Лабораторная работа №3.  
«Основы OpenMP»

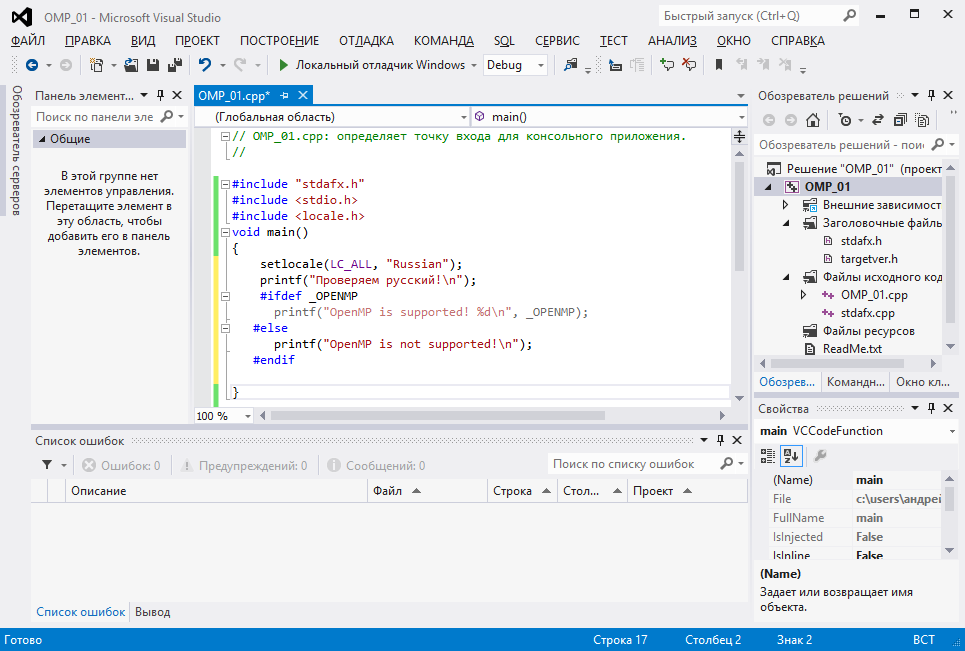
**Цель работы**

Знакомство с базовыми элементами технологии распараллеливания OpenMP.

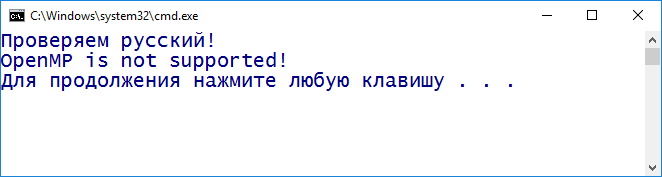
**Порядок выполнения работы.**

**Шаг 1**. Подготовим компьютер для работы с OpenMP. Будем использовать язык программирования С++ в среде MS Visual Studio.

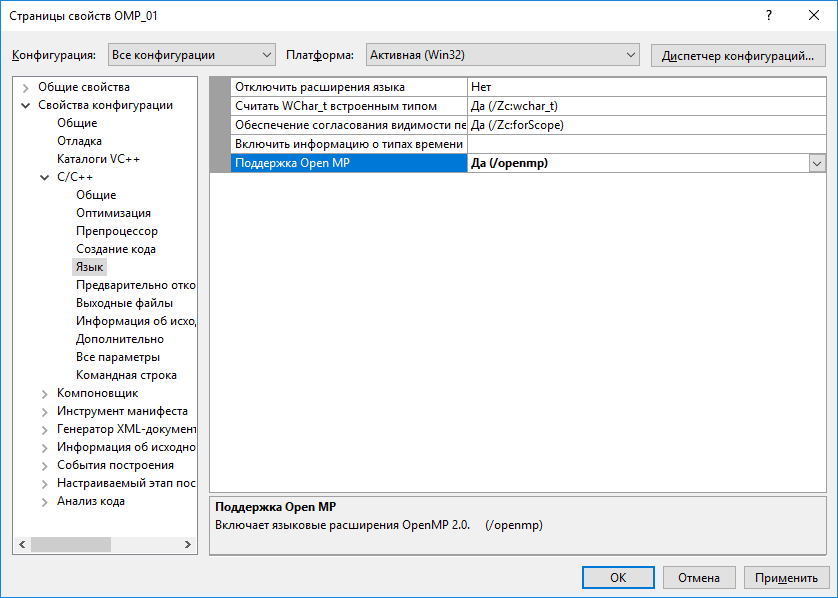
Создадим новый проект, в котором проверим, настроен ли компилятор на работу с OpenMP, а также вывод на экран кириллицы.



