Graphs Laboratory 1 Homework

Generated by Doxygen 1.8.17

1 Hierarchical Index

1.1 Class Hierarchy

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

2 Class Index

2.1 Class List

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

3 File Index

3.1 File List

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

4 Class Documentation

4.1 Graph Class Reference

the model for the graph class

`#include <Graph.h>`

Public Member Functions

- Graph ()
  The empty constructor of the graph. Returns an empty graph.
- Graph (const Graph &other)
  The copy constructor of the graph.
- Graph & operator= (const Graph &other)
  The assignment constructor of the graph.
- ~Graph ()
  the desturctor of the graph
- Graph (const std::vector< std::string > &vertices, const std::vector< std::tuple< std::string, std::string, int > > &edges)
  constructor from a given list of vertices and a given list of edges
- std::vector< std::string > parse_vertices () const

this function returns a vector containing all vertices

- bool is_edge (const std::string &x, const std::string &y)

  this function returns true if there is an edge from x to y and false otherwise
- int get_in_degree (const std::string &x)

  this function returns the in degree of a given vertex x
- int get_out_degree (const std::string &x)

  this function returns the out degree of a given vertex x
- std::vector< std::string > parse_outbound_edges (const std::string &x)

  This function returns a vector containing the endpoints of all edges outbound to x.
- std::vector< std::string > parse_inbound_edges (const std::string &x)

  This function returns a vector containing the endpoints of all edges inbound to x.
- int get_edge_cost (const std::string &x, const std::string &y)

  this function returns the cost of the edge from x to y
- void modify_edge_cost (const std::string &x, const std::string &y, int z)

  modifies the cost of the edge from x to y
- void add_vertex (const std::string &x)

  adds a vertex to the graph
- void remove_vertex (const std::string &x)

  removes a vertex from the graph
- void add_edge (const std::string &x, const std::string &y, int z)

  adds an edge from x to y with cost z
- void remove_edge (const std::string &x, const std::string &y)

  removes the edge from x to y
- bool operator== (const Graph &other) const

  graph equality operator. Checks if the graphs are basicly the same
- bool operator!= (const Graph &other) const

  graph not equal operator. Checks if the graphs are not the same

4.1.1 Detailed Description

the model for the graph class

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Graph()

```
Graph::Graph (
        const std::vector< std::string > & vertices,
        const std::vector< std::tuple< std::string, std::string, int > > & edges )
```

constructor from a given list of vertices and a given list of edges

Parameters

vertices	the vertices of the graph
edges	the edges of the graph, provided as tuples of (string, string, int)

4.1.3 Member Function Documentation

4.1.3.1 add_edge() `void Graph::add_edge (`

```
const std::string & x,
const std::string & y,
int z )
```

adds an edge from x to y with cost z

Parameters

x	the first vertex
y	the second vertex
z	the cost of the edge @raises GraphException if the vertices don't exist or if the edge already exists

4.1.3.2 add_vertex() `void Graph::add_vertex (`

```
const std::string & x )
```

adds a vertex to the graph

Parameters

x	the vertex @raises GraphException if the vertex already exists

4.1.3.3 get_edge_cost() `int Graph::get_edge_cost (`

```
const std::string & x,
const std::string & y )
```

this function returns the cost of the edge from x to y

Parameters

x	the first vertex
y	the second vertex

Returns

the cost of the edge from x to y @raises GraphException if vertices don't exist @raises GraphException if edge doesn't exist

4.1.3.4 get_in_degree() int Graph::get_in_degree (const std::string & x)

this function returns the in degree of a given vertex x

Parameters

x	the vertex

Returns

the in degree of the specified vertex

4.1.3.5 get_out_degree() int Graph::get_out_degree (const std::string & x)

this function returns the out degree of a given vertex x

Parameters

x	the vertex

Returns

the out degree of the specified vertex @raises GraphException if vertex doesn't exist

4.1.3.6 is_edge() bool Graph::is_edge (const std::string & x, const std::string & y)

this function returns true if there is an edge from x to y and false otherwise

Parameters

x	the first vertex
y	the second vertex

Returns

true if there is an edge from x to y false otherwise

4.1.3.7 modify_edge_cost() `void Graph::modify_edge_cost (`

```
const std::string & x,
const std::string & y,
int z )
```

modifies the cost of the edge from x to y

Parameters

x	the first vertex
y	the second vertex
z	the new cost @raises GraphException if vertices don't exist @raises GraphException if edge doesn't exist

4.1.3.8 operator"!=() `bool Graph::operator!= (`

```
const Graph & other ) const
```

graph not equal operator. Checks if the graphs are not the same

Parameters

other	the other graph

Returns

true if they are different, false otherwise

4.1.3.9 operator==() `bool Graph::operator== (`

```
const Graph & other ) const
```

graph equality operator. Checks if the graphs are basicly the same

Parameters

other	the other graph

Returns

true if they are equal, false otherwise

4.1.3.10 parse_inbound_edges() `vector< string > Graph::parse_inbound_edges (`

```
const std::string & x )
```

This function returns a vector containing the endpoints of all edges inbound to x.

