VGTL (Vienna Graph Template Library)

Version 1.4

Reference Manual

Hermann Schichl
University of Vienna, Faculty of Mathematics
Nordbergstr. 15
A-1090 Wien, Austria
email: Hermann.Schichl@univie.ac.at

Technical Report
January 2013

Contents

1 Introduction

The Vienna Graph Template Library (VGTL) is a generic graph library with generic programming structure. It uses STL containers like `map` and `vector` to organize the internal structure of the graphs.

A collection of walking algorithms for analyzing and working with the graphs has been implemented as generic algorithms. Similar to STL iterators, which are used to handle data in containers independently of the container implementation, for graphs the walker concept (see Section Walker) is introduced.

1.1 Walker

A **walker** is, like an STL iterator, a generalization of a pointer. It dereferences to the data a graph node stores.

There are two different kinds of walkers: **recursive** walker and **iterative** walker.

1.1.1 Recursive Walker

A recursive walker is a pointer to graph nodes, which can be moved around on the graph by changing the node it points to. Walkers can move along the edges of the graph to new nodes. The operators reserved for that are $<<$ for moving along in-edges and $>>$ for moving along out-edges. A recursive walker does not have an internal status, so the walking has to be done recursively.

1.1.2 Iterative Walker

An iterative walker (automatic walker) can walk through a graph without guidance. Simply using the operators $++$ and $--$, the walker itself searches for the next node in the walk.

1.2 Trees and Forests

The first few of the collection of graph containers are the n-ary trees and forests. These trees come in various flavors: standard trees, labelled trees, with and without data hooks. Trees provide iterative walkers and recursive walkers.

1.3 Directed Graphs and DAGs

The next more complicated graphs are **directed graphs. There are two classes implemented. Standard directed graphs and directed acyclic graphs (DAGs). Directed graphs provide recursive walkers only.**

1.4 Generic Graphs

Generic graphs don't have directed edges. They are the most general class of graphs, and special walking algorithms are provided for them. Generic graphs only have recursive walkers.

2 Module Index

2.1 Modules

Here is a list of all modules:

3 Namespace Index

3.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

4 Class Index

4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

5 Class Index

5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

6 File Index

6.1 File List

Here is a list of all documented files with brief descriptions:

7 Module Documentation

7.1 Classes and types for external use

Classes

- class dgraph

  unlabeled directed graph
- class dag

  unlabeled directed acyclic graph (DAG)
- class ldgraph

  labeled directed graph
- class ldag

  labeled directed acyclic graph (LDAG)
- class ntree< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >

  n-ary forest
- class rntree

  n-ary forest
- class atree< _Tp, _AssocCtr, _Key, _Compare, _PtrAlloc, _Alloc >

  n-ary forest with labelled edges
- class stree< _Key, _Compare, _AssocCtr, _PtrAlloc, _Alloc >

  n-ary forest with unsorted edges
- class ratree

  n-ary forest with labelled edges
- class rstree

  n-ary forest with unsorted edges
- class preorder_visitor

  preorder visitor base class
- class postorder_visitor

  postorder visitor base class
- class prepost_visitor

  pre+postorder visitor base class

Defines

- #define VGTL_VECTOR_IMPL

  STL vector wrapper for C array.

7.1.1 Detailed Description

The classes and types in this section are for external use.

7.1.2 Define Documentation

7.1.2.1 #define VGTL_VECTOR_IMPL

This class is a wrapper class, which builds a STL vector around a C array. Afterwards, this array_vector can be used lika a const std::vector of the same type.

Definition at line 55 of file array_vector.h.

7.2 Generic algorithms for external use

Functions

- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_safe_walk_if (_Walker __w, _Visitor __f)
- template<class _IterativeWalker , class _Function >
  _Function walk (_IterativeWalker __first, _IterativeWalker __last, _Function __f)
- template<class _PrePostWalker , class _Function >
  _Function pre_post_walk (_PrePostWalker __first, _PrePostWalker __last, _Function __f)
- template<class _PrePostWalker , class _Function1 , class _Function2 >
  _Function2 pre_post_walk (_PrePostWalker __first, _PrePostWalker __last, _Function1 __f1, _Function2 __f2)
- template<class _PrePostWalker , class _Function >
  _Function var_walk (_PrePostWalker __first, _PrePostWalker __last, _Function __f)
- template<class _PrePostWalker , class _Function1 , class _Function2 >
  _Function2 var_walk (_PrePostWalker __first, _PrePostWalker __last, _Function1 __f1, _Function2 __f2)
- template<class _PrePostWalker , class _Function , class _Predicate >
  _Function walk_if (_PrePostWalker __first, _PrePostWalker __last, _Function __f, _Predicate __pred)
- template<class _PrePostWalker , class _Function1 , class _Function2 , class _Predicate >
  _Function2 walk_if (_PrePostWalker __first, _PrePostWalker __last, _Function1 __f1, _Function2 __f2, _Predicate __pred)
- template<class _PrePostWalker , class _Function1 , class _Function2 , class _Predicate1 , class _Predicate2 >
  _Function2 walk_if (_PrePostWalker __first, _PrePostWalker __last, _Function1 __f1, _Function2 __f2, _Predicate1 __pred1, _Predicate2 __pred2)
- template<class _PrePostWalker , class _Function1 , class _Function2 , class _Predicate >
  _Function2 cached_walk_if (_PrePostWalker __first, _PrePostWalker __last, _Function1 __f1, _Function2 __f2, _Predicate __pred)
- template<class _PrePostWalker , class _Function1 , class _Function2 , class _Predicate >
  _Function2 multi_walk_if (_PrePostWalker __first, _PrePostWalker __last, _Function1 __f1, _Function2 __f2, _Predicate __pred)
- template<class _Walker , class _Function >
  _Function walk_up (_Walker __w, _Function __f)
- template<class _Walker , class _Function >
  _Function var_walk_up (_Walker __w, _Function __f)
- template<class _Walker , class _Function , class _Predicate >
  _Function walk_up_if (_Walker __w, _Function __f, _Predicate __p)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_preorder_walk (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_postorder_walk (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_walk (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_preorder_walk_if (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor , class _Predicate >
  _Visitor::return_value recursive_preorder_walk_if (_Walker __w, _Visitor __f, _Predicate __p)
- template<class _Walker , class _Visitor , class _Predicate >
  _Visitor::return_value recursive_postorder_walk_if (_Walker __w, _Visitor __f, _Predicate __p)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_walk_if (_Walker __w, _Visitor __f)

- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_cached_walk (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_multi_walk (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor , class _Predicate1 , class _Predicate2 >
  _Visitor::return_value recursive_walk_if (_Walker __w, _Visitor __f, _Predicate1 __p1, _Predicate2 __p2)
- template<class _Walker , class _Visitor , class _Predicate >
  _Visitor::return_value recursive_cached_walk (_Walker __w, _Visitor __f, _Predicate __p)
- template<class _Walker , class _Visitor , class _Predicate >
  _Visitor::return_value recursive_multi_walk (_Walker __w, _Visitor __f, _Predicate __p)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_preorder_walk_up (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_preorder_walk_up_if (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor , class _Predicate >
  _Visitor::return_value recursive_preorder_walk_up_if (_Walker __w, _Visitor __f, _Predicate __p)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_postorder_walk_up (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor , class _Predicate >
  _Visitor::return_value recursive_postorder_walk_up_if (_Walker __w, _Visitor __f, _Predicate __p)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_walk_up (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_walk_up_if (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor , class _Predicate1 , class _Predicate2 >
  _Visitor::return_value recursive_walk_up_if (_Walker __w, _Visitor __f, _Predicate1 __p1, _Predicate2 __p2)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_cached_walk_up (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_multi_walk_up (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor , class _Predicate >
  _Visitor::return_value recursive_cached_walk_up (_Walker __w, _Visitor __f, _Predicate __p)
- template<class _Walker , class _Visitor , class _Predicate >
  _Visitor::return_value recursive_multi_walk_up (_Walker __w, _Visitor __f, _Predicate __p)
- template<class _Walker , class _Visitor >
  _Visitor::return_value general_directed_walk (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value general_directed_walk_down (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value general_directed_walk_up (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_general_directed_walk (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_general_directed_walk_down (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_general_directed_walk_up (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value general_walk (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_general_walk (_Walker __w, _Visitor __f)

- template<class _BidirIter , class _Tp >
  _BidirIter rfind (_BidirIter __first, _BidirIter __last, const _Tp &__val)
- template<class _BidirIter , class _Predicate >
  _BidirIter rfind_if (_BidirIter __first, _BidirIter __last, _Predicate __pred)
- template<class _Walker , class _Test >
  void recursive_consistency_test (_Walker __w, const _Test &__t)

7.2.1 Detailed Description

The generic functions in this section are for external use.

7.2.2 Function Documentation

7.2.2.1 template<class _PrePostWalker , class _Function1 , class _Function2 , class _Predicate > _Function2 cached_walk_if (_PrePostWalker __first, _PrePostWalker __last, _Function1 __f1, _Function2 __f2, _Predicate __pred)

this tree walk is a pre+post walk, calling two functions at every node, one in the preorder and the other in the postorder visit. If the function returns true, the status of the walker is flipped from pre to post (or vice versa). If the status is changed from pre to post, the subtree originating from the current position is not visited, if the status change is the other way round, it is revisited. This allows for cached or partially multi pass walks.

Definition at line 394 of file vgtl_algo.h.

7.2.2.2 template<class _Walker , class _Visitor > _Visitor::return_value general_directed_walk (_Walker __w, _Visitor __f)

perform a general directed walk starting at `__w`. At every node various methods of the visitor `__f` are called:

- `analyze` is called before walking for every virtual node. While this function returns `true`, the walk goes on.
- `preorder` is called before a walk direction is being decided.
- `postorder` is called after the walk dircetion has been found.
- `walk_up` shall return whether the next step of the walk is upwards or downwards.
- `up` is called for an upwards step and decides which in-edge to take.
- `down` is called for a downwards step and decides which out-edge to take.
- `value` is called to compute the return value for this node.

Definition at line 2390 of file vgtl_algo.h.

7.2.2.3 template<class _Walker , class _Visitor > _Visitor::return_value general_directed_walk_down (_Walker __w, _Visitor __f)

perform a general directed walk starting at `__w`. At every node various methods of the visitor `__f` are called:

- `analyze` is called before walking for every virtual node. While this function returns `true`, the walk goes on.

- `preorder` is called before a walk direction is being decided.
- `postorder` is called after the walk dircetion has been found.
- `down` is called to decide which out-edge to take.
- `value` is called to compute the return value for this node.

Definition at line 2419 of file vgtl_algo.h.

7.2.2.4 template<class _Walker , class _Visitor > _Visitor::return_value general_directed_walk_up (_Walker __w, _Visitor __f)

perform a general directed walk starting at `__w`. At every node various methods of the visitor `__f` are called:

- `analyze` is called before walking for every virtual node. While this function returns `true`, the walk goes on.
- `preorder` is called before a walk direction is being decided.
- `postorder` is called after the walk dircetion has been found.
- `up` is called to decide which in-edge to take.
- `value` is called to compute the return value for this node.

Definition at line 2446 of file vgtl_algo.h.

7.2.2.5 template<class _Walker , class _Visitor > _Visitor::return_value general_walk (_Walker __w, _Visitor __f)

perform a general walk starting at `__w`. At every node various methods of the visitor `__f` are called:

- `analyze` is called before walking for every virtual node. While this function returns `true`, the walk goes on.
- `preorder` is called before a walk direction is being decided.
- `postorder` is called after the walk dircetion has been found.
- `next` is called to decide which edge to follow.
- `value` is called to compute the return value for this node.

Definition at line 2558 of file vgtl_algo.h.

7.2.2.6 template<class _PrePostWalker , class _Function1 , class _Function2 , class _Predicate > _Function2 multi_walk_if (_PrePostWalker __first, _PrePostWalker __last, _Function1 __f1, _Function2 __f2, _Predicate __pred)

this tree walk is a pre+post walk, calling two functions at every node, one in the preorder and the other in the postorder visit. If the function returns true, the status of the walker is flipped from pre to post (or vice versa). If the status is changed from pre to post, the subtree originating from the current position is not visited, if the status change is the other way round, it is revisited. This allows for cached or partially multi pass walks.

Definition at line 427 of file vgtl_algo.h.

7.2.2.7 template<class _PrePostWalker , class _Function > _Function pre_post_walk (_PrePostWalker __first, _PrePostWalker __last, _Function __f)

make a pre and post order tree walk, calling a function for every node.

Definition at line 206 of file vgtl_algo.h.

7.2.2.8 template<class _PrePostWalker , class _Function1 , class _Function2 > _Function2 pre_post_walk (_PrePostWalker __first, _PrePostWalker __last, _Function1 __f1, _Function2 __f2)

make a pre and post order tree walk, calling two different functions, one in the preorder step, the other in the postorder step.

Definition at line 224 of file vgtl_algo.h.

7.2.2.9 template<class _Walker , class _Visitor > _Visitor::return_value recursive_cached_walk (_Walker __w, _Visitor __f)

perform a recursive pre+post order walk starting at ___w. At every node various methods of the visitor ___f are called:

- `vinit` is called before walking for every virtual node
- `vcollect` is called after a child of a virtual node has finished
- `vvalue` is called to compute the return value of a virtual node
- `preorder` is called before the children are visited. If it returns `true`, the children are visited. If it returns `false`, the children are ignored
- `collect` is called everytime a child has finished
- `postorder` is called after the children have been visited.
- `value` is called to compute the return value for this node

Definition at line 1048 of file vgtl_algo.h.

7.2.2.10 template<class _Walker , class _Visitor , class _Predicate > _Visitor::return_value recursive_cached_walk (_Walker __w, _Visitor __f, _Predicate __p)

perform a recursive pre+post order walk starting at ___w. At every node various methods of the visitor ___f are called:

- `vinit` is called before walking for every virtual node
- `vcollect` is called after a child of a virtual node has finished
- `vvalue` is called to compute the return value of a virtual node
- `preorder` is called before the children are visited. If then predicate ___p returns `true`, the children are visited. If it returns `false`, the children are ignored
- `collect` is called everytime a child has finished
- `postorder` is called after the children have been visited.
- `value` is called to compute the return value for this node

Definition at line 1297 of file vgtl_algo.h.

7.2.2.11 template<class _Walker , class _Visitor > _Visitor::return_value recursive_cached_walk_up (_Walker __w, _Visitor __f)

perform a recursive pre+post order walk towards the root starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node
- vcollect is called after a child of a virtual node has finished
- vvalue is called to compute the return value of a virtual node
- preorder is called before the children are visited. If it returns true, the children are visited. If it returns false, the children are ignored
- collect is called everytime a child has finished
- postorder is called after the children have been visited.
- value is called to compute the return value for this node

Definition at line 2066 of file vgtl_algo.h.

7.2.2.12 template<class _Walker , class _Visitor , class _Predicate > _Visitor::return_value recursive_cached_walk_up (_Walker __w, _Visitor __f, _Predicate __p)

perform a recursive pre+post order walk towards the root starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node
- vcollect is called after a child of a virtual node has finished
- vvalue is called to compute the return value of a virtual node
- preorder is called before the children are visited. If then predicate __p returns true, the children are visited. If it returns false, the children are ignored
- collect is called everytime a child has finished
- postorder is called after the children have been visited.
- value is called to compute the return value for this node

Definition at line 2224 of file vgtl_algo.h.

7.2.2.13 template<class _Walker , class _Test > void recursive_consistency_test (_Walker __w, const _Test & __t)

perform a consistency test of the tree or DAG.

Definition at line 49 of file vgtl_test.h.

7.2.2.14 template<class _Walker , class _Visitor > _Visitor::return_value recursive_general_directed_walk (_Walker __w, _Visitor __f)

perform a recursive general directed walk starting at __w. At every node various methods of the visitor __f are called:

- preorder is called before any child is visited
- analyze is called everytime before a child node might be vistied. While this function returns true, the walk goes on at this node.
- collect is called everytime a child has finished.
- postorder is called after the walk dircetion has been found.
- walk_up shall return whether the next step of the walk is upwards or downwards.
- up is called for an upwards step and decides which in-edge to take.
- down is called for a downwards step and decides which out-edge to take.
- value is called to compute the return value for this node.

Definition at line 2479 of file vgtl_algo.h.

7.2.2.15 template<class _Walker , class _Visitor > _Visitor::return_value recursive_general_directed_walk_down (_Walker __w, _Visitor __f)

perform a recursive general directed walk starting at __w. At every node various methods of the visitor __f are called:

- preorder is called before any child is visited
- analyze is called everytime before a child node might be vistied. While this function returns true, the walk goes on at this node.
- collect is called everytime a child has finished.
- postorder is called after the walk dircetion has been found.
- down is called to decide which out-edge to take.
- value is called to compute the return value for this node.

Definition at line 2509 of file vgtl_algo.h.

7.2.2.16 template<class _Walker , class _Visitor > _Visitor::return_value recursive_general_directed_walk_up (_Walker __w, _Visitor __f)

perform a recursive general directed walk starting at __w. At every node various methods of the visitor __f are called:

- preorder is called before any child is visited
- analyze is called everytime before a child node might be vistied. While this function returns true, the walk goes on at this node.
- collect is called everytime a child has finished.
- postorder is called after the walk dircetion has been found.
- up is called to decide which in-edge to take.
- value is called to compute the return value for this node.

Definition at line 2534 of file vgtl_algo.h.

7.2.2.17 template<class _Walker , class _Visitor > _Visitor::return_value recursive_general_walk (_Walker __w, _Visitor __f)

perform a recursive general walk starting at __w. At every node various methods of the visitor __f are called:

- preorder is called before any child is visited
- analyze is called everytime before a child node might be vistied. While this function returns true, the walk goes on at this node.
- collect is called everytime a child has finished.
- postorder is called after the walk dircetion has been found.
- next is called to decide which edge to follow.
- value is called to compute the return value for this node.

Definition at line 2585 of file vgtl_algo.h.

7.2.2.18 template<class _Walker , class _Visitor > _Visitor::return_value recursive_multi_walk (_Walker __w, _Visitor __f)

perform a recursive pre+post order walk starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node.
- vcollect is called after a child of a virtual node has finished.
- vvalue is called to compute the return value of a virtual node.
- preorder is called before the children are visited.
- collect is called everytime a child has finished.
- postorder is called after the children have been visited. If it returns true, the walk is continued by switching back to preorder mode for this node. If it returns false, the walk is over for this node.
- value is called to compute the return value for this node.

Definition at line 1124 of file vgtl_algo.h.

7.2.2.19 template<class _Walker , class _Visitor , class _Predicate > _Visitor::return_value recursive_multi_walk (_Walker __w, _Visitor __f, _Predicate __p)

perform a recursive pre+post order walk starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node.
- vcollect is called after a child of a virtual node has finished.
- vvalue is called to compute the return value of a virtual node.
- preorder is called before the children are visited.
- collect is called everytime a child has finished.

- `postorder` is called after the children have been visited. If the predicate `__p` returns `true`, the walk is continued by switching back to preorder mode for this node. If it returns `false`, the walk is over for this node.
- `value` is called to compute the return value for this node.

Definition at line 1376 of file vgtl_algo.h.

7.2.2.20 template<class _Walker , class _Visitor > _Visitor::return_value recursive_multi_walk_up (_Walker __w, _Visitor __f)

perform a recursive pre+post order walk towards the root starting at `__w`. At every node various methods of the visitor `__f` are called:

- `vinit` is called before walking for every virtual node.
- `vcollect` is called after a child of a virtual node has finished.
- `vvalue` is called to compute the return value of a virtual node.
- `preorder` is called before the children are visited.
- `collect` is called everytime a child has finished.
- `postorder` is called after the children have been visited. If it returns `true`, the walk is continued by switching back to preorder mode for this node. If it returns `false`, the walk is over for this node.
- `value` is called to compute the return value for this node.

Definition at line 2143 of file vgtl_algo.h.

7.2.2.21 template<class _Walker , class _Visitor , class _Predicate > _Visitor::return_value recursive_multi_walk_up (_Walker __w, _Visitor __f, _Predicate __p)

perform a recursive pre+post order walk towards the root starting at `__w`. At every node various methods of the visitor `__f` are called:

- `vinit` is called before walking for every virtual node.
- `vcollect` is called after a child of a virtual node has finished.
- `vvalue` is called to compute the return value of a virtual node.
- `preorder` is called before the children are visited.
- `collect` is called everytime a child has finished.
- `postorder` is called after the children have been visited. If the predicate `__p` returns `true`, the walk is continued by switching back to preorder mode for this node. If it returns `false`, the walk is over for this node.
- `value` is called to compute the return value for this node.

Definition at line 2303 of file vgtl_algo.h.

7.2.2.22 template<class _Walker , class _Visitor > _Visitor::return_value recursive_postorder_walk (_Walker __w, _Visitor __f)

perform a recursive postorder walk starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node
- vcollect is called after a child of a virtual node has finished
- vvalue is called to compute the return value of a virtual node
- init is called before the children are visited
- collect is called everytime a child has finished
- postorder is called after all children have finished
- value is called to compute the return value for this node

Definition at line 596 of file vgtl_algo.h.

7.2.2.23 template<class _Walker , class _Visitor , class _Predicate > _Visitor::return_value recursive_postorder_walk_if (_Walker __w, _Visitor __f, _Predicate __p)

perform a recursive postorder walk starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node
- vcollect is called after a child of a virtual node has finished
- vvalue is called to compute the return value of a virtual node
- init is called before the children are visited. Then the predicate is called. If this predicate returns true, the children are visited. Otherwise, the node is treated as if it was a terminal node.
- postorder is called after all children have been visited.
- collect is called everytime a child has finished.
- value is called to compute the return value for this node.

Definition at line 881 of file vgtl_algo.h.

7.2.2.24 template<class _Walker , class _Visitor > _Visitor::return_value recursive_postorder_walk_up (_Walker __w, _Visitor __f)

perform a recursive postorder walk towards the root starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node
- vcollect is called after a child of a virtual node has finished
- vvalue is called to compute the return value of a virtual node
- init is called before the children are visited
- collect is called everytime a child has finished

- postorder is called after all children have finished
- value is called to compute the return value for this node

Definition at line 1669 of file vgtl_algo.h.

7.2.2.25 template<class _Walker , class _Visitor , class _Predicate > _Visitor::return_value recursive_postorder_walk_up_if (_Walker __w, _Visitor __f, _Predicate __p)

perform a recursive postorder walk towards the root starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node
- vcollect is called after a child of a virtual node has finished
- vvalue is called to compute the return value of a virtual node
- init is called before the children are visited. Then the predicate is called. If this predicate returns true, the children are visited. Otherwise, the node is treated as if it was a terminal node.
- postorder is called after all children have been visited.
- collect is called everytime a child has finished.
- value is called to compute the return value for this node.

Definition at line 1740 of file vgtl_algo.h.

7.2.2.26 template<class _Walker , class _Visitor > _Visitor::return_value recursive_preorder_walk (_Walker __w, _Visitor __f)

perform a recursive preorder walk starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node
- vcollect is called after a child of a virtual node has finished
- vvalue is called to compute the return value of a virtual node
- preorder is called before the children are visited
- collect is called everytime a child has finished
- value is called to compute the return value for this node

Definition at line 531 of file vgtl_algo.h.

7.2.2.27 template<class _Walker , class _Visitor > _Visitor::return_value recursive_preorder_walk_if (_Walker __w, _Visitor __f)

perform a recursive preorder walk starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node
- vcollect is called after a child of a virtual node has finished

- vvalue is called to compute the return value of a virtual node
- preorder is called before the children are visited. If this function returns true, the children are visited. Otherwise, the node is treated as if it was a terminal node.
- collect is called everytime a child has finished
- value is called to compute the return value for this node

Definition at line 731 of file vgtl_algo.h.

7.2.2.28 template<class _Walker , class _Visitor , class _Predicate > _Visitor::return_value recursive_preorder_walk_if (_Walker __w, _Visitor __f, _Predicate __p)

perform a recursive preorder walk starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node
- vcollect is called after a child of a virtual node has finished
- vvalue is called to compute the return value of a virtual node
- preorder is called before the children are visited. Then the predicate is called. If this predicate returns true, the children are visited. Otherwise, the node is treated as if it was a terminal node.
- collect is called everytime a child has finished
- value is called to compute the return value for this node

Definition at line 804 of file vgtl_algo.h.

7.2.2.29 template<class _Walker , class _Visitor > _Visitor::return_value recursive_preorder_walk_up (_Walker __w, _Visitor __f)

perform a recursive preorder walk towards the root starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node
- vcollect is called after a child of a virtual node has finished
- vvalue is called to compute the return value of a virtual node
- preorder is called before the children are visited
- collect is called everytime a child has finished
- value is called to compute the return value for this node

Definition at line 1456 of file vgtl_algo.h.

7.2.2.30 template<class _Walker , class _Visitor > _Visitor::return_value recursive_preorder_walk_up_if (_Walker __w, _Visitor __f)

perform a recursive preorder walk towards the root starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node

- vcollect is called after a child of a virtual node has finished
- vvalue is called to compute the return value of a virtual node
- preorder is called before the children are visited. If this function returns true, the children are visited. Otherwise, the node is treated as if it was a terminal node.
- collect is called everytime a child has finished
- value is called to compute the return value for this node

Definition at line 1522 of file vgtl_algo.h.

7.2.2.31 template<class _Walker , class _Visitor , class _Predicate > _Visitor::return_value recursive_preorder_walk_up_if (_Walker __w, _Visitor __f, _Predicate __p)

perform a recursive preorder walk towards the root starting at ___w. At every node various methods of the visitor ___f are called:

- vinit is called before walking for every virtual node
- vcollect is called after a child of a virtual node has finished
- vvalue is called to compute the return value of a virtual node
- preorder is called before the children are visited. Then the predicate is called. If this predicate returns true, the children are visited. Otherwise, the node is treated as if it was a terminal node.
- collect is called everytime a child has finished
- value is called to compute the return value for this node

Definition at line 1595 of file vgtl_algo.h.

7.2.2.32 template<class _Walker , class _Visitor > _Visitor::return_value recursive_safe_walk_if (_Walker __w, _Visitor __f)

perform a recursive pre+post order walk starting at ___w. At every node various methods of the visitor ___f are called:

- vinit is called before walking for every virtual node.
- vcollect is called after a child of a virtual node has finished.
- vvalue is called to compute the return value of a virtual node.
- preorder is called before the children are visited. If it returns true, the children are visited. If it returns false, the children are ignored.
- collect is called everytime a child has finished.
- postorder is called after the children have been visited. If it returns true, the walk is continued by switching back to preorder mode for this node. If it returns false, the walk is over for this node.
- value is called to compute the return value for this node.

Definition at line 59 of file vgtl_addalgo.h.

7.2.2.33 template<class _Walker , class _Visitor > _Visitor::return_value recursive_walk (_Walker __w, _Visitor __f)

perform a recursive pre+post order walk starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node
- vcollect is called after a child of a virtual node has finished
- vvalue is called to compute the return value of a virtual node
- preorder is called before the children are visited
- collect is called everytime a child has finished
- postorder is called after all children have been visited
- value is called to compute the return value for this node

Definition at line 664 of file vgtl_algo.h.

7.2.2.34 template<class _Walker , class _Visitor > _Visitor::return_value recursive_walk_if (_Walker __w, _Visitor __f)

perform a recursive pre+post order walk starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node.
- vcollect is called after a child of a virtual node has finished.
- vvalue is called to compute the return value of a virtual node.
- preorder is called before the children are visited. If it returns true, the children are visited. If it returns false, the children are ignored.
- collect is called everytime a child has finished.
- postorder is called after the children have been visited. If it returns true, the walk is continued by switching back to preorder mode for this node. If it returns false, the walk is over for this node.
- value is called to compute the return value for this node.

Definition at line 963 of file vgtl_algo.h.

7.2.2.35 template<class _Walker , class _Visitor , class _Predicate1 , class _Predicate2 > _Visitor::return_value recursive_walk_if (_Walker __w, _Visitor __f, _Predicate1 __p1, _Predicate2 __p2)

perform a recursive pre+post order walk starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node
- vcollect is called after a child of a virtual node has finished
- vvalue is called to compute the return value of a virtual node

- `preorder` is called before the children are visited. If then predicate `p1` returns `true`, the children are visited. If it returns `false`, the children are ignored
- `collect` is called everytime a child has finished
- `postorder` is called after the children have been visited. If then predicate `p2` returns `true`, the walk is continued by switching back to preorder mode for this node. If it returns `false`, the walk is over for this node.
- `value` is called to compute the return value for this node

Definition at line 1206 of file vgtl_algo.h.

7.2.2.36 template<class _Walker , class _Visitor > _Visitor::return_value recursive_walk_up (_Walker __w, _Visitor __f)

perform a recursive pre+post order walk towards the root starting at `__w`. At every node various methods of the visitor `__f` are called:

- `vinit` is called before walking for every virtual node
- `vcollect` is called after a child of a virtual node has finished
- `vvalue` is called to compute the return value of a virtual node
- `preorder` is called before the children are visited
- `collect` is called everytime a child has finished
- `postorder` is called after all children have been visited
- `value` is called to compute the return value for this node

Definition at line 1816 of file vgtl_algo.h.

7.2.2.37 template<class _Walker , class _Visitor > _Visitor::return_value recursive_walk_up_if (_Walker __w, _Visitor __f)

perform a recursive pre+post order walk towards the root starting at `__w`. At every node various methods of the visitor `__f` are called:

- `vinit` is called before walking for every virtual node.
- `vcollect` is called after a child of a virtual node has finished.
- `vvalue` is called to compute the return value of a virtual node.
- `preorder` is called before the children are visited. If it returns `true`, the children are visited. If it returns `false`, the children are ignored.
- `collect` is called everytime a child has finished.
- `postorder` is called after the children have been visited. If it returns `true`, the walk is continued by switching back to preorder mode for this node. If it returns `false`, the walk is over for this node.
- `value` is called to compute the return value for this node.

Definition at line 1887 of file vgtl_algo.h.

7.2.2.38 template<class _Walker , class _Visitor , class _Predicate1 , class _Predicate2 > _Visitor::return_value recursive_walk_up_if (_Walker __w, _Visitor __f, _Predicate1 __p1, _Predicate2 __p2)

perform a recursive pre+post order walk towards the root starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node
- vcollect is called after a child of a virtual node has finished
- vvalue is called to compute the return value of a virtual node
- preorder is called before the children are visited. If then predicate p1 returns true, the children are visited. If it returns false, the children are ignored
- collect is called everytime a child has finished
- postorder is called after the children have been visited. If then predicate p2 returns true, the walk is continued by switching back to preorder mode for this node. If it returns false, the walk is over for this node.
- value is called to compute the return value for this node

Definition at line 1975 of file vgtl_algo.h.

7.2.2.39 template<class _BidirIter , class _Tp > _BidirIter rfind (_BidirIter __first, _BidirIter __last, const _Tp & __val) [inline]

Find the last occurrence of a value in a sequence.

Parameters

__first	An input iterator.
__last	An input iterator.
__val	The value to find.

Returns

The last iterator i in the range [__first,__last) such that *i == val, or __last if no such iterator exists.

Definition at line 192 of file vgtl_helpers.h.

7.2.2.40 template<class _BidirIter , class _Predicate > _BidirIter rfind_if (_BidirIter __first, _BidirIter __last, _Predicate __pred) [inline]

Find the last element in a sequence for which a predicate is true.

Parameters

__first	An input iterator.
__last	An input iterator.
__pred	A predicate.

Returns

The last iterator `i` in the range [__first,__last) such that `__pred(*i)` is true, or `__last` if no such iterator exists.

Definition at line 208 of file vgtl_helpers.h.

7.2.2.41 template<class _PrePostWalker , class _Function > _Function var_walk (_PrePostWalker __first, _PrePostWalker __last, _Function __f)

this tree walk is a pre+post walk, calling a function at every node. If the function returns `true`, the status of the walker is flipped from pre to post (or vice versa). If the status is changed from pre to post, the subtree originating from the current position is not visited, if the status change is the other way round, it is revisited. This allows for cached or partially multi pass walks.

Definition at line 248 of file vgtl_algo.h.

7.2.2.42 template<class _PrePostWalker , class _Function1 , class _Function2 > _Function2 var_walk (_PrePostWalker __first, _PrePostWalker __last, _Function1 __f1, _Function2 __f2)

this tree walk is a pre+post walk, calling two functions at every node, one in the preorder and the other in the postorder step. If the function returns true, the status of the walker is flipped from pre to post (or vice versa). If the status is changed from pre to post, the subtree originating from the current position is not visited, if the status change is the other way round, it is revisited. This allows for cached or partially multi pass walks.

Definition at line 271 of file vgtl_algo.h.

7.2.2.43 template<class _Walker , class _Function > _Function var_walk_up (_Walker __w, _Function __f)

this tree walk is a pre+post walk towards the root, calling a function at every node. If the function returns `true`, the status of the walker is flipped from pre to post (or vice versa). If the status is changed from pre to post, the subtree originating from the current position is not visited, if the status change is the other way round, it is revisited. This allows for cached or partially multi pass walks.

Definition at line 476 of file vgtl_algo.h.

7.2.2.44 template<class _IterativeWalker , class _Function > _Function walk (_IterativeWalker __first, _IterativeWalker __last, _Function __f)

make a pre or post order tree walk, calling a function for every node it is also possible to perform a pre+post order walk. In that case the function `_f` must distinguish between the two calls by itself.

Definition at line 191 of file vgtl_algo.h.

7.2.2.45 template<class _PrePostWalker , class _Function , class _Predicate > _Function walk_if (_PrePostWalker __first, _PrePostWalker __last, _Function __f, _Predicate __pred)

this tree walk is a pre+post walk, calling a function at every node. If the predicate returns true, the status of the walker is flipped from pre to post (or vice versa). If the status is changed from pre to post, the subtree originating from the current position is not visited, if the status change is the other way round, it is revisited. This allows for cached or partially multi pass walks.

Definition at line 296 of file vgtl_algo.h.

7.2.2.46 template<class _PrePostWalker , class _Function1 , class _Function2 , class _Predicate > _Function2 walk_if (_PrePostWalker __first, _PrePostWalker __last, _Function1 __f1, _Function2 __f2, _Predicate __pred)

this tree walk is a pre+post walk, calling two functions at every node, one in the preorder and the other in the postorder visit. If the predicate returns true, the status of the walker is flipped from pre to post (or vice versa). If the status is changed from pre to post, the subtree originating from the current position is not visited, if the status change is the other way round, it is revisited. This allows for cached or partially multi pass walks.

Definition at line 323 of file vgtl_algo.h.

7.2.2.47 template<class _PrePostWalker , class _Function1 , class _Function2 , class _Predicate1 , class _Predicate2 > _Function2 walk_if (_PrePostWalker __first, _PrePostWalker __last, _Function1 __f1, _Function2 __f2, _Predicate1 __pred1, _Predicate2 __pred2)

this tree walk is a pre+post walk, calling two functions at every node, one in the preorder and the other in the postorder visit. If the predicates return true, the status of the walker is flipped from pre to post (or vice versa). If the status is changed from pre to post, the subtree originating from the current position is not visited, if the status change is the other way round, it is revisited. This allows for cached or partially multi pass walks. Predicate pred1 is called in the preorder phase, predicate pred2 in the postorder phase.

Definition at line 356 of file vgtl_algo.h.

7.2.2.48 template<class _Walker , class _Function > _Function walk_up (_Walker __w, _Function __f)

make a pre or post order tree walk towards the root node, calling a function for every node it is also possible to perform a pre+post order walk. In that case the function `_f` must distinguish between the two calls by itself.

Definition at line 456 of file vgtl_algo.h.

7.2.2.49 template<class _Walker , class _Function , class _Predicate > _Function walk_up_if (_Walker __w, _Function __f, _Predicate __p)

this tree walk is a pre+post walk towards the root, calling a function at every node. If the predicate returns true, the status of the walker is flipped from pre to post (or vice versa). If the status is changed from pre to post, the subtree originating from the current position is not visited, if the status change is the other way round, it is revisited. This allows for cached or partially multi pass walks.

Definition at line 497 of file vgtl_algo.h.

7.3 Classes and types for internal use

Classes

- class child_data_iterator< _Tree >

  Iterator which iterates through the data hooks of all children.
- class _DG_walker

  recursive directed graph walkers
- class _DG_iterator

  iterator through the directed graph
- class __DG

  Directed graph base class.
- class _DG_node

  directed graph node
- class _DG_base

  Directed graph base class for allocator encapsulation.
- class pointer_adaptor

  adaptor transforming a comparison predicate to pointers
- class pair_adaptor

  adaptor for an iterator over a pair to an iterator returning the second element
- class __one_iterator

  make an iterator out of one pointer
- class _G_compare_adaptor

  Adaptor for data comparison in graph nodes.
- class _LDG_walker

  recursive labelled directed graph walkers
- class _LDG_iterator

  iterator through the directed graph
- class __LDG

  Labelled directed graph base class.
- class _LDG_node

  labelled directed graph node
- class _LDG_edge

  labelled directed graph edge
- class _LDG_base

  Labelled directed graph base class for allocator encapsulation.
- class _Tree_node

  tree node for trees w/o data hooks
- class _ITree_node

  tree node for trees with data hooks
- class _Tree_walker_base

  base class for all tree walkers
- class _Tree_walker

  automatic tree walkers
- class _RTree_walker< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >

  recursive tree walkers
- class _Tree_iterator< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >

iterator through the tree

- class _Tree_alloc_base< _Tp, _Ctr, _TI, _Node, _Allocator, _IsStatic >

  Tree base class for general standard-conforming allocators.
- class _Tree_alloc_base< _Tp, _Ctr, _TI, _Node, _Allocator, true >

  Tree base class specialization for instanceless allocators.
- class _Tree_base< _Tp, _Ctr, _TI, _Node, _Alloc >

  Tree base class for allocator encapsulation.
- class __Tree_t

  Tree base class.
- class __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >

  Tree base class without data hooks.
- class __ITree

  Tree base class with data hooks.

7.3.1 Detailed Description

The classes and types in this section are used VDBL internally.

7.4 Generic algorithms for internal use

Functions

- template<class _Walker , class _Visitor >
  _Visitor::return_value _recursive_safe_walk_if (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value _recursive_preorder_walk (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value _recursive_postorder_walk (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value _recursive_walk (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value _recursive_preorder_walk_if (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor , class _Predicate >
  _Visitor::return_value _recursive_preorder_walk_if (_Walker __w, _Visitor __f, _Predicate __p)
- template<class _Walker , class _Visitor , class _Predicate >
  _Visitor::return_value _recursive_postorder_walk_if (_Walker __w, _Visitor __f, _Predicate __p)
- template<class _Walker , class _Visitor >
  _Visitor::return_value _recursive_walk_if (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value _recursive_cached_walk (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value _recursive_multi_walk (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor , class _Predicate1 , class _Predicate2 >
  _Visitor::return_value _recursive_walk_if (_Walker __w, _Visitor __f, _Predicate1 __p1, _Predicate2 __p2)
- template<class _Walker , class _Visitor , class _Predicate >
  _Visitor::return_value _recursive_cached_walk (_Walker __w, _Visitor __f, _Predicate __p)
- template<class _Walker , class _Visitor , class _Predicate >
  _Visitor::return_value _recursive_multi_walk (_Walker __w, _Visitor __f, _Predicate __p)
- template<class _Walker , class _Visitor >
  _Visitor::return_value _recursive_preorder_walk_up (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value _recursive_preorder_walk_up_if (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor , class _Predicate >
  _Visitor::return_value _recursive_preorder_walk_up_if (_Walker __w, _Visitor __f, _Predicate __p)
- template<class _Walker , class _Visitor >
  _Visitor::return_value _recursive_postorder_walk_up (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor , class _Predicate >
  _Visitor::return_value _recursive_postorder_walk_up_if (_Walker __w, _Visitor __f, _Predicate __p)
- template<class _Walker , class _Visitor >
  _Visitor::return_value _recursive_walk_up (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value _recursive_walk_up_if (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor , class _Predicate1 , class _Predicate2 >
  _Visitor::return_value _recursive_walk_up_if (_Walker __w, _Visitor __f, _Predicate1 __p1, _Predicate2 __p2)
- template<class _Walker , class _Visitor >
  _Visitor::return_value _recursive_cached_walk_up (_Walker __w, _Visitor __f)

- template<class _Walker , class _Visitor >
  _Visitor::return_value _recursive_multi_walk_up (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor , class _Predicate >
  _Visitor::return_value _recursive_cached_walk_up (_Walker __w, _Visitor __f, _Predicate __p)
- template<class _Walker , class _Visitor , class _Predicate >
  _Visitor::return_value _recursive_multi_walk_up (_Walker __w, _Visitor __f, _Predicate __p)
- template<class _BidirIter , class _Tp >
  _BidirIter rfind (_BidirIter __first, _BidirIter __last, const _Tp &__val, std::bidirectional_iterator_tag)
- template<class _BidirIter , class _Predicate >
  _BidirIter rfind_if (_BidirIter __first, _BidirIter __last, _Predicate __pred, std::bidirectional_iterator_tag)
- template<class _RandomAccessIter , class _Tp >
  _RandomAccessIter rfind (_RandomAccessIter __first, _RandomAccessIter __last, const _Tp &__val, std::random_access_iterator_tag)
- template<class _RandomAccessIter , class _Predicate >
  _RandomAccessIter rfind_if (_RandomAccessIter __first, _RandomAccessIter __last, _Predicate __pred, std::random_access_iterator_tag)

7.4.1 Detailed Description

The generic functions in this section are used by other generic algorithms and are not intended for external use.

7.4.2 Function Documentation

7.4.2.1 template<class _Walker , class _Visitor > _Visitor::return_value _recursive_cached_walk (_Walker __w, _Visitor __f)

perform a recursive pre+post order walk starting at `__w`. At every node various methods of the visitor `__f` are called:

- `vinit` is called before walking for every virtual node
- `vcollect` is called after a child of a virtual node has finished
- `vvalue` is called to compute the return value of a virtual node
- `preorder` is called before the children are visited. If it returns `true`, the children are visited. If it returns `false`, the children are ignored
- `collect` is called everytime a child has finished
- `postorder` is called after the children have been visited.
- `value` is called to compute the return value for this node this function does not check for hitting the virtual ground node.

Definition at line 1091 of file vgtl_algo.h.

7.4.2.2 template<class _Walker , class _Visitor , class _Predicate > _Visitor::return_value _recursive_cached_walk (_Walker __w, _Visitor __f, _Predicate __p)

perform a recursive pre+post order walk starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node
- vcollect is called after a child of a virtual node has finished
- vvalue is called to compute the return value of a virtual node
- preorder is called before the children are visited. If then predicate p returns true, the children are visited. If it returns false, the children are ignored
- collect is called everytime a child has finished
- postorder is called after the children have been visited.
- value is called to compute the return value for this node this function does not check for hitting the virtual ground node.

Definition at line 1342 of file vgtl_algo.h.

7.4.2.3 template<class _Walker , class _Visitor > _Visitor::return_value _recursive_cached_walk_up (_Walker __w, _Visitor __f)

perform a recursive pre+post order walk towards the root starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node
- vcollect is called after a child of a virtual node has finished
- vvalue is called to compute the return value of a virtual node
- preorder is called before the children are visited. If it returns true, the children are visited. If it returns false, the children are ignored
- collect is called everytime a child has finished
- postorder is called after the children have been visited.
- value is called to compute the return value for this node this function does not check for hitting the virtual sky node.

Definition at line 2110 of file vgtl_algo.h.

7.4.2.4 template<class _Walker , class _Visitor , class _Predicate > _Visitor::return_value _recursive_cached_walk_up (_Walker __w, _Visitor __f, _Predicate __p)

perform a recursive pre+post order walk towards the root starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node
- vcollect is called after a child of a virtual node has finished
- vvalue is called to compute the return value of a virtual node

- `preorder` is called before the children are visited. If then predicate `__p` returns `true`, the children are visited. If it returns `false`, the children are ignored
- `collect` is called everytime a child has finished
- `postorder` is called after the children have been visited.
- `value` is called to compute the return value for this node this function does not check for hitting the virtual sky node.

Definition at line 2269 of file vgtl_algo.h.

7.4.2.5 template<class _Walker , class _Visitor > _Visitor::return_value _recursive_multi_walk (_Walker __w, _Visitor __f)

perform a recursive pre+post order walk starting at `__w`. At every node various methods of the visitor `__f` are called:

- `vinit` is called before walking for every virtual node.
- `vcollect` is called after a child of a virtual node has finished.
- `vvalue` is called to compute the return value of a virtual node.
- `preorder` is called before the children are visited.
- `collect` is called everytime a child has finished.
- `postorder` is called after the children have been visited. If it returns `true`, the walk is continued by switching back to preorder mode for this node. If it returns `false`, the walk is over for this node.
- `value` is called to compute the return value for this node. this function does not check for hitting the virtual ground node.

Definition at line 1170 of file vgtl_algo.h.

7.4.2.6 template<class _Walker , class _Visitor , class _Predicate > _Visitor::return_value _recursive_multi_walk (_Walker __w, _Visitor __f, _Predicate __p)

perform a recursive pre+post order walk starting at `__w`. At every node various methods of the visitor `__f` are called:

- `vinit` is called before walking for every virtual node.
- `vcollect` is called after a child of a virtual node has finished.
- `vvalue` is called to compute the return value of a virtual node.
- `preorder` is called before the children are visited.
- `collect` is called everytime a child has finished.
- `postorder` is called after the children have been visited. If the predicate `__p` returns `true`, the walk is continued by switching back to preorder mode for this node. If it returns `false`, the walk is over for this node.
- `value` is called to compute the return value for this node. this function does not check for hitting the virtual ground node.

Definition at line 1424 of file vgtl_algo.h.

7.4.2.7 template<class _Walker , class _Visitor > _Visitor::return_value _recursive_multi_walk_up (_Walker __w, _Visitor __f)

perform a recursive pre+post order walk towards the root starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node.
- vcollect is called after a child of a virtual node has finished.
- vvalue is called to compute the return value of a virtual node.
- preorder is called before the children are visited.
- collect is called everytime a child has finished.
- postorder is called after the children have been visited. If it returns true, the walk is continued by switching back to preorder mode for this node. If it returns false, the walk is over for this node.
- value is called to compute the return value for this node. this function does not check for hitting the virtual sky node.

Definition at line 2190 of file vgtl_algo.h.

7.4.2.8 template<class _Walker , class _Visitor , class _Predicate > _Visitor::return_value _recursive_multi_walk_up (_Walker __w, _Visitor __f, _Predicate __p)

perform a recursive pre+post order walk towards the root starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node.
- vcollect is called after a child of a virtual node has finished.
- vvalue is called to compute the return value of a virtual node.
- preorder is called before the children are visited.
- collect is called everytime a child has finished.
- postorder is called after the children have been visited. If the predicate __p returns true, the walk is continued by switching back to preorder mode for this node. If it returns false, the walk is over for this node.
- value is called to compute the return value for this node. this function does not check for hitting the virtual sky node.

Definition at line 2352 of file vgtl_algo.h.

7.4.2.9 template<class _Walker , class _Visitor > _Visitor::return_value _recursive_postorder_walk (_Walker __w, _Visitor __f)

perform a recursive postorder walk starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node
- vcollect is called after a child of a virtual node has finished

- vvalue is called to compute the return value of a virtual node
- init is called before the children are visited
- collect is called everytime a child has finished
- postorder is called after all children have finished
- value is called to compute the return value for this node this function does not check for hitting the virtual ground node.

Definition at line 636 of file vgtl_algo.h.

7.4.2.10 template<class _Walker , class _Visitor , class _Predicate > _Visitor::return_value _recursive_postorder_walk_if (_Walker __w, _Visitor __f, _Predicate __p)

perform a recursive postorder walk starting at ___w. At every node various methods of the visitor ___f are called:

- vinit is called before walking for every virtual node
- vcollect is called after a child of a virtual node has finished
- vvalue is called to compute the return value of a virtual node
- init is called before the children are visited. Then the predicate is called. If this predicate returns true, the children are visited. Otherwise, the node is treated as if it was a terminal node.
- postorder is called after all children have been visited.
- collect is called everytime a child has finished.
- value is called to compute the return value for this node. this function does not check for hitting the virtual ground node.

Definition at line 927 of file vgtl_algo.h.

7.4.2.11 template<class _Walker , class _Visitor > _Visitor::return_value _recursive_postorder_walk_up (_Walker __w, _Visitor __f)

perform a recursive postorder walk towards the root starting at ___w. At every node various methods of the visitor ___f are called:

- vinit is called before walking for every virtual node
- vcollect is called after a child of a virtual node has finished
- vvalue is called to compute the return value of a virtual node
- init is called before the children are visited
- collect is called everytime a child has finished
- postorder is called after all children have finished
- value is called to compute the return value for this node this function does not check for hitting the virtual sky node.

Definition at line 1709 of file vgtl_algo.h.

7.4.2.12 template<class _Walker , class _Visitor , class _Predicate > _Visitor::return_value _recursive_postorder_walk_up_if (_Walker __w, _Visitor __f, _Predicate __p)

perform a recursive postorder walk towards the root starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node
- vcollect is called after a child of a virtual node has finished
- vvalue is called to compute the return value of a virtual node
- init is called before the children are visited. Then the predicate is called. If this predicate returns true, the children are visited. Otherwise, the node is treated as if it was a terminal node.
- postorder is called after all children have been visited.
- collect is called everytime a child has finished.
- value is called to compute the return value for this node.

Definition at line 1785 of file vgtl_algo.h.

7.4.2.13 template<class _Walker , class _Visitor > _Visitor::return_value _recursive_preorder_walk (_Walker __w, _Visitor __f)

perform a recursive preorder walk starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node
- vcollect is called after a child of a virtual node has finished
- vvalue is called to compute the return value of a virtual node
- preorder is called before the children are visited
- collect is called everytime a child has finished
- value is called to compute the return value for this node this function does not check for hitting the virtual ground node.

Definition at line 569 of file vgtl_algo.h.

7.4.2.14 template<class _Walker , class _Visitor > _Visitor::return_value _recursive_preorder_walk_if (_Walker __w, _Visitor __f)

perform a recursive preorder walk starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node
- vcollect is called after a child of a virtual node has finished
- vvalue is called to compute the return value of a virtual node
- preorder is called before the children are visited. If this function returns true, the children are visited. Otherwise, the node is treated as if it was a terminal node.

- collect is called everytime a child has finished
- value is called to compute the return value for this node this function does not check for hitting the virtual ground node.

Definition at line 773 of file vgtl_algo.h.

7.4.2.15 template<class _Walker , class _Visitor , class _Predicate > _Visitor::return_value _recursive_preorder_walk_if (_Walker __w, _Visitor __f, _Predicate __p)

perform a recursive preorder walk starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node
- vcollect is called after a child of a virtual node has finished
- vvalue is called to compute the return value of a virtual node
- preorder is called before the children are visited. Then the predicate is called. If this predicate returns true, the children are visited. Otherwise, the node is treated as if it was a terminal node.
- collect is called everytime a child has finished
- value is called to compute the return value for this node this function does not check for hitting the virtual ground node.

Definition at line 848 of file vgtl_algo.h.

7.4.2.16 template<class _Walker , class _Visitor > _Visitor::return_value _recursive_preorder_walk_up (_Walker __w, _Visitor __f)

perform a recursive preorder walk towards the root starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node
- vcollect is called after a child of a virtual node has finished
- vvalue is called to compute the return value of a virtual node
- preorder is called before the children are visited
- collect is called everytime a child has finished
- value is called to compute the return value for this node this function does not check for hitting the virtual sky node.

Definition at line 1494 of file vgtl_algo.h.

7.4.2.17 template<class _Walker , class _Visitor > _Visitor::return_value _recursive_preorder_walk_up_if (_Walker __w, _Visitor __f)

perform a recursive preorder walk towards the root starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node

- vcollect is called after a child of a virtual node has finished
- vvalue is called to compute the return value of a virtual node
- preorder is called before the children are visited. If this function returns true, the children are visited. Otherwise, the node is treated as if it was a terminal node.
- collect is called everytime a child has finished
- value is called to compute the return value for this node this function does not check for hitting the virtual sky node.

Definition at line 1564 of file vgtl_algo.h.

7.4.2.18 template<class _Walker , class _Visitor , class _Predicate > _Visitor::return_value _recursive_preorder_walk_up_if (_Walker __w, _Visitor __f, _Predicate __p)

perform a recursive preorder walk towards the root starting at ___w. At every node various methods of the visitor ___f are called:

- vinit is called before walking for every virtual node
- vcollect is called after a child of a virtual node has finished
- vvalue is called to compute the return value of a virtual node
- preorder is called before the children are visited. Then the predicate is called. If this predicate returns true, the children are visited. Otherwise, the node is treated as if it was a terminal node.
- collect is called everytime a child has finished
- value is called to compute the return value for this node this function does not check for hitting the virtual sky node.

Definition at line 1639 of file vgtl_algo.h.

7.4.2.19 template<class _Walker , class _Visitor > _Visitor::return_value _recursive_safe_walk_if (_Walker __w, _Visitor __f)

perform a recursive pre+post order walk starting at ___w. At every node various methods of the visitor ___f are called:

- vinit is called before walking for every virtual node
- vcollect is called after a child of a virtual node has finished
- vvalue is called to compute the return value of a virtual node
- preorder is called before the children are visited. If it returns true, the children are visited. If it returns false, the children are ignored
- collect is called everytime a child has finished
- postorder is called after the children have been visited. If it returns true, the walk is continued by switching back to preorder mode for this node. If it returns false, the walk is over for this node.
- value is called to compute the return value for this node this function does not check for hitting the virtual ground node.

Definition at line 114 of file vgtl_addalgo.h.

7.4.2.20 template<class _Walker , class _Visitor > _Visitor::return_value _recursive_walk (_Walker __w, _Visitor __f)

perform a recursive pre+post order walk starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node
- vcollect is called after a child of a virtual node has finished
- vvalue is called to compute the return value of a virtual node
- preorder is called before the children are visited
- collect is called everytime a child has finished
- postorder is called after all children have been visited
- value is called to compute the return value for this node this function does not check for hitting the virtual ground node.

Definition at line 703 of file vgtl_algo.h.

7.4.2.21 template<class _Walker , class _Visitor > _Visitor::return_value _recursive_walk_if (_Walker __w, _Visitor __f)

perform a recursive pre+post order walk starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node
- vcollect is called after a child of a virtual node has finished
- vvalue is called to compute the return value of a virtual node
- preorder is called before the children are visited. If it returns true, the children are visited. If it returns false, the children are ignored
- collect is called everytime a child has finished
- postorder is called after the children have been visited. If it returns true, the walk is continued by switching back to preorder mode for this node. If it returns false, the walk is over for this node.
- value is called to compute the return value for this node this function does not check for hitting the virtual ground node.

Definition at line 1013 of file vgtl_algo.h.

7.4.2.22 template<class _Walker , class _Visitor , class _Predicate1 , class _Predicate2 > _Visitor::return_value _recursive_walk_if (_Walker __w, _Visitor __f, _Predicate1 __p1, _Predicate2 __p2)

perform a recursive pre+post order walk starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node
- vcollect is called after a child of a virtual node has finished

- vvalue is called to compute the return value of a virtual node
- preorder is called before the children are visited. If then predicate p1 returns true, the children are visited. If it returns false, the children are ignored
- collect is called everytime a child has finished
- postorder is called after the children have been visited. If then predicate p2 returns true, the walk is continued by switching back to preorder mode for this node. If it returns false, the walk is over for this node.
- value is called to compute the return value for this node this function does not check for hitting the virtual ground node.

Definition at line 1259 of file vgtl_algo.h.

7.4.2.23 template<class _Walker , class _Visitor > _Visitor::return_value _recursive_walk_up (_Walker __w, _Visitor __f)

perform a recursive pre+post order walk towards the root starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node
- vcollect is called after a child of a virtual node has finished
- vvalue is called to compute the return value of a virtual node
- preorder is called before the children are visited
- collect is called everytime a child has finished
- postorder is called after all children have been visited
- value is called to compute the return value for this node this function does not check for hitting the virtual sky node.

Definition at line 1855 of file vgtl_algo.h.

7.4.2.24 template<class _Walker , class _Visitor > _Visitor::return_value _recursive_walk_up_if (_Walker __w, _Visitor __f)

perform a recursive pre+post order walk towards the root starting at __w. At every node various methods of the visitor __f are called:

- vinit is called before walking for every virtual node.
- vcollect is called after a child of a virtual node has finished.
- vvalue is called to compute the return value of a virtual node.
- preorder is called before the children are visited. If it returns true, the children are visited. If it returns false, the children are ignored.
- collect is called everytime a child has finished.
- postorder is called after the children have been visited. If it returns true, the walk is continued by switching back to preorder mode for this node. If it returns false, the walk is over for this node.
- value is called to compute the return value for this node. this function does not check for hitting the virtual sky node.

Definition at line 1937 of file vgtl_algo.h.

7.4.2.25 template<class _Walker , class _Visitor , class _Predicate1 , class _Predicate2 > _Visitor::return_value _recursive_walk_up_if (_Walker __w, _Visitor __f, _Predicate1 __p1, _Predicate2 __p2)

perform a recursive pre+post order walk towards the root starting at `__w`. At every node various methods of the visitor `__f` are called:

- `vinit` is called before walking for every virtual node
- `vcollect` is called after a child of a virtual node has finished
- `vvalue` is called to compute the return value of a virtual node
- `preorder` is called before the children are visited. If then predicate `p1` returns `true`, the children are visited. If it returns `false`, the children are ignored
- `collect` is called everytime a child has finished
- `postorder` is called after the children have been visited. If then predicate `p2` returns `true`, the walk is continued by switching back to preorder mode for this node. If it returns `false`, the walk is over for this node.
- `value` is called to compute the return value for this node this function does not check for hitting the virtual sky node.

Definition at line 2028 of file vgtl_algo.h.

7.4.2.26 template<class _BidirIter , class _Tp > _BidirIter rfind (_BidirIter __first, _BidirIter __last, const _Tp & __val, std::bidirectional_iterator_tag) [inline]

This is an overload used by `rfind()` (reverse find) for the Bidirectional Iterator case. `rfind()` works like the STL `find()` algorithm, just backwards.

Definition at line 45 of file vgtl_helpers.h.

7.4.2.27 template<class _RandomAccessIter , class _Tp > _RandomAccessIter rfind (_RandomAccessIter __first, _RandomAccessIter __last, const _Tp & __val, std::random_access_iterator_tag)

This is an overload used by `rfind()` (reverse find) for the Random Access Iterator case. `rfind()` works like the STL `find()` algorithm, just backwards.

Definition at line 87 of file vgtl_helpers.h.

7.4.2.28 template<class _BidirIter , class _Predicate > _BidirIter rfind_if (_BidirIter __first, _BidirIter __last, _Predicate __pred, std::bidirectional_iterator_tag) [inline]

This is an overload used by `rfind_if()` (reverse find if) for the Bidirectional Iterator case. `rfind_if()` works like the STL `find_if()` algorithm, just backwards.

Definition at line 65 of file vgtl_helpers.h.

7.4.2.29 template<class _RandomAccessIter , class _Predicate > _RandomAccessIter rfind_if (_RandomAccessIter __first, _RandomAccessIter __last, _Predicate __pred, std::random_access_iterator_tag)

This is an overload used by `rfind_if()` (reverse find if) for the Random Access Iterator case. `rfind_if()` works like the STL `find_if()` algorithm, just backwards.

Definition at line 137 of file vgtl_helpers.h.

8 Namespace Documentation

8.1 vgtl Namespace Reference

Main namespace of the VGTL.

8.1.1 Detailed Description

This is the main namespace holding all classes and functions of the Vienna Graph Template Library (VGTL)

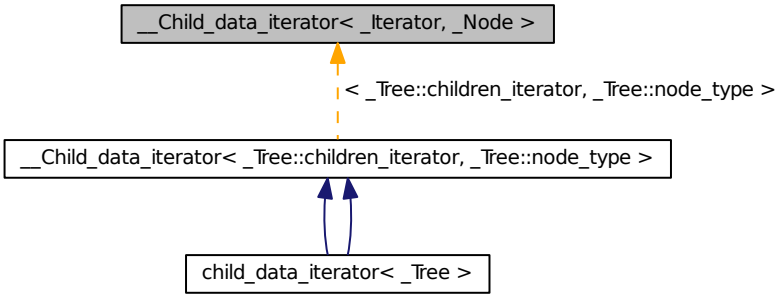
9 Class Documentation

9.1 __Child_data_iterator< _Iterator, _Node > Class Template Reference

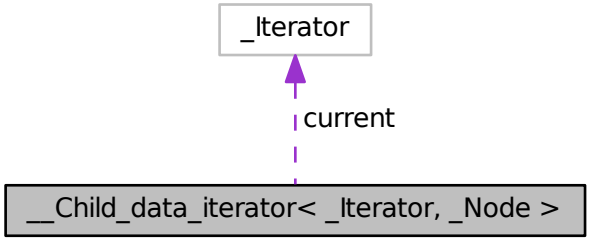
iterator adapter for iterating through children data hooks

`#include <vgtl_algo.h>`

Inheritance diagram for __Child_data_iterator< _Iterator, _Node >:



Collaboration diagram for __Child_data_iterator< _Iterator, _Node >:



Public Types

- typedef ctree_data_hook value_type
- typedef value_type * pointer
- typedef value_type & reference

- typedef ctree_data_hook value_type
- typedef value_type * pointer
- typedef value_type & reference

Public Member Functions

- __Child_data_iterator (const _Self &__x)
  standard destructor
- iterator_type base () const
  return the 'unwrapped' iterator
- reference operator* () const
  dereference to the `data_hook`.
- _Self & operator= (const iterator_type &it)
  assignment operator
- __Child_data_iterator (const _Self &__x)
  standard destructor
- iterator_type base () const
  return the 'unwrapped' iterator
- reference operator* () const
  dereference to the `data_hook`.
- _Self & operator= (const iterator_type &it)
  assignment operator

- __Child_data_iterator ()
  standard constructors
- __Child_data_iterator (iterator_type __x)
  standard constructors

- bool operator== (const _Self &__x) const
  standard comparison operator
- bool operator!= (const _Self &__x) const
  standard comparison operator

- _Self & operator++ ()
  standard in(de)crement operator
- _Self & operator++ (int)
  standard in(de)crement operator
- _Self & operator-- ()
  standard in(de)crement operator
- _Self & operator-- (int)
  standard in(de)crement operator

- _Self operator+ (difference_type __n) const
  additional operator for random access iterators
- _Self & operator+= (difference_type __n)
  additional operator for random access iterators
- _Self operator- (difference_type __n) const
  additional operator for random access iterators
- _Self & operator-= (difference_type __n)
  additional operator for random access iterators
- reference operator[ ] (difference_type __n) const
  additional operator for random access iterators

- __Child_data_iterator ()
  standard constructors
- __Child_data_iterator (iterator_type __x)
  standard constructors

- bool operator== (const _Self &__x) const
  standard comparison operator
- bool operator!= (const _Self &__x) const
  standard comparison operator

- _Self & operator++ ()
  standard in(de)crement operator
- _Self & operator++ (int)
  standard in(de)crement operator
- _Self & operator-- ()
  standard in(de)crement operator
- _Self & operator-- (int)
  standard in(de)crement operator

- _Self operator+ (difference_type __n) const
  additional operator for random access iterators
- _Self & operator+= (difference_type __n)
  additional operator for random access iterators
- _Self operator- (difference_type __n) const
  additional operator for random access iterators
- _Self & operator-= (difference_type __n)
  additional operator for random access iterators
- reference operator[ ] (difference_type __n) const
  additional operator for random access iterators

Protected Attributes

- _Iterator current
  that's where we are

9.1.1 Detailed Description

template<class _Iterator, class _Node>class __Child_data_iterator< _Iterator, _Node >

internal This class is an iterator adapter for iterating through the data hooks of all children of a given node

Definition at line 51 of file vgtl_lalgo.h.

9.1.2 Member Typedef Documentation

9.1.2.1 template<class _Iterator, class _Node> typedef value_type* __Child_data_iterator< _Iterator, _Node >::pointer

standard iterator definitions

Definition at line 64 of file vgtl_algo.h.

9.1.2.2 template<class _Iterator, class _Node> typedef value_type* __Child_data_iterator< _Iterator, _Node >::pointer

standard iterator definitions

Definition at line 64 of file vgtl_lalgo.h.

9.1.2.3 template<class _Iterator, class _Node> typedef value_type& __Child_data_iterator< _Iterator, _Node >::reference

standard iterator definitions

Definition at line 65 of file vgtl_lalgo.h.

9.1.2.4 template<class _Iterator, class _Node> typedef value_type& __Child_data_iterator< _Iterator, _Node >::reference

standard iterator definitions

Definition at line 65 of file vgtl_algo.h.

9.1.2.5 template<class _Iterator, class _Node> typedef ctree_data_hook __Child_data_iterator< _Iterator, _Node >::value_type

standard iterator definitions

Definition at line 63 of file vgtl_algo.h.

9.1.2.6 template<class _Iterator, class _Node> typedef ctree_data_hook __Child_data_iterator< _Iterator, _Node >::value_type

standard iterator definitions

Definition at line 63 of file vgtl_lalgo.h.

The documentation for this class was generated from the following files:

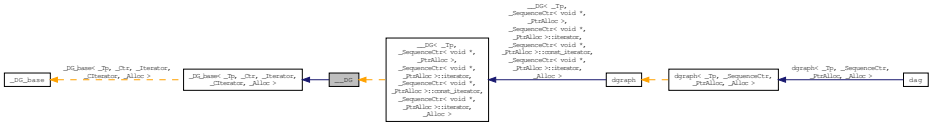
- vgtl_algo.h
- vgtl_lalgo.h

9.2 __DG Class Reference

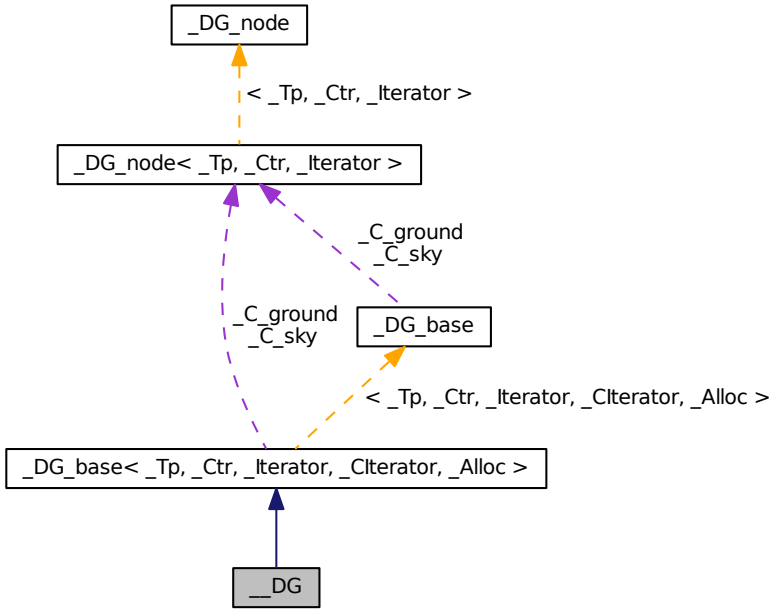
Directed graph base class.

`#include <vgtl_dag.h>`

Inheritance diagram for __DG:



Collaboration diagram for __DG:



Public Types

- typedef _Ctr container_type
- typedef _Iterator children_iterator
- typedef _Iterator parents_iterator
- typedef _Base::allocator_type allocator_type
- typedef _DG_iterator< _Tp, _Tp &, _Tp *, container_type, children_iterator, children_const_iterator > iterator
- typedef _DG_iterator< _Tp, const _Tp &, const _Tp *, container_type, children_iterator, children_const_iterator > const_iterator
- typedef std::reverse_iterator < const_iterator > const_reverse_iterator
- typedef std::reverse_iterator < iterator > reverse_iterator
- typedef _DG_walker< _Tp, _Tp &, _Tp *, container_type, children_iterator, children_const_iterator > walker
- typedef _DG_walker< _Tp, const _Tp &, const _Tp *, container_type, children_iterator, children_const_iterator > const_walker
- typedef std::pair< walker, walker > edge
- typedef std::pair< edge, bool > enhanced_edge

- typedef _Tp value_type
- typedef _Node node_type
- typedef value_type * pointer

- typedef const value_type * const_pointer
- typedef value_type & reference
- typedef const value_type & const_reference
- typedef size_t size_type
- typedef ptrdiff_t difference_type

Public Member Functions

- allocator_type get_allocator () const
- __DG (const allocator_type &__a=allocator_type())
- walker ground ()
- walker sky ()
- const_walker ground () const
- const_walker sky () const
- children_iterator root_begin ()
- children_iterator root_end ()
- children_const_iterator root_begin () const
- children_const_iterator root_end () const
- parents_iterator leaf_begin ()
- parents_iterator leaf_end ()
- parents_const_iterator leaf_begin () const
- parents_const_iterator leaf_end () const
- bool empty () const
- size_type size () const
- size_type max_size () const
- void swap (_Self &__x)
- walker insert_node_in_graph (_Node *__n, const walker &__parent, const walker &__child, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- walker insert_in_graph (const _Tp &__x, const walker &__parent, const walker &__child, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- walker insert_in_graph (const walker &__parent, const walker &__child, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- void insert_subgraph (_Self &__subgraph, const walker &__parent, const walker &__child, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr1, template< class __Tp, class __AllocTp > class __SequenceCtr2, class _Allocator1 , class _Allocator2 >
  walker insert_node_in_graph (_Node *__node, const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr1, template< class __Tp, class __AllocTp > class __SequenceCtr2, class _Allocator1 , class _Allocator2 >
  walker insert_in_graph (const _Tp &__x, const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr1, template< class __Tp, class __AllocTp > class __SequenceCtr2, class _Allocator1 , class _Allocator2 >
  walker insert_in_graph (const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker insert_node_in_graph (_Node *__node, const walker &__parent, const container_insert_arg &__pref, const __SequenceCtr< walker, _Allocator > &__children)

- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker insert_in_graph (const _Tp &__x, const walker &__parent, const container_insert_arg &__pref, const __SequenceCtr< walker, _Allocator > &__children)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker insert_in_graph (const walker &__parent, const container_insert_arg &__pref, const __SequenceCtr< walker, _Allocator > &__children)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker insert_node_in_graph (_Node *__node, const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const container_insert_arg &__cref)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker insert_in_graph (const _Tp &__x, const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const container_insert_arg &__cref)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker insert_in_graph (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const container_insert_arg &__cref)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr1, template< class __Tp, class __AllocTp > class __SequenceCtr2, class _Allocator1 , class _Allocator2 >
  void insert_subgraph (_Self &__subgraph, const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children)
- void add_edge (const edge &__edge, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- void add_edge (const walker &__parent, const walker &__child, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- void replace_edge_to_child (const walker &__parent, const walker &__child_old, const walker &__child_new)
- void replace_edge_to_parent (const walker &__parent_old, const walker &__parent_new, const walker &__child)
- void remove_edge (const edge &__edge)
- void remove_edge_and_deattach (const walker &__parent, const walker &__child)
- void remove_edge (const walker &__parent, const walker &__child)
- template<class Compare >
  void sort_child_edges (walker __position, children_iterator first, children_iterator last, Compare comp)
- template<class Compare >
  void sort_parent_edges (walker __position, parents_iterator first, parents_iterator last, Compare comp)
- template<class Compare >
  void sort_child_edges (walker __position, Compare comp)
- template<class Compare >
  void sort_parent_edges (walker __position, Compare comp)
- walker insert_node (_Node *_node, const walker &__position, const container_insert_arg &__It)
- walker insert_node (const _Tp &__x, const walker &__position, const container_insert_arg &__It)
- walker insert_node (const walker &__position, const container_insert_arg &__It)
- walker insert_node_before (_Node *_node, const walker &__position, const container_insert_arg &__It)
- void insert_node_before (const _Tp &__x, const walker &__position, const container_insert_arg &__It)
- void insert_node_before (const walker &__position, const container_insert_arg &__It)
- void merge (const walker &__position, const walker &__second, bool merge_parent_edges=true, bool merge_child_edges=true)
- void erase (const walker &__position)
- void partial_erase_to_parent (const walker &__position, const walker &__parent, unsigned int idx)
- void clear_erased_part (erased_part &_ep)

- erased_part erase_maximal_subgraph (const walker &__position)
- erased_part erase_minimal_subgraph (const walker &__position)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > erased_part erase_maximal_subgraph (const __SequenceCtr< walker, _Allocator > &__positions)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > erased_part erase_minimal_subgraph (const __SequenceCtr< walker, _Allocator > &__positions)
- erased_part erase_maximal_pregraph (const walker &__position)
- erased_part erase_minimal_pregraph (const walker &__position)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > erased_part erase_maximal_pregraph (const __SequenceCtr< walker, _Allocator > &__positions)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > erased_part erase_minimal_pregraph (const __SequenceCtr< walker, _Allocator > &__positions)
- bool erase_child (const walker &__position, const children_iterator &__It)
- bool erase_parent (const walker &__position, const parents_iterator &__It)
- void copy_maximal_subgraph (const walker &__x, const walker &__par, const walker &__bo, const walker &__bn)
- void clear ()
- __DG (const _Self &__x)
- ~__DG ()
- _Self & operator= (const _Self &__x)
- _Self & operator= (const _RV_DG &__rl)
- _Self & operator= (const erased_part &__ep)

Protected Types

- typedef std::pair< _RV_DG, std::vector< enhanced_edge > > erased_part

Protected Member Functions

- _Node * _C_create_node (const _Tp &__x)
- _Node * _C_create_node ()
- void _C_destroy_node (_Node *__p)
- void clear_graph (_DG_node< _Tp, _Ctr, _Iterator > *_node)
- _DG_node< _Tp, _Ctr, _Iterator > * _C_get_node ()
- void _C_put_node (_DG_node< _Tp, _Ctr, _Iterator > *__p)
- void clear_children ()
- void clear_parents ()
- void add_all_children (_Output_Iterator fi, _DG_node< _Tp, _Ctr, _Iterator > *_parent)
- void add_all_parents (_Output_Iterator fi, _DG_node< _Tp, _Ctr, _Iterator > *_child)

Protected Attributes

- _DG_node< _Tp, _Ctr, _Iterator > * _C_ground
- _DG_node< _Tp, _Ctr, _Iterator > * _C_sky
- int _C_mark

9.2.1 Detailed Description

This is the toplevel base class for all directed graphs independent of allocators

9.2.2 Member Typedef Documentation

9.2.2.1 typedef _Base::allocator_type __DG::allocator_type

allocator type

Reimplemented from _DG_base< _Tp, _Ctr, _Iterator, _CIterator, _Alloc >.

Definition at line 590 of file vgtl_dag.h.

9.2.2.2 typedef _Iterator __DG::children_iterator

iterator for accessing the children

Reimplemented from _DG_base< _Tp, _Ctr, _Iterator, _CIterator, _Alloc >.

Reimplemented in dgraph.

Definition at line 561 of file vgtl_dag.h.

9.2.2.3 typedef _DG_iterator<_Tp,const _Tp&,const _Tp*,container_type, children_iterator,children_const_iterator> __DG::const_iterator

the const iterator

Definition at line 600 of file vgtl_dag.h.

9.2.2.4 typedef const value_type* __DG::const_pointer

standard typedef

Definition at line 583 of file vgtl_dag.h.

9.2.2.5 typedef const value_type& __DG::const_reference

standard typedef

Definition at line 585 of file vgtl_dag.h.

9.2.2.6 typedef std::reverse_iterator<const_iterator> __DG::const_reverse_iterator

the const reverse iterator

Definition at line 604 of file vgtl_dag.h.

9.2.2.7 typedef _DG_walker<_Tp,const _Tp&,const _Tp*,container_type,children_iterator, children_const_iterator> __DG::const_walker

the (recursive) const walker

Reimplemented in dgraph.

Definition at line 623 of file vgtl_dag.h.

9.2.2.8 typedef _Ctr __DG::container_type

internal container used to store the children

Reimplemented from _DG_base< _Tp, _Ctr, _Iterator, _CIterator, _Alloc >.

Definition at line 560 of file vgtl_dag.h.

9.2.2.9 typedef ptrdiff_t __DG::difference_type

standard typedef

Definition at line 587 of file vgtl_dag.h.

9.2.2.10 typedef std::pair<walker,walker> __DG::edge

an edge of the graph (parent, child)

Definition at line 626 of file vgtl_dag.h.

9.2.2.11 typedef std::pair<edge,bool> __DG::enhanced_edge

an edge with additiona information about erased ground/sky edges

Definition at line 628 of file vgtl_dag.h.

9.2.2.12 typedef std::pair<_RV_DG, std::vector<enhanced_edge> > __DG::erased_part [protected]

an erased subgraph which is not yet a new directed graph

Reimplemented in dgraph.

Definition at line 632 of file vgtl_dag.h.

9.2.2.13 typedef _DG_iterator<_Tp,_Tp&,_Tp*,container_type,children_iterator, children_const_iterator> __DG::iterator

the iterator

Definition at line 597 of file vgtl_dag.h.

9.2.2.14 typedef _Node __DG::node_type

standard typedef

Definition at line 581 of file vgtl_dag.h.

9.2.2.15 typedef _Iterator __DG::parents_iterator

iterator for accessing the parents

Reimplemented from _DG_base< _Tp, _Ctr, _Iterator, _CIterator, _Alloc >.

Reimplemented in dgraph.

Definition at line 562 of file vgtl_dag.h.

9.2.2.16 typedef value_type* __DG::pointer

standard typedef

Definition at line 582 of file vgtl_dag.h.

9.2.2.17 typedef value_type& __DG::reference

standard typedef

Definition at line 584 of file vgtl_dag.h.

9.2.2.18 typedef std::reverse_iterator<iterator> __DG::reverse_iterator

the reverse iterator

Definition at line 606 of file vgtl_dag.h.

9.2.2.19 typedef size_t __DG::size_type

standard typedef

Definition at line 586 of file vgtl_dag.h.

9.2.2.20 typedef _Tp __DG::value_type

standard typedef

Definition at line 580 of file vgtl_dag.h.

9.2.2.21 typedef _DG_walker<_Tp,_Tp&,_Tp*,container_type,children_iterator, children_const_iterator> __DG::walker

the (recursive) walker

Reimplemented in dgraph.

Definition at line 620 of file vgtl_dag.h.

9.2.3 Constructor & Destructor Documentation

9.2.3.1 __DG::__DG (const allocator_type & __a = allocator_type()) [inline, explicit]

standard constructor

Definition at line 684 of file vgtl_dag.h.

9.2.3.2 __DG::__DG (const _Self & __x) [inline]

copy constructor

Definition at line 2009 of file vgtl_dag.h.

9.2.3.3 __DG::~__DG ( ) [inline]

standard destructor

Definition at line 2026 of file vgtl_dag.h.

9.2.4 Member Function Documentation

9.2.4.1 _Node* __DG::_C_create_node (const _Tp & __x) [inline, protected]

construct a new tree node containing data ___x

Definition at line 645 of file vgtl_dag.h.

9.2.4.2 _Node* __DG::_C_create_node ( ) [inline, protected]

construct a new tree node containing default data

Definition at line 659 of file vgtl_dag.h.

9.2.4.3 void __DG::_C_destroy_node (_Node * __p) [inline, protected]

construct a new tree node containing default data

Definition at line 673 of file vgtl_dag.h.

