УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №7.1

по предмету

Основы алгоритмизации и программирования

Вариант 14

Выполнил:

Крутько А.А.

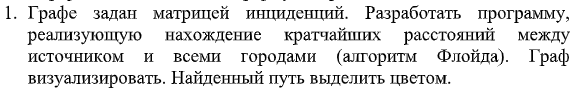
Проверила:

Данилова Г.В.

Группа 251004

Минск 2023

Задание:



Код программы на **Delphi**:

unit MainUnit;

interface

uses

Winapi.Windows, System.SysUtils, System.Classes, Vcl.Forms, Vcl.Grids, Vcl.StdCtrls, Vcl.NumberBox,

Vcl.Samples.Spin, Vcl.Buttons, ClipBRD, Vcl.ExtCtrls, Vcl.Menus, Math,

Vcl.Controls, Vcl.Graphics, Winapi.Messages, Vcl.Dialogs;

type

TArrOI = Array of Array of Integer;

TPointArr = Array of TPoint;

TRecord = Record

StartEdge, EndEdge: Integer;

end;

TRecordArr = Array of TRecord;

PEdge = ^TEdge;

TEdge = record

EndEdge: Integer;

Weight: Integer;

Next: PEdge;

end;

PArrEdge = Array of PEdge;

TIPEGrid = Class(TStringGrid);

TMainForm = class(TForm)

GridIncidence: TStringGrid;

SpinEdit1: TSpinEdit;

Label1: TLabel;

SpinEdit2: TSpinEdit;

Label2: TLabel;

BitBtn1: TBitBtn;

Timer1: TTimer;

Label3: TLabel;

WeightGrid: TStringGrid;

Label4: TLabel;

PaintBox1: TPaintBox;

AnswerGrid: TStringGrid;

MainMenu1: TMainMenu;

N1: TMenuItem;

N2: TMenuItem;

N3: TMenuItem;

N4: TMenuItem;

N5: TMenuItem;

SaveDialog1: TSaveDialog;

OpenDialog1: TOpenDialog;

procedure Timer1Timer(Sender: TObject);

procedure BitBtn1Click(Sender: TObject);

Procedure MakeEmptyHead;

Function CheckWeightGrid: Boolean;

Procedure FillHead(ArrIncidence: TArrOI);

Procedure InsertInHead(I, J, Weight: Integer);

Function GetFloidPaths: TArrOI;

procedure PaintBox1Paint(Sender: TObject);

Function GetVertex: TPointArr;

Procedure WriteAnswer;

procedure N2Click(Sender: TObject);

procedure N3Click(Sender: TObject);

procedure N5Click(Sender: TObject);

procedure N4Click(Sender: TObject);

private

Head: PArrEdge;

Dist: TArrOI;

public

end;

const

INF = 1000;

var

MainForm: TMainForm;

implementation

{$R \*.dfm}

Function GetIncidenceMatrix: TArrOI;

Var

Arr: TArrOI;

I, J: Integer;

Begin

SetLength(Arr, MainForm.GridIncidence.RowCount - 1, MainForm.GridIncidence.ColCount - 1);

I := 1;

While I < MainForm.GridIncidence.RowCount do

Begin

J := 1;

While J < MainForm.GridIncidence.ColCount do

Begin

If (MainForm.GridIncidence.Cells[J, I] <> '') and (MainForm.GridIncidence.Cells[J, I] <> '-') Then

Arr[I - 1, J - 1] := StrToInt(MainForm.GridIncidence.Cells[J, I]);

Inc(J);

End;

Inc(I);

End;

GetIncidenceMatrix := Arr;

End;

Function IsGraphCorrect: Boolean;

Var

Arr: TArrOI;

IsCorrect: Boolean;

I, J, Sum: Integer;

One, Two, MinusOne: Integer;

Begin

IsCorrect := True;

Try

Try

Arr := GetIncidenceMatrix;

J := 0;

While IsCorrect and (J < Length(Arr[0])) do

begin

One := 0;

Two := 0;

MinusOne := 0;

For I := 0 to High(Arr) do

Begin

Case Arr[I,J] of

1: Inc(One);

2: Inc(Two);

-1: Inc(MinusOne);

End;

End;

IsCorrect := ((One = 1) and (MinusOne = 1) and (Two = 0)) or

((Two = 1) and (One = 0) and (MinusOne = 0));

Inc(J);

end;

Except

IsCorrect := False;

End;

Finally

Arr := Nil;

End;

