

1 Лабораторная работа №1

Исследование способов анализа областей эквивалентности и построения тестовых последовательностей

1.1. Цель работы

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

1.2. Постановка задачи

Был выдан вариант 14.

Задача 1. Дана целочисленная прямоугольная матрица. Определить номер столбца, в котором находится самая длинная серия одинаковых элементов.

Задача 2. Дана строка. Преобразовать строку, заменив в ней все восклицательные знаки вопросительными.

Задача 3. Программа, которая считывает текст из файла и выводит на экран только строки, содержащие трехзначные числа.

1.3. Ход выполнения работы

1.3.1. Текст программных модулей

В листингах 1 — 9 представлены тексты программных модулей, написанных на языке Java.

Листинг 1 — Текст основного класса App

```

package org.cory7666.softwaretestingexample;

import java.io.IOException;

import org.cory7666.softwaretestingexample.task1.Task1;
import org.cory7666.softwaretestingexample.task2.Task2;
import org.cory7666.softwaretestingexample.task3.Task3;

public class App
{
    public static void main (String[] args) throws IOException
    {
        new Task1().execute();
        new Task2().execute();
        new Task3().execute();
    }
}

```

Листинг 2 — Текст класса task1.Task1

```

package org.cory7666.softwaretestingexample.task1;

import java.util.Collections;

public class Task1
{
    public Task1 ()
    {
        System.out.println("=== Тесты для Задания 1 ===");
    }

    public void execute ()
    {
        this
            .test(new Matrix(new int[][] { { 1 } }}, 0)
            .test(new Matrix(new int[][] { { 1, 1, 2, 1 }}, { 2, 4, 78, 100 }, { 3, 5, 11, 5 }}, 0)
            .test(
                new Matrix(new int[][] { { 1, -1, 10, 20 }, { 1, 2, 2, 20 }, { 1, -1, 10, 20 }, { 1, -1, 10, 20 } }},
                1)
            .test(
                new Matrix(
                    new int[][] { { 1, 2, 2, 4 }, { 1, 2, 3, 4 }, { 1, 2, 3, 4 }, { 1, 2, 3, 4 }, { 1, 10, 10, 10 } }},
                    4);
    }

    private Task1 test (Matrix matrix, int expect)
    {
        Series answer = new Series(-1, -1);
        for (var column : matrix)
        {
            var maxFromColumn = Collections.max(column.getAllPossibleSeries());
            if (maxFromColumn.counter() >= answer.counter())
            {
                answer = maxFromColumn;
            }
        }

        System.out.printf("Матрица: {\n%s\n}. Ожидается: <%d>. Получено: <%d>.\n", matrix, expect, answer.col);
        return this;
    }
}

```

Листинг 3 — Текст класса task2.Task2

```

package org.cory7666.softwaretestingexample.task2;

public class Task2
{
    public Task2 ()
    {
        System.out.println("=== Тесты для Задания 2 ===");
    }

    public void execute ()
    {
        this
            .test("", "")
            .test("simple string", "simple string")
            .test("!!!cómo estás", "???cómo estás")
            .test("How're you! I'm fine!", "How're you? I'm fine?");
    }

    private Task2 test (String toTest, String expect)
    {
        var result = new MyString(toTest).replaceCharacters();
        System.out
            .printf(
                "Тест для: <%s>. Ожидается: <%s>. Получено: <%s>. Итог: %b.\n",
                toTest,
                expect,
                result,
                result.equals(expect));
        return this;
    }
}

```

Листинг 4 — Текст класса task3.Task3

```

package org.cory7666.softwaretestingexample.task3;

public class Task3
{
    public Task3 ()
    {
        System.out.println("=== Тесты для Задания 3 ===");
    }

    public void execute ()
    {
        this
            .test("34")
            .test("Было подобрано число 33.")
            .test("101 далматинец")
            .test("2022 год")
            .test("13 литров 00509 миллилитров")
            .test("11 точка 0000900");
    }

    private Task3 test (String toTest)
    {
        System.out.printf("Тест для: <%s>. Итог: %b.\n", toTest, new MyLine(toTest).containsThreeDigitNumber());
        return this;
    }
}

```

Листинг 5 — Текст класса task1.Matrix

```

package org.cory7666.softwaretestingexample.task1;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

public class Matrix implements Iterable<MatrixColumn>
{

