

**Отчет по**

**Лабораторной работе № 2. (2021, весна)**

**Параллельные циклы OpenMP**

**Выполнил:**

Гломадов Антон – студент 3 курса Прикладной математики и информатики 1 подгруппа

2021г.

**Цель работы**: научиться пользоваться работать с for параллельно.

**Задачи**:

1. Изучите OpenMP - директиву параллельного выполнения цикла for. Напишите программу, в которой k нитей параллельно вычисляют сумму чисел от 1 до N. Распределите работу по нитям с помощью OpenMP - директивы for.  **Входные данные:** целое число k – количество нитей, целое число N – количество чисел.  **Выходные данные**: каждая нить выводит свою частичную сумму в формате «[Номер\_нити]: Sum = <частичная\_сумма>», один раз выводится общая сумма в формате «Sum = <сумма>».
2. Изучите параметр schedule директивы for. Модифицируйте программу «Сумма чисел» из задания выше таким образом, чтобы дополнительно выводилось на экран сообщение о том, какая нить, какую итерацию цикла выполняет: [<Номер нити>]: calculation of the iteration number <Номер итерации>. Задайте k = 4, N = 10. Выведите результаты для schedule (static; static,1; static,2; dynamic; dynamic,2).
3. Напишите OpenMP-программу, которая вычисляет произведение двух квадратных матриц размера N x N.
4. Перепишите программу, в которой параллельно вычисляется сумма чисел от 1 до N, без использования параметра reduction. Вместо параметра reduction используйте директиву atomic.

**Листинг для 1 задачи:**

#pragma once

#include <omp.h>

#include <iostream>

/\*\*

\*Изучите OpenMP - директиву параллельного выполнения цикла for. Напишите программу, в которой k нитей параллельно вычисляют сумму чисел от 1 до N. Распределите работу по нитям с помощью OpenMP - директивы for.

\* Входные данные: целое число k – количество нитей, целое число N – количество чисел.

\* Выходные данные: каждая нить выводит свою частичную сумму в формате

\* «[Номер\_нити]: Sum = <частичная\_сумма>», один раз выводится общая сумма в формате «Sum = <сумма>».

\*/

**void** **FirstTask**(){

std::cout<<"Задание 1**\n**";

**int** countOfThreads = **0**;

**int** nums = **0**;

std::cout<<"Введите количество потоков: ";

std::cin>>countOfThreads;

**if** (countOfThreads>omp\_get\_max\_threads()){

countOfThreads = omp\_get\_max\_threads();

std::cout<<"Максимальное допустимое число потоков: "<<omp\_get\_max\_threads()<<"**\n**Кол-во потов выставлено на максимальное допустимое значение**\n**";

}

std::cout<<"Введите количество чисел: ";

std::cin>>nums;

omp\_set\_num\_threads(countOfThreads);

**int** sum = **0**;

#pragma omp parallel reduction(+:sum)

{

#pragma omp **for**

**for** (**int** i = **0**; i <= nums; i++)

sum += i;

#pragma omp critical

{

std::cout << omp\_get\_thread\_num() << " Sum = " << sum << std::endl;