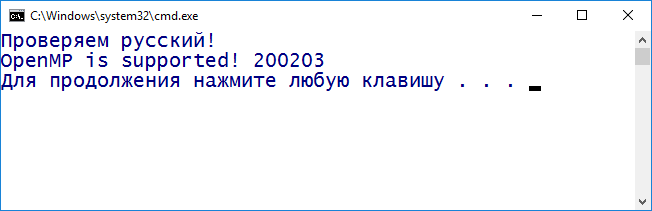
Запуск проекта скорее всего даст следующий вывод:



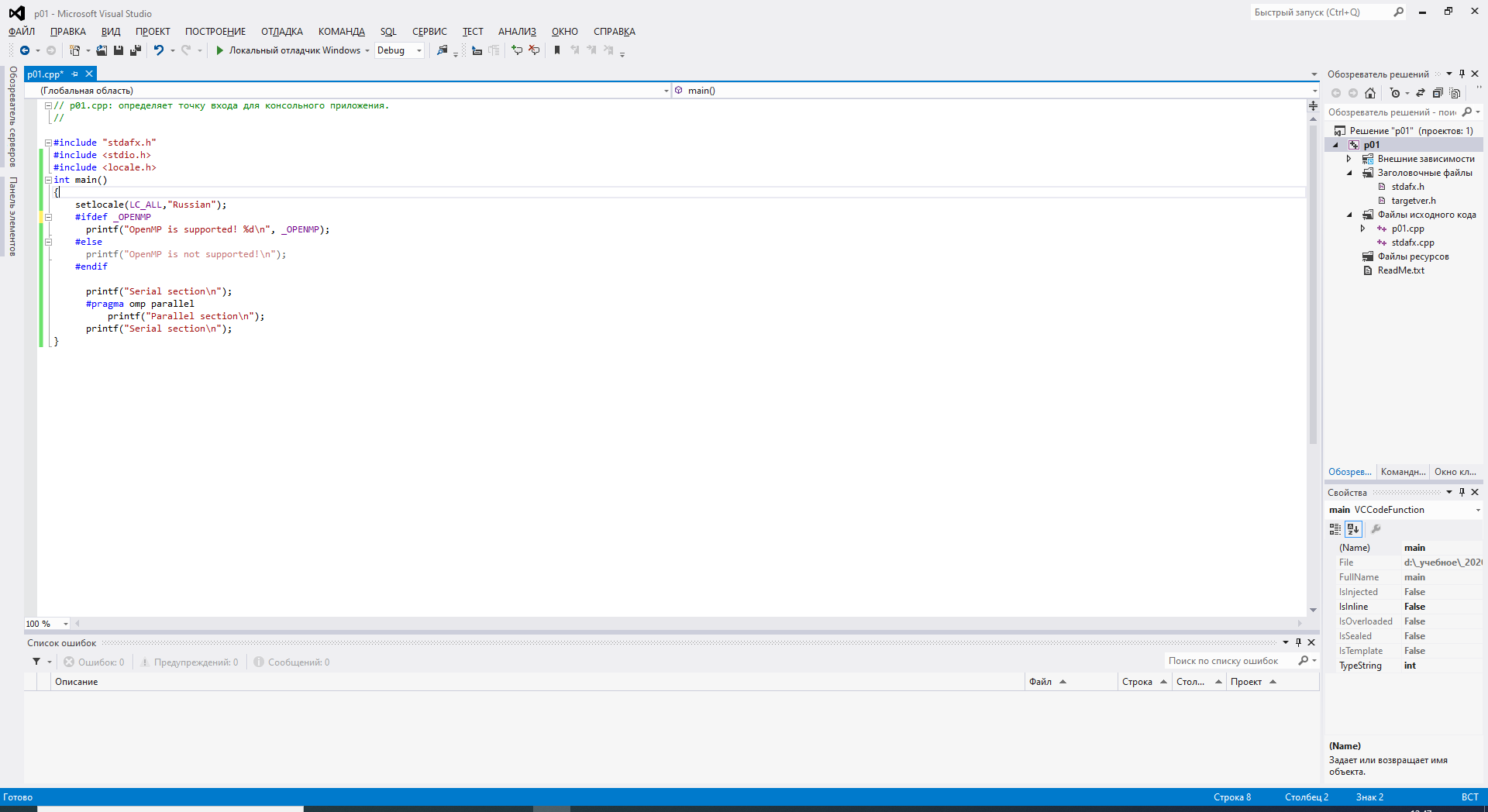
Для включения поддержки OpenMP в данном проекте выполним необходимую установку данного режима. Схематично это можно записать так «Project → Properties → All Configurations → Configuration Properties → C/C++ → Language → OpenMP Support → Yes». В окне свойств проекта это выглядит примерно так:



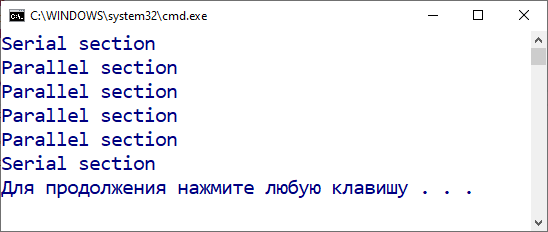
После настроек OpenMP работает:



**Шаг 2**. Проверим работу последовательных и параллельных участков программы (**не забываем директиву #include <omp.h>**).

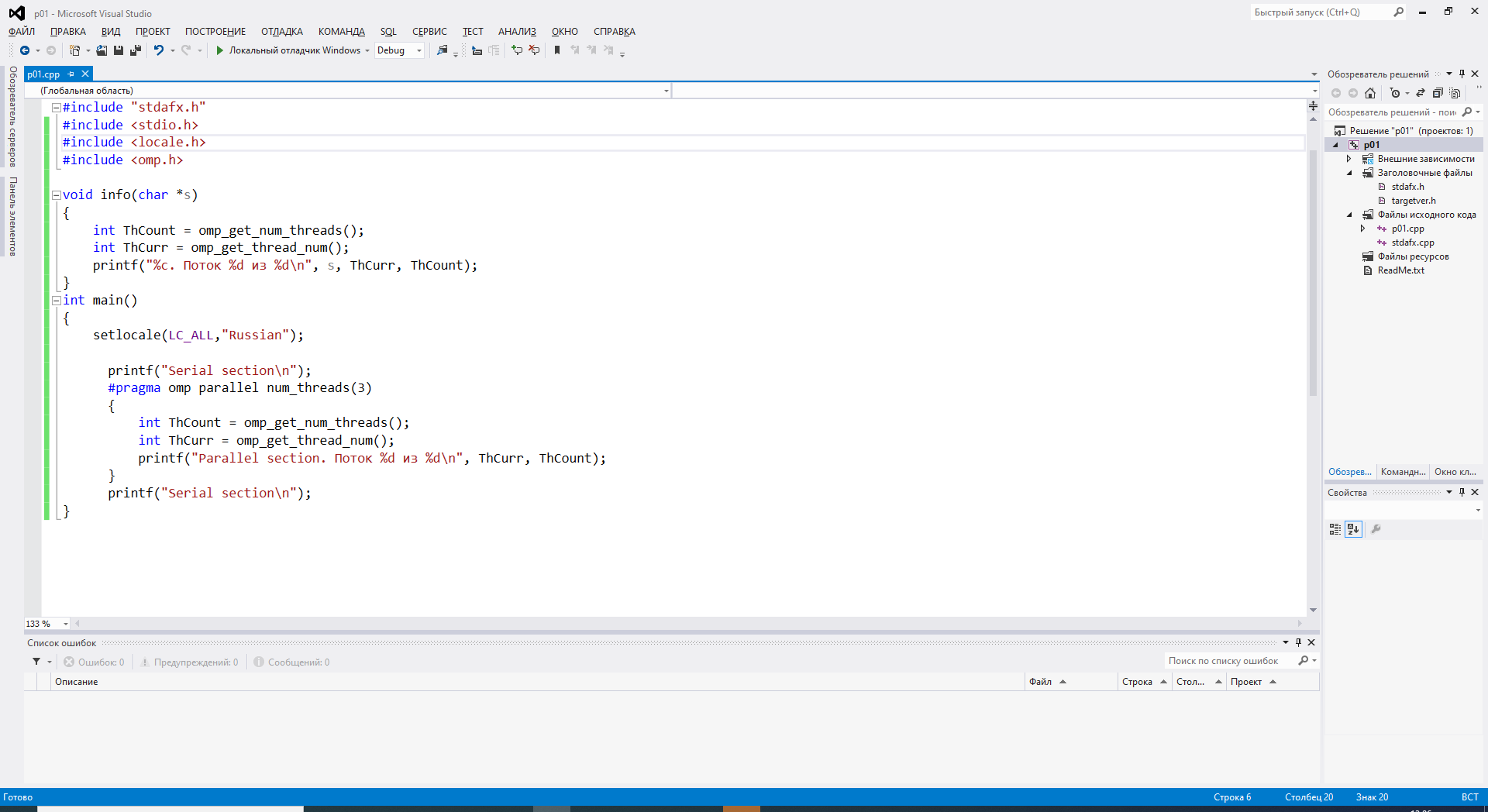


