УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №7.1

по предмету «Основы алгоритмизации и программирования»

Вариант 11

Выполнил:

Егоров А.С.

Гр. 351005

Проверил:

Данилова Г. В.

Минск 2024

**Задание:**

Преобразовать списки инцидентности в матрицу смежности.

**Код программы Delphi:**

Unit MatrixUnit;

Interface

Uses

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

System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Grids, Vcl.StdCtrls, Vcl.ComCtrls,

LinkedList;

Type

TuVCLMatrix = Class(TForm)

LbFormInfo: TLabel;

StrGrMatrix: TStringGrid;

Procedure FormShow(Sender: TObject);

Private

FNodes: TArray<TList<Integer>>;

Procedure PaintMatrix();

Procedure PaintForm();

Procedure PaintField();

Public

Property Nodes: TArray < TList < Integer >> Write FNodes;

End;

Var

UVCLMatrix: TuVCLMatrix;

Implementation

{$R \*.dfm}

Procedure TuVCLMatrix.PaintField;

Var

Size, I: Integer;

Begin

Size := Length(FNodes);

StrGrMatrix.FixedCols := 1;

StrGrMatrix.ColCount := StrGrMatrix.FixedCols + Size;

StrGrMatrix.FixedRows := 1;

StrGrMatrix.RowCount := StrGrMatrix.FixedRows + Size;

For I := 1 To Size Do

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

For I := 1 To Size Do

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

End;

procedure TuVCLMatrix.PaintForm;

Const

OffSet : Integer = 60;

Var

Size, SideWidth: Integer;

begin

SideWidth := StrGrMatrix.Margins.Left \* 2 + (StrGrMatrix.DefaultColWidth + 2) \*

(Length(FNodes) + 1);

Self.Constraints.MinHeight := SideWidth + OffSet;

Self.Constraints.MinWidth := SideWidth;

Self.ClientHeight := SideWidth + OffSet;

Self.ClientWidth := SideWidth;

end;

Procedure TuVCLMatrix.PaintMatrix;

Var

I, J, K: Integer;

TempArray: TArray<Integer>;

Begin

For I := 0 To High(FNodes) Do

Begin

TempArray := FNodes[I].ToArray();

K := 0;

For J := 0 To High(FNodes) Do

Begin

If (K < Length(TempArray)) And (TempArray[K] = J) Then

Begin

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

Inc(K);

End

Else

StrGrMatrix.Cells[J + 1, I + 1] := '0';

End;

End;

End;

Procedure TuVCLMatrix.FormShow(Sender: TObject);

Begin

// создание окна нужного размера

PaintForm();

// отрисовка полей

PaintField();

// отрисовка матрицы

PaintMatrix();

End;

End.

