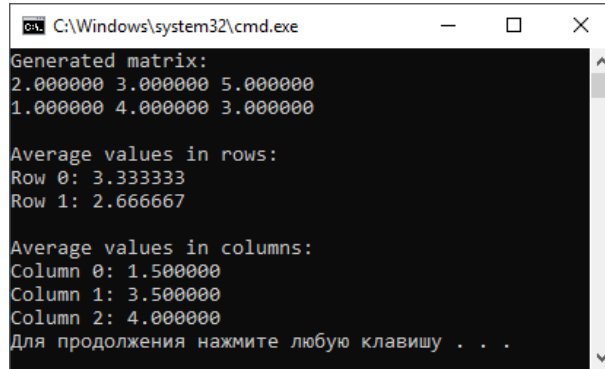


1. Разберите программу представленную в файле [task\\_for\\_lecture5.cpp](#). В программе создается 2 потока, каждый из которых вычисляет средние значения матрицы, один по строкам исходной матрицы *matrix*, а другой - по столбцам. Запустите программу и убедитесь в ее работоспособности.



```

C:\Windows\system32\cmd.exe
Generated matrix:
2.000000 3.000000 5.000000
1.000000 4.000000 3.000000

Average values in rows:
Row 0: 3.333333
Row 1: 2.666667

Average values in columns:
Column 0: 1.500000
Column 1: 3.500000
Column 2: 4.000000
Для продолжения нажмите любую клавишу . . .

```

Рис.1. Проверка работоспособности программы.

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

```

/// Функция FindAverageValues() находит средние значения в матрице <i>matrix</i>
/// по строкам, либо по столбцам в зависимости от значения параметра <i>proc_type</i>;
/// proc_type - признак, в зависимости от которого средние значения вычисляются
/// либо по строкам, либо по столбцам исходной матрицы <i>matrix</i>
/// matrix - исходная матрица
/// numb_rows - количество строк в исходной матрице <i>matrix</i>
/// numb_cols - количество столбцов в исходной матрице <i>matrix</i>
/// average_vals - массив, куда сохраняются вычисленные средние значения
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 );
                for( size_t j = 0; j < numb_cols; ++j )
                {
                    sum += matrix[i][j];
                }
                average_vals[i] = sum / numb_cols;
            }
            break;
        }
        case eprocess_type::by_cols:
        {
            cilk_for( size_t j = 0; j < numb_cols; ++j )
            {
                double sum( 0.0 );
                for( size_t i = 0; i < numb_rows; ++i )
                {
                    sum += matrix[i][j];
                }
                average_vals[j] = sum / numb_rows;
            }
            break;
        }
        default:
        {
            throw( "Incorrect value for parameter 'proc_type' in function FindAverageValues() call!" );
        }
    }
}

```

Рис.2. Внесённые изменения.

Изменения связаны с распараллеливанием циклов for.

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*).

