**УО «Белорусский государственный университет информатики и**

**радиоэлектроники»**

**Кафедра ПОИТ**

**Отчет по лабораторной работе №7.2**

**по предмету**

**Основы Алгоритмизации и Программирования**

**Вариант 1**

**Выполнил**

**Андросов И.С.**

**Проверила**

**Данилова Г.В.**

Группа:

**8**51001

**Минск 2019**

**Задание**

Преобразовать матрицу смежности в список инцидентности.

**Код программы**

**(Delphi)**

**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**

TMainForm = **class**(TForm)

SG: TStringGrid;

MainMenu: TMainMenu;

OpenFile: TOpenDialog;

PopupMenu: TPopupMenu;

FileMenu: TMenuItem;

Open: TMenuItem;

N: TMenuItem;

**Exit**: TMenuItem;

Help: TMenuItem;

AboutTheProgram: TMenuItem;

AboutTheDeveloper: TMenuItem;

TransformBtn: TButton;

SetSizeBtn: TButton;

SetSizeLbl: TLabel;

SizeEdit: TEdit;

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

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

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

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

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

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

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

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

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

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

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

**end**;

**var**

MainForm: TMainForm;

**implementation**

{$R \*.dfm}

**uses** Graph;

**procedure** TMainForm.AboutTheDeveloperClick(Sender: TObject);

**begin**

MessageDlg('This program is developed by Ilya Androsov' + #13#10 + 'BSUIR 2019',

mtInformation, [mbOk], 0);

**end**;

**procedure** TMainForm.AboutTheProgramClick(Sender: TObject);

**begin**

MessageDlg('This program implements a wide search in a graph from a given vertex.',

mtInformation, [mbOk], 0);

**end**;

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

**var**

i, j, k: 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);

GraphForm.ShowModal;

**end**;

**procedure** TMainForm.ExitClick(Sender: TObject);

**begin**

Close;

**end**;

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

**var**

i: ShortInt;

**begin**

**for** i := 1 **to** 9 **do**

**begin**

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

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

**end**;

SG.FixedCols := 1;

SG.FixedRows := 1;

**end**;

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

**var**

InputFile: TextFile;

i, j, Temp: ShortInt;

IsCorrect: Boolean;

**begin**

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

**begin**

SizeEdit.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);

i := 0;

**repeat**

Read(InputFile, j);

Inc(i);

**until** EoLn(InputFile);

SizeEdit.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** TMainForm.SetSizeBtnClick(Sender: TObject);

**begin**

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

SG.RowCount := SG.ColCount;

SetSizeBtn.Enabled := false;

SG.Enabled := true;

TransformBtn.Enabled := true;

**end**;

**procedure** TMainForm.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** TMainForm.SizeEditChange(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(SizeEdit.Text) = 1 **then**

SetSizeBtn.Enabled := true

**else**

SetSizeBtn.Enabled := false;

**end**;

**procedure** TMainForm.SizeEditKeyPress(Sender: TObject; **var** Key: Char);

**var**

Numerals: **set of** char;

**begin**

Numerals := ['2'..'9', #8];

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

Key := #0;

**if** (Length(SizeEdit.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,

Vcl.ExtCtrls;

**type**

TGraphForm = **class**(TForm)

PopupMenu: TPopupMenu;

MainMenu: TMainMenu;

FileMenu: TMenuItem;

Save: TMenuItem;

N: TMenuItem;

**Exit**: TMenuItem;

SaveFile: TSaveDialog;

SearchBtn: TButton;

ResultLbl: TLabel;

ValueEdit: TEdit;

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

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

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

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

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

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

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

**end**;

**var**

GraphForm: TGraphForm;

VertexArray: **array of** Boolean;

Angle: Real;

**implementation**

{$R \*.dfm}

**uses** Main, VertexQueue;

**procedure** TGraphForm.ExitClick(Sender: TObject);

**begin**

Close;

**end**;

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

**var**

i, j: ShortInt;

**begin**

Angle := 2 \* Pi / (MainForm.SG.ColCount - 1);

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

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

**for** j := 1 **to** MainForm.SG.ColCount - 1 **do**

**if** MainForm.SG.Cells[j, i] = '1' **then**

**begin**

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

120 + 168));

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

120 + 168));

**end**;

**for** i := 1 **to** MainForm.SG.ColCount - 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** TGraphForm.FormShow(Sender: TObject);

**var**

i: ShortInt;

**begin**

ValueEdit.Text := '';

ValueEdit.Enabled := true;

Save.Enabled := false;

ResultLbl.Caption := '';

SearchBtn.Enabled := false;

SetLength(VertexArray, MainForm.SG.ColCount - 1);

**for** i := 0 **to** High(VertexArray) **do**

VertexArray[i] := false;

**end**;

**procedure** TGraphForm.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** TGraphForm.SearchBtnClick(Sender: TObject);

