

Министерство цифрового развития, связи и массовых коммуникаций Российской
Федерации
Федеральное государственное бюджетное образовательное учреждение высшего
образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Лабораторная работа №3

Вариант №4

«Модульное тестирование программ на языке C++ в среде VisualStudio»

Выполнил: студент 4 курса

ИВТ, гр. ИП-013

Копытина Т.А.

Проверил: ассистент кафедры

ПМиК

Агалаков А.А.

Новосибирск, 2023 г.

Цель

Сформировать практические навыки разработки тестов и модульного тестирования на языке C++ с помощью средств автоматизации VisualStudio

Задание

Разработайте на языке C++ класс, содержащий набор функций в соответствии с вариантом задания. Разработайте тестовые наборы данных по критерию C2 для тестирования функций класса. Протестировать функции с помощью средств автоматизации модульного тестирования VisualStudio. Провести анализ выполненного теста и, если необходимо отладку кода. Написать отчёт о результатах проделанной работы.

1. Функция получает целое числа a . Находит и возвращает номер разряда, в котором находится минимальное значение r среди нечётных разрядов целого числа a с нечётным. Разряды числа, пронумерованы справа налево, начиная с единицы. Например, $a = 12543$, $r = 3$.
2. Функция получает целое числа a . Возвращает число, полученное циклическим сдвигом значений разрядов целого числа a на заданное число позиций влево. Например, сдвиг на две позиции: Исходное число: 123456 Результат: 345612
3. Функция получает целые числа a , b и n . Возвращает число, полученное путём вставки разрядов числа b в целое число a после разряда, заданного числом n . Разряды нумеруются слева направо, начиная с 1. Например, вставить после 2 разряда значение 6: Исходное число: 123457 вставить 6 после 2 разряда Результат: 1263457
4. Функция получает двумерный массив вещественных переменных A . Отыскивает и возвращает сумму чётных значений компонентов массива, лежащих ниже побочной диагонали

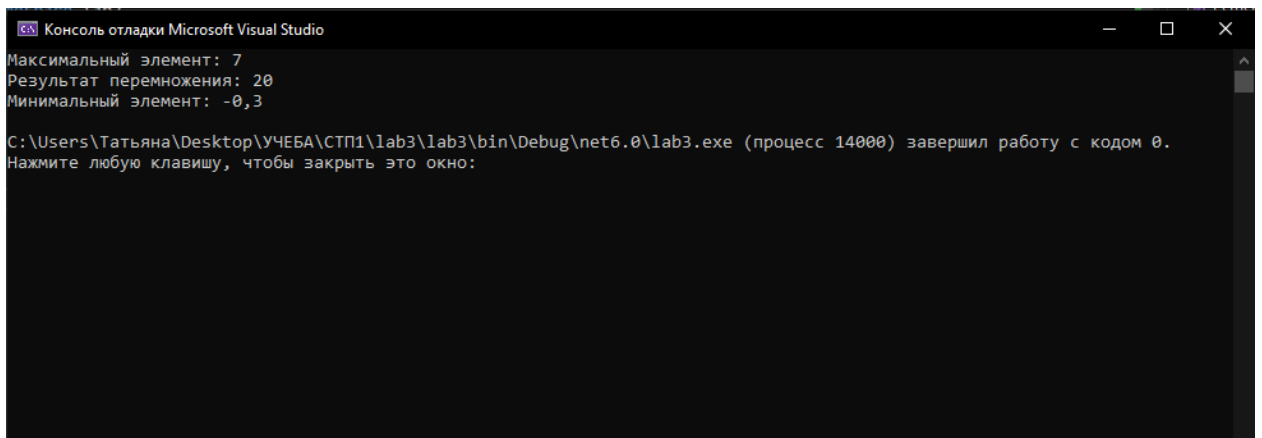
Реализация

В ходе выполнения задания был реализован класс с функциями в соответствии с заданием. Далее подробнее о каждом из реализованных методе:

`static int find Rank(int a)` – функция получает на вход три число, вычисляется минимальное число, стоящее на четных разрядах числа, читая справа налево и возвращается его индекс.

