My Project

Generated by Doxygen 1.8.20

Chapter 1

Namespace Index

1.1 Namespace List

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

Namespace Documentation

3.1 FunctorType Namespace Reference

Typedefs

- template<typename ... InputArgs>
  using **Consumer** = Functor< void(InputArgs...)>

3.1.1 Detailed Description

Represents different types of functors.

See also

Package java.util.function in Java

3.2 MakeGraph Namespace Reference

Classes

- struct From
- struct From< ADJACENCY_LIST, vertexes, adjacency_list >
- struct From< ADJACENCY_MATRIX, vertexes, adjacency_matrix >
- struct From< EDGE_LIST, vertexes, edge_list >
- struct From< EDGE_LIST, vertexes, EmptyTypeList >

3.2.1 Detailed Description

Methods to create graph from types, defined in

See also

GraphType

3.3 Objects Namespace Reference

Classes

- struct Boolean
- struct Integer

3.3.1 Detailed Description

Represents class holders of different objects

3.4 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 Contains
- struct ContainsConstructibleParent
- struct ContainsConstructibleParent< EmptyTypeList, T >
- struct ContainsParent
- struct ContainsParent< EmptyTypeList, T >
- struct FindParentTypeList
- struct FindTypeListByClass
- struct GenerateTypeLists
- struct GenerateTypeLists< 0 >
- 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 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 Size
- struct Size< EmptyTypeList >
- struct TypeAt
- struct TypeAt< type_list, 0 >

3.4.1 Detailed Description

Represents functions (as structs) for working with TypeList

Chapter 4

Class Documentation

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

4.1.1 Detailed Description

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

Returns

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

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

- TL/add.h

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

Public Types

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

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

- TL/add.h

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

Public Types

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

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

- TL/add.h

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

Public Types

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

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

- TL/add.h

4.5 GLib::AddEdge< 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** = Graph< typename graph::vertexes_, new_adjacency_list >

Static Public Attributes

- constexpr static int **vertex_num** = TL::IndexOf<typename graph::vertexes_, typename edge::from>::value

4.5.1 Detailed Description

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

Returns new graph with added edge

Parameters

graph	Template parameter, initial graph
edge	Template parameter, edge to add

Returns

Parameter result, new graph with added edge

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

- graph/add_edge.h

4.6 Objects::Boolean< boolean > Struct Template Reference

Static Public Attributes

- constexpr static bool **value** = boolean

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

- graph/objects.h

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

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

- TL/contains_constructible_parent.h

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

Static Public Attributes

- constexpr static bool **result**

4.8.1 Member Data Documentation

4.8.1.1 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
```

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

- TL/contains_constructible_parent.h

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

Static Public Attributes

- constexpr static bool **result** = true

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

- TL/contains_constructible_parent.h

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

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

- TL/contains_parent.h

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

Static Public Attributes

- constexpr static bool **result**

4.11.1 Member Data Documentation

4.11.1.1 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
```

