**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»**

*Факультет інформатики та обчислювальної техніки*

*Кафедра обчислювальної техніки*

**Лабораторна робота №4**

*з дискретної математики*

*Виконав студент групи ІО-44*

***Барабаш Т.А.***

*Перевірив:*

***Новотарський М.А****.*

**2015**

**ЛАБОРАТОРНА РОБОТА №4**

*Графи. Способи представлення графів. Остовні дерева. Пошук найкоротших шляхів*

**Мета:** Вивчення властивостей графів, способів їх представлення та основних алгоритмів на графах.

**I. Завдання**

**4403 Mod 8 + 1 = 4**

**Варіант 4. Пошук найкоротших шляхів у графі. Алгоритм Дейкстри**

А) Виконати завдання 4 до лабораторної роботи.

Б) За правилом, наданим викладачем, сформувати матрицю ваг *С*=[*ci,j*] графа.

В) Представити граф, заданий матрицею ваг C, у графічній формі та виділити найкоротший шлях

між заданими викладачем вершинами, сформований за допомогою алгоритму Дейкстри.

**ІI. Короткі теоретичні відомості**

**Опис машинного алгоритму Дейкстри**

Алгоритм використовує три масиви з n чисел кожний. Перший масив *Visited* містить мітки із двома значеннями: *False* (вершина ще не розглянута) і *True* (вершина вже розглянута); другий масив *Len* містить поточні найкоротші відстані від початкової до відповідної вершини; третій масив C містить номери вершин –k -й елемент C є номер передостанньої вершини на поточному найкоротшому шляху з початкової вершини в k -у. Використовується також *Matrix* – матриця відстаней.

**Опишемо алгоритм Дейкстри**

1 (ініціалізація). У циклі від 1 до n заповнити значенням False масив Visited; заповнити числом i масив C (i – номер стартової вершини); перенести i-й рядок матриці Matrix у масив Len; Visited[i]:=True; C[i]:=0;

2 (загальний крок). Знайти мінімум серед невідмічених (тобто тих k , для яких Visitid[k]=False); нехай мінімум досягається на індексі j, тобто Len[j]  Len[k]; Потім виконувати наступні операції: Visited[i]:=True; якщо Len[k]>Len[j]+Matrix[j, k], то (Len[k]:=Len[j]+Matrix[j, k]; C[k]:=j) {Якщо всі Visited[k] відзначені, то довжина шляху від i v до k v дорівнює C[k]. Тепер треба перелічити вершини, що входять у найкоротший шлях}.

3 (видача відповіді). {Шлях від iv до kv видається у зворотному порядку наступною процедурою:}

3.1 z:=C[k];

3.2 Видати z

3.3 z:=C[z]. Якщо z =0, то кінець, інакше перейти до 3.2.

**IIІ. Код програми**

unit Unit2;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs, ExtCtrls,

StdCtrls , Grids;

type

TNodeCoord=record

X:Integer;

Y:Integer;

end;

{ TOperForm }

TOperForm = class(TForm)

Input: TButton;

AllDis: TButton;

beginof: TEdit;

CertaindDis: TButton;

Clear: TButton;

endof: TEdit;

ExitButton: TButton;

GraphImage: TImage;

Label1: TLabel;

Label2: TLabel;

Label3: TLabel;

MatrixGrid: TStringGrid;

StringGrid1: TStringGrid;

procedure beginofChange ( Sender: TObject ) ;

procedure endofChange ( Sender: TObject ) ;

procedure InputClick ( Sender: TObject ) ;

procedure AllDisClick ( Sender: TObject ) ;

procedure CertaindDisClick ( Sender: TObject ) ;

procedure ClearClick ( Sender: TObject ) ;

procedure ExitButtonClick(Sender: TObject);

procedure NodesPaint;

procedure EdgesPaint;

private

{ private declarations }

public

{ public declarations }

end;

var

OperForm: TOperForm;

CoordArr: array [1..20] of TNodeCoord;

numNodes : integer = 15;

implementation

{$R \*.lfm}

{ TOperForm }

procedure TOperForm.ExitButtonClick(Sender: TObject);

begin

Close;

end;

procedure TOperForm.InputClick ( Sender: TObject ) ;

var

myfile1:TextFile;

str\_tmp1: String;

index\_tmp, i : integer;

begin

AssignFile(myfile1,'DATA\P4.TXT');

