

Федеральное агентство связи
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Сибирский государственный университет
телекоммуникаций и информатики»

Лабораторная работа №3
Вариант №4
«Модульное тестирование программ на языке C++ в среде
Visual Studio»

Выполнил: студент IV курса

ИВТ, гр. ИП-713

Михеев Н.А.

Проверил: ассистент кафедры

ПМиК

Агалаков А.А.

Новосибирск, 2020 г.

Цель

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

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 Visual Studio.

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

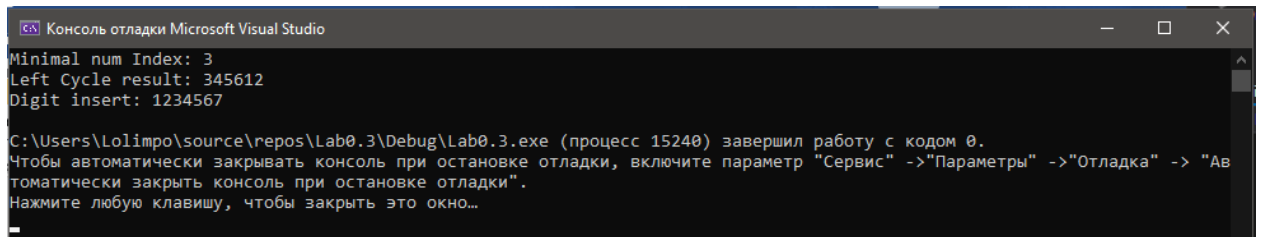
Реализация

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

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

static int digitInsert(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.

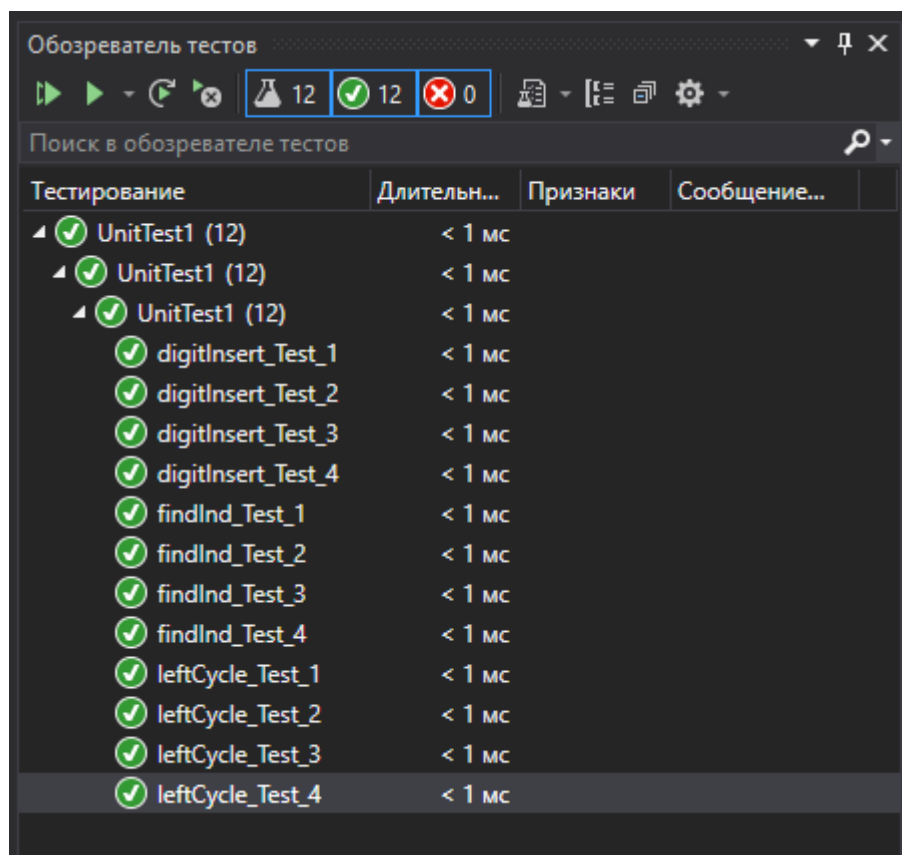


```
Консоль отладки Microsoft Visual Studio
Minimal num Index: 3
Left Cycle result: 345612
Digit insert: 1234567

C:\Users\Lolimpo\source\repos\Lab0.3\Debug\Lab0.3.exe (процесс 15240) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" -> "Параметры" -> "Отладка" -> "Автоматически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно...
```

Рис. 1 – демонстрация работоспособности реализованных функций.

