

Министерство науки и высшего образования Российской Федерации  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ  
ИМЕНИ В.Ф.УТКИНА»  
Рязанский станкостроительный колледж РГРТУ

Лабораторная работа №3

**Дополнительная профессиональная программа  
повышения квалификации  
«Тестирование программного обеспечения (с учетом стандарта Ворлдскиллс по  
компетенции «Программные решения для бизнеса»)»**

Мазуренко Валерия Витальевна

Рязань 2022

### Задание №3

#### Практическая работа №2

#### Виды тестирования. Планирование тестирования

##### Теория:

##### 1. Методы стратегии белого ящика'

Тестирование по принципу белого ящика характеризуется степенью, какой тесты выполняют или покрывают логику (исходный текст программы).

##### 2.1.1 Метод покрытия операторов

Целью этого метода тестирования является выполнение каждого оператора программы хотя бы один раз.

В этой программе можно выполнить каждый оператор, записав один единственный тест, который реализовал бы путь асе. Т.е., если бы на входе было:  $A=2$ ,  $B=0$ ,  $X=3$ , каждый оператор выполнялся бы один раз. Но этот критерий на самом деле хуже, чем он кажется на первый взгляд. Пусть в первом условии вместо "and"®"or" и во втором вместо " $x>1$ "®" $x<1$ " (блок-схема правильной программы приведена на рисунке 2.1, а неправильной - на рисунке 2.2). Результаты тестирования приведены в таблице 2.1. Обратите внимание: ожидаемый результат определяется по алгоритму на рисунке 2.1, а фактический - по алгоритму рисунка 2.2, поскольку определяется чувствительность метода тестирования к ошибкам программирования. Как видно из этой таблицы, ни одна из внесенных в алгоритм ошибок не будет обнаружена.

Таблица 2.1 - Результат тестирования методом покрытия операторов

Тест	Ожидаемый результат	Фактический результат	Результат тестирования
$A=2$ , $B=0$ , $X=3$	$X=2,5$	$X=2,5$	неуспешно

##### 2.2 Метод покрытия решений (покрытия переходов)

Более сильный метод тестирования известен как покрытие решений (покрытие переходов). Согласно данному методу каждое направление перехода должно быть реализовано по крайней мере один раз.

Покрытие решений обычно удовлетворяет критерию покрытия операторов. Поскольку каждый оператор лежит на некотором пути, исходящем либо из оператора перехода, либо из точки входа программы, при выполнении каждого направления перехода каждый оператор должен быть выполнен.

Для программы приведенной на рисунке 2.2 покрытие решений может быть выполнено двумя тестами, покрывающими пути {ace, abd}, либо {acd,abe}. Пути {acd,abe} покроем, выбрав следующие исходные данные:  $\{A=3, B=0, X=3\}$  и  $\{A=2, B=1, X=1\}$  (результаты тестирования - в таблице 2.2).

Таблица 2.2 - Результат тестирования методом покрытия решений

Тест	Ожидаемый результат	Фактический результат	Результат тестирования
$A=3$ , $B=0$ , $X=3$	$X=1$	$X=1$	неуспешно
$A=2$ , $B=1$ , $X=1$	$X=2$	$X=1,5$	успешно

##### 2.3 Метод покрытия условий

Лучшие результаты по сравнению с предыдущими может дать метод покрытия условий. В этом случае записывается число тестов, достаточное для того, чтобы все возможные результаты каждого условия в решении выполнялись по крайней мере один раз.

В предыдущем примере имеем четыре условия:  $\{A>1, B=0\}$ ,  $\{A=2, X>1\}$ . Следовательно, требуется достаточное число тестов, такое, чтобы реализовать ситуации, где  $A>1, A \neq 1, B=0$  и  $B \neq 0$  в точке  $A=2, A \neq 2, X>1$  и  $X \neq 1$  в точке  $B$ . Тесты, удовлетворяющие критерию покрытия условий и соответствующие им пути:

а)  $A=2, B=0, X=4$  ace

б)  $A=1, B=1, X=0$  abd

Таблица 2.3 - Результаты тестирования методом покрытия условий

Тест	Ожидаемый результат	Фактический результат	Результат тестирования
$A=2, B=0, X=4$	$X=3$	$X=3$	неуспешно
$A=1, B=1, X=0$	$X=0$	$X=1$	успешно

## 2.4 Критерий решений (условий)

Критерий покрытия решений/условий требует такого достаточного набора тестов, чтобы все возможные результаты каждого условия в решении выполнялись по крайней мере один раз, все результаты каждого решения выполнялись по крайней мере один раз и, кроме того, каждой точке входа передавалось управление по крайней мере один раз.