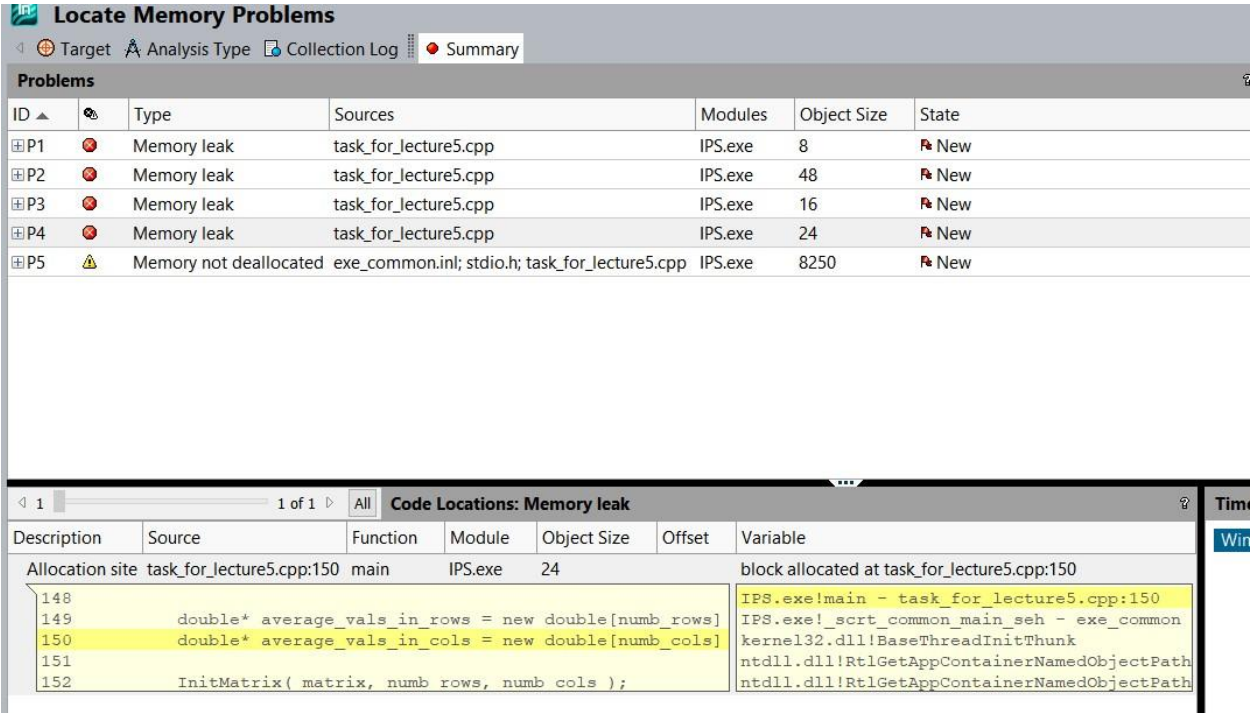


Рис.3. Анализ ошибок памяти. Быстрый.

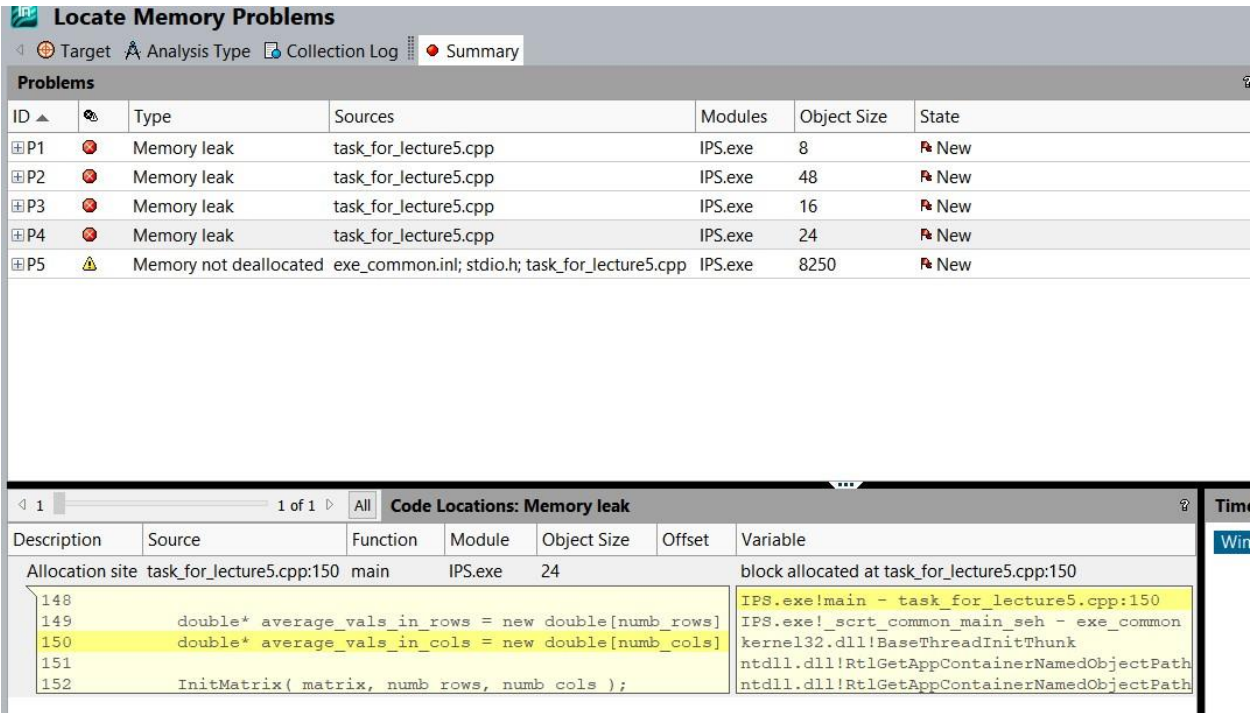


Рис.4. Анализ ошибок памяти. Средний.