Reset(myfile1);

index\_tmp:=-1;

readln( myfile1, str\_tmp1 );

readln( myfile1, str\_tmp1 );

while not eof(myfile1) do

begin

index\_tmp:=index\_tmp+1;

readln( myfile1, str\_tmp1 );

MatrixGrid.Cells[ (index\_tmp mod (MatrixGrid.ColCount-1))+1,

(index\_tmp div (MatrixGrid.RowCount-1))+1 ]:= str\_tmp1;

end;

closefile(myfile1);

for i:=0 to MatrixGrid.RowCount-1 do

begin

MatrixGrid.Cells[0,i] := IntToStr(i);

MatrixGrid.Cells[i,0] := IntToStr(i);

end;

NodesPaint;

EdgesPaint;

end;

procedure TOperForm.endofChange ( Sender: TObject ) ;

begin

if ( endof.Text <> '' ) and ( beginof.Text <> '' ) then

CertaindDis.Enabled := true

else

CertaindDis.Enabled := false;;

end;

procedure TOperForm.beginofChange ( Sender: TObject ) ;

begin

if beginof.Text <> '' then

AllDis.Enabled := true

else

AllDis.Enabled := false;

end;

procedure TOperForm.AllDisClick ( Sender: TObject ) ;

var

count, index, min, start, u, i,finish, endOfArray : integer;

distance : array [ 1..15 ] of integer;

visited : array [ 1..15 ] of boolean;

steps : array [ 1..20 ] of String;

begin

endOfArray := MatrixGrid.RowCount-1;

start := StrToInt(beginof.Text);

for i := 1 to endOfArray do

begin

distance[ i ] := maxint;

visited[ i ] := false;

end;

distance[ start ] := 0;

for count := 1 to endOfArray do

begin

min := maxint;

for i := 1 to endOfArray do

begin

if ( not visited[ i ] ) and ( distance [ i ] <=min ) then

begin

min := distance[ i ];

index := i;

end;

end;

u := index;

visited[ u ] := true;

for i := 1 to endOfArray do

if ( not visited[ i ] ) and ( MatrixGrid.Cells[ u , i] <> IntToStr(0) ) and

( distance[ u ] <> maxint ) and

( distance[ u ] + StrToInt( MatrixGrid.Cells[ u , i ] ) < distance [ i ] ) then

distance [ i ] := distance[ u ] + StrToInt( MatrixGrid.Cells[ u , i ] ) ;

end;

for i := 1 to endOfArray do

if distance[ i ] <> maxint then

StringGrid1.Cells[ 0 , i-1] := IntToStr( start ) + ' > ' + IntToStr( i ) +

' = ' + IntToStr( distance[ i ] )

else

StringGrid1.Cells[ 0 , i-1] := IntToStr( start ) + ' > ' + IntToStr( i ) +

' = ' + 'Перехід неможливий' ;

end;

procedure TOperForm.CertaindDisClick ( Sender: TObject ) ;

var

count, index, min, start, u, i,finish, endOfArray,j : integer;

distance : array [ 1..15 ] of integer;

visited : array [ 1..15 ] of boolean;

steps : array [ 1..20 ] of integer;

begin

for i := 0 to StringGrid1.RowCount-1 do

StringGrid1.Cells[0,i] := '';

endOfArray := MatrixGrid.RowCount-1;

start := StrToInt(beginof.Text);

finish := StrToInt(endof.Text);

if finish<15 then

if start<>finish then

begin

for i := 1 to endOfArray do

begin

distance[ i ] := maxint;

visited[ i ] := false;

steps[i] := start;

end;

distance[ start ] := 0;

steps[start] := 0;

for count := 1 to endOfArray do

begin

min := maxint;

for i := 1 to endOfArray do

begin

if ( not visited[ i ] ) and ( distance [ i ] <=min ) then

begin

min := distance[ i ];

index := i;

end;

end;

u := index;

visited[ u ] := true;

for i := 1 to endOfArray do

if ( not visited[ i ] ) and ( MatrixGrid.Cells[ u , i] <> IntToStr(0) ) and

( distance[ u ] <> maxint ) and

( distance[ u ] + StrToInt( MatrixGrid.Cells[ u , i ] ) < distance [ i ] ) then

begin

distance [ i ] := distance[ u ] + StrToInt( MatrixGrid.Cells[ u , i ] ) ;

steps[i]:=u;

end;

end;

j:=0;

if distance[StrToInt(endof.Text)]<=100 then

begin

GraphImage.Canvas.Pen.Width:= 100;

