МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Курский государственный университет»

кафедра программного обеспечения и администрирования информационных систем

-

Отчёт
по контрольной работе № 1

«Тестовое окружение»

по дисциплине

«Верификация программного обеспечения»

Выполнил: студент(ка) группы 411
Мусонда Салиму

Проверил: к.т.н., доцент
кафедры ПОиАИС
Макаров К.С.

Курск, 2020

Цель: изучение модульного тестирования с использованием метода «чёрный ящик».

Задачи:

1) Изучить основные понятия Лекций 2-4;

2) Составить тест-план и провести модульное тестирование (согласно варианту) одного из методов, представленных в разделе **Задания**.

В качестве примера тестируемого модуля используется программная библиотека функций CalcClass.dll, находящаяся в каталоге с данной контрольной работой.

В отчёте к контрольной работе в качестве экспериментальных результатов представить текст программы и экранную форму разработанного тестового драйвера с результатами тестирования.

Общие понятия

Основной объем тестирования практически любой сложной системы обычно выполняется в автоматическом режиме. Кроме того, тестируемая система обычно разбивается на отдельные модули, каждый из которых тестируется вначале отдельно от других, затем в комплексе.

Это означает, что для выполнения тестирования необходимо создать некоторую среду, которая обеспечит запуск и выполнение тестируемого модуля, передаст ему входные данные, соберет реальные выходные данные, полученные в результате работы системы на заданных входных данных (рисунок 1).

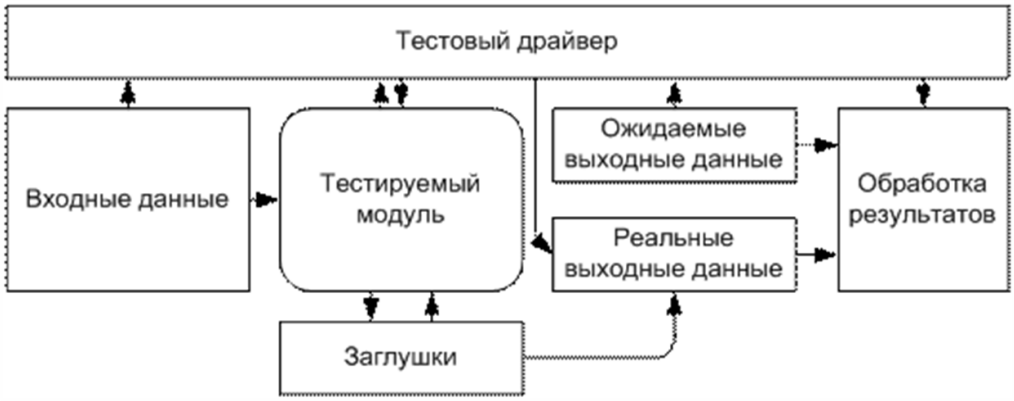

Рисунок 1 – Обобщённая схема среды тестирования

После этого среда должна сравнить реальные выходные данные с ожидаемыми и на основании данного сравнения сделать вывод о соответствии поведения модуля заданному.

Тестовое окружение также может использоваться для отчуждения отдельных модулей системы от всей системы. Разделение модулей системы на ранних этапах тестирования позволяет более точно локализовать проблемы, возникающие в их программном коде. Для поддержки работы модуля в отрыве от системы тестовое окружение должно моделировать поведение всех модулей, к функциям или данным которых обращается тестируемый модуль.

Поскольку тестовое окружение само является программой (причем, часто реализованной не на том языке программирования, на котором написана система), оно тоже должно быть протестировано. Целью тестирования тестового окружения является доказательство того, что тестовое окружение никаким образом не искажает выполнение тестируемого модуля и адекватно моделирует поведение системы.

