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**

**«Модульное тестирование программ на языке**

**С++ в среде Visual Studio»**

**Вариант №4**

Выполнил: студент 4 курса

ИВТ, гр. ИП-113

Шпилев Д. И.

Проверил: старший преподаватель кафедры ПМиК

Агалаков А.А.

Новосибирск, 2024 г.

**Цель**

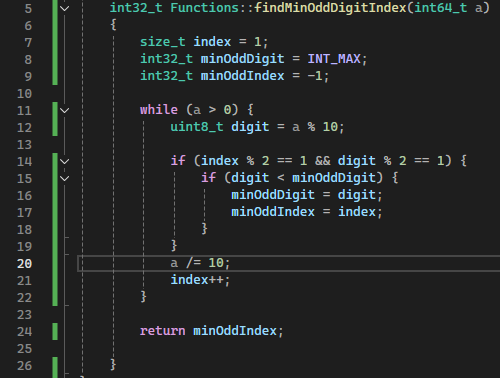
Сформировать практические навыки разработки тестов и модульного тестирования на языке С++ с помощью средств автоматизации Visual Studio

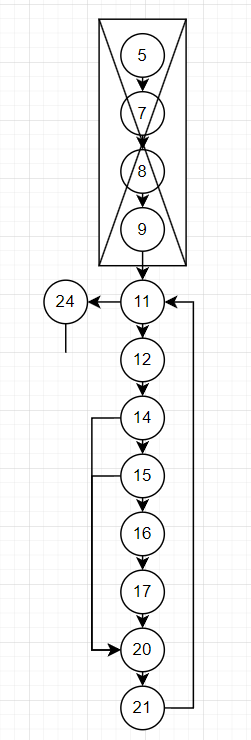
**Задание**

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.

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, а = 12543, r = 3.
2. Функция получает целое числа a. Возвращает число, полученное циклическим сдвигом значений разрядов целого числа а 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

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

****

****

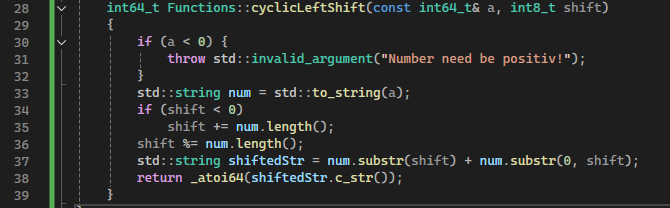
Пути:

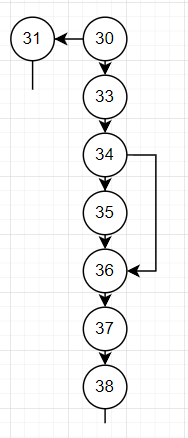
(5-7-8-9-11-24) a = 0

(5-7-8-9-11-12-14-20-21-24) a = 2

(5-7-8-9-11-12-14-15-20-21-24) a = 321

(5-7-8-9-11-12-14-15-16-17-20-21-24) a = 12543

****

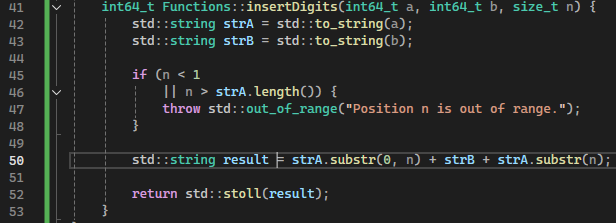
****

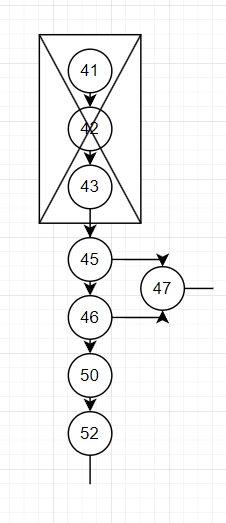
Пути:

(30-31) a = -1

(30-33-34-36-37-38) a = 12345 shift = 3

(30-33-34-35-36-37-38) a = 12345 shift = -2



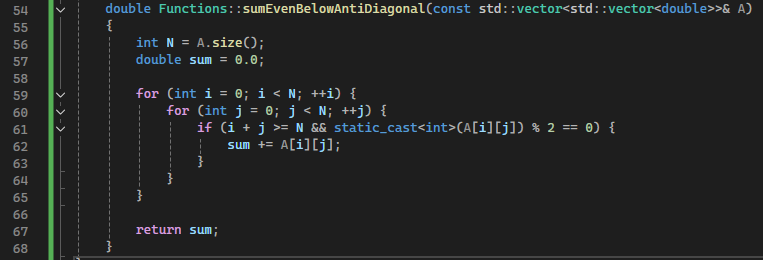


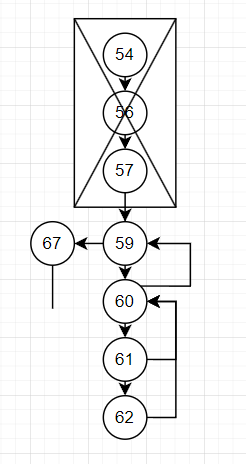
Пути:

(41-42-43-45-47) n = -1

(41-42-43-45-46-47) a = 12345 n = 7

(41-42-43-45-46-50-52) a = 123457 b = 6 n = 5





Пути:

(59-67) A = {{}}

(59-60-61-67) A = {{1.0, 1.0}, {1.0, 1.0}}

(59-60-61-62-67) A = {{1.0, 2.0, 3.0},{4.0, 6.0, 8.0},{10.0, 12.0, 14.0}}

**Листинг программы:**

**Functions.h:**

#pragma once

#include <string>

#include <stdexcept>

#include <vector>

namespace Lab3 {

class Functions {

Functions() = delete;

Functions(const Functions& other) = delete;

Functions(Functions&& other) = delete;

public:

static int32\_t findMinOddDigitIndex(int64\_t a);

static int64\_t cyclicLeftShift(const int64\_t& a, int16\_t shift);

static int64\_t insertDigits(int64\_t a, int64\_t b, int16\_t n);

static double sumEvenBelowAntiDiagonal(const std::vector<std::vector<double>>& A);

};

}

**Functions.cpp:**

#include "Functions.h"

namespace Lab3 {

int32\_t Functions::findMinOddDigitIndex(int64\_t a)

{

int32\_t index = 1;

int32\_t minOddDigit = INT\_MAX;

int32\_t minOddIndex = -1;

while (a > 0) {

uint16\_t digit = a % 10;

if (index % 2 == 1 && digit % 2 == 1) {

if (digit < minOddDigit) {

minOddDigit = digit;

minOddIndex = index;

}

}

a /= 10;

index++;

}

return minOddIndex;

}

int64\_t Functions::cyclicLeftShift(const int64\_t& a, int16\_t shift)

{

if (a < 0) {

throw std::invalid\_argument("Number need be positiv!");

}

std::string num = std::to\_string(a);

if (shift < 0)

shift += num.length();

shift %= num.length();

std::string shiftedStr = num.substr(shift) + num.substr(0, shift);

return std::stoll(shiftedStr);

}

int64\_t Functions::insertDigits(int64\_t a, int64\_t b, int16\_t n) {

std::string strA = std::to\_string(a);

std::string strB = std::to\_string(b);

if (n < 1 || n > strA.length()) {

throw std::out\_of\_range("Position n is out of range");

}

std::string result = strA.substr(0, n) + strB + strA.substr(n);

return std::stoll(result);

}

double Functions::sumEvenBelowAntiDiagonal(const std::vector<std::vector<double>>& A)

{

int N = A.size();

double sum = 0.0;

for (int i = 0; i < N; ++i) {

for (int j = 0; j < N; ++j) {

if (i + j >= N && static\_cast<int>(A[i][j]) % 2 == 0) {

sum += A[i][j];

}

}

}

return sum;

}

}

**UnitTest1.cpp:**

#include "CppUnitTest.h"

#include "../Lab3/Functions.h"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTest1

{

TEST\_CLASS(TestFindMinOdd)

{

public:

TEST\_METHOD(FindMinOddDigitInStart)

{

int32\_t result = Lab3::Functions::findMinOddDigitIndex(12543);

Assert::AreEqual(5, result);

}

TEST\_METHOD(ZeroMinOdd)

{

int32\_t result = Lab3::Functions::findMinOddDigitIndex(0);

Assert::AreEqual(-1, result);

}

TEST\_METHOD(OneEvenDigits)

{

int32\_t result = Lab3::Functions::findMinOddDigitIndex(2);

Assert::AreEqual(-1, result);

}

TEST\_METHOD(FindFirst)