GraphImage.Canvas.Pen.Color:=clRed;

nodespaint;

finish:=steps[finish];

StringGrid1.Cells[0, 0] := (endof.Text);

GraphImage.Canvas.MoveTo(CoordArr[StrToInt(endof.text)].X,CoordArr[StrToInt(endof.text)].Y);

GraphImage.Canvas.LineTo(CoordArr[finish].X,CoordArr[finish].Y);

while finish <> 0 do

begin

GraphImage.Canvas.MoveTo(CoordArr[finish].X,CoordArr[finish].Y);

StringGrid1.Cells[0,0] := StringGrid1.Cells[0,0] + ' <-' + IntToStr(finish);

finish:=steps[finish];

if finish <>0 then

GraphImage.Canvas.LineTo(CoordArr[finish].X,CoordArr[finish].Y)

end;

StringGrid1.Cells[0,0] := StringGrid1.Cells[0,0] + ' Відстань дорівнює: ' +

IntToStr( distance[StrToInt(endof.Text)]) ;

end

else

begin

StringGrid1.Cells[0,0]:='Неможливо знайти шлях';

end;

end

else

begin

StringGrid1.Cells[0,0]:='Вершини співвпадають';

end;

end;

procedure TOperForm.ClearClick ( Sender: TObject ) ;

Var

i : integer;

begin

endof.Text:= '';

beginof.Text:='';

for i := 0 to StringGrid1.RowCount-1 do

StringGrid1.Cells[0,i] := '';

NodesPaint;

EdgesPaint;

end;

procedure TOperForm.NodesPaint;

var i:Integer;

x,y: integer;

x0,y0: integer;

a: Extended;

h: integer;

begin

GraphImage.Canvas.Pen.Width:= 1;

GraphImage.Canvas.Pen.Color:=clBlack;

GraphImage.Canvas.Brush.Color := clwhite;

GraphImage.Canvas.Rectangle(0, 0, GraphImage.Width, GraphImage.Height);

GraphImage.Canvas.Font.Size:=12;

X0:=Round( GraphImage.Width / 2 );

Y0:=Round( GraphImage.Height / 2 );

a:=0;

h:=1;

GraphImage.Canvas.Pen.Width:= 2;

GraphImage.Canvas.Pen.Color:=clBlack;

GraphImage.Canvas.Font.Size:=12;

While a<360 do

begin

CoordArr[h].X:=x0+Round( 220 \* cos(a\*2\*pi/360));

CoordArr[h].Y:=x0-Round( 220 \* sin(a\*2\*pi/360));

GraphImage.Canvas.MoveTo(CoordArr[h].X,CoordArr[h].Y);

GraphImage.Canvas.Ellipse(CoordArr[h].X-4,CoordArr[h].Y-4,CoordArr[h].X+4,CoordArr[h].Y+4);

x:=x0+Round( (220+25) \* cos(a\*2\*pi/360));

y:=y0-Round( (220+25) \* sin(a\*2\*pi/360));

GraphImage.Canvas.TextOut(x-2,y-2,IntToStr(h));

Inc(h);

a:=a+360/NumNodes;

end;

end;

procedure TOperForm.EdgesPaint;

var i,j:Integer;

x1,y1:Integer;

Angle:Extended;

Weight:Integer;

WL1,WL2:TNodeCoord;

Len:Extended;

QX,QY:Integer;

begin

GraphImage.Canvas.Font.Size:=8;

GraphImage.Canvas.Pen.Color:=clBlack;

For i:=1 to NumNodes do

For j:=1 to NumNodes do

begin

Weight:=StrToInt(MatrixGrid.Cells[i,j]);

If Weight>0 then

begin

If i<>j then

begin

GraphImage.Canvas.MoveTo(CoordArr[i].X,CoordArr[i].Y);

GraphImage.Canvas.LineTo(CoordArr[j].X,CoordArr[j].Y);

GraphImage.Canvas.Pen.Width:= 4;

QX:=(CoordArr[j].X-CoordArr[i].X)\*(CoordArr[j].X-CoordArr[i].X);

QY:=(CoordArr[j].Y-CoordArr[i].Y)\*(CoordArr[j].Y-CoordArr[i].Y);

