

**УФИМСКИЙ  
УНИВЕРСИТЕТ  
НАУКИ И ТЕХНОЛОГИЙ**

Курс «Технологии  
параллельного  
программирования»

## **Лабораторная работа №1. Параллельное вычисление суммы числового ряда**

**Юлдашев Артур Владимирович**  
**art@ugatu.su**

**Спеле Владимир Владимирович**  
**spele.vv@ugatu.su**

**Добровольцев Александр Сергеевич**  
**dobrovolcev.as@ugatu.su**

**Сохатский Михаил Александрович**  
**sohatskii.ma@ugatu.su**

**Кафедра высокопроизводительных  
вычислений и дифференциальных  
уравнений (ВВиДУ)**

## Цель работы

На примере задачи сложения суммы ряда научиться использовать оптимизационные ключи компиляторов в операционных системах Windows (и Linux), а также инструмент для профилирования программ Intel Advisor.

Используемые компиляторы:

- Microsoft Visual C/C++,
- Intel Compiler Classic C/C++,
- Intel Clang/LLVM C/C++,
- GNU C/C++

## Компилятор Microsoft C/C++

В состав Visual Studio включен компилятор языка C/C++ позволяющий создавать все, от простых консольных приложений, до универсальных приложений Windows, приложений Магазина Windows и компонентов .NET.

Ключи оптимизации:

**/Od** – отключение оптимизаций (параметр по умолчанию)

**/O1** – максимальная оптимизация с приоритетом к уменьшению размера кода программы

**/O2** – максимальная оптимизация с приоритетом к увеличению скорости работы программы

**/Ox** – полная оптимизация,

**/Qpar** – автораспараллеливание на доступное число ядер и автовекторизация кода (происходит при выполнении определенных условий)

## Компилятор Intel Classic C/C++

Классический оптимизирующий компилятор от Intel. Входит в состав Intel OneAPI HPC Toolkit. В среде Windows возможна интеграция в Visual Studio.

В Linux сборка осуществляется в консоли командой:

**icc/icrc -ключ\_оптимизации имя\_файла.c/cpp**

Ключи оптимизации:

**/Od(O0** в Linux) – отключение оптимизаций

**/O1** – оптимизация по размеру

**/O2** – максимизация скорости

**/O3** – задействует оптимизации из /O2 и дополнительно более агрессивные методы оптимизации циклов и доступа к памяти

**/Ox** – максимальные оптимизации

**/QxHost** - обеспечивает генерацию максимально современных векторных инструкций, поддерживаемых платформой

**/Qparallel** – автораспараллеливание кода на доступное число ядер (происходит при выполнении определенных условий)

## Компилятор Intel Clang/LLVM C/C++

Компилятор нового поколения на базе LLVM от Intel. Входит в состав Intel OneAPI HPC Toolkit. В среде Windows возможна интеграция в Visual Studio.

В Linux сборка осуществляется в консоли командой:

**icx/icpx -ключ\_оптимизации имя\_файла.c/cpp**

Ключи оптимизации:

**/Od(O0** в Linux) – отключение оптимизаций

**/O1** – оптимизация по размеру

**/O2** – максимизация скорости

**/O3** – задействует оптимизации из /O2 и дополнительно более агрессивные методы оптимизации циклов и доступа к памяти

**/Ox** – максимальные оптимизации

**/QxHost** - обеспечивает генерацию максимально современных векторных инструкций, поддерживаемых платформой

# Компилятор GNU C/C++

GNU компилятор, входящий в состав операционной системы Linux.

Сборка программ осуществляется командой:  
**gcc/g++ -ключи\_оптимизации имя\_файла.c/cpp**

Ключи оптимизации:

- O0 – отключение оптимизаций
- O1 – оптимизация по размеру
- O2 – максимизация скорости
- O3 – максимальный уровень оптимизаций

## Задание

1. Написать последовательную версию программы вычисления суммы ряда, выбранного в соответствии со своим вариантом из задания к лабораторной работе, на языке C/C++. Предусмотреть замер времени выполнения основного вычислительного цикла, вывод на экран времени выполнения (в секундах) и вычисленной суммы.
2. Протестировать работоспособность программы при различных размерностях ( $N$ ), проверить корректность путем сравнения с каким-либо интернет-сервисом, позволяющим вычислить сумму ряда.
3. Подобрать  $N$  при которых программа будет работать ~ 30 сек. Провести анализ времени ее выполнения при использовании компиляторов различных производителей и различных ключей оптимизации под операционными системами Windows (и Linux).

## Задание

Оценить быстродействие в режимах сборки (Debug/Release).

Название компилятора	Время работы
Debug	
Release	

Оценить быстродействие архитектур (x86/x64)

Название компилятора	Время работы
x86	
x64	



## Задание

На лучшем варианте из (Debug/Release) и (x86/x64) оценить быстродействие с ключами оптимизации компилятора Microsoft C/C++.

Название компилятора	Время работы
/Od	
/O1	
/O2	
/Ox	

На лучшем варианте из ключей оптимизации оценить быстродействие с ключами оптимизации /Qpar компилятора Microsoft C/C++.

Название компилятора	Время работы
/Qpar	

## Задание

На лучшем варианте из (Debug/Release) и (x86/x64) оценить быстродействие с ключами оптимизации компилятора Intel Classic C/C++.

Название компилятора	Время работы
/Od	
/O1	
/O2	
/O3	
/Ox	

На лучшем варианте из ключей оптимизации оценить быстродействие с ключами оптимизации QxHost и Qparallel компилятора Intel Classic C/C++.

Название компилятора	Время работы
/QxHost	
/Qparallel	

## Задание

На лучшем варианте из (Debug/Release) и (x86/x64) оценить быстродействие с ключами оптимизации компилятора Intel Clang/LLVM C/C++.

Название компилятора	Время работы
/Od	
/O1	
/O2	
/O3	
/Ox	

На лучшем варианте из ключей оптимизации оценить быстродействие с ключом оптимизации QxHost компилятора Intel Clang/LLVM C/C++.

Название компилятора	Время работы
/QxHost	

## Задание

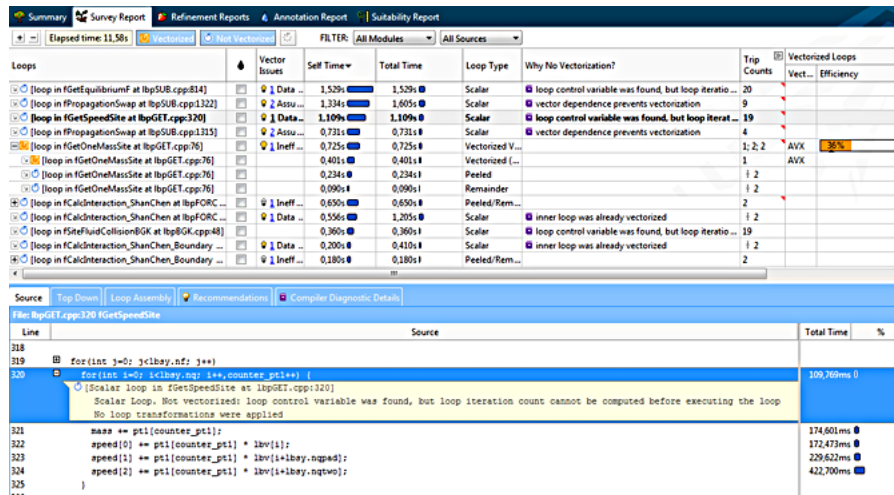
Оценить быстродействие с ключами оптимизации компилятора GNU C/C++.

