Лабораторная работа №5
Отладка отдельных модулей программного проекта

1 Цель работы

1.1Изучить процесс отладки приложений,

1.2Изучить процесс применения отладочных классов.

2 Литература

2.1 Гагарина, Л. Г. Технология разработки программного обеспечения : учебное пособие / Л.Г. Гагарина, Е.В. Кокорева, Б.Д. Сидорова-Виснадул ; под ред. Л.Г. Гагариной. — Москва : ФОРУМ : ИНФРА-М, 2022. — 400 с. — (Среднее профессиональное образование). – URL: https://znanium.com/catalog/product/1794453 . – Режим доступа: по подписке. – Текст : электронный. – гл.5.

3 Подготовка к работе

3.1 Повторить теоретический материал (см. п.2).

3.2 Изучить описание лабораторной работы.

4 Основное оборудование

4.1 Персональный компьютер.

5 Задание

5.1 Основы отладки и установка брейкпоинтов

5.1.1 Создать консольное приложение C# в Visual Studio, которое выводит сумму двух чисел, вводимых пользователем.

5.1.2 Установить брейкпоинты в строках, где происходит ввод данных и выполняется сложение чисел. Запустить отладку и убедиться, что выполнение программы останавливается на брейкпоинтах. Использовать панель "Locals" и "Watch" для анализа значений переменных во время выполнения программы.

5.1.3 Изменить программу, добавив цикл, который запрашивает сумму двух чисел, пока пользователь не введет "exit". Установить условный брейкпоинт на строке с выводом результата сложения. Настроить брейкпоинт так, чтобы программа останавливалась только если результат сложения больше 100. Запустить отладку, вводя различные значения, чтобы проверить, срабатывает ли брейкпоинт только при выполнении условия.

5.2 Использование трассировки и отладочного вывода

5.2.1 Добавить в программу из п.5.1 дополнительные сообщения, используя Debug.WriteLine и Trace.WriteLine.

Например: добавить Debug.WriteLine перед вводом значений, перед сложением и перед выводом результата.

5.2.2 Настроить проект так, чтобы вывод Debug.WriteLine работал только в конфигурации Debug, а Trace.WriteLine — в конфигурации Release. Переключить конфигурацию на Release и убедиться, что вывод Debug больше не отображается, а сообщения Trace остаются видимыми.

5.3 Использование Debug.Assert

5.3.1 Создать метод CalculateDiscount для расчета скидки в процентах, который принимает параметры price и discountRate и возвращает итоговую сумму.

5.3.2 Использовать Debug.Assert, чтобы проверить, что:

- параметр price больше нуля.
- параметр discountRate находится в диапазоне от 0 до 1.
- возвращаемое значение всегда меньше или равно price.

5.3.3 Вызвать метод с корректными и некорректными параметрами, чтобы увидеть, как Debug.Assert проверяет данные и реагирует на ожидания.

5.3.4 Запустить проект в режиме Debug и убедиться, что при некорректных значениях программа останавливается на ошибке.

5.4 Использование Call Stack и исследование стеков вызовов

5.4.1 Написать программу, которая вызывает несколько методов последовательно, например: Main -> MethodA -> MethodB -> MethodC.

5.4.2 В одном из методов (MethodC) добавить ошибку, например, выброс исключения DivideByZeroException.

5.4.3 Установить брейкпоинт перед вызовом MethodC и начать отладку. Когда программа остановится, открыть окно Call Stack (позволяет видеть последовательность вызовов методов) и проанализировать последовательность вызовов, которая привела к текущему методу.

5.4.4 Добавить обработку исключения в MethodC, используя блок try-catch, и залогировать стек вызовов (ex.StackTrace) в текстовый файл.

5.5 Отладка JavaScript

5.5.1 Пошаговая отладка:

Перейти на вкладку Sources и найти строку в коде, где выводится лог (или добавить его). Установить точку останова на этой строке, нажав рядом с номером строки.

Выполнить действие, которое приводит к выполнению кода с точкой останова, и начните пошаговую отладку, используя Step Over (F10), Step Into (F11) и Step Out (Shift+F11) для анализа пошагового выполнения кода.

Изучить текущие значения переменных во время выполнения кода, используя вкладку Scope.

5.5.2 Использование условной точки останова:

Добавить счетчик нажатий для кнопки:

```
let clickCount = 0;
document.querySelector('.button-class').addEventListener('click', () => {
```

```
    clickCount++;
    console.log("Кнопка нажата", clickCount, "раз");
});
```

Поставить точку останова с условием clickCount >= 3.

Нажать кнопку и изучить, что точка останова срабатывает только на третьем нажатии.

Добавить в JavaScript-файл цикл for, который выводит в консоль числа от 1 до 100.

Установить условный брейкпоинт на строке с выводом, чтобы он срабатывал только когда число делится на 10.

Запустить скрипт и проверить, что выполнение кода останавливается только на числах, кратных 10.

Использовать консоль для анализа значения переменной на каждой итерации, когда срабатывает брейкпоинт.

5.5.3 Настройка вывода разных типов сообщений:

Реализовать в консоли вывод сообщений, используя console.log для вывода информации, console.error для вывода ошибки и console.warn для предупреждения.

6 Порядок выполнения работы

6.1 Выполнить задание из п.5.

6.2 Ответить на контрольные вопросы.

7 Содержание отчета

7.1 Титульный лист

7.2 Цель работы

7.3 Ответы на контрольные вопросы

7.4 Вывод

8 Контрольные вопросы

8.1 Что такое «отладка»?

8.2 Какие этапы включает в себя отладка?

8.3 Что такое «точка останова»?

8.4 Какие отладочные классы могут использоваться в проекте?

8.5 Для чего могут потребоваться классы трассировки?