**Graph: Adjacency Matrix**

**LAB # 12**



**Data Structures & Algorithms**

Submitted by: **Shah Raza**

Registration No: **18PWCSE1658**

Class Section: **B**

“On my honor, as a student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted to: **Dr. Khurram Shehzad Khattak**

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

**Lab Objectives:**

Objectives of this lab are as follows:

* Implementing Graph using Adjacency Matrix .

**Task # 1:**

Implement graphs with the following operations

1. Graph Creation
2. Adding Vertex to Graph
3. Removal of Vertex from the graph
4. Checking whether an edge exist between two vertices
5. Printing graph

**Code:**

#include <iostream>

using namespace std;

class Graph

{

int \*\*AdjMatrix;

int vertices;

public:

void Create(int v)

{

vertices=v;

AdjMatrix = new int\*[vertices];

for(int i=0;i<vertices;i++)

{

AdjMatrix[i]= new int[vertices];

for(int j=0;j<vertices;j++)

AdjMatrix[i][j]=0;

}

}

void AddEdge(int i,int j)

{

AdjMatrix[i][j]=1;

AdjMatrix[j][i]=1;

}

void RemoveEdge(int i,int j)

{

AdjMatrix[i][j]=0;

AdjMatrix[j][i]=0;

}

void Print()

{

cout<<" ";

for(int i=0;i<vertices;i++)

cout<<i<<" ";

cout<<endl;

for(int i=0;i<vertices;i++)

{

cout<<i<<" : ";

for(int j=0;j<vertices;j++)

{

cout<<AdjMatrix[i][j]<<" ";

}

cout<<endl;

}

}

bool isEdge(int i,int j)

{

if(AdjMatrix[i][j])

return true;

return false;

}

~Graph() {

for (int i = 0; i < vertices; i++)

delete[] AdjMatrix[i];

delete[] AdjMatrix;

}

};

int main()

{

int choice,v,i,j;

Graph Grp;

do

{

cout<<"1. Create a Graph\n";

cout<<"2. Add an Edge\n";

cout<<"3. Remove an Edge\n";

cout<<"4. Check for Edge between vertices\n";

cout<<"5. Print\n";

cout<<"6. Exit\n";

cin>>choice;

switch(choice)

{

case 1:

cout<<"Enter the no of vertices: ";

cin>>v;

Grp.Create(v);

cout<<"Graph with vertices "<<v<<" has been created.\n";

break;

case 2:

cout<<"Enter Vertix No 1: ";

cin>>i;

cout<<"Enter Vertix No 2: ";

cin>>j;

Grp.AddEdge(i,j);

cout<<"Edge added between Vertix "<<i<<" and Vertix "<<j<<endl;

break;

case 3:

cout<<"Enter Vertix No 1: ";

cin>>i;

cout<<"Enter Vertix No 2: ";

cin>>j;

Grp.RemoveEdge(i,j);

cout<<"Edge Removed between Vertix "<<i<<" and Vertix "<<j<<endl;

break;

case 4:

cout<<"Enter Vertix No 1: ";

cin>>i;

cout<<"Enter Vertix No 2: ";

cin>>j;

if(Grp.isEdge(i,j))

cout<<"Edge exist between Vertix "<<i<<" and Vertix "<<j<<endl;

else

cout<<"Edge does not exist between Vertix "<<i<<" and Vertix "<<j<<endl;

break;

case 5:

Grp.Print();

break;

case 6:

return 0;

break;

default:

cout<<"Invalid Input.\n";

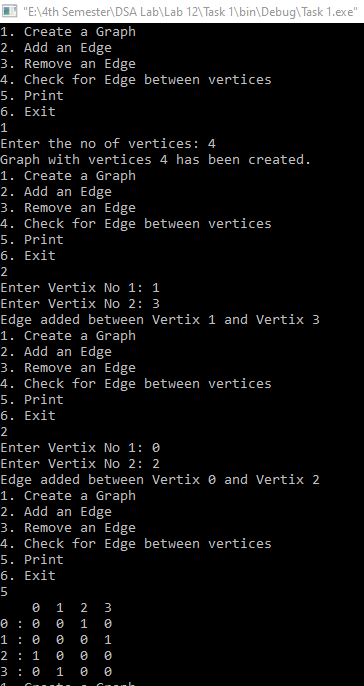
}

}while(choice!=6);

return 0;

}

**Output:**

****