Название компилятора	Время работы
-O0	
-O1	
-O2	
-O3	

Компилятор GNU C/C++ можно тестировать двумя способами:

1. В нативной Ubuntu.
2. Через WSL, запустив Ubuntu через меню Пуск.

# Intel OneAPI Advisor



The screenshot displays the Intel OneAPI Advisor interface. The top navigation bar includes tabs for Summary, Survey Report, Refinement Reports, Annotation Report, and Suitability Report. The main table lists various loops from different source files, showing their vectorization status, reasons for non-vectorization, and performance metrics. Below the table, the 'Source' tab is selected, showing the C++ code for the 'Fibonacci' loop. The code is as follows:

```
318 for(int j=0; j<lbay.nf; j++)
319 {
320     for(int i=0; i<lbay.nf; i++)
321     {
322         speed[0] += ptl[counter_ptl] * lbv[i];
323         speed[1] += ptl[counter_ptl] * lbv[i+lbay.ngpad];
324         speed[2] += ptl[counter_ptl] * lbv[i+lbay.ngtw];
325     }
326 }
```

Поддерживаемые языки:  
C, C++, Fortran

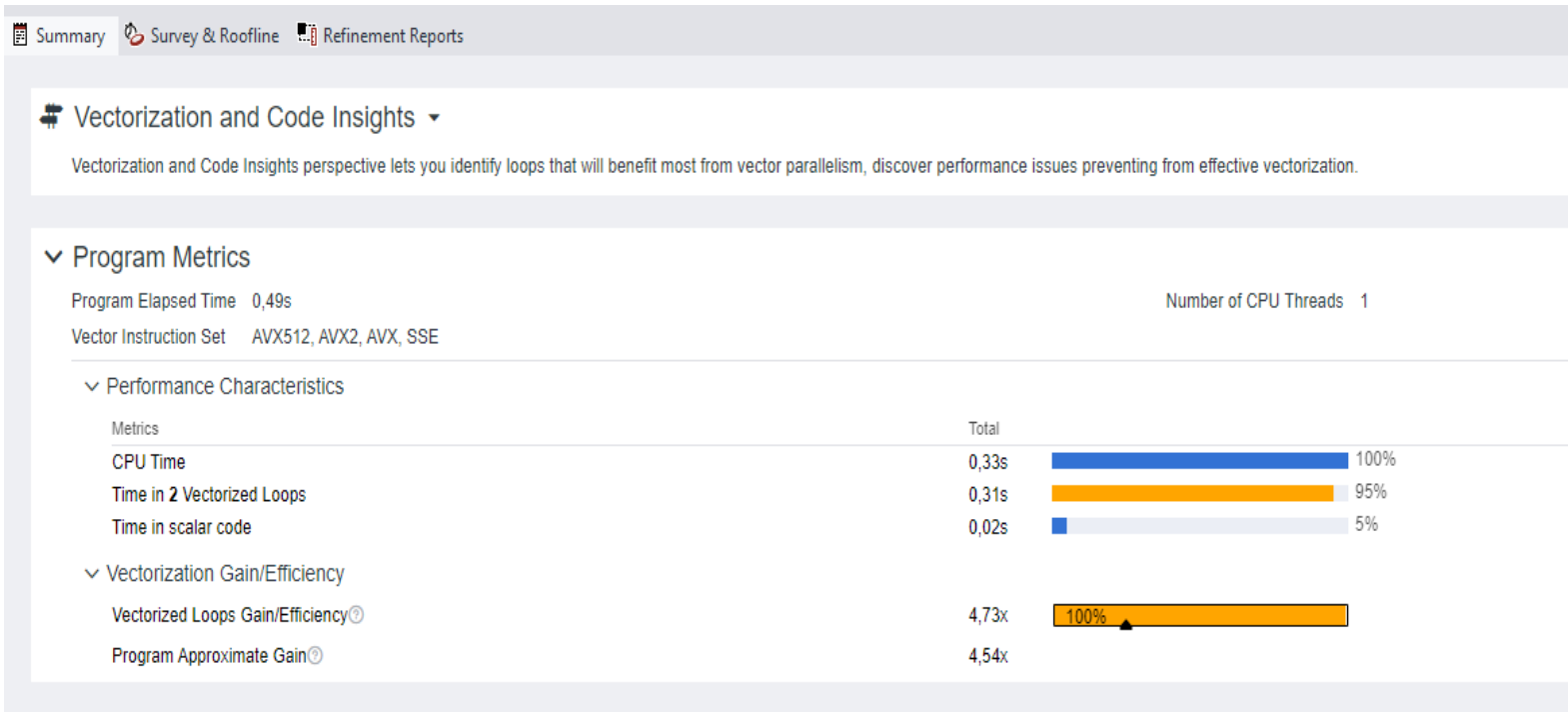
Поддерживаемые  
операционные системы:  
Windows, Linux

Инструмент Intel OneAPI Advisor является помощником разработчика вычислительных приложений.

Он дает советы разработчику по оптимизации приложения, автоматизируя анализ исходного кода, необходимый для быстрого и корректного внедрения векторизации и др.

# Задание

Пользуясь инструментом Intel Advisor добиться успешной векторизации кода и вставить скриншот в отчет



# Создание проекта

## Visual Studio 2019

### Открыть последние

Поиск в недавнем (ALT+"B")

#### Сегодня



lab1.sln

20.10.2020 10:59

C:\Users\Vova\source\repos\lab1

#### Вчера



Project3.sln

19.10.2020 15:38

C:\Users\Vova\source\repos\Project3

#### В этом месяце



sml.sln

06.10.2020 19:27

C:\Users\Vova\source\repos\stsim\build

#### В этом месяце



prj\_2017.sln

04.10.2020 18:55

C:\Users\Vova\Documents\usatu\_hardcoders\prj



Project2.sln

01.10.2020 1:42

C:\Users\Vova\source\repos\Project2

#### Ранее

### Начало работы



#### Клонирование или извлечение кода

Получить код из интернет-репозитория,  
например, GitHub или Azure DevOps



#### Открыть проект или решение

Открыть локальный проект Visual Studio или  
SLN-файл



#### Открыть локальную папку

Перейти и изменить код в любой папке



#### Создание проекта


Выберите шаблон проекта с формированием  
шаблонов кода, чтобы начать работу

Продолжить без кода →

# Создание проекта

## Создание проекта

### Последние шаблоны проектов

 Консольное приложение C++

 Пустой проект C++

 CUDA 10.2 Runtime

Все языки

Все платформы

Все типы проектов



#### Пустой проект

Начать с нуля, используя C++ для Windows. Начальные файлы отсутствуют.

