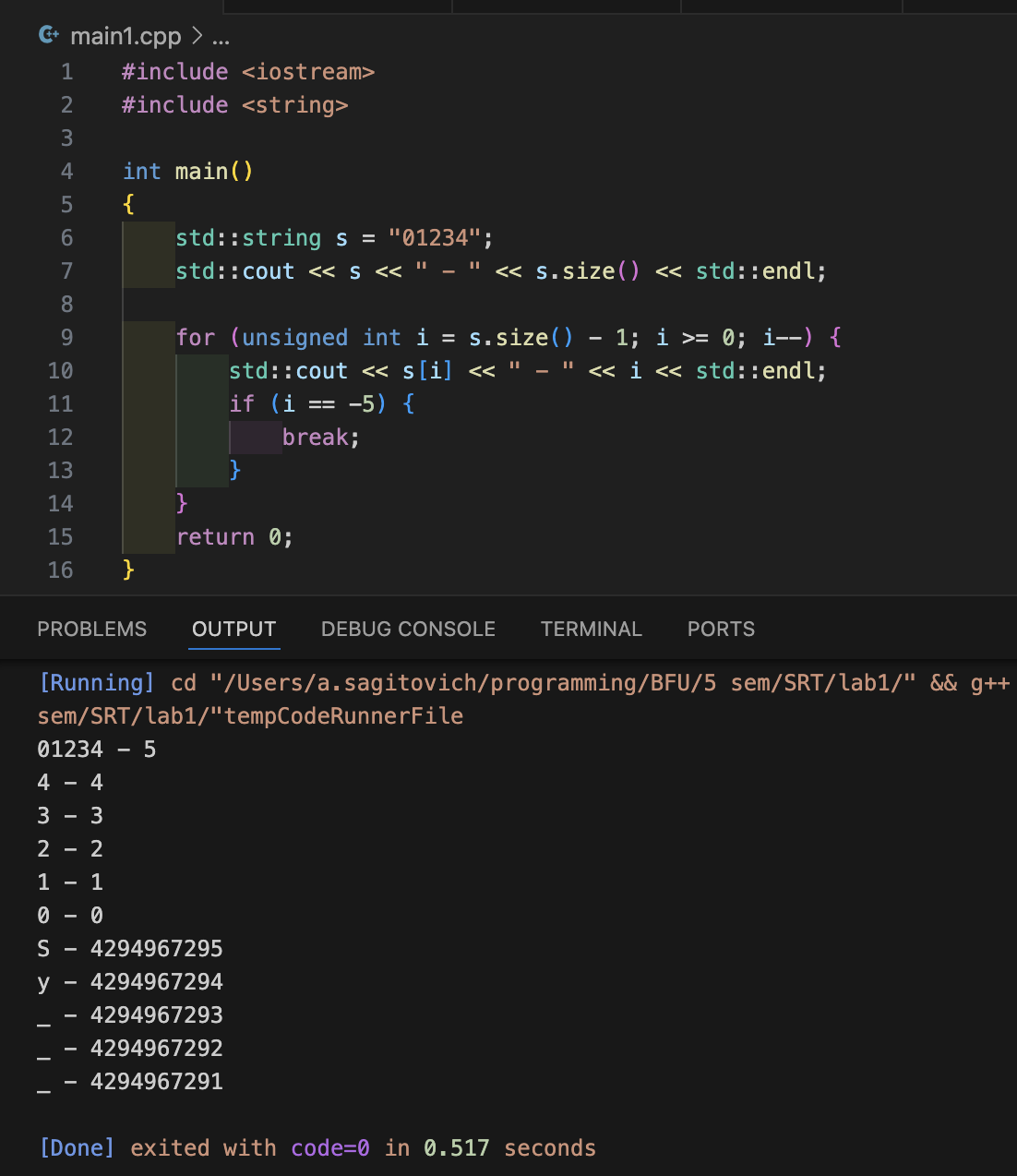
Системы реального времени  
Лабораторная работа №1  
Сагитов Александр

