# Описание: логотип

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

Институт информационных технологий, математики и механики

Отчет по лабораторной работе на тему:

«**Вычисление многомерных интегралов с использованием многошаговой схемы (метод прямоугольников) с помощью OpenMP и Intel Threading Building Blocks**»

**Выполнил:** студент группы *381706-3*

*Савин Дмитрий Владимирович*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*подпись*

**Проверил:**  преподаватель кафедры МОСТ ИИТММ

*Козинов Евгений Александрович*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*подпись*

Нижний Новгород

2020

# Содержание

* Постановка задачи
* Методы решения
* 1.OpenMP
* 2. Intel Threading Building Blocks
* Схема распараллеливания
* Описание программной реализации
* Подтверждение корректности
* Результаты экспериментов
* Заключение
* Список литературы
* Приложение

# Постановка задачи

Реализовать последовательный и параллельные методы вычислений многомерных интегралов с использованием многошаговой схемы методом прямоугольников с использованием OpenMP и TBB.

# Методы решения

## 1.OpenMP

OpenMP реализует [параллельные вычисления](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F) с помощью [многопоточности](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D0%BF%D0%BE%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D1%81%D1%82%D1%8C" \o "Многопоточность), в которой «главный» (master) [поток](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%82%D0%BE%D0%BA_%D0%B2%D1%8B%D0%BF%D0%BE%D0%BB%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F) создает набор подчиненных (slave) потоков и задача распределяется между ними. Предполагается, что потоки выполняются параллельно на [машине с несколькими процессорами](https://ru.wikipedia.org/wiki/%D0%9C%D1%83%D0%BB%D1%8C%D1%82%D0%B8%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80) (количество процессоров не обязательно должно быть больше или равно количеству потоков).

Задачи, выполняемые потоками параллельно, так же, как и данные, требуемые для выполнения этих задач, описываются с помощью специальных директив [препроцессора](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80) соответствующего языка — прагм.

Количество создаваемых потоков может регулироваться как самой программой при помощи вызова библиотечных процедур, так и извне, при помощи переменных окружения. [1]

## 2.Intel Threading Building Blocks

Intel Threading Building Blocks (также известная как TBB) — кроссплатформенная библиотека шаблонов С++, разработанная компанией [Intel](https://ru.wikipedia.org/wiki/Intel" \o "Intel) для параллельного программирования. Библиотека содержит алгоритмы и структуры данных, позволяющие программисту избежать многих сложностей, возникающих при использовании традиционных реализаций потоков, таких как [POSIX Threads](https://ru.wikipedia.org/wiki/POSIX_Threads), Windows threads или [Boost Threads](https://ru.wikipedia.org/wiki/Boost" \o "Boost), в которых создаются отдельные потоки исполнения, синхронизируемые и останавливаемые вручную. Библиотека TBB абстрагирует доступ к отдельным потокам. Все операции трактуются как «задачи», которые динамически распределяются между ядрами процессора. Кроме того, достигается эффективное использование [кэша](https://ru.wikipedia.org/wiki/%D0%9A%D1%8D%D1%88). Программа, написанная с использованием TBB, создаёт, синхронизирует и разрешает графы зависимостей задач в соответствии с алгоритмом. Затем задачи исполняются в соответствии с зависимостями. Этот подход позволяет программировать параллельные алгоритмы на высоком уровне, абстрагируясь от деталей архитектуры конкретной машины. [2]

# Схема распараллеливания

## 1.OpenMP

double calc\_integral\_1D\_openmp(double x, double y1, double y2, int ny)

{

double sum = 0;

double dy = (y2 - y1) / ny;

int iy = 0;

#pragma omp parallel reduction(+:sum)

{

#pragma omp for schedule(static)

for (iy = 0; iy < ny; iy++)

{

double y = y1 + iy \* dy + dy \* .5; // central

double f = fun\_xy(x, y); // значение функции в точке

double d = fun\_D(x, y); // проверка, попадает ли точка в область интегрирования

sum += f \* d \* dy;

}

}

return sum;

Функция распараллеливает подсчет суммы по dy.