{

int32\_t result = Lab3::Functions::findMinOddDigitIndex(321);

Assert::AreEqual(1, result);

}

};

TEST\_CLASS(TestShift)

{

public:

TEST\_METHOD(NegativNumber)

{

Assert::ExpectException<std::invalid\_argument>([] {Lab3::Functions::cyclicLeftShift(-1236, 1); }, L"Number need be positiv!");

}

TEST\_METHOD(StandartShift)

{

int64\_t a = 12345;

int8\_t shift = 3;

int64\_t result = Lab3::Functions::cyclicLeftShift(a, shift);

Assert::AreEqual(45123LL, result);

}

TEST\_METHOD(NegativShift)

{

int64\_t a = 12345;

int8\_t shift = -2;

int64\_t result = Lab3::Functions::cyclicLeftShift(a, shift);

Assert::AreEqual(45123LL, result);

}

};

TEST\_CLASS(TestInsert)

{

public:

TEST\_METHOD(NegativN)

{

Assert::ExpectException<std::out\_of\_range>([] {Lab3::Functions::insertDigits(12345, 1, -1); }, L"Position n is out of range");

}

TEST\_METHOD(OverN)

{

Assert::ExpectException<std::out\_of\_range>([] {Lab3::Functions::insertDigits(12345, 1, 6); }, L"Position n is out of range");

}

TEST\_METHOD(Insert)

{

int64\_t a = 123457;

int64\_t b = 6;

int8\_t n = 6;

int64\_t result = Lab3::Functions::insertDigits(123457, 6, 5);

Assert::AreEqual(1234567LL, result);

}

};

TEST\_CLASS(TestMatrix)

{

public:

TEST\_METHOD(ZeroElementSMatrix)

{

std::vector<std::vector<double>> A = { {} };

double result = Lab3::Functions::sumEvenBelowAntiDiagonal(A);

Assert::AreEqual(0.0, result);

}

TEST\_METHOD(MultipleElementMatrix)

{

std::vector<std::vector<double>> A = {

{1.0, 2.0, 3.0},

{4.0, 6.0, 8.0},

{10.0, 12.0, 14.0}

};

double result = Lab3::Functions::sumEvenBelowAntiDiagonal(A);

Assert::AreEqual(34.0, result);

}

TEST\_METHOD(NoEvenBelowAntiDiagonal)

{

std::vector<std::vector<double>> A = {

{1.1, 3.3},

{5.5, 7.7}

};

double result = Lab3::Functions::sumEvenBelowAntiDiagonal(A);

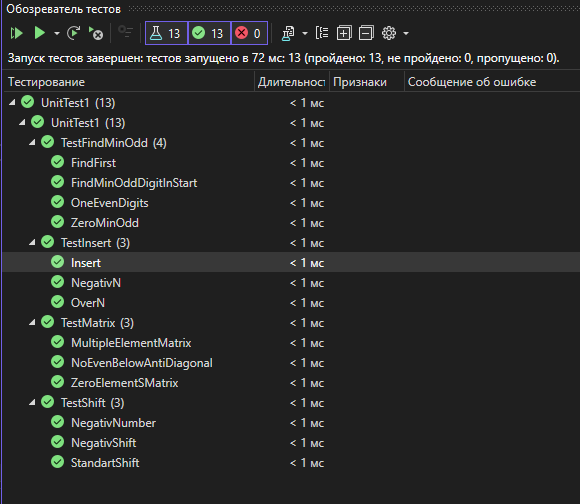
Assert::AreEqual(0.0, result);

}

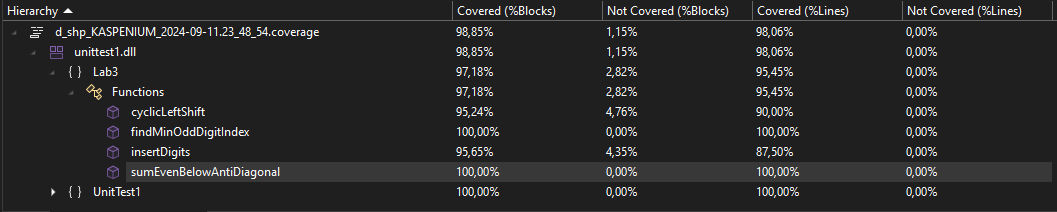
};

}

**Результаты выполнения модульных тестов**

****

**Результаты покрытия разработанного кода тестами.**



**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 С2.