TITLE

AUTHOR
Version
CREATEDATE

Table of Contents

Table of contents

Class Index

Class List

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

WeightedGraph::Vertex 4

Vertex 5

WeightedGraph 6

WtGraph 13

File Index

File List

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

config.h ..... **Error! Bookmark not defined.**
WeightedGraph.cpp (This program will implement a Weighted Graph) 14
WeightedGraph.h ..... **Error! Bookmark not defined.**
WeightedGraph2.h ..... **Error! Bookmark not defined.**
WeightedGraph3.h ..... **Error! Bookmark not defined.**

Class Documentation

WeightedGraph::Vertex Class Reference

Public Member Functions

- void **setLabel** (const string &newLabel)
- string **getLabel** () const
- void **setColor** (char newColor)
- char **getColor** () const

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

- WeightedGraph.h

Vertex Class Reference

Public Attributes

- char **label** [vertexLabelLength]
- char **color**

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

- WeightedGraph2.h
- WeightedGraph3.h

WeightedGraph Class Reference

Classes

- class **Vertex**

Public Member Functions

- **WeightedGraph** (int maxNumber=defMaxGraphSize)
- **WeightedGraph** (const **WeightedGraph** &other)
- **WeightedGraph** & **operator=** (const **WeightedGraph** &other)
- **~WeightedGraph** ()
- void **insertVertex** (const **Vertex** &newVertex) throw (logic_error)
- void **insertEdge** (const string &v1, const string &v2, int wt) throw (logic_error)
- bool **retrieveVertex** (const string &v, **Vertex** &vData) const
- bool **getEdgeWeight** (const string &v1, const string &v2, int &wt) const throw (logic_error)
- void **removeVertex** (const string &v) throw (logic_error)
- void **removeEdge** (const string &v1, const string &v2) throw (logic_error)
- void **clear** ()
- bool **isEmpty** () const
- bool **isFull** () const
- void **showStructure** () const
- void **showShortestPaths** () const
- bool **hasProperColoring** () const
- bool **areAllEven** () const
- **WeightedGraph** (int maxNumber=defMaxGraphSize)
- **WeightedGraph** (const **WeightedGraph** &other)
- **WeightedGraph** & **operator=** (const **WeightedGraph** &other)
- void **insertVertex** (**Vertex** newVertex) throw (logic_error)
- void **insertEdge** (char *v1, char *v2, int wt) throw (logic_error)
- bool **retrieveVertex** (char *v, **Vertex** &vData) const
- int **edgeWeight** (char *v1, char *v2, int &wt) const throw (logic_error)
- bool **getEdgeWeight** (char *v1, char *v2, int &wt) const throw (logic_error)
- void **removeVertex** (char *v) throw (logic_error)
- void **removeEdge** (char *v1, char *v2) throw (logic_error)
- void **clear** ()
- void **computePaths** ()
- bool **isEmpty** () const
- bool **isFull** () const
- void **showStructure** () const

Static Public Attributes

- static const int **defMaxGraphSize** = 10
- static const int **vertexLabelLength** = 11
- static const int **infiniteEdgeWt** = INT_MAX

Constructor & Destructor Documentation

WeightedGraph::WeightedGraph (int *maxNumber* = `defMaxGraphSize`)

Constructor for Weighted Graph. Allocates enough memory for a graph containing maxNumber vertices

Parameters:

maxNumber	: Number of vertices allowed in this graph

Returns:

none

Precondition:

none

Postcondition:

Graph will have memory allocated to it and maxSize will be set

WeightedGraph::WeightedGraph (const WeightedGraph & *other*)

Copy Constructor for Weighted Graph.

Parameters:

other	: Weighted which will be copied into this graph

Returns:

None

Precondition:

None

Postcondition:

This graph will be equivalent to other

WeightedGraph::~WeightedGraph ()

Destructor for Weighted Graph

Parameters:

None	

Returns:

None

Precondition:

None

Postcondition:

Memory of the graph will be deallocated

Member Function Documentation

bool WeightedGraph::areAllEven () const

Returns true if every vertex in a graph is of even degree. Otherwise, return false.

Parameters:

None	

Returns:

Returns true if every vertex in a graph is of even degree. Otherwise, return false.

Precondition:

None

Postcondition:

The contents of this graph will be unaffected

void WeightedGraph::clear ()

Removes all the vertices and edges in the graph.

Parameters:

None	

Returns:

None

Precondition:

None

Postcondition:

Removes all the vertices and edges in the graph.

bool WeightedGraph::getEdgeWeight (const string & *v1*, const string & *v2*, int & *wt*) const throw logic_error)

Searches the graph for the edge connecting vertices v1 and v2. If this edge exists, then places the weight of the edge in wt and returns true. Otherwise, returns false with wt undefined.

Parameters:

v1	: vertice 1 of edge to search for
v2	: vertice 2 of the edge to search for
wt	: If the given edge is found, its weight will be copied into this reference variable

Returns:

bool : If the edge exists, true; else false

Precondition:

Graph includes vertices v1 and v2

Postcondition:

If this edge exists, then places the weight of the edge in wt and returns true. Otherwise, returns false with wt undefined.

bool WeightedGraph::hasProperColoring () const

Returns true if no vertex in the graph has the same color as an adjacent vertex. Otherwise, returns false.

Parameters:

None	

Returns:

bool : Returns true if no vertex in the graph has the same color as an adjacent vertex. Otherwise, returns false.

