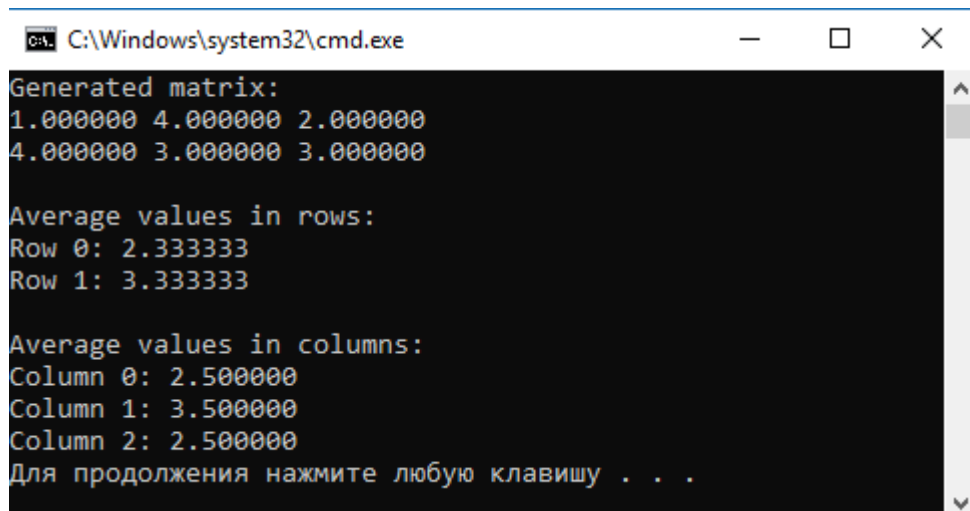


1. Разберите программу представленную в файле task_for_lecture5.cpp. В программе создается 2 потока, каждый из которых вычисляет средние значения матрицы, один по строкам исходной матрицы matrix, а другой - по столбцам. Запустите программу и убедитесь в ее работоспособности.

Запустили программу и убедились в ее работоспособности



```

C:\Windows\system32\cmd.exe

Generated matrix:
1.000000 4.000000 2.000000
4.000000 3.000000 3.000000

Average values in rows:
Row 0: 2.333333
Row 1: 3.333333

Average values in columns:
Column 0: 2.500000
Column 1: 3.500000
Column 2: 2.500000
Для продолжения нажмите любую клавишу . . .
  
```

Как видим, программа работает правильно.

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:
    {
        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.

Определили с помощью наличие в программе таких ошибок как: утечка памяти.

The screenshot displays the Intel Parallel Inspector 2019 interface. The 'Detect Leaks' window is active, showing a summary of detected memory leaks. The 'Problems' table lists four memory leaks (P1, P2, P3, P4) all of type 'Memory leak' in 'main.cpp' within the 'lab1.exe' module, with object sizes of 8, 48, 16, and 24 respectively, and all marked as 'Not fixed'. The 'Filters' panel on the right shows the current filter settings: Severity (Error), Type (Memory leak), Source (main.cpp), Module (lab1.exe), State (Not fixed), and Suppressed. The 'Code Locations: Memory leak' panel at the bottom shows the allocation site at 'main.cpp:550' in the 'main' function, with a description of the memory block allocated at 'main.cpp:550'. The 'Timeline' panel on the right shows a single event at '[Unknown] (6024)'.

ID	Type	Sources	Modules	Object Size	State
P1	Memory leak	main.cpp	lab1.exe	8	Not fixed
P2	Memory leak	main.cpp	lab1.exe	48	Not fixed
P3	Memory leak	main.cpp	lab1.exe	16	Not fixed
P4	Memory leak	main.cpp	lab1.exe	24	Not fixed