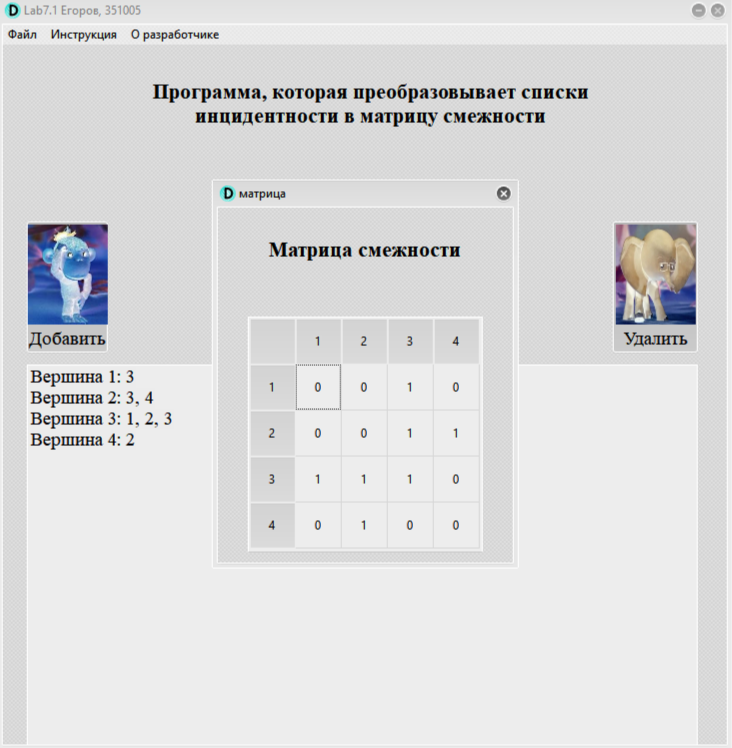
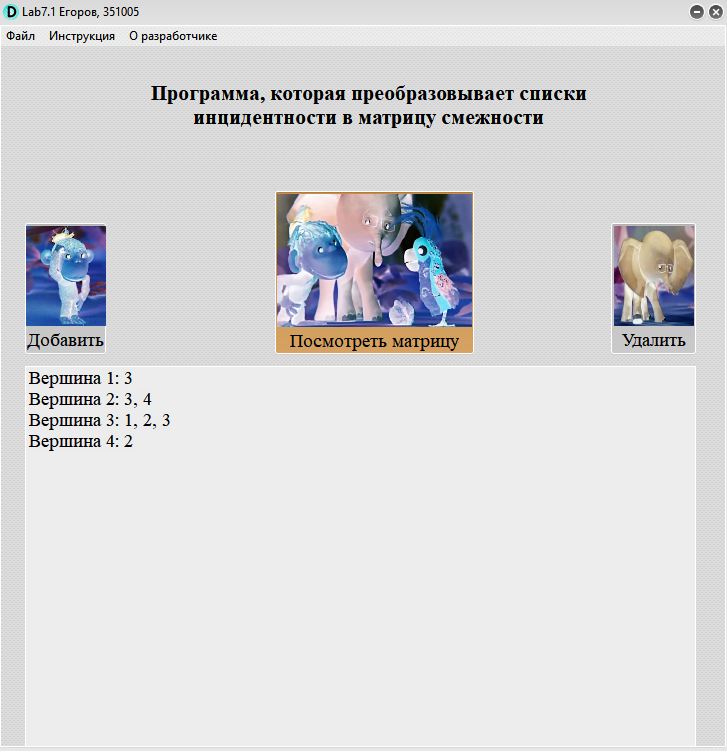
**Код программы CSharp:**

**MSquareBuilder.cs**

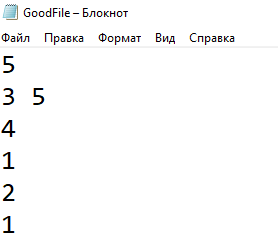
using Lab5\_2;  
  
LinkedList<LinkedList<int>> nodes = new LinkedList<LinkedList<int>>();  
MainMenu mainMenu = new MainMenu();  
int choose = 0;  
int size = 0;  
mainMenu.ShowProgramInfo();  
  
// input block  
do  
{  
 mainMenu.InputShowMenu();  
 choose = mainMenu.InputChoice();  
 // initialing reader  
 switch ((WorkingTypes)choose)  
 {  
 case WorkingTypes.WtConsole:  
 {  
 ConsoleReader consoleReader = new ConsoleReader();  
 size = consoleReader.InputSizeOfNodes();  
 nodes = consoleReader.InputElements(size);  
 }  
 break;  
 case WorkingTypes.WtFile:  
 {  
 FileReader fileReader = new FileReader();  
 do  
 {  
 fileReader.FilePath = mainMenu.InputFilePath();  
 // reading size and elements and checking status every operation  
 if (fileReader.FileStatus == FileStatus.FsGood)  
 {  
 size = fileReader.InputSizeOfNodes();  
 }  
 if (fileReader.FileStatus == FileStatus.FsGood)  
 {  
 nodes = fileReader.InputElements(size);  
 }  
 } while (fileReader.FileStatus != FileStatus.FsGood);  
 }  
 break;  
 default:  
 {  
 mainMenu.ShowWrongKeyMessage();  
 choose = 0;  
 }  
 break;  
 }  
} while (choose < (int)WorkingTypes.WtConsole || (int)WorkingTypes.WtFile < choose);  
  
// main block  
int[,] matrix = BuildMatrix(nodes,size);  
  