IsGraphCorrect := IsCorrect;

End;

Function GetAdjMatrix(Arr: TArrOI): TArrOI;

Var

ArrAdj: TArrOI;

I, J, SecondIndex: Integer;

Begin

SetLength(ArrAdj, Length(Arr), Length(Arr));

For I := 0 to High(Arr[0]) do

Begin

J := 0;

While Arr[J, I] = 0 do

Inc(J);

SecondIndex := J + 1;

While (SecondIndex < Length(Arr)) and (Arr[SecondIndex,I] = 0) do

Inc(SecondIndex);

If SecondIndex = Length(Arr) Then

Inc(ArrAdj[J,J])

Else

Begin

Inc(ArrAdj[J, SecondIndex]);

Inc(ArrAdj[SecondIndex, J]);

End;

End;

GetAdjMatrix := ArrAdj;

End;

Procedure TMainForm.MakeEmptyHead;

Var

I: Integer;

Begin

SetLength(Head, GridIncidence.RowCount - 1);

For I := 0 to High(Head) do

Head[I] := nil;

End;

procedure TMainForm.N2Click(Sender: TObject);

begin

ShowMessage('Крутько Андрей 251004');

end;

procedure TMainForm.N3Click(Sender: TObject);

begin

ShowMessage('Данная программа находит кратчайшие пути между всеми точками по методу Флойда-Уоршелла'#13#10 +

'1. В каждом столбце 1 и -1 или 2 может быть'#13#10 +

'2. 1 – начало ребра, -1 – конец ребра на графе');

end;

procedure Split(Delimiter: Char; Str: string; ListOfStrings: TStringList); StdCall;

begin

ListOfStrings.Clear;

ListOfStrings.Delimiter := Delimiter;

ListOfStrings.StrictDelimiter := True;

ListOfStrings.DelimitedText := Str;

end;

function CheckIncidenceMatrix(Arr: TArrOI): Boolean;

var

I, J, One, Two, MinusOne: Integer;

isCorrect: Boolean;

begin

isCorrect := True;

for I := 0 to High(Arr[0]) do

begin

One := 0;

Two := 0;

MinusOne := 0;

For J := 0 to High(Arr) do

case Arr[J, I] of

1:

Inc(One);

2:

Inc(Two);

-1:

Inc(MinusOne);

end;

isCorrect := ((One = 1) and (MinusOne = 1) and (Two = 0)) or ((Two = 1) and (One = 0) and (MinusOne = 0));

If Not isCorrect then

Break;

end;

CheckIncidenceMatrix := isCorrect;

end;

procedure TMainForm.N4Click(Sender: TObject);

Var

FileInput: TextFile;

Temp: String;

ListTemp: TStringList;

Arr: TArrOI;

ArrWeight: Array of Integer;

I, J, Count, Sum, SizeRow, SizeCol: Integer;

Begin

If OpenDialog1.Execute Then

Begin

Try

Try

AssignFile(FileInput, ChangeFileExt(OpenDialog1.FileName, '.txt'));

Reset(FileInput);

Readln(FileInput, Temp);

SizeRow := StrToInt(Temp);

If (SizeRow < 1) or (SizeRow > 10) Then

raise Exception.Create('Проверьте правильность размера массива');

Readln(FileInput, Temp);

SizeCol := StrToInt(Temp);

If (SizeCol < 1) or (SizeCol > 15) Then

raise Exception.Create('Проверьте правильность размера массива');

SetLength(Arr, SizeRow, SizeCol);

ListTemp := TStringList.Create;

For J := 0 to High(Arr) do

Begin

Readln(FileInput, Temp);

Split(' ', Temp, ListTemp);

If ListTemp.Count = SizeCol Then

Begin

For I := 0 to ListTemp.Count - 1 do

Arr[J, I] := StrToInt(ListTemp[I]);

End

Else

raise Exception.Create('Несовпадение элементов с размерностью списка');

End;

If Not(CheckIncidenceMatrix(Arr)) Then

raise Exception.Create('Проверьте матрицу инцидентности');

SetLength(ArrWeight, Length(Arr[0]));

Readln(FileInput, Temp);

Split(' ', Temp, ListTemp);

If ListTemp.Count = Length(ArrWeight) Then

For I := 0 to ListTemp.Count - 1 do

Begin

If ((StrToInt(ListTemp[I]) > 0) and (StrToInt(ListTemp[I]) < 100)) Then

ArrWeight[I] := StrToInt(ListTemp[I])

