PA10_Lab_12_Henry_Huffman

Generated by Doxygen 1.7.6.1

Thu Nov 20 2014 14:26:58

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 Vertex Class Reference

`#include <WeightedGraph2.h>`

Public Attributes

- char label [vertexLabelLength]
- char color

3.2.1 Member Data Documentation

3.2.1.1 char Vertex::color

3.2.1.2 char Vertex::label

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

- WeightedGraph2.h
- WeightedGraph3.h

3.3 WeightedGraph Class Reference

`#include <WeightedGraph.h>`

Classes

- class Vertex

Public Member Functions

- WeightedGraph (int maxNumber=MAX_GRAPH_SIZE)
- 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)
- ∼WeightedGraph ()
- 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 MAX_GRAPH_SIZE = 10
- static const int INFINITE_EDGE_WT = INT_MAX
- static const int DEF_MAX_GRAPH_SIZE = 10
- static const int VERTEX_LABEL_LENGTH = 11

Private Member Functions

- int getIndex (const string &v) const
- int getEdge (int row, int col) const
- void setEdge (int row, int col, int wt)
- int getIndex (char ∗v) const
- int getEdge (int row, int col) const
- int getPath (int row, int col) const
- void setEdge (int row, int col, int wt)
- void setPath (int row, int col, int wt)

Private Attributes

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

3.3.1 Constructor & Destructor Documentation

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

WeightedGraph Constructor

This constructor creates an array of vertices for the list of vertices. It also creates arrays for the adjacency matrix and path matrix. The values in the path matrix and adjacency matrix are set to the infinite edge weight. It also sets the maxSize and size of the WeightedGraph.

Parameters

maxNumber	- an integer that sets the maxSize of the number of dataItems in the array

Returns

none

Precondition

there will not be an initialized WeightedGraph

Postcondition

there will be an initialized Weighted with the maxSize and size set

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

Weighted Graph Copy Constructor

This copy constructor creates an array of new dataItems, sets the maxSize, and current size according to another specified weighted graph.

Parameters

other	- the specified weighted graph that is to be copied

Returns

none

Precondition

there will be one initialized weighted graph

Postcondition

there will be two initialized weighted graphs with identical values.

3.3.1.3 WeightedGraph::∼WeightedGraph ( )

Weighted Graph Destructor

This function deallocates all memory in the current weighted graph and sets everything to NULL or zero.

Parameters

none	

Returns

none

Precondition

the weighted graph will have memory allocated to it

Postcondition

the weighted graph will not have any memory allocated to it

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

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

3.3.1.6 WeightedGraph::∼WeightedGraph ( )

3.3.2 Member Function Documentation

3.3.2.1 bool WeightedGraph::areAllEven ( ) const

areAllEven function

This function checks to see if all of the vertices have an even number of edges

Parameters

none	

Returns

bool - returns whether or not all are even

Precondition

the weighted graph may or may not haven even number of edges

Postcondition

a bool that determines whether or not the weighted graph has an even number of edges is returned

3.3.2.2 void **WeightedGraph::clear** ( )

3.3.2.3 void **WeightedGraph::clear** ( )

clear function

This function sets the size equivalent to zero and sets the adjacency matrix at every location to infinite

Parameters

none	

Returns

none

Precondition

the size may or may not be set to zero, and the adjacency matrix may have varying edge weights.

Postcondition

the size will be zero and only infinite edge weights will be found in the adjacency matrix

3.3.2.4 void **WeightedGraph::computePaths** ( )

3.3.2.5 int WeightedGraph::edgeWeight (char ∗ *v1*, char ∗ *v2*, int & *wt*) const throw (logic_error)

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

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

getEdge function

This function simply returns the the weight at the specified location.

Parameters

row	- the row location in the adjacency matrix
col	- the column location in the adjacency matrix

Returns

int - the weight at the specified location in the adjacency matrix

Precondition

the edge at the location may or may not be known

Postcondition

the edge at the location will be known

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

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

getEdgeWeight function

This function searches for two vertices. If the vertices are located, the weight of the edge between the two vertices is passed back by reference through the parameter wt. If the weight at the specified location is not infinite, the function returns true. Otherwise, the function will return false if the weight is infinite or the vertices are not found

Parameters

v1	- the label of one of the vertices that this function searches for
v2	- the label of one of the vertices that this fucntion searches for
wt	- the edge weight found between the two vertices.

Returns

bool - returns true if vertices are found and the edge weight is not set to infinity. Othwerwise, returns false.

Precondition

there may or may not be a valied edge weight between the two specified vertices

Postcondition

the edge weight will be found between the vertices. If valid, the function will return true. It will return false in all other cases.

3.3.2.10 int WeightedGraph::getIndex (char ∗ *v*) const [private]

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

getIndex function

This function gets the index of the specified vertex.

Parameters

v	- the label of a vertex

Returns

int - the index of the vertex in the vertexList

Precondition

the location may or may not be know

Postcondition

the location of the vertex will be known

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

3.3.2.13 bool WeightedGraph::hasProperColoring ( ) const

hasProperColoroing function

This function checks to see if there are any vertices adjacent to one another that have the same color.