`static int left Cycle(int a, int step)` – функция получает на вход целое число и шаг сдвига влево, возвращает число с выполненным шагом.

`static int digit Insert(int a, int b, int n)` – функция получает на вход три целых числа, возвращает число, полученное путем вставки числа `b`, после разряда, заданного `n` нумеруя слева направо.



```
Консоль отладки Microsoft Visual Studio
Максимальный элемент: 7
Результат перемножения: 20
Минимальный элемент: -0,3

C:\Users\Татьяна\Desktop\УЧЕБА\СТП1\lab3\lab3\bin\Debug\net6.0\lab3.exe (процесс 14000) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно:
```

Рис. 1 – демонстрация работоспособности реализованных функций.

Так же были реализованы тесты всех методов по критерию С1 – набор тестов в совокупности должен обеспечить прохождение каждой ветви не менее одного раза.

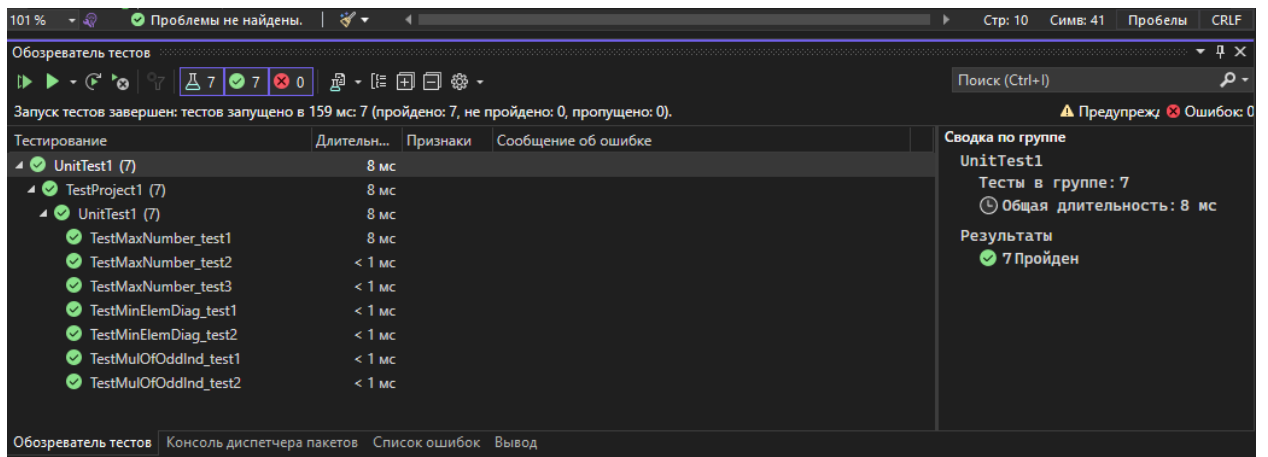


Рис. 2 – демонстрация результатов проведенного тестирования по критерию С1.

В конце НЕ были получены результаты выполненных тестов по объёму покрытия тестируемого кода, так данная проверка доступна лишь в Enterpriseверсии VisualStudio.

