**Практическая работа №2**

**Тестирование «Белым ящиком»**

**Цель работы:** изучить метод тестирования «Белым ящиком»

**WHITE BOX TESTING** — это тестирование внутренней структуры, дизайна и кодирования программного решения. В этом типе тестирования код виден тестеру. Основное внимание уделяется проверке потока входных и выходных данных через приложение, улучшению дизайна и удобства использования, усилению безопасности. Тестирование белого ящика также известно, как тестирование Clear Box, тестирование Open Box, структурное тестирование, тестирование прозрачного бокса, тестирование на основе кода и тестирование Glass Box. Это обычно выполняется разработчиками.

**Метод белого ящика** (white box testing, open box testing, clear box testing, glass box testing) – у тестировщика есть доступ к внутренней структуре и коду приложения, а также есть достаточно знаний для понимания увиденного.

**Разработка тестов методом белого ящика (white-box test design technique):**Процедура разработки или выбора тестовых сценариев на основании анализа внутренней структуры компонента или системы.

**Пример тестирования WhiteBox**

Рассмотрим следующий фрагмент кода

Printme (int a, int b) { ------------ Printme is a function

int result = a+ b;

If (result> 0)

Print ("Positive", result)

Else

Print ("Negative", result)

} ----------- End of the source code

Целью тестирования WhiteBox является проверка всех ветвей решений, циклов, операторов в коде.

Чтобы выполнить утверждения в приведенном выше коде, тестовые случаи WhiteBox будут

* A = 1, B = 1
* А = -1, В = -3

**Техники, основанные на структуре, или методе белого ящика**

* тестирование операторов;
* тестирование альтернатив.

**Альтернатива (decision):** Точка программы, в которой управление имеет два или более альтернативных путей. Узел с двумя или более связями для разделения ветвей.

**Тестирование условий альтернатив (decision condition testing):** Разработка тестов методом белого ящика, при котором тестовые сценарии проектируются для исходов условий и результатов альтернатив.

**Покрытие (coverage):** Уровень, выражаемый в процентах, на который определенный элемент покрытия был проверен набором тестов.

**Покрытие альтернатив (decision coverage):** Процент результатов альтернативы, который был проверен набором тестов. Стопроцентное покрытие решений подразумевает стопроцентное покрытие ветвей и стопроцентное покрытие операторов.

**Покрытие кода (code coverage):** Метод анализа, определяющий, какие части программного обеспечения были проверены (покрыты) набором тестов, а какие нет, например, покрытие операторов, покрытие альтернатив или покрытие условий.

***ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ И***

***ФОРМА ОТЧЕТНОСТИ:***

**Задание 1**. Разработать программу на C#.

Даны длины сторон треугольника, определить вид треугольника и его площадь. Выполнить контроль вводимых чисел.

1. Разносторонний треугольник
2. Равнобедренный треугольник
3. Равносторонний треугольник

**Ограничения:**

- три числа не могут быть определены как стороны треугольника;  
- если хотя бы одно из них меньше или равно 0;  
- сумма двух из них меньше третьего;

- если введены символы.

**Задание 2.** Подготовить набор тестовых вариантов для обнаружения ошибок в программе.

Результат оформить в следующем виде:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| А | В | С | Ожидаемый результат | Объект проверки |
| Значение | Значение | Значение | Что должно получится | Значения вводимых данных, либо ожидаемый результат |
| … | … | … | … | … |

**Задание 3.** Разработать программу на C#.

Даны длины сторон треугольника, определить вид треугольника и его площадь. Выполнить контроль вводимых чисел.

1. Остроугольный треугольник
2. Тупоугольный треугольник
3. Прямоугольный треугольник

**Ограничения:**

- три числа не могут быть определены как стороны треугольника;  
- если хотя бы одно из них меньше или равно 0;  
- сумма двух из них меньше третьего.

Подготовить набор тестовых вариантов для обнаружения ошибок в программе и оформить результат.

**Задание 4.**На основании проведенных тестов составьте рекомендации по исправлению ошибок, выявленных в ходе тестирования в виде отчета.

*Пример:*

1 тест. В ходе проведения первого теста было обнаружено, что при в ведении не корректных данных площадь все равно высчитывается.

Рекомендуется: в случае, если пользователь введет не корректные данные, следует выводить сообщение с просьбой исправить введенные значения. Добавить в программу проверку введенных значений на соответствие ограничения.