Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Кубанский государственный технологический университет»

Кафедра информационных систем и программирования

**ТЕСТИРОВАНИЕ И ОТЛАДКА  
ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

Отчет по лабораторной работе №7

«Тестирование с использованием тестовых двойников»

Выполнил:

студент 3 курса

группы 19-КБ-ПР1

Кравцов Олег Юрьевич

Проверил:

ст. преп. каф. ИСП

А. Г. Волик

Краснодар

2021

**1 Цель работы**

Изучить подход к автоматизации процесса тестирования с использованием тестовых двойников (Test Doubles).

1. **Задание**

1) Создать класс, реализующий граф/дерево, хранящий информацию о   
структуре графа на основе некоторого внутреннего представления   
указанного в задании из пункта 2.1.

2) В созданном классе реализовать один из методов обработки   
графа/дерева в соответствии с вариантом задания из пункта 2.2.

3) В качестве источника данных использовать класс, реализующий   
чтение из файла произвольного формата и возвращающего данные в   
виде, указанном в варианте задания из пункта 2.1.

4) Реализовать возможность сохранения данных в файл произвольного   
формата, с помощью отдельного класса, в формате, определяемом в   
соответствии с вариантом задания в пункте 2.1.

5) Для обработки файла создать отдельный класс/классы, реализующий   
методы загрузки из файла (возвращающий считанные данных в   
определенном формате) и сохранения в файл (сохраняющий матрицу   
в определенном формате, который может отличаться от внутреннего   
представления графа).

6) Протестировать класс для работы с заданным форматом файлов (при   
тестировании класса использовать подмену класса на StringReader   
/StringWriter для доступа к строковому потоку с последующей   
проверкой поученной выходной строки).

7) Протестировать класс, реализующий непосредственные вычисления   
(при получении данных и сохранения использовать заглушки и   
фиктивные объекты).

8) Оформить отчёт.

* 1. **Задание**

1) Обход графа в ширину

2) Обход графа в глубину

3) Нахождение минимального пути в графе

4) Проверка на наличие циклов в графе

5) Проверка графа на связность

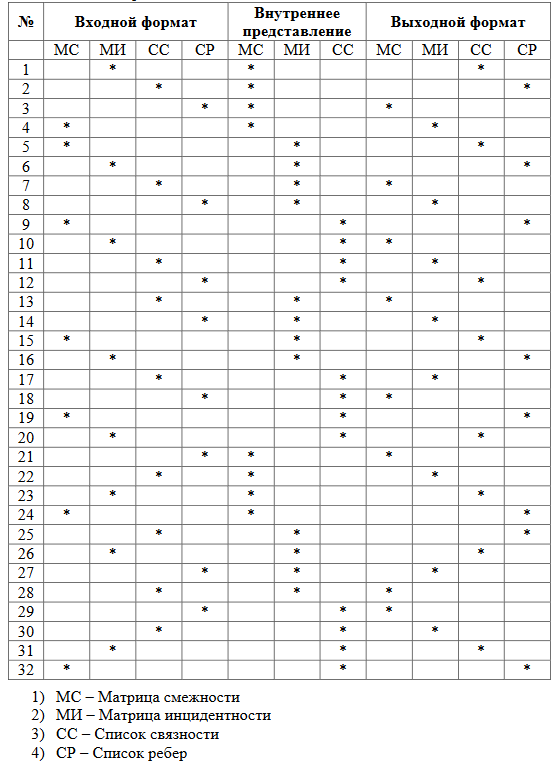
6) Поиск Эйлерова цикла в графе

7) Прямой обход дерева

8) Обратный обход дерева

9) Симметричный обход дерева

* 1. **Задание**



**3 Вариант задания**

Вариант №11.

**4 Текст программы**

Файл Program.cs

using System;

using System.IO;

namespace STLab07.Variant11

{

class Program

{

static void Main(string[] args)

{

string pathinput = @"C:\Users\User\Desktop\LR7TestsOleg\Input.odt";

StreamReader streamreader = new StreamReader(pathinput);

string pathoutput = @"C:\Users\User\Desktop\LR7TestsOleg\Output.txt";

StreamWriter streamwriter = new StreamWriter(pathoutput);

TxtTreeFile reader = new TxtTreeFile();

TxtTreeFile writer = new TxtTreeFile();

BinTree Tree = new BinTree(reader, writer);

Tree.LoadFromFile(streamreader);

Tree.SaveToFile(streamwriter);

