Отчёт по лабораторной работе №4

Жуков Вадим, ИВТ-12М

Технические характеристики

OС: Windows 10

VS: Visual Studio 2017 v15.9.18

IPS: v2019

Железо: 2 физических ядра, 4 логических, 3.4ГГц

# Задание 1

Разберите программу, представленную в файле task\_for\_lecture5.cpp. В программе создается 2 потока, каждый из которых вычисляет средние значения матрицы, один по строкам исходной матрицы ***matrix***, а другой - по столбцам.

Программа работает корректно, результат показан на рисунке 1. Также я немного изменил вывод, чтобы далее было более наглядно.

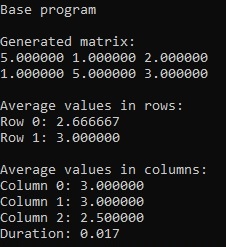


Рисунок 1. Результат работы 1 версии

# Задание 2

Проанализируйте программу и введите в нее изменения, которые, по Вашему мнению, повысят ее производительность.

Проанализировав программу, я решил попробовать улучшить её производительность используя cilk\_for вместо обычного for. Программа стала работать в 3 раза быстрее, чем в первом случае. Результат показан на рисунке 2.

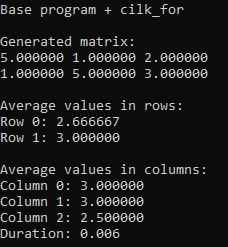


Рисунок 2. Результат работы улучшенной версии

# Задание 3

Определите с помощью ***Intel Parallel Inspector*** наличие в программе таких ошибок как: взаимная блокировка, гонка данных, утечка памяти. Сделайте скрины результатов анализа ***Parallel Inspector***(вкладки **Summary**, **Bottom-up**) для всех упомянутых ошибок, где отображаются обнаруженные ошибки, либо отражается их отсутствие. Запускайте анализы на разных уровнях (**Narrowest**, **Medium**, **Widest**).

Гонок данных и дедлоков не обнаружено, что видно на рисунке 3.

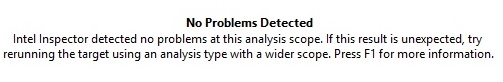


Рисунок 3. Результат поиска гонки данных и дедлоков

На 142 строке была обнаружена утечка памяти, что видно на рисунке 4.

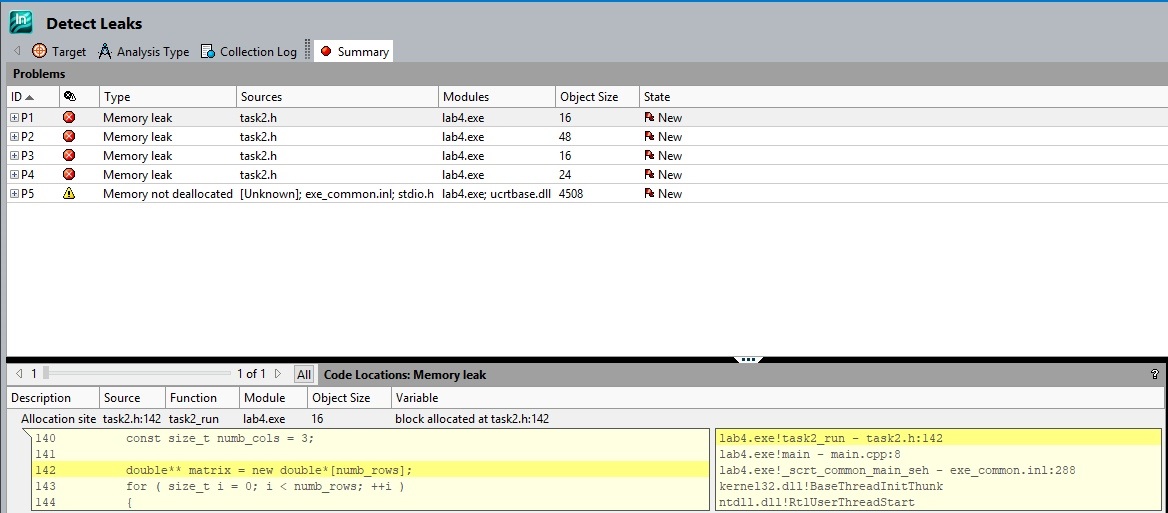


Рисунок 4. Результат поиска утечки памяти

# Задание 4

Измените код программы таким образом, чтобы **Inspector** при проверке не находил в программе ошибок, перечисленных в **п. 3.**Сделайте скрины результатов запуска ***Parallel Inspector***.

После добавления очистки выделенной памяти, выявленной в пункте 3, ошибки не стало, что видно на рисунке 5.

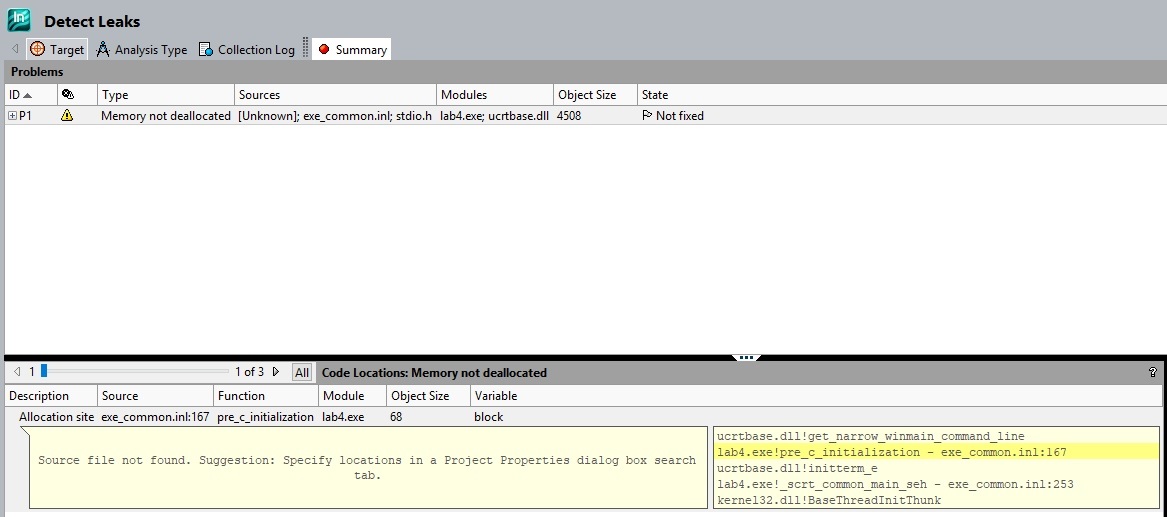


Рисунок 5. Результат поиска утечки памяти после исправления

Также я ещё раз запустил программу для измерения времени. В результате получилась ещё более производительная программа, в которой устранена утечка памяти. Время работы видно на рисунке 6.

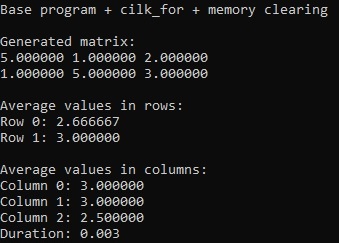


Рисунок 6. Конечный результат