

# Лабораторная работа №3

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

```
Generated matrix:
5.000000 1.000000 2.000000
2.000000 1.000000 4.000000

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

Average values in columns:
Column 0: 3.500000
Column 1: 1.000000
Column 2: 3.000000
```

Программа выдает верные результаты.

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

```
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:
        return;
}
}

```

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.

The screenshot shows the Intel Parallel Inspector interface. The top panel, titled "Problems", lists two memory leak issues:

ID	Type	Sources	Modules	Object Size	State
P1	Memory leak	IPS_labs.cpp	ips_labs.exe	16	New
P2	Memory leak	IPS_labs.cpp	ips_labs.exe	48	New

The bottom panel, titled "Code Locations: Memory leak", provides details for the first problem (P1):

Description	Source	Function	Module	Object Size	Offset	Variable
Allocation site	IPS_labs.cpp:434	run_lab3	ips_labs.exe	16		block allocated at IPS_labs.cpp:434

The code snippet for the allocation site is as follows:

```

432     const size_t numb_cols = 3;
433
434     double** matrix = new double* [numb_rows];
435     for (size_t i = 0; i < numb_rows; ++i)
436     {

```

The call stack on the right shows the execution path:

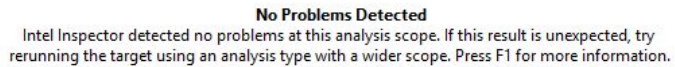
```

ips_labs.exe!run_lab3 - IPS_labs.cpp:434
ips_labs.exe!main - IPS_labs.cpp:476
ips_labs.exe!invoke_main - exe_common.inl:78
ips_labs.exe!_scrt_common_main_seh - exe_common.inl:288
ips_labs.exe!_scrt_common_main - exe_common.inl:330

```

Видим утечку из-за неосвобождения выделенной 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**.



**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.

Как видим, допущенная ошибка была устранена.