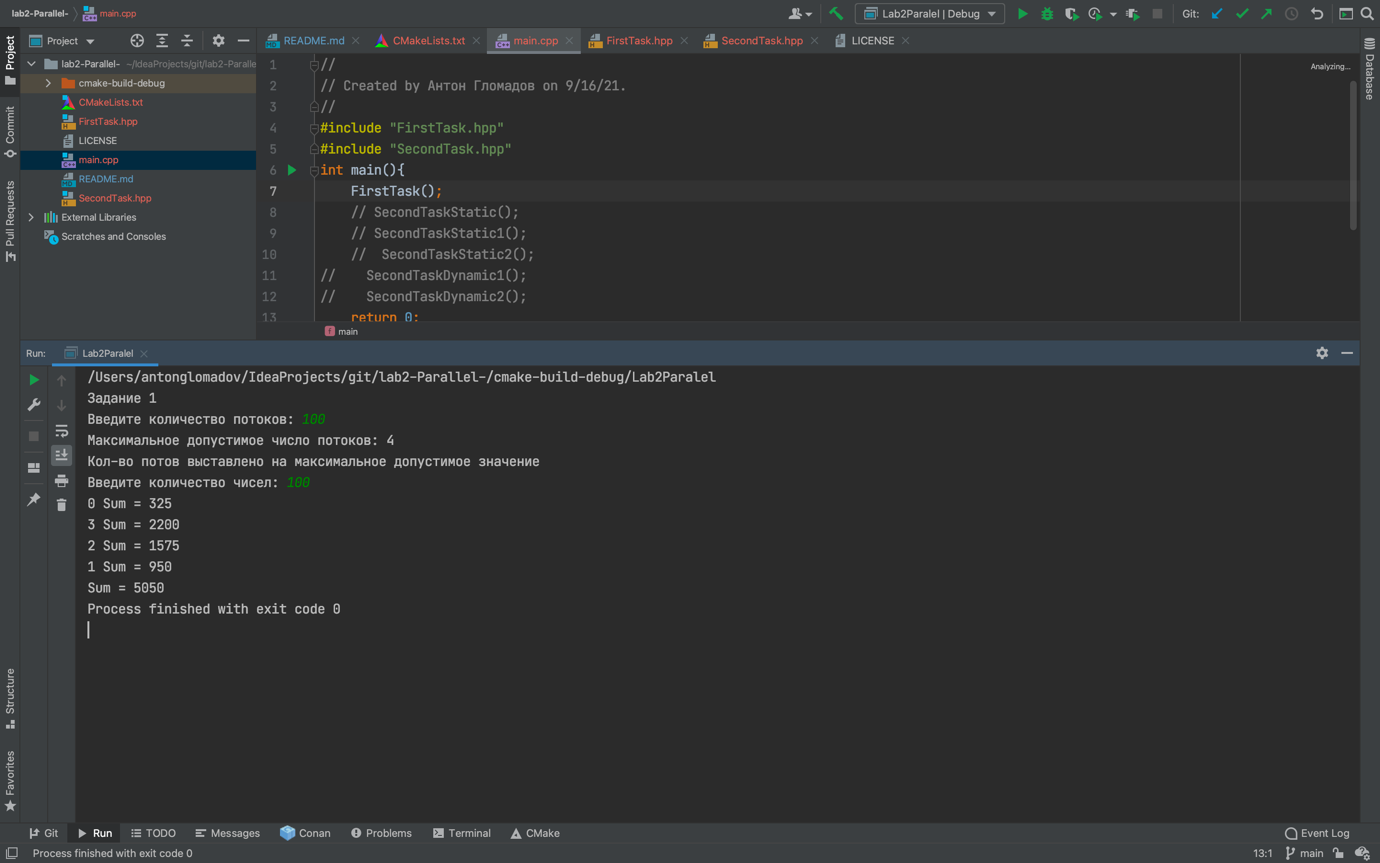
}

}

std::cout << "Sum = " << sum;

}

**Вывод для 1 задачи:**



**Листинг для 2 задачи:**

#pragma once

#include <omp.h>

#include <iostream>

/\*\*

\*Изучите параметр schedule директивы for. Модифицируйте программу «Сумма чисел» из задания выше таким образом,

\* чтобы дополнительно выводилось на экран сообщение о том, какая нить, какую итерацию цикла выполняет:

\* [<Номер нити>]: calculation of the iteration number <Номер итерации>.

\* Задайте k = 4, N = 10. Выведите результаты для schedule (static; static,1; static,2; dynamic; dynamic,2).

\*/

**void** **SecondTaskStatic**(){

std::cout<<"Задание 2.0**\n**";

**int** countOfThreads = **0**;

**int** nums = **0**;

std::cout<<"Введите количество потоков: ";

std::cin>>countOfThreads;

**if** (countOfThreads>omp\_get\_max\_threads()){

countOfThreads = omp\_get\_max\_threads();

std::cout<<"Максимальное допустимое число потоков: "<<omp\_get\_max\_threads()<<"**\n**Кол-во потов выставлено на максимальное допустимое значение**\n**";

}

std::cout<<"Введите количество чисел: ";

std::cin>>nums;

omp\_set\_num\_threads(countOfThreads);

**int** sum = **0**;

#pragma omp parallel reduction(+:sum)

{

#pragma omp **for** schedule(**static**)

**for** (**int** i = **0**; i <= nums; i++){

sum += i;

#pragma omp critical

{

std::cout << omp\_get\_thread\_num() << ": calculation of the iteration number " << i << std::endl;

}

}

#pragma omp critical

{

std::cout << omp\_get\_thread\_num() << " Sum = " << sum << std::endl;

}

}

std::cout << "Sum = " << sum;

}

**void** **SecondTaskStatic1**(){

std::cout<<"Задание 2.1**\n**";

**int** countOfThreads = **0**;

**int** nums = **0**;

std::cout<<"Введите количество потоков: ";

std::cin>>countOfThreads;

**if** (countOfThreads>omp\_get\_max\_threads()){

countOfThreads = omp\_get\_max\_threads();

std::cout<<"Максимальное допустимое число потоков: "<<omp\_get\_max\_threads()<<"**\n**Кол-во потов выставлено на максимальное допустимое значение**\n**";

}

std::cout<<"Введите количество чисел: ";

std::cin>>nums;

omp\_set\_num\_threads(countOfThreads);

**int** sum = **0**;

#pragma omp parallel reduction(+:sum)

{

#pragma omp **for** schedule(**static**,**1**)

**for** (**int** i = **0**; i <= nums; i++){

sum += i;

#pragma omp critical

{

std::cout << omp\_get\_thread\_num() << ": calculation of the iteration number " << i << std::endl;

}

}

#pragma omp critical

{

std::cout << omp\_get\_thread\_num() << " Sum = " << sum << std::endl;

}

}

std::cout << "Sum = " << sum;

}

**void** **SecondTaskStatic2**(){

std::cout<<"Задание 2.2**\n**";

**int** countOfThreads = **0**;

**int** nums = **0**;

std::cout<<"Введите количество потоков: ";

std::cin>>countOfThreads;

**if** (countOfThreads>omp\_get\_max\_threads()){

countOfThreads = omp\_get\_max\_threads();

std::cout<<"Максимальное допустимое число потоков: "<<omp\_get\_max\_threads()<<"**\n**Кол-во потов выставлено на максимальное допустимое значение**\n**";

}

std::cout<<"Введите количество чисел: ";

std::cin>>nums;

omp\_set\_num\_threads(countOfThreads);

