МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт информационных технологий и управления в технических системах

Кафедра «Информационные системы»

**ОТЧЕТ**

о выполнении лабораторной работы №3

по дисциплине “Тестирование Программного Обеспечения”

Выполнил: ст. гр. ИС/б-20-2-о

Белик Г. М.

Проверил: доц. каф.

«Информационные системы»

Строганов В. А.

Севастополь

2022

**ЛАБОРАТОРНАЯ РАБОТА №3**

**«Исследование способов модульного тестирования программного обеспечения»**

**Цель работы**

Исследовать основные подходы к модульному тестированию программного обеспечения. Приобрести практические навыки составления модульных тестов для объектно-ориентированных программ.

**Задачи**

1. Выбрать в качестве тестируемого один из классов, спроектированных в лабораторных работах № 1 – 4.
2. Составить спецификацию тестового случая для одного из методов выбранного класса, как показано в методических указаниях.
3. Реализовать тестируемый класс и необходимое тестовое окружение на языке С#.
4. Выполнить тестирование с выводом результатов на экран и сохранением в log-файл.
5. Проанализировать результаты тестирования, сделать выводы

**Ход работы**

Для тестирования был выбран класс печати результирующего вектора, который должен

выводить произведение элементов в строках матрицы, не содержащих отрицательных элементов.

После этого составили с спецификацию тестового случая для одного из методов выбранного класса.

Название тестируемого класса: T.

Название тестового случая: calculation.

Описание тестового случая: Тест проверяет правильность работы вычислений – полученные

элементы матрицы построчно проверяются на положительность и перемножаются, после записываются в результирующий одномерный массив, затем для наглядности результаты выводятся на экран и записываются в файл Test.log. Для примера была выбрана размерность матрицы 3\*3. В первой строке -2 -3 4; Во второй 1 0 5; В третьей 3 6 7. В первой строке находятся запрещенные значения, во второй ответ 0, в третьей 126.

Текст программы:

#include <iostream>

#include<Windows.h>

#include <fstream>

using namespace std;

template<class T>

T\* calculation(T\*\* matriza, T\* massiv, int N)

{

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

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

{

massiv[i] \*= matriza[i][j];

if (matriza[i][j] < 0)

{

massiv[i] = -1;

break;

}

}

return massiv;

}

template<class T>

void pechat(T\* massiv, int N)

{

ofstream Test;

Test.open("Test.log");

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

{

if (massiv[i] >= 0)

{

cout << "Произведение элементов в " << i << " строкe = " << massiv[i] << endl;

Test << "Произведение элементов в " << i << " строкe = " << massiv[i] << endl;

}

else

{

cout << "Строка " << i << " содержит отрицательные элементы\n";

Test << "Строка " << i << " содержит отрицательные элементы\n";

}

}

Test.close();

}

template<class T>

T\*\* read(T\*\* matriza, int N)

{

cout << "Введите значения " << N \* N << " элементов матрицы :\n";

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

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

{

cin >> matriza[i][j];

}

return matriza;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int N;

cout << "Введите положительное количество целых элементов маcсива = ";

cin >> N;

int\* massiv = new int[N];

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

massiv[i] = 1;

int\*\* matriza = new int\*[N];

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

matriza[i] = new int[N];

matriza = read(matriza, N);

massiv = calculation(matriza, massiv, N);

pechat(massiv, N);

}

Результат работы программы (рисунок 1).

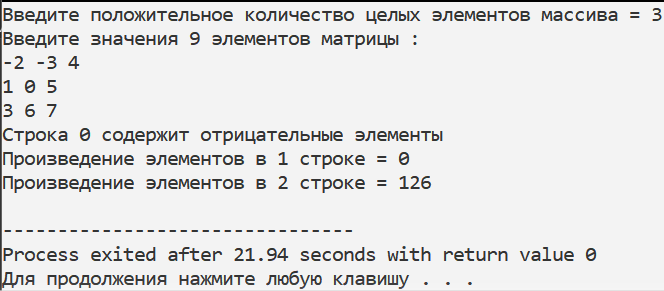


Рисунок 1 – Вычисления класса calculation

Запись расчетов в файл (рисунок 2).

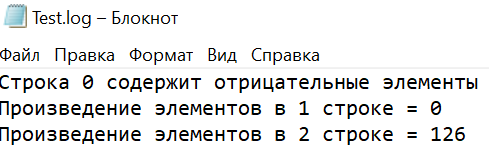


Рисунок 2 – Содержимое файла Test.log

На основе полученных результатов вычислений можно сказать о соответствии класса

calculation своей спецификации, так как ожидаемые и полученные результаты совпадают.

**Выводы**

В ходе лабораторной работы были исследованы основные подходы к модульному

тестированию программного обеспечения просмотр программного кода и тестовые прогоны. Были приобретены практические навыки составления модульных тестов для объектно-ориентированных программ: Осуществлен прогон программы, выявлена способность программы вести себя в соответствии со спецификацией. Выбрали наборы входных данных, определены соответствующие им правильные наборы выходных данных и сопоставили их с реально получаемыми выходными данными.