9.2.4.4 _DG_node<_Tp,_Ctr,_Iterator>* _DG_base::_C_get_node ( ) [inline, protected, inherited]

allocate a new node

Definition at line 405 of file vgtl_dagbase.h.

9.2.4.5 void _DG_base::_C_put_node (_DG_node< _Tp, _Ctr, _Iterator > * __p) [inline, protected, inherited]

deallocate a node

Definition at line 408 of file vgtl_dagbase.h.

9.2.4.6 void _DG_base< _Tp, _Ctr, _Iterator, _CIterator, _Alloc >::add_all_children (_Output_Iterator *fi,* _DG_node< _Tp, _Ctr, _Iterator > * *_parent*) [inline, protected, inherited]

add all children to the parent `_parent`. `fi` is a iterator to the children container of the parent

Definition at line 475 of file vgtl_dagbase.h.

9.2.4.7 void _DG_base< _Tp, _Ctr, _Iterator, _CIterator, _Alloc >::add_all_parents (_Output_Iterator *fi,* _DG_node< _Tp, _Ctr, _Iterator > * *_child*) [inline, protected, inherited]

add all parents to the child `_child`. `fi` is a iterator to the parents container of the child

Definition at line 484 of file vgtl_dagbase.h.

9.2.4.8 void __DG::add_edge (const edge & *__edge,* const container_insert_arg & *__itc,* const container_insert_arg & *__itp*) [inline]

add one edge between two nodes at the positions described by `__Itc` and `__Itp`.

Definition at line 1070 of file vgtl_dag.h.

9.2.4.9 void __DG::add_edge (const walker & *__parent,* const walker & *__child,* const container_insert_arg & *__itc,* const container_insert_arg & *__itp*) [inline]

add an edge between `__parent` and `__child` at positions `__Itc` and `__Itp`, respectively

Definition at line 1079 of file vgtl_dag.h.

9.2.4.10 void __DG::clear ( ) [inline]

erase all the nodes except sky and ground

Reimplemented from _DG_base< _Tp, _Ctr, _Iterator, _CIterator, _Alloc >.

Reimplemented in dgraph.

Definition at line 1967 of file vgtl_dag.h.

9.2.4.11 void _DG_base::clear_children ( ) [inline, protected, inherited]

clear all children of the root node

Definition at line 420 of file vgtl_dagbase.h.

9.2.4.12 void __DG::clear_erased_part (erased_part & _ep) [inline]

clear all nodes in an erased part

Definition at line 1751 of file vgtl_dag.h.

9.2.4.13 void _DG_base< _Tp, _Ctr, _Iterator, _CIterator, _Alloc >::clear_graph (_DG_node< _Tp, _Ctr, _Iterator > * _node) [protected, inherited]

removes recursively all nodes downward starting from _node.

Definition at line 444 of file vgtl_dagbase.h.

9.2.4.14 void _DG_base::clear_parents ( ) [inline, protected, inherited]

clear all parents of the leaf node

Definition at line 423 of file vgtl_dagbase.h.

9.2.4.15 void __DG::copy_maximal_subgraph (const walker & __x, const walker & __par, const walker & __bo, const walker & __bn) [inline]

This function returns a copy of the maximal subgraph between the nodes __xn and __bo. Here __bo is connected to the new node __bn. __par is the new parent of the copied subgraph.

Definition at line 1956 of file vgtl_dag.h.

9.2.4.16 bool __DG::empty ( ) const [inline]

returns true if the DG is empty

Definition at line 767 of file vgtl_dag.h.

9.2.4.17 void __DG::erase (const walker & __position) [inline]

erase a node from the DG except the sky and ground

Definition at line 1400 of file vgtl_dag.h.

9.2.4.18 bool __DG::erase_child (const walker & __position, const children_iterator & __it) [inline]

Erase a child of __position. This works if and only if the child has only one child and no other parents.

Definition at line 1904 of file vgtl_dag.h.

9.2.4.19 erased_part __DG::erase_maximal_pregraph (const walker & __position) [inline]

here every child is removed till the sky node. included the node at __position. The removed subgraph is returned. The subgraph is maximal, i.e. all nodes are removed, which are reachable from __position by walking upwards.

Definition at line 1834 of file vgtl_dag.h.

9.2.4.20 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > erased_part __DG::erase_maximal_pregraph (const __SequenceCtr< walker, _Allocator > & *__positions*) [inline]

here every child is removed till the sky included all nodes from __positions. The removed subgraph is returned. The subgraph is maximal, i.e. all nodes are removed, which are reachable from any node from __positions by walking up.

Definition at line 1868 of file vgtl_dag.h.

9.2.4.21 erased_part __DG::erase_maximal_subgraph (const walker & *__position*) [inline]

here every child is removed till the last base node, included the node at __position. The removed subgraph is returned. The subgraph is maximal, i.e. all nodes are removed, which are reachable from __position by walking down.

Definition at line 1763 of file vgtl_dag.h.

9.2.4.22 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > erased_part __DG::erase_maximal_subgraph (const __SequenceCtr< walker, _Allocator > & *__positions*) [inline]

here every child is removed till the last base node, included all nodes from __positions. The removed subgraph is returned. The subgraph is maximal, i.e. all nodes are removed, which are reachable from any node from __positions by walking down.

Definition at line 1797 of file vgtl_dag.h.

9.2.4.23 erased_part __DG::erase_minimal_pregraph (const walker & *__position*) [inline]

here every child is removed till the sky. included the node at __position. The removed subgraph is returned. The subgraph is minimal, i.e. only nodes are removed, which have no other descendant than __position. I.e., when walking towards the sky, there is no way which bypasses __position.

Definition at line 1850 of file vgtl_dag.h.

9.2.4.24 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > erased_part __DG::erase_minimal_pregraph (const __SequenceCtr< walker, _Allocator > & *__positions*) [inline]

here every child is removed till the sky. included all nodes from __positions. The removed subgraph is returned. The subgraph is minimal, i.e. only nodes are removed, which have no other ancestor than any node in __positions. I.e., when walking towards the ground, there is no way which bypasses all nodes in __positions.

Definition at line 1888 of file vgtl_dag.h.

9.2.4.25 erased_part __DG::erase_minimal_subgraph (const walker & *__position*) [inline]

here every child is removed till the last base node, included the node at __position. The removed subgraph is returned. The subgraph is minimal, i.e. only nodes are removed, which have no other ancestor than __position. I.e., when walking towards the ground, there is no way which bypasses __position.

Definition at line 1779 of file vgtl_dag.h.

9.2.4.26 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > erased_part __DG::erase_minimal_subgraph (const __SequenceCtr< walker, _Allocator > & *__positions*) [inline]

here every child is removed till the last base node, included all nodes from __positions. The removed subgraph is returned. The subgraph is minimal, i.e. only nodes are removed, which have no other ancestor than any node in __positions. I.e., when walking towards the ground, there is no way which bypasses all nodes in __positions.

Definition at line 1817 of file vgtl_dag.h.

9.2.4.27 bool __DG::erase_parent (const walker & *__position,* const parents_iterator & *__it*) [inline]

Erase a parent of __position. This works if and only if the parent has only one parent and no other children.

Definition at line 1930 of file vgtl_dag.h.

9.2.4.28 allocator_type __DG::get_allocator ( ) const [inline]

construct an allocator object

Reimplemented from _DG_base< _Tp, _Ctr, _Iterator, _CIterator, _Alloc >.

Definition at line 592 of file vgtl_dag.h.

9.2.4.29 walker __DG::ground ( ) [inline]

return a walker to the virtual ground node.

Definition at line 687 of file vgtl_dag.h.

9.2.4.30 const_walker __DG::ground ( ) const [inline]

return a const walker to the virtual ground node.

Definition at line 697 of file vgtl_dag.h.

9.2.4.31 walker __DG::insert_in_graph (const _Tp & *__x,* const walker & *__parent,* const walker & *__child,* const container_insert_arg & *__itc,* const container_insert_arg & *__itp*) [inline]

insert node with data __n into the graph between __parent and __child, the edge at the specific positions described by __Itc and __Itp.

Definition at line 807 of file vgtl_dag.h.

9.2.4.32 walker __DG::insert_in_graph (const walker & *__parent,* const walker & *__child,* const container_insert_arg & *__itc,* const container_insert_arg & *__itp*) [inline]

insert node with default data into the graph between __parent and __child, the edge at the specific positions described by __Itc and __Itp.

Definition at line 821 of file vgtl_dag.h.

9.2.4.33 template<template< class __Tp, class __AllocTp > class __SequenceCtr1, template< class __Tp, class __AllocTp > class __SequenceCtr2, class _Allocator1 , class _Allocator2 > walker __DG::insert_in_graph (const _Tp & *__x,* const __SequenceCtr1< walker, _Allocator1 > & *__parents,* const __SequenceCtr2< walker, _Allocator2 > & *__children*) [inline]

insert a node with data ___x into the graph between all parents from ___parents and all children from ___children.

Definition at line 885 of file vgtl_dag.h.

9.2.4.34 template<template< class __Tp, class __AllocTp > class __SequenceCtr1, template< class __Tp, class __AllocTp > class __SequenceCtr2, class _Allocator1 , class _Allocator2 > walker __DG::insert_in_graph (const __SequenceCtr1< walker, _Allocator1 > & *__parents,* const __SequenceCtr2< walker, _Allocator2 > & *__children*) [inline]

insert a node with default data into the graph between all parents from ___parents and all children from ___children.

Definition at line 900 of file vgtl_dag.h.

9.2.4.35 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker __DG::insert_in_graph (const _Tp & *__x,* const walker & *__parent,* const container_insert_arg & *__pref,* const __SequenceCtr< walker, _Allocator > & *__children*) [inline]

insert a node with data ___x into the graph between the parent ___parent and all children from ___children.

Definition at line 938 of file vgtl_dag.h.

9.2.4.36 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker __DG::insert_in_graph (const walker & *__parent,* const container_insert_arg & *__pref,* const __SequenceCtr< walker, _Allocator > & *__children*) [inline]

insert a node with data ___x into the graph between the parent ___parent and all children from ___children.

Definition at line 952 of file vgtl_dag.h.

9.2.4.37 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker __DG::insert_in_graph (const _Tp & *__x,* const __SequenceCtr< walker, _Allocator > & *__parents,* const walker & *__child,* const container_insert_arg & *__cref*) [inline]

insert a node with data ___x into the graph between all parents from ___parents and the child ___child.

Definition at line 991 of file vgtl_dag.h.

9.2.4.38 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker __DG::insert_in_graph (const __SequenceCtr< walker, _Allocator > & *__parents,* const walker & *__child,* const container_insert_arg & *__cref*) [inline]

insert a node with default data into the graph between all parents from ___parents and the child ___child.

Definition at line 1006 of file vgtl_dag.h.

9.2.4.39 walker __DG::insert_node (_Node * *__node*, const walker & *__position*, const container_insert_arg & *__lt*) [inline]

insert one node as child of __position

Definition at line 1261 of file vgtl_dag.h.

9.2.4.40 walker __DG::insert_node (const _Tp & *__x*, const walker & *__position*, const container_insert_arg & *__lt*) [inline]

insert a new node with data __x as child of __position

Definition at line 1275 of file vgtl_dag.h.

9.2.4.41 walker __DG::insert_node (const walker & *__position*, const container_insert_arg & *__lt*) [inline]

insert a new node with default data as child of __position

Definition at line 1281 of file vgtl_dag.h.

9.2.4.42 walker __DG::insert_node_before (_Node * *__node*, const walker & *__position*, const container_insert_arg & *__lt*) [inline]

insert a node as parent of __position

Definition at line 1286 of file vgtl_dag.h.

9.2.4.43 void __DG::insert_node_before (const _Tp & *__x*, const walker & *__position*, const container_insert_arg & *__lt*) [inline]

insert a new node with data __x as parent of __position

Definition at line 1300 of file vgtl_dag.h.

9.2.4.44 void __DG::insert_node_before (const walker & *__position*, const container_insert_arg & *__lt*) [inline]

insert a new node with default data as parent of __position

Definition at line 1305 of file vgtl_dag.h.

9.2.4.45 walker __DG::insert_node_in_graph (_Node * *__n*, const walker & *__parent*, const walker & *__child*, const container_insert_arg & *__ltc*, const container_insert_arg & *__ltp*) [inline]

insert node __n into the graph between __parent and __child, the edge at the specific positions described by __Itc and __Itp.

Definition at line 791 of file vgtl_dag.h.

9.2.4.46 template<template< class __Tp, class __AllocTp > class __SequenceCtr1, template< class __Tp, class __AllocTp > class __SequenceCtr2, class _Allocator1 , class _Allocator2 > walker __DG::insert_node_in_graph (_Node * *__node*, const __SequenceCtr1< walker, _Allocator1 > & *__parents*, const __SequenceCtr2< walker, _Allocator2 > & *__children*) [inline]

insert node __n into the graph between all parents from __parents and all children from __children.

Definition at line 854 of file vgtl_dag.h.

9.2.4.47 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker __DG::insert_node_in_graph (_Node * *__node,* const walker & *__parent,* const container_insert_arg & *__pref,* const __SequenceCtr< walker, _Allocator > & *__children*) [inline]

insert node __n into the graph between the parent __parent and all children from __children.

Definition at line 913 of file vgtl_dag.h.

9.2.4.48 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker __DG::insert_node_in_graph (_Node * *__node,* const __SequenceCtr< walker, _Allocator > & *__parents,* const walker & *__child,* const container_insert_arg & *__cref*) [inline]

insert node __n into the graph between all parents from __parents and the child __child.

Definition at line 966 of file vgtl_dag.h.

9.2.4.49 void __DG::insert_subgraph (_Self & *__subgraph,* const walker & *__parent,* const walker & *__child,* const container_insert_arg & *__itc,* const container_insert_arg & *__itp*) [inline]

insert a subgraph into the graph between __parent and __child, the edge at the specific positions described by __Itc and __Itp.

Definition at line 832 of file vgtl_dag.h.

9.2.4.50 template<template< class __Tp, class __AllocTp > class __SequenceCtr1, template< class __Tp, class __AllocTp > class __SequenceCtr2, class _Allocator1 , class _Allocator2 > void __DG::insert_subgraph (_Self & *__subgraph,* const __SequenceCtr1< walker, _Allocator1 > & *__parents,* const __SequenceCtr2< walker, _Allocator2 > & *__children*) [inline]

in this method one DG is inserted into another DG between the parents __parents and the children __children.

Definition at line 1020 of file vgtl_dag.h.

9.2.4.51 parents_iterator __DG::leaf_begin ( ) [inline]

return the first leaf of the directed graph

Definition at line 721 of file vgtl_dag.h.

9.2.4.52 parents_const_iterator __DG::leaf_begin ( ) const [inline]

return the first leaf of the directed graph

Definition at line 728 of file vgtl_dag.h.

9.2.4.53 parents_iterator __DG::leaf_end ( ) [inline]

return beyond the last leaf of the directed graph

Definition at line 724 of file vgtl_dag.h.

9.2.4.54 parents_const_iterator __DG::leaf_end ( ) const [inline]

return beyond the last leaf of the directed graph

Definition at line 731 of file vgtl_dag.h.

9.2.4.55 size_type __DG::max_size ( ) const [inline]

the maximum size of a DG is virtually unlimited

Definition at line 778 of file vgtl_dag.h.

9.2.4.56 void __DG::merge (const walker & *__position*, const walker & *__second*, bool *merge_parent_edges* = true, bool *merge_child_edges* = true) [inline]

merge two nodes, call also the merge method for the node data

Definition at line 1311 of file vgtl_dag.h.

9.2.4.57 _Self& __DG::operator= (const _Self & *__x*)

standard assignment operator

9.2.4.58 _Self& __DG::operator= (const _RV_DG & *__rl*) [inline]

assignment operator from a part of an erased part

Reimplemented in dgraph.

Definition at line 2032 of file vgtl_dag.h.

9.2.4.59 _Self& __DG::operator= (const erased_part & *__ep*) [inline]

assignment operator from an erased part

Reimplemented in dgraph.

Definition at line 2040 of file vgtl_dag.h.

9.2.4.60 void __DG::partial_erase_to_parent (const walker & *__position*, const walker & *__parent*, unsigned int *idx*) [inline]

split a node in two, the first connected to the __parent, the second connected to all other parents. Then erase the first node.

Definition at line 1461 of file vgtl_dag.h.

9.2.4.61 void __DG::remove_edge (const edge & *__edge*) [inline]

remove an edge with a particular parent and child

Definition at line 1197 of file vgtl_dag.h.

9.2.4.62 void __DG::remove_edge (const walker & *__parent*, const walker & *__child*) [inline]

just remove one edge between ___parent and ___child

Definition at line 1214 of file vgtl_dag.h.

9.2.4.63 void __DG::remove_edge_and_deattach (const walker & *__parent*, const walker & *__child*) [inline]

remove one egde and don't reconnect the node to sky/ground

Definition at line 1201 of file vgtl_dag.h.

9.2.4.64 void __DG::replace_edge_to_child (const walker & *__parent,* const walker & *__child_old,* const walker & *__child_new*) [inline]

change the edge from __parent to __child_old to an edge from __parent to __child_new.

Definition at line 1125 of file vgtl_dag.h.

9.2.4.65 void __DG::replace_edge_to_parent (const walker & *__parent_old,* const walker & *__parent_new,* const walker & *__child*) [inline]

change the edge from __parent_old to __child to an edge from __parent_new to __child.

Definition at line 1163 of file vgtl_dag.h.

9.2.4.66 children_iterator __DG::root_begin ( ) [inline]

return the first root of the directed graph

Definition at line 707 of file vgtl_dag.h.

9.2.4.67 children_const_iterator __DG::root_begin ( ) const [inline]

return the first root of the directed graph

Definition at line 714 of file vgtl_dag.h.

9.2.4.68 children_iterator __DG::root_end ( ) [inline]

return beyond the last root of the directed graph

Definition at line 710 of file vgtl_dag.h.

9.2.4.69 children_const_iterator __DG::root_end ( ) const [inline]

return beyond the last root of the directed graph

Definition at line 717 of file vgtl_dag.h.

9.2.4.70 size_type __DG::size ( ) const [inline]

returns the size of the DG (number of nodes)

Definition at line 771 of file vgtl_dag.h.

9.2.4.71 walker __DG::sky ( ) [inline]

return a walker to the virtual sky node.

Definition at line 692 of file vgtl_dag.h.

9.2.4.72 const_walker __DG::sky ( ) const [inline]

return a const walker to the virtual sky node.

Definition at line 702 of file vgtl_dag.h.

9.2.4.73 template<class Compare > void __DG::sort_child_edges (walker *__position,* children_iterator *first,* children_iterator *last,* Compare *comp*) [inline]

sort the child edges in the range [first,last) according to comp

Definition at line 1238 of file vgtl_dag.h.

9.2.4.74 template<class Compare > void __DG::sort_child_edges (walker __position, Compare comp) [inline]

sort all child edges according to comp

Definition at line 1250 of file vgtl_dag.h.

9.2.4.75 template<class Compare > void __DG::sort_parent_edges (walker __position, parents_iterator first, parents_iterator last, Compare comp) [inline]

sort the parent edges in the range [first,last) according to comp

Definition at line 1244 of file vgtl_dag.h.

9.2.4.76 template<class Compare > void __DG::sort_parent_edges (walker __position, Compare comp) [inline]

sort all parent edges according to comp

Definition at line 1256 of file vgtl_dag.h.

9.2.4.77 void __DG::swap (_Self & __x) [inline]

swap two DGs

Definition at line 781 of file vgtl_dag.h.

9.2.5 Member Data Documentation

9.2.5.1 _DG_node<_Tp,_Ctr,_Iterator>* _DG_base::_C_ground [protected, inherited]

the virtual ground node (below all roots)

Definition at line 413 of file vgtl_dagbase.h.

9.2.5.2 int _DG_base::_C_mark [protected, inherited]

an internal counter for setting marks during certain algorithms

Definition at line 417 of file vgtl_dagbase.h.

9.2.5.3 _DG_node<_Tp,_Ctr,_Iterator>* _DG_base::_C_sky [protected, inherited]

the virtual sky node (above all leafs)

Definition at line 415 of file vgtl_dagbase.h.

The documentation for this class was generated from the following file:

- vgtl_dag.h

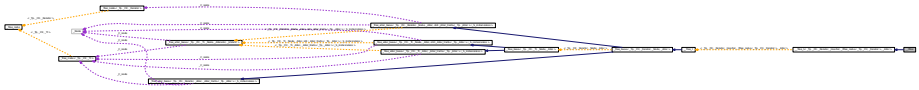
9.3 __ITree Class Reference

Tree base class with data hooks.

`#include <vgtl_tree.h>`

Inheritance diagram for __ITree:

Collaboration diagram for __ITree:



Public Types

- typedef _Node node_type
- typedef _Tree_iterator< _Tp, _Tp &, _Tp *, container_type, children_iterator, node_type > iterator
- typedef _Tree_iterator< _Tp, const _Tp &, const _Tp *, container_type, children_iterator, node_type > const_iterator
- typedef _Tree_walker< _Tp, _Tp &, _Tp *, container_type, children_iterator, _Node > iterative_walker
- typedef _Tree_walker< _Tp, const _Tp &, const _Tp *, container_type, children_iterator, _Node > const_iterative_walker
- typedef std::reverse_iterator < const_iterator > const_reverse_iterator
- typedef std::reverse_iterator < iterator > reverse_iterator
- typedef _RTree_walker< _Tp, _Tp &, _Tp *, container_type, children_iterator, node_type > walker
- typedef _RTree_walker< _Tp, const _Tp &, const _Tp *, container_type, children_iterator, node_type > const_walker

- typedef _Tp value_type
- typedef value_type * pointer
- typedef const value_type * const_pointer
- typedef value_type & reference
- typedef const value_type & const_reference
- typedef size_t size_type
- typedef ptrdiff_t difference_type

Public Member Functions

- __ITree (const allocator_type &__a=allocator_type())
- iterative_walker root (walker_type wt=cw_pre_post, bool front_to_back=true, bool depth_first=true)
- const_iterative_walker root (walker_type wt=cw_pre_post, bool front_to_back=true, bool depth_first=true) const
- iterative_walker through ()
- const_iterative_walker through () const
- iterative_walker begin (walker_type wt=cw_pre_post, bool front_to_back=true, bool depth_first=true)
- const_iterative_walker begin (walker_type wt=cw_pre_post, bool front_to_back=true, bool depth_first=true) const

- iterative_walker end (walker_type wt=cw_pre_post, bool front_to_back=true, bool depth_first=true)
- const_iterative_walker end (walker_type wt=cw_pre_post, bool front_to_back=true, bool depth_first=true) const
- reverse_iterator rbegin ()
- reverse_iterator rend ()
- const_reverse_iterator rbegin () const
- const_reverse_iterator rend () const
- size_type size () const
- reference getroot ()
- const_reference getroot () const
- size_type depth (const iterative_walker &__position)
- __ITree (size_type __n, const _Tp &__value, const allocator_type &__a=allocator_type())
- __ITree (size_type __n)
- __ITree (const _Self &__x)
- virtual ~__ITree ()
- _Self & operator= (const _Self &__x)
- _Self & operator= (_Node *__x)
- allocator_type get_allocator () const
- bool empty () const
- size_type max_size () const
- void swap (_Self &__x)
- void insert_child (const __walker_base &__position, const _Tp &__x, const container_insert_arg &__It)
- void insert_child (const __walker_base &__position, const container_insert_arg &__It)
- void insert_children (const __walker_base &__position, size_type __n, const _Tp &__x, const children_iterator &__It)
- void insert_subtree (const __walker_base &__position, _Self &__subtree, const children_iterator &__It)
- void erase (const __walker_base &__position)
- _Node * erase_tree (const __walker_base &__position)
- bool erase_child (const __walker_base &__position, const children_iterator &__It)
- _Node * erase_subtree (const __walker_base &__position, const children_iterator &__It)
- size_type depth (const walker &__position)
- void clear ()

Protected Member Functions

- _Node * _C_create_node (const _Tp &__x)
- _Node * _C_create_node ()

Friends

- bool operator==__VGTL_NULL_TMPL_ARGS (const __ITree &__x, const __ITree &__y)

9.3.1 Detailed Description

This is the base class for all trees with data hooks

9.3.2 Member Typedef Documentation

9.3.2.1 typedef _Tree_walker<_Tp,const _Tp&,const _Tp*,container_type,children_iterator,_Node> __ITree::const_iterative_walker

the const iterative walker

Definition at line 2065 of file vgtl_tree.h.

9.3.2.2 typedef _Tree_iterator<_Tp,const _Tp&,const _Tp*,container_type,children_iterator,node_type> __ITree::const_iterator

the const iterator

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _ITree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 2060 of file vgtl_tree.h.

9.3.2.3 typedef const value_type* __Tree_t::const_pointer [inherited]

standard typedef

Definition at line 1578 of file vgtl_tree.h.

9.3.2.4 typedef const value_type& __Tree_t::const_reference [inherited]

standard typedef

Definition at line 1580 of file vgtl_tree.h.

9.3.2.5 typedef std::reverse_iterator<const_iterator> __ITree::const_reverse_iterator

the const reverse iterator

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _ITree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 2069 of file vgtl_tree.h.

9.3.2.6 typedef _RTree_walker<_Tp,const _Tp&,const _Tp*,container_type,children_iterator,node_type> __Tree_t::const_walker [inherited]

the (recursive) const walker

Definition at line 1614 of file vgtl_tree.h.

9.3.2.7 typedef ptrdiff_t __Tree_t::difference_type [inherited]

standard typedef

Definition at line 1582 of file vgtl_tree.h.

9.3.2.8 typedef _Tree_walker<_Tp,_Tp&,_Tp*,container_type,children_iterator,_Node> __ITree::iterative_walker

the iterative walker

Definition at line 2063 of file vgtl_tree.h.

9.3.2.9 typedef _Tree_iterator<_Tp,_Tp&,_Tp*,container_type,children_iterator,node_type> __ITree::iterator

the iterator

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _ITree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 2058 of file vgtl_tree.h.

9.3.2.10 typedef _Node __ITree::node_type

standard typedef

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _ITree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 2055 of file vgtl_tree.h.

9.3.2.11 typedef value_type* __Tree_t::pointer [inherited]

standard typedef

Definition at line 1577 of file vgtl_tree.h.

9.3.2.12 typedef value_type& __Tree_t::reference [inherited]

standard typedef

Definition at line 1579 of file vgtl_tree.h.

9.3.2.13 typedef std::reverse_iterator<iterator> __ITree::reverse_iterator

the reverse iterator

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _ITree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 2071 of file vgtl_tree.h.

9.3.2.14 typedef size_t __Tree_t::size_type [inherited]

standard typedef

Definition at line 1581 of file vgtl_tree.h.

9.3.2.15 typedef _Tp __Tree_t::value_type [inherited]

standard typedef

Definition at line 1575 of file vgtl_tree.h.

9.3.2.16 typedef _RTree_walker<_Tp,_Tp&,_Tp*,container_type,children_iterator,node_type> __Tree_t::walker [inherited]

the (recursive) walker

Definition at line 1612 of file vgtl_tree.h.

9.3.3 Constructor & Destructor Documentation

9.3.3.1 __ITree::__ITree (const allocator_type & __a = allocator_type()) [inline, explicit]

standard constructor

Definition at line 2092 of file vgtl_tree.h.

9.3.3.2 __ITree::__ITree (size_type __n, const _Tp & __value, const allocator_type & __a = allocator_type()) [inline]

construct a tree containing ___n nodes with value ___value at the root spot.

Definition at line 2184 of file vgtl_tree.h.

9.3.3.3 __ITree::__ITree (size_type __n) [inline, explicit]

construct a tree containing ___n nodes with default value at the root spot.

Definition at line 2191 of file vgtl_tree.h.

9.3.3.4 __ITree::__ITree (const _Self & __x) [inline]

copy constructor

Definition at line 2196 of file vgtl_tree.h.

9.3.3.5 virtual __ITree::~__ITree ( ) [inline, virtual]

standard destructor

Definition at line 2199 of file vgtl_tree.h.

9.3.4 Member Function Documentation

9.3.4.1 _Node* __Tree_t::_C_create_node (const _Tp & __x) [inline, protected, inherited]

construct a new tree node containing data ___x

Definition at line 1629 of file vgtl_tree.h.

9.3.4.2 _Node* __Tree_t::_C_create_node ( ) [inline, protected, inherited]

construct a new tree node containing default data

Definition at line 1641 of file vgtl_tree.h.

9.3.4.3 iterative_walker __ITree::begin (walker_type wt = cw_pre_post, bool front_to_back = true, bool depth_first = true) [inline]

the walker to the first node of the complete walk

Definition at line 2122 of file vgtl_tree.h.

9.3.4.4 const_iterative_walker __ITree::begin (walker_type wt = cw_pre_post, bool front_to_back = true, bool depth_first = true) const [inline]

the const walker to the first node of the complete walk

Definition at line 2129 of file vgtl_tree.h.

9.3.4.5 void __Tree_t::clear ( ) [inline, inherited]

empty the tree

Definition at line 1817 of file vgtl_tree.h.

9.3.4.6 size_type __Tree_t::depth (const walker & *__position*) [inline, inherited]

return the depth of node __position in the tree

Definition at line 1805 of file vgtl_tree.h.

9.3.4.7 size_type __ITree::depth (const iterative_walker & *__position*) [inline]

return the depth of this __position in the tree

Definition at line 2177 of file vgtl_tree.h.

9.3.4.8 bool __Tree_t::empty ( ) const [inline, inherited]

is the tree empty?

Definition at line 1657 of file vgtl_tree.h.

9.3.4.9 iterative_walker __ITree::end (walker_type *wt* = cw_pre_post, bool *front_to_back* = true, bool *depth_first* = true) [inline]

the walker beyond the last node of the walk

Definition at line 2137 of file vgtl_tree.h.

9.3.4.10 const_iterative_walker __ITree::end (walker_type *wt* = cw_pre_post, bool *front_to_back* = true, bool *depth_first* = true) const [inline]

the const walker beyond the last node of the walk

Definition at line 2143 of file vgtl_tree.h.

9.3.4.11 void __Tree_t::erase (const __walker_base & *__position*) [inline, inherited]

erase the node at position __position.

Definition at line 1713 of file vgtl_tree.h.

9.3.4.12 bool __Tree_t::erase_child (const __walker_base & *__position*, const children_iterator & *__It*) [inline, inherited]

erase the (leaf) child __It of node __position. This works if and only if the child is a leaf.

Definition at line 1770 of file vgtl_tree.h.

9.3.4.13 _Node* __Tree_t::erase_subtree (const __walker_base & *__position*, const children_iterator & *__It*) [inline, inherited]

erase the subtree position __position, whose top node is the child at children_iterator position __It, and return its top node.

Definition at line 1790 of file vgtl_tree.h.

9.3.4.14 _Node* __Tree_t::erase_tree (const __walker_base & __position) [inline, inherited]

erase the subtree starting at position __position, and return its top node.

Definition at line 1743 of file vgtl_tree.h.

9.3.4.15 allocator_type __Tree_t::get_allocator ( ) const [inline, inherited]

construct an allocator object

Definition at line 1587 of file vgtl_tree.h.

9.3.4.16 reference __ITree::getroot ( ) [inline]

get a reference to the virtual root node

Definition at line 2172 of file vgtl_tree.h.

9.3.4.17 const_reference __ITree::getroot ( ) const [inline]

get a const reference to the virtual root node

Definition at line 2174 of file vgtl_tree.h.

9.3.4.18 void __Tree_t::insert_child (const __walker_base & __position, const _Tp & __x, const container_insert_arg & __It) [inline, inherited]

add a child below __position with data __x, at the __It position in the __position - node's children container

Definition at line 1668 of file vgtl_tree.h.

9.3.4.19 void __Tree_t::insert_child (const __walker_base & __position, const container_insert_arg & __It) [inline, inherited]

add a child below __position with default data, at the __It position in the __position - node's children container

Definition at line 1676 of file vgtl_tree.h.

9.3.4.20 void __Tree_t::insert_children (const __walker_base & __position, size_type __n, const _Tp & __x, const children_iterator & __It) [inline, inherited]

add __n children below __position with data __x, after the __It position in the __position - node's children container

Definition at line 1682 of file vgtl_tree.h.

9.3.4.21 void __Tree_t::insert_subtree (const __walker_base & __position, _Self & __subtree, const children_iterator & __It) [inline, inherited]

add a complete subtree __subtree below position __position and children iterator position __It.

Definition at line 1702 of file vgtl_tree.h.

9.3.4.22 size_type __Tree_t::max_size ( ) const [inline, inherited]

return the maximum possible size of the tree (theor. infinity)

Definition at line 1660 of file vgtl_tree.h.

9.3.4.23 _Self& __ITree::operator= (const _Self & __x)

standard assignment operator

9.3.4.24 _Self& __ITree::operator= (_Node * __x) [inline]

assign a tree from one node -> make this node the root node. This is useful for making trees out of erased subtrees.

Definition at line 2208 of file vgtl_tree.h.

9.3.4.25 reverse_iterator __ITree::rbegin ( ) [inline]

return a reverse iterator to the first node in walk

Definition at line 2151 of file vgtl_tree.h.

9.3.4.26 const_reverse_iterator __ITree::rbegin ( ) const [inline]

return a const reverse iterator to the first node in walk

Definition at line 2158 of file vgtl_tree.h.

9.3.4.27 reverse_iterator __ITree::rend ( ) [inline]

return a reverse iterator beyond the last node in walk

Definition at line 2154 of file vgtl_tree.h.

9.3.4.28 const_reverse_iterator __ITree::rend ( ) const [inline]

return a const reverse iterator beyond the last node in walk

Definition at line 2161 of file vgtl_tree.h.

9.3.4.29 iterative_walker __ITree::root (walker_type *wt* = cw_pre_post, bool *front_to_back* = true, bool *depth_first* = true) [inline]

return an iterative walker of type wt to the ground node

Definition at line 2099 of file vgtl_tree.h.

9.3.4.30 const_iterative_walker __ITree::root (walker_type *wt* = cw_pre_post, bool *front_to_back* = true, bool *depth_first* = true) const [inline]

return a const iterative walker of type wt to the ground node

Definition at line 2106 of file vgtl_tree.h.

9.3.4.31 size_type __ITree::size ( ) const [inline]

return the size of the tree (# of nodes)

Definition at line 2165 of file vgtl_tree.h.

9.3.4.32 void __Tree_t::swap (_Self & __x) [inline, inherited]

swap two trees

Definition at line 1663 of file vgtl_tree.h.

9.3.4.33 iterative_walker __ITree::through ( ) [inline]

the walker beyond the complete walk

Definition at line 2113 of file vgtl_tree.h.

9.3.4.34 const_iterative_walker __ITree::through ( ) const [inline]

the const walker beyond the complete walk

Definition at line 2117 of file vgtl_tree.h.

9.3.5 Friends And Related Function Documentation

9.3.5.1 bool operator==__VGTL_NULL_TMPL_ARGS (const __ITree & *__x*, const __ITree & *__y*) [friend]

comparison operator

The documentation for this class was generated from the following file:

- vgtl_tree.h

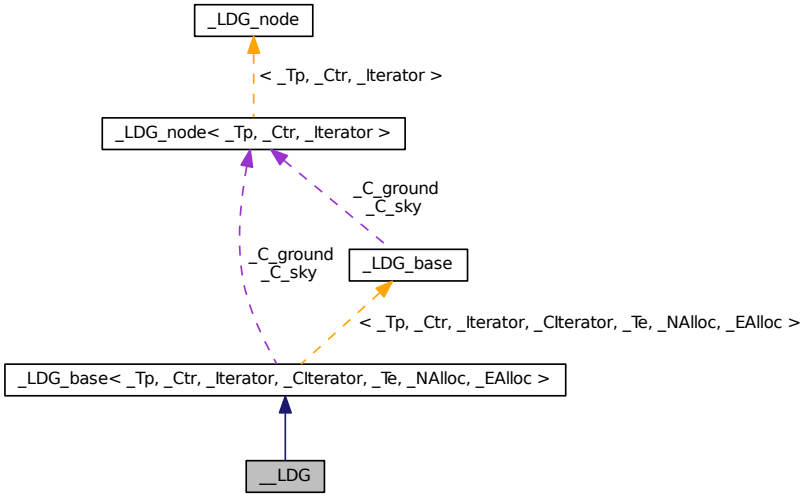
9.4 __LDG Class Reference

Labelled directed graph base class.

`#include <vgtl_ldag.h>`

Inheritance diagram for __LDG:

Collaboration diagram for __LDG:



Public Types

- typedef _Ctr container_type
- typedef _Iterator out_iterator
- typedef _Iterator in_iterator
- typedef _CIterator out_const_iterator
- typedef _CIterator in_const_iterator
- typedef _Base::node_allocator_type node_allocator_type
- typedef _Base::edge_allocator_type edge_allocator_type
- typedef _LDG_iterator< _Tp, _Tp &, _Tp *, container_type, children_iterator, children_const_iterator, _Te > iterator
- typedef _LDG_iterator< _Tp, const _Tp &, const _Tp *, container_type, children_iterator, children_const_iterator, _Te > const_iterator
- typedef std::reverse_iterator < const_iterator > const_reverse_iterator
- typedef std::reverse_iterator < iterator > reverse_iterator
- typedef _LDG_walker< _Tp, _Tp &, _Tp *, container_type, children_iterator, children_const_iterator, _Te > walker
- typedef _LDG_walker< _Tp, const _Tp &, const _Tp *, container_type, children_iterator, children_const_iterator, _Te > const_walker

- typedef _Tp value_type
- typedef _Node node_type
- typedef _Edge edge_type
- typedef value_type * pointer
- typedef const value_type * const_pointer
- typedef value_type & reference

- typedef const value_type & const_reference
- typedef size_t size_type
- typedef ptrdiff_t difference_type

Public Member Functions

- node_allocator_type get_node_allocator () const
- edge_allocator_type get_edge_allocator () const
- __LDG (const allocator_type &__a=allocator_type())
- walker ground ()
- walker sky ()
- const_walker ground () const
- const_walker sky () const
- bool empty () const
- size_type size () const
- size_type max_size () const
- void swap (_Self &__x)
- walker insert_node_in_graph (_Node *__n, const walker &__parent, const walker &__child, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- walker insert_in_graph (const _Tp &__x, const walker &__parent, const walker &__child, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- walker insert_in_graph (const walker &__parent, const walker &__child, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- void insert_subgraph (_Self &__subgraph, const walker &__parent, const walker &__child, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr1, template< class __Tp, class __AllocTp > class __SequenceCtr2, class _Allocator1 , class _Allocator2 >
  walker insert_node_in_graph (_Node *__node, const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr1, template< class __Tp, class __AllocTp > class __SequenceCtr2, class _Allocator1 , class _Allocator2 >
  walker insert_in_graph (const _Tp &__x, const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr1, template< class __Tp, class __AllocTp > class __SequenceCtr2, class _Allocator1 , class _Allocator2 >
  walker insert_in_graph (const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker insert_node_in_graph (_Node *__node, const walker &__parent, const container_insert_arg &__pref, const __SequenceCtr< walker, _Allocator > &__children)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker insert_in_graph (const _Tp &__x, const walker &__parent, const container_insert_arg &__pref, const __SequenceCtr< walker, _Allocator > &__children)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker insert_in_graph (const walker &__parent, const container_insert_arg &__pref, const __SequenceCtr< walker, _Allocator > &__children)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker insert_node_in_graph (_Node *__node, const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const container_insert_arg &__cref)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker insert_in_graph (const _Tp &__x, const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const container_insert_arg &__cref)

- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker insert_in_graph (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const container_insert_arg &__cref)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr1, template< class __Tp, class __AllocTp > class __SequenceCtr2, class _Allocator1 , class _Allocator2 >
  void insert_subgraph (_Self &__subgraph, const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children)
- void add_edge (const edge &__edge, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- void add_edge (const walker &__parent, const walker &__child, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- void replace_edge_to_child (const walker &__parent, const walker &__child_old, const walker &__child_new)
- void replace_edge_to_parent (const walker &__parent_old, const walker &__parent_new, const walker &__child)
- void remove_edge (const edge &__edge)
- void remove_edge_and_deattach (const walker &__parent, const walker &__child)
- void remove_edge (const walker &__parent, const walker &__child)
- template<class Compare >
  void sort_child_edges (walker __position, children_iterator first, children_iterator last, Compare comp)
- template<class Compare >
  void sort_parent_edges (walker __position, parents_iterator first, parents_iterator last, Compare comp)
- template<class Compare >
  void sort_child_edges (walker __position, Compare comp)
- template<class Compare >
  void sort_parent_edges (walker __position, Compare comp)
- walker insert_node (_Node *_node, const walker &__position, const container_insert_arg &__It)
- walker insert_node (const _Tp &__x, const walker &__position, const container_insert_arg &__It)
- walker insert_node (const walker &__position, const container_insert_arg &__It)
- walker insert_node_before (_Node *_node, const walker &__position, const container_insert_arg &__It)
- void insert_node_before (const _Tp &__x, const walker &__position, const container_insert_arg &__It)
- void insert_node_before (const walker &__position, const container_insert_arg &__It)
- void merge (const walker &__position, const walker &__second, bool merge_parent_edges=true, bool merge_child_edges=true)
- void erase (const walker &__position)
- void partial_erase_to_parent (const walker &__position, const walker &__parent, unsigned int idx)
- void clear_erased_part (erased_part &_ep)
- erased_part erase_maximal_subgraph (const walker &__position)
- erased_part erase_minimal_subgraph (const walker &__position)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  erased_part erase_maximal_subgraph (const __SequenceCtr< walker, _Allocator > &__positions)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  erased_part erase_minimal_subgraph (const __SequenceCtr< walker, _Allocator > &__positions)
- erased_part erase_maximal_pregraph (const walker &__position)
- erased_part erase_minimal_pregraph (const walker &__position)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  erased_part erase_maximal_pregraph (const __SequenceCtr< walker, _Allocator > &__positions)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  erased_part erase_minimal_pregraph (const __SequenceCtr< walker, _Allocator > &__positions)

- bool erase_child (const walker &__position, const children_iterator &__It)
- bool erase_parent (const walker &__position, const parents_iterator &__It)
- void clear ()
- __LDG (const _Self &__x)
- ~__LDG ()
- _Self & operator= (const _Self &__x)
- _Self & operator= (const _RV_LDG &__rl)
- _Self & operator= (const erased_part &__ep)

- out_iterator source_begin ()
- out_iterator root_begin ()

- out_iterator source_end ()
- out_iterator root_end ()

- out_const_iterator source_begin () const
- out_iterator root_begin ()

- out_const_iterator source_end () const
- out_iterator root_end ()

- in_iterator sink_begin ()
- in_iterator leaf_begin ()

- in_iterator sink_end ()
- in_iterator leaf_end ()

- in_const_iterator sink_begin () const
- in_iterator leaf_begin ()

- in_const_iterator sink_end () const
- in_iterator leaf_end ()

Protected Types

- typedef std::pair< _RV_LDG, std::vector< enhanced_edge > > erased_part

Protected Member Functions

- _Node * _C_create_node (const _Tp &__x)
- _Node * _C_create_node ()
- _Edge * _C_create_edge (const _Te &__x)
- _Edge * _C_create_edge ()
- _Edge * _C_create_edge (const _Te &__x, _Node *__s, _Node *__t)
- _Edge * _C_create_edge (_Node *__s, _Node *__t)
- void clear_graph (_LDG_node< _Tp, _Ctr, _Iterator > *_node)
- _LDG_node< _Tp, _Ctr, _Iterator > * _C_get_node ()
- void _C_put_node (_LDG_node< _Tp, _Ctr, _Iterator > *__p)
- _LDG_edge< _Te, _Node > * _C_get_edge ()
- void _C_put_edge (_LDG_edge< _Te, _Node > *__p)
- void clear_out_edges ()
- void clear_in_edges ()
- void add_all_out_edges (_Output_Iterator fi, _LDG_node< _Tp, _Ctr, _Iterator > *_parent)
- void add_all_in_edges (_Output_Iterator fi, _LDG_node< _Tp, _Ctr, _Iterator > *_child)

Protected Attributes

- _LDG_node< _Tp, _Ctr, _Iterator > * _C_ground
- _LDG_node< _Tp, _Ctr, _Iterator > * _C_sky
- int _C_mark

9.4.1 Detailed Description

This is the toplevel base class for all labelled directed graphs independent of allocators

9.4.2 Member Typedef Documentation

9.4.2.1 typedef _LDG_iterator<_Tp,const _Tp&,const _Tp*,container_type, children_iterator,children_const_iterator,_Te> __LDG::const_iterator

the const iterator

Definition at line 651 of file vgtl_ldag.h.

9.4.2.2 typedef const value_type* __LDG::const_pointer

standard typedef

Definition at line 630 of file vgtl_ldag.h.

9.4.2.3 typedef const value_type& __LDG::const_reference

standard typedef

Definition at line 632 of file vgtl_ldag.h.

9.4.2.4 typedef std::reverse_iterator<const_iterator> __LDG::const_reverse_iterator

the const reverse iterator

Definition at line 655 of file vgtl_ldag.h.

9.4.2.5 typedef _LDG_walker<_Tp,const _Tp&,const _Tp*,container_type,children_iterator, children_const_iterator,_Te> __LDG::const_walker

the (recursive) const walker

Reimplemented in ldgraph.

Definition at line 674 of file vgtl_ldag.h.

9.4.2.6 typedef _Ctr __LDG::container_type

internal container used to store the edges

Reimplemented from _LDG_base< _Tp, _Ctr, _Iterator, _CIterator, _Te, _NAlloc, _EAlloc >.

Definition at line 601 of file vgtl_ldag.h.

9.4.2.7 typedef ptrdiff_t __LDG::difference_type

standard typedef

Definition at line 634 of file vgtl_ldag.h.

9.4.2.8 typedef _Base::edge_allocator_type __LDG::edge_allocator_type

edge allocator type

Reimplemented from _LDG_base< _Tp, _Ctr, _Iterator, _CIterator, _Te, _NAlloc, _EAlloc >.

Definition at line 641 of file vgtl_ldag.h.

9.4.2.9 typedef _Edge __LDG::edge_type

standard typedef

Definition at line 628 of file vgtl_ldag.h.

9.4.2.10 typedef std::pair<_RV_LDG, std::vector<enhanced_edge> > __LDG::erased_part [protected]

an edge of the graph (parent, child) an edge with additiona information about erased ground/sky edges an erased subgraph which is not yet a new directed graph

Reimplemented in ldgraph.

Definition at line 683 of file vgtl_ldag.h.

9.4.2.11 typedef _CIterator __LDG::in_const_iterator

const iterator for accessing the out edges

Reimplemented from _LDG_base< _Tp, _Ctr, _Iterator, _CIterator, _Te, _NAlloc, _EAlloc >.

Definition at line 605 of file vgtl_ldag.h.

9.4.2.12 typedef _Iterator __LDG::in_iterator

iterator for accessing the in edges

Reimplemented from _LDG_base< _Tp, _Ctr, _Iterator, _CIterator, _Te, _NAlloc, _EAlloc >.

Definition at line 603 of file vgtl_ldag.h.

9.4.2.13 typedef _LDG_iterator<_Tp,_Tp&,_Tp*,container_type,children_iterator, children_const_iterator,_Te> __LDG::iterator

the iterator

Definition at line 648 of file vgtl_ldag.h.

9.4.2.14 typedef _Base::node_allocator_type __LDG::node_allocator_type

node allocator type

Reimplemented from _LDG_base< _Tp, _Ctr, _Iterator, _CIterator, _Te, _NAlloc, _EAlloc >.

Definition at line 637 of file vgtl_ldag.h.

9.4.2.15 typedef _Node __LDG::node_type

standard typedef

Definition at line 627 of file vgtl_ldag.h.

9.4.2.16 typedef _CIterator __LDG::out_const_iterator

const iterator for accessing the out edges

Reimplemented from _LDG_base< _Tp, _Ctr, _Iterator, _CIterator, _Te, _NAlloc, _EAlloc >.

Definition at line 604 of file vgtl_ldag.h.

9.4.2.17 typedef _Iterator __LDG::out_iterator

iterator for accessing the out edges

Reimplemented from _LDG_base< _Tp, _Ctr, _Iterator, _CIterator, _Te, _NAlloc, _EAlloc >.

Definition at line 602 of file vgtl_ldag.h.

9.4.2.18 typedef value_type* __LDG::pointer

standard typedef

Definition at line 629 of file vgtl_ldag.h.

9.4.2.19 typedef value_type& __LDG::reference

standard typedef

Definition at line 631 of file vgtl_ldag.h.

9.4.2.20 typedef std::reverse_iterator<iterator> __LDG::reverse_iterator

the reverse iterator

Definition at line 657 of file vgtl_ldag.h.

9.4.2.21 typedef size_t __LDG::size_type

standard typedef

Definition at line 633 of file vgtl_ldag.h.

9.4.2.22 typedef _Tp __LDG::value_type

standard typedef

Definition at line 626 of file vgtl_ldag.h.

9.4.2.23 typedef _LDG_walker<_Tp,_Tp&,_Tp*,container_type,children_iterator, children_const_iterator,_Te> __LDG::walker

the (recursive) walker

Reimplemented in ldgraph.

Definition at line 671 of file vgtl_ldag.h.

9.4.3 Constructor & Destructor Documentation

9.4.3.1 __LDG::__LDG (const allocator_type & __a = allocator_type()) [inline, explicit]

standard constructor

Definition at line 781 of file vgtl_ldag.h.

9.4.3.2 __LDG::__LDG (const _Self & __x) [inline]

copy constructor

Definition at line 2108 of file vgtl_ldag.h.

9.4.3.3 __LDG::~__LDG ( ) [inline]

standard destructor

Definition at line 2125 of file vgtl_ldag.h.

9.4.4 Member Function Documentation

9.4.4.1 _Edge* __LDG::_C_create_edge (const _Te & __x) [inline, protected]

construct a new graph edge containing data ___x

Definition at line 726 of file vgtl_ldag.h.

9.4.4.2 _Edge* __LDG::_C_create_edge ( ) [inline, protected]

construct a new graph edge containing default data

Definition at line 738 of file vgtl_ldag.h.

9.4.4.3 _Edge* __LDG::_C_create_edge (const _Te & __x, _Node * __s, _Node * __t) [inline, protected]

construct a new graph edge containing data ___x with source ___s and target ___t.

Definition at line 751 of file vgtl_ldag.h.

9.4.4.4 _Edge* __LDG::_C_create_edge (_Node * __s, _Node * __t) [inline, protected]

construct a new graph edge containing default data with source ___s and target ___t.

Definition at line 766 of file vgtl_ldag.h.

9.4.4.5 _Node* __LDG::_C_create_node (const _Tp & __x) [inline, protected]

construct a new graph node containing data ___x

Definition at line 698 of file vgtl_ldag.h.

9.4.4.6 _Node* __LDG::_C_create_node ( ) [inline, protected]

construct a new graph node containing default data

Definition at line 712 of file vgtl_ldag.h.

9.4.4.7 _LDG_edge<_Te,_Node>* _LDG_base::_C_get_edge ( ) [inline, protected, inherited]

allocate a new edge

Definition at line 533 of file vgtl_ldagbase.h.

9.4.4.8 _LDG_node<_Tp,_Ctr,_Iterator>* _LDG_base::_C_get_node ( ) [inline, protected, inherited]

allocate a new node

Definition at line 526 of file vgtl_ldagbase.h.

9.4.4.9 void _LDG_base::_C_put_edge (_LDG_edge< _Te, _Node > * __p) [inline, protected, inherited]

deallocate a edge

Definition at line 536 of file vgtl_ldagbase.h.

9.4.4.10 void _LDG_base::_C_put_node (_LDG_node< _Tp, _Ctr, _Iterator > * __p) [inline, protected, inherited]

deallocate a node

Definition at line 529 of file vgtl_ldagbase.h.

9.4.4.11 void _LDG_base::add_all_in_edges (_Output_Iterator *fi,* _LDG_node< _Tp, _Ctr, _Iterator > * *_child*) [protected, inherited]

add all in edges to the child _child. fi is a iterator to the in edges container of the child

9.4.4.12 void _LDG_base< _Tp, _Ctr, _Iterator, _CIterator, _Te, _NAlloc, _EAlloc >::add_all_out_edges (_Output_Iterator *fi,* _LDG_node< _Tp, _Ctr, _Iterator > * *_parent*) [inline, protected, inherited]

add all out edges to the parent _parent. fi is a iterator to the out edges container of the parent

Definition at line 603 of file vgtl_ldagbase.h.

9.4.4.13 void __LDG::add_edge (const edge & *__edge,* const container_insert_arg & *__itc,* const container_insert_arg & *__itp*) [inline]

add one edge between two nodes at the positions described by ___Itc and ___Itp.

Definition at line 1191 of file vgtl_ldag.h.

9.4.4.14 void __LDG::add_edge (const walker & *__parent,* const walker & *__child,* const container_insert_arg & *__itc,* const container_insert_arg & *__itp*) [inline]

add an edge between __parent and __child at positions __Itc and __Itp, respectively

Definition at line 1200 of file vgtl_ldag.h.

9.4.4.15 void __LDG::clear ( ) [inline]

erase all the nodes except sky and ground

Reimplemented from _LDG_base< _Tp, _Ctr, _Iterator, _CIterator, _Te, _NAlloc, _EAlloc >.

Reimplemented in ldgraph.

Definition at line 2068 of file vgtl_ldag.h.

9.4.4.16 void __LDG::clear_erased_part (erased_part & *_ep*) [inline]

clear all nodes in an erased part

Definition at line 1868 of file vgtl_ldag.h.

9.4.4.17 void _LDG_base< _Tp, _Ctr, _Iterator, _CIterator, _Te, _NAlloc, _EAlloc >::clear_graph (_LDG_node< _Tp, _Ctr, _Iterator > * *_node*) [protected, inherited]

removes recursively all nodes and edges downward starting from _node.

Definition at line 572 of file vgtl_ldagbase.h.

9.4.4.18 void _LDG_base::clear_in_edges ( ) [inline, protected, inherited]

clear all in edges of the sky node

Definition at line 551 of file vgtl_ldagbase.h.

9.4.4.19 void _LDG_base::clear_out_edges ( ) [inline, protected, inherited]

clear all out edges of the ground node

Definition at line 548 of file vgtl_ldagbase.h.

9.4.4.20 bool __LDG::empty ( ) const [inline]

returns true if the DG is empty

Definition at line 888 of file vgtl_ldag.h.

9.4.4.21 void __LDG::erase (const walker & *__position*) [inline]

erase a node from the DG except the sky and ground

Definition at line 1518 of file vgtl_ldag.h.

9.4.4.22 bool __LDG::erase_child (const walker & *__position,* const children_iterator & *__it*) [inline]

Erase a child of __position. This works if and only if the child has only one child and no other parents.

Definition at line 2020 of file vgtl_ldag.h.

9.4.4.23 erased_part __LDG::erase_maximal_pregraph (const walker & __position) [inline]

here every child is removed till the sky node. included the node at __position. The removed subgraph is returned. The subgraph is maximal, i.e. all nodes are removed, which are reachable from __position by walking upwards.

Definition at line 1950 of file vgtl_ldag.h.

9.4.4.24 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > erased_part __LDG::erase_maximal_pregraph (const __SequenceCtr< walker, _Allocator > & __positions) [inline]

here every child is removed till the sky included all nodes from __positions. The removed subgraph is returned. The subgraph is maximal, i.e. all nodes are removed, which are reachable from any node from __positions by walking up.

Definition at line 1984 of file vgtl_ldag.h.

9.4.4.25 erased_part __LDG::erase_maximal_subgraph (const walker & __position) [inline]

here every child is removed till the last base node, included the node at __position. The removed subgraph is returned. The subgraph is maximal, i.e. all nodes are removed, which are reachable from __position by walking down.

Definition at line 1879 of file vgtl_ldag.h.

9.4.4.26 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > erased_part __LDG::erase_maximal_subgraph (const __SequenceCtr< walker, _Allocator > & __positions) [inline]

here every child is removed till the last base node, included all nodes from __positions. The removed subgraph is returned. The subgraph is maximal, i.e. all nodes are removed, which are reachable from any node from __positions by walking down.

Definition at line 1913 of file vgtl_ldag.h.

9.4.4.27 erased_part __LDG::erase_minimal_pregraph (const walker & __position) [inline]

here every child is removed till the sky. included the node at __position. The removed subgraph is returned. The subgraph is minimal, i.e. only nodes are removed, which have no other descendant than __position. I.e., when walking towards the sky, there is no way which bypasses __position.

Definition at line 1966 of file vgtl_ldag.h.

9.4.4.28 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > erased_part __LDG::erase_minimal_pregraph (const __SequenceCtr< walker, _Allocator > & __positions) [inline]

here every child is removed till the sky. included all nodes from __positions. The removed subgraph is returned. The subgraph is minimal, i.e. only nodes are removed, which have no other ancestor than any node in __positions. I.e., when walking towards the ground, there is no way which bypasses all nodes in __positions.

Definition at line 2004 of file vgtl_ldag.h.

9.4.4.29 erased_part __LDG::erase_minimal_subgraph (const walker & __position) [inline]

here every child is removed till the last base node, included the node at __position. The removed subgraph is returned. The subgraph is minimal, i.e. only nodes are removed, which have no other ancestor than

__position. I.e., when walking towards the ground, there is no way which bypasses __position.

Definition at line 1895 of file vgtl_ldag.h.

9.4.4.30 template<template< class _Tp, class _AllocTp > class _SequenceCtr, class _Allocator > erased_part __LDG::erase_minimal_subgraph (const _SequenceCtr< walker, _Allocator > & *__positions*) [inline]

here every child is removed till the last base node, included all nodes from __positions. The removed subgraph is returned. The subgraph is minimal, i.e. only nodes are removed, which have no other ancestor than any node in __positions. I.e., when walking towards the ground, there is no way which bypasses all nodes in __positions.

Definition at line 1933 of file vgtl_ldag.h.

9.4.4.31 bool __LDG::erase_parent (const walker & *__position,* const parents_iterator & *__it*) [inline]

Erase a parent of __position. This works if and only if the parent has only one parent and no other children.

Definition at line 2046 of file vgtl_ldag.h.

9.4.4.32 edge_allocator_type __LDG::get_edge_allocator ( ) const [inline]

construct an edge allocator object

Reimplemented from _LDG_base< _Tp, _Ctr, _Iterator, _CIterator, _Te, _NAlloc, _EAlloc >.

Definition at line 643 of file vgtl_ldag.h.

9.4.4.33 node_allocator_type __LDG::get_node_allocator ( ) const [inline]

construct a node allocator object

Reimplemented from _LDG_base< _Tp, _Ctr, _Iterator, _CIterator, _Te, _NAlloc, _EAlloc >.

Definition at line 639 of file vgtl_ldag.h.

9.4.4.34 walker __LDG::ground ( ) [inline]

return a walker to the virtual ground node.

Definition at line 784 of file vgtl_ldag.h.

9.4.4.35 const_walker __LDG::ground ( ) const [inline]

return a const walker to the virtual ground node.

Definition at line 794 of file vgtl_ldag.h.

9.4.4.36 walker __LDG::insert_in_graph (const _Tp & *__x,* const walker & *__parent,* const walker & *__child,* const container_insert_arg & *__itc,* const container_insert_arg & *__itp*) [inline]

insert node with data __n into the graph between __parent and __child, the edge at the specific positions described by __Itc and __Itp.

Definition at line 928 of file vgtl_ldag.h.

9.4.4.37 walker __LDG::insert_in_graph (const walker & *__parent,* const walker & *__child,* const container_insert_arg & *__ltc,* const container_insert_arg & *__ltp*) [inline]

insert node with default data into the graph between __parent and __child, the edge at the specific positions described by __Itc and __Itp.

Definition at line 942 of file vgtl_ldag.h.

9.4.4.38 template<template< class __Tp, class __AllocTp > class __SequenceCtr1, template< class __Tp, class __AllocTp > class __SequenceCtr2, class _Allocator1 , class _Allocator2 > walker __LDG::insert_in_graph (const _Tp & *__x,* const __SequenceCtr1< walker, _Allocator1 > & *__parents,* const __SequenceCtr2< walker, _Allocator2 > & *__children*) [inline]

insert a node with data __x into the graph between all parents from __parents and all children from __children.

Definition at line 1006 of file vgtl_ldag.h.

9.4.4.39 template<template< class __Tp, class __AllocTp > class __SequenceCtr1, template< class __Tp, class __AllocTp > class __SequenceCtr2, class _Allocator1 , class _Allocator2 > walker __LDG::insert_in_graph (const __SequenceCtr1< walker, _Allocator1 > & *__parents,* const __SequenceCtr2< walker, _Allocator2 > & *__children*) [inline]

insert a node with default data into the graph between all parents from __parents and all children from __children.

Definition at line 1021 of file vgtl_ldag.h.

9.4.4.40 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker __LDG::insert_in_graph (const _Tp & *__x,* const walker & *__parent,* const container_insert_arg & *__pref,* const __SequenceCtr< walker, _Allocator > & *__children*) [inline]

insert a node with data __x into the graph between the parent __parent and all children from __children.

Definition at line 1059 of file vgtl_ldag.h.

9.4.4.41 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker __LDG::insert_in_graph (const walker & *__parent,* const container_insert_arg & *__pref,* const __SequenceCtr< walker, _Allocator > & *__children*) [inline]

insert a node with data __x into the graph between the parent __parent and all children from __children.

Definition at line 1073 of file vgtl_ldag.h.

9.4.4.42 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker __LDG::insert_in_graph (const _Tp & *__x,* const __SequenceCtr< walker, _Allocator > & *__parents,* const walker & *__child,* const container_insert_arg & *__cref*) [inline]

insert a node with data __x into the graph between all parents from __parents and the child __child.

Definition at line 1112 of file vgtl_ldag.h.

9.4.4.43 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker __LDG::insert_in_graph (const __SequenceCtr< walker, _Allocator > & *__parents,* const walker & *__child,* const container_insert_arg & *__cref*) [inline]

insert a node with default data into the graph between all parents from __parents and the child __child.

Definition at line 1127 of file vgtl_ldag.h.

9.4.4.44 walker __LDG::insert_node (_Node * *_node,* const walker & *__position,* const container_insert_arg & *__it*) [inline]

insert one node as child of __position

Definition at line 1379 of file vgtl_ldag.h.

9.4.4.45 walker __LDG::insert_node (const _Tp & *__x,* const walker & *__position,* const container_insert_arg & *__it*) [inline]

insert a new node with data __x as child of __position

Definition at line 1393 of file vgtl_ldag.h.

9.4.4.46 walker __LDG::insert_node (const walker & *__position,* const container_insert_arg & *__it*) [inline]

insert a new node with default data as child of __position

Definition at line 1399 of file vgtl_ldag.h.

9.4.4.47 walker __LDG::insert_node_before (_Node * *_node,* const walker & *__position,* const container_insert_arg & *__it*) [inline]

insert a node as parent of __position

Definition at line 1404 of file vgtl_ldag.h.

9.4.4.48 void __LDG::insert_node_before (const _Tp & *__x,* const walker & *__position,* const container_insert_arg & *__it*) [inline]

insert a new node with data __x as parent of __position

Definition at line 1418 of file vgtl_ldag.h.

9.4.4.49 void __LDG::insert_node_before (const walker & *__position,* const container_insert_arg & *__it*) [inline]

insert a new node with default data as parent of __position

Definition at line 1423 of file vgtl_ldag.h.

9.4.4.50 walker __LDG::insert_node_in_graph (_Node * *__n,* const walker & *__parent,* const walker & *__child,* const container_insert_arg & *__itc,* const container_insert_arg & *__itp*) [inline]

insert node __n into the graph between __parent and __child, the edge at the specific positions described by __Itc and __Itp.

Definition at line 912 of file vgtl_ldag.h.

9.4.4.51 template<template< class __Tp, class __AllocTp > class __SequenceCtr1, template< class __Tp, class __AllocTp > class __SequenceCtr2, class _Allocator1 , class _Allocator2 > walker __LDG::insert_node_in_graph (_Node * __node, const __SequenceCtr1< walker, _Allocator1 > & __parents, const __SequenceCtr2< walker, _Allocator2 > & __children) [inline]

insert node __n into the graph between all parents from __parents and all children from __children.

Definition at line 975 of file vgtl_ldag.h.

9.4.4.52 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker __LDG::insert_node_in_graph (_Node * __node, const walker & __parent, const container_insert_arg & __pref, const __SequenceCtr< walker, _Allocator > & __children) [inline]

insert node __n into the graph between the parent __parent and all children from __children.

Definition at line 1034 of file vgtl_ldag.h.

9.4.4.53 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker __LDG::insert_node_in_graph (_Node * __node, const __SequenceCtr< walker, _Allocator > & __parents, const walker & __child, const container_insert_arg & __cref) [inline]

insert node __n into the graph between all parents from __parents and the child __child.

Definition at line 1087 of file vgtl_ldag.h.

9.4.4.54 void __LDG::insert_subgraph (_Self & __subgraph, const walker & __parent, const walker & __child, const container_insert_arg & __ltc, const container_insert_arg & __ltp) [inline]

insert a subgraph into the graph between __parent and __child, the edge at the specific positions described by __Itc and __Itp.

Definition at line 953 of file vgtl_ldag.h.

9.4.4.55 template<template< class __Tp, class __AllocTp > class __SequenceCtr1, template< class __Tp, class __AllocTp > class __SequenceCtr2, class _Allocator1 , class _Allocator2 > void __LDG::insert_subgraph (_Self & __subgraph, const __SequenceCtr1< walker, _Allocator1 > & __parents, const __SequenceCtr2< walker, _Allocator2 > & __children) [inline]

in this method one DG is inserted into another DG between the parents __parents and the children __children.

Definition at line 1141 of file vgtl_ldag.h.

9.4.4.56 in_iterator __LDG::leaf_begin ( ) [inline]

return the first local sink of the directed graph

Definition at line 833 of file vgtl_ldag.h.

9.4.4.57 in_iterator __LDG::leaf_begin ( ) [inline]

return the first local sink of the directed graph

Definition at line 846 of file vgtl_ldag.h.

9.4.4.58 in_iterator __LDG::leaf_end ( ) [inline]

return beyond the last local sink of the directed graph

Definition at line 839 of file vgtl_ldag.h.

9.4.4.59 in_iterator __LDG::leaf_end ( ) [inline]

return beyond the last local sink of the directed graph

Definition at line 852 of file vgtl_ldag.h.

9.4.4.60 size_type __LDG::max_size ( ) const [inline]

the maximum size of a DG is virtually unlimited

Definition at line 899 of file vgtl_ldag.h.

9.4.4.61 void __LDG::merge (const walker & *__position*, const walker & *__second*, bool *merge_parent_edges* = `true`, bool *merge_child_edges* = `true`) [inline]

merge two nodes, call also the merge method for the node data

Definition at line 1429 of file vgtl_ldag.h.

9.4.4.62 _Self& __LDG::operator= (const _Self & *__x*)

standard assignment operator

9.4.4.63 _Self& __LDG::operator= (const _RV_LDG & *__rl*) [inline]

assignment operator from a part of an erased part

Reimplemented in ldgraph.

Definition at line 2131 of file vgtl_ldag.h.

9.4.4.64 _Self& __LDG::operator= (const erased_part & *__ep*) [inline]

assignment operator from an erased part

Reimplemented in ldgraph.

Definition at line 2139 of file vgtl_ldag.h.

9.4.4.65 void __LDG::partial_erase_to_parent (const walker & *__position*, const walker & *__parent*, unsigned int *idx*) [inline]

split a node in two, the first connected to the __parent, the second connected to all other parents. Then erase the first node.

Definition at line 1578 of file vgtl_ldag.h.

9.4.4.66 void __LDG::remove_edge (const edge & *__edge*) [inline]

remove an edge with a particular parent and child

Definition at line 1315 of file vgtl_ldag.h.

9.4.4.67 void __LDG::remove_edge (const walker & *__parent*, const walker & *__child*) [inline]

just remove one edge between `__parent` and `__child`

Definition at line 1332 of file vgtl_ldag.h.

9.4.4.68 void __LDG::remove_edge_and_deattach (const walker & *__parent,* const walker & *__child*) [inline]

remove one egde and don't reconnect the node to sky/ground

Definition at line 1319 of file vgtl_ldag.h.

9.4.4.69 void __LDG::replace_edge_to_child (const walker & *__parent,* const walker & *__child_old,* const walker & *__child_new*) [inline]

change the edge from __parent to __child_old to an edge from __parent to __child_new.

Definition at line 1243 of file vgtl_ldag.h.

9.4.4.70 void __LDG::replace_edge_to_parent (const walker & *__parent_old,* const walker & *__parent_new,* const walker & *__child*) [inline]

change the edge from __parent_old to __child to an edge from __parent_new to __child.

Definition at line 1281 of file vgtl_ldag.h.

9.4.4.71 out_iterator __LDG::root_begin ( ) [inline]

return the first local source of the directed graph

Definition at line 807 of file vgtl_ldag.h.

9.4.4.72 out_iterator __LDG::root_begin ( ) [inline]

return the first local source of the directed graph

Definition at line 820 of file vgtl_ldag.h.

9.4.4.73 out_iterator __LDG::root_end ( ) [inline]

return beyond the last local source of the directed graph

Definition at line 813 of file vgtl_ldag.h.

9.4.4.74 out_iterator __LDG::root_end ( ) [inline]

return beyond the last local source of the directed graph

Definition at line 826 of file vgtl_ldag.h.

9.4.4.75 in_iterator __LDG::sink_begin ( ) [inline]

return the first local sink of the directed graph

Definition at line 831 of file vgtl_ldag.h.

9.4.4.76 in_const_iterator __LDG::sink_begin ( ) const [inline]

return the first local sink of the directed graph

Definition at line 844 of file vgtl_ldag.h.

9.4.4.77 in_iterator __LDG::sink_end ( ) [inline]

return beyond the last local sink of the directed graph

Definition at line 837 of file vgtl_ldag.h.

9.4.4.78 in_const_iterator __LDG::sink_end ( ) const [inline]

return beyond the last local sink of the directed graph

Definition at line 850 of file vgtl_ldag.h.

9.4.4.79 size_type __LDG::size ( ) const [inline]

returns the size of the DG (number of nodes)

Definition at line 892 of file vgtl_ldag.h.

9.4.4.80 walker __LDG::sky ( ) [inline]

return a walker to the virtual sky node.

Definition at line 789 of file vgtl_ldag.h.

9.4.4.81 const_walker __LDG::sky ( ) const [inline]

return a const walker to the virtual sky node.

Definition at line 799 of file vgtl_ldag.h.

9.4.4.82 template<class Compare > void __LDG::sort_child_edges (walker *__position,* children_iterator *first,* children_iterator *last,* Compare *comp*) [inline]

sort the child edges in the range [first,last) according to `comp`

Definition at line 1356 of file vgtl_ldag.h.

9.4.4.83 template<class Compare > void __LDG::sort_child_edges (walker *__position,* Compare *comp*) [inline]

sort all child edges according to `comp`

Definition at line 1368 of file vgtl_ldag.h.

9.4.4.84 template<class Compare > void __LDG::sort_parent_edges (walker *__position,* parents_iterator *first,* parents_iterator *last,* Compare *comp*) [inline]

sort the parent edges in the range [first,last) according to `comp`

Definition at line 1362 of file vgtl_ldag.h.

9.4.4.85 template<class Compare > void __LDG::sort_parent_edges (walker *__position,* Compare *comp*) [inline]

sort all parent edges according to `comp`

Definition at line 1374 of file vgtl_ldag.h.

9.4.4.86 out_iterator __LDG::source_begin ( ) [inline]

return the first local source of the directed graph

Definition at line 805 of file vgtl_ldag.h.

9.4.4.87 out_const_iterator __LDG::source_begin ( ) const [inline]

return the first local source of the directed graph

Definition at line 818 of file vgtl_ldag.h.

9.4.4.88 out_iterator __LDG::source_end ( ) [inline]

return beyond the last local source of the directed graph

Definition at line 811 of file vgtl_ldag.h.

9.4.4.89 out_const_iterator __LDG::source_end ( ) const [inline]

return beyond the last local source of the directed graph

Definition at line 824 of file vgtl_ldag.h.

9.4.4.90 void __LDG::swap (_Self & __x) [inline]

swap two DGs

Definition at line 902 of file vgtl_ldag.h.

9.4.5 Member Data Documentation

9.4.5.1 _LDG_node<_Tp,_Ctr,_Iterator>* _LDG_base::_C_ground [protected, inherited]

the virtual ground node (below all roots)

Definition at line 541 of file vgtl_ldagbase.h.

9.4.5.2 int _LDG_base::_C_mark [protected, inherited]

an internal counter for setting marks during certain algorithms

Definition at line 545 of file vgtl_ldagbase.h.

9.4.5.3 _LDG_node<_Tp,_Ctr,_Iterator>* _LDG_base::_C_sky [protected, inherited]

the virtual sky node (above all leafs)

Definition at line 543 of file vgtl_ldagbase.h.

The documentation for this class was generated from the following file:

- vgtl_ldag.h

9.5 __one_iterator Class Reference

make an iterator out of one pointer

`#include <vgtl_intadapt.h>`

Public Types

- typedef std::random_access_iterator_tag iterator_category
  standard iterator definitions
- typedef ptrdiff_t difference_type
  standard iterator definitions
- typedef _Tp value_type
  standard iterator definitions
- typedef value_type * pointer
  standard iterator definitions
- typedef value_type & reference
  standard iterator definitions

Public Member Functions

- __one_iterator ()
  standard constructor
- __one_iterator (const value_type *__x)
  standard constructor setting the value
- __one_iterator (const _Self &__x)
  copy constructor
- __one_iterator (const pointer &__v, bool __a)
  constructor, explicitly setting value and iterator position
- reference operator* () const
  dereference operator

- _Self & operator++ ()
  standard increment, decrement, and access operators for random access
- _Self operator++ (int)
  standard increment, decrement, and access operators for random access
- _Self & operator-- ()
  standard increment, decrement, and access operators for random access
- _Self operator-- (int)
  standard increment, decrement, and access operators for random access
- _Self operator+ (difference_type __n) const
  standard increment, decrement, and access operators for random access
- _Self & operator+= (difference_type __n)
  standard increment, decrement, and access operators for random access
- _Self operator- (difference_type __n) const
  standard increment, decrement, and access operators for random access
- _Self & operator-= (difference_type __n)
  standard increment, decrement, and access operators for random access
- reference operator[ ] (difference_type __n) const
  standard increment, decrement, and access operators for random access

- bool operator== (const _Self &__x)

  comparsion operator
- bool operator!= (const _Self &__x)

  comparsion operator

Protected Attributes

- pointer __value_

  The single value of the 'sequence'.
- bool __at

  are we at begin()?

9.5.1 Detailed Description

This adaptor takes a pointer to a value of type `_Tp` and constructs an iterator, which only has two possibilities-:

- begin() points to the same place as the pointer
- end() is beyond the end. So a pointer is transformed to a sequence of length one, and this iterator iterates over it.

The documentation for this class was generated from the following file:

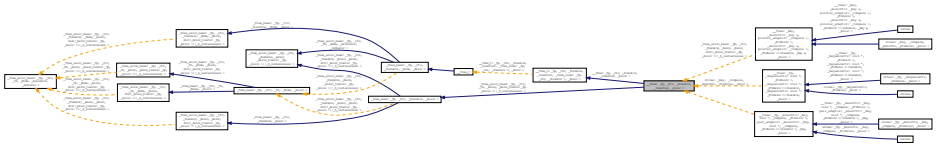
- vgtl_intadapt.h

9.6 __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc > Class Template Reference

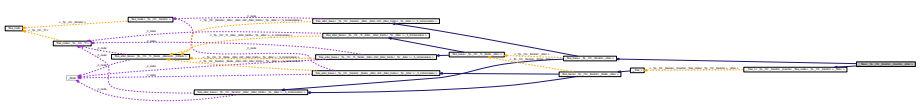
Tree base class without data hooks.

`#include <vgtl_tree.h>`

Inheritance diagram for __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >:



Collaboration diagram for __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >:



Public Types

- typedef _Tp value_type
- typedef _Node node_type
- typedef value_type * pointer
- typedef const value_type * const_pointer
- typedef value_type & reference
- typedef const value_type & const_reference
- typedef size_t size_type
- typedef ptrdiff_t difference_type
- typedef _Tree_iterator< _Tp, _Tp &, _Tp *, container_type, container_iterator > iterator
- typedef _Tree_iterator< _Tp, const _Tp &, const _Tp *, container_type, container_iterator > const_iterator
- typedef reverse_iterator < const_iterator > const_reverse_iterator
- typedef reverse_iterator < iterator > reverse_iterator
- typedef _Tree_walker< _Tp, _Tp &, _Tp *, container_type, container_iterator > walker
- typedef _Tree_walker< _Tp, const _Tp &, const _Tp *, container_type, container_iterator > const_walker
- typedef _Node node_type
- typedef _Tree_iterator< _Tp, _Tp &, _Tp *, container_type, children_iterator, node_type > iterator
- typedef _Tree_iterator< _Tp, const _Tp &, const _Tp *, container_type, children_iterator, node_type > const_iterator
- typedef std::reverse_iterator < const_iterator > const_reverse_iterator
- typedef std::reverse_iterator < iterator > reverse_iterator
- typedef _Iterator children_iterator
- typedef __one_iterator< void * > parents_iterator

Public Member Functions

- allocator_type get_allocator () const
- bool empty () const
- size_type max_size () const
- void insert_child (const __walker_base &__position, const _Tp &__x, const container_insert_arg &__It)
- void insert_child (const __walker_base &__position, const container_insert_arg &__It)
- void insert_children (const __walker_base &__position, size_type __n, const _Tp &__x, const container_iterator &__It)
- void erase (const __walker_base &__position)
- _Node * erase_tree (const __walker_base &__position)
- bool erase_child (const __walker_base &__position, const container_iterator &__It)
- _Node * erase_subtree (const __walker_base &__position, const container_iterator &__It)
- size_type depth (const recursive_walker &__position)
- __Tree (const allocator_type &__a=allocator_type())
- walker ground ()
- const_walker ground () const
- walker root (children_iterator __it)
- const_walker root (children_iterator __it) const
- walker root ()
- const_walker root () const
- iterator begin ()

- iterator end ()
- const_iterator begin () const
- const_iterator end () const
- reverse_iterator rbegin ()
- reverse_iterator rend ()
- const_reverse_iterator rbegin () const
- const_reverse_iterator rend () const
- reference getroot ()
- const_reference getroot () const
- __Tree (size_type __n, const _Tp &__value, const allocator_type &__a=allocator_type())
- __Tree (size_type __n)
- __Tree (const _Self &__x)
- virtual ~__Tree ()
- _Self & operator= (const _Self &__x)
- _Self & operator= (_Node *__x)
- void swap (_Self &__x)
- void insert_subtree (const __walker_base &__position, _Self &__subtree, const children_iterator &__It)
- void clear_children ()
- void add_all_children (_Output_Iterator fi, _Alloc *_parent)

Protected Member Functions

- _Node * _C_create_node (const _Tp &__x)
- _Node * _C_create_node ()
- _Alloc * _C_get_node ()
- void _C_put_node (_Alloc *__p)
- void _C_put_node (_Node *__p)

Protected Attributes

- _Alloc * _C_node

Friends

- bool operator==__VGTL_NULL_TMPL_ARGS (const __Tree &__x, const __Tree &__y)

9.6.1 Detailed Description

template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc>class __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >

This is the base class for all trees without data hooks

Definition at line 1885 of file vgtl_tree.h.