// output block  
do  
{  
 mainMenu.OutputShowMenu();  
 choose = mainMenu.InputChoice();  
 // initialing reader  
 switch ((WorkingTypes)choose)  
 {  
 case WorkingTypes.WtConsole:  
 {  
 ConsoleWriter consoleWriter = new ConsoleWriter();  
 consoleWriter.Output(matrix,size);  
 }  
 break;  
 case WorkingTypes.WtFile:  
 {  
 FileWriter fileWriter = new FileWriter();  
 do  
 {  
 fileWriter.FilePath = mainMenu.InputFilePath();  
 if (fileWriter.FileStatus == FileStatus.FsGood)  
 fileWriter.Output(matrix,size);  
 mainMenu.ShowFileStatusMessage(fileWriter.FileStatus);  
 } while (fileWriter.FileStatus != FileStatus.FsGood);  
 }  
 break;  
 default:  
 {  
 mainMenu.ShowWrongKeyMessage();  
 choose = 0;  
 }  
 break;  
 }  
} while (choose < 0 || (int)WorkingTypes.WtFile < choose);  
  
  
int[,] BuildMatrix(LinkedList<LinkedList<int>> linkedList, int size)  
{  
 int[,] matrix = new int[size,size];  
 int i = 0;  
 foreach (var list in linkedList)  
 {  
 int[] arr = list.ToArray();  
 int k = 0;  
 for (int j = 0; j < size; j++)  
 {  
 if (k < arr.Length && arr[k] - 1 == j)  
 {  
 matrix[i, j] = 1;  
 k++;  
 }  
 else  
 matrix[i, j] = 0;  
 }  
  