Else

raise Exception.Create('Неправильное значение у ребра');

End

Else

Raise Exception.Create('Несовпадение элементов с размерностью списка веса');

WeightGrid.ColCount := Length(ArrWeight);

For I := 0 to High(ArrWeight) do

WeightGrid.Cells[I, 0] := IntToStr(ArrWeight[I]);

GridIncidence.RowCount := 1 + Length(Arr);

GridIncidence.ColCount := 1 + Length(Arr[0]);

For I := 0 to High(Arr) do

For J := 0 to High(Arr[0]) do

GridIncidence.Cells[J + 1, I + 1] := IntToStr(Arr[I, J]);

BitBtn1.Enabled := True;

SpinEdit1.Value := Length(Arr[0]);

SpinEdit2.Value := Length(Arr);

Except

on E : Exception do

Begin

ShowMessage('Поднята ошибка, с сообщением : ' + E.Message);

End;

End;

Finally

CloseFile(FileInput);

End;

End;

end;

procedure TMainForm.N5Click(Sender: TObject);

Var

FileOutput: TextFile;

begin

If SaveDialog1.Execute Then

Begin

Try

Try

AssignFile(FileOutput, ChangeFileExt(SaveDialog1.FileName, '.txt'));

ReWrite(FileOutput);

For Var I := 0 to High(Dist) do

Begin

For Var J := 0 to High(Dist[0]) do

If Dist[I, J] <> Inf Then

Write(FileOutput, IntToStr(Dist[I, J]):4)

Else

Write(FileOutput, 'Infinity ');

Writeln(FileOutput);

End;

Except

On E: Exception do

ShowMessage('Проверьте файл, ошибка ' + E.Message);

End;

Finally

CloseFile(FileOutput);

End;

End;

end;

Procedure DrawCircle(X, Y, R: Integer);

Begin

MainForm.PaintBox1.Canvas.Ellipse(X - R, Y - R,X + R,Y + R);

End;

Function TMainForm.GetVertex: TPointArr;

Var

I: Integer;

Vertex: TPointArr;

Radius: Integer;

Begin

Radius := 40;

SetLength(Vertex, Length(Head));

For I := Low(Vertex) to High(Head) do

Begin

Repeat

Vertex[I].X := Random(PaintBox1.Width);

Until Not((Vertex[I].X < Radius) or (Vertex[I].X > PaintBox1.Width - Radius));

Repeat

Vertex[I].Y := Random(PaintBox1.Height);

Until Not((Vertex[I].Y < Radius) or (Vertex[I].Y > PaintBox1.Height - Radius));

End;

Result := Vertex;

End;

procedure DrawArrowHead(Canvas: TCanvas; X,Y: Integer; Angle,LW: Extended);

var

A1,A2: Extended;

Arrow: array[0..3] of TPoint;

OldWidth: Integer;

const

Beta=0.322;

LineLen=4.74;

CentLen=3;

begin

Angle:=Pi+Angle;

Arrow[0]:=Point(X,Y);

A1:=Angle-Beta;

A2:=Angle+Beta;

Arrow[1]:=Point(X+Round(LineLen\*LW\*Cos(A1)),Y-Round(LineLen\*LW\*Sin(A1)));

Arrow[2]:=Point(X+Round(CentLen\*LW\*Cos(Angle)),Y-Round(CentLen\*LW\*Sin(Angle)));

Arrow[3]:=Point(X+Round(LineLen\*LW\*Cos(A2)),Y-Round(LineLen\*LW\*Sin(A2)));

OldWidth:=Canvas.Pen.Width;

Canvas.Pen.Width:=1;

Canvas.Polygon(Arrow);

Canvas.Pen.Width:=OldWidth

end;

procedure DrawArrow(Canvas: TCanvas; StartPoint, EndPoint: TPoint; LW: Extended);

var

Angle: Extended;

begin

Angle := ArcTan2(StartPoint.Y - EndPoint.Y, EndPoint.X - StartPoint.X);

Canvas.MoveTo(StartPoint.X, StartPoint.Y);

Canvas.LineTo(EndPoint.X - Round(2\*LW\*Cos(Angle)), EndPoint.Y + Round(2\*LW\*Sin(Angle)));

Canvas.Pen.Color := ClBlack;

DrawArrowHead(Canvas, EndPoint.X, EndPoint.Y, Angle, LW);

end;

procedure TMainForm.PaintBox1Paint(Sender: TObject);