9.6.2 Member Typedef Documentation

9.6.2.1 typedef _Iterator _Tree_base< _Tp, _Ctr, _Iterator , _Alloc , _Alloc >::children_iterator [inherited]

iterator for accessing the children

Definition at line 1445 of file vgtl_tree.h.

9.6.2.2 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> typedef _Tree_iterator<_Tp,const _Tp&,const _Tp*,container_type,container_iterator> __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::const_iterator

the const iterator

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _Tree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 1263 of file vgtl_graph.h.

9.6.2.3 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> typedef _Tree_iterator<_Tp,const _Tp&,const _Tp*,container_type,children_iterator,node_type> __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::const_iterator

the const iterator

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _Tree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 1901 of file vgtl_tree.h.

9.6.2.4 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> typedef const value_type* __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::const_pointer

standard typedef

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _Tree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 1251 of file vgtl_graph.h.

9.6.2.5 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> typedef const value_type& __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::const_reference

standard typedef

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _Tree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 1253 of file vgtl_graph.h.

9.6.2.6 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> typedef reverse_iterator<const_iterator> __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::const_reverse_iterator

the const reverse iterator

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _Tree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 1266 of file vgtl_graph.h.

9.6.2.7 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> typedef std::reverse_iterator<const_iterator> __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::const_reverse_iterator

the const reverse iterator

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _Tree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 1905 of file vgtl_tree.h.

9.6.2.8 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> typedef _Tree_walker<_Tp,const _Tp&,const _Tp*,container_type,container_iterator> __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::const_walker

the (recursive) const walker

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _Tree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 1278 of file vgtl_graph.h.

9.6.2.9 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> typedef ptrdiff_t __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::difference_type

standard typedef

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _Tree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 1255 of file vgtl_graph.h.

9.6.2.10 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> typedef _Tree_iterator<_Tp,_Tp&,_Tp*,container_type,container_iterator> __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::iterator

the iterator

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _Tree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 1262 of file vgtl_graph.h.

9.6.2.11 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> typedef _Tree_iterator<_Tp,_Tp&,_Tp*,container_type,children_iterator,node_type> __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::iterator

the iterator

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _Tree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 1899 of file vgtl_tree.h.

9.6.2.12 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> typedef _Node __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::node_type

standard typedef

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _Tree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 1249 of file vgtl_graph.h.

9.6.2.13 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> typedef _Node __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::node_type

standard typedef

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _Tree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 1895 of file vgtl_tree.h.

9.6.2.14 typedef __one_iterator<void *> _Tree_base< _Tp, _Ctr, _Iterator , _Alloc , _Alloc >::parents_iterator [inherited]

iterator for accessing the parents

Definition at line 1447 of file vgtl_tree.h.

9.6.2.15 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> typedef value_type* __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::pointer

standard typedef

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _Tree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 1250 of file vgtl_graph.h.

9.6.2.16 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> typedef value_type& __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::reference

standard typedef

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _Tree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 1252 of file vgtl_graph.h.

9.6.2.17 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> typedef reverse_iterator<iterator> __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::reverse_iterator

the reverse iterator

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _Tree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 1267 of file vgtl_graph.h.

9.6.2.18 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> typedef std::reverse_iterator<iterator> __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::reverse_iterator

the reverse iterator

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _Tree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 1907 of file vgtl_tree.h.

9.6.2.19 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> typedef size_t __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::size_type

standard typedef

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _Tree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 1254 of file vgtl_graph.h.

9.6.2.20 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> typedef _Tp __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::value_type

standard typedef

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _Tree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 1248 of file vgtl_graph.h.

9.6.2.21 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> typedef _Tree_walker<_Tp,_Tp&,_Tp*,container_type,container_iterator> __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::walker

the (recursive) walker

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _Tree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 1277 of file vgtl_graph.h.

9.6.3 Constructor & Destructor Documentation

9.6.3.1 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::__Tree (const allocator_type & __a = allocator_type()) [inline, explicit]

standard constructor

Definition at line 1932 of file vgtl_tree.h.

9.6.3.2 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::__Tree (size_type __n, const _Tp & __value, const allocator_type & __a = allocator_type()) [inline]

construct a tree containing __n nodes with value __value at the root spot.

Definition at line 2004 of file vgtl_tree.h.

9.6.3.3 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::__Tree (size_type __n) [inline, explicit]

construct a tree containing __n nodes with default value at the root spot.

Definition at line 2011 of file vgtl_tree.h.

9.6.3.4 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::__Tree (const _Self & __x) [inline]

copy constructor

Definition at line 2016 of file vgtl_tree.h.

9.6.3.5 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> virtual __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::~__Tree ( ) [inline, virtual]

standard destructor

Definition at line 2019 of file vgtl_tree.h.

9.6.4 Member Function Documentation

9.6.4.1 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> _Node* __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::_C_create_node (const _Tp & __x) [inline, protected]

construct a new tree node containing data __x

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _Tree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 1295 of file vgtl_graph.h.

9.6.4.2 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> _Node* __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::_C_create_node ( ) [inline, protected]

construct a new tree node containing default data

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _Tree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 1308 of file vgtl_graph.h.

9.6.4.3 _Alloc * _Tree_alloc_base< _Tp, _Ctr, _Iterator , _Alloc , _Alloc , _IsStatic >::_C_get_node ( ) [inline, protected, inherited]

allocate a new node

Definition at line 1375 of file vgtl_tree.h.

9.6.4.4 void _Tree_alloc_base< _Tp, _Ctr, _Iterator , _Alloc , _Alloc , _IsStatic >::_C_put_node (_Alloc * __p) [inline, protected, inherited]

deallocate a node

Definition at line 1378 of file vgtl_tree.h.

9.6.4.5 template<class _Tp, class _Ctr, class _Tl, class _Node, class _Allocator, bool _IsStatic> void _Tree_alloc_base< _Tp, _Ctr, _Tl, _Node, _Allocator, _IsStatic >::_C_put_node (_Node * __p) [inline, protected, inherited]

deallocate a node

Definition at line 1378 of file vgtl_tree.h.

9.6.4.6 void _Tree_base< _Tp, _Ctr, _Iterator , _Alloc , _Alloc >::add_all_children (_Output_Iterator *fi,* _Alloc * *_parent*) [inherited]

add all children to the parent _parent. fi is a iterator to the children container of the parent

9.6.4.7 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> iterator __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::begin ( ) [inline]

return an iterator to the first node in walk

Definition at line 1964 of file vgtl_tree.h.

9.6.4.8 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> const_iterator __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::begin ( ) const [inline]

return a const iterator to the first node in walk

Definition at line 1973 of file vgtl_tree.h.

9.6.4.9 void _Tree_base< _Tp, _Ctr, _Iterator , _Alloc , _Alloc >::clear_children ( ) [inline, inherited]

clear all children of the root node

Definition at line 1466 of file vgtl_tree.h.

9.6.4.10 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> size_type __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::depth (const recursive_walker & *__position*) [inline]

return the depth of node __position in the tree

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _Tree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 1529 of file vgtl_graph.h.

9.6.4.11 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> bool __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::empty ( ) const [inline]

is the tree empty?

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _Tree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 1392 of file vgtl_graph.h.

9.6.4.12 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> iterator __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::end ( ) [inline]

return an iterator beyond the last node in walk

Definition at line 1968 of file vgtl_tree.h.

9.6.4.13 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> const_iterator __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::end ( ) const [inline]

return a const iterator beyond the last node in walk

Definition at line 1977 of file vgtl_tree.h.

9.6.4.14 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> void __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::erase (const __walker_base & __position) [inline]

erase the node at position __position.

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _Tree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 1444 of file vgtl_graph.h.

9.6.4.15 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> bool __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::erase_child (const __walker_base & __position, const container_iterator & __It) [inline]

erase the (leaf) child __It of node __position. This works if and only if the child is a leaf.

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _Tree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 1495 of file vgtl_graph.h.

9.6.4.16 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> _Node* __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::erase_subtree (const __walker_base & __position, const container_iterator & __It) [inline]

erase the subtree position __position, whose top node is the child at children_iterator position __It, and return its top node.

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _Tree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 1511 of file vgtl_graph.h.

9.6.4.17 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> _Node* __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::erase_tree (const __walker_base & __position) [inline]

erase the subtree starting at position __position, and return its top node.

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _Tree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 1471 of file vgtl_graph.h.

9.6.4.18 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> allocator_type __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::get_allocator ( ) const [inline]

construct an allocator object

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _Tree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 1259 of file vgtl_graph.h.

9.6.4.19 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> reference __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::getroot ( ) [inline]

get a reference to the virtual root node

Definition at line 1996 of file vgtl_tree.h.

9.6.4.20 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> const_reference __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::getroot ( ) const [inline]

get a const reference to the virtual root node

Definition at line 1998 of file vgtl_tree.h.

9.6.4.21 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> walker __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::ground ( ) [inline]

return a walker to the virtual root node.

Definition at line 1939 of file vgtl_tree.h.

9.6.4.22 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> const_walker __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::ground ( ) const [inline]

return a const walker to the virtual root node.

Definition at line 1943 of file vgtl_tree.h.

9.6.4.23 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> void __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::insert_child (const __walker_base & *__position,* const _Tp & *__x,* const container_insert_arg & *__It*) [inline]

add a child below __position with data __x, at the __It position in the __position - node's children container

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _Tree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 1409 of file vgtl_graph.h.

9.6.4.24 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> void __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::insert_child (const __walker_base & *__position,* const container_insert_arg & *__It*) [inline]

add a child below __position with default data, at the __It position in the __position - node's children container

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _Tree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 1415 of file vgtl_graph.h.

9.6.4.25 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> void __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::insert_children (const __walker_base & *__position,* size_type *__n,* const _Tp & *__x,* const container_iterator & *__It*) [inline]

add __n children below __position with data __x, after the __It position in the __position - node's children container

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _Tree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 1419 of file vgtl_graph.h.

9.6.4.26 void __Tree_t::insert_subtree (const __walker_base & __position, _Self & __subtree, const children_iterator & __It) [inline, inherited]

add a complete subtree __subtree below position __position and children iterator position __It.

Definition at line 1702 of file vgtl_tree.h.

9.6.4.27 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> size_type __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::max_size ( ) const [inline]

return the maximum possible size of the tree (theor. infinity)

Reimplemented from __Tree_t< _Tp, _Ctr, _Iterator, _Inserter, _Tree_node< _Tp, _Ctr, _Iterator >, _Alloc >.

Definition at line 1400 of file vgtl_graph.h.

9.6.4.28 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> _Self& __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::operator= (const _Self & __x)

standard assignment operator

9.6.4.29 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> _Self& __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::operator= (_Node * __x) [inline]

assign a tree from one node -> make this node the root node. This is useful for making trees out of erased subtrees.

Reimplemented in rstree, ratree, stree< _Key, _Compare, _AssocCtr, _PtrAlloc, _Alloc >, atree< _Tp, _AssocCtr, _Key, _Compare, _PtrAlloc, _Alloc >, and rntree.

Definition at line 2028 of file vgtl_tree.h.

9.6.4.30 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> reverse_iterator __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::rbegin ( ) [inline]

return a reverse iterator to the first node in walk

Definition at line 1982 of file vgtl_tree.h.

9.6.4.31 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> const_reverse_iterator __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::rbegin ( ) const [inline]

return a const reverse iterator to the first node in walk

Definition at line 1989 of file vgtl_tree.h.

9.6.4.32 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> reverse_iterator __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::rend ( ) [inline]

return a reverse iterator beyond the last node in walk

Definition at line 1985 of file vgtl_tree.h.

9.6.4.33 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> const_reverse_iterator __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::rend ( ) const [inline]

return a const reverse iterator beyond the last node in walk

Definition at line 1992 of file vgtl_tree.h.

9.6.4.34 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> walker __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::root (children_iterator __it) [inline]

return a walker to a root node.

Definition at line 1947 of file vgtl_tree.h.

9.6.4.35 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> const_walker __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::root (children_iterator __it) const [inline]

return a const walker to a root node.

Definition at line 1952 of file vgtl_tree.h.

9.6.4.36 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> walker __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::root ( ) [inline]

return a walker to the first non-virtual tree root

Definition at line 1957 of file vgtl_tree.h.

9.6.4.37 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> const_walker __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >::root ( ) const [inline]

return a const walker to the first non-virtual tree root

Definition at line 1960 of file vgtl_tree.h.

9.6.4.38 void __Tree_t::swap (_Self & __x) [inline, inherited]

swap two trees

Definition at line 1663 of file vgtl_tree.h.

9.6.5 Friends And Related Function Documentation

9.6.5.1 template<class _Tp, class _Ctr, class _Iterator, class _Inserter, class _Alloc> bool operator==__VGTL_NULL_TMPL_ARGS (const __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc > & __x, const __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc > & __y) [friend]

comparison operator

9.6.6 Member Data Documentation

9.6.6.1 _Alloc * _Tree_alloc_base< _Tp, _Ctr, _Iterator , _Alloc , _Alloc , _IsStatic >::_C_node [protected, inherited]

This is the node

Definition at line 1387 of file vgtl_tree.h.

The documentation for this class was generated from the following files:

- vgtl_graph.h
- vgtl_tree.h

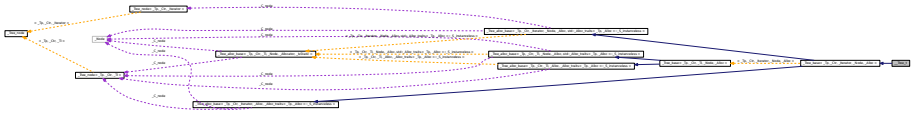
9.7 __Tree_t Class Reference

Tree base class.

`#include <vgtl_tree.h>`

Inheritance diagram for __Tree_t:

Collaboration diagram for __Tree_t:



Public Types

- typedef _Iterator children_iterator
- typedef __one_iterator< void * > parents_iterator
- typedef _Tree_iterator< _Tp, _Tp &, _Tp *, container_type, children_iterator, node_type > iterator
- typedef _Tree_iterator< _Tp, const _Tp &, const _Tp *, container_type, children_iterator, node_type > const_iterator
- typedef std::reverse_iterator < const_iterator > const_reverse_iterator
- typedef std::reverse_iterator < iterator > reverse_iterator
- typedef _RTree_walker< _Tp, _Tp &, _Tp *, container_type, children_iterator, node_type > walker
- typedef _RTree_walker< _Tp, const _Tp &, const _Tp *, container_type, children_iterator, node_type > const_walker

- typedef _Tp value_type
- typedef _Node node_type
- typedef value_type * pointer
- typedef const value_type * const_pointer
- typedef value_type & reference
- typedef const value_type & const_reference
- typedef size_t size_type
- typedef ptrdiff_t difference_type

Public Member Functions

- allocator_type get_allocator () const
- __Tree_t (const allocator_type &__a=allocator_type())
- bool empty () const
- size_type max_size () const

- void swap (_Self &__x)
- void insert_child (const __walker_base &__position, const _Tp &__x, const container_insert_arg &__It)
- void insert_child (const __walker_base &__position, const container_insert_arg &__It)
- void insert_children (const __walker_base &__position, size_type __n, const _Tp &__x, const children_iterator &__It)
- void insert_subtree (const __walker_base &__position, _Self &__subtree, const children_iterator &__It)
- void erase (const __walker_base &__position)
- _Node * erase_tree (const __walker_base &__position)
- bool erase_child (const __walker_base &__position, const children_iterator &__It)
- _Node * erase_subtree (const __walker_base &__position, const children_iterator &__It)
- size_type depth (const walker &__position)
- void clear ()
- __Tree_t (size_type __n, const _Tp &__value, const allocator_type &__a=allocator_type())
- __Tree_t (size_type __n)
- __Tree_t (const _Self &__x)
- virtual ~__Tree_t ()
- _Self & operator= (const _Self &__x)
- _Self & operator= (_Node *__x)
- void clear_children ()
- void add_all_children (_Output_Iterator fi, _Node *_parent)

Protected Member Functions

- _Node * _C_create_node (const _Tp &__x)
- _Node * _C_create_node ()
- _Node * _C_get_node ()
- void _C_put_node (_Node *__p)
- void _C_put_node (_Node *__p)

Protected Attributes

- _Node * _C_node

9.7.1 Detailed Description

This is the toplevel base class for all trees independent of allocators

9.7.2 Member Typedef Documentation

9.7.2.1 typedef _Iterator __Tree_t::children_iterator

iterator for accessing the children

Reimplemented from _Tree_base< _Tp, _Ctr, _Iterator, _Node, _Alloc >.

Definition at line 1563 of file vgtl_tree.h.

9.7.2.2 typedef _Tree_iterator<_Tp,const _Tp&,const _Tp*,container_type,children_iterator,node_type> __Tree_t::const_iterator

the const iterator

Reimplemented in __ITree, __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >, __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >, __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >, and __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >.

Definition at line 1593 of file vgtl_tree.h.

9.7.2.3 typedef const value_type* __Tree_t::const_pointer

standard typedef

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >, and __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >.

Definition at line 1578 of file vgtl_tree.h.

9.7.2.4 typedef const value_type& __Tree_t::const_reference

standard typedef

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >, and __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >.

Definition at line 1580 of file vgtl_tree.h.

9.7.2.5 typedef std::reverse_iterator<const_iterator> __Tree_t::const_reverse_iterator

the const reverse iterator

Reimplemented in __ITree, __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >, __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >, __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >-

::iterator, _Alloc >, and __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >.

Definition at line 1597 of file vgtl_tree.h.

9.7.2.6 typedef _RTree_walker<_Tp,const _Tp&,const _Tp*,container_type,children_iterator,node_type> __Tree_t::const_walker

the (recursive) const walker

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >, and __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >.

Definition at line 1614 of file vgtl_tree.h.

9.7.2.7 typedef ptrdiff_t __Tree_t::difference_type

standard typedef

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >, and __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >.

Definition at line 1582 of file vgtl_tree.h.

9.7.2.8 typedef _Tree_iterator<_Tp,_Tp&,_Tp*,container_type,children_iterator,node_type> __Tree_t::iterator

the iterator

Reimplemented in __ITree, __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >, __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >, __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >, and __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >.

Definition at line 1591 of file vgtl_tree.h.

9.7.2.9 typedef _Node __Tree_t::node_type

standard typedef

Reimplemented in __ITree, __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >,

__Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >, __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >, and __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >.

Definition at line 1576 of file vgtl_tree.h.

9.7.2.10 typedef __one_iterator<void *> __Tree_t::parents_iterator

iterator for accessing the parents

Reimplemented from _Tree_base< _Tp, _Ctr, _Iterator, _Node, _Alloc >.

Definition at line 1564 of file vgtl_tree.h.

9.7.2.11 typedef value_type* __Tree_t::pointer

standard typedef

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >, and __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >.

Definition at line 1577 of file vgtl_tree.h.

9.7.2.12 typedef value_type& __Tree_t::reference

standard typedef

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >, and __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >.

Definition at line 1579 of file vgtl_tree.h.

9.7.2.13 typedef std::reverse_iterator<iterator> __Tree_t::reverse_iterator

the reverse iterator

Reimplemented in __ITree, __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >, __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >, __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >, and __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >.

Definition at line 1599 of file vgtl_tree.h.

9.7.2.14 typedef size_t __Tree_t::size_type

standard typedef

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >, and __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >.

Definition at line 1581 of file vgtl_tree.h.

9.7.2.15 typedef _Tp __Tree_t::value_type

standard typedef

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >, and __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >.

Definition at line 1575 of file vgtl_tree.h.

9.7.2.16 typedef _RTree_walker<_Tp,_Tp&,_Tp*,container_type,children_iterator,node_type> __Tree_t::walker

the (recursive) walker

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >, and __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >.

Definition at line 1612 of file vgtl_tree.h.

9.7.3 Constructor & Destructor Documentation

9.7.3.1 __Tree_t::__Tree_t (const allocator_type & __a = allocator_type()) [inline, explicit]

standard constructor

Definition at line 1654 of file vgtl_tree.h.

9.7.3.2 __Tree_t::__Tree_t (size_type __n, const _Tp & __value, const allocator_type & __a = allocator_type()) [inline]

construct a tree containing __n nodes with value __value at the root spot.

Definition at line 1823 of file vgtl_tree.h.

9.7.3.3 __Tree_t::__Tree_t (size_type __n) [inline, explicit]

construct a tree containing ___n nodes with default value at the root spot.

Definition at line 1830 of file vgtl_tree.h.

9.7.3.4 __Tree_t::__Tree_t (const _Self & __x) [inline]

copy constructor

Definition at line 1849 of file vgtl_tree.h.

9.7.3.5 virtual __Tree_t::~__Tree_t ( ) [inline, virtual]

standard destructor

Definition at line 1858 of file vgtl_tree.h.

9.7.4 Member Function Documentation

9.7.4.1 _Node* __Tree_t::_C_create_node (const _Tp & __x) [inline, protected]

construct a new tree node containing data ___x

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >, and __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >.

Definition at line 1629 of file vgtl_tree.h.

9.7.4.2 _Node* __Tree_t::_C_create_node ( ) [inline, protected]

construct a new tree node containing default data

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >, and __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >.

Definition at line 1641 of file vgtl_tree.h.

9.7.4.3 _Node* _Tree_alloc_base< _Tp, _Ctr, _Iterator , _Node, _Alloc , _IsStatic >::_C_get_node ( ) [inline, protected, inherited]

allocate a new node

Definition at line 1375 of file vgtl_tree.h.

9.7.4.4 void _Tree_alloc_base< _Tp, _Ctr, _Iterator , _Node, _Alloc , _IsStatic >::_C_put_node (_Node * __p) [inline, protected, inherited]

deallocate a node

Definition at line 1378 of file vgtl_tree.h.

9.7.4.5 template<class _Tp, class _Ctr, class _Tl, class _Node, class _Allocator, bool _IsStatic> void _Tree_alloc_base< _Tp, _Ctr, _Tl, _Node, _Allocator, _IsStatic >::_C_put_node (_Node * __p) [inline, protected, inherited]

deallocate a node

Definition at line 1378 of file vgtl_tree.h.

9.7.4.6 void _Tree_base< _Tp, _Ctr, _Iterator, _Node, _Alloc >::add_all_children (_Output_Iterator *fi,* _Node * *_parent*) [inline, inherited]

add all children to the parent _parent. fi is a iterator to the children container of the parent

Definition at line 1539 of file vgtl_tree.h.

9.7.4.7 void __Tree_t::clear ( ) [inline]

empty the tree

Reimplemented from _Tree_base< _Tp, _Ctr, _Iterator, _Node, _Alloc >.

Definition at line 1817 of file vgtl_tree.h.

9.7.4.8 void _Tree_base< _Tp, _Ctr, _Iterator , _Node, _Alloc >::clear_children ( ) [inline, inherited]

clear all children of the root node

Definition at line 1466 of file vgtl_tree.h.

9.7.4.9 size_type __Tree_t::depth (const walker & *__position*) [inline]

return the depth of node __position in the tree

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >, and __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >.

Definition at line 1805 of file vgtl_tree.h.

9.7.4.10 bool __Tree_t::empty ( ) const [inline]

is the tree empty?

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >, and __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >.

Definition at line 1657 of file vgtl_tree.h.

9.7.4.11 void __Tree_t::erase (const __walker_base & *__position*) [inline]

erase the node at position __position.

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Key, _AssocCtr< _Key &,

pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >, and __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >.

Definition at line 1713 of file vgtl_tree.h.

9.7.4.12 bool __Tree_t::erase_child (const __walker_base & __position, const children_iterator & __It) [inline]

erase the (leaf) child ___It of node ___position. This works if and only if the child is a leaf.

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >.

Definition at line 1770 of file vgtl_tree.h.

9.7.4.13 _Node* __Tree_t::erase_subtree (const __walker_base & __position, const children_iterator & __It) [inline]

erase the subtree position ___position, whose top node is the child at children_iterator position ___It, and return its top node.

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >.

Definition at line 1790 of file vgtl_tree.h.

9.7.4.14 _Node* __Tree_t::erase_tree (const __walker_base & __position) [inline]

erase the subtree starting at position ___position, and return its top node.

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >, and __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >.

Definition at line 1743 of file vgtl_tree.h.

9.7.4.15 allocator_type __Tree_t::get_allocator ( ) const [inline]

construct an allocator object

Reimplemented from _Tree_alloc_base< _Tp, _Ctr, _TI, _Node, _Allocator, _IsStatic >.

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >, and __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >.

Definition at line 1587 of file vgtl_tree.h.

9.7.4.16 void __Tree_t::insert_child (const __walker_base & __position, const _Tp & __x, const container_insert_arg & __It) [inline]

add a child below ___position with data ___x, at the ___It position in the ___position - node's children container

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >, and __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >.

Definition at line 1668 of file vgtl_tree.h.

9.7.4.17 void __Tree_t::insert_child (const __walker_base & __position, const container_insert_arg & __It) [inline]

add a child below __position with default data, at the __It position in the __position - node's children container

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >, and __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >.

Definition at line 1676 of file vgtl_tree.h.

9.7.4.18 void __Tree_t::insert_children (const __walker_base & __position, size_type __n, const _Tp & __x, const children_iterator & __It) [inline]

add __n children below __position with data __x, after the __It position in the __position - node's children container

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >.

Definition at line 1682 of file vgtl_tree.h.

9.7.4.19 void __Tree_t::insert_subtree (const __walker_base & __position, _Self & __subtree, const children_iterator & __It) [inline]

add a complete subtree __subtree below position __position and children iterator position __It.

Definition at line 1702 of file vgtl_tree.h.

9.7.4.20 size_type __Tree_t::max_size ( ) const [inline]

return the maximum possible size of the tree (theor. infinity)

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >, and __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >.

Definition at line 1660 of file vgtl_tree.h.

9.7.4.21 _Self& __Tree_t::operator= (const _Self & __x)

standard assignment operator

Reimplemented in __Tree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >, and __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc

>, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >.

9.7.4.22 _Self& __Tree_t::operator= (_Node * __x) [inline]

assign a tree from one node -> make this node the root node. This is useful for making trees out of erased subtrees.

Definition at line 1867 of file vgtl_tree.h.

9.7.4.23 void __Tree_t::swap (_Self & __x) [inline]

swap two trees

Definition at line 1663 of file vgtl_tree.h.

9.7.5 Member Data Documentation

9.7.5.1 _Node* _Tree_alloc_base< _Tp, _Ctr, _Iterator , _Node, _Alloc , _IsStatic >::_C_node [protected, inherited]

This is the node

Definition at line 1387 of file vgtl_tree.h.

The documentation for this class was generated from the following file:

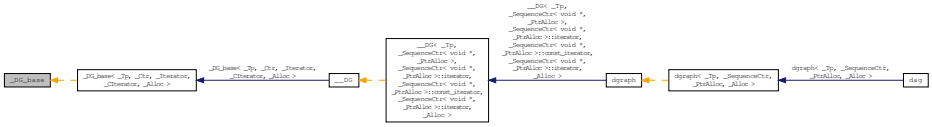
- vgtl_tree.h

9.8 _DG_base Class Reference

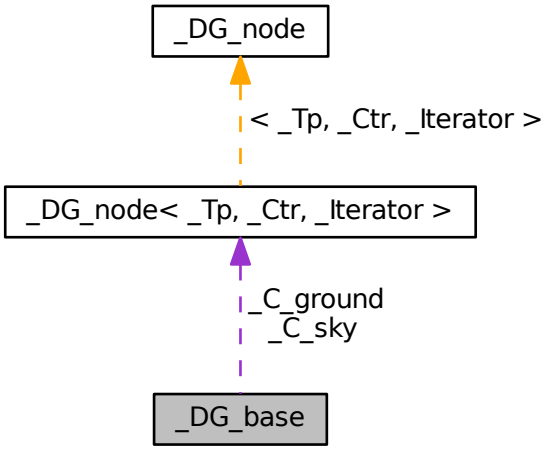
Directed graph base class for allocator encapsulation.

`#include <vgtl_dagbase.h>`

Inheritance diagram for _DG_base:



Collaboration diagram for _DG_base:



Public Types

- typedef _Alloc allocator_type
- typedef _Ctr container_type
- typedef _Iterator children_iterator
- typedef _Iterator parents_iterator

Public Member Functions

- allocator_type get_allocator () const
- _DG_base (const allocator_type &)
- ~_DG_base ()
- void clear ()

Protected Member Functions

- void clear_graph (_DG_node< _Tp, _Ctr, _Iterator > *_node)
- _DG_node< _Tp, _Ctr, _Iterator > * _C_get_node ()
- void _C_put_node (_DG_node< _Tp, _Ctr, _Iterator > *__p)
- void clear_children ()
- void clear_parents ()
- template<class _Output_Iterator >
  void add_all_children (_Output_Iterator fi, _DG_node< _Tp, _Ctr, _Iterator > *_parent)
- template<class _Output_Iterator >
  void add_all_parents (_Output_Iterator fi, _DG_node< _Tp, _Ctr, _Iterator > *_child)

Protected Attributes

- _DG_node< _Tp, _Ctr, _Iterator > * _C_ground
- _DG_node< _Tp, _Ctr, _Iterator > * _C_sky
- int _C_mark

9.8.1 Detailed Description

Base directed graph class top level that encapsulates details of allocators. This class is same as _DG_base and _DG_alloc_base if STL doesn't support standard allocators.

9.8.2 Member Typedef Documentation

9.8.2.1 typedef _Alloc _DG_base::allocator_type

allocator type

Reimplemented in __DG.

Definition at line 353 of file vgtl_dagbase.h.

9.8.2.2 typedef _Iterator _DG_base::children_iterator

iterator for accessing the children

Reimplemented in __DG.

Definition at line 360 of file vgtl_dagbase.h.

9.8.2.3 typedef _Ctr _DG_base::container_type

internal container used to store the children

Reimplemented in __DG.

Definition at line 358 of file vgtl_dagbase.h.

9.8.2.4 typedef _Iterator _DG_base::parents_iterator

iterator for accessing the parents

Reimplemented in __DG.

Definition at line 363 of file vgtl_dagbase.h.

9.8.3 Constructor & Destructor Documentation

9.8.3.1 _DG_base::_DG_base (const allocator_type &) [inline]

constructor initializing the allocator and the virtual nodes

Definition at line 367 of file vgtl_dagbase.h.

9.8.3.2 _DG_base::~_DG_base ( ) [inline]

standard destructor

Definition at line 388 of file vgtl_dagbase.h.

9.8.4 Member Function Documentation

9.8.4.1 _DG_node<_Tp,_Ctr,_Iterator>* _DG_base::_C_get_node ( ) [inline, protected]

allocate a new node

Definition at line 405 of file vgtl_dagbase.h.

9.8.4.2 void _DG_base::_C_put_node (_DG_node< _Tp, _Ctr, _Iterator > * __p) [inline, protected]

deallocate a node

Definition at line 408 of file vgtl_dagbase.h.

9.8.4.3 template<class _Output_Iterator > void _DG_base::add_all_children (_Output_Iterator *fi,* _DG_node< _Tp, _Ctr, _Iterator > * *_parent*) [protected]

add all children to the parent _parent. fi is a iterator to the children container of the parent

9.8.4.4 template<class _Output_Iterator > void _DG_base::add_all_parents (_Output_Iterator *fi,* _DG_node< _Tp, _Ctr, _Iterator > * *_child*) [protected]

add all parents to the child _child. fi is a iterator to the parents container of the child

9.8.4.5 void _DG_base::clear ( )

empty the graph

Reimplemented in __DG.

9.8.4.6 void _DG_base::clear_children ( ) [inline, protected]

clear all children of the root node

Definition at line 420 of file vgtl_dagbase.h.

9.8.4.7 void _DG_base::clear_graph (_DG_node< _Tp, _Ctr, _Iterator > * *_node*) [protected]

removes recursively all nodes downward starting from _node.

9.8.4.8 void _DG_base::clear_parents ( ) [inline, protected]

clear all parents of the leaf node

Definition at line 423 of file vgtl_dagbase.h.

9.8.4.9 allocator_type _DG_base::get_allocator ( ) const [inline]

get an allocator object

Reimplemented in __DG.

Definition at line 355 of file vgtl_dagbase.h.

9.8.5 Member Data Documentation

9.8.5.1 _DG_node<_Tp,_Ctr,_Iterator>* _DG_base::_C_ground [protected]

the virtual ground node (below all roots)

Definition at line 413 of file vgtl_dagbase.h.

9.8.5.2 int _DG_base::_C_mark [protected]

an internal counter for setting marks during certain algorithms

Definition at line 417 of file vgtl_dagbase.h.

9.8.5.3 _DG_node<_Tp,_Ctr,_Iterator>* _DG_base::_C_sky [protected]

the virtual sky node (above all leafs)

Definition at line 415 of file vgtl_dagbase.h.

The documentation for this class was generated from the following file:

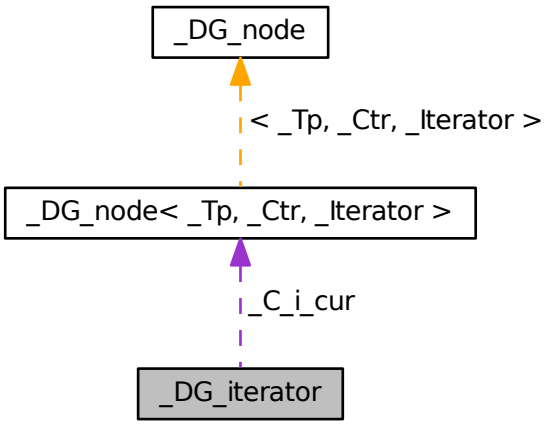
- vgtl_dagbase.h

9.9 _DG_iterator Class Reference

iterator through the directed graph

`#include <vgtl_dag.h>`

Collaboration diagram for _DG_iterator:



Public Types

- typedef std::bidirectional_iterator_tag iterator_category

- typedef _Tp value_type
- typedef _Ptr pointer
- typedef _Ref reference
- typedef _DG_node< _Tp, _Ctr, _Iterator > _Node
- typedef size_t size_type
- typedef ptrdiff_t difference_type

Public Member Functions

- _DG_iterator ()
- _DG_iterator (const iterator &__x)
- reference operator* () const
- pointer operator-> () const
- _Self & operator= (const _Walk &__x)

- bool operator== (const _Self &__x) const
- bool operator!= (const _Self &__x) const

- _Self & operator++ ()
- _Self operator++ (int)
- _Self & operator-- ()
- _Self operator-- (int)

Protected Attributes

- _Node * _C_i_cur
- std::vector< _Ctr_iterator > _C_i_cur_it

9.9.1 Detailed Description

This is an iterator, which visits each node of a directed graph once. It is based on a preorder depth-first automatic walker which visits a child if and only if the parent is the first in the list.

9.9.2 Member Typedef Documentation

9.9.2.1 typedef _DG_node<_Tp,_Ctr,_Iterator> _DG_iterator::_Node

standard iterator definition

Definition at line 292 of file vgtl_dag.h.

9.9.2.2 typedef ptrdiff_t _DG_iterator::difference_type

standard iterator definition

Definition at line 294 of file vgtl_dag.h.

9.9.2.3 typedef std::bidirectional_iterator_tag _DG_iterator::iterator_category

standard iterator definition

Definition at line 288 of file vgtl_dag.h.

9.9.2.4 typedef _Ptr _DG_iterator::pointer

standard iterator definition

Definition at line 290 of file vgtl_dag.h.

9.9.2.5 typedef _Ref _DG_iterator::reference

standard iterator definition

Definition at line 291 of file vgtl_dag.h.

9.9.2.6 typedef size_t _DG_iterator::size_type

standard iterator definition

Definition at line 293 of file vgtl_dag.h.

9.9.2.7 typedef _Tp _DG_iterator::value_type

standard iterator definition

Definition at line 289 of file vgtl_dag.h.

9.9.3 Constructor & Destructor Documentation

9.9.3.1 _DG_iterator::_DG_iterator ( ) [inline]

standard constructor

Definition at line 307 of file vgtl_dag.h.

9.9.3.2 _DG_iterator::_DG_iterator (const iterator & __x) [inline]

copy constructor

Definition at line 309 of file vgtl_dag.h.

9.9.4 Member Function Documentation

9.9.4.1 bool _DG_iterator::operator!= (const _Self & __x) const [inline]

comparison operator

Definition at line 322 of file vgtl_dag.h.

9.9.4.2 reference _DG_iterator::operator* ( ) const [inline]

dereference operator

Definition at line 332 of file vgtl_dag.h.

9.9.4.3 _Self& _DG_iterator::operator++ ( ) [inline]

in(de)crement operator

Definition at line 360 of file vgtl_dag.h.

9.9.4.4 _Self _DG_iterator::operator++ (int) [inline]

in(de)crement operator

Definition at line 364 of file vgtl_dag.h.

9.9.4.5 _Self& _DG_iterator::operator-- ( ) [inline]

in(de)crement operator

Definition at line 370 of file vgtl_dag.h.

9.9.4.6 _Self _DG_iterator::operator-- (int) [inline]

in(de)crement operator

Definition at line 374 of file vgtl_dag.h.

9.9.4.7 pointer _DG_iterator::operator-> ( ) const [inline]

pointer operator

Definition at line 336 of file vgtl_dag.h.

9.9.4.8 _Self& _DG_iterator::operator= (const _Walk & __x) [inline]

assignment to iterator from walker

Definition at line 349 of file vgtl_dag.h.

9.9.4.9 bool _DG_iterator::operator== (const _Self & __x) const [inline]

comparison operator

Definition at line 314 of file vgtl_dag.h.

9.9.5 Member Data Documentation

9.9.5.1 _Node* _DG_iterator::_C_i_cur [protected]

The current node

Definition at line 301 of file vgtl_dag.h.

9.9.5.2 std::vector<_Ctr_iterator> _DG_iterator::_C_i_cur_it [protected]

The internal stack

Definition at line 303 of file vgtl_dag.h.

The documentation for this class was generated from the following file:

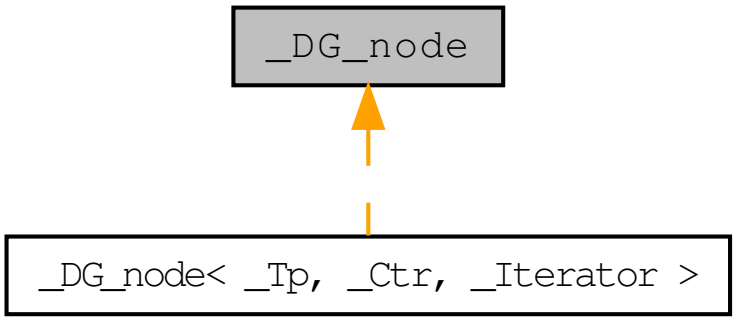
- vgtl_dag.h

9.10 _DG_node Class Reference

directed graph node

`#include <vgtl_dagbase.h>`

Inheritance diagram for _DG_node:



Public Member Functions

- _DG_node ()
- ~_DG_node ()
- void clear_children ()
- void clear_parents ()
- _Ctr_iterator get_childentry_iterator (const _Void_pointer __p)
- _Ctr_iterator get_parententry_iterator (const _Void_pointer __p)
- template<class _Output_Iterator >
  void add_all_children (_Output_Iterator fi, _Self *_parent)
- template<class _Output_Iterator >
  void add_all_parents (_Output_Iterator fi, _Self *_child)
- template<class Compare >
  void sort_child_edges (_Ctr_iterator first, _Ctr_iterator last, Compare comp)
- template<class Compare >
  void sort_parent_edges (_Ctr_iterator first, _Ctr_iterator last, Compare comp)

Public Attributes

- _Tp _C_data
- _Ctr _C_parents
- _Ctr _C_children
- int _C_visited

9.10.1 Detailed Description

This is the node for a directed graph

9.10.2 Constructor & Destructor Documentation

9.10.2.1 _DG_node::_DG_node ( ) [inline]

standard constructor

Definition at line 63 of file vgtl_dagbase.h.

9.10.2.2 _DG_node::~_DG_node ( ) [inline]

standard destructor

Definition at line 74 of file vgtl_dagbase.h.

9.10.3 Member Function Documentation

9.10.3.1 template<class _Output_Iterator > void _DG_node::add_all_children (_Output_Iterator *fi,* _Self * *_parent*)

add all children to parent _parent. fi is an iterator to the children container of _parent

9.10.3.2 template<class _Output_Iterator > void _DG_node::add_all_parents (_Output_Iterator *fi,* _Self * *_child*)

add all parents to child _child. fi is an iterator to the parents container of _child

9.10.3.3 void _DG_node::clear_children ( ) [inline]

erase all children entries

Definition at line 81 of file vgtl_dagbase.h.

9.10.3.4 void _DG_node::clear_parents ( ) [inline]

erase all parents entries

Definition at line 84 of file vgtl_dagbase.h.

9.10.3.5 _Ctr_iterator _DG_node::get_childentry_iterator (const _Void_pointer *__p*) [inline]

find the iterator into the children container for child ___p

Definition at line 88 of file vgtl_dagbase.h.

9.10.3.6 _Ctr_iterator _DG_node::get_parententry_iterator (const _Void_pointer *__p*) [inline]

find the iterator into the parents container for parent ___p

Definition at line 97 of file vgtl_dagbase.h.

9.10.3.7 template<class Compare > void _DG_node::sort_child_edges (_Ctr_iterator *first,* _Ctr_iterator *last,* Compare *comp*) [inline]

sort the children according to comp

Definition at line 124 of file vgtl_dagbase.h.

9.10.3.8 template<class Compare > void _DG_node::sort_parent_edges (_Ctr_iterator *first*, _Ctr_iterator *last*, Compare *comp*) [inline]

sort the parents according to comp

Definition at line 131 of file vgtl_dagbase.h.

9.10.4 Member Data Documentation

9.10.4.1 _Ctr _DG_node::_C_children

the edges to the children

Definition at line 58 of file vgtl_dagbase.h.

9.10.4.2 _Tp _DG_node::_C_data

the node data

Definition at line 54 of file vgtl_dagbase.h.

9.10.4.3 _Ctr _DG_node::_C_parents

the edges to the parents

Definition at line 56 of file vgtl_dagbase.h.

9.10.4.4 int _DG_node::_C_visited

internal counter for marks in algorithms

Definition at line 60 of file vgtl_dagbase.h.

The documentation for this class was generated from the following file:

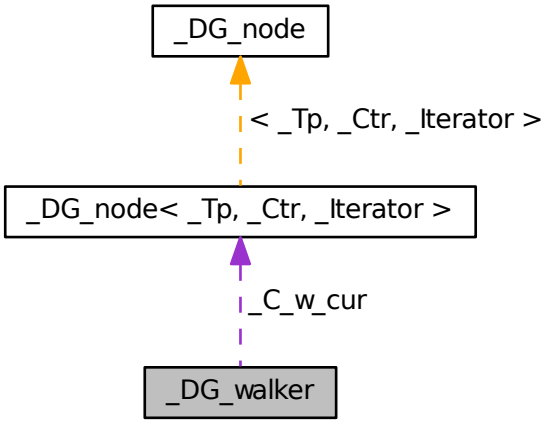
- vgtl_dagbase.h

9.11 _DG_walker Class Reference

recursive directed graph walkers

```
#include <vgtl_dag.h>
```

Collaboration diagram for _DG_walker:



Public Types

- typedef _Tp value_type
- typedef _Ptr pointer
- typedef _Ref reference

- typedef _Ctr_iterator children_iterator
- typedef _Ctr_iterator parents_iterator
- typedef _Ctr_const_iterator children_const_iterator
- typedef _Ctr_const_iterator parents_const_iterator
- typedef _Node node_type
- typedef size_t size_type
- typedef ptrdiff_t difference_type

Public Member Functions

- _DG_walker ()
- _DG_walker (_Node *__x)
- _DG_walker (const walker &__x)
- reference operator* () const
- pointer operator-> () const
- const _Node * node ()
- size_type n_children () const
- size_type n_parents () const

- bool is_root () const
- bool is_leaf () const
- bool is_ground () const
- bool is_sky () const
- children_iterator child_begin ()
- children_iterator child_end ()
- parents_iterator parent_begin ()
- parents_iterator parent_end ()
- template<class _Function >
  _Function for_each_child (_Function __f)
- template<class _Function >
  _Function for_each_parent (_Function __f)
- _Self operator<< (parents_iterator __i)
- _Self operator>> (children_iterator __i)
- _Self & operator<<= (parents_iterator __i)
- _Self & operator>>= (children_iterator __i)
- _Self operator<< (parents_const_iterator __i)
- _Self operator>> (children_const_iterator __i)
- _Self & operator<<= (parents_const_iterator __i)
- _Self & operator>>= (children_const_iterator __i)
- _Self & operator= (const _Itr &__x)
- _Self & operator= (const _Self &__x)
- _Self & operator= (const _Node &__n)

- bool operator== (const _Self &__x) const
- bool operator!= (const _Self &__x) const

Public Attributes

- _Node * _C_w_cur

9.11.1 Detailed Description

This is the class defining reursive directed graph walkers, which walk directed graphs under guidance.

9.11.2 Member Typedef Documentation

9.11.2.1 typedef _Ctr_const_iterator _DG_walker::children_const_iterator

standard walker definition

Definition at line 91 of file vgtl_dag.h.

9.11.2.2 typedef _Ctr_iterator _DG_walker::children_iterator

standard walker definition

Definition at line 89 of file vgtl_dag.h.

9.11.2.3 typedef ptrdiff_t _DG_walker::difference_type

standard walker definition

Definition at line 96 of file vgtl_dag.h.

9.11.2.4 typedef _Node _DG_walker::node_type

standard walker definition

Definition at line 93 of file vgtl_dag.h.

9.11.2.5 typedef _Ctr_const_iterator _DG_walker::parents_const_iterator

standard walker definition

Definition at line 92 of file vgtl_dag.h.

9.11.2.6 typedef _Ctr_iterator _DG_walker::parents_iterator

standard walker definition

Definition at line 90 of file vgtl_dag.h.

9.11.2.7 typedef _Ptr _DG_walker::pointer

standard walker definition

Definition at line 77 of file vgtl_dag.h.

9.11.2.8 typedef _Ref _DG_walker::reference

standard walker definition

Definition at line 78 of file vgtl_dag.h.

9.11.2.9 typedef size_t _DG_walker::size_type

standard walker definition

Definition at line 95 of file vgtl_dag.h.

9.11.2.10 typedef _Tp _DG_walker::value_type

standard walker definition

Definition at line 76 of file vgtl_dag.h.

9.11.3 Constructor & Destructor Documentation

9.11.3.1 _DG_walker::_DG_walker ( ) [inline]

standard constructor

Definition at line 105 of file vgtl_dag.h.

9.11.3.2 _DG_walker::_DG_walker (_Node * __x) [inline]

constructor setting the position

Definition at line 109 of file vgtl_dag.h.

9.11.3.3 _DG_walker::_DG_walker (const walker & __x) [inline]

copy constructor

Definition at line 112 of file vgtl_dag.h.

9.11.4 Member Function Documentation

9.11.4.1 children_iterator _DG_walker::child_begin ( ) [inline]

return children_iterator to first child

Definition at line 158 of file vgtl_dag.h.

9.11.4.2 children_iterator _DG_walker::child_end ( ) [inline]

return children_iterator beyond last child

Definition at line 162 of file vgtl_dag.h.

9.11.4.3 template<class _Function > _Function _DG_walker::for_each_child (_Function __f) [inline]

apply the function ___f to all children

Definition at line 177 of file vgtl_dag.h.

9.11.4.4 template<class _Function > _Function _DG_walker::for_each_parent (_Function __f) [inline]

apply the function ___f to all parents

Definition at line 183 of file vgtl_dag.h.

9.11.4.5 bool _DG_walker::is_ground ( ) const [inline]

is this node a virtual node - the ground (below all roots)?

Definition at line 153 of file vgtl_dag.h.

9.11.4.6 bool _DG_walker::is_leaf ( ) const [inline]

is this node a leaf?

Definition at line 142 of file vgtl_dag.h.

9.11.4.7 bool _DG_walker::is_root ( ) const [inline]

is this node a root?

Definition at line 132 of file vgtl_dag.h.

9.11.4.8 bool _DG_walker::is_sky ( ) const [inline]

is this node a virtual node - the sky (above all leafs)?

Definition at line 155 of file vgtl_dag.h.

9.11.4.9 size_type _DG_walker::n_children ( ) const [inline]

return the number of children

Definition at line 127 of file vgtl_dag.h.

9.11.4.10 size_type _DG_walker::n_parents ( ) const [inline]

return the number of parents

Definition at line 129 of file vgtl_dag.h.

9.11.4.11 const _Node* _DG_walker::node ( ) [inline]

retrieve the full node

Definition at line 124 of file vgtl_dag.h.

9.11.4.12 bool _DG_walker::operator!= (const _Self & __x) const [inline]

comparison operator

Definition at line 193 of file vgtl_dag.h.

9.11.4.13 reference _DG_walker::operator* ( ) const [inline]

dereference operator

Definition at line 115 of file vgtl_dag.h.

9.11.4.14 pointer _DG_walker::operator-> ( ) const [inline]

pointer operator

Definition at line 119 of file vgtl_dag.h.

9.11.4.15 _Self _DG_walker::operator<< (parents_iterator __i) [inline]

this function returns the walker pointing to the required parent

Definition at line 198 of file vgtl_dag.h.

9.11.4.16 _Self _DG_walker::operator<< (parents_const_iterator __i) [inline]

this function returns the walker pointing to the required parent

Definition at line 224 of file vgtl_dag.h.

9.11.4.17 _Self& _DG_walker::operator<<= (parents_iterator __i) [inline]

here the original walker goes to the required parent

Definition at line 212 of file vgtl_dag.h.

9.11.4.18 _Self& _DG_walker::operator<<= (parents_const_iterator __i) [inline]

here the original walker goes to the required parent

Definition at line 238 of file vgtl_dag.h.

9.11.4.19 _Self& _DG_walker::operator= (const _Itr & __x) [inline]

new walker is assigned from that particular iterator

Definition at line 250 of file vgtl_dag.h.

9.11.4.20 _Self& _DG_walker::operator= (const _Self & __x) [inline]

standard assignment operator

Definition at line 256 of file vgtl_dag.h.

9.11.4.21 _Self& _DG_walker::operator= (const _Node & __n) [inline]

a walker is assigned to any pointer to a graph node

Definition at line 262 of file vgtl_dag.h.

9.11.4.22 bool _DG_walker::operator== (const _Self & __x) const [inline]

comparison operator

Definition at line 191 of file vgtl_dag.h.

9.11.4.23 _Self _DG_walker::operator>> (children_iterator __i) [inline]

this function returns the walker pointing to the required child

Definition at line 205 of file vgtl_dag.h.

9.11.4.24 _Self _DG_walker::operator>> (children_const_iterator __i) [inline]

this function returns the walker pointing to the required child

Definition at line 231 of file vgtl_dag.h.

9.11.4.25 _Self& _DG_walker::operator>>= (children_iterator __i) [inline]

here the original walker goes to the required child

Definition at line 218 of file vgtl_dag.h.

9.11.4.26 _Self& _DG_walker::operator>>= (children_const_iterator __i) [inline]

here the original walker goes to the required child

Definition at line 244 of file vgtl_dag.h.

9.11.4.27 parents_iterator _DG_walker::parent_begin ( ) [inline]

return parents_iterator to first parent

Definition at line 167 of file vgtl_dag.h.

9.11.4.28 parents_iterator _DG_walker::parent_end ( ) [inline]

return parents_iterator beyond last parent

Definition at line 171 of file vgtl_dag.h.

9.11.5 Member Data Documentation

9.11.5.1 _Node* _DG_walker::_C_w_cur

pointer to the current node

Definition at line 101 of file vgtl_dag.h.

The documentation for this class was generated from the following file:

- vgtl_dag.h

9.12 _G_compare_adaptor Class Reference

Adaptor for data comparison in graph nodes.

`#include <vgtl_intadapt.h>`

Public Member Functions

- _G_compare_adaptor (const Predicate &__p)
  constructor
- bool operator() (const void *r, const void *l) const
  make it a function object on the nodes

9.12.1 Detailed Description

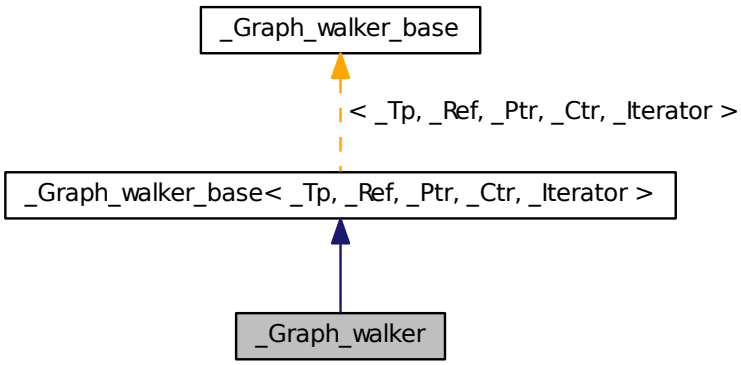
This adaptor takes a binary predicate for node data and transforms it to a binary predicate on the nodes.

The documentation for this class was generated from the following file:

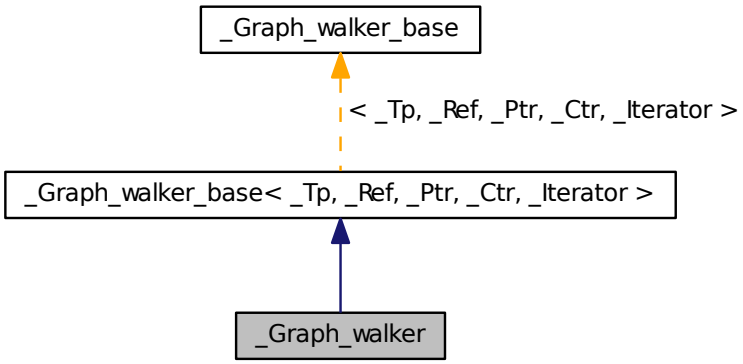
- vgtl_intadapt.h

9.13 _Graph_walker Class Reference

Inheritance diagram for _Graph_walker:



Collaboration diagram for _Graph_walker:

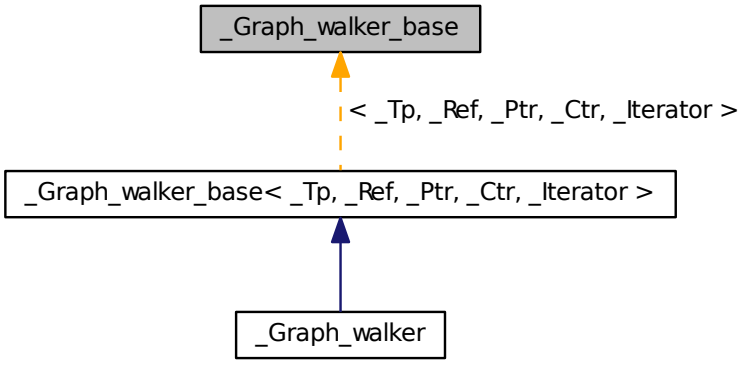


The documentation for this class was generated from the following file:

- vgtl_graph.h

9.14 _Graph_walker_base Class Reference

Inheritance diagram for _Graph_walker_base:



The documentation for this class was generated from the following file:

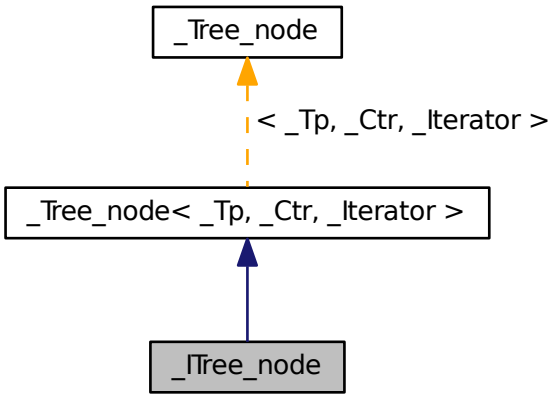
- vgtl_graph.h

9.15 _ITree_node Class Reference

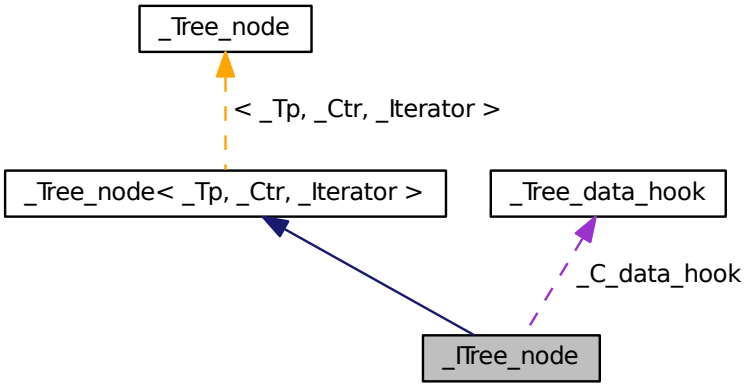
tree node for trees with data hooks

`#include <vgtl_tree.h>`

Inheritance diagram for _ITree_node:



Collaboration diagram for _ITree_node:



Public Member Functions

- _ITree_node ()

- void initialize ()
- void get_rid_of ()
- ctree_data_hook & data_hook ()
- void clear_tree ()
- void clear_children ()
- _Ctr_iterator get_childentry_iterator (_Void_pointer __p)
- void add_all_children (_Output_Iterator fi, _Self *_parent)
- void sort_children (_Ctr_iterator first, _Ctr_iterator last, Compare comp)
- void sort_parents (_Ctr_iterator first, _Ctr_iterator last, Compare comp)

Public Attributes

- ctree_data_hook _C_data_hook
- _Tp _C_data
- _Void_pointer _C_parent
- _Ctr _C_children

9.15.1 Detailed Description

This is the tree node for a tree with data hooks

9.15.2 Constructor & Destructor Documentation

9.15.2.1 _ITree_node::_ITree_node ( ) [inline]

standard constructor

Definition at line 151 of file vgtl_tree.h.

9.15.3 Member Function Documentation

9.15.3.1 void _Tree_node< _Tp, _Ctr, _Iterator >::add_all_children (_Output_Iterator *fi*, _Self * *_parent*) [inherited]

add all children to parent _parent. fi is an iterator to the children container of _parent

Definition at line 181 of file vgtl_tree.h.

9.15.3.2 void _Tree_node::clear_children ( ) [inline, inherited]

erase all children entries

Definition at line 101 of file vgtl_tree.h.

9.15.3.3 void _Tree_node< _Tp, _Ctr, _Iterator >::clear_tree ( ) [inherited]

remove the whole subtree below this node

Definition at line 196 of file vgtl_tree.h.

9.15.3.4 ctree_data_hook& _ITree_node::data_hook ( ) [inline]

return the data of the data hook

Definition at line 172 of file vgtl_tree.h.

9.15.3.5 _Ctr_iterator _Tree_node::get_childentry_iterator (_Void_pointer __p) [inline, inherited]

find the iterator into the children container for child __p

Definition at line 105 of file vgtl_tree.h.

9.15.3.6 void _ITree_node::get_rid_of ( ) [inline]

remove the children container

Reimplemented from _Tree_node< _Tp, _Ctr, _Iterator >.

Definition at line 166 of file vgtl_tree.h.

9.15.3.7 void _ITree_node::initialize ( ) [inline]

initialize the data structure

Reimplemented from _Tree_node< _Tp, _Ctr, _Iterator >.

Definition at line 159 of file vgtl_tree.h.

9.15.3.8 void _Tree_node::sort_children (_Ctr_iterator *first*, _Ctr_iterator *last*, Compare *comp*) [inline, inherited]

sort the children according to comp

Definition at line 122 of file vgtl_tree.h.

9.15.3.9 void _Tree_node::sort_parents (_Ctr_iterator *first*, _Ctr_iterator *last*, Compare *comp*) [inline, inherited]

sort the children according to comp, i.e. do nothing here

Definition at line 129 of file vgtl_tree.h.

9.15.4 Member Data Documentation

9.15.4.1 _Ctr _Tree_node::_C_children [inherited]

the edges to the children

Definition at line 77 of file vgtl_tree.h.

9.15.4.2 _Tp _Tree_node::_C_data [inherited]

the node data

Definition at line 73 of file vgtl_tree.h.

9.15.4.3 ctree_data_hook _ITree_node::_C_data_hook

the data hook for trees with data hook

Definition at line 148 of file vgtl_tree.h.

9.15.4.4 _Void_pointer _Tree_node::_C_parent [inherited]

the edge to the parent

Definition at line 75 of file vgtl_tree.h.

The documentation for this class was generated from the following file:

- vgtl_tree.h

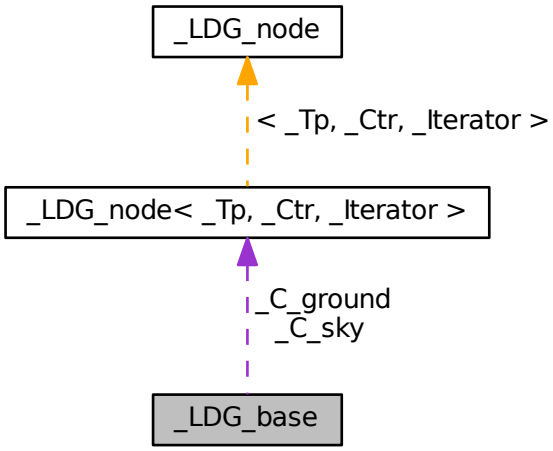
9.16 _LDG_base Class Reference

Labelled directed graph base class for allocator encapsulation.

`#include <vgtl_ldagbase.h>`

Inheritance diagram for _LDG_base:

Collaboration diagram for _LDG_base:



Public Types

- typedef _NAlloc node_allocator_type
- typedef _EAlloc edge_allocator_type
- typedef _Ctr container_type
- typedef _Iterator out_iterator
- typedef _CIterator out_const_iterator
- typedef _Iterator in_iterator
- typedef _CIterator in_const_iterator

Public Member Functions

- node_allocator_type get_node_allocator () const
- edge_allocator_type get_edge_allocator () const
- _LDG_base (const node_allocator_type &, const edge_allocator_type &)
- ~_LDG_base ()
- void clear ()

Protected Member Functions

- void clear_graph (_LDG_node< _Tp, _Ctr, _Iterator > *_node)
- _LDG_node< _Tp, _Ctr, _Iterator > * _C_get_node ()
- void _C_put_node (_LDG_node< _Tp, _Ctr, _Iterator > *__p)
- _LDG_edge< _Te, _Node > * _C_get_edge ()
- void _C_put_edge (_LDG_edge< _Te, _Node > *__p)
- void clear_out_edges ()
- void clear_in_edges ()
- template<class _Output_Iterator >
  void add_all_out_edges (_Output_Iterator fi, _LDG_node< _Tp, _Ctr, _Iterator > *_parent)
- template<class _Output_Iterator >
  void add_all_in_edges (_Output_Iterator fi, _LDG_node< _Tp, _Ctr, _Iterator > *_child)

Protected Attributes

- _LDG_node< _Tp, _Ctr, _Iterator > * _C_ground
- _LDG_node< _Tp, _Ctr, _Iterator > * _C_sky
- int _C_mark

9.16.1 Detailed Description

Base directed graph class top level that encapsulates details of allocators. This class is same as _LDG_base and _LDG_alloc_base if STL doesn't support standard allocators.

9.16.2 Member Typedef Documentation

9.16.2.1 typedef _Ctr _LDG_base::container_type

internal container used to store the edges

Reimplemented in __LDG.

Definition at line 468 of file vgtl_ldagbase.h.

9.16.2.2 typedef _EAlloc _LDG_base::edge_allocator_type

edge allocator type

Reimplemented in __LDG.

Definition at line 461 of file vgtl_ldagbase.h.

9.16.2.3 typedef _CIterator _LDG_base::in_const_iterator

const iterator for accessing the out edges

Reimplemented in __LDG.

Definition at line 476 of file vgtl_ldagbase.h.

9.16.2.4 typedef _Iterator _LDG_base::in_iterator

iterator for accessing the in edges

Reimplemented in __LDG.

Definition at line 474 of file vgtl_ldagbase.h.

9.16.2.5 typedef _NAlloc _LDG_base::node_allocator_type

node allocator type

Reimplemented in __LDG.

Definition at line 459 of file vgtl_ldagbase.h.

9.16.2.6 typedef _CIterator _LDG_base::out_const_iterator

const iterator for accessing the out edges

Reimplemented in __LDG.

Definition at line 472 of file vgtl_ldagbase.h.

9.16.2.7 typedef _Iterator _LDG_base::out_iterator

iterator for accessing the out edges

Reimplemented in __LDG.

Definition at line 470 of file vgtl_ldagbase.h.

9.16.3 Constructor & Destructor Documentation

9.16.3.1 _LDG_base::_LDG_base (const node_allocator_type & , const edge_allocator_type &) [inline]

constructor initializing the allocator and the virtual nodes

Definition at line 479 of file vgtl_ldagbase.h.

9.16.3.2 _LDG_base::~_LDG_base ( ) [inline]

standard destructor

Definition at line 505 of file vgtl_ldagbase.h.

9.16.4 Member Function Documentation

9.16.4.1 _LDG_edge<_Te,_Node>* _LDG_base::_C_get_edge ( ) [inline, protected]

allocate a new edge

Definition at line 533 of file vgtl_ldagbase.h.

9.16.4.2 _LDG_node<_Tp,_Ctr,_Iterator>* _LDG_base::_C_get_node ( ) [inline, protected]

allocate a new node

Definition at line 526 of file vgtl_ldagbase.h.

9.16.4.3 void _LDG_base::_C_put_edge (_LDG_edge< _Te, _Node > * __p) [inline, protected]

deallocate a edge

Definition at line 536 of file vgtl_ldagbase.h.

9.16.4.4 void _LDG_base::_C_put_node (_LDG_node< _Tp, _Ctr, _Iterator > * __p) [inline, protected]

deallocate a node

Definition at line 529 of file vgtl_ldagbase.h.

9.16.4.5 template<class _Output_Iterator > void _LDG_base::add_all_in_edges (_Output_Iterator *fi*, _LDG_node< _Tp, _Ctr, _Iterator > * *_child*) [protected]

add all in edges to the child _child. fi is a iterator to the in edges container of the child

9.16.4.6 template<class _Output_Iterator > void _LDG_base::add_all_out_edges (_Output_Iterator *fi*, _LDG_node< _Tp, _Ctr, _Iterator > * *_parent*) [protected]

add all out edges to the parent _parent. fi is a iterator to the out edges container of the parent

9.16.4.7 void _LDG_base::clear ( )

empty the graph

Reimplemented in __LDG.

9.16.4.8 void _LDG_base::clear_graph (_LDG_node< _Tp, _Ctr, _Iterator > * *_node*) [protected]

removes recursively all nodes and edges downward starting from _node.

9.16.4.9 void _LDG_base::clear_in_edges ( ) [inline, protected]

clear all in edges of the sky node

Definition at line 551 of file vgtl_ldagbase.h.

9.16.4.10 void _LDG_base::clear_out_edges ( ) [inline, protected]

clear all out edges of the ground node

Definition at line 548 of file vgtl_ldagbase.h.

9.16.4.11 edge_allocator_type _LDG_base::get_edge_allocator ( ) const [inline]

get an edge allocator object

Reimplemented in __LDG.

Definition at line 465 of file vgtl_ldagbase.h.

9.16.4.12 node_allocator_type _LDG_base::get_node_allocator ( ) const [inline]

get a node allocator object

Reimplemented in __LDG.

Definition at line 463 of file vgtl_ldagbase.h.

9.16.5 Member Data Documentation

9.16.5.1 _LDG_node<_Tp,_Ctr,_Iterator>* _LDG_base::_C_ground [protected]

the virtual ground node (below all roots)

Definition at line 541 of file vgtl_ldagbase.h.

9.16.5.2 int _LDG_base::_C_mark [protected]

an internal counter for setting marks during certain algorithms

Definition at line 545 of file vgtl_ldagbase.h.

9.16.5.3 _LDG_node<_Tp,_Ctr,_Iterator>* _LDG_base::_C_sky [protected]

the virtual sky node (above all leafs)

Definition at line 543 of file vgtl_ldagbase.h.

The documentation for this class was generated from the following file:

- vgtl_ldagbase.h

9.17 _LDG_edge Class Reference

labelled directed graph edge

`#include <vgtl_ldagbase.h>`

Public Member Functions

- _LDG_edge ()
- ~_LDG_node ()

Public Attributes

- _Te * _E_data
- _TN * _E_snode
- _TN * _E_tnode

9.17.1 Detailed Description

This is the edge for a labelled directed graph

9.17.2 Constructor & Destructor Documentation

9.17.2.1 _LDG_edge::_LDG_edge ( ) [inline]

standard constructor

Definition at line 195 of file vgtl_ldagbase.h.

9.17.2.2 _LDG_edge::~_LDG_node ( ) [inline]

standard destructor

Definition at line 205 of file vgtl_ldagbase.h.

9.17.3 Member Data Documentation

9.17.3.1 _Te* _LDG_edge::_E_data

the edge data

Definition at line 188 of file vgtl_ldagbase.h.

9.17.3.2 _TN* _LDG_edge::_E_snode

the pointer to the source node

Definition at line 190 of file vgtl_ldagbase.h.

9.17.3.3 _TN* _LDG_edge::_E_tnode

the pointer to the target node

Definition at line 192 of file vgtl_ldagbase.h.

The documentation for this class was generated from the following file:

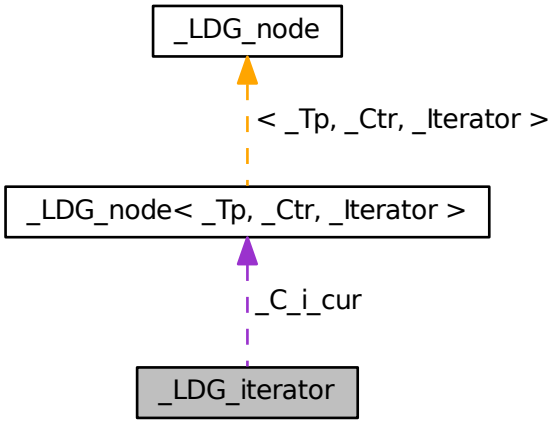
- vgtl_ldagbase.h

9.18 _LDG_iterator Class Reference

iterator through the directed graph

`#include <vgtl_ldag.h>`

Collaboration diagram for _LDG_iterator:



Public Types

- typedef std::bidirectional_iterator_tag iterator_category
- typedef _Tp value_type
- typedef _Ptr pointer
- typedef _Ref reference
- typedef _LDG_node< _Tp, _Ctr, _Iterator > _Node
- typedef size_t size_type
- typedef ptrdiff_t difference_type

Public Member Functions

- _LDG_iterator ()
- _LDG_iterator (const iterator &__x)
- reference operator* () const
- pointer operator-> () const
- _Self & operator= (const _Walk &__x)

- bool operator== (const _Self &__x) const
- bool operator!= (const _Self &__x) const

- _Self & operator++ ()
- _Self operator++ (int)
- _Self & operator-- ()
- _Self operator-- (int)

Protected Attributes

- _Node * _C_i_cur
- std::vector< _Ctr_iterator > _C_i_cur_it

9.18.1 Detailed Description

This is an iterator, which visits each node of a directed graph once. It is based on a preorder depth-first automatic walker which visits a child if and only if the parent is the first in the list.

9.18.2 Member Typedef Documentation

9.18.2.1 typedef _LDG_node<_Tp,_Ctr,_Iterator> _LDG_iterator::_Node

standard iterator definition

Definition at line 333 of file vgtl_ldag.h.

9.18.2.2 typedef ptrdiff_t _LDG_iterator::difference_type

standard iterator definition

Definition at line 335 of file vgtl_ldag.h.

9.18.2.3 typedef std::bidirectional_iterator_tag _LDG_iterator::iterator_category

standard iterator definition

Definition at line 329 of file vgtl_ldag.h.

9.18.2.4 typedef _Ptr _LDG_iterator::pointer

standard iterator definition

Definition at line 331 of file vgtl_ldag.h.

9.18.2.5 typedef _Ref _LDG_iterator::reference

standard iterator definition

Definition at line 332 of file vgtl_ldag.h.

9.18.2.6 typedef size_t _LDG_iterator::size_type

standard iterator definition

Definition at line 334 of file vgtl_ldag.h.

9.18.2.7 typedef _Tp _LDG_iterator::value_type

standard iterator definition

Definition at line 330 of file vgtl_ldag.h.

9.18.3 Constructor & Destructor Documentation

9.18.3.1 _LDG_iterator::_LDG_iterator ( ) [inline]

standard constructor

Definition at line 348 of file vgtl_ldag.h.

9.18.3.2 _LDG_iterator::_LDG_iterator (const iterator & __x) [inline]

copy constructor

Definition at line 350 of file vgtl_ldag.h.

9.18.4 Member Function Documentation

9.18.4.1 bool _LDG_iterator::operator!= (const _Self & __x) const [inline]

comparison operator

Definition at line 363 of file vgtl_ldag.h.

9.18.4.2 reference _LDG_iterator::operator* ( ) const [inline]

dereference operator

Definition at line 373 of file vgtl_ldag.h.

9.18.4.3 _Self& _LDG_iterator::operator++ ( ) [inline]

in(de)crement operator

Definition at line 401 of file vgtl_ldag.h.

9.18.4.4 _Self _LDG_iterator::operator++ (int) [inline]

in(de)crement operator

Definition at line 405 of file vgtl_ldag.h.

9.18.4.5 _Self& _LDG_iterator::operator-- ( ) [inline]

in(de)crement operator

Definition at line 411 of file vgtl_ldag.h.

9.18.4.6 _Self _LDG_iterator::operator-- (int) [inline]

in(de)crement operator

Definition at line 415 of file vgtl_ldag.h.

9.18.4.7 pointer _LDG_iterator::operator->( ) const [inline]

pointer operator

Definition at line 377 of file vgtl_ldag.h.

9.18.4.8 _Self& _LDG_iterator::operator= (const _Walk & __x) [inline]

assignment to iterator from walker

Definition at line 390 of file vgtl_ldag.h.

9.18.4.9 bool _LDG_iterator::operator== (const _Self & __x) const [inline]

comparison operator

Definition at line 355 of file vgtl_ldag.h.

9.18.5 Member Data Documentation

9.18.5.1 _Node* _LDG_iterator::_C_i_cur [protected]

The current node

Definition at line 342 of file vgtl_ldag.h.

9.18.5.2 std::vector<_Ctr_iterator> _LDG_iterator::_C_i_cur_it [protected]

The internal stack

Definition at line 344 of file vgtl_ldag.h.

The documentation for this class was generated from the following file:

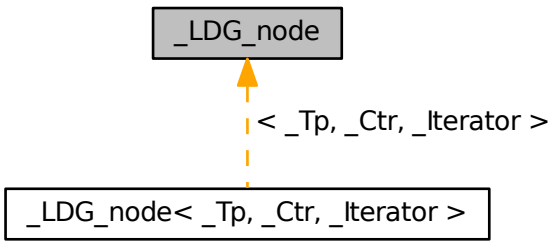
- vgtl_ldag.h

9.19 _LDG_node Class Reference

labelled directed graph node

`#include <vgtl_ldagbase.h>`

Inheritance diagram for _LDG_node:



Public Member Functions

- _LDG_node ()
- ~_LDG_node ()
- void clear_in_edges ()
- void clear_out_edges ()
- _Ctr_iterator get_childentry_iterator (const _Void_pointer __p)
- _Ctr_iterator get_parententry_iterator (const _Void_pointer __p)
- template<class _Output_Iterator >
  void add_all_children (_Output_Iterator fi, _Self *_parent)
- template<class _Output_Iterator >
  void add_all_parents (_Output_Iterator fi, _Self *_child)
- template<class Compare >
  void sort_in_edges (_Ctr_iterator first, _Ctr_iterator last, Compare comp)
- template<class Compare >
  void sort_out_edges (_Ctr_iterator first, _Ctr_iterator last, Compare comp)

Public Attributes

- _Tp * _C_data
- _Ctr _C_inedges
- _Ctr _C_outedges
- int _C_visited

9.19.1 Detailed Description

This is the node for a directed graph

9.19.2 Constructor & Destructor Documentation

9.19.2.1 _LDG_node::_LDG_node ( ) [inline]

standard constructor

Definition at line 63 of file vgtl_ldagbase.h.

9.19.2.2 _LDG_node::~_LDG_node ( ) [inline]

standard destructor

Definition at line 74 of file vgtl_ldagbase.h.

9.19.3 Member Function Documentation

9.19.3.1 template<class _Output_Iterator > void _LDG_node::add_all_children (_Output_Iterator *fi,* _Self * *_parent*)

add all children to parent _parent. fi is an iterator to the children container of _parent

9.19.3.2 template<class _Output_Iterator > void _LDG_node::add_all_parents (_Output_Iterator *fi,* _Self * *_child*)

add all parents to child _child. fi is an iterator to the parents container of _child

9.19.3.3 void _LDG_node::clear_in_edges ( ) [inline]

erase all in edges

Definition at line 84 of file vgtl_ldagbase.h.

9.19.3.4 void _LDG_node::clear_out_edges ( ) [inline]

erase all out edges

Definition at line 87 of file vgtl_ldagbase.h.

9.19.3.5 _Ctr_iterator _LDG_node::get_childentry_iterator (const _Void_pointer *__p*) [inline]

find the iterator into the children container for child __p

Definition at line 91 of file vgtl_ldagbase.h.

9.19.3.6 _Ctr_iterator _LDG_node::get_parententry_iterator (const _Void_pointer *__p*) [inline]

find the iterator into the parents container for parent __p

Definition at line 100 of file vgtl_ldagbase.h.

9.19.3.7 template<class Compare > void _LDG_node::sort_in_edges (_Ctr_iterator *first,* _Ctr_iterator *last,* Compare *comp*) [inline]

sort the children according to comp

Definition at line 127 of file vgtl_ldagbase.h.

9.19.3.8 template<class Compare > void _LDG_node::sort_out_edges (_Ctr_iterator *first,* _Ctr_iterator *last,* Compare *comp*) [inline]

sort the parents according to comp

Definition at line 134 of file vgtl_ldagbase.h.

9.19.4 Member Data Documentation

9.19.4.1 _Tp* _LDG_node::_C_data

the node data

Definition at line 54 of file vgtl_ldagbase.h.

9.19.4.2 _Ctr _LDG_node::_C_inedges

the edges to the parents

Definition at line 56 of file vgtl_ldagbase.h.

9.19.4.3 _Ctr _LDG_node::_C_outedges

the edges to the children

Definition at line 58 of file vgtl_ldagbase.h.

9.19.4.4 int _LDG_node::_C_visited

internal counter for marks in algorithms

Definition at line 60 of file vgtl_ldagbase.h.

