DTS Lab 7  
graph

Contents

[Objective 2](#_Toc358034637)

[Prototypes 2](#_Toc358034638)

[Desired Output 4](#_Toc358034639)

[Submission 6](#_Toc358034640)

# Objective

Today you will implement a template class, *Graph*, with two publicly nested classes (or structures) named *Edge* and *Vertex*. You will reuse your *DynArray* class from Lab 1 and your *SLList* class from Lab 3. The graph class will function as an adjacency list. Each *Edge* contains the index of a vertex the edge connects to. Each *Vertex* contains a piece of data and a singly linked list of edges that stores all of that vertex’s adjacencies. The *Graph* itself contains a *DynArray* of vertices representing all the vertices in the graph.

Place all your code in a file named *Graph.h*

# Data Members

The Graph class should contain the following private data members:

DynArray<Vertex> theGraph;

# Prototypes

The *Edge* class/structure should contain a single unsigned integer to represent the edge it connects to

unsigned int toVertex;

The *Vertex* class/structure should contain a list of edges, a data element, and a function that allows the user to add a new edge to the list of edges

// the data that this vertex is storing  
Type element;  
  
// the list of edges that connect this vertex to another vertex  
SLList<Edge> edges;  
  
///////////////////////////////////////////////////////////////////////////  
// Function : addEdge  
// Parameters : toVertex - the index of the vertex we are adjacent to  
// Notes: Declare an Edge variable, assign its .toVertex to the parameter,  
// then add it to the list of edges  
///////////////////////////////////////////////////////////////////////////  
void addEdge(const unsigned int& toVertex)

The Graph class should contain the following public interface:

/////////////////////////////////////////////////////////////////////////////  
// Function : addVertex  
// Parameters : value - the data to store in this vertex  
// Return : unsigned int - the index this vertex was added at  
// Notes: Declare a Vertex variable, assign its .element to the parameter,  
// then add it to the dynarray of vertex  
/////////////////////////////////////////////////////////////////////////////  
unsigned int addVertex(const Type& value)

/////////////////////////////////////////////////////////////////////////////  
// Function : operator[]  
// Parameters : index - the index in the graph to access  
// Return : Vertex& - the vertex stored at the specified index  
// Notes: Calls [] on the dynarray  
/////////////////////////////////////////////////////////////////////////////  
Vertex& operator[](const unsigned int& index)  
  
/////////////////////////////////////////////////////////////////////////////  
// Function : size  
// Return : unsiged int - the number of vertices in the graph  
// Notes: Calls the dynarray's size()  
/////////////////////////////////////////////////////////////////////////////  
unsigned int size() const  
  
/////////////////////////////////////////////////////////////////////////////  
// Function : clear  
// Notes : call the dynarray's clear()  
/////////////////////////////////////////////////////////////////////////////  
void clear()  
  
/////////////////////////////////////////////////////////////////////////////  
// Function : printBreadthFirst  
// Parameters : startVertex - the vertex to start the traversal at  
// Notes : prints contents of the graph in depth order   
// (from closest to farthest)  
/////////////////////////////////////////////////////////////////////////////  
void printBreadthFirst(const unsigned int& startVertex)

# Desired Output

Compile and run your code with the DTSLab7.cpp file provided via FSO. Your console output should match the following block identically:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*\* LAB 7: \*\*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
\*\*\* TEST 1 \*\*\*  
Vertex 0 (h) : 1   
Vertex 1 (a) : 2 0   
Vertex 2 (i) : 3 1   
Vertex 3 (r) : 2   
  
  
\*\*\* TEST 2 \*\*\*  
Breadth First Graph Traversal  
h : 0  
a : 1  
i : 2  
r : 3  
  
Breadth First Graph Traversal  
a : 0  
i : 1  
h : 1  
r : 2  
  
Breadth First Graph Traversal  
i : 0  
r : 1  
a : 1  
h : 2  
  
Breadth First Graph Traversal  
r : 0  
i : 1  
a : 2  
h : 3  
  
  
\*\*\* TEST 3 \*\*\*  
Vertex 0 (h) : 1   
Vertex 1 (a) : 2 0   
Vertex 2 (i) : 3 1   
Vertex 3 (r) : 2   
  
Vertex 0 (h) : 1   
Vertex 1 (a) : 2 0   
Vertex 2 (i) : 3 1   
Vertex 3 (r) : 2

\*\*\* TEST 4 \*\*\*  
Vertex 0 (This) : 1   
Vertex 1 (is) : 0 2   
Vertex 2 (only) : 3 1   
Vertex 3 (a) : 4 2   
Vertex 4 (test) : 3   
is : 0  
This : 1  
only : 1  
a : 2  
test : 3

# Submission

To submit the lab assignment:

1. Clean, build, and run DTSLab7.cpp with your Graph.h, SLList.h, and DynArray.h files in Visual Studio (debug mode).
   1. clear up any warnings you encounter.
   2. verify that your output is correct by comparing it to the lab document's Desired Output section, line-by-line.
   3. ensure there are no memory leaks.
2. Submit the 'Graph.h', 'SLList.h', and 'DynArray.h' file via FSO. (you may zip these into a single file if you like