МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«Севастопольский государственный университет»

Институт информационных технологий и управления в технических системах

кафедра Информационные системы

09.03.02 Информационные системы и технологии (уровень бакалавриата)

Лабораторная работа №1

по дисциплине: «Тестирование программного обеспечения»

на тему: «Исследование способов анализа областей эквивалентности и

построения тестовых последовательностей»

Вариант – 2

Выполнил

студент 3 курса группы ИС/б-33-о

Генералов Николай Николаевич

Отметка о зачете\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

(дата)

Проверил

ст. пр. \_\_\_\_\_\_\_\_\_\_\_ Строганов В. А.

(должность) (подпись) (фамилия, инициалы)

г. Севастополь

2018 г.

# ЦЕЛЬ РАБОТЫ

Исследовать способы анализа областей эквивалентности входных данных для тестирования программного обеспечения. Приобрести практические навыки составления построения тестовых последовательностей.

# ПОСТАНОВКА ЗАДАЧИ

Задача 1**.** Дана целочисленная прямоугольная матрица. Определить

количество столбцов, не содержащих ни одного нулевого элемента.

Задача 2**.** Дана строка. Подсчитать, сколько раз среди данных символов

встречается символ + и сколько раз символ \* .

Задача 3. Программа, которая считывает текст из файла и выводит на экран только предложения, содержащие введенное с клавиатуры слово.

# ТЕКСТ ПРОГРАММНЫХ МОДУЛЕЙ

1. Текст программных модулей задачи №1.

“Program.cs”

using System;

using System.Collections.Generic;

namespace TPO\_lab1\_1{

static class Program{

static void Main(string[] args){

Console.WriteLine("Test ");

MatrixOpener dataFile = new MatrixOpener("test"1.txt");

List<List<int>> tempMatrix = dataFile.Load();

MyMatrix mainMatrix = new MyMatrix(tempMatrix);

Console.WriteLine("Matrix: ");

mainMatrix.Show();

if (!mainMatrix.IsEmpty()){

Console.WriteLine("Matrix has{0} not contains zero cols", mainMatrix.AmountOfNotContainZeroCols());

}

Console.WriteLine();

} }}

“MatrixOpener.cs”

using System;

using System.Collections.Generic;

using System.IO;

namespace TPO\_lab1\_1{

public class MatrixOpener{

public string FileName{ get; private set; }

protected StreamReader \_fread{ get; private set; }

public MatrixOpener(string \_filename){

FileName = \_filename;

try{

\_fread = new StreamReader(new FileStream(FileName, FileMode.Open));

}

catch (FileNotFoundException err){

Console.WriteLine(err);

Environment.Exit(0);

} }

public List<List<int>> Load(){

try{

List<List<int>> matrix = new List<List<int>>();

int matrixRowIndex = 0;

while (!\_fread.EndOfStream){

string tempString = \_fread.ReadLine();

matrix.Add(new List<int>());

string[] NumericStrings = tempString.Split(" ");

foreach(var num in NumericStrings){

int resultNum = 0;

if (Int32.TryParse(num,out resultNum)){

matrix[matrixRowIndex].Add(resultNum);

}

}

++matrixRowIndex;

}

return matrix;

} catch(IOException err){

Console.WriteLine(err);

return null;

}

}

~MatrixOpener(){

if (\_fread != null){

\_fread.Close();

}

}

}}

“MyMatrix.cs”

using System;

using System.Collections.Generic;

namespace TPO\_lab1\_1{

public class MyMatrix{

public int RowsAmount{get; private set;}

public int ColsAmount{get; private set;}

private List<List<int>> matrix = null;

public MyMatrix(List<List<int>> \_matrix){

RowsAmount = \_matrix.Count;

if (RowsAmount > 0){

ColsAmount = \_matrix[0].Count;

} else{

ColsAmount = 0;

}

matrix = \_matrix;

}

public void Show(){

try{

if (this.IsEmpty()){

Console.WriteLine("Matrix is empty");

return;

}

foreach (var i in matrix){

foreach (var j in i){

Console.Write("{0, 2}" + " ", j);

}

Console.WriteLine();

}

} catch(Exception err){

Console.WriteLine(err);

}

}

public bool IsEmpty(){

return (RowsAmount == 0 || ColsAmount == 0) ? true : false;

}

public int AmountOfNotContainZeroCols(){

try{

int result = 0;

if (!this.IsEmpty()){

for (int i = 0; i < this.ColsAmount; ++i){

bool ColContainZero = false;

for (int j = 0; j < this.RowsAmount; ++j){

if (matrix[j][i] == 0){

ColContainZero = true;

break;

}

}

if (!ColContainZero){

++result;

}

}

}

return result;

}

catch (Exception err){

Console.WriteLine(err);

return 0;

} } }}

1. Текст программных модулей задачи №2.