C++ Windows Консоль



#### Консольное приложение

Выполнить код в терминале Windows. По умолчанию выводится фраза "Hello World".

C++ Windows Консоль



#### Мастер классических приложений Windows

Создание собственного приложения Windows с помощью мастера.

C++ Windows Рабочий стол Консоль Библиотека



#### Классическое приложение Windows

Проект приложения с графическим интерфейсом в Windows.

C++ Windows Рабочий стол



#### Проект общих элементов

Проект общих элементов используется для совместного использования файлов в нескольких проектах.

C++ Windows Android iOS Linux Рабочий стол

Консоль Библиотека GMP Игры Мобильный

Назад

Далее



# Пример программы

## Пример

```
#include <iostream>
#define _USE_MATH_DEFINES
#include <math.h>
#include <time.h>
using namespace std;
int main()
{
    int N = 700000000;
    double start_time = clock();
    double sum = 0.0;
    for (int i = 1; i < N; i++)
    {
        sum += pow(-1, i) / (i - log10(i));
    }
    double end_time = clock();
    cout << "time = " << (end_time - start_time) / CLK_TCK << endl;
    cout << "SUM = " << sum << endl;
    return 0;
}
```

## Результат работы программы

```
time = 17.981
SUM = -0.641846
```

## Пример программы

В случае знакопеременного ряда можно разделить цикл на 2 (может повлиять на точность вычислений!).

```
#include <iostream>
#define _USE_MATH_DEFINES
#include <math.h>
#include <time.h>
using namespace std;
int main()
{
    int N = 700000000;
    double start_time = clock();
    double sum = 0.0;
    for (int i = 1; i < N; i += 2)
    {
        sum -= 1 / (i - log10(i));
    }
    for (int i = 2; i < N; i += 2)
    {
        sum += 1 / (i - log10(i));
    }
    double end_time = clock();
    cout << "time = " << (end_time - start_time) / CLK_TCK << endl;
    cout << "SUM = " << sum << endl;
    return 0;
}
```

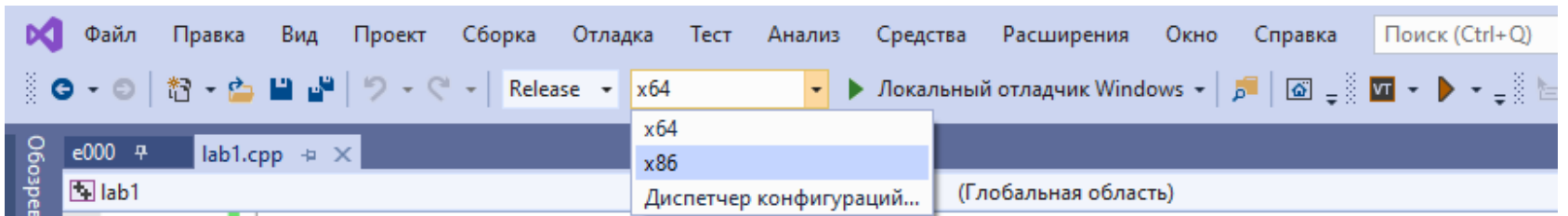
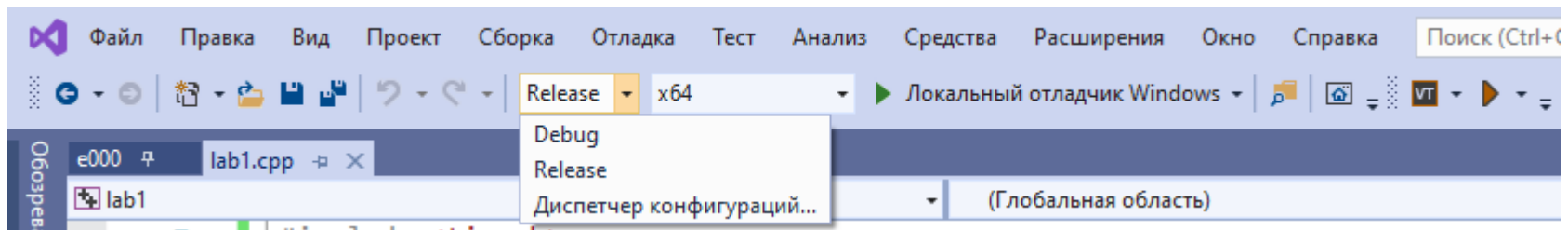
Результат работы программы

```
time = 8.502
SUM = -0.641846
```

# Компилятор Microsoft C/C++

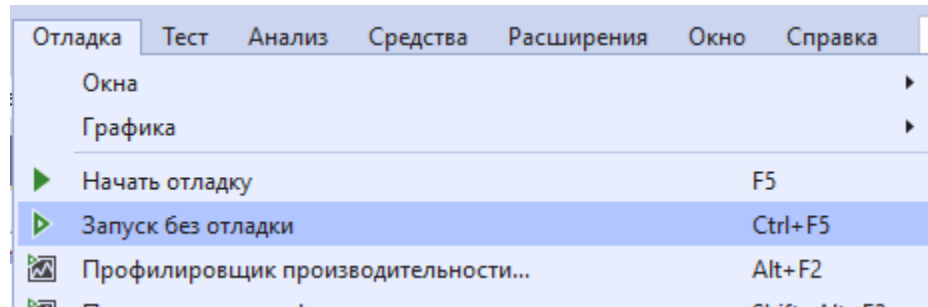
Выберите наиболее производительную конфигурацию:

1. Режим сборки (Debug, Release)
2. Архитектуры (x86/x64)



# Компилятор Microsoft C/C++

Запуск программы с отладкой (**F5**) и без отладки (**CTRL+F5**)



Запуск с **F5**

```
time = 22.202  
SUM = -0.641846
```

Запуск с **CTRL + F5**

```
time = 8.45  
SUM = -0.641846
```

Запуск программы без отладки (**CTRL + F5**) значительно быстрее запуска программы с отладкой (**F5**).

