Compile-Time Graph Library

Generated by Doxygen 1.8.20

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 GLib Namespace Reference

Classes

- struct AddEdge
- struct AddEdge< ADJACENCY_LIST, graph, edge >
- struct DFS
- struct Filter
- struct Filter< id, graph, EmptyTypeList >
- struct FindNodeByVertex
- struct FindNodeByVertex< vertex, EmptyTypeList >
- struct FindPath
- struct ForEach
- struct ForEach< id, graph, EmptyTypeList >
- struct GetNodesFromRoots
- struct GetNodesFromRoots< EmptyTypeList, graph >
- struct GetReachedVertexes

5.2 Objects Namespace Reference

Classes

- struct Boolean
- struct Integer

5.2.1 Detailed Description

Represents class holders of different objects

5.3 TL Namespace Reference

Classes

- struct Add
- struct Add< T, 0, TypeList< Arg, Args... > >
- struct Add< T, 0, TypeList< Args... > >
- struct Add< T, ind, TypeList< Arg, Args... > >
- struct Concatenate
- struct Contains
- struct ContainsConstructibleParent
- struct ContainsConstructibleParent< EmptyTypeList, T >
- struct ContainsParent
- struct ContainsParent< EmptyTypeList, T >
- struct FillTypeListWithObject
- struct FillTypeListWithObject< obj, 0 >
- struct FindParentTypeList
- struct FindTypeListByClass
- struct HasDerivedAndConstructible
- struct HasDerivedAndConstructible< EmptyTypeList, T >
- struct IndexOf
- struct IndexOf< EmptyTypeList, T >
- struct IndexOf< type_list, typename type_list::Head >
- struct IsBaseOf
- struct IsBaseOf< EmptyTypeList, derived >
- struct IsBaseOf< EmptyTypeList, EmptyTypeList >
- struct IsBaseOf< parent, EmptyTypeList >
- struct IsTypeList
- struct IsTypeList< TypeList< Args... > >
- struct MostDerived
- struct MostDerived< EmptyTypeList, T >
- struct MostDerivedAndConstructible
- struct MostDerivedAndConstructible< EmptyTypeList, T >
- struct NoDuplicates
- struct NoDuplicates< EmptyTypeList >
- struct Remove
- struct Remove< EmptyTypeList, T >
- struct Remove< type_list, typename type_list::Head >
- struct RemoveAll
- struct RemoveAll< type_list, typename type_list::Head >
- struct Replace
- struct Replace< T, 0, TypeList< Arg, Args... > >
- struct Replace< T, ind, TypeList< Arg, Args... > >
- struct Reverse
- struct Size
- struct Size< EmptyTypeList >
- struct TypeAt
- struct TypeAt< type_list, 0 >

5.3.1 Detailed Description

Represents functions (as structs) for working with TypeList

Chapter 6

Class Documentation

6.1 TL::Add< T, ind, Arg, Args > Struct Template Reference

6.1.1 Detailed Description

template<typename T, size_t ind, class Arg, class ... Args>
struct TL::Add< T, ind, Arg, Args >

See also

Add<T, ind, TypeList<Arg, Args...>>

Definition at line 10 of file add.h.

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

- TL/add.h

6.2 TL::Add< T, 0, TypeList< Arg, Args... > > Struct Template Reference

`#include <add.h>`

Public Types

- using result = TypeList< T, Arg, Args... >

6.2.1 Detailed Description

template<typename T, class Arg, class ... Args>
struct TL::Add< T, 0, TypeList< Arg, Args... > >

See also

Add<T, ind, TypeList<Arg, Args...>>

Definition at line 33 of file add.h.

6.2.2 Member Typedef Documentation

6.2.2.1 result

```
template<typename T , class Arg , class ...  Args>
using TL::Add< T, 0, TypeList< Arg, Args...  > >::result = TypeList<T, Arg, Args...>
```

Definition at line 34 of file add.h.

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

- TL/add.h

6.3 TL::Add< T, 0, TypeList< Args... > > Struct Template Reference

```
#include <add.h>
```

Public Types

- using result = TypeList< T, Args... >

6.3.1 Detailed Description

template<typename T, class ... Args>
struct TL::Add< T, 0, TypeList< Args... > >

See also

Add<T, ind, TypeList<Arg, Args...>>

Definition at line 41 of file add.h.

6.3.2 Member Typedef Documentation

6.3.2.1 result

```
template<typename T , class ...  Args>
using TL::Add< T, 0, TypeList< Args...  > >::result = TypeList<T, Args...>
```

Definition at line 42 of file add.h.

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

- TL/add.h

6.4 TL::Add< T, ind, TypeList< Arg, Args... > > Struct Template Reference

`#include <add.h>`

Public Types

- using end = typename Add< T, ind - 1, TypeList< Args... > >::result
- using result = typename Add< Arg, 0, end >::result

6.4.1 Detailed Description

template<typename T, size_t ind, class Arg, class ... Args>
struct TL::Add< T, ind, TypeList< Arg, Args... > >

Adds typename to a specific position in TypeList

Parameters

T	Typename to add to a specific position in TypeList
ind	Number of this position
TypeList<Arg,Args...>	This TypeList @value Parameter value, new type list with typename added to position ind

Definition at line 19 of file add.h.

6.4.2 Member Typedef Documentation

6.4.2.1 end

```
template<typename T , size_t ind, class Arg , class ...  Args>
using TL::Add< T, ind, TypeList< Arg, Args...  > >::end = typename Add< T, ind - 1, TypeList<Args...>
>::result
```

Definition at line 20 of file add.h.

6.4.2.2 result

```
template<typename T , size_t ind, class Arg , class ...  Args>
using TL::Add< T, ind, TypeList< Arg, Args...  > >::result = typename Add<Arg, 0, end>↩
::result
```

Definition at line 26 of file add.h.

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

- TL/add.h

6.5 GLib::AddEdge< GraphType, graph, edge > Struct Template Reference

6.5.1 Detailed Description

template<GraphType, class graph, class edge>
struct GLib::AddEdge< GraphType, graph, edge >

Returns new graph with added edge

Parameters

GraphType	Template parameter, type of a graph
graph	Template parameter, initial graph
edge	Template parameter, edge to add

See also

Edge
GraphType

Returns

Parameter result, new graph with added edge

Definition at line 20 of file add_edge.h.

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

- graph/GLib/add_edge.h

6.6 GLib::AddEdge< ADJACENCY_LIST, graph, edge > Struct Template Reference

`#include <add_edge.h>`

Public Types

- using adjacent_vertexes = typename TL::TypeAt< typename graph::adjacency_list_, vertex_num >::value
- using new_adjacent_vertexes = typename TL::Add< edge, 0, adjacent_vertexes >::result
- using new_adjacency_list = typename TL::Replace< new_adjacent_vertexes, vertex_num, typename graph::adjacency_list_ >::result
- using result = AdjacencyListGraph< typename graph::vertexes_, new_adjacency_list >

Static Public Attributes

- constexpr static size_t vertex_num = graph::template GetVertexIndex<typename edge::from>()

6.6.1 Detailed Description

template<class graph, class edge>
struct GLib::AddEdge< ADJACENCY_LIST, graph, edge >

See also

AddEdge

Definition at line 26 of file add_edge.h.

6.6.2 Member Typedef Documentation

6.6.2.1 adjacent_vertexes

```
template<class graph , class edge >
using GLib::AddEdge< ADJACENCY_LIST, graph, edge >::adjacent_vertexes = typename TL::TypeAt<typename
graph::adjacency_list_, vertex_num>::value
```

Definition at line 28 of file add_edge.h.

6.6.2.2 new_adjacency_list

```
template<class graph , class edge >
using GLib::AddEdge< ADJACENCY_LIST, graph, edge >::new_adjacency_list = typename TL::Replace<new_adjacent_ve
vertex_num, typename graph::adjacency_list_>::result
```

Definition at line 31 of file add_edge.h.

6.6.2.3 new_adjacent_vertexes

```
template<class graph , class edge >
using GLib::AddEdge< ADJACENCY_LIST, graph, edge >::new_adjacent_vertexes = typename TL::Add<edge,
0, adjacent_vertexes>::result
```

Definition at line 30 of file add_edge.h.

6.6.2.4 result

```
template<class graph , class edge >
using GLib::AddEdge< ADJACENCY_LIST, graph, edge >::result = AdjacencyListGraph<typename
graph::vertexes_, new_adjacency_list>
```

Definition at line 32 of file add_edge.h.

6.6.3 Member Data Documentation

6.6.3.1 vertex_num

```
template<class graph , class edge >
constexpr static size_t GLib::AddEdge< ADJACENCY_LIST, graph, edge >::vertex_num = graph↩
::template GetVertexIndex<typename edge::from>()  [static], [constexpr]
```

Definition at line 27 of file add_edge.h.

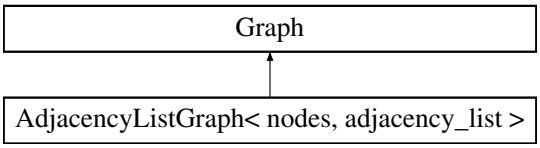
The documentation for this struct was generated from the following file:

- graph/GLib/add_edge.h

6.7 AdjacencyListGraph< nodes, adjacency_list > Struct Template Reference

```
#include <adjacency_list_graph.h>
```

Inheritance diagram for AdjacencyListGraph< nodes, adjacency_list >:



Classes

- struct ConvertTo

Public Types

- using vertexes_ = nodes

  TypeList of vertexes in graph.
- using adjacency_list_ = adjacency_list

  TypeList of TypeLists of edges, which are grouped by starting vertex.

Public Member Functions

- template<class edge >
  constexpr bool HasEdge ()

Static Public Member Functions

- template<typename vertex >
  constexpr static size_t GetVertexIndex ()

Static Public Attributes

- constexpr static GraphType TYPE = ADJACENCY_LIST

6.7.1 Detailed Description

template<class nodes, class adjacency_list>
struct AdjacencyListGraph< nodes, adjacency_list >

Represents graph vertexes defined in vertexes_, and edges, which are derived from adjacency_list_ Size of an adjacency list must be equal to amount of vertexes

See also

Graph

Parameters

*vertexes_*	TypeList of vertexes in graph.
*adjacency_list_*	- TypeList of TypeLists of edges, which are grouped by starting vertex i.e. edge (from, to, weight) goes to adjacency_list_[from]

Definition at line 24 of file adjacency_list_graph.h.

6.7.2 Member Typedef Documentation

6.7.2.1 adjacency_list_

```
template<class nodes , class adjacency_list >
using AdjacencyListGraph< nodes, adjacency_list >::adjacency_list_ = adjacency_list
```

TypeList of TypeLists of edges, which are grouped by starting vertex.

Definition at line 28 of file adjacency_list_graph.h.

6.7.2.2 vertexes_

```
template<class nodes , class adjacency_list >
using AdjacencyListGraph< nodes, adjacency_list >::vertexes_ = nodes
```

TypeList of vertexes in graph.

Definition at line 27 of file adjacency_list_graph.h.

6.7.3 Member Function Documentation

6.7.3.1 GetVertexIndex()

```
template<class nodes , class adjacency_list >
template<typename vertex >
constexpr static size_t AdjacencyListGraph< nodes, adjacency_list >::GetVertexIndex ( )  [inline],
[static], [constexpr]
```

Gets index of a passed vertex, throws assert if there is no such vertex

Parameters

vertex	Template parameter

Returns

Position of this vertex in vertexes_ TypeList

Definition at line 51 of file adjacency_list_graph.h.

6.7.3.2 HasEdge()

```
template<class nodes , class adjacency_list >
template<class edge >
constexpr bool AdjacencyListGraph< nodes, adjacency_list >::HasEdge ( )  [inline], [constexpr]
```

Checks if edge, passed as a template, is located in this graph

Parameters

edge	Template parameter, represents an edge to check

Returns

true if this edge in the graph, false otherwise

Definition at line 39 of file adjacency_list_graph.h.

6.7.4 Member Data Documentation

6.7.4.1 TYPE

```
template<class nodes , class adjacency_list >
constexpr static GraphType AdjacencyListGraph< nodes, adjacency_list >::TYPE = ADJACENCY_LIST
[static], [constexpr]
```

Definition at line 25 of file adjacency_list_graph.h.

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

- graph/graphs/adjacency_list_graph.h

6.8 AdjacencyMatrixGraph< vertexes, matrix > Struct Template Reference

```
#include <adjacency_matrix_graph.h>
```

Classes

- struct ConvertTo

Public Types

- using vertexes_ = vertexes

  TypeList of vertexes in graph.
- using matrix_ = matrix

  TypeList of TypeLists of Edges.

Static Public Attributes

- constexpr static GraphType TYPE = ADJACENCY_MATRIX

6.8.1 Detailed Description

template<class vertexes, class matrix>
struct AdjacencyMatrixGraph< vertexes, matrix >

Represents graph as an adjacency matrix. Element in row i, column j must be Objects::Boolean. If it's Boolean<false>, then there's no edge between them. Otherwise there is. If element is Boolean<true>, then the weight is NullType. Otherwise it's equal to this element. If you wish to have Boolean as a weight, then consider wrapping it in some other class.

See also

Objects::Boolean

Graph

Edge

Parameters

vertexes	Template parameter, vertexes of a graph
matrix	Template parameter, an adjacency matrix

Returns

Parameter result, resulting graph

Definition at line 25 of file adjacency_matrix_graph.h.

6.8.2 Member Typedef Documentation

6.8.2.1 matrix_

```
template<class vertexes , class matrix >
using AdjacencyMatrixGraph< vertexes, matrix >::matrix_ = matrix
```

TypeList of TypeLists of Edges.

Definition at line 30 of file adjacency_matrix_graph.h.

6.8.2.2 vertexes_

```
template<class vertexes , class matrix >
using AdjacencyMatrixGraph< vertexes, matrix >::vertexes_ = vertexes
```

TypeList of vertexes in graph.

Definition at line 29 of file adjacency_matrix_graph.h.

6.8.3 Member Data Documentation

6.8.3.1 TYPE

```
template<class vertexes , class matrix >
constexpr static GraphType AdjacencyMatrixGraph< vertexes, matrix >::TYPE = ADJACENCY_MATRIX
[static], [constexpr]
```

Definition at line 26 of file adjacency_matrix_graph.h.

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

- graph/graphs/adjacency_matrix_graph.h

6.9 Objects::Boolean< boolean > Struct Template Reference

```
#include <objects.h>
```

Static Public Attributes

- constexpr static bool value = boolean

6.9.1 Detailed Description

template<bool boolean>
struct Objects::Boolean< boolean >

Definition at line 13 of file objects.h.

6.9.2 Member Data Documentation

6.9.2.1 value

```
template<bool boolean>
constexpr static bool Objects::Boolean< boolean >::value = boolean  [static], [constexpr]
```

Definition at line 14 of file objects.h.

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

- graph/objects.h

6.10 CheckContainsConstructibleParent< type_list, T, is_parent > Struct Template Reference

`#include <contains_constructible_parent.h>`

6.10.1 Detailed Description

template<class type_list, typename T, bool is_parent>
struct CheckContainsConstructibleParent< type_list, T, is_parent >

Definition at line 18 of file contains_constructible_parent.h.

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

- TL/contains_constructible_parent.h

6.11 CheckContainsConstructibleParent< type_list, T, false > Struct Template Reference

`#include <contains_constructible_parent.h>`

Static Public Attributes

- constexpr static bool result

6.11.1 Detailed Description

template<class type_list, typename T>
struct CheckContainsConstructibleParent< type_list, T, false >

Definition at line 21 of file contains_constructible_parent.h.

6.11.2 Member Data Documentation

6.11.2.1 result

```
template<class type_list , typename T >
constexpr static bool CheckContainsConstructibleParent< type_list, T, false >::result  [static],
[constexpr]
```

Initial value:

```
= TL::ContainsConstructibleParent<
        typename type_list::Tail,
        T
    >::result
```

Definition at line 22 of file contains_constructible_parent.h.

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

- TL/contains_constructible_parent.h

6.12 CheckContainsConstructibleParent< type_list, T, true > Struct Template Reference

```
#include <contains_constructible_parent.h>
```

Static Public Attributes

- constexpr static bool result = true

6.12.1 Detailed Description

template<class type_list, typename T>
struct CheckContainsConstructibleParent< type_list, T, true >

Definition at line 29 of file contains_constructible_parent.h.

6.12.2 Member Data Documentation

6.12.2.1 result

```
template<class type_list , typename T >
constexpr static bool CheckContainsConstructibleParent< type_list, T, true >::result = true
[static], [constexpr]
```

Definition at line 30 of file contains_constructible_parent.h.

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

- TL/contains_constructible_parent.h

6.13 CheckContainsParent< type_list, T, is_parent > Struct Template Reference

`#include <contains_parent.h>`

6.13.1 Detailed Description

template<class type_list, typename T, bool is_parent>
struct CheckContainsParent< type_list, T, is_parent >

Definition at line 19 of file contains_parent.h.

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

- TL/contains_parent.h

6.14 CheckContainsParent< type_list, T, false > Struct Template Reference

`#include <contains_parent.h>`

Static Public Attributes

- constexpr static bool result

6.14.1 Detailed Description

template<class type_list, typename T>
struct CheckContainsParent< type_list, T, false >

Definition at line 22 of file contains_parent.h.

6.14.2 Member Data Documentation

6.14.2.1 result

```
template<class type_list , typename T >
constexpr static bool CheckContainsParent< type_list, T, false >::result  [static], [constexpr]
```

Initial value:

```
= TL::ContainsParent<
        typename type_list::Tail,
        T
    >::result
```

Definition at line 23 of file contains_parent.h.

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

- TL/contains_parent.h

6.15 CheckContainsParent< type_list, T, true > Struct Template Reference

`#include <contains_parent.h>`

Static Public Attributes

- constexpr static bool result = true

6.15.1 Detailed Description

template<class type_list, typename T>
struct CheckContainsParent< type_list, T, true >

Definition at line 30 of file contains_parent.h.

6.15.2 Member Data Documentation

6.15.2.1 result

```
template<class type_list , typename T >
constexpr static bool CheckContainsParent< type_list, T, true >::result = true  [static],
[constexpr]
```

Definition at line 31 of file contains_parent.h.

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

- TL/contains_parent.h

6.16 CheckFindParentTypeList< contains_class, T, type_list, type_lists > Struct Template Reference

`#include <find_parent_type_list.h>`

Public Types

- using result = NullType

6.16.1 Detailed Description

template<bool contains_class, typename T, class type_list, class ... type_lists>
struct CheckFindParentTypeList< contains_class, T, type_list, type_lists >

Definition at line 19 of file find_parent_type_list.h.

6.16.2 Member Typedef Documentation

6.16.2.1 result

```
template<bool contains_class, typename T , class type_list , class ...  type_lists>
using CheckFindParentTypeList< contains_class, T, type_list, type_lists >::result = NullType
```

Definition at line 20 of file find_parent_type_list.h.

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

- TL/find_parent_type_list.h

6.17 CheckFindParentTypeList< false, T, type_list, type_lists... > Struct Template Reference

```
#include <find_parent_type_list.h>
```

Public Types

- using result = typename TL::FindParentTypeList< T, type_lists... >::result

6.17.1 Detailed Description

template<typename T, class type_list, class ... type_lists>
struct CheckFindParentTypeList< false, T, type_list, type_lists... >

Definition at line 29 of file find_parent_type_list.h.

6.17.2 Member Typedef Documentation

6.17.2.1 result

```
template<typename T , class type_list , class ...  type_lists>
using CheckFindParentTypeList< false, T, type_list, type_lists...  >::result = typename TL::FindParentTypeLis
type_lists...>::result
```

Definition at line 30 of file find_parent_type_list.h.

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

- TL/find_parent_type_list.h

6.18 CheckFindParentTypeList< true, T, type_list, type_lists... > Struct Template Reference

```
#include <find_parent_type_list.h>
```

Public Types

- using result = type_list

6.18.1 Detailed Description

template<typename T, class type_list, class ... type_lists>
struct CheckFindParentTypeList< true, T, type_list, type_lists... >

Definition at line 24 of file find_parent_type_list.h.

6.18.2 Member Typedef Documentation

6.18.2.1 result

```
template<typename T , class type_list , class ...  type_lists>
using CheckFindParentTypeList< true, T, type_list, type_lists...  >::result = type_list
```

Definition at line 25 of file find_parent_type_list.h.