}

}

}

Файл BinTree.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Text;

namespace STLab07.Variant11

{

public class BinTree

{

public ITreeReader Reader;

public ITreeWriter Writer;

public List<Connection> connections;

public List<int> TraversePreordeList;

public List<Unit> IncidenceMatrixList;

public int[,] IncidenceMatrix; //Матрица инцедентности

public BinTree(ITreeReader reader, ITreeWriter writer)

{

this.Reader = reader;

this.Writer = writer;

connections = new List<Connection>();

}

public void LoadFromFile(StreamReader reader)

{

//Получения списка связности

connections = Reader.LoadTree(reader);

}

public void SaveToFile(StreamWriter writer)

{

ConvertIncidenceMatrixFromListToArray();

Writer.SaveTree(writer, IncidenceMatrix);

}

public void ConvertConnectionListToIncidenceMatrixList()

{

foreach (Connection c in connections)

{

if (c.Root == true)

{

IncidenceMatrixList = new List<Unit>();

TraversePreordeList = new List<int>();

ConvertToIncidenceMatrixList(c.Index);

break;

}

}

}

public void ConvertToIncidenceMatrixList(int index)

{

if (index != -1)

{

var ID = SearchOfIndexInList(index, connections);

TraversePreordeList.Add(connections[ID].Index);

if (connections[ID].LeftConn\_Index != -1)

{

IncidenceMatrixList.Add(new Unit(index, connections[ID].LeftConn\_Index));

}

ConvertToIncidenceMatrixList(connections[ID].LeftConn\_Index);

if (connections[ID].RightConn\_Index != -1)

{

IncidenceMatrixList.Add(new Unit(index, connections[ID].RightConn\_Index));

}

ConvertToIncidenceMatrixList(connections[ID].RightConn\_Index);

}

}

public int SearchOfIndexInList(int index, List<Connection> connections)

{

for (int i = 0; i <= connections.Count; i++)

{

if (connections[i].Index == index) return i;

}

return -1;

}

public void ConvertIncidenceMatrixFromListToArray()

{

ConvertConnectionListToIncidenceMatrixList();

int[,] array = new int[connections.Count, connections.Count];

for (int i = 0; i < IncidenceMatrixList.Count; i++)

{

array[IncidenceMatrixList[i].First - 1, i] = 1;

array[IncidenceMatrixList[i].Second - 1, i] = 1;

}

foreach (Unit u in IncidenceMatrixList)

{

Console.WriteLine("First - {0,2} , Second - {1,2}", u.First, u.Second);

}//\*/

IncidenceMatrix = array;

}

}

public struct Unit

{

public int First;

public int Second;

public Unit(int first, int second)

{

First = first;

Second = second;

}

}

}

Файл Connection.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace STLab07.Variant11

{

public class Connection

{

public int Value;

public int Index;

public int LeftConn\_Index;

public int RightConn\_Index;

public bool Root;

public Connection(int value, int index, int leftconn\_index, int rightconn\_index, bool root = false)

{

Value = value;

Index = index;

LeftConn\_Index = leftconn\_index;

RightConn\_Index = rightconn\_index;

Root = root;

}

}

}

Файл ITreeReader.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Text;

namespace STLab07.Variant11

{

public interface ITreeReader

{

List<Connection> LoadTree(StreamReader reader);

}

}

Файл ITreeWriter.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Text;

namespace STLab07.Variant11

{

public interface ITreeWriter

{

void SaveTree(StreamWriter writer, int[,] adjacencyMatrix);

}

}

Файл OdtTreeFile.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Text;

namespace STLab07.Variant11

{

class OdtTreeFile : ITreeReader, ITreeWriter

{

public List<Connection> LoadTree(StreamReader reader)

{

List<Connection> res = new List<Connection>();

string[] data = reader.ReadToEnd().Split(new char[] { ';' }, StringSplitOptions.RemoveEmptyEntries);

foreach (string d in data)

{

var c = d.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

res.Add(new Connection(Convert.ToInt32(c[0]), Convert.ToInt32(c[1]),

Convert.ToInt32(c[2]), Convert.ToInt32(c[3]), Convert.ToBoolean(c[4])));

}

reader.Close();

return res;

}

public void SaveTree(StreamWriter writer, int[,] adjacencyMatrix)

{

string text = "";

for (int i = 0; i < Math.Pow(adjacencyMatrix.Length, 0.5); i++)

{

string res = "";

for (int j = 0; j < Math.Pow(adjacencyMatrix.Length, 0.5); j++)

{

res += String.Format(" {0,2} ", adjacencyMatrix[i, j]);

