Отчёт  
по практической работе № 11  
по дисциплине «МДК 02.02 Инструментальные средства разработки программного обеспечения»  
Тема: «Выполнение функционального тестирования»

|  |  |  |
| --- | --- | --- |
| Студент гр. 22 ИТ |  | Липинский К.С. |
| Преподаватель |  | Быковский З.С. |

### Цель работы

Изучение вопросов, связанных с функциональным тестированием.

### Выполнение работы

### Задание 1 (3)

Программа для проверки паролей.

Входные данные: Пароль (строка)

Выходные данные: Логическое значение (True или False), указывающее, соответствует ли пароль требованиям

|  |  |  |
| --- | --- | --- |
| **Входное условие** | **Правильные классы эквивалентности** | **Неправильные классы эквивалентности** |
| Пароль (строка) | Строка, длиной 8 символов, содержащая заглавные и прописные буквы, цифры. | Пароль длиной более 8 символов.  Пароль длиной менее восьми символов.  Пароль без заглавных букв.  Пароль без прописных букв.  Пароль без цифр.  Пароль с недопустимыми символами. |

Тестовый вариант для каждого класса эквивалентности

|  |  |  |
| --- | --- | --- |
| **Входное значение** | **Класс** | **Ожидаемый результат** |
| 12GHjk89 | Правильный | True |
| Kasofjkde | Неправильный | False |
| 11111Hj8 | Правильный | True |
| dfdsj132 | Неправильный | False |

Код программы (рис. 1)

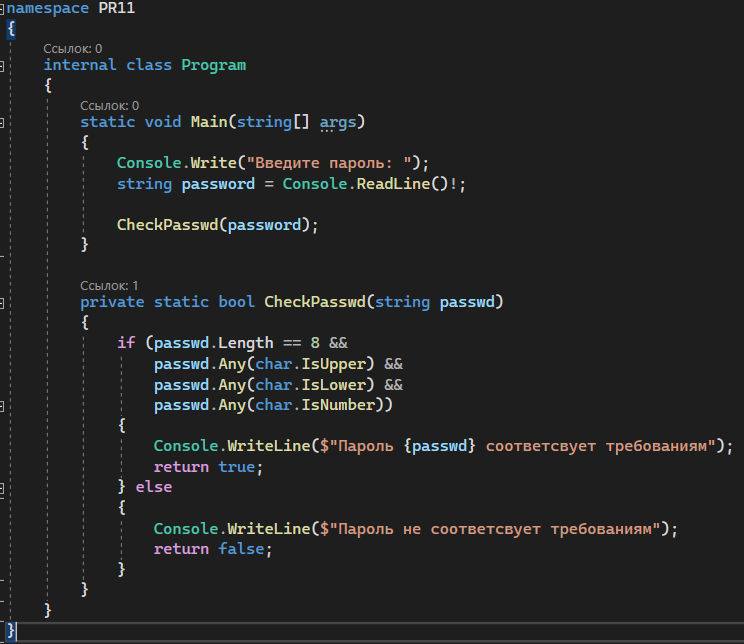


Рисунок 1

Результаты выполнения программы (рис. 2 и рис. 3)



Рисунок 2



Рисунок 3

### Задание 2 (2)

1. Реализация анализа граничных значений для входных данных:=

* Установление минимальной границы: имя должно содержать не менее 6 символов.
* Установление максимальной границы: имя должно содержать не более 12 символов.
* Определение допустимых данных: имя должно состоять из 6-12 букв кириллицы.
* Определение недопустимых данных: имя содержит меньше 6 или больше 12 букв, либо включает буквы других алфавитов или их комбинации.

1. Создание тестовых сценариев:

* Имя длиной 6 символов (минимальная граница):

Ввод: Имя = "Александр"

Ожидаемый результат: Успешное принятие ввода.

* Имя длиной 12 символов (максимальная граница):

Ввод: Имя = "Александрина"

Ожидаемый результат: Успешное принятие ввода.

* Имя длиной 5 символов (ниже минимальной границы):

Ввод: Имя = "Анна"

Ожидаемый результат: Ошибка ввода.

* Имя длиной 13 символов (выше максимальной границы):

Ввод: Имя = "Александринина"

Ожидаемый результат: Ошибка ввода.

* Имя длиной 7 символов (в пределах допустимого диапазона):

Ввод: Имя = "Марина"

Ожидаемый результат: Успешное принятие ввода.

* Имя длиной 11 символов (в пределах допустимого диапазона):

Ввод: Имя = "Елизавета"

Ожидаемый результат: Успешное принятие ввода.