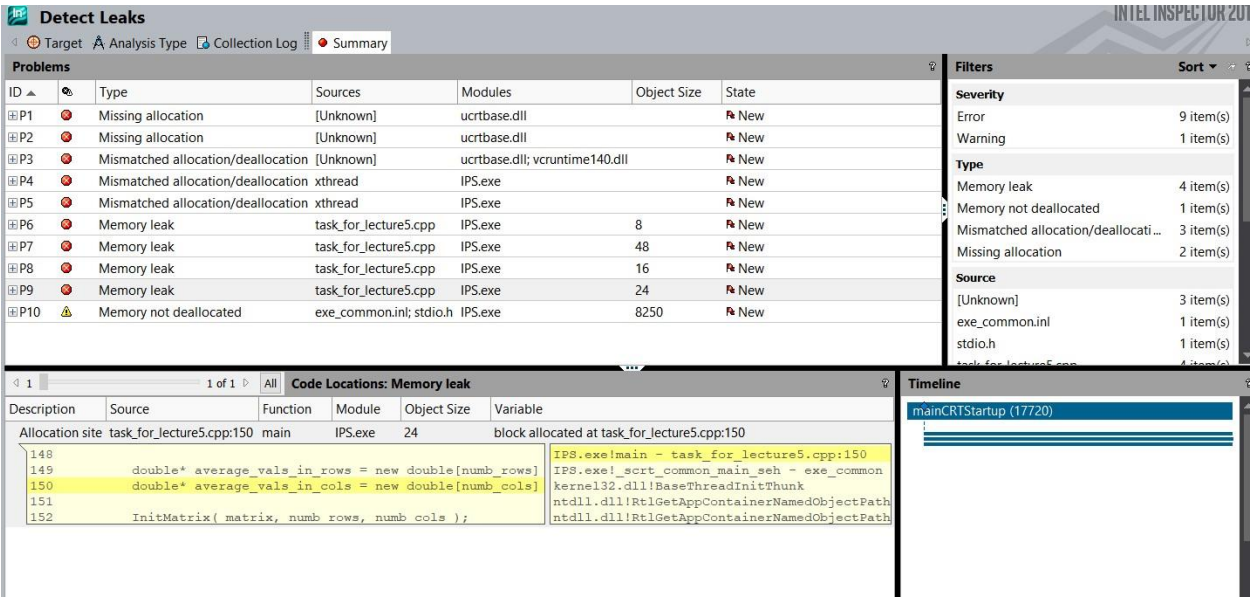


Рис.5. Анализ ошибок памяти. Полный.

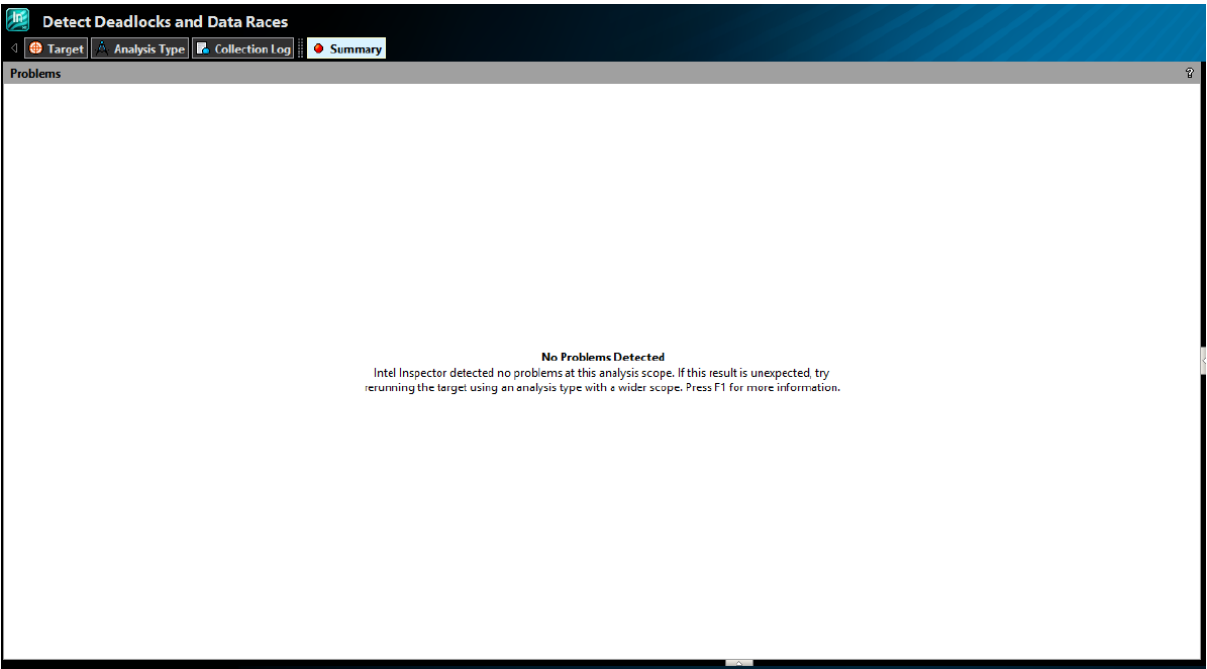


Рис.6. Обнаружение тупиков и гонки данных.

