Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет

телекоммуникаций и информатики»

Кафедра ПМиК

Современные технологии программирования 1

Лабораторная работа №3

Модульное тестирование программ на языке С++ в среде Visual Studio

Вариант №2

Выполнил: студент 4 курса

Ф. ИВТ, группа: ИП-711

Мартасов И. О.

Проверил: доцент кафедры ПМиК

Зайцев Михаил Георгиевич

Новосибирск, 2020

**Содержание**

1. Задание.
2. УГП и тестовые наборы данных для тестирования функций класса.
3. Исходные тексты программ на языке C#
4. Результаты выполнения модульных тестов.
5. Результаты покрытия разработанного кода тестами.
6. Выводы по выполненной работе.

**Задание**

Разработайте на языке С++ класс, содержащий набор функций в соответствии с вариантом задания.

Разработайте тестовые наборы данных по критерию С2 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.

**Вариант № 2**

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.
3. Функция получает целое числа a. Находит и возвращает минимальное значение r среди разрядов целого числа a.
4. Функция получает двумерный массив целых переменных A. Отыскивает и возвращает сумму нечётных значений компонентов массива, лежащих ниже главной диагонали.

**УГП и тестовые наборы данных для тестирования функций класса**

**Функция 1 (max\_of\_three\_numbers):**

**(x, y) = {(x = 3, y = 5, z = 7, m = 7), (x = 5, y = 3, m = 5)}**

**Функция 2 (max\_index):**

**(a) = {(a = 12345, b = 42), (a = 0, b = 0)}**

**Функция 3 (min\_index):**

**(a) = {(a = 62345, b = 2), (a = 0, b = 0)}**

**Функция 4 (diag):**

**(mas, n, m) = {(1.0, 2.0, 3.0, 4.0 ,5.0), {(2.0, 3.0, 4.0, 5.0 ,6.0) , {(3.0, 4.0, 5.0, 6.0 ,7.0) , {(4.0, 5.0, 6.0, 7.0 ,8.0) , {(5.0, 6.0, 7.0, 8.0 ,9.0) вх. массив}, n= 5, m=5, sum = 24), (n = 0, m = 0, sum = 0), (n = 5, m = 0, sum = 0), (n = 0, m = 5, sum = 0)}**

**Исходные тексты программ на языке C#**

**MyClass.cpp**

#include "pch.h"

#include "MyClass.h"

int MyClass::max\_of\_three\_numbers(int x, int y, int z)

{

if (x > y && x > z)

{

return x;

}

else if (y > z)

{

return y;

}

else

{

return z;

}

}

int MyClass::max\_index(int a)

{

int b = 0;

while (0 != a)

{

if (0 == ((a % 10) % 2))

{

b \*= 10;

b += (a % 10);

}

a /= 10;

}

return b;

}

int MyClass::min\_index(int a)

{

int r = 9;

while (0 != a)

{

if ((a % 10) < r)

{

r = a % 10;

}

a /= 10;

}

return r;

}

int MyClass::diag(int\*\* a, int n, int m)

{

int sum = 0;

for (int i = 0; i < n; i++)

{

for (int j = 0; j < m; j++)

{

if ((j < i) && (1 == (a[i][j] % 2)))

{

sum += a[i][j];

}

}

}

return sum;

}

**MyClass.h**

#pragma once

class MyClass

{

public:

static int max\_of\_three\_numbers(int x, int y, int z);

static int max\_index(int a);

static int min\_index(int a);

static int diag(int\*\* a, int n, int m);

};

**TestSTP-Lab3.cpp**

#include "pch.h"

#include "CppUnitTest.h"

#include "../STP-Lab3/MyClass.h"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace TestSTPLab3

{

TEST\_CLASS(TestSTPLab3)

{

public:

TEST\_METHOD(max\_of\_three\_numbers\_3\_5\_7\_expected\_7)

{

int x = 3;

int y = 5;

int z = 7;

int expected = 7;

int actual = MyClass::max\_of\_three\_numbers(x, y, z);

Assert::AreEqual(expected, actual);

}

TEST\_METHOD(max\_of\_three\_numbers\_5\_7\_3\_expected\_7)

{

int x = 5;

int y = 7;

int z = 3;

int expected = 7;

int actual = MyClass::max\_of\_three\_numbers(x, y, z);