**var**

MyQueue: TMyQueue;

Item, i, j: ShortInt;

**begin**

ValueEdit.Enabled := false;

Save.Enabled := true;

SearchBtn.Enabled := false;

ResultLbl.Caption := 'The path is: ';

MyQueue := TMyQueue.Create;

MyQueue.Enqueue(StrToInt(ValueEdit.Text));

VertexArray[StrToInt(ValueEdit.Text) - 1] := true;

**Repeat**

i := MyQueue.Dequeue;

ResultLbl.Caption := ResultLbl.Caption + IntToStr(i) + ' ';

Canvas.Brush.Color := RGB(255, 255, 255);

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));

**for** j := 1 **to** MainForm.SG.ColCount **do**

**if** (MainForm.SG.Cells[j, i] = '1') **and** (**not**(VertexArray[j - 1])) **then**

**begin**

MyQueue.Enqueue(j);

VertexArray[j - 1] := true;

**end**;

sleep(500);

**Until** MyQueue.IsEmpty;

MyQueue.Destroy;

**end**;

**procedure** TGraphForm.ValueEditChange(Sender: TObject);

**var**

i, j: ShortInt;

**begin**

SearchBtn.Enabled := false;

Save.Enabled := false;

ResultLbl.Caption := '';

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

SearchBtn.Enabled := true

**else**

SearchBtn.Enabled := false;

**end**;

**procedure** TGraphForm.ValueEditKeyPress(Sender: TObject; **var** Key: Char);

**var**

Numerals: **set of** char;

i: ShortInt;

**begin**

Numerals := [#8];

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

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

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

Key := #0;

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

Key := #0;

**end**;

**end**.

**VertexQueue.pas**

**unit** VertexQueue;

**interface**

**type**

TElement = ShortInt;

PNodePointer = ^TNode;

TNode = **record**

Value: TElement;

Next: PNodePointer;

**end**;

TMyQueue = **class**(TObject)

**private**

Head: PNodePointer;

Tail: PNodePointer;

**public**

**constructor** Create;

**destructor** Destroy;

**procedure** Enqueue(Item: TElement);

**function** Dequeue: TElement;

**function** IsEmpty: Boolean;

**end**;

**implementation**

**uses** Main;

**constructor** TMyQueue.Create;

**begin**

Head:= nil;

Tail:= nil;

**end**;

**destructor** TMyQueue.Destroy;

**begin**

**while not** IsEmpty **do**

Dequeue;

Head:= nil;

Tail:= nil;

**end**;

**procedure** TMyQueue.Enqueue(Item: TElement);

**var**

Temp: PNodePointer;

**begin**

**New**(Temp);

Temp^.Value := Item;

Temp^.Next := nil;

**if** (IsEmpty) **then**

**begin**

Head:= Temp;

Tail:= Temp;

**end**

**else**

**begin**

Tail^.next := Temp;

Tail := Temp;

**end**;

**end**;

**function** TMyQueue.Dequeue: TElement;

**var**

Temp: PNodePointer;

**begin**

**if not** IsEmpty **then**

**begin**

Temp := Head;

Head := Head^.Next;

Result := temp^.value;

Dispose(temp);

**end**

**else**

Result := 0;

**end**;

**function** TMyQueue.IsEmpty: boolean;

**begin**

**if** Head = nil **then**

IsEmpty := true

**else**

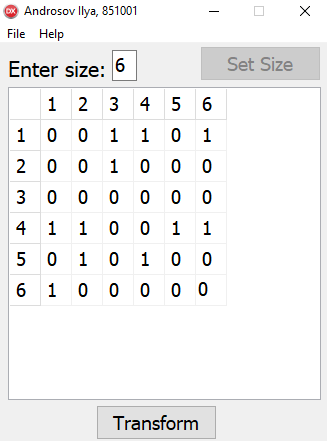
IsEmpty := false;

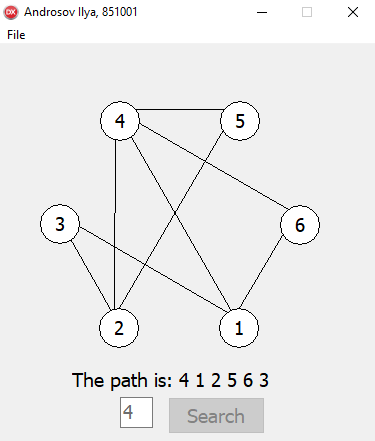
**end**;

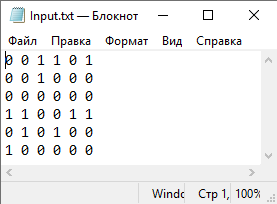
**end**.

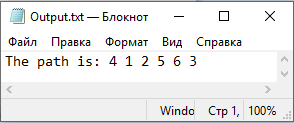
**Скриншоты**

**(Delphi)**

****

****

****

****

**Схема алгоритма**