Len:=sqrt(QX+QY);

x1:=Round((CoordArr[j].X-CoordArr[i].X)\*20/Len)+CoordArr[i].X;

y1:=Round((CoordArr[j].Y-CoordArr[i].Y)\*20/Len)+CoordArr[i].Y;

GraphImage.Canvas.MoveTo(x1,y1);

GraphImage.Canvas.LineTo(CoordArr[i].X,CoordArr[i].Y);

GraphImage.Canvas.Pen.Width:= 2;

WL1.X:=(CoordArr[j].X+CoordArr[i].X) div 2;

WL1.Y:=(CoordArr[j].Y+CoordArr[i].Y) div 2;

WL2.X:=(CoordArr[j].X+WL1.X) div 2;

WL2.Y:=(CoordArr[j].Y+WL1.Y) div 2;

GraphImage.Canvas.TextOut(WL2.X,WL2.Y,MatrixGrid.Cells[i,j]);

end else

begin

Angle:=(j-1)\*360/NumNodes;

x1:=CoordArr[i].X+Round( 30 \* cos(Angle\*2\*pi/360));

y1:=CoordArr[i].Y-Round( 30 \* sin(Angle\*2\*pi/360));

GraphImage.Canvas.Ellipse(x1-30,y1-30,x1+30,y1+30);

WL2.X:=300+Round( 280 \* cos(Angle\*2\*pi/360));

WL2.Y:=300-Round( 280 \* sin(Angle\*2\*pi/360));

GraphImage.Canvas.TextOut(WL2.X,WL2.Y,MatrixGrid.Cells[i,j]);

end;

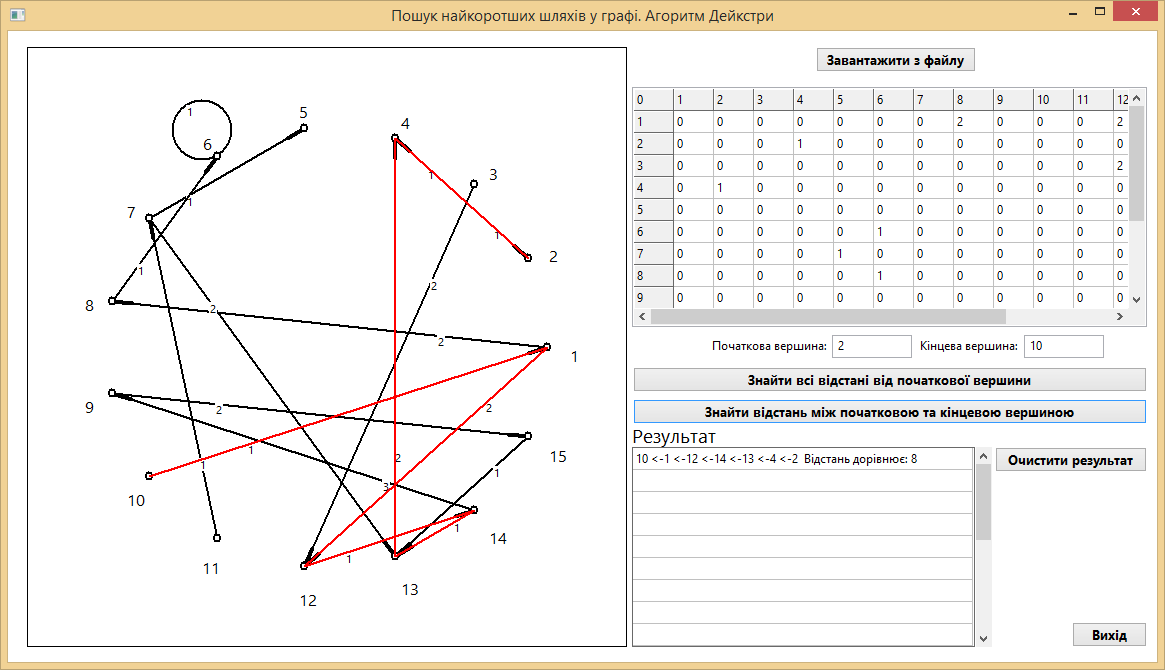
end;

end;

end;

end.

**IV. Результат**



**V. Висновок та аналіз результатів**

У ході лабораторної роботи було закріплено знання з теорії графів та було розроблено програму для пошуку найкоротшого шляху між двома вершинами у графі на основі алгоритму Дейкстри.

Отримані результати підтверджують правильність роботи програми.