Var

Vertex: TPointArr;

I, J, Radius: Integer;

IsCorrect: Boolean;

Temp: PEdge;

StartPoint, EndPoint, TextPos: TPoint;

ArrowPoints: TRecordArr;

begin

Radius := 30;

Repeat

IsCorrect := True;

Vertex := GetVertex;

For I := Low(Vertex) to High(Vertex) do

Begin

For J := I + 1 to High(Head) do

If TPoint.PointInCircle(Vertex[I], Vertex[J], Radius) Then

IsCorrect := False;

End;

Until IsCorrect;

For I := Low(Vertex) to High(Vertex) do

Begin

DrawCircle(Vertex[I].X, Vertex[I].Y, Radius);

PaintBox1.Canvas.TextOut(Vertex[I].X - PaintBox1.Canvas.TextWidth(IntToStr(I + 1)) div 2,

Vertex[I].Y - PaintBox1.Canvas.TextHeight(IntToStr(I + 1)) div 2, IntToStr(I + 1));

End;

for I := Low(Head) to High(Head) do

begin

Temp := Head[I];

while Temp <> nil do

begin

StartPoint := Point(Vertex[I].X + Sign(Vertex[Temp^.EndEdge].X - Vertex[I].X) \* Radius,

Vertex[I].Y);

EndPoint := Point(Vertex[Temp^.EndEdge].X - Sign(Vertex[Temp^.EndEdge].X - Vertex[I].X) \* Radius,

Vertex[Temp^.EndEdge].Y);

For J := Low(ArrowPoints) to High(ArrowPoints) do

Begin

If (I in [ArrowPoints[J].StartEdge, ArrowPoints[J].EndEdge])

and (Temp.EndEdge in [ArrowPoints[J].StartEdge, ArrowPoints[J].EndEdge]) then

Begin

StartPoint.X := StartPoint.X + Sign(StartPoint.X - EndPoint.X) \* Radius;

EndPoint.Y := EndPoint.Y + Sign(EndPoint.Y - StartPoint.Y) \* Radius;

End;

End;

PaintBox1.Canvas.Pen.Color := ClSkyBlue;

PaintBox1.Canvas.Pen.width := 2;

SetLength(ArrowPoints, Length(ArrowPoints) + 1);

ArrowPoints[High(ArrowPoints)].StartEdge := I;

ArrowPoints[High(ArrowPoints)].EndEdge := Temp.EndEdge;

DrawArrow(PaintBox1.Canvas, StartPoint, EndPoint, 4);

Var TempStyle := PaintBox1.Canvas.Brush.Style;

Var TempColor := PaintBox1.Canvas.Font.Color;

PaintBox1.Canvas.Brush.Style := BsClear;

PaintBox1.Canvas.Font.Color := ClLime;

TextPos := Point((StartPoint.X + EndPoint.X - PaintBox1.Canvas.TextWidth(IntToStr(Temp^.Weight))) div 2,

(StartPoint.Y + EndPoint.Y - PaintBox1.Canvas.TextHeight(IntToStr(Temp^.Weight))) div 2);

PaintBox1.Canvas.Font.Orientation := Trunc(StartPoint.Angle(EndPoint) \* 180 / Pi);

PaintBox1.Canvas.TextOut(TextPos.X, TextPos.Y, IntToStr(Temp^.Weight));

PaintBox1.Canvas.Brush.Style := TempStyle;

PaintBox1.Canvas.Font.Color := TempColor;

Temp := Temp^.Next;

end;

end;

end;

Procedure TMainForm.InsertInHead(I, J, Weight: Integer);

Var

Temp: PEdge;

Begin

New(Temp);

Temp^.EndEdge := J;

Temp^.Weight := Weight;

Temp^.Next := Head[I];

Head[I] := Temp;

End;

Procedure TMainForm.FillHead(ArrIncidence: TArrOI);

Var

I, J, StartIndexEdge, EndIndexEdge, Weight: Integer;

Begin

For J := 0 to High(ArrIncidence[0]) do

Begin

I := 0;

StartIndexEdge := -1;

EndIndexEdge := -1;

While ArrIncidence[I, J] = 0 do

Inc(I);

If ArrIncidence[I, J] > 0 Then

StartIndexEdge := I

Else

EndIndexEdge := I;

Inc(I);

While (I < Length(ArrIncidence)) and (ArrIncidence[I, J] = 0) do

Inc(I);