Два теста метода покрытия условий

а)  $A=2, B=0, X=4$  ace

б)  $A=1, B=1, X=0$  abd

отвечают и критерию покрытия решений/условий. Это является следствием того, что одни условия приведенных решений скрывают другие условия в этих решениях. Так, если условие  $A>1$  будет ложным, транслятор может не проверять условия  $B=0$ , поскольку при любом результате условия  $B=0$ , результат решения  $((A>1) \& (B=0))$  примет значение ложь. Следовательно, недостатком критерия покрытия решений/условий является невозможность его применения для выполнения всех результатов всех условий.

Другая реализация рассматриваемого примера приведена на рисунке 2.3. Многоусловные решения исходной программы разбиты на отдельные решения и переходы. Наиболее полное покрытие тестами в этом случае выполняется так, чтобы выполнялись все возможные результаты каждого простого решения. Для этого нужно покрыть пути  $HILP$ (тест  $A=2, B=0, X=4$ ),  $HIMKT$ (тест  $A=3, B=1, X=0$ ),  $HJKT$ (тест  $A=0, B=0, X=0$ ),  $HJKR$ (тест  $A=0, B=0, X=2$ )..

## 2.5 Метод комбинаторного покрытия условий.

Критерием, который решает эти и некоторые другие проблемы, является комбинаторное покрытие условий. Он требует создания такого числа тестов, чтобы все возможные комбинации результатов условия в каждом решении выполнялись по крайней мере один раз. Набор тестов, удовлетворяющих критерию комбинаторного покрытия условий, удовлетворяет также и критериям покрытия решений, покрытия условий и покрытия решений/условий.

По этому критерию в рассматриваемом примере должны быть покрыты тестами следующие восемь комбинаций:

- а)  $A > 1, B = 0$ ;
- б)  $A > 1, B \neq 0$ ;
- в)  $A \neq 1, B = 0$ ;
- г)  $A \neq 1, B \neq 0$ ;
- д)  $A = 2, X > 1$ ;
- е)  $A = 2, X \neq 1$ ;
- ж)  $A \neq 2, X > 1$ ;

з)  $A \neq 2, X \neq 1$ ; Для того чтобы протестировать эти комбинации, необязательно использовать все 8 тестов. Фактически они могут быть покрыты четырьмя тестами:

- $A = 2, B = 0, X = 4$  {покрывает а, д};
- $A = 2, B = 1, X = 1$  {покрывает б, е};
- $A = 0,5, B = 0, X = 2$  (покрывает в, ж);
- $A = 1, B = 0, X = 1$  {покрывает г, з}.

Таблица 2.4 - Результаты тестирования методом комбинаторного покрытия условий

Тест	Ожидаемый результат	Фактический результат	Результат тестирования
$A = 2, B = 0, X = 4$	$X = 3$	$X = 3$	неуспешно
$A = 2, B = 1, X = 1$	$X = 2$	$X = 1,5$	успешно
$A = 0,5, B = 0, X = 2$	$X = 3$	$X = 4$	успешно
$A = 1, B = 0, X = 1$	$X = 1$	$X = 1$	неуспешно

#### Задание

1. Написать программу, реализующую заданный преподавателем алгоритм обработки данных (взять программы из задания 2).
2. Провести тестирование программного продукта рассмотренными методами.
3. Выписать пути алгоритма, которые должны быть проверены тестами для рассматриваемого метода тестирования.
4. Записать тесты, которые позволят пройти по путям алгоритма.
5. Протестировать разработанную Вами программу. Результаты оформить в виде таблиц (таблицы 2.1-2.4).
6. Сделать скриншоты программы
7. Оформить отчет по лабораторной работе

## ЛАБОРАТОРНАЯ РАБОТА №3

### Задания 2

1. Даны числа  $a$  и  $b$ . Если оба значения положительны, то каждое из них удвоить; если оба отрицательны, то отбросить их знаки; в противном случае оставить числа без изменения.

№ теста	Входные данные	Формула	Ожидаемый результат	Полученный результат
1	$a=2$ $b=3$	$x=a*2$ $y=b*2$	$x=4$ $y=6$	верно
8	$a=-2$ $b=й$	$x=a$ $y=b$	Ошибка неверно значение введено	неверно
9	$a=й$ $b=й$	$x=a$ $y=b$	Ошибка неверно значение введено	неверно

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static System.Net.Mime.MediaTypeNames;

namespace lera2
{
    internal class Program
    {
        static void Main(string[] args)
        {

            string s1, s2;
            double a, b, x, y;

            Console.Write("a = ");

            s1 = Console.ReadLine();

            Console.Write("b = ");
            s2 = Console.ReadLine();

            /* if (!double.TryParse(s1, out a))
            {
                Console.WriteLine("Значения А не является числом");
            }
            if (!double.TryParse(s2, out b))
            {
                Console.WriteLine("Значения В не является числом");
            }
            */
            if (double.TryParse(s1, out a) && double.TryParse(s2, out b))
            {
                if (a >= 0 && b >= 0)
                {
                    x = a * 2;
                    y = b * 2;
                }
                else if (a < 0 && b < 0)
                {
                    x = -a;
                    y = -b;
                }
            }
        }
    }
}
```