 i++;  
 }  
 return matrix;  
}

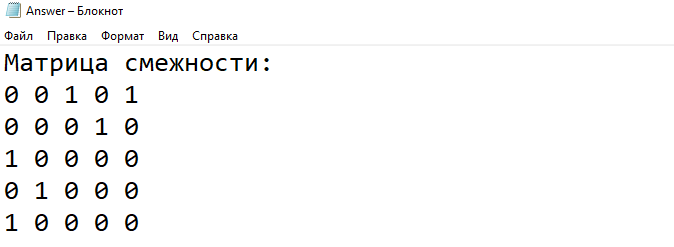
**Скриншоты:**

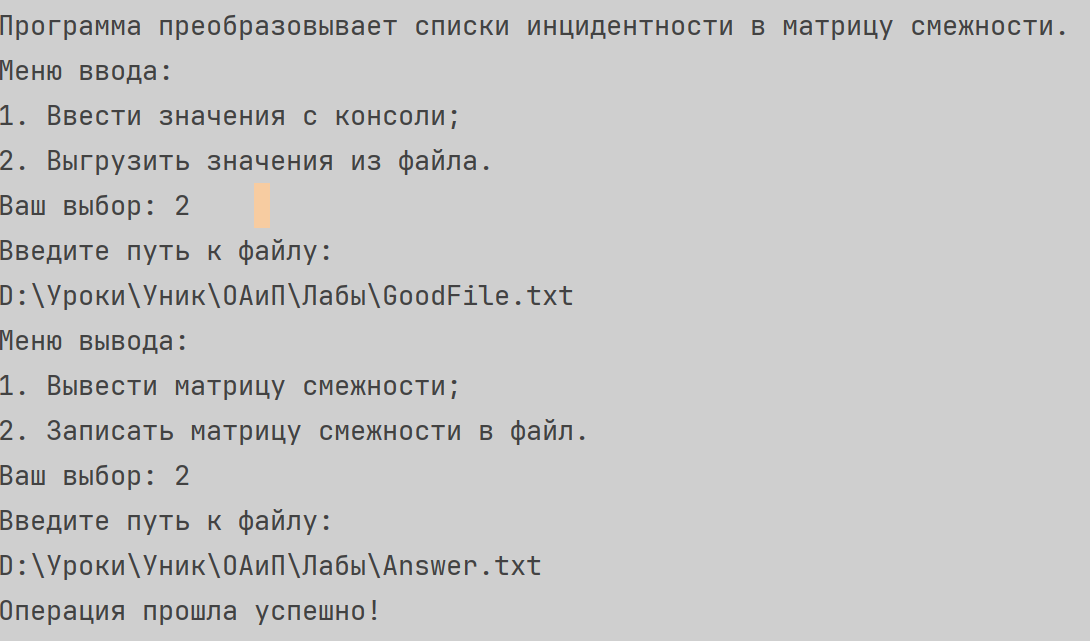
**Delphi:**

** **

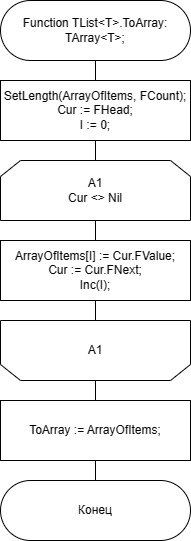
**CSharp:**

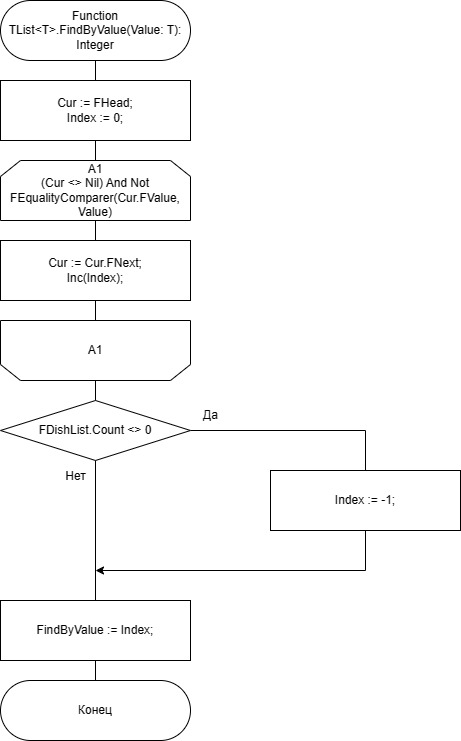
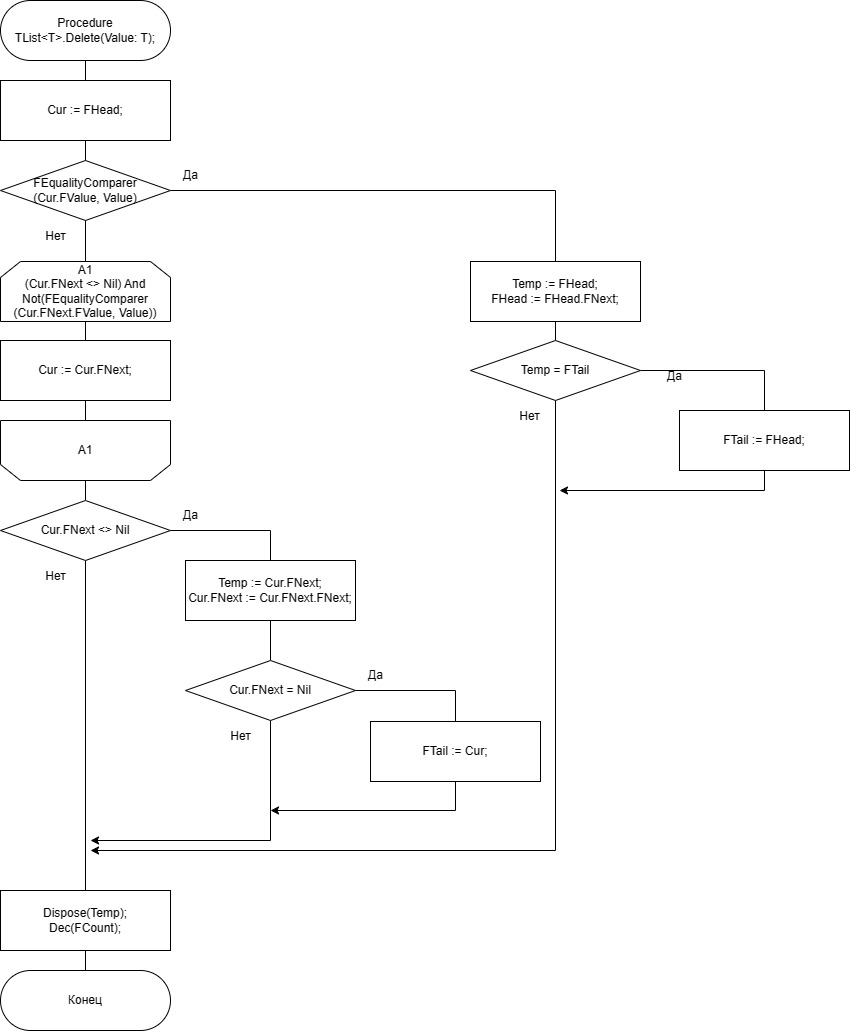
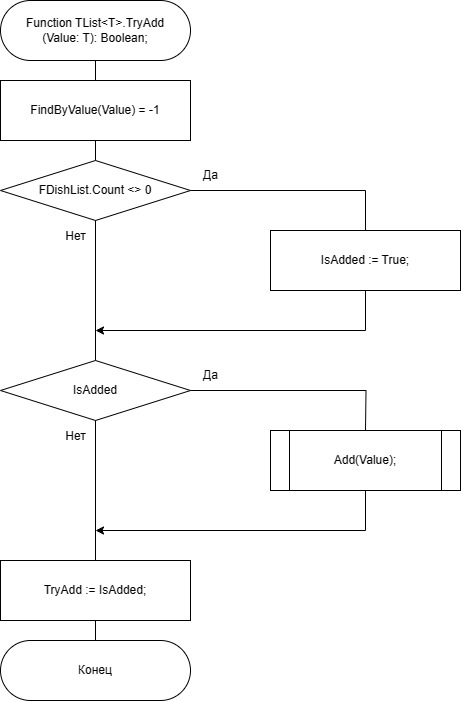
****