Тестовое окружение для программного кода на структурных языках программирования состоит из двух компонентов – драйвера, который обеспечивает запуск и выполнение тестируемого модуля, и заглушек, которые моделируют функции, вызываемые из данного модуля. Разработка тестового драйвера представляет собой отдельную задачу тестирования, сам драйвер должен быть протестирован, дабы исключить неверное тестирование. Драйвер и заглушки могут иметь различные уровни сложности, требуемый уровень

сложности выбирается в зависимости от сложности тестируемого модуля и уровня тестирования. Так, драйвер может выполнять следующие функции:

1 вызов тестируемого модуля;

2 1 + передача в тестируемый модуль входных значений и прием результатов;

3 2 + вывод выходных значений;

4 3 + протоколирование процесса тестирования и ключевых точек программы;

Заглушки могут выполнять следующие функции:

1 не производить никаких действий (такие заглушки нужны для корректной сборки тестируемого модуля);

2 выводить сообщения о том, что заглушка была вызвана;

3 1 + выводить сообщения со значениями параметров, переданных в функцию;

4 2 + возвращать значение, заранее заданное во входных параметрах теста;

5 3 + выводить значение, заранее заданное во входных параметрах теста;

6 3 + принимать от тестируемого ПО значения и передавать их в драйвер.

Тестовое окружение для объектно-ориентированного ПО выполняет те же самые функции, что и для структурных программ (на процедурных языках). Однако, оно имеет некоторые особенности, связанные с применением наследования и инкапсуляции.

Если при тестировании структурных программ минимальным тестируемым объектом является функция, то в объектно-ориентированном программном обеспечении минимальным объектом является класс. При применении принципа инкапсуляции все внутренние данные класса и некоторая часть его методов недоступна извне. В этом случае тестировщик лишен возможности обращаться в своих тестах к данным класса и

произвольным образом вызывать методы; единственное, что ему доступно – вызывать методы внешнего интерфейса класса.

Существует несколько подходов к тестированию классов, каждый из них накладывает свои ограничения на структуру драйвера и заглушек.

1 Драйвер создает один или больше объектов тестируемого класса, все обращения к объектам происходят только с использованием их внешнего интерфейса. Текст драйвера в этом случае представляет собой так называемый тестирующий класс, который содержит по одному методу для каждого тестового примера. Процесс тестирования заключается в последовательном вызове этих методов. Вместо заглушек в состав тестового окружения входит программный код реальной системы, соответственно, отсутствует изоляция тестируемого класса. Именно такой подход к тестированию принят сейчас в большинстве методологий и сред разработки. Его классическое название – unit testing (тестирование модулей).

2 Аналогично предыдущему подходу, но для всех классов, которые использует тестируемый класс, создаются заглушки

3 Программный код тестируемого класса модифицируется таким образом, чтобы открыть доступ ко всем его свойствам и методам. Строение тестового окружения в этом случае полностью аналогично окружению для тестирования структурных программ.

4 Используются специальные средства доступа к закрытым данным и методам класса на уровне объектного или исполняемого кода – скрипты отладчика или accessors в Visual Studio.

Основное достоинство первых двух методов – при их использовании класс работает точно таким же образом, как в реальной системе. Однако в этом случае нельзя гарантировать того, что в процессе тестирования будет выполнен весь программный код класса и не останется непротестированных методов.

Основной недостаток 3-го метода – после изменения исходных текстов тестируемого модуля нельзя дать гарантии того, что класс будет вести себя

таким же образом, как и исходный. В частности, это связано с тем, что изменение защиты данных класса влияет на наследование данных и методов другими классами.

Тестирование наследования – отдельная сложная задача в объектно-ориентированных системах. После того, как протестирован базовый класс, необходимо тестировать классы-потомки. Однако, для базового класса нельзя создавать заглушки, так как в этом случае можно пропустить возможные проблемы полиморфизма. Если класс-потомок использует методы базового класса для обработки собственных данных, необходимо убедиться в том, что эти методы работают.

Таким образом, иерархия классов может тестироваться сверху вниз, начиная от базового класса. Тестовое окружение при этом может меняться для каждой тестируемой конфигурации классов.