# Компилятор Microsoft C/C++

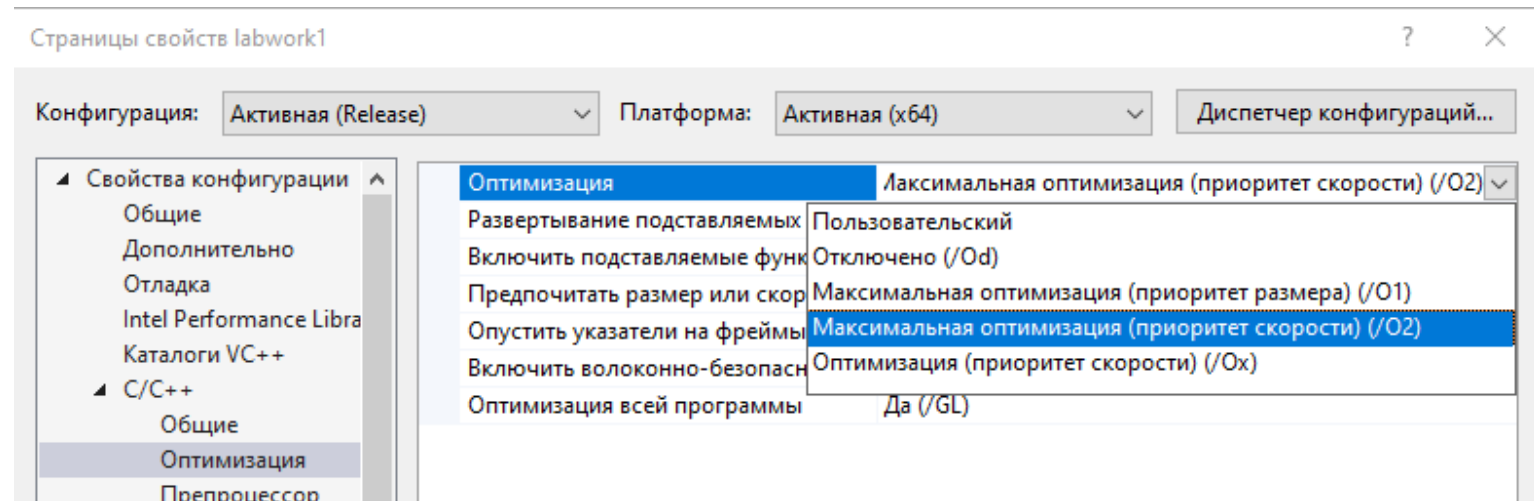
Ключи оптимизации:

**/Od** – отключение оптимизаций (параметр по умолчанию)

**/O1** – максимальная оптимизация с приоритетом к уменьшению размера кода программы

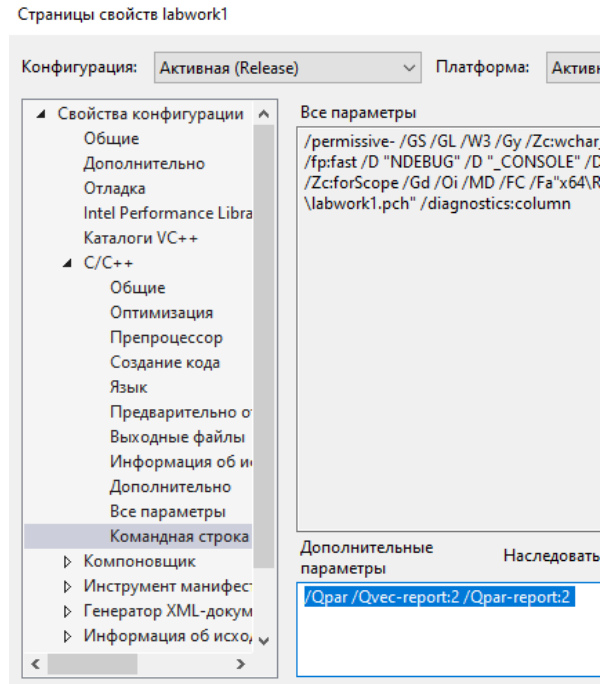
**/O2** – максимальная оптимизация с приоритетом к увеличению скорости работы программы

**/Ox** – полная оптимизация,



[Справка о ключах оптимизации /O](#)

# Компилятор Microsoft C/C++



**/Qpar** – включает автоматическое распараллеливание циклов в коде (происходит при выполнении определенных условий),

[Справка по /Qpar](#)

**/Qpar-report, Qvec-report** – вывод информации об автораспараллеливании и автовекторизации кода

[Справка по /Qpar-report](#)

[Справка по /Qvec-report](#)

