GraphLib

Generated by Doxygen 1.8.16

Chapter 1

Todo List

Member gl::Edge::weight () const

This function must be overwritten to adapt the behavior of the Djikstra algorithm

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

Class Documentation

4.1 gl::Edge Class Reference

Edge is the base clase representing unidirectional edges (arrows, flows) in a graph.

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

Public Member Functions

- Edge (Graph &graph, Node &src, Node &dst)

    Constructor to create a new edge between source and destination node The constructor registers the edge with the parent graph and with the connected nodes.
- Edge (const Edge &other)

    Constructor to create a new edge as a copy of an existing edge The constructor registers the edge with the parent graph and with the connected edges.
- virtual ∼Edge ()

    Virtual destructor for edge The destructor deregisters the edge from the connected nodes and from the graph before destroying.
- bool isConnectedTo (const Node &node) const

    Constant member function checking if a node is connected to this edge.
- Node & source ()

    Member function returning a reference of the source node.
- Node & destination ()

    Member function returning a reference of the destination node.
- virtual double weight () const

    Virtual constant member function defining the weight (or distance, or cost) of the edge Currently set to 1.0, so the shortest path algorithm searches for the path with the least number of edges involved.

Protected Member Functions

- virtual const std::string name () const

    Virtual constant member function returning a description of the edge as string Edge as base class does not have a name by its own, but builds the string from the names of the connected nodes.

Friends

- std::ostream & operator<< (std::ostream &os, const Edge &edge)

  Streaming operator as friend function, streaming the return value of the name() function to the output stream.

4.1.1 Detailed Description

Edge is the base clase representing unidirectional edges (arrows, flows) in a graph.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Edge() `[1/2]`

```
gl::Edge::Edge (
             Graph & graph,
             Node & src,
             Node & dst )
```

Constructor to create a new edge between source and destination node The constructor registers the edge with the parent graph and with the connected nodes.

Parameters

graph	parent graph
src	source node
dst	destination node

4.1.2.2 Edge() `[2/2]`

```
gl::Edge::Edge (
             const Edge & other )
```

Constructor to create a new edge as a copy of an existing edge The constructor registers the edge with the parent graph and with the connected edges.

Parameters

other	existing edge

4.1.3 Member Function Documentation

4.1.3.1 destination()

```
Node& gl::Edge::destination ( )  [inline]
```

Member function returning a reference of the destination node.

Returns

Node& reference of the destination node

4.1.3.2 isConnectedTo()

```
bool gl::Edge::isConnectedTo (
             const Node & node ) const
```

Constant member function checking if a node is connected to this edge.

Parameters

node	const reference of node

Returns

bool true if node is connected to this edge (either as source or destination node), and false otherwise

4.1.3.3 name()

```
const std::string gl::Edge::name ( ) const  [protected], [virtual]
```

Virtual constant member function returning a description of the edge as string Edge as base class does not have a name by its own, but builds the string from the names of the connected nodes.

Returns

std::string the description of the edge, in the format "source -> destination"

4.1.3.4 source()

```
Node& gl::Edge::source ( )  [inline]
```

Member function returning a reference of the source node.

Returns

Node& reference of the source node

4.1.3.5 weight()

```
virtual double gl::Edge::weight ( ) const  [inline], [virtual]
```

Virtual constant member function defining the weight (or distance, or cost) of the edge Currently set to 1.0, so the shortest path algorithm searches for the path with the least number of edges involved.

Todo This function must be overwritten to adapt the behavior of the Djikstra algorithm

Returns

double the value representing the weight (or distance or cost) of the edge

4.1.4 Friends And Related Function Documentation

4.1.4.1 operator<<

```
std::ostream& operator<< (
            std::ostream & os,
            const Edge & edge )  [friend]
```

Streaming operator as friend function, streaming the return value of the name() function to the output stream.

Returns

std::ostream& reference to the output stream

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

- C:/Users/admin/source/repos/lectures/C23/GraphLibDev/GraphLib/edge.h
- C:/Users/admin/source/repos/lectures/C23/GraphLibDev/GraphLib/edge.cpp

4.2 gl::Graph Class Reference

Graph is the base clase representing a graph as parent of nodes and edges.