Parameters

none	

Returns

bool - returns whether or not the colors are proper

Precondition

the coloring may or may not be proper

Postcondition

a bool that determines whether or not the weighted graph has proper coloring will be returned

3.3.2.14 void WeightedGraph::insertEdge (char ∗ *v1,* char ∗ *v2,* int *wt*) throw (logic_error)

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

insertEdge function

This function inserts the edge into the adjacency matrix. It first checks to see if both verices are in the vertex list. If so, it will update the will update the weight.

Parameters

v1	- the label of the first vertex
v2	- the label of the second vertex
wt	- an integer that specifies the weight

Returns

none

Precondition

the graph may not have the specified weight between the two vertices

Postcondition

the specified weight will be placed into the correct location of the adjacency matrix.

3.3.2.16 void **WeightedGraph::insertVertex** (Vertex *newVertex*) throw (logic_error)

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

insertVertex Function

This function inserts a new vertex into the vertex list. In doing so, it also updates the adjacency matrix and path matrix.

Parameters

newVertex	- the vertex that is to be inserted into the vertex list, adjacency matrix, and path matrix.

Returns

none

Precondition

the graph may or may not include the newVertex

Postcondition

the graph will include the new vertex, or update the specified vertex's weight

3.3.2.18 bool **WeightedGraph::isEmpty** ( ) const

3.3.2.19 bool **WeightedGraph::isEmpty** ( ) const

isEmpty function

This function checks to see if current weighted graph is empty.

Parameters

none	

Returns

bool - returns whether or not the size is set to zero

Precondition

the weighted graph may or may not be empty

Postcondition

a bool that determines whether or not the weighted graph is empty is returned

3.3.2.20 bool WeightedGraph::isFull ( ) const

3.3.2.21 bool WeightedGraph::isFull ( ) const

isFull function

This function checks to see if current weighted graph is full.

Parameters

none	

Returns

bool - returns whether or not the size is equal to the maxSize

Precondition

the weighted graph may or may not be full

Postcondition

a bool that determines whether or not the weighted graph is full is returned

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

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

overloaded operator=

This copy constructor creates an array of new dataItems, sets the maxSize, and current size according to another specified weighted graph.

Parameters

other	- the specified weighted graph that is to be copied

Returns

WeightedGraph - the graph with updated values

Precondition

there will be one initialized weighted graph

Postcondition

there will be two initialized weighted graphs with identical values.

3.3.2.24 void **WeightedGraph::removeEdge** (char ∗ *v1,* char ∗ *v2*) throw (logic_error)

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

3.3.2.26 void **WeightedGraph::removeVertex** (char ∗ *v*) throw (logic_error)

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

3.3.2.28 bool **WeightedGraph::retrieveVertex** (char ∗ *v,* **Vertex** & *vData*) const

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

retrieveVertex function

This function searches for a vertex that matches the given label. If the vertex is found, the parameter vData is assigned the vertex and returns true. If the vertex is not found, it will return false.

Parameters

v	- the label used to search for the matching vertex
vData	- used to pass the data of the located vertex back by reference

Returns

bool - returns true if vertex is found, else it returns false

Precondition

the vertex may or may not be in the vertex list

Postcondition

the vertex may or may not be found. If found, the data of the vertex will be passed back by reference through the parameter vData.

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

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

setEdge function

This function sets the edge weight at a specified location in the adjacency matrix

Parameters

row	- the row location in the adjacency matrix
col	- the column location in the adjacency matrix
wt	- the weight that is to be inserted

Returns

none

Precondition

the edge may not contain the weight specified

Postcondition

the edge will contain the weight specified

3.3.2.32 void **WeightedGraph::setPath** (int *row,* int *col,* int *wt*) `[private]`

3.3.2.33 void **WeightedGraph::showShortestPaths** ( ) const

showShortestPath function

This function finds the shortest path between two vertices. The path matrice is used to find said path.

Parameters

none	

Returns

none

Precondition

the shortest path between two vertices may or may not be known

Postcondition

the shortest path will be found and output

3.3.2.34 void WeightedGraph::showStructure ( ) const

3.3.2.35 void WeightedGraph::showStructure ( ) const

3.3.3 Member Data Documentation

3.3.3.1 int ∗ WeightedGraph::adjMatrix `[private]`

3.3.3.2 const int WeightedGraph::DEF_MAX_GRAPH_SIZE = 10 `[static]`

3.3.3.3 static const int WeightedGraph::INFINITE_EDGE_WT = INT_MAX `[static]`

3.3.3.4 const int WeightedGraph::MAX_GRAPH_SIZE = 10 `[static]`

3.3.3.5 int WeightedGraph::maxSize `[private]`

3.3.3.6 int ∗ WeightedGraph::pathMatrix `[private]`

3.3.3.7 int WeightedGraph::size `[private]`

3.3.3.8 const int WeightedGraph::VERTEX_LABEL_LENGTH = 11 `[static]`

3.3.3.9 Vertex ∗ WeightedGraph::vertexList `[private]`

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

- WeightedGraph.h
- WeightedGraph2.h
- show12.cpp
- WeightedGraph.cpp

3.4 WtGraph Class Reference

`#include <WeightedGraph3.h>`

Public Member Functions

- WtGraph (int maxNumber=defMaxGraphSize) throw (bad_alloc)
- ∼WtGraph ()
- 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