## }2.Intel Threading Building Blocks

double calc\_integral\_tbb(double x1, double x2, double y1, double y2, int nx, int ny) {

double dx = (x2 - x1) / nx;

double sum = 0;

parallel\_for (blocked\_range<int>(0, nx),

[&](const blocked\_range<int>& r)

{

for (int ix = r.begin(); ix != r.end(); ++ix)

{

double x = x1 + ix \* dx + dx \* .5; // central

double integral\_1D = calc\_integral\_1D(x, y1, y2, ny);

sum += integral\_1D \* dx;

}

}

);

return sum;

}

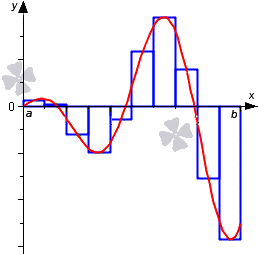


Рис 1. График функции с выделенными прямоугольниками.

# Описание программной реализации

double fun\_xy(double x, double y) – вычисление двойного интеграла от функции.

double fun\_D(double x, double y) – область интегрирования.

double calc\_integral(double x1, double x2, double y1, double y2, int nx, int ny) – функция вычисляющая центр прямоугольников по координатам X и Y,и считающая площадь прямоугольников, а так же проверяющая попадает ли точка в область интегрирования;

double calc\_integral\_openmp(double x1, double x2, double y1, double y2, int nx, int ny) – функция вычисляющая центр прямоугольников по координатам X и Y,и считающая площадь прямоугольников, а так же проверяющая попадает ли точка в область интегрирования для версии OpenMP.

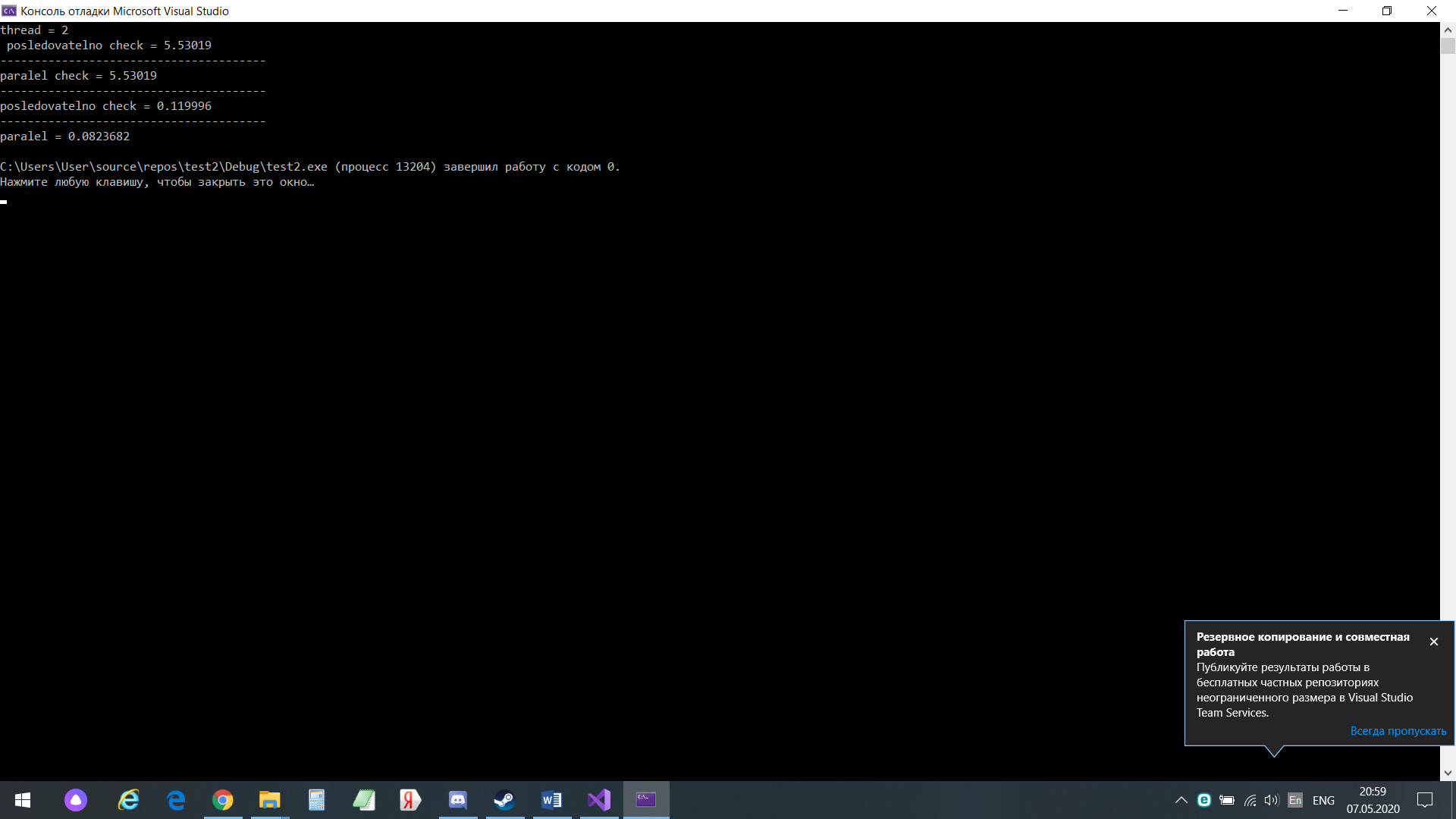
double calc\_integral\_tbb(double x1, double x2, double y1, double y2, int nx, int ny) – функция вычисляющая центр прямоугольников по координатам X и Y,и считающая площадь прямоугольников, а так же проверяющая попадает ли точка в область интегрирования для версии tbb.