The documentation for this class was generated from the following file:

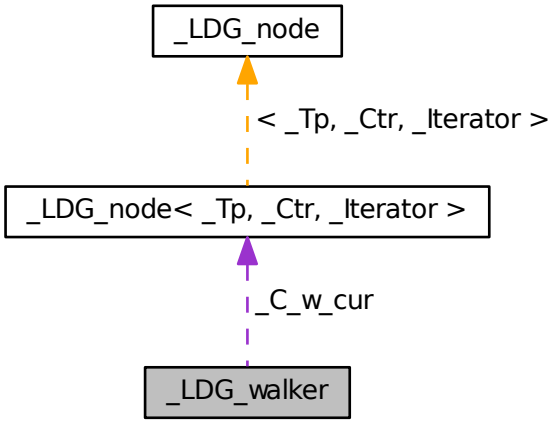
- vgtl_ldagbase.h

9.20 _LDG_walker Class Reference

recursive labelled directed graph walkers

```
#include <vgtl_ldag.h>
```

Collaboration diagram for _LDG_walker:



Public Types

- typedef _Tp value_type
- typedef _Ptr pointer
- typedef _Ref reference
- typedef _Te edge_value_type

- typedef _Ctr_iterator out_edge_iterator
- typedef _Ctr_iterator in_edge_iterator
- typedef _Ctr_const_iterator out_edge_const_iterator
- typedef _Ctr_const_iterator in_edge_const_iterator
- typedef _Node node_type
- typedef _Edge edge_type
- typedef out_edge_iterator children_iterator
- typedef in_edge_iterator parents_iterator
- typedef out_edge_const_iterator children_const_iterator
- typedef in_edge_const_iterator parents_const_iterator
- typedef size_t size_type
- typedef ptrdiff_t difference_type

Public Member Functions

- _LDG_walker ()
- _LDG_walker (_Node *__x)

- _LDG_walker (const walker &__x)
- reference operator* () const
- pointer operator-> () const
- const _Node * node ()
- size_type out_degree () const
- size_type in_degree () const
- size_type n_children () const
- size_type n_children () const
- bool is_source () const
- bool is_root () const
- bool is_sink () const
- bool is_leaf () const
- bool is_ground () const
- bool is_sky () const
- template<class _Function >
  _Function for_each_child (_Function __f)
- template<class _Function >
  _Function for_each_parent (_Function __f)
- _Self operator<< (in_iterator __i)
- _Self operator>> (out_iterator __i)
- _Self & operator<<= (in_iterator __i)
- _Self & operator>>= (out_iterator __i)
- _Self operator<< (in_const_iterator __i)
- _Self operator>> (out_const_iterator __i)
- _Self & operator<<= (parents_const_iterator __i)
- _Self & operator>>= (children_const_iterator __i)
- _Self & operator= (const _Itr &__x)
- _Self & operator= (const _Self &__x)
- _Self & operator= (const _Node &__n)

- out_iterator out_begin ()
- out_const_iterator out_begin () const
- out_iterator child_begin ()
- out_const_iterator child_begin () const

- out_iterator out_end ()
- out_const_iterator out_end () const
- out_iterator child_end ()
- out_const_iterator child_end () const

- in_iterator in_begin ()
- in_const_iterator in_begin () const
- in_iterator parent_begin ()
- in_const_iterator parent_begin () const

- in_iterator in_end ()
- in_const_iterator in_end () const
- in_iterator in_end ()
- in_const_iterator in_end () const

- bool operator== (const _Self &__x) const
- bool operator!= (const _Self &__x) const

Public Attributes

- _Node * _C_w_cur

9.20.1 Detailed Description

This is the class defining recursive labelled directed graph walkers, which walk labelled directed graphs under guidance.

9.20.2 Member Typedef Documentation

9.20.2.1 typedef out_edge_const_iterator _LDG_walker::children_const_iterator

standard walker definition

Definition at line 100 of file vgtl_ldag.h.

9.20.2.2 typedef out_edge_iterator _LDG_walker::children_iterator

standard walker definition

Definition at line 98 of file vgtl_ldag.h.

9.20.2.3 typedef ptrdiff_t _LDG_walker::difference_type

standard walker definition

Definition at line 104 of file vgtl_ldag.h.

9.20.2.4 typedef _Edge _LDG_walker::edge_type

standard walker definition

Definition at line 96 of file vgtl_ldag.h.

9.20.2.5 typedef _Te _LDG_walker::edge_value_type

standard walker definition

Definition at line 79 of file vgtl_ldag.h.

9.20.2.6 typedef _Ctr_const_iterator _LDG_walker::in_edge_const_iterator

standard walker definition

Definition at line 94 of file vgtl_ldag.h.

9.20.2.7 typedef _Ctr_iterator _LDG_walker::in_edge_iterator

standard walker definition

Definition at line 92 of file vgtl_ldag.h.

9.20.2.8 typedef _Node _LDG_walker::node_type

standard walker definition

Definition at line 95 of file vgtl_ldag.h.

9.20.2.9 typedef _Ctr_const_iterator _LDG_walker::out_edge_const_iterator

standard walker definition

Definition at line 93 of file vgtl_ldag.h.

9.20.2.10 typedef _Ctr_iterator _LDG_walker::out_edge_iterator

standard walker definition

Definition at line 91 of file vgtl_ldag.h.

9.20.2.11 typedef in_edge_const_iterator _LDG_walker::parents_const_iterator

standard walker definition

Definition at line 101 of file vgtl_ldag.h.

9.20.2.12 typedef in_edge_iterator _LDG_walker::parents_iterator

standard walker definition

Definition at line 99 of file vgtl_ldag.h.

9.20.2.13 typedef _Ptr _LDG_walker::pointer

standard walker definition

Definition at line 77 of file vgtl_ldag.h.

9.20.2.14 typedef _Ref _LDG_walker::reference

standard walker definition

Definition at line 78 of file vgtl_ldag.h.

9.20.2.15 typedef size_t _LDG_walker::size_type

standard walker definition

Definition at line 103 of file vgtl_ldag.h.

9.20.2.16 typedef _Tp _LDG_walker::value_type

standard walker definition

Definition at line 76 of file vgtl_ldag.h.

9.20.3 Constructor & Destructor Documentation

9.20.3.1 _LDG_walker::_LDG_walker ( ) [inline]

standard constructor

Definition at line 113 of file vgtl_ldag.h.

9.20.3.2 _LDG_walker::_LDG_walker (_Node * __x) [inline]

constructor setting the position

Definition at line 117 of file vgtl_ldag.h.

9.20.3.3 _LDG_walker::_LDG_walker (const walker & __x) [inline]

copy constructor

Definition at line 120 of file vgtl_ldag.h.

9.20.4 Member Function Documentation

9.20.4.1 out_iterator _LDG_walker::child_begin ( ) [inline]

return out_iterator to first child

Definition at line 179 of file vgtl_ldag.h.

9.20.4.2 out_const_iterator _LDG_walker::child_begin ( ) const [inline]

return out_iterator to first child

Definition at line 180 of file vgtl_ldag.h.

9.20.4.3 out_iterator _LDG_walker::child_end ( ) [inline]

return out_iterator beyond last child

Definition at line 187 of file vgtl_ldag.h.

9.20.4.4 out_const_iterator _LDG_walker::child_end ( ) const [inline]

return out_iterator beyond last child

Definition at line 188 of file vgtl_ldag.h.

9.20.4.5 template<class _Function > _Function _LDG_walker::for_each_child (_Function __f) [inline]

apply the function __f to all children

Definition at line 210 of file vgtl_ldag.h.

9.20.4.6 template<class _Function > _Function _LDG_walker::for_each_parent (_Function __f) [inline]

apply the function __f to all parents

Definition at line 216 of file vgtl_ldag.h.

9.20.4.7 in_iterator _LDG_walker::in_begin ( ) [inline]

return in_iterator to first parent

Definition at line 193 of file vgtl_ldag.h.

9.20.4.8 in_const_iterator _LDG_walker::in_begin ( ) const [inline]

return in_iterator to first parent

Definition at line 194 of file vgtl_ldag.h.

9.20.4.9 size_type _LDG_walker::in_degree ( ) const [inline]

return the in degree

Definition at line 137 of file vgtl_ldag.h.

9.20.4.10 in_iterator _LDG_walker::in_end ( ) [inline]

return in_iterator beyond last parent

Definition at line 201 of file vgtl_ldag.h.

9.20.4.11 in_const_iterator _LDG_walker::in_end ( ) const [inline]

return in_iterator beyond last parent

Definition at line 202 of file vgtl_ldag.h.

9.20.4.12 in_iterator _LDG_walker::in_end ( ) [inline]

return in_iterator beyond last parent

Definition at line 204 of file vgtl_ldag.h.

9.20.4.13 in_const_iterator _LDG_walker::in_end ( ) const [inline]

return in_iterator beyond last parent

Definition at line 205 of file vgtl_ldag.h.

9.20.4.14 bool _LDG_walker::is_ground ( ) const [inline]

is this node a virtual node - the ground (below all roots)?

Definition at line 170 of file vgtl_ldag.h.

9.20.4.15 bool _LDG_walker::is_leaf ( ) const [inline]

is this node a leaf?

Definition at line 167 of file vgtl_ldag.h.

9.20.4.16 bool _LDG_walker::is_root ( ) const [inline]

is this node a root?

Definition at line 155 of file vgtl_ldag.h.

9.20.4.17 bool _LDG_walker::is_sink ( ) const [inline]

is this node a local sink?

Definition at line 157 of file vgtl_ldag.h.

9.20.4.18 bool _LDG_walker::is_sky ( ) const [inline]

is this node a virtual node - the sky (above all leafs)?

Definition at line 172 of file vgtl_ldag.h.

9.20.4.19 bool _LDG_walker::is_source ( ) const [inline]

is this node a local source?

Definition at line 145 of file vgtl_ldag.h.

9.20.4.20 size_type _LDG_walker::n_children ( ) const [inline]

return the number of children (the out degree)

Definition at line 140 of file vgtl_ldag.h.

9.20.4.21 size_type _LDG_walker::n_children ( ) const [inline]

return the number of parents (the in degree)

Definition at line 142 of file vgtl_ldag.h.

9.20.4.22 const _Node* _LDG_walker::node ( ) [inline]

retrieve the full node

Definition at line 132 of file vgtl_ldag.h.

9.20.4.23 bool _LDG_walker::operator!= (const _Self & __x) const [inline]

comparison operator

Definition at line 226 of file vgtl_ldag.h.

9.20.4.24 reference _LDG_walker::operator* ( ) const [inline]

dereference operator

Definition at line 123 of file vgtl_ldag.h.

9.20.4.25 pointer _LDG_walker::operator-> ( ) const [inline]

pointer operator

Definition at line 127 of file vgtl_ldag.h.

9.20.4.26 _Self _LDG_walker::operator<< (in_iterator __i) [inline]

this function returns the walker pointing to the required parent

Definition at line 231 of file vgtl_ldag.h.

9.20.4.27 _Self _LDG_walker::operator<< (in_const_iterator __i) [inline]

this function returns the walker pointing to the required parent

Definition at line 261 of file vgtl_ldag.h.

9.20.4.28 _Self& _LDG_walker::operator<<= (in_iterator __i) [inline]

here the original walker goes to the required parent

Definition at line 247 of file vgtl_ldag.h.

9.20.4.29 _Self& _LDG_walker::operator<<= (parents_const_iterator __i) [inline]

here the original walker goes to the required parent

Definition at line 277 of file vgtl_ldag.h.

9.20.4.30 _Self& _LDG_walker::operator= (const _Itr & __x) [inline]

new walker is assigned from that particular iterator

Definition at line 291 of file vgtl_ldag.h.

9.20.4.31 _Self& _LDG_walker::operator= (const _Self & __x) [inline]

standard assignment operator

Definition at line 297 of file vgtl_ldag.h.

9.20.4.32 _Self& _LDG_walker::operator= (const _Node & __n) [inline]

a walker is assigned to any pointer to a graph node

Definition at line 303 of file vgtl_ldag.h.

9.20.4.33 bool _LDG_walker::operator== (const _Self & __x) const [inline]

comparison operator

Definition at line 224 of file vgtl_ldag.h.

9.20.4.34 _Self _LDG_walker::operator>> (out_iterator __i) [inline]

this function returns the walker pointing to the required child

Definition at line 239 of file vgtl_ldag.h.

9.20.4.35 _Self _LDG_walker::operator>> (out_const_iterator __i) [inline]

this function returns the walker pointing to the required child

Definition at line 269 of file vgtl_ldag.h.

9.20.4.36 _Self& _LDG_walker::operator>>= (out_iterator __i) [inline]

here the original walker goes to the required child

Definition at line 254 of file vgtl_ldag.h.

9.20.4.37 _Self& _LDG_walker::operator>>= (children_const_iterator __i) [inline]

here the original walker goes to the required child

Definition at line 284 of file vgtl_ldag.h.

9.20.4.38 out_iterator _LDG_walker::out_begin ( ) [inline]

return out_iterator to first child

Definition at line 176 of file vgtl_ldag.h.

9.20.4.39 out_const_iterator _LDG_walker::out_begin ( ) const [inline]

return out_iterator to first child

Definition at line 177 of file vgtl_ldag.h.

9.20.4.40 size_type _LDG_walker::out_degree ( ) const [inline]

return the out degree

Definition at line 135 of file vgtl_ldag.h.

9.20.4.41 out_iterator _LDG_walker::out_end ( ) [inline]

return out_iterator beyond last child

Definition at line 184 of file vgtl_ldag.h.

9.20.4.42 out_const_iterator _LDG_walker::out_end ( ) const [inline]

return out_iterator beyond last child

Definition at line 185 of file vgtl_ldag.h.

9.20.4.43 in_iterator _LDG_walker::parent_begin ( ) [inline]

return in_iterator to first parent

Definition at line 196 of file vgtl_ldag.h.

9.20.4.44 in_const_iterator _LDG_walker::parent_begin ( ) const [inline]

return in_iterator to first parent

Definition at line 197 of file vgtl_ldag.h.

9.20.5 Member Data Documentation

9.20.5.1 _Node* _LDG_walker::_C_w_cur

pointer to the current node

Definition at line 109 of file vgtl_ldag.h.

The documentation for this class was generated from the following file:

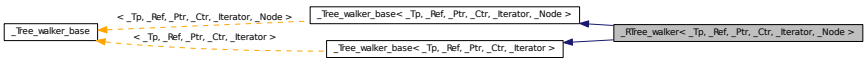
- vgtl_ldag.h

9.21 _RTree_walker< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node > Class Template Reference

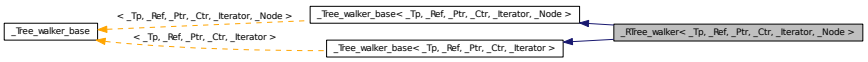
recursive tree walkers

`#include <vgtl_tree.h>`

Inheritance diagram for _RTree_walker< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >:



Collaboration diagram for _RTree_walker< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >:



Public Types

- typedef _Tp value_type
- typedef _Ptr pointer
- typedef _Ref reference
- typedef _Tp value_type
- typedef _Ptr pointer
- typedef _Ref reference

- typedef __one_iterator< void * > parents_iterator
- typedef _Ctr_iterator children_iterator
- typedef _Node node_type
- typedef size_t size_type
- typedef ptrdiff_t difference_type
- typedef __one_iterator< void * > parents_iterator
- typedef _Ctr_iterator children_iterator
- typedef _Node node_type
- typedef size_t size_type
- typedef ptrdiff_t difference_type

Public Member Functions

- _RTree_walker ()
- _RTree_walker (_Node *__x)
- _RTree_walker (const walker &__x)
- _Self operator<< (const parents_iterator &__dummy)

  go to parent operator
- _Self operator>> (const children_iterator &__i)

  go to child operator
- _Self & operator<<= (const parents_iterator &__dummy)
- _Self & operator>>= (const children_iterator &__i)
- _Self & operator= (const _Itr &__x)
- _Self & operator= (const _Tree_walker< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node > &__x)
- reference operator* () const
- pointer operator-> () const
- ctree_data_hook & data_hook ()
- ctree_data_hook & parent_data_hook ()
- const _Node * parent ()
- const _Node * node ()
- size_type n_children ()
- size_type n_parents ()
- bool is_leaf ()
- bool is_root ()
- bool is_ground ()
- bool is_sky ()
- children_iterator child_begin ()
- children_iterator child_end ()
- parents_iterator parent_begin ()
- parents_iterator parent_end ()
- _Function for_each_child (_Function __f)
- _Function for_each_parent (_Function __f)
- void sort_children (children_iterator first, children_iterator last, Compare comp)
- void sort_children (Compare comp)
- void sort_parents (parents_iterator first, parents_iterator last, Compare comp)
- void sort_parents (Compare comp)
- reference operator* () const
- pointer operator-> () const
- ctree_data_hook & data_hook ()
- ctree_data_hook & parent_data_hook ()
- const _Node * parent ()
- const _Node * node ()
- size_type n_children ()
- size_type n_parents ()
- bool is_leaf ()
- bool is_root ()
- bool is_ground ()
- bool is_sky ()
- children_iterator child_begin ()
- children_iterator child_end ()
- parents_iterator parent_begin ()

- parents_iterator parent_end ()
- _Function for_each_child (_Function __f)
- _Function for_each_parent (_Function __f)
- void sort_children (children_iterator first, children_iterator last, Compare comp)
- void sort_children (Compare comp)
- void sort_parents (parents_iterator first, parents_iterator last, Compare comp)
- void sort_parents (Compare comp)

- bool operator== (const _Self &__x) const
- bool operator!= (const _Self &__x) const

Public Attributes

- _Node * _C_w_cur
- _Node * _C_w_cur

9.21.1 Detailed Description

template<class _Tp, class _Ref, class _Ptr, class _Ctr, class _Iterator, class _Node>class _RTree_walker< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >

This is the class defining reursive tree walkers, which walk trees under guidance.

Definition at line 1061 of file vgtl_tree.h.

9.21.2 Member Typedef Documentation

9.21.2.1 typedef _Ctr_iterator _Tree_walker_base::children_iterator [inherited]

standard walker definition

Definition at line 243 of file vgtl_tree.h.

9.21.2.2 typedef _Ctr_iterator _Tree_walker_base::children_iterator [inherited]

standard walker definition

Definition at line 243 of file vgtl_tree.h.

9.21.2.3 typedef ptrdiff_t _Tree_walker_base::difference_type [inherited]

standard walker definition

Definition at line 247 of file vgtl_tree.h.

9.21.2.4 typedef ptrdiff_t _Tree_walker_base::difference_type [inherited]

standard walker definition

Definition at line 247 of file vgtl_tree.h.

9.21.2.5 typedef _Node _Tree_walker_base::node_type [inherited]

standard walker definition

Definition at line 244 of file vgtl_tree.h.

9.21.2.6 typedef _Node _Tree_walker_base::node_type [inherited]

standard walker definition

Definition at line 244 of file vgtl_tree.h.

9.21.2.7 typedef __one_iterator<void *> _Tree_walker_base::parents_iterator [inherited]

standard walker definition

Definition at line 242 of file vgtl_tree.h.

9.21.2.8 typedef __one_iterator<void *> _Tree_walker_base::parents_iterator [inherited]

standard walker definition

Definition at line 242 of file vgtl_tree.h.

9.21.2.9 typedef _Ptr _Tree_walker_base::pointer [inherited]

standard walker definition

Definition at line 233 of file vgtl_tree.h.

9.21.2.10 typedef _Ptr _Tree_walker_base::pointer [inherited]

standard walker definition

Definition at line 233 of file vgtl_tree.h.

9.21.2.11 typedef _Ref _Tree_walker_base::reference [inherited]

standard walker definition

Definition at line 234 of file vgtl_tree.h.

9.21.2.12 typedef _Ref _Tree_walker_base::reference [inherited]

standard walker definition

Definition at line 234 of file vgtl_tree.h.

9.21.2.13 typedef size_t _Tree_walker_base::size_type [inherited]

standard walker definition

Definition at line 246 of file vgtl_tree.h.

9.21.2.14 typedef size_t _Tree_walker_base::size_type [inherited]

standard walker definition

Definition at line 246 of file vgtl_tree.h.

9.21.2.15 typedef _Tp _Tree_walker_base::value_type [inherited]

standard walker definition

Definition at line 232 of file vgtl_tree.h.

9.21.2.16 typedef _Tp _Tree_walker_base::value_type [inherited]

standard walker definition

Definition at line 232 of file vgtl_tree.h.

9.21.3 Constructor & Destructor Documentation

9.21.3.1 template<class _Tp , class _Ref , class _Ptr , class _Ctr , class _Iterator , class _Node > _RTree_walker< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >::_RTree_walker ( ) [inline]

standard constructor

Definition at line 1070 of file vgtl_tree.h.

9.21.3.2 template<class _Tp , class _Ref , class _Ptr , class _Ctr , class _Iterator , class _Node > _RTree_walker< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >::_RTree_walker (_Node * __x) [inline]

constructor setting the position

Definition at line 1073 of file vgtl_tree.h.

9.21.3.3 template<class _Tp , class _Ref , class _Ptr , class _Ctr , class _Iterator , class _Node > _RTree_walker< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >::_RTree_walker (const walker & __x) [inline]

copy constructor

Definition at line 1076 of file vgtl_tree.h.

9.21.4 Member Function Documentation

9.21.4.1 children_iterator _Tree_walker_base::child_begin ( ) [inline, inherited]

return children_iterator to first child

Definition at line 307 of file vgtl_tree.h.

9.21.4.2 children_iterator _Tree_walker_base::child_begin ( ) [inline, inherited]

return children_iterator to first child

Definition at line 307 of file vgtl_tree.h.

9.21.4.3 children_iterator _Tree_walker_base::child_end ( ) [inline, inherited]

return children_iterator beyond last child

Definition at line 309 of file vgtl_tree.h.

9.21.4.4 children_iterator _Tree_walker_base::child_end ( ) [inline, inherited]

return children_iterator beyond last child

Definition at line 309 of file vgtl_tree.h.

9.21.4.5 ctree_data_hook& _Tree_walker_base::data_hook ( ) [inline, inherited]

retrieve the data hook

Definition at line 280 of file vgtl_tree.h.

9.21.4.6 ctree_data_hook& _Tree_walker_base::data_hook ( ) [inline, inherited]

retrieve the data hook

Definition at line 280 of file vgtl_tree.h.

9.21.4.7 _Function _Tree_walker_base::for_each_child (_Function __f) [inline, inherited]

apply the function __f to all children

Definition at line 320 of file vgtl_tree.h.

9.21.4.8 _Function _Tree_walker_base::for_each_child (_Function __f) [inline, inherited]

apply the function __f to all children

Definition at line 320 of file vgtl_tree.h.

9.21.4.9 _Function _Tree_walker_base::for_each_parent (_Function __f) [inline, inherited]

apply the function __f to all parents

Definition at line 326 of file vgtl_tree.h.

9.21.4.10 _Function _Tree_walker_base::for_each_parent (_Function __f) [inline, inherited]

apply the function __f to all parents

Definition at line 326 of file vgtl_tree.h.

9.21.4.11 bool _Tree_walker_base::is_ground ( ) [inline, inherited]

is this node a virtual node - the ground (below all roots)?

Definition at line 302 of file vgtl_tree.h.

9.21.4.12 bool _Tree_walker_base::is_ground ( ) [inline, inherited]

is this node a virtual node - the ground (below all roots)?

Definition at line 302 of file vgtl_tree.h.

9.21.4.13 bool _Tree_walker_base::is_leaf ( ) [inline, inherited]

is this node a leaf?

Definition at line 296 of file vgtl_tree.h.

9.21.4.14 bool _Tree_walker_base::is_leaf ( ) [inline, inherited]

is this node a leaf?

Definition at line 296 of file vgtl_tree.h.

9.21.4.15 bool _Tree_walker_base::is_root ( ) [inline, inherited]

is this node a root?

Definition at line 298 of file vgtl_tree.h.

9.21.4.16 bool _Tree_walker_base::is_root ( ) [inline, inherited]

is this node a root?

Definition at line 298 of file vgtl_tree.h.

9.21.4.17 bool _Tree_walker_base::is_sky ( ) [inline, inherited]

is this node a virtual node - the sky (above all leafs)?

Definition at line 304 of file vgtl_tree.h.

9.21.4.18 bool _Tree_walker_base::is_sky ( ) [inline, inherited]

is this node a virtual node - the sky (above all leafs)?

Definition at line 304 of file vgtl_tree.h.

9.21.4.19 size_type _Tree_walker_base::n_children ( ) [inline, inherited]

return the number of children

Definition at line 291 of file vgtl_tree.h.

9.21.4.20 size_type _Tree_walker_base::n_children ( ) [inline, inherited]

return the number of children

Definition at line 291 of file vgtl_tree.h.

9.21.4.21 size_type _Tree_walker_base::n_parents ( ) [inline, inherited]

return the number of parents (0 or 1)

Definition at line 293 of file vgtl_tree.h.

9.21.4.22 size_type _Tree_walker_base::n_parents ( ) [inline, inherited]

return the number of parents (0 or 1)

Definition at line 293 of file vgtl_tree.h.

9.21.4.23 const _Node* _Tree_walker_base::node ( ) [inline, inherited]

retrieve the full node

Definition at line 288 of file vgtl_tree.h.

9.21.4.24 const _Node* _Tree_walker_base::node ( ) [inline, inherited]

retrieve the full node

Definition at line 288 of file vgtl_tree.h.

9.21.4.25 template<class _Tp , class _Ref , class _Ptr , class _Ctr , class _Iterator , class _Node > bool _RTree_walker< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >::operator!= (const _Self & __x) const [inline]

comparison operator

Definition at line 1083 of file vgtl_tree.h.

9.21.4.26 reference _Tree_walker_base::operator* ( ) const [inline, inherited]

dereference operator

Definition at line 265 of file vgtl_tree.h.

9.21.4.27 reference _Tree_walker_base::operator* ( ) const [inline, inherited]

dereference operator

Definition at line 265 of file vgtl_tree.h.

9.21.4.28 pointer _Tree_walker_base::operator-> ( ) const [inline, inherited]

pointer operator

Definition at line 269 of file vgtl_tree.h.

9.21.4.29 pointer _Tree_walker_base::operator-> ( ) const [inline, inherited]

pointer operator

Definition at line 269 of file vgtl_tree.h.

9.21.4.30 template<class _Tp , class _Ref , class _Ptr , class _Ctr , class _Iterator , class _Node > _Self _RTree_walker< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >::operator<< (const parents_iterator & __dummy) [inline]

This operator moves the walker to the parent

Definition at line 1089 of file vgtl_tree.h.

9.21.4.31 template<class _Tp , class _Ref , class _Ptr , class _Ctr , class _Iterator , class _Node > _Self& _RTree_walker< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >::operator<<= (const parents_iterator & __dummy) [inline]

go to parent assignment operator

Definition at line 1106 of file vgtl_tree.h.

9.21.4.32 template<class _Tp , class _Ref , class _Ptr , class _Ctr , class _Iterator , class _Node > _Self& _RTree_walker< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >::operator= (const _Itr & __x) [inline]

assignment from iterator

Reimplemented from _Tree_walker_base< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >.

Definition at line 1120 of file vgtl_tree.h.

9.21.4.33 template<class _Tp , class _Ref , class _Ptr , class _Ctr , class _Iterator , class _Node > _Self& _RTree_walker< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >::operator= (const _Tree_walker< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node > & __x) [inline]

assignment from automatic iterator

Definition at line 1126 of file vgtl_tree.h.

9.21.4.34 template<class _Tp , class _Ref , class _Ptr , class _Ctr , class _Iterator , class _Node > bool _RTree_walker< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >::operator== (const _Self & __x) const [inline]

comparison operator

Definition at line 1081 of file vgtl_tree.h.

9.21.4.35 template<class _Tp , class _Ref , class _Ptr , class _Ctr , class _Iterator , class _Node > _Self _RTree_walker< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >::operator>> (const children_iterator & __i) [inline]

This operator moves the walker to the child pointed to by ___i

Definition at line 1099 of file vgtl_tree.h.

9.21.4.36 template<class _Tp , class _Ref , class _Ptr , class _Ctr , class _Iterator , class _Node > _Self& _RTree_walker< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >::operator>>= (const children_iterator & __i) [inline]

go to child assignment operator

Definition at line 1114 of file vgtl_tree.h.

9.21.4.37 const _Node* _Tree_walker_base::parent ( ) [inline, inherited]

retrieve the parent node

Definition at line 286 of file vgtl_tree.h.

9.21.4.38 const _Node* _Tree_walker_base::parent ( ) [inline, inherited]

retrieve the parent node

Definition at line 286 of file vgtl_tree.h.

9.21.4.39 parents_iterator _Tree_walker_base::parent_begin ( ) [inline, inherited]

return parents_iterator to first parent (the parent)

Definition at line 312 of file vgtl_tree.h.

9.21.4.40 parents_iterator _Tree_walker_base::parent_begin ( ) [inline, inherited]

return parents_iterator to first parent (the parent)

Definition at line 312 of file vgtl_tree.h.

9.21.4.41 ctree_data_hook& _Tree_walker_base::parent_data_hook ( ) [inline, inherited]

retrieve the parent's data hook

Definition at line 282 of file vgtl_tree.h.

9.21.4.42 ctree_data_hook& _Tree_walker_base::parent_data_hook ( ) [inline, inherited]

retrieve the parent's data hook

Definition at line 282 of file vgtl_tree.h.

9.21.4.43 parents_iterator _Tree_walker_base::parent_end ( ) [inline, inherited]

return parents_iterator beyond last parent

Definition at line 315 of file vgtl_tree.h.

9.21.4.44 parents_iterator _Tree_walker_base::parent_end ( ) [inline, inherited]

return parents_iterator beyond last parent

Definition at line 315 of file vgtl_tree.h.

9.21.4.45 void _Tree_walker_base::sort_children (children_iterator *first*, children_iterator *last*, Compare *comp*) [inline, inherited]

sort the children in the range [first,last) according to comp

Definition at line 333 of file vgtl_tree.h.

9.21.4.46 void _Tree_walker_base::sort_children (children_iterator *first*, children_iterator *last*, Compare *comp*) [inline, inherited]

sort the children in the range [first,last) according to comp

Definition at line 333 of file vgtl_tree.h.

9.21.4.47 void _Tree_walker_base::sort_children (Compare *comp*) [inline, inherited]

sort all children according to comp

Definition at line 344 of file vgtl_tree.h.

9.21.4.48 void _Tree_walker_base::sort_children (Compare *comp*) [inline, inherited]

sort all children according to comp

Definition at line 344 of file vgtl_tree.h.

9.21.4.49 void _Tree_walker_base::sort_parents (parents_iterator *first*, parents_iterator *last*, Compare *comp*) [inline, inherited]

sort the parents in the range [first,last) according to comp (NOP)

Definition at line 339 of file vgtl_tree.h.

9.21.4.50 void _Tree_walker_base::sort_parents (parents_iterator *first*, parents_iterator *last*, Compare *comp*) [inline, inherited]

sort the parents in the range [first,last) according to comp (NOP)

Definition at line 339 of file vgtl_tree.h.

9.21.4.51 void _Tree_walker_base::sort_parents (Compare *comp*) [inline, inherited]

sort all parents according to `comp` (NOP = do nothing)

Definition at line 349 of file vgtl_tree.h.

9.21.4.52 void _Tree_walker_base::sort_parents (Compare *comp*) [inline, inherited]

sort all parents according to `comp` (NOP = do nothing)

Definition at line 349 of file vgtl_tree.h.

9.21.5 Member Data Documentation

9.21.5.1 _Node* _Tree_walker_base::_C_w_cur [inherited]

pointer to the current node

Definition at line 252 of file vgtl_tree.h.

9.21.5.2 _Node* _Tree_walker_base::_C_w_cur [inherited]

pointer to the current node

Definition at line 252 of file vgtl_tree.h.

The documentation for this class was generated from the following files:

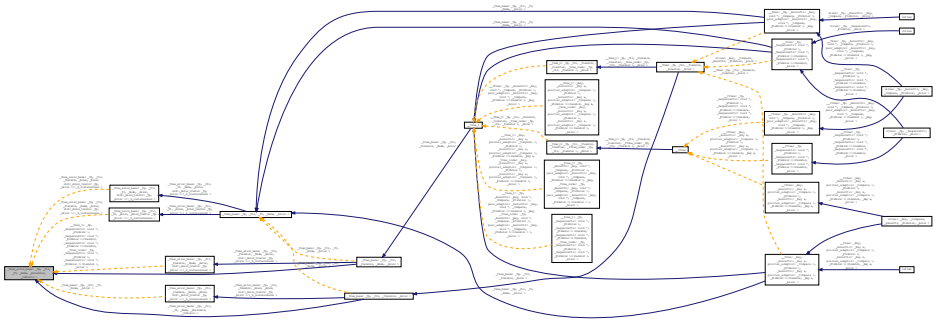
- vgtl_graph.h
- vgtl_tree.h

9.22 _Tree_alloc_base< _Tp, _Ctr, _TI, _Node, _Allocator, _IsStatic > Class Template Reference

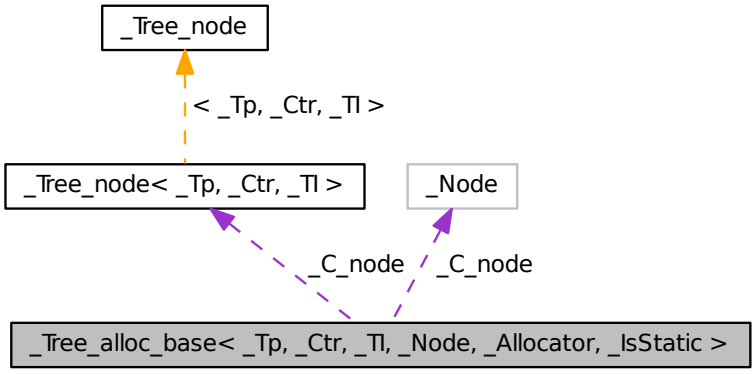
Tree base class for general standard-conforming allocators.

`#include <vgtl_tree.h>`

Inheritance diagram for _Tree_alloc_base< _Tp, _Ctr, _TI, _Node, _Allocator, _IsStatic >:



Collaboration diagram for _Tree_alloc_base< _Tp, _Ctr, _TI, _Node, _Allocator, _IsStatic >:



Protected Member Functions

- _Node * _C_get_node ()
- void _C_put_node (_Node *__p)

Protected Attributes

- _Node * _C_node

9.22.1 Detailed Description

template<class _Tp, class _Ctr, class _TI, class _Node, class _Allocator, bool _IsStatic>class _Tree_alloc_base< _Tp, _Ctr, _TI, _Node, _Allocator, _IsStatic >

Base tree class that encapsulates details of allocators. Three cases: an ordinary standard-conforming allocator, a standard-conforming allocator with no non-static data, and an SGI-style allocator. This complexity is necessary only because we're worrying about STL compatibility and because we want to avoid wasting storage on an allocator instance if it isn't necessary. Base for general standard-conforming allocators.

Definition at line 1365 of file vgtl_tree.h.

9.22.2 Member Function Documentation

9.22.2.1 template<class _Tp, class _Ctr, class _TI, class _Node, class _Allocator, bool _IsStatic> _Node* _Tree_alloc_base< _Tp, _Ctr, _TI, _Node, _Allocator, _IsStatic >::_C_get_node ( ) [inline, protected]

allocate a new node

Definition at line 1375 of file vgtl_tree.h.

9.22.2.2 template<class _Tp, class _Ctr, class _TI, class _Node, class _Allocator, bool _IsStatic> void _Tree_alloc_base< _Tp, _Ctr, _TI, _Node, _Allocator, _IsStatic >::_C_put_node (_Node * _p) [inline, protected]

deallocate a node

Definition at line 1378 of file vgtl_tree.h.

9.22.3 Member Data Documentation

9.22.3.1 template<class _Tp, class _Ctr, class _TI, class _Node, class _Allocator, bool _IsStatic> _Node* _Tree_alloc_base< _Tp, _Ctr, _TI, _Node, _Allocator, _IsStatic >::_C_node [protected]

This is the node

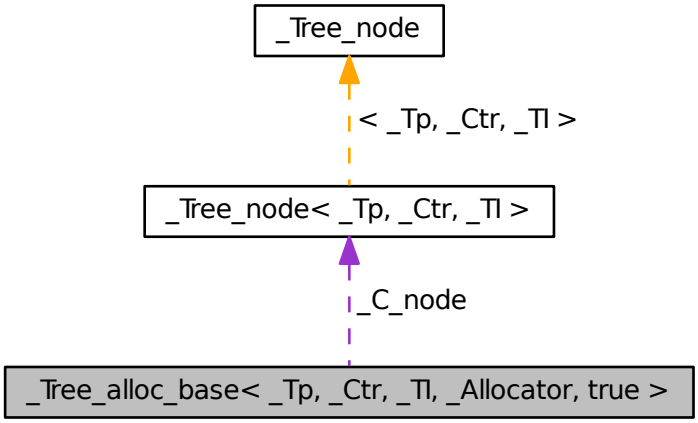
Definition at line 1387 of file vgtl_tree.h.

The documentation for this class was generated from the following files:

- vgtl_graph.h
- vgtl_tree.h

9.23 _Tree_alloc_base< _Tp, _Ctr, _TI, _Allocator, true > Class Reference

Collaboration diagram for _Tree_alloc_base< _Tp, _Ctr, _TI, _Allocator, true >:



The documentation for this class was generated from the following file:

- vgtl_graph.h

9.24 _Tree_alloc_base< _Tp, _Ctr, _TI, _Node, _Allocator, true > Class Reference

Tree base class specialization for instanceless allocators.

`#include <vgtl_tree.h>`

Protected Member Functions

- _Node * _C_get_node ()
- void _C_put_node (_Node *__p)

Protected Attributes

- _Node * _C_node

9.24.1 Detailed Description

Base tree class that encapsulates details of allocators. Three cases: an ordinary standard-conforming allocator, a standard-conforming allocator with no non-static data, and an SGI-style allocator. This complexity is necessary only because we're worrying about STL compatibility and because we want to avoid wasting storage on an allocator instance if it isn't necessary. Base class specialization for instanceless allocators.

9.24.2 Member Function Documentation

9.24.2.1 _Node* _Tree_alloc_base< _Tp, _Ctr, _TI, _Node, _Allocator, true >::_C_get_node ( )

`[inline, protected]`

allocate a new node

Definition at line 1414 of file vgtl_tree.h.

9.24.2.2 void _Tree_alloc_base< _Tp, _Ctr, _TI, _Node, _Allocator, true >::_C_put_node (_Node * __p)

`[inline, protected]`

deallocate a node

Definition at line 1417 of file vgtl_tree.h.

9.24.3 Member Data Documentation

9.24.3.1 _Node* _Tree_alloc_base< _Tp, _Ctr, _TI, _Node, _Allocator, true >::_C_node [protected]

This is the root node

Definition at line 1422 of file vgtl_tree.h.

The documentation for this class was generated from the following file:

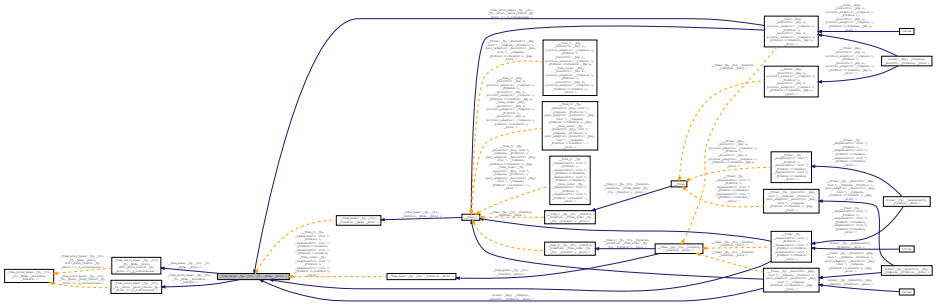
- vgtl_tree.h

9.25 _Tree_base< _Tp, _Ctr, _TI, _Node, _Alloc > Class Template Reference

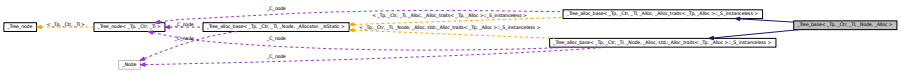
Tree base class for allocator encapsulation.

`#include <vgtl_tree.h>`

Inheritance diagram for _Tree_base< _Tp, _Ctr, _TI, _Node, _Alloc >:



Collaboration diagram for _Tree_base< _Tp, _Ctr, _TI, _Node, _Alloc >:



Public Types

- typedef _Base::allocator_type allocator_type
- typedef _Ctr container_type
- typedef _TI children_iterator
- typedef __one_iterator< void * > parents_iterator

Public Member Functions

- _Tree_base (const allocator_type &__a)
- virtual ~_Tree_base ()
- void clear ()
- void clear_children ()
- template<class _Output_Iterator >
  void add_all_children (_Output_Iterator fi, _Node *_parent)

Protected Member Functions

- _Node * _C_get_node ()
- void _C_put_node (_Node *__p)
- void _C_put_node (_Alloc *__p)

Protected Attributes

- _Node * _C_node

9.25.1 Detailed Description

template<class _Tp, class _Ctr, class _TI, class _Node, class _Alloc>class _Tree_base< _Tp, _Ctr, _TI, _Node, _Alloc >

Base tree class top level that encapsulates details of allocators.

Definition at line 1431 of file vgtl_tree.h.

9.25.2 Member Typedef Documentation

9.25.2.1 template<class _Tp, class _Ctr, class _TI, class _Node, class _Alloc> typedef _Base::allocator_type _Tree_base< _Tp, _Ctr, _TI, _Node, _Alloc >::allocator_type

allocator type

Reimplemented from _Tree_alloc_base< _Tp, _Ctr, _TI, _Alloc, _Alloc_traits< _Tp, _Alloc >::_S_instanceless >.

Definition at line 1440 of file vgtl_tree.h.

9.25.2.2 template<class _Tp, class _Ctr, class _TI, class _Node, class _Alloc> typedef _TI _Tree_base< _Tp, _Ctr, _TI, _Node, _Alloc >::children_iterator

iterator for accessing the children

Reimplemented in __Tree_t.

Definition at line 1445 of file vgtl_tree.h.

9.25.2.3 template<class _Tp, class _Ctr, class _TI, class _Node, class _Alloc> typedef _Ctr _Tree_base< _Tp, _Ctr, _TI, _Node, _Alloc >::container_type

internal container used to store the children

Definition at line 1443 of file vgtl_tree.h.

9.25.2.4 template<class _Tp, class _Ctr, class _TI, class _Node, class _Alloc> typedef __one_iterator<void *> _Tree_base< _Tp, _Ctr, _TI, _Node, _Alloc >::parents_iterator

iterator for accessing the parents

Reimplemented in __Tree_t.

Definition at line 1447 of file vgtl_tree.h.

9.25.3 Constructor & Destructor Documentation

9.25.3.1 template<class _Tp, class _Ctr, class _TI, class _Node, class _Alloc> _Tree_base< _Tp, _Ctr, _TI, _Node, _Alloc >::_Tree_base (const allocator_type & __a) [inline]

constructor initializing the allocator and the root

Definition at line 1450 of file vgtl_tree.h.

9.25.3.2 template<class _Tp, class _Ctr, class _TI, class _Node, class _Alloc> virtual _Tree_base< _Tp, _Ctr, _TI, _Node, _Alloc >::~_Tree_base ( ) [inline, virtual]

standard destructor

Definition at line 1458 of file vgtl_tree.h.

9.25.4 Member Function Documentation

9.25.4.1 _Node* _Tree_alloc_base< _Tp, _Ctr, _TI, _Node, _Alloc , _IsStatic >::_C_get_node ( ) [inline, protected, inherited]

allocate a new node

Definition at line 1375 of file vgtl_tree.h.

9.25.4.2 void _Tree_alloc_base< _Tp, _Ctr, _TI, _Alloc , _Alloc_traits< _Tp, _Alloc >::_S_instanceless , _IsStatic >::_C_put_node (_Alloc * __p) [inline, protected, inherited]

deallocate a node

Definition at line 1378 of file vgtl_tree.h.

9.25.4.3 void _Tree_alloc_base< _Tp, _Ctr, _TI, _Node, _Alloc , _IsStatic >::_C_put_node (_Node * __p) [inline, protected, inherited]

deallocate a node

Definition at line 1378 of file vgtl_tree.h.

9.25.4.4 template<class _Tp, class _Ctr, class _TI, class _Node, class _Alloc> template<class _Output_Iterator > void _Tree_base< _Tp, _Ctr, _TI, _Node, _Alloc >::add_all_children (_Output_Iterator *fi*, _Node * *_parent*)

add all children to the parent `_parent`. `fi` is a iterator to the children container of the parent

9.25.4.5 template<class _Tp, class _Ctr, class _TI, class _Node, class _Alloc> void _Tree_base< _Tp, _Ctr, _TI, _Node, _Alloc >::clear ( )

empty the tree

Reimplemented in __Tree_t.

9.25.4.6 template<class _Tp, class _Ctr, class _TI, class _Node, class _Alloc> void _Tree_base< _Tp, _Ctr, _TI, _Node, _Alloc >::clear_children ( ) [inline]

clear all children of the root node

Definition at line 1466 of file vgtl_tree.h.

9.25.5 Member Data Documentation

9.25.5.1 _Node* _Tree_alloc_base< _Tp, _Ctr, _TI, _Node, _Alloc , _IsStatic >::_C_node [protected, inherited]

This is the node

Definition at line 1387 of file vgtl_tree.h.

The documentation for this class was generated from the following files:

- vgtl_graph.h
- vgtl_tree.h

9.26 _Tree_data_hook Union Reference

`#include <vgtl_gdata.h>`

9.26.1 Detailed Description

This is a mixed-type union for data hooks on trees. A data hook can be used for non-recursive walks.

The documentation for this union was generated from the following file:

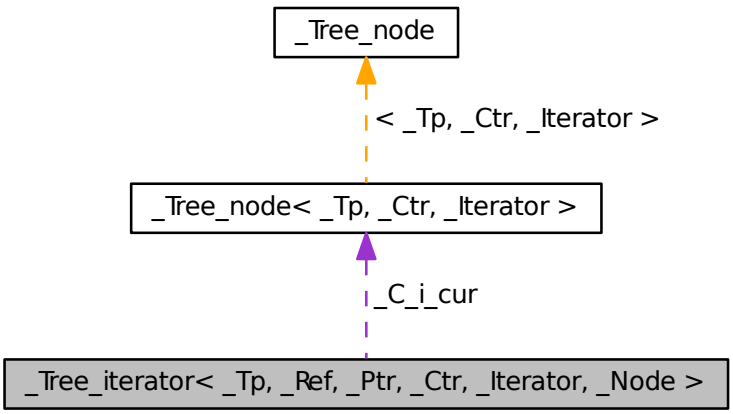
- vgtl_gdata.h

9.27 _Tree_iterator< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node > Class Template Reference

iterator through the tree

`#include <vgtl_tree.h>`

Collaboration diagram for _Tree_iterator< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >:



Public Types

- typedef std::bidirectional_iterator_tag iterator_category
- typedef _Tp value_type

- typedef _Ptr pointer
- typedef _Ref reference
- typedef size_t size_type
- typedef ptrdiff_t difference_type

Public Member Functions

- _Tree_iterator ()
- _Tree_iterator (const iterator &__x)
- _Tree_iterator (const _Node *__n, bool st=false)
- reference operator* () const
- pointer operator-> () const
- ctree_data_hook & data_hook ()
- _Self & operator= (const _Walk &__x)

- bool operator== (const _Self &__x) const
- bool operator!= (const _Self &__x) const

- _Self & operator++ ()
- _Self operator++ (int)
- _Self & operator-- ()
- _Self operator-- (int)

Protected Attributes

- _Node * _C_i_cur
- std::vector< _Ctr_iterator > _C_i_cur_it

9.27.1 Detailed Description

template<class _Tp, class _Ref, class _Ptr, class _Ctr, class _Iterator, class _Node>class _Tree_iterator< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >

This is an iterator, which visits each node of a tree once. It is based on a preorder depth-first automatic walker.

Definition at line 1141 of file vgtl_tree.h.

9.27.2 Member Typedef Documentation

9.27.2.1 template<class _Tp , class _Ref , class _Ptr , class _Ctr , class _Iterator , class _Node > typedef ptrdiff_t _Tree_iterator< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >::difference_type

standard iterator definition

Definition at line 1156 of file vgtl_tree.h.

9.27.2.2 template<class _Tp , class _Ref , class _Ptr , class _Ctr , class _Iterator , class _Node > typedef std::bidirectional_iterator_tag _Tree_iterator< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >::iterator_category

standard iterator definition

Definition at line 1151 of file vgtl_tree.h.

9.27.2.3 template<class _Tp , class _Ref , class _Ptr , class _Ctr , class _Iterator , class _Node > typedef _Ptr _Tree_iterator< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >::pointer

standard iterator definition

Definition at line 1153 of file vgtl_tree.h.

9.27.2.4 template<class _Tp , class _Ref , class _Ptr , class _Ctr , class _Iterator , class _Node > typedef _Ref _Tree_iterator< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >::reference

standard iterator definition

Definition at line 1154 of file vgtl_tree.h.

9.27.2.5 template<class _Tp , class _Ref , class _Ptr , class _Ctr , class _Iterator , class _Node > typedef size_t _Tree_iterator< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >::size_type

standard iterator definition

Definition at line 1155 of file vgtl_tree.h.

9.27.2.6 template<class _Tp , class _Ref , class _Ptr , class _Ctr , class _Iterator , class _Node > typedef _Tp _Tree_iterator< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >::value_type

standard iterator definition

Definition at line 1152 of file vgtl_tree.h.

9.27.3 Constructor & Destructor Documentation

9.27.3.1 template<class _Tp , class _Ref , class _Ptr , class _Ctr , class _Iterator , class _Node > _Tree_iterator< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >::_Tree_iterator ( ) [inline]

standard constructor

Definition at line 1168 of file vgtl_tree.h.

9.27.3.2 template<class _Tp , class _Ref , class _Ptr , class _Ctr , class _Iterator , class _Node > _Tree_iterator< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >::_Tree_iterator (const iterator & *__x*) [inline]

copy constructor

Definition at line 1170 of file vgtl_tree.h.

9.27.3.3 template<class _Tp , class _Ref , class _Ptr , class _Ctr , class _Iterator , class _Node > _Tree_iterator< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >::_Tree_iterator (const _Node * *__n,* bool *st* = false) [inline]

constructor setting a specific position

Definition at line 1173 of file vgtl_tree.h.

9.27.4 Member Function Documentation

9.27.4.1 template<class _Tp , class _Ref , class _Ptr , class _Ctr , class _Iterator , class _Node > ctree_data_hook& _Tree_iterator< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >::data_hook ( ) [inline]

access to the data hook of the node

Definition at line 1199 of file vgtl_tree.h.

9.27.4.2 template<class _Tp , class _Ref , class _Ptr , class _Ctr , class _Iterator , class _Node > bool _Tree_iterator< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >::operator!= (const _Self & __x) const [inline]

comparison operator

Definition at line 1184 of file vgtl_tree.h.

9.27.4.3 template<class _Tp , class _Ref , class _Ptr , class _Ctr , class _Iterator , class _Node > reference _Tree_iterator< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >::operator* ( ) const [inline]

dereference operator

Definition at line 1192 of file vgtl_tree.h.

9.27.4.4 template<class _Tp , class _Ref , class _Ptr , class _Ctr , class _Iterator , class _Node > _Self& _Tree_iterator< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >::operator++ ( ) [inline]

in(de)crement operator

Definition at line 1222 of file vgtl_tree.h.

9.27.4.5 template<class _Tp , class _Ref , class _Ptr , class _Ctr , class _Iterator , class _Node > _Self _Tree_iterator< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >::operator++ (int) [inline]

in(de)crement operator

Definition at line 1226 of file vgtl_tree.h.

9.27.4.6 template<class _Tp , class _Ref , class _Ptr , class _Ctr , class _Iterator , class _Node > _Self& _Tree_iterator< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >::operator-- ( ) [inline]

in(de)crement operator

Definition at line 1232 of file vgtl_tree.h.

9.27.4.7 template<class _Tp , class _Ref , class _Ptr , class _Ctr , class _Iterator , class _Node > _Self _Tree_iterator< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >::operator-- (int) [inline]

in(de)crement operator

Definition at line 1236 of file vgtl_tree.h.

9.27.4.8 template<class _Tp , class _Ref , class _Ptr , class _Ctr , class _Iterator , class _Node > pointer _Tree_iterator< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >::operator-> ( ) const [inline]

pointer operator

Definition at line 1196 of file vgtl_tree.h.

9.27.4.9 template<class _Tp , class _Ref , class _Ptr , class _Ctr , class _Iterator , class _Node > _Self& _Tree_iterator< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >::operator= (const _Walk & __x) [inline]

assignment to iterator from walker

Definition at line 1211 of file vgtl_tree.h.

9.27.4.10 template<class _Tp , class _Ref , class _Ptr , class _Ctr , class _Iterator , class _Node > bool _Tree_iterator< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >::operator== (const _Self & __x) const [inline]

comparison operator

Definition at line 1178 of file vgtl_tree.h.

9.27.5 Member Data Documentation

9.27.5.1 template<class _Tp , class _Ref , class _Ptr , class _Ctr , class _Iterator , class _Node > _Node * _Tree_iterator< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >::_C_i_cur [protected]

current position

Definition at line 915 of file vgtl_graph.h.

9.27.5.2 template<class _Tp , class _Ref , class _Ptr , class _Ctr , class _Iterator , class _Node > std::vector<_Ctr_iterator> _Tree_iterator< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >::_C_i_cur_it [protected]

internal stack

Definition at line 1164 of file vgtl_tree.h.

The documentation for this class was generated from the following files:

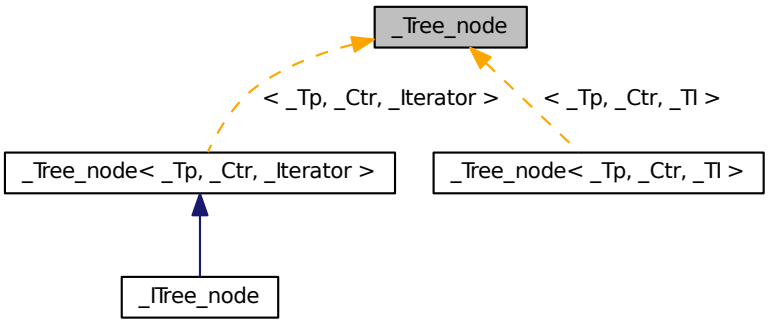
- vgtl_graph.h
- vgtl_tree.h

9.28 _Tree_node Class Reference

tree node for trees w/o data hooks

`#include <vgtl_tree.h>`

Inheritance diagram for _Tree_node:



Public Member Functions

- _Tree_node ()
- void initialize ()
- void get_rid_of ()
- void clear_tree ()
- void clear_children ()
- _Ctr_iterator get_childentry_iterator (_Void_pointer __p)
- template<class _Output_Iterator >
  void add_all_children (_Output_Iterator fi, _Self *_parent)
- template<class Compare >
  void sort_children (_Ctr_iterator first, _Ctr_iterator last, Compare comp)
- template<class Compare >
  void sort_parents (_Ctr_iterator first, _Ctr_iterator last, Compare comp)

Public Attributes

- _Tp _C_data
- _Void_pointer _C_parent
- _Ctr _C_children

9.28.1 Detailed Description

This is the tree node for a tree without data hooks

9.28.2 Constructor & Destructor Documentation

9.28.2.1 _Tree_node::_Tree_node ( ) [inline]

standard constructor

Definition at line 80 of file vgtl_tree.h.

9.28.3 Member Function Documentation

9.28.3.1 template<class _Output_Iterator > void _Tree_node::add_all_children (_Output_Iterator *fi*, _Self * *_parent*)

add all children to parent `_parent`. `fi` is an iterator to the children container of `_parent`

9.28.3.2 void _Tree_node::clear_children ( ) [inline]

erase all children entries

Definition at line 101 of file vgtl_tree.h.

9.28.3.3 void _Tree_node::clear_tree ( )

remove the whole subtree below this node

9.28.3.4 _Ctr_iterator _Tree_node::get_childentry_iterator (_Void_pointer *__p*) [inline]

find the iterator into the children container for child `__p`

Definition at line 105 of file vgtl_tree.h.

9.28.3.5 void _Tree_node::get_rid_of ( ) [inline]

remove the children container

Reimplemented in _ITree_node.

Definition at line 94 of file vgtl_tree.h.

9.28.3.6 void _Tree_node::initialize ( ) [inline]

initialize the data structure

Reimplemented in _ITree_node.

Definition at line 88 of file vgtl_tree.h.

9.28.3.7 template<class Compare > void _Tree_node::sort_children (_Ctr_iterator *first*, _Ctr_iterator *last*, Compare *comp*) [inline]

sort the children according to `comp`

Definition at line 122 of file vgtl_tree.h.

9.28.3.8 template<class Compare > void _Tree_node::sort_parents (_Ctr_iterator *first*, _Ctr_iterator *last*, Compare *comp*) [inline]

sort the children according to `comp`, i.e. do nothing here

Definition at line 129 of file vgtl_tree.h.

9.28.4 Member Data Documentation

9.28.4.1 _Ctr _Tree_node::_C_children

the edges to the children

Definition at line 77 of file vgtl_tree.h.

9.28.4.2 _Tp _Tree_node::_C_data

the node data

Definition at line 73 of file vgtl_tree.h.

9.28.4.3 _Void_pointer _Tree_node::_C_parent

the edge to the parent

Definition at line 75 of file vgtl_tree.h.

The documentation for this class was generated from the following file:

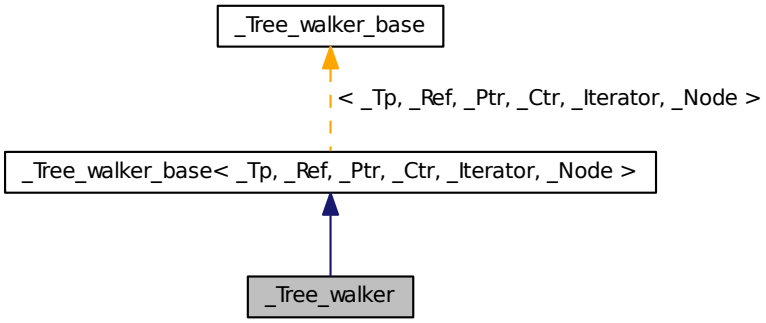
- vgtl_tree.h

9.29 _Tree_walker Class Reference

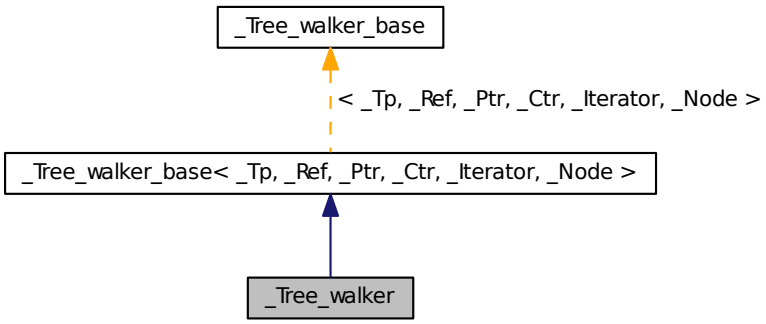
automatic tree walkers

`#include <vgtl_tree.h>`

Inheritance diagram for _Tree_walker:



Collaboration diagram for _Tree_walker:



Public Types

- typedef _Tp value_type
- typedef _Ptr pointer
- typedef _Ref reference

- typedef __one_iterator< void * > parents_iterator
- typedef _Ctr_iterator children_iterator
- typedef _Node node_type
- typedef size_t size_type
- typedef ptrdiff_t difference_type

Public Member Functions

- _Tree_walker ()
- _Tree_walker (_Node *__x, int order=(_C_W_preorder|_C_W_postorder), bool front_to_back=true, bool depth_first=true, bool find_start=true)
- _Tree_walker (const walker &__x)
- _Self operator<< (const parents_iterator &__dummy)

  go to parent operator
- _Self operator>> (const children_iterator &__i)

  go to child operator
- _Self & operator<<= (const parents_iterator &__dummy)
- _Self & operator>>= (const children_iterator &__i)
- _Self & operator~ ()
- _Self & operator= (const _Itr &__x)
- bool in_preorder ()
- reference operator* () const

- pointer operator-> () const
- ctree_data_hook & data_hook ()
- ctree_data_hook & parent_data_hook ()
- const _Node * parent ()
- const _Node * node ()
- size_type n_children ()
- size_type n_parents ()
- bool is_leaf ()
- bool is_root ()
- bool is_ground ()
- bool is_sky ()
- children_iterator child_begin ()
- children_iterator child_end ()
- parents_iterator parent_begin ()
- parents_iterator parent_end ()
- _Function for_each_child (_Function __f)
- _Function for_each_parent (_Function __f)
- void sort_children (children_iterator first, children_iterator last, Compare comp)
- void sort_children (Compare comp)
- void sort_parents (parents_iterator first, parents_iterator last, Compare comp)
- void sort_parents (Compare comp)

- bool operator== (const _Self &__x) const
- bool operator!= (const _Self &__x) const

- _Self & operator++ ()
- _Self operator++ (int)
- _Self & operator-- ()
- _Self operator-- (int)

Public Attributes

- struct {
  } _C_w_t

- bool _C_w_in_preorder
- std::vector< _Iterator > _C_w_cur_it
- _Node * _C_w_cur

9.29.1 Detailed Description

This is the class defining automatic (iterative) tree walkers, which walk trees without guidance.

9.29.2 Member Typedef Documentation

9.29.2.1 typedef _Ctr_iterator _Tree_walker_base::children_iterator [inherited]

standard walker definition

Definition at line 243 of file vgtl_tree.h.

9.29.2.2 typedef ptrdiff_t _Tree_walker_base::difference_type [inherited]

standard walker definition

Definition at line 247 of file vgtl_tree.h.

9.29.2.3 typedef _Node _Tree_walker_base::node_type [inherited]

standard walker definition

Definition at line 244 of file vgtl_tree.h.

9.29.2.4 typedef __one_iterator<void *> _Tree_walker_base::parents_iterator [inherited]

standard walker definition

Definition at line 242 of file vgtl_tree.h.

9.29.2.5 typedef _Ptr _Tree_walker_base::pointer [inherited]

standard walker definition

Definition at line 233 of file vgtl_tree.h.

9.29.2.6 typedef _Ref _Tree_walker_base::reference [inherited]

standard walker definition

Definition at line 234 of file vgtl_tree.h.

9.29.2.7 typedef size_t _Tree_walker_base::size_type [inherited]

standard walker definition

Definition at line 246 of file vgtl_tree.h.

9.29.2.8 typedef _Tp _Tree_walker_base::value_type [inherited]

standard walker definition

Definition at line 232 of file vgtl_tree.h.

9.29.3 Constructor & Destructor Documentation

9.29.3.1 _Tree_walker::_Tree_walker ( ) [inline]

standard constructor

Definition at line 381 of file vgtl_tree.h.

9.29.3.2 _Tree_walker::_Tree_walker (_Node * __x, int order = (_C_W_preorder|_C_W_postorder), bool front_to_back = true, bool depth_first = true, bool find_start = true) [inline]

This is the main constructor for an automatic walker. It sets the starting position and, optionally, the walker type.

Definition at line 406 of file vgtl_tree.h.

9.29.3.3 _Tree_walker::_Tree_walker (const walker & __x) [inline]

copy constructor

Definition at line 423 of file vgtl_tree.h.

9.29.4 Member Function Documentation

9.29.4.1 children_iterator _Tree_walker_base::child_begin ( ) [inline, inherited]

return children_iterator to first child

Definition at line 307 of file vgtl_tree.h.

9.29.4.2 children_iterator _Tree_walker_base::child_end ( ) [inline, inherited]

return children_iterator beyond last child

Definition at line 309 of file vgtl_tree.h.

9.29.4.3 ctree_data_hook& _Tree_walker_base::data_hook ( ) [inline, inherited]

retrieve the data hook

Definition at line 280 of file vgtl_tree.h.

9.29.4.4 _Function _Tree_walker_base::for_each_child (_Function __f) [inline, inherited]

apply the function ___f to all children

Definition at line 320 of file vgtl_tree.h.

9.29.4.5 _Function _Tree_walker_base::for_each_parent (_Function __f) [inline, inherited]

apply the function ___f to all parents

Definition at line 326 of file vgtl_tree.h.

9.29.4.6 bool _Tree_walker::in_preorder ( ) [inline]

are we in the preorder phase of a pre+post walk?

Definition at line 587 of file vgtl_tree.h.

9.29.4.7 bool _Tree_walker_base::is_ground ( ) [inline, inherited]

is this node a virtual node - the ground (below all roots)?

Definition at line 302 of file vgtl_tree.h.

9.29.4.8 bool _Tree_walker_base::is_leaf ( ) [inline, inherited]

is this node a leaf?

Definition at line 296 of file vgtl_tree.h.

9.29.4.9 bool _Tree_walker_base::is_root ( ) [inline, inherited]

is this node a root?

Definition at line 298 of file vgtl_tree.h.

9.29.4.10 bool _Tree_walker_base::is_sky ( ) [inline, inherited]

is this node a virtual node - the sky (above all leafs)?

Definition at line 304 of file vgtl_tree.h.

9.29.4.11 size_type _Tree_walker_base::n_children ( ) [inline, inherited]

return the number of children

Definition at line 291 of file vgtl_tree.h.

9.29.4.12 size_type _Tree_walker_base::n_parents ( ) [inline, inherited]

return the number of parents (0 or 1)

Definition at line 293 of file vgtl_tree.h.

9.29.4.13 const _Node* _Tree_walker_base::node ( ) [inline, inherited]

retrieve the full node

Definition at line 288 of file vgtl_tree.h.

9.29.4.14 bool _Tree_walker::operator!= (const _Self & __x) const [inline]

comparison operator

Definition at line 439 of file vgtl_tree.h.

9.29.4.15 reference _Tree_walker_base::operator* ( ) const [inline, inherited]

dereference operator

Definition at line 265 of file vgtl_tree.h.

9.29.4.16 _Self& _Tree_walker::operator++ ( ) [inline]

in(de)crement operator

Definition at line 452 of file vgtl_tree.h.

9.29.4.17 _Self _Tree_walker::operator++ (int) [inline]

in(de)crement operator

Definition at line 474 of file vgtl_tree.h.

9.29.4.18 _Self& _Tree_walker::operator-- ( ) [inline]

in(de)crement operator

Definition at line 480 of file vgtl_tree.h.

9.29.4.19 _Self _Tree_walker::operator-- (int) [inline]

in(de)crement operator

Definition at line 502 of file vgtl_tree.h.

9.29.4.20 pointer _Tree_walker_base::operator-> ( ) const [inline, inherited]

pointer operator

Definition at line 269 of file vgtl_tree.h.

9.29.4.21 _Self _Tree_walker::operator<< (const parents_iterator & *__dummy*) [inline]

This operator moves the walker to the parent

Definition at line 511 of file vgtl_tree.h.

9.29.4.22 _Self& _Tree_walker::operator<<= (const parents_iterator & *__dummy*) [inline]

go to parent assignment operator

Definition at line 542 of file vgtl_tree.h.

9.29.4.23 _Self& _Tree_walker::operator= (const _Itr & *__x*) [inline]

assignment from iterator

Reimplemented from _Tree_walker_base< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >.

Definition at line 577 of file vgtl_tree.h.

9.29.4.24 bool _Tree_walker::operator== (const _Self & *__x*) const [inline]

comparison operator

Definition at line 431 of file vgtl_tree.h.

9.29.4.25 _Self _Tree_walker::operator>> (const children_iterator & *__i*) [inline]

This operator moves the walker to the child pointed to by ___i

Definition at line 531 of file vgtl_tree.h.

9.29.4.26 _Self& _Tree_walker::operator>>= (const children_iterator & *__i*) [inline]

go to child assignment operator

Definition at line 560 of file vgtl_tree.h.

9.29.4.27 _Self& _Tree_walker::operator~ ( ) [inline]

switch from preorder to postorder phase

Definition at line 570 of file vgtl_tree.h.

9.29.4.28 const _Node* _Tree_walker_base::parent ( ) [inline, inherited]

retrieve the parent node

Definition at line 286 of file vgtl_tree.h.

9.29.4.29 parents_iterator _Tree_walker_base::parent_begin ( ) [inline, inherited]

return parents_iterator to first parent (the parent)

Definition at line 312 of file vgtl_tree.h.

9.29.4.30 ctree_data_hook& _Tree_walker_base::parent_data_hook ( ) [inline, inherited]

retrieve the parent's data hook

Definition at line 282 of file vgtl_tree.h.

9.29.4.31 parents_iterator _Tree_walker_base::parent_end ( ) [inline, inherited]

return parents_iterator beyond last parent

Definition at line 315 of file vgtl_tree.h.

9.29.4.32 void _Tree_walker_base::sort_children (children_iterator *first*, children_iterator *last*, Compare *comp*) [inline, inherited]

sort the children in the range [first,last) according to `comp`

Definition at line 333 of file vgtl_tree.h.

9.29.4.33 void _Tree_walker_base::sort_children (Compare *comp*) [inline, inherited]

sort all children according to `comp`

Definition at line 344 of file vgtl_tree.h.

9.29.4.34 void _Tree_walker_base::sort_parents (parents_iterator *first*, parents_iterator *last*, Compare *comp*) [inline, inherited]

sort the parents in the range [first,last) according to `comp` (NOP)

Definition at line 339 of file vgtl_tree.h.

9.29.4.35 void _Tree_walker_base::sort_parents (Compare *comp*) [inline, inherited]

sort all parents according to `comp` (NOP = do nothing)

Definition at line 349 of file vgtl_tree.h.

9.29.5 Member Data Documentation

9.29.5.1 _Node* _Tree_walker_base::_C_w_cur [inherited]

pointer to the current node

Definition at line 252 of file vgtl_tree.h.

9.29.5.2 std::vector<_Iterator> _Tree_walker::_C_w_cur_it

internal stack

Definition at line 377 of file vgtl_tree.h.

9.29.5.3 bool _Tree_walker::_C_w_in_preorder

walker is in preorder mode?

Definition at line 375 of file vgtl_tree.h.

9.29.5.4 struct { ... } _Tree_walker::_C_w_t

walker type (order, front to back/back to front, depth/breath first

The documentation for this class was generated from the following files:

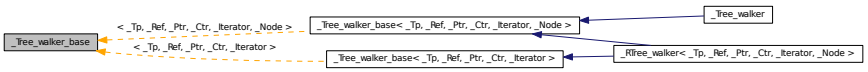
- vgtl_tree.h
- vgtl_graph.h

9.30 _Tree_walker_base Class Reference

base class for all tree walkers

`#include <vgtl_tree.h>`

Inheritance diagram for _Tree_walker_base:



Public Types

- typedef _Tp value_type
- typedef _Ptr pointer
- typedef _Ref reference

- typedef __one_iterator< void * > parents_iterator
- typedef _Ctr_iterator children_iterator
- typedef _Node node_type
- typedef size_t size_type
- typedef ptrdiff_t difference_type

Public Member Functions

- _Tree_walker_base ()

- _Tree_walker_base (_Node *__x)
- _Tree_walker_base (const walker &__x)
- reference operator* () const
- pointer operator-> () const
- _Self & operator= (const _Itr &__x)
- ctree_data_hook & data_hook ()
- ctree_data_hook & parent_data_hook ()
- const _Node * parent ()
- const _Node * node ()
- size_type n_children ()
- size_type n_parents ()
- bool is_leaf ()
- bool is_root ()
- bool is_ground ()
- bool is_sky ()
- children_iterator child_begin ()
- children_iterator child_end ()
- parents_iterator parent_begin ()
- parents_iterator parent_end ()
- template<class _Function >
  _Function for_each_child (_Function __f)
- template<class _Function >
  _Function for_each_parent (_Function __f)
- template<class Compare >
  void sort_children (children_iterator first, children_iterator last, Compare comp)
- template<class Compare >
  void sort_parents (parents_iterator first, parents_iterator last, Compare comp)
- template<class Compare >
  void sort_children (Compare comp)
- template<class Compare >
  void sort_parents (Compare comp)

Public Attributes

- _Node * _C_w_cur

9.30.1 Detailed Description

This is the base class for all tree walkers.

9.30.2 Member Typedef Documentation

9.30.2.1 typedef _Ctr_iterator _Tree_walker_base::children_iterator

standard walker definition

Definition at line 243 of file vgtl_tree.h.

9.30.2.2 typedef ptrdiff_t _Tree_walker_base::difference_type

standard walker definition

Definition at line 247 of file vgtl_tree.h.

9.30.2.3 typedef _Node _Tree_walker_base::node_type

standard walker definition

Definition at line 244 of file vgtl_tree.h.

9.30.2.4 typedef __one_iterator<void *> _Tree_walker_base::parents_iterator

standard walker definition

Definition at line 242 of file vgtl_tree.h.

9.30.2.5 typedef _Ptr _Tree_walker_base::pointer

standard walker definition

Definition at line 233 of file vgtl_tree.h.

9.30.2.6 typedef _Ref _Tree_walker_base::reference

standard walker definition

Definition at line 234 of file vgtl_tree.h.

9.30.2.7 typedef size_t _Tree_walker_base::size_type

standard walker definition

Definition at line 246 of file vgtl_tree.h.

9.30.2.8 typedef _Tp _Tree_walker_base::value_type

standard walker definition

Definition at line 232 of file vgtl_tree.h.

9.30.3 Constructor & Destructor Documentation

9.30.3.1 _Tree_walker_base::_Tree_walker_base ( ) [inline]

standard constructor

Definition at line 256 of file vgtl_tree.h.

9.30.3.2 _Tree_walker_base::_Tree_walker_base (_Node * __x) [inline]

constructor setting the position

Definition at line 259 of file vgtl_tree.h.

9.30.3.3 _Tree_walker_base::_Tree_walker_base (const walker & __x) [inline]

copy constructor

Definition at line 262 of file vgtl_tree.h.

9.30.4 Member Function Documentation

9.30.4.1 children_iterator _Tree_walker_base::child_begin ( ) [inline]

return children_iterator to first child

Definition at line 307 of file vgtl_tree.h.

9.30.4.2 children_iterator _Tree_walker_base::child_end ( ) [inline]

return children_iterator beyond last child

Definition at line 309 of file vgtl_tree.h.

9.30.4.3 ctree_data_hook& _Tree_walker_base::data_hook ( ) [inline]

retrieve the data hook

Definition at line 280 of file vgtl_tree.h.

9.30.4.4 template<class _Function > _Function _Tree_walker_base::for_each_child (_Function __f) [inline]

apply the function __f to all children

Definition at line 320 of file vgtl_tree.h.

9.30.4.5 template<class _Function > _Function _Tree_walker_base::for_each_parent (_Function __f) [inline]

apply the function __f to all parents

Definition at line 326 of file vgtl_tree.h.

9.30.4.6 bool _Tree_walker_base::is_ground ( ) [inline]

is this node a virtual node - the ground (below all roots)?

Definition at line 302 of file vgtl_tree.h.

9.30.4.7 bool _Tree_walker_base::is_leaf ( ) [inline]

is this node a leaf?

Definition at line 296 of file vgtl_tree.h.

9.30.4.8 bool _Tree_walker_base::is_root ( ) [inline]

is this node a root?

Definition at line 298 of file vgtl_tree.h.

9.30.4.9 bool _Tree_walker_base::is_sky ( ) [inline]

is this node a virtual node - the sky (above all leafs)?

Definition at line 304 of file vgtl_tree.h.

9.30.4.10 size_type _Tree_walker_base::n_children ( ) [inline]

return the number of children

Definition at line 291 of file vgtl_tree.h.

9.30.4.11 size_type _Tree_walker_base::n_parents ( ) [inline]

return the number of parents (0 or 1)

Definition at line 293 of file vgtl_tree.h.

9.30.4.12 const _Node* _Tree_walker_base::node ( ) [inline]

retrieve the full node

Definition at line 288 of file vgtl_tree.h.

9.30.4.13 reference _Tree_walker_base::operator* ( ) const [inline]

dereference operator

Definition at line 265 of file vgtl_tree.h.

9.30.4.14 pointer _Tree_walker_base::operator-> ( ) const [inline]

pointer operator

Definition at line 269 of file vgtl_tree.h.

9.30.4.15 _Self& _Tree_walker_base::operator= (const _Itr & *__x*) [inline]

assignment operator from iterator to walker

Reimplemented in _RTree_walker< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >, and _Tree_walker.

Definition at line 274 of file vgtl_tree.h.

9.30.4.16 const _Node* _Tree_walker_base::parent ( ) [inline]

retrieve the parent node

Definition at line 286 of file vgtl_tree.h.

9.30.4.17 parents_iterator _Tree_walker_base::parent_begin ( ) [inline]

return parents_iterator to first parent (the parent)

Definition at line 312 of file vgtl_tree.h.

9.30.4.18 ctree_data_hook& _Tree_walker_base::parent_data_hook ( ) [inline]

retrieve the parent's data hook

Definition at line 282 of file vgtl_tree.h.

9.30.4.19 parents_iterator _Tree_walker_base::parent_end ( ) [inline]

return parents_iterator beyond last parent

Definition at line 315 of file vgtl_tree.h.

9.30.4.20 template<class Compare > void _Tree_walker_base::sort_children (children_iterator *first*, children_iterator *last*, Compare *comp*) [inline]

sort the children in the range [first,last) according to `comp`

Definition at line 333 of file vgtl_tree.h.

9.30.4.21 template<class Compare > void _Tree_walker_base::sort_children (Compare *comp*) [inline]

sort all children according to comp

Definition at line 344 of file vgtl_tree.h.

9.30.4.22 template<class Compare > void _Tree_walker_base::sort_parents (parents_iterator *first*, parents_iterator *last*, Compare *comp*) [inline]

sort the parents in the range [first,last) according to comp (NOP)

Definition at line 339 of file vgtl_tree.h.

9.30.4.23 template<class Compare > void _Tree_walker_base::sort_parents (Compare *comp*) [inline]

sort all parents according to comp (NOP = do nothing)

Definition at line 349 of file vgtl_tree.h.

9.30.5 Member Data Documentation

9.30.5.1 _Node* _Tree_walker_base::_C_w_cur

pointer to the current node

Definition at line 252 of file vgtl_tree.h.

The documentation for this class was generated from the following file:

- vgtl_tree.h

9.31 array_vector Class Reference

Public Member Functions

- array_vector ()
- array_vector (_TT *__a, int n)
- ~array_vector ()
- void assignvector (_TT *__a, int n)

9.31.1 Constructor & Destructor Documentation

9.31.1.1 array_vector::array_vector ( ) [inline]

standard constructor

Definition at line 65 of file array_vector.h.

9.31.1.2 array_vector::array_vector (_TT * *__a*, int *n*) [inline]

constructor building an array_vector from pointer __a with size n

Definition at line 73 of file array_vector.h.

9.31.1.3 array_vector::~array_vector ( ) [inline]

standard destructor

Definition at line 83 of file array_vector.h.

9.31.2 Member Function Documentation

9.31.2.1 void array_vector::assignvector (_TT * __a, int n) [inline]

assign an array (__a) of length n to this array_vector.

Definition at line 90 of file array_vector.h.