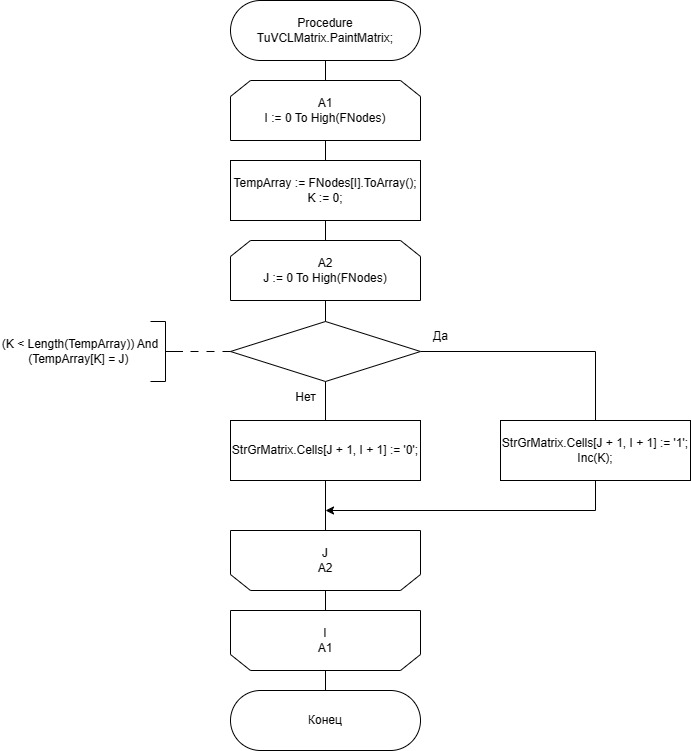


****

**Блок Схема:**

****



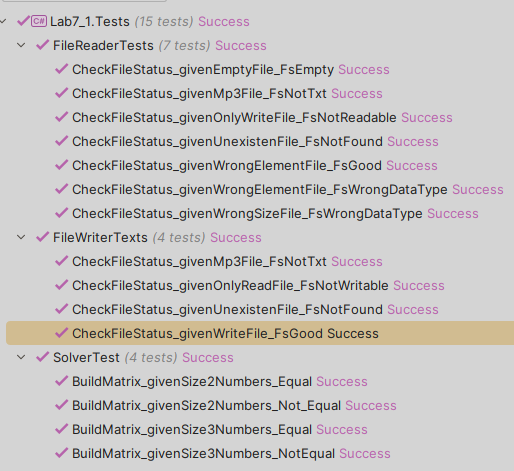
****

**Юнит Тесты**

**Код**

using System;  
using System.Collections.Generic;  
using JetBrains.Annotations;  
using Microsoft.VisualStudio.TestTools.UnitTesting;  
using Lab7\_1;  
using Microsoft.VisualStudio.TestPlatform.TestHost;  
  