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

- TL/find_parent_type_list.h

6.19 CheckFindTypeListByClass< contains_class, T, type_list, type_lists > Struct Template Reference

`#include <find_type_list_by_class.h>`

Public Types

- using result = NullType

6.19.1 Detailed Description

template<bool contains_class, typename T, class type_list, class ... type_lists>
struct CheckFindTypeListByClass< contains_class, T, type_list, type_lists >

Definition at line 19 of file find_type_list_by_class.h.

6.19.2 Member Typedef Documentation

6.19.2.1 result

```
template<bool contains_class, typename T , class type_list , class ...  type_lists>
using CheckFindTypeListByClass< contains_class, T, type_list, type_lists >::result = NullType
```

Definition at line 20 of file find_type_list_by_class.h.

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

- TL/find_type_list_by_class.h

6.20 CheckFindTypeListByClass< false, T, type_list, type_lists... > Struct Template Reference

`#include <find_type_list_by_class.h>`

Public Types

- using result = typename TL::FindTypeListByClass< T, type_lists... >::result

6.20.1 Detailed Description

template<typename T, class type_list, class ... type_lists>
struct CheckFindTypeListByClass< false, T, type_list, type_lists... >

Definition at line 29 of file find_type_list_by_class.h.

6.20.2 Member Typedef Documentation

6.20.2.1 result

```
template<typename T , class type_list , class ...  type_lists>
using CheckFindTypeListByClass< false, T, type_list, type_lists...  >::result = typename
TL::FindTypeListByClass<T, type_lists...>::result
```

Definition at line 30 of file find_type_list_by_class.h.

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

- TL/find_type_list_by_class.h

6.21 CheckFindTypeListByClass< true, T, type_list, type_lists... > Struct Template Reference

```
#include <find_type_list_by_class.h>
```

Public Types

- using result = type_list

6.21.1 Detailed Description

template<typename T, class type_list, class ... type_lists>
struct CheckFindTypeListByClass< true, T, type_list, type_lists... >

Definition at line 24 of file find_type_list_by_class.h.

6.21.2 Member Typedef Documentation

6.21.2.1 result

```
template<typename T , class type_list , class ...  type_lists>
using CheckFindTypeListByClass< true, T, type_list, type_lists...  >::result = type_list
```

Definition at line 25 of file find_type_list_by_class.h.

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

- TL/find_type_list_by_class.h

6.22 CheckHasDerivedAndConstructible< type_list, T, is_head_parent_of_T > Struct Template Reference

```
#include <has_derived_and_constructible.h>
```

6.22.1 Detailed Description

template<class type_list, typename T, bool is_head_parent_of_T>
struct CheckHasDerivedAndConstructible< type_list, T, is_head_parent_of_T >

Definition at line 19 of file has_derived_and_constructible.h.

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

- TL/has_derived_and_constructible.h

6.23 CheckHasDerivedAndConstructible< type_list, T, false > Struct Template Reference

```
#include <has_derived_and_constructible.h>
```

Static Public Attributes

- constexpr static bool result = TL::HasDerivedAndConstructible<typename type_list::Tail, T>::result

6.23.1 Detailed Description

template<class type_list, typename T>
struct CheckHasDerivedAndConstructible< type_list, T, false >

Definition at line 27 of file has_derived_and_constructible.h.

6.23.2 Member Data Documentation

6.23.2.1 result

```
template<class type_list , typename T >
constexpr static bool CheckHasDerivedAndConstructible< type_list, T, false >::result = TL::HasDerivedAndConst
type_list::Tail, T>::result  [static], [constexpr]
```

Definition at line 28 of file has_derived_and_constructible.h.

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

- TL/has_derived_and_constructible.h

6.24 CheckHasDerivedAndConstructible< type_list, T, true > Struct Template Reference

```
#include <has_derived_and_constructible.h>
```

Static Public Attributes

- constexpr static bool result = true

6.24.1 Detailed Description

template<class type_list, typename T>
struct CheckHasDerivedAndConstructible< type_list, T, true >

Definition at line 22 of file has_derived_and_constructible.h.

6.24.2 Member Data Documentation

6.24.2.1 result

```
template<class type_list , typename T >
constexpr static bool CheckHasDerivedAndConstructible< type_list, T, true >::result = true
[static], [constexpr]
```

Definition at line 23 of file has_derived_and_constructible.h.

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

- TL/has_derived_and_constructible.h

6.25 CheckIsBaseOf< has_parent, parent, derived > Struct Template Reference

`#include <is_base_of.h>`

6.25.1 Detailed Description

template<bool has_parent, class parent, class derived>
struct CheckIsBaseOf< has_parent, parent, derived >

Definition at line 21 of file is_base_of.h.

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

- TL/is_base_of.h

6.26 CheckIsBaseOf< false, parent, derived > Struct Template Reference

`#include <is_base_of.h>`

Static Public Attributes

- constexpr static bool result = false

6.26.1 Detailed Description

template<class parent, class derived>
struct CheckIsBaseOf< false, parent, derived >

Definition at line 24 of file is_base_of.h.

6.26.2 Member Data Documentation

6.26.2.1 result

```
template<class parent , class derived >
constexpr static bool CheckIsBaseOf< false, parent, derived >::result = false  [static], [constexpr]
```

Definition at line 25 of file is_base_of.h.

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

- TL/is_base_of.h

6.27 CheckIsBaseOf< true, parent, derived > Struct Template Reference

`#include <is_base_of.h>`

Static Public Attributes

- constexpr static bool result

6.27.1 Detailed Description

template<class parent, class derived>
struct CheckIsBaseOf< true, parent, derived >

Definition at line 29 of file is_base_of.h.

6.27.2 Member Data Documentation

6.27.2.1 result

```
template<class parent , class derived >
constexpr static bool CheckIsBaseOf< true, parent, derived >::result  [static], [constexpr]
```

Initial value:

```
= TL::IsBaseOf<
        parent,
        typename derived::Tail
    >::result
```

Definition at line 30 of file is_base_of.h.

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

- TL/is_base_of.h

6.28 CheckMostDerived< type_list, T, is_head_parent_of_T > Struct Template Reference

`#include <most_derived.h>`

Public Types

- using result = NullType

6.28.1 Detailed Description

template<class type_list, typename T, bool is_head_parent_of_T>
struct CheckMostDerived< type_list, T, is_head_parent_of_T >

Definition at line 19 of file most_derived.h.

6.28.2 Member Typedef Documentation

6.28.2.1 result

```
template<class type_list , typename T , bool is_head_parent_of_T>
using CheckMostDerived< type_list, T, is_head_parent_of_T >::result = NullType
```

Definition at line 20 of file most_derived.h.

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

- TL/most_derived.h

6.29 CheckMostDerived< type_list, T, false > Struct Template Reference

```
#include <most_derived.h>
```

Public Types

- using result = typename TL::MostDerived< typename type_list::Tail, T >::result

6.29.1 Detailed Description

template<class type_list, typename T>
struct CheckMostDerived< type_list, T, false >

Definition at line 29 of file most_derived.h.

6.29.2 Member Typedef Documentation

6.29.2.1 result

```
template<class type_list , typename T >
using CheckMostDerived< type_list, T, false >::result = typename TL::MostDerived<typename
type_list::Tail, T>::result
```

Definition at line 30 of file most_derived.h.

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

- TL/most_derived.h

6.30 CheckMostDerived< type_list, T, true > Struct Template Reference

```
#include <most_derived.h>
```

Public Types

- using result = typename TL::MostDerived< typename type_list::Tail, typename type_list::Head >::result

6.30.1 Detailed Description

template<class type_list, typename T>
struct CheckMostDerived< type_list, T, true >

Definition at line 24 of file most_derived.h.

6.30.2 Member Typedef Documentation

6.30.2.1 result

```
template<class type_list , typename T >
using CheckMostDerived< type_list, T, true >::result = typename TL::MostDerived<typename
type_list::Tail, typename type_list::Head>::result
```

Definition at line 25 of file most_derived.h.

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

- TL/most_derived.h

6.31 CheckMostDerivedAndConstructible< type_list, T, is_head_parent_of_T > Struct Template Reference

`#include <most_derived_and_constructible.h>`

6.31.1 Detailed Description

template<class type_list, typename T, bool is_head_parent_of_T>
struct CheckMostDerivedAndConstructible< type_list, T, is_head_parent_of_T >

Definition at line 19 of file most_derived_and_constructible.h.

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

- TL/most_derived_and_constructible.h

6.32 CheckMostDerivedAndConstructible< type_list, T, false > Struct Template Reference

`#include <most_derived_and_constructible.h>`

Public Types

- using result = typename TL::MostDerivedAndConstructible< typename type_list::Tail, T >::result

6.32.1 Detailed Description

template<class type_list, typename T>
struct CheckMostDerivedAndConstructible< type_list, T, false >

Definition at line 27 of file most_derived_and_constructible.h.

6.32.2 Member Typedef Documentation

6.32.2.1 result

```
template<class type_list , typename T >
using CheckMostDerivedAndConstructible< type_list, T, false >::result = typename TL::MostDerivedAndConstructi
type_list::Tail, T>::result
```

Definition at line 28 of file most_derived_and_constructible.h.

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

- TL/most_derived_and_constructible.h

6.33 CheckMostDerivedAndConstructible< type_list, T, true > Struct Template Reference

`#include <most_derived_and_constructible.h>`

Public Types

- using result = typename TL::MostDerivedAndConstructible< typename type_list::Tail, typename type_list::↩ Head >::result

6.33.1 Detailed Description

template<class type_list, typename T>
struct CheckMostDerivedAndConstructible< type_list, T, true >

Definition at line 22 of file most_derived_and_constructible.h.

6.33.2 Member Typedef Documentation

6.33.2.1 result

```
template<class type_list , typename T >
using CheckMostDerivedAndConstructible< type_list, T, true >::result = typename TL::MostDerivedAndConstructib
type_list::Tail, typename type_list::Head>::result
```

Definition at line 23 of file most_derived_and_constructible.h.

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

- TL/most_derived_and_constructible.h

6.34 Class< value_ > Struct Template Reference

`#include <class.h>`

Public Types

- using value = value_

6.34.1 Detailed Description

template<typename value_>
struct Class< value_ >

Represents wrapper of a class object.

Parameters

value↩ _	Template parameter, value that should be wrapped

Definition at line 8 of file class.h.

6.34.2 Member Typedef Documentation

6.34.2.1 value

```
template<typename value_ >
using Class< value_ >::value = value_
```

Definition at line 9 of file class.h.

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

- graph/class.h

6.35 TL::Concatenate< front, back > Struct Template Reference

```
#include <concatenate.h>
```

Classes

- struct IterateThroughReversedFront
- struct IterateThroughReversedFront< EmptyTypeList, current >

Public Types

- using reversed_front = typename Reverse< front >::result
- using result = typename IterateThroughReversedFront< reversed_front, back >::result

6.35.1 Detailed Description

template<class front, class back>
struct TL::Concatenate< front, back >

Concatenates two TypeLists

Parameters

front	Template parameter
back	Template parameter

Returns

Parameter result, Concatenated TypeList

Definition at line 17 of file concatenate.h.

6.35.2 Member Typedef Documentation

6.35.2.1 result

```
template<class front , class back >
using TL::Concatenate< front, back >::result = typename IterateThroughReversedFront<reversed_front,
back>::result
```

Definition at line 42 of file concatenate.h.

6.35.2.2 reversed_front

```
template<class front , class back >
using TL::Concatenate< front, back >::reversed_front = typename Reverse<front>::result
```

Definition at line 21 of file concatenate.h.

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

- TL/concatenate.h

6.36 TL::Contains< type_list, T > Struct Template Reference

```
#include <contains.h>
```

Static Public Attributes

- constexpr static bool value = IndexOf<type_list, T>::value >= 0

6.36.1 Detailed Description

template<class type_list, typename T>
struct TL::Contains< type_list, T >

Checks if type_list contains typename T

Parameters

type_list	Template parameter
T	Template parameter

Returns

Parameter value, true if type_list contains typename T, false otherwise

Definition at line 14 of file contains.h.

6.36.2 Member Data Documentation

6.36.2.1 value

```
template<class type_list , typename T >
constexpr static bool TL::Contains< type_list, T >::value = IndexOf<type_list, T>::value >=
0  [static], [constexpr]
```

Definition at line 15 of file contains.h.

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

- TL/contains.h

6.37 TL::ContainsConstructibleParent< type_list, T > Struct Template Reference

```
#include <contains_constructible_parent.h>
```

Static Public Attributes

- constexpr static bool result

6.37.1 Detailed Description

template<class type_list, typename T>
struct TL::ContainsConstructibleParent< type_list, T >

Checks if type_list contains constructible parent of T

Parameters

type_list	Template parameter
T	Template parameter

Returns

Parameter result, true if type_list contains constructible parent of T, false otherwise

Definition at line 35 of file contains_constructible_parent.h.

6.37.2 Member Data Documentation

6.37.2.1 result

```
template<class type_list , typename T >
constexpr static bool TL::ContainsConstructibleParent< type_list, T >::result  [static], [constexpr]
```

Initial value:

```
= CheckContainsConstructibleParent<
            type_list,
            T,
            std::is_base_of<typename type_list::Head, T>::value &&
            std::is_constructible<typename type_list::Head>::value
        >::result
```

Definition at line 36 of file contains_constructible_parent.h.

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

- TL/contains_constructible_parent.h

6.38 TL::ContainsConstructibleParent< EmptyTypeList, T > Struct Template Reference

```
#include <contains_constructible_parent.h>
```

Static Public Attributes

- constexpr static bool result = false

6.38.1 Detailed Description

template<typename T>
struct TL::ContainsConstructibleParent< EmptyTypeList, T >

Definition at line 45 of file contains_constructible_parent.h.

6.38.2 Member Data Documentation

6.38.2.1 result

```
template<typename T >
constexpr static bool TL::ContainsConstructibleParent< EmptyTypeList, T >::result = false
[static], [constexpr]
```

Definition at line 46 of file contains_constructible_parent.h.

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

- TL/contains_constructible_parent.h

6.39 TL::ContainsParent< type_list, T > Struct Template Reference

```
#include <contains_parent.h>
```

Static Public Attributes

- constexpr static bool result

6.39.1 Detailed Description

template<class type_list, typename T>
struct TL::ContainsParent< type_list, T >

Checks if type_list contains parent of T

Parameters

type_list	Template parameter
T	Template parameter

Returns

Parameter result, true if type_list contains parent of T, false otherwise

Definition at line 36 of file contains_parent.h.

6.39.2 Member Data Documentation

6.39.2.1 result

```
template<class type_list , typename T >
constexpr static bool TL::ContainsParent< type_list, T >::result  [static], [constexpr]
```

Initial value:

```
= CheckContainsParent<
            type_list,
            T,
            std::is_base_of<typename type_list::Head, T>::value
        >::result
```

Definition at line 37 of file contains_parent.h.

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

- TL/contains_parent.h

6.40 TL::ContainsParent< EmptyTypeList, T > Struct Template Reference

```
#include <contains_parent.h>
```

Static Public Attributes

- constexpr static bool result = false

6.40.1 Detailed Description

template<typename T>
struct TL::ContainsParent< EmptyTypeList, T >

Definition at line 45 of file contains_parent.h.

6.40.2 Member Data Documentation

6.40.2.1 result

```
template<typename T >
constexpr static bool TL::ContainsParent< EmptyTypeList, T >::result = false  [static], [constexpr]
```

Definition at line 46 of file contains_parent.h.

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

- TL/contains_parent.h

6.41 ConvertGraph< From, To, graph > Struct Template Reference

6.41.1 Detailed Description

template<GraphType From, GraphType To, class graph>
struct ConvertGraph< From, To, graph >

An adapter to convert graph of type From to type To. Defined separate from its realizations in order to avoid cycle dependency. Before convertion, don't forget to include file of a required implementation. Is used as a visitor in Visitor pattern.

See also

GraphType

Parameters

From	Template parameter
To	Template parameter
graph	Template parameter

Returns

Parameter result, resulting graph.

Definition at line 17 of file convert_graph.h.

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

- graph/convert/convert_graph.h

6.42 ConvertGraph< ADJACENCY_LIST, POINTER_STRUCTURE, graph > Struct Template Reference

`#include <convert_to_pointer_structure.h>`

Classes

- struct MakePointerStructureGraph
- struct MakePointerStructureGraph< EmptyTypeList, EmptyTypeList >

Public Types

- using result = PointerStructureGraph< typename MakePointerStructureGraph< typename graph::vertexes↩ _, typename graph::adjacency_list_ >::result >

6.42.1 Detailed Description

template<class graph>
struct ConvertGraph< ADJACENCY_LIST, POINTER_STRUCTURE, graph >

See also

ConvertGraph

Definition at line 24 of file convert_to_pointer_structure.h.

6.42.2 Member Typedef Documentation

6.42.2.1 result

```
template<class graph >
using ConvertGraph< ADJACENCY_LIST, POINTER_STRUCTURE, graph >::result = PointerStructureGraph<
typename MakePointerStructureGraph< typename graph::vertexes_, typename graph::adjacency_↩
list_ >::result >
```

Definition at line 49 of file convert_to_pointer_structure.h.

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

- graph/convert/convert_to_pointer_structure.h

6.43 ConvertGraph< ADJACENCY_MATRIX, EDGE_LIST, graph > Struct Template Reference

```
#include <convert_to_edge_list.h>
```

Classes

- struct IterateThroughMatrix
- struct IterateThroughMatrix<-1 >

Public Types

- using vertexes = typename graph::vertexes_
- using edges = typename IterateThroughMatrix< n *n - 1 >::result
- using result = EdgeListGraph< vertexes, edges >

Static Public Attributes

- constexpr static int n = TL::Size<vertexes>::size

6.43.1 Detailed Description

template<class graph>
struct ConvertGraph< ADJACENCY_MATRIX, EDGE_LIST, graph >

See also

ConvertGraph

Definition at line 64 of file convert_to_edge_list.h.

6.43.2 Member Typedef Documentation

6.43.2.1 edges

```
template<class graph >
using ConvertGraph< ADJACENCY_MATRIX, EDGE_LIST, graph >::edges = typename IterateThrough↩
Matrix<n * n - 1>::result
```

Definition at line 105 of file convert_to_edge_list.h.

6.43.2.2 result

```
template<class graph >
using ConvertGraph< ADJACENCY_MATRIX, EDGE_LIST, graph >::result = EdgeListGraph<vertexes,
edges>
```

Definition at line 106 of file convert_to_edge_list.h.

6.43.2.3 vertexes

```
template<class graph >
using ConvertGraph< ADJACENCY_MATRIX, EDGE_LIST, graph >::vertexes = typename graph::vertexes↩
_
```

Definition at line 67 of file convert_to_edge_list.h.

6.43.3 Member Data Documentation

6.43.3.1 n

```
template<class graph >
constexpr static int ConvertGraph< ADJACENCY_MATRIX, EDGE_LIST, graph >::n = TL::Size<vertexes>↩
::size  [static], [constexpr]
```

Definition at line 68 of file convert_to_edge_list.h.

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

- graph/convert/convert_to_edge_list.h

6.44 ConvertGraph< EDGE_LIST, ADJACENCY_LIST, graph > Struct Template Reference

```
#include <convert_to_adjacency_list.h>
```

Classes

- struct IterateThroughEdges
- struct IterateThroughEdges< EmptyTypeList >

Public Types

- using result = typename IterateThroughEdges< typename graph::edge_list_ >::result

6.44.1 Detailed Description

template<class graph>
struct ConvertGraph< EDGE_LIST, ADJACENCY_LIST, graph >

See also

ConvertGraph

Definition at line 21 of file convert_to_adjacency_list.h.

6.44.2 Member Typedef Documentation

6.44.2.1 result

```
template<class graph >
using ConvertGraph< EDGE_LIST, ADJACENCY_LIST, graph >::result = typename IterateThrough↩
Edges<typename graph::edge_list_>::result
```

Definition at line 44 of file convert_to_adjacency_list.h.

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

- graph/convert/convert_to_adjacency_list.h

6.45 ConvertGraph< EDGE_LIST, ADJACENCY_MATRIX, graph > Struct Template Reference

```
#include <convert_to_adjacency_matrix.h>
```

Classes

- struct IterateThroughEdges
- struct IterateThroughEdges< EmptyTypeList >

Public Types

- using vertexes = typename graph::vertexes_
- using matrix = typename IterateThroughEdges< typename graph::edge_list_ >::result
- using result = AdjacencyMatrixGraph< vertexes, matrix >

6.45.1 Detailed Description

template<class graph>
struct ConvertGraph< EDGE_LIST, ADJACENCY_MATRIX, graph >

See also

ConvertGraph

Definition at line 14 of file convert_to_adjacency_matrix.h.

