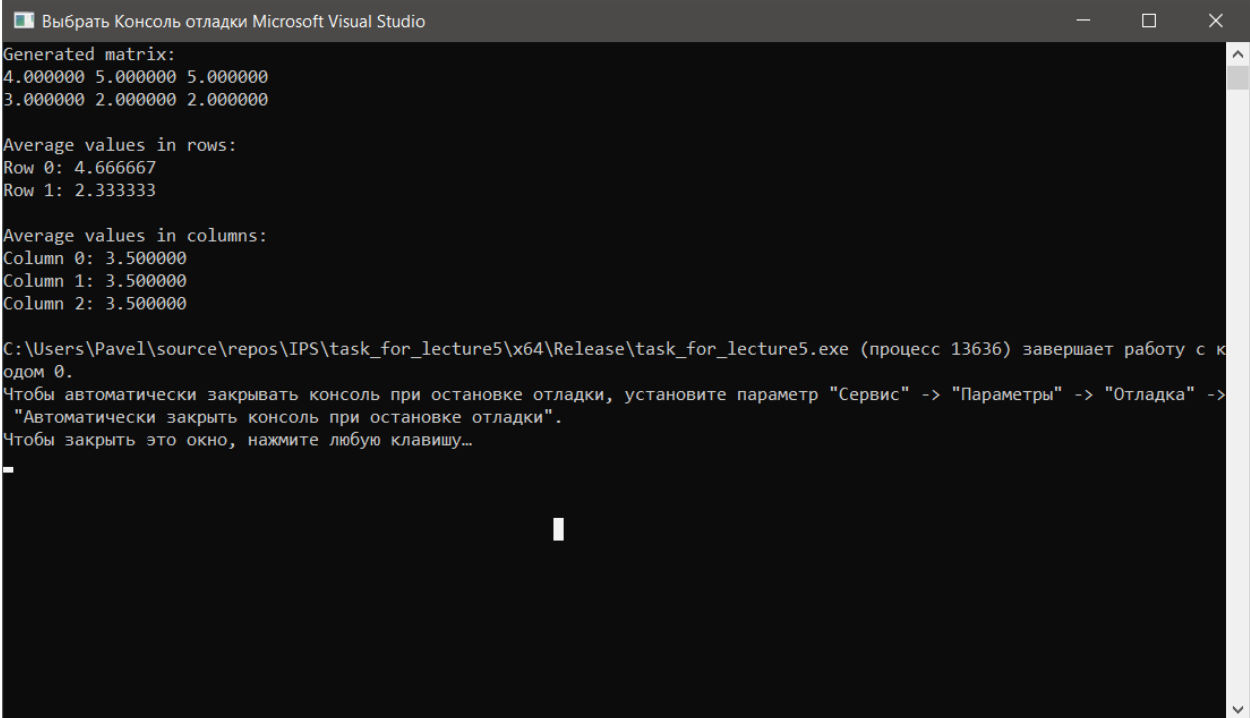


Программное обеспечение, использованное при выполнении: Visual Studio 2017, IPS 2019, Windows 10 – 64x

Процессор – четырехъядерный Intel Core i5 8250U с частотой 1.6 ГГц

1) Разберите программу, представленную в файле `task_for_lecture5.cpp`.

Предварительный просмотр документа. В программе создается 2 потока, каждый из которых вычисляет средние значения матрицы, один по строкам исходной матрицы `matrix`, а другой - по столбцам. Запустите программу и убедитесь в ее работоспособности.



```
Выбрать Консоль отладки Microsoft Visual Studio
Generated matrix:
4.000000 5.000000 5.000000
3.000000 2.000000 2.000000

Average values in rows:
Row 0: 4.666667
Row 1: 2.333333

Average values in columns:
Column 0: 3.500000
Column 1: 3.500000
Column 2: 3.500000

C:\Users\PaveI\source\repos\IPS\task_for_lecture5\x64\Release\task_for_lecture5.exe (процесс 13636) завершает работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, установите параметр "Сервис" -> "Параметры" -> "Отладка" -> "Автоматически закрыть консоль при остановке отладки".
Чтобы закрыть это окно, нажмите любую клавишу...
```

Рисунок 1 – результат работы программы при первом запуске

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

Модифицируем программу, добавив циклы **cilk_for** для распараллеливания и необходимые **reducer** для суммирования:

```
case eprocess_type::by_rows:
{
    cilk_for ( size_t i = 0; i < numb_rows; ++i )
    {
        cilk::reducer<cilk::op_add<double>> sum(0.0);
        cilk_for(size_t j = 0; j < numb_cols; ++j)
        {
            *sum += matrix[i][j];
        }
        average_vals[i] = sum.get_value() / numb_cols;
    }
    break;
}
case eprocess_type::by_cols:
{
    cilk_for ( size_t j = 0; j < numb_cols; ++j )
    {
        cilk::reducer<cilk::op_add<double>> sum(0.0);
        cilk_for(size_t i = 0; i < numb_rows; ++i)
```

```

        {
            *sum += matrix[i][j];
        }
        average_vals[j] = sum.get_value() / numb_rows;
    }
    break;
}

```

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**).