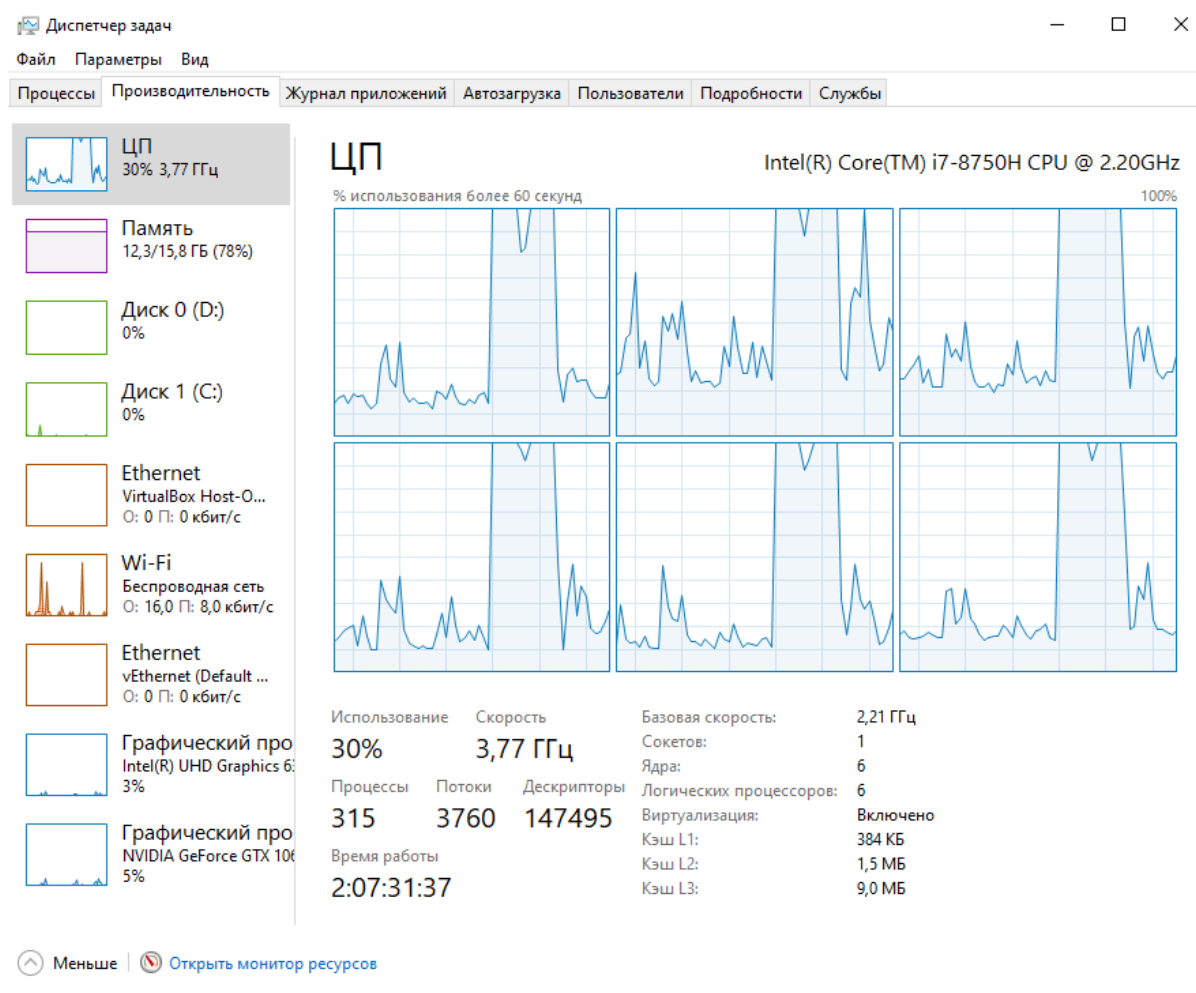
## Компилятор Microsoft C/C++

При компиляции с ключами **/Qpar-report:2** и **/Qvec-report:2** в вывод компилятора выводятся информационные сообщения о результатах автопараллелизации и автовекторизации.

```
1>--- Анализ функции: main
1>C:\Users\Vova\source\repos\labwork1\labwork1.cpp(24) : info C5002: цикл не векторизирован по следующей причине: "1301"
1>C:\Users\Vova\source\repos\labwork1\labwork1.cpp(28) : info C5002: цикл не векторизирован по следующей причине: "1301"
1>C:\Users\Vova\source\repos\labwork1\labwork1.cpp(24) : info C5012: цикл не параллелизован по следующей причине: "1001"
1>C:\Users\Vova\source\repos\labwork1\labwork1.cpp(28) : info C5012: цикл не параллелизован по следующей причине: "1001"
1>All 12 functions were compiled because no usable IPDB/IOBJ from previous compilation was found.
1>Создание кода завершено
1>labwork1.vcxproj -> C:\Users\Vova\source\repos\labwork1\x64\Release\labwork1.exe
```

[Справка по кодам сообщений /Qpar-report и /Qvec-report](#)

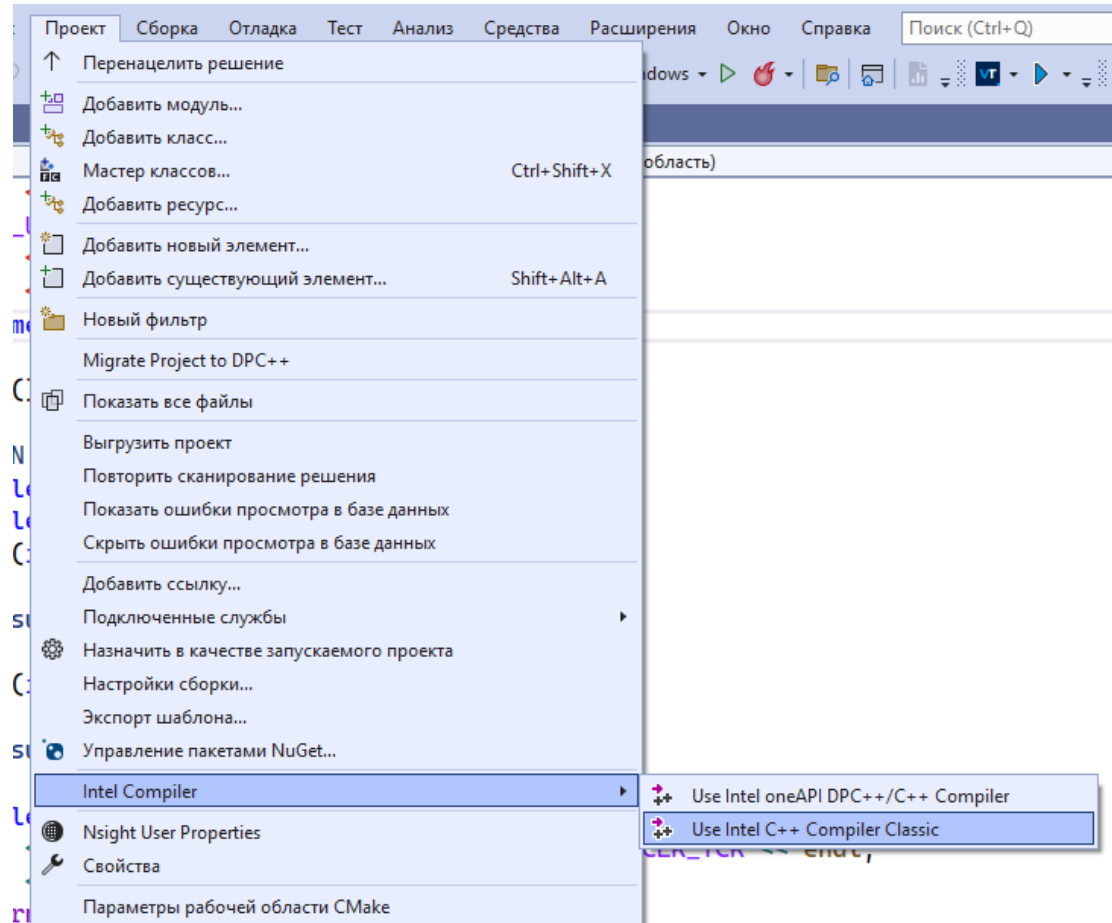
# Мониторинг загрузки CPU





# Компилятор Intel Classic C/C++

Переход на компилятор Intel.



# Компилятор Intel Classic C/C++

Ключи оптимизации:

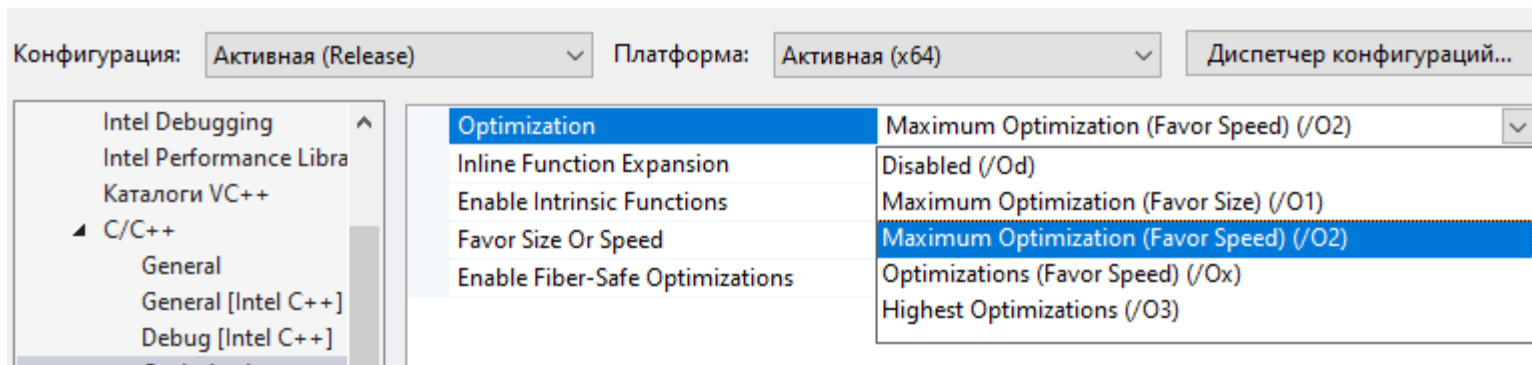
**/Od(O0** в Linux) – отключение оптимизаций

