В данной работе при помощи автоматических драйверов тестирования будут протестированы две составляющие части программы:

1. Логическая составляющая, отвечающая за правильный ввод, хранение и обработку

данных, реализованная в виде динамически подключаемой библиотеки (SwapArray.dll) будет тестироваться при помощи юнит – тестов(unit tests), использующих framework и визуальную среду исполнения тестов программы NUnit.

1. Графический интерфейс программы, отвечающий за последовательность ввода

пользователем информации, вывод подсказок и итогового результата будет тестироваться при помощи встроенного в среду разработки MS Visual Studio 2010 Enterprise Edition построителя кодированных тестов интерфейса пользователя.

**Тестирование логической составляющей.**

Динамически подключаемая библиотека SwapArray.dll содержит описание класса SwapArray, являющегося основным объектом тестирования в нашей программе.

Для его тестирования был создан новый проект MS Visual Studio - Nunit1, содержащий классы для тестирования логики (LogicControl) и корректного ввода данных (DataIntroduction) – это и будет наш драйвер тестирования (см. Приложение А).

В каждом из классов описаны методы, тестирующие одну из функциональностей программы (реализованных в классе SwapArray), либо тестирующие общую работу алгоритма на определенном наборе входных данных (например метод ComonWork).

**Алгоритм драйвера**: Каждый метод инициализирует переменную – экземпляр класса SwapArray, затем передает ей требуемые входные параметры и сравнивает возвращаемый результат с эталонным, либо ожидает сообщения об ошибке если были введены заведомо неправильные данные. В случае если возвращаемый результат совпадает с ожидаемым, данный тест помечается как успешно пройденный.

**Варианты тестирования**:

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя тест-кейса | Ввод (массив, размер сегмента, i1, i1) | Ожидаемый вывод |
| 1 | ConstructorTest | Вызов SwapArray() | Переменная типа SwapArray |
| 2 | ComonWork | "0123456789",2,2,7 | "0673451289" |
| 3 | InitialArray | "asdfghjkl" | "asdfghjkl" |
| 4 | Size\_Valid | "asdfghjk",3 | 3 |
| 5 | Size\_Overrange | "asdfghjk",6 | Сообщение об ошибке |
| 6 | Size\_CannotConvert | "asdfghjk",”f” | Сообщение об ошибке |
| 7 | Size\_WithoutArray | ,1 | Сообщение об ошибке |
| 8 | I1\_WithoutSize | "asdfghjk", ,1 | Сообщение об ошибке |
| 9 | I1\_CannotConvert | "asdfghjk",4,”t” | Сообщение об ошибке |
| 10 | I1\_Overrange | "asdfghjk",4,5 | Сообщение об ошибке |
| 11 | I1\_Valid | "asdfghjk",4,1 | 1 |
| 12 | I2\_Valid | "asdfghjk",4,1,5 | 5 |
| 13 | I2\_Overrange | "asdfghjk",4,1,3 | Сообщение об ошибке |
| 14 | I2\_CannotConvert | "asdfghjk",4,1,”n” | Сообщение об ошибке |
| 15 | I2\_WithoutI1 | "asdfghjk",4, ,3 | Сообщение об ошибке |

Таким образом, составленные тесты покрывают все методы класса SwapArray, учитывают классы эквивалентности вводимых значений и различные последовательности вызова методов класса.