**int** sum = **0**;

#pragma omp parallel reduction(+:sum)

{

#pragma omp **for** schedule(**static**,**2**)

**for** (**int** i = **0**; i <= nums; i++){

sum += i;

#pragma omp critical

{

std::cout << omp\_get\_thread\_num() << ": calculation of the iteration number " << i << std::endl;

}

}

#pragma omp critical

{

std::cout << omp\_get\_thread\_num() << " Sum = " << sum << std::endl;

}

}

std::cout << "Sum = " << sum;

}

**void** **SecondTaskDynamic1**(){

std::cout<<"Задание 2.3**\n**";

**int** countOfThreads = **0**;

**int** nums = **0**;

std::cout<<"Введите количество потоков: ";

std::cin>>countOfThreads;

**if** (countOfThreads>omp\_get\_max\_threads()){

countOfThreads = omp\_get\_max\_threads();

std::cout<<"Максимальное допустимое число потоков: "<<omp\_get\_max\_threads()<<"**\n**Кол-во потов выставлено на максимальное допустимое значение**\n**";

}

std::cout<<"Введите количество чисел: ";

std::cin>>nums;

omp\_set\_num\_threads(countOfThreads);

**int** sum = **0**;

#pragma omp parallel reduction(+:sum)

{

#pragma omp **for** schedule(dynamic,**1**)

**for** (**int** i = **0**; i <= nums; i++){

sum += i;

#pragma omp critical

{

std::cout << omp\_get\_thread\_num() << ": calculation of the iteration number " << i << std::endl;

}

}

#pragma omp critical

{

std::cout << omp\_get\_thread\_num() << " Sum = " << sum << std::endl;

}

}

std::cout << "Sum = " << sum;

}

**void** **SecondTaskDynamic2**(){

std::cout<<"Задание 2.4**\n**";

**int** countOfThreads = **0**;

**int** nums = **0**;

std::cout<<"Введите количество потоков: ";

std::cin>>countOfThreads;

**if** (countOfThreads>omp\_get\_max\_threads()){

countOfThreads = omp\_get\_max\_threads();

std::cout<<"Максимальное допустимое число потоков: "<<omp\_get\_max\_threads()<<"**\n**Кол-во потов выставлено на максимальное допустимое значение**\n**";

}

std::cout<<"Введите количество чисел: ";

std::cin>>nums;

omp\_set\_num\_threads(countOfThreads);

**int** sum = **0**;

#pragma omp parallel reduction(+:sum)

{

#pragma omp **for** schedule(**static**)

**for** (**int** i = **0**; i <= nums; i++){

sum += i;

#pragma omp critical

{

std::cout << omp\_get\_thread\_num() << ": calculation of the iteration number " << i << std::endl;

}

}

#pragma omp critical

{

std::cout << omp\_get\_thread\_num() << " Sum = " << sum << std::endl;