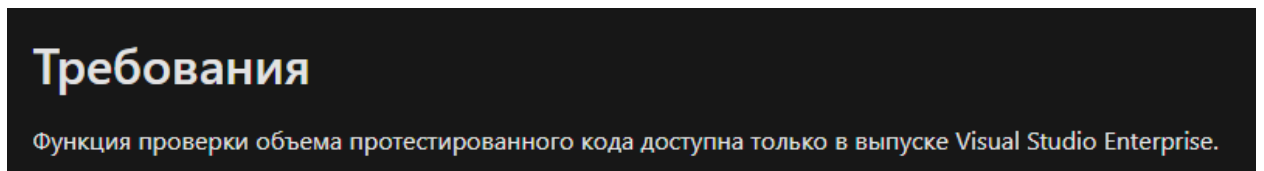


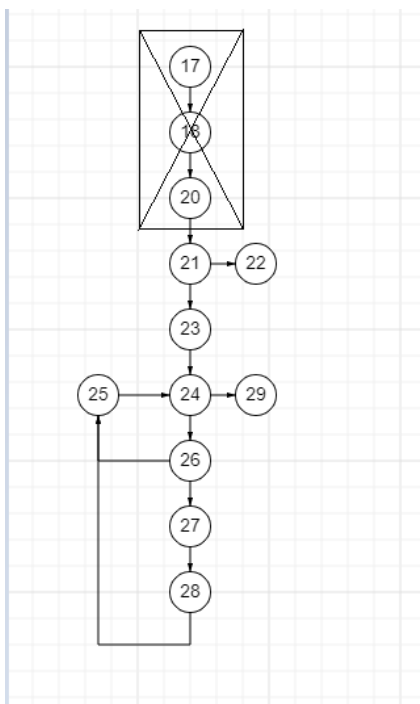
Рис.3 – доказательство, информация с официальной документации VisualStudioc сайта Microsoft.

УГП и тестовые наборы данных

Для функции `public static int findRank(int a):`

```
17. std::string numStr = std::to_string(a);
18. int min = INT_MAX;
19. int len = numStr.length();
20. size_t minIndex = 0;
21. if (len < 2)
22.     return 0;
23. for (int i = len-1;
24.      i >= 0;
25.      i -= 2) {
26.     if (numStr[i] < min) {
27.         min = numStr[i];
28.         minIndex = i;    }    }
29. return len - minIndex;
```

УГП:



Содержит следующие ветви: 17-18-20-21, 21-22, 21-23-24, 24-25, 24-26, 26-25-24, 26-27-28-25-24 и пути: 17-16-20-21-22, 17-16-20-21-23-24-29, 17-16-20-21-23-24-26-27-28-25-24-29.

Тестовые наборы, которые подойдут, по критерию C2:

1. a=1; (покроем путь 17-16-20-21-22);

2.a=12(покроем путь17-16-20-21-23-24-29);

3.a=12543;(покроем путь17-16-20-21-23-24-26-27-28-25-24-29);

Для функции *public static double leftCycle(int a, int step):*

```
36 . std::string numStr = std::to_string(a);
```

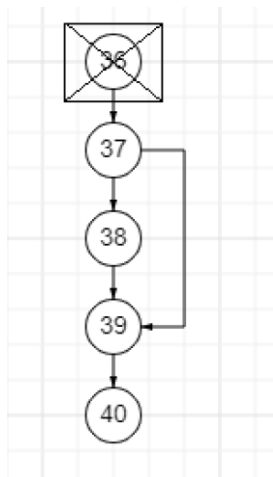
```
37 . if (step< 0)
```

```
38 . step = numStr.length() - (abs(step) % numStr.length());
```

```
39 . step %= numStr.length();
```

```
40 . return std::stoi(numStr.substr(step) + numStr.substr(0, step));
```

УГП:



Содержит следующие ветви:36-37,37-38-39-40,37-39-40 и пути 36-37-38-39-40,36-37-39-40.

Тестовый набор, который подойдет, по критерию C1:

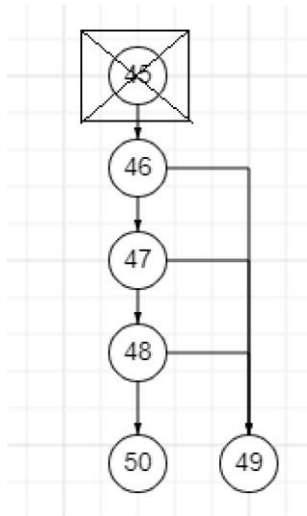
1.a=123456, step = -1;(покроем путь36-37-38-39-40);

1.a=123456, step = 1;(покроем путь36-37-39-40);

Для функции *public static intdigitInsert(int a, int b, int c):*

```
45 . std::string numStr = std::to_string(a);  
46 . if (numStr.length() < n  
47 . || n < 0  
48 . || b < 0)  
49 . return a;  
50 . return std::stoi(numStr.substr(0, n) + std::to_string(b) + numStr.substr(n));
```

УГП:



Содержит следующие ветви: 45-46,46-47,46-49,47-48, 47-49,48-50,48-50 и пути 45-46-47-48-50, 45-46-49, 45-46-47-49, 45-46-47-48-49.