Тестовое окружение применительно к системе «Калькулятор»

Тесты, проделанные в ходе контрольной работы №1, как правило, проводятся не вручную. Для целей тестирования пишут специальную программу – тестовый драйвер, который и проводит тестирование. Более того, такие программы часто пишутся на другом языке, нежели тестируемая программа, или создаются автоматически, с помощью специальных утилит.

В ходе данной контрольной работы создадим простой тестовый драйвер на C# для тестирования функций "Калькулятора".

Рассмотрим тестирование функции сложения. Для простоты будем пользоваться лишь четырьмя общими тест- требованиями.

1 Оба входных параметра принадлежат допустимой области, и выходное значение принадлежит допустимой области.

2 Первый входной параметр принадлежит допустимой области, второй не принадлежит допустимой области

3 Первый входной параметр не принадлежит допустимой области, второй принадлежит допустимой области

4 Оба входных параметров принадлежат допустимой области, а значение функции не принадлежит допустимой области.

Листинг программы, реализующей проверку данных тест-требований имеет следующий вид:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using BaseCalculator;

namespace WindowsFormsApplication2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void buttonStartDel_Click(object sender, EventArgs e)
        {
            try
            {
                richTextBox1.Text = "";
                richTextBox1.Text += "Test Case 1\n";
                richTextBox1.Text += "Входные данные a=78508, b=-304\n";
                richTextBox1.Text += "Ожидаемый результат: res=78204 &&
error=\"\"" + "\n";
                int res = CalcClass.Add(78508, -304);
                string error = CalcClass.lastError;
                richTextBox1.Text += "Код ошибки:" + error + "\n";
                richTextBox1.Text += "Получившийся результат:" + "res=" +
res.ToString() + "error=" + error.ToString() + "\n";
                if (res == 78204 && error == "")
                {
                    richTextBox1.Text += "Тест пройден\n\n";
                }
                else
                {
                    richTextBox1.Text += "Тест не пройден\n\n";
                }
            }
            catch (Exception ex)
            {
                richTextBox1.Text += "Перехвачено исключение:" +
ex.ToString() + "\nТест не пройден.\n";
            }
            try
            {
    //          richTextBox1.Text = "";
                richTextBox1.Text += "Test Case 2\n";
                richTextBox1.Text += "Входные данные a=-2850800078,
b=3000000000\n";
                richTextBox1.Text += "Ожидаемый результат: res=0 &&
error=\"Error 06\"\n";
```

```
                int res = CalcClass.Add(-2850800078, 3000000000);
                string error = CalcClass.lastError;
                richTextBox1.Text += "Код ошибки:" + error + "\n";
                richTextBox1.Text += "Получившийся результат:" + "res=" +
res.ToString() + "error=" + error.ToString() + "\n";
                if (res == 0 && error == "Error 06")
                {
                    richTextBox1.Text += "Тест пройден\n\n";
                }
                else
                {
                    richTextBox1.Text += "Тест не пройден\n\n";
                }
            }
            catch (Exception ex)
            {
                richTextBox1.Text += "Перехвачено исключение:" +
ex.ToString() + "\nТест не пройден.\n";
            }
            try
            {
     //          richTextBox1.Text = "";
                richTextBox1.Text += "Test Case 3\n";
                richTextBox1.Text += "Входные данные a=3000000000, b=-
2850800078\n";
                richTextBox1.Text += "Ожидаемый результат: res=0 &&
error=\"Error 06\"\n";
                int res = CalcClass.Add(3000000000, -2850800078);
                string error = CalcClass.lastError;
                richTextBox1.Text += "Код ошибки:" + error + "\n";
                richTextBox1.Text += "Получившийся результат:" + "res=" +
res.ToString() + "error=" + error.ToString() + "\n";
                if (res == 0 && error == "Error 06")
                {
                    richTextBox1.Text += "Тест пройден\n\n";
                }
                else
                {
                    richTextBox1.Text += "Тест не пройден\n\n";
                }
            }
            catch (Exception ex)
            {
                richTextBox1.Text += "Перехвачено исключение:" +
ex.ToString() + "\nТест не пройден.\n";
            }
            try
            {
     //          richTextBox1.Text = "";
                richTextBox1.Text += "Test Case 4\n";
                richTextBox1.Text += "Входные данные a=2000000000,
b=2000000000\n";
                richTextBox1.Text += "Ожидаемый результат: res=0 &&
error=\"Error 06\"\n";
                int res = CalcClass.Add(2000000000, 2000000000);
                string error = CalcClass.lastError;
                richTextBox1.Text += "Код ошибки:" + error + "\n";
                richTextBox1.Text += "Получившийся результат:" + "res=" +
res.ToString() + "error=" + error.ToString() + "\n";
                if (res == 0 && error == "Error 06")
                {
                    richTextBox1.Text += "Тест пройден\n\n";
                }
                else
```