Parameters

x	the vertex

Returns

a vector containing the inbound neighbors of x @raises GraphException if vertex doesn't exist

4.1.3.11 parse_outbound_edges()

```
vector< string > Graph::parse_outbound_edges (
        const std::string & x )
```

This function returns a vector containing the endpoints of all edges outbound to x.

Parameters

x	the vertex

Returns

a vector containing the outbound neighbors of x @raises GraphException if vertex doesn't exist

4.1.3.12 remove_edge()

```
void Graph::remove_edge (
        const std::string & x,
        const std::string & y )
```

removes the edge from x to y

Parameters

x	the first vertex
y	the second vertex @raises GraphException if the vertices don't exist or the edge doesn't exist

4.1.3.13 remove_vertex()

```
void Graph::remove_vertex (
        const std::string & x )
```

removes a vertex from the graph

Parameters

x	the vertex @raises GraphException if the vertex doesn't exist

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

- Graph.h
- Graph.cpp

4.2 GraphException Class Reference

the graph exception class

`#include <Graph.h>`

Public Member Functions

- GraphException (const std::string &msg)

  the constructor of the exception
- std::string what () const

  the message of the exception

4.2.1 Detailed Description

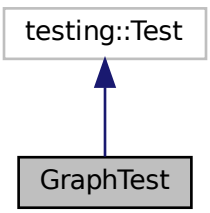
the graph exception class

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

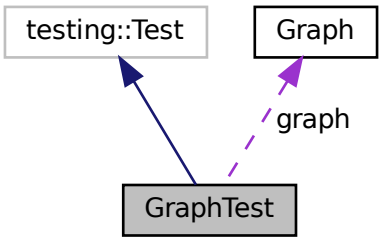
- Graph.h
- Graph.cpp

4.3 GraphTest Class Reference

Inheritance diagram for GraphTest:



Collaboration diagram for GraphTest:



Protected Member Functions

- void **SetUp** () override

Protected Attributes

- Graph **graph**

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

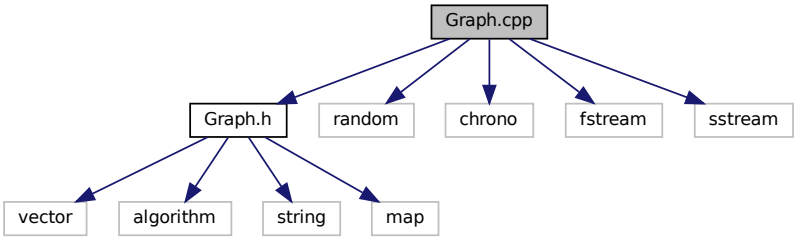
- tests/graph_tests.cpp

5 File Documentation

5.1 Graph.cpp File Reference

```
#include "Graph.h"
#include <random>
#include <chrono>
#include <fstream>
#include <sstream>
```

Include dependency graph for Graph.cpp:



Functions

- bool **endswith** (string s, string ends)
- Graph **read_graph** (const string &file)
- void **write_graph** (const string &file, Graph &graph)
- Graph random_graph (int n, int m)

  This function generates a random graph with n vertices and m edges.

5.1.1 Detailed Description

this file contains the implementation described in Graph.h

5.1.2 Function Documentation

5.1.2.1 random_graph()

```
Graph random_graph (
        int n,
        int m )
```

This function generates a random graph with n vertices and m edges.