The documentation for this class was generated from the following file:

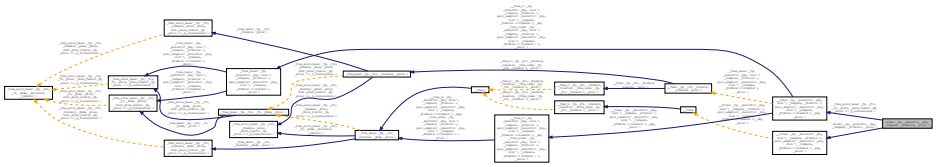
- array_vector.h

9.32 atree< _Tp, _AssocCtr, _Key, _Compare, _PtrAlloc, _Alloc > Class Template Reference

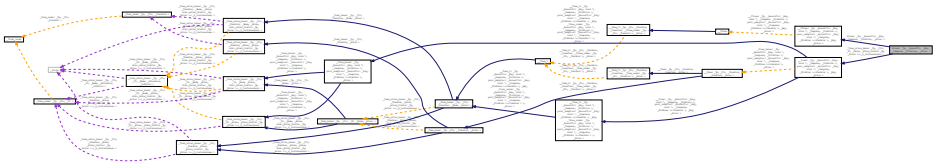
n-ary forest with labelled edges

`#include <vgtl_tree.h>`

Inheritance diagram for atree< _Tp, _AssocCtr, _Key, _Compare, _PtrAlloc, _Alloc >:



Collaboration diagram for atree< _Tp, _AssocCtr, _Key, _Compare, _PtrAlloc, _Alloc >:



Public Types

- typedef _Tree_iterator< _Tp, _Tp &, _Tp *, container_type, children_iterator, node_type > iterator
- typedef _Tree_iterator< _Tp, const _Tp &, const _Tp *, container_type, children_iterator, node_type > const_iterator
- typedef _Tree_walker< _Tp, _Tp &, _Tp *, container_type, children_iterator, _Node > iterative_walker
- typedef _Tree_walker< _Tp, const _Tp &, const _Tp *, container_type, children_iterator, _Node > const_iterative_walker
- typedef std::reverse_iterator < const_iterator > const_reverse_iterator
- typedef std::reverse_iterator < iterator > reverse_iterator

- typedef _Tp value_type
- typedef _Node node_type
- typedef value_type * pointer
- typedef const value_type * const_pointer
- typedef value_type & reference
- typedef const value_type & const_reference
- typedef size_t size_type
- typedef ptrdiff_t difference_type
- typedef _Tree_iterator< _Tp, _Tp &, _Tp *, container_type, container_iterator > iterator
- typedef _Tree_iterator< _Tp, const _Tp &, const _Tp *, container_type, container_iterator > const_iterator
- typedef reverse_iterator < const_iterator > const_reverse_iterator
- typedef reverse_iterator < iterator > reverse_iterator
- typedef _Tree_walker< _Tp, _Tp &, _Tp *, container_type, container_iterator > walker
- typedef _Tree_walker< _Tp, const _Tp &, const _Tp *, container_type, container_iterator > const_walker
- typedef _Iterator children_iterator
- typedef _TI children_iterator
- typedef __one_iterator< void * > parents_iterator
- typedef __one_iterator< void * > parents_iterator

Public Member Functions

- _Self & operator= (_Node *__x)
- void insert (const __walker_base &__position, const _Tp &__x, const _Key &__k)
- void insert (const __walker_base &__position, const _Key &__k)
- iterative_walker root (walker_type wt=cw_pre_post, bool front_to_back=true, bool depth_first=true)
- const_iterative_walker root (walker_type wt=cw_pre_post, bool front_to_back=true, bool depth_first=true) const
- iterative_walker through ()
- const_iterative_walker through () const
- iterative_walker begin (walker_type wt=cw_pre_post, bool front_to_back=true, bool depth_first=true)
- const_iterative_walker begin (walker_type wt=cw_pre_post, bool front_to_back=true, bool depth_first=true) const
- iterative_walker end (walker_type wt=cw_pre_post, bool front_to_back=true, bool depth_first=true)
- const_iterative_walker end (walker_type wt=cw_pre_post, bool front_to_back=true, bool depth_first=true) const
- reverse_iterator rbegin ()
- const_reverse_iterator rbegin () const
- reverse_iterator rend ()
- const_reverse_iterator rend () const
- size_type size () const
- reference getroot ()
- const_reference getroot () const
- size_type depth (const iterative_walker &__position)
- allocator_type get_allocator () const
- walker root (children_iterator __it)
- const_walker root (children_iterator __it) const
- walker root ()
- const_walker root () const

- iterator begin ()
- const_iterator begin () const
- iterator end ()
- const_iterator end () const
- bool empty () const
- size_type max_size () const
- void **swap** (_Self &__x)
- void insert_child (const __walker_base &__position, const _Tp &__x, const container_insert_arg &__It)
- void insert_child (const __walker_base &__position, const container_insert_arg &__It)
- void insert_children (const __walker_base &__position, size_type __n, const _Tp &__x, const children-_iterator &__It)
- void insert_subtree (const __walker_base &__position, _Self &__subtree, const children_iterator &__It)
- void erase (const __walker_base &__position)
- _Node * erase_tree (const __walker_base &__position)
- bool erase_child (const __walker_base &__position, const children_iterator &__It)
- _Node * erase_subtree (const __walker_base &__position, const children_iterator &__It)
- size_type **depth** (const walker &__position)
- walker ground ()
- const_walker ground () const
- void clear_children ()
- void add_all_children (_Output_Iterator fi, _Node *_parent)
- template<class _Output_Iterator >
  void add_all_children (_Output_Iterator fi, _Node *_parent)

Protected Member Functions

- _Node * _C_create_node (const _Tp &__x)
- _Node * _C_create_node ()
- _Node * _C_get_node ()
- void _C_put_node (_Node *__p)
- void _C_put_node (_Node *__p)
- void _C_put_node (_Node *__p)
- void _C_put_node (_Alloc *__p)

Protected Attributes

- _Node * _C_node

Friends

- bool operator==__VGTL_NULL_TMPL_ARGS (const __ITree &__x, const __ITree &__y)

9.32.1 Detailed Description

template<class _Tp, template< class __Key, class __Ty, class __Compare, class __AllocT > class _AssocCtr = std::multimap, class _Key = string, class _Compare = less<_Key>, class _PtrAlloc = __VGTL_DEFAULT_ALLOCATOR(void *), class _Alloc = __VGTL_DEFAULT_ALLOCATOR(_Tp)>class atree< _Tp, _AssocCtr, _Key, _Compare, _PtrAlloc, _Alloc >

This class constructs an n-ary forest with data hooks and labelled edges. By default, the children are collected in a STL multimap, but the container can be replaced by any other associative map container.

Definition at line 2700 of file vgtl_tree.h.

9.32.2 Member Typedef Documentation

9.32.2.1 template<class _Tp, class _Ctr, class _Tl, class _Node, class _Alloc> typedef _Tl _Tree_base< _Tp, _Ctr, _Tl, _Node, _Alloc >::children_iterator [inherited]

iterator for accessing the children

Reimplemented in __Tree_t.

Definition at line 1445 of file vgtl_tree.h.

9.32.2.2 typedef _Iterator __Tree_t::children_iterator [inherited]

iterator for accessing the children

Reimplemented from _Tree_base< _Tp, _Ctr, _Iterator, _Node, _Alloc >.

Definition at line 1563 of file vgtl_tree.h.

9.32.2.3 typedef _Tree_walker<_Tp,const _Tp&,const _Tp*,container_type,children_iterator,_Node> __ITree::const_iterative_walker [inherited]

the const iterative walker

Definition at line 2065 of file vgtl_tree.h.

9.32.2.4 typedef _Tree_iterator<_Tp,const _Tp&,const _Tp*,container_type,container_iterator> __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::const_iterator [inherited]

the const iterator

Reimplemented from __Tree_t.

Definition at line 1263 of file vgtl_graph.h.

9.32.2.5 typedef _Tree_iterator<_Tp,const _Tp&,const _Tp*,container_type,children_iterator,node_type> __ITree::const_iterator [inherited]

the const iterator

Definition at line 2060 of file vgtl_tree.h.

9.32.2.6 typedef const value_type* __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::const_pointer [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1251 of file vgtl_graph.h.

9.32.2.7 typedef const value_type& __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::const_reference [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1253 of file vgtl_graph.h.

9.32.2.8 typedef reverse_iterator<const_iterator> __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::const_reverse_iterator [inherited]

the const reverse iterator

Reimplemented from __Tree_t.

Definition at line 1266 of file vgtl_graph.h.

9.32.2.9 typedef std::reverse_iterator<const_iterator> __ITree::const_reverse_iterator [inherited]

the const reverse iterator

Definition at line 2069 of file vgtl_tree.h.

9.32.2.10 typedef _Tree_walker<_Tp,const _Tp&,const _Tp*,container_type,container_iterator> __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::const_walker [inherited]

the (recursive) const walker

Reimplemented from __Tree_t.

Definition at line 1278 of file vgtl_graph.h.

9.32.2.11 typedef ptrdiff_t __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::difference_type [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1255 of file vgtl_graph.h.

9.32.2.12 typedef _Tree_walker<_Tp,_Tp&,_Tp*,container_type,children_iterator,_Node> __ITree::iterative_walker [inherited]

the iterative walker

Definition at line 2063 of file vgtl_tree.h.

9.32.2.13 typedef _Tree_iterator<_Tp,_Tp&,_Tp*,container_type,container_iterator> __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::iterator [inherited]

the iterator

Reimplemented from __Tree_t.

Definition at line 1262 of file vgtl_graph.h.

9.32.2.14 typedef _Tree_iterator<_Tp,_Tp&,_Tp*,container_type,children_iterator,node_type> __ITree::iterator [inherited]

the iterator

Definition at line 2058 of file vgtl_tree.h.

9.32.2.15 typedef _Node __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::node_type [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1249 of file vgtl_graph.h.

9.32.2.16 template<class _Tp, class _Ctr, class _Tl, class _Node, class _Alloc> typedef __one_iterator<void *> _Tree_base< _Tp, _Ctr, _Tl, _Node, _Alloc >::parents_iterator [inherited]

iterator for accessing the parents

Reimplemented in __Tree_t.

Definition at line 1447 of file vgtl_tree.h.

9.32.2.17 typedef __one_iterator<void *> __Tree_t::parents_iterator [inherited]

iterator for accessing the parents

Reimplemented from _Tree_base< _Tp, _Ctr, _Iterator, _Node, _Alloc >.

Definition at line 1564 of file vgtl_tree.h.

9.32.2.18 typedef value_type* __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::pointer [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1250 of file vgtl_graph.h.

9.32.2.19 typedef value_type& __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::reference [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1252 of file vgtl_graph.h.

9.32.2.20 typedef reverse_iterator<iterator> __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::reverse_iterator [inherited]

the reverse iterator

Reimplemented from __Tree_t.

Definition at line 1267 of file vgtl_graph.h.

9.32.2.21 typedef std::reverse_iterator<iterator> __ITree::reverse_iterator [inherited]

the reverse iterator

Definition at line 2071 of file vgtl_tree.h.

9.32.2.22 typedef size_t __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::size_type [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1254 of file vgtl_graph.h.

9.32.2.23 typedef _Tp __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::value_type [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1248 of file vgtl_graph.h.

9.32.2.24 typedef _Tree_walker<_Tp,_Tp&,_Tp*,container_type,container_iterator> __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::walker [inherited]

the (recursive) walker

Reimplemented from __Tree_t.

Definition at line 1277 of file vgtl_graph.h.

9.32.3 Member Function Documentation

9.32.3.1 _Node* __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::_C_create_node (const _Tp & __x) [inline, protected, inherited]

construct a new tree node containing data __x

Reimplemented from __Tree_t.

Definition at line 1295 of file vgtl_graph.h.

9.32.3.2 _Node* __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::_C_create_node ( ) [inline, protected, inherited]

construct a new tree node containing default data

Reimplemented from __Tree_t.

Definition at line 1308 of file vgtl_graph.h.

9.32.3.3 _Node* _Tree_alloc_base< _Tp, _Ctr, _Iterator , _Node, _Alloc , _IsStatic >::_C_get_node ( ) [inline, protected, inherited]

allocate a new node

Definition at line 1375 of file vgtl_tree.h.

9.32.3.4 void _Tree_alloc_base< _Tp, _Ctr, _Tl, _Node, _Alloc , _IsStatic >::_C_put_node (_Node * __p) [inline, protected, inherited]

deallocate a node

Definition at line 1378 of file vgtl_tree.h.

9.32.3.5 void _Tree_alloc_base< _Tp, _Ctr, _Iterator , _Node, _Alloc , _IsStatic >::_C_put_node (_Node * __p) [inline, protected, inherited]

deallocate a node

Definition at line 1378 of file vgtl_tree.h.

9.32.3.6 template<class _Tp, class _Ctr, class _Tl, class _Node, class _Allocator, bool _IsStatic> void _Tree_alloc_base< _Tp, _Ctr, _Tl, _Node, _Allocator, _IsStatic >::_C_put_node (_Node * __p) [inline, protected, inherited]

deallocate a node

Definition at line 1378 of file vgtl_tree.h.

9.32.3.7 void _Tree_alloc_base< _Tp, _Ctr, _Tl, _Alloc , _Alloc_traits< _Tp, _Alloc >::_S_instanceless , _IsStatic >::_C_put_node (_Alloc * __p) [inline, protected, inherited]

deallocate a node

Definition at line 1378 of file vgtl_tree.h.

9.32.3.8 template<class _Tp, class _Ctr, class _TI, class _Node, class _Alloc> template<class _Output_Iterator > void _Tree_base< _Tp, _Ctr, _TI, _Node, _Alloc >::add_all_children (_Output_Iterator *fi,* _Node * *_parent*) [inherited]

add all children to the parent _parent. fi is a iterator to the children container of the parent

9.32.3.9 void _Tree_base< _Tp, _Ctr, _Iterator, _Node, _Alloc >::add_all_children (_Output_Iterator *fi,* _Node * *_parent*) [inline, inherited]

add all children to the parent _parent. fi is a iterator to the children container of the parent

Definition at line 1539 of file vgtl_tree.h.

9.32.3.10 iterator __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::begin ( ) [inline, inherited]

return an iterator to the first node in walk

Definition at line 1964 of file vgtl_tree.h.

9.32.3.11 const_iterator __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::begin ( ) const [inline, inherited]

return a const iterator to the first node in walk

Definition at line 1973 of file vgtl_tree.h.

9.32.3.12 iterative_walker __ITree::begin (walker_type *wt* = cw_pre_post, bool *front_to_back* = true, bool *depth_first* = true) [inline, inherited]

the walker to the first node of the complete walk

Definition at line 2122 of file vgtl_tree.h.

9.32.3.13 const_iterative_walker __ITree::begin (walker_type *wt* = cw_pre_post, bool *front_to_back* = true, bool *depth_first* = true) const [inline, inherited]

the const walker to the first node of the complete walk

Definition at line 2129 of file vgtl_tree.h.

9.32.3.14 void _Tree_base< _Tp, _Ctr, _Iterator , _Node, _Alloc >::clear_children ( ) [inline, inherited]

clear all children of the root node

Definition at line 1466 of file vgtl_tree.h.

9.32.3.15 size_type __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::depth (const walker & *__position*) [inline, inherited]

Reimplemented from __Tree_t.

Definition at line 1526 of file vgtl_graph.h.

9.32.3.16 size_type __ITree::depth (const iterative_walker & *__position*) [inline, inherited]

return the depth of this __position in the tree

Definition at line 2177 of file vgtl_tree.h.

9.32.3.17 bool __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::empty ( ) const [inline, inherited]

is the tree empty?

Reimplemented from __Tree_t.

Definition at line 1392 of file vgtl_graph.h.

9.32.3.18 iterator __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::end ( ) [inline, inherited]

return an iterator beyond the last node in walk

Definition at line 1968 of file vgtl_tree.h.

9.32.3.19 const_iterator __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::end ( ) const [inline, inherited]

return a const iterator beyond the last node in walk

Definition at line 1977 of file vgtl_tree.h.

9.32.3.20 iterative_walker __ITree::end (walker_type *wt* = cw_pre_post, bool *front_to_back* = true, bool *depth_first* = true) [inline, inherited]

the walker beyond the last node of the walk

Definition at line 2137 of file vgtl_tree.h.

9.32.3.21 const_iterative_walker __ITree::end (walker_type *wt* = cw_pre_post, bool *front_to_back* = true, bool *depth_first* = true) const [inline, inherited]

the const walker beyond the last node of the walk

Definition at line 2143 of file vgtl_tree.h.

9.32.3.22 void __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::erase (const __walker_base & *__position*) [inline, inherited]

erase the node at position __position.

Reimplemented from __Tree_t.

Definition at line 1444 of file vgtl_graph.h.

9.32.3.23 bool __Tree_t::erase_child (const __walker_base & *__position*, const children_iterator & *__it*) [inline, inherited]

erase the (leaf) child __It of node __position. This works if and only if the child is a leaf.

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >.

Definition at line 1770 of file vgtl_tree.h.

9.32.3.24 _Node* __Tree_t::erase_subtree (const __walker_base & __position, const children_iterator & __It) [inline, inherited]

erase the subtree position __position, whose top node is the child at children_iterator position __It, and return its top node.

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >.

Definition at line 1790 of file vgtl_tree.h.

9.32.3.25 _Node* __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::erase_tree (const __walker_base & __position) [inline, inherited]

erase the subtree starting at position __position, and return its top node.

Reimplemented from __Tree_t.

Definition at line 1471 of file vgtl_graph.h.

9.32.3.26 allocator_type __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::get_allocator ( ) const [inline, inherited]

construct an allocator object

Reimplemented from __Tree_t.

Definition at line 1259 of file vgtl_graph.h.

9.32.3.27 reference __ITree::getroot ( ) [inline, inherited]

get a reference to the virtual root node

Definition at line 2172 of file vgtl_tree.h.

9.32.3.28 const_reference __ITree::getroot ( ) const [inline, inherited]

get a const reference to the virtual root node

Definition at line 2174 of file vgtl_tree.h.

9.32.3.29 walker __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::ground ( ) [inline, inherited]

return a walker to the virtual root node.

Definition at line 1939 of file vgtl_tree.h.

9.32.3.30 const_walker __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::ground ( ) const [inline, inherited]

return a const walker to the virtual root node.

Definition at line 1943 of file vgtl_tree.h.

9.32.3.31 template<class _Tp , template< class __Key, class __Ty, class __Compare, class __AllocT > class _AssocCtr = std::multimap, class _Key = string, class _Compare = less<_Key>, class _PtrAlloc = __VGTL_DEFAULT_ALLOCATOR(void *), class _Alloc = __VGTL_DEFAULT_ALLOCATOR(_Tp)> void atree< _Tp, _AssocCtr, _Key, _Compare, _PtrAlloc, _Alloc >::insert (const __walker_base & *__position,* const _Tp & *__x,* const _Key & *__k*) [inline]

Insert a node with data __x and key _k at position __position.

Definition at line 2722 of file vgtl_tree.h.

9.32.3.32 template<class _Tp , template< class __Key, class __Ty, class __Compare, class __AllocT > class _AssocCtr = std::multimap, class _Key = string, class _Compare = less<_Key>, class _PtrAlloc = __VGTL_DEFAULT_ALLOCATOR(void *), class _Alloc = __VGTL_DEFAULT_ALLOCATOR(_Tp)> void atree< _Tp, _AssocCtr, _Key, _Compare, _PtrAlloc, _Alloc >::insert (const __walker_base & *__position,* const _Key & *__k*) [inline]

Insert a node with default data and key __k at position __position.

Definition at line 2748 of file vgtl_tree.h.

9.32.3.33 void __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::insert_child (const __walker_base & *__position,* const _Tp & *__x,* const container_insert_arg & *__It*) [inline, inherited]

add a child below __position with data __x, at the __It position in the __position - node's children container

Reimplemented from __Tree_t.

Definition at line 1409 of file vgtl_graph.h.

9.32.3.34 void __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::insert_child (const __walker_base & *__position,* const container_insert_arg & *__It*) [inline, inherited]

add a child below __position with default data, at the __It position in the __position - node's children container

Reimplemented from __Tree_t.

Definition at line 1415 of file vgtl_graph.h.

9.32.3.35 void __Tree_t::insert_children (const __walker_base & *__position,* size_type *__n,* const _Tp & *__x,* const children_iterator & *__It*) [inline, inherited]

add __n children below __position with data __x, after the __It position in the __position - node's children container

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >.

Definition at line 1682 of file vgtl_tree.h.

9.32.3.36 void __Tree_t::insert_subtree (const __walker_base & *__position,* _Self & *__subtree,* const children_iterator & *__It*) [inline, inherited]

add a complete subtree __subtree below position __position and children iterator position __It.

Definition at line 1702 of file vgtl_tree.h.

9.32.3.37 size_type __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::max_size ( ) const [inline, inherited]

return the maximum possible size of the tree (theor. infinity)

Reimplemented from __Tree_t.

Definition at line 1400 of file vgtl_graph.h.

9.32.3.38 template<class _Tp , template< class __Key, class __Ty, class __Compare, class __AllocT > class _AssocCtr = std::multimap, class _Key = string, class _Compare = less<_Key>, class _PtrAlloc = __VGTL_DEFAULT_ALLOCATOR(void *), class _Alloc = __VGTL_DEFAULT_ALLOCATOR(_Tp)> _Self& atree< _Tp, _AssocCtr, _Key, _Compare, _PtrAlloc, _Alloc >::operator= (_Node * __x) [inline]

assign a tree from one node -> make this node the root node. This is useful for making trees out of erased subtrees.

Reimplemented from __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >.

Definition at line 2713 of file vgtl_tree.h.

9.32.3.39 reverse_iterator __ITree::rbegin ( ) [inline, inherited]

return a reverse iterator to the first node in walk

Definition at line 2151 of file vgtl_tree.h.

9.32.3.40 const_reverse_iterator __ITree::rbegin ( ) const [inline, inherited]

return a const reverse iterator to the first node in walk

Definition at line 2158 of file vgtl_tree.h.

9.32.3.41 reverse_iterator __ITree::rend ( ) [inline, inherited]

return a reverse iterator beyond the last node in walk

Definition at line 2154 of file vgtl_tree.h.

9.32.3.42 const_reverse_iterator __ITree::rend ( ) const [inline, inherited]

return a const reverse iterator beyond the last node in walk

Definition at line 2161 of file vgtl_tree.h.

9.32.3.43 walker __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::root (children_iterator __it) [inline, inherited]

return a walker to a root node.

Definition at line 1947 of file vgtl_tree.h.

9.32.3.44 const_walker __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::root (children_iterator __it) const [inline, inherited]

return a const walker to a root node.

Definition at line 1952 of file vgtl_tree.h.

9.32.3.45 walker __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::root ( ) [inline, inherited]

return a walker to the first non-virtual tree root

Definition at line 1957 of file vgtl_tree.h.

9.32.3.46 const_walker __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::root ( ) const [inline, inherited]

return a const walker to the first non-virtual tree root

Definition at line 1960 of file vgtl_tree.h.

9.32.3.47 iterative_walker __ITree::root (walker_type *wt* = cw_pre_post, bool *front_to_back* = true, bool *depth_first* = true) [inline, inherited]

return an iterative walker of type `wt` to the ground node

Definition at line 2099 of file vgtl_tree.h.

9.32.3.48 const_iterative_walker __ITree::root (walker_type *wt* = cw_pre_post, bool *front_to_back* = true, bool *depth_first* = true) const [inline, inherited]

return a const iterative walker of type `wt` to the ground node

Definition at line 2106 of file vgtl_tree.h.

9.32.3.49 size_type __ITree::size ( ) const [inline, inherited]

return the size of the tree (# of nodes)

Definition at line 2165 of file vgtl_tree.h.

9.32.3.50 void __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::swap (_Self & __x) [inline, inherited]

Reimplemented from __Tree_t.

Definition at line 1405 of file vgtl_graph.h.

9.32.3.51 iterative_walker __ITree::through ( ) [inline, inherited]

the walker beyond the complete walk

Definition at line 2113 of file vgtl_tree.h.

9.32.3.52 const_iterative_walker __ITree::through ( ) const [inline, inherited]

the const walker beyond the complete walk

Definition at line 2117 of file vgtl_tree.h.

9.32.4 Friends And Related Function Documentation

9.32.4.1 bool operator==__VGTL_NULL_TMPL_ARGS (const __ITree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc > & __x, const __ITree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc > & __y) [friend, inherited]

comparison operator

9.32.5 Member Data Documentation

9.32.5.1 _Node* _Tree_alloc_base< _Tp, _Ctr, _Iterator , _Node, _Alloc , _IsStatic >::_C_node [protected, inherited]

This is the node

Definition at line 1387 of file vgtl_tree.h.

The documentation for this class was generated from the following files:

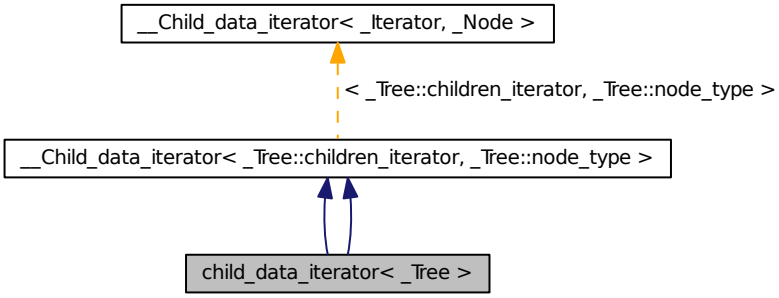
- vgtl_graph.h
- vgtl_tree.h

9.33 child_data_iterator< _Tree > Class Template Reference

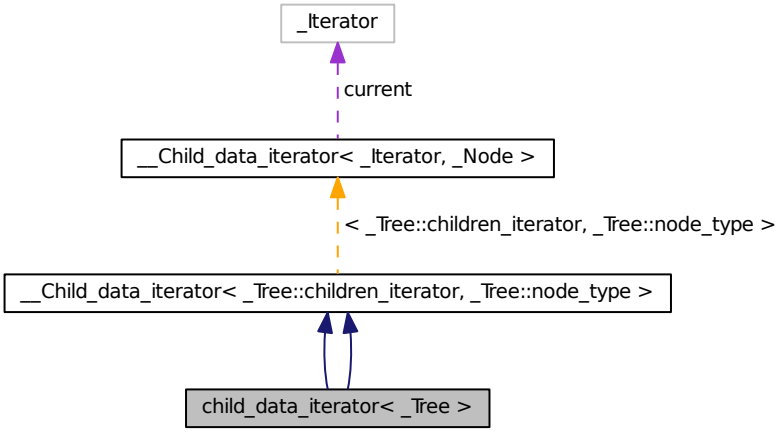
Iterator which iterates through the data hooks of all children.

`#include <vgtl_algo.h>`

Inheritance diagram for child_data_iterator< _Tree >:



Collaboration diagram for child_data_iterator< _Tree >:



Public Types

- typedef ctree_data_hook value_type
- typedef value_type * pointer
- typedef value_type & reference

- typedef ctree_data_hook value_type
- typedef value_type * pointer
- typedef value_type & reference

Public Member Functions

- child_data_iterator ()
  standard constructor
- child_data_iterator (const _Self &__x)
  constructor presetting the position
- _Self & operator= (const iterator_type &it)
  assignment operator for setting the position
- child_data_iterator ()
  standard constructor
- child_data_iterator (iterator_type __x)
  constructor presetting the position
- child_data_iterator (const _Self &__x)
  copy constructor

- _Self & operator= (const iterator_type &it)

  assignment operator for setting the position
- iterator_type base () const

  return the 'unwrapped' iterator
- iterator_type base () const

  return the 'unwrapped' iterator
- reference operator* () const

  dereference to the `data_hook`.
- reference operator* () const

  dereference to the `data_hook`.

- bool operator== (const _Self &__x) const

  standard comparison operator
- bool operator!= (const _Self &__x) const

  standard comparison operator

- bool operator== (const _Self &__x) const

  standard comparison operator
- bool operator!= (const _Self &__x) const

  standard comparison operator

- _Self & operator++ ()

  standard in(de)crement operator
- _Self & operator++ (int)

  standard in(de)crement operator
- _Self & operator-- ()

  standard in(de)crement operator
- _Self & operator-- (int)

  standard in(de)crement operator

- _Self & operator++ ()

  standard in(de)crement operator
- _Self & operator++ (int)

  standard in(de)crement operator
- _Self & operator-- ()

  standard in(de)crement operator
- _Self & operator-- (int)

  standard in(de)crement operator

- _Self operator+ (difference_type __n) const
  additional operator for random access iterators
- _Self & operator+= (difference_type __n)
  additional operator for random access iterators
- _Self operator- (difference_type __n) const
  additional operator for random access iterators
- _Self & operator-= (difference_type __n)
  additional operator for random access iterators
- reference operator[ ] (difference_type __n) const
  additional operator for random access iterators

- _Self operator+ (difference_type __n) const
  additional operator for random access iterators
- _Self & operator+= (difference_type __n)
  additional operator for random access iterators
- _Self operator- (difference_type __n) const
  additional operator for random access iterators
- _Self & operator-= (difference_type __n)
  additional operator for random access iterators
- reference operator[ ] (difference_type __n) const
  additional operator for random access iterators

Protected Attributes

- _Tree::children_iterator current
  that's where we are

9.33.1 Detailed Description

template<class _Tree>class child_data_iterator< _Tree >

This class defines an iterator for iterating through all data hooks of a node's children.

Definition at line 156 of file vgtl_lalgo.h.

9.33.2 Member Typedef Documentation

9.33.2.1 typedef value_type* __Child_data_iterator< _Tree::children_iterator , _Tree::node_type >::pointer [inherited]

standard iterator definitions

Definition at line 64 of file vgtl_lalgo.h.

9.33.2.2 typedef value_type* __Child_data_iterator< _Tree::children_iterator , _Tree::node_type >::pointer [inherited]

standard iterator definitions

Definition at line 64 of file vgtl_algo.h.

9.33.2.3 typedef value_type& __Child_data_iterator< _Tree::children_iterator , _Tree::node_type >::reference [inherited]

standard iterator definitions

Definition at line 65 of file vgtl_lalgo.h.

9.33.2.4 typedef value_type& __Child_data_iterator< _Tree::children_iterator , _Tree::node_type >::reference [inherited]

standard iterator definitions

Definition at line 65 of file vgtl_algo.h.

9.33.2.5 typedef ctree_data_hook __Child_data_iterator< _Tree::children_iterator , _Tree::node_type >::value_type [inherited]

standard iterator definitions

Definition at line 63 of file vgtl_lalgo.h.

9.33.2.6 typedef ctree_data_hook __Child_data_iterator< _Tree::children_iterator , _Tree::node_type >::value_type [inherited]

standard iterator definitions

Definition at line 63 of file vgtl_algo.h.

9.33.3 Constructor & Destructor Documentation

9.33.3.1 template<class _Tree> child_data_iterator< _Tree >::child_data_iterator (const _Self & __x) [inline]

copy constructor

Definition at line 174 of file vgtl_algo.h.

The documentation for this class was generated from the following files:

- vgtl_algo.h
- vgtl_lalgo.h

9.34 dag Class Reference

unlabeled directed acyclic graph (DAG)

`#include <vgtl_dag.h>`

Inheritance diagram for dag:

Collaboration diagram for dag:

Public Types

- typedef _Base::walker walker
- typedef _Base::const_walker const_walker
- typedef _Base::children_iterator children_iterator
- typedef _Base::parents_iterator parents_iterator
- typedef _Base::children_const_iterator children_const_iterator
- typedef _Base::parents_const_iterator parents_const_iterator
- typedef _Base::erased_part erased_part

Public Member Functions

- dag (const allocator_type &__a=allocator_type())
- dag (const _Self &__dag)
- dag (const _Base &__dag)
- dag (const erased_part &__ep)
- bool check_acyclicity (const walker &__parent, const walker &__child)
- _Self & operator= (const _RV_DG &__rl)
- _Self & operator= (const erased_part &__ep)
- void clear ()
- walker between (const walker &__parent, const children_iterator &__cit, const walker &__child, const parents_iterator &__pit, const _Tp &__x)
- walker between (const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children, const _Tp &__x)
- walker between (const walker &__parent, const children_iterator &__cit, const __SequenceCtr< walker, _Allocator > &__children, const _Tp &__x)
- walker between (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const parents_iterator &__pit, const _Tp &__x)
- walker split (const walker &__parent, const children_iterator &__ch_it, const walker &__child, const parents_iterator &__pa_it, const _Tp &__x)
- walker split (const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children, const _Tp &__x)
- walker split (const walker &__parent, const children_iterator &__ch_it, const __SequenceCtr< walker, _Allocator > &__children, const _Tp &__x)
- walker split (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const parents_iterator &__pr_it, const _Tp &__x)
- walker between_back (const walker &__parent, const walker &__child, const _Tp &__x)
- walker between_back (const walker &__parent, const __SequenceCtr< walker, _Allocator > &__children, const _Tp &__x)
- walker between_back (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const _Tp &__x)
- walker split_back (const walker &__parent, const walker &__child, const _Tp &__x)
- walker split_back (const walker &__parent, const __SequenceCtr< walker, _Allocator > &__children, const _Tp &__x)
- walker split_back (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const _Tp &__x)

- walker between_front (const walker &__parent, const walker &__child, const _Tp &__x)
- walker between_front (const walker &__parent, const __SequenceCtr< walker, _Allocator > &__children, const _Tp &__x)
- walker between_front (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const _Tp &__x)
- walker split_front (const walker &__parent, const walker &__child, const _Tp &__x)
- walker split_front (const walker &__parent, const __SequenceCtr< walker, _Allocator > &__children, const _Tp &__x)
- walker split_front (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const _Tp &__x)
- void insert_subgraph (_Self &__subgraph, const walker &__parent, const children_iterator &__ch_it, const walker &__child, const parents_iterator &__pa_it)
- void insert_back_subgraph (_Self &__subgraph, const walker &__parent, const walker &__child)
- void insert_front_subgraph (_Self &__subgraph, const walker &__parent, const walker &__child)
- void add_edge (const walker &__parent, const children_iterator &__ch_it, const walker &__child, const parents_iterator &__pa_it)
- void add_edge_back (const walker &__parent, const walker &__child)
- void add_edge_front (const walker &__parent, const walker &__child)

9.34.1 Detailed Description

This class constructs an unlabeled directed acyclic graph (DAG). By default, the children and the parents are collected in an STL vector, but the container can be replaced by any other sequential container.

9.34.2 Member Typedef Documentation

9.34.2.1 typedef _Base::children_const_iterator dag::children_const_iterator

the children const iterator

Reimplemented from dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

Definition at line 2673 of file vgtl_dag.h.

9.34.2.2 typedef _Base::children_iterator dag::children_iterator

the children iterator

Reimplemented from dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

Definition at line 2669 of file vgtl_dag.h.

9.34.2.3 typedef _Base::const_walker dag::const_walker

the const walker

Reimplemented from dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

Definition at line 2667 of file vgtl_dag.h.

9.34.2.4 typedef _Base::erased_part dag::erased_part

the erased part constructed in erasing subgraphs

Reimplemented from dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

Definition at line 2678 of file vgtl_dag.h.

9.34.2.5 typedef _Base::parents_const_iterator dag::parents_const_iterator

the parents const iterator

Reimplemented from dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

Definition at line 2675 of file vgtl_dag.h.

9.34.2.6 typedef _Base::parents_iterator dag::parents_iterator

the parents iterator

Reimplemented from dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

Definition at line 2671 of file vgtl_dag.h.

9.34.2.7 typedef _Base::walker dag::walker

the walker

Reimplemented from dgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

Definition at line 2665 of file vgtl_dag.h.

9.34.3 Constructor & Destructor Documentation

9.34.3.1 dag::dag (const allocator_type & __a = allocator_type()) [inline, explicit]

standard constructor

Definition at line 2682 of file vgtl_dag.h.

9.34.3.2 dag::dag (const _Self & __dag) [inline]

copy constructor

Definition at line 2685 of file vgtl_dag.h.

9.34.3.3 dag::dag (const _Base & __dag) [inline]

construct dag from directed graph

Definition at line 2691 of file vgtl_dag.h.

9.34.3.4 dag::dag (const erased_part & __ep) [inline]

construct dag from erased part

Definition at line 2699 of file vgtl_dag.h.

9.34.4 Member Function Documentation

9.34.4.1 void dgraph::add_edge (const walker & __parent, const children_iterator & __ch_it, const walker & __child, const parents_iterator & __pa_it) [inline, inherited]

add an edge between __parent and __child at specific positions __ch_it and __pa_it.

Definition at line 2389 of file vgtl_dag.h.

9.34.4.2 void dgraph::add_edge_back (const walker & __parent, const walker & __child) [inline, inherited]

add an edge between __parent and __child at the end of the children and parents containers.

Definition at line 2399 of file vgtl_dag.h.

9.34.4.3 void dgraph::add_edge_front (const walker & __parent, const walker & __child) [inline, inherited]

add an edge between __parent and __child at the beginning of the children and parents containers.

Definition at line 2409 of file vgtl_dag.h.

9.34.4.4 walker dgraph::between (const walker & __parent, const children_iterator & __cit, const walker & __child, const parents_iterator & __pit, const _Tp & __x) [inline, inherited]

here a new node is inserted between a parent node and a child node but the previous bonds between the two are not broken, the node is always new with data __x.

Definition at line 2177 of file vgtl_dag.h.

9.34.4.5 walker dgraph::between (const __SequenceCtr1< walker, _Allocator1 > & __parents, const __SequenceCtr2< walker, _Allocator2 > & __children, const _Tp & __x) [inline, inherited]

here a new node is inserted between many parents and many children but the previous bonds are not broken, the node is always new

Definition at line 2279 of file vgtl_dag.h.

9.34.4.6 walker dgraph::between (const walker & __parent, const children_iterator & __cit, const __SequenceCtr< walker, _Allocator > & __children, const _Tp & __x) [inline, inherited]

here a new node is inserted between one parent and many children but the previous bonds are not broken, the node is always new

Definition at line 2425 of file vgtl_dag.h.

9.34.4.7 walker dgraph::between (const __SequenceCtr< walker, _Allocator > & __parents, const walker & __child, const parents_iterator & __pit, const _Tp & __x) [inline, inherited]

here a new node is inserted between many parents and one child but the previous bonds are not broken, the node is always new

Definition at line 2525 of file vgtl_dag.h.

9.34.4.8 walker dgraph::between_back (const walker & __parent, const walker & __child, const _Tp & __x) [inline, inherited]

insert the node as the last child between parent and child, without breaking old bonds.

Definition at line 2212 of file vgtl_dag.h.

9.34.4.9 walker dgraph::between_back (const walker & __parent, const __SequenceCtr< walker, _Allocator > & __children, const _Tp & __x) [inline, inherited]

here a new node is inserted between one parent and many children but the previous bonds are not broken, the node is always new. At the parent the new child is put last.

Definition at line 2480 of file vgtl_dag.h.

9.34.4.10 walker dgraph::between_back (const __SequenceCtr< walker, _Allocator > & __parents, const walker & __child, const _Tp & __x) [inline, inherited]

here a new node is inserted between many parents and one child but the previous bonds are not broken, the node is always new. At the child the new parent is put last.

Definition at line 2579 of file vgtl_dag.h.

9.34.4.11 walker dgraph::between_front (const walker & __parent, const walker & __child, const _Tp & __x) [inline, inherited]

Here the inserted node is the first child of its parent and first parent of its child. Insert the node without breaking old bonds.

Definition at line 2243 of file vgtl_dag.h.

9.34.4.12 walker dgraph::between_front (const walker & __parent, const __SequenceCtr< walker, _Allocator > & __children, const _Tp & __x) [inline, inherited]

here a new node is inserted between one parent and many children but the previous bonds are not broken, the node is always new. At the parent the new child is put first.

Definition at line 2510 of file vgtl_dag.h.

9.34.4.13 walker dgraph::between_front (const __SequenceCtr< walker, _Allocator > & __parents, const walker & __child, const _Tp & __x) [inline, inherited]

here a new node is inserted between many parents and one child but the previous bonds are not broken, the node is always new. At the child the new parent is put first.

Definition at line 2607 of file vgtl_dag.h.

9.34.4.14 bool dag::check_acyclicity (const walker & __parent, const walker & __child) [inline]

This method checks, whether the dag is indeed acyclic. This is NYI!

Definition at line 2722 of file vgtl_dag.h.

9.34.4.15 void dgraph::clear ( ) [inline, inherited]

empty the graph

Definition at line 2170 of file vgtl_dag.h.

9.34.4.16 void dgraph::insert_back_subgraph (_Self & __subgraph, const walker & __parent, const walker & __child) [inline, inherited]

here a subgraph is inserted between a parent and a child, at the end of the children resp. parents lists.

Definition at line 2348 of file vgtl_dag.h.

9.34.4.17 void dgraph::insert_front_subgraph (_Self & __subgraph, const walker & __parent, const walker & __child) [inline, inherited]

here a subgraph is inserted between a parent and a child, at the front of the children resp. parents lists.

Definition at line 2361 of file vgtl_dag.h.

9.34.4.18 void dgraph::insert_subgraph (_Self & __subgraph, const walker & __parent, const children_iterator & __ch_it, const walker & __child, const parents_iterator & __pa_it) [inline, inherited]

here a subgraph is inserted between a parent and a child, at specific positions ___ch_it and ___pa_it.

Definition at line 2337 of file vgtl_dag.h.

9.34.4.19 _Self& dag::operator= (const _RV_DG & __rl) [inline]

assignment from part of an erased part

Definition at line 2738 of file vgtl_dag.h.

9.34.4.20 _Self& dag::operator= (const erased_part & __ep) [inline]

assignment from erased part

Definition at line 2746 of file vgtl_dag.h.

9.34.4.21 walker dgraph::split (const walker & __parent, const children_iterator & __ch_it, const walker & __child, const parents_iterator & __pa_it, const _Tp & __x) [inline, inherited]

here a new node is inserted between a parent node and a child node and the previous bonds between them are broken, the node is always new with data ___x.

Definition at line 2190 of file vgtl_dag.h.

9.34.4.22 walker dgraph::split (const __SequenceCtr1< walker, _Allocator1 > & __parents, const __SequenceCtr2< walker, _Allocator2 > & __children, const _Tp & __x) [inline, inherited]

here a new node is inserted between many parents and many children, and the previous bonds are broken, the node is always new.

Definition at line 2311 of file vgtl_dag.h.

9.34.4.23 walker dgraph::split (const walker & __parent, const children_iterator & __ch_it, const __SequenceCtr< walker, _Allocator > & __children, const _Tp & __x) [inline, inherited]

here a new node is inserted between one parent and many children, and the previous bonds are broken, the node is always new.

Definition at line 2438 of file vgtl_dag.h.

9.34.4.24 walker dgraph::split (const __SequenceCtr< walker, _Allocator > & __parents, const walker & __child, const parents_iterator & __pr_it, const _Tp & __x) [inline, inherited]

here a new node is inserted between many parents and one child, and the previous bonds are broken, the node is always new.

Definition at line 2538 of file vgtl_dag.h.

9.34.4.25 walker dgraph::split_back (const walker & *__parent,* const walker & *__child,* const _Tp & *__x*) [inline, inherited]

insert the node as the last child between parent and child, with breaking old bonds.

Definition at line 2225 of file vgtl_dag.h.

9.34.4.26 walker dgraph::split_back (const walker & *__parent,* const __SequenceCtr< walker, _Allocator > & *__children,* const _Tp & *__x*) [inline, inherited]

here a new node is inserted between one parent and many children, and the previous bonds are broken, the node is always new. At the parent the new child is put last.

Definition at line 2465 of file vgtl_dag.h.

9.34.4.27 walker dgraph::split_back (const __SequenceCtr< walker, _Allocator > & *__parents,* const walker & *__child,* const _Tp & *__x*) [inline, inherited]

here a new node is inserted between many parents and one child, and the previous bonds are broken, the node is always new. At the child the new parent is put last.

Definition at line 2565 of file vgtl_dag.h.

9.34.4.28 walker dgraph::split_front (const walker & *__parent,* const walker & *__child,* const _Tp & *__x*) [inline, inherited]

Here the inserted node is the first child of its parent and first parent of its child. Insert the node and break old bonds.

Definition at line 2256 of file vgtl_dag.h.

9.34.4.29 walker dgraph::split_front (const walker & *__parent,* const __SequenceCtr< walker, _Allocator > & *__children,* const _Tp & *__x*) [inline, inherited]

here a new node is inserted between one parent and many children, and the previous bonds are broken, the node is always new. At the parent the new child is put first.

Definition at line 2495 of file vgtl_dag.h.

9.34.4.30 walker dgraph::split_front (const __SequenceCtr< walker, _Allocator > & *__parents,* const walker & *__child,* const _Tp & *__x*) [inline, inherited]

here a new node is inserted between many parents and one child, and the previous bonds are broken, the node is always new. At the child the new parent is put first.

Definition at line 2593 of file vgtl_dag.h.

The documentation for this class was generated from the following file:

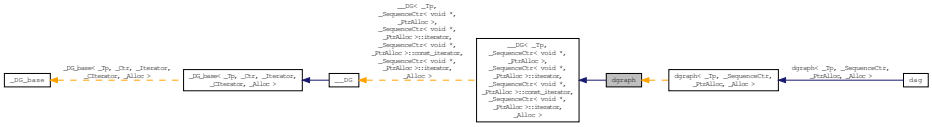
- vgtl_dag.h

9.35 dgraph Class Reference

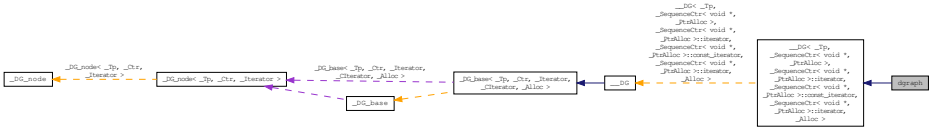
unlabeled directed graph

```
#include <vgtl_dag.h>
```

Inheritance diagram for dgraph:



Collaboration diagram for dgraph:



Public Types

- typedef _Base::walker walker
- typedef _Base::const_walker const_walker
- typedef _Base::children_iterator children_iterator
- typedef _Base::parents_iterator parents_iterator
- typedef _Base::parents_const_iterator parents_const_iterator
- typedef _Base::children_const_iterator children_const_iterator
- typedef _DG_iterator< _Tp, _Tp &, _Tp *, container_type, children_iterator, children_const_iterator > iterator
- typedef _DG_iterator< _Tp, const _Tp &, const _Tp *, container_type, children_iterator, children_const_iterator > const_iterator
- typedef std::reverse_iterator < const_iterator > const_reverse_iterator
- typedef std::reverse_iterator < iterator > reverse_iterator
- typedef std::pair< walker, walker > edge
- typedef std::pair< edge, bool > enhanced_edge

- typedef _Tp value_type
- typedef _Node node_type
- typedef value_type * pointer
- typedef const value_type * const_pointer
- typedef value_type & reference
- typedef const value_type & const_reference
- typedef size_t size_type
- typedef ptrdiff_t difference_type

Public Member Functions

- dgraph (const allocator_type &__a=allocator_type())
- dgraph (const _Self &__dg)
- dgraph (const erased_part &__ep, const allocator_type &__a=allocator_type())
- void clear ()

- walker between (const walker &__parent, const children_iterator &__cit, const walker &__child, const parents_iterator &__pit, const _Tp &__x)
- walker split (const walker &__parent, const children_iterator &__ch_it, const walker &__child, const parents_iterator &__pa_it, const _Tp &__x)
- walker between_back (const walker &__parent, const walker &__child, const _Tp &__x)
- walker split_back (const walker &__parent, const walker &__child, const _Tp &__x)
- walker between_front (const walker &__parent, const walker &__child, const _Tp &__x)
- walker split_front (const walker &__parent, const walker &__child, const _Tp &__x)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr1, template< class __Tp, class __AllocTp > class __SequenceCtr2, class _Allocator1 , class _Allocator2 >
  walker between (const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children, const _Tp &__x)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr1, template< class __Tp, class __AllocTp > class __SequenceCtr2, class _Allocator1 , class _Allocator2 >
  walker split (const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children, const _Tp &__x)
- void insert_subgraph (_Self &__subgraph, const walker &__parent, const children_iterator &__ch_it, const walker &__child, const parents_iterator &__pa_it)
- void insert_back_subgraph (_Self &__subgraph, const walker &__parent, const walker &__child)
- void insert_front_subgraph (_Self &__subgraph, const walker &__parent, const walker &__child)
- void add_edge (const walker &__parent, const children_iterator &__ch_it, const walker &__child, const parents_iterator &__pa_it)
- void add_edge_back (const walker &__parent, const walker &__child)
- void add_edge_front (const walker &__parent, const walker &__child)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker between (const walker &__parent, const children_iterator &__cit, const __SequenceCtr< walker, _Allocator > &__children, const _Tp &__x)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker split (const walker &__parent, const children_iterator &__ch_it, const __SequenceCtr< walker, _Allocator > &__children, const _Tp &__x)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker split_back (const walker &__parent, const __SequenceCtr< walker, _Allocator > &__children, const _Tp &__x)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker between_back (const walker &__parent, const __SequenceCtr< walker, _Allocator > &__children, const _Tp &__x)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker split_front (const walker &__parent, const __SequenceCtr< walker, _Allocator > &__children, const _Tp &__x)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker between_front (const walker &__parent, const __SequenceCtr< walker, _Allocator > &__children, const _Tp &__x)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker between (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const parents_iterator &__pit, const _Tp &__x)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker split (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const parents_iterator &__pr_it, const _Tp &__x)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker split_back (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const _Tp &__x)

- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker between_back (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const _Tp &__x)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker split_front (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const _Tp &__x)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker between_front (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const _Tp &__x)
- _Self & operator= (const _RV_DG &__rl)
- _Self & operator= (const erased_part &__ep)
- allocator_type get_allocator () const
- walker ground ()
- const_walker ground () const
- walker sky ()
- const_walker sky () const
- children_iterator root_begin ()
- children_const_iterator root_begin () const
- children_iterator root_end ()
- children_const_iterator root_end () const
- parents_iterator leaf_begin ()
- parents_const_iterator leaf_begin () const
- parents_iterator leaf_end ()
- parents_const_iterator leaf_end () const
- bool empty () const
- size_type size () const
- size_type max_size () const
- void swap (_Self &__x)
- walker insert_node_in_graph (_Node *__n, const walker &__parent, const walker &__child, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- walker insert_node_in_graph (_Node *__node, const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children)
- walker insert_node_in_graph (_Node *__node, const walker &__parent, const container_insert_arg &__pref, const __SequenceCtr< walker, _Allocator > &__children)
- walker insert_node_in_graph (_Node *__node, const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const container_insert_arg &__cref)
- walker insert_in_graph (const _Tp &__x, const walker &__parent, const walker &__child, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- walker insert_in_graph (const walker &__parent, const walker &__child, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- walker insert_in_graph (const _Tp &__x, const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children)
- walker insert_in_graph (const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children)
- walker insert_in_graph (const _Tp &__x, const walker &__parent, const container_insert_arg &__pref, const __SequenceCtr< walker, _Allocator > &__children)
- walker insert_in_graph (const walker &__parent, const container_insert_arg &__pref, const __SequenceCtr< walker, _Allocator > &__children)
- walker insert_in_graph (const _Tp &__x, const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const container_insert_arg &__cref)
- walker insert_in_graph (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const container_insert_arg &__cref)

- void insert_subgraph (_Self &__subgraph, const walker &__parent, const walker &__child, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- void insert_subgraph (_Self &__subgraph, const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children)
- void add_edge (const edge &__edge, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- void add_edge (const walker &__parent, const walker &__child, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- void replace_edge_to_child (const walker &__parent, const walker &__child_old, const walker &__child_new)
- void replace_edge_to_parent (const walker &__parent_old, const walker &__parent_new, const walker &__child)
- void remove_edge (const edge &__edge)
- void remove_edge (const walker &__parent, const walker &__child)
- void remove_edge_and_deattach (const walker &__parent, const walker &__child)
- void sort_child_edges (walker __position, children_iterator first, children_iterator last, Compare comp)
- void sort_child_edges (walker __position, Compare comp)
- void sort_parent_edges (walker __position, parents_iterator first, parents_iterator last, Compare comp)
- void sort_parent_edges (walker __position, Compare comp)
- walker insert_node (_Node *_node, const walker &__position, const container_insert_arg &__It)
- walker insert_node (const _Tp &__x, const walker &__position, const container_insert_arg &__It)
- walker insert_node (const walker &__position, const container_insert_arg &__It)
- walker insert_node_before (_Node *_node, const walker &__position, const container_insert_arg &__It)
- void insert_node_before (const _Tp &__x, const walker &__position, const container_insert_arg &__It)
- void insert_node_before (const walker &__position, const container_insert_arg &__It)
- void merge (const walker &__position, const walker &__second, bool merge_parent_edges=true, bool merge_child_edges=true)
- void erase (const walker &__position)
- void partial_erase_to_parent (const walker &__position, const walker &__parent, unsigned int idx)
- void clear_erased_part (erased_part &_ep)
- erased_part erase_maximal_subgraph (const walker &__position)
- erased_part erase_maximal_subgraph (const __SequenceCtr< walker, _Allocator > &__positions)
- erased_part erase_minimal_subgraph (const walker &__position)
- erased_part erase_minimal_subgraph (const __SequenceCtr< walker, _Allocator > &__positions)
- erased_part erase_maximal_pregraph (const walker &__position)
- erased_part erase_maximal_pregraph (const __SequenceCtr< walker, _Allocator > &__positions)
- erased_part erase_minimal_pregraph (const walker &__position)
- erased_part erase_minimal_pregraph (const __SequenceCtr< walker, _Allocator > &__positions)
- bool erase_child (const walker &__position, const children_iterator &__It)
- bool erase_parent (const walker &__position, const parents_iterator &__It)
- void copy_maximal_subgraph (const walker &__x, const walker &__par, const walker &__bo, const walker &__bn)

Protected Types

- typedef _Base::erased_part erased_part

Protected Member Functions

- _Node * _C_create_node (const _Tp &__x)
- _Node * _C_create_node ()
- void _C_destroy_node (_Node *__p)

9.35.1 Detailed Description

This class constructs an unlabeled directed graph. By default, the children and the parents are collected in an STL vector, but the container can be replaced by any other sequential container.

9.35.2 Member Typedef Documentation

9.35.2.1 typedef _Base::children_const_iterator dgraph::children_const_iterator

the children const iterator

Reimplemented from __DG< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::const_iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >.

Reimplemented in dag.

Definition at line 2150 of file vgtl_dag.h.

9.35.2.2 typedef _Base::children_iterator dgraph::children_iterator

the children iterator

Reimplemented from __DG< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::const_iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >.

Reimplemented in dag.

Definition at line 2144 of file vgtl_dag.h.

9.35.2.3 typedef _DG_iterator<_Tp,const _Tp&,const _Tp*,container_type, children_iterator,children_const_iterator> __DG::const_iterator [inherited]

the const iterator

Definition at line 600 of file vgtl_dag.h.

9.35.2.4 typedef const value_type* __DG::const_pointer [inherited]

standard typedef

Definition at line 583 of file vgtl_dag.h.

9.35.2.5 typedef const value_type& __DG::const_reference [inherited]

standard typedef

Definition at line 585 of file vgtl_dag.h.

9.35.2.6 typedef std::reverse_iterator<const_iterator> __DG::const_reverse_iterator [inherited]

the const reverse iterator

Definition at line 604 of file vgtl_dag.h.

9.35.2.7 typedef _Base::const_walker dgraph::const_walker

the const walker

Reimplemented from __DG< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::const_iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >.

Reimplemented in dag.

Definition at line 2142 of file vgtl_dag.h.

9.35.2.8 typedef ptrdiff_t __DG::difference_type [inherited]

standard typedef

Definition at line 587 of file vgtl_dag.h.

9.35.2.9 typedef std::pair<walker,walker> __DG::edge [inherited]

an edge of the graph (parent, child)

Definition at line 626 of file vgtl_dag.h.

9.35.2.10 typedef std::pair<edge,bool> __DG::enhanced_edge [inherited]

an edge with additiona information about erased ground/sky edges

Definition at line 628 of file vgtl_dag.h.

9.35.2.11 typedef _Base::erased_part dgraph::erased_part [protected]

an erased subgraph which is not yet a new directed graph

Reimplemented from __DG< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::const_iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >.

Reimplemented in dag.

Definition at line 2136 of file vgtl_dag.h.

9.35.2.12 typedef _DG_iterator<_Tp,_Tp&,_Tp*,container_type,children_iterator, children_const_iterator> __DG::iterator [inherited]

the iterator

Definition at line 597 of file vgtl_dag.h.

9.35.2.13 typedef _Node __DG::node_type [inherited]

standard typedef

Definition at line 581 of file vgtl_dag.h.

9.35.2.14 typedef _Base::parents_const_iterator dgraph::parents_const_iterator

the parents const iterator

Reimplemented from __DG< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::const_iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >.

Reimplemented in dag.

Definition at line 2148 of file vgtl_dag.h.

9.35.2.15 typedef _Base::parents_iterator dgraph::parents_iterator

the parents iterator

Reimplemented from __DG< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::const_iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >.

Reimplemented in dag.

Definition at line 2146 of file vgtl_dag.h.

9.35.2.16 typedef value_type* __DG::pointer [inherited]

standard typedef

Definition at line 582 of file vgtl_dag.h.

9.35.2.17 typedef value_type& __DG::reference [inherited]

standard typedef

Definition at line 584 of file vgtl_dag.h.

9.35.2.18 typedef std::reverse_iterator<iterator> __DG::reverse_iterator [inherited]

the reverse iterator

Definition at line 606 of file vgtl_dag.h.

9.35.2.19 typedef size_t __DG::size_type [inherited]

standard typedef

Definition at line 586 of file vgtl_dag.h.

9.35.2.20 typedef _Tp __DG::value_type [inherited]

standard typedef

Definition at line 580 of file vgtl_dag.h.

9.35.2.21 typedef _Base::walker dgraph::walker

the walker

Reimplemented from __DG< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::const_iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >.

Reimplemented in dag.

Definition at line 2140 of file vgtl_dag.h.

9.35.3 Constructor & Destructor Documentation

9.35.3.1 dgraph::dgraph (const allocator_type & __a = allocator_type()) [inline, explicit]

standard constructor

Definition at line 2154 of file vgtl_dag.h.

9.35.3.2 dgraph::dgraph (const _Self & __dg) [inline]

copy constructor

Definition at line 2157 of file vgtl_dag.h.

9.35.3.3 dgraph::dgraph (const erased_part & __ep, const allocator_type & __a = allocator_type()) [inline]

constructor from an erased_part

Definition at line 2160 of file vgtl_dag.h.

9.35.4 Member Function Documentation

9.35.4.1 _Node* __DG::_C_create_node (const _Tp & __x) [inline, protected, inherited]

construct a new tree node containing data __x

Definition at line 645 of file vgtl_dag.h.

9.35.4.2 _Node* __DG::_C_create_node ( ) [inline, protected, inherited]

construct a new tree node containing default data

Definition at line 659 of file vgtl_dag.h.

9.35.4.3 void __DG::_C_destroy_node (_Node * __p) [inline, protected, inherited]

construct a new tree node containing default data

Definition at line 673 of file vgtl_dag.h.

9.35.4.4 void __DG::add_edge (const edge & __edge, const container_insert_arg & __itc, const container_insert_arg & __itp) [inline, inherited]

add one edge between two nodes at the positions described by __itc and __itp.

Definition at line 1070 of file vgtl_dag.h.

9.35.4.5 void __DG::add_edge (const walker & __parent, const walker & __child, const container_insert_arg & __itc, const container_insert_arg & __itp) [inline, inherited]

add an edge between __parent and __child at positions __itc and __itp, respectively

Definition at line 1079 of file vgtl_dag.h.

9.35.4.6 void dgraph::add_edge (const walker & __parent, const children_iterator & __ch_it, const walker & __child, const parents_iterator & __pa_it) [inline]

add an edge between __parent and __child at specific positions __ch_it and __pa_it.

Definition at line 2389 of file vgtl_dag.h.

9.35.4.7 void dgraph::add_edge_back (const walker & __parent, const walker & __child) [inline]

add an edge between __parent and __child at the end of the children and parents containers.

Definition at line 2399 of file vgtl_dag.h.

9.35.4.8 void dgraph::add_edge_front (const walker & __parent, const walker & __child) [inline]

add an edge between __parent and __child at the beginning of the children and parents containers.

Definition at line 2409 of file vgtl_dag.h.

9.35.4.9 walker dgraph::between (const walker & __parent, const children_iterator & __cit, const walker & __child, const parents_iterator & __pit, const _Tp & __x) [inline]

here a new node is inserted between a parent node and a child node but the previous bonds between the two are not broken, the node is always new with data __x.

Definition at line 2177 of file vgtl_dag.h.

9.35.4.10 template<template< class __Tp, class __AllocTp > class __SequenceCtr1, template< class __Tp, class __AllocTp > class __SequenceCtr2, class _Allocator1 , class _Allocator2 > walker dgraph::between (const __SequenceCtr1< walker, _Allocator1 > & __parents, const __SequenceCtr2< walker, _Allocator2 > & __children, const _Tp & __x) [inline]

here a new node is inserted between many parents and many children but the previous bonds are not broken, the node is always new

Definition at line 2279 of file vgtl_dag.h.

9.35.4.11 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker dgraph::between (const walker & __parent, const children_iterator & __cit, const __SequenceCtr< walker, _Allocator > & __children, const _Tp & __x) [inline]

here a new node is inserted between one parent and many children but the previous bonds are not broken, the node is always new

Definition at line 2425 of file vgtl_dag.h.

9.35.4.12 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker dgraph::between (const __SequenceCtr< walker, _Allocator > & __parents, const walker & __child, const parents_iterator & __pit, const _Tp & __x) [inline]

here a new node is inserted between many parents and one child but the previous bonds are not broken, the node is always new

Definition at line 2525 of file vgtl_dag.h.

9.35.4.13 walker dgraph::between_back (const walker & __parent, const walker & __child, const _Tp & __x) [inline]

insert the node as the last child between parent and child, without breaking old bonds.

Definition at line 2212 of file vgtl_dag.h.

9.35.4.14 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker dgraph::between_back (const walker & *__parent,* const __SequenceCtr< walker, _Allocator > & *__children,* const _Tp & *__x*) [inline]

here a new node is inserted between one parent and many children but the previous bonds are not broken, the node is always new. At the parent the new child is put last.

Definition at line 2480 of file vgtl_dag.h.

9.35.4.15 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker dgraph::between_back (const __SequenceCtr< walker, _Allocator > & *__parents,* const walker & *__child,* const _Tp & *__x*) [inline]

here a new node is inserted between many parents and one child but the previous bonds are not broken, the node is always new. At the child the new parent is put last.

Definition at line 2579 of file vgtl_dag.h.

9.35.4.16 walker dgraph::between_front (const walker & *__parent,* const walker & *__child,* const _Tp & *__x*) [inline]

Here the inserted node is the first child of its parent and first parent of its child. Insert the node without breaking old bonds.

Definition at line 2243 of file vgtl_dag.h.

9.35.4.17 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker dgraph::between_front (const walker & *__parent,* const __SequenceCtr< walker, _Allocator > & *__children,* const _Tp & *__x*) [inline]

here a new node is inserted between one parent and many children but the previous bonds are not broken, the node is always new. At the parent the new child is put first.

Definition at line 2510 of file vgtl_dag.h.

9.35.4.18 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker dgraph::between_front (const __SequenceCtr< walker, _Allocator > & *__parents,* const walker & *__child,* const _Tp & *__x*) [inline]

here a new node is inserted between many parents and one child but the previous bonds are not broken, the node is always new. At the child the new parent is put first.

Definition at line 2607 of file vgtl_dag.h.

9.35.4.19 void dgraph::clear ( ) [inline]

empty the graph

Reimplemented from __DG< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::const_iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >.

Definition at line 2170 of file vgtl_dag.h.

9.35.4.20 void __DG::clear_erased_part (erased_part & *_ep*) [inline, inherited]

clear all nodes in an erased part

Definition at line 1751 of file vgtl_dag.h.

9.35.4.21 void __DG::copy_maximal_subgraph (const walker & __x, const walker & __par, const walker & __bo, const walker & __bn) [inline, inherited]

This function returns a copy of the maximal subgraph between the nodes __xn and __bo. Here __bo is connected to the new node __bn. __par is the new parent of the copied subgraph.

Definition at line 1956 of file vgtl_dag.h.

9.35.4.22 bool __DG::empty ( ) const [inline, inherited]

returns true if the DG is empty

Definition at line 767 of file vgtl_dag.h.

9.35.4.23 void __DG::erase (const walker & __position) [inline, inherited]

erase a node from the DG except the sky and ground

Definition at line 1400 of file vgtl_dag.h.

9.35.4.24 bool __DG::erase_child (const walker & __position, const children_iterator & __it) [inline, inherited]

Erase a child of __position. This works if and only if the child has only one child and no other parents.

Definition at line 1904 of file vgtl_dag.h.

9.35.4.25 erased_part __DG::erase_maximal_pregraph (const walker & __position) [inline, inherited]

here every child is removed till the sky node. included the node at __position. The removed subgraph is returned. The subgraph is maximal, i.e. all nodes are removed, which are reachable from __position by walking upwards.

Definition at line 1834 of file vgtl_dag.h.

9.35.4.26 erased_part __DG::erase_maximal_pregraph (const __SequenceCtr< walker, _Allocator > & __positions) [inline, inherited]

here every child is removed till the sky included all nodes from __positions. The removed subgraph is returned. The subgraph is maximal, i.e. all nodes are removed, which are reachable from any node from __positions by walking up.

Definition at line 1868 of file vgtl_dag.h.

9.35.4.27 erased_part __DG::erase_maximal_subgraph (const walker & __position) [inline, inherited]

here every child is removed till the last base node, included the node at __position. The removed subgraph is returned. The subgraph is maximal, i.e. all nodes are removed, which are reachable from __position by walking down.

Definition at line 1763 of file vgtl_dag.h.

9.35.4.28 erased_part __DG::erase_maximal_subgraph (const __SequenceCtr< walker, _Allocator > & __positions) [inline, inherited]

here every child is removed till the last base node, included all nodes from __positions. The removed subgraph is returned. The subgraph is maximal, i.e. all nodes are removed, which are reachable from any node from __positions by walking down.

Definition at line 1797 of file vgtl_dag.h.

9.35.4.29 erased_part __DG::erase_minimal_pregraph (const walker & __position) [inline, inherited]

here every child is removed till the sky. included the node at __position. The removed subgraph is returned. The subgraph is minimal, i.e. only nodes are removed, which have no other descendant than __position. I.e., when walking towards the sky, there is no way which bypasses __position.

Definition at line 1850 of file vgtl_dag.h.

9.35.4.30 erased_part __DG::erase_minimal_pregraph (const __SequenceCtr< walker, _Allocator > & __positions) [inline, inherited]

here every child is removed till the sky. included all nodes from __positions. The removed subgraph is returned. The subgraph is minimal, i.e. only nodes are removed, which have no other ancestor than any node in __positions. I.e., when walking towards the ground, there is no way which bypasses all nodes in __positions.

Definition at line 1888 of file vgtl_dag.h.

9.35.4.31 erased_part __DG::erase_minimal_subgraph (const walker & __position) [inline, inherited]

here every child is removed till the last base node, included the node at __position. The removed subgraph is returned. The subgraph is minimal, i.e. only nodes are removed, which have no other ancestor than __position. I.e., when walking towards the ground, there is no way which bypasses __position.

Definition at line 1779 of file vgtl_dag.h.

9.35.4.32 erased_part __DG::erase_minimal_subgraph (const __SequenceCtr< walker, _Allocator > & __positions) [inline, inherited]

here every child is removed till the last base node, included all nodes from __positions. The removed subgraph is returned. The subgraph is minimal, i.e. only nodes are removed, which have no other ancestor than any node in __positions. I.e., when walking towards the ground, there is no way which bypasses all nodes in __positions.

Definition at line 1817 of file vgtl_dag.h.

9.35.4.33 bool __DG::erase_parent (const walker & __position, const parents_iterator & __lt) [inline, inherited]

Erase a parent of __position. This works if and only if the parent has only one parent and no other children.

Definition at line 1930 of file vgtl_dag.h.

9.35.4.34 allocator_type __DG::get_allocator ( ) const [inline, inherited]

construct an allocator object

Definition at line 592 of file vgtl_dag.h.

9.35.4.35 walker __DG::ground ( ) [inline, inherited]

return a walker to the virtual ground node.

Definition at line 687 of file vgtl_dag.h.

9.35.4.36 const_walker __DG::ground ( ) const [inline, inherited]

return a const walker to the virtual ground node.

Definition at line 697 of file vgtl_dag.h.

9.35.4.37 void dgraph::insert_back_subgraph (_Self & *__subgraph*, const walker & *__parent*, const walker & *__child*) [inline]

here a subgraph is inserted between a parent and a child, at the end of the children resp. parents lists.

Definition at line 2348 of file vgtl_dag.h.

9.35.4.38 void dgraph::insert_front_subgraph (_Self & *__subgraph*, const walker & *__parent*, const walker & *__child*) [inline]

here a subgraph is inserted between a parent and a child, at the front of the children resp. parents lists.

Definition at line 2361 of file vgtl_dag.h.

9.35.4.39 walker __DG::insert_in_graph (const _Tp & *__x*, const walker & *__parent*, const walker & *__child*, const container_insert_arg & *__itc*, const container_insert_arg & *__itp*) [inline, inherited]

insert node with data `__n` into the graph between `__parent` and `__child`, the edge at the specific positions described by `__Itc` and `__Itp`.

Definition at line 807 of file vgtl_dag.h.

9.35.4.40 walker __DG::insert_in_graph (const walker & *__parent*, const walker & *__child*, const container_insert_arg & *__itc*, const container_insert_arg & *__itp*) [inline, inherited]

insert node with default data into the graph between `__parent` and `__child`, the edge at the specific positions described by `__Itc` and `__Itp`.

Definition at line 821 of file vgtl_dag.h.

9.35.4.41 walker __DG::insert_in_graph (const _Tp & *__x*, const __SequenceCtr1< walker, _Allocator1 > & *__parents*, const __SequenceCtr2< walker, _Allocator2 > & *__children*) [inline, inherited]

insert a node with data `__x` into the graph between all parents from `__parents` and all children from `__children`.

Definition at line 885 of file vgtl_dag.h.

9.35.4.42 walker __DG::insert_in_graph (const __SequenceCtr1< walker, _Allocator1 > & *__parents*, const __SequenceCtr2< walker, _Allocator2 > & *__children*) [inline, inherited]

insert a node with default data into the graph between all parents from `__parents` and all children from `__children`.

Definition at line 900 of file vgtl_dag.h.

9.35.4.43 walker __DG::insert_in_graph (const _Tp & __x, const walker & __parent, const container_insert_arg & __pref, const __SequenceCtr< walker, _Allocator > & __children) [inline, inherited]

insert a node with data __x into the graph between the parent __parent and all children from __children.

Definition at line 938 of file vgtl_dag.h.

9.35.4.44 walker __DG::insert_in_graph (const walker & __parent, const container_insert_arg & __pref, const __SequenceCtr< walker, _Allocator > & __children) [inline, inherited]

insert a node with data __x into the graph between the parent __parent and all children from __children.

Definition at line 952 of file vgtl_dag.h.

9.35.4.45 walker __DG::insert_in_graph (const _Tp & __x, const __SequenceCtr< walker, _Allocator > & __parents, const walker & __child, const container_insert_arg & __cref) [inline, inherited]

insert a node with data __x into the graph between all parents from __parents and the child __child.

Definition at line 991 of file vgtl_dag.h.

9.35.4.46 walker __DG::insert_in_graph (const __SequenceCtr< walker, _Allocator > & __parents, const walker & __child, const container_insert_arg & __cref) [inline, inherited]

insert a node with default data into the graph between all parents from __parents and the child __child.

Definition at line 1006 of file vgtl_dag.h.

9.35.4.47 walker __DG::insert_node (_Node * _node, const walker & __position, const container_insert_arg & __lt) [inline, inherited]

insert one node as child of __position

Definition at line 1261 of file vgtl_dag.h.

9.35.4.48 walker __DG::insert_node (const _Tp & __x, const walker & __position, const container_insert_arg & __lt) [inline, inherited]

insert a new node with data __x as child of __position

Definition at line 1275 of file vgtl_dag.h.

9.35.4.49 walker __DG::insert_node (const walker & __position, const container_insert_arg & __lt) [inline, inherited]

insert a new node with default data as child of __position

Definition at line 1281 of file vgtl_dag.h.

9.35.4.50 walker __DG::insert_node_before (_Node * _node, const walker & __position, const container_insert_arg & __lt) [inline, inherited]

insert a node as parent of __position

Definition at line 1286 of file vgtl_dag.h.

9.35.4.51 void __DG::insert_node_before (const _Tp & __x, const walker & __position, const container_insert_arg & __lt) [inline, inherited]

insert a new node with data __x as parent of __position

Definition at line 1300 of file vgtl_dag.h.

9.35.4.52 void __DG::insert_node_before (const walker & __position, const container_insert_arg & __lt) [inline, inherited]

insert a new node with default data as parent of __position

Definition at line 1305 of file vgtl_dag.h.

9.35.4.53 walker __DG::insert_node_in_graph (_Node * __n, const walker & __parent, const walker & __child, const container_insert_arg & __ltc, const container_insert_arg & __ltp) [inline, inherited]

insert node __n into the graph between __parent and __child, the edge at the specific positions described by __Itc and __Itp.

Definition at line 791 of file vgtl_dag.h.

9.35.4.54 walker __DG::insert_node_in_graph (_Node * __node, const __SequenceCtr1< walker, _Allocator1 > & __parents, const __SequenceCtr2< walker, _Allocator2 > & __children) [inline, inherited]

insert node __n into the graph between all parents from __parents and all children from __children.

Definition at line 854 of file vgtl_dag.h.

9.35.4.55 walker __DG::insert_node_in_graph (_Node * __node, const walker & __parent, const container_insert_arg & __pref, const __SequenceCtr< walker, _Allocator > & __children) [inline, inherited]

insert node __n into the graph between the parent __parent and all children from __children.

Definition at line 913 of file vgtl_dag.h.

9.35.4.56 walker __DG::insert_node_in_graph (_Node * __node, const __SequenceCtr< walker, _Allocator > & __parents, const walker & __child, const container_insert_arg & __cref) [inline, inherited]

insert node __n into the graph between all parents from __parents and the child __child.

Definition at line 966 of file vgtl_dag.h.

9.35.4.57 void __DG::insert_subgraph (_Self & __subgraph, const walker & __parent, const walker & __child, const container_insert_arg & __ltc, const container_insert_arg & __ltp) [inline, inherited]

insert a subgraph into the graph between __parent and __child, the edge at the specific positions described by __Itc and __Itp.

Definition at line 832 of file vgtl_dag.h.

9.35.4.58 void __DG::insert_subgraph (_Self & __subgraph, const __SequenceCtr1< walker, _Allocator1 > & __parents, const __SequenceCtr2< walker, _Allocator2 > & __children) [inline, inherited]

in this method one DG is inserted into another DG between the parents ___parents and the children ___children.

Definition at line 1020 of file vgtl_dag.h.

9.35.4.59 void dgraph::insert_subgraph (_Self & __subgraph, const walker & __parent, const children_iterator & __ch_it, const walker & __child, const parents_iterator & __pa_it) [inline]

here a subgraph is inserted between a parent and a child, at specific positions ___ch_it and ___pa_it.

Definition at line 2337 of file vgtl_dag.h.

9.35.4.60 parents_iterator __DG::leaf_begin ( ) [inline, inherited]

return the first leaf of the directed graph

Definition at line 721 of file vgtl_dag.h.

9.35.4.61 parents_const_iterator __DG::leaf_begin ( ) const [inline, inherited]

return the first leaf of the directed graph

Definition at line 728 of file vgtl_dag.h.

9.35.4.62 parents_iterator __DG::leaf_end ( ) [inline, inherited]

return beyond the last leaf of the directed graph

Definition at line 724 of file vgtl_dag.h.

9.35.4.63 parents_const_iterator __DG::leaf_end ( ) const [inline, inherited]

return beyond the last leaf of the directed graph

Definition at line 731 of file vgtl_dag.h.

9.35.4.64 size_type __DG::max_size ( ) const [inline, inherited]

the maximum size of a DG is virtually unlimited

Definition at line 778 of file vgtl_dag.h.

9.35.4.65 void __DG::merge (const walker & __position, const walker & __second, bool merge_parent_edges = true, bool merge_child_edges = true) [inline, inherited]

merge two nodes, call also the merge method for the node data

Definition at line 1311 of file vgtl_dag.h.

9.35.4.66 _Self& dgraph::operator= (const _RV_DG & __rl) [inline]

assignment operator from a part of an erased part

Reimplemented from __DG< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::const_iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >.

Definition at line 2616 of file vgtl_dag.h.

9.35.4.67 _Self& dgraph::operator= (const erased_part & __ep) [inline]

assignment operator from an erased part

Reimplemented from __DG< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::const_iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >.

Definition at line 2624 of file vgtl_dag.h.

9.35.4.68 void __DG::partial_erase_to_parent (const walker & __position, const walker & __parent, unsigned int idx) [inline, inherited]

split a node in two, the first connected to the __parent, the second connected to all other parents. Then erase the first node.

Definition at line 1461 of file vgtl_dag.h.

9.35.4.69 void __DG::remove_edge (const edge & __edge) [inline, inherited]

remove an edge with a particular parent and child

Definition at line 1197 of file vgtl_dag.h.

9.35.4.70 void __DG::remove_edge (const walker & __parent, const walker & __child) [inline, inherited]

just remove one edge between __parent and __child

Definition at line 1214 of file vgtl_dag.h.

9.35.4.71 void __DG::remove_edge_and_deattach (const walker & __parent, const walker & __child) [inline, inherited]

remove one egde and don't reconnect the node to sky/ground

Definition at line 1201 of file vgtl_dag.h.

9.35.4.72 void __DG::replace_edge_to_child (const walker & __parent, const walker & __child_old, const walker & __child_new) [inline, inherited]

change the edge from __parent to __child_old to an edge from __parent to __child_new.

Definition at line 1125 of file vgtl_dag.h.

9.35.4.73 void __DG::replace_edge_to_parent (const walker & __parent_old, const walker & __parent_new, const walker & __child) [inline, inherited]

change the edge from __parent_old to __child to an edge from __parent_new to __child.

Definition at line 1163 of file vgtl_dag.h.

9.35.4.74 children_iterator __DG::root_begin ( ) [inline, inherited]

return the first root of the directed graph

Definition at line 707 of file vgtl_dag.h.

9.35.4.75 children_const_iterator __DG::root_begin ( ) const [inline, inherited]

return the first root of the directed graph

Definition at line 714 of file vgtl_dag.h.

9.35.4.76 children_iterator __DG::root_end ( ) [inline, inherited]

return beyond the last root of the directed graph

Definition at line 710 of file vgtl_dag.h.

9.35.4.77 children_const_iterator __DG::root_end ( ) const [inline, inherited]

return beyond the last root of the directed graph

Definition at line 717 of file vgtl_dag.h.

9.35.4.78 size_type __DG::size ( ) const [inline, inherited]

returns the size of the DG (number of nodes)

Definition at line 771 of file vgtl_dag.h.

9.35.4.79 walker __DG::sky ( ) [inline, inherited]

return a walker to the virtual sky node.

Definition at line 692 of file vgtl_dag.h.

9.35.4.80 const_walker __DG::sky ( ) const [inline, inherited]

return a const walker to the virtual sky node.

Definition at line 702 of file vgtl_dag.h.

9.35.4.81 void __DG::sort_child_edges (walker __*position*, children_iterator *first*, children_iterator *last*, Compare *comp*) [inline, inherited]

sort the child edges in the range [first,last) according to `comp`

Definition at line 1238 of file vgtl_dag.h.