```
                {
                    richTextBox1.Text += "Тест не пройден\n\n";
                }
            }
            catch (Exception ex)
            {
                richTextBox1.Text += "Перехвачено исключение:" +
ex.ToString() + "\nТест не пройден.\n";
            }
        }
    }
}
```

Каждый тестовый пример находится внутри блока try-catch для того, чтобы перехватить любое сгенерированное исключение внутри методов Add().

При этом файл CalcClass.dll, в котором и реализованы все математические методы, необходимо добавить в References проекта.

Проведем тестирование и получим результат, представленный на рисунке 2.

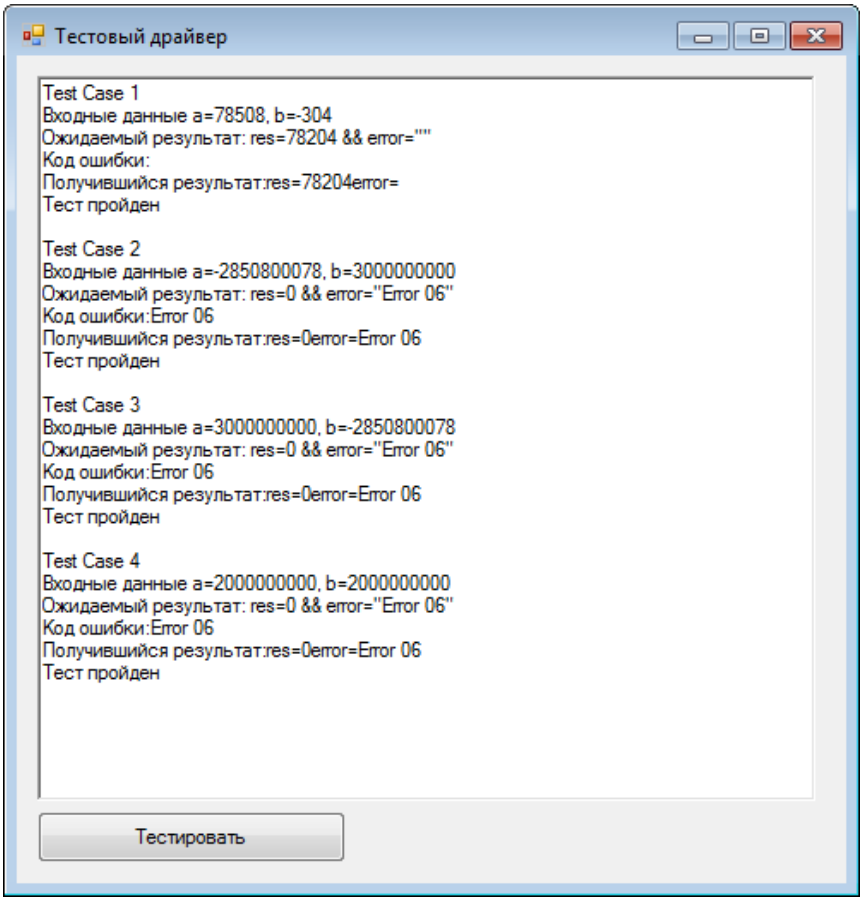

Рисунок 2 – Результат работы тестового драйвера, тестовый драйвер, проверяющий функцию деления

Заметим, что при таком подходе к тестированию нам удается локализовать ошибки. Если что-то работает не так, как надо, то можно с

уверенностью утверждать, что ошибка содержится именно в функции деления, в то время, как при ручном тестировании нельзя определённо сказать, где именно она произошла.

Задания

Составить тест-план и провести модульное тестирование следующих метод:

Унарный плюс
/// <summary>
/// унарный плюс
/// </summary>
/// <param name="a"></param>
/// <returns></returns>
public static int IABS(long a)

Экспериментальные результаты

Текст программы

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using BaseCalculator;

namespace WindowsFormsApp1
{



        public partial class Form1 : Form
        {
            public Form1()
            {
                InitializeComponent();
            }




    private void Form1_Load(object sender, EventArgs e)
        {

        }

        private void button2_Click(object sender, EventArgs e)
        {
```

```
            try
            {
                richTextBox1.Text = "";
                richTextBox1.Text += "Test Case 1\n";
                richTextBox1.Text += "Входные данные a=-3456\n";
                richTextBox1.Text += "Ожидаемый результат: res=3456 && error=\"\"" +
"\n";
                int res = CalcClass.ABS(-3456);
                string error = CalcClass.lastError;
                richTextBox1.Text += "Код ошибки:" + error + "\n";
                richTextBox1.Text += "Получившийся результат:" + "res=" + res.ToString()
+ "error=" + error.ToString() + "\n";
                if (res == 3456 && error == "")
                {
                    richTextBox1.Text += "Тест пройден\n\n";
                }
                else
                {
                    richTextBox1.Text += "Тест не пройден\n\n";
                }
            }
            catch (Exception ex)
            {
                richTextBox1.Text += "Перехвачено исключение:" + ex.ToString() + "\nТест
не пройден.\n";
            }
            try
            {

                richTextBox1.Text += "Test Case 2\n";
                richTextBox1.Text += "Входные данные a=6789\n";
                richTextBox1.Text += "Ожидаемый результат: res=6789 && error=\"\"" +