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

- TL/contains_parent.h

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

Static Public Attributes

- constexpr static bool **result** = true

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

- TL/contains_parent.h

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

Public Types

- using **result** = NullType

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

- TL/find_parent_type_list.h

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

Public Types

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

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

- TL/find_parent_type_list.h

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

Public Types

- using **result** = type_list

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

- TL/find_parent_type_list.h

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

Public Types

- using **result** = NullType

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

- TL/find_type_list_by_class.h

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

Public Types

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

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

- TL/find_type_list_by_class.h

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

Public Types

- using **result** = type_list

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

- TL/find_type_list_by_class.h

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

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

- TL/has_derived_and_constructible.h

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

Static Public Attributes

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

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

- TL/has_derived_and_constructible.h

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

Static Public Attributes

- constexpr static bool **result** = true

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

- TL/has_derived_and_constructible.h

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

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

- TL/is_base_of.h

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

Static Public Attributes

- constexpr static bool **result** = false

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

- TL/is_base_of.h

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

Static Public Attributes

- constexpr static bool **result**

4.24.1 Member Data Documentation

4.24.1.1 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
```

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

- TL/is_base_of.h

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

Public Types

- using **result** = NullType

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

- TL/most_derived.h

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

Public Types

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

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

- TL/most_derived.h

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

Public Types

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

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

- TL/most_derived.h

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

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

- TL/most_derived_and_constructible.h

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

Public Types

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

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

- TL/most_derived_and_constructible.h

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

Public Types

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

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

- TL/most_derived_and_constructible.h

4.31 Class< T > Struct Template Reference

`#include <class.h>`

Public Types

- using value = T
  Holder of a typename.

4.31.1 Detailed Description

template<typename T>
struct Class< T >

Represents holder for a typename

Parameters

T	Template parameter, typename that should be contained

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

- class.h

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

`#include <contains.h>`

Static Public Attributes

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

4.32.1 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 result, true if type_list contains typename T, false otherwise

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

- TL/contains.h

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

Static Public Attributes

- constexpr static bool **result**

4.33.1 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

4.33.2 Member Data Documentation

4.33.2.1 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
```

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

- TL/contains_constructible_parent.h

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

Static Public Attributes

- constexpr static bool **result** = false

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

- TL/contains_constructible_parent.h

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

Static Public Attributes

- constexpr static bool **result**

4.35.1 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

4.35.2 Member Data Documentation

4.35.2.1 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
```

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

- TL/contains_parent.h

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

Static Public Attributes

- constexpr static bool **result** = false

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

- TL/contains_parent.h

4.37 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.

4.37.1 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

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

- graph/edge.h

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

Public Types

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

4.38.1 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

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

- TL/find_parent_type_list.h

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

Public Types

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

4.39.1 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

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

- TL/find_type_list_by_class.h

4.40 MakeGraph::From< graph_type, vertexes, edges > Struct Template Reference

4.40.1 Detailed Description

template<GraphType graph_type, class vertexes, class edges>
struct MakeGraph::From< graph_type, vertexes, edges >

Creates graph (V, E) with V = vertexes, E is defined by graph_type and edges

Parameters

graph_type	Template parameter, how representation of edges looks like
vertexes	Template parameter, vertexes of a graph
edge	Template parameter, representation of edges

Returns

Parameter result, resulting graph

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

- graph/make_graph.h

4.41 MakeGraph::From< ADJACENCY_LIST, vertexes, adjacency_list > Struct Template Reference

`#include <make_graph.h>`

Public Types

- using **result** = Graph< vertexes, adjacency_list >

4.41.1 Detailed Description

template<class vertexes, class adjacency_list>
struct MakeGraph::From< ADJACENCY_LIST, vertexes, adjacency_list >

Creates graph from an adjacency list.

Parameters

vertexes	Template parameter, vertexes of a graph
edge	Template parameter, adjacency list of a graph

Returns

Parameter result, resulting graph

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

- graph/make_graph.h

4.42 MakeGraph::From< ADJACENCY_MATRIX, vertexes, adjacency_matrix > Struct Template Reference

`#include <make_graph.h>`

Public Types

- using **result** = typename IterateThroughAdjacencyMatrix< vertex_size *vertex_size - 1 >::result

4.42.1 Detailed Description

template<class vertexes, class adjacency_matrix>
struct MakeGraph::From< ADJACENCY_MATRIX, vertexes, adjacency_matrix >

Creates graph from an adjacency matrix.

Parameters

vertexes	Template parameter, vertexes of a graph
adjacency_matrix	Template parameter, adjacency matrix Recommended to be represented as a matrix of

See also

Objects::Boolean

Returns

Parameter result, resulting graph

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

- graph/make_graph.h

4.43 MakeGraph::From< EDGE_LIST, vertexes, edge_list > Struct Template Reference

`#include <make_graph.h>`

Public Types

- using **result** = typename GLib::AddEdge< typename From< EDGE_LIST, vertexes, typename edge_list::Tail >::result, typename edge_list::Head >::result

4.43.1 Detailed Description

template<class vertexes, class edge_list>
struct MakeGraph::From< EDGE_LIST, vertexes, edge_list >

Creates graph from list of edges.

Parameters

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

See also

Edge

Returns

Parameter result, resulting graph

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

- graph/make_graph.h

4.44 MakeGraph::From< EDGE_LIST, vertexes, EmptyTypeList > Struct Template Reference

`#include <make_graph.h>`

Public Types

- using **result** = Graph< vertexes, typename TL::GenerateTypeLists< TL::Size< vertexes >::size >::result >

4.44.1 Detailed Description

template<class vertexes>
struct MakeGraph::From< EDGE_LIST, vertexes, EmptyTypeList >

See also

From<EDGE_LIST, vertexes, edge_list>

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

- graph/make_graph.h

4.45 Functor< ResultType, ArgTypes > Class Template Reference

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

- functor.h

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

`#include <functor.h>`

Public Member Functions

- 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)

4.46.1 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

4.46.2 Member Function Documentation

4.46.2.1 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

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

- functor.h

4.47 TL::GenerateTypeLists< n > Struct Template Reference

`#include <generate_type_lists.h>`

Public Types

- using **result** = typename Add< EmptyTypeList, 0, typename GenerateTypeLists< n - 1 >::result >::result

4.47.1 Detailed Description

template<int n>
struct TL::GenerateTypeLists< n >

Generates TypeList of n EmptyTypeLists

See also

EmptyTypeList

Parameters

n	Template parameter, a number of EmptyTypeLists to generate

Returns

Parameter result, TypeList of n EmptyTypeList

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

- TL/generate_type_lists.h

4.48 TL::GenerateTypeLists< 0 > Struct Reference

Public Types

- using **result** = EmptyTypeList

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

- TL/generate_type_lists.h

4.49 Graph< vertexes, adjacency_list > Struct Template Reference

`#include <graph.h>`

Public Types

- using vertexes_ = vertexes

  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 ()

4.49.1 Detailed Description

template<class vertexes, class adjacency_list>
struct Graph< vertexes, 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

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]

4.49.2 Member Function Documentation

4.49.2.1 HasEdge()