If I = Length(ArrIncidence) Then

Begin

If StartIndexEdge <> -1 Then

EndIndexEdge := StartIndexEdge

Else

StartIndexEdge := EndIndexEdge;

End

Else

Begin

If StartIndexEdge <> -1 Then

EndIndexEdge := I

Else

StartIndexEdge := I;

End;

Weight := StrToInt(WeightGrid.Cells[J, 0]);

InsertInHead(StartIndexEdge, EndIndexEdge, Weight);

End;

End;

Function TMainForm.GetFloidPaths: TArrOI;

Var

Dist: TArrOI;

I, J, K: Integer;

Temp: PEdge;

Begin

SetLength(Dist, Length(Head), Length(Head));

for I := 0 to High(Dist) do

for J := 0 to High(Dist[0]) do

Dist[I,J] := Inf;

for I := 0 to High(Dist) do

begin

Temp := Head[I];

while (Temp <> nil) do

begin

J := Temp^.EndEdge;

Dist[I,J] := Min(Dist[I,J], Temp^.Weight);

Temp := Temp^.Next;

end;

end;

For I := 0 to High(Dist) do

Dist[I,I] := 0;

//Само начало

for K := 0 to High(Dist) do

for I := 0 to High(Dist) do

for J := 0 to High(Dist[0]) do

Dist[I,J] := Min(Dist[I,J], Dist[I,K] + Dist[K,J]);

GetFloidPaths := Dist;

End;

Procedure TMainForm.WriteAnswer;

Var

I, J: Integer;

Begin

AnswerGrid.RowCount := Length(Dist) + 1;

AnswerGrid.ColCount := Length(Dist[0]) + 1;

For I := 0 to High(Dist) do

For J := 0 to High(Dist[0]) do

If Dist[I, J] <> Inf Then

AnswerGrid.Cells[J + 1, I + 1] := IntToStr(Dist[I, J])

Else

AnswerGrid.Cells[J + 1, I + 1] := 'Infinity';

End;

procedure TMainForm.BitBtn1Click(Sender: TObject);

Var

ArrIncidence: TArrOI;

begin

ArrIncidence := GetIncidenceMatrix;

MakeEmptyHead;

FillHead(ArrIncidence);

Dist := GetFloidPaths;

WriteAnswer;

PaintBox1.Invalidate;

end;

procedure TMainForm.Timer1Timer(Sender: TObject);

Var

I: Integer;

begin

For I := 1 to GridIncidence.RowCount - 1 do

GridIncidence.Cells[0,I] := IntToStr(I);

For I := 1 to GridIncidence.ColCount - 1 do

GridIncidence.Cells[I,0] := Chr(I + 96);

For I := 1 to AnswerGrid.RowCount - 1 do

Begin

AnswerGrid.Cells[I, 0] := IntToStr(I);

AnswerGrid.Cells[0, I] := IntToStr(I);

End;

end;