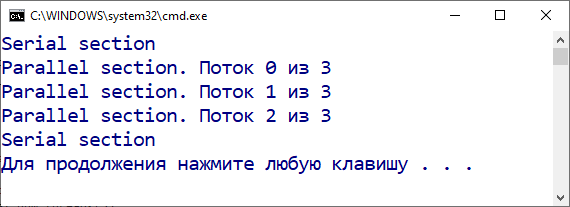
В случае запуска на моём компьютере результат запуска был такой



Т.е., было организовано 4 параллельных потока по числу процессоров.

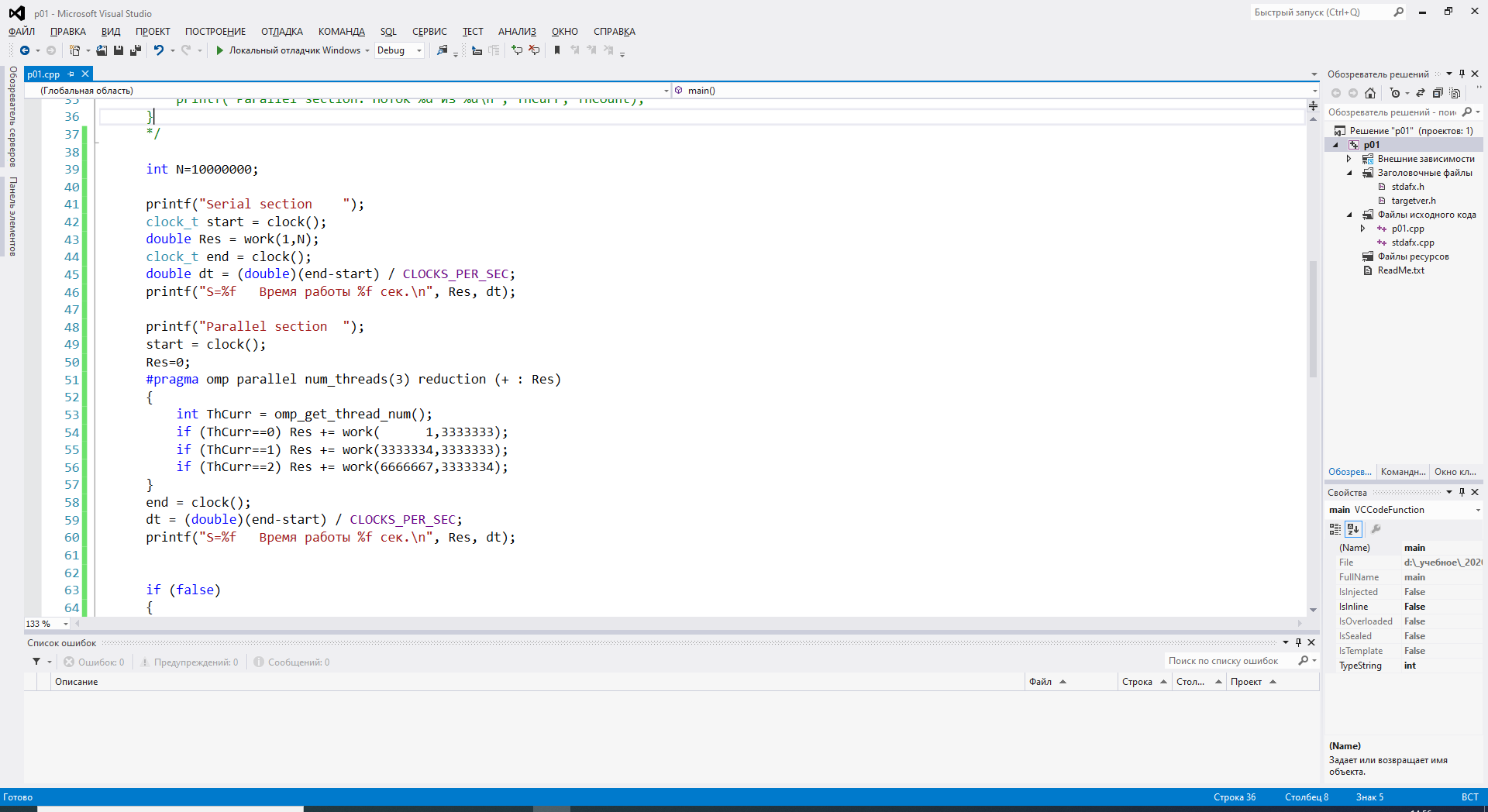
Изменим число потоков до трёх и выведем их номера на консоль.