```
template<class vertexes , class adjacency_list >
template<class edge >
constexpr bool Graph< vertexes, 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

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

- graph/graph.h

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

Static Public Attributes

- constexpr static bool **result**

4.50.1 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

4.50.2 Member Data Documentation

4.50.2.1 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
```

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

- TL/has_derived_and_constructible.h

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

`#include <has_derived_and_constructible.h>`

Static Public Attributes

- constexpr static bool **result** = false

4.51.1 Detailed Description

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

See also

HasDerivedAndConstructible

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

- TL/has_derived_and_constructible.h

4.52 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

4.52.1 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

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

- TL/index_of.h

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

`#include <index_of.h>`

Static Public Attributes

- constexpr static int **value** = INT32_MIN

4.53.1 Detailed Description

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

See also

IndexOf

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

- TL/index_of.h

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

`#include <index_of.h>`

Static Public Attributes

- constexpr static int **value** = 0

4.54.1 Detailed Description

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

See also

IndexOf

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

- TL/index_of.h

4.55 Objects::Integer< integer > Struct Template Reference

Static Public Attributes

- constexpr static int **value** = integer

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

- graph/objects.h

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

Static Public Attributes

- constexpr static bool **result**

4.56.1 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

4.56.2 Member Data Documentation

4.56.2.1 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
```