```

    {
        x = Math.Abs(a);
        y = Math.Abs(b);
    }
    else
    {
        x = a;
        y = b;
    }
    Console.WriteLine("x = " + x);
    Console.WriteLine("y = " + y);
}
else
{
    Console.WriteLine("Ошибка введено неверно значение");
}
Console.ReadKey();
}
}
}

```

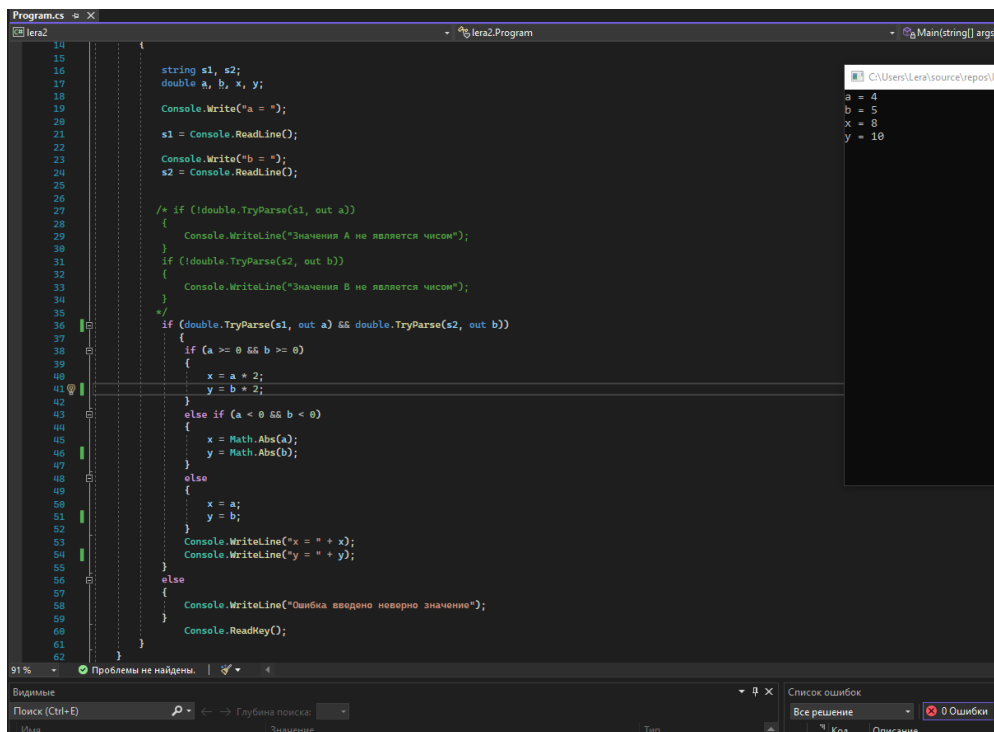


Рисунок 1 –Код программы

## 2.1 Метод покрытия операторов

Целью этого метода тестирования является выполнение каждого оператора программы хотя бы один раз.

Таблица 2.1 - Результат тестирования методом покрытия решений

№	Тест	Ожидаемый результат	Фактический результат	Результат тестирования
1	a=2, b=0	x=4, y=0	x=4, y=0	успешно

```
C:\Users\Lera\sou
a = 2
b = 0
x = 4
y = 0
```

Рисунок 2-2.1 Метод покрытия операторов

## 2.2 Метод покрытия решений (покрытия переходов)

Каждое направление перехода должно быть реализовано по крайней мере один раз.

Оператор должен лежать на некотором пути, исходящем либо из оператора перехода, либо из точки входа программы, при выполнении каждого направления перехода каждый оператор должен быть выполнен.

Таблица 2.2 - Результат тестирования методом покрытия решений

№	Тест	Ожидаемый результат	Фактический результат	Результат тестирования
1	a=2, b=-3	x=2 y=-3	x=2 y=-3	успешно
2	a=й b=3	x=й y=3	ошибка	неуспешно

```
C:\Users\Lera\source\repos\lera2\lera2\bin\Debug\lera2.exe
a = 2
b = -3
x = 2
y = -3
```

Рисунок 3- 2.2Метод покрытия решений (покрытия переходов) №1 тест

```
C:\Users\Lera\source\repos\lera2\lera2\bin\Debug\lera2.exe
a = й
b = 3
Ошибка введено неверно значение
```

Рисунок 4- 2.2Метод покрытия решений (покрытия переходов) №2 тест

## 2.3 Метод покрытия условий

В этом случае записывается число тестов, достаточное для того, чтобы все возможные результаты каждого условия в решении выполнялись по крайней мере один раз.

Таблица 2.3 - Результаты тестирования методом покрытия условий

№	Тест	Ожидаемый результат	Фактический результат	Результат тестирования
1	a=-2 b=3	x=-2 y=3	x=-2 y=3	успешно

2	a=й b=-3	x=й y=-3	ошибка	неуспешно
---	-------------	-------------	--------	-----------