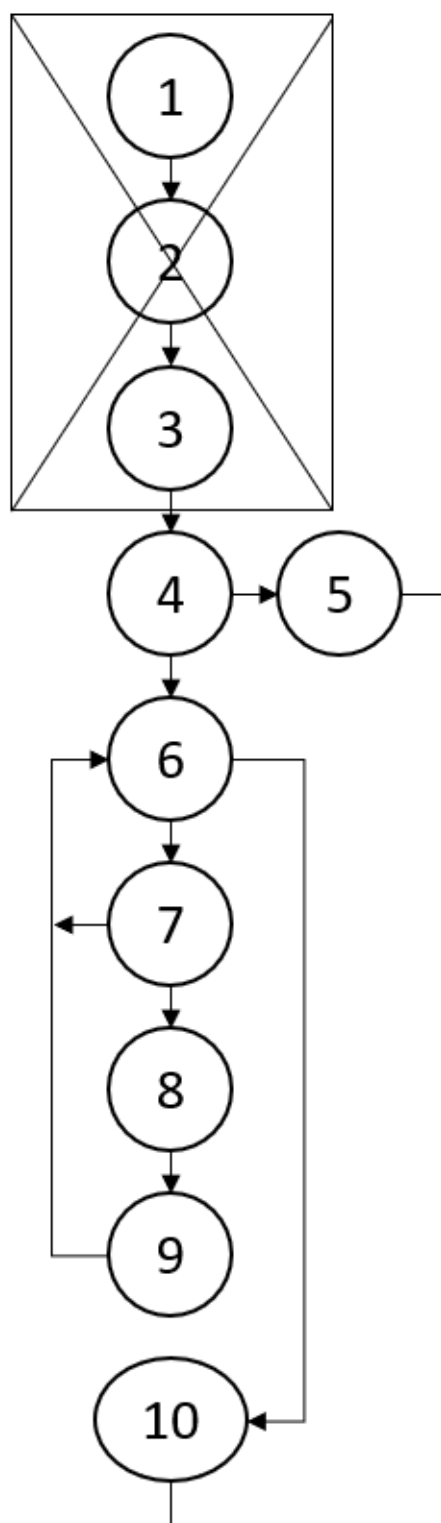
Так же были реализованы тесты всех методов по критерию С2 – набор тестов в совокупности должен обеспечить прохождение каждой ветви не менее одного раза.



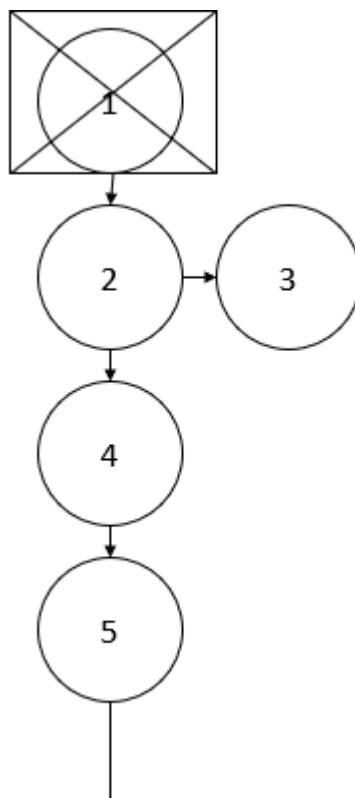
Обозреватель тестов			
Поиск в обозревателе тестов			
Тестирование	Длительн...	Признаки	Сообщение...
✔ UnitTest1 (12)	< 1 мс		
✔ UnitTest1 (12)	< 1 мс		
✔ UnitTest1 (12)	< 1 мс		
✔ digitInsert_Test_1	< 1 мс		
✔ digitInsert_Test_2	< 1 мс		
✔ digitInsert_Test_3	< 1 мс		
✔ digitInsert_Test_4	< 1 мс		
✔ findInd_Test_1	< 1 мс		
✔ findInd_Test_2	< 1 мс		
✔ findInd_Test_3	< 1 мс		
✔ findInd_Test_4	< 1 мс		
✔ leftCycle_Test_1	< 1 мс		
✔ leftCycle_Test_2	< 1 мс		
✔ leftCycle_Test_3	< 1 мс		
✔ leftCycle_Test_4	< 1 мс		

Рис. 2 – демонстрация результатов проведенного тестирования по критерию С2.