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

- TL/is_base_of.h

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

Static Public Attributes

- constexpr static bool **result** = false

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

- TL/is_base_of.h

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

Static Public Attributes

- constexpr static bool **result** = true

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

- TL/is_base_of.h

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

Static Public Attributes

- constexpr static bool **result** = true

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

- TL/is_base_of.h

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

Public Types

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

4.60.1 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

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

- TL/most_derived.h

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

Public Types

- using **result** = T

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

- TL/most_derived.h

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

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

4.62.1 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

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

- TL/most_derived_and_constructible.h

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

Public Types

- using **result** = T

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

- TL/most_derived_and_constructible.h

4.64 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 >

4.64.1 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

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

- TL/no_duplicates.h

4.65 TL::NoDuplicates< EmptyTypeList > Struct Reference

`#include <no_duplicates.h>`

Public Types

- using **result** = EmptyTypeList

4.65.1 Detailed Description

See also

NoDuplicates

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

- TL/no_duplicates.h

4.66 NullType Struct Reference

`#include <null_type.h>`

4.66.1 Detailed Description

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

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

- TL/null_type.h

4.67 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 >

4.67.1 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

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

- TL/remove.h

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

`#include <remove.h>`

Public Types

- using **result** = EmptyTypeList

4.68.1 Detailed Description

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

See also

Remove

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

- TL/remove.h

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

`#include <remove.h>`

Public Types

- using **result** = typename type_list::Tail

4.69.1 Detailed Description

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

See also

Remove

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

- TL/remove.h

4.70 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 >

4.70.1 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

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

- TL/remove.h

4.71 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

4.71.1 Detailed Description

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

See also

RemoveAll

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

- TL/remove.h

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

`#include <replace.h>`

4.72.1 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

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

- TL/replace.h

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

`#include <replace.h>`

Public Types

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

4.73.1 Detailed Description

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

See also

Replace

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

- TL/replace.h

4.74 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

4.74.1 Detailed Description

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

See also

Replace

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

- TL/replace.h

4.75 TL::Size< TypeList > Struct Template Reference

`#include <size.h>`

Static Public Attributes

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

4.75.1 Detailed Description

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

Gets length of a TypeList

Parameters

TypeList	Template parameter

Returns

Parameter size, amount of elements in TypeList

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

- TL/size.h

4.76 TL::Size< EmptyTypeList > Struct Reference

`#include <size.h>`

Static Public Attributes

- constexpr static size_t **size** = 0

4.76.1 Detailed Description

See also

Size

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

- TL/size.h

4.77 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

4.77.1 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

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

- TL/type_at.h

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

`#include <type_at.h>`

Public Types

- using **value** = typename type_list::Head

4.78.1 Detailed Description

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

See also

TypeAt

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

- TL/type_at.h

4.79 TypeList< Args > Struct Template Reference

`#include <type_list.h>`

Public Types

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

4.79.1 Detailed Description

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

See also

TypeList<H, T...>

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

- TL/type_list.h

4.80 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.

4.80.1 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

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

- TL/type_list.h

4.81 TypeList< T > Struct Template Reference

`#include <type_list.h>`

Public Types

- using **Head** = T
- using **Tail** = EmptyTypeList

4.81.1 Detailed Description

template<typename T>
struct TypeList< T >

See also

TypeList<H, T...>

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

- TL/type_list.h

4.82 VertexStream< stream, graph > Struct Template Reference

`#include <vertex_stream.h>`

Public Types

- using stream_ = stream

  TypeList of vertexes of a graph.

Public Member Functions

- template<class Consumer >
  void ForEach (Consumer consumer)

4.82.1 Detailed Description

template<class stream, class graph>
struct VertexStream< stream, graph >

Represents a stream of vertexes of a graph

Parameters

stream	Template parameter, TypeList of vertexes of a graph
graph	Template parameter

4.82.2 Member Function Documentation

4.82.2.1 ForEach()

```
template<class stream , class graph >
template<class Consumer >
void VertexStream< stream, graph >::ForEach (
            Consumer consumer )  [inline]
```

Calls consumer on every vertex in a stream It's recommended that this object must be of type FunctorTypes::↩ Consumer Also it's based on a variation of "Chain of a responsibility" pattern

Parameters

consumer	Consumer, that accepts Class object of a graph and index of a current vertex

See also

FunctorType::Consumer

Class

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

- graph/vertex_stream.h

4.83 VertexStream< EmptyTypeList, graph > Struct Template Reference

```
#include <vertex_stream.h>
```

Public Types

- using **stream** = EmptyTypeList

Public Member Functions

- template<class Consumer >
  void **ForEach** (Consumer consumer)

4.83.1 Detailed Description

template<class graph>
struct VertexStream< EmptyTypeList, graph >

See also

VertexStream

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

- graph/vertex_stream.h

Index