Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГТУ»)**

**Институт компьютерных систем и информационной безопасности**

**Кафедра информационных систем и программирования**

**ЛАБОРАТОРНАЯ РАБОТА №4**

**Дисциплина: тестирование и отладка программного обеспечения**

Работу выполнил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Я.С Давков

Направление подготовки: 09.03.04 Программная инженерия

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Г. Волик

**1 Цель работы**

Изучить подход к тестированию методом белого ящика.

**2 Краткая теория**

Тестирование программного кода – процесс выполнения программного кода, направленный на выявление существующих в нем дефектов. Под дефектом здесь понимается участок программного кода, выполнение которого при определенных условиях приводит к неожиданному поведению системы (т.е. поведению, не соответствующему требованиям). Задача тестирования – определение условий, при которых проявляются дефекты системы и протоколирование этих условий. В задачи тестирования обычно не входит выявление конкретных дефектных участков программного кода и никогда не входит исправление дефектов – это задача отладки, которая выполняется по результатам тестирования системы. Поскольку современные программные системы имеют весьма значительные размеры, при тестировании их программного кода используется метод функциональной декомпозиции. Система разбивается на отдельные модули (классы, пространства имен и т.п.), имеющие определенную требованиями функциональность и интерфейсы. После этого по отдельности тестируется каждый модуль – выполняется модульное тестирование. Затем выполняется сборка отдельных модулей в более крупные конфигурации – выполняется интеграционное тестирование, и наконец, тестируется система в целом – выполняется системное тестирование.

**3 Задание**

1) Создать класс (в соответствии с вариантом задания из п.5), реализующий обработку с текстовой информации. 2) Построить управляющий граф программы (УГП). 3) Используя знания о структуре программы определить набор тестов для тестирования методом белого ящика. 4) Указать какие пути реализуются (проверяются) тестом (см. таблица 4.1). 5) Указать какие критерии покрываются путем (см. таблица 4.2). 6) Оценить качество полученных тестовых данных с точки зрения покрытия кода тестами. 7) Протестировать метод с использованием средств автоматизации тестирования. 8) Составить отчет о результатах проведенного тестирования.

**4 Выполнение работы**

Мой вариант – 4. Создать класс, реализующий метод для вставки символа в произвольную позицию строки.

Листинг класса:

public class TextProcessor  
{  
 public string InsertChar(string input, char ch, int position)  
 {  
 if (input == null)  
 throw new ArgumentNullException(nameof(input));  
  
 if (position < 0)  
 throw new ArgumentOutOfRangeException(nameof(position), "Position must be >= 0.");  
  
 if (position > input.Length)  
 throw new ArgumentOutOfRangeException(nameof(position), "Position must be <= input.Length.");  
  
 int n = input.Length;  
 char[] result = new char[n + 1];  
  
 for (int i = 0; i < position; i++)  
 {  
 result[i] = input[i];  
 }  
  
 result[position] = ch;  
  
 for (int i = position; i < n; i++)  
 {  
 result[i + 1] = input[i];  
 }  
  
 return new string(result);  
 }  
}

Создадим управляющий граф.

Изображение выглядит как диаграмма, План, текст, Технический чертеж

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 1 - Управляющий граф программы

Подберём тесты так, чтобы покрыть все логические ветви и характерные варианты поведения:

1. InputNull: input = null, ch = 'a', position = 0  
   Ожидание: ArgumentNullException.
2. PositionNegative: input = "abc", ch = 'x', position = -1  
   Ожидание: ArgumentOutOfRangeException.
3. PositionGreaterThanLength: input = "abc", ch = 'x', position = 4 (len=3)  
   Ожидание: ArgumentOutOfRangeException.
4. InsertAtBeginning: input = "hello", ch = 'X', position = 0  
   Ожидание: "Xhello" — покрывает случай, когда первый цикл (copy 0..position-1) делает 0 итераций.
5. InsertAtEnd: input = "hello", ch = '!', position = 5 (position == length)  
   Ожидание: "hello!" — проверка вставки в конец (второй цикл копирует все n элементов в сдвиг).
6. InsertInMiddle: input = "abcde", ch = 'Z', position = 2  
   Ожидание: "abZcde" — проверка срединной вставки (оба цикла выполняют >0 итераций).
7. InsertIntoEmptyString: input = "", ch = 'q', position = 0  
   Ожидание: "q" — крайний случай пустой строки.

Сделаем таблицу

|  |  |  |
| --- | --- | --- |
| **№** | **Критерий** | **Пути** |
| 1 | Покрытие решений, условий, путей | 1,2 |
| 2 | Покрытие решений, условий, путей | 1,3,4 |
| 3 | Покрытие решений, условий, путей | 1,3,5,13 |
| 4 | Покрытие операторов, решений, условий, путей | 1,3,5,6,7,8,9,10,11,12 |
| 5 | Покрытие операторов, решений, условий, путей | 1,3,5,6,7,8,9,12 |
| 6 | Покрытие операторов, решений, условий, путей | 1,3,5,6,7,8,9,10,11,12 |
| 7 | Покрытие операторов, решений, условий, путей | 1,3,5,6,7,9,12 |

Код покрыт на 100% тестами, запустим тесты и проверим их выполнение.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 2 – результат запуска тестов

Листинг тестов:

[TestFixture]  
public class TextProcessorTests  
{  
 private TextProcessor \_tp;  
  
 [SetUp]  
 public void SetUp()  
 {  
 \_tp = new TextProcessor();  
 }  
  
 [Test]  
 public void InsertChar\_InputNull\_ThrowsArgumentNullException()  
 {  
 Assert.That(() => \_tp.InsertChar(null, 'a', 0),  
 Throws.TypeOf<ArgumentNullException>());  
 }  
  
 [Test]  
 public void InsertChar\_PositionNegative\_ThrowsArgumentOutOfRangeException()  
 {  
 Assert.That(() => \_tp.InsertChar("abc", 'x', -1),  
 Throws.TypeOf<ArgumentOutOfRangeException>());  
 }  
  
 [Test]  
 public void InsertChar\_PositionGreaterThanLength\_ThrowsArgumentOutOfRangeException()  
 {  
 Assert.That(() => \_tp.InsertChar("abc", 'x', 4),  
 Throws.TypeOf<ArgumentOutOfRangeException>());  
 }  
  
 [Test]  
 public void InsertChar\_InsertAtBeginning\_ReturnsCorrectString()  
 {  
 string result = \_tp.InsertChar("hello", 'X', 0);  
 Assert.That(result, Is.EqualTo("Xhello"));  
 }  
  
 [Test]  
 public void InsertChar\_InsertAtEnd\_ReturnsCorrectString()  
 {  
 string result = \_tp.InsertChar("hello", '!', 5);  
 Assert.That(result, Is.EqualTo("hello!"));  
 }  
  
 [Test]  
 public void InsertChar\_InsertInMiddle\_ReturnsCorrectString()  
 {  
 string result = \_tp.InsertChar("abcde", 'Z', 2);  
 Assert.That(result, Is.EqualTo("abZcde"));  
 }  
  
 [Test]  
 public void InsertChar\_EmptyString\_InsertAtZero\_ReturnsSingleChar()  
 {  
 string result = \_tp.InsertChar("", 'q', 0);  
 Assert.That(result, Is.EqualTo("q"));  
 }  
}