“Program.cs”

using System;

namespace TPO\_lab1\_2{

class Program{

static void Main(string[] args){

Console.WriteLine("Test ");

OneStringOpener dataFile = new OneStringOpener("test1.txt");

String tempStr = dataFile.Load();

MyString mainString = new MyString(tempStr);

mainString.ShowCharsAmount('+','\*');

Console.WriteLine();

} }}

“OneStringOpener.cs”

using System;

using System.IO;

namespace TPO\_lab1\_2{

class OneStringOpener{

public string FileName{ get; private set; }

protected StreamReader \_fread{ get; private set; }

public OneStringOpener(string \_filename){

FileName = \_filename;

try{

\_fread = new StreamReader(new FileStream(FileName, FileMode.Open));

}

catch (FileNotFoundException err){

Console.WriteLine(err);

Environment.Exit(0);

}

}

public string Load(){

try{

String tempString = "";

if (!\_fread.EndOfStream){

tempString = \_fread.ReadLine();

}

return tempString;

}

catch (IOException err){

Console.WriteLine(err);

return "";

} }

~OneStringOpener(){

if (\_fread != null){

\_fread.Close();

}

} }}

“MyString.cs”

using System;

namespace TPO\_lab1\_2{

class MyString{

public String ContentString{ get; private set; }

public MyString(String str){

ContentString = str;

}

private bool IsEmpty(){

return ContentString.Length == 0 ? true : false;

}

public int ToCountChar(char ch){

int resultCounter = 0;

foreach(var n in ContentString)

{

if (n == ch){

++resultCounter;

}

}

return resultCounter;

}

public void ShowCharsAmount(params char[] \_chArr){

if (!this.IsEmpty()){

Console.WriteLine("String: \n" + ContentString);

Console.WriteLine("{0, 10} |{1, 7}", "Character", "Amount");

foreach (var ch in \_chArr){

Console.WriteLine("{0, 10} |{1,7}", ch, this.ToCountChar(ch));

}

} else{

Console.WriteLine("String is empty");

}

} }}

1. Текст программных модулей задачи №3.

“Program.cs”

using System;

namespace TPO\_lab1\_3{

class Program{

static void Main(string[] args){

Console.WriteLine("Enter file name: ");

WordHandler fileName = new WordHandler();

fileName.ReadWordFromConsole();

TextOpener dataFile = new TextOpener(fileName.word);

String tempStr = dataFile.Load();

MyString mainString = new MyString(tempStr);

WordHandler WordToSearch = new WordHandler();

Console.WriteLine("Enter word to found: ");

WordToSearch.ReadWordFromConsole(); } }}

“MyString.cs”

using System;

namespace TPO\_lab1\_3{

class MyString{

public String ContentString{ get; private set; }

public MyString(String str){

ContentString = str;

}

private bool IsEmpty(){

return ContentString.Length == 0 ? true : false;

}

public void ShowSentence(string word){

char[] endSentenceChars ={'.','!','?'};

String[] sentencesArr = ContentString.Split(endSentenceChars, StringSplitOptions.RemoveEmptyEntries);

foreach(var str in sentencesArr){

if (str.Contains(word,StringComparison.CurrentCultureIgnoreCase)){

Console.WriteLine(str);

Console.WriteLine();

}

} } }}

“TextOpener.cs”

using System;

using System.IO;

namespace TPO\_lab1\_3{

class TextOpener{

public string FileName{ get; private set; }

protected StreamReader \_fread{ get; private set; }

public TextOpener(string \_filename){

FileName = \_filename;

try{

\_fread = new StreamReader(new FileStream(FileName, FileMode.Open));

}

catch (FileNotFoundException err){

Console.WriteLine(err);

Environment.Exit(0);

}

}

public string Load(){

try{

String tempString = "";

while (!\_fread.EndOfStream)

{

tempString += \_fread.ReadLine();

}

return tempString;

}

catch (IOException err){

Console.WriteLine(err);

return "";

}

}

~TextOpener(){

if (\_fread != null){

\_fread.Close();

}

} }}

“WordHandler.cs”

using System;

namespace TPO\_lab1\_3{

class WordHandler{

public String word{ get; set; }

public WordHandler(){

word = "";

}

public string ReadWordFromConsole(){

try{

string \_word = "";

do{

Console.WriteLine("Enter word to found: ");

\_word = Console.ReadLine();

\_word = \_word.Trim();

} while (\_word.Length < 2);

word = \_word;

return \_word;

}

catch(Exception err){

Console.WriteLine(err);

return "";

}

} }}

* 1. ОПИСАНИЕ ОБЛАСТЕЙ ЭКВИВАЛЕНТНОСТИ ЗАДАНИЯ

Для задачи №1 можно выделить такие области эквивалентности.

