# Задание

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

Разработайте тестовые наборы данных по критерию С2 для тестирования функций класса.

Протестируйте функции с помощью средств автоматизации модульного тестирования Visual Studio.

Провести анализ выполненного теста и, если необходимо отладку кода.

Напишите отчёт о результатах проделанной работы.

## Вариант

1. Функция получает целое числа a. Формирует и возвращает целое число b из значений нечётных разрядов целого числа a, следующих в обратном порядке. Например: a = 12345, b = 531.
2. Функция получает целое числа a. Находит и возвращает номер разряда, в котором находится максимальное значение r среди чётных разрядов целого числа a с чётным значением. Разряды числа, пронумерованы справа налево, начиная с единицы. Например, а = 62543, r = 4.
3. Функция получает целое числа a. Возвращает число, полученное циклическим сдвигом значений разрядов целого числа а на заданное число позиций вправо. Например, сдвиг на две позиции: Исходное число: 123456 Результат: 561234.
4. Функция получает двумерный массив вещественных переменных A. Отыскивает и возвращает сумму чётных значений компонентов массива, лежащих выше побочной диагонали.

# УГП и тестовые наборы данных для тестирования функций класса

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер функции** | **УГП** | **Тестовый набор** | |
|  | **Y** |
| 1 |  |  |  |
|  |  |
| 2 |  |  |  |
|  |  |
|  |  |
| 3 |  |  |  |
|  |  |
|  |  |
|  |  |
| 4 |  |  |  |
|  |  |
|  |  |
|  |  |

# Исходные тексты программ на языке C#

Тестируемые функции:

Class.h:

#pragma once

class Class

{

public:

// Функция получает целое числа a.

// Формирует и возвращает целое число b из значений нечётных разрядов целого числа a, следующих в

// обратном порядке.Например: a = 12345, b = 531.

static int getOddRankAndReverse(int a);

//Функция получает целое числа a.Находит и возвращает номер

//разряда, в котором находится максимальное значение r среди чётных

//разрядов целого числа a с чётным значением.Разряды числа,

//пронумерованы справа налево, начиная с единицы.Например, а =

//62543, r = 4.

static int getEvenRankWithMaxEvenValue(int a);

//Функция получает целое числа a.Возвращает число, полученное

//циклическим сдвигом значений разрядов целого числа а на заданное

//число позиций вправо.Например, сдвиг на две позиции :

//Исходное число : 123456

//Результат : 561234

static int getCycleRightShiftNumber(int a, int i);

//Функция получает двумерный массив вещественных переменных A.

//Отыскивает и возвращает сумму чётных значений компонентов

//массива, лежащих выше побочной диагонали

static double sumEvenNumbers(double\*\* A, int size);

};

Class.cpp:

#include "Class.h"

int Class::getOddRankAndReverse(int a)

{

int b = 0;

1 for (;

2 a != 0;

3 a /= 100)

4 b = b \* 10 + (a % 10);

5 return b;

}

int Class::getEvenRankWithMaxEvenValue(int a)

{

int maxR = 0;

int maxValue = 0;

1 a = a / 10;

2 for (int r = a;

3 a != 0;

4 a /= 100,

5 r += 2) {

6 int value = a % 10;

7 if (value % 2 == 0 && maxValue < value) {

8 maxR = r;

9 maxValue = value;

}

}

10 return maxR;

}

int Class::getCycleRightShiftNumber(int a, int shift) {

int highDigit = 1;

// Подсчет кол-ва разрядов

1 while (a / (highDigit \* 10))

2 highDigit \*= 10;

// Сдвиг числа

3 for (int i = 0;

4 i < shift;

5 i++) {

6 int temp = a % 10;

7 a = a / 10 + highDigit \* temp;

}

8 return a;

}

double Class::sumEvenNumbers(double\*\* A, int size)

{

double sumA = 0;

1 for (int i = 0;

2 i < size;

3 i++)

4 for (int j = 0;

5 j < size - i - 1;

6 j++)

7 if ((int)A[i][j] % 2 == 0)

8 sumA += A[i][j];

9 return sumA;

}

UnitTest1.css:

#include "pch.h"

#include "CppUnitTest.h"

#include "../ConsoleApplication1/Class.h"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTest1

{

TEST\_CLASS(UnitTest1)

{

public:

TEST\_METHOD(getOddRankAndReverseAIsZero\_Test\_C2)

{

//arrange

int a = 0;

int expected = 0;

// act

int result = Class::getOddRankAndReverse(a);

// assert

Assert::AreEqual(expected, result);

}

TEST\_METHOD(getOddRankAndReverseAIsNotZero\_Test\_C2)

{

//arrange

int a = 1;

int expected = 1;

// act

int result = Class::getOddRankAndReverse(a);

// assert

Assert::AreEqual(expected, result);

}

TEST\_METHOD(getEvenRankWithMaxEvenValueAIsZero\_Test\_C2)

{

//arrange

int a = 0;

int expected = 0;

// act

int result = Class::getEvenRankWithMaxEvenValue(a);

// assert

Assert::AreEqual(expected, result);

}

TEST\_METHOD(getEvenRankWithMaxEvenValueAIsNotZeroWithoutEvenValues\_Test\_C2)

{

//arrange

int a = 10;

int expected = 0;

// act

int result = Class::getEvenRankWithMaxEvenValue(a);

// assert

Assert::AreEqual(expected, result);

}

TEST\_METHOD(getEvenRankWithMaxEvenValueAIsNotZeroWithEvenValue\_Test\_C2)

{

//arrange

int a = 20;

int expected = 2;