6.45.2 Member Typedef Documentation

6.45.2.1 matrix

```
template<class graph >
using ConvertGraph< EDGE_LIST, ADJACENCY_MATRIX, graph >::matrix = typename IterateThrough↩
Edges<typename graph::edge_list_>::result
```

Definition at line 67 of file convert_to_adjacency_matrix.h.

6.45.2.2 result

```
template<class graph >
using ConvertGraph< EDGE_LIST, ADJACENCY_MATRIX, graph >::result = AdjacencyMatrixGraph<vertexes,
matrix>
```

Definition at line 68 of file convert_to_adjacency_matrix.h.

6.45.2.3 vertexes

```
template<class graph >
using ConvertGraph< EDGE_LIST, ADJACENCY_MATRIX, graph >::vertexes = typename graph::vertexes↩
_
```

Definition at line 16 of file convert_to_adjacency_matrix.h.

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

- graph/convert/convert_to_adjacency_matrix.h

6.46 ConvertGraph< EDGE_LIST, POINTER_STRUCTURE, graph > Struct Template Reference

```
#include <convert_to_pointer_structure.h>
```

Public Types

- using adjacency_list = typename ConvertGraph< EDGE_LIST, ADJACENCY_LIST, graph >::result
- using result = typename ConvertGraph< ADJACENCY_LIST, POINTER_STRUCTURE, adjacency_list >↩ ::result

6.46.1 Detailed Description

template<class graph>
struct ConvertGraph< EDGE_LIST, POINTER_STRUCTURE, graph >

See also

ConvertGraph

Definition at line 61 of file convert_to_pointer_structure.h.

6.46.2 Member Typedef Documentation

6.46.2.1 adjacency_list

```
template<class graph >
using ConvertGraph< EDGE_LIST, POINTER_STRUCTURE, graph >::adjacency_list = typename ConvertGraph<EDGE_LIST,
ADJACENCY_LIST, graph>::result
```

Definition at line 63 of file convert_to_pointer_structure.h.

6.46.2.2 result

```
template<class graph >
using ConvertGraph< EDGE_LIST, POINTER_STRUCTURE, graph >::result = typename ConvertGraph<ADJACENCY_LIST,
POINTER_STRUCTURE, adjacency_list>::result
```

Definition at line 64 of file convert_to_pointer_structure.h.

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

- graph/convert/convert_to_pointer_structure.h

6.47 ConvertGraph< POINTER_STRUCTURE, EDGE_LIST, graph > Struct Template Reference

```
#include <convert_to_edge_list.h>
```

Classes

- struct IterateThroughNodes
- struct IterateThroughNodes< EmptyTypeList >

Public Types

- using iterate_result = IterateThroughNodes< typename graph::nodes_ >
- using result = EdgeListGraph< typename iterate_result::vertexes, typename iterate_result::edges >

6.47.1 Detailed Description

template<class graph>
struct ConvertGraph< POINTER_STRUCTURE, EDGE_LIST, graph >

See also

ConvertGraph

Definition at line 25 of file convert_to_edge_list.h.

6.47.2 Member Typedef Documentation

6.47.2.1 iterate_result

```
template<class graph >
using ConvertGraph< POINTER_STRUCTURE, EDGE_LIST, graph >::iterate_result = IterateThrough↩
Nodes<typename graph::nodes_>
```

Definition at line 52 of file convert_to_edge_list.h.

6.47.2.2 result

```
template<class graph >
using ConvertGraph< POINTER_STRUCTURE, EDGE_LIST, graph >::result = EdgeListGraph< typename
iterate_result::vertexes, typename iterate_result::edges >
```

Definition at line 54 of file convert_to_edge_list.h.

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

- graph/convert/convert_to_edge_list.h

6.48 ConvertGraph< type, type, graph > Struct Template Reference

```
#include <convert_graph.h>
```

Public Types

- using result = graph

6.48.1 Detailed Description

template<GraphType type, class graph>
struct ConvertGraph< type, type, graph >

Definition at line 20 of file convert_graph.h.

6.48.2 Member Typedef Documentation

6.48.2.1 result

```
template<GraphType type, class graph >
using ConvertGraph< type, type, graph >::result = graph
```

Definition at line 22 of file convert_graph.h.

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

- graph/convert/convert_graph.h

6.49 AdjacencyListGraph< nodes, adjacency_list >::ConvertTo< type > Struct Template Reference

```
#include <adjacency_list_graph.h>
```

Public Types

- using result = typename ConvertGraph< TYPE, type, AdjacencyListGraph< vertexes_, adjacency_list_ > >::result

6.49.1 Detailed Description

template<class nodes, class adjacency_list>
template<GraphType type>
struct AdjacencyListGraph< nodes, adjacency_list >::ConvertTo< type >

Represents an adapter, which converts one type of a graph into another. Is used as an element in Visitor pattern.

Parameters

GraphType	Template parameter, type of a resulting graph

Returns

Parameter result, resulting graph

Definition at line 63 of file adjacency_list_graph.h.

6.49.2 Member Typedef Documentation

6.49.2.1 result

```
template<class nodes , class adjacency_list >
template<GraphType type>
using AdjacencyListGraph< nodes, adjacency_list >::ConvertTo< type >::result = typename ConvertGraph<
TYPE, type, AdjacencyListGraph<vertexes_, adjacency_list_> >::result
```

Definition at line 64 of file adjacency_list_graph.h.

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

- graph/graphs/adjacency_list_graph.h

6.50 AdjacencyMatrixGraph< vertexes, matrix >::ConvertTo< type > Struct Template Reference

```
#include <adjacency_matrix_graph.h>
```

Public Types

- using result = typename ConvertGraph< TYPE, type, AdjacencyMatrixGraph< vertexes, matrix > >::result

6.50.1 Detailed Description

template<class vertexes, class matrix>
template<GraphType type>
struct AdjacencyMatrixGraph< vertexes, matrix >::ConvertTo< type >

Represents an adapter, which converts one type of a graph into another. Is used as an element in Visitor pattern.

Parameters

GraphType	Template parameter, type of a resulting graph

Returns

Parameter result, resulting graph

Definition at line 39 of file adjacency_matrix_graph.h.

6.50.2 Member Typedef Documentation

6.50.2.1 result

```
template<class vertexes , class matrix >
template<GraphType type>
using AdjacencyMatrixGraph< vertexes, matrix >::ConvertTo< type >::result = typename ConvertGraph<
TYPE, type, AdjacencyMatrixGraph<vertexes, matrix> >::result
```

Definition at line 40 of file adjacency_matrix_graph.h.

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

- graph/graphs/adjacency_matrix_graph.h

6.51 EdgeListGraph< nodes, edge_list >::ConvertTo< type > Struct Template Reference

```
#include <edge_list_graph.h>
```

Public Types

- using result = typename ConvertGraph< TYPE, type, EdgeListGraph< vertexes_, edge_list_ > >::result

6.51.1 Detailed Description

template<class nodes, class edge_list>
template<GraphType type>
struct EdgeListGraph< nodes, edge_list >::ConvertTo< type >

Represents an adapter, which converts one type of a graph into another. Is used as an element in Visitor pattern.

Parameters

GraphType	Template parameter, type of a resulting graph

Returns

Parameter result, resulting graph

Definition at line 36 of file edge_list_graph.h.

6.51.2 Member Typedef Documentation

6.51.2.1 result

```
template<class nodes , class edge_list >
template<GraphType type>
using EdgeListGraph< nodes, edge_list >::ConvertTo< type >::result = typename ConvertGraph<
TYPE, type, EdgeListGraph<vertexes_, edge_list_> >::result
```

Definition at line 37 of file edge_list_graph.h.

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

- graph/graphs/edge_list_graph.h

6.52 PointerStructureGraph< nodes >::ConvertTo< type > Struct Template Reference

```
#include <pointer_structure_graph.h>
```

Public Types

- using result = typename ConvertGraph< TYPE, type, PointerStructureGraph< nodes > >::result

6.52.1 Detailed Description

template<class nodes>
template<GraphType type>
struct PointerStructureGraph< nodes >::ConvertTo< type >

Represents an adapter, which converts one type of a graph into another. Is used as an element in Visitor pattern.

Parameters

GraphType	Template parameter, type of a resulting graph

Returns

Parameter result, resulting graph

Definition at line 36 of file pointer_structure_graph.h.

6.52.2 Member Typedef Documentation

6.52.2.1 result

```
template<class nodes >
template<GraphType type>
using PointerStructureGraph< nodes >::ConvertTo< type >::result = typename ConvertGraph<
TYPE, type, PointerStructureGraph<nodes> >::result
```

Definition at line 37 of file pointer_structure_graph.h.

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

- graph/graphs/pointer_structure_graph.h

6.53 GLib::DFS< cur_node, graph, visited_nodes > Struct Template Reference

```
#include <dfs.h>
```

Classes

- struct IterateThroughChildren
- struct IterateThroughChildren< EmptyTypeList, cur_unvisited >

Public Types

- using upd_visited = typename TL::Add< cur_node, 0, visited_nodes >::result
- using iterate_through_children = IterateThroughChildren< typename cur_node::children, upd_visited >
- using new_visited = typename iterate_through_children::new_visited
- using result = typename iterate_through_children::result

6.53.1 Detailed Description

template<class cur_node, class graph, class visited_nodes = EmptyTypeList>
struct GLib::DFS< cur_node, graph, visited_nodes >

Performs Depth-First Search, starting from passed vertex. It doesn't visit vertexes that have been visited already. It returns visited edges in chronological order, from which it's easy to deduce DFS. It's more versatile than one may think) Also a variation of Composite pattern.

Parameters

cur_nod	Template parameter, starting node in DFS.
graph	Graph, where DFS should be performed.
visited_nodes	Optional template parameter, nodes that are not allowed to be visited.

Returns

Parameter result, TypeList of visited edges in chronological order. Also returns parameter new_visited as a side effect, which is a TypeList of visited nodes.

Definition at line 22 of file dfs.h.

6.53.2 Member Typedef Documentation

6.53.2.1 iterate_through_children

```
template<class cur_node , class graph , class visited_nodes = EmptyTypeList>
using GLib::DFS< cur_node, graph, visited_nodes >::iterate_through_children = IterateThroughChildren<
typename cur_node::children, upd_visited >
```

Definition at line 67 of file dfs.h.

6.53.2.2 new_visited

```
template<class cur_node , class graph , class visited_nodes = EmptyTypeList>
using GLib::DFS< cur_node, graph, visited_nodes >::new_visited = typename iterate_through_children::new_visit
```

Definition at line 71 of file dfs.h.

6.53.2.3 result

```
template<class cur_node , class graph , class visited_nodes = EmptyTypeList>
using GLib::DFS< cur_node, graph, visited_nodes >::result = typename iterate_through_children::result
```

Definition at line 72 of file dfs.h.

6.53.2.4 upd_visited

```
template<class cur_node , class graph , class visited_nodes = EmptyTypeList>
using GLib::DFS< cur_node, graph, visited_nodes >::upd_visited = typename TL::Add<cur_node,
0, visited_nodes>::result
```

Definition at line 25 of file dfs.h.

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

- graph/GLib/dfs.h

6.54 Edge< from_, to_, weight_ > Struct Template Reference

```
#include <edge.h>
```

Public Types

- using from = from_
  Starting vertex of an edge.
- using to = to_
  Ending vertex of an edge.
- using weight = weight_
  Additional property of an edge.

6.54.1 Detailed Description

template<typename from_, typename to_, typename weight_ = NullType>
struct Edge< from_, to_, weight_ >

Represents an edge in the graph.

Parameters

*from_*	Template parameter, starting vertex of an edge
*to_*	Template parameter, ending vertex of an edge
*weight_*	Template parameter, additional property of an edge

Definition at line 12 of file edge.h.

6.54.2 Member Typedef Documentation

6.54.2.1 from

```
template<typename from_ , typename to_ , typename weight_ = NullType>
using Edge< from_, to_, weight_ >::from = from_
```

Starting vertex of an edge.

Definition at line 13 of file edge.h.

6.54.2.2 to

```
template<typename from_ , typename to_ , typename weight_ = NullType>
using Edge< from_, to_, weight_ >::to = to_
```

Ending vertex of an edge.

Definition at line 14 of file edge.h.

6.54.2.3 weight

```
template<typename from_ , typename to_ , typename weight_ = NullType>
using Edge< from_, to_, weight_ >::weight = weight_
```

Additional property of an edge.

Definition at line 15 of file edge.h.

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

- graph/edge.h

6.55 EdgeListGraph< nodes, edge_list > Struct Template Reference

```
#include <edge_list_graph.h>
```

Classes

- struct ConvertTo

Public Types

- using vertexes_ = nodes

  TypeList of vertexes in graph.
- using edge_list_ = edge_list

  TypeList of edges.
- using edges_ = edge_list_

Static Public Attributes

- constexpr static GraphType TYPE = EDGE_LIST

6.55.1 Detailed Description

template<class nodes, class edge_list>
struct EdgeListGraph< nodes, edge_list >

Represents graph as a list of edges.

See also

Graph

Edge

Parameters

vertexes	Template parameter, vertexes of a graph
edge_list	Template parameter, TypeList of Edge

Returns

Parameter result, resulting graph

Definition at line 20 of file edge_list_graph.h.

6.55.2 Member Typedef Documentation

6.55.2.1 edge_list_

```
template<class nodes , class edge_list >
using EdgeListGraph< nodes, edge_list >::edge_list_ = edge_list
```

TypeList of edges.

Definition at line 26 of file edge_list_graph.h.

6.55.2.2 edges_

```
template<class nodes , class edge_list >
using EdgeListGraph< nodes, edge_list >::edges_ = edge_list_
```

Definition at line 27 of file edge_list_graph.h.

6.55.2.3 vertexes_

```
template<class nodes , class edge_list >
using EdgeListGraph< nodes, edge_list >::vertexes_ = nodes
```

TypeList of vertexes in graph.

Definition at line 25 of file edge_list_graph.h.

6.55.3 Member Data Documentation

6.55.3.1 TYPE

```
template<class nodes , class edge_list >
constexpr static GraphType EdgeListGraph< nodes, edge_list >::TYPE = EDGE_LIST  [static],
[constexpr]
```

Definition at line 21 of file edge_list_graph.h.

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

- graph/graphs/edge_list_graph.h

6.56 TL::FillTypeListWithObject< obj, n > Struct Template Reference

```
#include <fill_type_list_with_object.h>
```

Public Types

- using result = typename Add< obj, 0, typename FillTypeListWithObject< obj, n - 1 >::result >::result

6.56.1 Detailed Description

template<typename obj, int n>
struct TL::FillTypeListWithObject< obj, n >

Generates TypeList of n objects of type obj.

See also

EmptyTypeList

Parameters

obj	Template parameter, an object to fill TypeList with.
n	Template parameter, a number of EmptyTypeLists to generate.

Returns

Parameter result, TypeList of n EmptyTypeList.

Definition at line 15 of file fill_type_list_with_object.h.

6.56.2 Member Typedef Documentation

6.56.2.1 result

```
template<typename obj , int n>
using TL::FillTypeListWithObject< obj, n >::result = typename Add< obj, 0, typename FillTypeListWithObject<o
n - 1>::result >::result
```

Definition at line 16 of file fill_type_list_with_object.h.

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

- TL/fill_type_list_with_object.h

6.57 TL::FillTypeListWithObject< obj, 0 > Struct Template Reference

```
#include <fill_type_list_with_object.h>
```

Public Types

- using result = EmptyTypeList

6.57.1 Detailed Description

template<typename obj>
struct TL::FillTypeListWithObject< obj, 0 >

Definition at line 24 of file fill_type_list_with_object.h.

6.57.2 Member Typedef Documentation

6.57.2.1 result

```
template<typename obj >
using TL::FillTypeListWithObject< obj, 0 >::result = EmptyTypeList
```

Definition at line 25 of file fill_type_list_with_object.h.

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

- TL/fill_type_list_with_object.h

6.58 GLib::Filter< id, graph, vertexes > Struct Template Reference

```
#include <filter.h>
```

Public Types

- using tail_result = typename Filter< id, graph, typename vertexes::Tail >::result
- using result = std::conditional_t< ProcessVertex< id, graph, typename vertexes::Head >::result, typename TL::Add< typename vertexes::Head, 0, tail_result >::result, tail_result >

6.58.1 Detailed Description

template<int id, class graph, class vertexes = typename graph::vertexes_>
struct GLib::Filter< id, graph, vertexes >

Leaves only vertexes that return true when called with ProcessVertex. ProcessVertex must have constexpr static parameter called "result" that can be casted to bool.

See also

ProcessVertex

Parameters

id	Template parameter, integer, which ForEach uses to access specific ProcessVertex implementation.
graph	Template parameter, graph, where vertexes are located.
vertexes	Optional template parameter, vertexes to process. Is equal to graph's vertexes by default.

Returns

Parameter result, TypeList of vertexes which return true on ProcessVertex.

Definition at line 20 of file filter.h.

6.58.2 Member Typedef Documentation

6.58.2.1 result

```
template<int id, class graph , class vertexes = typename graph::vertexes_>
using GLib::Filter< id, graph, vertexes >::result = std::conditional_t< ProcessVertex<id,
graph, typename vertexes::Head>::result, typename TL::Add<typename vertexes::Head, 0, tail_result>↩
::result, tail_result >
```

Definition at line 23 of file filter.h.

6.58.2.2 tail_result

```
template<int id, class graph , class vertexes = typename graph::vertexes_>
using GLib::Filter< id, graph, vertexes >::tail_result = typename Filter<id, graph, typename
vertexes::Tail>::result
```

Definition at line 21 of file filter.h.

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

- graph/GLib/filter.h

6.59 GLib::Filter< id, graph, EmptyTypeList > Struct Template Reference

```
#include <filter.h>
```

Public Types

- using result = EmptyTypeList

6.59.1 Detailed Description

template<int id, class graph>
struct GLib::Filter< id, graph, EmptyTypeList >

See also

Filter

Definition at line 34 of file filter.h.

6.59.2 Member Typedef Documentation

6.59.2.1 result

```
template<int id, class graph >
using GLib::Filter< id, graph, EmptyTypeList >::result = EmptyTypeList
```

Definition at line 35 of file filter.h.

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

- graph/GLib/filter.h

6.60 GLib::FindNodeByVertex< vertex, graph > Struct Template Reference

```
#include <find_node_by_vertex.h>
```

Classes

- struct IterateThroughNodes
- struct IterateThroughNodes< EmptyTypeList >

Public Types

- using result = typename IterateThroughNodes< typename graph::nodes_ >::result

6.60.1 Detailed Description

template<typename vertex, class graph>
struct GLib::FindNodeByVertex< vertex, graph >

Finds node corresponding to this vertex.

Parameters

vertex	Template parameter, vertex, node of which to find.
graph	Template parameter, graph that should be passed.

Returns

Parameter result, required node if found, NullType otherwise.

Definition at line 13 of file find_node_by_vertex.h.

6.60.2 Member Typedef Documentation

6.60.2.1 result

```
template<typename vertex , class graph >
using GLib::FindNodeByVertex< vertex, graph >::result = typename IterateThroughNodes<typename
graph::nodes_>::result
```

Definition at line 31 of file find_node_by_vertex.h.

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

- graph/GLib/find_node_by_vertex.h

6.61 GLib::FindNodeByVertex< vertex, EmptyTypeList > Struct Template Reference

```
#include <find_node_by_vertex.h>
```

Public Types

- using result = NullType

6.61.1 Detailed Description

template<typename vertex>
struct GLib::FindNodeByVertex< vertex, EmptyTypeList >

Definition at line 35 of file find_node_by_vertex.h.

6.61.2 Member Typedef Documentation

6.61.2.1 result

```
template<typename vertex >
using GLib::FindNodeByVertex< vertex, EmptyTypeList >::result = NullType
```

Definition at line 36 of file find_node_by_vertex.h.

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

- graph/GLib/find_node_by_vertex.h

6.62 TL::FindParentTypeList< T, type_list, type_lists > Struct Template Reference

`#include <find_parent_type_list.h>`

Public Types

- using result = typename CheckFindParentTypeList< TL::IsBaseOf< type_list, T >::result, T, type_list, type_lists... >::result

6.62.1 Detailed Description

template<typename T, class type_list, class ... type_lists>
struct TL::FindParentTypeList< T, type_list, type_lists >

Finds and returns TypeList that has the parent of T

Parameters

T	
type_list	First TypeList among other TypeLists
...type_lists	Other TypeLists to check

Returns

Parameter result, first TypeList that contains the parent of T, compilation error otherwise

Definition at line 35 of file find_parent_type_list.h.