}

text += res + "\n";

}

writer.WriteLine(text);

writer.Close();

}

}

}

Файл ReaderStub.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Text;

namespace STLab07.Variant11

{

public class ReaderStub : ITreeReader

{

public List<Connection> LoadTree(StreamReader reader)

{

reader.Close();

return new List<Connection>()

{

new Connection(1,1,-1,-1,false),

new Connection(2,2,1,3,false),

new Connection(3,3,-1,-1,false),

new Connection(4,4,2,5,true),

new Connection(5,5,-1,7,false),

new Connection(6,6,-1,-1,false),

new Connection(7,7,6,8,false),

new Connection(8,8,-1,-1,false)

};

}

}

}

Файл TxtTreeFile.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Text;

namespace STLab07.Variant11

{

class TxtTreeFile : ITreeReader, ITreeWriter

{

public List<Connection> LoadTree(StreamReader reader)

{

List<Connection> res = new List<Connection>();

string[] data = reader.ReadToEnd().Split(new char[] { ';' }, StringSplitOptions.RemoveEmptyEntries);

foreach (string d in data)

{

var c = d.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

res.Add(new Connection(Convert.ToInt32(c[0]), Convert.ToInt32(c[1]),

Convert.ToInt32(c[2]), Convert.ToInt32(c[3]), Convert.ToBoolean(c[4])));

}

reader.Close();

return res;

}

public void SaveTree(StreamWriter writer, int[,] adjacencyMatrix)

{

string text = "";

for (int i = 0; i < Math.Pow(adjacencyMatrix.Length, 0.5); i++)

{

string res = "";

for (int j = 0; j < Math.Pow(adjacencyMatrix.Length, 0.5); j++)

{

res += String.Format(" {0,2} ", adjacencyMatrix[i, j]);

}

text += res + "\n";

}

writer.WriteLine(text);

writer.Close();

}

}

}

Файл WriterStub.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Text;

namespace STLab07.Variant11

{

public class WriterStub : ITreeWriter

{

public int SaveWasInvoked;

public void SaveTree(StreamWriter writer, int[,] adjacencyMatrix)

{

SaveWasInvoked++;

writer.Close();

}

}

}

**5 Текст модульных тестов**

Файл TreeTests.cs

using NUnit.Framework;

using System.IO;

using System.Collections.Generic;

using STLab07.Variant11;

namespace TreeTests

{

public class Tests

{

[Test]

public void TreeReadTest()

{

ReaderStub reader = new ReaderStub();

WriterStub writer = new WriterStub();

BinTree tree = new BinTree(reader, writer);

tree.LoadFromFile(new StreamReader(@"Example.txt"));

tree.SaveToFile(new StreamWriter(@"Example.txt"));

Assert.AreEqual(8, tree.connections.Count);

}

[Test]

public void TreeWriteTest()

{

ReaderStub reader = new ReaderStub();

WriterStub writer = new WriterStub();

BinTree tree = new BinTree(reader, writer);

tree.LoadFromFile(new StreamReader(@"Example.txt"));

tree.SaveToFile(new StreamWriter(@"Example.txt"));

tree.SaveToFile(new StreamWriter(@"Example.txt"));

tree.SaveToFile(new StreamWriter(@"Example.txt"));

Assert.AreEqual(3, writer.SaveWasInvoked);

}

[Test]

public void IncidenceMatrixMethodsTest()

{

ReaderStub reader = new ReaderStub();

WriterStub writer = new WriterStub();

BinTree tree = new BinTree(reader, writer);

tree.LoadFromFile(new StreamReader(@"Example.txt"));

tree.SaveToFile(new StreamWriter(@"Example.txt"));

Assert.IsNotNull(tree.IncidenceMatrix);

Assert.IsNotNull(tree.IncidenceMatrixList);

}

[Test]

public void TraversePreordeTest()

{

ReaderStub reader = new ReaderStub();

WriterStub writer = new WriterStub();

BinTree tree = new BinTree(reader, writer);

tree.LoadFromFile(new StreamReader(@"Example.txt"));

tree.SaveToFile(new StreamWriter(@"Example.txt"));