Тестовый набор, который подойдет, по критерию C1:

1. a=123456, b = 6, n = 8;(покроем путь 45-46-49);
2. a=123456, b = 6, n = -8;(покроем путь 45-46-47-49);
3. a=123456, b = -6, n = 8;(покроем путь 45-46-47-48-49);
3. a=123456, b = 6, n = 2;(покроем путь 45-46-47-48-50);

Для функции *public static int SumOddDiag (double **mas, int n, intm):*

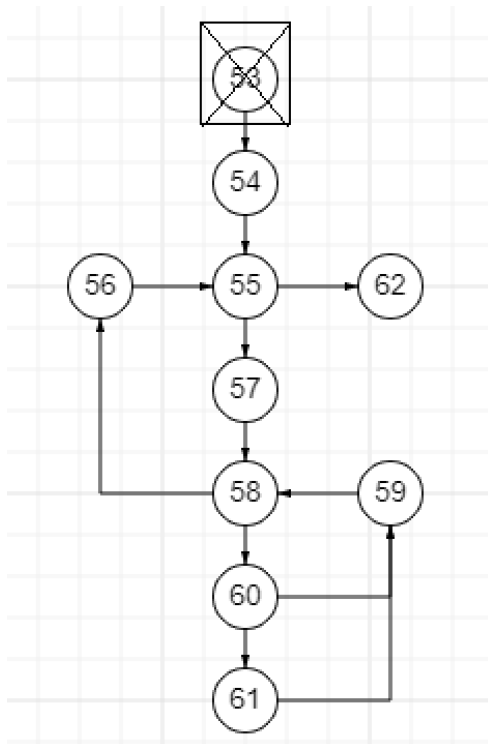
```
53 . double result = 0;  
54 . for (int i = 0;  
55 . i < n;  
56 . i++){  
57 . for (int j = m - 1;
```

```

58 .j > m - 1 - i;
59 .j--){
60 . if ((int)(mas[i][j]) % 2 == 0)
61 . result += mas[i][j]; }}
62 . return result;

```

УГП:



Содержит следующие ветви: 53-54-55, 55-57-58, 55-62, 58-60, 60-59, 60-61-59-58, 58-56-55 и пути 53-54-55-62, 53-54-55-57-58-56-55-62, 53-54-55-57-58-60-59-56-55-62, 53-54-55-57-58-60-61-59-56-55-62.

Тестовый набор, который подойдет, по критерию C1:

1. double[,] mas={{ 1.0,1.0}, {1.0,1.0}}, n = 0, m = 0;
(покроем путь 53-54-55-62);

2. double[,] mas={{ 1.0,1.0}, {1.0,1.0}}, n = 2, m = 0;
(покроем путь 53-54-55-57-58-56-55-62);

3. double[,] mas={{ 1.0,1.0, 1.0}, {{ 1.0,1.0, 1.0}, {{ 1.0,1.0, 2.0}}}, n = 5, m = 5; (покроем путь 53-54-55-57-58-60-61-59-56-55-62);

4. double[,] mas={{ 1.0,1.0, 1.0}, {{ 1.0,1.0, 1.0}, {{ 1.0,1.0, 1.0}}}, n = 5, m = 5; (покроем путь 53-54-55-57-58-60-59-56-55-62)

Вывод

Были сформированы практические навыки разработки и выполнения модульного тестирования с помощью средств автоматизации VisualStudio, разработан класс на языке C++, содержащий функции в соответствии с вариантом задания, разработаны тестовые наборы данных для тестирования функций класса, по критерию C2.

