



Курс «Параллельное
программирование»

Лабораторная работа №1. Параллельное вычисление суммы числового ряда

Юлдашев Артур Владимирович
art@ugatu.su

Спеле Владимир Владимирович
spele.vv@ugatu.su

**Кафедра
высокопроизводительных
вычислительных технологий и
систем (ВВТиС)**

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 DPC++/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 используется из консоли.

Сборка осуществляется командой:

icc/icpc -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 используется из консоли.

Сборка осуществляется 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 Intel oneAPI DPC++/C++

Входит в состав Intel OneAPI Base Toolkit и Intel OneAPI HPC Toolkit. В среде Windows возможна интеграция в Visual Studio.

В Linux используется из консоли.

Сборка осуществляется командой:

dpcpp -ключ_оптимизации имя_файла.c/cpp

Ключи оптимизации:

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

/O1 – оптимизация по размеру

/O2 – максимизация скорости

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 ключей оптимизации компилятора Microsoft C/C++.

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

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

Название компилятора	Время работы
/Qpar	

Задание

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

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

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

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

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

Название компилятора	Время работы
/QxHost	

Задание

Оценить быстродействие ключей оптимизации компилятора GNU C/C++.

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

Intel OneAPI Advisor

Summary

Survey Report

Refinement Reports

Annotation Report

Suitability Report

Elapsed time: 11.58s

Vectorized

Not Vectorized

FILTER: All Modules

All Sources

Loops	Vector Issues	Self Time	Total Time	Loop Type	Why No Vectorization?	Trip Counts	Vectorized Loops
							Vect. Efficiency
[loop in fGetEquilibriumF at lbpSUB.cpp:814]	1 Data ...	1.529s	1.529s	Scalar	loop control variable was found, but loop iteration ...	20	
[loop in fPropagationSwap at lbpSUB.cpp:1322]	2 Assu ...	1.334s	1.605s	Scalar	vector dependence prevents vectorization	9	
[loop in fGetSpeedSite at lbpGET.cpp:320]	1 Data ...	1.109s	1.109s	Scalar	loop control variable was found, but loop iteration ...	19	
[loop in fPropagationSwap at lbpSUB.cpp:1315]	2 Assu ...	0.731s	0.731s	Scalar	vector dependence prevents vectorization	4	
[loop in fGetOneMassSite at lbpGET.cpp:76]	1 Ineff ...	0.725s	0.725s	Vectorized V...		1; 2; 2	AVX 100%
[loop in fGetOneMassSite at lbpGET.cpp:76]		0.401s	0.401s	Vectorized V...		1	AVX
[loop in fGetOneMassSite at lbpGET.cpp:76]		0.234s	0.234s	Peel		1; 2	
[loop in fGetOneMassSite at lbpGET.cpp:76]		0.090s	0.090s	Remainder		1; 2	
[loop in fCalcInteraction_ShanChen at lbpFORC...	1 Ineff ...	0.650s	0.650s	Peel/Rem...		2	
[loop in fCalcInteraction_ShanChen at lbpFORC...	1 Data ...	0.556s	1.205s	Scalar	inner loop was already vectorized	1; 2	
[loop in fSiteFluidCollisionOK at lbp6GK.cpp:48]		0.360s	0.360s	Scalar	loop control variable was found, but loop iteration ...	19	
[loop in fCalcInteraction_ShanChen_Boundary ...]	1 Data ...	0.200s	0.410s	Scalar	inner loop was already vectorized	1; 2	
[loop in fCalcInteraction_ShanChen_Boundary ...]	1 Ineff ...	0.180s	0.180s	Peel/Rem...		2	