Private Member Functions

- int index (char ∗v) const
- int getEdge (int row, int col) const
- void setEdge (int row, int col, int wt)

Private Attributes

- int maxSize
- int size
- Vertex ∗ vertexList
- int ∗ adjMatrix

3.4.1 Constructor & Destructor Documentation

3.4.1.1 WtGraph::WtGraph (int *maxNumber* = defMaxGraphSize) throw (bad_alloc)

3.4.1.2 WtGraph::∼WtGraph ( )

3.4.2 Member Function Documentation

3.4.2.1 void WtGraph::clear ( )

3.4.2.2 bool WtGraph::edgeWeight (char ∗ *v1,* char ∗ *v2,* int & *wt*) const throw (logic_error)

3.4.2.3 int WtGraph::getEdge (int *row,* int *col*) const `[private]`

3.4.2.4 bool WtGraph::getEdgeWeight (char ∗ *v1,* char ∗ *v2,* int & *wt*) const throw (logic_error)

3.4.2.5 bool WtGraph::hasProperColoring ( ) const

3.4.2.6 int WtGraph::index (char ∗ *v*) const `[private]`

3.4.2.7 void WtGraph::insertEdge (char ∗ *v1,* char ∗ *v2,* int *wt*) throw (logic_error)

3.4.2.8 void WtGraph::insertVertex (Vertex *newVertex*) throw (logic_error)

3.4.2.9 bool WtGraph::isEmpty ( ) const

3.4.2.10 bool WtGraph::isFull ( ) const

3.4.2.11 void WtGraph::removeEdge (char ∗ *v1*, char ∗ *v2*) throw (logic_error)

3.4.2.12 void WtGraph::removeVertex (char ∗ *v*) throw (logic_error)

3.4.2.13 bool WtGraph::retrieveVertex (char ∗ *v*, Vertex & *vData*) const

3.4.2.14 void WtGraph::setEdge (int *row*, int *col*, int *wt*) `[private]`

3.4.2.15 void WtGraph::showStructure ( ) const

3.4.3 Member Data Documentation

3.4.3.1 int∗ WtGraph::adjMatrix `[private]`

3.4.3.2 int WtGraph::maxSize `[private]`

3.4.3.3 int WtGraph::size `[private]`

3.4.3.4 Vertex∗ WtGraph::vertexList `[private]`

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

- WeightedGraph3.h
- WeightedGraph.cs

Chapter 4

File Documentation

4.1 config.h File Reference

Defines

- #define LAB12_TEST1 1

    set all config testing to one to enable all programming exercises
- #define LAB12_TEST2 1
- #define LAB12_TEST3 1

4.1.1 Define Documentation

4.1.1.1 #define LAB12_TEST1 1

set all config testing to one to enable all programming exercises

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 show12.cpp File Reference

4.3 test12.cpp File Reference

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

Functions

- void print_help ()
- int main ()

4.3.1 Function Documentation

4.3.1.1 int main ( )

4.3.1.2 void print_help ( )

4.4 WeightedGraph.cpp File Reference

This program builds a graph. This graph uses an adjacency matrix to keep track of the vertices and weights of the graph.

`#include "WeightedGraph.h"`

4.4.1 Detailed Description

This program builds a graph. This graph uses an adjacency matrix to keep track of the vertices and weights of the graph.

Author

Henry Huffman

Version

1.1

More specifically, this program has the following basic member functions: constructor, copy constructor, overloaded = operator, and destructor. This program also contains areAllEven, clear, getEdge, getEdgeWeight, getIndex, hasProperColoring, insertEdge, insertVertex, isEmpty, isFull, removeEdge, removeVertex, retrieveVertex, setEdge, showShortestPath, and showStructure fuctions.

Date

Monday, November 18th, 2014

4.5 WeightedGraph.cs File Reference

`#include <iostream>` `#include <cstring>` `#include "wtgraph.h"`

4.6 WeightedGraph.h File Reference

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

Classes

- class WeightedGraph
- class WeightedGraph::Vertex

4.7 WeightedGraph2.h File Reference

```
#include <climits> #include <new> #include <stdexcept>
```

Classes

- class Vertex
- class WeightedGraph

4.8 WeightedGraph3.h File Reference

```
#include <climits> #include <new> #include <stdexcept>
```

Classes

- class Vertex
- class WtGraph

Variables

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

4.8.1 Variable Documentation

4.8.1.1 const int **defMaxGraphSize** = 10

4.8.1.2 const int **infiniteEdgeWt** = INT_MAX

4.8.1.3 const int **vertexLabelLength** = 11