**УО «Белорусский государственный университет информатики и радиоэлектроники»**

**Кафедра ПОИТ**

**Отчёт по лабораторной работе №7.1**

**По предмету**

**Основы алгоритмизации и программирования**

**Вариант 6 (9)**

**Выполнил:**

**Гладкий М.Г.**

**Проверила:**

**Данилова Г.В.**

**Группа 851001**

**Минск 2019**

**Задание:**

Граф задан матрицей инциденций. Разработать программу нахождения всех гамильтоновых циклов в графе. Граф визуализировать. Найденный цикл выделить цветом.

**Delphi 10:**

**Main.pas**

**unit** Main;

**interface**

**uses**

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs,

Vcl.Menus, Vcl.Grids, Vcl.StdCtrls;

**type**

TMainF = **class**(TForm)

SG: TStringGrid;

MainMenu: TMainMenu;

OpenFile: TOpenDialog;

PopupMenu: TPopupMenu;

FileMenu: TMenuItem;

Open: TMenuItem;

Help: TMenuItem;

Info: TMenuItem;

Developer: TMenuItem;

TransformBtn: TButton;

SetSizeBtn: TButton;

VerticesLbl: TLabel;

VertEdit: TEdit;

EdgesLbl: TLabel;

EdgeEdit: TEdit;

**procedure** InfoClick(Sender: TObject);

**procedure** DeveloperClick(Sender: TObject);

**procedure** OpenClick(Sender: TObject);

**procedure** FormCloseQuery(Sender: TObject; **var** CanClose: Boolean);

**procedure** SGKeyPress(Sender: TObject; **var** Key: Char);

**procedure** FormCreate(Sender: TObject);

**procedure** TransformBtnClick(Sender: TObject);

**procedure** VertEditChange(Sender: TObject);

**procedure** VertEditKeyPress(Sender: TObject; **var** Key: Char);

**procedure** SetSizeBtnClick(Sender: TObject);

**procedure** EdgeEditChange(Sender: TObject);

**procedure** EdgeEditKeyPress(Sender: TObject; **var** Key: Char);

**procedure** EdgeEditKeyUp(Sender: TObject; **var** Key: Word;

Shift: TShiftState);

**end**;

**var**

MainF: TMainF;

**implementation**

{$R \*.dfm}

**uses** Graph;

**procedure** TMainF.DeveloperClick(Sender: TObject);

**begin**

MessageDlg('Developer: Gladkiy Maksim, gp.851001', mtInformation,

[mbOk], 0);

**end**;

**procedure** TMainF.InfoClick(Sender: TObject);

**begin**

MessageDlg('This program finds Hamiltonian cycle in the graph.' + #13#10

+ 'Vertices[3..5].' + #13#10 + 'Edges[3..10].' + #13#10 +

'In table cells use 0 or 1.', mtInformation, [mbOk], 0);

**end**;

**procedure** TMainF.TransformBtnClick(Sender: TObject);

**var**

i, j: ShortInt;

IsEmpty: Boolean;

**begin**

IsEmpty := false;

**for** j := 1 **to** SG.RowCount - 1 **do**

**for** i := 1 **to** SG.ColCount - 1 **do**

**if** (SG.Cells[i, j] = '') **then**

**begin**

IsEmpty := true;

SG.Cells[i, j] := '0';

**end**;

**if** IsEmpty **then**

MessageDlg('Empty cells were filled with zeros.',

mtConfirmation, [mbOk], 0);

GraphF.Showmodal;

**end**;

**procedure** TMainF.FormCloseQuery(Sender: TObject; **var** CanClose: Boolean);

**var**

ButtonSelected: Byte;

**begin**

ButtonSelected := MessageDlg('Are you sure you want to exit?',

mtConfirmation, [mbYes,mbNo], 0);

**if** ButtonSelected <> mrYes **then**

CanClose := False;