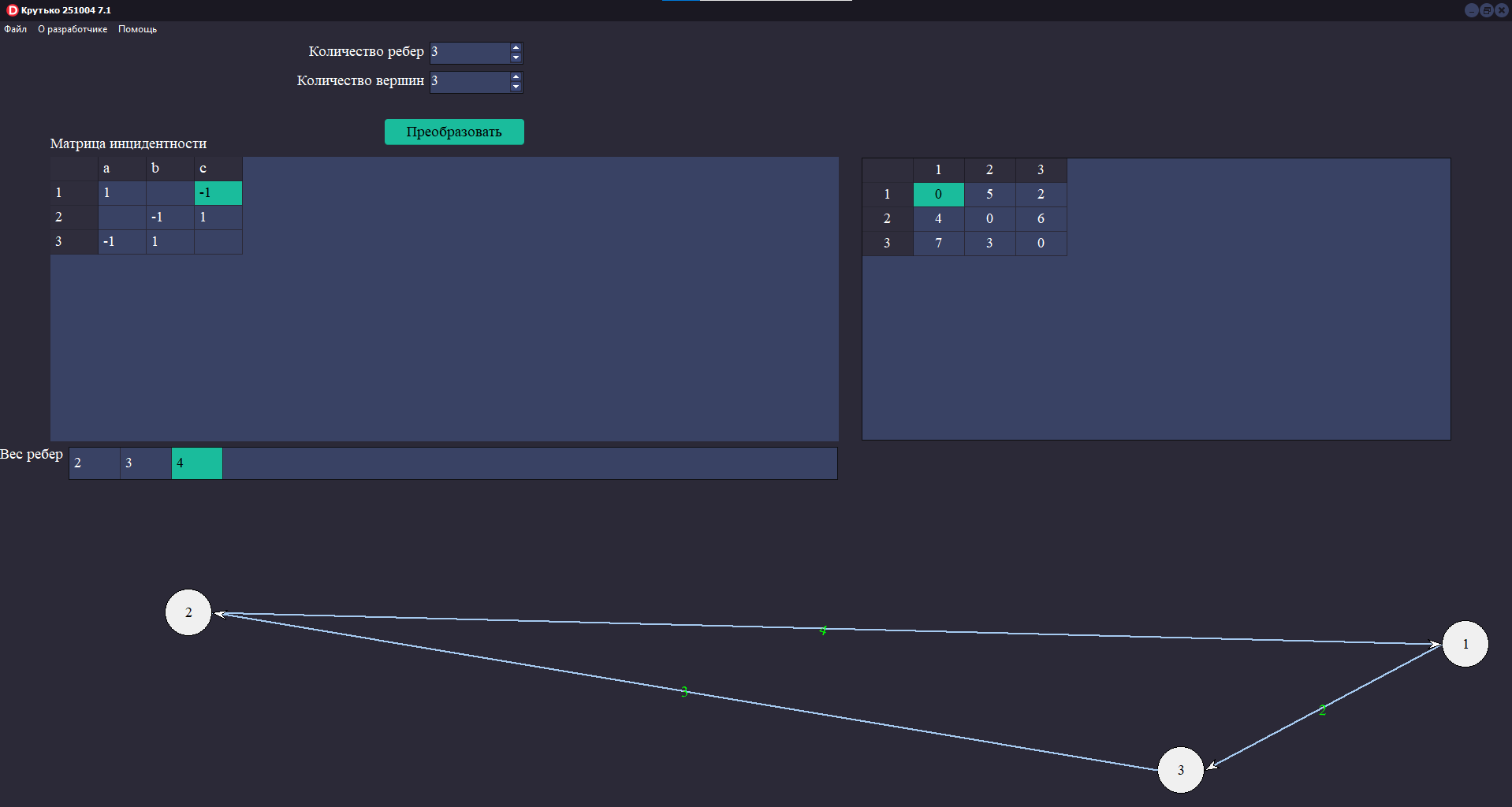
Код программы на **C#**:

// See https://aka.ms/new-console-template for more information  
  
using System.IO.IsolatedStorage;  
using System.Runtime.Versioning;  
using Lab7\_1;  
  
const int Inf = 1000;  
  
Console.WriteLine("Данная программа находит кратчайшие пути между всеми точками по методу Флойда-Уоршелла");  
int[,] dist = GetKey().Key == ConsoleKey.C ? GetAnswerFromConsole() : GetAnswerFromFile();  
  
if (GetKey().Key == ConsoleKey.C)  
 Print(Console.Write, dist);  
else  
 SaveToFile(dist);  
  
void SaveToFile(int[,] dist)  
{  
 bool isIncorrect;  
 do {  
 isIncorrect = true;  
 try  
 {  
 using (var sw = new StreamWriter(TakeFilePath(), false))  
 Print(sw.Write, dist);  
 }  
 catch (Exception e)  
 {  
 isIncorrect = false;  
 Console.WriteLine(e.Message);  
 }  
 } while (!isIncorrect);  
}  
  
int[,] GetAnswerFromConsole()  
{  
 Console.WriteLine("Введите число ребер от 1 до 15: ");  
 int numberRibs = GetNumber(1, 15);  
 Console.WriteLine("Введите число вершин от 2 до 10: ");  
 int numberEdges = GetNumber(2, 10);  
 return GetFloidPaths(FillHead(GetIncidenceMatrix(numberEdges, numberRibs, Console.ReadLine, false), GetWeightArray(numberRibs, Console.ReadLine, false)));  
}  
  
int[,] GetAnswerFromFile()  
{  
 int[,] dist = {};  
 bool isIncorrect;  
 do {  
 isIncorrect = true;  
 try  
 {  
 using (var st = new StreamReader(TakeFilePath()))  
 {  
 int numberRibs = int.Parse(st.ReadLine());  
 if (numberRibs > 15 || numberRibs < 1)  
 throw new Exception("Количество ребер некорректно");  
 int numberEdges = int.Parse(st.ReadLine());  
 if (numberEdges > 10 || numberEdges < 1)  
 throw new Exception("Количество вершин некорректно");  
 dist = GetFloidPaths(FillHead(GetIncidenceMatrix(numberEdges, numberRibs, st.ReadLine, true), GetWeightArray(numberRibs, st.ReadLine, true)));  
 }  
 }  
 catch (Exception e)  
 {  
 Console.WriteLine(e.Message);  
 isIncorrect = false;  
 }  
 } while (!isIncorrect);  
 return dist;  
}  
  