9.35.4.82 void __DG::sort_child_edges (walker __*position*, Compare *comp*) [inline, inherited]

sort all child edges according to `comp`

Definition at line 1250 of file vgtl_dag.h.

9.35.4.83 void __DG::sort_parent_edges (walker __*position*, parents_iterator *first*, parents_iterator *last*, Compare *comp*) [inline, inherited]

sort the parent edges in the range [first,last) according to `comp`

Definition at line 1244 of file vgtl_dag.h.

9.35.4.84 void __DG::sort_parent_edges (walker __*position*, Compare *comp*) [inline, inherited]

sort all parent edges according to `comp`

Definition at line 1256 of file vgtl_dag.h.

9.35.4.85 walker dgraph::split (const walker & *__parent,* const children_iterator & *__ch_it,* const walker & *__child,* const parents_iterator & *__pa_it,* const _Tp & *__x*) [inline]

here a new node is inserted between a parent node and a child node and the previous bonds between them are broken, the node is always new with data ___x.

Definition at line 2190 of file vgtl_dag.h.

9.35.4.86 template<template< class __Tp, class __AllocTp > class __SequenceCtr1, template< class __Tp, class __AllocTp > class __SequenceCtr2, class _Allocator1 , class _Allocator2 > walker dgraph::split (const __SequenceCtr1< walker, _Allocator1 > & *__parents,* const __SequenceCtr2< walker, _Allocator2 > & *__children,* const _Tp & *__x*) [inline]

here a new node is inserted between many parents and many children, and the previous bonds are broken, the node is always new.

Definition at line 2311 of file vgtl_dag.h.

9.35.4.87 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker dgraph::split (const walker & *__parent,* const children_iterator & *__ch_it,* const __SequenceCtr< walker, _Allocator > & *__children,* const _Tp & *__x*) [inline]

here a new node is inserted between one parent and many children, and the previous bonds are broken, the node is always new.

Definition at line 2438 of file vgtl_dag.h.

9.35.4.88 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker dgraph::split (const __SequenceCtr< walker, _Allocator > & *__parents,* const walker & *__child,* const parents_iterator & *__pr_it,* const _Tp & *__x*) [inline]

here a new node is inserted between many parents and one child, and the previous bonds are broken, the node is always new.

Definition at line 2538 of file vgtl_dag.h.

9.35.4.89 walker dgraph::split_back (const walker & *__parent,* const walker & *__child,* const _Tp & *__x*) [inline]

insert the node as the last child between parent and child, with breaking old bonds.

Definition at line 2225 of file vgtl_dag.h.

9.35.4.90 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker dgraph::split_back (const walker & *__parent,* const __SequenceCtr< walker, _Allocator > & *__children,* const _Tp & *__x*) [inline]

here a new node is inserted between one parent and many children, and the previous bonds are broken, the node is always new. At the parent the new child is put last.

Definition at line 2465 of file vgtl_dag.h.

9.35.4.91 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker dgraph::split_back (const __SequenceCtr< walker, _Allocator > & *__parents,* const walker & *__child,* const _Tp & *__x*) [inline]

here a new node is inserted between many parents and one child, and the previous bonds are broken, the node is always new. At the child the new parent is put last.

Definition at line 2565 of file vgtl_dag.h.

9.35.4.92 walker dgraph::split_front (const walker & *__parent,* const walker & *__child,* const _Tp & *__x*) [inline]

Here the inserted node is the first child of its parent and first parent of its child. Insert the node and break old bonds.

Definition at line 2256 of file vgtl_dag.h.

9.35.4.93 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker dgraph::split_front (const walker & *__parent,* const __SequenceCtr< walker, _Allocator > & *__children,* const _Tp & *__x*) [inline]

here a new node is inserted between one parent and many children, and the previous bonds are broken, the node is always new. At the parent the new child is put first.

Definition at line 2495 of file vgtl_dag.h.

9.35.4.94 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker dgraph::split_front (const __SequenceCtr< walker, _Allocator > & *__parents,* const walker & *__child,* const _Tp & *__x*) [inline]

here a new node is inserted between many parents and one child, and the previous bonds are broken, the node is always new. At the child the new parent is put first.

Definition at line 2593 of file vgtl_dag.h.

9.35.4.95 void __DG::swap (_Self & *__x*) [inline, inherited]

swap two DGs

Definition at line 781 of file vgtl_dag.h.

The documentation for this class was generated from the following file:

- vgtl_dag.h

9.36 ldag Class Reference

labeled directed acyclic graph (LDAG)

`#include <vgtl_ldag.h>`

Inheritance diagram for ldag:

Collaboration diagram for ldag:

Public Types

- typedef _Base::walker walker
- typedef _Base::const_walker const_walker
- typedef _Base::children_iterator children_iterator
- typedef _Base::parents_iterator parents_iterator
- typedef _Base::children_const_iterator children_const_iterator
- typedef _Base::parents_const_iterator parents_const_iterator
- typedef _Base::erased_part erased_part

Public Member Functions

- ldag (const allocator_type &__a=allocator_type())
- ldag (const _Self &__ldag)
- ldag (const _Base &__ldag)
- ldag (const erased_part &__ep)
- bool check_acyclicity (const walker &__parent, const walker &__child)
- _Self & operator= (const _RV_LDG &__rl)
- _Self & operator= (const erased_part &__ep)
- void clear ()
- walker between (const walker &__parent, const children_iterator &__cit, const walker &__child, const parents_iterator &__pit, const _Tp &__x)
- walker between (const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children, const _Tp &__x)
- walker between (const walker &__parent, const children_iterator &__cit, const __SequenceCtr< walker, _Allocator > &__children, const _Tp &__x)
- walker between (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const parents_iterator &__pit, const _Tp &__x)
- walker split (const walker &__parent, const children_iterator &__ch_it, const walker &__child, const parents_iterator &__pa_it, const _Tp &__x)
- void split (const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children, const _Tp &__x)
- walker split (const walker &__parent, const children_iterator &__ch_it, const __SequenceCtr< walker, _Allocator > &__children, const _Tp &__x)
- walker split (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const parents_iterator &__pr_it, const _Tp &__x)
- walker between_back (const walker &__parent, const walker &__child, const _Tp &__x)
- walker between_back (const walker &__parent, const __SequenceCtr< walker, _Allocator > &__children, const _Tp &__x)
- walker between_back (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const _Tp &__x)
- walker split_back (const walker &__parent, const walker &__child, const _Tp &__x)
- walker split_back (const walker &__parent, const __SequenceCtr< walker, _Allocator > &__children, const _Tp &__x)
- walker split_back (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const _Tp &__x)

- walker between_front (const walker &__parent, const walker &__child, const _Tp &__x)
- walker between_front (const walker &__parent, const __SequenceCtr< walker, _Allocator > &__children, const _Tp &__x)
- walker between_front (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const _Tp &__x)
- walker split_front (const walker &__parent, const walker &__child, const _Tp &__x)
- walker split_front (const walker &__parent, const __SequenceCtr< walker, _Allocator > &__children, const _Tp &__x)
- walker split_front (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const _Tp &__x)
- void insert_subgraph (_Self &__subgraph, const walker &__parent, const children_iterator &__ch_it, const walker &__child, const parents_iterator &__pa_it)
- void insert_back_subgraph (_Self &__subgraph, const walker &__parent, const walker &__child)
- void insert_front_subgraph (_Self &__subgraph, const walker &__parent, const walker &__child)
- void add_edge (const walker &__parent, const children_iterator &__ch_it, const walker &__child, const parents_iterator &__pa_it)
- void add_edge_back (const walker &__parent, const walker &__child)
- void add_edge_front (const walker &__parent, const walker &__child)

9.36.1 Detailed Description

This class constructs a labeled directed acyclic graph (LDAG). By default, the children and the parents are collected in an STL vector, but the container can be replaced by any other sequential container.

9.36.2 Member Typedef Documentation

9.36.2.1 typedef _Base::children_const_iterator ldag::children_const_iterator

the children const iterator

Reimplemented from ldgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

Definition at line 2772 of file vgtl_ldag.h.

9.36.2.2 typedef _Base::children_iterator ldag::children_iterator

the children iterator

Reimplemented from ldgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

Definition at line 2768 of file vgtl_ldag.h.

9.36.2.3 typedef _Base::const_walker ldag::const_walker

the const walker

Reimplemented from ldgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

Definition at line 2766 of file vgtl_ldag.h.

9.36.2.4 typedef _Base::erased_part ldag::erased_part

the erased part constructed in erasing subgraphs

Reimplemented from ldgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

Definition at line 2777 of file vgtl_ldag.h.

9.36.2.5 typedef _Base::parents_const_iterator ldag::parents_const_iterator

the parents const iterator

Reimplemented from ldgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

Definition at line 2774 of file vgtl_ldag.h.

9.36.2.6 typedef _Base::parents_iterator ldag::parents_iterator

the parents iterator

Reimplemented from ldgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

Definition at line 2770 of file vgtl_ldag.h.

9.36.2.7 typedef _Base::walker ldag::walker

the walker

Reimplemented from ldgraph< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >.

Definition at line 2764 of file vgtl_ldag.h.

9.36.3 Constructor & Destructor Documentation

9.36.3.1 ldag::ldag (const allocator_type & __a = allocator_type()) [inline, explicit]

standard constructor

Definition at line 2781 of file vgtl_ldag.h.

9.36.3.2 ldag::ldag (const _Self & __ldag) [inline]

copy constructor

Definition at line 2784 of file vgtl_ldag.h.

9.36.3.3 ldag::ldag (const _Base & __ldag) [inline]

construct ldag from directed graph

Definition at line 2790 of file vgtl_ldag.h.

9.36.3.4 ldag::ldag (const erased_part & __ep) [inline]

construct ldag from erased part

Definition at line 2798 of file vgtl_ldag.h.

9.36.4 Member Function Documentation

9.36.4.1 void ldgraph::add_edge (const walker & __parent, const children_iterator & __ch_it, const walker & __child, const parents_iterator & __pa_it) [inline, inherited]

add an edge between __parent and __child at specific positions __ch_it and __pa_it.

Definition at line 2488 of file vgtl_ldag.h.

9.36.4.2 void ldgraph::add_edge_back (const walker & __parent, const walker & __child) [inline, inherited]

add an edge between __parent and __child at the end of the children and parents containers.

Definition at line 2498 of file vgtl_ldag.h.

9.36.4.3 void ldgraph::add_edge_front (const walker & __parent, const walker & __child) [inline, inherited]

add an edge between __parent and __child at the beginning of the children and parents containers.

Definition at line 2508 of file vgtl_ldag.h.

9.36.4.4 walker ldgraph::between (const walker & __parent, const children_iterator & __cit, const walker & __child, const parents_iterator & __pit, const _Tp & __x) [inline, inherited]

here a new node is inserted between a parent node and a child node but the previous bonds between the two are not broken, the node is always new with data __x.

Definition at line 2276 of file vgtl_ldag.h.

9.36.4.5 walker ldgraph::between (const __SequenceCtr1< walker, _Allocator1 > & __parents, const __SequenceCtr2< walker, _Allocator2 > & __children, const _Tp & __x) [inline, inherited]

here a new node is inserted between many parents and many children but the previous bonds are not broken, the node is always new

Definition at line 2378 of file vgtl_ldag.h.

9.36.4.6 walker ldgraph::between (const walker & __parent, const children_iterator & __cit, const __SequenceCtr< walker, _Allocator > & __children, const _Tp & __x) [inline, inherited]

here a new node is inserted between one parent and many children but the previous bonds are not broken, the node is always new

Definition at line 2524 of file vgtl_ldag.h.

9.36.4.7 walker ldgraph::between (const __SequenceCtr< walker, _Allocator > & __parents, const walker & __child, const parents_iterator & __pit, const _Tp & __x) [inline, inherited]

here a new node is inserted between many parents and one child but the previous bonds are not broken, the node is always new

Definition at line 2624 of file vgtl_ldag.h.

9.36.4.8 walker ldgraph::between_back (const walker & __parent, const walker & __child, const _Tp & __x) [inline, inherited]

insert the node as the last child between parent and child, without breaking old bonds.

Definition at line 2311 of file vgtl_ldag.h.

9.36.4.9 walker ldgraph::between_back (const walker & __parent, const __SequenceCtr< walker, _Allocator > & __children, const _Tp & __x) [inline, inherited]

here a new node is inserted between one parent and many children but the previous bonds are not broken, the node is always new. At the parent the new child is put last.

Definition at line 2579 of file vgtl_ldag.h.

9.36.4.10 walker ldgraph::between_back (const __SequenceCtr< walker, _Allocator > & __parents, const walker & __child, const _Tp & __x) [inline, inherited]

here a new node is inserted between many parents and one child but the previous bonds are not broken, the node is always new. At the child the new parent is put last.

Definition at line 2678 of file vgtl_ldag.h.

9.36.4.11 walker ldgraph::between_front (const walker & __parent, const walker & __child, const _Tp & __x) [inline, inherited]

Here the inserted node is the first child of its parent and first parent of its child. Insert the node without breaking old bonds.

Definition at line 2342 of file vgtl_ldag.h.

9.36.4.12 walker ldgraph::between_front (const walker & __parent, const __SequenceCtr< walker, _Allocator > & __children, const _Tp & __x) [inline, inherited]

here a new node is inserted between one parent and many children but the previous bonds are not broken, the node is always new. At the parent the new child is put first.

Definition at line 2609 of file vgtl_ldag.h.

9.36.4.13 walker ldgraph::between_front (const __SequenceCtr< walker, _Allocator > & __parents, const walker & __child, const _Tp & __x) [inline, inherited]

here a new node is inserted between many parents and one child but the previous bonds are not broken, the node is always new. At the child the new parent is put first.

Definition at line 2706 of file vgtl_ldag.h.

9.36.4.14 bool ldag::check_acyclicity (const walker & __parent, const walker & __child) [inline]

This method checks, whether the ldag is indeed acyclic. This is NYI!

Definition at line 2822 of file vgtl_ldag.h.

9.36.4.15 void ldgraph::clear ( ) [inline, inherited]

empty the graph

Definition at line 2269 of file vgtl_ldag.h.

9.36.4.16 void ldgraph::insert_back_subgraph (_Self & __subgraph, const walker & __parent, const walker & __child) [inline, inherited]

here a subgraph is inserted between a parent and a child, at the end of the children resp. parents lists.

Definition at line 2447 of file vgtl_ldag.h.

9.36.4.17 void ldgraph::insert_front_subgraph (_Self & __subgraph, const walker & __parent, const walker & __child) [inline, inherited]

here a subgraph is inserted between a parent and a child, at the front of the children resp. parents lists.

Definition at line 2460 of file vgtl_ldag.h.

9.36.4.18 void ldgraph::insert_subgraph (_Self & __subgraph, const walker & __parent, const children_iterator & __ch_it, const walker & __child, const parents_iterator & __pa_it) [inline, inherited]

here a subgraph is inserted between a parent and a child, at specific positions ___ch_it and ___pa_it.

Definition at line 2436 of file vgtl_ldag.h.

9.36.4.19 _Self& ldag::operator= (const _RV_LDG & __rl) [inline]

assignment from part of an erased part

Definition at line 2838 of file vgtl_ldag.h.

9.36.4.20 _Self& ldag::operator= (const erased_part & __ep) [inline]

assignment from erased part

Definition at line 2846 of file vgtl_ldag.h.

9.36.4.21 walker ldgraph::split (const walker & __parent, const children_iterator & __ch_it, const walker & __child, const parents_iterator & __pa_it, const _Tp & __x) [inline, inherited]

here a new node is inserted between a parent node and a child node and the previous bonds between them are broken, the node is always new with data ___x.

Definition at line 2289 of file vgtl_ldag.h.

9.36.4.22 void ldgraph::split (const __SequenceCtr1< walker, _Allocator1 > & __parents, const __SequenceCtr2< walker, _Allocator2 > & __children, const _Tp & __x) [inline, inherited]

here a new node is inserted between many parents and many children, and the previous bonds are broken, the node is always new.

Definition at line 2410 of file vgtl_ldag.h.

9.36.4.23 walker ldgraph::split (const walker & __parent, const children_iterator & __ch_it, const __SequenceCtr< walker, _Allocator > & __children, const _Tp & __x) [inline, inherited]

here a new node is inserted between one parent and many children, and the previous bonds are broken, the node is always new.

Definition at line 2537 of file vgtl_ldag.h.

9.36.4.24 walker ldgraph::split (const __SequenceCtr< walker, _Allocator > & __parents, const walker & __child, const parents_iterator & __pr_it, const _Tp & __x) [inline, inherited]

here a new node is inserted between many parents and one child, and the previous bonds are broken, the node is always new.

Definition at line 2637 of file vgtl_ldag.h.

9.36.4.25 walker ldgraph::split_back (const walker & __parent, const walker & __child, const _Tp & __x) [inline, inherited]

insert the node as the last child between parent and child, with breaking old bonds.

Definition at line 2324 of file vgtl_ldag.h.

9.36.4.26 walker ldgraph::split_back (const walker & __parent, const __SequenceCtr< walker, _Allocator > & __children, const _Tp & __x) [inline, inherited]

here a new node is inserted between one parent and many children, and the previous bonds are broken, the node is always new. At the parent the new child is put last.

Definition at line 2564 of file vgtl_ldag.h.

9.36.4.27 walker ldgraph::split_back (const __SequenceCtr< walker, _Allocator > & __parents, const walker & __child, const _Tp & __x) [inline, inherited]

here a new node is inserted between many parents and one child, and the previous bonds are broken, the node is always new. At the child the new parent is put last.

Definition at line 2664 of file vgtl_ldag.h.

9.36.4.28 walker ldgraph::split_front (const walker & __parent, const walker & __child, const _Tp & __x) [inline, inherited]

Here the inserted node is the first child of its parent and first parent of its child. Insert the node and break old bonds.

Definition at line 2355 of file vgtl_ldag.h.

9.36.4.29 walker ldgraph::split_front (const walker & __parent, const __SequenceCtr< walker, _Allocator > & __children, const _Tp & __x) [inline, inherited]

here a new node is inserted between one parent and many children, and the previous bonds are broken, the node is always new. At the parent the new child is put first.

Definition at line 2594 of file vgtl_ldag.h.

9.36.4.30 walker ldgraph::split_front (const __SequenceCtr< walker, _Allocator > & __parents, const walker & __child, const _Tp & __x) [inline, inherited]

here a new node is inserted between many parents and one child, and the previous bonds are broken, the node is always new. At the child the new parent is put first.

Definition at line 2692 of file vgtl_ldag.h.

The documentation for this class was generated from the following file:

- vgtl_ldag.h

9.37 ldgraph Class Reference

labeled directed graph

```
#include <vgtl_ldag.h>
```

Inheritance diagram for ldgraph:

Collaboration diagram for ldgraph:

Public Types

- typedef _Base::walker walker
- typedef _Base::const_walker const_walker
- typedef _Base::children_iterator children_iterator
- typedef _Base::parents_iterator parents_iterator
- typedef _Base::parents_const_iterator parents_const_iterator
- typedef _Base::children_const_iterator children_const_iterator
- typedef _LDG_iterator< _Tp, _Tp &, _Tp *, container_type, children_iterator, children_const_iterator, _Te > iterator
- typedef _LDG_iterator< _Tp, const _Tp &, const _Tp *, container_type, children_iterator, children_const_iterator, _Te > const_iterator
- typedef std::reverse_iterator < const_iterator > const_reverse_iterator
- typedef std::reverse_iterator < iterator > reverse_iterator

- typedef _Tp value_type
- typedef _Node node_type
- typedef _Edge edge_type
- typedef value_type * pointer
- typedef const value_type * const_pointer
- typedef value_type & reference
- typedef const value_type & const_reference
- typedef size_t size_type
- typedef ptrdiff_t difference_type

Public Member Functions

- ldgraph (const allocator_type &__a=allocator_type())
- ldgraph (const _Self &__dg)
- ldgraph (const erased_part &__ep, const allocator_type &__a=allocator_type())
- void clear ()
- walker between (const walker &__parent, const children_iterator &__cit, const walker &__child, const parents_iterator &__pit, const _Tp &__x)
- walker split (const walker &__parent, const children_iterator &__ch_it, const walker &__child, const parents_iterator &__pa_it, const _Tp &__x)
- walker between_back (const walker &__parent, const walker &__child, const _Tp &__x)
- walker split_back (const walker &__parent, const walker &__child, const _Tp &__x)
- walker between_front (const walker &__parent, const walker &__child, const _Tp &__x)

- walker split_front (const walker &__parent, const walker &__child, const _Tp &__x)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr1, template< class __Tp, class __AllocTp > class __SequenceCtr2, class _Allocator1 , class _Allocator2 >
  walker between (const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children, const _Tp &__x)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr1, template< class __Tp, class __AllocTp > class __SequenceCtr2, class _Allocator1 , class _Allocator2 >
  void split (const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children, const _Tp &__x)
- void insert_subgraph (_Self &__subgraph, const walker &__parent, const children_iterator &__ch-_it, const walker &__child, const parents_iterator &__pa_it)
- void insert_back_subgraph (_Self &__subgraph, const walker &__parent, const walker &__child)
- void insert_front_subgraph (_Self &__subgraph, const walker &__parent, const walker &__child)
- void add_edge (const walker &__parent, const children_iterator &__ch_it, const walker &__child, const parents_iterator &__pa_it)
- void add_edge_back (const walker &__parent, const walker &__child)
- void add_edge_front (const walker &__parent, const walker &__child)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker between (const walker &__parent, const children_iterator &__cit, const __SequenceCtr< walker, _Allocator > &__children, const _Tp &__x)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker split (const walker &__parent, const children_iterator &__ch_it, const __SequenceCtr< walker, _Allocator > &__children, const _Tp &__x)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker split_back (const walker &__parent, const __SequenceCtr< walker, _Allocator > &__-children, const _Tp &__x)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker between_back (const walker &__parent, const __SequenceCtr< walker, _Allocator > &__-children, const _Tp &__x)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker split_front (const walker &__parent, const __SequenceCtr< walker, _Allocator > &__-children, const _Tp &__x)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker between_front (const walker &__parent, const __SequenceCtr< walker, _Allocator > &__-children, const _Tp &__x)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker between (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const parents_iterator &__pit, const _Tp &__x)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker split (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const parents_iterator &__pr_it, const _Tp &__x)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker split_back (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const _Tp &__x)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker between_back (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__-child, const _Tp &__x)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker split_front (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const _Tp &__x)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator >
  walker between_front (const __SequenceCtr< walker, _Allocator > &__parents, const walker &_-_child, const _Tp &__x)

- _Self & operator= (const _RV_LDG &__rl)
- _Self & operator= (const erased_part &__ep)
- node_allocator_type get_node_allocator () const
- edge_allocator_type get_edge_allocator () const
- walker ground ()
- const_walker ground () const
- walker sky ()
- const_walker sky () const
- bool empty () const
- size_type size () const
- size_type max_size () const
- void swap (_Self &__x)
- walker insert_node_in_graph (_Node *__n, const walker &__parent, const walker &__child, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- walker insert_node_in_graph (_Node *__node, const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children)
- walker insert_node_in_graph (_Node *__node, const walker &__parent, const container_insert_arg &__pref, const __SequenceCtr< walker, _Allocator > &__children)
- walker insert_node_in_graph (_Node *__node, const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const container_insert_arg &__cref)
- walker insert_in_graph (const _Tp &__x, const walker &__parent, const walker &__child, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- walker insert_in_graph (const walker &__parent, const walker &__child, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- walker insert_in_graph (const _Tp &__x, const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children)
- walker insert_in_graph (const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children)
- walker insert_in_graph (const _Tp &__x, const walker &__parent, const container_insert_arg &__pref, const __SequenceCtr< walker, _Allocator > &__children)
- walker insert_in_graph (const walker &__parent, const container_insert_arg &__pref, const __SequenceCtr< walker, _Allocator > &__children)
- walker insert_in_graph (const _Tp &__x, const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const container_insert_arg &__cref)
- walker insert_in_graph (const __SequenceCtr< walker, _Allocator > &__parents, const walker &__child, const container_insert_arg &__cref)
- void insert_subgraph (_Self &__subgraph, const walker &__parent, const walker &__child, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- void insert_subgraph (_Self &__subgraph, const __SequenceCtr1< walker, _Allocator1 > &__parents, const __SequenceCtr2< walker, _Allocator2 > &__children)
- void add_edge (const edge &__edge, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- void add_edge (const walker &__parent, const walker &__child, const container_insert_arg &__Itc, const container_insert_arg &__Itp)
- void replace_edge_to_child (const walker &__parent, const walker &__child_old, const walker &__child_new)
- void replace_edge_to_parent (const walker &__parent_old, const walker &__parent_new, const walker &__child)
- void remove_edge (const edge &__edge)
- void remove_edge (const walker &__parent, const walker &__child)
- void remove_edge_and_deattach (const walker &__parent, const walker &__child)

- void sort_child_edges (walker __position, children_iterator first, children_iterator last, Compare comp)
- void sort_child_edges (walker __position, Compare comp)
- void sort_parent_edges (walker __position, parents_iterator first, parents_iterator last, Compare comp)
- void sort_parent_edges (walker __position, Compare comp)
- walker insert_node (_Node *_node, const walker &__position, const container_insert_arg &__It)
- walker insert_node (const _Tp &__x, const walker &__position, const container_insert_arg &__It)
- walker insert_node (const walker &__position, const container_insert_arg &__It)
- walker insert_node_before (_Node *_node, const walker &__position, const container_insert_arg &__It)
- void insert_node_before (const _Tp &__x, const walker &__position, const container_insert_arg &__It)
- void insert_node_before (const walker &__position, const container_insert_arg &__It)
- void merge (const walker &__position, const walker &__second, bool merge_parent_edges=true, bool merge_child_edges=true)
- void erase (const walker &__position)
- void partial_erase_to_parent (const walker &__position, const walker &__parent, unsigned int idx)
- void clear_erased_part (erased_part &_ep)
- erased_part erase_maximal_subgraph (const walker &__position)
- erased_part erase_maximal_subgraph (const __SequenceCtr< walker, _Allocator > &__positions)
- erased_part erase_minimal_subgraph (const walker &__position)
- erased_part erase_minimal_subgraph (const __SequenceCtr< walker, _Allocator > &__positions)
- erased_part erase_maximal_pregraph (const walker &__position)
- erased_part erase_maximal_pregraph (const __SequenceCtr< walker, _Allocator > &__positions)
- erased_part erase_minimal_pregraph (const walker &__position)
- erased_part erase_minimal_pregraph (const __SequenceCtr< walker, _Allocator > &__positions)
- bool erase_child (const walker &__position, const children_iterator &__It)
- bool erase_parent (const walker &__position, const parents_iterator &__It)

- out_iterator source_begin ()
- out_iterator root_begin ()

- out_const_iterator source_begin () const
- out_iterator root_begin ()

- out_iterator source_end ()
- out_iterator root_end ()

- out_const_iterator source_end () const
- out_iterator root_end ()

- in_iterator sink_begin ()
- in_iterator leaf_begin ()

- in_const_iterator sink_begin () const
- in_iterator leaf_begin ()

- in_iterator sink_end ()
- in_iterator leaf_end ()

- in_const_iterator sink_end () const
- in_iterator leaf_end ()

Protected Types

- typedef _Base::erased_part erased_part

Protected Member Functions

- _Node * _C_create_node (const _Tp &__x)
- _Node * _C_create_node ()
- _Edge * _C_create_edge (const _Te &__x)
- _Edge * _C_create_edge ()
- _Edge * _C_create_edge (const _Te &__x, _Node *__s, _Node *__t)
- _Edge * _C_create_edge (_Node *__s, _Node *__t)

9.37.1 Detailed Description

This class constructs a labeled directed graph. By default, the children and the parents are collected in an STL vector, but the container can be replaced by any other sequential container.

9.37.2 Member Typedef Documentation

9.37.2.1 typedef _Base::children_const_iterator ldgraph::children_const_iterator

the children const iterator

Reimplemented from __LDG< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::const_iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >.

Reimplemented in ldag.

Definition at line 2249 of file vgtl_ldag.h.

9.37.2.2 typedef _Base::children_iterator ldgraph::children_iterator

the children iterator

Reimplemented from __LDG< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::const_iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >.

Reimplemented in ldag.

Definition at line 2243 of file vgtl_ldag.h.

9.37.2.3 typedef _LDG_iterator<_Tp,const _Tp&,const _Tp*,container_type, children_iterator,children_const_iterator,_Te> __LDG::const_iterator [inherited]

the const iterator

Definition at line 651 of file vgtl_ldag.h.

9.37.2.4 typedef const value_type* __LDG::const_pointer [inherited]

standard typedef

Definition at line 630 of file vgtl_ldag.h.

9.37.2.5 typedef const value_type& __LDG::const_reference [inherited]

standard typedef

Definition at line 632 of file vgtl_ldag.h.

9.37.2.6 typedef std::reverse_iterator<const_iterator> __LDG::const_reverse_iterator [inherited]

the const reverse iterator

Definition at line 655 of file vgtl_ldag.h.

9.37.2.7 typedef _Base::const_walker ldgraph::const_walker

the const walker

Reimplemented from __LDG< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::const_iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >.

Reimplemented in ldag.

Definition at line 2241 of file vgtl_ldag.h.

9.37.2.8 typedef ptrdiff_t __LDG::difference_type [inherited]

standard typedef

Definition at line 634 of file vgtl_ldag.h.

9.37.2.9 typedef _Edge __LDG::edge_type [inherited]

standard typedef

Definition at line 628 of file vgtl_ldag.h.

9.37.2.10 typedef _Base::erased_part ldgraph::erased_part [protected]

an edge of the graph (parent, child) an edge with additiona information about erased ground/sky edges an erased subgraph which is not yet a new directed graph

Reimplemented from __LDG< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::const_iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >.

Reimplemented in ldag.

Definition at line 2235 of file vgtl_ldag.h.

9.37.2.11 typedef _LDG_iterator<_Tp,_Tp&,_Tp*,container_type,children_iterator, children_const_iterator,_Te> __LDG::iterator [inherited]

the iterator

Definition at line 648 of file vgtl_ldag.h.

9.37.2.12 typedef _Node __LDG::node_type [inherited]

standard typedef

Definition at line 627 of file vgtl_ldag.h.

9.37.2.13 typedef _Base::parents_const_iterator ldgraph::parents_const_iterator

the parents const iterator

Reimplemented from __LDG< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::const_iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >.

Reimplemented in ldag.

Definition at line 2247 of file vgtl_ldag.h.

9.37.2.14 typedef _Base::parents_iterator ldgraph::parents_iterator

the parents iterator

Reimplemented from __LDG< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::const_iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >.

Reimplemented in ldag.

Definition at line 2245 of file vgtl_ldag.h.

9.37.2.15 typedef value_type* __LDG::pointer [inherited]

standard typedef

Definition at line 629 of file vgtl_ldag.h.

9.37.2.16 typedef value_type& __LDG::reference [inherited]

standard typedef

Definition at line 631 of file vgtl_ldag.h.

9.37.2.17 typedef std::reverse_iterator<iterator> __LDG::reverse_iterator [inherited]

the reverse iterator

Definition at line 657 of file vgtl_ldag.h.

9.37.2.18 typedef size_t __LDG::size_type [inherited]

standard typedef

Definition at line 633 of file vgtl_ldag.h.

9.37.2.19 typedef _Tp __LDG::value_type [inherited]

standard typedef

Definition at line 626 of file vgtl_ldag.h.

9.37.2.20 typedef _Base::walker ldgraph::walker

the walker

Reimplemented from __LDG< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::const_iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >.

Reimplemented in ldag.

Definition at line 2239 of file vgtl_ldag.h.

9.37.3 Constructor & Destructor Documentation

9.37.3.1 ldgraph::ldgraph (const allocator_type & __a = allocator_type()) [inline, explicit]

standard constructor

Definition at line 2253 of file vgtl_ldag.h.

9.37.3.2 ldgraph::ldgraph (const _Self & __dg) [inline]

copy constructor

Definition at line 2256 of file vgtl_ldag.h.

9.37.3.3 ldgraph::ldgraph (const erased_part & __ep, const allocator_type & __a = allocator_type()) [inline]

constructor from an erased_part

Definition at line 2259 of file vgtl_ldag.h.

9.37.4 Member Function Documentation

9.37.4.1 _Edge* __LDG::_C_create_edge (const _Te & __x) [inline, protected, inherited]

construct a new graph edge containing data ___x

Definition at line 726 of file vgtl_ldag.h.

9.37.4.2 _Edge* __LDG::_C_create_edge ( ) [inline, protected, inherited]

construct a new graph edge containing default data

Definition at line 738 of file vgtl_ldag.h.

9.37.4.3 _Edge* __LDG::_C_create_edge (const _Te & __x, _Node * __s, _Node * __t) [inline, protected, inherited]

construct a new graph edge containing data __x with source __s and target __t.

Definition at line 751 of file vgtl_ldag.h.

9.37.4.4 _Edge* __LDG::_C_create_edge (_Node * __s, _Node * __t) [inline, protected, inherited]

construct a new graph edge containing default data with source __s and target __t.

Definition at line 766 of file vgtl_ldag.h.

9.37.4.5 _Node* __LDG::_C_create_node (const _Tp & __x) [inline, protected, inherited]

construct a new graph node containing data __x

Definition at line 698 of file vgtl_ldag.h.

9.37.4.6 _Node* __LDG::_C_create_node ( ) [inline, protected, inherited]

construct a new graph node containing default data

Definition at line 712 of file vgtl_ldag.h.

9.37.4.7 void __LDG::add_edge (const edge & __edge, const container_insert_arg & __itc, const container_insert_arg & __itp) [inline, inherited]

add one edge between two nodes at the positions described by __Itc and __Itp.

Definition at line 1191 of file vgtl_ldag.h.

9.37.4.8 void __LDG::add_edge (const walker & __parent, const walker & __child, const container_insert_arg & __itc, const container_insert_arg & __itp) [inline, inherited]

add an edge between __parent and __child at positions __Itc and __Itp, respectively

Definition at line 1200 of file vgtl_ldag.h.

9.37.4.9 void ldgraph::add_edge (const walker & __parent, const children_iterator & __ch_it, const walker & __child, const parents_iterator & __pa_it) [inline]

add an edge between __parent and __child at specific positions __ch_it and __pa_it.

Definition at line 2488 of file vgtl_ldag.h.

9.37.4.10 void ldgraph::add_edge_back (const walker & __parent, const walker & __child) [inline]

add an edge between __parent and __child at the end of the children and parents containers.

Definition at line 2498 of file vgtl_ldag.h.

9.37.4.11 void ldgraph::add_edge_front (const walker & __parent, const walker & __child) [inline]

add an edge between __parent and __child at the beginning of the children and parents containers.

Definition at line 2508 of file vgtl_ldag.h.

9.37.4.12 walker ldgraph::between (const walker & *__parent,* const children_iterator & *__cit,* const walker & *__child,* const parents_iterator & *__pit,* const _Tp & *__x*) [inline]

here a new node is inserted between a parent node and a child node but the previous bonds between the two are not broken, the node is always new with data ___x.

Definition at line 2276 of file vgtl_ldag.h.

9.37.4.13 template<template< class __Tp, class __AllocTp > class __SequenceCtr1, template< class __Tp, class __AllocTp > class __SequenceCtr2, class _Allocator1 , class _Allocator2 > walker ldgraph::between (const __SequenceCtr1< walker, _Allocator1 > & *__parents,* const __SequenceCtr2< walker, _Allocator2 > & *__children,* const _Tp & *__x*) [inline]

here a new node is inserted between many parents and many children but the previous bonds are not broken, the node is always new

Definition at line 2378 of file vgtl_ldag.h.

9.37.4.14 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker ldgraph::between (const walker & *__parent,* const children_iterator & *__cit,* const __SequenceCtr< walker, _Allocator > & *__children,* const _Tp & *__x*) [inline]

here a new node is inserted between one parent and many children but the previous bonds are not broken, the node is always new

Definition at line 2524 of file vgtl_ldag.h.

9.37.4.15 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker ldgraph::between (const __SequenceCtr< walker, _Allocator > & *__parents,* const walker & *__child,* const parents_iterator & *__pit,* const _Tp & *__x*) [inline]

here a new node is inserted between many parents and one child but the previous bonds are not broken, the node is always new

Definition at line 2624 of file vgtl_ldag.h.

9.37.4.16 walker ldgraph::between_back (const walker & *__parent,* const walker & *__child,* const _Tp & *__x*) [inline]

insert the node as the last child between parent and child, without breaking old bonds.

Definition at line 2311 of file vgtl_ldag.h.

9.37.4.17 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker ldgraph::between_back (const walker & *__parent,* const __SequenceCtr< walker, _Allocator > & *__children,* const _Tp & *__x*) [inline]

here a new node is inserted between one parent and many children but the previous bonds are not broken, the node is always new. At the parent the new child is put last.

Definition at line 2579 of file vgtl_ldag.h.

9.37.4.18 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker ldgraph::between_back (const __SequenceCtr< walker, _Allocator > & *__parents,* const walker & *__child,* const _Tp & *__x*) [inline]

here a new node is inserted between many parents and one child but the previous bonds are not broken, the node is always new. At the child the new parent is put last.

Definition at line 2678 of file vgtl_ldag.h.

9.37.4.19 walker ldgraph::between_front (const walker & __parent, const walker & __child, const _Tp & __x) [inline]

Here the inserted node is the first child of its parent and first parent of its child. Insert the node without breaking old bonds.

Definition at line 2342 of file vgtl_ldag.h.

9.37.4.20 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker ldgraph::between_front (const walker & __parent, const __SequenceCtr< walker, _Allocator > & __children, const _Tp & __x) [inline]

here a new node is inserted between one parent and many children but the previous bonds are not broken, the node is always new. At the parent the new child is put first.

Definition at line 2609 of file vgtl_ldag.h.

9.37.4.21 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker ldgraph::between_front (const __SequenceCtr< walker, _Allocator > & __parents, const walker & __child, const _Tp & __x) [inline]

here a new node is inserted between many parents and one child but the previous bonds are not broken, the node is always new. At the child the new parent is put first.

Definition at line 2706 of file vgtl_ldag.h.

9.37.4.22 void ldgraph::clear ( ) [inline]

empty the graph

Reimplemented from __LDG< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::const_iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >.

Definition at line 2269 of file vgtl_ldag.h.

9.37.4.23 void __LDG::clear_erased_part (erased_part & _ep) [inline, inherited]

clear all nodes in an erased part

Definition at line 1868 of file vgtl_ldag.h.

9.37.4.24 bool __LDG::empty ( ) const [inline, inherited]

returns `true` if the DG is empty

Definition at line 888 of file vgtl_ldag.h.

9.37.4.25 void __LDG::erase (const walker & __position) [inline, inherited]

erase a node from the DG except the sky and ground

Definition at line 1518 of file vgtl_ldag.h.

9.37.4.26 bool __LDG::erase_child (const walker & __position, const children_iterator & __it) [inline, inherited]

Erase a child of `__position`. This works if and only if the child has only one child and no other parents.

Definition at line 2020 of file vgtl_ldag.h.

9.37.4.27 erased_part __LDG::erase_maximal_pregraph (const walker & __position) [inline, inherited]

here every child is removed till the sky node. included the node at __position. The removed subgraph is returned. The subgraph is maximal, i.e. all nodes are removed, which are reachable from __position by walking upwards.

Definition at line 1950 of file vgtl_ldag.h.

9.37.4.28 erased_part __LDG::erase_maximal_pregraph (const __SequenceCtr< walker, _Allocator > & __positions) [inline, inherited]

here every child is removed till the sky included all nodes from __positions. The removed subgraph is returned. The subgraph is maximal, i.e. all nodes are removed, which are reachable from any node from __positions by walking up.

Definition at line 1984 of file vgtl_ldag.h.

9.37.4.29 erased_part __LDG::erase_maximal_subgraph (const walker & __position) [inline, inherited]

here every child is removed till the last base node, included the node at __position. The removed subgraph is returned. The subgraph is maximal, i.e. all nodes are removed, which are reachable from __position by walking down.

Definition at line 1879 of file vgtl_ldag.h.

9.37.4.30 erased_part __LDG::erase_maximal_subgraph (const __SequenceCtr< walker, _Allocator > & __positions) [inline, inherited]

here every child is removed till the last base node, included all nodes from __positions. The removed subgraph is returned. The subgraph is maximal, i.e. all nodes are removed, which are reachable from any node from __positions by walking down.

Definition at line 1913 of file vgtl_ldag.h.

9.37.4.31 erased_part __LDG::erase_minimal_pregraph (const walker & __position) [inline, inherited]

here every child is removed till the sky. included the node at __position. The removed subgraph is returned. The subgraph is minimal, i.e. only nodes are removed, which have no other descendant than __position. I.e., when walking towards the sky, there is no way which bypasses __position.

Definition at line 1966 of file vgtl_ldag.h.

9.37.4.32 erased_part __LDG::erase_minimal_pregraph (const __SequenceCtr< walker, _Allocator > & __positions) [inline, inherited]

here every child is removed till the sky. included all nodes from __positions. The removed subgraph is returned. The subgraph is minimal, i.e. only nodes are removed, which have no other ancestor than any node in __positions. I.e., when walking towards the ground, there is no way which bypasses all nodes in __positions.

Definition at line 2004 of file vgtl_ldag.h.

9.37.4.33 erased_part __LDG::erase_minimal_subgraph (const walker & __position) [inline, inherited]

here every child is removed till the last base node, included the node at __position. The removed subgraph is returned. The subgraph is minimal, i.e. only nodes are removed, which have no other ancestor than __position. I.e., when walking towards the ground, there is no way which bypasses __position.

Definition at line 1895 of file vgtl_ldag.h.

9.37.4.34 erased_part __LDG::erase_minimal_subgraph (const __SequenceCtr< walker, _Allocator > & __positions) [inline, inherited]

here every child is removed till the last base node, included all nodes from __positions. The removed subgraph is returned. The subgraph is minimal, i.e. only nodes are removed, which have no other ancestor than any node in __positions. I.e., when walking towards the ground, there is no way which bypasses all nodes in __positions.

Definition at line 1933 of file vgtl_ldag.h.

9.37.4.35 bool __LDG::erase_parent (const walker & __position, const parents_iterator & __it) [inline, inherited]

Erase a parent of __position. This works if and only if the parent has only one parent and no other children.

Definition at line 2046 of file vgtl_ldag.h.

9.37.4.36 edge_allocator_type __LDG::get_edge_allocator ( ) const [inline, inherited]

construct an edge allocator object

Definition at line 643 of file vgtl_ldag.h.

9.37.4.37 node_allocator_type __LDG::get_node_allocator ( ) const [inline, inherited]

construct a node allocator object

Definition at line 639 of file vgtl_ldag.h.

9.37.4.38 walker __LDG::ground ( ) [inline, inherited]

return a walker to the virtual ground node.

Definition at line 784 of file vgtl_ldag.h.

9.37.4.39 const_walker __LDG::ground ( ) const [inline, inherited]

return a const walker to the virtual ground node.

Definition at line 794 of file vgtl_ldag.h.

9.37.4.40 void ldgraph::insert_back_subgraph (_Self & __subgraph, const walker & __parent, const walker & __child) [inline]

here a subgraph is inserted between a parent and a child, at the end of the children resp. parents lists.

Definition at line 2447 of file vgtl_ldag.h.

9.37.4.41 void ldgraph::insert_front_subgraph (_Self & __subgraph, const walker & __parent, const walker & __child) [inline]

here a subgraph is inserted between a parent and a child, at the front of the children resp. parents lists.

Definition at line 2460 of file vgtl_ldag.h.

9.37.4.42 walker __LDG::insert_in_graph (const _Tp & __x, const walker & __parent, const walker & __child, const container_insert_arg & __ltc, const container_insert_arg & __ltp) [inline, inherited]

insert node with data __n into the graph between __parent and __child, the edge at the specific positions described by __Itc and __Itp.

Definition at line 928 of file vgtl_ldag.h.

9.37.4.43 walker __LDG::insert_in_graph (const walker & __parent, const walker & __child, const container_insert_arg & __ltc, const container_insert_arg & __ltp) [inline, inherited]

insert node with default data into the graph between __parent and __child, the edge at the specific positions described by __Itc and __Itp.

Definition at line 942 of file vgtl_ldag.h.

9.37.4.44 walker __LDG::insert_in_graph (const _Tp & __x, const __SequenceCtr1< walker, _Allocator1 > & __parents, const __SequenceCtr2< walker, _Allocator2 > & __children) [inline, inherited]

insert a node with data __x into the graph between all parents from __parents and all children from __children.

Definition at line 1006 of file vgtl_ldag.h.

9.37.4.45 walker __LDG::insert_in_graph (const __SequenceCtr1< walker, _Allocator1 > & __parents, const __SequenceCtr2< walker, _Allocator2 > & __children) [inline, inherited]

insert a node with default data into the graph between all parents from __parents and all children from __children.

Definition at line 1021 of file vgtl_ldag.h.

9.37.4.46 walker __LDG::insert_in_graph (const _Tp & __x, const walker & __parent, const container_insert_arg & __pref, const __SequenceCtr< walker, _Allocator > & __children) [inline, inherited]

insert a node with data __x into the graph between the parent __parent and all children from __children.

Definition at line 1059 of file vgtl_ldag.h.

9.37.4.47 walker __LDG::insert_in_graph (const walker & __parent, const container_insert_arg & __pref, const __SequenceCtr< walker, _Allocator > & __children) [inline, inherited]

insert a node with data __x into the graph between the parent __parent and all children from __children.

Definition at line 1073 of file vgtl_ldag.h.

9.37.4.48 walker __LDG::insert_in_graph (const _Tp & __x, const __SequenceCtr< walker, _Allocator > & __parents, const walker & __child, const container_insert_arg & __cref) [inline, inherited]

insert a node with data __x into the graph between all parents from __parents and the child __child.

Definition at line 1112 of file vgtl_ldag.h.

9.37.4.49 walker __LDG::insert_in_graph (const __SequenceCtr< walker, _Allocator > & __parents, const walker & __child, const container_insert_arg & __cref) [inline, inherited]

insert a node with default data into the graph between all parents from __parents and the child __child.

Definition at line 1127 of file vgtl_ldag.h.

9.37.4.50 walker __LDG::insert_node (_Node * _node, const walker & __position, const container_insert_arg & __it) [inline, inherited]

insert one node as child of __position

Definition at line 1379 of file vgtl_ldag.h.

9.37.4.51 walker __LDG::insert_node (const _Tp & __x, const walker & __position, const container_insert_arg & __it) [inline, inherited]

insert a new node with data __x as child of __position

Definition at line 1393 of file vgtl_ldag.h.

9.37.4.52 walker __LDG::insert_node (const walker & __position, const container_insert_arg & __it) [inline, inherited]

insert a new node with default data as child of __position

Definition at line 1399 of file vgtl_ldag.h.

9.37.4.53 walker __LDG::insert_node_before (_Node * _node, const walker & __position, const container_insert_arg & __it) [inline, inherited]

insert a node as parent of __position

Definition at line 1404 of file vgtl_ldag.h.

9.37.4.54 void __LDG::insert_node_before (const _Tp & __x, const walker & __position, const container_insert_arg & __it) [inline, inherited]

insert a new node with data __x as parent of __position

Definition at line 1418 of file vgtl_ldag.h.

9.37.4.55 void __LDG::insert_node_before (const walker & __position, const container_insert_arg & __it) [inline, inherited]

insert a new node with default data as parent of __position

Definition at line 1423 of file vgtl_ldag.h.

9.37.4.56 walker __LDG::insert_node_in_graph (_Node * __n, const walker & __parent, const walker & __child, const container_insert_arg & __Itc, const container_insert_arg & __Itp) [inline, inherited]

insert node __n into the graph between __parent and __child, the edge at the specific positions described by __Itc and __Itp.

Definition at line 912 of file vgtl_ldag.h.

9.37.4.57 walker __LDG::insert_node_in_graph (_Node * __node, const __SequenceCtr1< walker, _Allocator1 > & __parents, const __SequenceCtr2< walker, _Allocator2 > & __children) [inline, inherited]

insert node __n into the graph between all parents from __parents and all children from __children.

Definition at line 975 of file vgtl_ldag.h.

9.37.4.58 walker __LDG::insert_node_in_graph (_Node * __node, const walker & __parent, const container_insert_arg & __pref, const __SequenceCtr< walker, _Allocator > & __children) [inline, inherited]

insert node __n into the graph between the parent __parent and all children from __children.

Definition at line 1034 of file vgtl_ldag.h.

9.37.4.59 walker __LDG::insert_node_in_graph (_Node * __node, const __SequenceCtr< walker, _Allocator > & __parents, const walker & __child, const container_insert_arg & __cref) [inline, inherited]

insert node __n into the graph between all parents from __parents and the child __child.

Definition at line 1087 of file vgtl_ldag.h.

9.37.4.60 void __LDG::insert_subgraph (_Self & __subgraph, const walker & __parent, const walker & __child, const container_insert_arg & __Itc, const container_insert_arg & __Itp) [inline, inherited]

insert a subgraph into the graph between __parent and __child, the edge at the specific positions described by __Itc and __Itp.

Definition at line 953 of file vgtl_ldag.h.

9.37.4.61 void __LDG::insert_subgraph (_Self & __subgraph, const __SequenceCtr1< walker, _Allocator1 > & __parents, const __SequenceCtr2< walker, _Allocator2 > & __children) [inline, inherited]

in this method one DG is inserted into another DG between the parents __parents and the children __children.

Definition at line 1141 of file vgtl_ldag.h.

9.37.4.62 void ldgraph::insert_subgraph (_Self & __subgraph, const walker & __parent, const children_iterator & __ch_it, const walker & __child, const parents_iterator & __pa_it) [inline]

here a subgraph is inserted between a parent and a child, at specific positions __ch_it and __pa_it.

Definition at line 2436 of file vgtl_ldag.h.

9.37.4.63 in_iterator __LDG::leaf_begin ( ) [inline, inherited]

return the first local sink of the directed graph

Definition at line 833 of file vgtl_ldag.h.

9.37.4.64 in_iterator __LDG::leaf_begin ( ) [inline, inherited]

return the first local sink of the directed graph

Definition at line 846 of file vgtl_ldag.h.

9.37.4.65 in_iterator __LDG::leaf_end ( ) [inline, inherited]

return beyond the last local sink of the directed graph

Definition at line 839 of file vgtl_ldag.h.

9.37.4.66 in_iterator __LDG::leaf_end ( ) [inline, inherited]

return beyond the last local sink of the directed graph

Definition at line 852 of file vgtl_ldag.h.

9.37.4.67 size_type __LDG::max_size ( ) const [inline, inherited]

the maximum size of a DG is virtually unlimited

Definition at line 899 of file vgtl_ldag.h.

9.37.4.68 void __LDG::merge (const walker & *__position*, const walker & *__second*, bool *merge_parent_edges* = true, bool *merge_child_edges* = true) [inline, inherited]

merge two nodes, call also the merge method for the node data

Definition at line 1429 of file vgtl_ldag.h.

9.37.4.69 _Self& ldgraph::operator= (const _RV_LDG & *__rl*) [inline]

assignment operator from a part of an erased part

Reimplemented from __LDG< _Tp, _SequenceCtr< void *>, _PtrAlloc >, _SequenceCtr< void *>, _PtrAlloc >::iterator, _SequenceCtr< void *>, _PtrAlloc >::const_iterator, _SequenceCtr< void *>, _PtrAlloc >::iterator, _Alloc >.

Definition at line 2715 of file vgtl_ldag.h.

9.37.4.70 _Self& ldgraph::operator= (const erased_part & *__ep*) [inline]

assignment operator from an erased part

Reimplemented from __LDG< _Tp, _SequenceCtr< void *>, _PtrAlloc >, _SequenceCtr< void *>, _PtrAlloc >::iterator, _SequenceCtr< void *>, _PtrAlloc >::const_iterator, _SequenceCtr< void *>, _PtrAlloc >::iterator, _Alloc >.

Definition at line 2723 of file vgtl_ldag.h.

9.37.4.71 void __LDG::partial_erase_to_parent (const walker & *__position,* const walker & *__parent,* unsigned int *idx*) [inline, inherited]

split a node in two, the first connected to the __parent, the second connected to all other parents. Then erase the first node.

Definition at line 1578 of file vgtl_ldag.h.

9.37.4.72 void __LDG::remove_edge (const edge & *__edge*) [inline, inherited]

remove an edge with a particular parent and child

Definition at line 1315 of file vgtl_ldag.h.

9.37.4.73 void __LDG::remove_edge (const walker & *__parent,* const walker & *__child*) [inline, inherited]

just remove one edge between __parent and __child

Definition at line 1332 of file vgtl_ldag.h.

9.37.4.74 void __LDG::remove_edge_and_deattach (const walker & *__parent,* const walker & *__child*) [inline, inherited]

remove one egde and don't reconnect the node to sky/ground

Definition at line 1319 of file vgtl_ldag.h.

9.37.4.75 void __LDG::replace_edge_to_child (const walker & *__parent,* const walker & *__child_old,* const walker & *__child_new*) [inline, inherited]

change the edge from __parent to __child_old to an edge from __parent to __child_new.

Definition at line 1243 of file vgtl_ldag.h.

9.37.4.76 void __LDG::replace_edge_to_parent (const walker & *__parent_old,* const walker & *__parent_new,* const walker & *__child*) [inline, inherited]

change the edge from __parent_old to __child to an edge from __parent_new to __child.

Definition at line 1281 of file vgtl_ldag.h.

9.37.4.77 out_iterator __LDG::root_begin ( ) [inline, inherited]

return the first local source of the directed graph

Definition at line 807 of file vgtl_ldag.h.

9.37.4.78 out_iterator __LDG::root_begin ( ) [inline, inherited]

return the first local source of the directed graph

Definition at line 820 of file vgtl_ldag.h.

9.37.4.79 out_iterator __LDG::root_end ( ) [inline, inherited]

return beyond the last local source of the directed graph

Definition at line 813 of file vgtl_ldag.h.

9.37.4.80 out_iterator __LDG::root_end ( ) [inline, inherited]

return beyond the last local source of the directed graph

Definition at line 826 of file vgtl_ldag.h.

9.37.4.81 in_iterator __LDG::sink_begin ( ) [inline, inherited]

return the first local sink of the directed graph

Definition at line 831 of file vgtl_ldag.h.

9.37.4.82 in_const_iterator __LDG::sink_begin ( ) const [inline, inherited]

return the first local sink of the directed graph

Definition at line 844 of file vgtl_ldag.h.

9.37.4.83 in_iterator __LDG::sink_end ( ) [inline, inherited]

return beyond the last local sink of the directed graph

Definition at line 837 of file vgtl_ldag.h.

9.37.4.84 in_const_iterator __LDG::sink_end ( ) const [inline, inherited]

return beyond the last local sink of the directed graph

Definition at line 850 of file vgtl_ldag.h.

9.37.4.85 size_type __LDG::size ( ) const [inline, inherited]

returns the size of the DG (number of nodes)

Definition at line 892 of file vgtl_ldag.h.

9.37.4.86 walker __LDG::sky ( ) [inline, inherited]

return a walker to the virtual sky node.

Definition at line 789 of file vgtl_ldag.h.

9.37.4.87 const_walker __LDG::sky ( ) const [inline, inherited]

return a const walker to the virtual sky node.

Definition at line 799 of file vgtl_ldag.h.

9.37.4.88 void __LDG::sort_child_edges (walker *__position*, children_iterator *first*, children_iterator *last*, Compare *comp*) [inline, inherited]

sort the child edges in the range [first,last) according to `comp`

Definition at line 1356 of file vgtl_ldag.h.

9.37.4.89 void __LDG::sort_child_edges (walker *__position*, Compare *comp*) [inline, inherited]

sort all child edges according to `comp`

Definition at line 1368 of file vgtl_ldag.h.

9.37.4.90 void __LDG::sort_parent_edges (walker __*position*, parents_iterator *first*, parents_iterator *last*, Compare *comp*) [inline, inherited]

sort the parent edges in the range [first,last) according to comp

Definition at line 1362 of file vgtl_ldag.h.

9.37.4.91 void __LDG::sort_parent_edges (walker __*position*, Compare *comp*) [inline, inherited]

sort all parent edges according to comp

Definition at line 1374 of file vgtl_ldag.h.

9.37.4.92 out_iterator __LDG::source_begin ( ) [inline, inherited]

return the first local source of the directed graph

Definition at line 805 of file vgtl_ldag.h.

9.37.4.93 out_const_iterator __LDG::source_begin ( ) const [inline, inherited]

return the first local source of the directed graph

Definition at line 818 of file vgtl_ldag.h.

9.37.4.94 out_iterator __LDG::source_end ( ) [inline, inherited]

return beyond the last local source of the directed graph

Definition at line 811 of file vgtl_ldag.h.

9.37.4.95 out_const_iterator __LDG::source_end ( ) const [inline, inherited]

return beyond the last local source of the directed graph

Definition at line 824 of file vgtl_ldag.h.

9.37.4.96 walker ldgraph::split (const walker & __*parent*, const children_iterator & __*ch_it*, const walker & __*child*, const parents_iterator & __*pa_it*, const _Tp & __*x*) [inline]

here a new node is inserted between a parent node and a child node and the previous bonds between them are broken, the node is always new with data ___x.

Definition at line 2289 of file vgtl_ldag.h.

9.37.4.97 template<template< class __Tp, class __AllocTp > class __SequenceCtr1, template< class __Tp, class __AllocTp > class __SequenceCtr2, class _Allocator1 , class _Allocator2 > void ldgraph::split (const __SequenceCtr1< walker, _Allocator1 > & __*parents*, const __SequenceCtr2< walker, _Allocator2 > & __*children*, const _Tp & __*x*) [inline]

here a new node is inserted between many parents and many children, and the previous bonds are broken, the node is always new.

Definition at line 2410 of file vgtl_ldag.h.

9.37.4.98 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker ldgraph::split (const walker & *__parent,* const children_iterator & *__ch_it,* const __SequenceCtr< walker, _Allocator > & *__children,* const _Tp & *__x*) [inline]

here a new node is inserted between one parent and many children, and the previous bonds are broken, the node is always new.

Definition at line 2537 of file vgtl_ldag.h.

9.37.4.99 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker ldgraph::split (const __SequenceCtr< walker, _Allocator > & *__parents,* const walker & *__child,* const parents_iterator & *__pr_it,* const _Tp & *__x*) [inline]

here a new node is inserted between many parents and one child, and the previous bonds are broken, the node is always new.

Definition at line 2637 of file vgtl_ldag.h.

9.37.4.100 walker ldgraph::split_back (const walker & *__parent,* const walker & *__child,* const _Tp & *__x*) [inline]

insert the node as the last child between parent and child, with breaking old bonds.

Definition at line 2324 of file vgtl_ldag.h.

9.37.4.101 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker ldgraph::split_back (const walker & *__parent,* const __SequenceCtr< walker, _Allocator > & *__children,* const _Tp & *__x*) [inline]

here a new node is inserted between one parent and many children, and the previous bonds are broken, the node is always new. At the parent the new child is put last.

Definition at line 2564 of file vgtl_ldag.h.

9.37.4.102 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker ldgraph::split_back (const __SequenceCtr< walker, _Allocator > & *__parents,* const walker & *__child,* const _Tp & *__x*) [inline]

here a new node is inserted between many parents and one child, and the previous bonds are broken, the node is always new. At the child the new parent is put last.

Definition at line 2664 of file vgtl_ldag.h.

9.37.4.103 walker ldgraph::split_front (const walker & *__parent,* const walker & *__child,* const _Tp & *__x*) [inline]

Here the inserted node is the first child of its parent and first parent of its child. Insert the node and break old bonds.

Definition at line 2355 of file vgtl_ldag.h.

9.37.4.104 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker ldgraph::split_front (const walker & *__parent,* const __SequenceCtr< walker, _Allocator > & *__children,* const _Tp & *__x*) [inline]

here a new node is inserted between one parent and many children, and the previous bonds are broken, the node is always new. At the parent the new child is put first.

Definition at line 2594 of file vgtl_ldag.h.

9.37.4.105 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class _Allocator > walker ldgraph::split_front (const __SequenceCtr< walker, _Allocator > & *__parents,* const walker & *__child,* const _Tp & *__x*) [inline]

here a new node is inserted between many parents and one child, and the previous bonds are broken, the node is always new. At the child the new parent is put first.

Definition at line 2692 of file vgtl_ldag.h.

9.37.4.106 void __LDG::swap (_Self & *__x*) [inline, inherited]

swap two DGs

Definition at line 902 of file vgtl_ldag.h.

The documentation for this class was generated from the following file:

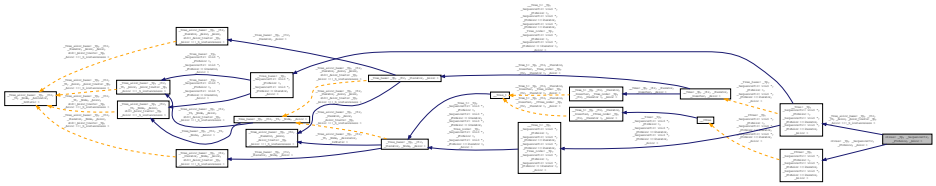
- vgtl_ldag.h

9.38 ntree< _Tp, _SequenceCtr, _PtrAlloc, _Alloc > Class Template Reference

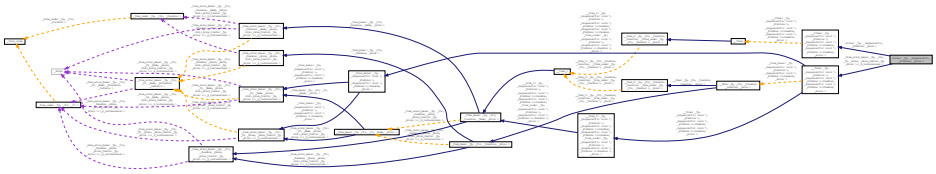
n-ary forest

`#include <vgtl_tree.h>`

Inheritance diagram for ntree< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >:



Collaboration diagram for ntree< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >:



Public Types

- typedef _Tree_iterator< _Tp, _Tp &, _Tp *, container_type, children_iterator, node_type > iterator
- typedef _Tree_iterator< _Tp, const _Tp &, const _Tp *, container_type, children_iterator, node_type > const_iterator
- typedef _Tree_walker< _Tp, _Tp &, _Tp *, container_type, children_iterator, _Node > iterative_walker
- typedef _Tree_walker< _Tp, const _Tp &, const _Tp *, container_type, children_iterator, _Node > const_iterative_walker
- typedef std::reverse_iterator < const_iterator > const_reverse_iterator

- typedef std::reverse_iterator < iterator > reverse_iterator
- typedef _Tp value_type
- typedef _Node node_type
- typedef value_type * pointer
- typedef const value_type * const_pointer
- typedef value_type & reference
- typedef const value_type & const_reference
- typedef size_t size_type
- typedef ptrdiff_t difference_type
- typedef _Tree_iterator< _Tp, _Tp &, _Tp *, container_type, container_iterator > iterator
- typedef _Tree_iterator< _Tp, const _Tp &, const _Tp *, container_type, container_iterator > const_iterator
- typedef reverse_iterator < const_iterator > const_reverse_iterator
- typedef reverse_iterator < iterator > reverse_iterator
- typedef _Tree_walker< _Tp, _Tp &, _Tp *, container_type, container_iterator > walker
- typedef _Tree_walker< _Tp, const _Tp &, const _Tp *, container_type, container_iterator > const_walker
- typedef _Iterator children_iterator
- typedef _TI children_iterator
- typedef __one_iterator< void * > parents_iterator
- typedef __one_iterator< void * > parents_iterator

Public Member Functions

- void insert (const __walker_base &__position, const _Tp &__x)
- void insert (const __walker_base &__position)
- void push_child (const __walker_base &__position, const _Tp &__x)
- void push_child (const __walker_base &__position)
- void push_children (const __walker_base &__position, size_type __n, const _Tp &__x)
- void push_children (const __walker_base &__position, size_type __n)
- void unshift_child (const __walker_base &__position, const _Tp &__x)
- void unshift_child (const __walker_base &__position)
- void unshift_children (const __walker_base &__position, size_type __n, const _Tp &__x)
- void unshift_children (const __walker_base &__position, size_type __n)
- void push_subtree (const __walker_base &__position, _Self &__subtree)
- void unshift_subtree (const __walker_base &__position, _Self &__subtree)
- bool pop_child (const __walker_base &__position)
- bool shift_child (const __walker_base &__position)
- _Node * pop_subtree (const __walker_base &__position)
- _Node * shift_subtree (const __walker_base &__position)
- _Self & operator= (_Node *__x)
- iterative_walker root (walker_type wt=cw_pre_post, bool front_to_back=true, bool depth_first=true)
- const_iterative_walker root (walker_type wt=cw_pre_post, bool front_to_back=true, bool depth_first=true) const
- iterative_walker through ()
- const_iterative_walker through () const
- iterative_walker begin (walker_type wt=cw_pre_post, bool front_to_back=true, bool depth_first=true)
- const_iterative_walker begin (walker_type wt=cw_pre_post, bool front_to_back=true, bool depth_first=true) const
- iterative_walker end (walker_type wt=cw_pre_post, bool front_to_back=true, bool depth_first=true)

- const_iterative_walker end (walker_type wt=cw_pre_post, bool front_to_back=true, bool depth_-first=true) const
- reverse_iterator rbegin ()
- const_reverse_iterator rbegin () const
- reverse_iterator rend ()
- const_reverse_iterator rend () const
- size_type size () const
- reference getroot ()
- const_reference getroot () const
- size_type depth (const iterative_walker &__position)
- allocator_type get_allocator () const
- walker root (children_iterator __it)
- const_walker root (children_iterator __it) const
- walker root ()
- const_walker root () const
- iterator begin ()
- const_iterator begin () const
- iterator end ()
- const_iterator end () const
- bool empty () const
- size_type max_size () const
- void **swap** (_Self &__x)
- void insert_child (const __walker_base &__position, const _Tp &__x, const container_insert_arg &__It)
- void insert_child (const __walker_base &__position, const container_insert_arg &__It)
- void insert_children (const __walker_base &__position, size_type __n, const _Tp &__x, const children-_iterator &__It)
- void insert_subtree (const __walker_base &__position, _Self &__subtree, const children_iterator &__It)
- void erase (const __walker_base &__position)
- _Node * erase_tree (const __walker_base &__position)
- bool erase_child (const __walker_base &__position, const children_iterator &__It)
- _Node * erase_subtree (const __walker_base &__position, const children_iterator &__It)
- size_type **depth** (const walker &__position)
- walker ground ()
- const_walker ground () const
- void clear_children ()
- void add_all_children (_Output_Iterator fi, _Node *_parent)
- template<class _Output_Iterator >
  void add_all_children (_Output_Iterator fi, _Node *_parent)

Protected Member Functions

- _Node * _C_create_node (const _Tp &__x)
- _Node * _C_create_node ()
- _Node * _C_get_node ()
- void _C_put_node (_Node *__p)
- void _C_put_node (_Node *__p)
- void _C_put_node (_Node *__p)
- void _C_put_node (_Alloc *__p)

Protected Attributes

- _Node * _C_node

Friends

- bool operator==__VGTL_NULL_TMPL_ARGS (const __ITree &__x, const __ITree &__y)

9.38.1 Detailed Description

template<class _Tp, template< class __Ty, class __AllocT > class _SequenceCtr = std::vector, class _PtrAlloc = __VGTL_DEFAULT_ALLOCATOR(void *), class _Alloc = __VGTL_DEFAULT_ALLOCATOR(_Tp)>class ntree< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >

This class constructs an n-ary forest with data hooks. By default, the children are collected in a STL vector, but the container can be replaced by any other sequential container.

Definition at line 2322 of file vgtl_tree.h.

9.38.2 Member Typedef Documentation

9.38.2.1 template<class _Tp, class _Ctr, class _Tl, class _Node, class _Alloc> typedef _Tl _Tree_base< _Tp, _Ctr, _Tl, _Node, _Alloc >::children_iterator [inherited]

iterator for accessing the children

Reimplemented in __Tree_t.

Definition at line 1445 of file vgtl_tree.h.

9.38.2.2 typedef _Iterator __Tree_t::children_iterator [inherited]

iterator for accessing the children

Reimplemented from _Tree_base< _Tp, _Ctr, _Iterator, _Node, _Alloc >.

Definition at line 1563 of file vgtl_tree.h.

9.38.2.3 typedef _Tree_walker<_Tp,const _Tp&,const _Tp*,container_type,children_iterator,_Node> __ITree::const_iterative_walker [inherited]

the const iterative walker

Definition at line 2065 of file vgtl_tree.h.

9.38.2.4 typedef _Tree_iterator<_Tp,const _Tp&,const _Tp*,container_type,container_iterator> __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::const_iterator [inherited]

the const iterator

Reimplemented from __Tree_t.

Definition at line 1263 of file vgtl_graph.h.

9.38.2.5 typedef _Tree_iterator<_Tp,const _Tp&,const _Tp*,container_type,children_iterator,node_type> __ITree::const_iterator [inherited]

the const iterator

Definition at line 2060 of file vgtl_tree.h.

9.38.2.6 typedef const value_type* __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::const_pointer [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1251 of file vgtl_graph.h.

9.38.2.7 typedef const value_type& __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::const_reference [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1253 of file vgtl_graph.h.

9.38.2.8 typedef reverse_iterator<const_iterator> __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::const_reverse_iterator [inherited]

the const reverse iterator

Reimplemented from __Tree_t.

Definition at line 1266 of file vgtl_graph.h.

9.38.2.9 typedef std::reverse_iterator<const_iterator> __ITree::const_reverse_iterator [inherited]

the const reverse iterator

Definition at line 2069 of file vgtl_tree.h.

9.38.2.10 typedef _Tree_walker<_Tp,const _Tp&,const _Tp*,container_type,container_iterator> __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::const_walker [inherited]

the (recursive) const walker

Reimplemented from __Tree_t.

Definition at line 1278 of file vgtl_graph.h.

9.38.2.11 typedef ptrdiff_t __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::difference_type [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1255 of file vgtl_graph.h.

9.38.2.12 typedef _Tree_walker<_Tp,_Tp&,_Tp*,container_type,children_iterator,_Node> __ITree::iterative_walker [inherited]

the iterative walker

Definition at line 2063 of file vgtl_tree.h.

9.38.2.13 typedef _Tree_iterator<_Tp,_Tp&,_Tp*,container_type,container_iterator> __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::iterator [inherited]

the iterator

Reimplemented from __Tree_t.

Definition at line 1262 of file vgtl_graph.h.

9.38.2.14 typedef _Tree_iterator<_Tp,_Tp&,_Tp*,container_type,children_iterator,node_type> __ITree::iterator [inherited]

the iterator

Definition at line 2058 of file vgtl_tree.h.

9.38.2.15 typedef _Node __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::node_type [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1249 of file vgtl_graph.h.

9.38.2.16 template<class _Tp, class _Ctr, class _Tl, class _Node, class _Alloc> typedef __one_iterator<void *> _Tree_base< _Tp, _Ctr, _Tl, _Node, _Alloc >::parents_iterator [inherited]

iterator for accessing the parents

Reimplemented in __Tree_t.

Definition at line 1447 of file vgtl_tree.h.

9.38.2.17 typedef __one_iterator<void *> __Tree_t::parents_iterator [inherited]

iterator for accessing the parents

Reimplemented from _Tree_base< _Tp, _Ctr, _Iterator, _Node, _Alloc >.

Definition at line 1564 of file vgtl_tree.h.

9.38.2.18 typedef value_type* __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::pointer [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1250 of file vgtl_graph.h.

9.38.2.19 typedef value_type& __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::reference [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1252 of file vgtl_graph.h.

9.38.2.20 typedef reverse_iterator<iterator> __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::reverse_iterator [inherited]

the reverse iterator

Reimplemented from __Tree_t.

Definition at line 1267 of file vgtl_graph.h.

9.38.2.21 typedef std::reverse_iterator<iterator> __ITree::reverse_iterator [inherited]

the reverse iterator

Definition at line 2071 of file vgtl_tree.h.

9.38.2.22 typedef size_t __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::size_type [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1254 of file vgtl_graph.h.

9.38.2.23 typedef _Tp __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::value_type [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1248 of file vgtl_graph.h.

9.38.2.24 typedef _Tree_walker<_Tp,_Tp&,_Tp*,container_type,container_iterator> __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::walker [inherited]

the (recursive) walker

Reimplemented from __Tree_t.

Definition at line 1277 of file vgtl_graph.h.

9.38.3 Member Function Documentation

9.38.3.1 _Node* __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::_C_create_node (const _Tp & __x) [inline, protected, inherited]

construct a new tree node containing data __x

Reimplemented from __Tree_t.

Definition at line 1295 of file vgtl_graph.h.

9.38.3.2 _Node* __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::_C_create_node ( ) [inline, protected, inherited]

construct a new tree node containing default data

Reimplemented from __Tree_t.

Definition at line 1308 of file vgtl_graph.h.

9.38.3.3 _Node* _Tree_alloc_base< _Tp, _Ctr, _Iterator , _Node, _Alloc , _IsStatic >::_C_get_node ( ) [inline, protected, inherited]

allocate a new node

Definition at line 1375 of file vgtl_tree.h.

9.38.3.4 void _Tree_alloc_base< _Tp, _Ctr, _Iterator , _Node, _Alloc , _IsStatic >::_C_put_node (_Node * __p) [inline, protected, inherited]

deallocate a node

Definition at line 1378 of file vgtl_tree.h.

9.38.3.5 template<class _Tp, class _Ctr, class _Tl, class _Node, class _Allocator, bool _IsStatic> void _Tree_alloc_base< _Tp, _Ctr, _Tl, _Node, _Allocator, _IsStatic >::_C_put_node (_Node * __p) [inline, protected, inherited]

deallocate a node

Definition at line 1378 of file vgtl_tree.h.

9.38.3.6 void _Tree_alloc_base< _Tp, _Ctr, _Tl, _Node, _Alloc , _IsStatic >::_C_put_node (_Node * __p) [inline, protected, inherited]

deallocate a node

Definition at line 1378 of file vgtl_tree.h.

9.38.3.7 void _Tree_alloc_base< _Tp, _Ctr, _Tl, _Alloc , _Alloc_traits< _Tp, _Alloc >::_S_instanceless , _IsStatic >::_C_put_node (_Alloc * __p) [inline, protected, inherited]

deallocate a node

Definition at line 1378 of file vgtl_tree.h.

9.38.3.8 void _Tree_base< _Tp, _Ctr, _Iterator, _Node, _Alloc >::add_all_children (_Output_Iterator *fi,* _Node * *_parent*) [inline, inherited]

add all children to the parent `_parent`. `fi` is a iterator to the children container of the parent

Definition at line 1539 of file vgtl_tree.h.

9.38.3.9 template<class _Tp, class _Ctr, class _TI, class _Node, class _Alloc> template<class _Output_Iterator > void _Tree_base< _Tp, _Ctr, _TI, _Node, _Alloc >::add_all_children (_Output_Iterator *fi*, _Node * *_parent*) [inherited]

add all children to the parent `_parent`. `fi` is a iterator to the children container of the parent

9.38.3.10 iterator __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::begin ( ) [inline, inherited]

return an iterator to the first node in walk

Definition at line 1964 of file vgtl_tree.h.

9.38.3.11 const_iterator __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::begin ( ) const [inline, inherited]

return a const iterator to the first node in walk

Definition at line 1973 of file vgtl_tree.h.

9.38.3.12 iterative_walker __ITree::begin (walker_type *wt* = `cw_pre_post`, bool *front_to_back* = `true`, bool *depth_first* = `true`) [inline, inherited]

the walker to the first node of the complete walk

Definition at line 2122 of file vgtl_tree.h.

9.38.3.13 const_iterative_walker __ITree::begin (walker_type *wt* = `cw_pre_post`, bool *front_to_back* = `true`, bool *depth_first* = `true`) const [inline, inherited]

the const walker to the first node of the complete walk

Definition at line 2129 of file vgtl_tree.h.

9.38.3.14 void _Tree_base< _Tp, _Ctr, _Iterator , _Node, _Alloc >::clear_children ( ) [inline, inherited]

clear all children of the root node

Definition at line 1466 of file vgtl_tree.h.

9.38.3.15 size_type __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::depth (const walker & *__position*) [inline, inherited]

Reimplemented from __Tree_t.

Definition at line 1526 of file vgtl_graph.h.

9.38.3.16 size_type __ITree::depth (const iterative_walker & *__position*) [inline, inherited]

return the depth of this `__position` in the tree

Definition at line 2177 of file vgtl_tree.h.

9.38.3.17 bool __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::empty ( ) const [inline, inherited]

is the tree empty?

Reimplemented from __Tree_t.

Definition at line 1392 of file vgtl_graph.h.

9.38.3.18 iterator __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::end ( ) [inline, inherited]

return an iterator beyond the last node in walk

Definition at line 1968 of file vgtl_tree.h.

9.38.3.19 const_iterator __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::end ( ) const [inline, inherited]

return a const iterator beyond the last node in walk

Definition at line 1977 of file vgtl_tree.h.

9.38.3.20 iterative_walker __lTree::end (walker_type *wt* = cw_pre_post, bool *front_to_back* = true, bool *depth_first* = true) [inline, inherited]

the walker beyond the last node of the walk

Definition at line 2137 of file vgtl_tree.h.

9.38.3.21 const_iterative_walker __lTree::end (walker_type *wt* = cw_pre_post, bool *front_to_back* = true, bool *depth_first* = true) const [inline, inherited]

the const walker beyond the last node of the walk

Definition at line 2143 of file vgtl_tree.h.

9.38.3.22 void __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::erase (const __walker_base & *__position*) [inline, inherited]

erase the node at position ___position.

Reimplemented from __Tree_t.

Definition at line 1444 of file vgtl_graph.h.

9.38.3.23 bool __Tree_t::erase_child (const __walker_base & *__position*, const children_iterator & *__lt*) [inline, inherited]

erase the (leaf) child ___It of node ___position. This works if and only if the child is a leaf.

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >.

Definition at line 1770 of file vgtl_tree.h.

9.38.3.24 _Node* __Tree_t::erase_subtree (const __walker_base & *__position,* const children_iterator & *__It*) [inline, inherited]

erase the subtree position ___position, whose top node is the child at children_iterator position ___It, and return its top node.

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >.

Definition at line 1790 of file vgtl_tree.h.

9.38.3.25 _Node* __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::erase_tree (const __walker_base & *__position*) [inline, inherited]

erase the subtree starting at position ___position, and return its top node.

Reimplemented from __Tree_t.

Definition at line 1471 of file vgtl_graph.h.

9.38.3.26 allocator_type __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::get_allocator ( ) const [inline, inherited]

construct an allocator object

Reimplemented from __Tree_t.

Definition at line 1259 of file vgtl_graph.h.

9.38.3.27 reference __ITree::getroot ( ) [inline, inherited]

get a reference to the virtual root node

Definition at line 2172 of file vgtl_tree.h.

9.38.3.28 const_reference __ITree::getroot ( ) const [inline, inherited]

get a const reference to the virtual root node

Definition at line 2174 of file vgtl_tree.h.

9.38.3.29 walker __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::ground ( ) [inline, inherited]

return a walker to the virtual root node.

Definition at line 1939 of file vgtl_tree.h.

9.38.3.30 const_walker __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::ground ( ) const [inline, inherited]

return a const walker to the virtual root node.

Definition at line 1943 of file vgtl_tree.h.