- Запустим Intel Inspector на выявление взаимных блокировок и гонок данных

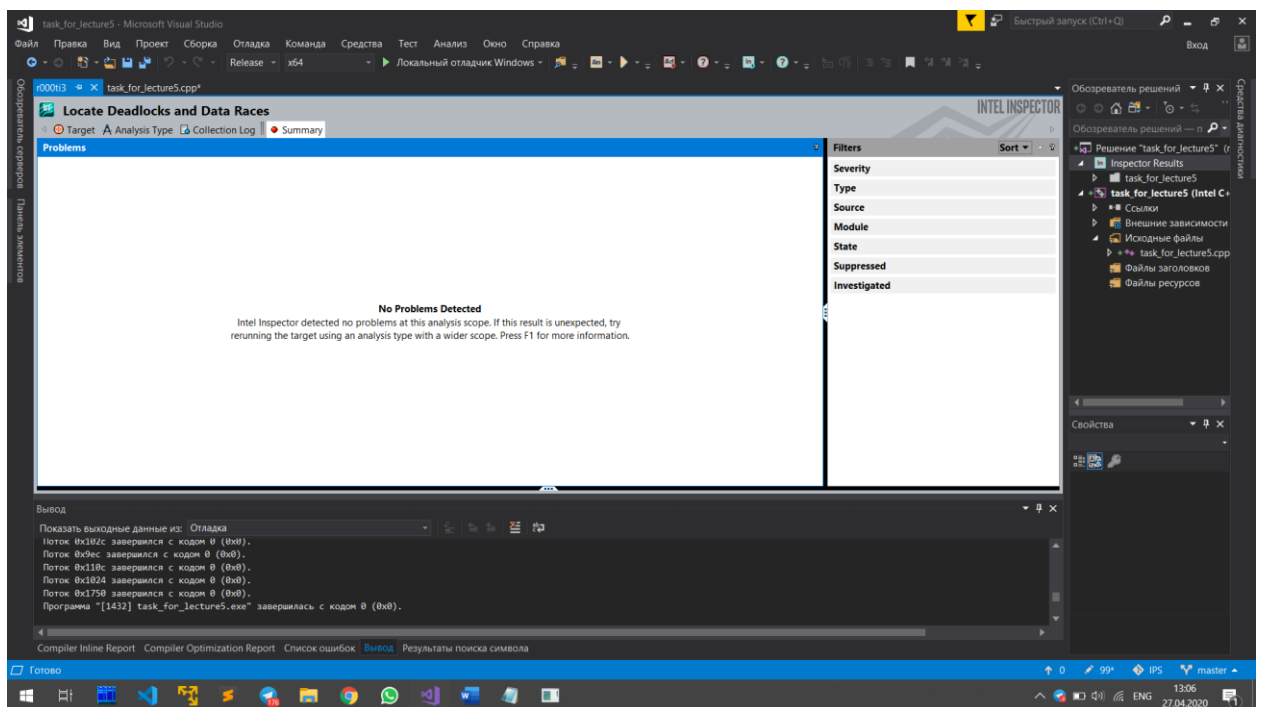


Рисунок 2 – результат работы Inspector на наличие гонок данных

Проблем не обнаружено

- Запустим Intel Inspector на выявление утечек данных

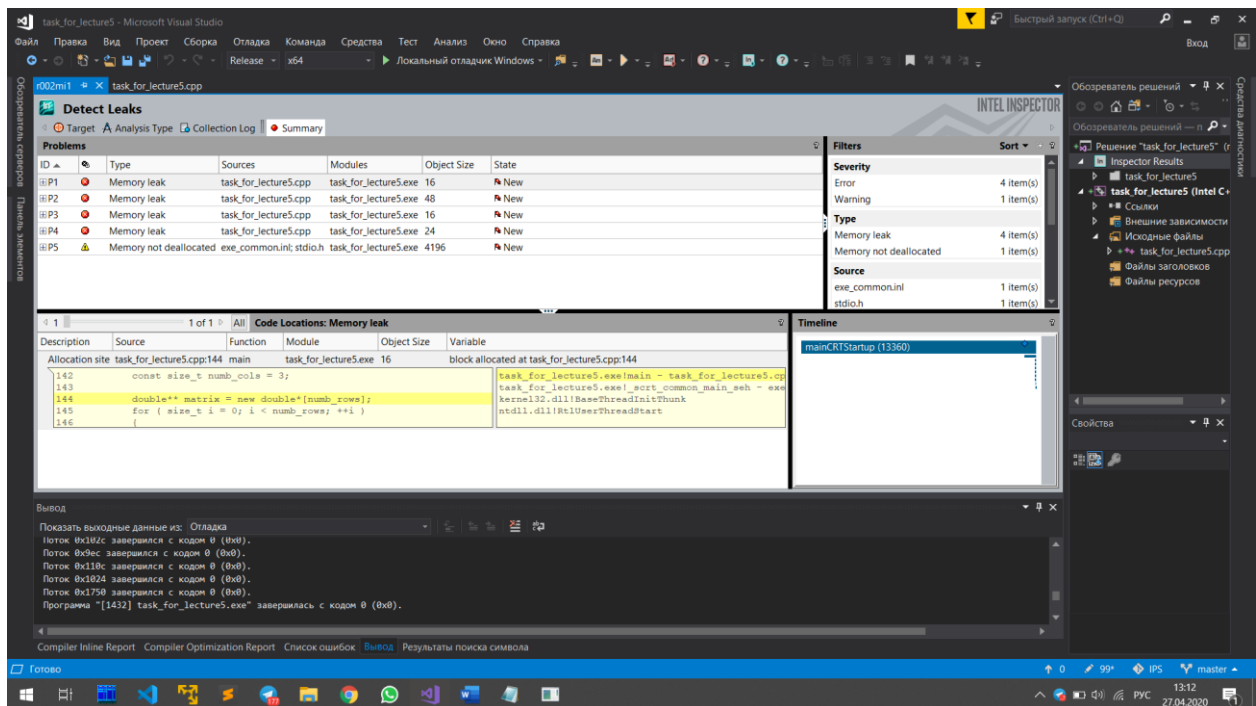


Рисунок 3 – результат работы Inspector на наличие утечек памяти

Как можно увидеть на скриншоте обнаружены утечки памяти, скорее всего разработчик забыл освободить выделенную под данные память. Исправим эту ошибку в следующем пункте.

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**.

Внесем следующие изменения в конец программы:

```
for (size_t i = 0; i < numb_rows; ++i)