**/O1** – оптимизация по размеру

**/O2** – максимизация скорости

**/O3** – задействует оптимизации из /O2 и дополнительно более агрессивные методы оптимизации циклов и доступа к памяти

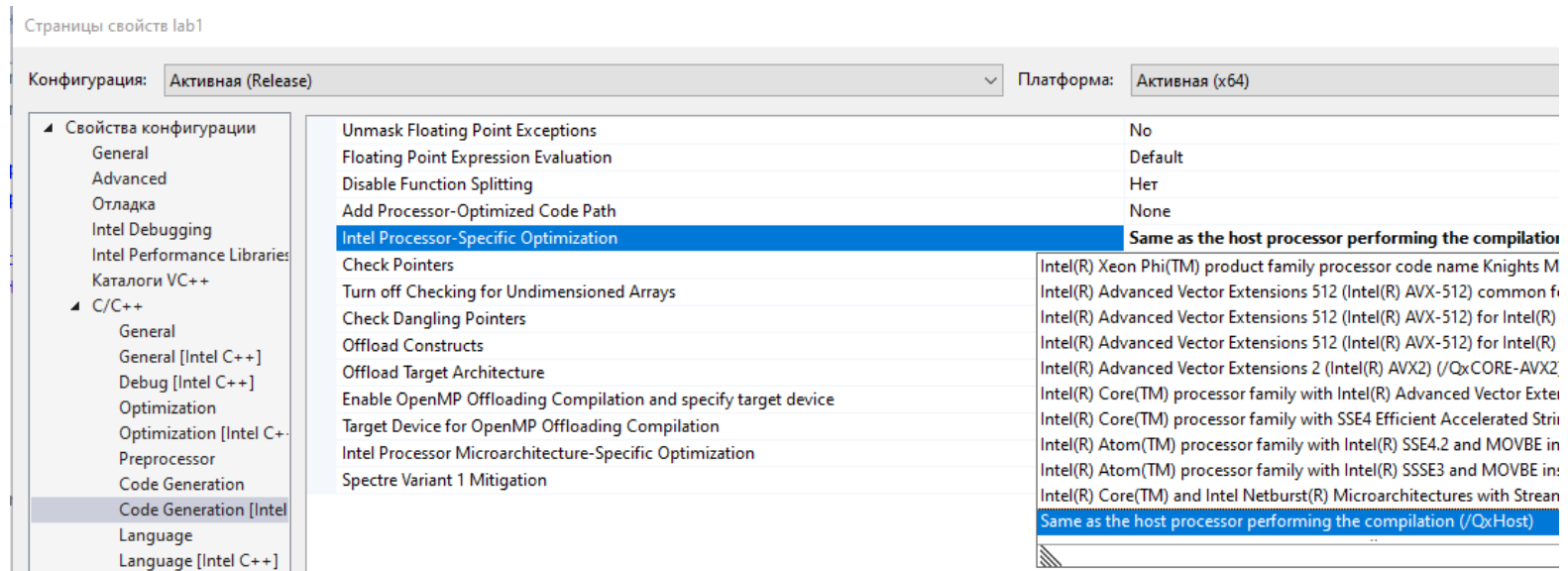
Страницы свойств labwork1



[Справка по ключам оптимизации /O](#)

# Компилятор Intel Classic C/C++

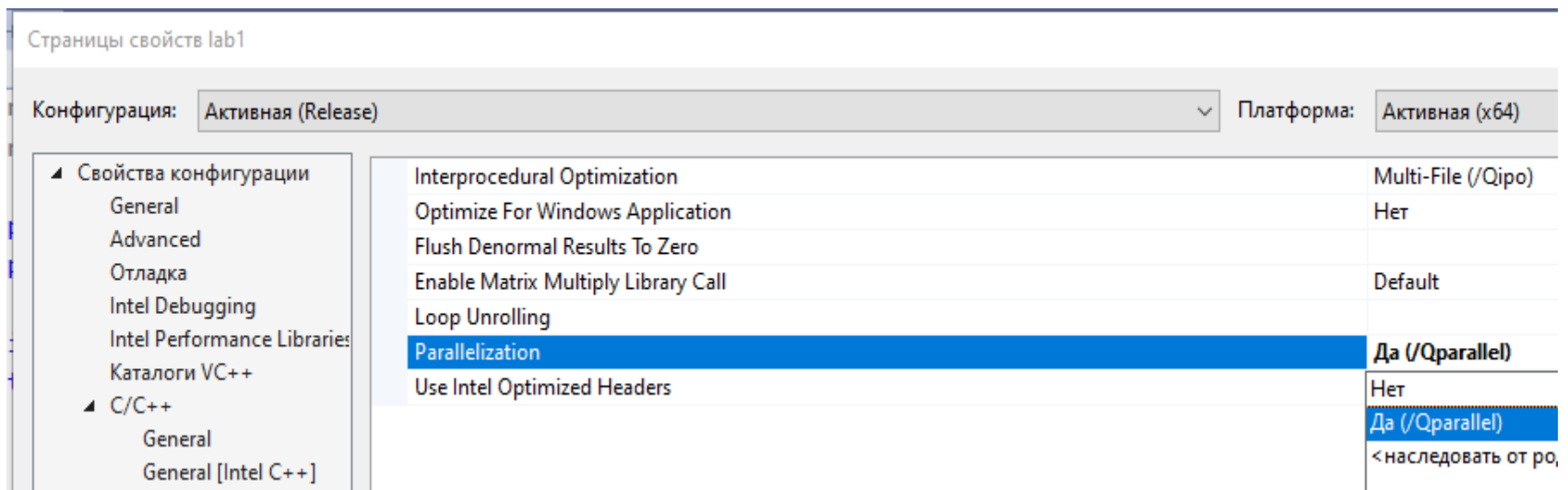
**/QxHost** - обеспечивает генерацию максимально современных векторных инструкций, поддерживаемых платформой



Справка по /QxHost

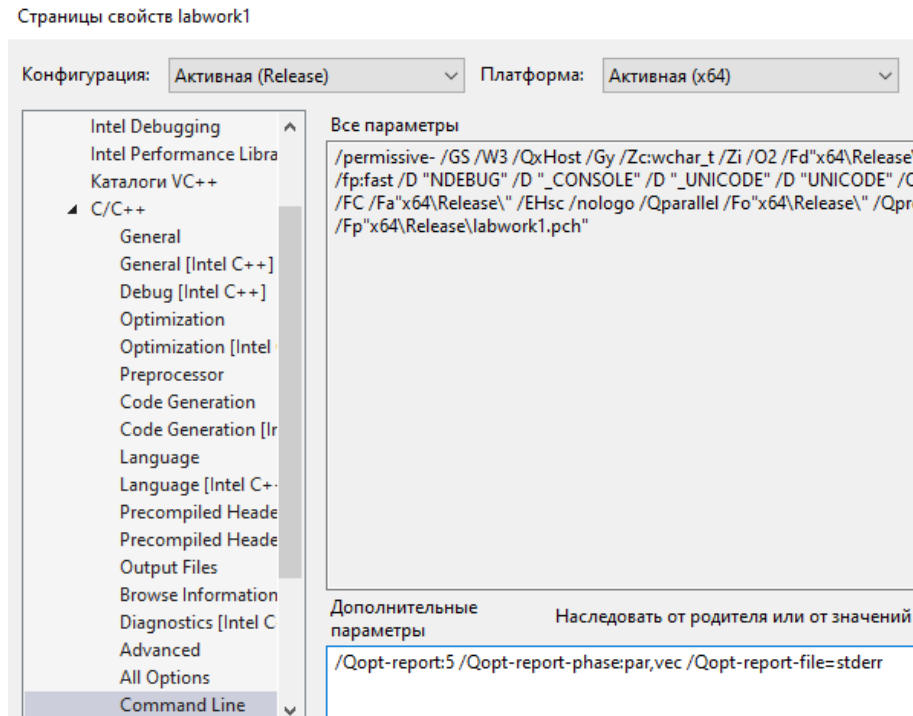
# Компилятор Intel Classic C/C++

**/Qparallel** – автораспараллеливание кода для использования многоядерности (происходит при выполнении определенных условий)



