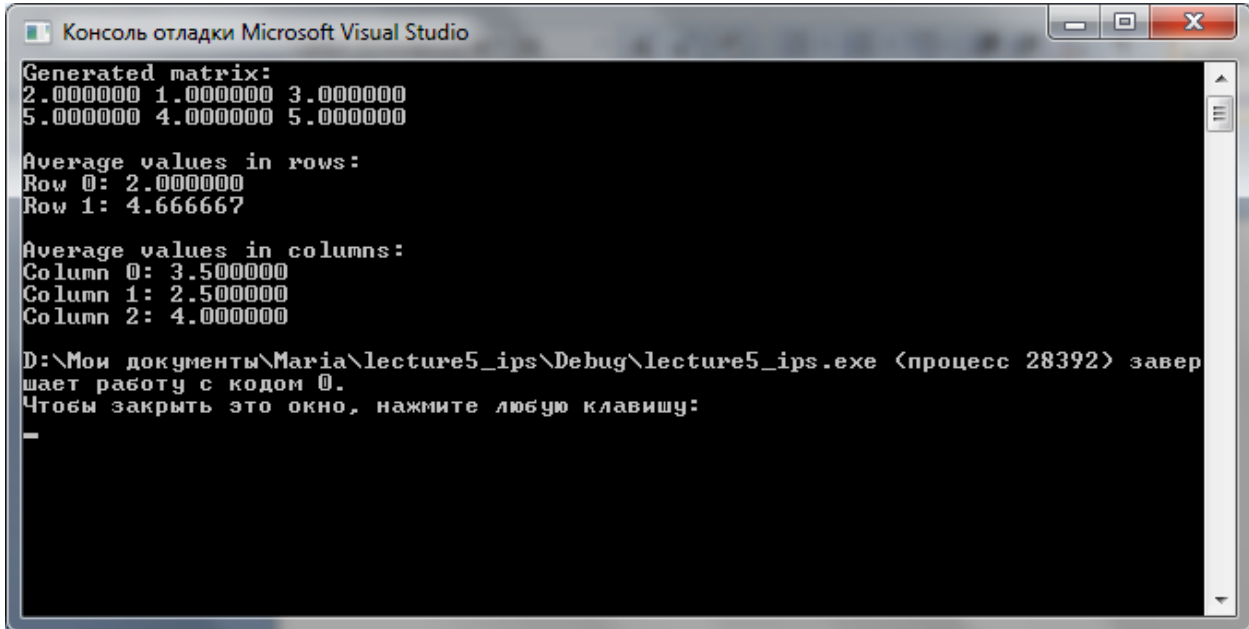


Задание к занятию 5

Выполнила: Шахманова Мария, ПМ-21м

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. Запустите программу и убедитесь в ее работоспособности.



```
Консоль отладки Microsoft Visual Studio
Generated matrix:
2.000000 1.000000 3.000000
5.000000 4.000000 5.000000

Average values in rows:
Row 0: 2.000000
Row 1: 4.666667

Average values in columns:
Column 0: 3.500000
Column 1: 2.500000
Column 2: 4.000000

D:\Мои документы\Maria\lecture5_ips\Debug\lecture5_ips.exe (процесс 28392) завер
шает работу с кодом 0.
Чтобы закрыть это окно, нажмите любую клавишу:
-
```

Программа работает корректно.

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

Используем `cilk_for`:

```
void FindAverageValues( eprocess_type proc_type, double** matrix, const size_t numb_rows, const
size_t numb_cols, double* average_vals )
{
    switch ( proc_type )
    {
        case eprocess_type::by_rows:
        {
            cilk_for ( size_t i = 0; i < numb_rows; ++i )
            {
                //double sum( 0.0 );
                cilk::reducer_opadd<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 )
            {
                //double sum( 0.0 );
```

```

        cilk::reducer_opadd<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;
}
default:
{
    throw("Incorrect value for parameter 'proc_type' in function FindAverageValues()
call!");
}
}
}
}

```

```

Консоль отладки Microsoft Visual Studio

Generated matrix:
3.000000 2.000000 4.000000
1.000000 1.000000 5.000000

Average values in rows:
Row 0: 3.000000
Row 1: 2.333333

Average values in columns:
Column 0: 2.000000
Column 1: 1.500000
Column 2: 4.500000

D:\Мои документы\Maria\lecture5_ips\Debug\lecture5_ips.exe (процесс 30156) завер
шает работу с кодом 0.
Чтобы закрыть это окно, нажмите любую клавишу:

```

Программа работает корректно.

- 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**).