    delete[] matrix[i];
delete[] matrix;
delete[] average_vals_in_rows;
delete[] average_vals_in_cols;
```

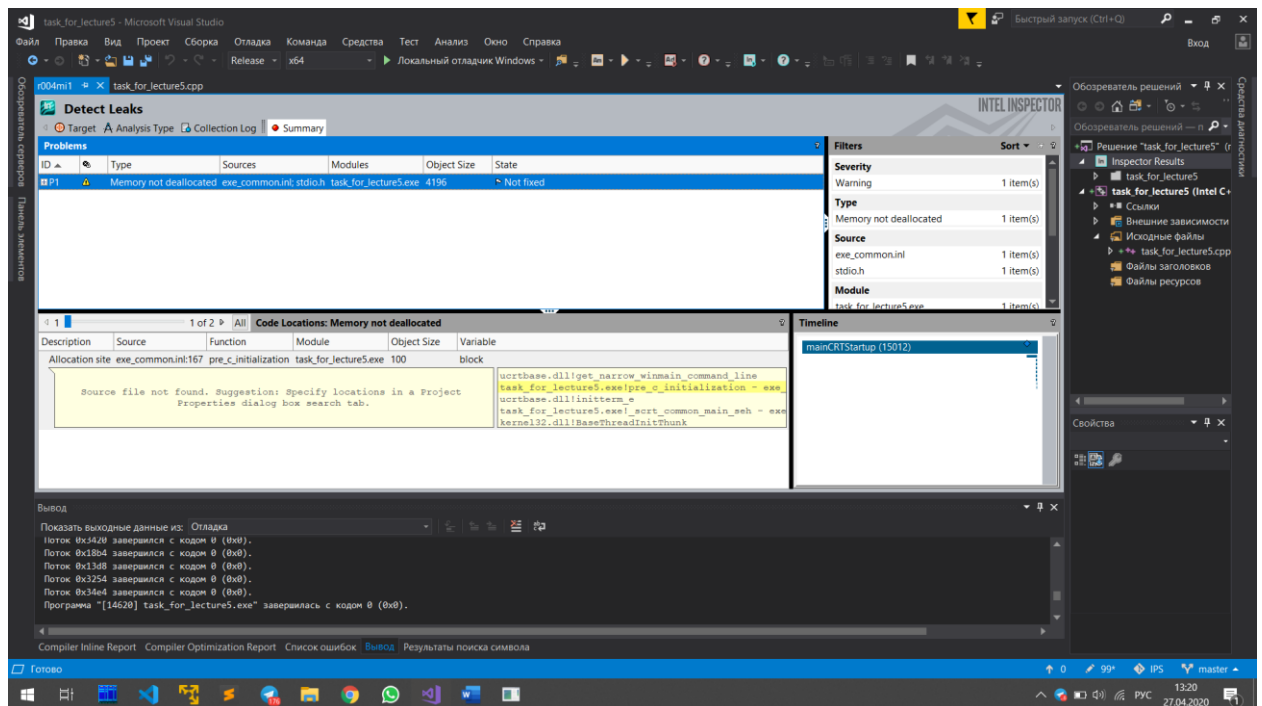


Рисунок 4 – результат работы Inspector на наличие утечек памяти

Как видно на скриншоте ошибки теперь отсутствуют.

Вывод: таким образом, в рамках данной лабораторной работы я проверил работоспособность программы, которая вычисляет средние значения матрицы по строкам и столбцам, убедился в ее работоспособности. Модифицировал программу, добавив `cilk_for` и `reducer`, повысив ее производительность. Произвел анализ на предмет гонок данных и утечек памяти. Обнаружил и исправил ошибку, связанную с наличием утечек памяти.