/\*

***Name* :-** Abrar Mahedavi

***Roll No.* :-** 21035

***Title* :-** Write a program in C to find shortest path between source and destination using Dijkstra's algorithm.

*/\**

***Program :***

#include <stdio.h>

#define INFINITY 999 //Macro to assign 999 to INFINITY.

#define MAX 30 //Macro to assign 30 to MAX.

void input(int Graph[MAX][MAX]); //Prototype for input function

void DIJKSTRA(int Graph[MAX][MAX],int Numberofvertices); //Prototype for DIJKSTRA function

void output(int Graph[MAX][MAX],int Numberofvertices); //Prototype for output function

void input(int Graph[MAX][MAX]) //Function Definition for input

{

int Numberofvertices;

int Weight,i,j;

printf("\n\n\tEnter the number of vertices you want \n");

scanf("%d",&Numberofvertices);

printf("\n\n\tNumber of vertices are %d is ",Numberofvertices);

for(i=1;i<=Numberofvertices;i++)

{

for(j=1;j<=Numberofvertices;j++)

{

Graph[i][j]=INFINITY;

}

}

printf("IF THERE IS NO DIRECT CONNECTION BETWEEN THE NODES PLEASE WRITE AS 999\n");

for(i=1;i<=Numberofvertices;i++)

{

for(j=i+1;j<=Numberofvertices;j++)

{

printf("\n\n\tEnter the weight between %d and %d \n",i,j);

scanf("%d",&Weight);

Graph[i][j]=Weight;

Graph[j][i]=Graph[i][j];

}

}

output(Graph,Numberofvertices); //Calling of output function

DIJKSTRA(Graph,Numberofvertices); //Calling of DIJKSTRA function

}

void output(int Graph[MAX][MAX],int Numberofvertices)

//Function Definition for output

{

int i,j;

for(i=1;i<=Numberofvertices;i++)

{

for(j=1;j<=Numberofvertices;j++)

{

printf("\t%d\t",Graph[i][j]);

}

printf("\n");

}

}

void DIJKSTRA(int Graph[MAX][MAX],int Numberofvertices)

//Function Definition for DIJKSTRA

{

int Source,Destination;

int Current,Start,Minimum=0,temp,Shortestpath=0,NewDistance=0;

int i,middle,c,END=1234;

int Visited[MAX],Distance[MAX],Path[MAX];

printf("\n\n\t Enter the Source ");

scanf("%d",&Source);

printf("\n\n\t Enter the Destination ");

scanf("%d",&Destination);

printf("\n\n\t Source is %d \n",Source);

printf("\n\n\t Destination is %d \n ",Destination);

for(i=1;i<=Numberofvertices;i++) //Initialize the Visited and Distance array //to 0 and 999 respectively

{

Visited[i]=0;

Distance[i]=INFINITY;

}

Visited[Source]=1;

Distance[Source]=0;

Current=Source;

while(Current!=Destination) //Loop till we get destination

{

Minimum=INFINITY;

Start=Distance[Current];

for(i=1;i<=Numberofvertices;i++)

{

if(Visited[i]==0)

{

NewDistance = Start + Graph[Current][i];

if(NewDistance<Distance[i])

//Distance Array being updated

{

Distance[i]=NewDistance;

}

if(Distance[i]<Minimum) //Minimum distance stored

{

Minimum=Distance[i];

temp=i;

}

}

}

Path[c++]=Current; //Storing the path in Array

Current=temp;

Visited[Current]=1;

}

printf("\n");

Path [c] =END ;

printf("The shortest path is\n");

for(i=0;Path[i]!=END;i++)

{

printf("%d====>",Path[i]);

}

printf("%d\n",Destination);

printf("Shortest Distance Between %d and %d is %d \n",Source,Destination,Minimum);

}

int main()

{

int Graph[MAX][MAX];

input(Graph);

return 0;

}

***Output :***

compeng-sl2-08@compeng-sl2-08:~$ cd Abrar

compeng-sl2-08@compeng-sl2-08:~/Abrar$ gcc DIJKSTRA.c

compeng-sl2-08@compeng-sl2-08:~/Abrar$ ./a.out

Enter the number of vertices you want

5

Number of vertices are 5 is IF THERE IS NO DIRECT CONNECTION BETWEEN THE NODES PLEASE WRITE AS 999

Enter the weight between 1 and 2

1

Enter the weight between 1 and 3

5

Enter the weight between 1 and 4

999

Enter the weight between 1 and 5

999

Enter the weight between 2 and 3

10

Enter the weight between 2 and 4

2

Enter the weight between 2 and 5

999

Enter the weight between 3 and 4

2

Enter the weight between 3 and 5

3

Enter the weight between 4 and 5

8

999 1 5 999 999

1 999 10 2 999

5 10 999 2 3

999 2 2 999 8

999 999 3 8 999

Enter the Source 1

Enter the Destination 5

Source is 1

Destination is 5

The shortest path is

1====>2====>4====>3====>5

Shortest Distance Between 1 and 5 is 8