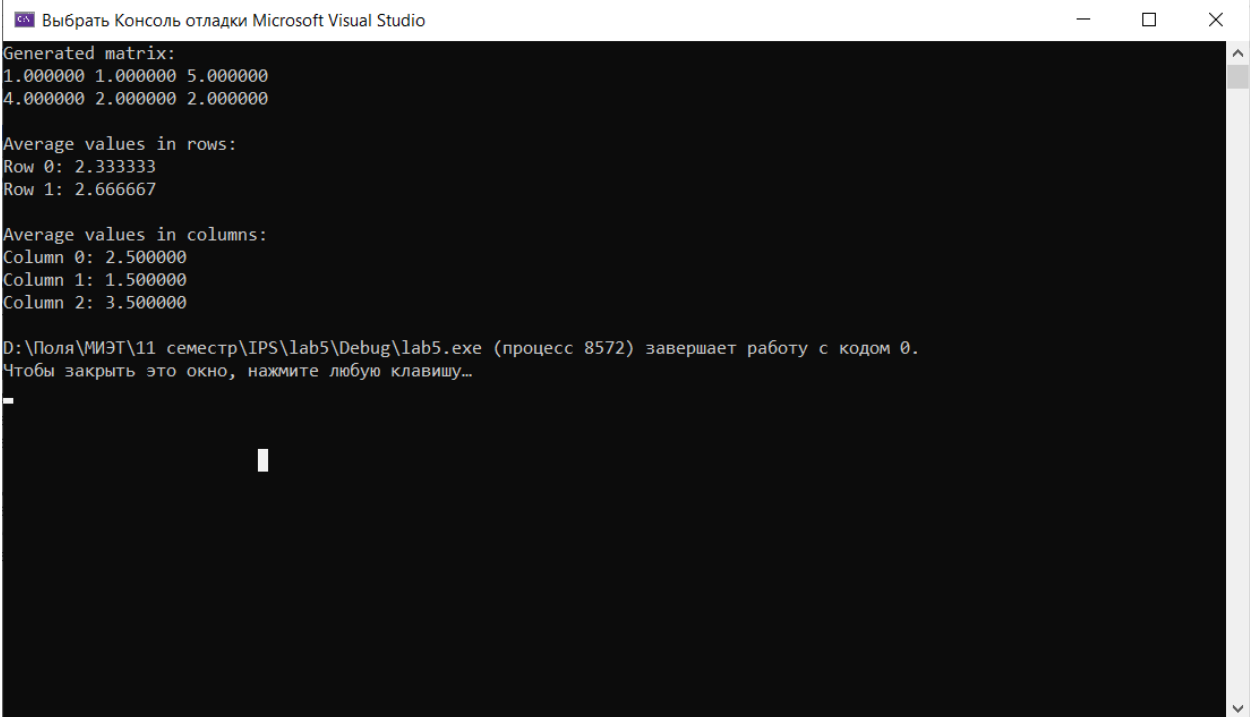


Отчет к занятию 5.

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.

Запускаем программу. Видим, что она работает корректно.



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

Average values in rows:
Row 0: 2.333333
Row 1: 2.666667

Average values in columns:
Column 0: 2.500000
Column 1: 1.500000
Column 2: 3.500000

D:\Поля\МИЭТ\11 семестр\IPS\lab5\Debug\lab5.exe (процесс 8572) завершает работу с кодом 0.
Чтобы закрыть это окно, нажмите любую клавишу...
```

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

Внесем изменения:

```
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) {
                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!");
}
}
}
}

```

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).

С помощью Intel Parallel Inspector определили ошибки утечки памяти.

The screenshot shows the Intel Parallel Inspector interface. The top bar indicates 'Detect Memory Problems'. Below it, the 'Problems' table lists four memory leaks (P1, P2, P3, P4) all of type 'Memory leak' and state 'New'. The 'Filters' panel on the right shows 'Severity' (Error, 4 item(s)), 'Type' (Memory leak, 4 item(s)), 'Source' (lab5.cpp, 4 item(s)), 'Module' (lab5.exe, 4 item(s)), and 'State' (New, 4 item(s)). The bottom panel shows the 'Code Locations: Memory leak' for the first leak (P1). It displays the allocation site in lab5.cpp:122, where a block of 16 bytes is allocated. The code snippet shows the allocation of a 2D array 'matrix'.

ID	Type	Sources	Modules	Object Size	State
P1	Memory leak	lab5.cpp	lab5.exe	16	New
P2	Memory leak	lab5.cpp	lab5.exe	192	New
P3	Memory leak	lab5.cpp	lab5.exe	32	New
P4	Memory leak	lab5.cpp	lab5.exe	48	New

Description	Source	Function	Module	Object Size	Offset	Variable
Allocation site	lab5.cpp:122	main	lab5.exe	16		block allocated at lab5.cpp:122

```

120 const size_t numb_cols = 3*2;
121
122 double** matrix = new double* [numb_rows]
123 for (size_t i = 0; i < numb_rows; ++i) {
124     matrix[i] = new double[numb_cols];

```

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.

Исправим проблемы утечки памяти.

