Лабораторная работа №7
Отладка проекта

1 Цель работы

1.1 Изучить процесс отладки приложений,

1.2 Научиться применять логирование в ходе отладки.

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 Логирование исключений

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

5.1.2 Обернуть код сложения в блок try-catch, который будет перехватывать исключения разных типов (например, FormatException). При возникновении исключения вывести в консоль сообщение об ошибке и текст исключения.

5.1.3 Добавить логирование исключений в текстовый файл log.txt, где будет сохраняться информация об исключении (текст исключения и время его возникновения).

5.1.4 Проверить, что логирование срабатывает, если пользователь вводит некорректные данные, например, текст вместо числа или числа, которые в результате дадут переполнение.

5.2 Применение логирования для отслеживания времени выполнения операций

5.2.1 Создать программу, которая выполняет несколько операций, например, чтение и обработку данных из файла или выполнение сложных математических расчетов.

5.2.2 Перед началом каждой операции запустить Stopwatch, а после завершения остановить его.

Stopwatch — удобный инструмент для измерения времени выполнения операций, особенно полезный в отладке производительности:

```
Stopwatch stopWatch = new Stopwatch();
stopWatch.Start();
… // длительная операция
stopWatch.Stop();
TimeSpan ts = stopWatch.Elapsed; // получение прошедшего времени
```

5.2.3 Использовать Trace.WriteLine или Debug.WriteLine, чтобы записать в лог время, потраченное на каждую операцию. Вывести в конце общее время выполнения всех операций.

5.2.4 Проанализировать лог и сделать выводы, какие операции выполняются дольше других.

5.3 Применение Trace.Listeners для сохранения информации в файл

5.3.1 Создать консольное приложение, которое выполняет несколько простых операций, например, сложение, вычитание и деление.

5.3.2 Добавить TextWriterTraceListener в Trace.Listeners, чтобы записывать все сообщения трассировки в файл trace.log.

```
Trace.Listeners.Add(new TextWriterTraceListener(
    "trace.log", "myListener"));
Trace.TraceInformation("Test message.");
```

5.3.3 Применить Trace.WriteLine для логирования операций в файл, включая данные о входных параметрах, выполненных операциях и результатах. По завершении выполнения программы убедиться, что файл trace.log содержит все сообщения о выполнении операций.

5.3.4 Отключить логирование файла после завершения работы, используя Trace.Close() или Trace.Flush().

```
Trace.Flush(); // закрыть/очистить трассировку, для очистки буфера
```

5.4 Использование TraceSwitch для централизованного управления уровнем детализации логирования

5.4.1 Создать приложение, которое выполняет операции, такие как загрузка, сохранение и удаление данных, и добавить подробные сообщения в Trace.WriteLine.

5.4.2 Настроить TraceSwitch с уровнями (Off, Error, Warning, Info, Verbose) и использовать его для управления уровнем детализации логирования.

5.4.3 Настроить TraceSwitch так, чтобы при запуске логировались только сообщения с уровня Warning и выше. Изменить уровень логирования на Verbose и убедиться, что теперь логируются все сообщения.

5.4.4 Запустить приложение и проанализировать, как изменяется количество выводимых сообщений при разных уровнях детализации.

6 Порядок выполнения работы

6.1 6.1Выполнить 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 Как правильно организовать обработку исключений при вводе пользовательских данных и записывать информацию об ошибках в лог-файл?

8.2 Какие методы класса Stopwatch следует использовать для измерения времени выполнения операций?

8.3 Как настроить Trace.Listeners для записи отладочной информации в файл и какие данные следует включать в логи при выполнении операций?

8.4 Какие уровни детализации предоставляет TraceSwitch и как его использовать для гибкого управления логированием в зависимости от потребностей?

8.5 В чем преимущества использования Trace.WriteLine по сравнению с простой записью в файл?