Source

Top Down

Loop Assembly

Recommendations

Compiler Diagnostic Details

File: lbpGET.cpp:320 fGetSpeedSite

Source

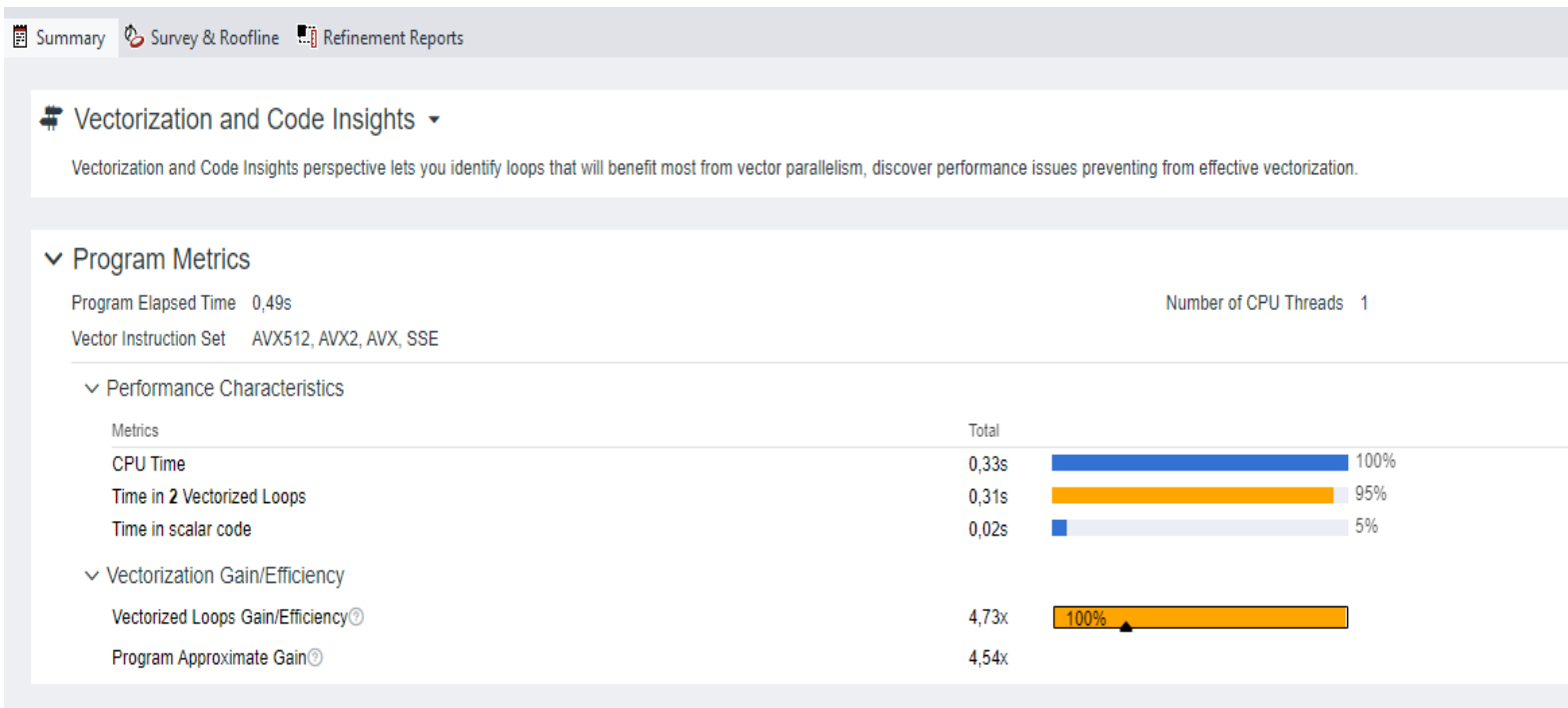
Total Time

%

318			
319	for(int i=0; i<lbay.nf; i++)		
320	{	109,76ms	0
	for(int i=0; i<lbay.nf; i++)		
	{		
	// Scalar loop in fGetSpeedSite at lbpGET.cpp:320		
	// Scalar Loop. Not vectorized: loop control variable was found, but loop iteration count cannot be computed before executing the loop		
	// No loop transformations were applied		
	}		
321	mass += pti[counter_ptil];	174,601ms	0
322	speed[0] += pti[counter_ptil] * lbv[i];	172,473ms	0
323	speed[1] += pti[counter_ptil] * lbv[i+lbay.npadd];	229,622ms	0
324	speed[2] += pti[counter_ptil] * lbv[i+lbay.npadd];	422,700ms	0
325	}		

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 UWP 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 лучше разделить вычислительный цикл на 2.

```
#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;
}
```

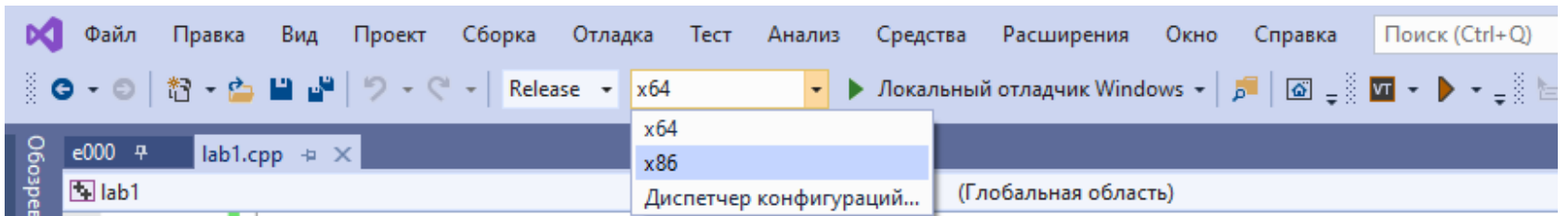
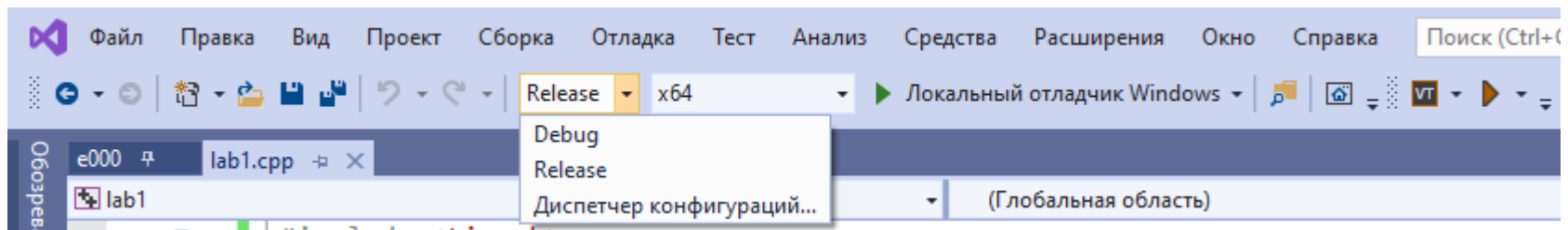
Результат работы программы