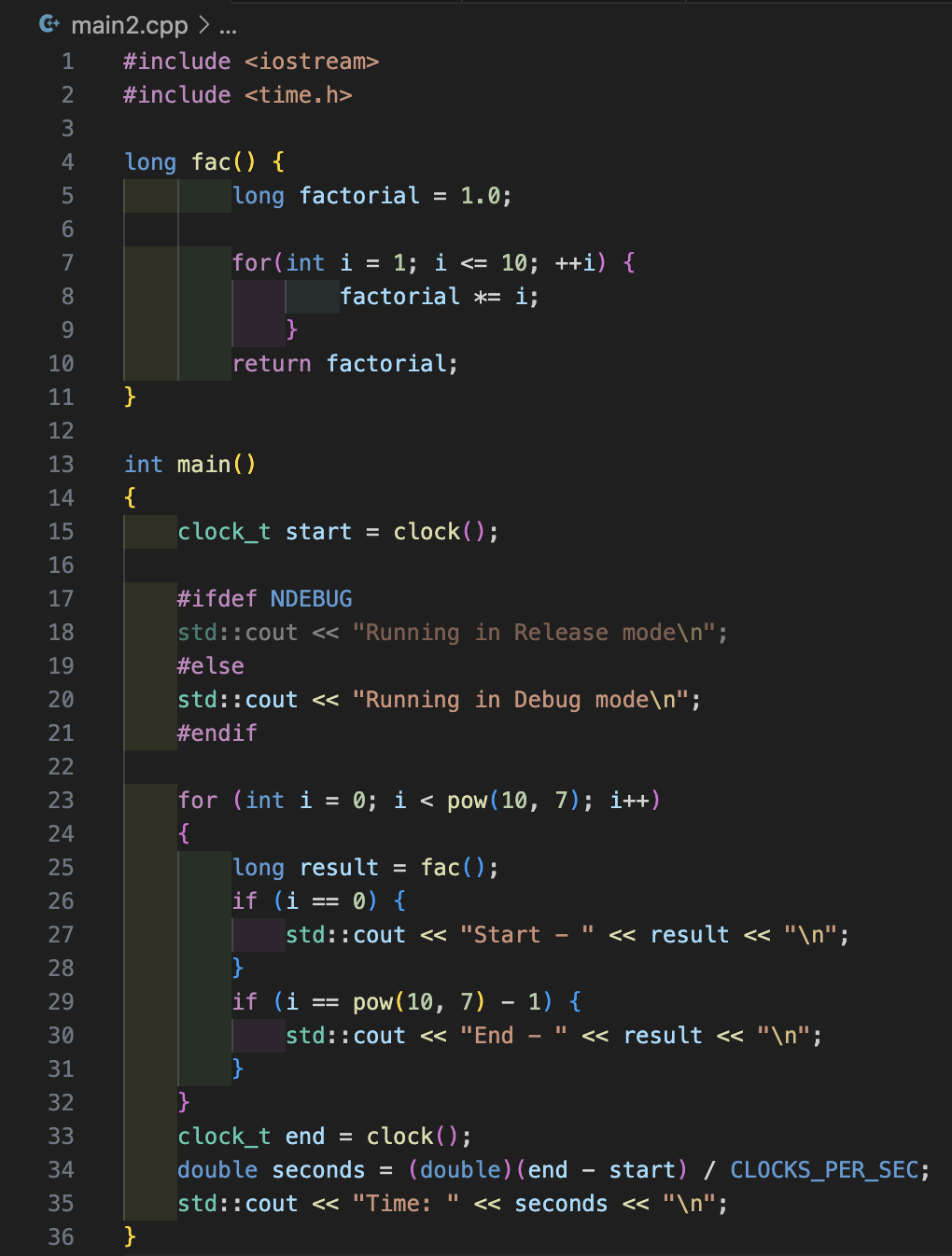
**Задание №1**Что выведет следующая программа и почему?

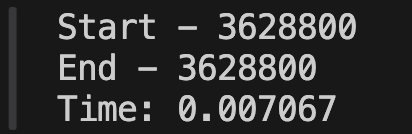
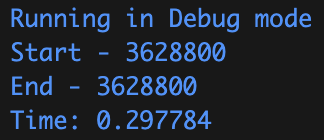


Когда переменная i становится равной -1 начинаются проблемы: i имеет тип ***unsigned int***, который не может содержать отрицательные значения, ввиду чего при уменьшении i до -1 происходит “переполнение”. В результате вместо -1, переменная i получает максимальное значение для типа unsigned int, которое равно = 4294967295.

Т. к. i стало слишком большим числом, программа пытается вывести символы s[i] строки s, где i уже выходит за границы строки. Это приводит к чтению произвольных данных из памяти, что объясняет странные символы в выводе.

**Задание №2**  
Вычислить факториал 10! - 10 млн. раз. Замерить время выполнения программы в режиме *debug* и *release* версии. Объяснить результаты.