Precondition:

All vertices have been assigned a color

Postcondition:

The contents of the graph will be unaffected

void WeightedGraph::insertEdge (const string & *v1*, const string & *v2*, int *wt*) throw logic_error)

Inserts an undirected edge connecting vertices v1 and v2 into the graph. The weight of the edge is wt. If there is already an edge connecting these vertices, then updates the weight of the edge.

Parameters:

v1	: vertice 1
v2	: vertice 2
wt	: weight of the edge

Returns:

none

Precondition:

Graph includes vertices v1 and v2.

Postcondition:

Inserts an undirected edge connecting vertices v1 and v2 into the graph. The weight of the edge is wt. If there is already an edge connecting these vertices, then updates the weight of the edge.

Exceptions:

logic_error	: If graph does not included vertices v1 and v2, through "Vertex not found" error.

void WeightedGraph::insertVertex (const Vertex & *newVertex*) throw logic_error)

Insert newVertex into a graph. If it already exists in the graph, then updates it.

Parameters:

newVertex	: the new vertex to be inserted into the graph

Returns:

None

Precondition:

Graph is not full

Postcondition:

newVertex will be inserted or will be updated (if it already exists in the graph)

Exceptions:

logic_error	: Throws "The Graph is full" is the graph is full

bool WeightedGraph::isEmpty () const

Returns true if the graph is empty (no vertices). Otherwise, returns false.

Parameters:

None	

Returns:

bool : Returns true if the graph is empty (no vertices). Otherwise, returns false.

Precondition:

None

Postcondition:

Contents of graph will be unaffected

bool WeightedGraph::isFull () const

Returns true if the graph is full (cannot add any more vertices). Otherwise, returns false.

Parameters:

None	

Returns:

bool : Returns true if the graph is full (cannot add any more vertices). Otherwise, returns false.

Precondition:

None

Postcondition:

Contents of graph will be unaffected

WeightedGraph & WeightedGraph::operator= (const WeightedGraph & *other*)

Overloaded assignment operator for Weighted Graph

Parameters:

other	: Weighted Graph which will be copied into this graph

Returns:

WeightedGraph& : A copy of this graph

Precondition:

None

Postcondition:

This graph will be equivalent to other

void WeightedGraph::removeEdge (const string & *v1*, const string & *v2*) throw logic_error)

Removes the edge connecting vertices v1 and v2 from the graph.

Parameters:

v1	: vertice one of edge to remove
v2	: vertice two of edge to remove

Returns:

None

Precondition:

Graph includes vertices v1 and v2

Postcondition:

Removes the edge connecting vertices v1 and v2 from the graph.

void WeightedGraph::removeVertex (const string & *v*) throw logic_error)

Removes vertex v from the graph and any edges connected to v.

Parameters:

v	: vertex to be removed

Returns:

none

Precondition:

Graph includes vertex v

Postcondition:

Removes vertex v from the graph and any edges connected to v.

Exceptions:

logic_error	: If the given vertex does not exist in the graph, throw a logic_error

bool WeightedGraph::retrieveVertex (const string & *v*, Vertex & *vData*) const

Searches a graph for vertex v. If this vertex is found, then places the value of the vertex's data in vData and returns true. Otherwise, returns false with vData undefined.

Parameters:

v	: vertex string label to look for
vData	: if the given vertex is found, it will be copied into this reference variable

Returns:

bool : true if the given vertex is in the graph, false otherwise

Precondition:

None

Postcondition:

If this vertex is found, then places the value of the vertex's data in vData and returns true. Otherwise, returns false with vData undefined.

void WeightedGraph::showShortestPaths () const

Computes and displays the graph's path matrix If there exists a path (a to c) in which there are subsequent paths (a to b) and (b to c), in which ((a to b) + (b to c)) < (a to c), replace a to c with ((a to b) + (b + c)).

Parameters:

None	

Returns:

None

Precondition:

None

Postcondition:

Contents of the graph will be unaffected

void WeightedGraph::showStructure () const

Outputs a graph's vertex list and adjacency matrix. This operation is intended for testing/debugging purposes only.

Parameters:

None	

Returns:

None

Precondition:

None

Postcondition:

Contents of graph will be unaffected

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

- WeightedGraph.h
- WeightedGraph2.h

- show12.cpp
- **WeightedGraph.cpp**

WtGraph Class Reference

Public Member Functions

- **WtGraph** (int maxNumber=defMaxGraphSize) throw (bad_alloc)
- void **insertVertex** (**Vertex** newVertex) throw (logic_error)
- void **insertEdge** (char *v1, char *v2, int wt) throw (logic_error)
- bool **retrieveVertex** (char *v, **Vertex** &vData) const
- bool **edgeWeight** (char *v1, char *v2, int &wt) const throw (logic_error)
- bool **getEdgeWeight** (char *v1, char *v2, int &wt) const throw (logic_error)
- void **removeVertex** (char *v) throw (logic_error)
- void **removeEdge** (char *v1, char *v2) throw (logic_error)
- void **clear** ()
- bool **isEmpty** () const
- bool **isFull** () const
- bool **hasProperColoring** () const
- void **showStructure** () const

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

- WeightedGraph3.h
- WeightedGraph.cs

File Documentation

WeightedGraph.cpp File Reference

This program will implement a Weighted Graph.

```
#include <stdexcept>
#include <iostream>
#include <climits>
#include <string>
#include "WeightedGraph.h"
```

Detailed Description

This program will implement a Weighted Graph.

Author:

Tim Kwist

Version:

1.0

The specifications of this program are defined by C++ Data Structures: A Laboratory Course (3rd edition) by Brandle, JGeisler, Roberge, Whittington, lab 12.

Date:

Wednesday, November 21, 2014

Index

INDEX