9.38.3.31 template<class _Tp , template< class __Ty, class __AllocT > class _SequenceCtr = std::vector, class _PtrAlloc = __VGTL_DEFAULT_ALLOCATOR(void *), class _Alloc = __VGTL_DEFAULT_ALLOCATOR(_Tp)> void ntree< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >::insert (const __walker_base & *__position,* const _Tp & *__x*) [inline]

Insert a node with data __x at position __position.

Definition at line 2336 of file vgtl_tree.h.

9.38.3.32 template<class _Tp , template< class __Ty, class __AllocT > class _SequenceCtr = std::vector, class _PtrAlloc = __VGTL_DEFAULT_ALLOCATOR(void *), class _Alloc = __VGTL_DEFAULT_ALLOCATOR(_Tp)> void ntree< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >::insert (const __walker_base & *__position*) [inline]

Insert a node with default data at position __position.

Definition at line 2364 of file vgtl_tree.h.

9.38.3.33 void __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::insert_child (const __walker_base & *__position,* const _Tp & *__x,* const container_insert_arg & *__It*) [inline, inherited]

add a child below __position with data __x, at the __It position in the __position - node's children container

Reimplemented from __Tree_t.

Definition at line 1409 of file vgtl_graph.h.

9.38.3.34 void __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::insert_child (const __walker_base & *__position,* const container_insert_arg & *__It*) [inline, inherited]

add a child below __position with default data, at the __It position in the __position - node's children container

Reimplemented from __Tree_t.

Definition at line 1415 of file vgtl_graph.h.

9.38.3.35 void __Tree_t::insert_children (const __walker_base & *__position,* size_type *__n,* const _Tp & *__x,* const children_iterator & *__It*) [inline, inherited]

add __n children below __position with data __x, after the __It position in the __position - node's children container

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >.

Definition at line 1682 of file vgtl_tree.h.

9.38.3.36 void __Tree_t::insert_subtree (const __walker_base & *__position,* _Self & *__subtree,* const children_iterator & *__It*) [inline, inherited]

add a complete subtree __subtree below position __position and children iterator position __It.

Definition at line 1702 of file vgtl_tree.h.

9.38.3.37 size_type __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::max_size ( ) const [inline, inherited]

return the maximum possible size of the tree (theor. infinity)

Reimplemented from __Tree_t.

Definition at line 1400 of file vgtl_graph.h.

9.38.3.38 template<class _Tp , template< class __Ty, class __AllocT > class _SequenceCtr = std::vector, class _PtrAlloc = __VGTL_DEFAULT_ALLOCATOR(void *), class _Alloc = __VGTL_DEFAULT_ALLOCATOR(_Tp)> _Self& ntree< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >::operator= (_Node * __x) [inline]

assign a tree from one node -> make this node the root node. This is useful for making trees out of erased subtrees.

Definition at line 2491 of file vgtl_tree.h.

9.38.3.39 template<class _Tp , template< class __Ty, class __AllocT > class _SequenceCtr = std::vector, class _PtrAlloc = __VGTL_DEFAULT_ALLOCATOR(void *), class _Alloc = __VGTL_DEFAULT_ALLOCATOR(_Tp)> bool ntree< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >::pop_child (const __walker_base & __position) [inline]

erase the last (leaf) child of node __position. This works if and only if the child is a leaf.

Definition at line 2433 of file vgtl_tree.h.

9.38.3.40 template<class _Tp , template< class __Ty, class __AllocT > class _SequenceCtr = std::vector, class _PtrAlloc = __VGTL_DEFAULT_ALLOCATOR(void *), class _Alloc = __VGTL_DEFAULT_ALLOCATOR(_Tp)> _Node* ntree< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >::pop_subtree (const __walker_base & __position) [inline]

erase the subtree position __position, whose top node is the last child of the node, and return its top node.

Definition at line 2461 of file vgtl_tree.h.

9.38.3.41 template<class _Tp , template< class __Ty, class __AllocT > class _SequenceCtr = std::vector, class _PtrAlloc = __VGTL_DEFAULT_ALLOCATOR(void *), class _Alloc = __VGTL_DEFAULT_ALLOCATOR(_Tp)> void ntree< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >::push_child (const __walker_base & __position, const _Tp & __x) [inline]

add a child below __position with data __x, at the last position in the __position - node's children container

Definition at line 2369 of file vgtl_tree.h.

9.38.3.42 template<class _Tp , template< class __Ty, class __AllocT > class _SequenceCtr = std::vector, class _PtrAlloc = __VGTL_DEFAULT_ALLOCATOR(void *), class _Alloc = __VGTL_DEFAULT_ALLOCATOR(_Tp)> void ntree< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >::push_child (const __walker_base & __position) [inline]

add a child below __position with default data, at the last position in the __position - node's children container

Definition at line 2374 of file vgtl_tree.h.

9.38.3.43 template<class _Tp , template< class __Ty, class __AllocT > class _SequenceCtr = std::vector, class _PtrAlloc = __VGTL_DEFAULT_ALLOCATOR(void *), class _Alloc = __VGTL_DEFAULT_ALLOCATOR(_Tp)> void ntree< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >::push_children (const __walker_base & *__position,* size_type *__n,* const _Tp & *__x*) [inline]

add __n children below __position with data __x, after the last position in the __position - node's children container

Definition at line 2379 of file vgtl_tree.h.

9.38.3.44 template<class _Tp , template< class __Ty, class __AllocT > class _SequenceCtr = std::vector, class _PtrAlloc = __VGTL_DEFAULT_ALLOCATOR(void *), class _Alloc = __VGTL_DEFAULT_ALLOCATOR(_Tp)> void ntree< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >::push_children (const __walker_base & *__position,* size_type *__n*) [inline]

add __n children below __position with default data, after the last position in the __position - node's children container

Definition at line 2385 of file vgtl_tree.h.

9.38.3.45 template<class _Tp , template< class __Ty, class __AllocT > class _SequenceCtr = std::vector, class _PtrAlloc = __VGTL_DEFAULT_ALLOCATOR(void *), class _Alloc = __VGTL_DEFAULT_ALLOCATOR(_Tp)> void ntree< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >::push_subtree (const __walker_base & *__position,* _Self & *__subtree*) [inline]

add a complete subtree __subtree below position __position and last children iterator position.

Definition at line 2413 of file vgtl_tree.h.

9.38.3.46 reverse_iterator __ITree::rbegin ( ) [inline, inherited]

return a reverse iterator to the first node in walk

Definition at line 2151 of file vgtl_tree.h.

9.38.3.47 const_reverse_iterator __ITree::rbegin ( ) const [inline, inherited]

return a const reverse iterator to the first node in walk

Definition at line 2158 of file vgtl_tree.h.

9.38.3.48 reverse_iterator __ITree::rend ( ) [inline, inherited]

return a reverse iterator beyond the last node in walk

Definition at line 2154 of file vgtl_tree.h.

9.38.3.49 const_reverse_iterator __ITree::rend ( ) const [inline, inherited]

return a const reverse iterator beyond the last node in walk

Definition at line 2161 of file vgtl_tree.h.

9.38.3.50 walker __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::root (children_iterator *__it*) [inline, inherited]

return a walker to a root node.

Definition at line 1947 of file vgtl_tree.h.

9.38.3.51 const_walker __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::root (children_iterator __*it*) const `[inline, inherited]`

return a const walker to a root node.

Definition at line 1952 of file vgtl_tree.h.

9.38.3.52 walker __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::root ( ) `[inline, inherited]`

return a walker to the first non-virtual tree root

Definition at line 1957 of file vgtl_tree.h.

9.38.3.53 const_walker __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::root ( ) const `[inline, inherited]`

return a const walker to the first non-virtual tree root

Definition at line 1960 of file vgtl_tree.h.

9.38.3.54 iterative_walker __ITree::root (walker_type *wt* = `cw_pre_post`, bool *front_to_back* = `true`, bool *depth_first* = `true`) `[inline, inherited]`

return an iterative walker of type `wt` to the ground node

Definition at line 2099 of file vgtl_tree.h.

9.38.3.55 const_iterative_walker __ITree::root (walker_type *wt* = `cw_pre_post`, bool *front_to_back* = `true`, bool *depth_first* = `true`) const `[inline, inherited]`

return a const iterative walker of type `wt` to the ground node

Definition at line 2106 of file vgtl_tree.h.

9.38.3.56 template<class _Tp , template< class __Ty, class __AllocT > class _SequenceCtr = std::vector, class _PtrAlloc = __VGTL_DEFAULT_ALLOCATOR(void *), class _Alloc = __VGTL_DEFAULT_ALLOCATOR(_Tp)> bool ntree< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >::shift_child (const __walker_base & __*position*) `[inline]`

erase the first (leaf) child of node `__position`. This works if and only if the child is a leaf.

Definition at line 2447 of file vgtl_tree.h.

9.38.3.57 template<class _Tp , template< class __Ty, class __AllocT > class _SequenceCtr = std::vector, class _PtrAlloc = __VGTL_DEFAULT_ALLOCATOR(void *), class _Alloc = __VGTL_DEFAULT_ALLOCATOR(_Tp)> _Node* ntree< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >::shift_subtree (const __walker_base & __*position*) `[inline]`

erase the subtree position `__position`, whose top node is the last child of the node, and return its top node.

Definition at line 2476 of file vgtl_tree.h.

9.38.3.58 size_type __ITree::size ( ) const [inline, inherited]

return the size of the tree (# of nodes)

Definition at line 2165 of file vgtl_tree.h.

9.38.3.59 void __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::swap (_Self & __x) [inline, inherited]

Reimplemented from __Tree_t.

Definition at line 1405 of file vgtl_graph.h.

9.38.3.60 iterative_walker __ITree::through ( ) [inline, inherited]

the walker beyond the complete walk

Definition at line 2113 of file vgtl_tree.h.

9.38.3.61 const_iterative_walker __ITree::through ( ) const [inline, inherited]

the const walker beyond the complete walk

Definition at line 2117 of file vgtl_tree.h.

9.38.3.62 template<class _Tp , template< class __Ty, class __AllocT > class _SequenceCtr = std::vector, class _PtrAlloc = __VGTL_DEFAULT_ALLOCATOR(void *), class _Alloc = __VGTL_DEFAULT_ALLOCATOR(_Tp)> void ntree< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >::unshift_child (const __walker_base & *__position,* const _Tp & *__x*) [inline]

add a child below ___position with data ___x, at the first position in the ___position - node's children container

Definition at line 2390 of file vgtl_tree.h.

9.38.3.63 template<class _Tp , template< class __Ty, class __AllocT > class _SequenceCtr = std::vector, class _PtrAlloc = __VGTL_DEFAULT_ALLOCATOR(void *), class _Alloc = __VGTL_DEFAULT_ALLOCATOR(_Tp)> void ntree< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >::unshift_child (const __walker_base & *__position*) [inline]

add a child below ___position with default data, at the first position in the ___position - node's children container

Definition at line 2395 of file vgtl_tree.h.

9.38.3.64 template<class _Tp , template< class __Ty, class __AllocT > class _SequenceCtr = std::vector, class _PtrAlloc = __VGTL_DEFAULT_ALLOCATOR(void *), class _Alloc = __VGTL_DEFAULT_ALLOCATOR(_Tp)> void ntree< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >::unshift_children (const __walker_base & *__position,* size_type *__n,* const _Tp & *__x*) [inline]

add ___n children below ___position with data ___x, after the first position in the ___position - node's children container

Definition at line 2400 of file vgtl_tree.h.

9.38.3.65 template<class _Tp , template< class __Ty, class __AllocT > class _SequenceCtr = std::vector, class _PtrAlloc = __VGTL_DEFAULT_ALLOCATOR(void *), class _Alloc = __VGTL_DEFAULT_ALLOCATOR(_Tp)> void ntree< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >::unshift_children (const __walker_base & *__position*, size_type *__n*) [inline]

add __n children below __position with default data, after the first position in the __position - node's children container

Definition at line 2406 of file vgtl_tree.h.

9.38.3.66 template<class _Tp , template< class __Ty, class __AllocT > class _SequenceCtr = std::vector, class _PtrAlloc = __VGTL_DEFAULT_ALLOCATOR(void *), class _Alloc = __VGTL_DEFAULT_ALLOCATOR(_Tp)> void ntree< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >::unshift_subtree (const __walker_base & *__position*, _Self & *__subtree*) [inline]

add a complete subtree __subtree below position __position and first children iterator position.

Definition at line 2423 of file vgtl_tree.h.

9.38.4 Friends And Related Function Documentation

9.38.4.1 bool operator==__VGTL_NULL_TMPL_ARGS (const __ITree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc > & *__x*, const __ITree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc > & *__y*) [friend, inherited]

comparison operator

9.38.5 Member Data Documentation

9.38.5.1 _Node* _Tree_alloc_base< _Tp, _Ctr, _Iterator , _Node, _Alloc , _IsStatic >::_C_node [protected, inherited]

This is the node

Definition at line 1387 of file vgtl_tree.h.

The documentation for this class was generated from the following files:

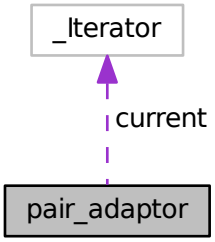
- vgtl_graph.h
- vgtl_tree.h

9.39 pair_adaptor Class Reference

adaptor for an iterator over a pair to an iterator returning the second element

`#include <vgtl_intadapt.h>`

Collaboration diagram for pair_adaptor:



Public Types

- typedef std::iterator_traits < _Iterator > ::iterator_category iterator_category
  standard iterator definitions
- typedef std::iterator_traits < _Iterator >::difference_type difference_type
  standard iterator definitions
- typedef std::iterator_traits < _Iterator >::value_type p_value_type
  standard iterator definitions
- typedef std::iterator_traits < _Iterator >::pointer p_pointer
  standard iterator definitions
- typedef std::iterator_traits < _Iterator >::reference p_reference
  standard iterator definitions
- typedef p_value_type::second_type value_type
  standard iterator definitions
- typedef value_type & reference
  standard iterator definitions
- typedef value_type * pointer
  standard iterator definitions

- typedef p_value_type::first_type key_type
  additional definitions for the key type
- typedef key_type & key_reference
  additional definitions for the key type
- typedef key_type * key_pointer
  additional definitions for the key type

Public Member Functions

- pair_adaptor ()
  standard constructor
- pair_adaptor (iterator_type __x)
  constructor setting the position
- pair_adaptor (const _Self &__x)
  copy constructor
- template<class _Iter >
  pair_adaptor (const pair_adaptor< _Iter > &__x)
  a copy constructor setting the position from another pair adaptor
- iterator_type base () const
  return the base iterator
- reference operator* () const
  dereference operator
- pointer operator-> () const
  pointer operator
- key_reference operator~ () const
  dereference to the key value
- _Self & operator= (const iterator_type &__x)
  assignment operator setting the position from base iterator

- _Self & operator++ ()
  standard increment, decrement operators
- _Self operator++ (int)
  standard increment, decrement operators
- _Self & operator-- ()
  standard increment, decrement operators
- _Self operator-- (int)
  standard increment, decrement operators

- _Self operator+ (difference_type __n) const
  standard random access operators
- _Self & operator+= (difference_type __n)
  standard random access operators
- _Self operator- (difference_type __n) const
  standard random access operators
- _Self & operator-= (difference_type __n)
  standard random access operators
- reference operator[ ] (difference_type __n) const
  standard random access operators

- bool operator== (const iterator_type &__x)
  standard comparison operator
- bool operator!= (const iterator_type &__x)
  standard comparison operator

Protected Attributes

- _Iterator current
  the original iterator

9.39.1 Detailed Description

This adaptor transforms an iterator returning a pair (e.g. a `map` or `multimap` iterator) to an iterator returning only the value part. There is another operator (~), which returns the key value for a given position.

The documentation for this class was generated from the following file:

- vgtl_intadapt.h

9.40 pointer_adaptor Class Reference

adaptor transforming a comparison predicate to pointers

`#include <vgtl_intadapt.h>`

Public Types

- typedef __a1 * first_argument_type
  standard binary predicate definitions
- typedef __a2 * second_argument_type
  standard binary predicate definitions
- typedef _Compare::result_type result_type
  standard binary predicate definitions

Public Member Functions

- result_type operator() (__a1 *arg1, __a2 *arg2) const
  the real adaptor

9.40.1 Detailed Description

This adaptor transforms a binary comparison predicate for two data types `__a1` and `__a2` to a comparison predicate on the pointers to `__a1` and `__a2`, respectively.

The documentation for this class was generated from the following file:

- vgtl_intadapt.h

9.41 postorder_visitor Class Reference

postorder visitor base class

`#include <vgtl_visitor.h>`

Public Member Functions

- postorder_visitor ()
- virtual ~postorder_visitor ()

- virtual void vinit ()
- virtual return_value vvalue () VGTL_PURE_VIRTUAL virtual void vcollect(collect_value __r)

- virtual void init ()
- virtual bool postorder (const _Node &__n)
- virtual void collect (const _Node &__n, collect_value __r)

9.41.1 Detailed Description

This is the base class of all postorder visitors. They can be used in all recursive postorder walks.

9.41.2 Constructor & Destructor Documentation

9.41.2.1 postorder_visitor::postorder_visitor ( ) [inline]

standard constructor

Definition at line 94 of file vgtl_visitor.h.

9.41.2.2 virtual postorder_visitor::~postorder_visitor ( ) [inline, virtual]

standard destructor

Definition at line 96 of file vgtl_visitor.h.

9.41.3 Member Function Documentation

9.41.3.1 virtual void postorder_visitor::collect (const _Node & *__n*, collect_value *__r*) [inline, virtual]

virtual functions for ordinary nodes

Definition at line 109 of file vgtl_visitor.h.

9.41.3.2 virtual void postorder_visitor::init ( ) [inline, virtual]

virtual functions for ordinary nodes

Definition at line 107 of file vgtl_visitor.h.

9.41.3.3 virtual bool postorder_visitor::postorder (const _Node & __n) [inline, virtual]

virtual functions for ordinary nodes

Definition at line 108 of file vgtl_visitor.h.

9.41.3.4 virtual void postorder_visitor::vinit ( ) [inline, virtual]

virtual functions for virtual nodes

Definition at line 100 of file vgtl_visitor.h.

9.41.3.5 virtual return_value postorder_visitor::vvalue ( ) [inline, virtual]

virtual functions for virtual nodes

Definition at line 101 of file vgtl_visitor.h.

The documentation for this class was generated from the following file:

- vgtl_visitor.h

9.42 preorder_visitor Class Reference

preorder visitor base class

`#include <vgtl_visitor.h>`

Public Types

- typedef _Ret return_value

Public Member Functions

- preorder_visitor ()
- virtual ~preorder_visitor ()

- virtual void vinit ()
- virtual return_value vvalue () VGTL_PURE_VIRTUAL virtual void vcollect(collect_value __r)

- virtual bool preorder (const _Node &__n)
- virtual void collect (const _Node &__n, collect_value __r)

9.42.1 Detailed Description

This is the base class of all preorder visitors. They can be used in all recursive preorder walks.

9.42.2 Member Typedef Documentation

9.42.2.1 typedef _Ret preorder_visitor::return_value

the return value type

Definition at line 57 of file vgtl_visitor.h.

9.42.3 Constructor & Destructor Documentation

9.42.3.1 preorder_visitor::preorder_visitor ( ) [inline]

standard constructor

Definition at line 61 of file vgtl_visitor.h.

9.42.3.2 virtual preorder_visitor::~preorder_visitor ( ) [inline, virtual]

standard destructor

Definition at line 63 of file vgtl_visitor.h.

9.42.4 Member Function Documentation

9.42.4.1 virtual void preorder_visitor::collect (const _Node & __n, collect_value __r) [inline, virtual]

virtual functions for ordinary nodes

Definition at line 75 of file vgtl_visitor.h.

9.42.4.2 virtual bool preorder_visitor::preorder (const _Node & __n) [inline, virtual]

virtual functions for ordinary nodes

Definition at line 74 of file vgtl_visitor.h.

9.42.4.3 virtual void preorder_visitor::vinit ( ) [inline, virtual]

virtual functions for virtual nodes

Definition at line 67 of file vgtl_visitor.h.

9.42.4.4 virtual return_value preorder_visitor::vvalue ( ) [inline, virtual]

virtual functions for virtual nodes

Definition at line 68 of file vgtl_visitor.h.

The documentation for this class was generated from the following file:

- vgtl_visitor.h

9.43 prepost_visitor Class Reference

pre+postorder visitor base class

`#include <vgtl_visitor.h>`

Public Member Functions

- prepost_visitor ()
- virtual ~prepost_visitor ()

- virtual void vinit ()
- virtual return_value vvalue () VGTL_PURE_VIRTUAL virtual void vcollect(collect_value __r)

- virtual bool preorder (const _Node &__n)
- virtual bool postorder (const _Node &__n)
- virtual void collect (const _Node &__n, collect_value __r)

9.43.1 Detailed Description

This is the base class of all pre+postorder visitors. They can be used in all recursive walks.

9.43.2 Constructor & Destructor Documentation

9.43.2.1 prepost_visitor::prepost_visitor ( ) [inline]

standard constructor

Definition at line 128 of file vgtl_visitor.h.

9.43.2.2 virtual prepost_visitor::~prepost_visitor ( ) [inline, virtual]

standard destructor

Definition at line 130 of file vgtl_visitor.h.

9.43.3 Member Function Documentation

9.43.3.1 virtual void prepost_visitor::collect (const _Node & *__n*, collect_value *__r*) [inline, virtual]

virtual functions for ordinary nodes

Definition at line 143 of file vgtl_visitor.h.

9.43.3.2 virtual bool prepost_visitor::postorder (const _Node & *__n*) [inline, virtual]

virtual functions for ordinary nodes

Definition at line 142 of file vgtl_visitor.h.

9.43.3.3 virtual bool prepost_visitor::preorder (const _Node & *__n*) [inline, virtual]

virtual functions for ordinary nodes

Definition at line 141 of file vgtl_visitor.h.

9.43.3.4 virtual void prepost_visitor::vinit ( ) [inline, virtual]

virtual functions for virtual nodes

Definition at line 134 of file vgtl_visitor.h.

9.43.3.5 virtual return_value prepost_visitor::vvalue ( ) [inline, virtual]

virtual functions for virtual nodes

Definition at line 135 of file vgtl_visitor.h.

The documentation for this class was generated from the following file:

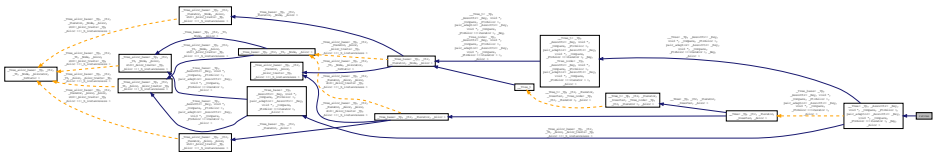
- vgtl_visitor.h

9.44 ratree Class Reference

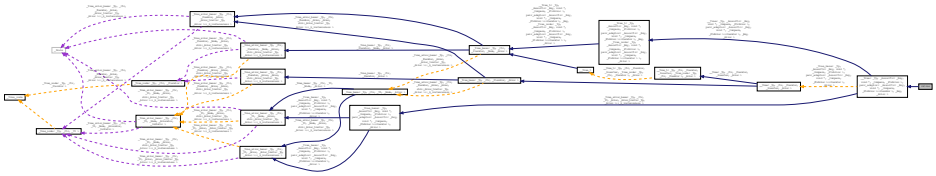
n-ary forest with labelled edges

`#include <vgtl_tree.h>`

Inheritance diagram for ratree:



Collaboration diagram for ratree:



Public Types

- typedef _Tp value_type
- typedef _Node node_type
- typedef _Node node_type
- typedef value_type * pointer
- typedef const value_type * const_pointer
- typedef value_type & reference
- typedef const value_type & const_reference
- typedef size_t size_type
- typedef ptrdiff_t difference_type
- typedef _Tree_iterator< _Tp, _Tp &, _Tp *, container_type, container_iterator > iterator
- typedef _Tree_iterator< _Tp, _Tp &, _Tp *, container_type, children_iterator, node_type > iterator

- typedef _Tree_iterator< _Tp, const _Tp &, const _Tp *, container_type, container_iterator > const_iterator
- typedef _Tree_iterator< _Tp, const _Tp &, const _Tp *, container_type, children_iterator, node_type > const_iterator
- typedef reverse_iterator < const_iterator > const_reverse_iterator
- typedef std::reverse_iterator < const_iterator > const_reverse_iterator
- typedef reverse_iterator < iterator > reverse_iterator
- typedef std::reverse_iterator < iterator > reverse_iterator
- typedef _Tree_walker< _Tp, _Tp &, _Tp *, container_type, container_iterator > walker
- typedef _Tree_walker< _Tp, const _Tp &, const _Tp *, container_type, container_iterator > const_walker
- typedef _Iterator children_iterator
- typedef _TI children_iterator
- typedef __one_iterator< void * > parents_iterator
- typedef __one_iterator< void * > parents_iterator

Public Member Functions

- _Self & operator= (_Node *__x)
- void insert (const __walker_base &__position, const _Tp &__x, const _Key &__k)
- void insert (const __walker_base &__position, const _Key &__k)
- allocator_type get_allocator () const
- walker root (children_iterator __it)
- const_walker root (children_iterator __it) const
- walker root ()
- const_walker root () const
- iterator begin ()
- const_iterator begin () const
- iterator end ()
- const_iterator end () const
- reverse_iterator rbegin ()
- const_reverse_iterator rbegin () const
- reverse_iterator rend ()
- const_reverse_iterator rend () const
- bool empty () const
- size_type max_size () const
- reference getroot ()
- const_reference getroot () const
- void **swap** (_Self &__x)
- void insert_child (const __walker_base &__position, const _Tp &__x, const container_insert_arg &__It)
- void insert_child (const __walker_base &__position, const container_insert_arg &__It)
- void insert_children (const __walker_base &__position, size_type __n, const _Tp &__x, const children_iterator &__It)
- void insert_subtree (const __walker_base &__position, _Self &__subtree, const children_iterator &__It)
- void erase (const __walker_base &__position)
- _Node * erase_tree (const __walker_base &__position)
- bool erase_child (const __walker_base &__position, const children_iterator &__It)
- _Node * erase_subtree (const __walker_base &__position, const children_iterator &__It)

- size_type **depth** (const walker &__position)
- size_type depth (const recursive_walker &__position)
- walker ground ()
- const_walker ground () const
- void clear_children ()
- void add_all_children (_Output_Iterator fi, _Node *_parent)
- template<class _Output_Iterator >
  void add_all_children (_Output_Iterator fi, _Node *_parent)

Protected Member Functions

- _Node * _C_create_node (const _Tp &__x)
- _Node * _C_create_node ()
- _Node * _C_get_node ()
- void _C_put_node (_Node *__p)
- void _C_put_node (_Node *__p)
- void _C_put_node (_Node *__p)
- void _C_put_node (_Alloc *__p)

Protected Attributes

- _Node * _C_node

Friends

- bool operator==__VGTL_NULL_TMPL_ARGS (const __Tree &__x, const __Tree &__y)

9.44.1 Detailed Description

This class constructs an n-ary forest without data hooks and labelled edges. By default, the children are collected in a STL multimap, but the container can be replaced by any other associative map container.

9.44.2 Member Typedef Documentation

9.44.2.1 template<class _Tp, class _Ctr, class _Tl, class _Node, class _Alloc> typedef _Tl _Tree_base< _Tp, _Ctr, _Tl, _Node, _Alloc >::children_iterator [inherited]

iterator for accessing the children

Reimplemented in __Tree_t.

Definition at line 1445 of file vgtl_tree.h.

9.44.2.2 typedef _Iterator __Tree_t::children_iterator [inherited]

iterator for accessing the children

Reimplemented from _Tree_base< _Tp, _Ctr, _Iterator, _Node, _Alloc >.

Definition at line 1563 of file vgtl_tree.h.

9.44.2.3 typedef _Tree_iterator<_Tp,const _Tp&,const _Tp*,container_type,container_iterator> __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::const_iterator [inherited]

the const iterator

Reimplemented from __Tree_t.

Definition at line 1263 of file vgtl_graph.h.

9.44.2.4 typedef _Tree_iterator<_Tp,const _Tp&,const _Tp*,container_type,children_iterator,node_type> __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::const_iterator [inherited]

the const iterator

Reimplemented from __Tree_t.

Definition at line 1901 of file vgtl_tree.h.

9.44.2.5 typedef const value_type* __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::const_pointer [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1251 of file vgtl_graph.h.

9.44.2.6 typedef const value_type& __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::const_reference [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1253 of file vgtl_graph.h.

9.44.2.7 typedef reverse_iterator<const_iterator> __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::const_reverse_iterator [inherited]

the const reverse iterator

Reimplemented from __Tree_t.

Definition at line 1266 of file vgtl_graph.h.

9.44.2.8 typedef std::reverse_iterator<const_iterator> __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::const_reverse_iterator [inherited]

the const reverse iterator

Reimplemented from __Tree_t.

Definition at line 1905 of file vgtl_tree.h.

9.44.2.9 typedef _Tree_walker<_Tp,const _Tp&,const _Tp*,container_type,container_iterator> __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::const_walker [inherited]

the (recursive) const walker

Reimplemented from __Tree_t.

Definition at line 1278 of file vgtl_graph.h.

9.44.2.10 typedef ptrdiff_t __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::difference_type [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1255 of file vgtl_graph.h.

9.44.2.11 typedef _Tree_iterator<_Tp,_Tp&,_Tp*,container_type,container_iterator> __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::iterator [inherited]

the iterator

Reimplemented from __Tree_t.

Definition at line 1262 of file vgtl_graph.h.

9.44.2.12 typedef _Tree_iterator<_Tp,_Tp&,_Tp*,container_type,children_iterator,node_type> __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::iterator [inherited]

the iterator

Reimplemented from __Tree_t.

Definition at line 1899 of file vgtl_tree.h.

9.44.2.13 typedef _Node __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::node_type [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1249 of file vgtl_graph.h.

9.44.2.14 typedef _Node __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::node_type [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1895 of file vgtl_tree.h.

9.44.2.15 template<class _Tp, class _Ctr, class _Tl, class _Node, class _Alloc> typedef __one_iterator<void *> _Tree_base< _Tp, _Ctr, _Tl, _Node, _Alloc >::parents_iterator [inherited]

iterator for accessing the parents

Reimplemented in __Tree_t.

Definition at line 1447 of file vgtl_tree.h.

9.44.2.16 typedef __one_iterator<void *> __Tree_t::parents_iterator [inherited]

iterator for accessing the parents

Reimplemented from _Tree_base< _Tp, _Ctr, _Iterator, _Node, _Alloc >.

Definition at line 1564 of file vgtl_tree.h.

9.44.2.17 typedef value_type* __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::pointer [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1250 of file vgtl_graph.h.

9.44.2.18 typedef value_type& __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::reference [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1252 of file vgtl_graph.h.

9.44.2.19 typedef reverse_iterator<iterator> __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::reverse_iterator [inherited]

the reverse iterator

Reimplemented from __Tree_t.

Definition at line 1267 of file vgtl_graph.h.

9.44.2.20 typedef std::reverse_iterator<iterator> __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::reverse_iterator [inherited]

the reverse iterator

Reimplemented from __Tree_t.

Definition at line 1907 of file vgtl_tree.h.

9.44.2.21 typedef size_t __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::size_type [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1254 of file vgtl_graph.h.

9.44.2.22 typedef _Tp __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::value_type [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1248 of file vgtl_graph.h.

9.44.2.23 typedef _Tree_walker<_Tp,_Tp&,_Tp*,container_type,container_iterator> __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::walker [inherited]

the (recursive) walker

Reimplemented from __Tree_t.

Definition at line 1277 of file vgtl_graph.h.

9.44.3 Member Function Documentation

9.44.3.1 _Node* __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::_C_create_node (const _Tp & __x) [inline, protected, inherited]

construct a new tree node containing data __x

Reimplemented from __Tree_t.

Definition at line 1295 of file vgtl_graph.h.

9.44.3.2 _Node* __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::_C_create_node ( ) [inline, protected, inherited]

construct a new tree node containing default data

Reimplemented from __Tree_t.

Definition at line 1308 of file vgtl_graph.h.

9.44.3.3 _Node* _Tree_alloc_base< _Tp, _Ctr, _Iterator , _Node, _Alloc , _IsStatic >::_C_get_node ( ) [inline, protected, inherited]

allocate a new node

Definition at line 1375 of file vgtl_tree.h.

9.44.3.4 void _Tree_alloc_base< _Tp, _Ctr, _Tl, _Node, _Alloc , _IsStatic >::_C_put_node (_Node * __p) [inline, protected, inherited]

deallocate a node

Definition at line 1378 of file vgtl_tree.h.

9.44.3.5 void _Tree_alloc_base< _Tp, _Ctr, _Iterator , _Node, _Alloc , _IsStatic >::_C_put_node (_Node * __p) [inline, protected, inherited]

deallocate a node

Definition at line 1378 of file vgtl_tree.h.

9.44.3.6 template<class _Tp, class _Ctr, class _Tl, class _Node, class _Allocator, bool _IsStatic> void _Tree_alloc_base< _Tp, _Ctr, _Tl, _Node, _Allocator, _IsStatic >::_C_put_node (_Node * __p) [inline, protected, inherited]

deallocate a node

Definition at line 1378 of file vgtl_tree.h.

9.44.3.7 void _Tree_alloc_base< _Tp, _Ctr, _Tl, _Alloc , _Alloc_traits< _Tp, _Alloc >::_S_instanceless , _IsStatic >::_C_put_node (_Alloc * __p) [inline, protected, inherited]

deallocate a node

Definition at line 1378 of file vgtl_tree.h.

9.44.3.8 void _Tree_base< _Tp, _Ctr, _Iterator, _Node, _Alloc >::add_all_children (_Output_Iterator *fi,* _Node * *_parent*) [inline, inherited]

add all children to the parent `_parent`. `fi` is a iterator to the children container of the parent

Definition at line 1539 of file vgtl_tree.h.

9.44.3.9 template<class _Tp, class _Ctr, class _Tl, class _Node, class _Alloc> template<class _Output_Iterator > void _Tree_base< _Tp, _Ctr, _Tl, _Node, _Alloc >::add_all_children (_Output_Iterator *fi,* _Node * *_parent*) [inherited]

add all children to the parent `_parent`. `fi` is a iterator to the children container of the parent

9.44.3.10 iterator __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::begin ( ) [inline, inherited]

return an iterator to the first node in walk

Definition at line 1964 of file vgtl_tree.h.

9.44.3.11 const_iterator __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::begin ( ) const [inline, inherited]

return a const iterator to the first node in walk

Definition at line 1973 of file vgtl_tree.h.

9.44.3.12 void _Tree_base< _Tp, _Ctr, _Iterator , _Node, _Alloc >::clear_children ( ) [inline, inherited]

clear all children of the root node

Definition at line 1466 of file vgtl_tree.h.

9.44.3.13 size_type __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::depth (const walker & __position) [inline, inherited]

Reimplemented from __Tree_t.

Definition at line 1526 of file vgtl_graph.h.

9.44.3.14 size_type __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::depth (const recursive_walker & __position) [inline, inherited]

return the depth of node __position in the tree

Reimplemented from __Tree_t.

Definition at line 1529 of file vgtl_graph.h.

9.44.3.15 bool __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::empty ( ) const [inline, inherited]

is the tree empty?

Reimplemented from __Tree_t.

Definition at line 1392 of file vgtl_graph.h.

9.44.3.16 iterator __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::end ( ) [inline, inherited]

return an iterator beyond the last node in walk

Definition at line 1968 of file vgtl_tree.h.

9.44.3.17 const_iterator __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::end ( ) const [inline, inherited]

return a const iterator beyond the last node in walk

Definition at line 1977 of file vgtl_tree.h.

9.44.3.18 void __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::erase (const __walker_base & __position) [inline, inherited]

erase the node at position __position.

Reimplemented from __Tree_t.

Definition at line 1444 of file vgtl_graph.h.

9.44.3.19 bool __Tree_t::erase_child (const __walker_base & __position, const children_iterator & __it) [inline, inherited]

erase the (leaf) child __It of node __position. This works if and only if the child is a leaf.

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >.

Definition at line 1770 of file vgtl_tree.h.

9.44.3.20 _Node* __Tree_t::erase_subtree (const __walker_base & __position, const children_iterator & __it) [inline, inherited]

erase the subtree position __position, whose top node is the child at children_iterator position __It, and return its top node.

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >.

Definition at line 1790 of file vgtl_tree.h.

9.44.3.21 _Node* __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::erase_tree (const __walker_base & __position) [inline, inherited]

erase the subtree starting at position __position, and return its top node.

Reimplemented from __Tree_t.

Definition at line 1471 of file vgtl_graph.h.

9.44.3.22 allocator_type __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::get_allocator ( ) const [inline, inherited]

construct an allocator object

Reimplemented from __Tree_t.

Definition at line 1259 of file vgtl_graph.h.

9.44.3.23 reference __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::getroot ( ) [inline, inherited]

get a reference to the virtual root node

Definition at line 1996 of file vgtl_tree.h.

9.44.3.24 const_reference __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::getroot ( ) const [inline, inherited]

get a const reference to the virtual root node

Definition at line 1998 of file vgtl_tree.h.

9.44.3.25 walker __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::ground ( ) [inline, inherited]

return a walker to the virtual root node.

Definition at line 1939 of file vgtl_tree.h.

9.44.3.26 const_walker __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::ground ( ) const [inline, inherited]

return a const walker to the virtual root node.

Definition at line 1943 of file vgtl_tree.h.

9.44.3.27 void ratree::insert (const __walker_base & *__position,* const _Tp & *__x,* const _Key & *__k*) [inline]

Insert a node with data __x and key _k at position __position.

Definition at line 2823 of file vgtl_tree.h.

9.44.3.28 void ratree::insert (const __walker_base & *__position,* const _Key & *__k*) [inline]

Insert a node with default data and key __k at position __position.

Definition at line 2849 of file vgtl_tree.h.

9.44.3.29 void __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::insert_child (const __walker_base & *__position,* const _Tp & *__x,* const container_insert_arg & *__It*) [inline, inherited]

add a child below __position with data __x, at the __It position in the __position - node's children container

Reimplemented from __Tree_t.

Definition at line 1409 of file vgtl_graph.h.

9.44.3.30 void __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::insert_child (const __walker_base & *__position,* const container_insert_arg & *__It*) [inline, inherited]

add a child below __position with default data, at the __It position in the __position - node's children container

Reimplemented from __Tree_t.

Definition at line 1415 of file vgtl_graph.h.

9.44.3.31 void __Tree_t::insert_children (const __walker_base & *__position,* size_type *__n,* const _Tp & *__x,* const children_iterator & *__It*) [inline, inherited]

add __n children below __position with data __x, after the __It position in the __position - node's children container

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >.

Definition at line 1682 of file vgtl_tree.h.

9.44.3.32 void __Tree_t::insert_subtree (const __walker_base & *__position,* _Self & *__subtree,* const children_iterator & *__It*) [inline, inherited]

add a complete subtree __subtree below position __position and children iterator position __It.

Definition at line 1702 of file vgtl_tree.h.

9.44.3.33 size_type __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::max_size ( ) const [inline, inherited]

return the maximum possible size of the tree (theor. infinity)

Reimplemented from __Tree_t.

Definition at line 1400 of file vgtl_graph.h.

9.44.3.34 _Self& ratree::operator= (_Node * *__x*) [inline]

assign a tree from one node -> make this node the root node. This is useful for making trees out of erased subtrees.

Reimplemented from __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >.

Definition at line 2814 of file vgtl_tree.h.

9.44.3.35 reverse_iterator __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::rbegin ( ) [inline, inherited]

return a reverse iterator to the first node in walk

Definition at line 1982 of file vgtl_tree.h.

9.44.3.36 const_reverse_iterator __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::rbegin ( ) const [inline, inherited]

return a const reverse iterator to the first node in walk

Definition at line 1989 of file vgtl_tree.h.

9.44.3.37 reverse_iterator __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::rend ( ) [inline, inherited]

return a reverse iterator beyond the last node in walk

Definition at line 1985 of file vgtl_tree.h.

9.44.3.38 const_reverse_iterator __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::rend ( ) const [inline, inherited]

return a const reverse iterator beyond the last node in walk

Definition at line 1992 of file vgtl_tree.h.

9.44.3.39 walker __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::root (children_iterator __it) [inline, inherited]

return a walker to a root node.

Definition at line 1947 of file vgtl_tree.h.

9.44.3.40 const_walker __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::root (children_iterator __it) const [inline, inherited]

return a const walker to a root node.

Definition at line 1952 of file vgtl_tree.h.

9.44.3.41 walker __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::root ( ) [inline, inherited]

return a walker to the first non-virtual tree root

Definition at line 1957 of file vgtl_tree.h.

9.44.3.42 const_walker __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::root ( ) const [inline, inherited]

return a const walker to the first non-virtual tree root

Definition at line 1960 of file vgtl_tree.h.

9.44.3.43 void __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc > , pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator > , _Key , _Alloc >::swap (_Self & __x) [inline, inherited]

Reimplemented from __Tree_t.

Definition at line 1405 of file vgtl_graph.h.

9.44.4 Friends And Related Function Documentation

9.44.4.1 bool operator==__VGTL_NULL_TMPL_ARGS (const __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc > & __x, const __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc > & __y) [friend, inherited]

comparison operator

9.44.5 Member Data Documentation

9.44.5.1 _Node* _Tree_alloc_base< _Tp, _Ctr, _Iterator , _Node, _Alloc , _IsStatic >::_C_node [protected, inherited]

This is the node

Definition at line 1387 of file vgtl_tree.h.

The documentation for this class was generated from the following file:

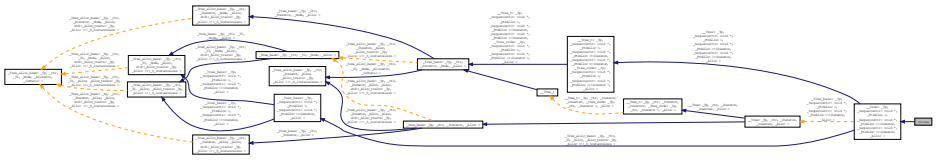
- vgtl_tree.h

9.45 rntree Class Reference

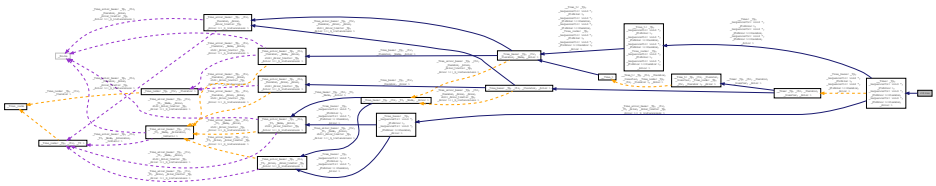
n-ary forest

`#include <vgtl_tree.h>`

Inheritance diagram for rntree:



Collaboration diagram for rntree:



Public Types

- typedef _Tp value_type
- typedef _Node node_type
- typedef _Node node_type
- typedef value_type * pointer
- typedef const value_type * const_pointer
- typedef value_type & reference
- typedef const value_type & const_reference
- typedef size_t size_type
- typedef ptrdiff_t difference_type
- typedef _Tree_iterator< _Tp, _Tp &, _Tp *, container_type, container_iterator > iterator
- typedef _Tree_iterator< _Tp, _Tp &, _Tp *, container_type, children_iterator, node_type > iterator
- typedef _Tree_iterator< _Tp, const _Tp &, const _Tp *, container_type, container_iterator > const_iterator
- typedef _Tree_iterator< _Tp, const _Tp &, const _Tp *, container_type, children_iterator, node_type > const_iterator
- typedef reverse_iterator < const_iterator > const_reverse_iterator
- typedef std::reverse_iterator < const_iterator > const_reverse_iterator
- typedef reverse_iterator < iterator > reverse_iterator
- typedef std::reverse_iterator < iterator > reverse_iterator

- typedef _Tree_walker< _Tp, _Tp &, _Tp *, container_type, container_iterator > walker
- typedef _Tree_walker< _Tp, const _Tp &, const _Tp *, container_type, container_iterator > const_walker
- typedef _Iterator children_iterator
- typedef _TI children_iterator
- typedef __one_iterator< void * > parents_iterator
- typedef __one_iterator< void * > parents_iterator

Public Member Functions

- void insert (const __walker_base &__position, const _Tp &__x)
- void insert (const __walker_base &__position)
- void push_child (const __walker_base &__position, const _Tp &__x)
- void push_child (const __walker_base &__position)
- void push_children (const __walker_base &__position, size_type __n, const _Tp &__x)
- void push_children (const __walker_base &__position, size_type __n)
- void unshift_child (const __walker_base &__position, const _Tp &__x)
- void unshift_child (const __walker_base &__position)
- void unshift_children (const __walker_base &__position, size_type __n, const _Tp &__x)
- void unshift_children (const __walker_base &__position, size_type __n)
- void push_subtree (const __walker_base &__position, _Self &__subtree)
- void unshift_subtree (const __walker_base &__position, _Self &__subtree)
- bool pop_child (const __walker_base &__position)
- bool shift_child (const __walker_base &__position)
- _Node * pop_subtree (const __walker_base &__position)
- _Node * shift_subtree (const __walker_base &__position)
- _Self & operator= (_Node *__x)
- allocator_type get_allocator () const
- walker root (children_iterator __it)
- const_walker root (children_iterator __it) const
- walker root ()
- const_walker root () const
- iterator begin ()
- const_iterator begin () const
- iterator end ()
- const_iterator end () const
- reverse_iterator rbegin ()
- const_reverse_iterator rbegin () const
- reverse_iterator rend ()
- const_reverse_iterator rend () const
- bool empty () const
- size_type max_size () const
- reference getroot ()
- const_reference getroot () const
- void **swap** (_Self &__x)
- void insert_child (const __walker_base &__position, const _Tp &__x, const container_insert_arg &__It)
- void insert_child (const __walker_base &__position, const container_insert_arg &__It)
- void insert_children (const __walker_base &__position, size_type __n, const _Tp &__x, const children-_iterator &__It)

- void insert_subtree (const __walker_base &__position, _Self &__subtree, const children_iterator &__It)
- void erase (const __walker_base &__position)
- _Node * erase_tree (const __walker_base &__position)
- bool erase_child (const __walker_base &__position, const children_iterator &__It)
- _Node * erase_subtree (const __walker_base &__position, const children_iterator &__It)
- size_type **depth** (const walker &__position)
- size_type depth (const recursive_walker &__position)
- walker ground ()
- const_walker ground () const
- void clear_children ()
- void add_all_children (_Output_Iterator fi, _Node *_parent)
- template<class _Output_Iterator >
  void add_all_children (_Output_Iterator fi, _Node *_parent)

Protected Member Functions

- _Node * _C_create_node (const _Tp &__x)
- _Node * _C_create_node ()
- _Node * _C_get_node ()
- void _C_put_node (_Node *__p)
- void _C_put_node (_Node *__p)
- void _C_put_node (_Node *__p)
- void _C_put_node (_Alloc *__p)

Protected Attributes

- _Node * _C_node

Friends

- bool operator==__VGTL_NULL_TMPL_ARGS (const __Tree &__x, const __Tree &__y)

9.45.1 Detailed Description

This class constructs an n-ary forest without data hooks. By default, the children are collected in a STL vector, but the container can be replaced by any other sequential container.

9.45.2 Member Typedef Documentation

9.45.2.1 template<class _Tp, class _Ctr, class _Tl, class _Node, class _Alloc> typedef _Tl _Tree_base< _Tp, _Ctr, _Tl, _Node, _Alloc >::children_iterator [inherited]

iterator for accessing the children

Reimplemented in __Tree_t.

Definition at line 1445 of file vgtl_tree.h.

9.45.2.2 typedef _Iterator __Tree_t::children_iterator [inherited]

iterator for accessing the children

Reimplemented from _Tree_base< _Tp, _Ctr, _Iterator, _Node, _Alloc >.

Definition at line 1563 of file vgtl_tree.h.

9.45.2.3 typedef _Tree_iterator<_Tp,const _Tp&,const _Tp*,container_type,container_iterator> __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::const_iterator [inherited]

the const iterator

Reimplemented from __Tree_t.

Definition at line 1263 of file vgtl_graph.h.

9.45.2.4 typedef _Tree_iterator<_Tp,const _Tp&,const _Tp*,container_type,children_iterator,node_type> __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::const_iterator [inherited]

the const iterator

Reimplemented from __Tree_t.

Definition at line 1901 of file vgtl_tree.h.

9.45.2.5 typedef const value_type* __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::const_pointer [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1251 of file vgtl_graph.h.

9.45.2.6 typedef const value_type& __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::const_reference [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1253 of file vgtl_graph.h.

9.45.2.7 typedef reverse_iterator<const_iterator> __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::const_reverse_iterator [inherited]

the const reverse iterator

Reimplemented from __Tree_t.

Definition at line 1266 of file vgtl_graph.h.

9.45.2.8 typedef std::reverse_iterator<const_iterator> __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::const_reverse_iterator [inherited]

the const reverse iterator

Reimplemented from __Tree_t.

Definition at line 1905 of file vgtl_tree.h.

9.45.2.9 typedef _Tree_walker<_Tp,const _Tp&,const _Tp*,container_type,container_iterator> __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::const_walker [inherited]

the (recursive) const walker

Reimplemented from __Tree_t.

Definition at line 1278 of file vgtl_graph.h.

9.45.2.10 typedef ptrdiff_t __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::difference_type [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1255 of file vgtl_graph.h.

9.45.2.11 typedef _Tree_iterator<_Tp,_Tp&,_Tp*,container_type,container_iterator> __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::iterator [inherited]

the iterator

Reimplemented from __Tree_t.

Definition at line 1262 of file vgtl_graph.h.

9.45.2.12 typedef _Tree_iterator<_Tp,_Tp&,_Tp*,container_type,children_iterator,node_type> __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::iterator [inherited]

the iterator

Reimplemented from __Tree_t.

Definition at line 1899 of file vgtl_tree.h.

9.45.2.13 typedef _Node __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::node_type [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1249 of file vgtl_graph.h.

9.45.2.14 typedef _Node __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::node_type [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1895 of file vgtl_tree.h.

9.45.2.15 template<class _Tp, class _Ctr, class _Tl, class _Node, class _Alloc> typedef __one_iterator<void *> _Tree_base< _Tp, _Ctr, _Tl, _Node, _Alloc >::parents_iterator [inherited]

iterator for accessing the parents

Reimplemented in __Tree_t.

Definition at line 1447 of file vgtl_tree.h.

9.45.2.16 typedef __one_iterator<void *> __Tree_t::parents_iterator [inherited]

iterator for accessing the parents

Reimplemented from _Tree_base< _Tp, _Ctr, _Iterator, _Node, _Alloc >.

Definition at line 1564 of file vgtl_tree.h.

9.45.2.17 typedef value_type* __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::pointer [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1250 of file vgtl_graph.h.

9.45.2.18 typedef value_type& __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::reference [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1252 of file vgtl_graph.h.

9.45.2.19 typedef reverse_iterator<iterator> __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::reverse_iterator [inherited]

the reverse iterator

Reimplemented from __Tree_t.

Definition at line 1267 of file vgtl_graph.h.

9.45.2.20 typedef std::reverse_iterator<iterator> __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::reverse_iterator [inherited]

the reverse iterator

Reimplemented from __Tree_t.

Definition at line 1907 of file vgtl_tree.h.

9.45.2.21 typedef size_t __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::size_type [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1254 of file vgtl_graph.h.

9.45.2.22 typedef _Tp __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::value_type [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1248 of file vgtl_graph.h.

9.45.2.23 typedef _Tree_walker<_Tp,_Tp&,_Tp*,container_type,container_iterator> __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::walker [inherited]

the (recursive) walker

Reimplemented from __Tree_t.

Definition at line 1277 of file vgtl_graph.h.

9.45.3 Member Function Documentation

9.45.3.1 _Node* __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::_C_create_node (const _Tp & __x) [inline, protected, inherited]

construct a new tree node containing data __x

Reimplemented from __Tree_t.

Definition at line 1295 of file vgtl_graph.h.

9.45.3.2 _Node* __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::_C_create_node ( ) [inline, protected, inherited]

construct a new tree node containing default data

Reimplemented from __Tree_t.

Definition at line 1308 of file vgtl_graph.h.

9.45.3.3 _Node* _Tree_alloc_base< _Tp, _Ctr, _Iterator , _Node, _Alloc , _IsStatic >::_C_get_node ( ) [inline, protected, inherited]

allocate a new node

Definition at line 1375 of file vgtl_tree.h.

9.45.3.4 void _Tree_alloc_base< _Tp, _Ctr, _Iterator , _Node, _Alloc , _IsStatic >::_C_put_node (_Node * __p) [inline, protected, inherited]

deallocate a node

Definition at line 1378 of file vgtl_tree.h.

9.45.3.5 template<class _Tp, class _Ctr, class _Tl, class _Node, class _Allocator, bool _IsStatic> void _Tree_alloc_base< _Tp, _Ctr, _Tl, _Node, _Allocator, _IsStatic >::_C_put_node (_Node * __p) [inline, protected, inherited]

deallocate a node

Definition at line 1378 of file vgtl_tree.h.

9.45.3.6 void _Tree_alloc_base< _Tp, _Ctr, _Tl, _Node, _Alloc , _IsStatic >::_C_put_node (_Node * __p) [inline, protected, inherited]

deallocate a node

Definition at line 1378 of file vgtl_tree.h.

9.45.3.7 void _Tree_alloc_base< _Tp, _Ctr, _Tl, _Alloc , _Alloc_traits< _Tp, _Alloc >::_S_instanceless , _IsStatic >::_C_put_node (_Alloc * __p) [inline, protected, inherited]

deallocate a node

Definition at line 1378 of file vgtl_tree.h.

9.45.3.8 template<class _Tp, class _Ctr, class _Tl, class _Node, class _Alloc> template<class _Output_Iterator > void _Tree_base< _Tp, _Ctr, _Tl, _Node, _Alloc >::add_all_children (_Output_Iterator *fi,* _Node * *_parent*) [inherited]

add all children to the parent _parent. fi is a iterator to the children container of the parent

9.45.3.9 void _Tree_base< _Tp, _Ctr, _Iterator, _Node, _Alloc >::add_all_children (_Output_Iterator *fi,* _Node * *_parent*) [inline, inherited]

add all children to the parent _parent. fi is a iterator to the children container of the parent

Definition at line 1539 of file vgtl_tree.h.

9.45.3.10 iterator __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::begin ( ) [inline, inherited]

return an iterator to the first node in walk

Definition at line 1964 of file vgtl_tree.h.

9.45.3.11 const_iterator __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::begin ( ) const [inline, inherited]

return a const iterator to the first node in walk

Definition at line 1973 of file vgtl_tree.h.

9.45.3.12 void _Tree_base< _Tp, _Ctr, _Iterator , _Node, _Alloc >::clear_children ( ) [inline, inherited]

clear all children of the root node

Definition at line 1466 of file vgtl_tree.h.

9.45.3.13 size_type __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::depth (const walker & __position) [inline, inherited]

Reimplemented from __Tree_t.

Definition at line 1526 of file vgtl_graph.h.

9.45.3.14 size_type __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::depth (const recursive_walker & __position) [inline, inherited]

return the depth of node `__position` in the tree

Reimplemented from __Tree_t.

Definition at line 1529 of file vgtl_graph.h.

9.45.3.15 bool __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::empty ( ) const [inline, inherited]

is the tree empty?

Reimplemented from __Tree_t.

Definition at line 1392 of file vgtl_graph.h.

9.45.3.16 iterator __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::end ( ) [inline, inherited]

return an iterator beyond the last node in walk

Definition at line 1968 of file vgtl_tree.h.

9.45.3.17 const_iterator __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::end ( ) const [inline, inherited]

return a const iterator beyond the last node in walk

Definition at line 1977 of file vgtl_tree.h.

9.45.3.18 void __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::erase (const __walker_base & *__position*) [inline, inherited]

erase the node at position __position.

Reimplemented from __Tree_t.

Definition at line 1444 of file vgtl_graph.h.

9.45.3.19 bool __Tree_t::erase_child (const __walker_base & *__position,* const children_iterator & *__It*) [inline, inherited]

erase the (leaf) child __It of node __position. This works if and only if the child is a leaf.

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >.

Definition at line 1770 of file vgtl_tree.h.

9.45.3.20 _Node* __Tree_t::erase_subtree (const __walker_base & *__position,* const children_iterator & *__It*) [inline, inherited]

erase the subtree position __position, whose top node is the child at children_iterator position __It, and return its top node.

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >.

Definition at line 1790 of file vgtl_tree.h.

9.45.3.21 _Node* __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::erase_tree (const __walker_base & *__position*) [inline, inherited]

erase the subtree starting at position __position, and return its top node.

Reimplemented from __Tree_t.

Definition at line 1471 of file vgtl_graph.h.

9.45.3.22 allocator_type __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::get_allocator ( ) const [inline, inherited]

construct an allocator object

Reimplemented from __Tree_t.

Definition at line 1259 of file vgtl_graph.h.

9.45.3.23 reference __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::getroot ( ) [inline, inherited]

get a reference to the virtual root node

Definition at line 1996 of file vgtl_tree.h.

9.45.3.24 const_reference __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::getroot ( ) const [inline, inherited]

get a const reference to the virtual root node

Definition at line 1998 of file vgtl_tree.h.

9.45.3.25 walker __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::ground ( ) [inline, inherited]

return a walker to the virtual root node.

Definition at line 1939 of file vgtl_tree.h.

9.45.3.26 const_walker __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::ground ( ) const [inline, inherited]

return a const walker to the virtual root node.

Definition at line 1943 of file vgtl_tree.h.

9.45.3.27 void rntree::insert (const __walker_base & *__position,* const _Tp & *__x*) [inline]

Insert a node with data __x at position __position.

Definition at line 2523 of file vgtl_tree.h.

9.45.3.28 void rntree::insert (const __walker_base & *__position*) [inline]

Insert a node with default data at position __position.

Definition at line 2551 of file vgtl_tree.h.

9.45.3.29 void __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::insert_child (const __walker_base & *__position,* const _Tp & *__x,* const container_insert_arg & *__It*) [inline, inherited]

add a child below __position with data __x, at the __It position in the __position - node's children container

Reimplemented from __Tree_t.

Definition at line 1409 of file vgtl_graph.h.

9.45.3.30 void __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::insert_child (const __walker_base & *__position,* const container_insert_arg & *__It*) [inline, inherited]

add a child below __position with default data, at the __It position in the __position - node's children container

Reimplemented from __Tree_t.

Definition at line 1415 of file vgtl_graph.h.

9.45.3.31 void __Tree_t::insert_children (const __walker_base & __position, size_type __n, const _Tp & __x, const children_iterator & __It) [inline, inherited]

add __n children below __position with data __x, after the __It position in the __position - node's children container

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >.

Definition at line 1682 of file vgtl_tree.h.

9.45.3.32 void __Tree_t::insert_subtree (const __walker_base & __position, _Self & __subtree, const children_iterator & __It) [inline, inherited]

add a complete subtree __subtree below position __position and children iterator position __It.

Definition at line 1702 of file vgtl_tree.h.

9.45.3.33 size_type __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::max_size ( ) const [inline, inherited]

return the maximum possible size of the tree (theor. infinity)

Reimplemented from __Tree_t.

Definition at line 1400 of file vgtl_graph.h.

9.45.3.34 _Self& rntree::operator= (_Node * __x) [inline]

assign a tree from one node -> make this node the root node. This is useful for making trees out of erased subtrees.

Reimplemented from __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >.

Definition at line 2678 of file vgtl_tree.h.

9.45.3.35 bool rntree::pop_child (const __walker_base & __position) [inline]

erase the last (leaf) child of node __position. This works if and only if the child is a leaf.

Definition at line 2620 of file vgtl_tree.h.

9.45.3.36 _Node* rntree::pop_subtree (const __walker_base & __position) [inline]

erase the subtree position __position, whose top node is the last child of the node, and return its top node.

Definition at line 2648 of file vgtl_tree.h.

9.45.3.37 void rntree::push_child (const __walker_base & __position, const _Tp & __x) [inline]

add a child below __position with data __x, at the last position in the __position - node's children container

Definition at line 2556 of file vgtl_tree.h.

9.45.3.38 void rntree::push_child (const __walker_base & __position) [inline]

add a child below __position with default data, at the last position in the __position - node's children container

Definition at line 2561 of file vgtl_tree.h.

9.45.3.39 void rntree::push_children (const __walker_base & __position, size_type __n, const _Tp & __x) [inline]

add __n children below __position with data __x, after the last position in the __position - node's children container

Definition at line 2566 of file vgtl_tree.h.

9.45.3.40 void rntree::push_children (const __walker_base & __position, size_type __n) [inline]

add __n children below __position with default data, after the last position in the __position - node's children container

Definition at line 2572 of file vgtl_tree.h.

9.45.3.41 void rntree::push_subtree (const __walker_base & __position, _Self & __subtree) [inline]

add a complete subtree __subtree below position __position and last children iterator position.

Definition at line 2600 of file vgtl_tree.h.

9.45.3.42 reverse_iterator __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::rbegin ( ) [inline, inherited]

return a reverse iterator to the first node in walk

Definition at line 1982 of file vgtl_tree.h.

9.45.3.43 const_reverse_iterator __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::rbegin ( ) const [inline, inherited]

return a const reverse iterator to the first node in walk

Definition at line 1989 of file vgtl_tree.h.

9.45.3.44 reverse_iterator __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::rend ( ) [inline, inherited]

return a reverse iterator beyond the last node in walk

Definition at line 1985 of file vgtl_tree.h.

9.45.3.45 const_reverse_iterator __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::rend ( ) const [inline, inherited]

return a const reverse iterator beyond the last node in walk

Definition at line 1992 of file vgtl_tree.h.

9.45.3.46 walker __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::root (children_iterator __it) [inline, inherited]

return a walker to a root node.

Definition at line 1947 of file vgtl_tree.h.

9.45.3.47 const_walker __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::root (children_iterator __it) const [inline, inherited]

return a const walker to a root node.

Definition at line 1952 of file vgtl_tree.h.

9.45.3.48 walker __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::root ( ) [inline, inherited]

return a walker to the first non-virtual tree root

Definition at line 1957 of file vgtl_tree.h.

9.45.3.49 const_walker __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::root ( ) const [inline, inherited]

return a const walker to the first non-virtual tree root

Definition at line 1960 of file vgtl_tree.h.

9.45.3.50 bool rntree::shift_child (const __walker_base & __position) [inline]

erase the first (leaf) child of node __position. This works if and only if the child is a leaf.

Definition at line 2634 of file vgtl_tree.h.

9.45.3.51 _Node* rntree::shift_subtree (const __walker_base & __position) [inline]

erase the subtree position __position, whose top node is the last child of the node, and return its top node.

Definition at line 2663 of file vgtl_tree.h.

9.45.3.52 void __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc > , _SequenceCtr< void *, _PtrAlloc >::iterator , _SequenceCtr< void *, _PtrAlloc >::iterator , _Alloc >::swap (_Self & __x) [inline, inherited]

Reimplemented from __Tree_t.

Definition at line 1405 of file vgtl_graph.h.

9.45.3.53 void rntree::unshift_child (const __walker_base & __position, const _Tp & __x) [inline]

add a child below __position with data __x, at the first position in the __position - node's children container

Definition at line 2577 of file vgtl_tree.h.

9.45.3.54 void rntree::unshift_child (const __walker_base & __position) [inline]

add a child below __position with default data, at the first position in the __position - node's children container

Definition at line 2582 of file vgtl_tree.h.

9.45.3.55 void rntree::unshift_children (const __walker_base & __position, size_type __n, const _Tp & __x) [inline]

add __n children below __position with data __x, after the first position in the __position - node's children container

Definition at line 2587 of file vgtl_tree.h.

9.45.3.56 void rntree::unshift_children (const __walker_base & __position, size_type __n) [inline]

add __n children below __position with default data, after the first position in the __position - node's children container

Definition at line 2593 of file vgtl_tree.h.

9.45.3.57 void rntree::unshift_subtree (const __walker_base & __position, _Self & __subtree) [inline]

add a complete subtree __subtree below position __position and first children iterator position.

Definition at line 2610 of file vgtl_tree.h.

9.45.4 Friends And Related Function Documentation

9.45.4.1 bool operator==__VGTL_NULL_TMPL_ARGS (const __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc > & __x, const __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc > & __y) [friend, inherited]

comparison operator

9.45.5 Member Data Documentation

9.45.5.1 _Node* _Tree_alloc_base< _Tp, _Ctr, _Iterator , _Node, _Alloc , _IsStatic >::_C_node [protected, inherited]

This is the node

Definition at line 1387 of file vgtl_tree.h.

The documentation for this class was generated from the following file:

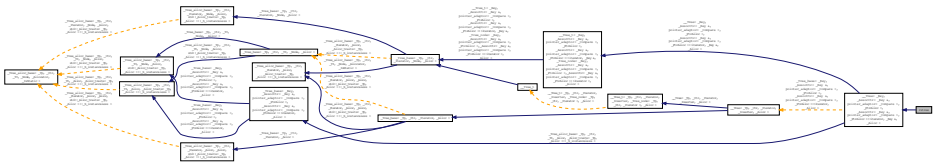
- vgtl_tree.h

9.46 rstree Class Reference

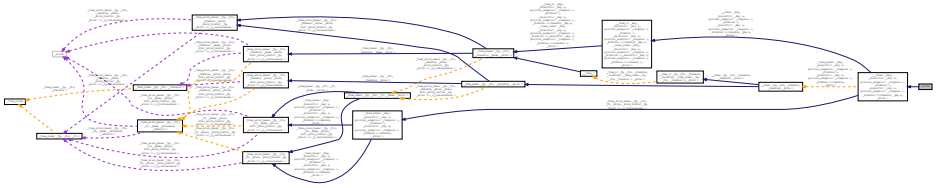
n-ary forest with unsorted edges

`#include <vgtl_tree.h>`

Inheritance diagram for rstree:



Collaboration diagram for rstree:



Public Types

- typedef _Key value_type
- typedef _Node node_type
- typedef _Node node_type
- typedef value_type * pointer
- typedef const value_type * const_pointer
- typedef value_type & reference
- typedef const value_type & const_reference
- typedef size_t size_type
- typedef ptrdiff_t difference_type
- typedef _Tree_iterator< _Key, _Key &, _Key *, container_type, container_iterator > iterator
- typedef _Tree_iterator< _Key, _Key &, _Key *, container_type, children_iterator, node_type > iterator
- typedef _Tree_iterator< _Key, const _Key &, const _Key *, container_type, container_iterator > const_iterator
- typedef _Tree_iterator< _Key, const _Key &, const _Key *, container_type, children_iterator, node_type > const_iterator
- typedef reverse_iterator < const_iterator > const_reverse_iterator
- typedef std::reverse_iterator < const_iterator > const_reverse_iterator
- typedef reverse_iterator < iterator > reverse_iterator
- typedef std::reverse_iterator < iterator > reverse_iterator
- typedef _Tree_walker< _Key, _Key &, _Key *, container_type, container_iterator > walker
- typedef _Tree_walker< _Key, const _Key &, const _Key *, container_type, container_iterator > const_walker
- typedef _Iterator children_iterator
- typedef _TI children_iterator
- typedef __one_iterator< void * > parents_iterator
- typedef __one_iterator< void * > parents_iterator

Public Member Functions

- _Self & operator= (_Node *__x)
- allocator_type get_allocator () const
- walker root (children_iterator __it)
- const_walker root (children_iterator __it) const
- walker root ()
- const_walker root () const
- iterator begin ()
- const_iterator begin () const
- iterator end ()
- const_iterator end () const
- reverse_iterator rbegin ()
- const_reverse_iterator rbegin () const
- reverse_iterator rend ()
- const_reverse_iterator rend () const
- bool empty () const
- size_type max_size () const
- reference getroot ()
- const_reference getroot () const
- void **swap** (_Self &__x)
- void insert_child (const __walker_base &__position, const container_insert_arg &__It)
- void insert_child (const __walker_base &__position, const _Tp &__x, const container_insert_arg &__It)
- void insert_children (const __walker_base &__position, size_type __n, const _Tp &__x, const children-_iterator &__It)
- void insert_subtree (const __walker_base &__position, _Self &__subtree, const children_iterator &__It)
- void erase (const __walker_base &__position)
- _Node * erase_tree (const __walker_base &__position)
- bool erase_child (const __walker_base &__position, const children_iterator &__It)
- _Node * erase_subtree (const __walker_base &__position, const children_iterator &__It)
- size_type **depth** (const walker &__position)
- size_type depth (const recursive_walker &__position)
- walker ground ()
- const_walker ground () const
- void clear_children ()
- void add_all_children (_Output_Iterator fi, _Node *_parent)
- template<class _Output_Iterator >
  void add_all_children (_Output_Iterator fi, _Node *_parent)

Protected Member Functions

- _Node * _C_create_node ()
- _Node * _C_create_node (const _Tp &__x)
- _Node * _C_get_node ()
- void _C_put_node (_Node *__p)
- void _C_put_node (_Node *__p)
- void _C_put_node (_Node *__p)
- void _C_put_node (_Alloc *__p)

Protected Attributes

- _Node * _C_node

Friends

- bool operator==__VGTL_NULL_TMPL_ARGS (const __Tree &__x, const __Tree &__y)

9.46.1 Detailed Description

This class constructs an n-ary forest without data hooks and unsorted edges. By default, the children are collected in a STL multiset, but the container can be replaced by any other associative set container.

9.46.2 Member Typedef Documentation

9.46.2.1 template<class _Tp, class _Ctr, class _Tl, class _Node, class _Alloc> typedef _Tl _Tree_base< _Tp, _Ctr, _Tl, _Node, _Alloc >::children_iterator [inherited]

iterator for accessing the children

Reimplemented in __Tree_t.

Definition at line 1445 of file vgtl_tree.h.

9.46.2.2 typedef _Iterator __Tree_t::children_iterator [inherited]

iterator for accessing the children

Reimplemented from _Tree_base< _Tp, _Ctr, _Iterator, _Node, _Alloc >.

Definition at line 1563 of file vgtl_tree.h.

9.46.2.3 typedef _Tree_iterator<_Key ,const _Key &,const _Key *,container_type,container_iterator> __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::const_iterator [inherited]

the const iterator

Reimplemented from __Tree_t.

Definition at line 1263 of file vgtl_graph.h.

9.46.2.4 typedef _Tree_iterator<_Key ,const _Key &,const _Key *,container_type,children_- iterator,node_type> __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::const_iterator [inherited]

the const iterator

Reimplemented from __Tree_t.

Definition at line 1901 of file vgtl_tree.h.

9.46.2.5 typedef const value_type* __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::const_pointer [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1251 of file vgtl_graph.h.

9.46.2.6 typedef const value_type& __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::const_reference [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1253 of file vgtl_graph.h.

9.46.2.7 typedef reverse_iterator<const_iterator> __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::const_reverse_iterator [inherited]

the const reverse iterator

Reimplemented from __Tree_t.

Definition at line 1266 of file vgtl_graph.h.

9.46.2.8 typedef std::reverse_iterator<const_iterator> __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::const_reverse_iterator [inherited]

the const reverse iterator

Reimplemented from __Tree_t.

Definition at line 1905 of file vgtl_tree.h.

9.46.2.9 typedef _Tree_walker<_Key ,const _Key &,const _Key *,container_type,container_iterator> __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::const_walker [inherited]

the (recursive) const walker

Reimplemented from __Tree_t.

Definition at line 1278 of file vgtl_graph.h.

9.46.2.10 typedef ptrdiff_t __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::difference_type [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1255 of file vgtl_graph.h.

9.46.2.11 typedef _Tree_iterator<_Key ,_Key &,_Key *,container_type,container_iterator> __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::iterator [inherited]

the iterator

Reimplemented from __Tree_t.

Definition at line 1262 of file vgtl_graph.h.

9.46.2.12 typedef _Tree_iterator<_Key ,_Key &,_Key *,container_type,children_iterator,node_type> __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::iterator [inherited]

the iterator

Reimplemented from __Tree_t.

Definition at line 1899 of file vgtl_tree.h.

9.46.2.13 typedef _Node __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::node_type [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1249 of file vgtl_graph.h.

9.46.2.14 typedef _Node __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::node_type [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1895 of file vgtl_tree.h.

9.46.2.15 template<class _Tp, class _Ctr, class _Tl, class _Node, class _Alloc> typedef __one_iterator<void *> _Tree_base< _Tp, _Ctr, _Tl, _Node, _Alloc >::parents_iterator [inherited]

iterator for accessing the parents

Reimplemented in __Tree_t.

Definition at line 1447 of file vgtl_tree.h.

9.46.2.16 typedef __one_iterator<void *> __Tree_t::parents_iterator [inherited]

iterator for accessing the parents

Reimplemented from _Tree_base< _Tp, _Ctr, _Iterator, _Node, _Alloc >.

Definition at line 1564 of file vgtl_tree.h.

9.46.2.17 typedef value_type* __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::pointer [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1250 of file vgtl_graph.h.

9.46.2.18 typedef value_type& __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::reference [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1252 of file vgtl_graph.h.

9.46.2.19 typedef reverse_iterator<iterator> __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::reverse_iterator [inherited]

the reverse iterator

Reimplemented from __Tree_t.

Definition at line 1267 of file vgtl_graph.h.

9.46.2.20 typedef std::reverse_iterator<iterator> __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::reverse_iterator [inherited]

the reverse iterator

Reimplemented from __Tree_t.

Definition at line 1907 of file vgtl_tree.h.

9.46.2.21 typedef size_t __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::size_type [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1254 of file vgtl_graph.h.

9.46.2.22 typedef _Key __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::value_type [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1248 of file vgtl_graph.h.

9.46.2.23 typedef _Tree_walker<_Key ,_Key &,_Key *,container_type,container_iterator> __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::walker [inherited]

the (recursive) walker

Reimplemented from __Tree_t.

Definition at line 1277 of file vgtl_graph.h.

9.46.3 Member Function Documentation

9.46.3.1 _Node* __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::_C_create_node ( ) [inline, protected, inherited]

construct a new tree node containing default data

Reimplemented from __Tree_t.

Definition at line 1308 of file vgtl_graph.h.

9.46.3.2 _Node* __Tree_t::_C_create_node (const _Tp & __x) [inline, protected, inherited]

construct a new tree node containing data ___x

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >, and __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >.

Definition at line 1629 of file vgtl_tree.h.

9.46.3.3 _Node* _Tree_alloc_base< _Tp, _Ctr, _Iterator , _Node, _Alloc , _IsStatic >::_C_get_node ( ) [inline, protected, inherited]

allocate a new node

Definition at line 1375 of file vgtl_tree.h.

9.46.3.4 void _Tree_alloc_base< _Tp, _Ctr, _TI, _Node, _Alloc , _IsStatic >::_C_put_node (_Node * __p) [inline, protected, inherited]

deallocate a node

Definition at line 1378 of file vgtl_tree.h.

9.46.3.5 void _Tree_alloc_base< _Tp, _Ctr, _Iterator , _Node, _Alloc , _IsStatic >::_C_put_node (_Node * __p) [inline, protected, inherited]

deallocate a node

Definition at line 1378 of file vgtl_tree.h.

9.46.3.6 template<class _Tp, class _Ctr, class _Tl, class _Node, class _Allocator, bool _IsStatic> void _Tree_alloc_base< _Tp, _Ctr, _Tl, _Node, _Allocator, _IsStatic >::_C_put_node (_Node * __p) [inline, protected, inherited]

deallocate a node

Definition at line 1378 of file vgtl_tree.h.

9.46.3.7 void _Tree_alloc_base< _Tp, _Ctr, _Tl, _Alloc , _Alloc_traits< _Tp, _Alloc >::_S_instanceless , _IsStatic >::_C_put_node (_Alloc * __p) [inline, protected, inherited]

deallocate a node

Definition at line 1378 of file vgtl_tree.h.

9.46.3.8 void _Tree_base< _Tp, _Ctr, _Iterator, _Node, _Alloc >::add_all_children (_Output_Iterator *fi,* _Node * *_parent*) [inline, inherited]

add all children to the parent _parent. fi is a iterator to the children container of the parent

Definition at line 1539 of file vgtl_tree.h.

9.46.3.9 template<class _Tp, class _Ctr, class _Tl, class _Node, class _Alloc> template<class _Output_Iterator > void _Tree_base< _Tp, _Ctr, _Tl, _Node, _Alloc >::add_all_children (_Output_Iterator *fi,* _Node * *_parent*) [inherited]

add all children to the parent _parent. fi is a iterator to the children container of the parent

9.46.3.10 iterator __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::begin ( ) [inline, inherited]

return an iterator to the first node in walk

Definition at line 1964 of file vgtl_tree.h.

9.46.3.11 const_iterator __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::begin ( ) const [inline, inherited]

return a const iterator to the first node in walk

Definition at line 1973 of file vgtl_tree.h.

9.46.3.12 void _Tree_base< _Tp, _Ctr, _Iterator , _Node, _Alloc >::clear_children ( ) [inline, inherited]

clear all children of the root node

Definition at line 1466 of file vgtl_tree.h.

9.46.3.13 size_type __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::depth (const walker & *__position*) [inline, inherited]

Reimplemented from __Tree_t.

Definition at line 1526 of file vgtl_graph.h.

9.46.3.14 size_type __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::depth (const recursive_walker & __position) [inline, inherited]

return the depth of node __position in the tree

Reimplemented from __Tree_t.

Definition at line 1529 of file vgtl_graph.h.

9.46.3.15 bool __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::empty ( ) const [inline, inherited]

is the tree empty?

Reimplemented from __Tree_t.

Definition at line 1392 of file vgtl_graph.h.

9.46.3.16 iterator __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::end ( ) [inline, inherited]

return an iterator beyond the last node in walk

Definition at line 1968 of file vgtl_tree.h.

9.46.3.17 const_iterator __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::end ( ) const [inline, inherited]

return a const iterator beyond the last node in walk

Definition at line 1977 of file vgtl_tree.h.

9.46.3.18 void __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::erase (const __walker_base & __position) [inline, inherited]

erase the node at position __position.

Reimplemented from __Tree_t.

Definition at line 1444 of file vgtl_graph.h.

9.46.3.19 bool __Tree_t::erase_child (const __walker_base & __position, const children_iterator & __It) [inline, inherited]

erase the (leaf) child __It of node __position. This works if and only if the child is a leaf.

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >.

Definition at line 1770 of file vgtl_tree.h.

9.46.3.20 _Node* __Tree_t::erase_subtree (const __walker_base & __position, const children_iterator & __It) [inline, inherited]

erase the subtree position __position, whose top node is the child at children_iterator position __It, and return its top node.

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >.

Definition at line 1790 of file vgtl_tree.h.

9.46.3.21 _Node* __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::erase_tree (const __walker_base & __position) [inline, inherited]

erase the subtree starting at position __position, and return its top node.

Reimplemented from __Tree_t.

Definition at line 1471 of file vgtl_graph.h.

9.46.3.22 allocator_type __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::get_allocator ( ) const [inline, inherited]

construct an allocator object

Reimplemented from __Tree_t.

Definition at line 1259 of file vgtl_graph.h.

9.46.3.23 reference __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::getroot ( ) [inline, inherited]

get a reference to the virtual root node

Definition at line 1996 of file vgtl_tree.h.

9.46.3.24 const_reference __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::getroot ( ) const [inline, inherited]

get a const reference to the virtual root node

Definition at line 1998 of file vgtl_tree.h.