```

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

```

Intel

Detect Deadlocks and Data Races

TargetAnalysis TypeCollection LogSummary

Problems

No Problems Detected

Intel Inspector detected no problems at this analysis scope. If this result is unexpected, try rerunning the target using an analysis type with a wider scope. Press F1 for more information.

Filters

Severity

Type

Source

Module

State

Suppressed

Investigated

Intel

Detect Memory Problems

TargetAnalysis TypeCollection LogSummary

Problems

ID	Type	Sources	Modules	Object Size	State
P1	Memory leak	thread	lab5.exe	32	New

Filters

Severity

Type

Source

Module

State

1 of 1

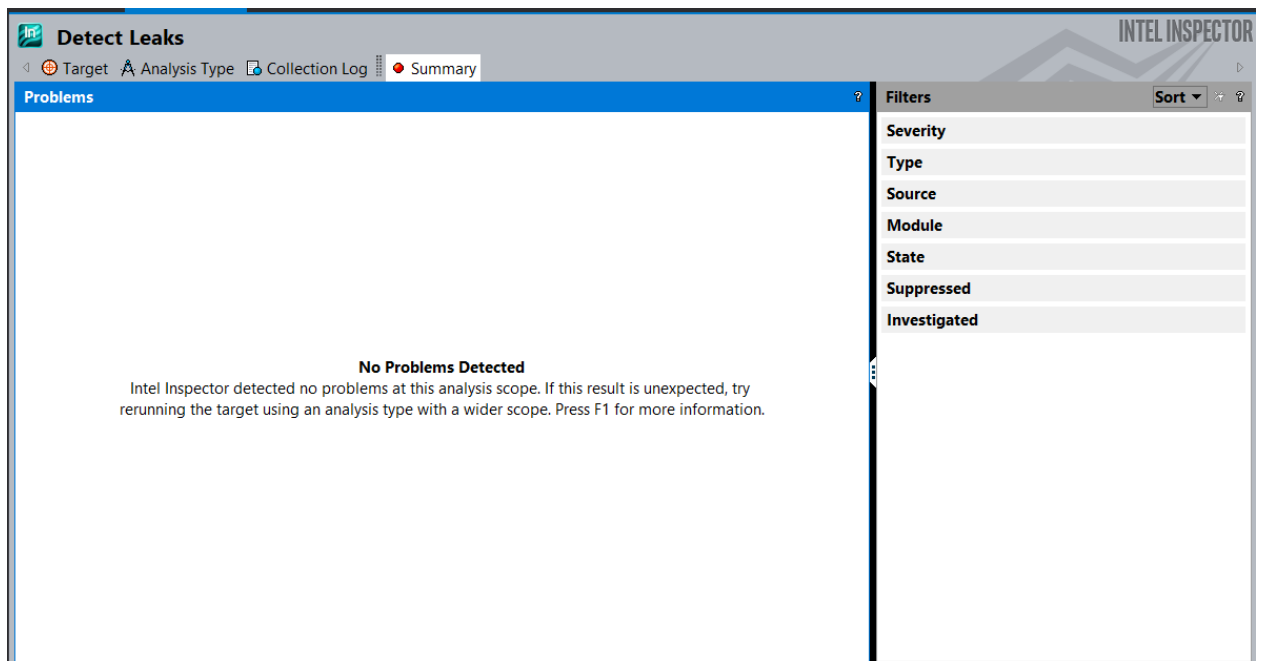
All

Code Locations: Memory leak

Description	Source	Function	Module	Object Size	Offset	Variable
Allocation site	thread:41	_Invoke	lab5.exe	32		block
<div>39 _Tuple& _Tup = * _FnVals; 40 _STD invoke(_STD move(_STD get< Ind 41 _Cnd_do_broadcast_at_thread_exit(); 42 return 0; 43 }</div>						

Timeline

register_onexit_function (6028)



Как видим, ошибки исправлены.