* По размеру матрицы:
  1. матрица строка;
  2. матрица столбец;
  3. матрица состоящая из 1 элемента;
  4. матрица 0 размера;
  5. матрица размера MxN;
* По положению 0 элемента в столбце:
  1. в столбце средний элемент 0;
  2. в столбце последний элемент 0;
  3. в столбце все элементы 0;

На основе данных областей эквивалентности, для задачи №1, были составлены такие тестовые последовательности:

матрица 0 размера;

матрица содержит 1 элемент = 0;

матрица содержит 1 элемент не равный0;

матрица строка содержит 1 или более 0;

матрица строка не содержит 0 элементов;

матрица столбец содержит 1 или более 0;

матрица столбец не содержит 0;

матрица размера MxN – нулевая;

матрица размера MxN – не содержит 0;

матрица размера MxN – содержит 1 или более 0 в столбце.

Для задачи №2 можно выделить такие области эквивалентности.

* По размеру строки:
  1. строка пустая;
  2. строка содержит 1 символ;
  3. строка содержит >1 символа;
* По наличию соответствующих элементов:
  1. строка состоит только из символов \*/+;
  2. строка не содержит символов \*/+;
  3. строка содержит >1 каждого из символов;
* По позиции соответствующих элементов:
  1. символ в начале строки;
  2. символ в конце строки;
  3. символ в середине строки;

На основе данных областей эквивалентности, для задачи №2, были составлены такие тестовые последовательности:

пустая строка;

строка содержит 1 символ \* или +;

строка содержит 1 символ отличный от \* и +;

строка > 1 символа состоит из + и \*;

строка > 1 символа не содержит + и \*;

строка > 1 символа содержит случайное число + и \*, в начале, середине и конце строки;

Для задачи №3 можно выделить такие области эквивалентности.

* По размеру содержимого файла:

1. файл пустой;
2. файл содержит 1 предложение;
3. файл содержит несколько предложений;

* По содержимому предложений:

1. предложение не содержит искомое слово;
2. предложение содержит искомое слово;

* По позиции слова в предложении:

1. слово стоит в начале предложения;
2. слово стоит в середине предложения;
3. слово стоит в конце предложения;

* По вводимому с клавиатуры слову:

1. слово не введено;
2. слово состоит из пробелов;
3. введено несколько слов;
4. введено одно слово;

На основе данных областей эквивалентности, для задачи №3, были составлены такие тестовые последовательности:

Слово не введено

Вместо слова введены пробелы

Файл пустой

Файл содержит 1 предложение, но не содержит искомого слова

Файл содержит 1 предложение и содержит 1 слово

Файл содержит несколько предложений, каждое из которых содержит несколько искомых слов, в разных частях предложений.

* 1. ПРИМЕРЫ ТЕСТОВЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ



Рисунок 1 – Результат выполнения программы для решения задачи №1, при передаче пустой матрицы в качестве входных данных

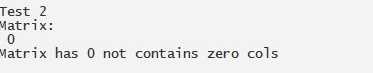


Рисунок 2 – Результат выполнения программы для решения задачи №1, при передаче нуля, в качестве входных данных

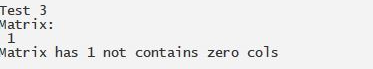


Рисунок 3 - Результат выполнения программы для решения задачи №1, при передаче числа не равного 0, в качестве входных данных

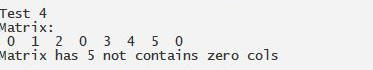


Рисунок 4 – Результат выполнения программы для решения задачи №1, при передаче матрицы-строки, содержащей нулевые элементы

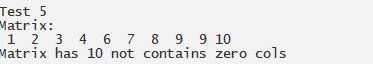


Рисунок 5 – Результат выполнения программы для решения задачи №1, при передаче матрицы-строки, не содержащей нулевые элементы

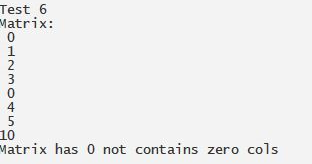


Рисунок 6 – Результат выполнения программы для решения задачи №1, при передаче матрицы-столбца, содержащей нулевые элементы