```

C:\Users\Lera\source\repos\lera2\lera2\bin\Debug\lera2.exe
a = й
b = -3
Ошибка введено неверно значение
x = -2
y = 3

```

Рисунок 5- 2.3 Метод покрытия

условий №1 тест

Рисунок 6- 2.3 Метод покрытия условий №2 тест

## 2.4 Критерий решений (условий)

Критерий покрытия решений/условий требует такого достаточного набора тестов, чтобы все возможные результаты каждого условия в решении выполнялись по крайней мере один раз, все результаты каждого решения выполнялись по крайней мере один раз и, кроме того, каждой точке входа передавалось управление по крайней мере один раз.

Таблица 2.4 - Критерий решений (условий)

№	Тест	Ожидаемый результат	Фактический результат	Результат тестирования
1	a=-2 b=-3	x=2 y=3	x=2 y=3	успешно
2	a=2 b=й	x=2 y=й	ошибка	неуспешно

```

C:\Users\Lera\source\repos\lera2\lera2\bin\Debug\lera2.exe
a = -2
b = -3
x = 2
y = 3

```

Рисунок 7- 2.4 - Критерий решений (условий ) №1 тест

```

C:\Users\Lera\source\repos\lera2\lera2\bin\Debug\lera2.exe
a = 2
b = й
Ошибка введено неверно значение

```

Рисунок 8- 2.4 - Критерий решений (условий ) №2 тест

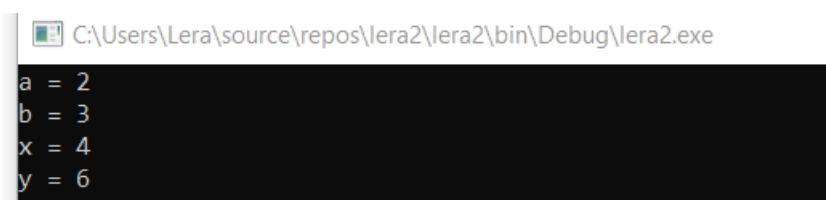


## 2.5 Метод комбинаторного покрытия условий.

Критерием, который решает эти и некоторые другие проблемы, является комбинаторное покрытие условий. Он требует создания такого числа тестов, чтобы все возможные комбинации результатов условия в каждом решении выполнялись по крайней мере один раз. Набор тестов, удовлетворяющих критерию комбинаторного покрытия условий, удовлетворяет также и критериям покрытия решений, покрытия условий и покрытия решений/условий.

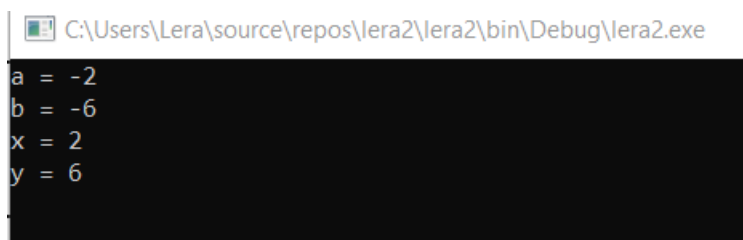
Таблица 2.5 - Метод комбинаторного покрытия условий.

№	Тест	Ожидаемый результат	Фактический результат	Результат тестирования
1	a=2 b=3	x=2 y=3	x=2 y=3	успешно
2	a=-2 b=-6	x=-2 y=-6	x=-2 y=-6	успешно
3	a=-6 b=4	x=-6 y=4	x=-6 y=4	успешно
4	a=6 b=й	x=6 y=й	ошибка	неуспешно



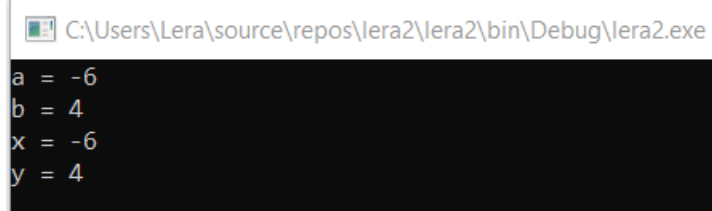
```
C:\Users\Lera\source\repos\lera2\lera2\bin\Debug\lera2.exe
a = 2
b = 3
x = 4
y = 6
```

Рисунок 9- 2.4 - Критерий решений (условий ) №1 тест



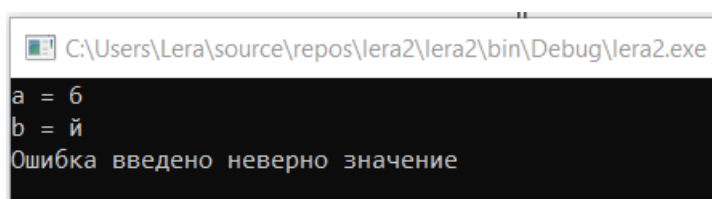
```
C:\Users\Lera\source\repos\lera2\lera2\bin\Debug\lera2.exe
a = -2
b = -6
x = 2
y = 6
```

Рисунок 10- 2.4 - Критерий решений (условий ) №2 тест