Locate Memory Problems

Target: Analysis Type: Collection Log: Summary

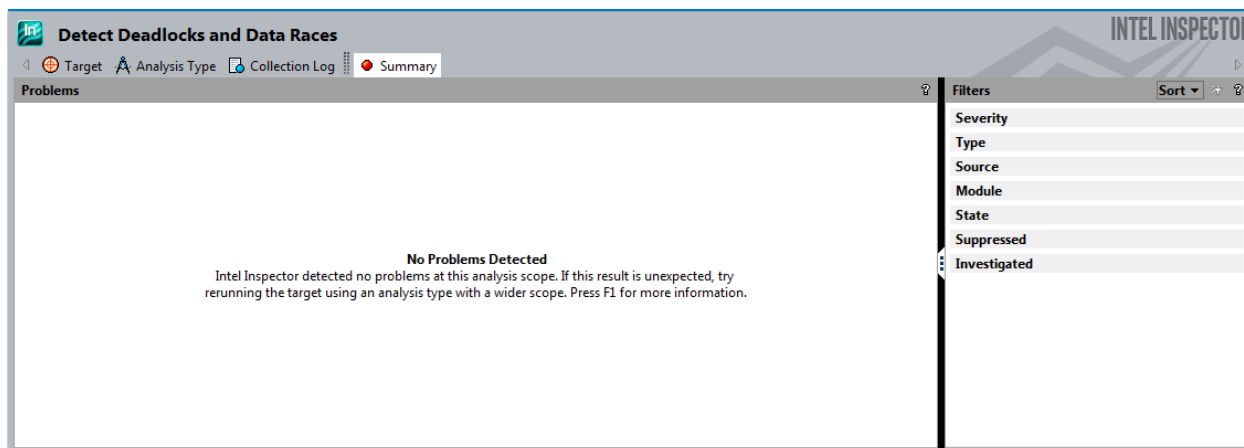
ID	Type	Sources	Modules	Object Size	State
P1	Memory leak	task_for_lecture5.cpp	lecture5_ips.exe	8	New
P2	Memory leak	task_for_lecture5.cpp	lecture5_ips.exe	48	New
P3	Memory leak	task_for_lecture5.cpp	lecture5_ips.exe	16	New
P4	Memory leak	task_for_lecture5.cpp	lecture5_ips.exe	24	New

Filters: Severity: Error (4 item(s)), Type: Memory leak (4 item(s)), Source: task_for_lecture5.cpp (4 item(s)), Module: (4 item(s))

Code Locations: Memory leak

Description	Source	Function	Module	Object Size	Offset	Variable
Allocation site	task_for_lecture5.cpp:143	main	lecture5_ips.exe	8		block allocated at task_for_lecture5.cpp:143
141	const size_t numb_cols = 3;					lecture5_ips.exe!main - task_for_lecture5.
142						lecture5_ips.exe!0x7bbe
143	double** matrix = new double*[numb_rows];					lecture5_ips.exe!0x7a12
144	for (size_t i = 0; i < numb_rows; ++i)					lecture5_ips.exe!0x78a8
145	{					lecture5_ips.exe!mainCRTStartup - exe_main

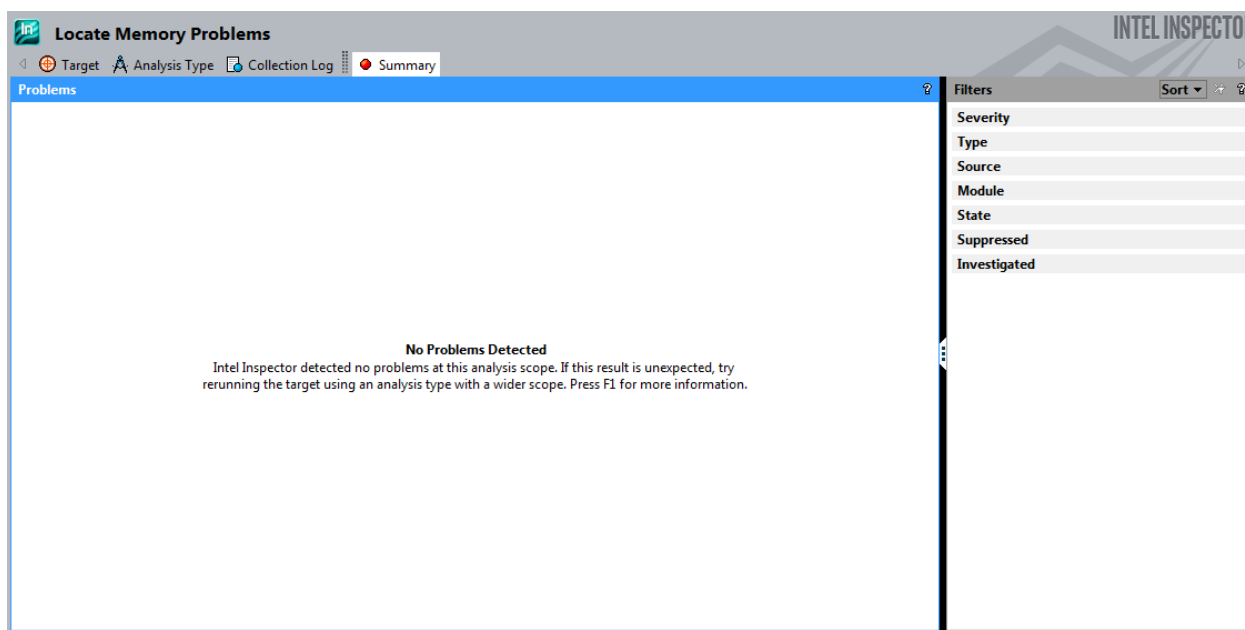
Timeline: RtlQueryEnvironmentVariable (31766)



Запуская на разных уровнях, получим ошибку: утечка памяти.

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*.

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



Как видим, после корректного освобождения памяти ошибок более не возникает.