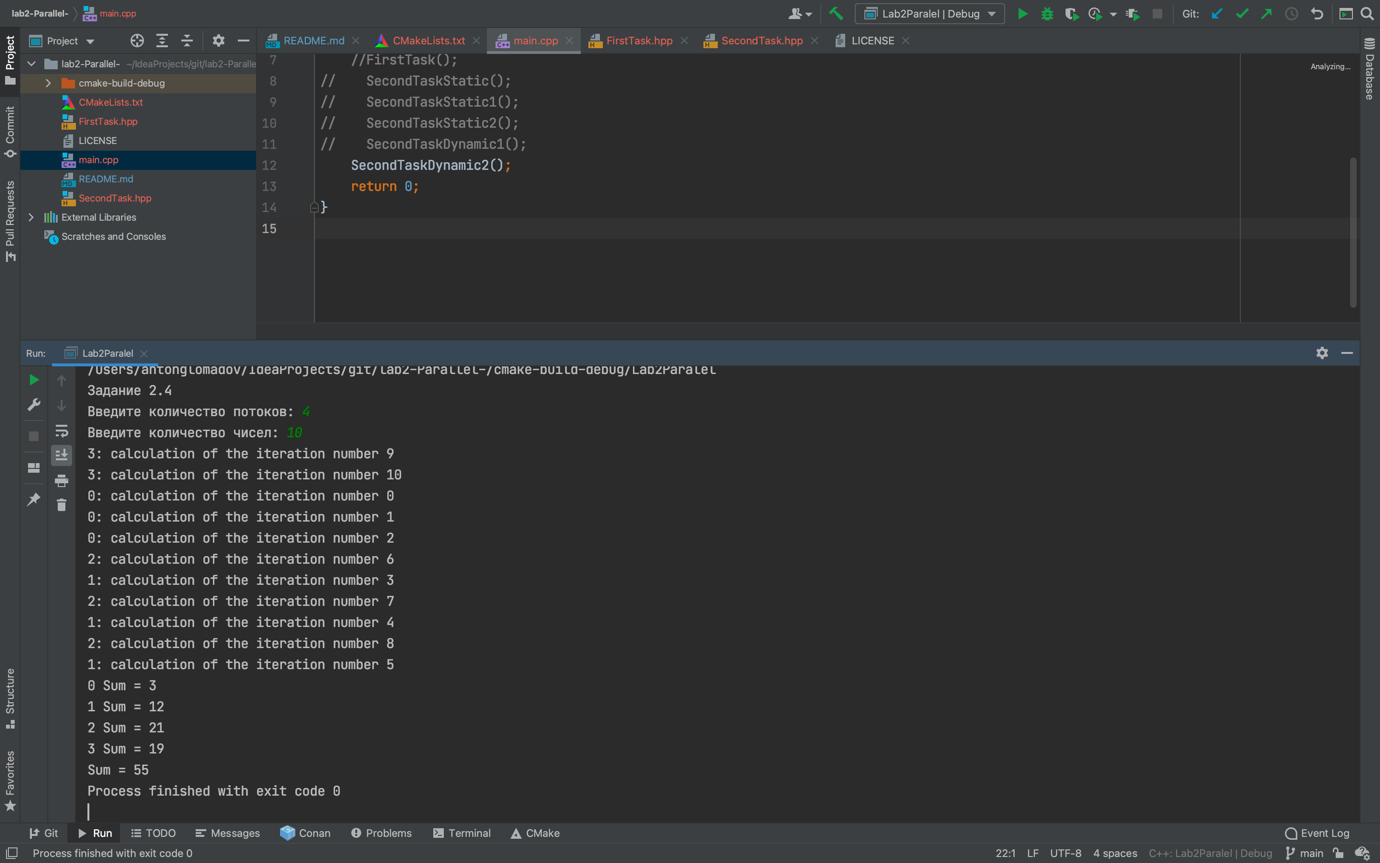
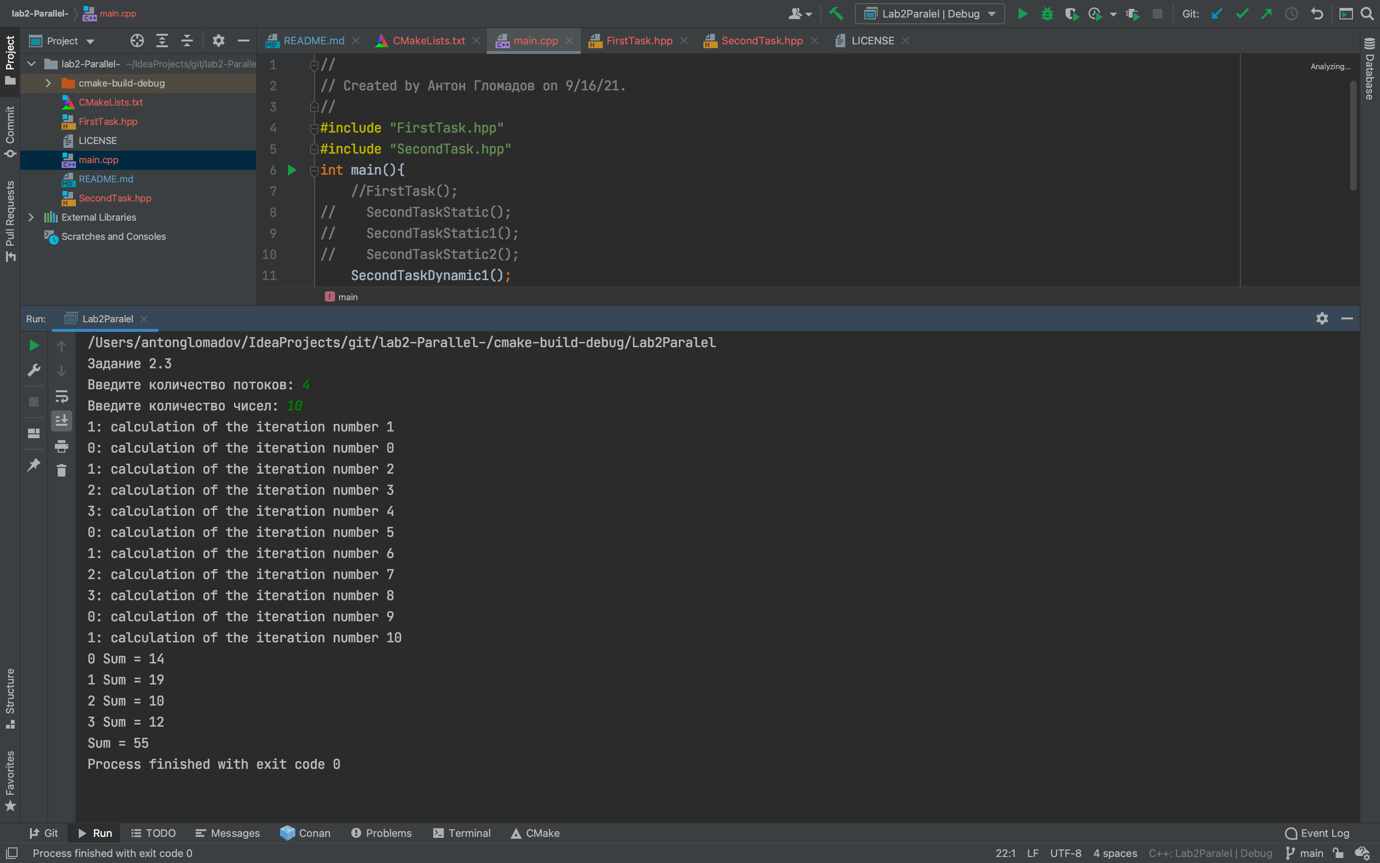