```
C:\Users\Lera\source\repos\lera2\lera2\bin\Debug\lera2.exe
a = -6
b = 4
x = -6
y = 4
```

Рисунок 11- 2.4 - Критерий решений (условий ) №3 тест



```
C:\Users\Lera\source\repos\lera2\lera2\bin\Debug\lera2.exe
a = 6
b = й
Ошибка введено неверно значение
```

Рисунок 12- 2.4 - Критерий решений (условий ) №4 тест

7. Для натурального числа  $k \leq 7$ , выводит на экран день недели и сообщение, рабочий день или выходной.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ypr2_Lera
{
    internal class Program
    {
        static void Main(string[] args)
        {
            int k;
            string s1;
            Console.WriteLine("Понедельник\nВторник\nСреда\nЧетверг\nПятница\nСуббота\nВоскресенье\n");
            Console.Write("Введите число месяца ");
            //k = Convert.ToInt32(Console.ReadLine());
            s1 = Console.ReadLine();
            if (int.TryParse(s1, out k))
            {
                switch (k)
                {
                    case 1:
                        Console.WriteLine("Понедельник");
                        break;
                    case 2:
                        Console.WriteLine("Вторник");
                        break;
                    case 3:
                        Console.WriteLine("Среда");
                        break;
                    case 4:
                        Console.WriteLine("Четверг");
                        break;
                    case 5:
                        Console.WriteLine("Пятница");
                        break;
                    case 6:
                        Console.WriteLine("Суббота");
                        break;
                    case 7:
                        Console.WriteLine("Воскресенье");
                        break;
                }
                if (k > 0 && k <= 5)
                {
                    Console.WriteLine("Рабочий день");
                }
                else if (k >= 6 && k <= 7)
                {
                    Console.WriteLine("Выходной");
                }
                else
                {
                    Console.WriteLine("Ошибка введено неверно значение");
                }
            }
            else
            {

```

```

        Console.WriteLine("Ошибка введено неверно значение");
    }
    Console.ReadKey();
}
}
}