6.62.2 Member Typedef Documentation

6.62.2.1 result

```
template<typename T , class type_list , class ...  type_lists>
using TL::FindParentTypeList< T, type_list, type_lists >::result = typename CheckFindParentTypeList<
TL::IsBaseOf<type_list, T>::result, T, type_list, type_lists...  >::result
```

Definition at line 36 of file find_parent_type_list.h.

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

- TL/find_parent_type_list.h

6.63 GLib::FindPath< graph_raw, start, finish > Struct Template Reference

`#include <find_path.h>`

Classes

- struct IterateThroughEdges
- struct IterateThroughEdges< EmptyTypeList, wanted_node >

Public Types

- using graph = typename ConvertGraph< graph_raw::TYPE, POINTER_STRUCTURE, graph_raw >::result
- using start_node = typename FindNodeByVertex< start, graph >::result
- using finish_node = typename FindNodeByVertex< finish, graph >::result
- using dfs_search = typename DFS< start_node, graph >::result
- using reversed = typename TL::Reverse< dfs_search >::result
- using iterate_through_edges = IterateThroughEdges< reversed, finish_node >
- using reversed_path = typename iterate_through_edges::path
- using reversed_weights = typename iterate_through_edges::weights
- using path = typename TL::Add< start, 0, typename TL::Reverse< reversed_path >::result >::result
- using weights = typename TL::Reverse< reversed_weights >::result

6.63.1 Detailed Description

template<class graph_raw, typename start, typename finish>
struct GLib::FindPath< graph_raw, start, finish >

Finds path in graph between vertexes start and finish.

See also

DFS

Parameters

graph	Template parameter
start	Template parameter
finish	Template parameter

Returns

Two parameters: path and weights. "path" is a TypeList of vertexes that make this path. "weights" is a TypeList of weights, that were on the edges in this path. If there's no path, path and weights are EmptyTypeList.

Definition at line 26 of file find_path.h.

6.63.2 Member Typedef Documentation

6.63.2.1 dfs_search

```
template<class graph_raw , typename start , typename finish >
using GLib::FindPath< graph_raw, start, finish >::dfs_search = typename DFS<start_node, graph>↩
::result
```

Definition at line 32 of file find_path.h.

6.63.2.2 finish_node

```
template<class graph_raw , typename start , typename finish >
using GLib::FindPath< graph_raw, start, finish >::finish_node = typename FindNodeByVertex<finish,
graph>::result
```

Definition at line 30 of file find_path.h.

6.63.2.3 graph

```
template<class graph_raw , typename start , typename finish >
using GLib::FindPath< graph_raw, start, finish >::graph = typename ConvertGraph<graph_raw::↩
TYPE, POINTER_STRUCTURE, graph_raw>::result
```

Definition at line 27 of file find_path.h.

6.63.2.4 iterate_through_edges

```
template<class graph_raw , typename start , typename finish >
using GLib::FindPath< graph_raw, start, finish >::iterate_through_edges = IterateThroughEdges<reversed,
finish_node>
```

Definition at line 78 of file find_path.h.

6.63.2.5 path

```
template<class graph_raw , typename start , typename finish >
using GLib::FindPath< graph_raw, start, finish >::path = typename TL::Add< start, 0, typename
TL::Reverse<reversed_path>::result >::result
```

Definition at line 82 of file find_path.h.

6.63.2.6 reversed

```
template<class graph_raw , typename start , typename finish >
using GLib::FindPath< graph_raw, start, finish >::reversed = typename TL::Reverse<dfs_search>↩
::result
```

Definition at line 33 of file find_path.h.

6.63.2.7 reversed_path

```
template<class graph_raw , typename start , typename finish >
using GLib::FindPath< graph_raw, start, finish >::reversed_path = typename iterate_through_edges::path
```

Definition at line 79 of file find_path.h.

6.63.2.8 reversed_weights

```
template<class graph_raw , typename start , typename finish >
using GLib::FindPath< graph_raw, start, finish >::reversed_weights = typename iterate_through_edges::weights
```

Definition at line 80 of file find_path.h.

6.63.2.9 start_node

```
template<class graph_raw , typename start , typename finish >
using GLib::FindPath< graph_raw, start, finish >::start_node = typename FindNodeByVertex<start,
graph>::result
```

Definition at line 29 of file find_path.h.

6.63.2.10 weights

```
template<class graph_raw , typename start , typename finish >
using GLib::FindPath< graph_raw, start, finish >::weights = typename TL::Reverse<reversed_weights>↩
::result
```

Definition at line 87 of file find_path.h.

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

- graph/GLib/find_path.h

6.64 TL::FindTypeListByClass< T, type_list, type_lists > Struct Template Reference

`#include <find_type_list_by_class.h>`

Public Types

- using result = typename CheckFindTypeListByClass< TL::Contains< type_list, T >::value, T, type_list, type_lists... >::result

6.64.1 Detailed Description

template<typename T, class type_list, class ... type_lists>
struct TL::FindTypeListByClass< T, type_list, type_lists >

Finds and returns TypeList that has T

Parameters

T	Template parameter
type_list	Template parameter, first TypeList among other TypeLists
...type_lists	Template parameter, other TypeLists to check

Returns

Parameter result, first TypeList that contains T, compilation error otherwise

Definition at line 35 of file find_type_list_by_class.h.

6.64.2 Member Typedef Documentation

6.64.2.1 result

```
template<typename T , class type_list , class ...  type_lists>
using TL::FindTypeListByClass< T, type_list, type_lists >::result = typename CheckFindTypeListByClass<
TL::Contains<type_list, T>::value, T, type_list, type_lists...  >::result
```

Definition at line 36 of file find_type_list_by_class.h.

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

- TL/find_type_list_by_class.h

6.65 GLib::ForEach< id, graph, vertexes > Struct Template Reference

`#include <for_each.h>`

Public Types

- using result = typename TL::Add< ProcessVertex< id, graph, typename vertexes::Head >, 0, typename ForEach< id, graph, typename vertexes::Tail >::result >::result

6.65.1 Detailed Description

template<int id, class graph, class vertexes = typename graph::vertexes_>
struct GLib::ForEach< id, graph, vertexes >

Processes each vertex by using ProcessVertex.

See also

ProcessVertex

Parameters

id	Template parameter, integer, which ForEach uses to access specific ProcessVertex implementation.
graph	Template parameter, graph, where vertexes are located.
vertexes	Optional template parameter, vertexes to process. Is equal to graph's vertexes by default.

Returns

Parameter result, TypeList of ProcessVertex, applied to each vertex.

Definition at line 17 of file for_each.h.

6.65.2 Member Typedef Documentation

6.65.2.1 result

```
template<int id, class graph , class vertexes = typename graph::vertexes_>
using GLib::ForEach< id, graph, vertexes >::result = typename TL::Add< ProcessVertex<id,
graph, typename vertexes::Head>, 0, typename ForEach<id, graph, typename vertexes::Tail>↩
::result >::result
```

Definition at line 18 of file for_each.h.

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

- graph/GLib/for_each.h

6.66 GLib::ForEach< id, graph, EmptyTypeList > Struct Template Reference

`#include <for_each.h>`

Public Types

- using result = EmptyTypeList

6.66.1 Detailed Description

template<int id, class graph>
struct GLib::ForEach< id, graph, EmptyTypeList >

See also

ForEach

Definition at line 29 of file for_each.h.

6.66.2 Member Typedef Documentation

6.66.2.1 result

```
template<int id, class graph >
using GLib::ForEach< id, graph, EmptyTypeList >::result = EmptyTypeList
```

Definition at line 30 of file for_each.h.

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

- graph/GLib/for_each.h

6.67 Functor< ResultType, ArgTypes > Class Template Reference

6.67.1 Detailed Description

template<typename ResultType, typename ... ArgTypes>
class Functor< ResultType, ArgTypes >

Definition at line 7 of file functor.h.

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

- functor.h

6.68 Functor< ResultType(ArgTypes...)> Class Template Reference

`#include <functor.h>`

Public Member Functions

- Functor ()=default
- template<typename Function >
  Functor (Function function)
- template<typename Function , class Class >
  Functor (Function Class::*function)
- Functor (const Functor &other)
- Functor & operator= (const Functor &other)
- ResultType operator() (ArgTypes... args)

6.68.1 Detailed Description

template<typename ResultType, typename ... ArgTypes>
class Functor< ResultType(ArgTypes...)>

Provides an object that contains a function

Parameters

ResultType	Template parameter, type of an object function returns
ArgTypes	Template parameters, types of an object function accepts

Definition at line 15 of file functor.h.

6.68.2 Constructor & Destructor Documentation

6.68.2.1 Functor() [1/4]

```
template<typename ResultType , typename ...  ArgTypes>
Functor< ResultType(ArgTypes...)>::Functor ( )  [default]
```

6.68.2.2 Functor() [2/4]

```
template<typename ResultType , typename ...  ArgTypes>
template<typename Function >
Functor< ResultType(ArgTypes...)>::Functor (
            Function function )  [inline]
```

Definition at line 20 of file functor.h.

6.68.2.3 Functor() [3/4]

```
template<typename ResultType , typename ...  ArgTypes>
template<typename Function , class Class >
Functor< ResultType(ArgTypes...)>::Functor (
            Function Class::* function )  [inline]
```

Definition at line 23 of file functor.h.

6.68.2.4 Functor() [4/4]

```
template<typename ResultType , typename ...  ArgTypes>
Functor< ResultType(ArgTypes...)>::Functor (
            const Functor< ResultType(ArgTypes...)> & other )  [inline]
```

Definition at line 25 of file functor.h.

6.68.3 Member Function Documentation

6.68.3.1 operator()()

```
template<typename ResultType , typename ...  ArgTypes>
ResultType Functor< ResultType(ArgTypes...)>::operator() (
            ArgTypes...  args )  [inline]
```

Invokes function

Parameters

args	Arguments for a function

Returns

Result of a function with passed args as arguments

Definition at line 36 of file functor.h.

6.68.3.2 operator=()

```
template<typename ResultType , typename ...  ArgTypes>
Functor& Functor< ResultType(ArgTypes...)>::operator= (
            const Functor< ResultType(ArgTypes...)> & other )  [inline]
```

Definition at line 27 of file functor.h.

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

- functor.h

6.69 GLib::GetNodesFromRoots< nodes, graph > Struct Template Reference

`#include <get_nodes_from_roots.h>`

Public Types

- using tail_result = typename GetNodesFromRoots< typename nodes::Tail, graph >::result
- using new_visited = typename DFS< typename nodes::Head, graph, tail_result >::new_visited
- using result = typename TL::Concatenate< new_visited, tail_result >::result

6.69.1 Detailed Description

template<class nodes, class graph>
struct GLib::GetNodesFromRoots< nodes, graph >

Gets all nodes that can be reached from roots. It's a good way to get full structure of a graph without entering every node.

Parameters

nodes	Template parameter, initial nodes.

Returns

Parameter result, nodes for bui

Definition at line 13 of file get_nodes_from_roots.h.

6.69.2 Member Typedef Documentation

6.69.2.1 new_visited

```
template<class nodes , class graph >
using GLib::GetNodesFromRoots< nodes, graph >::new_visited = typename DFS< typename nodes::↩
Head, graph, tail_result >::new_visited
```

Definition at line 16 of file get_nodes_from_roots.h.

6.69.2.2 result

```
template<class nodes , class graph >
using GLib::GetNodesFromRoots< nodes, graph >::result = typename TL::Concatenate<new_visited,
tail_result>::result
```

Definition at line 22 of file get_nodes_from_roots.h.

6.69.2.3 tail_result

```
template<class nodes , class graph >
using GLib::GetNodesFromRoots< nodes, graph >::tail_result = typename GetNodesFromRoots<typename
nodes::Tail, graph>::result
```

Definition at line 14 of file get_nodes_from_roots.h.

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

- graph/GLib/get_nodes_from_roots.h

6.70 GLib::GetNodesFromRoots< EmptyTypeList, graph > Struct Template Reference

`#include <get_nodes_from_roots.h>`

Public Types

- using result = EmptyTypeList

6.70.1 Detailed Description

template<class graph>
struct GLib::GetNodesFromRoots< EmptyTypeList, graph >

See also

GetNodesFromRoots

Definition at line 29 of file get_nodes_from_roots.h.

6.70.2 Member Typedef Documentation

6.70.2.1 result

```
template<class graph >
using GLib::GetNodesFromRoots< EmptyTypeList, graph >::result = EmptyTypeList
```

Definition at line 30 of file get_nodes_from_roots.h.

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

- graph/GLib/get_nodes_from_roots.h

6.71 GLib::GetReachedVertexes< graph, start > Struct Template Reference

```
#include <get_reached_vertexes.h>
```

Classes

- struct IterateThroughEdges
- struct IterateThroughEdges< EmptyTypeList >

Public Types

- using start_node = typename FindNodeByVertex< start, graph >::result
- using dfs_search = typename DFS< start_node, graph >::result
- using result = typename TL::Add< start, 0, typename IterateThroughEdges< dfs_search >::result >::result

6.71.1 Detailed Description

template<class graph, typename start>
struct GLib::GetReachedVertexes< graph, start >

Gets all vertexes that can be reached from vertex start.

See also

DFS

Parameters

graph	Template parameter, graph to process
start	Template parameter

Returns

Parameter result, all vertexes that are reached from start.

Definition at line 17 of file get_reached_vertexes.h.

6.71.2 Member Typedef Documentation

6.71.2.1 dfs_search

```
template<class graph , typename start >
using GLib::GetReachedVertexes< graph, start >::dfs_search = typename DFS<start_node, graph>↩
::result
```

Definition at line 24 of file get_reached_vertexes.h.

6.71.2.2 result

```
template<class graph , typename start >
using GLib::GetReachedVertexes< graph, start >::result = typename TL::Add< start, 0, typename
IterateThroughEdges<dfs_search>::result >::result
```

Definition at line 43 of file get_reached_vertexes.h.

6.71.2.3 start_node

```
template<class graph , typename start >
using GLib::GetReachedVertexes< graph, start >::start_node = typename FindNodeByVertex<start,
graph>::result
```

Definition at line 22 of file get_reached_vertexes.h.

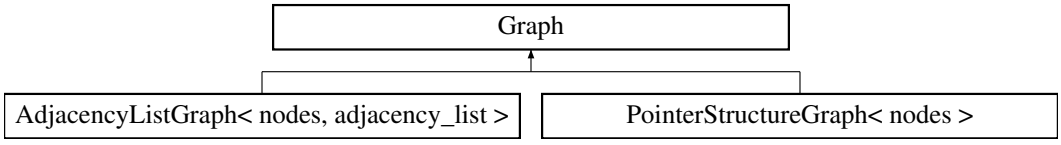
The documentation for this struct was generated from the following file:

- graph/GLib/get_reached_vertexes.h

6.72 Graph Struct Reference

```
#include <graph.h>
```

Inheritance diagram for Graph:



6.72.1 Detailed Description

Represents placeholder for a graph. Graph is a pair of vertexes (collection of some elements), and edges (collection of pairs of vertexes). Graph can be represented in multiple ways in code. This library provides several preexisting implementaions. Also it should be noted that preexisting implementations are built in compile-time, and it's recommended to follow this rule.

Definition at line 11 of file graph.h.

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

- graph/graphs/graph.h

6.73 TL::HasDerivedAndConstructible< type_list, T > Struct Template Reference

`#include <has_derived_and_constructible.h>`

Static Public Attributes

- constexpr static bool result

6.73.1 Detailed Description

template<class type_list, typename T>
struct TL::HasDerivedAndConstructible< type_list, T >

Checks if type_list contains derived and constructible child of T

Parameters

type_list	Template parameter
T	Template parameter

Returns

Parameter result, true if type_list ccontains derived and constructible child of T, false otherwise

Definition at line 33 of file has_derived_and_constructible.h.

6.73.2 Member Data Documentation

6.73.2.1 result

```
template<class type_list , typename T >
constexpr static bool TL::HasDerivedAndConstructible< type_list, T >::result  [static], [constexpr]
```

Initial value:

```
= CheckHasDerivedAndConstructible<
            type_list,
            T,
            std::is_base_of<T, typename type_list::Head>::value&&
            std::is_constructible<typename type_list::Head>::value
        >::result
```

Definition at line 34 of file has_derived_and_constructible.h.

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

- TL/has_derived_and_constructible.h

6.74 TL::HasDerivedAndConstructible< EmptyTypeList, T > Struct Template Reference

`#include <has_derived_and_constructible.h>`

Static Public Attributes

- constexpr static bool result = false

6.74.1 Detailed Description

template<typename T>
struct TL::HasDerivedAndConstructible< EmptyTypeList, T >

See also

HasDerivedAndConstructible

Definition at line 46 of file has_derived_and_constructible.h.

6.74.2 Member Data Documentation

6.74.2.1 result

```
template<typename T >
constexpr static bool TL::HasDerivedAndConstructible< EmptyTypeList, T >::result = false
[static], [constexpr]
```

Definition at line 47 of file has_derived_and_constructible.h.

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

- TL/has_derived_and_constructible.h

6.75 TL::IndexOf< type_list, T > Struct Template Reference

`#include <index_of.h>`

Static Public Attributes

- constexpr static int value = 1 + IndexOf<typename type_list::Tail, T>::value

6.75.1 Detailed Description

template<class type_list, typename T>
struct TL::IndexOf< type_list, T >

Gets index of a first ocurrence of typename T in type_list

Parameters

type_list	Template parameter
T	Template parameter

Returns

Parameter value, index of a first ocurrence of typename T in type_list, INT32_MIN otherwise

Definition at line 17 of file index_of.h.

6.75.2 Member Data Documentation

6.75.2.1 value

```
template<class type_list , typename T >
constexpr static int TL::IndexOf< type_list, T >::value = 1 + IndexOf<typename type_list::↩
Tail, T>::value  [static], [constexpr]
```

Definition at line 19 of file index_of.h.

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

- TL/index_of.h

6.76 TL::IndexOf< EmptyTypeList, T > Struct Template Reference

`#include <index_of.h>`

Static Public Attributes

- constexpr static int value = INT32_MIN

6.76.1 Detailed Description

template<typename T>
struct TL::IndexOf< EmptyTypeList, T >

See also

IndexOf

Definition at line 35 of file index_of.h.

6.76.2 Member Data Documentation

6.76.2.1 value

```
template<typename T >
constexpr static int TL::IndexOf< EmptyTypeList, T >::value = INT32_MIN  [static], [constexpr]
```

Definition at line 36 of file index_of.h.

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

- TL/index_of.h

6.77 TL::IndexOf< type_list, typename type_list::Head > Struct Template Reference

```
#include <index_of.h>
```

Static Public Attributes

- constexpr static int value = 0

6.77.1 Detailed Description

template<class type_list>
struct TL::IndexOf< type_list, typename type_list::Head >

See also

IndexOf

Definition at line 26 of file index_of.h.

6.77.2 Member Data Documentation

6.77.2.1 value

```
template<class type_list >
constexpr static int TL::IndexOf< type_list, typename type_list::Head >::value = 0  [static],
[constexpr]
```

Definition at line 28 of file index_of.h.

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

- TL/index_of.h

6.78 Objects::Integer< integer > Struct Template Reference

```
#include <objects.h>
```

Static Public Attributes

- constexpr static int value = integer

6.78.1 Detailed Description

template<int integer>
struct Objects::Integer< integer >

Definition at line 8 of file objects.h.

6.78.2 Member Data Documentation

6.78.2.1 value

```
template<int integer>
constexpr static int Objects::Integer< integer >::value = integer  [static], [constexpr]
```

Definition at line 9 of file objects.h.

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

- graph/objects.h

6.79 TL::IsBaseOf< parent, derived > Struct Template Reference

`#include <is_base_of.h>`

Static Public Attributes

- constexpr static bool result

6.79.1 Detailed Description

template<class parent, class derived>
struct TL::IsBaseOf< parent, derived >

Checks if TypeList "parent" is in fact parent of another TypeList "derived" "parent" is parent of "derived" if and only if for every class C in "derived", "parent" has parent of C

Parameters

parent	Template parameter
derived	Template parameter

Returns

true if TypeList "parent" is in fact parent of another TypeList "derived", false otherwise

Definition at line 38 of file is_base_of.h.