* Пустое поле (крайний случай недопустимого ввода):

Ввод: Имя = ""

Ожидаемый результат: Ошибка ввода.

* Имя длиной 1 символ (крайний случай недопустимого ввода):

Ввод: Имя = "А"

Ожидаемый результат: Ошибка ввода.

### Задание 3 (1)

1. Формирование классов эквивалентности исходных данных:
   * + Правильные классы эквивалентности: нечетные числа в диапазоне от -18 до 18.
     + Неправильные классы эквивалентности исходных данных:
   * Числа, которые не являются целыми;
   * Числа, которые не находятся в диапазоне от -18 до 18;
   * Буквы, символы и другие не числовые значения.
2. Разработка тестовых вариантов для каждого класса эквивалентности:

* Правильные тесты: Произведение всех нечетных числе в диапазоне от -18 до 18;
* Неправильные тесты:
  + Ввод чисел вне диапазона;
  + Ввод нецелых чисел;
  + Ввод символов;

Программа (рис. 4)



Рисунок

Тестирование программы (таблица 1)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № теста | Входные данные (диапазон) | Ожидаемый результат | Фактический результат | Результат тестирования | Предложение по исправлению найденных ошибок |
| 1 | -18 до 18 | Произведение всех нечетных чисел |  | Успешно | Нет |
| 2 | -20 до 20 | Ошибка: вне диапазона |  | Успешно | Нет |
| 3 | -18 до 18,5 | Ошибка: нецелое число |  | Успешно | Нет |
| 4 | -18 до а | Ошибка ввода |  | Успешно | Нет |
| 5 | -10 до 10 | Произведение всех нечетных чисел |  | Успешно | Нет |
| 6 | 0 до 0 | Ошибка: нет нечетных чисел |  | Успешно | Нет |

Таблица 1

### Контрольные вопросы

1. **Что понимается под функциональным тестированием?**

Функциональное тестирование — это процесс проверки программного обеспечения для убедительности в том, что оно соответствует функциональным требованиям и спецификациям. Оно направлено на проверку функций или характеристик программы, таких как корректность вычислений, обработка данных и взаимодействие с пользователем.

1. **В чём сущность тестирования «черного ящика» (blackbox)?**

Тестирование «черного ящика» фокусируется на проверке функциональности программного обеспечения без учета его внутренней структуры или реализации. Тестировщик взаимодействует с программой через её интерфейсы и проверяет, соответствует ли её поведение ожидаемым результатам.

1. **В чём сущность тестирования «белого ящика» (whitebox)?**

Тестирование «белого ящика» включает проверку внутренней логики программы, такой как код, структуры данных и алгоритмы. Оно требует доступа к исходному коду и направлено на выявление скрытых ошибок, таких как неинициализированные переменные или ошибки в логике.

1. **В чём сущность тестирования «серого ящика» (gray-box)?**

Тестирование «серого ящика» сочетает элементы тестирования «черного» и «белого» ящиков. Тестировщик имеет частичное представление о внутренней структуре программы, что позволяет ему разрабатывать более точные тесты, чем в случае «черного ящика», но не требует полного понимания кода.

1. **Методы отбора тестов для black-box тестирования.**

* Тестирование классов эквивалентности: Разделение входных данных на группы, которые считаются эквивалентными по поведению.
* Анализ граничных значений: Проверка граничных значений входных данных, где часто возникают ошибки.
* Таблицы решений: Использование таблиц для определения логических условий и соответствующих действий.
* Тестирование на основе сценариев использования (use cases).

1. **Тестирование сценариев использования юз-кейсов (usecases).**

Тестирование сценариев использования включает проверку программы на основе реальных сценариев её использования. Это помогает убедиться, что программа работает корректно в условиях, близких к реальным условиям эксплуатации.

1. **Тестирование классов эквивалентности.**

Тестирование классов эквивалентности заключается в разделении входных данных на группы, которые ожидается будут обработаны программой одинаково. Это позволяет сократить количество тестов, сосредоточившись на представителях каждого класса.

1. **В чём сущность попарного тестирования?**

Попарное тестирование — это метод, при котором тестируются все возможные пары входных параметров. Это помогает выявить ошибки, которые могут возникнуть при взаимодействии различных параметров, и позволяет значительно сократить количество тестов по сравнению с исчерпывающим тестированием всех комбинаций.

### Вывод

В ходе выполнения лабораторной работы были изучены основные принципы функционального тестирования, включая методы разбиения по эквивалентности и таблицы решений. Были разработаны и выполнены тесты для простого консольного приложения на C#, которое вычисляет произведение нечетных чисел в заданном диапазоне.