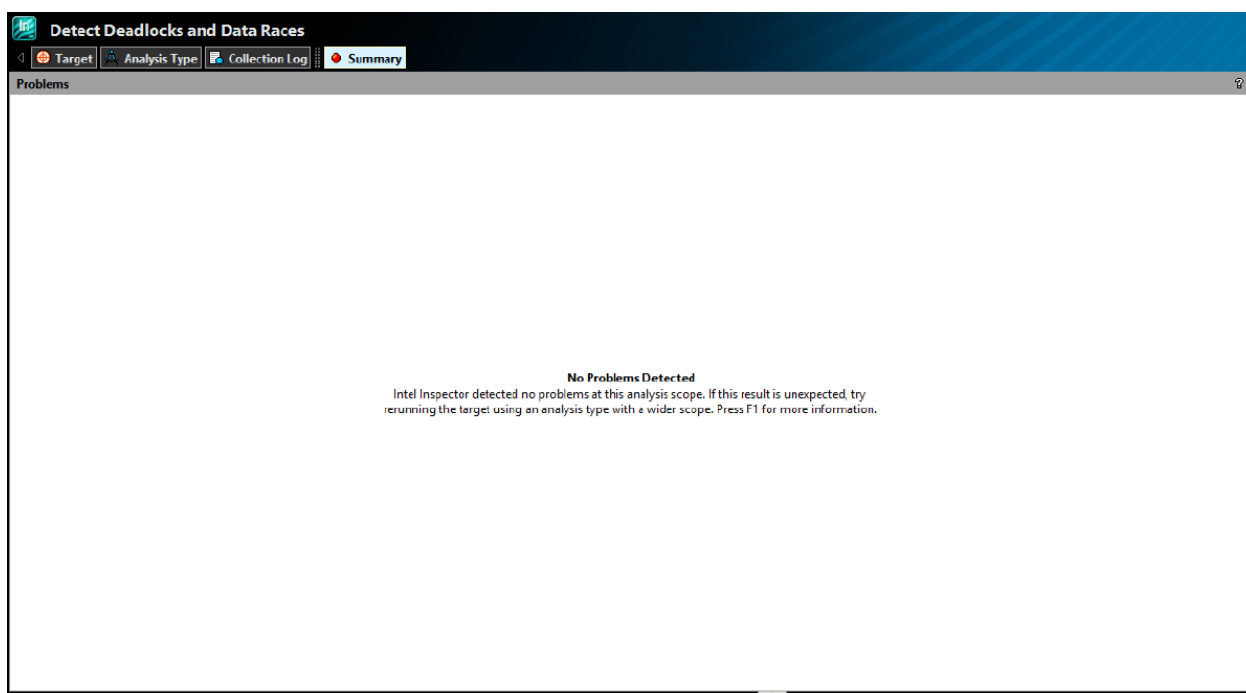


Рис.7. Обнаружение тупиков и гонки данных.