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

Public Member Functions

- virtual ∼Graph ()

  Virtual destructor for graph The destructor destroys all remaining edges and nodes connected for this graph.
- std::list< Node ∗ > & nodes ()

  Member function returning a reference to the list of nodes connected to this graph.
- std::list< Edge ∗ > & edges ()

  Member function returning a reference to the list of edges connected to this graph.
- Node ∗ findNode (const std::string &name) const

  Member function returning the pointer to the node with the given name, if such node exists.
- std::vector< Edge ∗ > findEdges (const Node &src, const Node &dst) const

  Member function returning a vector of pointers to all edges connecting the source and destination node, and an empty vector if no such edges exist.
- std::deque< Edge ∗ > dijkstra (const Node &src, const Node &dst) const

  Member function returning a vector of pointers to all edges connecting the source and destination node, and an empty vector if no such edges exist.

Protected Attributes

- std::list< Node ∗ > **m_nodes**
- std::list< Edge ∗ > **m_edges**

4.2.1 Detailed Description

Graph is the base clase representing a graph as parent of nodes and edges.

4.2.2 Member Function Documentation

4.2.2.1 dijkstra()

```
std::deque< Edge * > gl::Graph::dijkstra (
            const Node & src,
            const Node & dst ) const
```

Member function returning a vector of pointers to all edges connecting the source and destination node, and an empty vector if no such edges exist.

Returns

std::vector<Edge∗> vector of pointers to all edges connecting the given node, or empty vector

4.2.2.2 edges()

```
std::list<Edge*>& gl::Graph::edges ( )  [inline]
```

Member function returning a reference to the list of edges connected to this graph.

Returns

std::list<Edge∗>& reference of the list of edges

4.2.2.3 findEdges()

```
std::vector< Edge * > gl::Graph::findEdges (
             const Node & src,
             const Node & dst ) const
```

Member function returning a vector of pointers to all edges connecting the source and destination node, and an empty vector if no such edges exist.

Returns

std::vector<Edge∗> vector of pointers to all edges connecting the given node, or empty vector

4.2.2.4 findNode()

```
Node * gl::Graph::findNode (
             const std::string & name ) const
```

Member function returning the pointer to the node with the given name, if such node exists.

Returns

Node∗ pointer to the node with the specified name if such nodes exists, or nullptr otherwise

4.2.2.5 nodes()

```
std::list<Node*>& gl::Graph::nodes ( )  [inline]
```

Member function returning a reference to the list of nodes connected to this graph.

Returns

std::list<Node∗>& reference of the list of nodes

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

- C:/Users/admin/source/repos/lectures/C23/GraphLibDev/GraphLib/graph.h
- C:/Users/admin/source/repos/lectures/C23/GraphLibDev/GraphLib/graph.cpp

4.3 gl::GraphLibException Class Reference

GraphLibException is a specific exception class for this library.

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

Public Member Functions

- GraphLibException (const char ∗error)

  Constructor to create a library exception carrying the error string.
- const char ∗ what () const

  Member function returning reason for error.

4.3.1 Detailed Description

GraphLibException is a specific exception class for this library.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 GraphLibException()

```
gl::GraphLibException::GraphLibException (
            const char * error )  [inline]
```

Constructor to create a library exception carrying the error string.

Parameters

error	error string

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

- C:/Users/admin/source/repos/lectures/C23/GraphLibDev/GraphLib/graph.h

4.4 gl::Node Class Reference

Node is the base clase representing nodes (vertices) in a graph.

```
#include <node.h>
```

Public Member Functions

- Node (Graph &graph, std::string name)

  Constructor to create a new named node The constructor registers the node with the parent graph.
- Node (Graph &graph)

  Constructor to create a new node and assign a auto-generated name to it The constructor registers the node with the parent graph The auto-generated name follows the "Node_0001", "Node_0002", etc. convention.
- virtual ∼Node ()

  Virtual destructor for node The destructor destroys the connected edges and deregisters the node from the parent graph before destroying.
- virtual std::string name () const

  Virtual constant member function returning the name (description) of the node as string.
- std::list< Edge ∗ > & outEdges ()

  Member function returning a reference to the list of outgoing edges.
- std::list< Edge ∗ > & inEdges ()

  Member function returning a reference to the list of incoming edges.
- const Graph & graph () const

  Constant member function returning a const reference to the parent graph of the node.