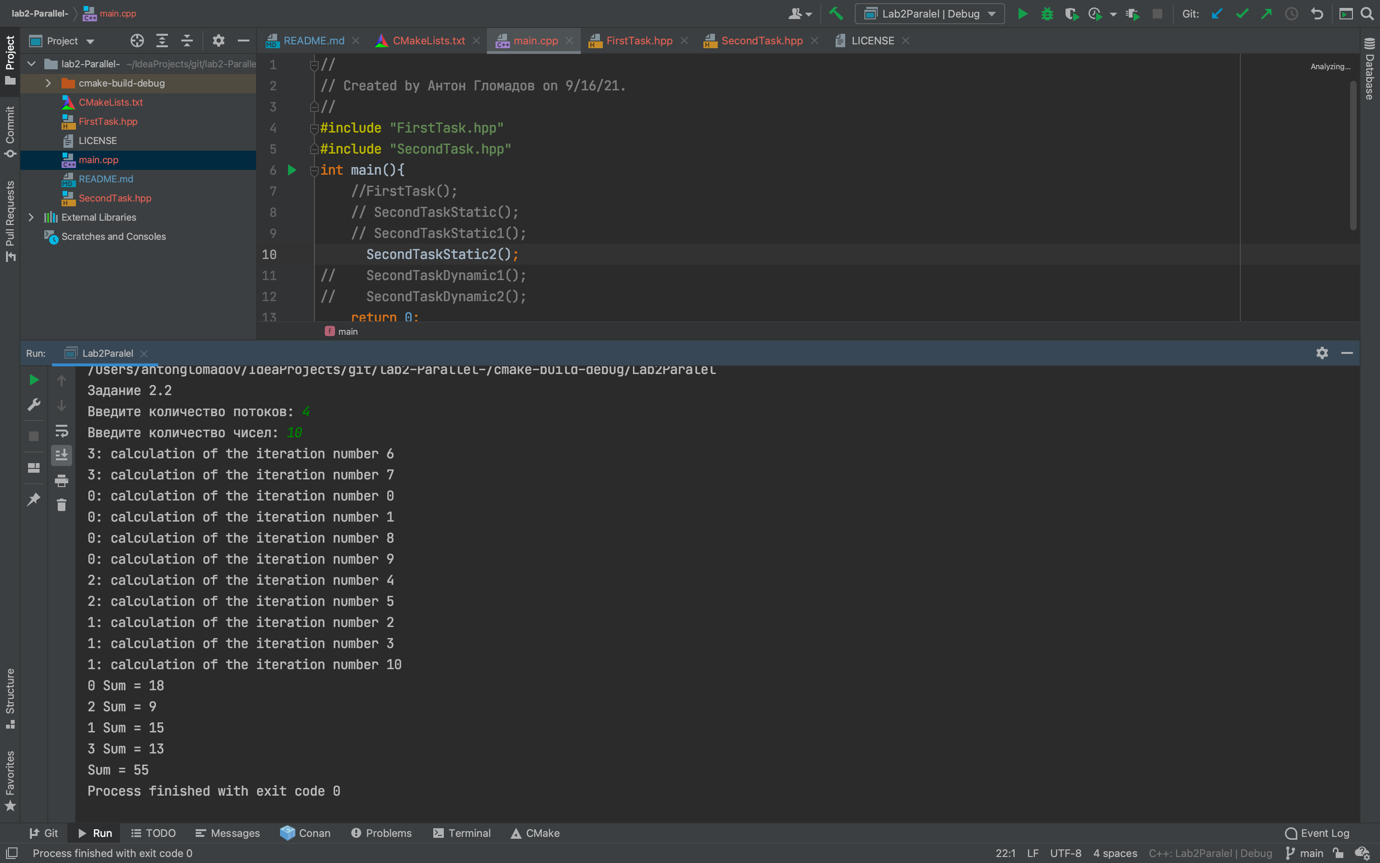