**Результат тестирования** – все тесты пройдены успешно:

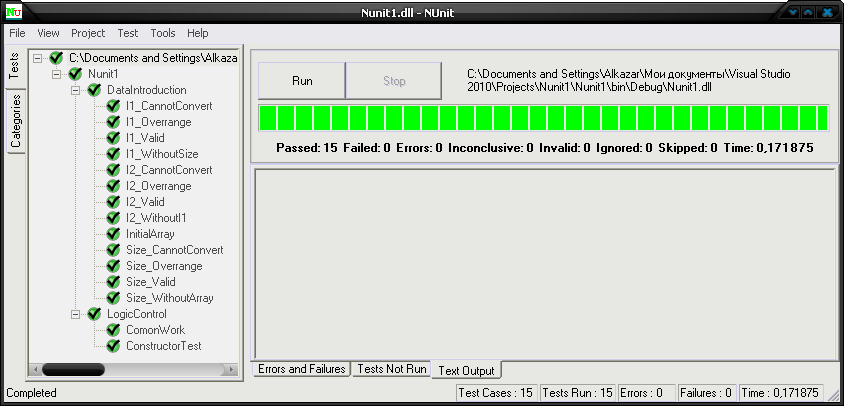


Рисунок 1 - Результат тестирования

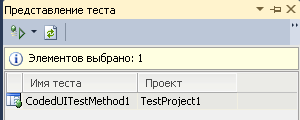
**Тестирование графического интерфейса программы**

Для тестирования пользовательского интерфейса программы использовались встроенные средства среды разработки MS Visual Studio, а именно построитель кодированных тестов интерфейса пользователя. Для его применения был создан отдельный проект MS Visual Studio – TestProject1, далее при помощи встроенных средств была записана последовательность действий пользователя по запуску тестируемого приложения, вводу данных, инициализации основного алгоритма обработки и закрытия приложения после получения результата.

****

Рисунок 2 - Построитель кодированных тестов интерфейса пользователя

Затем в сгенерированный код были добавлены инструкции для проверки соответствия полученного результата ожидаемому (см. Приложение Б). В результате был получен скрипт, взаимодействующий с программой на уровне графического интерфейса пользователя и способный воспроизводить пользовательские сценарии. Путем несложных изменений кода, можно будет добиться автоматической проверки работы программы для многочисленных и разнообразных входных данных.

****

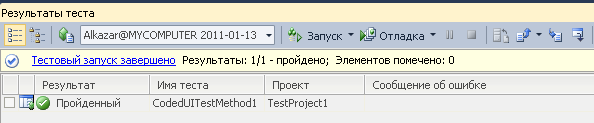
****

Рисунок 3 – Созданный тест и результат его выполнения

**Вывод**: В процессе выполнения данной лабораторной работы я ознакомился с принципами работы различных программ для автоматизации тестирования программ и их составных частей. Также я составил код нескольких тестов для проверки алгоритма и графического интерфейса созданной ранее программы.

**Приложение А**

using System;

using System.Collections.Generic;

using System.Text;

using NUnit.Framework;

using SwapArray;

namespace Nunit1

{

[TestFixture]

public class LogicControl

{

[Test]

public void ConstructorTest()

{

SwapArray.SwapArray sa = new SwapArray.SwapArray();

Assert.IsNotNull(sa);

}

[Test]

public void ComonWork()

{

string array = "0123456789";

string etalon = "0673451289";

string checkArray = null;

SwapArray.SwapArray sa = new SwapArray.SwapArray();

checkArray= sa.ProceedSwap(array, 1, 6, 2);

Assert.AreEqual(etalon, checkArray);

}

}

[TestFixture]

public class DataIntroduction

{

[Test]

public void InitialArray()

{

SwapArray.SwapArray sa = new SwapArray.SwapArray();

string s = "asdfghjkl";

sa.SetAr(s);

Assert.AreEqual(sa.GetAr(),s);

}

[Test]

public void Size\_Valid()

{

SwapArray.SwapArray sa = new SwapArray.SwapArray();

sa.SetAr("asdfghjk");

sa.SetSize("3",1);

Assert.AreEqual(sa.GetSize(), 3);

}

[Test]

[ExpectedException("System.Exception")]

public void Size\_Overrange()

{

SwapArray.SwapArray sa = new SwapArray.SwapArray();

sa.SetAr("asdfghjk");

sa.SetSize("6",1);

}

[Test]

[ExpectedException("System.Exception")]

public void Size\_CannotConvert()

{

SwapArray.SwapArray sa = new SwapArray.SwapArray();

sa.SetAr("asdfghjk");

sa.SetSize("f",1);

}

[Test]

[ExpectedException("System.Exception")]

public void Size\_WithoutArray()

{

SwapArray.SwapArray sa = new SwapArray.SwapArray();

sa.SetSize("1",1);

}

[Test]

[ExpectedException("System.Exception")]

public void I1\_WithoutSize()

