PA10-L12 Catherine Pollock

Generated by Doxygen 1.7.6.1

Thu Nov 20 2014 03:28:38

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Class Documentation

3.1 WeightedGraph::Vertex Class Reference

`#include <WeightedGraph.h>`

Public Member Functions

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

Private Attributes

- string label
- char color

3.1.1 Member Function Documentation

3.1.1.1 char WeightedGraph::Vertex::getColor ( ) const [inline]

3.1.1.2 string WeightedGraph::Vertex::getLabel ( ) const [inline]

3.1.1.3 void WeightedGraph::Vertex::setColor (char *newColor*) [inline]

3.1.1.4 void WeightedGraph::Vertex::setLabel (const string & *newLabel*) [inline]

3.1.2 Member Data Documentation

3.1.2.1 char WeightedGraph::Vertex::color [private]

3.1.2.2 string WeightedGraph::Vertex::label [private]

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

- WeightedGraph.h

3.2 WeightedGraph Class Reference

`#include <WeightedGraph.h>`

Classes

- class Vertex

Public Member Functions

- WeightedGraph (int maxNumber=MAX_GRAPH_SIZE)

  GRAPH PUBLIC FUNCTION IMPLEMENTATION ----------.
- 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

Static Public Attributes

- static const int MAX_GRAPH_SIZE = 10
- static const int INFINITE_EDGE_WT = INT_MAX

Private Member Functions

- int getIndex (const string &v) const

  GRAPH FACILITATOR FUNCTIONS ----------.
- int getEdge (int row, int col) const
- void setEdge (int row, int col, int wt)
- int getPath (int row, int col) const
- void setPath (int row, int col, int wt) const

Private Attributes

- int maxSize
- int size
- Vertex * vertexList
- int * adjMatrix
- int * pathMatrix

3.2.1 Constructor & Destructor Documentation

3.2.1.1 WeightedGraph::WeightedGraph (int *maxNumber* = MAX_GRAPH_SIZE)

GRAPH PUBLIC FUNCTION IMPLEMENTATION ----------.

default constructor

Creates an empty graph. Allocates enough memory for a graph containing maxNumber verticies.

Parameters

maxNumber	int of max number of verticies

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

copy constructor

Initializes the graph to be equivalent to the other graph parameter.

Parameters

other	reference to a graph to be copied from

set this graph to equal other

3.2.1.3 WeightedGraph::~WeightedGraph ( )

destructor

Dellocates (frees) the memory used to store the graph. free memory for lists

3.2.2 Member Function Documentation

3.2.2.1 bool WeightedGraph::areAllEven ( ) const

areAllEven

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

Returns

bool if all verticies are of even degree

loop throught all values in graph

if the edge has a value, incriment k

if there are an odd number of valued edges, return false

3.2.2.2 void WeightedGraph::clear ( )

clear

Removes all verticies and edges from graph. for all rows

clear vertexList values

set all edges to infinity

reset size

3.2.2.3 int WeightedGraph::getEdge (int *row,* int *col*) const [private]

getEdge

Get edge weight using adjacency matrix indicies.

Returns

int of weight

Parameters

row	(int) index of row
col	(int) index of column

return edge value at index

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

getEdgeWeight

Searches the graph for the edge containing verticies 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

wt	(int) weight of edge found
v1	(const string&) first vertex to find
v2	(const string&) second vertex to find

Precondition

graph contains v1 and v2

Exceptions

logic_error	thrown when graph does not contain v1 and/or v2

Returns

bool if found edge with a weight

get index of verticies

get the weight of the edge

return if weight found on edge

3.2.2.5 int WeightedGraph::getIndex (const string & *v*) const [private]

GRAPH FACILITATOR FUNCTIONS ----------.

getIndex

Returns the adjacency matrix index for vertex v. Returns size if the vertex does not exist.

Returns

int of index

Parameters

v	(char∗) character pointer to vertex label

search for passed string and return index

3.2.2.6 int WeightedGraph::getPath (int *row,* int *col*) const [private]

getPath

Get edge path using adjacency matrix indicies.

Returns

int of weight

Parameters

row	(int) index of row
col	(int) index of column

return edge value at index

3.2.2.7 bool WeightedGraph::hasProperColoring ( ) const

hasProperColoring

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

Returns

bool if no vertex has same color as adjacent vertex

Precondition

all verticies have been assigned a color.

loop through all values

if the vertex has a value

if the vetex has same value, return false

otherwise return true

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

insertEdge

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

Parameters

wt	(int) weight of edge
v1	(const string&) first vertex to connect
v2	(const string&) second vertex to connect

Precondition

graph contains v1 and v2

Exceptions

logic_error	thrown when graph does not contain v1 and/or v2

get indexes of verticies

set the edge weight for verticies

stuff needed to work i guess

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

insertVertex

Inserts newVertex into graph. If the vertex already exists in graph, then updates it.