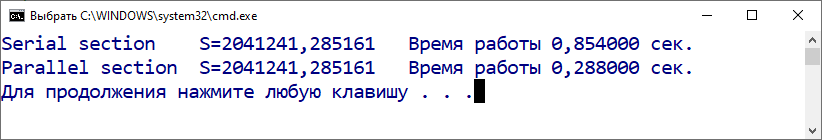


**Шаг 3**. Проведём небольшой эксперимент. Попробуем распараллелить расчёт суммы

Для примера возьмём 3 потока и, разделив N слагаемых данной суммы на примерно равные группы, доверим каждую группу отдельному потоку. Для проверки корректности вначале найдём эту сумму в последовательном режиме. Попутно оценим время работы в последовательной и параллельной реализациях. Чтобы почувствовать разницу выберем довольно большое N=107.



Результат работы показывает, что, организовав три потока, мы сократили время расчёта суммы почти в 3 раза: **K=2.97** (**коэффициент ускорения** К равен отношению времени последовательного расчёта ко времени параллельного расчёта).



**Задание для самостоятельной работы в классе.**

Исследуйте эффективность распараллеливания расчёта суммы из 3-го шага лабораторной работы организовав программу так, чтобы она показала время расчёта и коэффициент ускорения для числа потоков от 1 до М+2 (М-число процессоров Вашего компьютера – определяется программой!). При этом в качестве времени работы брать среднее время по 10 испытаний для каждого числа потоков. Результат программы – таблица на консоли со следующими графами: число потоков, значение суммы в последнем эксперименте, среднее время расчёта, коэффициент ускорения).

**Задания для самостоятельной работы дома.**

**Задание 1. Общие и частные переменные в OpenMP: параметр reduction.**

А) Напишите программу, в которой две нити параллельно вычисляют сумму чисел от 1 до N. Распределите работу по нитям с помощью оператора if языка С. Для сложения результатов вычисления нитей воспользуйтесь OpenMP-параметром reduction. Выполните расчёты для N ∈ {1,2,5,10,100,1000}

Входные данные: 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>».

**[0]: Sum = 3**

**[1]: Sum = 7**

**Sum = 10**

**Задание 2. Общие и частные переменные в OpenMP: параметр reduction.**

Б) Модифицируйте программу таким образом, чтобы она работала для k нитей. Покажите работу программы для N ∈ {2,10,100,1000}, k ∈ {2,4,8,16}.

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

**[0]: Sum = 1**

**[1]: Sum = 2**

**[2]: Sum = 0**

**Sum = 3**

**Задание 3. Распараллеливание циклов в OpenMP: программа «Сумма чисел»**

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. Покажите работу программы для N ∈ {2,10,100,1000}, k ∈ {2,4,8,16}.

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>».

**Задание 4. Распараллеливание циклов в OpenMP: параметр schedule**

Изучите параметр schedule директивы for. Модифицируйте программу «Сумма чисел» из задания 3 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. Заполните следующую таблицу распределения итераций цикла по нитям в зависимости от параметра schedule:



Объясните полученные результаты.

**Задание 5. Распараллеливание циклов в OpenMP: параметр schedule**

Реализовать вычисление числа π с требуемой точностью ε по формуле

Для вычисления интеграла использовать метод левых прямоугольников.

Исследовать зависимость времени работы алгоритма от числа потоков {k=1,2,3,…,20} и желаемой точности (ε=0.001, 0.0001, 0.00001, …, 0.000000001, … - остановка, когда время последнего расчёта превысило 2 мин). Полученную зависимость показать на графике.

Для достижения необходимой точности на каждом шаге увеличивать количество разбиений в 2 раза, пока уточнение результата не станет изменяться меньше, чем на ε.

**Требования к отчету**

Отчет по лабораторной работе должен состоять из разделов:

1. информация о студенте – ФИО, группа;

2. цель работы;

3. блоки ответов по каждому из заданий, включающие следующие пункты:

* текст задания
* скриншоты результатов работы программы;
* код метода, реализующего конкретное задание (void Task1(int N) – для задания №1);
* таблица результатов (для заданий 4 и 5);
* график зависимости (задание 5);
* выводы по заданию.

Отчёт высылается в электронном виде на почту [redregion@mail.ru](mailto:redregion@mail.ru) в срок до даты, указанной в имени файла (включительно).