Static Protected Attributes

- static int **s_instance_number** = 0

Friends

- bool operator== (const Node &lhs, const Node &rhs)

  Comparison operator as friend function, returning true if two nodes are identical (not just having the same member variable content)
- std::ostream & operator<< (std::ostream &os, const Node &node)

  Streaming operator as friend function, streaming the return value of the name() function to the output stream.

4.4.1 Detailed Description

Node is the base clase representing nodes (vertices) in a graph.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Node() [1/2]

```
gl::Node::Node (
            Graph & graph,
            std::string name )
```

Constructor to create a new named node The constructor registers the node with the parent graph.

Parameters

graph	parent graph
name	name of the node

4.4.2.2 Node() [2/2]

```
gl::Node::Node (
            Graph & graph )
```

Constructor to create a new node and assign a auto-generated name to it The constructor registers the node with the parent graph The auto-generated name follows the "Node_0001", "Node_0002", etc. convention.

Parameters

graph	parent graph

4.4.3 Member Function Documentation

4.4.3.1 graph()

```
const Graph& gl::Node::graph ( ) const  [inline]
```

Constant member function returning a const reference to the parent graph of the node.

Returns

Graph& const reference to the parent graph of the node

4.4.3.2 inEdges()

```
std::list<Edge*>& gl::Node::inEdges ( )  [inline]
```

Member function returning a reference to the list of incoming edges.

Returns

std::list<Edge*>& reference of the list of incoming edges

4.4.3.3 name()

```
virtual std::string gl::Node::name ( ) const  [inline], [virtual]
```

Virtual constant member function returning the name (description) of the node as string.

Returns

std::string the name of the node

4.4.3.4 outEdges()

```
std::list<Edge*>& gl::Node::outEdges ( )  [inline]
```

Member function returning a reference to the list of outgoing edges.

Returns

std::list<Edge*>& reference of the list of outgoing edges

4.4.4 Friends And Related Function Documentation

4.4.4.1 operator<<

```
std::ostream& operator<< (
             std::ostream & os,
             const Node & node )  [friend]
```

Streaming operator as friend function, streaming the return value of the name() function to the output stream.

Returns

std::ostream& reference to the output stream

4.4.4.2 operator==

```
bool operator== (
             const Node & lhs,
             const Node & rhs )  [friend]
```

Comparison operator as friend function, returning true if two nodes are identical (not just having the same member variable content)

Returns

bool true if the two nodes are identical, false otherwise

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

- C:/Users/admin/source/repos/lectures/C23/GraphLibDev/GraphLib/node.h
- C:/Users/admin/source/repos/lectures/C23/GraphLibDev/GraphLib/node.cpp

Chapter 5

File Documentation

5.1 C:/Users/admin/source/repos/lectures/C23/GraphLibDev/Graph↩ Lib/edge.h File Reference

```
#include "node.h"
```

Classes

- class gl::Edge

  Edge is the base clase representing unidirectional edges (arrows, flows) in a graph.

5.1.1 Detailed Description

Version

1.0

Author

Carsten Thomas

Date

December 2020

5.2 C:/Users/admin/source/repos/lectures/C23/GraphLibDev/Graph↩ Lib/graph.h File Reference

```
#include <list>
#include <vector>
#include <deque>
#include <string>
#include <exception>
#include <iostream>
#include "node.h"
#include "edge.h"
```

Classes

- class gl::Graph

  Graph is the base clase representing a graph as parent of nodes and edges.
- class gl::GraphLibException

  GraphLibException is a specific exception class for this library.

5.2.1 Detailed Description

Version

1.0

Author

Carsten Thomas

Date

December 2020

5.3 C:/Users/admin/source/repos/lectures/C23/GraphLibDev/Graph↩ Lib/node.h File Reference

```
#include <string>
#include <list>
#include <iostream>
```

Classes

- class gl::Node

  Node is the base clase representing nodes (vertices) in a graph.

5.3.1 Detailed Description

Version

1.0

Author

Carsten Thomas

Date

December 2020

Index