Parameters

n	the number of vertices
m	the number of edges

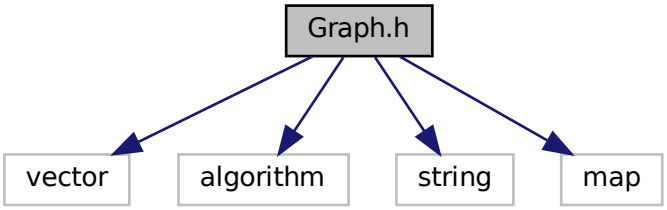
Returns

the graph

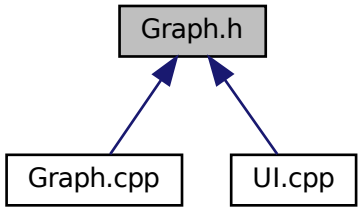
5.2 Graph.h File Reference

```
#include <vector>
#include <algorithm>
#include <string>
#include <map>
```

Include dependency graph for Graph.h:



This graph shows which files directly or indirectly include this file:



Classes

- class GraphException

  the graph exception class
- class Graph

  the model for the graph class

Functions

- Graph read_graph (const std::string &file)

  reads a graph from file It supports 2 formats .txt and .modified.txt
- void write_graph (const std::string &file, Graph &graph)

  writes a graph to file This function writes a graph from a file. It supports 1 format .modified.txt
- Graph random_graph (int n, int m)

  This function generates a random graph with n vertices and m edges.

5.2.1 Detailed Description

this file contains the model for a Graph class

5.2.2 Function Documentation

5.2.2.1 random_graph()

```
Graph random_graph (
        int n,
        int m )
```

This function generates a random graph with n vertices and m edges.

Parameters

n	the number of vertices
m	the number of edges

Returns

the graph

5.2.2.2 read_graph()

```
Graph read_graph (
        const std::string & file )
```

reads a graph from file It supports 2 formats .txt and .modified.txt

In case of .txt, the file is supposed to look like this:

On the first line, the number n of vertices and the number m of edges; On each of the following m lines, three numbers, x, y and c, describing an edge: the origin, the target and the cost of that edge.

In case of .modified.txt, the file is supposed to look like this:

On the first line, the number n of vertices and the number m of edges

On the second line, a list of the n vertices separated by space On each of the following m lines, three numbers, x, y and c, describing an edge: the origin, the target and the cost of that edge.

Parameters

filename	the file from which to read(name, relative path or absolute path)

Returns

Graph @raises GraphException in case of invalid format

5.2.2.3 write_graph()

```
void write_graph (
        const std::string & file,
        Graph & graph )
```

writes a graph to file This function writes a graph from a file. It supports 1 format .modified.txt

On the first line, the number n of vertices and the number m of edges On the second line, a list of the n vertices separated by space On each of the following m lines, three numbers, x, y and c, describing an edge: the origin, the target and the cost of that edge.

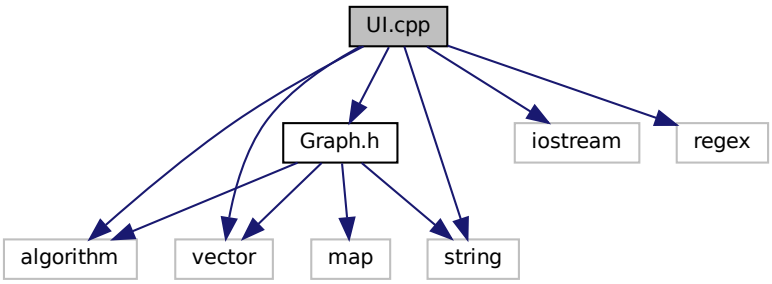
Parameters

filename	the filename to which to read(name, relative path or absolute path), MUST end in .modified.txt
graph	the graph to be written @raises GraphException: if invalid data

5.3 UI.cpp File Reference

```
#include <algorithm>
#include <vector>
#include <string>
#include <iostream>
#include "Graph.h"
#include <regex>
```

Include dependency graph for UI.cpp:



Functions

- void display_edges (const vector< tuple< string, string, int > > &edges)
  this function displays a vector o edges
- void display_vertices (const vector< string > &vertices)
- void display_help (vector< string > &help_prompt)
- vector< string > my_split (string s, char separator)
- int main ()

5.3.1 Detailed Description

this file contains the ui of the application. It also serves as the main file

5.3.2 Function Documentation

5.3.2.1 display_edges() `void display_edges ( const vector< tuple< string, string, int > > & edges )`

this function displays a vector o edges

Parameters

edges	the vector containing the edges

5.3.2.2 display_help() `void display_help ( vector< string > & help_prompt )`

This function displays a given help prompt

Parameters

help_prompt	

5.3.2.3 display_vertices() `void display_vertices ( const vector< string > & vertices )`

This function displays the given vertices

Parameters

vertices	the vector of vertices

5.3.2.4 main() `int main ( )`

the main of the file

Returns

0 hopefully

5.3.2.5 my_split()

```
vector<string> my_split (
        string s,
        char separator )
```

This function splits a given string by a separator

Parameters

s	the string
separator	the separator

Returns

a vector containing the fields of the result

Index