6.79.2 Member Data Documentation

6.79.2.1 result

```
template<class parent , class derived >
constexpr static bool TL::IsBaseOf< parent, derived >::result  [static], [constexpr]
```

Initial value:

```
= CheckIsBaseOf<
            ContainsParent<parent, typename derived::Head>::result,
            parent,
            derived
        >::result
```

Definition at line 39 of file is_base_of.h.

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

- TL/is_base_of.h

6.80 TL::IsBaseOf< EmptyTypeList, derived > Struct Template Reference

`#include <is_base_of.h>`

Static Public Attributes

- constexpr static bool result = false

6.80.1 Detailed Description

template<class derived>
struct TL::IsBaseOf< EmptyTypeList, derived >

Definition at line 52 of file is_base_of.h.

6.80.2 Member Data Documentation

6.80.2.1 result

```
template<class derived >
constexpr static bool TL::IsBaseOf< EmptyTypeList, derived >::result = false  [static], [constexpr]
```

Definition at line 53 of file is_base_of.h.

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

- TL/is_base_of.h

6.81 TL::IsBaseOf< EmptyTypeList, EmptyTypeList > Struct Reference

`#include <is_base_of.h>`

Static Public Attributes

- constexpr static bool result = true

6.81.1 Detailed Description

Definition at line 57 of file is_base_of.h.

6.81.2 Member Data Documentation

6.81.2.1 result

```
constexpr static bool TL::IsBaseOf< EmptyTypeList, EmptyTypeList >::result = true  [static],
[constexpr]
```

Definition at line 58 of file is_base_of.h.

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

- TL/is_base_of.h

6.82 TL::IsBaseOf< parent, EmptyTypeList > Struct Template Reference

```
#include <is_base_of.h>
```

Static Public Attributes

- constexpr static bool result = true

6.82.1 Detailed Description

template<class parent>
struct TL::IsBaseOf< parent, EmptyTypeList >

Definition at line 47 of file is_base_of.h.

6.82.2 Member Data Documentation

6.82.2.1 result

```
template<class parent >
constexpr static bool TL::IsBaseOf< parent, EmptyTypeList >::result = true  [static], [constexpr]
```

Definition at line 48 of file is_base_of.h.

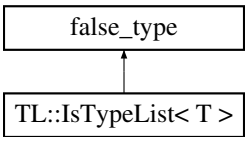
The documentation for this struct was generated from the following file:

- TL/is_base_of.h

6.83 TL::IsTypeList< T > Struct Template Reference

`#include <is_type_list.h>`

Inheritance diagram for TL::IsTypeList< T >:



6.83.1 Detailed Description

template<class T>
struct TL::IsTypeList< T >

Checks if passed class T is a TypeList

Parameters

T	Template argument

Returns

Parameter value, true if T is a TypeList, false otherwise

Definition at line 14 of file is_type_list.h.

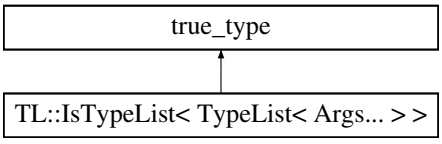
The documentation for this struct was generated from the following file:

- TL/is_type_list.h

6.84 TL::IsTypeList< TypeList< Args... > > Struct Template Reference

`#include <is_type_list.h>`

Inheritance diagram for TL::IsTypeList< TypeList< Args... > >:



6.84.1 Detailed Description

template<class ... Args>
struct TL::IsTypeList< TypeList< Args... > >

Definition at line 17 of file is_type_list.h.

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

- TL/is_type_list.h

6.85 GLib::DFS< cur_node, graph, visited_nodes >::IterateThroughChildren< cur_children, cur_visited > Struct Template Reference

`#include <dfs.h>`

Public Types

- using cur_edge = typename cur_children::Head
- using cur_child = typename GLib::FindNodeByVertex< typename cur_edge::to, graph >::result
- using new_visited = std::conditional_t< TL::Contains< cur_visited, cur_child >::result, upd_visited, typename DFS< cur_child, graph, upd_visited >::new_visited >
- using result = std::conditional_t< TL::Contains< upd_visited, cur_child >::result, typename IterateThroughChildren< typename cur_children::Tail, new_visited >::result, typename TL::Add< cur_edge, 0, typename TL::Concatenate< typename DFS< cur_child, graph, upd_visited >::result, typename IterateThroughChildren< typename cur_children::Tail, new_visited >::result >::result >::result >

6.85.1 Detailed Description

template<class cur_node, class graph, class visited_nodes = EmptyTypeList>
template<class cur_children, class cur_visited>
struct GLib::DFS< cur_node, graph, visited_nodes >::IterateThroughChildren< cur_children, cur_visited >

Definition at line 28 of file dfs.h.

6.85.2 Member Typedef Documentation

6.85.2.1 cur_child

```
template<class cur_node , class graph , class visited_nodes = EmptyTypeList>
template<class cur_children , class cur_visited >
using GLib::DFS< cur_node, graph, visited_nodes >::IterateThroughChildren< cur_children,
cur_visited >::cur_child = typename GLib::FindNodeByVertex< typename cur_edge::to, graph >↩
::result
```

Definition at line 30 of file dfs.h.

6.85.2.2 cur_edge

```
template<class cur_node , class graph , class visited_nodes = EmptyTypeList>
template<class cur_children , class cur_visited >
using GLib::DFS< cur_node, graph, visited_nodes >::IterateThroughChildren< cur_children,
cur_visited >::cur_edge = typename cur_children::Head
```

Definition at line 29 of file dfs.h.

6.85.2.3 new_visited

```
template<class cur_node , class graph , class visited_nodes = EmptyTypeList>
template<class cur_children , class cur_visited >
using GLib::DFS< cur_node, graph, visited_nodes >::IterateThroughChildren< cur_children,
cur_visited >::new_visited = std::conditional_t< TL::Contains<cur_visited, cur_child>::result,
upd_visited, typename DFS<cur_child, graph, upd_visited>::new_visited >
```

Definition at line 35 of file dfs.h.

6.85.2.4 result

```
template<class cur_node , class graph , class visited_nodes = EmptyTypeList>
template<class cur_children , class cur_visited >
using GLib::DFS< cur_node, graph, visited_nodes >::IterateThroughChildren< cur_children,
cur_visited >::result = std::conditional_t< TL::Contains<upd_visited, cur_child>::result,
typename IterateThroughChildren< typename cur_children::Tail, new_visited >::result, typename
TL::Add< cur_edge, 0, typename TL::Concatenate< typename DFS<cur_child, graph, upd_visited>::result, typename IterateThroughChildren< typename cur_children::Tail, new_visited >::result
>::result >::result >
```

Definition at line 41 of file dfs.h.

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

- graph/GLib/dfs.h

6.86 GLib::DFS< cur_node, graph, visited_nodes >::IterateThroughChildren< EmptyTypeList, cur_unvisited > Struct Template Reference

```
#include <dfs.h>
```

Public Types

- using result = EmptyTypeList
- using new_visited = upd_visited

6.86.1 Detailed Description

template<class cur_node, class graph, class visited_nodes = EmptyTypeList>
template<class cur_unvisited>
struct GLib::DFS< cur_node, graph, visited_nodes >::IterateThroughChildren< EmptyTypeList, cur_unvisited >

Definition at line 62 of file dfs.h.

6.86.2 Member Typedef Documentation

6.86.2.1 new_visited

```
template<class cur_node , class graph , class visited_nodes = EmptyTypeList>
template<class cur_unvisited >
using GLib::DFS< cur_node, graph, visited_nodes >::IterateThroughChildren< EmptyTypeList,
cur_unvisited >::new_visited = upd_visited
```

Definition at line 64 of file dfs.h.

6.86.2.2 result

```
template<class cur_node , class graph , class visited_nodes = EmptyTypeList>
template<class cur_unvisited >
using GLib::DFS< cur_node, graph, visited_nodes >::IterateThroughChildren< EmptyTypeList,
cur_unvisited >::result = EmptyTypeList
```

Definition at line 63 of file dfs.h.

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

- graph/GLib/dfs.h

6.87 ConvertGraph< EDGE_LIST, ADJACENCY_MATRIX, graph >::IterateThroughEdges< cur_edges > Struct Template Reference

```
#include <convert_to_adjacency_matrix.h>
```

Public Types

- using cur_edge = typename cur_edges::Head
- using from = typename cur_edge::from
- using to = typename cur_edge::to
- using new_weight = std::conditional_t< std::is_same_v< NullType, typename cur_edge::weight >, Objects::Boolean< true >, std::conditional_t< std::is_same_v< Objects::Boolean< false >, typename cur_edge::weight >, Class< Objects::Boolean< false > >, std::conditional_t< std::is_same_v< Objects::Boolean< true >, typename cur_edge::weight >, Class< Objects::Boolean< true > >, typename cur_edge::weight > > >
- using tail_result = typename IterateThroughEdges< typename cur_edges::Tail >::result
- using result = typename TL::Replace< typename TL::Replace< new_weight, to_ind, typename TL::TypeAt< tail_result, from_ind >::value >::result, from_ind, tail_result >::result

Static Public Attributes

- constexpr static size_t from_ind = TL::IndexOf<vertexes, typename cur_edge::from>::value
- constexpr static size_t to_ind = TL::IndexOf<vertexes, typename cur_edge::to>::value

6.87.1 Detailed Description

template<class graph>
template<class cur_edges>
struct ConvertGraph< EDGE_LIST, ADJACENCY_MATRIX, graph >::IterateThroughEdges< cur_edges >

Definition at line 19 of file convert_to_adjacency_matrix.h.

6.87.2 Member Typedef Documentation

6.87.2.1 cur_edge

```
template<class graph >
template<class cur_edges >
using ConvertGraph< EDGE_LIST, ADJACENCY_MATRIX, graph >::IterateThroughEdges< cur_edges >↩
::cur_edge = typename cur_edges::Head
```

Definition at line 20 of file convert_to_adjacency_matrix.h.

6.87.2.2 from

```
template<class graph >
template<class cur_edges >
using ConvertGraph< EDGE_LIST, ADJACENCY_MATRIX, graph >::IterateThroughEdges< cur_edges >↩
::from = typename cur_edge::from
```

Definition at line 22 of file convert_to_adjacency_matrix.h.

6.87.2.3 new_weight

```
template<class graph >
template<class cur_edges >
using ConvertGraph< EDGE_LIST, ADJACENCY_MATRIX, graph >::IterateThroughEdges< cur_edges
>::new_weight = std::conditional_t< std::is_same_v<NullType, typename cur_edge::weight>,
Objects::Boolean<true>, std::conditional_t< std::is_same_v<Objects::Boolean<false>, typename
cur_edge::weight>, Class<Objects::Boolean<false> >, std::conditional_t< std::is_same_↩
v<Objects::Boolean<true>, typename cur_edge::weight>, Class<Objects::Boolean<true> >, typename
cur_edge::weight > > >
```

Definition at line 30 of file convert_to_adjacency_matrix.h.

6.87.2.4 result

```
template<class graph >
template<class cur_edges >
using ConvertGraph< EDGE_LIST, ADJACENCY_MATRIX, graph >::IterateThroughEdges< cur_edges >↩
::result = typename TL::Replace< typename TL::Replace< new_weight, to_ind, typename TL::TypeAt<tail_result,
from_ind>::value >::result, from_ind, tail_result >::result
```

Definition at line 45 of file convert_to_adjacency_matrix.h.

6.87.2.5 tail_result

```
template<class graph >
template<class cur_edges >
using ConvertGraph< EDGE_LIST, ADJACENCY_MATRIX, graph >::IterateThroughEdges< cur_edges >↩
::tail_result = typename IterateThroughEdges<typename cur_edges::Tail>::result
```

Definition at line 44 of file convert_to_adjacency_matrix.h.

6.87.2.6 to

```
template<class graph >
template<class cur_edges >
using ConvertGraph< EDGE_LIST, ADJACENCY_MATRIX, graph >::IterateThroughEdges< cur_edges >↩
::to = typename cur_edge::to
```

Definition at line 23 of file convert_to_adjacency_matrix.h.

6.87.3 Member Data Documentation

6.87.3.1 from_ind

```
template<class graph >
template<class cur_edges >
constexpr static size_t ConvertGraph< EDGE_LIST, ADJACENCY_MATRIX, graph >::IterateThrough↩
Edges< cur_edges >::from_ind = TL::IndexOf<vertexes, typename cur_edge::from>::value  [static],
[constexpr]
```

Definition at line 27 of file convert_to_adjacency_matrix.h.

6.87.3.2 to_ind

```
template<class graph >
template<class cur_edges >
constexpr static size_t ConvertGraph< EDGE_LIST, ADJACENCY_MATRIX, graph >::IterateThrough↩
Edges< cur_edges >::to_ind = TL::IndexOf<vertexes, typename cur_edge::to>::value  [static],
[constexpr]
```

Definition at line 28 of file convert_to_adjacency_matrix.h.

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

- graph/convert/convert_to_adjacency_matrix.h

6.88 GLib::FindPath< graph_raw, start, finish >::IterateThroughEdges< cur_edges, wanted_node > Struct Template Reference

```
#include <find_path.h>
```

Public Types

- using cur_edge = typename cur_edges::Head
- using path = typename std::conditional_t< found, typename TL::Add< typename wanted_node::vertex, 0, typename IterateThroughEdges< typename cur_edges::Tail, typename FindNodeByVertex< typename cur_edge::from, graph >::result >::path >::result, typename IterateThroughEdges< typename cur_edges↩ ::Tail, wanted_node >::path >
- using weights = typename std::conditional_t< found, typename TL::Add< typename cur_edge::weight, 0, typename IterateThroughEdges< typename cur_edges::Tail, typename FindNodeByVertex< typename cur_edge::from, graph >::result >::weights >::result, typename IterateThroughEdges< typename cur_↩ edges::Tail, wanted_node >::weights >

Static Public Attributes

- constexpr static bool found

6.88.1 Detailed Description

template<class graph_raw, typename start, typename finish>
template<class cur_edges, class wanted_node>
struct GLib::FindPath< graph_raw, start, finish >::IterateThroughEdges< cur_edges, wanted_node >

Definition at line 36 of file find_path.h.

6.88.2 Member Typedef Documentation

6.88.2.1 cur_edge

```
template<class graph_raw , typename start , typename finish >
template<class cur_edges , class wanted_node >
using GLib::FindPath< graph_raw, start, finish >::IterateThroughEdges< cur_edges, wanted_node
>::cur_edge = typename cur_edges::Head
```

Definition at line 37 of file find_path.h.

6.88.2.2 path

```
template<class graph_raw , typename start , typename finish >
template<class cur_edges , class wanted_node >
using GLib::FindPath< graph_raw, start, finish >::IterateThroughEdges< cur_edges, wanted_node
>::path = typename std::conditional_t<found, typename TL::Add< typename wanted_node::vertex,
0, typename IterateThroughEdges< typename cur_edges::Tail, typename FindNodeByVertex< typename
cur_edge::from, graph >::result >::path >::result, typename IterateThroughEdges<typename
cur_edges::Tail, wanted_node>::path >
```

Definition at line 43 of file find_path.h.

6.88.2.3 weights

```
template<class graph_raw , typename start , typename finish >
template<class cur_edges , class wanted_node >
using GLib::FindPath< graph_raw, start, finish >::IterateThroughEdges< cur_edges, wanted_node
>::weights = typename std::conditional_t<found, typename TL::Add< typename cur_edge::weight,
0, typename IterateThroughEdges< typename cur_edges::Tail, typename FindNodeByVertex< typename
cur_edge::from, graph >::result >::weights >::result, typename IterateThroughEdges<typename
cur_edges::Tail, wanted_node>::weights >
```

Definition at line 57 of file find_path.h.

6.88.3 Member Data Documentation

6.88.3.1 found

```
template<class graph_raw , typename start , typename finish >
template<class cur_edges , class wanted_node >
constexpr static bool GLib::FindPath< graph_raw, start, finish >::IterateThroughEdges< cur_↩
edges, wanted_node >::found  [static], [constexpr]
```

Initial value:

```
= std::is_same<
                typename cur_edge::to,
                typename wanted_node::vertex
            >::value
```

Definition at line 38 of file find_path.h.

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

- graph/GLib/find_path.h

6.89 ConvertGraph< EDGE_LIST, ADJACENCY_LIST, graph >::IterateThroughEdges< edge_list > Struct Template Reference

```
#include <convert_to_adjacency_list.h>
```

Public Types

- using result = typename GLib::AddEdge< ADJACENCY_LIST, typename IterateThroughEdges< typename edge_list::Tail >::result, typename edge_list::Head >::result

6.89.1 Detailed Description

template<class graph>
template<class edge_list>
struct ConvertGraph< EDGE_LIST, ADJACENCY_LIST, graph >::IterateThroughEdges< edge_list >

Definition at line 25 of file convert_to_adjacency_list.h.

6.89.2 Member Typedef Documentation

6.89.2.1 result

```
template<class graph >
template<class edge_list >
using ConvertGraph< EDGE_LIST, ADJACENCY_LIST, graph >::IterateThroughEdges< edge_list >↩
::result = typename GLib::AddEdge< ADJACENCY_LIST, typename IterateThroughEdges<typename
edge_list::Tail>::result, typename edge_list::Head >::result
```

Definition at line 26 of file convert_to_adjacency_list.h.

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

- graph/convert/convert_to_adjacency_list.h

6.90 GLib::GetReachedVertexes< graph, start >::IterateThroughEdges< cur_edges > Struct Template Reference

```
#include <get_reached_vertexes.h>
```

Public Types

- using cur_edge = typename cur_edges::Head
- using result = typename TL::Add< typename cur_edge::to, 0, typename IterateThroughEdges< typename cur_edges::Tail >::result >::result

6.90.1 Detailed Description

template<class graph, typename start>
template<class cur_edges>
struct GLib::GetReachedVertexes< graph, start >::IterateThroughEdges< cur_edges >

Definition at line 28 of file get_reached_vertexes.h.

6.90.2 Member Typedef Documentation

6.90.2.1 cur_edge

```
template<class graph , typename start >
template<class cur_edges >
using GLib::GetReachedVertexes< graph, start >::IterateThroughEdges< cur_edges >::cur_edge =
typename cur_edges::Head
```

Definition at line 29 of file get_reached_vertexes.h.

6.90.2.2 result

```
template<class graph , typename start >
template<class cur_edges >
using GLib::GetReachedVertexes< graph, start >::IterateThroughEdges< cur_edges >::result =
typename TL::Add< typename cur_edge::to, 0, typename IterateThroughEdges<typename cur_edges↩
::Tail>::result >::result
```

Definition at line 30 of file get_reached_vertexes.h.

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

- graph/GLib/get_reached_vertexes.h

6.91 GLib::GetReachedVertexes< graph, start >::IterateThroughEdges< EmptyTypeList > Struct Reference

```
#include <get_reached_vertexes.h>
```

Public Types

- using result = EmptyTypeList

6.91.1 Detailed Description

template<class graph, typename start>
struct GLib::GetReachedVertexes< graph, start >::IterateThroughEdges< EmptyTypeList >

Definition at line 38 of file get_reached_vertexes.h.

6.91.2 Member Typedef Documentation

6.91.2.1 result

```
template<class graph , typename start >
using GLib::GetReachedVertexes< graph, start >::IterateThroughEdges< EmptyTypeList >::result
= EmptyTypeList
```

Definition at line 39 of file get_reached_vertexes.h.

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

- graph/GLib/get_reached_vertexes.h

6.92 ConvertGraph< EDGE_LIST, ADJACENCY_MATRIX, graph >::IterateThroughEdges< EmptyTypeList > Struct Reference

`#include <convert_to_adjacency_matrix.h>`

Public Types

- using result = typename TL::FillTypeListWithObject< typename TL::FillTypeListWithObject< Objects::Boolean< false >, TL::Size< vertexes >::size >::result, TL::Size< vertexes >::size >::result

6.92.1 Detailed Description

template<class graph>
struct ConvertGraph< EDGE_LIST, ADJACENCY_MATRIX, graph >::IterateThroughEdges< EmptyTypeList >

Definition at line 57 of file convert_to_adjacency_matrix.h.

6.92.2 Member Typedef Documentation

6.92.2.1 result

```
template<class graph >
using ConvertGraph< EDGE_LIST, ADJACENCY_MATRIX, graph >::IterateThroughEdges< EmptyTypeList
>::result = typename TL::FillTypeListWithObject< typename TL::FillTypeListWithObject< Objects::Boolean<false
TL::Size<vertexes>::size >::result, TL::Size<vertexes>::size >::result
```

Definition at line 58 of file convert_to_adjacency_matrix.h.