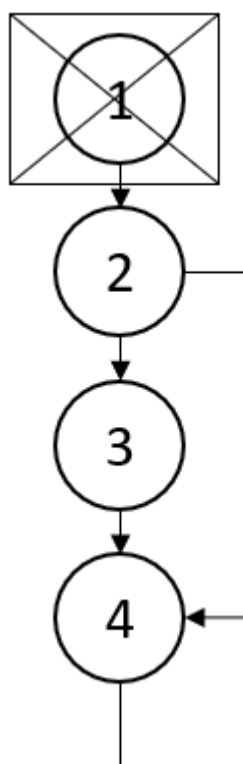
Так же были построены управляющие графы программы для всех функций.
Для static int findInd(int a):



Для static int leftCycle(int a, int step):



Для последней функции static int digitInsert(int a, int b, int n):



Вывод

Были сформированы практические навыки разработки и выполнения модульного тестирования с помощью средств автоматизации Visual Studio, 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:

Functions.h:

```
#pragma once
#include <string>
#include <stdexcept>

class Functions
{
private:
    Functions() = delete;
public:
    static int findInd(int a)
    {
        std::string numStr = std::to_string(a);
        int mini = INT_MAX;
        size_t minInd = 0;
        if (numStr.length() < 2)
            return 0;
        for (int i = numStr.length(); i >= 0; i -= 2)
        {
            if (numStr[i] < mini)
            {
                mini = numStr[i];
                minInd = i;
            }
        }
        return minInd;
    }

    static int leftCycle(int a, int step)
    {
        std::string numStr = std::to_string(a);
        step %= numStr.length();
        if (step < 0)
            step += numStr.length();
        return std::stoi(numStr.substr(step) + numStr.substr(0, step));
    }

    static int digitInsert(int a, int b, int n)
    {
        std::string numStr = std::to_string(a);
        if (numStr.length() < n || n < 0 || b < 0)
            return a;
        return std::stoi(numStr.substr(0, n) + std::to_string(b) +
numStr.substr(n));
    }

    static double sumOddDiag(double **A);
};
```

UnitTest1.cpp:

```

#include "pch.h"
#include "CppUnitTest.h"

#include "../Lab0.3/Functions.h"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTest1
{
    TEST_CLASS(UnitTest1)
    {
    public:

        TEST_METHOD(findInd_Test_1)
        {
            int expect = 5;
            int result = Functions::findInd(12543);
            Assert::AreEqual(expect, result);
        }

        TEST_METHOD(findInd_Test_2)
        {
            int expect = 0;
            int result = Functions::findInd(1);
            Assert::AreEqual(expect, result);
        }

        TEST_METHOD(findInd_Test_3)
        {
            int expect = 2;
            int result = Functions::findInd(12);
            Assert::AreEqual(expect, result);
        }

        TEST_METHOD(findInd_Test_4)
        {
            int expect = 3;
            int result = Functions::findInd(123);
            Assert::AreEqual(expect, result);
        }

        TEST_METHOD(leftCycle_Test_1)
        {
            int expect = 345612;
            int result = Functions::leftCycle(123456, 2);
            Assert::AreEqual(expect, result);
        }

        TEST_METHOD(leftCycle_Test_2)
        {
            int expect = 123456;
            int result = Functions::leftCycle(123456, 0);
            Assert::AreEqual(expect, result);
        }

        TEST_METHOD(leftCycle_Test_3)
        {
            int expect = 345612;
            int result = Functions::leftCycle(123456, -2);
            Assert::AreEqual(expect, result);
        }

        TEST_METHOD(leftCycle_Test_4)
        {
            int expect = 123456;
            int result = Functions::leftCycle(123456, 6);
            Assert::AreEqual(expect, result);
        }
    }
}

```

```

TEST_METHOD(digitInsert_Test_1)
{
    int expect = 1234567;
    int result = Functions::digitInsert(123457, 6, 5);
    Assert::AreEqual(expect, result);
}

TEST_METHOD(digitInsert_Test_2)
{
    int expect = 123457;
    int result = Functions::digitInsert(123457, 6, 8);
    Assert::AreEqual(expect, result);
}

TEST_METHOD(digitInsert_Test_3)
{
    int expect = 123457;
    int result = Functions::digitInsert(123457, 6, -1);
    Assert::AreEqual(expect, result);
}

TEST_METHOD(digitInsert_Test_4)
{
    int expect = 123457;
    int result = Functions::digitInsert(123457, -6, 5);
    Assert::AreEqual(expect, result);
}
};
}

```