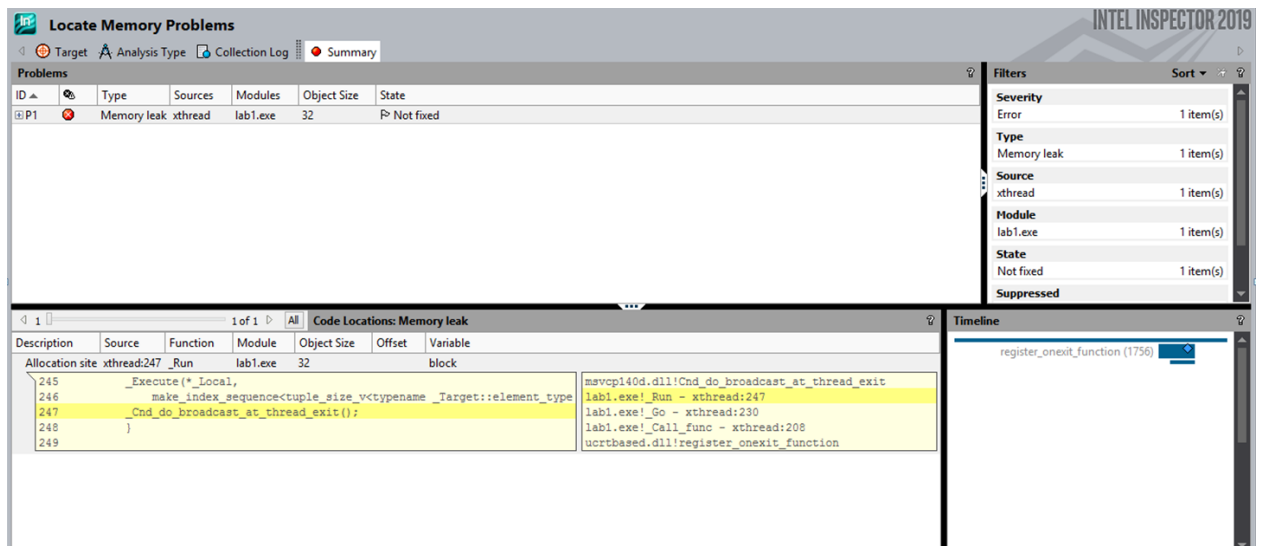
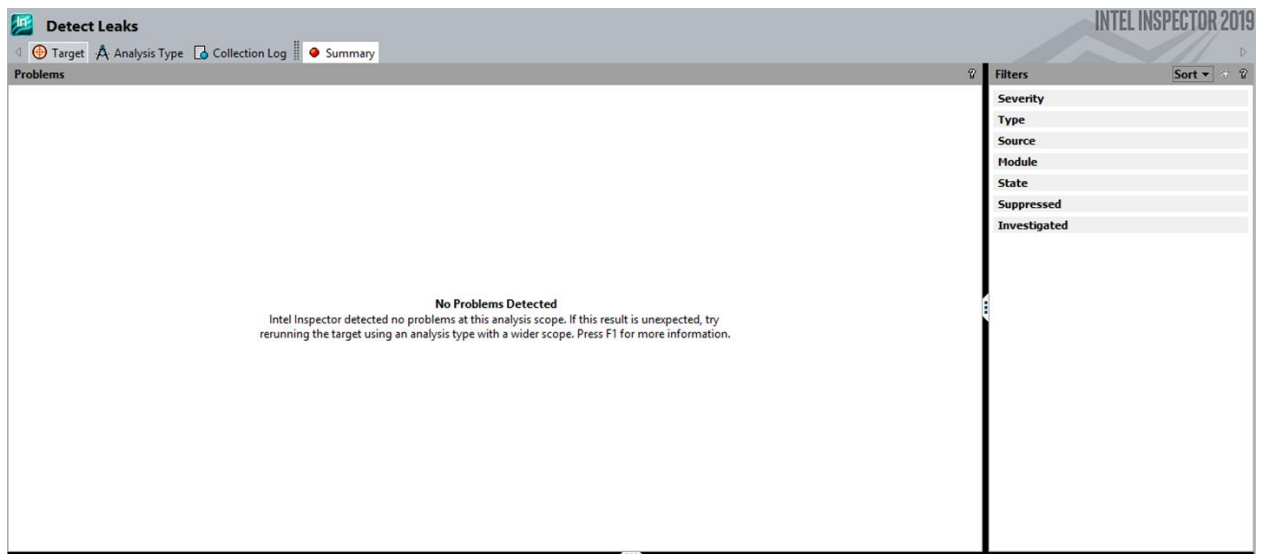
Description	Source	Function	Module	Object Size	Variable
Allocation site	main.cpp:550	main	lab1.exe	8	block allocated at main.cpp:550

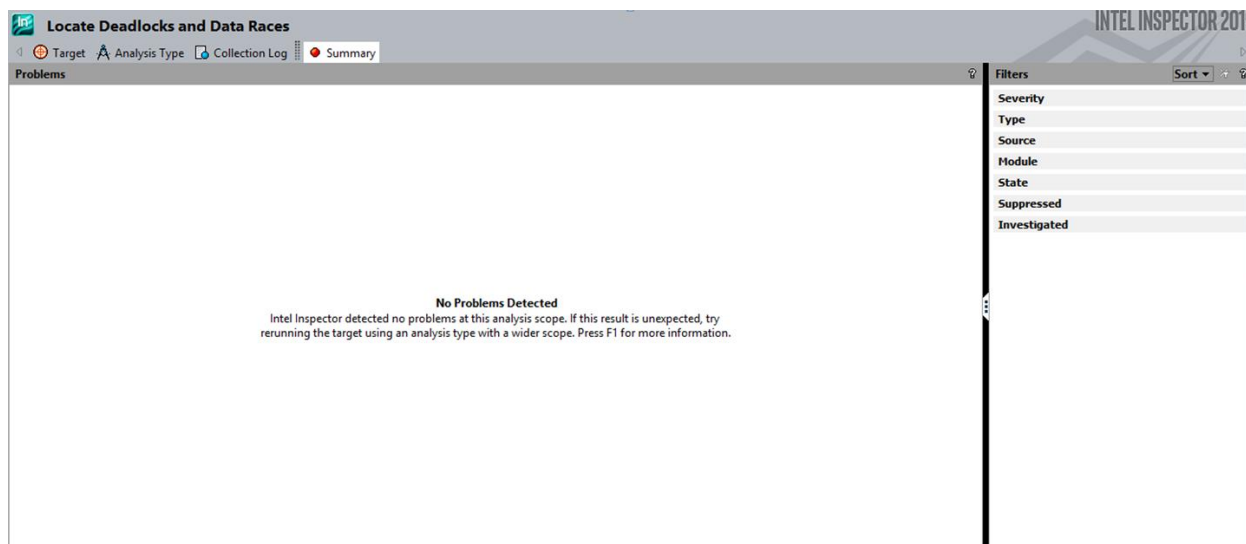
4. Измените код программы таким образом, чтобы Inspector при проверке не находил в программе ошибок, перечисленных в п. 3. Сделайте скрины результатов запуска Parallel Inspector.

Исправим ошибку, которая давала утечку памяти.

```
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 видно, ошибки отсутствуют.