```

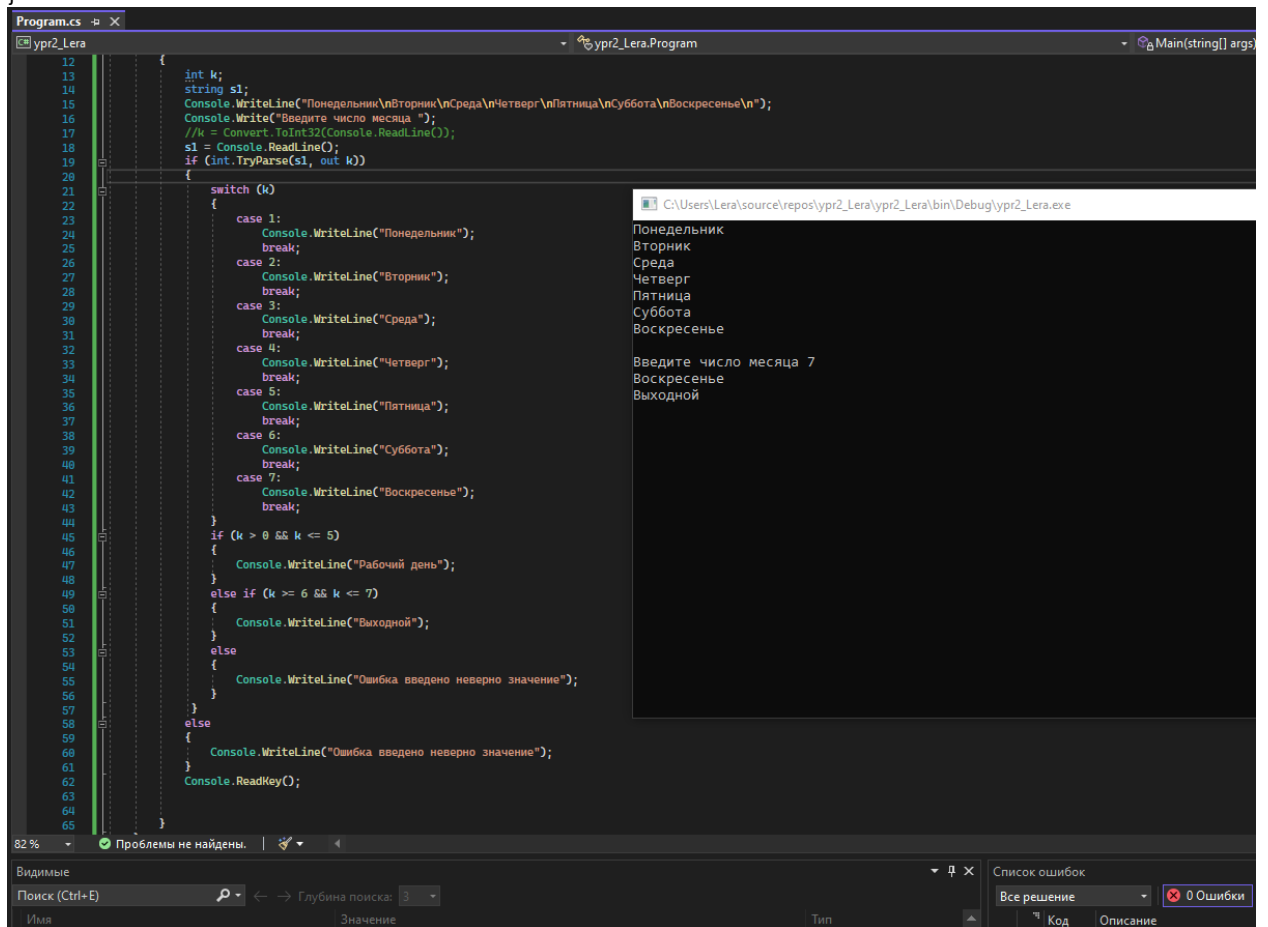


Рисунок 13- Код программы

## 2.1 Метод покрытия операторов

Целью этого метода тестирования является выполнение каждого оператора программы хотя бы один раз.

Таблица 2.1 - Результат тестирования методом покрытия решений

№	Тест	Ожидаемый результат	Фактический результат	Результат тестирования
1	k=1	Понедельник Рабочий день	Понедельник Рабочий день	успешно

```
C:\Users\Lera\source\repos\ypr2_Lera\ypr2_Lera\bin\Debug\ypr2_Lera.exe
Понедельник
Вторник
Среда
Четверг
Пятница
Суббота
Воскресенье

Введите число месяца 1
Понедельник
Рабочий день
```

Рисунок 14-2.1 Метод покрытия операторов

## 2.2 Метод покрытия решений (покрытия переходов)

Каждое направление перехода должно быть реализовано по крайней мере один раз.

Оператор должен лежать на некотором пути, исходящем либо из оператора перехода, либо из точки входа программы, при выполнении каждого направления перехода каждый оператор должен быть выполнен.

Таблица 2.2 - Результат тестирования методом покрытия решений

№	Тест	Ожидаемый результат	Фактический результат	Результат тестирования
1	k=2	Вторник Рабочий день	Вторник Рабочий день	успешно
2	k=9	Вторник Рабочий день	ошибка	неуспешно

```
C:\Users\Lera\source\repos\ypr2_Lera\ypr2_Lera\bin\Debug\ypr2_Lera.exe
Понедельник
Вторник
Среда
Четверг
Пятница
Суббота
Воскресенье

Введите число месяца 2
Вторник
Рабочий день
```

Рисунок 15- 2.2 Метод покрытия решений (покрытия переходов) №1 тест

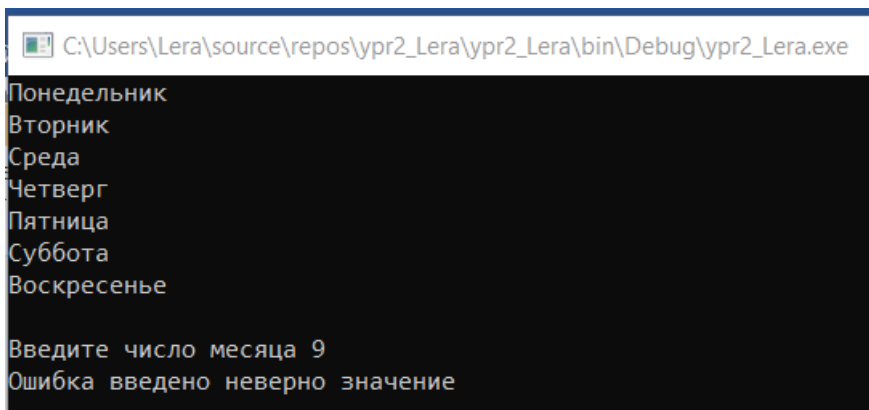


Рисунок 16- 2.2Метод покрытия решений (покрытия переходов) №2 тест

### 2.3 Метод покрытия условий

В этом случае записывается число тестов, достаточное для того, чтобы все возможные результаты каждого условия в решении выполнялись по крайней мере один раз.