**end**;

**procedure** TMainF.FormCreate(Sender: TObject);

**var**

i: ShortInt;

**begin**

**for** i := 1 **to** 5 **do**

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

**for** i := 1 **to** 10 **do**

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

SG.Cells[0, 0] := 'H';

SG.FixedCols := 1;

SG.FixedRows := 1;

**end**;

**procedure** TMainF.OpenClick(Sender: TObject);

**var**

InputFile: TextFile;

i, j, Temp: ShortInt;

IsCorrect: Boolean;

**begin**

**if** OpenFile.Execute **then**

**begin**

VertEdit.Text := '';

EdgeEdit.Text := '';

**try**

AssignFile(InputFile, OpenFile.FileName);

Reset(InputFile);

**if** EoF(InputFile) **then**

**begin**

MessageDlg('This file is empty. Try again.',

mtError, [mbRetry], 0);

CloseFile(InputFile);

**end**

**else**

**begin**

**repeat**

Read(InputFile, Temp);

**if** (Temp <> 0) **and** (Temp <> 1) **then**

IsCorrect := false

**until** EoF(InputFile) **or not**(IsCorrect);

**if** IsCorrect **then**

**begin**

Reset(InputFile);

j := 0;

**repeat**

i := 0;

**repeat**

Read(InputFile, Temp);

Inc(i);

**until** EoLn(InputFile);

inc(j);

**until** EoF(InputFile);

VertEdit.Text := IntToStr(j);

EdgeEdit.Text := IntToStr(i);

SetSizeBtn.Click;

Reset(InputFile);

**for** j := 1 **to** SG.RowCount - 1 **do**

**for** i := 1 **to** SG.ColCount - 1 **do**

**begin**

Read(InputFile, Temp);

SG.Cells[i, j] := IntToStr(Temp);

**end**;

TransformBtn.Click;

CloseFile(InputFile);

**end**

**else**

**begin**

MessageDlg('Check entered data. Try again.',

mtError, [mbRetry], 0);

CloseFile(InputFile);

**end**;

**end**;

**except**

MessageDlg('Check entered data. Try again.', mtError,

[mbRetry], 0);

CloseFile(InputFile);

**end**;

**end**;

**end**;

**procedure** TMainF.SetSizeBtnClick(Sender: TObject);

**begin**

SG.ColCount := StrToInt(EdgeEdit.Text) + 1;

SG.RowCount := StrToInt(VertEdit.Text) + 1;

SetSizeBtn.Enabled := false;

SG.Enabled := true;

TransformBtn.Enabled := true;

**end**;

**procedure** TMainF.SGKeyPress(Sender: TObject; **var** Key: Char);

**begin**

**if** (Length(SG.Cells[SG.Col, SG.Row]) = 1) **and** (Key <> #8) **then**

Key := #0;

**if** (Key <> '0') **and** (Key <> '1') **and** (Key <> #8) **then**

Key := #0;

**end**;

**procedure** TMainF.VertEditChange(Sender: TObject);

**var**

i, j: ShortInt;

**begin**

TransformBtn.Enabled := false;

SG.Enabled := false;

**for** j := 1 **to** SG.RowCount - 1 **do**

**for** i := 1 **to** SG.ColCount - 1 **do**

SG.Cells[i, j] := '';

**if** (Length(VertEdit.Text) = 1)**and**(Length(EdgeEdit.Text) > 0) **then**

SetSizeBtn.Enabled := true

**else**

SetSizeBtn.Enabled := false;

**end**;

**procedure** TMainF.EdgeEditChange(Sender: TObject);

**var**

i, j: ShortInt;

**begin**

TransformBtn.Enabled := false;

SG.Enabled := false;

**for** j := 1 **to** SG.RowCount - 1 **do**

**for** i := 1 **to** SG.ColCount - 1 **do**

SG.Cells[i, j] := '';