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

Компилятор Microsoft C/C++

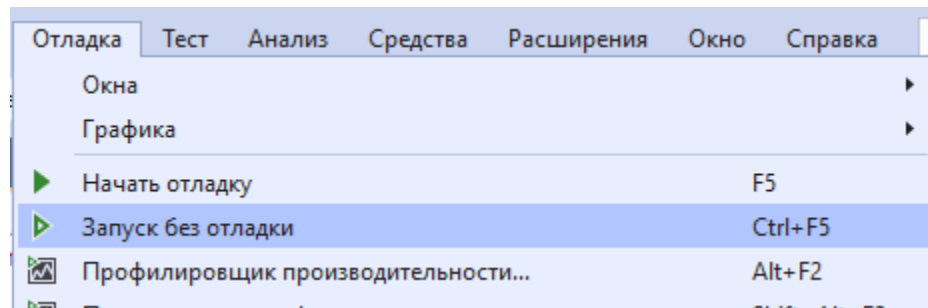
Выберите наиболее производительную конфигурацию:

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++

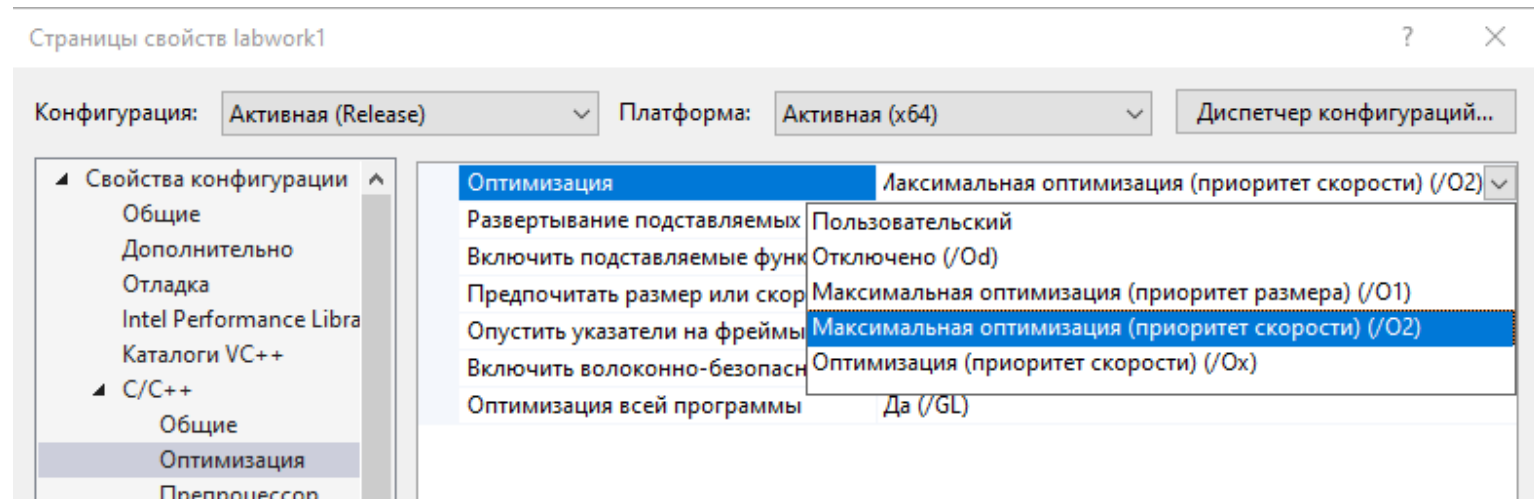
Ключи оптимизации:

/Od – отключение оптимизаций (параметр по умолчанию)

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

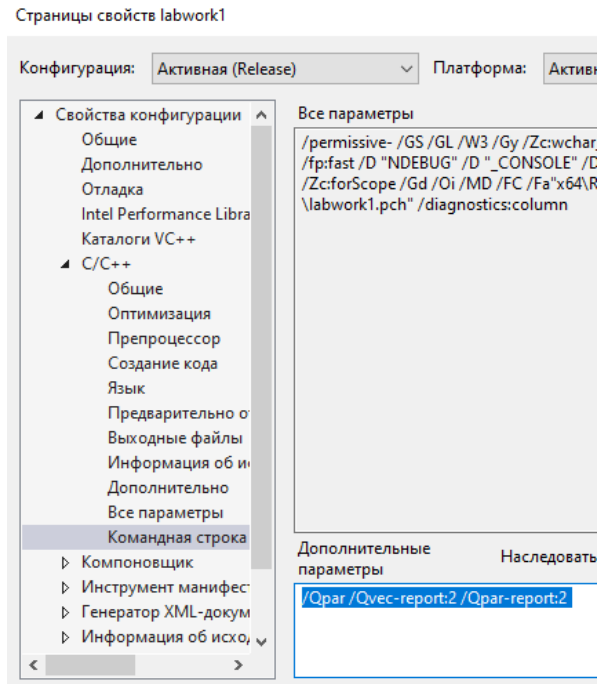
/O2 – максимальная оптимизация с приоритетом к увеличению скорости работы программы

/Ox – полная оптимизация,



[Справка о ключах оптимизации /O](#)

Компилятор Microsoft C/C++



/Qpar – включает автоматическую параллелизацию циклов в коде и векторизацию операций (происходит при выполнении определенных условий),

[Справка по /Qpar](#)

/Qpar-report, Qvec-report – вывод информации об автораспараллеливании и автовекторизации кода (происходит при выполнении определенных условий)