int[] GetWeightArray(int ribs, Func<string?> readLine, bool isFile)  
{  
 int[] arr = new int[ribs];  
 if (!isFile)  
 {  
 Console.WriteLine("Введите ваш вес всех ребер:");  
 Console.WriteLine("Ввод веса происходит в одну строку!");  
 }  
  
 bool isValid;  
 do  
 {  
 isValid = true;  
 var input = readLine().Split(' ');  
 if (input.Length != arr.GetLength(0))  
 isValid = GetWarningMessage();  
 if (isValid)  
 foreach (int col in Enumerable.Range(0, input.Length))  
 if (!int.TryParse(input[col], out arr[col]))  
 {  
 isValid = GetWarningMessage();  
 break;  
 }  
 if (isValid)  
 {  
 if (isFile && !isValid)  
 return null;  
 }  
 } while (!isValid);  
 return arr;  
}  
  
ConsoleKeyInfo GetKey()  
{  
 ConsoleKeyInfo key;  
 Console.WriteLine("Введите \"F\", если хотите работать с файлом;\nВведите \"C\", если хотите работать с консолью.");  
 do {  
 key = Console.ReadKey();  
 } while (key.Key != ConsoleKey.C && key.Key != ConsoleKey.F);  
 return key;  
}  
  
void Print(Action<object> action, int[,] arr)  
{  
 action("Ваши найденные пути:" + "\n");  
 for (int i = 0; i < arr.GetLength(0); i++) {  
 for (int j = 0; j < arr.GetLength(1); j++)  
 if (arr[i, j] == Inf)  
 action("Null" + "\t");  
 else   
 action(arr[i,j] + "\t");  
 action("\n");  
 }  
}  
  
string? TakeFilePath()  
{  
 string? path;  
 bool isIncorrect;  
 do {  
 isIncorrect = false;  
 Console.WriteLine("Введите путь к файлу");  
 path = Console.ReadLine();  
 if (!File.Exists(path) || !path.EndsWith(".txt"))  
 {  
 isIncorrect = true;  
 Console.WriteLine("Проверьте параметры файла");  
 }  
 } while (isIncorrect);  
 return path;  
}  
  
int[,] GetFloidPaths(AdjList[] list)  
{  
 int[,] dist = new int[list.Length, list.Length];  
 for (int i = 0; i < dist.GetLength(0); i++)  
 for (int j = 0; j < dist.GetLength(1); j++)  
 dist[i, j] = Inf;  
  
 for (int i = 0; i < dist.GetLength(0); i++)  
 foreach (var node in list[i])  
 dist[i, node.Edge] = Math.Min(dist[i, node.Edge], node.Weight);  
  
 for (int i = 0; i < dist.GetLength(0); i++)  
 dist[i, i] = 0;  
  
 for (int k = 0; k < dist.GetLength(0); k++)  
 for (int i = 0; i < dist.GetLength(0); i++)  
 for (int j = 0; j < dist.GetLength(1); j++)  
 dist[i, j] = Math.Min(dist[i, j], dist[i, k] + dist[k, j]);  
   
 return dist;  
}  
  
AdjList[] FillHead(int[,] arr, int[] weightArr)  
{  
 AdjList[] list = new AdjList[arr.GetLength(0)];  
 for (int i = 0; i < list.GetLength(0); i++)  
 list[i] = new AdjList();  
 for (int j = 0; j < arr.GetLength(1); j++)  
 {  
 var i = 0;  
 var startIndexEdge = -1;  
 var endIndexEdge = -1;  
 while (arr[i, j] == 0)  
 i++;  
 if (arr[i, j] > 0)  
 startIndexEdge = i;  
 else  
 endIndexEdge = i;  
 i++;  
 while (i < arr.GetLength(0) && arr[i, j] == 0)  
 i++;  
 if (i == arr.GetLength(0))  
 {  
 if (startIndexEdge != -1)  
 endIndexEdge = startIndexEdge;  
 else  
 startIndexEdge = endIndexEdge;  
 }  
 else  
 {  
 if (startIndexEdge != -1)  
 endIndexEdge = i;  
 else  
 startIndexEdge = i;  
 }  
 list[startIndexEdge].Add(weightArr[j], endIndexEdge);  
 }  
 return list;  
}  
  