# Подтверждение корректности

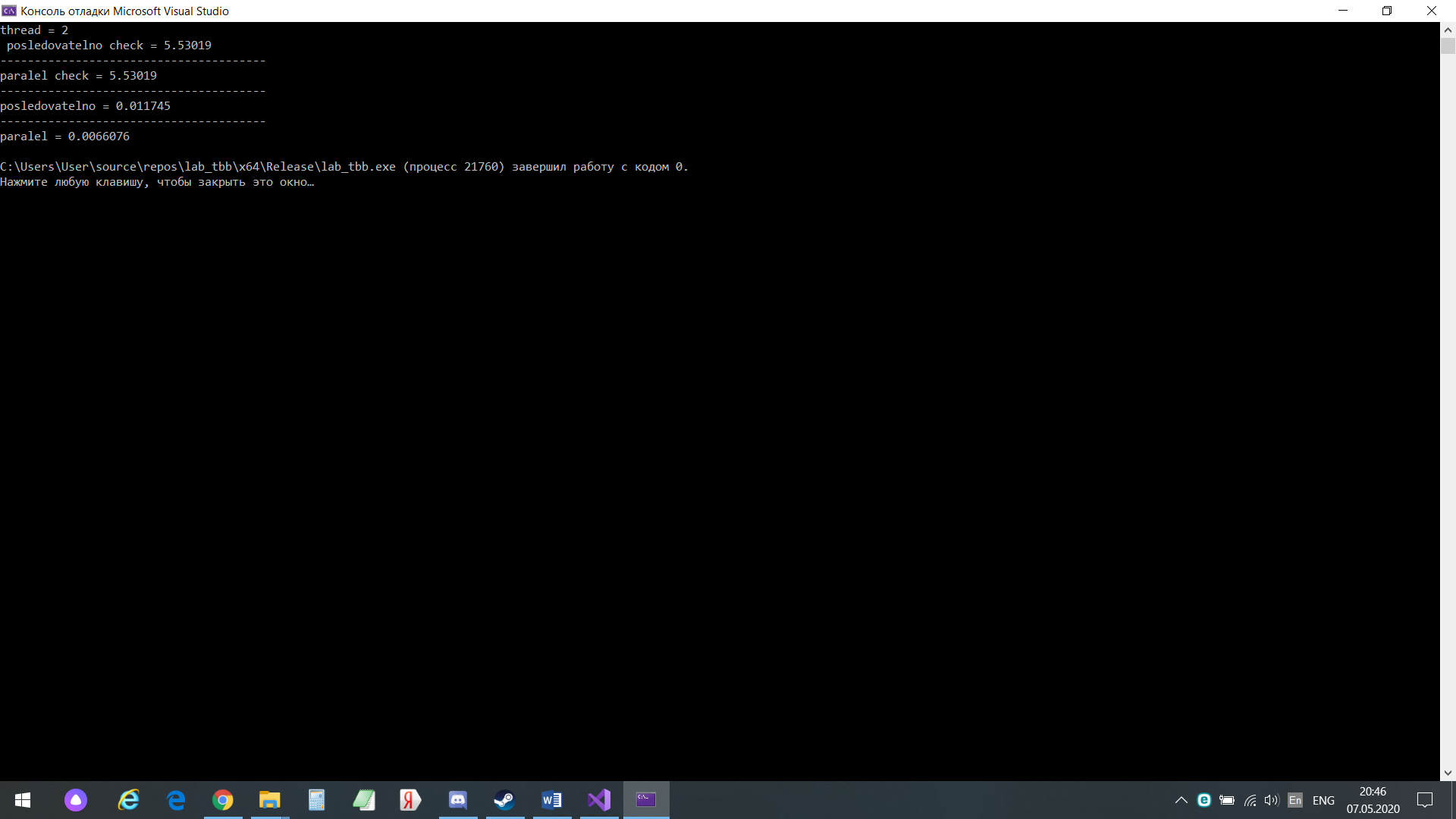
double check\_sum = calc\_integral(x1, x2, y1, y2, 1000, 1000) – функция выполнения подсчета площади последовательным методом работы программы, а double check\_sum\_openmp = calc\_integral\_openmp(x1, x2, y1, y2, 1000, 1000) - функция выполнения подсчета площади параллельным методом OpenMP работы программы и double check\_sum\_tbb = calc\_integral\_tbb(x1, x2, y1, y2, 1000, 1000) - функция выполнения подсчета площади параллельным методом TBB работы программы.