[Справка по /Qpar-report](#)

[Справка по /Qvec-report](#)

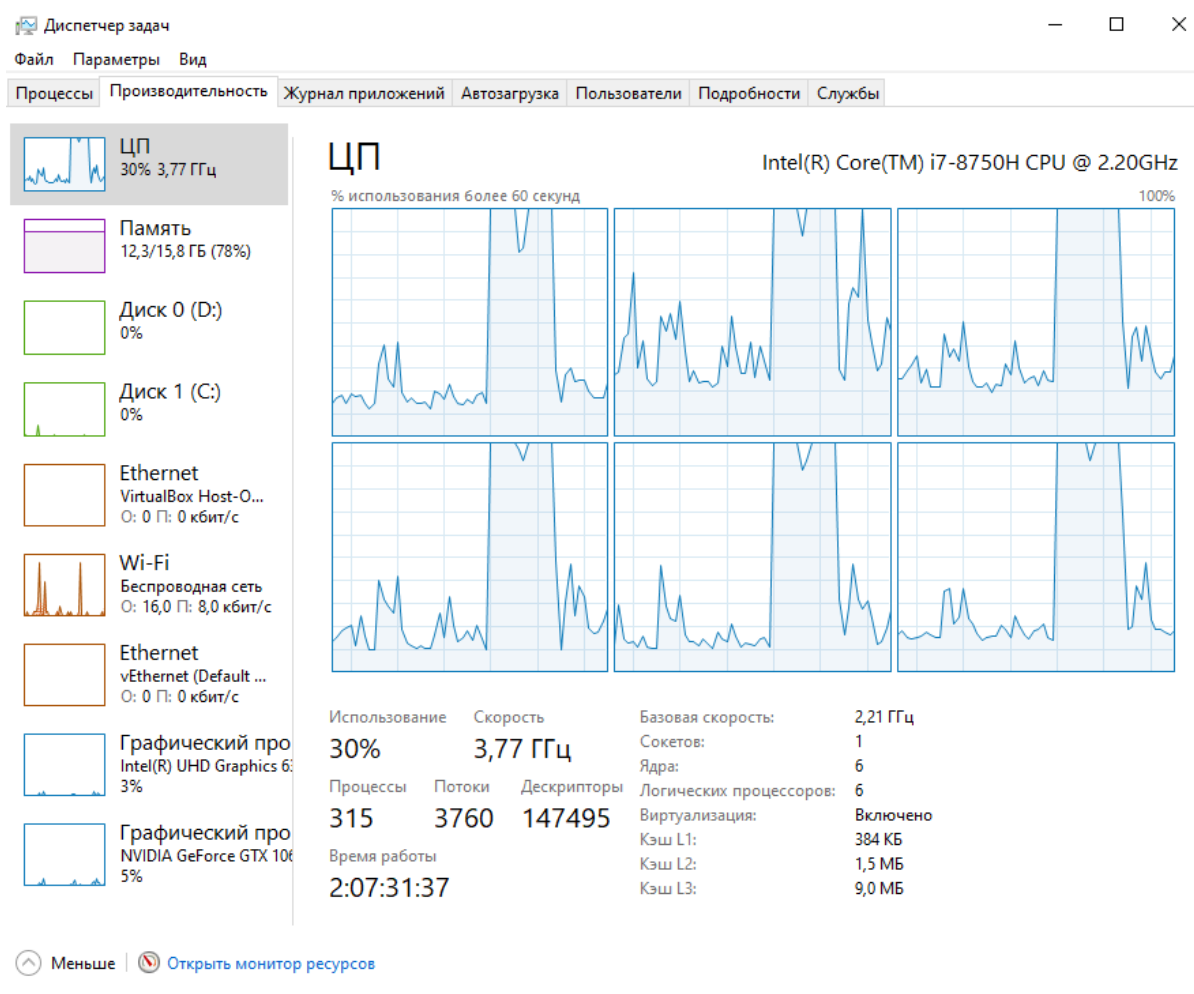
Компилятор Microsoft C/C++

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
```

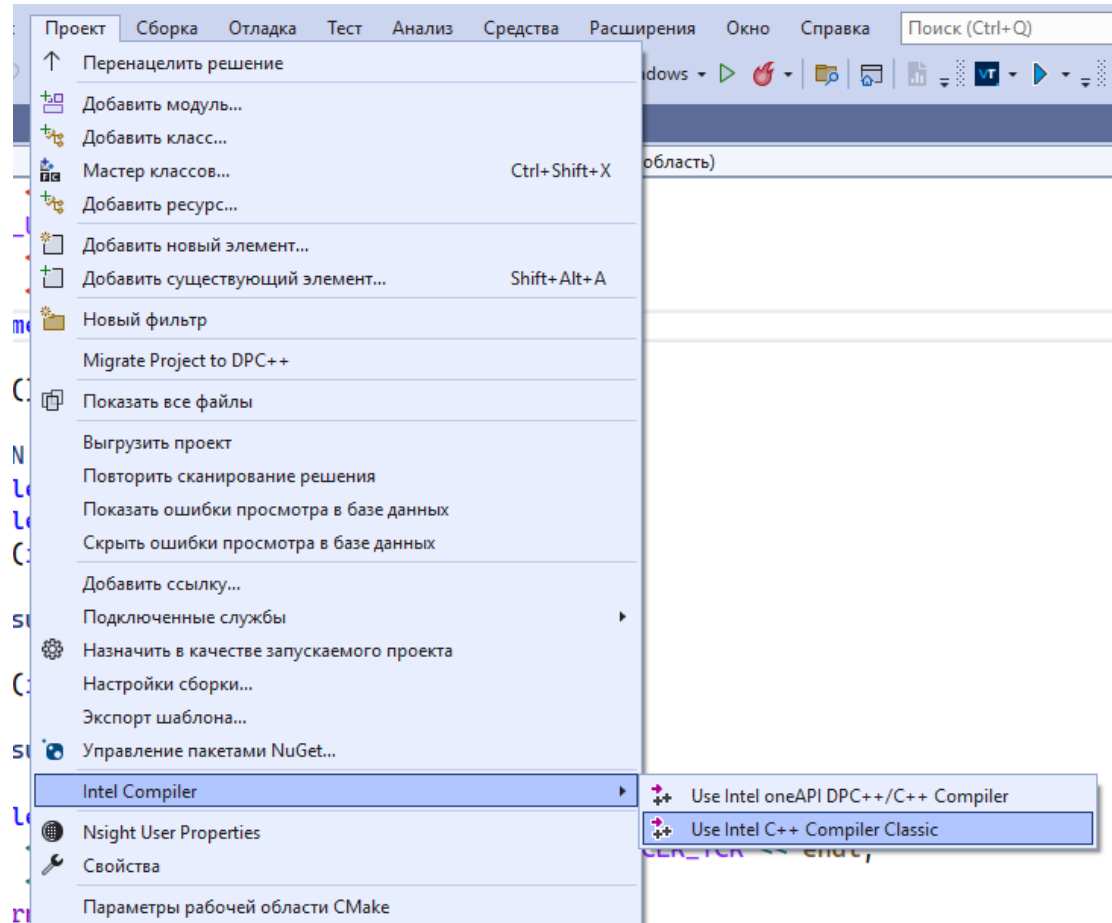
[Справка по кодам сообщений /Qpar-report и /Qvec-report](#)

Мониторинг загрузки CPU



Компилятор Intel Classic C/C++

Переход на компилятор Intel.



Компилятор Intel Classic C/C++

Ключи оптимизации:

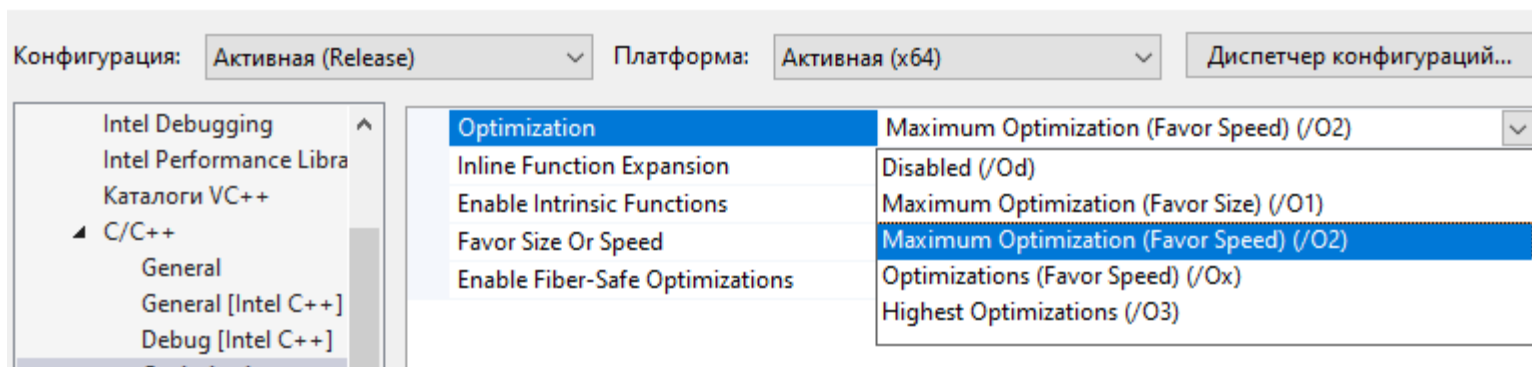
/Od(O0 в Linux) – отключение оптимизаций

/O1 – оптимизация по размеру

/O2 – максимизация скорости

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

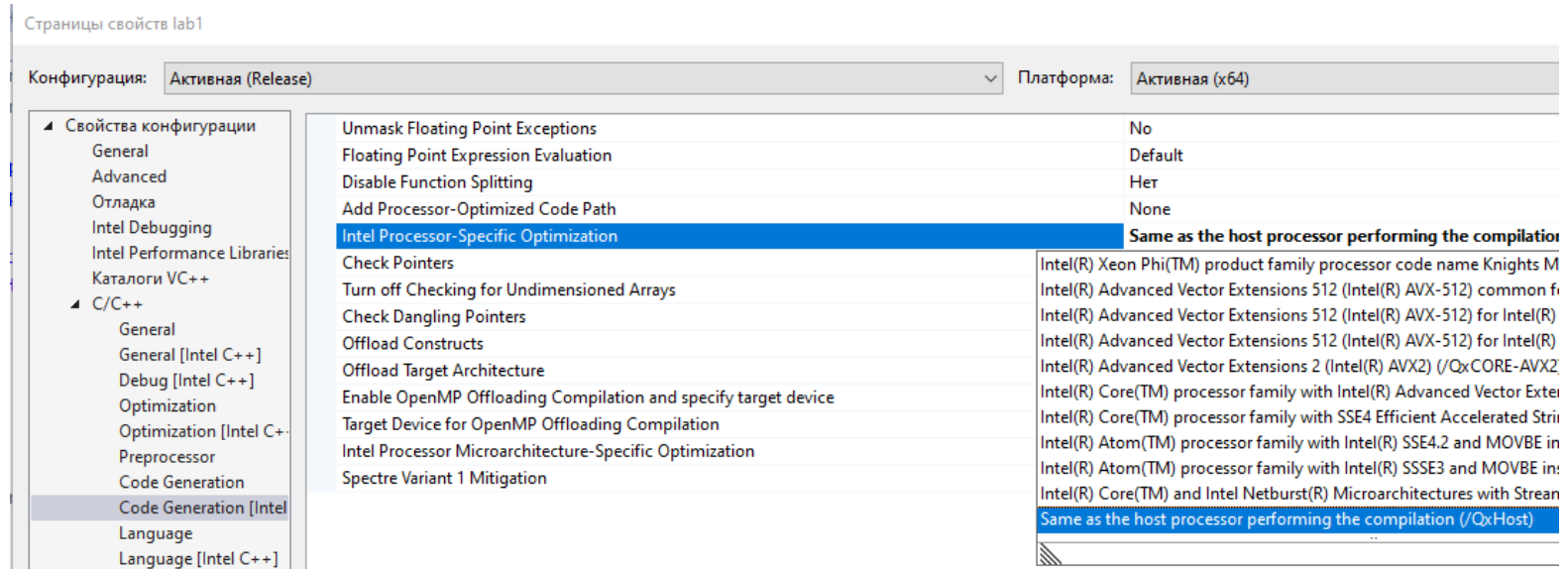
Страницы свойств labwork1



[Справка по ключам оптимизации /O](#)