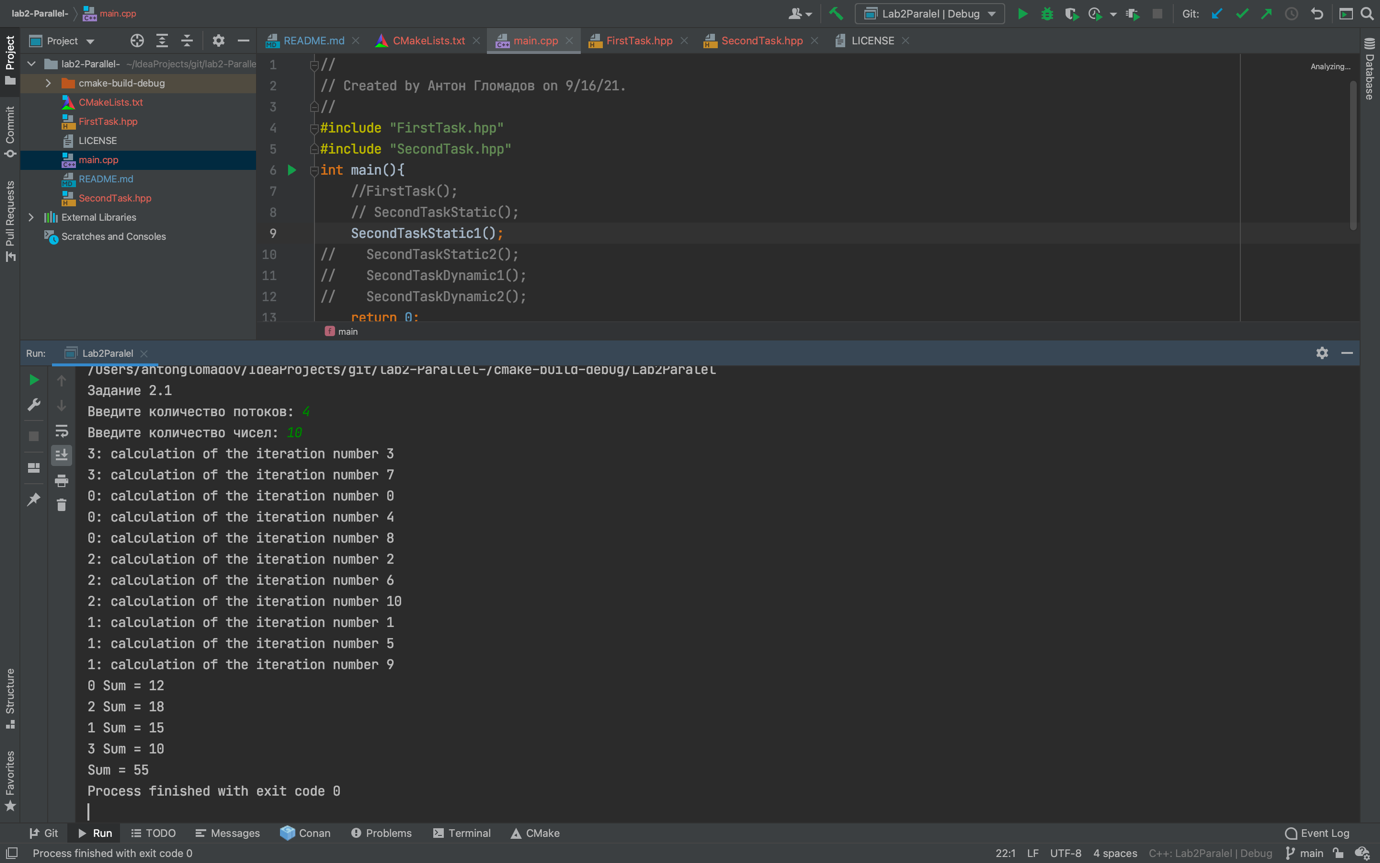
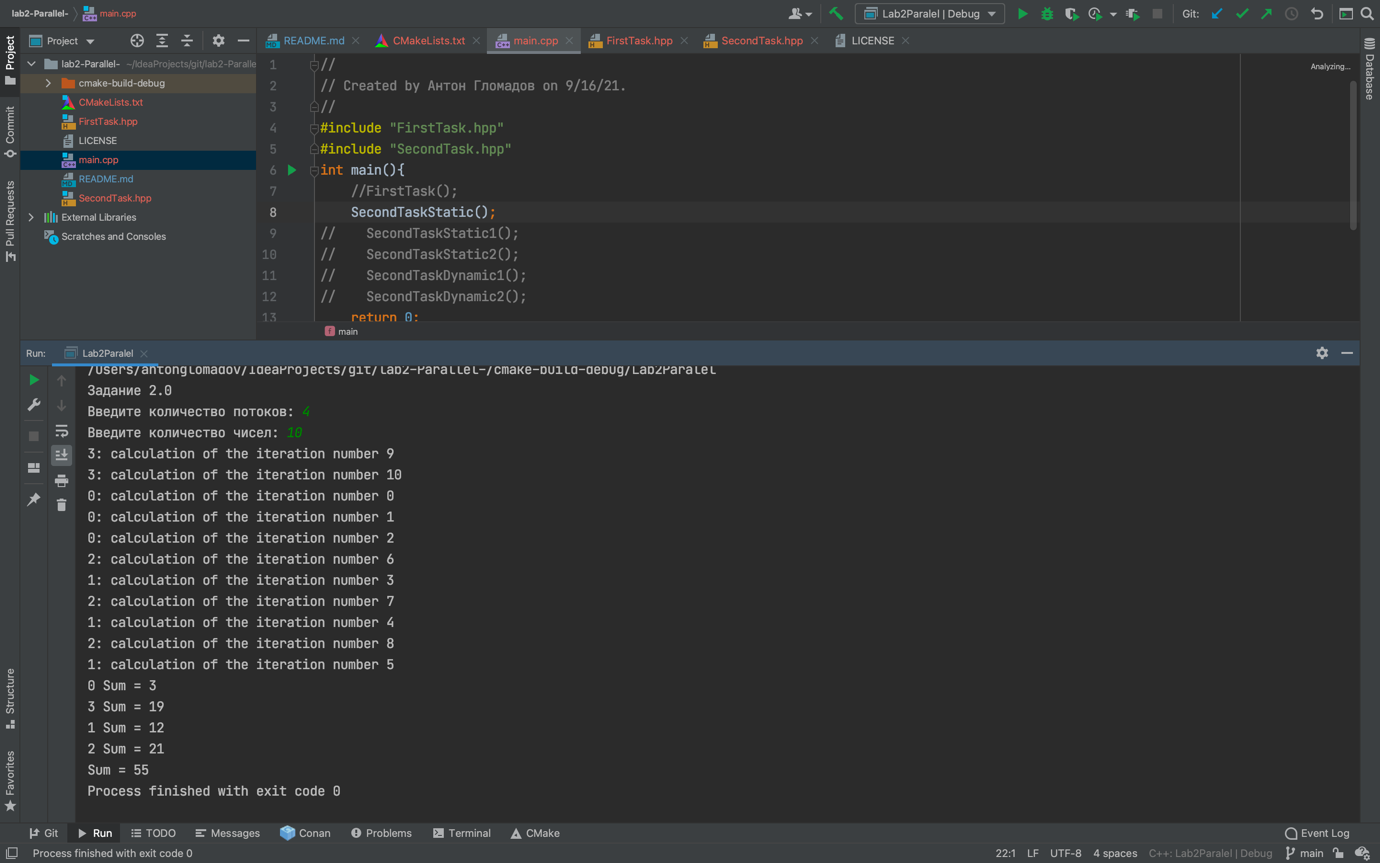
}