  
[TestClass]  
[TestSubject(typeof(FileReader))]  
public class FileReaderTests  
{  
 private static FileReader fileReader = new FileReader();   
  
 [TestMethod]  
 public void CheckFileStatus\_givenOnlyWriteFile\_FsNotReadable()  
 {  
 fileReader.FilePath = "D:\\Уроки\\Уник\\ОАиП\\Лабы\\onlyWriteFile.txt";  
 fileReader.InputSizeOfNodes();  
 Assert.AreEqual(fileReader.FileStatus,FileStatus.FsNotReadable);  
 }  
   
 [TestMethod]  
 public void CheckFileStatus\_givenMp3File\_FsNotTxt()  
 {  
 fileReader.FilePath = "D:\\Уроки\\Уник\\ОАиП\\Лабы\\music.mp3";  
 Assert.AreEqual(fileReader.FileStatus,FileStatus.FsNotTxt);  
 }  
   
 [TestMethod]  
 public void CheckFileStatus\_givenUnexistenFile\_FsNotFound()  
 {  
 fileReader.FilePath = "D:\\Уроки\\Уник\\ОАиП\\Лабы\\NotExist.txt";  
 Assert.AreEqual(fileReader.FileStatus,FileStatus.FsNotFound);  
 }  
   
 [TestMethod]  
 public void CheckFileStatus\_givenWrongSizeFile\_FsWrongDataType()  
 {  
 fileReader.FilePath = "D:\\Уроки\\Уник\\ОАиП\\Лабы\\WrongSizeFile.txt";  
 fileReader.InputSizeOfNodes();  
 Assert.AreEqual(fileReader.FileStatus,FileStatus.FsWrongDataType);  
 }  
   
 [TestMethod]  
 public void CheckFileStatus\_givenWrongElementFile\_FsWrongDataType()  
 {  
 fileReader.FilePath = "D:\\Уроки\\Уник\\ОАиП\\Лабы\\WrongTypeFile.txt";  
 fileReader.InputElements(fileReader.InputSizeOfNodes());  
 Assert.AreEqual(fileReader.FileStatus,FileStatus.FsWrongDataType);  
 }  
   
 [TestMethod]  
 public void CheckFileStatus\_givenEmptyFile\_FsEmpty()  
 {  
 fileReader.FilePath = "D:\\Уроки\\Уник\\ОАиП\\Лабы\\EmptyFile.txt";  
 fileReader.InputSizeOfNodes();  
 Assert.AreEqual(fileReader.FileStatus,FileStatus.FsEmpty);  
 }  
   
 [TestMethod]  
 public void CheckFileStatus\_givenWrongElementFile\_FsGood()  
 {  
 fileReader.FilePath = "D:\\Уроки\\Уник\\ОАиП\\Лабы\\GoodFile.txt";  
 fileReader.InputElements(fileReader.InputSizeOfNodes());  
 Assert.AreEqual(fileReader.FileStatus,FileStatus.FsGood);  
 }  
}  
  
[TestClass]  
[TestSubject(typeof(FileWriter))]  
public class FileWriterTexts  
{  
 private static FileWriter fileWriter = new FileWriter();  
   
 [TestMethod]  
 public void CheckFileStatus\_givenOnlyReadFile\_FsNotWritable()  
 {  
 fileWriter.FilePath = "D:\\Уроки\\Уник\\ОАиП\\Лабы\\onlyReadFile.txt";  
 const int SIZE = 3;  
 int[,] testMatrix = { {0,1,0},{1,0,0},{0,0,0} };  
 fileWriter.Output(testMatrix,SIZE);  
 Assert.AreEqual( fileWriter.FileStatus,FileStatus.FsNotWritable);  
 }   
   
 [TestMethod]  
 public void CheckFileStatus\_givenMp3File\_FsNotTxt()  
 {  
 fileWriter.FilePath = "D:\\Уроки\\Уник\\ОАиП\\Лабы\\music.mp3";  
 Assert.AreEqual(fileWriter.FileStatus,FileStatus.FsNotTxt);  
 }  
   