Компилятор Intel Classic C/C++

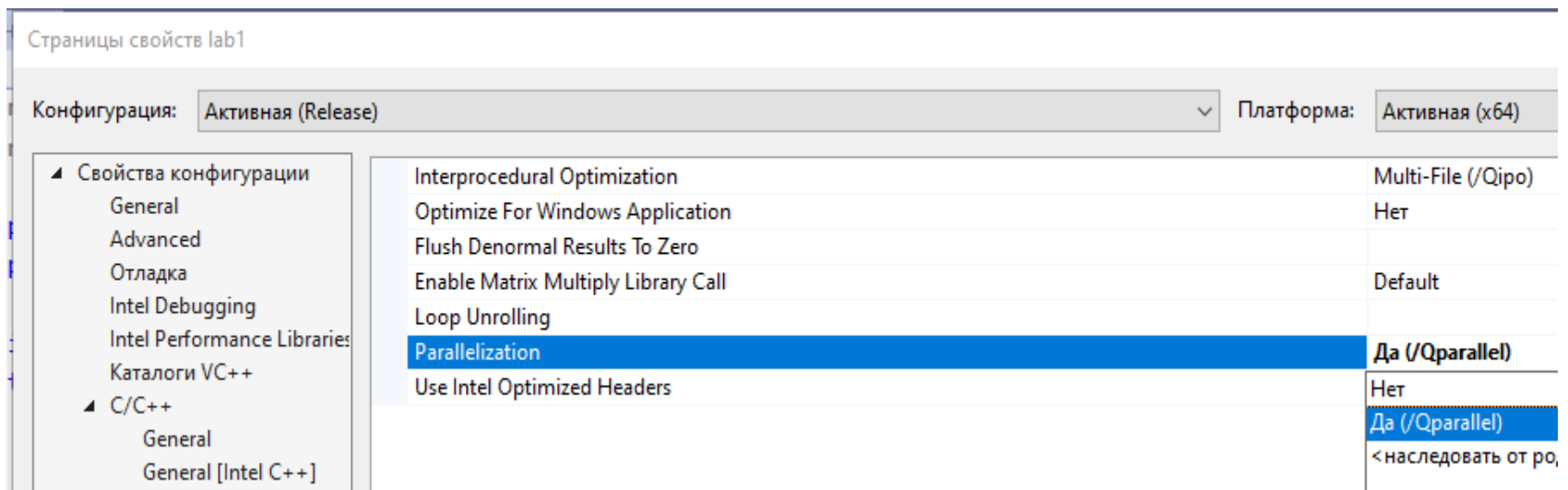
/QxHost - обеспечивает генерацию максимально современных векторных инструкций, поддерживаемых платформой



Справка по /QxHost

Компилятор Intel Classic C/C++

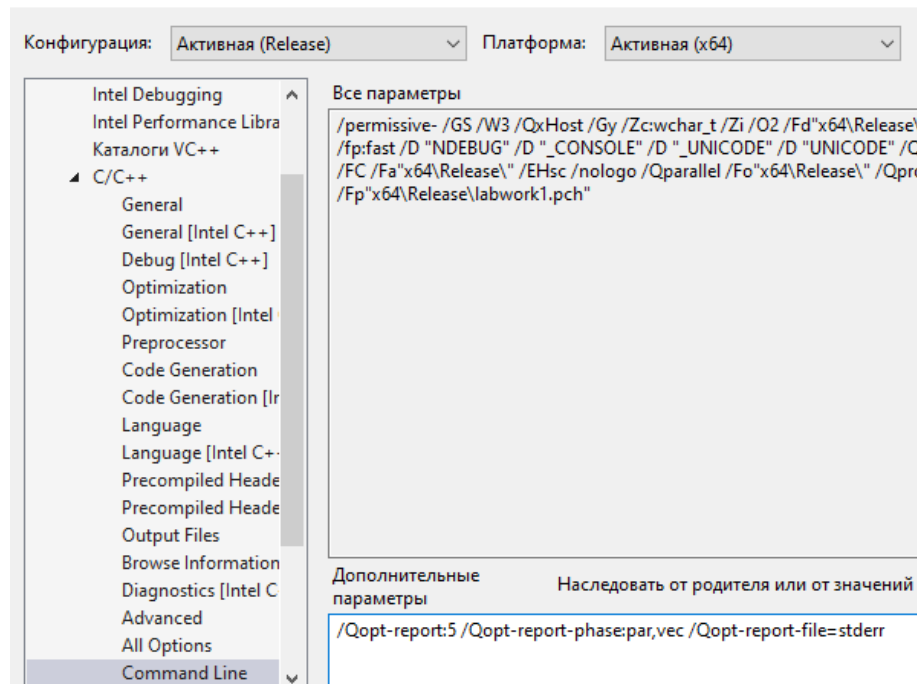
/Qparallel – автораспараллеливание кода на доступное число ядер (происходит при выполнении определенных условий)



[Справка по /Qparallel](#)

Компилятор Intel Classic C/C++

Страницы свойств labwork1



/Qopt-report – включает генерацию отчета об оптимизации,

[Справка по /Qopt-report](#)

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

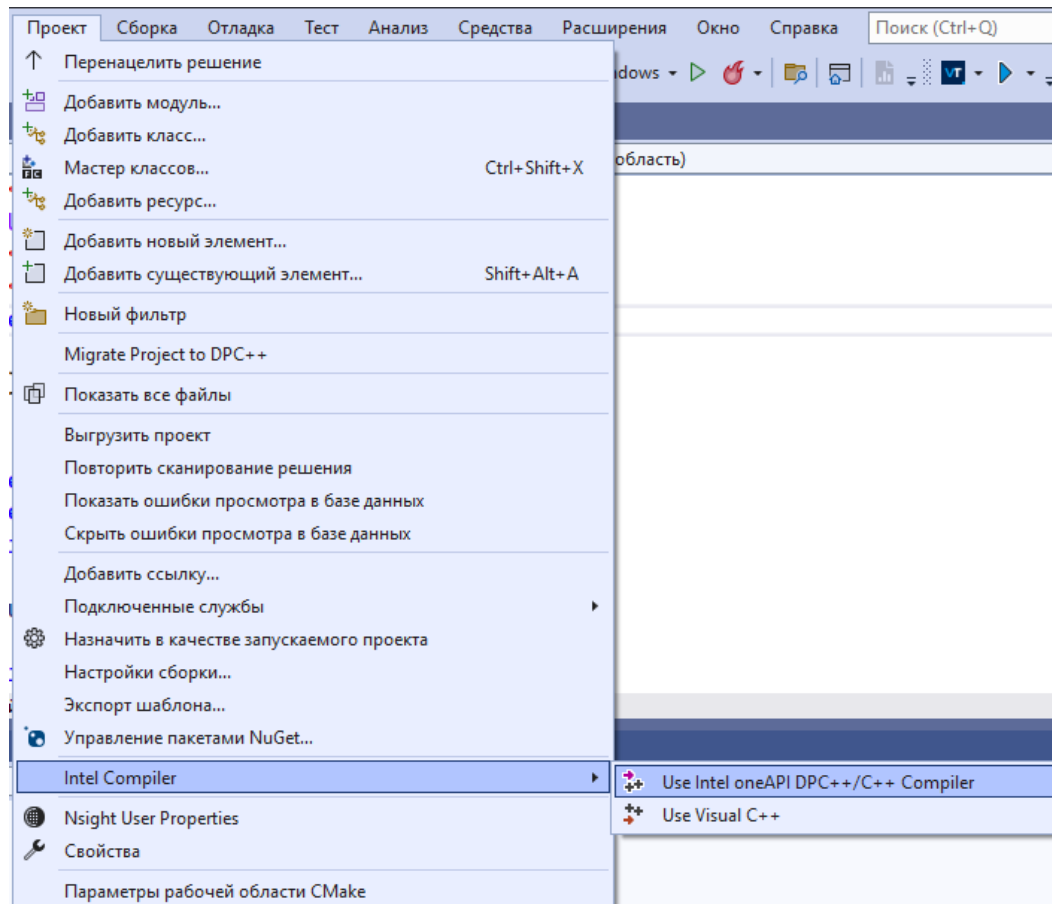
[Справка по /Qopt-report-phase](#)

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