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

- graph/convert/convert_to_adjacency_matrix.h

6.93 ConvertGraph< EDGE_LIST, ADJACENCY_LIST, graph >::IterateThroughEdges< EmptyTypeList > Struct Reference

`#include <convert_to_adjacency_list.h>`

Public Types

- using result = AdjacencyListGraph< typename graph::vertexes_, typename TL::FillTypeListWithObject< EmptyTypeList, TL::Size< typename graph::vertexes_ >::size >::result >

6.93.1 Detailed Description

template<class graph>
struct ConvertGraph< EDGE_LIST, ADJACENCY_LIST, graph >::IterateThroughEdges< EmptyTypeList >

Definition at line 34 of file convert_to_adjacency_list.h.

6.93.2 Member Typedef Documentation

6.93.2.1 result

```
template<class graph >
using ConvertGraph< EDGE_LIST, ADJACENCY_LIST, graph >::IterateThroughEdges< EmptyTypeList
>::result = AdjacencyListGraph< typename graph::vertexes_, typename TL::FillTypeListWithObject<
EmptyTypeList, TL::Size<typename graph::vertexes_>::size >::result >
```

Definition at line 35 of file convert_to_adjacency_list.h.

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

- graph/convert/convert_to_adjacency_list.h

6.94 GLib::FindPath< graph_raw, start, finish >::IterateThroughEdges< EmptyTypeList, wanted_node > Struct Template Reference

```
#include <find_path.h>
```

Public Types

- using path = EmptyTypeList
- using weights = EmptyTypeList

6.94.1 Detailed Description

template<class graph_raw, typename start, typename finish>
template<class wanted_node>
struct GLib::FindPath< graph_raw, start, finish >::IterateThroughEdges< EmptyTypeList, wanted_node >

Definition at line 73 of file find_path.h.

6.94.2 Member Typedef Documentation

6.94.2.1 path

```
template<class graph_raw , typename start , typename finish >
template<class wanted_node >
using GLib::FindPath< graph_raw, start, finish >::IterateThroughEdges< EmptyTypeList, wanted↩
_node >::path = EmptyTypeList
```

Definition at line 74 of file find_path.h.

6.94.2.2 weights

```
template<class graph_raw , typename start , typename finish >
template<class wanted_node >
using GLib::FindPath< graph_raw, start, finish >::IterateThroughEdges< EmptyTypeList, wanted↩
_node >::weights = EmptyTypeList
```

Definition at line 75 of file find_path.h.

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

- graph/GLib/find_path.h

6.95 TL::Reverse< type_list >::IterateThroughElements< cur_type_list, cur_result > Struct Template Reference

```
#include <reverse.h>
```

Public Types

- using result = typename IterateThroughElements< typename cur_type_list::Tail, typename TL::Add< typename cur_type_list::Head, 0, cur_result >::result >::result

6.95.1 Detailed Description

template<class type_list>
template<class cur_type_list, class cur_result>
struct TL::Reverse< type_list >::IterateThroughElements< cur_type_list, cur_result >

Definition at line 14 of file reverse.h.

6.95.2 Member Typedef Documentation

6.95.2.1 result

```
template<class type_list >
template<class cur_type_list , class cur_result >
using TL::Reverse< type_list >::IterateThroughElements< cur_type_list, cur_result >::result
= typename IterateThroughElements < typename cur_type_list::Tail, typename TL::Add< typename
cur_type_list::Head, 0, cur_result >::result >::result
```

Definition at line 15 of file reverse.h.

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

- TL/reverse.h

6.96 TL::Reverse< type_list >::IterateThroughElements< EmptyTypeList, cur_result > Struct Template Reference

```
#include <reverse.h>
```

Public Types

- using result = cur_result

6.96.1 Detailed Description

template<class type_list>
template<class cur_result>
struct TL::Reverse< type_list >::IterateThroughElements< EmptyTypeList, cur_result >

Definition at line 26 of file reverse.h.

6.96.2 Member Typedef Documentation

6.96.2.1 result

```
template<class type_list >
template<class cur_result >
using TL::Reverse< type_list >::IterateThroughElements< EmptyTypeList, cur_result >::result =
cur_result
```

Definition at line 27 of file reverse.h.

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

- TL/reverse.h

6.97 ConvertGraph< ADJACENCY_MATRIX, EDGE_LIST, graph >::IterateThroughMatrix< cur_index > Struct Template Reference

`#include <convert_to_edge_list.h>`

Public Types

- using from = typename TL::TypeAt< vertexes, row >::value
- using to = typename TL::TypeAt< vertexes, col >::value
- using cell = typename TL::TypeAt< typename TL::TypeAt< typename graph::matrix_, row >::value, col >::value
- using weight = std::conditional_t< std::is_same< Objects::Boolean< true >, cell >::value, NullType, cell >
- using tail_result = typename IterateThroughMatrix< cur_index - 1 >::result
- using result = std::conditional_t< std::is_same< Objects::Boolean< false >, cell >::value, tail_result, typename TL::Add< Edge< from, to, weight >, 0, tail_result >::result >

Static Public Attributes

- constexpr static int row = cur_index / n
- constexpr static int col = cur_index % n

6.97.1 Detailed Description

template<class graph>
template<int cur_index>
struct ConvertGraph< ADJACENCY_MATRIX, EDGE_LIST, graph >::IterateThroughMatrix< cur_index >

Definition at line 71 of file convert_to_edge_list.h.

6.97.2 Member Typedef Documentation

6.97.2.1 cell

```
template<class graph >
template<int cur_index>
using ConvertGraph< ADJACENCY_MATRIX, EDGE_LIST, graph >::IterateThroughMatrix< cur_index >
::cell = typename TL::TypeAt< typename TL::TypeAt<typename graph::matrix_, row>::value, col
>::value
```

Definition at line 77 of file convert_to_edge_list.h.

6.97.2.2 from

```
template<class graph >
template<int cur_index>
using ConvertGraph< ADJACENCY_MATRIX, EDGE_LIST, graph >::IterateThroughMatrix< cur_index >↩
::from = typename TL::TypeAt<vertexes, row>::value
```

Definition at line 74 of file convert_to_edge_list.h.

6.97.2.3 result

```
template<class graph >
template<int cur_index>
using ConvertGraph< ADJACENCY_MATRIX, EDGE_LIST, graph >::IterateThroughMatrix< cur_index >↩
::result = std::conditional_t< std::is_same<Objects::Boolean<false>, cell>::value, tail_result,
typename TL::Add< Edge<from, to, weight>, 0, tail_result >::result >
```

Definition at line 89 of file convert_to_edge_list.h.

6.97.2.4 tail_result

```
template<class graph >
template<int cur_index>
using ConvertGraph< ADJACENCY_MATRIX, EDGE_LIST, graph >::IterateThroughMatrix< cur_index >↩
::tail_result = typename IterateThroughMatrix<cur_index - 1>::result
```

Definition at line 87 of file convert_to_edge_list.h.

6.97.2.5 to

```
template<class graph >
template<int cur_index>
using ConvertGraph< ADJACENCY_MATRIX, EDGE_LIST, graph >::IterateThroughMatrix< cur_index >↩
::to = typename TL::TypeAt<vertexes, col>::value
```

Definition at line 75 of file convert_to_edge_list.h.

6.97.2.6 weight

```
template<class graph >
template<int cur_index>
using ConvertGraph< ADJACENCY_MATRIX, EDGE_LIST, graph >::IterateThroughMatrix< cur_index >↩
::weight = std::conditional_t< std::is_same<Objects::Boolean<true>, cell>::value, NullType,
cell >
```

Definition at line 81 of file convert_to_edge_list.h.

6.97.3 Member Data Documentation

6.97.3.1 col

```
template<class graph >
template<int cur_index>
constexpr static int ConvertGraph< ADJACENCY_MATRIX, EDGE_LIST, graph >::IterateThrough↩
Matrix< cur_index >::col = cur_index % n  [static]
```

Definition at line 72 of file convert_to_edge_list.h.

6.97.3.2 row

```
template<class graph >
template<int cur_index>
constexpr static int ConvertGraph< ADJACENCY_MATRIX, EDGE_LIST, graph >::IterateThrough↩
Matrix< cur_index >::row = cur_index / n  [static], [constexpr]
```

Definition at line 72 of file convert_to_edge_list.h.

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

- graph/convert/convert_to_edge_list.h

6.98 ConvertGraph< ADJACENCY_MATRIX, EDGE_LIST, graph >::IterateThroughMatrix<-1 > Struct Reference

```
#include <convert_to_edge_list.h>
```

Public Types

- using result = EmptyTypeList

6.98.1 Detailed Description

template<class graph>
struct ConvertGraph< ADJACENCY_MATRIX, EDGE_LIST, graph >::IterateThroughMatrix<-1 >

Definition at line 101 of file convert_to_edge_list.h.

6.98.2 Member Typedef Documentation

6.98.2.1 result

```
template<class graph >
using ConvertGraph< ADJACENCY_MATRIX, EDGE_LIST, graph >::IterateThroughMatrix<-1 >::result =
EmptyTypeList
```

Definition at line 102 of file convert_to_edge_list.h.

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

- graph/convert/convert_to_edge_list.h

6.99 GLib::FindNodeByVertex< vertex, graph >::IterateThroughNodes< cur_nodes > Struct Template Reference

```
#include <find_node_by_vertex.h>
```

Public Types

- using cur_node = typename cur_nodes::Head
- using result = std::conditional_t< std::is_same< vertex, typename cur_node::vertex >::value, cur_node, typename IterateThroughNodes< typename cur_nodes::Tail >::result >

6.99.1 Detailed Description

template<typename vertex, class graph>
template<class cur_nodes>
struct GLib::FindNodeByVertex< vertex, graph >::IterateThroughNodes< cur_nodes >

Definition at line 17 of file find_node_by_vertex.h.

6.99.2 Member Typedef Documentation

6.99.2.1 cur_node

```
template<typename vertex , class graph >
template<class cur_nodes >
using GLib::FindNodeByVertex< vertex, graph >::IterateThroughNodes< cur_nodes >::cur_node =
typename cur_nodes::Head
```

Definition at line 18 of file find_node_by_vertex.h.

6.99.2.2 result

```
template<typename vertex , class graph >
template<class cur_nodes >
using GLib::FindNodeByVertex< vertex, graph >::IterateThroughNodes< cur_nodes >::result =
std::conditional_t< std::is_same<vertex, typename cur_node::vertex>::value, cur_node, typename
IterateThroughNodes<typename cur_nodes::Tail>::result >
```

Definition at line 19 of file find_node_by_vertex.h.

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

- graph/GLib/find_node_by_vertex.h

6.100 ConvertGraph< POINTER_STRUCTURE, EDGE_LIST, graph >::IterateThroughNodes< cur_nodes > Struct Template Reference

```
#include <convert_to_edge_list.h>
```

Public Types

- using tail_call = IterateThroughNodes< typename cur_nodes::Tail >
- using cur_node = typename cur_nodes::Head
- using vertexes = typename TL::Add< typename cur_node::vertex, 0, typename tail_call::vertexes >::result
- using edges = typename TL::Concatenate< typename cur_node::children, typename tail_call::edges >↩ ::result

6.100.1 Detailed Description

template<class graph>
template<class cur_nodes>
struct ConvertGraph< POINTER_STRUCTURE, EDGE_LIST, graph >::IterateThroughNodes< cur_nodes >

Definition at line 29 of file convert_to_edge_list.h.

6.100.2 Member Typedef Documentation

6.100.2.1 cur_node

```
template<class graph >
template<class cur_nodes >
using ConvertGraph< POINTER_STRUCTURE, EDGE_LIST, graph >::IterateThroughNodes< cur_nodes >↩
::cur_node = typename cur_nodes::Head
```

Definition at line 31 of file convert_to_edge_list.h.

6.100.2.2 edges

```
template<class graph >
template<class cur_nodes >
using ConvertGraph< POINTER_STRUCTURE, EDGE_LIST, graph >::IterateThroughNodes< cur_nodes >↩
::edges = typename TL::Concatenate< typename cur_node::children, typename tail_call::edges >↩
::result
```

Definition at line 40 of file convert_to_edge_list.h.

6.100.2.3 tail_call

```
template<class graph >
template<class cur_nodes >
using ConvertGraph< POINTER_STRUCTURE, EDGE_LIST, graph >::IterateThroughNodes< cur_nodes >↩
::tail_call = IterateThroughNodes<typename cur_nodes::Tail>
```

Definition at line 30 of file convert_to_edge_list.h.

6.100.2.4 vertexes

```
template<class graph >
template<class cur_nodes >
using ConvertGraph< POINTER_STRUCTURE, EDGE_LIST, graph >::IterateThroughNodes< cur_nodes >↩
::vertexes = typename TL::Add< typename cur_node::vertex, 0, typename tail_call::vertexes >↩
::result
```

Definition at line 34 of file convert_to_edge_list.h.

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

- graph/convert/convert_to_edge_list.h

6.101 GLib::FindNodeByVertex< vertex, graph >::IterateThroughNodes< EmptyTypeList > Struct Reference

```
#include <find_node_by_vertex.h>
```

Public Types

- using result = NullType

6.101.1 Detailed Description

template<typename vertex, class graph>
struct GLib::FindNodeByVertex< vertex, graph >::IterateThroughNodes< EmptyTypeList >

Definition at line 27 of file find_node_by_vertex.h.

6.101.2 Member Typedef Documentation

6.101.2.1 result

```
template<typename vertex , class graph >
using GLib::FindNodeByVertex< vertex, graph >::IterateThroughNodes< EmptyTypeList >::result =
NullType
```

Definition at line 28 of file find_node_by_vertex.h.

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

- graph/GLib/find_node_by_vertex.h

6.102 ConvertGraph< POINTER_STRUCTURE, EDGE_LIST, graph >::IterateThroughNodes< EmptyTypeList > Struct Reference

```
#include <convert_to_edge_list.h>
```

Public Types

- using vertexes = EmptyTypeList
- using edges = EmptyTypeList

6.102.1 Detailed Description

template<class graph>
struct ConvertGraph< POINTER_STRUCTURE, EDGE_LIST, graph >::IterateThroughNodes< EmptyTypeList >

Definition at line 47 of file convert_to_edge_list.h.

6.102.2 Member Typedef Documentation

6.102.2.1 edges

```
template<class graph >
using ConvertGraph< POINTER_STRUCTURE, EDGE_LIST, graph >::IterateThroughNodes< EmptyTypeList
>::edges = EmptyTypeList
```

Definition at line 49 of file convert_to_edge_list.h.

6.102.2.2 vertexes

```
template<class graph >
using ConvertGraph< POINTER_STRUCTURE, EDGE_LIST, graph >::IterateThroughNodes< EmptyTypeList
>::vertexes = EmptyTypeList
```

Definition at line 48 of file convert_to_edge_list.h.

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

- graph/convert/convert_to_edge_list.h

6.103 TL::Concatenate< front, back >::IterateThroughReversedFront< elements, current > Struct Template Reference

```
#include <concatenate.h>
```

Public Types

- using added = typename Add< typename elements::Head, 0, current >::result
- using result = typename IterateThroughReversedFront< typename elements::Tail, added >::result

6.103.1 Detailed Description

template<class front, class back>
template<class elements, class current>
struct TL::Concatenate< front, back >::IterateThroughReversedFront< elements, current >

Definition at line 24 of file concatenate.h.

6.103.2 Member Typedef Documentation

6.103.2.1 added

```
template<class front , class back >
template<class elements , class current >
using TL::Concatenate< front, back >::IterateThroughReversedFront< elements, current >::added
= typename Add< typename elements::Head, 0, current >::result
```

Definition at line 25 of file concatenate.h.

6.103.2.2 result

```
template<class front , class back >
template<class elements , class current >
using TL::Concatenate< front, back >::IterateThroughReversedFront< elements, current >↩
::result = typename IterateThroughReversedFront< typename elements::Tail, added >::result
```

Definition at line 31 of file concatenate.h.

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

- TL/concatenate.h

6.104 TL::Concatenate< front, back >::IterateThroughReversedFront< EmptyTypeList, current > Struct Template Reference

```
#include <concatenate.h>
```

Public Types

- using result = current

6.104.1 Detailed Description

template<class front, class back>
template<class current>
struct TL::Concatenate< front, back >::IterateThroughReversedFront< EmptyTypeList, current >

Definition at line 38 of file concatenate.h.

6.104.2 Member Typedef Documentation

6.104.2.1 result

```
template<class front , class back >
template<class current >
using TL::Concatenate< front, back >::IterateThroughReversedFront< EmptyTypeList, current >↩
::result = current
```

Definition at line 39 of file concatenate.h.

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

- TL/concatenate.h

6.105 ConvertGraph< ADJACENCY_LIST, POINTER_STRUCTURE, graph >::MakePointerStructureGraph< current_vertexes, current_adjacency_list > Struct Template Reference

```
#include <convert_to_pointer_structure.h>
```

Public Types

- using type_list_without_first = typename MakePointerStructureGraph< typename current_vertexes::Tail, typename current_adjacency_list::Tail >::result
- using result = typename TL::Add< PointerStructureNode< typename current_vertexes::Head, typename current_adjacency_list::Head >, 0, type_list_without_first >::result

6.105.1 Detailed Description

template<class graph>
template<class current_vertexes, class current_adjacency_list>
struct ConvertGraph< ADJACENCY_LIST, POINTER_STRUCTURE, graph >::MakePointerStructureGraph< current_vertexes, current_adjacency_list >

Definition at line 28 of file convert_to_pointer_structure.h.

6.105.2 Member Typedef Documentation

6.105.2.1 result

```
template<class graph >
template<class current_vertexes , class current_adjacency_list >
using ConvertGraph< ADJACENCY_LIST, POINTER_STRUCTURE, graph >::MakePointerStructureGraph<
current_vertexes, current_adjacency_list >::result = typename TL::Add< PointerStructureNode<
typename current_vertexes::Head, typename current_adjacency_list::Head >, 0, type_list_without_first
>::result
```

Definition at line 34 of file convert_to_pointer_structure.h.

6.105.2.2 type_list_without_first

```
template<class graph >
template<class current_vertexes , class current_adjacency_list >
using ConvertGraph< ADJACENCY_LIST, POINTER_STRUCTURE, graph >::MakePointerStructureGraph<
current_vertexes, current_adjacency_list >::type_list_without_first = typename MakePointer↩
StructureGraph< typename current_vertexes::Tail, typename current_adjacency_list::Tail >↩
::result
```

Definition at line 29 of file convert_to_pointer_structure.h.

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

- graph/convert/convert_to_pointer_structure.h

6.106 ConvertGraph< ADJACENCY_LIST, POINTER_STRUCTURE, graph >::MakePointerStructureGraph< EmptyTypeList, EmptyTypeList > Struct Reference

```
#include <convert_to_pointer_structure.h>
```

Public Types

- using result = EmptyTypeList

6.106.1 Detailed Description

template<class graph>
struct ConvertGraph< ADJACENCY_LIST, POINTER_STRUCTURE, graph >::MakePointerStructureGraph< EmptyTypeList, EmptyTypeList >

Definition at line 45 of file convert_to_pointer_structure.h.

6.106.2 Member Typedef Documentation

6.106.2.1 result

```
template<class graph >
using ConvertGraph< ADJACENCY_LIST, POINTER_STRUCTURE, graph >::MakePointerStructureGraph<
EmptyTypeList, EmptyTypeList >::result = EmptyTypeList
```

Definition at line 46 of file convert_to_pointer_structure.h.

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

- graph/convert/convert_to_pointer_structure.h

6.107 TL::MostDerived< type_list, T > Struct Template Reference

`#include <most_derived.h>`

Public Types

- using result = typename CheckMostDerived< type_list, T, std::is_base_of< T, typename type_list::Head >::value >::result

6.107.1 Detailed Description

template<class type_list, typename T>
struct TL::MostDerived< type_list, T >

Finds the most derived child of T in type_list

Parameters

type_list	Template parameter
T	Template parameter

Returns

Parameter result, the most derived child of T in type_list

Definition at line 35 of file most_derived.h.

6.107.2 Member Typedef Documentation

6.107.2.1 result

```
template<class type_list , typename T >
using TL::MostDerived< type_list, T >::result = typename CheckMostDerived< type_list, T,
std::is_base_of<T, typename type_list::Head>::value >::result
```

Definition at line 36 of file most_derived.h.

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

- TL/most_derived.h

6.108 TL::MostDerived< EmptyTypeList, T > Struct Template Reference

`#include <most_derived.h>`

Public Types

- using result = T

6.108.1 Detailed Description

template<typename T>
struct TL::MostDerived< EmptyTypeList, T >

Definition at line 44 of file most_derived.h.