Таблица 2.3 - Результаты тестирования методом покрытия условий

№	Тест	Ожидаемый результат	Фактический результат	Результат тестирования
1	k=3	Среда Рабочий день	Среда Рабочий день	успешно
2	k=0	ошибка	ошибка	успешно

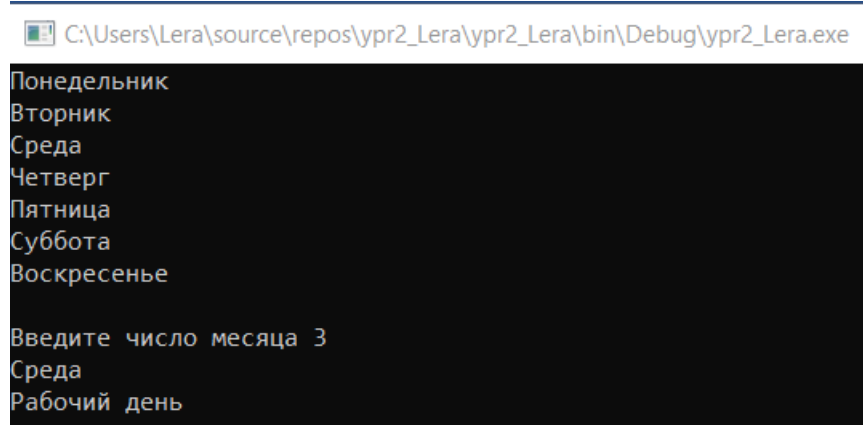


Рисунок 17- 2.3 Метод покрытия условий №1 тест

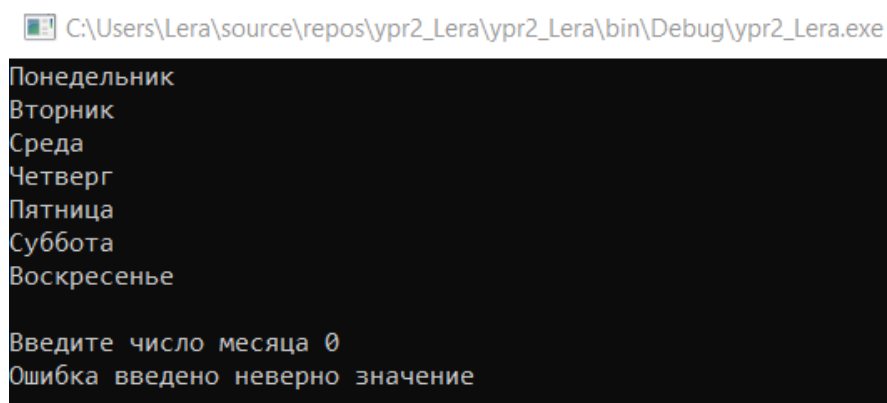


Рисунок 18- 2.3 Метод покрытия условий №2 тест

## 2.4 Критерий решений (условий)

Критерий покрытия решений/условий требует такого достаточного набора тестов, чтобы все возможные результаты каждого условия в решении выполнялись по крайней мере один раз, все результаты каждого решения выполнялись по крайней мере один раз и, кроме того, каждой точке входа передавалось управление по крайней мере один раз.

Таблица 2.4 - Критерий решений (условий)

№	Тест	Ожидаемый результат	Фактический результат	Результат тестирования
1	k=4	Четверг Рабочий день	Четверг Рабочий день	успешно
2	k=55	Воскресенье Рабочий день	ошибка	неуспешно