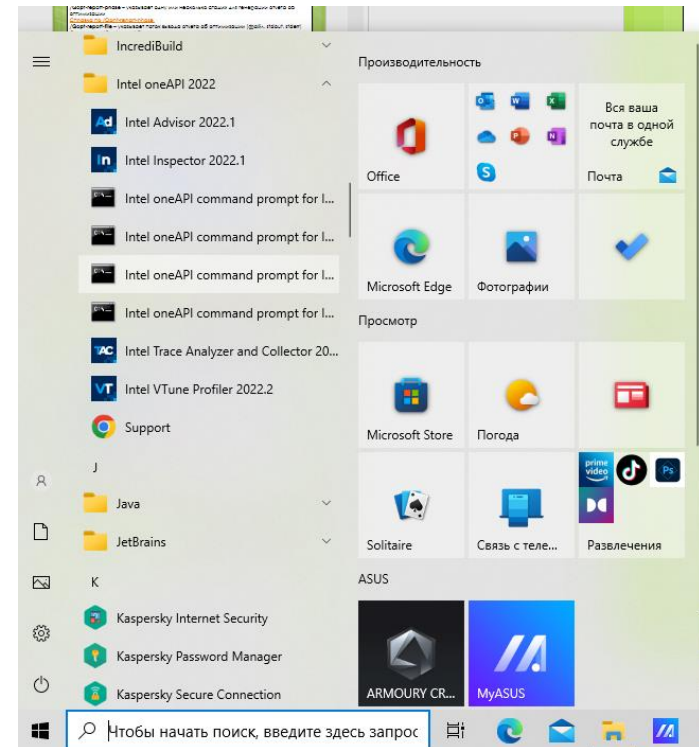
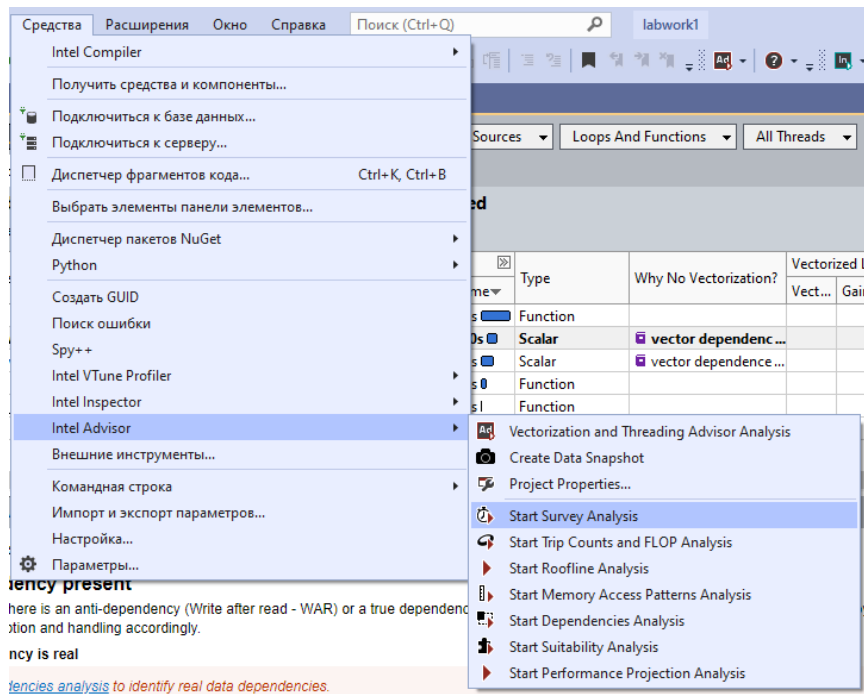
[Справка по /Qopt-report-file](#)

Компилятор Intel Clang/LLVM C++

Переход на компилятор Intel.



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 checkboxes for Analysis Types (Survey, Characterization, Memory Access Patterns, Dependencies, Reduction Detection).
- Performance Issues:** A table listing issues such as "1 Data type conversions present" and "3 Unoptimized floating point operation processing possible".
- Source Code:** A view of the source code for `Source.cpp:12 main`, showing a loop that processes `Float64` and `Int32` data types.
- Table:** A table summarizing the analysis results, including CPU Time, Type, Why No Vectorization?, Vectorized Loops, and Instruction Set Analysis.