 [TestMethod]  
 public void CheckFileStatus\_givenUnexistenFile\_FsNotFound()  
 {  
 fileWriter.FilePath = "D:\\Уроки\\Уник\\ОАиП\\Лабы\\NotExist.txt";  
 Assert.AreEqual(fileWriter.FileStatus,FileStatus.FsNotFound);  
 }  
   
 [TestMethod]  
 public void CheckFileStatus\_givenWriteFile\_FsGood()  
 {  
 fileWriter.FilePath = "D:\\Уроки\\Уник\\ОАиП\\Лабы\\OnlyWriteFile.txt";  
 Assert.AreEqual(fileWriter.FileStatus,FileStatus.FsGood);  
 }  
}  
  
[TestClass]  
[TestSubject(typeof(Solver))]  
public class SolverTest  
{  
  
 [TestMethod]  
 public void BuildMatrix\_givenSize2Numbers\_Equal()  
 {  
 LinkedList<LinkedList<int>> allNodes = new LinkedList<LinkedList<int>>();  
 int SIZE = 2;  
 LinkedList<int> tempList1 = new LinkedList<int>();  
 tempList1.AddLast(2);  
 allNodes.AddLast(tempList1);  
 LinkedList<int> tempList2 = new LinkedList<int>();  
 tempList2.AddLast(1);  
 allNodes.AddLast(tempList2);  
 int[,] answer = { { 0, 1 }, { 1, 0 } };  
 CollectionAssert.AreEquivalent(Solver.BuildMatrix(allNodes, SIZE),answer);  
 }  
 [TestMethod]   
 public void BuildMatrix\_givenSize2Numbers\_Not\_Equal()  
 {  
 LinkedList<LinkedList<int>> allNodes = new LinkedList<LinkedList<int>>();  
 int SIZE = 2;  
 LinkedList<int> tempList1 = new LinkedList<int>();  
 tempList1.AddLast(1);  
 tempList1.AddLast(2);  
 allNodes.AddLast(tempList1);  
 LinkedList<int> tempList2 = new LinkedList<int>();  
 tempList2.AddLast(1);  
 tempList2.AddLast(2);  
 allNodes.AddLast(tempList2);  
 int[,] answer = { { 0, 1 }, { 1, 0 } };  
 CollectionAssert.AreNotEquivalent(Solver.BuildMatrix(allNodes, SIZE),answer);  
 }  
   
 [TestMethod]  
 public void BuildMatrix\_givenSize3Numbers\_Equal()  
 {  
 LinkedList<LinkedList<int>> allNodes = new LinkedList<LinkedList<int>>();  
 int SIZE = 3;  
 LinkedList<int> tempList1 = new LinkedList<int>();  
 tempList1.AddLast(1);  
 tempList1.AddLast(3);  
 allNodes.AddLast(tempList1);  
 LinkedList<int> tempList2 = new LinkedList<int>();  
 tempList2.AddLast(1);  
 allNodes.AddLast(tempList2);  
 int[,] answer = { { 1, 0, 1 }, { 0, 0, 0 }, { 1, 0, 0 } };  
 CollectionAssert.AreEquivalent(Solver.BuildMatrix(allNodes, SIZE),answer);  
 }  
   
 [TestMethod]  
 public void BuildMatrix\_givenSize3Numbers\_NotEqual()  
 {  
 LinkedList<LinkedList<int>> allNodes = new LinkedList<LinkedList<int>>();  
 int SIZE = 3;  
 LinkedList<int> tempList1 = new LinkedList<int>();  
 tempList1.AddLast(1);  
 allNodes.AddLast(tempList1);  
 LinkedList<int> tempList2 = new LinkedList<int>();  
 tempList2.AddLast(1);  
 allNodes.AddLast(tempList2);  
 LinkedList<int> tempList3 = new LinkedList<int>();  
 tempList3.AddLast(1);  
 allNodes.AddLast(tempList3);  
 int[,] answer = { { 1, 0, 1 }, { 0, 0, 0 }, { 0, 0, 1 } };  
 CollectionAssert.AreEquivalent(Solver.BuildMatrix(allNodes, SIZE),answer);  
 }  
}

**Скриншоты:**

****