"\n";
                int res = CalcClass.ABS(6789);
                string error = CalcClass.lastError;
                richTextBox1.Text += "Код ошибки:" + error + "\n";
                richTextBox1.Text += "Получившийся результат:" + "res=" + res.ToString()
+ "error=" + error.ToString() + "\n";
                if (res == 6789 && error == "")
                {
                    richTextBox1.Text += "Тест пройден\n\n";
                }
                else
                {
                    richTextBox1.Text += "Тест не пройден\n\n";
                }
            }
            catch (Exception ex)
            {
                richTextBox1.Text += "Перехвачено исключение:" + ex.ToString() + "\nТест
не пройден.\n";
            }
            try
            {

               // richTextBox1.Text = "";
                richTextBox1.Text += "Test Case 3\n";
                richTextBox1.Text += "Входные данные a=2147483648\n";
                richTextBox1.Text += "Ожидаемый результат: res=0 && error=\"\"" + "\n";
                int res = CalcClass.ABS(2147483648);
                string error = CalcClass.lastError;
                richTextBox1.Text += "Код ошибки:" + error + "\n";
                richTextBox1.Text += "Получившийся результат:" + "res=" + res.ToString()
+ "error=" + error.ToString() + "\n";
                if (res == 0 && error == "Error 06")
```

```
                {
                    richTextBox1.Text += "Тест пройден\n\n";
                }
                else
                {
                    richTextBox1.Text += "Тест не пройден\n\n";
                }
            }
            catch (Exception ex)
            {
                richTextBox1.Text += "Перехвачено исключение:" + ex.ToString() + "\nТест
не пройден.\n";
            }
            try
            {

                //richTextBox1.Text = "";
                richTextBox1.Text += "Test Case 4\n";
                richTextBox1.Text += "Входные данные a=2147483647\n";
                richTextBox1.Text += "Ожидаемый результат: res=2147483647 && error=\"\""
+ "\n";
                int res = CalcClass.ABS(2147483647);
                string error = CalcClass.lastError;
                richTextBox1.Text += "Код ошибки:" + error + "\n";
                richTextBox1.Text += "Получившийся результат:" + "res=" + res.ToString()
+ "error=" + error.ToString() + "\n";
                if (res == 2147483647 && error == "")
                {
                    richTextBox1.Text += "Тест пройден\n\n";
                }
                else
                {
                    richTextBox1.Text += "Тест не пройден\n\n";
                }
            }
            catch (Exception ex)
            {
                richTextBox1.Text += "Перехвачено исключение:" + ex.ToString() + "\nТест
не пройден.\n";
            }


            try
            {

                // richTextBox1.Text = "";
                richTextBox1.Text += "Test Case 5\n";
                richTextBox1.Text += "Входные данные a=2147483648\n";