Function Call Sites and Loops	Performance Issues	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>	<input type="checkbox"/> 1 Data type conversions present	0,144s	0,144s	Vector Function	AVX2	Appr. Reciprocals(AVX...
<code>[loop in main at Source.cpp:12]</code>	<input type="checkbox"/> 3 Unoptimized floating point operation processing possible	0,156s	0,084s	Vectorized (B...	AVX	Divisions; Type Conv...
<code>[loop in main at Source.cpp:16]</code>	<input type="checkbox"/> 3 Unoptimized floating point operation processing possible	0,158s	0,072s	Vectorized (Bo...	AVX	Divisions; Type Conve...
<code>[Import thunk _svm_log104_I9]</code>		0,014s	0,014s	Vector Function		
<code>invoke_main</code>		0,314s	0,000s	Inlined Function		
<code>_scr_common_main_seh</code>		0,331s	0,000s	Function		
<code>main</code>	<input type="checkbox"/> 1 Data type conversions present	0,314s	0,000s	Function		Divisions; Extracts; Ty...

```
1 #include <iostream>
2 #define _USE_MATH_DEFINES
3 #include <math.h>
4 #include <time.h>
5 using namespace std;
6
7 int main()
8 {
9     int N = 210000000;
10    double start_time = clock();
11    double sum = 0;
12    for (int i = 1; i < N; i+=2)
13    {
14        sum += -1. / (1 - log10(i));
15    }
16    for (int i = 2; i < N; i+=2)
17    {
18        sum += 1. / (1 - log10(i));
19    }
20    double end_time = clock();
21    cout << "TIME = " << (end_time - start_time) / CLK_TCK << endl;
```

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.