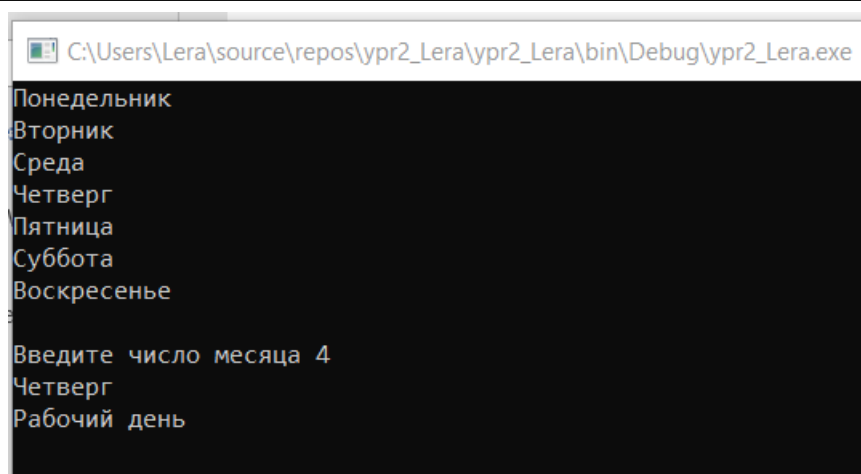


Рисунок 19- 2.4 - Критерий решений (условий ) №1 тест

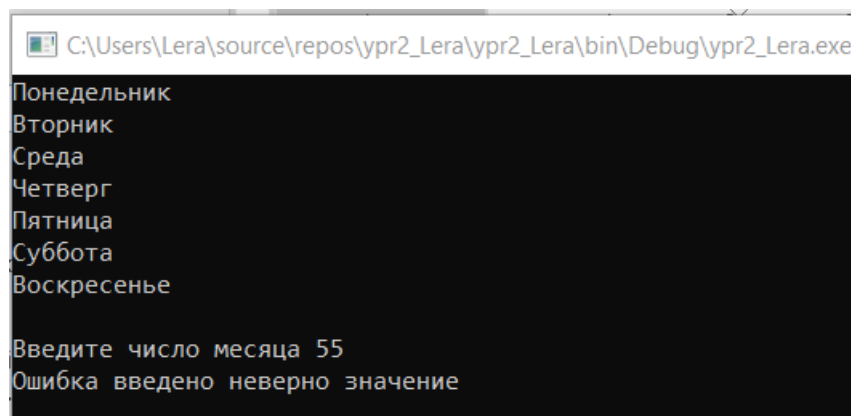


Рисунок 20- 2.4 - Критерий решений (условий ) №2 тест

## 2.5 Метод комбинаторного покрытия условий.

Критерием, который решает эти и некоторые другие проблемы, является комбинаторное покрытие условий. Он требует создания такого числа тестов, чтобы все возможные комбинации результатов условия в каждом решении выполнялись по крайней мере один раз. Набор тестов, удовлетворяющих критерию комбинаторного покрытия условий, удовлетворяет также и критериям покрытия решений, покрытия условий и покрытия решений/условий.

Таблица 2.5 - Метод комбинаторного покрытия условий.

№	Тест	Ожидаемый результат	Фактический результат	Результат тестирования
1	k=5	Пятница Рабочий день	Пятница Рабочий день	успешно
2	k=6	Суббота Выходной	Суббота Выходной	успешно
3	k=10	Среда Рабочий день	ошибка	неуспешно
4	k=месяц	Ошибка	ошибка	неуспешно

```

C:\Users\Lera\source\repos\ypr2_Lera\ypr2_Lera\bin\Debug\ypr2_Lera.exe
Понедельник
Вторник
Среда
Четверг
Пятница
Суббота
Воскресенье

Введите число месяца 5
Пятница
Рабочий день

```

Рисунок 21- 2.4 - Критерий решений (условий ) №1 тест

```

C:\Users\Lera\source\repos\ypr2_Lera\ypr2_Lera\bin\Debug\ypr2_Lera.exe
Понедельник
Вторник
Среда
Четверг
Пятница
Суббота
Воскресенье

Введите число месяца 6
Суббота
Выходной

```

Рисунок 22- 2.4 - Критерий решений (условий ) №2 тест



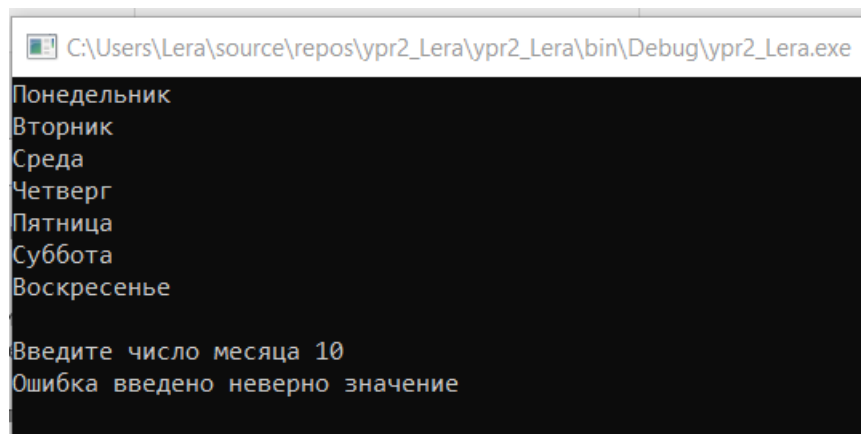


Рисунок 23- 2.4 - Критерий решений (условий ) №3 тест

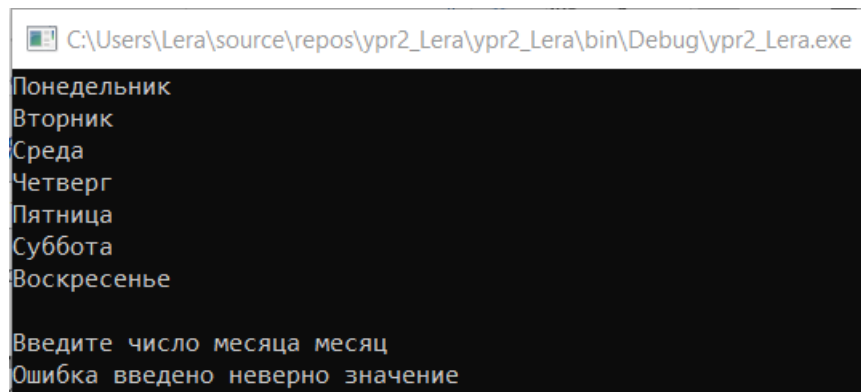


Рисунок 24- 2.4 - Критерий решений (условий ) №4 тест