                richTextBox1.Text += "Ожидаемый результат: res=0 && error=\"\"" + "\n";
                int res = CalcClass.ABS(2147483648);
                string error = CalcClass.lastError;
                richTextBox1.Text += "Код ошибки:" + error + "\n";
                richTextBox1.Text += "Получившийся результат:" + "res=" + res.ToString()
+ "error=" + error.ToString() + "\n";
                if (res == 0 && error == "Error 06")
                {
                    richTextBox1.Text += "Тест пройден\n\n";
                }
                else
                {
                    richTextBox1.Text += "Тест не пройден\n\n";
                }
            }
            catch (Exception ex)
            {
```

```
                richTextBox1.Text += "Перехвачено исключение:" + ex.ToString() + "\nТест
не пройден.\n";
            }



            try
            {

                //richTextBox1.Text = "";
                richTextBox1.Text += "Test Case 6\n";
                richTextBox1.Text += "Входные данные a=-2147483647\n";
                richTextBox1.Text += "Ожидаемый результат: res=2147483647 && error=\"\""
+ "\n";
                int res = CalcClass.ABS(-2147483647);
                string error = CalcClass.lastError;
                richTextBox1.Text += "Код ошибки:" + error + "\n";
                richTextBox1.Text += "Получившийся результат:" + "res=" + res.ToString()
+ "error=" + error.ToString() + "\n";
                if (res == 2147483647 && error == "")
                {
                    richTextBox1.Text += "Тест пройден\n\n";
                }
                else
                {
                    richTextBox1.Text += "Тест не пройден\n\n";
                }







            }
            catch (Exception ex)
            {
                richTextBox1.Text += "Перехвачено исключение:" + ex.ToString() + "\nТест
не пройден.\n";
            }


            try
            {

                //richTextBox1.Text = "";
                richTextBox1.Text += "Test Case 7\n";
                richTextBox1.Text += "Входные данные a=-2147483648\n";
                richTextBox1.Text += "Ожидаемый результат: res=0 && error=\"\"" + "\n";
                int res = CalcClass.ABS(-2147483648);
                string error = CalcClass.lastError;
                richTextBox1.Text += "Код ошибки:" + error + "\n";
                richTextBox1.Text += "Получившийся результат:" + "res=" + res.ToString()
+ "error=" + error.ToString() + "\n";
                if (res == 0 && error == "Error 06")
                {
                    richTextBox1.Text += "Тест пройден\n\n";
                }
                else
                {
                    richTextBox1.Text += "Тест не пройден\n\n";
                }
```

```
            }
            catch (Exception ex)
            {
                richTextBox1.Text += "Перехвачено исключение:" + ex.ToString() + "\nТест
не пройден.\n";
            }

        }

    }
}
```