[Справка по /Qparallel](#)

# Компилятор Intel Classic C/C++



**/Qopt-report** – включает генерацию отчета об оптимизации,

[Справка по /Qopt-report](#)

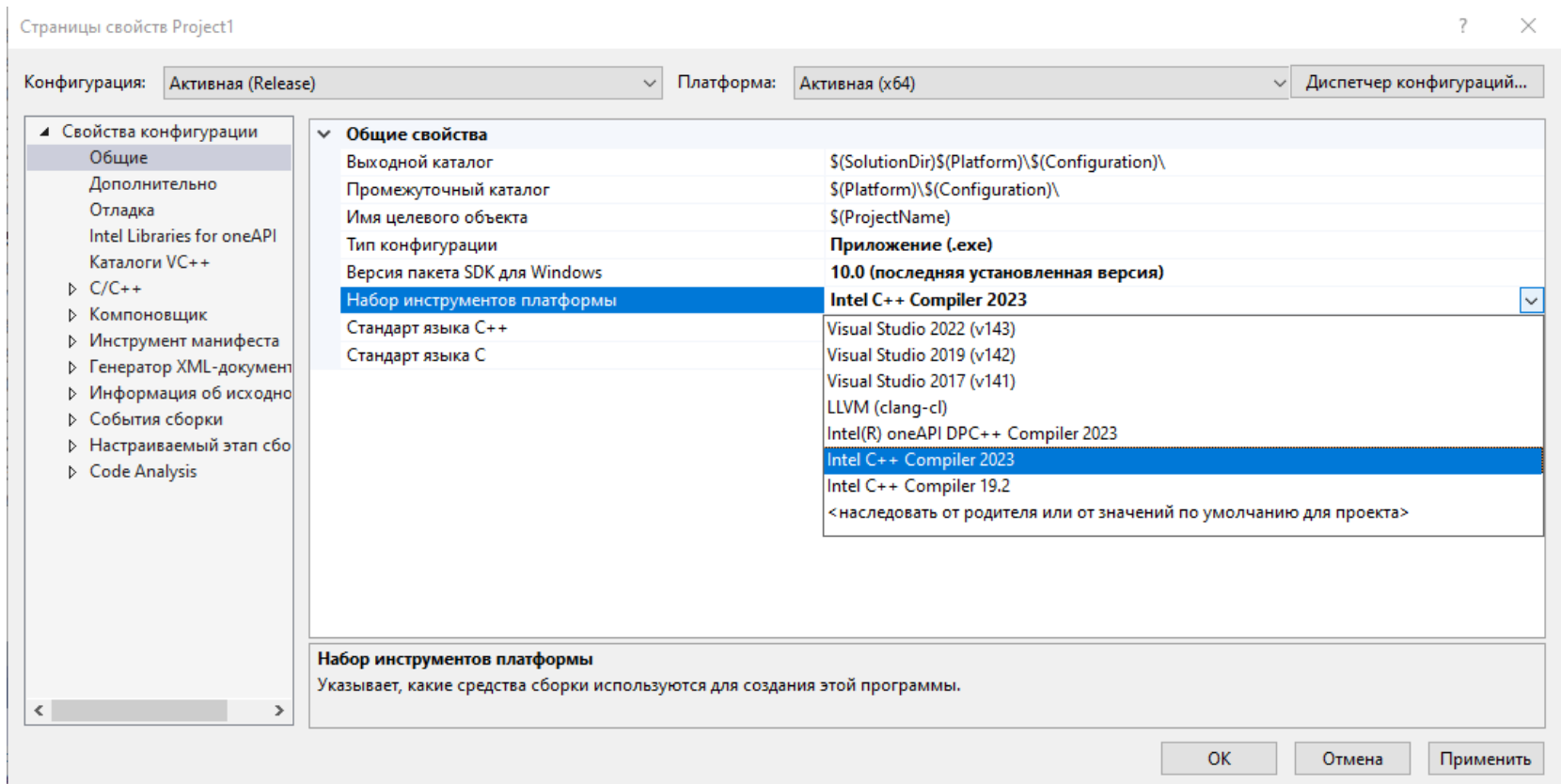
**/Qopt-report-phase** – указывает одну или несколько стадий для генерации отчета об оптимизации

[Справка по /Qopt-report-phase](#)