6.108.2 Member Typedef Documentation

6.108.2.1 result

```
template<typename T >
using TL::MostDerived< EmptyTypeList, T >::result = T
```

Definition at line 45 of file most_derived.h.

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

- TL/most_derived.h

6.109 TL::MostDerivedAndConstructible< type_list, T > Struct Template Reference

```
#include <most_derived_and_constructible.h>
```

Public Types

- using result = typename CheckMostDerivedAndConstructible< type_list, T, std::is_base_of< T, typename type_list::Head >::value &&std::is_constructible< typename type_list::Head >::value >::result

6.109.1 Detailed Description

template<class type_list, typename T>
struct TL::MostDerivedAndConstructible< type_list, T >

Finds the most derived and constructible child of T in type_list

Parameters

type_list	Template parameter
T	Template parameter

Returns

Parameter result, the most derived and constructible child of T in type_list

Definition at line 33 of file most_derived_and_constructible.h.

6.109.2 Member Typedef Documentation

6.109.2.1 result

```
template<class type_list , typename T >
using TL::MostDerivedAndConstructible< type_list, T >::result = typename CheckMostDerivedAndConstructible<
type_list, T, std::is_base_of<T, typename type_list::Head>::value && std::is_constructible<typename
type_list::Head>::value >::result
```

Definition at line 34 of file most_derived_and_constructible.h.

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

- TL/most_derived_and_constructible.h

6.110 TL::MostDerivedAndConstructible< EmptyTypeList, T > Struct Template Reference

```
#include <most_derived_and_constructible.h>
```

Public Types

- using result = T

6.110.1 Detailed Description

template<typename T>
struct TL::MostDerivedAndConstructible< EmptyTypeList, T >

Definition at line 43 of file most_derived_and_constructible.h.

6.110.2 Member Typedef Documentation

6.110.2.1 result

```
template<typename T >
using TL::MostDerivedAndConstructible< EmptyTypeList, T >::result = T
```

Definition at line 44 of file most_derived_and_constructible.h.

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

- TL/most_derived_and_constructible.h

6.111 TL::NoDuplicates< type_list > Struct Template Reference

```
#include <no_duplicates.h>
```

Public Types

- using result = TypeList< typename type_list::Head, typename NoDuplicates< typename RemoveAll< typename type_list::Tail, typename type_list::Head >::result >::result >

6.111.1 Detailed Description

template<class type_list>
struct TL::NoDuplicates< type_list >

Removes duplicated from TypeList type_list

Parameters

type_list	Template parameter

Returns

Parameter result, new TypeList without any duplicates

Definition at line 11 of file no_duplicates.h.

6.111.2 Member Typedef Documentation

6.111.2.1 result

```
template<class type_list >
using TL::NoDuplicates< type_list >::result = TypeList< typename type_list::Head, typename
NoDuplicates< typename RemoveAll< typename type_list::Tail, typename type_list::Head >↩
::result >::result >
```

Definition at line 12 of file no_duplicates.h.

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

- TL/no_duplicates.h

6.112 TL::NoDuplicates< EmptyTypeList > Struct Reference

```
#include <no_duplicates.h>
```

Public Types

- using result = EmptyTypeList

6.112.1 Detailed Description

See also

NoDuplicates

Definition at line 26 of file no_duplicates.h.

6.112.2 Member Typedef Documentation

6.112.2.1 result

```
using TL::NoDuplicates< EmptyTypeList >::result = EmptyTypeList
```

Definition at line 27 of file no_duplicates.h.

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

- TL/no_duplicates.h

6.113 NullType Struct Reference

```
#include <null_type.h>
```

6.113.1 Detailed Description

Represents nothing. If there is an absence of some template, it should be represented by NullType.

Definition at line 7 of file null_type.h.

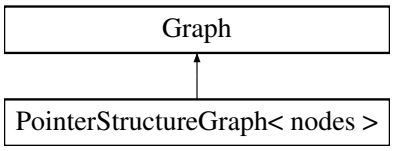
The documentation for this struct was generated from the following file:

- TL/null_type.h

6.114 PointerStructureGraph< nodes > Struct Template Reference

`#include <pointer_structure_graph.h>`

Inheritance diagram for PointerStructureGraph< nodes >:



Classes

- struct ConvertTo

Public Types

- using nodes_ = nodes

  All accounted vertexes in this graph.

Static Public Attributes

- constexpr static GraphType TYPE = POINTER_STRUCTURE

6.114.1 Detailed Description

template<class nodes>
struct PointerStructureGraph< nodes >

Represents graph as a structure with pointers Every vertex must be contained within node. Node must have a TypeList "children", which is a TypeList of Edges, showing who can be reached from this vertex. Also node must have a field "vertex" ~— the vertex this node contains.

Parameters

nodes	Template parameter, nodes in this graph

Definition at line 23 of file pointer_structure_graph.h.

6.114.2 Member Typedef Documentation

6.114.2.1 nodes_

```
template<class nodes >
using PointerStructureGraph< nodes >::nodes_ = nodes
```

All accounted vertexes in this graph.

Definition at line 27 of file pointer_structure_graph.h.

6.114.3 Member Data Documentation

6.114.3.1 TYPE

```
template<class nodes >
constexpr static GraphType PointerStructureGraph< nodes >::TYPE = POINTER_STRUCTURE  [static],
[constexpr]
```

Definition at line 24 of file pointer_structure_graph.h.

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

- graph/graphs/pointer_structure_graph.h

6.115 PointerStructureNode< vertex_, children_ > Struct Template Reference

```
#include <pointer_structure_node.h>
```

Public Types

- using vertex = vertex_
- using children = children_

6.115.1 Detailed Description

template<class vertex_, class children_>
struct PointerStructureNode< vertex_, children_ >

Default version of a suitable class for PointerStructureGraph. It's not necessary to use this one. In fact, it's encouraged to make your objects suitable to PointerStructureGraph. It can be done by adding field "children" to every vertex in the graph.

Parameters

*vertex_*	Template parameter, vertex that this node represents
*children_*	Template parameter, TypeList of Edges, showing who can be reached from this vertex.

Definition at line 13 of file pointer_structure_node.h.

6.115.2 Member Typedef Documentation

6.115.2.1 children

```
template<class vertex_ , class children_ >
using PointerStructureNode< vertex_, children_ >::children = children_
```

Definition at line 16 of file pointer_structure_node.h.

6.115.2.2 vertex

```
template<class vertex_ , class children_ >
using PointerStructureNode< vertex_, children_ >::vertex = vertex_
```

Definition at line 15 of file pointer_structure_node.h.

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

- graph/graphs/pointer_structure_node.h

6.116 ProcessVertex< id, graph, vertex > Struct Template Reference

6.116.1 Detailed Description

template<int id, class graph, typename vertex>
struct ProcessVertex< id, graph, vertex >

Processes vertex in a given graph. id represents unique key to implementation of this struct. Id is used to make specific implementation for ForEach function.

See also

ForEach

Parameters

id	Template parameter, unique key for implementations of this struct.
graph	Template parameter
vertex	Template parameter

Returns

Implementation-defined.

Definition at line 14 of file process_vertex.h.

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

- graph/process_vertex.h

6.117 ProcessVertex< 34, graph, vertex > Struct Template Reference

Static Public Attributes

- constexpr static int number = vertex::value
- constexpr static int edge_count = TL::Size<typename graph::edges_>::size
- constexpr static bool result = number > edge_count

6.117.1 Detailed Description

template<class graph, typename vertex>
struct ProcessVertex< 34, graph, vertex >

Definition at line 15 of file vertex_stream_example.cpp.

6.117.2 Member Data Documentation

6.117.2.1 edge_count

```
template<class graph , typename vertex >
constexpr static int ProcessVertex< 34, graph, vertex >::edge_count = TL::Size<typename graph↩
::edges_>::size  [static], [constexpr]
```

Definition at line 17 of file vertex_stream_example.cpp.

6.117.2.2 number

```
template<class graph , typename vertex >
constexpr static int ProcessVertex< 34, graph, vertex >::number = vertex::value  [static],
[constexpr]
```

Definition at line 16 of file vertex_stream_example.cpp.

6.117.2.3 result

```
template<class graph , typename vertex >
constexpr static bool ProcessVertex< 34, graph, vertex >::result = number > edge_count  [static],
[constexpr]
```

Definition at line 18 of file vertex_stream_example.cpp.

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

- graph/examples/vertex_stream_example.cpp

6.118 ProcessVertex< 42, graph, vertex > Struct Template Reference

Static Public Attributes

- constexpr static int result = vertex::value

6.118.1 Detailed Description

template<class graph, typename vertex>
struct ProcessVertex< 42, graph, vertex >

Definition at line 23 of file vertex_stream_example.cpp.

6.118.2 Member Data Documentation

6.118.2.1 result

```
template<class graph , typename vertex >
constexpr static int ProcessVertex< 42, graph, vertex >::result = vertex::value  [static],
[constexpr]
```

Definition at line 24 of file vertex_stream_example.cpp.

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

- graph/examples/vertex_stream_example.cpp

6.119 TL::Remove< type_list, T > Struct Template Reference

`#include <remove.h>`

Public Types

- using result = TypeList< typename type_list::Head, typename Remove< typename type_list::Tail, T >::result >

6.119.1 Detailed Description

template<class type_list, typename T>
struct TL::Remove< type_list, T >

Removes first ocurrence of T in type_list

Parameters

type_list	Template parameter
T	Template parameter

Returns

Parameter result, new TypeList without first ocurrence of T

Definition at line 13 of file remove.h.

6.119.2 Member Typedef Documentation

6.119.2.1 result

```
template<class type_list , typename T >
using TL::Remove< type_list, T >::result = TypeList<typename type_list::Head, typename Remove<typename
type_list::Tail, T>::result>
```

Definition at line 14 of file remove.h.

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

- TL/remove.h

6.120 TL::Remove< EmptyTypeList, T > Struct Template Reference

`#include <remove.h>`

Public Types

- using result = EmptyTypeList

6.120.1 Detailed Description

template<typename T>
struct TL::Remove< EmptyTypeList, T >

See also

Remove

Definition at line 30 of file remove.h.

6.120.2 Member Typedef Documentation

6.120.2.1 result

```
template<typename T >
using TL::Remove< EmptyTypeList, T >::result = EmptyTypeList
```

Definition at line 31 of file remove.h.

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

- TL/remove.h

6.121 TL::Remove< type_list, typename type_list::Head > Struct Template Reference

```
#include <remove.h>
```

Public Types

- using result = typename type_list::Tail

6.121.1 Detailed Description

template<class type_list>
struct TL::Remove< type_list, typename type_list::Head >

See also

Remove

Definition at line 22 of file remove.h.

6.121.2 Member Typedef Documentation

6.121.2.1 result

```
template<class type_list >
using TL::Remove< type_list, typename type_list::Head >::result = typename type_list::Tail
```

Definition at line 23 of file remove.h.

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

- TL/remove.h

6.122 TL::RemoveAll< type_list, T > Struct Template Reference

```
#include <remove.h>
```

Public Types

- using result = TypeList< typename type_list::Head, typename RemoveAll< typename type_list::Tail, T >::result >

6.122.1 Detailed Description

template<class type_list, class T>
struct TL::RemoveAll< type_list, T >

Removes all ocurrences of T in type_list

Parameters

type_list	Template parameter
T	Template parameter

Returns

Parameter result, new TypeList without ocurrences of T

Definition at line 41 of file remove.h.

6.122.2 Member Typedef Documentation

6.122.2.1 result

```
template<class type_list , class T >
using TL::RemoveAll< type_list, T >::result = TypeList<typename type_list::Head, typename
RemoveAll< typename type_list::Tail, T >::result >
```

Definition at line 42 of file remove.h.

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

- TL/remove.h

6.123 TL::RemoveAll< type_list, typename type_list::Head > Struct Template Reference

```
#include <remove.h>
```

Public Types

- using result = typename RemoveAll< typename type_list::Tail, typename type_list::Head >::result

6.123.1 Detailed Description

template<class type_list>
struct TL::RemoveAll< type_list, typename type_list::Head >

See also

RemoveAll

Definition at line 53 of file remove.h.

6.123.2 Member Typedef Documentation

6.123.2.1 result

```
template<class type_list >
using TL::RemoveAll< type_list, typename type_list::Head >::result = typename RemoveAll<
typename type_list::Tail, typename type_list::Head >::result
```

Definition at line 54 of file remove.h.

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

- TL/remove.h

6.124 TL::Replace< T, ind, Arg, Args > Struct Template Reference

`#include <replace.h>`

6.124.1 Detailed Description

template<typename T, size_t ind, class Arg, class ... Args>
struct TL::Replace< T, ind, Arg, Args >

Replaces typename on a specific position in TypeList

Parameters

T	Typename that will be on a specific position in TypeList
ind	Number of this position
TypeList<Arg,Args...>	This TypeList

Returns

Parameter result, new type list with typename added to position ind

Definition at line 14 of file replace.h.

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

- TL/replace.h

6.125 TL::Replace< T, 0, TypeList< Arg, Args... > > Struct Template Reference

`#include <replace.h>`

Public Types

- using result = TypeList< T, Args... >

6.125.1 Detailed Description

template<typename T, class Arg, class ... Args>
struct TL::Replace< T, 0, TypeList< Arg, Args... > >

See also

Replace

Definition at line 34 of file replace.h.

6.125.2 Member Typedef Documentation

6.125.2.1 result

```
template<typename T , class Arg , class ...  Args>
using TL::Replace< T, 0, TypeList< Arg, Args...  > >::result = TypeList<T, Args...>
```

Definition at line 35 of file replace.h.

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

- TL/replace.h

6.126 TL::Replace< T, ind, TypeList< Arg, Args... > > Struct Template Reference

```
#include <replace.h>
```

Public Types

- using end = typename Replace< T, ind - 1, TypeList< Args... > >::result
- using result = typename Add< Arg, 0, end >::result

6.126.1 Detailed Description

template<typename T, size_t ind, class Arg, class ... Args>
struct TL::Replace< T, ind, TypeList< Arg, Args... > >

See also

Replace

Definition at line 20 of file replace.h.

6.126.2 Member Typedef Documentation

6.126.2.1 end

```
template<typename T , size_t ind, class Arg , class ...  Args>
using TL::Replace< T, ind, TypeList< Arg, Args...  > >::end = typename Replace< T, ind - 1,
TypeList<Args...> >::result
```

Definition at line 21 of file replace.h.

6.126.2.2 result

```
template<typename T , size_t ind, class Arg , class ...  Args>
using TL::Replace< T, ind, TypeList< Arg, Args...  > >::result = typename Add<Arg, 0, end>↩
::result
```

Definition at line 27 of file replace.h.

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

- TL/replace.h

6.127 TL::Reverse< type_list > Struct Template Reference

```
#include <reverse.h>
```

Classes

- struct IterateThroughElements
- struct IterateThroughElements< EmptyTypeList, cur_result >

Public Types

- using result = typename IterateThroughElements< type_list, EmptyTypeList >::result

6.127.1 Detailed Description

template<class type_list>
struct TL::Reverse< type_list >

Reverses type_list

Parameters

type_list	Template parameter

Returns

Parameter result, reversed type_list

Definition at line 12 of file reverse.h.

6.127.2 Member Typedef Documentation

6.127.2.1 result

```
template<class type_list >
using TL::Reverse< type_list >::result = typename IterateThroughElements<type_list, EmptyTypeList>↩
::result
```

Definition at line 30 of file reverse.h.

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

- TL/reverse.h

6.128 TL::Size< type_list > Struct Template Reference

```
#include <size.h>
```

Static Public Attributes

- constexpr static size_t size = 1 + Size<typename type_list::Tail>::size

6.128.1 Detailed Description

template<class type_list>
struct TL::Size< type_list >

Gets length of a TypeList

Parameters

TypeList	Template parameter

Returns

Parameter size, amount of elements in TypeList

Definition at line 12 of file size.h.

6.128.2 Member Data Documentation

6.128.2.1 size

```
template<class type_list >
constexpr static size_t TL::Size< type_list >::size = 1 + Size<typename type_list::Tail>↩
::size  [static], [constexpr]
```

Definition at line 14 of file size.h.

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

- TL/size.h

6.129 TL::Size< EmptyTypeList > Struct Reference

```
#include <size.h>
```

Static Public Attributes

- constexpr static size_t size = 0

6.129.1 Detailed Description

See also

Size

Definition at line 21 of file size.h.

6.129.2 Member Data Documentation

6.129.2.1 size

```
constexpr static size_t TL::Size< EmptyTypeList >::size = 0  [static], [constexpr]
```

Definition at line 22 of file size.h.

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

- TL/size.h

6.130 TL::TypeAt< type_list, ind > Struct Template Reference

`#include <type_at.h>`

Public Types

- using value = typename TypeAt< typename type_list::Tail, ind - 1 >::value

6.130.1 Detailed Description

template<class type_list, size_t ind>
struct TL::TypeAt< type_list, ind >

Get class at specific index of TypeList

Parameters

type_list	Template parameter, where required class is located
ind	Template parameter, shows position where required class is located

Returns

Parameter value, class at a specific index of TypeList

Definition at line 15 of file type_at.h.

6.130.2 Member Typedef Documentation

6.130.2.1 value

```
template<class type_list , size_t ind>
using TL::TypeAt< type_list, ind >::value = typename TypeAt<typename type_list::Tail, ind -
1>::value
```

Definition at line 18 of file type_at.h.

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

- TL/type_at.h

6.131 TL::TypeAt< type_list, 0 > Struct Template Reference

`#include <type_at.h>`

Public Types

- using value = typename type_list::Head

6.131.1 Detailed Description

template<class type_list>
struct TL::TypeAt< type_list, 0 >

See also

TypeAt

Definition at line 25 of file type_at.h.

6.131.2 Member Typedef Documentation

6.131.2.1 value

```
template<class type_list >
using TL::TypeAt< type_list, 0 >::value = typename type_list::Head
```

Definition at line 26 of file type_at.h.

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

- TL/type_at.h

6.132 TypeList< Args > Struct Template Reference

```
#include <type_list.h>
```

Public Types

- using Head = NullType
- using Tail = TypeList<>

6.132.1 Detailed Description

template<typename ... Args>
struct TypeList< Args >

See also

TypeList<H, T...>

Definition at line 9 of file type_list.h.

6.132.2 Member Typedef Documentation

6.132.2.1 Head

```
template<typename ...  Args>
using TypeList< Args >::Head = NullType
```

Definition at line 10 of file type_list.h.

6.132.2.2 Tail

```
template<typename ...  Args>
using TypeList< Args >::Tail = TypeList<>
```

Definition at line 11 of file type_list.h.

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

- TL/type_list.h

6.133 TypeList< H, T... > Struct Template Reference

```
#include <type_list.h>
```

Public Types

- using Head = H
  First type in a type list.
- using Tail = TypeList< T... >
  TypeList of other types.

6.133.1 Detailed Description

template<typename H, typename ... T>
struct TypeList< H, T... >

Represents a list of various types

Parameters

H	Template parameter, first object in a type list
T	Template parameter, other objects in a type list

Definition at line 35 of file type_list.h.

6.133.2 Member Typedef Documentation

6.133.2.1 Head

```
template<typename H , typename ...  T>
using TypeList< H, T...  >::Head = H
```

First type in a type list.

Definition at line 36 of file type_list.h.

6.133.2.2 Tail

```
template<typename H , typename ...  T>
using TypeList< H, T...  >::Tail = TypeList<T...>
```

TypeList of other types.

Definition at line 37 of file type_list.h.

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

- TL/type_list.h

6.134 TypeList< T > Struct Template Reference

```
#include <type_list.h>
```

Public Types

- using Head = T
- using Tail = EmptyTypeList

6.134.1 Detailed Description

template<typename T>
struct TypeList< T >

See also

TypeList<H, T...>

Definition at line 24 of file type_list.h.

6.134.2 Member Typedef Documentation

6.134.2.1 Head

```
template<typename T >
using TypeList< T >::Head = T
```

Definition at line 25 of file type_list.h.

6.134.2.2 Tail

```
template<typename T >
using TypeList< T >::Tail = EmptyTypeList
```

Definition at line 26 of file type_list.h.