```

```

private final List<MatrixColumn> matrix;
public final int rows, cols;

public Matrix (int[][] data)
{
    try
    {
        this.cols = data.length;
        this.rows = data[0].length;

        this.matrix = new ArrayList<>(this.cols);
        for (int i = 0; i < data.length; ++i)
        {
            this.matrix.add(new MatrixColumn(i, data[i]));
        }
    }
    catch (NullPointerException e)
    {
        throw new IllegalArgumentException("Argument data cann't be null.");
    }
}

public MatrixColumn getColumn (int col)
{
    return matrix.get(col);
}

@Override
public Iterator<MatrixColumn> iterator ()
{
    return matrix.iterator();
}

@Override
public String toString ()
{
    var ret = new StringBuilder();

    for (int i = 0; i < rows; ++i)
    {
        for (var col : matrix)
        {
            ret.append(col.get(i) + "\t");
        }

        if (i + 1 != rows)
            ret.append('\n');
    }

    return ret.toString();
}
}

```

Листинг 6 — Текст класса task1.Series

```

package org.cory7666.softwaretestingexample.task1;

public class Series implements Comparable<Series>
{
    public final int col, value;
    private int counter;

    public Series (int col, int value)
    {
        this.col = col;
        this.value = value;
        this.counter = 1;
    }

    public int counter ()
    {
        return counter;
    }
}

```

```

public void increase ()
{
    ++counter;
}

@Override
public int compareTo (Series o)
{
    return this.counter() - o.counter();
}
}

```

Листинг 7 — Текст класса task1.MatrixColumn

```

package org.cory7666.softwaretestingexample.task1;

import java.util.HashSet;
import java.util.Set;

public class MatrixColumn
{
    public final int colIndex;
    private final int[] data;

    public MatrixColumn (int colIndex, int[] data)
    {
        this.data = data;
        this.colIndex = colIndex;
    }

    public Set<Series> getAllPossibleSeries ()
    {
        Set<Series> ret = new HashSet<>();
        var currentSeries = new Series(colIndex, data[0]);

        for (int i = 1; i < data.length; ++i)
        {
            if (data[i] == currentSeries.value)
            {
                currentSeries.increase();
            }
            else
            {
                ret.add(currentSeries);
                currentSeries = new Series(colIndex, data[i]);
            }
        }

        ret.add(currentSeries);
        return ret;
    }

    public int get (int i)
    {
        return data[i];
    }
}

```

Листинг 8 — Текст класса task2.MyString

```

package org.cory7666.softwaretestingexample.task2;

public class MyString
{
    private final String string;

    public MyString (String string)
    {

```

```

    if (string == null)
        throw new IllegalArgumentException("Passed string is null.");
    this.string = string;
}

public String replaceCharacters ()
{
    return string.replaceAll("!", "?");
}
}

```

Листинг 9 — Текст класса task3.MyLine

```

package org.cory7666.softwaretestingexample.task3;

public class MyLine
{
    public final String line;

    public MyLine (String line)
    {
        this.line = line;
    }

    public boolean containsThreeDigitNumber ()
    {
        int digitCounter = 0;
        for (int i = 0; i < line.length(); ++i)
        {
            if ('0' <= line.charAt(i) && line.charAt(i) <= '9')
            {
                if (('1' <= line.charAt(i) && line.charAt(i) <= '9') || (line.charAt(i) == '0' && digitCounter != 0))
                {
                    ++digitCounter;
                }
            }
            else
            {
                digitCounter = 0;
            }

            if (digitCounter == 3 && (i + 1 < line.length() && !('0' <= line.charAt(i + 1) && line.charAt(i + 1) <=
'9')
                || i + 1 == line.length()))
            {
                return true;
            }
        }
        return false;
    }
}

```

1.3.2. Описание областей эквивалентностей

Для каждого из предложенных заданий была составлена своя область эквивалентностей.

Для первого задания были составлены следующие области эквивалентности:

1. по размеру матрицы:
 1. из одного столбца и одной строки;

2. из нескольких столбцов и строк;
2. по положению в матрице элементов, входящих в серию:
 1. начиная с первой строки первого столбца;
 2. начиная со строки, расположенной посередине, в столбце, расположенном посередине;
 3. начиная с предпоследней строки в последнем столбце.

Для второго задания были выделены следующие области эквивалентности:

1. по длине строки:
 1. строка не содержит символов;
 2. строка содержит один или более символов;
2. по количеству искомым символов:
 1. нуль искомым символов;
 2. один и более искомым символов;
3. по положению искомого символа в строке:
 1. в начале строки;
 2. в середине строки;
 3. в конце строки.