}

std::cout << "Sum = " << sum;

}

**Вывод для 2 задачи:**

****

**Листинг для 3 задачи:**

#pragma once

#include <omp.h>

#include <iostream>

#include <vector>

/\*\*

\*Напишите OpenMP-программу, которая вычисляет произведение двух квадратных матриц размера N x N.

\*/

**void** **ThirdTask**() {

srand(time(**0**));

**int** matrixSize;

**double** startParallel;

**double** endParallel;

**double** startNormal;

**double** endNormal;

std::cout << "Введите размер матрицы: ";

std::cin >> matrixSize;

**static** **int** a[**10000**][**10000**];

**static** **int** b[**10000**][**10000**];

**static** **int** c[**10000**][**10000**];

**for** (**int** i = **0**; i < matrixSize; ++i) {

**for** (**int** j = **0**; j < matrixSize; ++j) {

a[i][j] = rand() % **100000**;;

b[i][j] = rand() % **100000**;;

c[i][j] = **0**;

}

}

startParallel = omp\_get\_wtime();

omp\_set\_num\_threads(omp\_get\_max\_threads());

#pragma omp parallel for schedule(static)

{

**for** (**int** i = **0**; i < matrixSize; ++i)

**for** (**int** m = **0**; m < matrixSize; ++m)

**for** (**int** j = **0**; j < matrixSize; ++j)

c[i][m] += a[i][j] \* b[j][m];

}

endParallel = omp\_get\_wtime();

std::cout << "Паралельное время : " << endParallel - startParallel << std::endl;

startNormal = omp\_get\_wtime();

**for** (**int** i = **0**; i < matrixSize; ++i) {

**for** (**int** m = **0**; m < matrixSize; ++m)

**for** (**int** j = **0**; j < matrixSize; ++j)

c[i][m] += a[i][j] \* b[j][m];

}

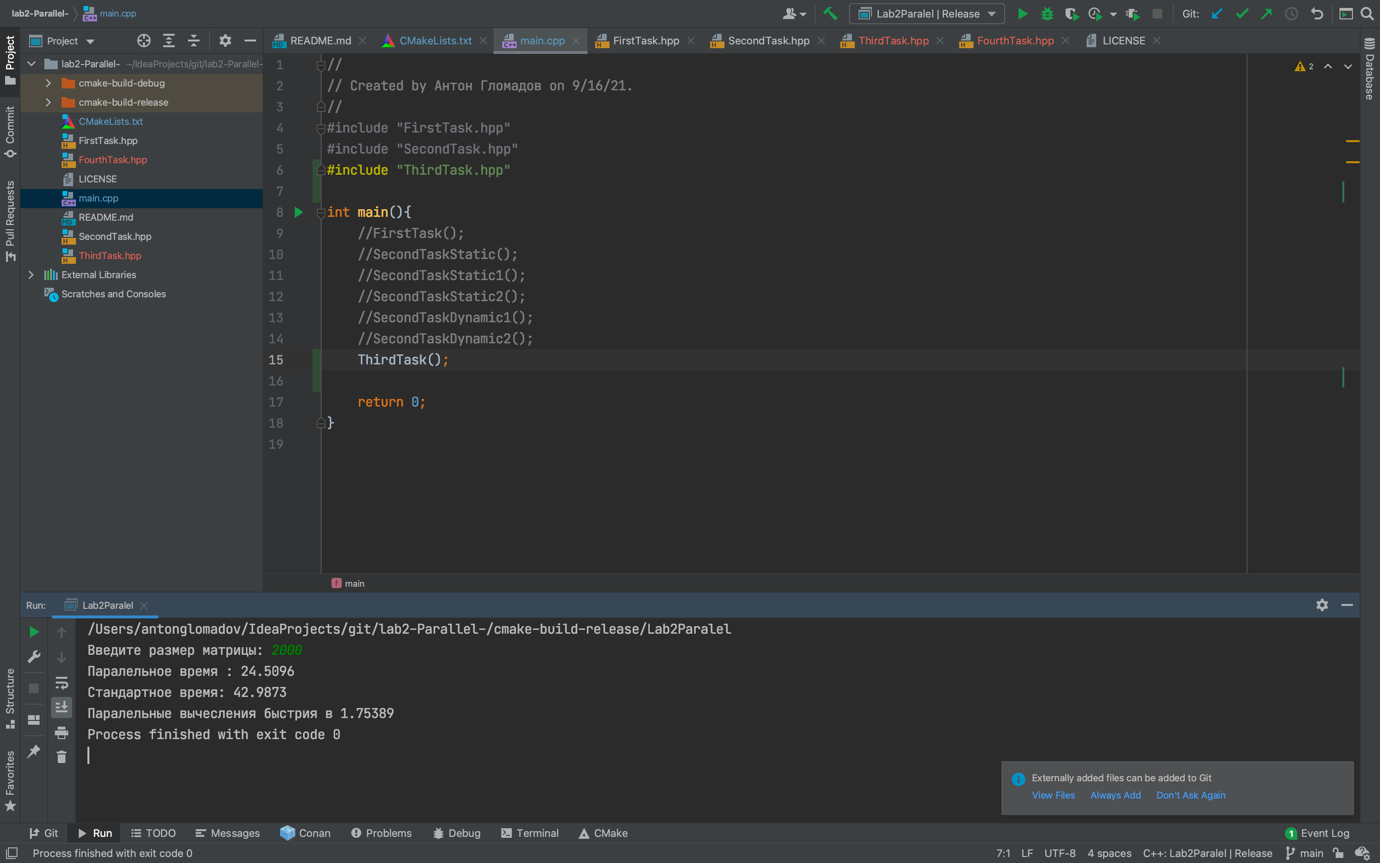
endNormal = omp\_get\_wtime();