**if** (Length(VertEdit.Text) = 1)**and**(Length(EdgeEdit.Text) > 0) **then**

SetSizeBtn.Enabled := true

**else**

SetSizeBtn.Enabled := false;

**end**;

**procedure** TMainF.EdgeEditKeyPress(Sender: TObject; **var** Key: Char);

**var**

Numerals: **set of** char;

**begin**

Numerals := ['0', '1', '3'..'9', #8];

**if not** (Key **in** Numerals) **then**

Key := #0;

**if** (Length(EdgeEdit.Text) = 2) **and** (Key <> #8) **then**

Key := #0;

**if** (Length(EdgeEdit.Text) = 0) **and** (Key = '0') **then**

Key := #0;

**end**;

**procedure** TMainF.EdgeEditKeyUp(Sender: TObject; **var** Key: Word;

Shift: TShiftState);

**begin**

**if** (Length(EdgeEdit.Text) > 1) **and** (StrToInt(EdgeEdit.Text) > 10) **then**

EdgeEdit.Clear;

**end**;

**procedure** TMainF.VertEditKeyPress(Sender: TObject; **var** Key: Char);

**var**

Numerals: **set of** char;

**begin**

Numerals := ['3'..'5', #8];

**if not** (Key **in** Numerals) **then**

Key := #0;

**if** (Length(VertEdit.Text) = 1) **and** (Key <> #8) **then**

Key := #0;

**end**;

**end**.

**Graph.pas**

**unit** Graph;

**interface**

**uses**

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs,

Vcl.Menus, Vcl.StdCtrls;

**type**

TGraphF = **class**(TForm)

PopupMenu: TPopupMenu;

MainMenu: TMainMenu;

FileMenu: TMenuItem;

Save: TMenuItem;

SaveFile: TSaveDialog;

NumEdit: TEdit;

FindBtn: TButton;

ResultLbl: TLabel;

Inf: TMenuItem;

**procedure** FormPaint(Sender: TObject);

**procedure** SaveClick(Sender: TObject);

**procedure** NumEditChange(Sender: TObject);

**procedure** NumEditKeyPress(Sender: TObject; **var** Key: Char);

**procedure** FormShow(Sender: TObject);

**procedure** FindBtnClick(Sender: TObject);

**function** HamiltCycle(k: Byte): String;

**procedure** InfClick(Sender: TObject);

**end**;

**var**

GraphF: TGraphF;

**implementation**

{$R \*.dfm}

**uses** Main;

**var**

Angle: Real;

VertArr: **array**[1..5]**of array**[1..10] **of** ShortInt;

BuffArr: **array**[1..5] **of** Boolean;

l: ShortInt;

ResultL: String;

**function** TGraphF.HamiltCycle(k: Byte): String;

**var**

m: ShortInt;

**begin**

m := 1;

**while** l <> MainF.SG.RowCount - 1 **do**

**if** (BuffArr[VertArr[k][m]]) **then**

**begin**

**if** l <> MainF.SG.RowCount - 2 **then**

ResultL := ResultL + IntToStr(VertArr[k][m]) + ' '

**else**

ResultL := ResultL + IntToStr(VertArr[k][m]) + ' '

+ NumEdit.Text;

BuffArr[VertArr[k][m]] := false;

inc(l);

Canvas.Brush.Color := RGB(250, 0, 0);

Canvas.Ellipse(Trunc(Cos(Angle \* VertArr[k][m]) \* 120 + 200),

Trunc(Sin(Angle \* VertArr[k][m]) \* 120 + 200),

Trunc(Cos(Angle \* VertArr[k][m]) \* 120 + 160),

Trunc(Sin(Angle \* VertArr[k][m]) \* 120 + 160));

Canvas.Font.Size := 14;

Canvas.TextOut(Trunc(Cos(Angle \* VertArr[k][m]) \* 120 + 175),

Trunc(Sin(Angle \* VertArr[k][m]) \* 120 + 168),

IntToStr(VertArr[k][m]));

