Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Институт информационных технологий

Кафедра ИС

# ОТЧЕТ

по лабораторной работе №6

ИССЛЕДОВАНИЕ СПОСОБОВ ПРОФИЛИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Выполнил:

ст. гр. ИС/б-21-2-о

Мовенко К. М.

Проверил:

Петраков В. А.

Севастополь

2023

# ЦЕЛЬ РАБОТЫ

Исследовать критические по времени выполнения участки программного кода и возможности их устранения. Приобрести практические навыки анализа программ с помощью профайлера.

# ЗАДАНИЕ

* 1. Разработать программу на основе библиотеки классов, реализованной и протестированной в предыдущей работе. Программа должна как можно более полно использовать функциональность класса. При необходимости для наглядности профилирования в методы класса следует искусственно внести задержку выполнения;
  2. Выполнить профилирование разработанной программы, выявить функции, на выполнение которых тратится наибольшее время;
  3. Модифицировать программу с целью оптимизации времени выполнения;
  4. Выполнить повторное профилирование программы, сравнить новые результаты и полученные ранее, сделать выводы;

# ХОД РАБОТЫ

Написанная в предыдущей работе библиотека классов была преобразована в консольное приложения, для которого был написан класс MainClass. Его метод Main вызывается при запуске исполняемого файла, в нём происходит поочерёдный вызов методов класса StringTransformer. Также в метод ReverseString была введена искусственная задержка – 7 секунд.

namespace LAB\_6

{

internal class MainClass

{

public static void Main()

{

String s = "AaAaAaAa";

StringTransformer transformer = new StringTransformer();

Console.WriteLine(transformer.ReverseString(s));

Console.WriteLine(transformer.ToLowerCase(s));

Console.WriteLine(transformer.ToUpperCase(s));

}

}

}

Поскольку профайлер EQATEC, рекомендованный в методических указаниях, не поддерживает современные версии .NET, для профилирования была выбрана программа dotTrace.

Программа была отлажена и скомпилирована в исполняемый exe-файл. Данный файл был выбран для профилирования. В результате был создан отчёт, в котором было построено дерево вызовов для метода Main (Рисунок 1).

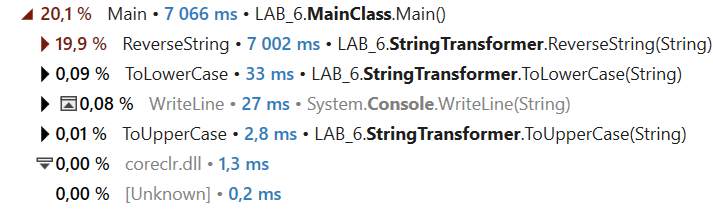


Рисунок 1 – Дерево вызовов программы

На дереве видно, что наиболее затратным по времени выполнения является вызов метода ReverseString класс StringTransformer – того, что был искусственно замедлен. Было открыто его внутреннее дерево вызовов.

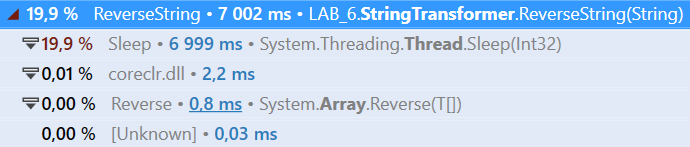


Рисунок 2 – Дерево вызовов метода ReverseString

В нём видно, что основная нагрузка приходится на потоковый метод Sleep, занимающий 6999 миллисекунд – как раз почти 7 секунд. После этого из программы была убрана искусственная задержка и она была спрофилирована вновь (Рисунок 3). Без искусственной задержки метод ReverseString оказался наиболее быстродействующим.

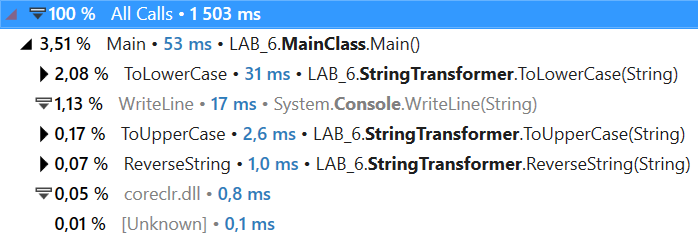


Рисунок 3 – Обновлённое дерево вызовов программы

# ВЫВОД

В ходе работы было проведено ознакомление с процессом профилирования разрабатываемого программного обеспечения. Были получены навыки анализа эффективности модулей программ на платформе .NETс помощью профилировщика dotTrace. Полученные навыки позволяют определять наиболее и наименее эффективные модули и операции программ, искать и исправлять «узкие горлышки».