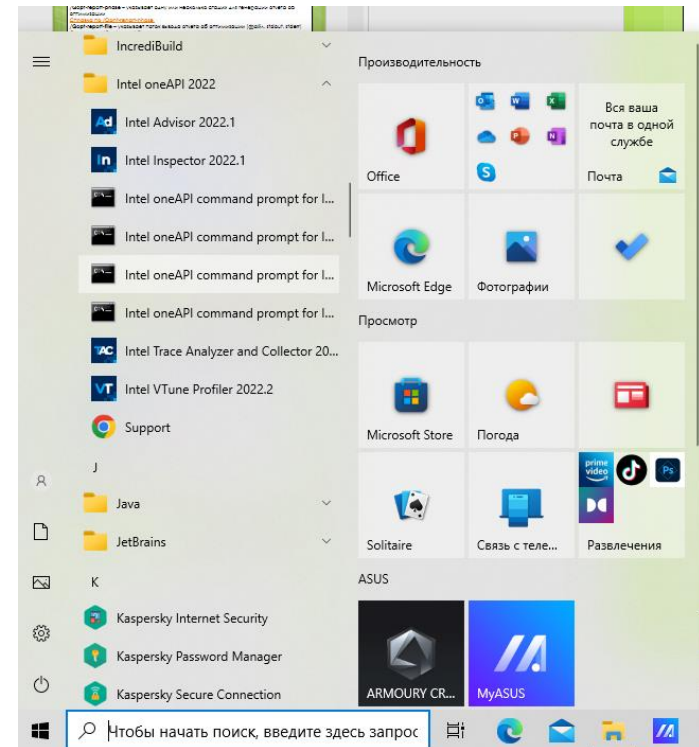
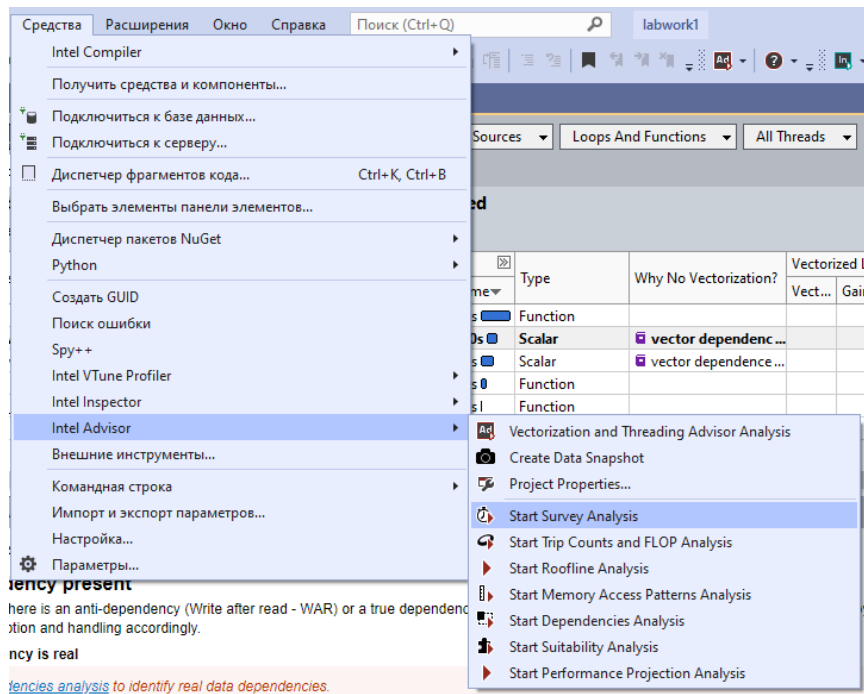
**/Qopt-report-file** – указывает поток вывода отчета об оптимизации (файл, stdout, stderr)

[Справка по /Qopt-report-file](#)

# Компилятор Intel Clang/LLVM C++



# Запуск Intel Advisor



## Справка по Intel Advisor

# Intel Advisor

The screenshot displays the Intel Advisor interface with the following components:

- Analysis Workflow:** Includes buttons for Accuracy (Low, Medium, High, Custom) and Overhead, and Analysis Types (Survey, Characterization, Memory Access Patterns, Dependencies).
- Performance Issues:** A table listing issues such as "1 Data type conversions present" and "3 Unoptimized floating point operation processing possible".
- Vectorization Summary:** A table showing the status of vectorization for various functions, including CPU Time, Type, Why No Vectorization?, and Vectorized Loops.
- Source Code:** A view of the source code for `Source.cpp:12 main`, highlighting a loop that can be vectorized.

Function Call Sites and Loops	CPU Time	Type	Why No Vectorization?	Vectorized Loops	Instruction Set Analysis
	Total Time	Self Time		Vect... Efficiency Gain... VL (...)	Traits
<code>_svm_log104_I9</code>	0,144s	0,144s	Vector Function	AVX2	
<code>[loop in main at Source.cpp:12]</code>	0,156s	0,084s	Vectorized (B...	AVX ~100% 4,73x 4	Divisions; Type Conv... Float6...
<code>[loop in main at Source.cpp:16]</code>	0,158s	0,072s	Vectorized (Bo...	AVX ~100% 4,73x 4	Divisions; Type Conve... Float64...
<code>[Import thunk _svm_log104_I9]</code>	0,014s	0,014s	Vector Function		
<code>invoke_main</code>	0,314s	0,000s	Inlined Function		
<code>__scrt_common_main_seh</code>	0,331s	0,000s	Function		
<code>main</code>	0,314s	0,000s	Function		Divisions; Extracts; Ty... Float32...

Line	Source	Total Time	%	Loop/Function Time	%	Traits
12	<code>for (int i = 1; i &lt; N; i+=2)</code>	26,814ms		156,170ms		
14	<code>sum += -1. / (1 - log10(i));</code>	56,989ms				Divisions

Справка по Vectorization Advisor



# Индивидуальное задание

№	Ряд	№	Ряд	№	Ряд
1.	$\sum_{n=1}^N \frac{(-1)^n}{(3n-1)^2}$	10.	$\sum_{n=1}^N \frac{(-1)^n}{(3n-2)(3n+1)}$	19.	$\sum_{n=1}^N \frac{(-1)^{n-1}}{\sqrt{n}}$
2.	$\sum_{n=1}^N \frac{\sqrt[3]{n}}{(n+1)\sqrt{n}}$	11.	$\sum_{n=1}^N (-1)^{n-1} \frac{2n+1}{n(n+1)}$	20.	$\sum_{n=1}^N \frac{(-1)^n}{(2n+1)^3 - 1}$
3.	$\sum_{n=1}^N \frac{(-1)^n}{(n+1)^2 - 1}$	12.	$\sum_{n=1}^N (-1)^n \frac{n+1}{(n+1)\sqrt{n+1} - 1}$	21.	$\sum_{n=1}^N \frac{(-1)^{n+1}}{2n - \sqrt{n}}$
4.	$\sum_{n=1}^N \frac{(-1)^{n-1}}{n^2}$	13.	$\sum_{n=1}^N \frac{\sin(2n+1)}{(n+1)^2 (n+2)^2}$	22.	$\sum_{n=1}^N \frac{(-1)^{n-1}}{\ln(n+1)}$
5.	$\sum_{n=1}^N (-1)^n \frac{\ln n}{n}$	14.	$\sum_{n=1}^N (-1)^{n-1} \operatorname{tg}\left(\frac{1}{n\sqrt{n}}\right)$	23.	$\sum_{n=2}^N \frac{(-1)^n}{n^3\sqrt{n} - \sqrt{n}}$
6.	$\sum_{n=1}^N \frac{\sin(1/n^2)}{(5n-1)^2}$	15.	$\sum_{n=1}^N \frac{\sin(5n+1)}{(6n+4)^2 (7n-1)^3}$	24.	$\sum_{n=1}^N \frac{\sin(2n-1)}{(2n-1)^2}$
7.	$\sum_{n=1}^N \frac{(-1)^{n-1}}{(2n-1)^2}$	16.	$\sum_{n=1}^N (-1)^n \ln\left(\frac{n^2+1}{n^2}\right)$	25.	$\sum_{n=1}^N \frac{(-1)^n}{n - \ln n}$
8.	$\sum_{n=2}^N \frac{1}{n \ln^2 n}$	17.	$\sum_{n=1}^N \frac{(-1)^n}{n(n+1)(n+2)}$		
9.	$\sum_{n=2}^N \frac{(-1)^{n-1}}{n^2 - n}$	18.	$\sum_{n=1}^N \left(1 - \cos\left(\frac{\pi}{n}\right)\right)$		

## Требования к оформлению отчета

- В отчет по проделанной работе включить:
  - 1) описание используемых компиляторов;
  - 2) скриншоты проверки корректности вычислений при различных размерностях;
  - 3) заполненные таблицы;
  - 4) скриншот успешной векторизации кода в Advisor,
  - 5) вывод.