Тестирование

Тестирование унарного плюса представлено на рисунке 1-2.

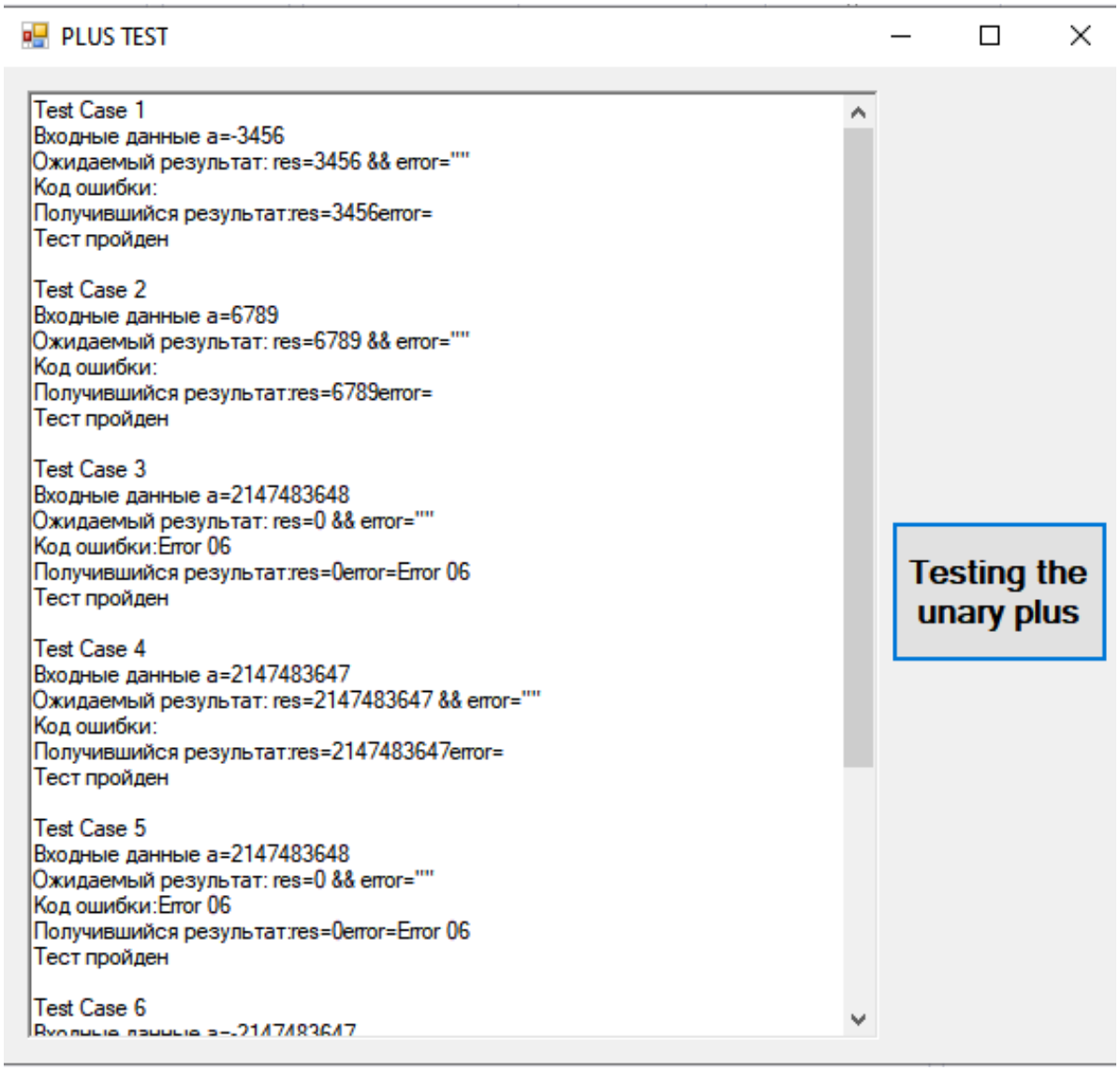