Листинг программы:

Program.cs:

```
using System;

namespace lab2
{
    public class Program
    {
        static void Main(string[] args)
        {
            int a = 7, b = 2, c = 3;
            double[,] mas1 = { { 1, 2, 3.5 }, { 4.2, 5, 6.7 } };
            double[,] mas2 = { { 1.6, 2, -2.4 }, { 0.15, 1.5, 15 }, { 0.54, -0.3, 5.42 } };
            Console.WriteLine("Максимальный элемент: " + MaxNumber(a, b, c));
            Console.WriteLine("Результат перемножения: " + MultOfOddInd(mas1));
            Console.WriteLine("Минимальный элемент: " + MinElemDiag(mas2));

        }

        public static int MaxNumber(int a, int b, int c)
        {
            if (a > b)
            {
                if (a > c)
                    return a;
                else
                    return c;
            }
            else if (b > c)
                return b;
            else
                return c;
        }

        public static double MultOfOddInd(double[,] mas)
        {
            double result = 1;
            for (int i = 0; i < mas.GetLength(0); i++)
                for (int j = 0; j < mas.GetLength(1); j++)
                    if ((i + j) % 2 == 0)
                        result *= mas[i, j];
            return result;
        }

        public static double MinElemDiag(double[,] mas)
        {
            double result = float.MaxValue;
            for (int i = 0; i < mas.GetLength(0); i++)
                for (int j = 0; j < mas.GetLength(1); j++)
                    if (i == j || i > j)
                        if (mas[i, j] < result)
                            result = mas[i, j];
            return result;
        }
    }
}
```

UnitTest1.cs:

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using lab2;
using System;

namespace TestProject1
{
    [TestClass]
    public class UnitTest1
    {
        [TestMethod]
        public void TestMaxNumber_test1()
        {
            int a = 7, b = 2, c = 3;
            int expected = 7;
            int result = Program.MaxNumber(a, b, c);
            Assert.AreEqual(expected, result);
        }

        [TestMethod]
        public void TestMaxNumber_test2()
        {
            int a = -7, b = 2, c = -3;
            int expected = 2;
            int result = Program.MaxNumber(a, b, c);
            Assert.AreEqual(expected, result);
        }

        [TestMethod]
        public void TestMaxNumber_test3()
        {
            int a = -7, b = 2, c = 3;
            int expected = 3;
            int result = Program.MaxNumber(a, b, c);
            Assert.AreEqual(expected, result);
        }

        [TestMethod]
        public void TestMulOfOddInd_test1()
        {
            double[,] mas = { { 1, 2, 3.5 }, { 4.2, 5, 6.7 } };
            double expected = 17.5;
            double result = Program.MultOfOddInd(mas);
            Assert.AreEqual(expected, result);
        }

        [TestMethod]
        public void TestMulOfOddInd_test2()
        {
            double[,] mas = { { 1, 300002, 1 }, { 300002, 1, 300002 } };
            double expected = 1;
            double result = Program.MultOfOddInd(mas);
            Assert.AreEqual(expected, result);
        }

        [TestMethod]
        public void TestMinElemDiag_test1()
        {
            double[,] mas = { { 1.6, 2, -2.4 }, { 0.15, 1.5, 15 }, { 0.54, -
0.3, 5.42 } };
            double expected = -0.3;
            double result = Program.MinElemDiag(mas);
            Assert.AreEqual(expected, result);
        }
    }
}
```

```

    }

    [TestMethod]
    public void TestMinElemDiag_test2()
    {
        double[,] mas = { { 1.6, 2, -2.4 }, { 0.15, 1.5, -150 }, { 0.54,
0.3, 5.42 } };
        double expected = 0.15;
        double result = Program.MinElemDiag(mas);
        Assert.AreEqual(expected, result);
    }
}

```