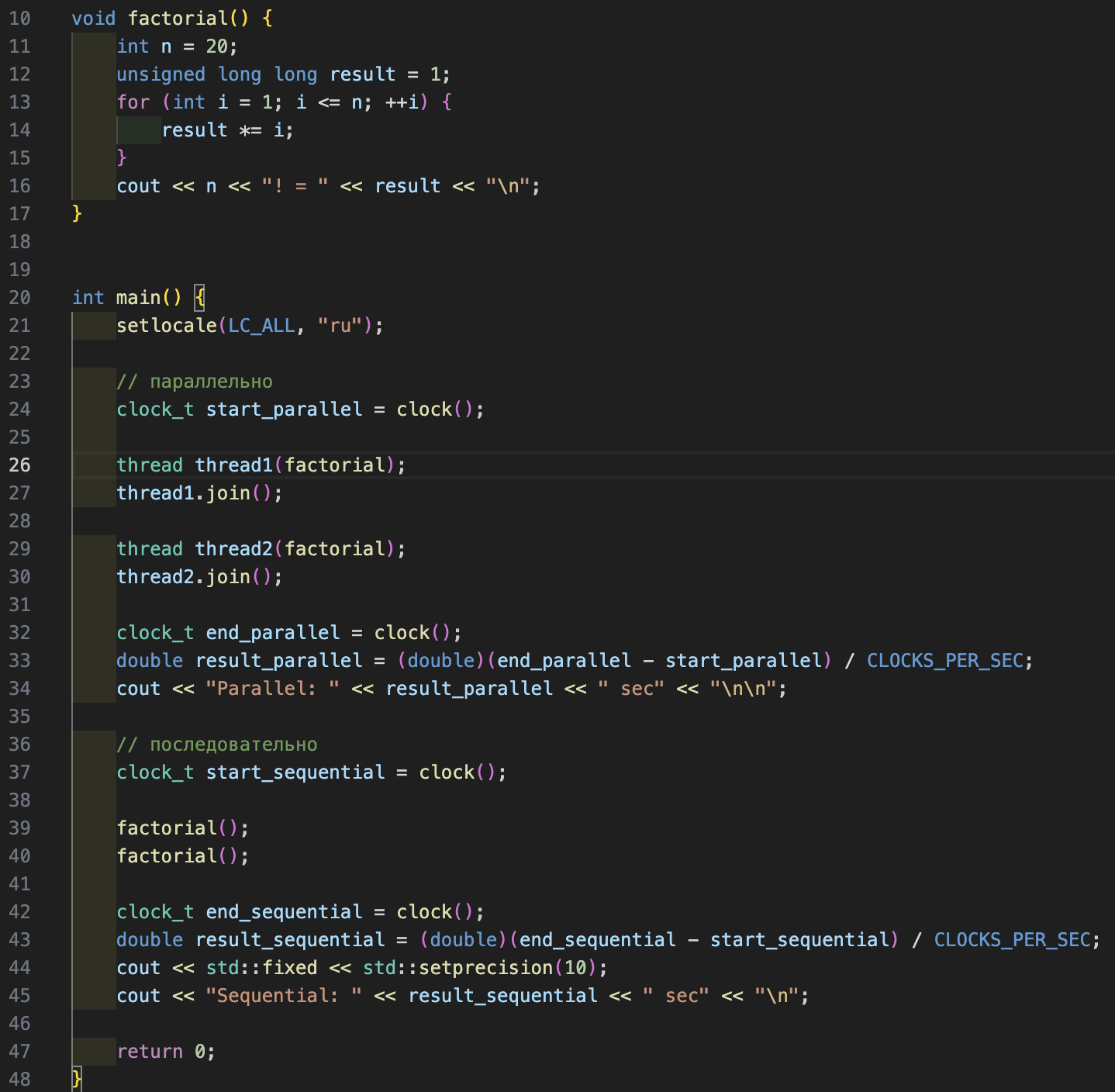
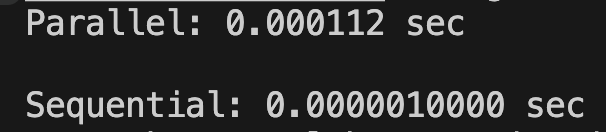
В ***release*** программа работает быстрее.  
Это происходит потому, что компилятор оптимизирует код. Он вставляет функции прямо в место их вызова, убирает ненужные проверки и делает циклы проще и быстрее. Также компилятор может использовать специальные команды процессора для ускорения.

В ***debug***все эти улучшения отключены, чтобы программа была легче для отладки, поэтому она работает медленнее.

**Задание №3**

Замерьте время выполнения всей программы.  
Запустите вызов функции два раза последовательно.

Замерьте время и сравните с запуском в параллельных потоках.

Последовательное выполнение функции оказалось быстрее, чем параллельное, потому что factorial() выполняется очень быстро (не требуется много ресурсов на подсчёт 20!).

В таких случаях создание и управление потоками добавляют дополнительные задержки, которые превышают пользу от параллельности. Когда функция запускается последовательно, программа просто выполняет один вызов за другим без лишних затрат на потоки, поэтому это оказывается быстрее для маленьких задач, как в данном примере.