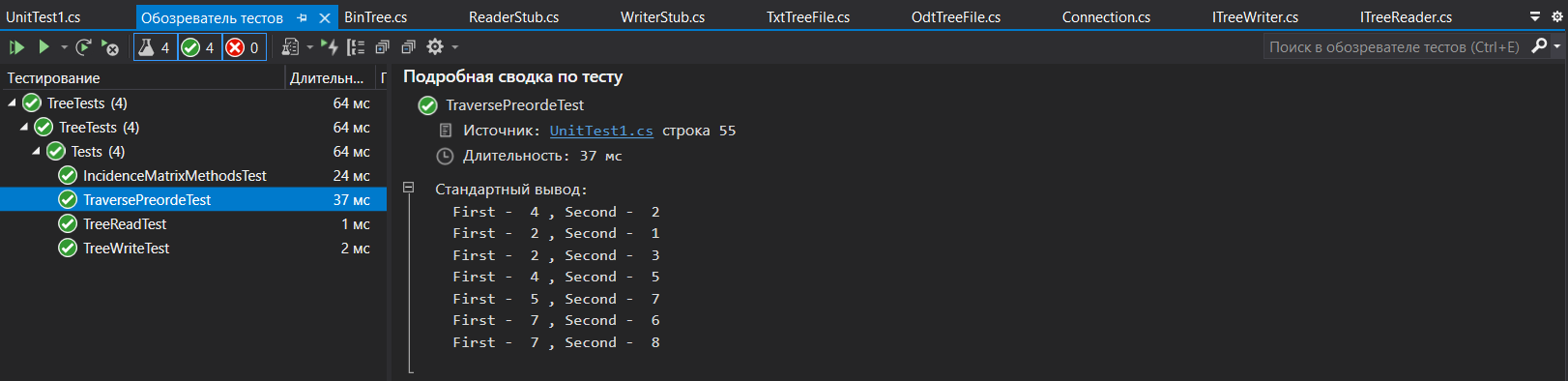
Assert.AreEqual(new List<int>() { 4, 2, 1, 3, 5, 7, 6, 8 }, tree.TraversePreordeList);

}

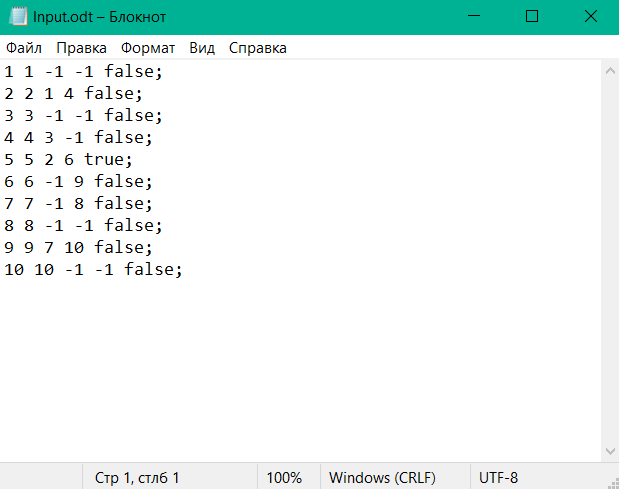
}

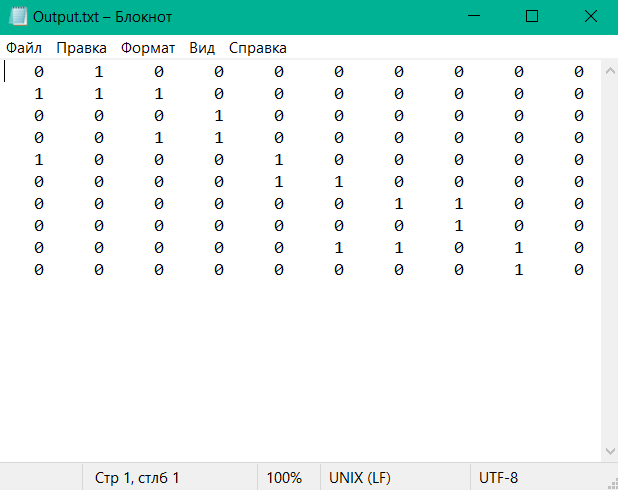
}

**6 Результаты тестирования программы**

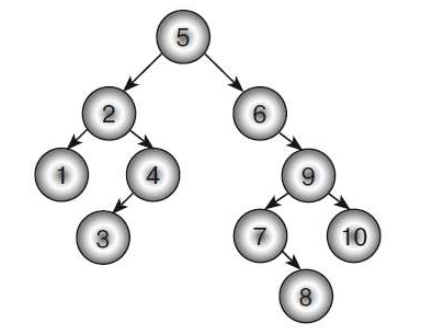


**7 Результат работы программы**





**Дерево из Program.cs**



**Дерево из модульных тестов**

