Лабораторная работа №4: Исправление ошибок в коде программы после выполнения тест-кейсов

Цель работы:

- Научиться выявлять и исправлять ошибки в коде программы на основе результатов выполнения тест-кейсов.
- Закрепить навыки работы с отладчиком, анализа логов и написания корректного кода.

Задние:

1. Выполнить тест-кейсы для программы по вашему варианту.
2. Выявить ошибки на основе результатов тестирования.
3. Исправить ошибки в коде программы.
4. Повторно протестировать программу, чтобы убедиться в устранении ошибок.

Оборудование и инструменты:

- Среда разработки Visual Studio.
- Отладчик.
- Система управления версиями Git.
- Тест-кейсы, подготовленные на основе функциональных требований.

Этапы выполнения:

1. Подготовка

- Ознакомьтесь с функциональными требованиями к программе.
- Изучите исходный код программы.
- Подготовьте тест-кейсы для проверки функциональности программы.

---

2. Выполнение тест-кейсов

- Запустите программу.
- Выполните тест-кейсы, описанные в предыдущей лабораторной работе и зафиксируйте результаты.

---

3. Анализ результатов тестирования

- Определите, какие тест-кейсы завершились с ошибками.
- Используйте отладчик для поиска проблемных участков кода.

---

4. Исправление ошибок

- Внесите изменения в код программы, чтобы устранить выявленные ошибки.
- Проверьте, что изменения не нарушают другие части программы.

---

5. Повторное тестирование

- Повторно выполните тест-кейсы, чтобы убедиться, что ошибки исправлены.
- Зафиксируйте результаты повторного тестирования.

Пример выполнения лабораторной работы

Исходный код программы :

```
public class Product
{
   public string Name { get; set; }
   public decimal Price { get; set; }
   public int Quantity { get; set; }

   public Product(string name, decimal price, int quantity)
   {
      Name = name;
      Price = price;
      Quantity = quantity;
   }

   public override string ToString()
   {
      return $"Товар: {Name}, Цена: {Price}, Количество: {Quantity}";
   }

   public bool IsAvailable()
   {
      return Quantity > 0;
   }
}
```

Тест-кейсы и выявленные ошибки:

1. **TC-001: Создание товара с корректными данными**
    - Ошибка: Нет проверки на пустое название товара.
2. **TC-002: Создание товара с нулевой ценой**
    - Ошибка: Программа допускает создание товара с нулевой ценой.
3. **TC-003: Создание товара с отрицательным количеством**
    - Ошибка: Программа допускает создание товара с отрицательным количеством.

Исправленный код программы:

```
public class Product
{
    public string Name { get; set; }
    public decimal Price { get; set; }
    public int Quantity { get; set; }

    public Product(string name, decimal price, int quantity)
    {
        if (string.IsNullOrEmpty(name))
            throw new ArgumentException("Название товара не может быть пустым.");

        if (price <= 0)
            throw new ArgumentException("Цена должна быть положительной.");

        if (quantity < 0)
            throw new ArgumentException("Количество не может быть отрицательным.");

        Name = name;
        Price = price;
        Quantity = quantity;
    }

    public override string ToString()
    {
        return $"Товар: {Name}, Цена: {Price}, Количество: {Quantity}";
    }

    public bool IsAvailable()
    {
        return Quantity > 0;
    }
}
```

Результаты повторного тестирования:

1. **ТС-001: Создание товара с корректными данными**
    - Результат: Товар успешно создан. Ошибок нет.
2. **ТС-002: Создание товара с нулевой ценой**
    - Результат: Программа выдает ошибку: "Цена должна быть положительной."
3. **ТС-003: Создание товара с отрицательным количеством**
    - Результат: Программа выдает ошибку: "Количество не может быть отрицательным."

Выводы:

- В ходе лабораторной работы были выявлены и исправлены ошибки в коде программы.
- После исправления кода все тест-кейсы выполняются корректно.
- Программа теперь соответствует функциональным требованиям.

Вопросы для самоконтроля:

1. Какие ошибки были выявлены в ходе тестирования?
2. Какие изменения были внесены в код для исправления ошибок?
3. Как проверить, что исправления не нарушили другие части программы?
4. Какие инструменты помогли в поиске и исправлении ошибок?

Варианты

1. Управление банковским счётом
2. Конвертер валют
3. Управление задачами
4. Календарь событий
5. Управление книгами в библиотеке
6. Управление заметками
7. Управление контактами
8. Управление покупками
9. Управление задачами с приоритетом
10. Управление продажами
11. Управление инвентарём
12. Управление заказами
13. Управление сотрудниками
14. Управление проектами
15. Управление клиентами

16. Управление доставкой
17. Управление отчётами
18. Управление резервированием
19. Управление бюджетом
20. Управление файлами и папками
21. Управление системой безопасности данных
22. Управление здоровьем
23. Управление музыкальной коллекцией
24. Управление путешествиями
25. Управление фотографиями
26. Управление рецептами и планированием меню
27. Управление обучением и курсами
28. Управление резюме и поиском работы
29. Управление встречами и мероприятиями
30. Управление техникой и оборудованием
31. Управление автомобилем и его обслуживанием
32. Управление энергопотреблением
33. Управление личными целями и задачами
34. Управление коллекцией игр
35. Управление доставкой