Parameters

newVertex	(const Vertex&) value given to insert.

Exceptions

logic_error	thrown when graph is full

Precondition

graph is not full

if graph is full, throw logic error

if existing vertex matches, update edges

otherwise, insert new vertex and incriment size

3.2.2.10 bool WeightedGraph::isEmpty ( ) const

isEmpty

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

Returns

bool if graph is empty

return if size is equal to zero

3.2.2.11 bool WeightedGraph::isFull ( ) const

isFull

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

Returns

bool if graph is full

return if size is equal to maxSize

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

overloaded assignment operator

Sets the graph to be equivalent to the other graph parameter and returns a reference to this object.

Parameters

other	WeightedGraph reference to a heap to be copied from

Returns

WeightedGraph& reference to this graph

if this does not have the address of other

clear this graph

copy data from other

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

removeEdge

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

Parameters

v1	(const string&) first vertex of edge
v2	(const string&) second vertex of edge

Precondition

graph contains v1 and v2

Exceptions

logic_error	thrown when graph does not contain v1 and/or v2

get index of verticies

set edge to infinity

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

removeVertex

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

Parameters

v	(const string&) vertex to find and remove

Exceptions

logic_error	thrown if vertex v not in graph

Precondition

graph includes vertex v

shift following matrix cols left

shift following matrix rows up

shift verticies left

decrement size

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

retrieveVertex

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

Parameters

vData	(Vertex&) data found at found vertex
v	(const string&) vertex to find

Returns

bool if vertex was found

if empty, return false

iterate through vertexList

if matching vertex, set vData and return true;

otherwise, return false

3.2.2.16 void WeightedGraph::setEdge (int *row,* int *col,* int *wt*) [private]

setEdge

Set edge weight using adjacency matrix indicies.

Parameters

wt	(int) weight value
row	(int) index of row
col	(int) index of column

set edge values for specific row and column to wt

3.2.2.17 void WeightedGraph::setPath (int *row,* int *col,* int *wt*) const [private]

setPath

Set edge path using adjacency matrix indicies.

Parameters

wt	(int) weight value
row	(int) index of row
col	(int) index of column

set path values for specific row and column to wt

3.2.2.18 void WeightedGraph::showShortestPaths ( ) const

showShortestPaths

Computes and displays the graph's path matrix. set values in path matrix

set values in path matrix to those of edge matrix

give main diagonal values 0 for path

loop through and set path values based on value

if the path does not have infinity weight

if the path is less than previous calculated path

set the path with the new value

print path matrix

print path value but '-' if infinity

3.2.2.19 void WeightedGraph::showStructure ( ) const

showStructure

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

3.2.3 Member Data Documentation

3.2.3.1 int* WeightedGraph::adjMatrix [private]

3.2.3.2 const int WeightedGraph::INFINITE_EDGE_WT = INT_MAX [static]

3.2.3.3 const int WeightedGraph::MAX_GRAPH_SIZE = 10 [static]

3.2.3.4 int WeightedGraph::maxSize [private]

3.2.3.5 int* WeightedGraph::pathMatrix [private]

3.2.3.6 int WeightedGraph::size [private]

3.2.3.7 Vertex* WeightedGraph::vertexList [private]

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

- WeightedGraph.h
- WeightedGraph.cpp

Chapter 4

File Documentation

4.1 config.h File Reference

Defines

- #define LAB12_TEST1 1
- #define LAB12_TEST2 1
- #define LAB12_TEST3 1

4.1.1 Define Documentation

4.1.1.1 #define LAB12_TEST1 1

WeightedGraph class configuration file. Activate test #N by defining the corresponding LAB12_TESTN to have the value 1.

4.1.1.2 #define LAB12_TEST2 1

4.1.1.3 #define LAB12_TEST3 1

4.2 test12.cpp File Reference

```
#include <iostream> #include <cstring> #include <cctype> ×
#include "WeightedGraph.h" #include "config.h"
```

Functions

- void print_help ()
- int main ()

4.2.1 Function Documentation

4.2.1.1 int main ( )

4.2.1.2 void print_help ( )

4.3 WeightedGraph.cpp File Reference

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

4.3.1 Detailed Description

Author

CatherinePollock

Date

11/19/14

This is the implementation file for the WeightedGraph.h file.

4.4 WeightedGraph.h File Reference

```
#include <stdexcept> #include <iostream> #include <climits> ×
#include <string>
```

Classes

- class WeightedGraph
- class WeightedGraph::Vertex