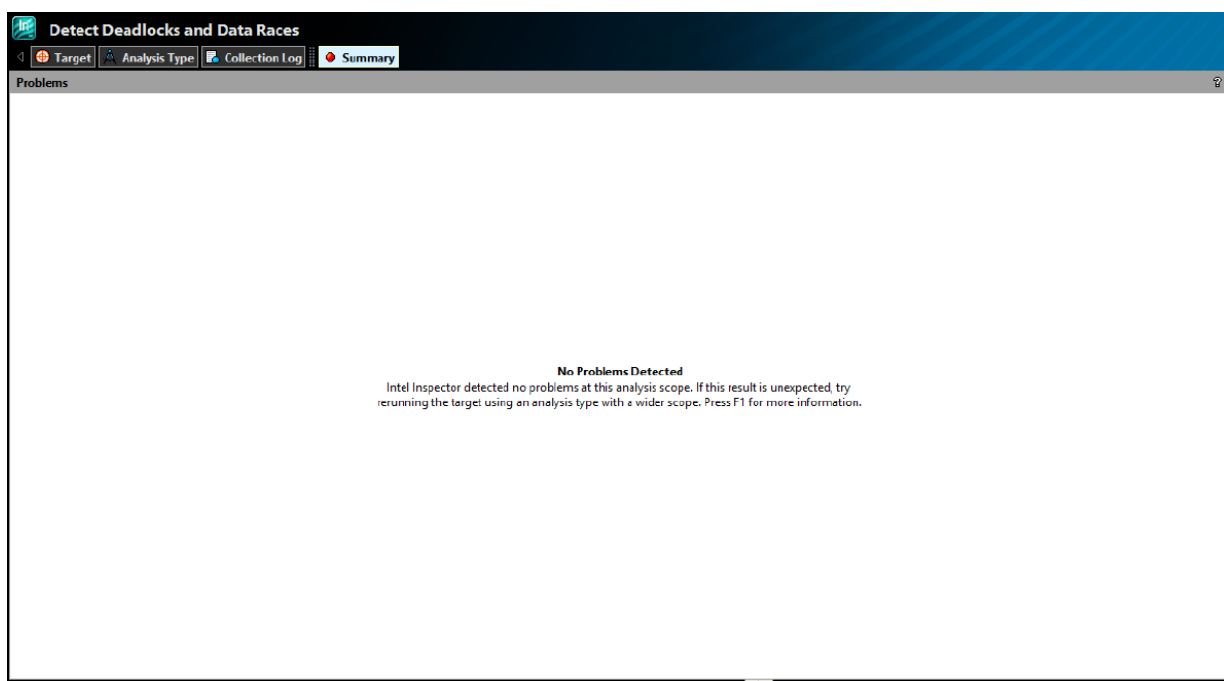


Рис.8. Обнаружение тупиков и гонки данных.

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*.

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

Рис.8. Изменения кода программы.

Добавлена очистка памяти.

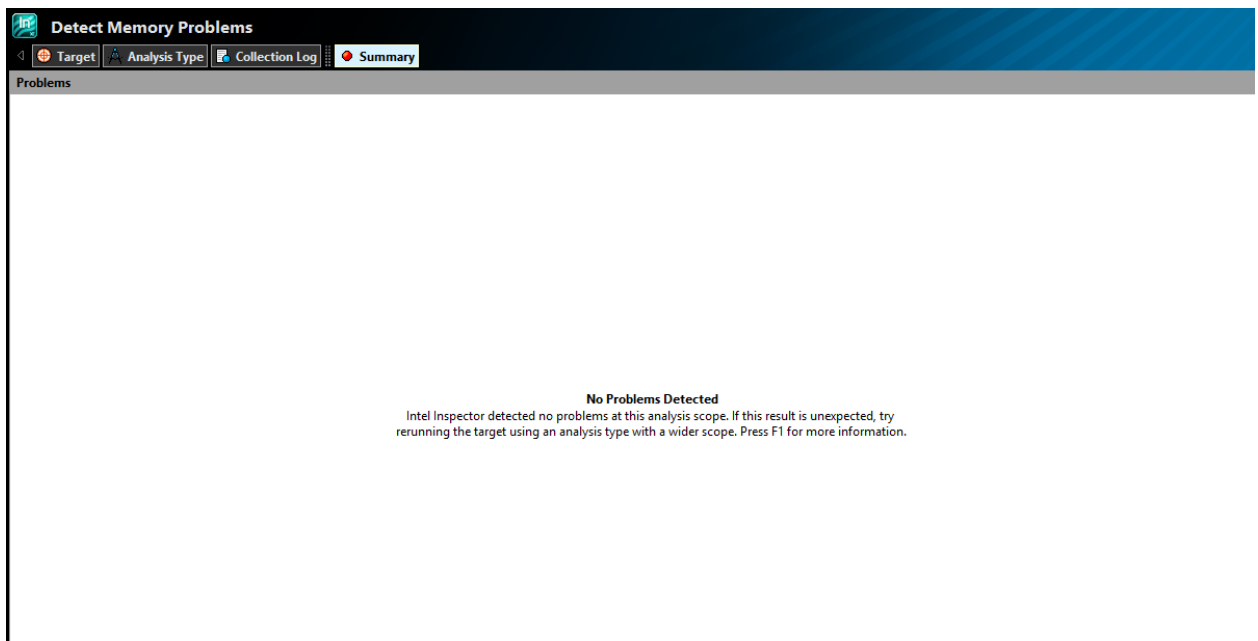


Рис. 9. Окно ошибок по памяти.

5. Результатом работы должна быть ссылка на Ваш профиль *GitHub* где представлен код программы с введенными изменениями согласно заданиям. Скрины, отображающие работу с *Intel Parallel Inspector* необходимо представить в отдельном документе .pdf и загрузить в систему в качестве отчета к этому заданию.

Ссылка на **GitHub** [https://github.com/vdemcnehko/ips\\_3](https://github.com/vdemcnehko/ips_3)