Sleep(1000);

HamiltCycle(VertArr[k][m]);

**end**

**else**

**begin**

**if** (VertArr[k][m + 1] <> 0)**and**(m < 30) **then**

inc(m)

**else**

**begin**

MessageDlg('There is no Hamilton Cycle.',

mtInformation, [mbOk], 0);

Save.Enabled := false;

ResultL := '';

l := MainF.SG.RowCount - 1;

**break**;

**end**;

**end**;

HamiltCycle := ResultL;

**end**;

**procedure** TGraphF.InfClick(Sender: TObject);

**begin**

MessageDlg('Use numbers corresponding to vertex numbers.',

mtInformation, [mbOk], 0);

**end**;

**procedure** TGraphF.FindBtnClick(Sender: TObject);

**begin**

Save.Enabled := true;

NumEdit.Enabled := false;

FindBtn.Enabled := false;

l := 1;

ResultL := 'Hamilton Cycle is: ' + NumEdit.Text + ' ';

BuffArr[StrToInt(NumEdit.Text)] := false;

Canvas.Brush.Color := RGB(250, 0, 0);

Canvas.Ellipse(Trunc(Cos(Angle \* StrToInt(NumEdit.Text)) \* 120 + 200),

Trunc(Sin(Angle \* StrToInt(NumEdit.Text)) \* 120 + 200),

Trunc(Cos(Angle \* StrToInt(NumEdit.Text)) \* 120 + 160),

Trunc(Sin(Angle \* StrToInt(NumEdit.Text)) \* 120 + 160));

Canvas.Font.Size := 14;

Canvas.TextOut(Trunc(Cos(Angle \* StrToInt(NumEdit.Text)) \* 120 + 175),

Trunc(Sin(Angle \* StrToInt(NumEdit.Text)) \* 120 + 168),

IntToStr(StrToInt(NumEdit.Text)));

Sleep(1000);

ResultLbl.Caption := HamiltCycle(StrToInt(NumEdit.Text));

**end**;

**procedure** TGraphF.FormPaint(Sender: TObject);

**var**

i, j, k: ShortInt;

**begin**

Angle := 2 \* Pi / (MainF.SG.RowCount - 1);

Canvas.Pen.Color := RGB(0, 0, 0);

**for** j := 1 **to** MainF.SG.RowCount **do**

**begin**

l := 1;

**for** i := 1 **to** MainF.SG.ColCount **do**

**if** (MainF.SG.Cells[i, j] = '1') **then**

**for** k := 1 **to** MainF.SG.RowCount **do**

**if** (MainF.SG.Cells[i, k] = '1')**and**(k <> j) **then**

**begin**

Canvas.MoveTo(Trunc(Cos(Angle \* j) \* 120 + 175),

Trunc(Sin(Angle \* j) \* 120 + 168));

Canvas.LineTo(Trunc(Cos(Angle \* k) \* 120 + 175),

Trunc(Sin(Angle \* k) \* 120 + 168));

**end**;

**end**;

**for** i := 1 **to** MainF.SG.RowCount - 1 **do**

**begin**

Canvas.Brush.Color := RGB(Random(256), Random(256), Random(256));

Canvas.Ellipse(Trunc(Cos(Angle \* i) \* 120 + 200), Trunc(Sin(Angle \*

i) \* 120 + 200), Trunc(Cos(Angle \* i) \* 120 + 160),

Trunc(Sin(Angle \* i) \* 120 + 160));  
 Canvas.Font.Size := 14;

Canvas.TextOut(Trunc(Cos(Angle \* i) \* 120 + 175), Trunc(Sin(Angle \*

i) \* 120 + 168), IntToStr(i));

**end**;

**end**;

**procedure** TGraphF.SaveClick(Sender: TObject);

**var**

OutputFile: TextFile;

MyFile: String;