{

SwapArray.SwapArray sa = new SwapArray.SwapArray();

sa.SetAr("asdfghjk");

sa.SetI1("1", 1);

}

[Test]

[ExpectedException("System.Exception")]

public void I1\_Overrange()

{

SwapArray.SwapArray sa = new SwapArray.SwapArray();

sa.SetAr("asdfghjk");

sa.SetSize("4");

sa.SetI1("5", 1);

}

[Test]

[ExpectedException("System.Exception")]

public void I1\_CannotConvert()

{

SwapArray.SwapArray sa = new SwapArray.SwapArray();

sa.SetAr("asdfghjk");

sa.SetSize("4");

sa.SetI1("t", 1);

}

[Test]

public void I1\_Valid()

{

SwapArray.SwapArray sa = new SwapArray.SwapArray();

sa.SetAr("asdfghjk");

sa.SetSize("4");

sa.SetI1("1", 1);

}

[Test]

public void I2\_Valid()

{

SwapArray.SwapArray sa = new SwapArray.SwapArray();

sa.SetAr("asdfghjk");

sa.SetSize("4");

sa.SetI1("1");

sa.SetI2("5",1);

}

[Test]

[ExpectedException("System.Exception")]

public void I2\_Overrange()

{

SwapArray.SwapArray sa = new SwapArray.SwapArray();

sa.SetAr("asdfghjk");

sa.SetSize("4");

sa.SetI1("1");

sa.SetI2("3", 1);

}

[Test]

[ExpectedException("System.Exception")]

public void I2\_CannotConvert()

{

SwapArray.SwapArray sa = new SwapArray.SwapArray();

sa.SetAr("asdfghjk");

sa.SetSize("4");

sa.SetI1("1");

sa.SetI2("n", 1);

}

[Test]

[ExpectedException("System.Exception")]

public void I2\_WithoutI1()

{

SwapArray.SwapArray sa = new SwapArray.SwapArray();

sa.SetAr("asdfghjk");

sa.SetSize("4");

sa.SetI2("3", 1);

}

}

}

**Приложение Б**

public void RecordedMethod1()

{

// Двойной щелчок "WindowsFormsApplication6.exe" элемент списка

Mouse.DoubleClick(uIWindowsFormsApplicatListItem, new Point(30, 12));

// Ввести "12345678" в "textBox1" надпись

uITextBox1Edit.Text = this.RecordedMethod1Params.UITextBox1EditText;

// Ввести "{Enter}" в "textBox1" надпись

Keyboard.SendKeys(uITextBox1Edit, this.RecordedMethod1Params.UITextBox1EditSendKeys, ModifierKeys.None);

// Ввести "4" в "textBox2" надпись

uITextBox2Edit.Text = this.RecordedMethod1Params.UITextBox2EditText;

// Ввести "{Enter}" в "textBox2" надпись

Keyboard.SendKeys(uITextBox2Edit, this.RecordedMethod1Params.UITextBox2EditSendKeys, ModifierKeys.None);

// Ввести "1" в "textBox3" надпись

uITextBox3Edit.Text = this.RecordedMethod1Params.UITextBox3EditText;

// Ввести "{Enter}" в "textBox3" надпись

Keyboard.SendKeys(uITextBox3Edit, this.RecordedMethod1Params.UITextBox3EditSendKeys, ModifierKeys.None);

// Ввести "5" в "textBox4" надпись

uITextBox4Edit.Text = this.RecordedMethod1Params.UITextBox4EditText;

// Ввести "{Enter}" в "textBox4" надпись

Keyboard.SendKeys(uITextBox4Edit, this.RecordedMethod1Params.UITextBox4EditSendKeys, ModifierKeys.None);

// Щелчок "Обменять" кнопка

Mouse.Click(uIОбменятьButton, new Point(28, 13));

// Убедитесь, что у надписи "textBox5" свойство "Text" равно "56781234"

Assert.AreEqual(this.AssertMethod1ExpectedValues.UITextBox5EditText, uITextBox5Edit.Text);

// Щелчок "Закрыть" кнопка

Mouse.Click(uIЗакрытьButton, new Point(6, 9));

}