int[,] GetIncidenceMatrix(int edges, int ribs, Func<string?> readLine, bool isFile)  
{  
 if (!isFile)  
 {  
 Console.WriteLine("Введите вашу матрицу инцидентности:");  
 Console.WriteLine("Вводите элементы построчно!");  
 }  
 int[,] incidenceMatrix = new int[ribs, edges];  
 bool isValid;  
 do  
 {  
 if (!isFile)   
 Console.WriteLine("Считывание матрицы сначала: ");  
 isValid = true;  
 foreach (int row in Enumerable.Range(0, incidenceMatrix.GetLength(0)))  
 {  
 var input = readLine().Split(' ');  
 if (input.Length != incidenceMatrix.GetLength(1))  
 {  
 isValid = GetWarningMessage();  
 break;  
 }  
 foreach (int col in Enumerable.Range(0, input.Length))  
 if (!int.TryParse(input[col], out incidenceMatrix[row, col]))  
 {  
 isValid = GetWarningMessage();  
 break;  
 }  
 }  
 if (isValid)  
 {  
 isValid = CheckIncidenceMatrix(incidenceMatrix);  
 if (isFile && !isValid)  
 return null;  
 }  
 } while (!isValid);  
 return incidenceMatrix;  
}  
  
bool GetWarningMessage()  
{  
 Console.WriteLine("Incorrect input!");  
 return false;  
}  
  
bool CheckIncidenceMatrix(int[,] arr)  
{  
 bool isCorrect = true;  
 for (int i = 0; i < arr.GetLength(1) && isCorrect; i++)  
 {  
 int one = 0, two = 0, minusOne = 0;  
 for (int j = 0; j < arr.GetLength(0); j++)  
 switch (arr[j, i])  
 {  
 case 1:  
 one++;  
 break;  
 case 2:  
 two++;  
 break;  
 case -1:  
 minusOne++;  
 break;  
 }  
  
 isCorrect = (one == 1 && minusOne == 1 && two == 0) || (two == 1 && one == 0 && minusOne == 0);  
 }  
 return isCorrect;  
}  
  
int GetNumber(in int min, in int max)  
{  
 int numb;  
 do {  
 } while (!int.TryParse(Console.ReadLine(), out numb) || numb < min || numb > max);  
 return numb;  
}

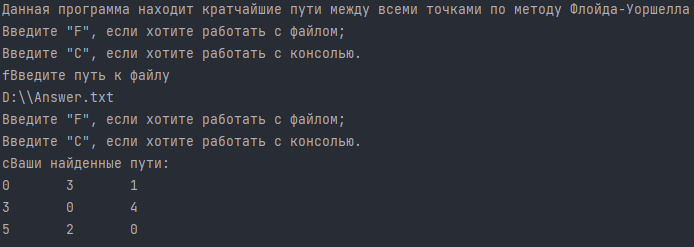
namespace Lab7\_1;  
  
public class Node  
{  
 public Node(int weight, int edge)  
 {  
 Weight = weight;  
 Edge = edge;  
 }  
 public int Weight { get; set; }  
 public int Edge { get; set; }  
 public Node? Next { get; set; }  
}

using System.Collections;  
  
namespace Lab7\_1  
{  
 public class AdjList : IEnumerable<Node>  
 {  
 Node? Head;  
 Node? Tail;  
  
 public void Add(int weight, int edge)  
 {  
 if (Head == null)  
 {  
 Head = new Node(weight, edge);  
 Tail = Head;  
 }  
 else  
 {  
 Tail.Next = new Node(weight, edge);  
 Tail = Tail.Next;  
 }  
 }  
  
 public void Clear()  
 {  
 Head = null;  
 Tail = null;  
 }  
  
 public AdjList()  
 {  
 Head = null;  
 Tail = null;  
 }  
  
 public IEnumerator<Node> GetEnumerator()  
 {  
 Node current = Head;  
  
 while (current != null)  
 {  
 yield return current;  
 current = current.Next;  
 }  
 }  
  
 IEnumerator IEnumerable.GetEnumerator()  
 {  
 return GetEnumerator();  
 }  
 }  
}

Результат на **Delphi**:



Результат на **C#**:



**Блок-cхема**:

