Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики» (СибГУТИ)

Кафедра прикладной математики и кибернетики

Современные технологии программирования

Лабораторная работа №3

«Модульное тестирование программ на языке C++ в среде Visual Studio»

Вариант №2

Выполнил: студент 4 курса группы ИП-111

Кузьменок Денис Витальевич

Проверил преподаватель: Зайцев Михаил Георгиевич

Новосибирск, 2024 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 находит максимальное из трёх значений целых переменных.
2) Функция получает целое числа a. Формирует и возвращает целое число b из значений чётных разрядов целого числа a, следующих в обратном порядке. Например: a = 12345, b = 42.
3) Функция получает целое числа a. Находит и возвращает минимальное значение r среди разрядов целого числа a. Например, a = 62543, r = 2.
4) Функция получает двумерный массив целых переменных A. Отыскивает и возвращает сумму нечётных значений компонентов массива, лежащих ниже главной диагонали.

Реализация

В ходе выполнения лабораторной работы мною были созданы два файла: lab3.cpp, содержащий в себе точку входа в программу – функцию main – и реализацию методов класса NumberOperations. Сам класс находится в отдельном файле – Library.h. С помощью `#include "Library.h"` я подключаю библиотеку класса в исходный файл.

Содержащиеся методы:

`static int FindMax(int a, int b, int c);` - Метод принимает три целочисленных параметра. Среди них находит максимальное значение и возвращает его.

`static int ExtractEvenDigitsReversed(int a);` - Метод принимает единственное значение целого числа. Возвращает целое число, представляющее собой “склеивание” четных цифр в числе a, расположенных в обратном порядке. Предусмотрена генерация исключения `invalid_argument_1`, которое срабатывает в случае, если a – отрицательное.

`static int FindMinDigit(int a);` - Метод принимает единственное значение целого числа. Возвращает целое число, которое представляет из себя минимальное значение, найденное в числе a. Предусмотрена генерация исключения `invalid_argument_1`, которое срабатывает в случае, если a – отрицательное.

`static int SumOddBelowDiagonal(int** A, int n);` - Метод принимает два параметра: двумерный массив A и его размерность n. Возвращает сумму нечетных элементов матрицы, стоящих на позициях ниже главной диагонали. Присутствует генерация двух исключений:

1) `invalid_argument_2` – срабатывает в случае, если матрица пустая;
2) `invalid_argument_3` – срабатывает в случае, если матрица не является квадратной (т.к. главная диагональ может быть только в квадратной матрице).

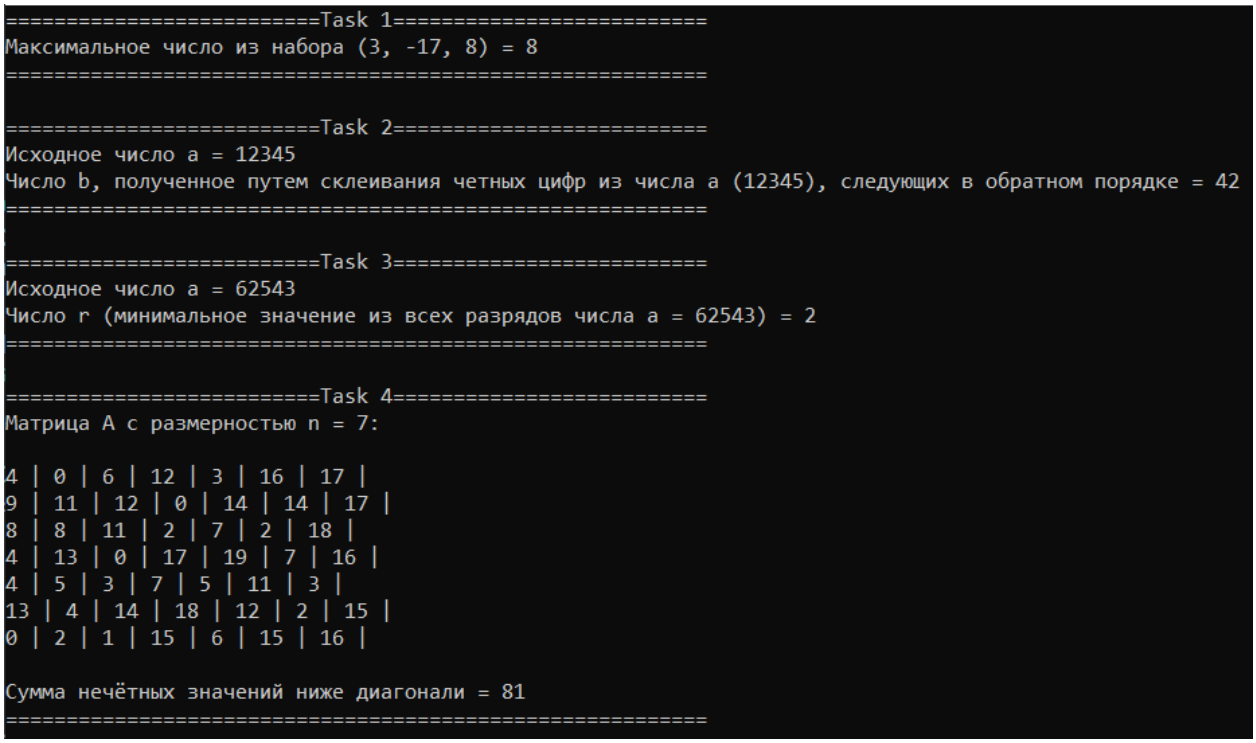
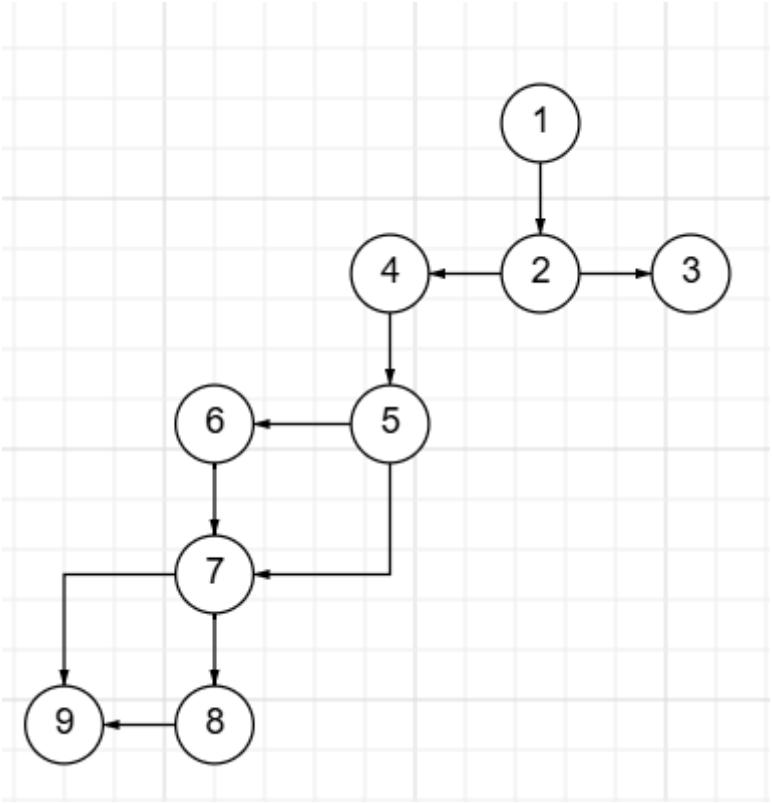


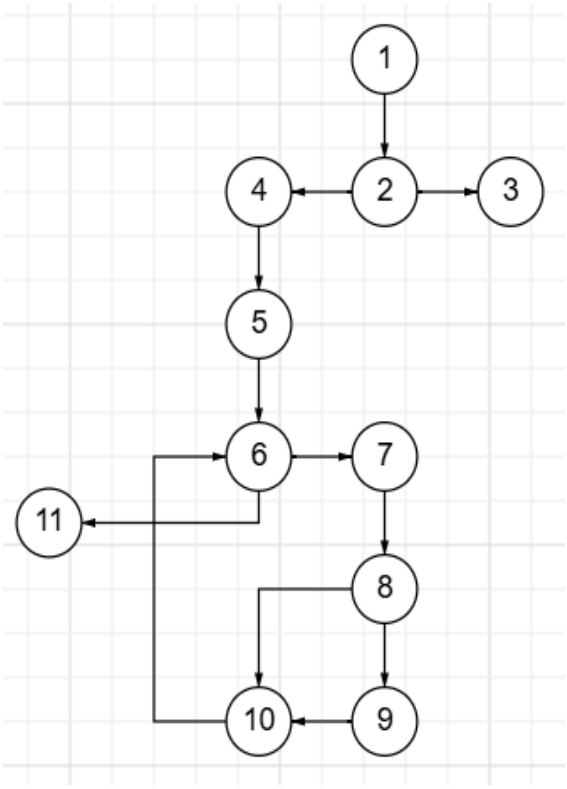
Рис. 1 – Результат запуска программы

В результате написания методов были созданы управляющие графы для каждого из них:

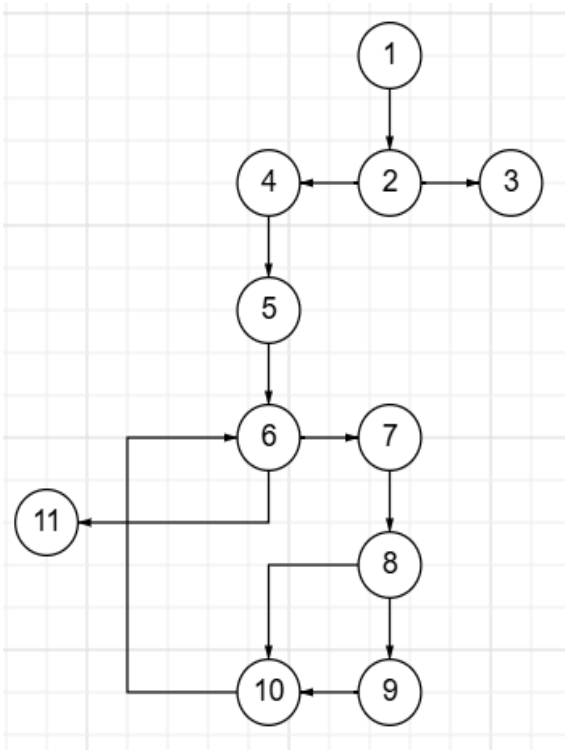
Для `static int FindMax(int a, int b, int c):`



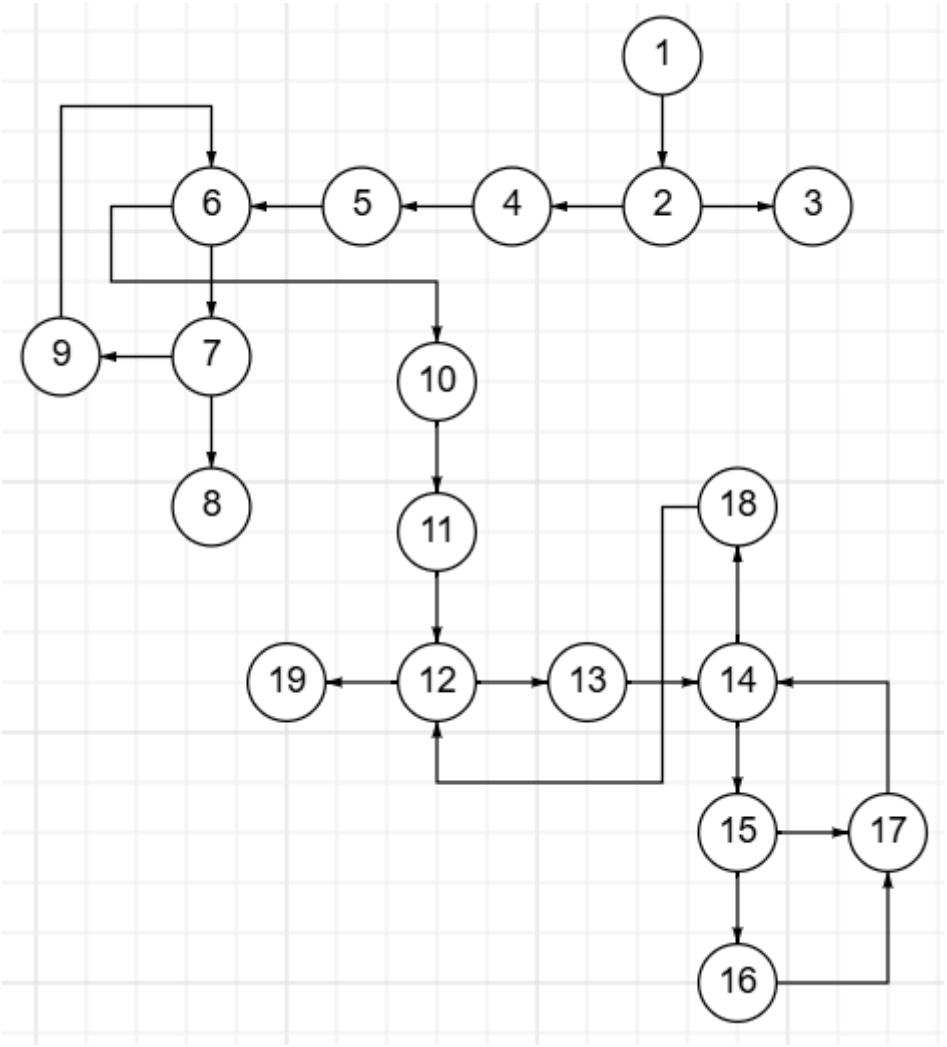
Для `static int ExtractEvenDigitsReversed(int a)`:



Для `static int FindMinDigit(int a)`:



Для `static int SumOddBelowDiagonal(int** A, int n):`



1. **FindMax**
    a) **Корректные данные:**
        - Входные данные: a = 5, b = -10, c = 2
        - Ожидаемый результат: максимальное значение – a = 5
    b) **Три одинаковых числа:**
        - Входные данные: a = 10, b = 10, c = 10
        - Ожидаемый результат: возвращение любого из трех значений (в моем случае возвращается переменная a)
2. **ExtractEvenDigitsReversed**
    a) **Корректные данные:**
        - Входные данные: a = 12345
        - Ожидаемый результат: генерация нового числа b, полученная путем “склеивания” четных цифр в числе a, записанных в обратном порядке
    b) **Число a меньше нуля:**
        - Входные данные: a = -7832
        - Ожидаемый результат: генерация исключения `invalid_argument_1`
    c) **Однозначное нечетное число:**
        - Входные данные: a = 7
        - Ожидаемый результат: возвращение b = 0
    d) **Однозначное четное число, в т.ч. 0:**
        - Входные данные: a = 6
        - Ожидаемый результат: возвращение b = a
3. **FindMinDigit**
    a) **Корректные данные:**
        - Входные данные: a = 62543
        - Ожидаемый результат: нахождение минимального цифрового значения в числе a, т.е. r = 2
    b) **Число a меньше нуля:**
        - Входные данные: -4320
        - Ожидаемый результат: генерация исключения `invalid_argument_1`
    c) **Однозначное число:**
        - Входные данные: a = 5
        - Ожидаемый результат: возвращение r = a
4. **SumOddBelowDiagonal**
    a) **Корректные данные:**
        - Входные данные: A = [[1, 2, 3], [4, 5, 6], [7, 8, 9]], n = 3
        - Ожидаемый результат: нахождение суммы нечетных элементов матрицы A, расположенных ниже главной диагонали. Sum = 7
    b) **Матрица не заполнена значениями:**
        - Входные данные: A = [[]]
        - Ожидаемый результат: генерация исключения `invalid_argument_2`
    c) **Матрица не является квадратной:**
        - Входные данные: A = [[1, 2], [3, 4], [5, 6]]
        - Ожидаемый результат: генерация исключения `invalid_argument_3`

Тестирование по структуре критериев C2 для каждого из трех методов предполагает:

1) Метод `FindMax:`
   - Можно выделить следующие пути: (1, 2, 3), (1, 2, 4, 5, 7, 9), (1, 2, 4, 5, 6, 7, 8, 9), (1, 2, 4, 5, 6, 7, 9). Необходимо будет реализовать четыре тестовых метода (X, W, Y, Z) = {(a = 5, b = 5, c = 5, “5”), (a = 31, b = 5, c = 18, “31”), (a = 2, b = 21, c = 90, “90”), (a = -65, b = 100, c = 0, “100”)}, чтобы все вышеуказанные покрыть пути.
2) Метод `ExtractEvenDigitsReversed:`
   - Можно выделить следующие пути: (1, 2, 3), (1, 2, 4, 5, 6, 11), (1, 2, 4, 5, 6, 7, 8, 9, 10, 6), (1, 2, 4, 5, 6, 7, 8, 10, 6). Необходимо будет реализовать два тестовых метода (X, Y) = {(a = 12345, “42”), (a = -90621, invalid_argument_1)}, чтобы покрыть все указанные пути.
3) Метод `FindMinDigit:`
   - Можно выделить следующие пути: (1, 2, 3), (1, 2, 4, 5, 6, 11), (1, 2, 4, 5, 6, 7, 8, 9, 10, 6), (1, 2, 4, 5, 6, 7, 8, 10, 6). Необходимо будет реализовать два тестовых метода (X, Y) = {(a = 62543, “2”), (a = -90621, invalid_argument_1)}, чтобы покрыть все указанные пути.
4) Метод `SumOddBelowDiagonal:`
   - Можно выделить следующие пути: (1, 2, 3), (1, 2, 4, 5, 6, 7, 8), (1, 2, 4, 5, 6, 7, 9, 6), (1, 2, 4, 5, 6, 10, 11, 12, 19), (1, 2, 4, 5, 6, 10, 11, 12, 13, 14, 18, 12), (1, 2, 4, 5, 6, 10, 11, 12, 13, 14, 15, 16, 17, 14), (1, 2, 4, 5, 6, 10, 11, 12, 13, 14, 15, 17, 14). Необходимо будет реализовать три тестовых метода (X, Y, Z) = {(A = [[1, 2, 3], [4, 5, 6], [7, 8, 9]], “7”), (A = [[]], invalid_argument_2), (A = [[1, 2, 3], [], [4, 5, 6]], invalid_argument_3)}, чтобы покрыть все указанные пути.

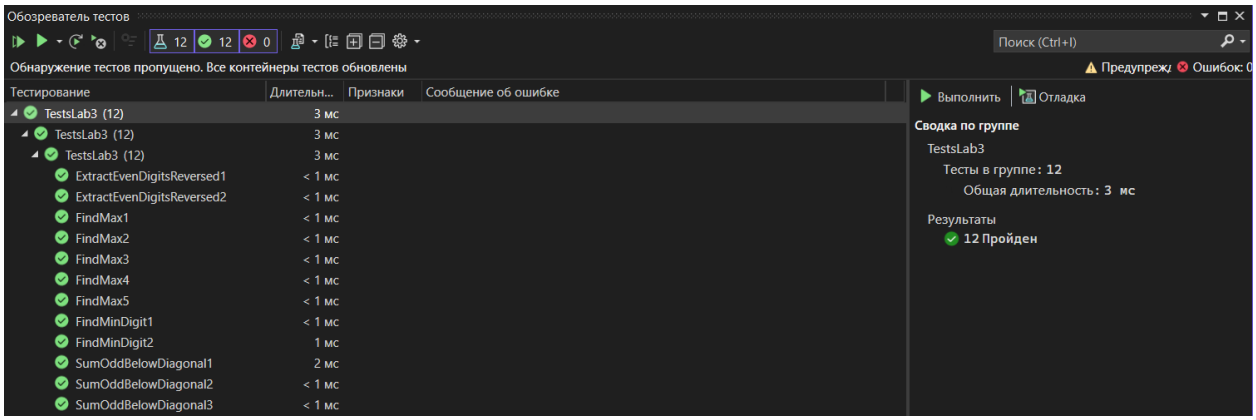


Рис. 2 – Результат выполнения модульных тестов.

Вывод:

В ходе выполнения данной работы была сформирована практическая база по разработке модульных тестов для программ на языке программирования C++ с использованием средств автоматизации Visual Studio.

В результате проведенной работы были достигнуты следующие навыки:

1) Сформированы практические навыки разработки тестов и модульного тестирования на C++.;
2) Получены знания о возможностях Visual Studio для автоматизации тестирования;
3) Разработаны тесты для заданного кода и проведено модульное тестирование на основе критериев C2.

Листинг программы:

lab3.cpp

```
#include <iostream>
#include <Windows.h>
#include <ctime>
#include <cstdlib>
#include "Library.h"
using namespace std;

int NumberOperations::FindMax(int a, int b, int c) {

    if (a == b && b == c) return a;

    int max = a;

    if (b > max) {
        max = b;
    }
    if (c > max) {
        max = c;
    }
    return max;
}

int NumberOperations::ExtractEvenDigitsReversed(int a) {
    const char* s1 = "Число должно быть неотрицательно.";
    if (a < 0) {
        throw invalid_argument_1(s1);
    }

    int b = 0;
    int digit;

    while (a > 0) {
        digit = a % 10;
        if (digit % 2 == 0) {
            b = b * 10 + digit;
        }
        a /= 10;
    }
    return b;
}

int NumberOperations::FindMinDigit(int a) {
    const char* s1 = "Число должно быть неотрицательно.";
    if (a < 0) {
        throw invalid_argument_1(s1);
    }
    int min = 9;
    int digit;
    while (a > 0) {
        digit = a % 10;
        if (digit < min) {
            min = digit;
        }
        a /= 10;
    }
    return min;
}

int NumberOperations::SumOddBelowDiagonal(int** A, int n) {
    const char* s2 = "Матрица не может быть пустым.";
    const char* s4 = "Матрица должен быть квадратным.";
```

```
    if (A == nullptr || n == 0) {
        throw invalid_argument_2(s2);
    }

    for (int i = 1; i < n; i++) {
        if (A[i] == nullptr) {
            throw invalid_argument_3(s4);
        }
    }

    long sum = 0;

    for (int i = 1; i < n; i++) {
        for (int j = 0; j < i; j++) {
            if (A[i][j] % 2 != 0) {
                sum += A[i][j];
            }
        }
    }
    return sum;
}

int main()
{

    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    NumberOperations numberOperations = NumberOperations();
    int numbers[] = {3, -17, 8};
    int task1 = numberOperations.FindMax(numbers[0], numbers[1], numbers[2]);
    cout << "==========================Task 1==========================" << endl;
    cout << "Максимальное число из набора (" << numbers[0] << ", " << numbers[1] <<
", " << numbers[2] << ")" << " = " << task1 << endl;
    cout << "==========================================================" << endl <<
endl;

    int a = 12345;
    int task2 = numberOperations.ExtractEvenDigitsReversed(a);
    cout << "==========================Task 2==========================" << endl;
    cout << "Исходное число a = " << a << endl;
    cout << "Число b, полученное путем склеивания четных цифр из числа a (" << a <<
"), следующих в обратном порядке = " << task2 << endl;
    cout << "==========================================================" << endl <<
endl;

    int r = 62543;
    int task3 = numberOperations.FindMinDigit(r);
    cout << "==========================Task 3==========================" << endl;
    cout << "Исходное число a = " << r << endl;
    cout << "Число r (минимальное значение из всех разрядов числа a = " << r << ") =
" << task3 << endl;
    cout << "==========================================================" << endl <<
endl;

    srand(time(0));
    int n = rand() % 5 + 3;
    int** A = new int* [n];
    for (int i = 0; i < n; i++) {
        A[i] = new int[n];
        for (int j = 0; j < n; j++) {
            A[i][j] = rand() % 20;
        }
    }
```

```
    cout << "=========================Task 4=========================" << endl;
    cout << "Матрица A с размерностью n = " << n << ": " << endl << endl;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cout << A[i][j] << " | ";
        }
        cout << endl;
    }

    int task4 = numberOperations.SumOddBelowDiagonal(A, n);

    cout << endl << "Сумма нечётных значений ниже диагонали = " << task4 << endl;
    cout << "========================================================" << endl;
    return 0;
}
```

Library.h

```
#pragma once
#include <stdexcept>
class NumberOperations {

public:

	class invalid_argument_1 : public std::invalid_argument {
	public:
		invalid_argument_1(const char* s) : invalid_argument(s) { }
	};

	class invalid_argument_2 : public std::invalid_argument {
	public:
		invalid_argument_2(const char* s) : invalid_argument(s) { }
	};

	class invalid_argument_3 : public std::invalid_argument {
	public:
		invalid_argument_3(const char* s) : invalid_argument(s) { }
	};

	class invalid_argument_4 : public std::invalid_argument {
	public:
		invalid_argument_4(const char* s) : invalid_argument(s) { }
	};

	static int FindMax(int a, int b, int c);
	static int ExtractEvenDigitsReversed(int a);
	static int FindMinDigit(int a);
	static int SumOddBelowDiagonal(int** A, int n);
};
```

TestsLab3.cpp

```
#include "pch.h"
#include "CppUnitTest.h"
#include "../lab3/Library.h"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace TestsLab3
{
	TEST_CLASS(TestsLab3)
	{
	public:

		TEST_METHOD(FindMax1)
		{
			NumberOperations numberOperations;
			//arrange
			int first = 52;
			int second = 0;
			int third = -87;
			int expected = 52;
			//act
			int actual = numberOperations.FindMax(first, second, third);
			//assert
			Assert::AreEqual(expected, actual);
		}

		TEST_METHOD(FindMax2)
		{
			NumberOperations numberOperations;
			//arrange
			int first = 0;
			int second = 52;
			int third = -87;
			int expected = 52;
			//act
			int actual = numberOperations.FindMax(first, second, third);
			//assert
			Assert::AreEqual(expected, actual);
		}

		TEST_METHOD(FindMax3)
		{
			NumberOperations numberOperations;
			//arrange
			int first = 0;
			int second = -87;
			int third = 52;
			int expected = 52;
			//act
			int actual = numberOperations.FindMax(first, second, third);
			//assert
			Assert::AreEqual(expected, actual);
		}

		TEST_METHOD(FindMax4)
		{
			NumberOperations numberOperations;
			//arrange
			int first = 0;
			int second = 0;
			int third = 0;
			int expected = 0;
			//act
```

```
		int actual = numberOperations.FindMax(first, second, third);
		//assert
		Assert::AreEqual(expected, actual);
	}

	TEST_METHOD(FindMax5)
	{
		NumberOperations numberOperations;
		//arrange
		int first = 52;
		int second = 52;
		int third = 52;
		int expected = 52;
		//act
		int actual = numberOperations.FindMax(first, second, third);
		//assert
		Assert::AreEqual(expected, actual);
	}

	TEST_METHOD(ExtractEvenDigitsReversed1)
	{
		//arrange
		int a = -9467;
		//act
		auto func = [a] {
			NumberOperations numberOperations;
			numberOperations.ExtractEvenDigitsReversed(a);
			};
		//assert

Assert::ExpectException<NumberOperations::invalid_argument_1>(func);
	}

	TEST_METHOD(ExtractEvenDigitsReversed2)
	{
		//arrange
		int a = 12345;
		int expected = 42;
		//act
		int actual = NumberOperations::ExtractEvenDigitsReversed(a);
		//assert
		Assert::AreEqual(expected, actual);
	}

	TEST_METHOD(FindMinDigit1)
	{
		//arrange
		int a = 62543;
		int expected = 2;
		//act
		int actual = NumberOperations::FindMinDigit(a);
		//assert
		Assert::AreEqual(expected, actual);
	}

	TEST_METHOD(FindMinDigit2)
	{
		//arrange
		int a = -62543;
		//act
		auto func = [a] {
			NumberOperations numberOperations;
			numberOperations.FindMinDigit(a);
			};
		//assert
```

```
Assert::ExpectException<NumberOperations::invalid_argument_1>(func);
	}

	TEST_METHOD(SumOddBelowDiagonal1)
	{
		//arrange
		int n = 3;
		int** a = new int* [n];
		a[0] = new int[n] {1, 2, 3};
		a[1] = new int[n] {4, 5, 6};
		a[2] = new int[n] {7, 8, 9};
		int expected = 7;
		//act
		int actual = NumberOperations::SumOddBelowDiagonal(a, n);
		//assert
		Assert::AreEqual(expected, actual);
		delete[] a[0];
		delete[] a[1];
		delete[] a[2];
		delete[] a;
	}

	TEST_METHOD(SumOddBelowDiagonal2)
	{
		//arrange
		int n = 0;
		int** a = new int*[3];
		a[0] = new int[n];
		a[1] = new int[n];
		a[2] = new int[n];
		//act
		auto func = [a] {
			NumberOperations numberOperations;
			int size = 0;
			numberOperations.SumOddBelowDiagonal(a, size);
			};
		//assert

Assert::ExpectException<NumberOperations::invalid_argument_2>(func);
		delete[] a[0];
		delete[] a[1];
		delete[] a[2];
		delete[] a;
	}

	TEST_METHOD(SumOddBelowDiagonal3)
	{
		//arrange
		int n = 3;
		int** a = new int* [n];
		a[0] = new int[n] {1, 2, 3};
		a[1] = nullptr;
		a[2] = new int[n] {4, 5, 6};
		//act
		auto func = [a] {
			NumberOperations numberOperations;
			numberOperations.SumOddBelowDiagonal(a, 3);
			};
		//assert

Assert::ExpectException<NumberOperations::invalid_argument_3>(func);
		delete[] a[0];
		delete[] a[1];
		delete[] a[2];
```

```
                    delete[] a;
                }
        };
}
```