ButtonSelected , i, j: byte;

**begin**

**if** SaveFile.Execute **then**

**begin**

MyFile := SaveFile.FileName;

**if** FileExists(MyFile) **then**

**begin**

ButtonSelected := MessageDlg('Do you want to rewrite the file?',

mtConfirmation, [mbYes,mbNo], 0);

AssignFile(OutputFile, MyFile);

**if** ButtonSelected = MrYes **then**

Rewrite(OutputFile)

**else**

**begin**

Append(outputFile);

WriteLn(OutputFile);

**end**;

Write(OutputFile, ResultLbl.Caption);

CloseFile(OutputFile);

**end**;

**end**;

**end**;

**procedure** TGraphF.FormShow(Sender: TObject);

**var**

i, j, k: ShortInt;

**begin**

NumEdit.Text := '';

NumEdit.Enabled := true;

Save.Enabled := false;

ResultLbl.Caption := '';

FindBtn.Enabled := false;

**for** j := 1 **to** MainF.SG.RowCount **do**

**begin**

l := 1;

**for** i := 1 **to** MainF.SG.ColCount **do**

**if** (MainF.SG.Cells[i, j] = '1') **then**

**for** k := 1 **to** MainF.SG.RowCount **do**

**begin**

**if** (MainF.SG.Cells[i, k] = '1')**and**(k <> j) **then**

**begin**

VertArr[j][l] := k;

inc(l);

**end**;

**end**;

**end**;

**for** i := 1 **to** MainF.SG.RowCount **do**

BuffArr[i] := true;

**end**;

**procedure** TGraphF.NumEditChange(Sender: TObject);

**begin**

**if** Length(NumEdit.Text) = 1 **then**

FindBtn.Enabled := true

**else**

FindBtn.Enabled := false;

**end**;

**procedure** TGraphF.NumEditKeyPress(Sender: TObject; **var** Key: Char);

**var**

Numerals: **set of** char;

i: ShortInt;

**begin**

Numerals := [#8];

**for** i := 1 **to** MainF.SG.RowCount - 1 **do**

Include(Numerals, AnsiChar(i + 48));

**if not** (Key **in** Numerals) **then**

Key := #0;

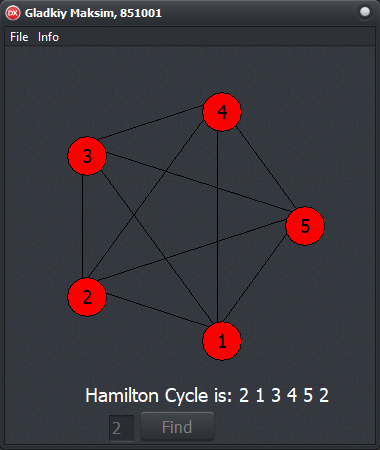
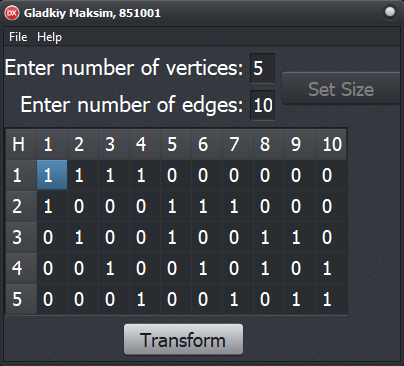
**if** (Length(NumEdit.Text) = 1) **and** (Key <> #8) **then**

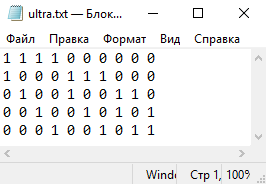
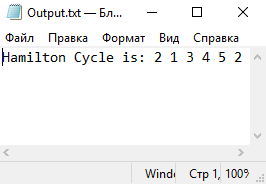
Key := #0;

**end**;

**end**.

**Скриншоты:**

****

** **

**Блок-схема:**