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

- TL/type_list.h

Chapter 7

File Documentation

7.1 Debug/CodeAnalysisResultManifest.txt File Reference

7.2 Debug/library.vcxproj.FileListAbsolute.txt File Reference

7.3 functor.h File Reference

```
#include <cassert>
#include <memory>
```

Classes

- class Functor< ResultType(ArgTypes...)>

7.4 graph/class.h File Reference

Classes

- struct Class< value_ >

7.5 graph/convert/convert_graph.h File Reference

```
#include "../graphs/graph_type.h"
```

Classes

- struct ConvertGraph< type, type, graph >

7.6 graph/convert/convert_to_adjacency_list.h File Reference

```
#include "convert_graph.h"
#include "../../TL/type_list.h"
#include "../../TL/fill_type_list_with_object.h"
#include "../../TL/size.h"
#include "../GLib/add_edge.h"
#include "../graphs/adjacency_list_graph.h"
#include "../graphs/adjacency_matrix_graph.h"
#include "../graphs/edge_list_graph.h"
#include "../graphs/pointer_structure_graph.h"
#include "../graphs/graph_type.h"
```

Classes

- struct ConvertGraph< EDGE_LIST, ADJACENCY_LIST, graph >
- struct ConvertGraph< EDGE_LIST, ADJACENCY_LIST, graph >::IterateThroughEdges< edge_list >
- struct ConvertGraph< EDGE_LIST, ADJACENCY_LIST, graph >::IterateThroughEdges< EmptyTypeList >

7.7 graph/convert/convert_to_adjacency_matrix.h File Reference

```
#include "../../TL/fill_type_list_with_object.h"
#include "../../TL/index_of.h"
#include "../class.h"
#include "../graphs/adjacency_matrix_graph.h"
#include "../objects.h"
```

Classes

- struct ConvertGraph< EDGE_LIST, ADJACENCY_MATRIX, graph >
- struct ConvertGraph< EDGE_LIST, ADJACENCY_MATRIX, graph >::IterateThroughEdges< cur_edges >
- struct ConvertGraph< EDGE_LIST, ADJACENCY_MATRIX, graph >::IterateThroughEdges< EmptyTypeList >

7.8 graph/convert/convert_to_edge_list.h File Reference

```
#include "convert_graph.h"
#include "../../TL/add.h"
#include "../../TL/concatenate.h"
#include "../../TL/size.h"
#include "../../TL/type_list.h"
#include "../GLib/add_edge.h"
#include "../graphs/adjacency_list_graph.h"
#include "../graphs/adjacency_matrix_graph.h"
#include "../graphs/edge_list_graph.h"
#include "../graphs/pointer_structure_graph.h"
#include "../graphs/graph_type.h"
#include "../edge.h"
#include "../objects.h"
```

Classes

- struct ConvertGraph< POINTER_STRUCTURE, EDGE_LIST, graph >
- struct ConvertGraph< POINTER_STRUCTURE, EDGE_LIST, graph >::IterateThroughNodes< cur_nodes >
- struct ConvertGraph< POINTER_STRUCTURE, EDGE_LIST, graph >::IterateThroughNodes< EmptyTypeList >
- struct ConvertGraph< ADJACENCY_MATRIX, EDGE_LIST, graph >
- struct ConvertGraph< ADJACENCY_MATRIX, EDGE_LIST, graph >::IterateThroughMatrix< cur_index >
- struct ConvertGraph< ADJACENCY_MATRIX, EDGE_LIST, graph >::IterateThroughMatrix<-1 >

7.9 graph/convert/convert_to_pointer_structure.h File Reference

```
#include "convert_graph.h"
#include "../../TL/add.h"
#include "../../TL/fill_type_list_with_object.h"
#include "../../TL/size.h"
#include "../../TL/type_list.h"
#include "../GLib/add_edge.h"
#include "../graphs/adjacency_list_graph.h"
#include "../graphs/adjacency_matrix_graph.h"
#include "../graphs/edge_list_graph.h"
#include "../graphs/pointer_structure_graph.h"
#include "../graphs/pointer_structure_node.h"
#include "../graphs/graph_type.h"
```

Classes

- struct ConvertGraph< ADJACENCY_LIST, POINTER_STRUCTURE, graph >
- struct ConvertGraph< ADJACENCY_LIST, POINTER_STRUCTURE, graph >::MakePointerStructureGraph< current_vertexes
- struct ConvertGraph< ADJACENCY_LIST, POINTER_STRUCTURE, graph >::MakePointerStructureGraph< EmptyTypeList,
- struct ConvertGraph< EDGE_LIST, POINTER_STRUCTURE, graph >

7.10 graph/edge.h File Reference

```
#include "../TL/null_type.h"
```

Classes

- struct Edge< from_, to_, weight_ >

7.11 graph/examples/graph_examples.cpp File Reference

```
#include "../graphs/adjacency_list_graph.h"
#include "../graphs/adjacency_matrix_graph.h"
#include "../graphs/edge_list_graph.h"
#include "../graphs/pointer_structure_graph.h"
#include "../edge.h"
#include "../objects.h"
#include "../graphs/pointer_structure_node.h"
```

Functions

- int main ()

7.11.1 Function Documentation

7.11.1.1 main()

```
int main ( )
```

Definition at line 12 of file graph_examples.cpp.

7.12 graph/examples/vertex_stream_example.cpp File Reference

```
#include <iostream>
#include "../graphs/adjacency_list_graph.h"
#include "../graphs/edge_list_graph.h"
#include "../GLib/filter.h"
#include "../GLib/for_each.h"
#include "../edge.h"
#include "../process_vertex.h"
#include "../objects.h"
```

Classes

- struct ProcessVertex< 34, graph, vertex >
- struct ProcessVertex< 42, graph, vertex >

Functions

- int main ()

7.12.1 Function Documentation

7.12.1.1 main()

```
int main ( )
```

Definition at line 29 of file vertex_stream_example.cpp.

7.13 graph/GLib/add_edge.h File Reference

```
#include "../../TL/replace.h"
#include "../../TL/type_list.h"
#include "../../TL/type_at.h"
#include "../../TL/index_of.h"
#include "../graphs/graph_type.h"
#include "../graphs/adjacency_list_graph.h"
```

Classes

- struct GLib::AddEdge< ADJACENCY_LIST, graph, edge >

Namespaces

- GLib

7.14 graph/GLib/dfs.h File Reference

```
#include "../../TL/contains.h"
#include "../graphs/pointer_structure_graph.h"
```

Classes

- struct GLib::DFS< cur_node, graph, visited_nodes >
- struct GLib::DFS< cur_node, graph, visited_nodes >::IterateThroughChildren< cur_children, cur_visited >
- struct GLib::DFS< cur_node, graph, visited_nodes >::IterateThroughChildren< EmptyTypeList, cur_unvisited >

Namespaces

- GLib

7.15 graph/GLib/filter.h File Reference

```
#include <type_traits>
#include "../process_vertex.h"
#include "../../TL/add.h"
```

Classes

- struct GLib::Filter< id, graph, vertexes >
- struct GLib::Filter< id, graph, EmptyTypeList >

Namespaces

- GLib

7.16 graph/GLib/find_node_by_vertex.h File Reference

```
#include "../graphs/pointer_structure_graph.h"
```

Classes

- struct GLib::FindNodeByVertex< vertex, graph >
- struct GLib::FindNodeByVertex< vertex, graph >::IterateThroughNodes< cur_nodes >
- struct GLib::FindNodeByVertex< vertex, graph >::IterateThroughNodes< EmptyTypeList >
- struct GLib::FindNodeByVertex< vertex, EmptyTypeList >

Namespaces

- GLib

7.17 graph/GLib/find_path.h File Reference

```
#include <type_traits>
#include "../../TL/concatenate.h"
#include "../../TL/reverse.h"
#include "find_node_by_vertex.h"
#include "dfs.h"
#include "../graphs/convert_graph.h"
#include "../graphs/convert_to_pointer_structure.h"
```

Classes

- struct GLib::FindPath< graph_raw, start, finish >
- struct GLib::FindPath< graph_raw, start, finish >::IterateThroughEdges< cur_edges, wanted_node >
- struct GLib::FindPath< graph_raw, start, finish >::IterateThroughEdges< EmptyTypeList, wanted_node >

Namespaces

- GLib

7.18 graph/GLib/for_each.h File Reference

```
#include "../process_vertex.h"
#include "../../TL/add.h"
```

Classes

- struct GLib::ForEach< id, graph, vertexes >
- struct GLib::ForEach< id, graph, EmptyTypeList >

Namespaces

- GLib

7.19 graph/GLib/get_nodes_from_roots.h File Reference

```
#include "dfs.h"
```

Classes

- struct GLib::GetNodesFromRoots< nodes, graph >
- struct GLib::GetNodesFromRoots< EmptyTypeList, graph >

Namespaces

- GLib

7.20 graph/GLib/get_reached_vertexes.h File Reference

```
#include <type_traits>
#include "find_node_by_vertex.h"
#include "dfs.h"
```

Classes

- struct GLib::GetReachedVertexes< graph, start >
- struct GLib::GetReachedVertexes< graph, start >::IterateThroughEdges< cur_edges >
- struct GLib::GetReachedVertexes< graph, start >::IterateThroughEdges< EmptyTypeList >

Namespaces

- GLib

7.21 graph/GLib/map_indexes_to_vertexes.h File Reference

7.22 graph/graphs/adjacency_list_graph.h File Reference

```
#include "graph.h"
#include "../../TL/add.h"
#include "../../TL/contains.h"
#include "../../TL/index_of.h"
#include "../../TL/is_type_list.h"
#include "../../TL/size.h"
#include "../../TL/type_at.h"
#include "../../TL/type_list.h"
#include "../convert/convert_graph.h"
```

Classes

- struct AdjacencyListGraph< nodes, adjacency_list >
- struct AdjacencyListGraph< nodes, adjacency_list >::ConvertTo< type >

7.23 graph/graphs/adjacency_matrix_graph.h File Reference

```
#include "graph.h"
#include "../../TL/is_type_list.h"
#include "../../TL/fill_type_list_with_object.h"
#include "../GLib/add_edge.h"
#include "../convert/convert_graph.h"
```

Classes

- struct AdjacencyMatrixGraph< vertexes, matrix >
- struct AdjacencyMatrixGraph< vertexes, matrix >::ConvertTo< type >

7.24 graph/graphs/edge_list_graph.h File Reference

```
#include "graph.h"
#include "../../TL/is_type_list.h"
#include "../../TL/fill_type_list_with_object.h"
#include "../GLib/add_edge.h"
#include "../convert/convert_graph.h"
```

Classes

- struct EdgeListGraph< nodes, edge_list >
- struct EdgeListGraph< nodes, edge_list >::ConvertTo< type >

7.25 graph/graphs/graph.h File Reference

```
#include "graph_type.h"
```

Classes

- struct Graph

7.26 graph/graphs/graph_type.h File Reference

Enumerations

- enum GraphType { ADJACENCY_MATRIX, ADJACENCY_LIST, EDGE_LIST, POINTER_STRUCTURE }

7.26.1 Enumeration Type Documentation

7.26.1.1 GraphType

```
enum GraphType
```

Types, by which graph can be created. For more details, see corresponding file.

See also

AdjacencyMatrixGraph

AdjacencyListGraph

EdgeListGraph

PointerStructureGraph

Enumerator

ADJACENCY_MATRIX	Graph is represented as an adjacency matrix of booleans.
ADJACENCY_LIST	Graph is represented as an adjacency list (TypeList of TypeLists of edges).
EDGE_LIST	Graph is represented by a collection of edges.
POINTER_STRUCTURE	Graph is represented by a pointer structure.

Definition at line 11 of file graph_type.h.

7.27 graph/graphs/pointer_structure_graph.h File Reference

```
#include "graph.h"
```

```
#include "../../TL/concatenate.h"
#include "../../TL/is_type_list.h"
#include "../../TL/remove.h"
#include "../GLib/find_node_by_vertex.h"
#include "pointer_structure_node.h"
#include "../convert/convert_graph.h"
```

Classes

- struct PointerStructureGraph< nodes >
- struct PointerStructureGraph< nodes >::ConvertTo< type >

7.28 graph/graphs/pointer_structure_node.h File Reference

```
#include "../../TL/is_type_list.h"
```

Classes

- struct PointerStructureNode< vertex_, children_ >

7.29 graph/objects.h File Reference

Classes

- struct Objects::Integer< integer >
- struct Objects::Boolean< boolean >

Namespaces

- Objects

7.30 graph/process_vertex.h File Reference

7.31 TL/add.h File Reference

```
#include "type_list.h"
```

Classes

- struct TL::Add< T, ind, TypeList< Arg, Args... > >
- struct TL::Add< T, 0, TypeList< Arg, Args... > >
- struct TL::Add< T, 0, TypeList< Args... > >

Namespaces

- TL

7.32 TL/concatenate.h File Reference

```
#include "add.h"
#include "reverse.h"
#include "size.h"
#include "is_type_list.h"
```

Classes

- struct TL::Concatenate< front, back >
- struct TL::Concatenate< front, back >::IterateThroughReversedFront< elements, current >
- struct TL::Concatenate< front, back >::IterateThroughReversedFront< EmptyTypeList, current >

Namespaces

- TL

7.33 TL/contains.h File Reference

```
#include "index_of.h"
#include "type_list.h"
```

Classes

- struct TL::Contains< type_list, T >

Namespaces

- TL

7.34 TL/contains_constructible_parent.h File Reference

```
#include <type_traits>
#include "type_list.h"
```

Classes

- struct CheckContainsConstructibleParent< type_list, T, is_parent >
- struct CheckContainsConstructibleParent< type_list, T, false >
- struct CheckContainsConstructibleParent< type_list, T, true >
- struct TL::ContainsConstructibleParent< type_list, T >
- struct TL::ContainsConstructibleParent< EmptyTypeList, T >

Namespaces

- TL

7.35 TL/contains_parent.h File Reference

```
#include <type_traits>
#include "type_list.h"
```

Classes

- struct CheckContainsParent< type_list, T, is_parent >
- struct CheckContainsParent< type_list, T, false >
- struct CheckContainsParent< type_list, T, true >
- struct TL::ContainsParent< type_list, T >
- struct TL::ContainsParent< EmptyTypeList, T >

Namespaces

- TL

7.36 TL/fill_type_list_with_object.h File Reference

```
#include "add.h"
#include "type_list.h"
```

Classes

- struct TL::FillTypeListWithObject< obj, n >
- struct TL::FillTypeListWithObject< obj, 0 >

Namespaces

- TL

7.37 TL/find_parent_type_list.h File Reference

```
#include "is_base_of.h"
#include "type_list.h"
```

Classes

- struct CheckFindParentTypeList< contains_class, T, type_list, type_lists >
- struct CheckFindParentTypeList< true, T, type_list, type_lists... >
- struct CheckFindParentTypeList< false, T, type_list, type_lists... >
- struct TL::FindParentTypeList< T, type_list, type_lists >

Namespaces

- TL

7.38 TL/find_type_list_by_class.h File Reference

```
#include "contains.h"
#include "type_list.h"
```

Classes

- struct CheckFindTypeListByClass< contains_class, T, type_list, type_lists >
- struct CheckFindTypeListByClass< true, T, type_list, type_lists... >
- struct CheckFindTypeListByClass< false, T, type_list, type_lists... >
- struct TL::FindTypeListByClass< T, type_list, type_lists >

Namespaces

- TL

7.39 TL/has_derived_and_constructible.h File Reference

```
#include <type_traits>
#include "type_list.h"
```

Classes

- struct CheckHasDerivedAndConstructible< type_list, T, is_head_parent_of_T >
- struct CheckHasDerivedAndConstructible< type_list, T, true >
- struct CheckHasDerivedAndConstructible< type_list, T, false >
- struct TL::HasDerivedAndConstructible< type_list, T >
- struct TL::HasDerivedAndConstructible< EmptyTypeList, T >

Namespaces

- TL

7.40 TL/index_of.h File Reference

```
#include <cstdint>
#include "is_type_list.h"
#include "type_list.h"
```

Classes

- struct TL::IndexOf< type_list, T >
- struct TL::IndexOf< type_list, typename type_list::Head >
- struct TL::IndexOf< EmptyTypeList, T >

Namespaces

- TL

7.41 TL/is_base_of.h File Reference

```
#include <type_traits>
#include "contains_parent.h"
#include "type_list.h"
```

Classes

- struct CheckIsBaseOf< has_parent, parent, derived >
- struct CheckIsBaseOf< false, parent, derived >
- struct CheckIsBaseOf< true, parent, derived >
- struct TL::IsBaseOf< parent, derived >
- struct TL::IsBaseOf< parent, EmptyTypeList >
- struct TL::IsBaseOf< EmptyTypeList, derived >
- struct TL::IsBaseOf< EmptyTypeList, EmptyTypeList >

Namespaces

- TL

7.42 TL/is_type_list.h File Reference

```
#include <type_traits>
#include "type_list.h"
```

Classes

- struct TL::IsTypeList< T >
- struct TL::IsTypeList< TypeList< Args... > >

Namespaces

- TL

7.43 TL/most_derived.h File Reference

```
#include <type_traits>
#include "type_list.h"
```

Classes

- struct CheckMostDerived< type_list, T, is_head_parent_of_T >
- struct CheckMostDerived< type_list, T, true >
- struct CheckMostDerived< type_list, T, false >
- struct TL::MostDerived< type_list, T >
- struct TL::MostDerived< EmptyTypeList, T >

Namespaces

- TL

7.44 TL/most_derived_and_constructible.h File Reference

```
#include <type_traits>
#include "type_list.h"
```

Classes

- struct CheckMostDerivedAndConstructible< type_list, T, is_head_parent_of_T >
- struct CheckMostDerivedAndConstructible< type_list, T, true >
- struct CheckMostDerivedAndConstructible< type_list, T, false >
- struct TL::MostDerivedAndConstructible< type_list, T >
- struct TL::MostDerivedAndConstructible< EmptyTypeList, T >

Namespaces

- TL

7.45 TL/no_duplicates.h File Reference

```
#include "remove.h"
#include "type_list.h"
```

Classes

- struct TL::NoDuplicates< type_list >
- struct TL::NoDuplicates< EmptyTypeList >

Namespaces

- TL

7.46 TL/null_type.h File Reference

Classes

- struct NullType

7.47 TL/remove.h File Reference

```
#include "type_list.h"
```

Classes

- struct TL::Remove< type_list, T >
- struct TL::Remove< type_list, typename type_list::Head >
- struct TL::Remove< EmptyTypeList, T >
- struct TL::RemoveAll< type_list, T >
- struct TL::RemoveAll< type_list, typename type_list::Head >

Namespaces

- TL

7.48 TL/replace.h File Reference

```
#include "add.h"
#include "type_list.h"
```

Classes

- struct TL::Replace< T, ind, Arg, Args >
- struct TL::Replace< T, ind, TypeList< Arg, Args... > >
- struct TL::Replace< T, 0, TypeList< Arg, Args... > >

Namespaces

- TL

7.49 TL/reverse.h File Reference

```
#include "add.h"
```

Classes

- struct TL::Reverse< type_list >
- struct TL::Reverse< type_list >::IterateThroughElements< cur_type_list, cur_result >
- struct TL::Reverse< type_list >::IterateThroughElements< EmptyTypeList, cur_result >

Namespaces

- TL

7.50 TL/size.h File Reference

```
#include "is_type_list.h"
#include "type_list.h"
```

Classes

- struct TL::Size< type_list >
- struct TL::Size< EmptyTypeList >

Namespaces

- TL

7.51 TL/type_at.h File Reference

```
#include "is_type_list.h"
#include "size.h"
#include "type_list.h"
```

Classes

- struct TL::TypeAt< type_list, ind >
- struct TL::TypeAt< type_list, 0 >

Namespaces

- TL

7.52 TL/type_list.h File Reference

```
#include "null_type.h"
```

Classes

- struct TypeList< Args >
- struct TypeList< T >
- struct TypeList< H, T... >

Namespaces

- TL

Typedefs

- using EmptyTypeList = TypeList<>
- template<typename ... Args>
  using Typelist = TypeList< Args... >

7.52.1 Typedef Documentation

7.52.1.1 EmptyTypeList

```
using EmptyTypeList = TypeList<>
```

Represents TypeList with no data

See also

TypeList

Definition at line 18 of file type_list.h.

7.52.1.2 Typelist

```
template<typename ...  Args>
using Typelist = TypeList<Args...>
```

See also

TypeList<H, T...>

Definition at line 44 of file type_list.h.

Chapter 8

Example Documentation

8.1 get_nodes_from_roots.h

An example of how to use DFS.

8.2 get_reached_vertexes.h

An example of how to use DFS.

8.3 graph_examples.cpp

An example of how graph be created.

8.4 vertex_stream_example.cpp

An example of how to use Filter.

Index