Assert::AreEqual(expected, actual);

}

TEST\_METHOD(max\_of\_three\_numbers\_7\_3\_5\_expected\_7)

{

int x = 7;

int y = 3;

int z = 5;

int expected = 7;

int actual = MyClass::max\_of\_three\_numbers(x, y, z);

Assert::AreEqual(expected, actual);

}

TEST\_METHOD(max\_index\_12345\_expected\_42)

{

int expected = 42;

int actual = MyClass::max\_index(12345);

Assert::AreEqual(expected, actual);

}

TEST\_METHOD(max\_index\_0\_expected\_0)

{

int expected = 0;

int actual = MyClass::max\_index(0);

Assert::AreEqual(expected, actual);

}

TEST\_METHOD(min\_index\_62432\_expected\_2)

{

int expected = 2;

int actual = MyClass::min\_index(62432);

Assert::AreEqual(expected, actual);

}

TEST\_METHOD(min\_index\_1110101010\_expected\_0)

{

int expected = 0;

int actual = MyClass::max\_index(1110101010);

Assert::AreEqual(expected, actual);

}

TEST\_METHOD(min\_index\_0\_expected\_0)

{

int expected = 0;

int actual = MyClass::max\_index(0);

Assert::AreEqual(expected, actual);

}

TEST\_METHOD(diag\_5\_5\_expected\_24)

{

int n = 5;

int m = 5;

int\*\* a = new int\* [n];

for (int i = 0; i < n; i++)

{

a[i] = new int[m];

}

for (int i = 0; i < n; i++)

{

for (int j = 0; j < m; j++)

{

a[i][j] = i + j;

}

}

int expected = 24;

int actual = MyClass::diag(a, n, m);

Assert::AreEqual(expected, actual);

}

TEST\_METHOD(diag\_0\_0\_expected\_0)

{

int n = 0;

int m = 0;

int\*\* a = new int\* [n];

for (int i = 0; i < n; i++)

{

a[i] = new int[m];

}

for (int i = 0; i < n; i++)

{

for (int j = 0; j < m; j++)

{

a[i][j] = i + j;

}

}

int expected = 0;

int actual = MyClass::diag(a, n, m);

Assert::AreEqual(expected, actual);

}

TEST\_METHOD(diag\_5\_0\_expected\_0)

{

int n = 5;

int m = 0;

int\*\* a = new int\* [n];

for (int i = 0; i < n; i++)

{

a[i] = new int[m];

}

for (int i = 0; i < n; i++)

{

for (int j = 0; j < m; j++)

{

a[i][j] = i + j;

}

}

int expected = 0;

int actual = MyClass::diag(a, n, m);

Assert::AreEqual(expected, actual);

}

TEST\_METHOD(diag\_0\_5\_expected\_0)

{

int n = 0;

int m = 5;

int\*\* a = new int\* [n];

for (int i = 0; i < n; i++)

{

a[i] = new int[m];

}

for (int i = 0; i < n; i++)

{

for (int j = 0; j < m; j++)

{

a[i][j] = i + j;

}

}

int expected = 0;

int actual = MyClass::diag(a, n, m);

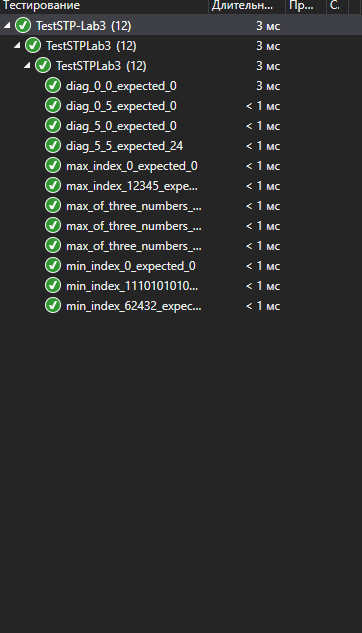
Assert::AreEqual(expected, actual);

}

};

}

**Результаты выполнения модульных тестов и результаты покрытия разработанного кода тестам**



**Выводы по выполненной работе**

Были сформированы практические навыки разработки тестов и модульного тестирования на языке С++ с помощью средств автоматизации Visual Studio.