Для третьего задания было выделено несколько областей эквивалентностей:

- 1 по количеству символов в строке:
 - 1 строки, длиной не более 2-х символов;
 - 2 строки, длиной 3 и более символов;
- 2 по количеству цифр в числе:
 - 1 две цифры в числе;
 - 2 три цифры в числе;
 - 3 четыре цифры в числе;
 - 4 число содержит $(n + 3)$ цифр, из которых n являются ведущими нулями;
- 3 по положению ключевого числа в строке:
 - 1 в начале строки;
 - 2 в середине строки;

3 в конце строки.

1.3.3.Примеры тестовых последовательностей

После определения классов входных данных были составлены тестовые последовательности, представленные в таблицах 1.1 — 1.3. Для таблиц 1.1 и 1.2 ожидается число или строка как возвращаемое из функции значение соответственно. Для таблицы 1.3 в столбце «Ожидаемый результат» значение true означает, что строка будет выведена на экран, false в противоположном случае.

Таблица 1.1 — Тестовые последовательности для задания 1

Соответствие классам	Тестовая последовательность	Ожидаемый результат
1a	[1]	0
1б, 2a	1 2 3 (1 4 5) (2 78 11) 1 100 5	0
1б, 2б	1 1 1 1 (-1 2 -1 -1) (10 2 10 10) 20 20 20 20	1
1б, 2в	1 1 1 1 1 (2 2 2 2 10) (2 3 3 3 10) 4 4 4 4 10	4

Таблица 1.2 — Тестовая последовательность для задания 2

Соответствие классам	Тестовая последовательность	Ожидаемый результат
1a	«»	«»
1б, 2a	«simple string»	«simple string»
1б, 2б, 3a	«!!!cómo estás»	«???cómo estás»
1б, 2б, 3б, 3в	«How're you! I'm fine!»	«How're you? I'm fine?»


Таблица 1.3 — Тестовая последовательность для задания 3

Соответствие классам	Тестовая последовательность	Ожидаемый результат
----------------------	-----------------------------	---------------------

1a	«34»	false
1б, 2а	«Было подобрано число 33.»	false
1б, 2б, 3а	«101 далматинец»	true
1б, 2в	«2022 год»	false
1б, 2г, 3б	«13 литров 00509 миллилитров»	true
1б, 2г, 3в	«11 точка 0000900»	true

Тестовые последовательности были перенесены в программу. Программа была скомпилирована и запущена. Результат тестирования продемонстрирован на рисунке. Все тесты были пройдены, из чего следует, что разработанные алгоритмы с большой вероятностью дадут верный результат при любых входных данных.

```

 /tmp java -jar Lab 1.jar
=== Тесты для Задания 1 ===
Матрица: {
1
}. Ожидается: <0>. Получено: <0>.
Матрица: {
1      2      3
1      4      5
2      78     11
1      100    5
}. Ожидается: <0>. Получено: <0>.
Матрица: {
1      1      1      1
-1     2      -1     -1
10     2      10     10
20     20     20     20
}. Ожидается: <1>. Получено: <1>.
Матрица: {
1      1      1      1      1
2      2      2      2      10
2      3      3      3      10
4      4      4      4      10
}. Ожидается: <4>. Получено: <4>.
=== Тесты для Задания 2 ===
Тест для: <>. Ожидается: <>. Получено: <>. Итог: true.
Тест для: <simple string>. Ожидается: <simple string>. Получено: <simple string>. Итог: true.
Тест для: <!!!cómo estás>. Ожидается: <???cómo estás>. Получено: <???cómo estás>. Итог: true.
Тест для: <How're you! I'm fine!>. Ожидается: <How're you? I'm fine?>. Получено: <How're you? I'm fine?>. Итог: true.
=== Тесты для Задания 3 ===
Тест для: <34>. Итог: false.
Тест для: <Было подобрано число 33.>. Итог: false.
Тест для: <101 далматинец>. Итог: true.
Тест для: <2022 год>. Итог: false.
Тест для: <13 литров 00509 миллилитров>. Итог: true.
Тест для: <11 точка 0000900>. Итог: true.

```

Рисунок 1.1 — Тестовый запуск программы

Выводы по результатам работы

При выполнении данной лабораторной работы были освоены навыки составления тестов, нахождения областей эквивалентности. Получено понимание принципов выделения областей эквивалентности и построения тестов на их основе, что позволит составлять небольшое количество тестов, покрывающих большое количество комбинаций данных.