Рисунок 1 - Тестирование унарного плюса

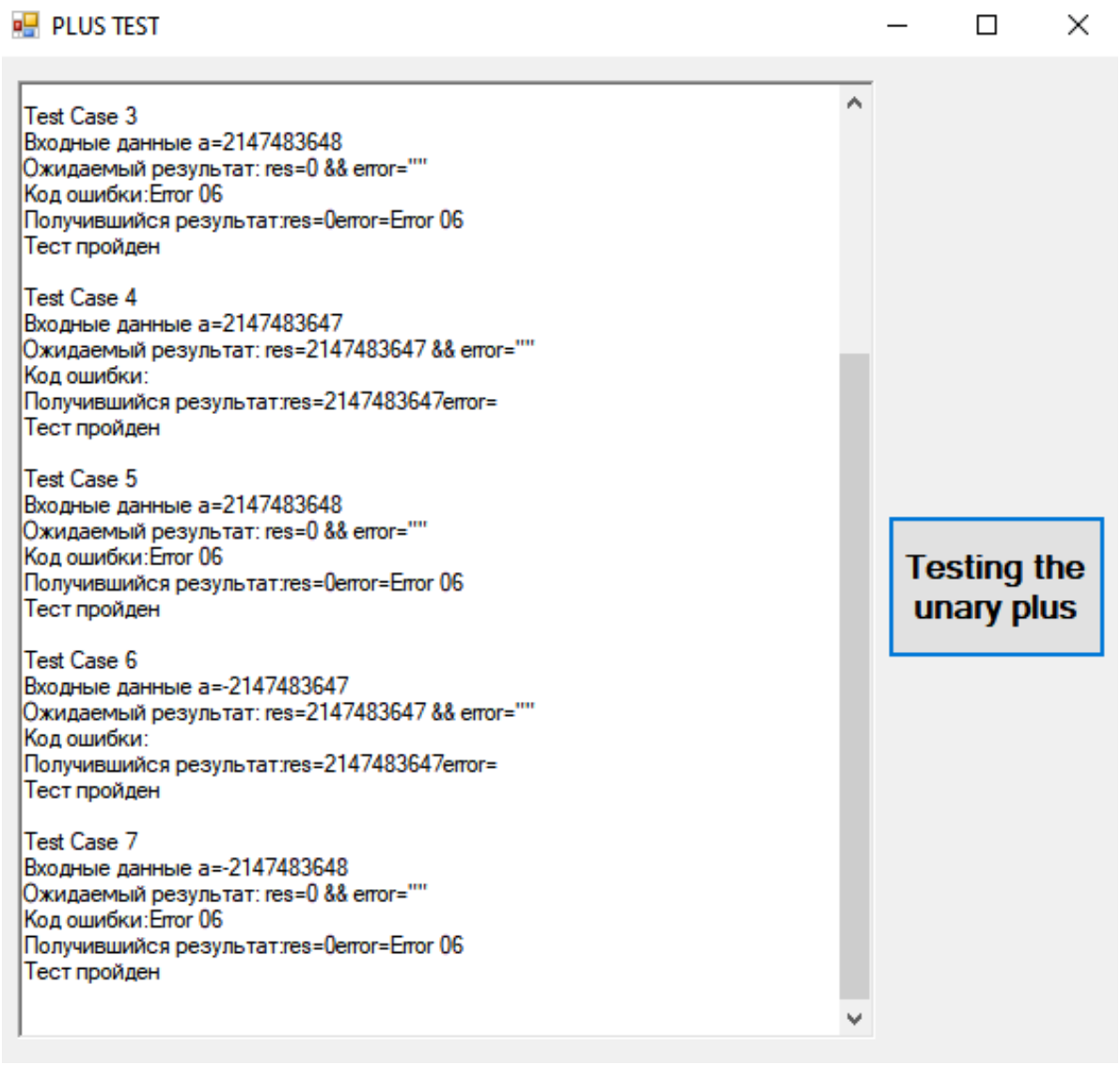

Рисунок 2 - Тестирование унарного плюса