# Результаты экспериментов

Запуск программы с 2 потоками OpenMP.



Запуск программы с 2 потоками TBB.



**Результаты экспериментов**

Результат экспериментов на персоанальном компьютере.

Характеристики:

* Intel(R) Core(TM) i7-4750HQ @ 2.00GHz
* 12 GB RAM

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Количество процессов | Линейный алгоритм | 2  процесса  OpenMP | 8  процесса  OpenMP | 12  процессов  OpenMP | 2 процесса  TBB | 8  процесса  TBB | 12 процессов  TBB |
| Время работы | 0,13 | 0.09 | 0.035 | 0.03 | 0,065 | 0,041 | 0.028 |
| Ускорение | 1 | 1,44 | 3,22 | 4.33 | 2 | 3,75 | 4,6 |

# Заключение

Таким образом, наилучший показатель производительности получился с использованием Threading Building Blocks нежели OpenMP. Несмотря на это обе библиотека показали себя с наилучшей стороны, что выразилось в ускорении времени работы программы. Однако Threading Building Blocks и OpenMP имеют следующие достоинства и недостатки:

OpenMP:

Достоинства: за счет идеи "инкрементального распараллеливания" OpenMP идеально подходит для разработчиков, желающих быстро распараллелить свои вычислительные программы с большими параллельными циклами. Разработчик не создает новую параллельную программу, а просто последовательно добавляет в текст последовательной программы OpenMP-директивы.

Недостатки: не использует GPU, не поддерживает многие нужные для «продвинутого» контроля и синхронизации потоков функции; не всегда обеспечивает хорошую производительность; неоптимально «комбинируется» в различных компонентах приложения и с другими средствами распараллеливания.

Threading Building Blocks:

Достоинства: отличная производительность, независимость от компилятора, возможность использовать отдельные компоненты библиотеки независимо.

Недостаток: не использует GPU.

# Список литературы

[1]. <https://www.openmp.org/>

[2]. <https://ru.wikipedia.org/wiki/Intel_Threading_Building_Blocks>

# Приложение