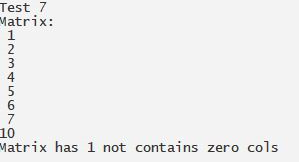


Рисунок 7 – Результат выполнения программы для решения задачи №1, при передаче матрицы-столбца, не содержащей нулевые элементы

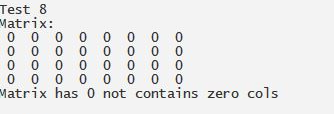


Рисунок 8 – Результат выполнения программы для решения задачи №1, при передаче прямоугольной нулевой матрицы

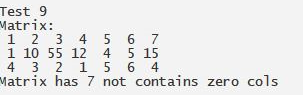


Рисунок 9 – Результат выполнения программы для решения задачи №1, при передаче прямоугольной матрицы, не имеющей нулевых элементов

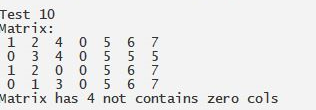


Рисунок 10 – Результат выполнения программы для решения задачи №1, при передаче прямоугольной матрицы, содержащей нулевые элементы в различных позициях



Рисунок 11 – Результат выполнения программы для решения задачи №2, при передаче пустой строки в качестве входных данных

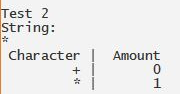


Рисунок 12 – Результат выполнения программы для решения задачи №2, при передаче искомого символа



Рисунок 13 – Результат выполнения программы для решения задачи №2, при передаче символа, отличного от искомого

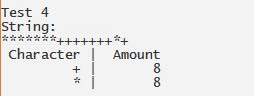


Рисунок 14 – Результат выполнения программы для решения задачи №2, при передаче строки, состоящей из искомых символов

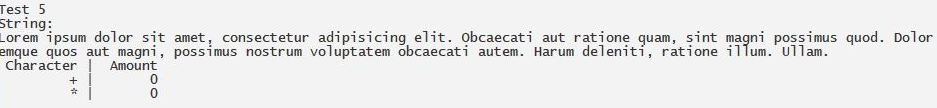


Рисунок 15 – Результат выполнения программы для решения задачи №2, при передаче строки, не содержащей искомых символов

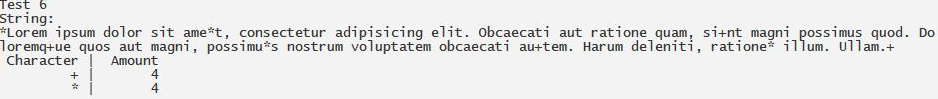


Рисунок 16 – Результат выполнения программы для решения задачи №2, при передаче строки, содержащей искомые символы в различных позициях

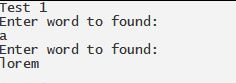


Рисунок 17 – Результат выполнения программы для решения задачи №3, с вводом одного символа, и поиска слова в пустом файле



Рисунок 18 – Результат выполнения программы для решения задачи №3 и поиска в файле, не содержащем искомого слова

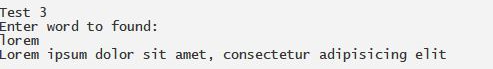


Рисунок 19 – Результат выполнения программы для решения задачи №3 и поиска слова в файле, содержащем искомое слово в одном предложении

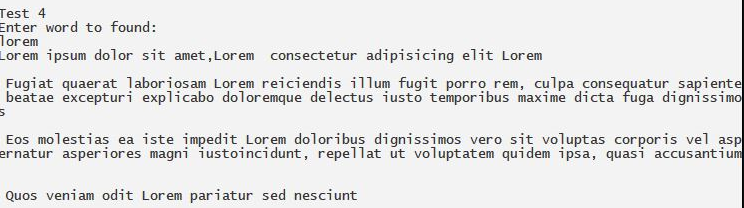


Рисунок 20 – Результат выполнения программы для решения задачи №3 и поиска слова в файле, содержащем искомое слово в нескольких предложениях и различных позициях

# ВЫВОД

В ходе выполнения лабораторной работы были исследованы способы анализа областей эквивалентности входных данных для тестирования программного обеспечения. Были приобретены практические навыки построения тестовых примеров на основании анализа данных областей.

Для решения поставленных задач, были составлены и проанализированы области эквивалентности каждой из них. Так же были разработаны и написаны программы на языке С#, по результатам выполнения которых видно, что при выполнении программ учитываются и обрабатываются все составленные тестовые примеры, поэтому можно сделать вывод, что поставленная цель достигнута.