9.46.3.25 walker __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::ground ( ) [inline, inherited]

return a walker to the virtual root node.

Definition at line 1939 of file vgtl_tree.h.

9.46.3.26 const_walker __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::ground ( ) const [inline, inherited]

return a const walker to the virtual root node.

Definition at line 1943 of file vgtl_tree.h.

9.46.3.27 void __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::insert_child (const __walker_base & __position, const container_insert_arg & __It) [inline, inherited]

add a child below __position with default data, at the __It position in the __position - node's children container

Reimplemented from __Tree_t.

Definition at line 1415 of file vgtl_graph.h.

9.46.3.28 void __Tree_t::insert_child (const __walker_base & __position, const _Tp & __x, const container_insert_arg & __It) [inline, inherited]

add a child below __position with data __x, at the __It position in the __position - node's children container

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >, and __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >.

Definition at line 1668 of file vgtl_tree.h.

9.46.3.29 void __Tree_t::insert_children (const __walker_base & __position, size_type __n, const _Tp & __x, const children_iterator & __It) [inline, inherited]

add __n children below __position with data __x, after the __It position in the __position - node's children container

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >.

Definition at line 1682 of file vgtl_tree.h.

9.46.3.30 void __Tree_t::insert_subtree (const __walker_base & __position, _Self & __subtree, const children_iterator & __It) [inline, inherited]

add a complete subtree __subtree below position __position and children iterator position __It.

Definition at line 1702 of file vgtl_tree.h.

9.46.3.31 size_type __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::max_size ( ) const [inline, inherited]

return the maximum possible size of the tree (theor. infinity)

Reimplemented from __Tree_t.

Definition at line 1400 of file vgtl_graph.h.

9.46.3.32 _Self& rstree::operator= (_Node * __x) [inline]

assign a tree from one node -> make this node the root node. This is useful for making trees out of erased subtrees.

Reimplemented from __Tree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc >.

Definition at line 2881 of file vgtl_tree.h.

9.46.3.33 reverse_iterator __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::rbegin ( ) [inline, inherited]

return a reverse iterator to the first node in walk

Definition at line 1982 of file vgtl_tree.h.

9.46.3.34 const_reverse_iterator __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::rbegin ( ) const [inline, inherited]

return a const reverse iterator to the first node in walk

Definition at line 1989 of file vgtl_tree.h.

9.46.3.35 reverse_iterator __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::rend ( ) [inline, inherited]

return a reverse iterator beyond the last node in walk

Definition at line 1985 of file vgtl_tree.h.

9.46.3.36 const_reverse_iterator __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::rend ( ) const [inline, inherited]

return a const reverse iterator beyond the last node in walk

Definition at line 1992 of file vgtl_tree.h.

9.46.3.37 walker __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::root (children_iterator __it) [inline, inherited]

return a walker to a root node.

Definition at line 1947 of file vgtl_tree.h.

9.46.3.38 const_walker __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::root (children_iterator __it) const [inline, inherited]

return a const walker to a root node.

Definition at line 1952 of file vgtl_tree.h.

9.46.3.39 walker __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::root ( ) [inline, inherited]

return a walker to the first non-virtual tree root

Definition at line 1957 of file vgtl_tree.h.

9.46.3.40 const_walker __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::root ( ) const [inline, inherited]

return a const walker to the first non-virtual tree root

Definition at line 1960 of file vgtl_tree.h.

9.46.3.41 void __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::swap (_Self & __x) [inline, inherited]

Reimplemented from __Tree_t.

Definition at line 1405 of file vgtl_graph.h.

9.46.4 Friends And Related Function Documentation

9.46.4.1 bool operator==__VGTL_NULL_TMPL_ARGS (const __Tree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc > & __x, const __Tree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc > & __y) [friend, inherited]

comparison operator

9.46.5 Member Data Documentation

9.46.5.1 _Node* _Tree_alloc_base< _Tp, _Ctr, _Iterator , _Node, _Alloc , _IsStatic >::_C_node [protected, inherited]

This is the node

Definition at line 1387 of file vgtl_tree.h.

The documentation for this class was generated from the following file:

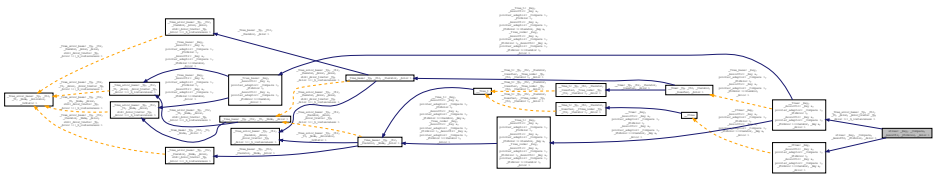
- vgtl_tree.h

9.47 stree< _Key, _Compare, _AssocCtr, _PtrAlloc, _Alloc > Class Template Reference

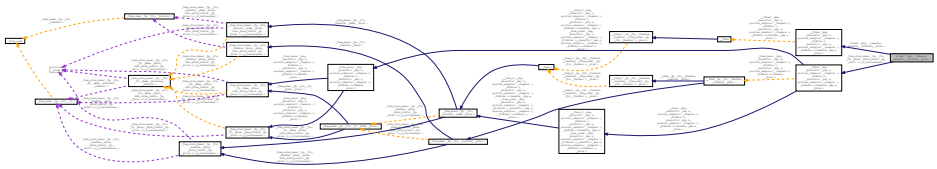
n-ary forest with unsorted edges

`#include <vgtl_tree.h>`

Inheritance diagram for stree< _Key, _Compare, _AssocCtr, _PtrAlloc, _Alloc >:



Collaboration diagram for stree< _Key, _Compare, _AssocCtr, _PtrAlloc, _Alloc >:



Public Types

- typedef _Tree_iterator< _Tp, _Tp &, _Tp *, container_type, children_iterator, node_type > iterator
- typedef _Tree_iterator< _Tp, const _Tp &, const _Tp *, container_type, children_iterator, node_type > const_iterator
- typedef _Tree_walker< _Tp, _Tp &, _Tp *, container_type, children_iterator, _Node > iterative_walker
- typedef _Tree_walker< _Tp, const _Tp &, const _Tp *, container_type, children_iterator, _Node > const_iterative_walker
- typedef std::reverse_iterator < const_iterator > const_reverse_iterator
- typedef std::reverse_iterator < iterator > reverse_iterator
- typedef _Key value_type
- typedef _Node node_type
- typedef value_type * pointer
- typedef const value_type * const_pointer
- typedef value_type & reference
- typedef const value_type & const_reference
- typedef size_t size_type
- typedef ptrdiff_t difference_type
- typedef _Tree_iterator< _Key, _Key &, _Key *, container_type, container_iterator > iterator
- typedef _Tree_iterator< _Key, const _Key &, const _Key *, container_type, container_iterator > const_iterator
- typedef reverse_iterator < const_iterator > const_reverse_iterator
- typedef reverse_iterator < iterator > reverse_iterator
- typedef _Tree_walker< _Key, _Key &, _Key *, container_type, container_iterator > walker
- typedef _Tree_walker< _Key, const _Key &, const _Key *, container_type, container_iterator > const_walker
- typedef _Iterator children_iterator
- typedef _TI children_iterator
- typedef __one_iterator< void * > parents_iterator
- typedef __one_iterator< void * > parents_iterator

Public Member Functions

- _Self & operator= (_Node *__x)
- iterative_walker root (walker_type wt=cw_pre_post, bool front_to_back=true, bool depth_first=true)
- const_iterative_walker root (walker_type wt=cw_pre_post, bool front_to_back=true, bool depth_first=true) const
- iterative_walker through ()
- const_iterative_walker through () const
- iterative_walker begin (walker_type wt=cw_pre_post, bool front_to_back=true, bool depth_first=true)

- const_iterative_walker begin (walker_type wt=cw_pre_post, bool front_to_back=true, bool depth_-first=true) const
- iterative_walker end (walker_type wt=cw_pre_post, bool front_to_back=true, bool depth_first=true)
- const_iterative_walker end (walker_type wt=cw_pre_post, bool front_to_back=true, bool depth_-first=true) const
- reverse_iterator rbegin ()
- const_reverse_iterator rbegin () const
- reverse_iterator rend ()
- const_reverse_iterator rend () const
- size_type size () const
- reference getroot ()
- const_reference getroot () const
- size_type depth (const iterative_walker &__position)
- allocator_type get_allocator () const
- walker root (children_iterator __it)
- const_walker root (children_iterator __it) const
- walker root ()
- const_walker root () const
- iterator begin ()
- const_iterator begin () const
- iterator end ()
- const_iterator end () const
- bool empty () const
- size_type max_size () const
- void **swap** (_Self &__x)
- void insert_child (const __walker_base &__position, const container_insert_arg &__It)
- void insert_child (const __walker_base &__position, const _Tp &__x, const container_insert_arg &__It)
- void insert_children (const __walker_base &__position, size_type __n, const _Tp &__x, const children-_iterator &__It)
- void insert_subtree (const __walker_base &__position, _Self &__subtree, const children_iterator &__It)
- void erase (const __walker_base &__position)
- _Node * erase_tree (const __walker_base &__position)
- bool erase_child (const __walker_base &__position, const children_iterator &__It)
- _Node * erase_subtree (const __walker_base &__position, const children_iterator &__It)
- size_type **depth** (const walker &__position)
- walker ground ()
- const_walker ground () const
- void clear_children ()
- void add_all_children (_Output_Iterator fi, _Node *_parent)
- template<class _Output_Iterator >
  void add_all_children (_Output_Iterator fi, _Node *_parent)

Protected Member Functions

- _Node * _C_create_node ()
- _Node * _C_create_node (const _Tp &__x)
- _Node * _C_get_node ()
- void _C_put_node (_Node *__p)
- void _C_put_node (_Node *__p)
- void _C_put_node (_Node *__p)
- void _C_put_node (_Alloc *__p)

Protected Attributes

- _Node * _C_node

Friends

- bool operator==__VGTL_NULL_TMPL_ARGS (const __ITree &__x, const __ITree &__y)

9.47.1 Detailed Description

template<class _Key, class _Compare = less<_Key>, template< class __Key, class __Compare, class __Alloc-T > class _AssocCtr = std::multiset, class _PtrAlloc = __VGTL_DEFAULT_ALLOCATOR(void *), class _Alloc = __VGTL_DEFAULT_ALLOCATOR(_Key&)>class stree< _Key, _Compare, _AssocCtr, _PtrAlloc, _Alloc >

This class constructs an n-ary forest with data hooks and unsorted edges. By default, the children are collected in a STL multiset, but the container can be replaced by any other associative set container.

Definition at line 2766 of file vgtl_tree.h.

9.47.2 Member Typedef Documentation

9.47.2.1 template<class _Tp, class _Ctr, class _Tl, class _Node, class _Alloc> typedef _Tl _Tree_base< _Tp, _Ctr, _Tl, _Node, _Alloc >::children_iterator [inherited]

iterator for accessing the children

Reimplemented in __Tree_t.

Definition at line 1445 of file vgtl_tree.h.

9.47.2.2 typedef _Iterator __Tree_t::children_iterator [inherited]

iterator for accessing the children

Reimplemented from _Tree_base< _Tp, _Ctr, _Iterator, _Node, _Alloc >.

Definition at line 1563 of file vgtl_tree.h.

9.47.2.3 typedef _Tree_walker<_Tp,const _Tp&,const _Tp*,container_type,children_iterator,_Node> __ITree::const_iterative_walker [inherited]

the const iterative walker

Definition at line 2065 of file vgtl_tree.h.

9.47.2.4 typedef _Tree_iterator<_Key ,const _Key &,const _Key *,container_type,container_iterator> __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::const_iterator [inherited]

the const iterator

Reimplemented from __Tree_t.

Definition at line 1263 of file vgtl_graph.h.

9.47.2.5 typedef _Tree_iterator<_Tp,const _Tp&,const _Tp*,container_type,children_iterator,node_type> __ITree::const_iterator [inherited]

the const iterator

Definition at line 2060 of file vgtl_tree.h.

9.47.2.6 typedef const value_type* __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::const_pointer [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1251 of file vgtl_graph.h.

9.47.2.7 typedef const value_type& __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::const_reference [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1253 of file vgtl_graph.h.

9.47.2.8 typedef reverse_iterator<const_iterator> __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::const_reverse_iterator [inherited]

the const reverse iterator

Reimplemented from __Tree_t.

Definition at line 1266 of file vgtl_graph.h.

9.47.2.9 typedef std::reverse_iterator<const_iterator> __ITree::const_reverse_iterator [inherited]

the const reverse iterator

Definition at line 2069 of file vgtl_tree.h.

9.47.2.10 typedef _Tree_walker<_Key ,const _Key &,const _Key *,container_type,container_iterator> __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::const_walker [inherited]

the (recursive) const walker

Reimplemented from __Tree_t.

Definition at line 1278 of file vgtl_graph.h.

9.47.2.11 typedef ptrdiff_t __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::difference_type [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1255 of file vgtl_graph.h.

9.47.2.12 typedef _Tree_walker<_Tp,_Tp&,_Tp*,container_type,children_iterator,_Node> __ITree::iterative_walker [inherited]

the iterative walker

Definition at line 2063 of file vgtl_tree.h.

9.47.2.13 typedef _Tree_iterator<_Key ,_Key &,_Key *,container_type,container_iterator> __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::iterator [inherited]

the iterator

Reimplemented from __Tree_t.

Definition at line 1262 of file vgtl_graph.h.

9.47.2.14 typedef _Tree_iterator<_Tp,_Tp&,_Tp*,container_type,children_iterator,node_type> __ITree::iterator [inherited]

the iterator

Definition at line 2058 of file vgtl_tree.h.

9.47.2.15 typedef _Node __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::node_type [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1249 of file vgtl_graph.h.

9.47.2.16 template<class _Tp, class _Ctr, class _Tl, class _Node, class _Alloc> typedef __one_iterator<void *> _Tree_base< _Tp, _Ctr, _Tl, _Node, _Alloc >::parents_iterator [inherited]

iterator for accessing the parents

Reimplemented in __Tree_t.

Definition at line 1447 of file vgtl_tree.h.

9.47.2.17 typedef __one_iterator<void *> __Tree_t::parents_iterator [inherited]

iterator for accessing the parents

Reimplemented from _Tree_base< _Tp, _Ctr, _Iterator, _Node, _Alloc >.

Definition at line 1564 of file vgtl_tree.h.

9.47.2.18 typedef value_type* __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::pointer [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1250 of file vgtl_graph.h.

9.47.2.19 typedef value_type& __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::reference [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1252 of file vgtl_graph.h.

9.47.2.20 typedef reverse_iterator<iterator> __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::reverse_iterator [inherited]

the reverse iterator

Reimplemented from __Tree_t.

Definition at line 1267 of file vgtl_graph.h.

9.47.2.21 typedef std::reverse_iterator<iterator> __ITree::reverse_iterator [inherited]

the reverse iterator

Definition at line 2071 of file vgtl_tree.h.

9.47.2.22 typedef size_t __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::size_type [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1254 of file vgtl_graph.h.

9.47.2.23 typedef _Key __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::value_type [inherited]

standard typedef

Reimplemented from __Tree_t.

Definition at line 1248 of file vgtl_graph.h.

9.47.2.24 typedef _Tree_walker<_Key ,_Key &,_Key *,container_type,container_iterator> __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::walker [inherited]

the (recursive) walker

Reimplemented from __Tree_t.

Definition at line 1277 of file vgtl_graph.h.

9.47.3 Member Function Documentation

9.47.3.1 _Node* __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::_C_create_node ( ) [inline, protected, inherited]

construct a new tree node containing default data

Reimplemented from __Tree_t.

Definition at line 1308 of file vgtl_graph.h.

9.47.3.2 _Node* __Tree_t::_C_create_node (const _Tp & __x) [inline, protected, inherited]

construct a new tree node containing data ___x

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >, and __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >.

Definition at line 1629 of file vgtl_tree.h.

9.47.3.3 _Node* _Tree_alloc_base< _Tp, _Ctr, _Iterator , _Node, _Alloc , _IsStatic >::_C_get_node ( ) [inline, protected, inherited]

allocate a new node

Definition at line 1375 of file vgtl_tree.h.

9.47.3.4 void _Tree_alloc_base< _Tp, _Ctr, _Iterator , _Node, _Alloc , _IsStatic >::_C_put_node (_Node * __p) [inline, protected, inherited]

deallocate a node

Definition at line 1378 of file vgtl_tree.h.

9.47.3.5 template<class _Tp, class _Ctr, class _TI, class _Node, class _Allocator, bool _IsStatic> void _Tree_alloc_base< _Tp, _Ctr, _TI, _Node, _Allocator, _IsStatic >::_C_put_node (_Node * __p) [inline, protected, inherited]

deallocate a node

Definition at line 1378 of file vgtl_tree.h.

9.47.3.6 void _Tree_alloc_base< _Tp, _Ctr, _Tl, _Node, _Alloc , _IsStatic >::_C_put_node (_Node * __p) [inline, protected, inherited]

deallocate a node

Definition at line 1378 of file vgtl_tree.h.

9.47.3.7 void _Tree_alloc_base< _Tp, _Ctr, _Tl, _Alloc , _Alloc_traits< _Tp, _Alloc >::_S_instanceless , _IsStatic >::_C_put_node (_Alloc * __p) [inline, protected, inherited]

deallocate a node

Definition at line 1378 of file vgtl_tree.h.

9.47.3.8 void _Tree_base< _Tp, _Ctr, _Iterator, _Node, _Alloc >::add_all_children (_Output_Iterator *fi,* _Node * *_parent*) [inline, inherited]

add all children to the parent _parent. fi is a iterator to the children container of the parent

Definition at line 1539 of file vgtl_tree.h.

9.47.3.9 template<class _Tp, class _Ctr, class _Tl, class _Node, class _Alloc> template<class _Output_Iterator > void _Tree_base< _Tp, _Ctr, _Tl, _Node, _Alloc >::add_all_children (_Output_Iterator *fi,* _Node * *_parent*) [inherited]

add all children to the parent _parent. fi is a iterator to the children container of the parent

9.47.3.10 iterator __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::begin ( ) [inline, inherited]

return an iterator to the first node in walk

Definition at line 1964 of file vgtl_tree.h.

9.47.3.11 const_iterator __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::begin ( ) const [inline, inherited]

return a const iterator to the first node in walk

Definition at line 1973 of file vgtl_tree.h.

9.47.3.12 iterative_walker __ITree::begin (walker_type *wt =* cw_pre_post, bool *front_to_back =* true, bool *depth_first =* true) [inline, inherited]

the walker to the first node of the complete walk

Definition at line 2122 of file vgtl_tree.h.

9.47.3.13 const_iterative_walker __ITree::begin (walker_type *wt =* cw_pre_post, bool *front_to_back =* true, bool *depth_first =* true) const [inline, inherited]

the const walker to the first node of the complete walk

Definition at line 2129 of file vgtl_tree.h.

9.47.3.14 void _Tree_base< _Tp, _Ctr, _Iterator , _Node, _Alloc >::clear_children ( ) [inline, inherited]

clear all children of the root node

Definition at line 1466 of file vgtl_tree.h.

9.47.3.15 size_type __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::depth (const walker & __position) [inline, inherited]

Reimplemented from __Tree_t.

Definition at line 1526 of file vgtl_graph.h.

9.47.3.16 size_type __ITree::depth (const iterative_walker & __position) [inline, inherited]

return the depth of this `__position` in the tree

Definition at line 2177 of file vgtl_tree.h.

9.47.3.17 bool __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::empty ( ) const [inline, inherited]

is the tree empty?

Reimplemented from __Tree_t.

Definition at line 1392 of file vgtl_graph.h.

9.47.3.18 iterator __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::end ( ) [inline, inherited]

return an iterator beyond the last node in walk

Definition at line 1968 of file vgtl_tree.h.

9.47.3.19 const_iterator __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::end ( ) const [inline, inherited]

return a const iterator beyond the last node in walk

Definition at line 1977 of file vgtl_tree.h.

9.47.3.20 iterative_walker __ITree::end (walker_type *wt* = `cw_pre_post`, bool *front_to_back* = `true`, bool *depth_first* = `true`) [inline, inherited]

the walker beyond the last node of the walk

Definition at line 2137 of file vgtl_tree.h.

9.47.3.21 const_iterative_walker __ITree::end (walker_type *wt* = `cw_pre_post`, bool *front_to_back* = `true`, bool *depth_first* = `true`) const [inline, inherited]

the const walker beyond the last node of the walk

Definition at line 2143 of file vgtl_tree.h.

9.47.3.22 void __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::erase (const __walker_base & __position) [inline, inherited]

erase the node at position __position.

Reimplemented from __Tree_t.

Definition at line 1444 of file vgtl_graph.h.

9.47.3.23 bool __Tree_t::erase_child (const __walker_base & __position, const children_iterator & __It) [inline, inherited]

erase the (leaf) child __It of node __position. This works if and only if the child is a leaf.

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >.

Definition at line 1770 of file vgtl_tree.h.

9.47.3.24 _Node* __Tree_t::erase_subtree (const __walker_base & __position, const children_iterator & __It) [inline, inherited]

erase the subtree position __position, whose top node is the child at children_iterator position __It, and return its top node.

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >.

Definition at line 1790 of file vgtl_tree.h.

9.47.3.25 _Node* __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::erase_tree (const __walker_base & __position) [inline, inherited]

erase the subtree starting at position __position, and return its top node.

Reimplemented from __Tree_t.

Definition at line 1471 of file vgtl_graph.h.

9.47.3.26 allocator_type __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::get_allocator ( ) const [inline, inherited]

construct an allocator object

Reimplemented from __Tree_t.

Definition at line 1259 of file vgtl_graph.h.

9.47.3.27 reference __ITree::getroot ( ) [inline, inherited]

get a reference to the virtual root node

Definition at line 2172 of file vgtl_tree.h.

9.47.3.28 const_reference __ITree::getroot ( ) const [inline, inherited]

get a const reference to the virtual root node

Definition at line 2174 of file vgtl_tree.h.

9.47.3.29 walker __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::ground ( ) [inline, inherited]

return a walker to the virtual root node.

Definition at line 1939 of file vgtl_tree.h.

9.47.3.30 const_walker __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::ground ( ) const [inline, inherited]

return a const walker to the virtual root node.

Definition at line 1943 of file vgtl_tree.h.

9.47.3.31 void __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::insert_child (const __walker_base & *__position,* const container_insert_arg & *__It*) [inline, inherited]

add a child below __position with default data, at the __It position in the __position - node's children container

Reimplemented from __Tree_t.

Definition at line 1415 of file vgtl_graph.h.

9.47.3.32 void __Tree_t::insert_child (const __walker_base & *__position,* const _Tp & *__x,* const container_insert_arg & *__It*) [inline, inherited]

add a child below __position with data __x, at the __It position in the __position - node's children container

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >, __Tree< _Tp, _SequenceCtr< void *, _PtrAlloc >, _SequenceCtr< void *, _PtrAlloc >::iterator, _SequenceCtr< void *, _PtrAlloc >::iterator, _Alloc >, and __Tree< _Tp, _AssocCtr< _Key, void *, _Compare, _PtrAlloc >, pair_adaptor< _AssocCtr< _Key, void *, _Compare, _PtrAlloc >::iterator >, _Key, _Alloc >.

Definition at line 1668 of file vgtl_tree.h.

9.47.3.33 void __Tree_t::insert_children (const __walker_base & *__position,* size_type *__n,* const _Tp & *__x,* const children_iterator & *__It*) [inline, inherited]

add __n children below __position with data __x, after the __It position in the __position - node's children container

Reimplemented in __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >.

Definition at line 1682 of file vgtl_tree.h.

9.47.3.34 void __Tree_t::insert_subtree (const __walker_base & *__position,* _Self & *__subtree,* const children_iterator & *__It*) [inline, inherited]

add a complete subtree __subtree below position __position and children iterator position __It.

Definition at line 1702 of file vgtl_tree.h.

9.47.3.35 size_type __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::max_size ( ) const [inline, inherited]

return the maximum possible size of the tree (theor. infinity)

Reimplemented from __Tree_t.

Definition at line 1400 of file vgtl_graph.h.

9.47.3.36 template<class _Key , class _Compare = less<_Key>, template< class __Key, class __Compare, class __AllocT > class _AssocCtr = std::multiset, class _PtrAlloc = __VGTL_DEFAULT_ALLOCATOR(void *), class _Alloc = __VGTL_DEFAULT_ALLOCATOR(_Key&)> _Self& stree< _Key, _Compare, _AssocCtr, _PtrAlloc, _Alloc >::operator= (_Node * __x) [inline]

assign a tree from one node -> make this node the root node. This is useful for making trees out of erased subtrees.

Reimplemented from __Tree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc >.

Definition at line 2780 of file vgtl_tree.h.

9.47.3.37 reverse_iterator __lTree::rbegin ( ) [inline, inherited]

return a reverse iterator to the first node in walk

Definition at line 2151 of file vgtl_tree.h.

9.47.3.38 const_reverse_iterator __lTree::rbegin ( ) const [inline, inherited]

return a const reverse iterator to the first node in walk

Definition at line 2158 of file vgtl_tree.h.

9.47.3.39 reverse_iterator __lTree::rend ( ) [inline, inherited]

return a reverse iterator beyond the last node in walk

Definition at line 2154 of file vgtl_tree.h.

9.47.3.40 const_reverse_iterator __lTree::rend ( ) const [inline, inherited]

return a const reverse iterator beyond the last node in walk

Definition at line 2161 of file vgtl_tree.h.

9.47.3.41 walker __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::root (children_iterator __it) [inline, inherited]

return a walker to a root node.

Definition at line 1947 of file vgtl_tree.h.

9.47.3.42 const_walker __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::root (children_iterator __it) const [inline, inherited]

return a const walker to a root node.

Definition at line 1952 of file vgtl_tree.h.

9.47.3.43 walker __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::root ( ) [inline, inherited]

return a walker to the first non-virtual tree root

Definition at line 1957 of file vgtl_tree.h.

9.47.3.44 const_walker __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::root ( ) const [inline, inherited]

return a const walker to the first non-virtual tree root

Definition at line 1960 of file vgtl_tree.h.

9.47.3.45 iterative_walker __ITree::root (walker_type *wt* = cw_pre_post, bool *front_to_back* = true, bool *depth_first* = true) [inline, inherited]

return an iterative walker of type wt to the ground node

Definition at line 2099 of file vgtl_tree.h.

9.47.3.46 const_iterative_walker __ITree::root (walker_type *wt* = cw_pre_post, bool *front_to_back* = true, bool *depth_first* = true) const [inline, inherited]

return a const iterative walker of type wt to the ground node

Definition at line 2106 of file vgtl_tree.h.

9.47.3.47 size_type __ITree::size ( ) const [inline, inherited]

return the size of the tree (# of nodes)

Definition at line 2165 of file vgtl_tree.h.

9.47.3.48 void __Tree< _Key , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc > , _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator , _Key & , _Alloc >::swap (_Self & __x) [inline, inherited]

Reimplemented from __Tree_t.

Definition at line 1405 of file vgtl_graph.h.

9.47.3.49 iterative_walker __ITree::through ( ) [inline, inherited]

the walker beyond the complete walk

Definition at line 2113 of file vgtl_tree.h.

9.47.3.50 const_iterative_walker __ITree::through ( ) const [inline, inherited]

the const walker beyond the complete walk

Definition at line 2117 of file vgtl_tree.h.

9.47.4 Friends And Related Function Documentation

9.47.4.1 bool operator==__VGTL_NULL_TMPL_ARGS (const __ITree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc > & __x, const __ITree< _Key, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >, _AssocCtr< _Key &, pointer_adaptor< _Compare >, _PtrAlloc >::iterator, _Key &, _Alloc > & __y) [friend, inherited]

comparison operator

9.47.5 Member Data Documentation

9.47.5.1 _Node* _Tree_alloc_base< _Tp, _Ctr, _Iterator , _Node, _Alloc , _IsStatic >::_C_node [protected, inherited]

This is the node

Definition at line 1387 of file vgtl_tree.h.

The documentation for this class was generated from the following files:

- vgtl_graph.h
- vgtl_tree.h

10 File Documentation

10.1 array_vector.h File Reference

Classes

- class array_vector

Defines

- #define VGTL_VECTOR_IMPL

  STL vector wrapper for C array.

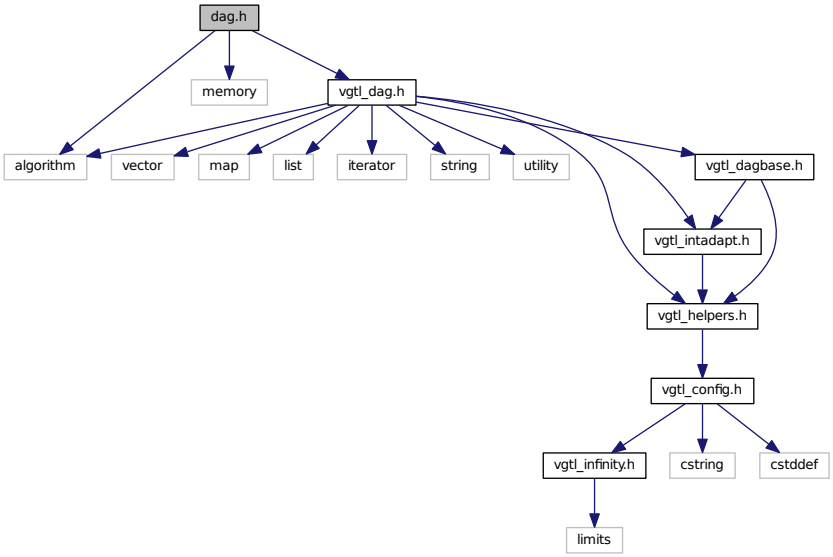
10.1.1 Detailed Description

This is the external header file intended for direct use.

Definition in file array_vector.h.

10.2 dag.h File Reference

`#include <algorithm>` `#include <memory>` `#include <vgtl_dag.h>` Include dependency graph for dag.h:



10.2.1 Detailed Description

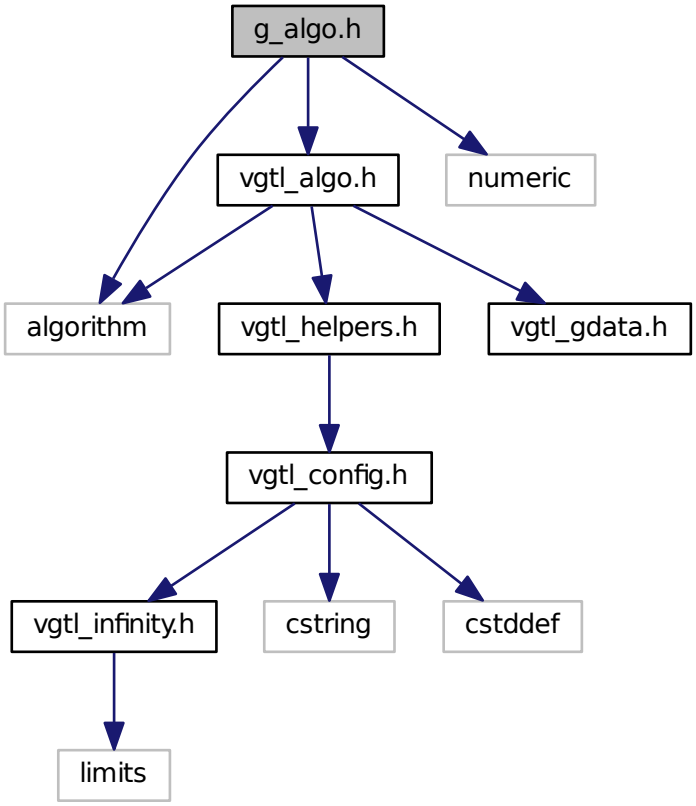
This is the external header file intended for direct use.

Definition in file dag.h.

10.3 g_algo.h File Reference

`#include <algorithm>` `#include <vgtl_algo.h>` `#include <numeric>` Include de-

pendency graph for g_algo.h:



10.3.1 Detailed Description

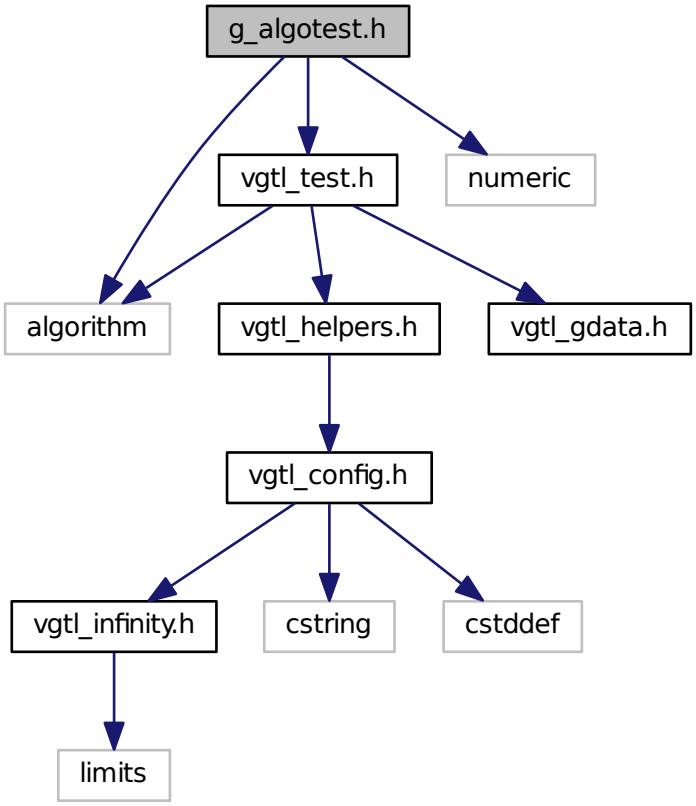
This is the external header file intended for direct use.

Definition in file g_algo.h.

10.4 g_algotest.h File Reference

`#include <algorithm>` `#include <vgtl_test.h>` `#include <numeric>` Include de-

pendency graph for g_algotest.h:



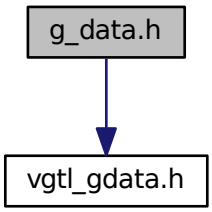
10.4.1 Detailed Description

This is the external header file intended for direct use.

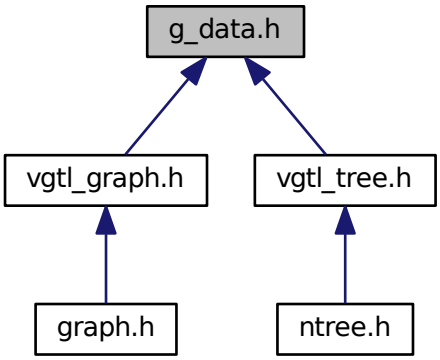
Definition in file g_algotest.h.

10.5 g_data.h File Reference

`#include <vgtl_gdata.h>` Include dependency graph for g_data.h:



This graph shows which files directly or indirectly include this file:



10.5.1 Detailed Description

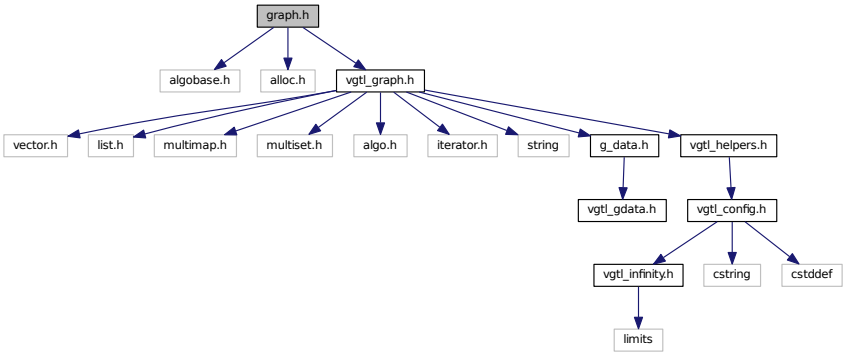
This is the external header file intended for direct use.

Definition in file g_data.h.

10.6 graph.h File Reference

`#include <algobase.h>` `#include <alloc.h>` `#include <vgtl_graph.h>` Include

dependency graph for graph.h:



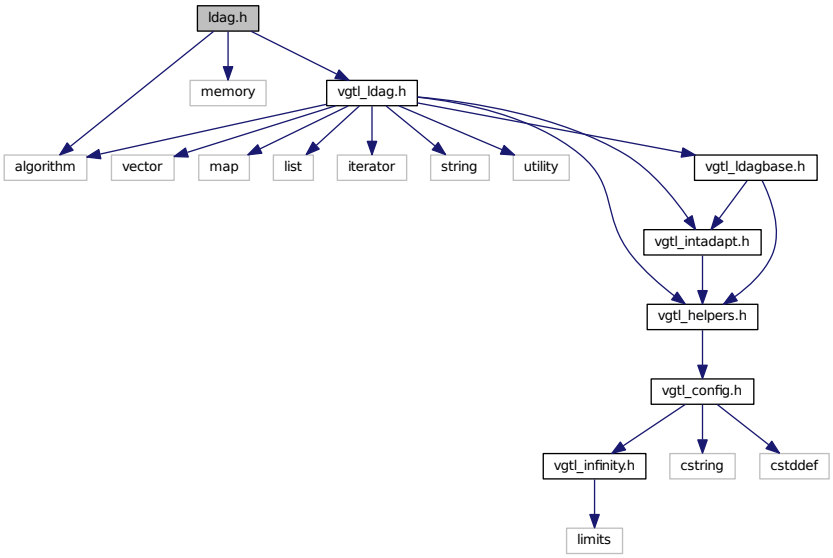
10.6.1 Detailed Description

This is the external header file intended for direct use.

Definition in file graph.h.

10.7 ldag.h File Reference

`#include <algorithm> #include <memory> #include <vgtl_ldag.h>` Include dependency graph for ldag.h:



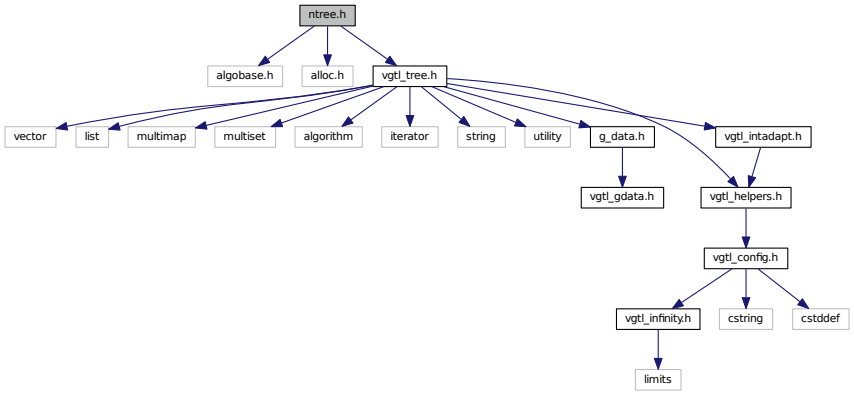
10.7.1 Detailed Description

This is the external header file intended for direct use.

Definition in file ldag.h.

10.8 ntree.h File Reference

`#include <algobase.h>` `#include <alloc.h>` `#include <vgtl_tree.h>` Include dependency graph for ntree.h:



10.8.1 Detailed Description

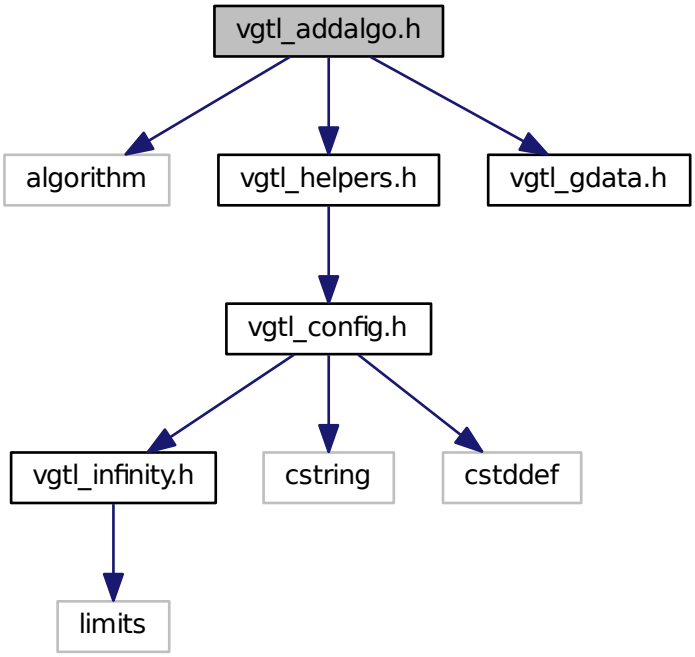
This is the external header file intended for direct use.

Definition in file ntree.h.

10.9 vgtl_addalgo.h File Reference

`#include <algorithm>` `#include <vgtl_helpers.h>` `#include <vgtl_gdata.h>` ×

Include dependency graph for vgtl_addalgo.h:



Functions

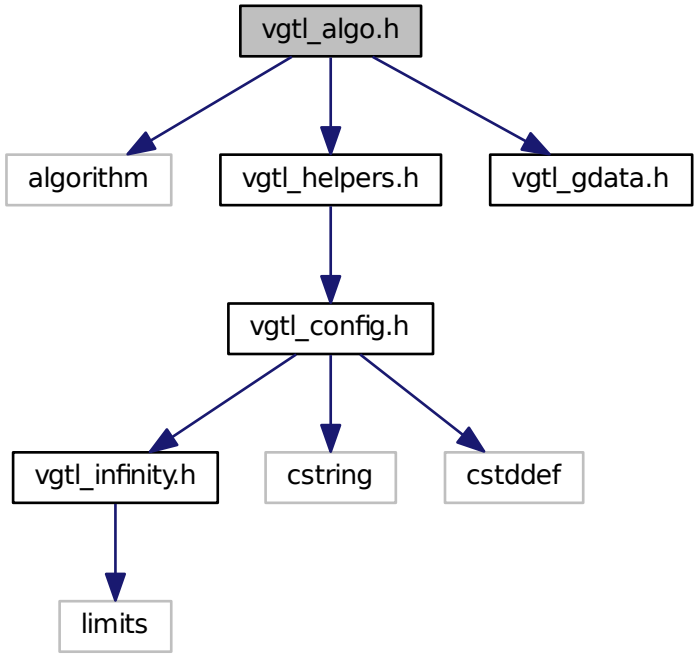
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_safe_walk_if (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value _recursive_safe_walk_if (_Walker __w, _Visitor __f)

10.9.1 Detailed Description

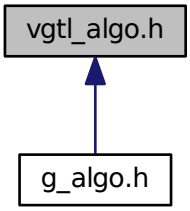
Definition in file vgtl_addalgo.h.

10.10 vgtl_algo.h File Reference

`#include <algorithm>` `#include <vgtl_helpers.h>` `#include <vgtl_gdata.h>` ×
Include dependency graph for vgtl_algo.h:



This graph shows which files directly or indirectly include this file:



Classes

- class __Child_data_iterator< _Iterator, _Node >

  iterator adapter for iterating through children data hooks
- class child_data_iterator< _Tree >

  Iterator which iterates through the data hooks of all children.

Functions

- template<class _IterativeWalker , class _Function >
  _Function walk (_IterativeWalker __first, _IterativeWalker __last, _Function __f)
- template<class _PrePostWalker , class _Function >
  _Function pre_post_walk (_PrePostWalker __first, _PrePostWalker __last, _Function __f)
- template<class _PrePostWalker , class _Function1 , class _Function2 >
  _Function2 pre_post_walk (_PrePostWalker __first, _PrePostWalker __last, _Function1 __f1, _Function2 __f2)
- template<class _PrePostWalker , class _Function >
  _Function var_walk (_PrePostWalker __first, _PrePostWalker __last, _Function __f)
- template<class _PrePostWalker , class _Function1 , class _Function2 >
  _Function2 var_walk (_PrePostWalker __first, _PrePostWalker __last, _Function1 __f1, _Function2 __f2)
- template<class _PrePostWalker , class _Function , class _Predicate >
  _Function walk_if (_PrePostWalker __first, _PrePostWalker __last, _Function __f, _Predicate __pred)
- template<class _PrePostWalker , class _Function1 , class _Function2 , class _Predicate >
  _Function2 walk_if (_PrePostWalker __first, _PrePostWalker __last, _Function1 __f1, _Function2 __f2, _Predicate __pred)
- template<class _PrePostWalker , class _Function1 , class _Function2 , class _Predicate1 , class _Predicate2 >
  _Function2 walk_if (_PrePostWalker __first, _PrePostWalker __last, _Function1 __f1, _Function2 __f2, _Predicate1 __pred1, _Predicate2 __pred2)
- template<class _PrePostWalker , class _Function1 , class _Function2 , class _Predicate >
  _Function2 cached_walk_if (_PrePostWalker __first, _PrePostWalker __last, _Function1 __f1, _Function2 __f2, _Predicate __pred)
- template<class _PrePostWalker , class _Function1 , class _Function2 , class _Predicate >
  _Function2 multi_walk_if (_PrePostWalker __first, _PrePostWalker __last, _Function1 __f1, _Function2 __f2, _Predicate __pred)
- template<class _Walker , class _Function >
  _Function walk_up (_Walker __w, _Function __f)
- template<class _Walker , class _Function >
  _Function var_walk_up (_Walker __w, _Function __f)
- template<class _Walker , class _Function , class _Predicate >
  _Function walk_up_if (_Walker __w, _Function __f, _Predicate __p)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_preorder_walk (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value _recursive_preorder_walk (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_postorder_walk (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value _recursive_postorder_walk (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_walk (_Walker __w, _Visitor __f)

- template<class _Walker , class _Visitor >
  _Visitor::return_value _recursive_walk (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_preorder_walk_if (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value _recursive_preorder_walk_if (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor , class _Predicate >
  _Visitor::return_value recursive_preorder_walk_if (_Walker __w, _Visitor __f, _Predicate __p)
- template<class _Walker , class _Visitor , class _Predicate >
  _Visitor::return_value _recursive_preorder_walk_if (_Walker __w, _Visitor __f, _Predicate __p)
- template<class _Walker , class _Visitor , class _Predicate >
  _Visitor::return_value recursive_postorder_walk_if (_Walker __w, _Visitor __f, _Predicate __p)
- template<class _Walker , class _Visitor , class _Predicate >
  _Visitor::return_value _recursive_postorder_walk_if (_Walker __w, _Visitor __f, _Predicate __p)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_walk_if (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value _recursive_walk_if (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_cached_walk (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value _recursive_cached_walk (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_multi_walk (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value _recursive_multi_walk (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor , class _Predicate1 , class _Predicate2 >
  _Visitor::return_value recursive_walk_if (_Walker __w, _Visitor __f, _Predicate1 __p1, _Predicate2 __p2)
- template<class _Walker , class _Visitor , class _Predicate1 , class _Predicate2 >
  _Visitor::return_value _recursive_walk_if (_Walker __w, _Visitor __f, _Predicate1 __p1, _Predicate2 __p2)
- template<class _Walker , class _Visitor , class _Predicate >
  _Visitor::return_value recursive_cached_walk (_Walker __w, _Visitor __f, _Predicate __p)
- template<class _Walker , class _Visitor , class _Predicate >
  _Visitor::return_value _recursive_cached_walk (_Walker __w, _Visitor __f, _Predicate __p)
- template<class _Walker , class _Visitor , class _Predicate >
  _Visitor::return_value recursive_multi_walk (_Walker __w, _Visitor __f, _Predicate __p)
- template<class _Walker , class _Visitor , class _Predicate >
  _Visitor::return_value _recursive_multi_walk (_Walker __w, _Visitor __f, _Predicate __p)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_preorder_walk_up (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value _recursive_preorder_walk_up (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_preorder_walk_up_if (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value _recursive_preorder_walk_up_if (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor , class _Predicate >
  _Visitor::return_value recursive_preorder_walk_up_if (_Walker __w, _Visitor __f, _Predicate __p)
- template<class _Walker , class _Visitor , class _Predicate >
  _Visitor::return_value _recursive_preorder_walk_up_if (_Walker __w, _Visitor __f, _Predicate __p)

- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_postorder_walk_up (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value _recursive_postorder_walk_up (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor , class _Predicate >
  _Visitor::return_value recursive_postorder_walk_up_if (_Walker __w, _Visitor __f, _Predicate __p)
- template<class _Walker , class _Visitor , class _Predicate >
  _Visitor::return_value _recursive_postorder_walk_up_if (_Walker __w, _Visitor __f, _Predicate __p)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_walk_up (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value _recursive_walk_up (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_walk_up_if (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value _recursive_walk_up_if (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor , class _Predicate1 , class _Predicate2 >
  _Visitor::return_value recursive_walk_up_if (_Walker __w, _Visitor __f, _Predicate1 __p1, _Predicate2 __p2)
- template<class _Walker , class _Visitor , class _Predicate1 , class _Predicate2 >
  _Visitor::return_value _recursive_walk_up_if (_Walker __w, _Visitor __f, _Predicate1 __p1, _Predicate2 __p2)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_cached_walk_up (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value _recursive_cached_walk_up (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_multi_walk_up (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value _recursive_multi_walk_up (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor , class _Predicate >
  _Visitor::return_value recursive_cached_walk_up (_Walker __w, _Visitor __f, _Predicate __p)
- template<class _Walker , class _Visitor , class _Predicate >
  _Visitor::return_value _recursive_cached_walk_up (_Walker __w, _Visitor __f, _Predicate __p)
- template<class _Walker , class _Visitor , class _Predicate >
  _Visitor::return_value recursive_multi_walk_up (_Walker __w, _Visitor __f, _Predicate __p)
- template<class _Walker , class _Visitor , class _Predicate >
  _Visitor::return_value _recursive_multi_walk_up (_Walker __w, _Visitor __f, _Predicate __p)
- template<class _Walker , class _Visitor >
  _Visitor::return_value general_directed_walk (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value general_directed_walk_down (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value general_directed_walk_up (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_general_directed_walk (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_general_directed_walk_down (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_general_directed_walk_up (_Walker __w, _Visitor __f)

- template<class _Walker , class _Visitor >
  _Visitor::return_value general_walk (_Walker __w, _Visitor __f)
- template<class _Walker , class _Visitor >
  _Visitor::return_value recursive_general_walk (_Walker __w, _Visitor __f)

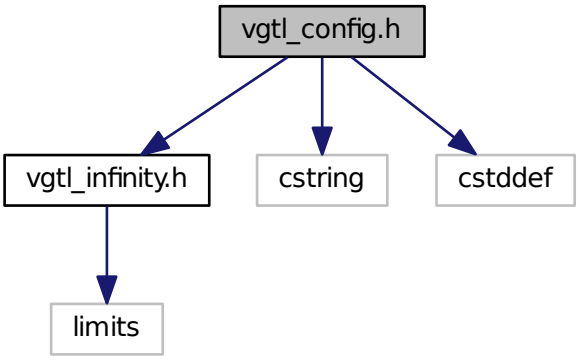
10.10.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

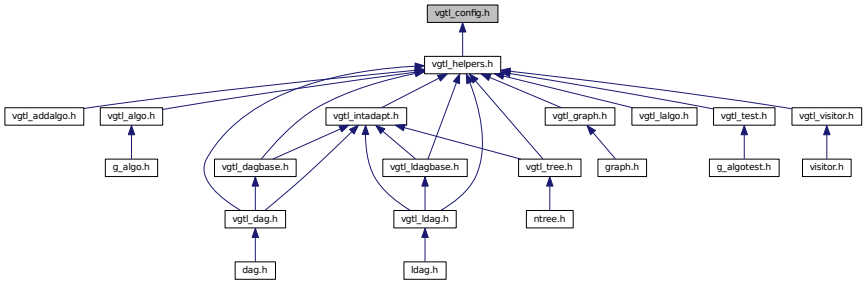
Definition in file vgtl_algo.h.

10.11 vgtl_config.h File Reference

`#include <vgtl_infinity.h>` `#include <cstring>` `#include <cstddef>` Include dependency graph for vgtl_config.h:



This graph shows which files directly or indirectly include this file:

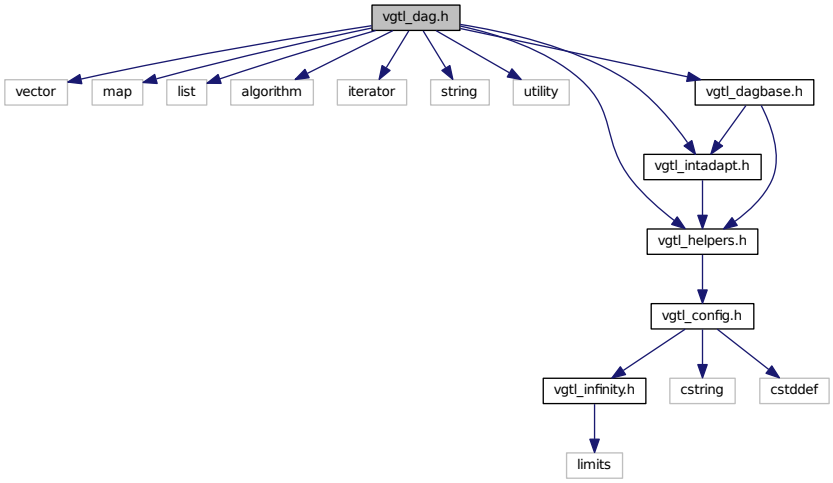


10.11.1 Detailed Description

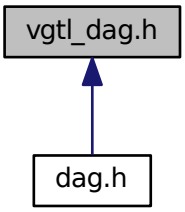
Definition in file vgtl_config.h.

10.12 vgtl_dag.h File Reference

`#include <vector>` `#include <map>` `#include <list>` `#include <algorithm>` × `#include <iterator>` `#include <string>` `#include <utility>` `#include <vgtl_helpers.h>` `#include <vgtl_intadapt.h>` `#include <vgtl_dagbase.h>` Include dependency graph for vgtl_dag.h:



This graph shows which files directly or indirectly include this file:



Classes

- class _DG_walker

recursive directed graph walkers

- class _DG_iterator

  iterator through the directed graph
- class __DG

  Directed graph base class.
- class dgraph

  unlabeled directed graph
- class dag

  unlabeled directed acyclic graph (DAG)

10.12.1 Detailed Description

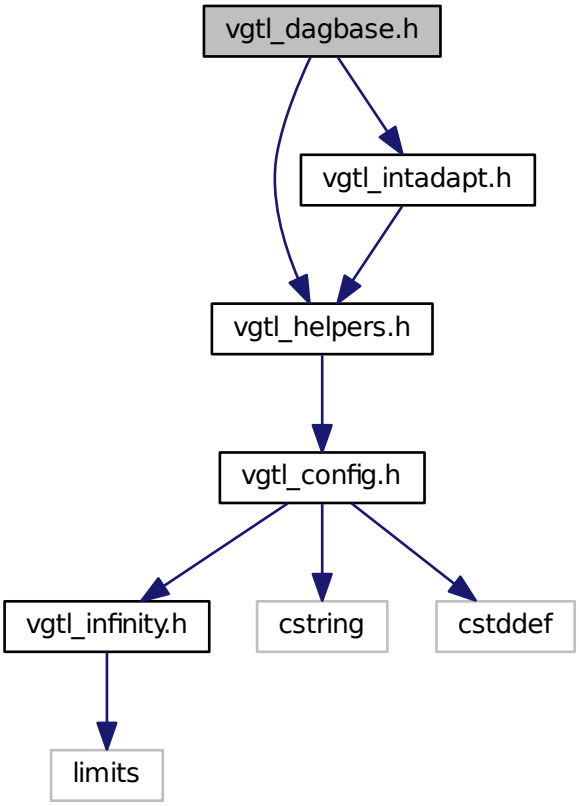
This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file vgtl_dag.h.

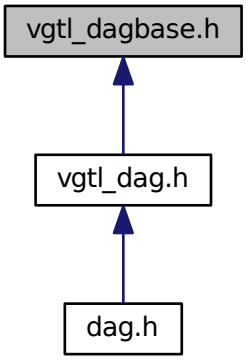
10.13 vgtl_dagbase.h File Reference

`#include <vgtl_helpers.h>` `#include <vgtl_intadapt.h>` Include dependency graph

for vgtl_dagbase.h:



This graph shows which files directly or indirectly include this file:



Classes

- class _DG_node
  directed graph node
- class _DG_base
  Directed graph base class for allocator encapsulation.

10.13.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file vgtl_dagbase.h.

10.14 vgtl_extradocu.h File Reference

Namespaces

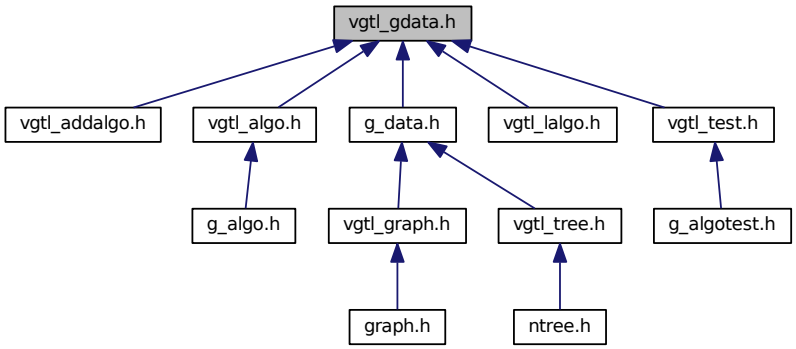
- namespace vgtl
  Main namespace of the VGTL.

10.14.1 Detailed Description

Definition in file vgtl_extradocu.h.

10.15 vgtl_gdata.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- union _Tree_data_hook

Typedefs

- typedef union _Tree_data_hook ctree_data_hook

10.15.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file vgtl_gdata.h.

10.15.2 Typedef Documentation

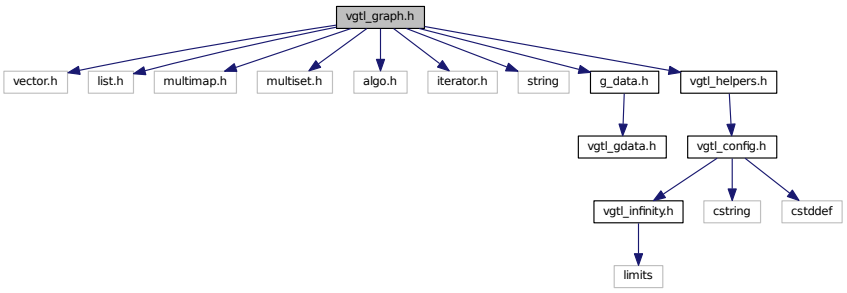
10.15.2.1 typedef union _Tree_data_hook ctree_data_hook

This is a mixed-type union for data hooks on trees. A data hook can be used for non-recursive walks.

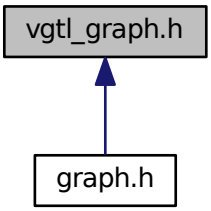
10.16 vgtl_graph.h File Reference

```
#include <vector.h> #include <list.h> #include <multimap.h> #include <multiset.-
h> #include <algo.h> #include <iterator.h> #include <string> #include <g-
```

_data.h> #include <vgtl_helpers.h> Include dependency graph for vgtl_graph.h:



This graph shows which files directly or indirectly include this file:



Classes

- class _Graph_walker_base
- class _Graph_walker
- class _RTree_walker< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >
  recursive tree walkers
- class _Tree_iterator< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >
  iterator through the tree
- class _Tree_alloc_base< _Tp, _Ctr, _TI, _Node, _Allocator, _IsStatic >
  Tree base class for general standard-conforming allocators.
- class _Tree_alloc_base< _Tp, _Ctr, _TI, _Allocator, true >
- class _Tree_base< _Tp, _Ctr, _TI, _Node, _Alloc >
  Tree base class for allocator encapsulation.
- class __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >
  Tree base class without data hooks.
- class ntree< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >
  n-ary forest
- class atree< _Tp, _AssocCtr, _Key, _Compare, _PtrAlloc, _Alloc >

n-ary forest with labelled edges

- class stree< _Key, _Compare, _AssocCtr, _PtrAlloc, _Alloc >

  n-ary forest with unsorted edges

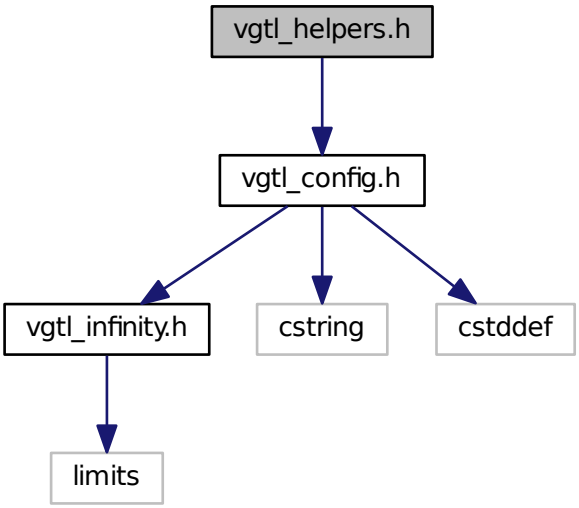
10.16.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

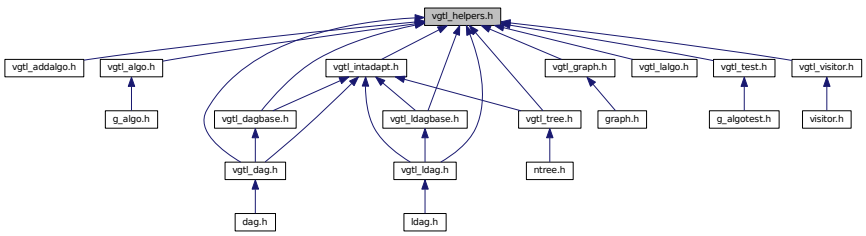
Definition in file vgtl_graph.h.

10.17 vgtl_helpers.h File Reference

`#include <vgtl_config.h>` Include dependency graph for vgtl_helpers.h:



This graph shows which files directly or indirectly include this file:



Functions

- template<class _BidirIter , class _Tp >
  _BidirIter rfind (_BidirIter __first, _BidirIter __last, const _Tp &__val, std::bidirectional_iterator_-tag)
- template<class _BidirIter , class _Predicate >
  _BidirIter rfind_if (_BidirIter __first, _BidirIter __last, _Predicate __pred, std::bidirectional_iterator-_tag)
- template<class _RandomAccessIter , class _Tp >
  _RandomAccessIter rfind (_RandomAccessIter __first, _RandomAccessIter __last, const _Tp &__-val, std::random_access_iterator_tag)
- template<class _RandomAccessIter , class _Predicate >
  _RandomAccessIter rfind_if (_RandomAccessIter __first, _RandomAccessIter __last, _Predicate _-_pred, std::random_access_iterator_tag)
- template<class _BidirIter , class _Tp >
  _BidirIter rfind (_BidirIter __first, _BidirIter __last, const _Tp &__val)
- template<class _BidirIter , class _Predicate >
  _BidirIter rfind_if (_BidirIter __first, _BidirIter __last, _Predicate __pred)

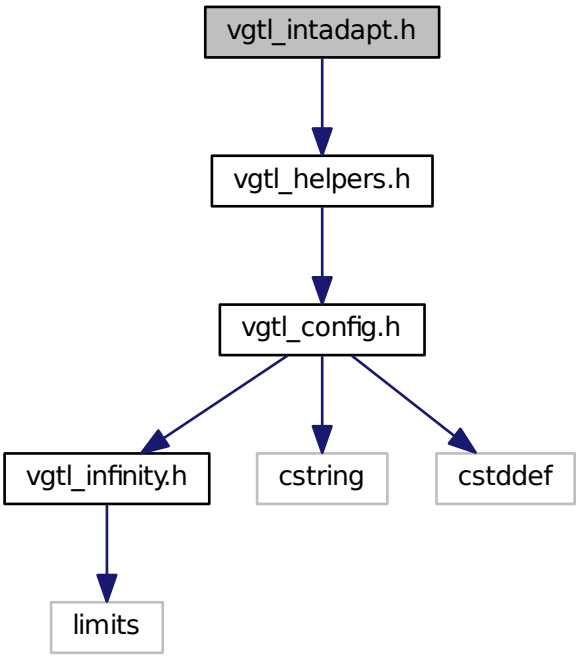
10.17.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

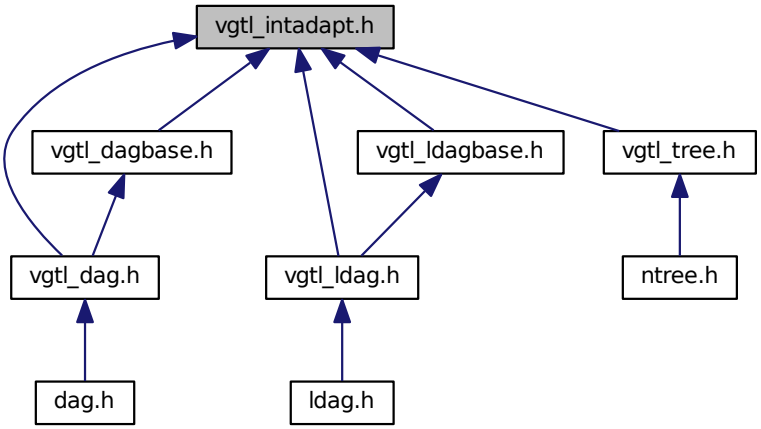
Definition in file vgtl_helpers.h.

10.18 vgtl_intadapt.h File Reference

`#include <vgtl_helpers.h>` Include dependency graph for vgtl_intadapt.h:



This graph shows which files directly or indirectly include this file:



Classes

- class pointer_adaptor

  adaptor transforming a comparison predicate to pointers
- class pair_adaptor

  adaptor for an iterator over a pair to an iterator returning the second element
- class __one_iterator

  make an iterator out of one pointer
- class _G_compare_adaptor

  Adaptor for data comparison in graph nodes.

10.18.1 Detailed Description

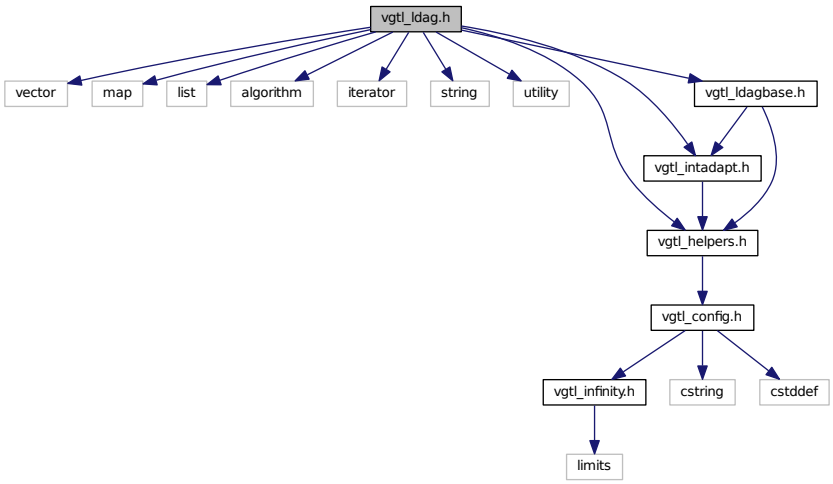
This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file vgtl_intadapt.h.

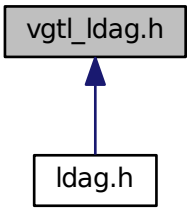
10.19 vgtl_ldag.h File Reference

`#include <vector>` `#include <map>` `#include <list>` `#include <algorithm>` × `#include <iterator>` `#include <string>` `#include <utility>` `#include <vgtl_helpers.h>` `#include <vgtl_intadapt.h>` `#include <vgtl_ldagbase.h>` Include

dependency graph for vgtl_ldag.h:



This graph shows which files directly or indirectly include this file:



Classes

- class _LDG_walker

  recursive labelled directed graph walkers
- class _LDG_iterator

  iterator through the directed graph
- class __LDG

  Labelled directed graph base class.
- class ldgraph

  labeled directed graph
- class ldag

  labeled directed acyclic graph (LDAG)

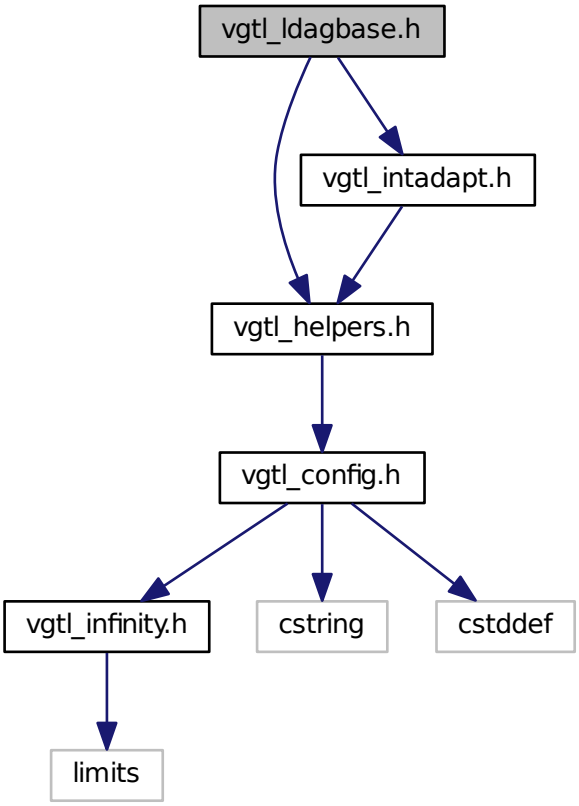
10.19.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

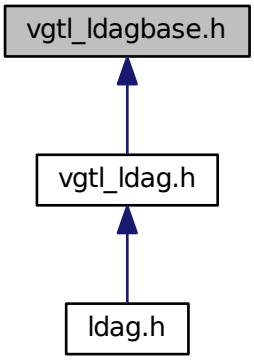
Definition in file vgtl_ldag.h.

10.20 vgtl_ldagbase.h File Reference

`#include <vgtl_helpers.h>` `#include <vgtl_intadapt.h>` Include dependency graph for vgtl_ldagbase.h:



This graph shows which files directly or indirectly include this file:



Classes

- class _LDG_node
  labelled directed graph node
- class _LDG_edge
  labelled directed graph edge
- class _LDG_base
  Labelled directed graph base class for allocator encapsulation.

10.20.1 Detailed Description

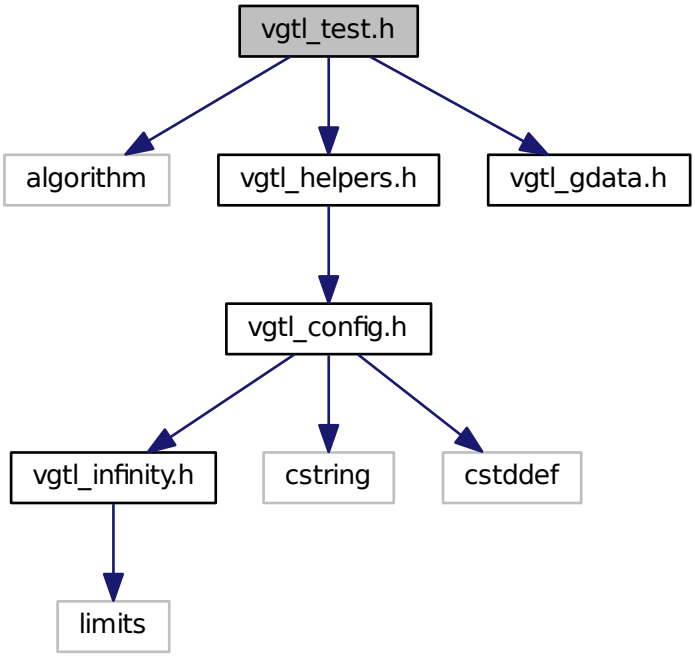
This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file vgtl_ldagbase.h.

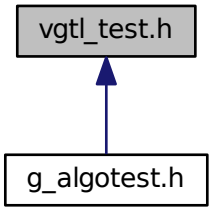
10.21 vgtl_test.h File Reference

```
#include <algorithm> #include <vgtl_helpers.h> #include <vgtl_gdata.h> ×
```

Include dependency graph for vgtl_test.h:



This graph shows which files directly or indirectly include this file:



Functions

- template<class _Walker , class _Test >
  void recursive_consistency_test (_Walker __w, const _Test &__t)

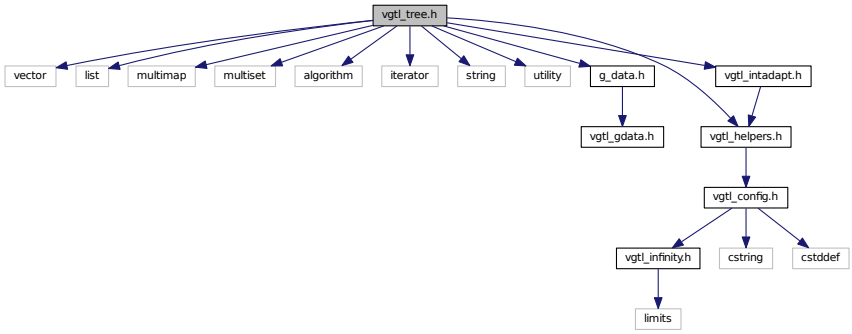
10.21.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

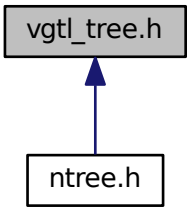
Definition in file vgtl_test.h.

10.22 vgtl_tree.h File Reference

`#include <vector> #include <list> #include <multimap> #include <multiset> #include <algorithm> #include <iterator> #include <string> #include <utility> #include <g_data.h> #include <vgtl_helpers.h> #include <vgtl_intadapt.h>` Include dependency graph for vgtl_tree.h:



This graph shows which files directly or indirectly include this file:



Classes

- class _Tree_node

  tree node for trees w/o data hooks
- class _ITree_node

  tree node for trees with data hooks
- class _Tree_walker_base

base class for all tree walkers

- class _Tree_walker

  automatic tree walkers
- class _RTree_walker< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >

  recursive tree walkers
- class _Tree_iterator< _Tp, _Ref, _Ptr, _Ctr, _Iterator, _Node >

  iterator through the tree
- class _Tree_alloc_base< _Tp, _Ctr, _TI, _Node, _Allocator, _IsStatic >

  Tree base class for general standard-conforming allocators.
- class _Tree_alloc_base< _Tp, _Ctr, _TI, _Node, _Allocator, true >

  Tree base class specialization for instanceless allocators.
- class _Tree_base< _Tp, _Ctr, _TI, _Node, _Alloc >

  Tree base class for allocator encapsulation.
- class __Tree_t

  Tree base class.
- class __Tree< _Tp, _Ctr, _Iterator, _Inserter, _Alloc >

  Tree base class without data hooks.
- class __ITree

  Tree base class with data hooks.
- class ntree< _Tp, _SequenceCtr, _PtrAlloc, _Alloc >

  n-ary forest
- class rntree

  n-ary forest
- class atree< _Tp, _AssocCtr, _Key, _Compare, _PtrAlloc, _Alloc >

  n-ary forest with labelled edges
- class stree< _Key, _Compare, _AssocCtr, _PtrAlloc, _Alloc >

  n-ary forest with unsorted edges
- class ratree

  n-ary forest with labelled edges
- class rstree

  n-ary forest with unsorted edges

Defines

- #define _C_W_preorder 1
- #define _C_W_postorder 2

Enumerations

- enum walker_type

10.22.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file vgtl_tree.h.

10.22.2 Define Documentation

10.22.2.1 #define _C_W_postorder 2

The walker is in postorder mode

Definition at line 47 of file vgtl_tree.h.

10.22.2.2 #define _C_W_preorder 1

The walker is in preorder mode

Definition at line 45 of file vgtl_tree.h.

10.22.3 Enumeration Type Documentation

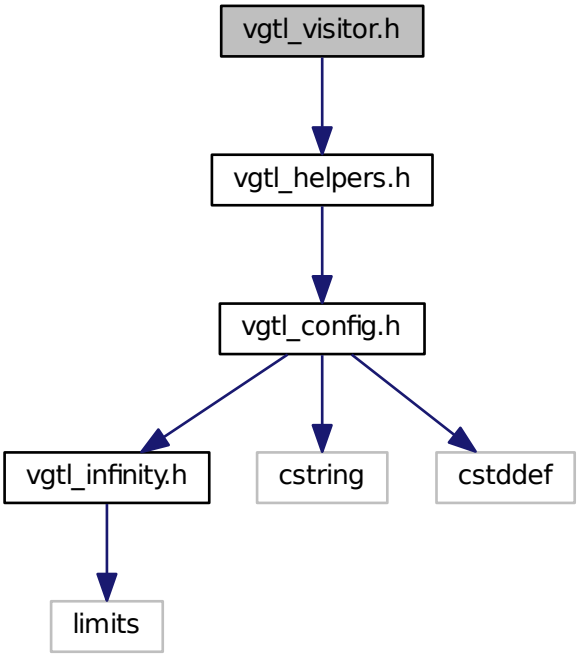
10.22.3.1 enum walker_type

enum for walker types: preorder, postorder, pre+postorder

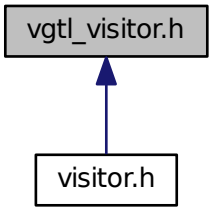
Definition at line 50 of file vgtl_tree.h.

10.23 vgtl_visitor.h File Reference

`#include <vgtl_helpers.h>` Include dependency graph for vgtl_visitor.h:



This graph shows which files directly or indirectly include this file:



Classes

- class preorder_visitor

  preorder visitor base class
- class postorder_visitor

  postorder visitor base class
- class prepost_visitor

  pre+postorder visitor base class

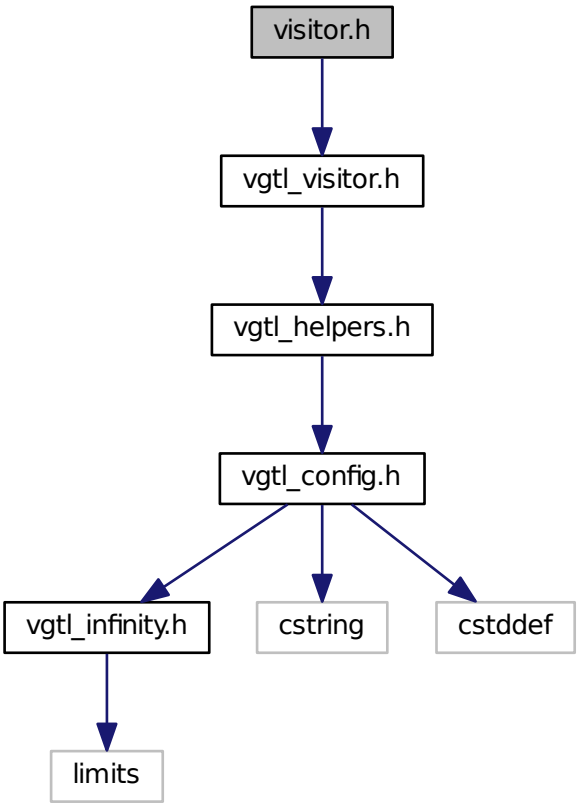
10.23.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file vgtl_visitor.h.

10.24 visitor.h File Reference

`#include <vgtl_visitor.h>` Include dependency graph for visitor.h:



10.24.1 Detailed Description

This is the external header file intended for direct use.

Definition in file visitor.h.