// act

int result = Class::getEvenRankWithMaxEvenValue(a);

// assert

Assert::AreEqual(expected, result);

}

TEST\_METHOD(getCycleRightShiftTwoDigitNumberWithZeroShirt\_Test\_C2) {

//arrange

int a = 12;

int shift = 0;

int expected = 12;

// act

int result = Class::getCycleRightShiftNumber(a, shift);

// assert

Assert::AreEqual(expected, result);

}

TEST\_METHOD(getCycleRightShiftTwoDigitNumberWithOneShirt\_Test\_C2) {

//arrange

int a = 12;

int shift = 1;

int expected = 21;

// act

int result = Class::getCycleRightShiftNumber(a, shift);

// assert

Assert::AreEqual(expected, result);

}

TEST\_METHOD(getCycleRightShiftOneDigitNumberWithZeroShirt\_Test\_C2) {

//arrange

int a = 9;

int shift = 0;

int expected = 9;

// act

int result = Class::getCycleRightShiftNumber(a, shift);

// assert

Assert::AreEqual(expected, result);

}

TEST\_METHOD(getCycleRightShiftOneDigitNumberWithOneShirt\_Test\_C2) {

//arrange

int a = 9;

int shift = 1;

int expected = 9;

// act

int result = Class::getCycleRightShiftNumber(a, shift);

// assert

Assert::AreEqual(expected, result);

}

TEST\_METHOD(sumEvenNumbers\_Test\_C2\_0) {

//arrange

double expected = 0;

double\*\* M;

// act

double result = Class::sumEvenNumbers(M, 0);

// assert

Assert::AreEqual(expected, result);

}

TEST\_METHOD(sumEvenNumbers\_Test\_C2\_1) {

//arrange

double expected = 0;

int size = 1;

double\*\* M = new double\* [size];

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

M[i] = new double[size];

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

M[i][j] = 1;

}

// act

double result = Class::sumEvenNumbers(M, size);

// assert

Assert::AreEqual(expected, result);

}

TEST\_METHOD(sumEvenNumbers\_Test\_C2\_21) {

//arrange

double expected = 0;

int size = 2;

double\*\* M = new double\* [size];

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

M[i] = new double[size];

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

M[i][j] = 1;

}

// act

double result = Class::sumEvenNumbers(M, size);

// assert

Assert::AreEqual(expected, result);

}

TEST\_METHOD(sumEvenNumbers\_Test\_C2\_22) {

//arrange

double expected = 2;

int size = 2;

double\*\* M = new double\* [size];

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

M[i] = new double[size];

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

M[i][j] = 2;

}

// act

double result = Class::sumEvenNumbers(M, size);

// assert

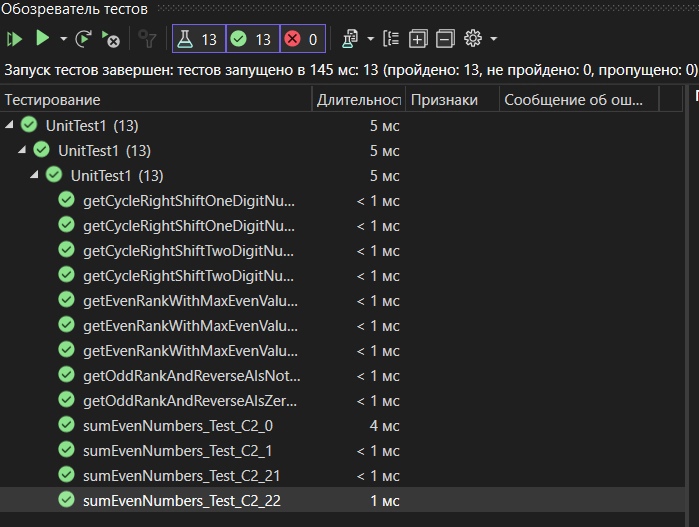
Assert::AreEqual(expected, result);

}

};

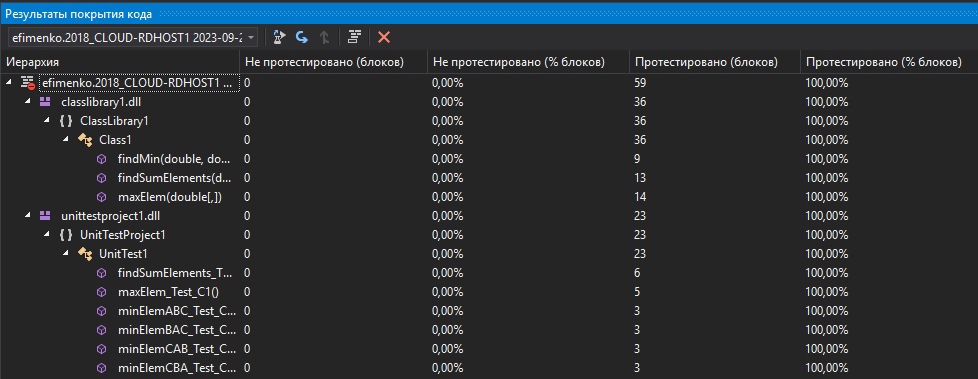
}

# Результаты выполнения модульных тестов



Все тесты выполнились успешно.

# Результаты покрытия разработанного кода тестами



Таким образом, по рисунку выше видно, что написанные тесты по критерию С2 покрыли весь код (100%).

# Вывод

В результате выполнения практической работы был изучен структурный критерий С2. Для его практического применения были разработаны 3 функции на языке C++ и написано 13 тестов.

Тестирование выполнилось успешно и покрыло весь код.