std::cout << "Стандартное время: " << endNormal - startNormal << std::endl;

std::cout << "Паралельные вычесления быстрия в " << (endNormal - startNormal)/(endParallel - startParallel);

};

**Вывод для 3 задачи:**

****

**Листинг для 4 задачи:**

//

// Created by Антон Гломадов on 9/16/21.

//

#pragma once

#include <omp.h>

#include <iostream>

**void** **FourthTask**() {

std::cout<<"Задание 4**\n**";

**int** countOfThreads = **0**;

**int** nums = **0**;

std::cout<<"Введите количество потоков: ";

std::cin>>countOfThreads;

**if** (countOfThreads>omp\_get\_max\_threads()){

countOfThreads = omp\_get\_max\_threads();

std::cout<<"Максимальное допустимое число потоков: "<<omp\_get\_max\_threads()<<"**\n**Кол-во потов выставлено на максимальное допустимое значение**\n**";

}

std::cout<<"Введите количество чисел: ";

std::cin>>nums;

omp\_set\_num\_threads(countOfThreads);

**int** sum = **0**;

#pragma omp parallel **for**

**for** (**int** i = **1**; i <= nums; ++i) {

#pragma omp atomic

sum += i;

#pragma omp critical

{

std::cout << omp\_get\_thread\_num() << " Sum = " << sum << std::endl;

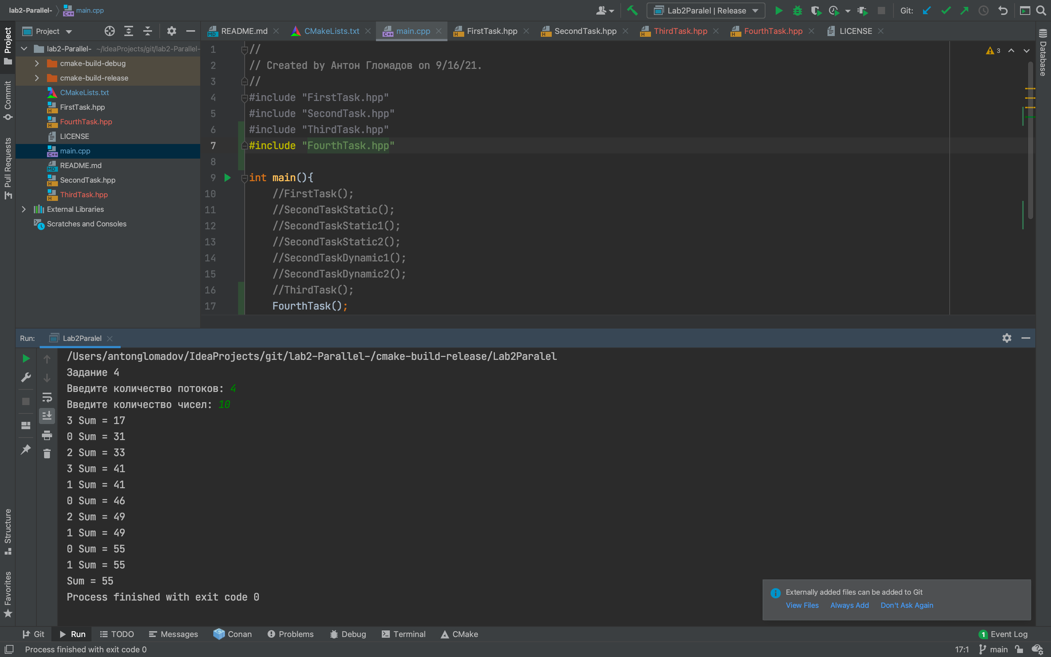
}

}

std::cout << "Sum = " << sum;

}

**Вывод для 3 задачи:**



Вывод: были получены навыки работы с циклом for в OpenMP.