Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Операционные системы**

Студент: Козека Е. М.

ФИТ 3 курс 4 группа

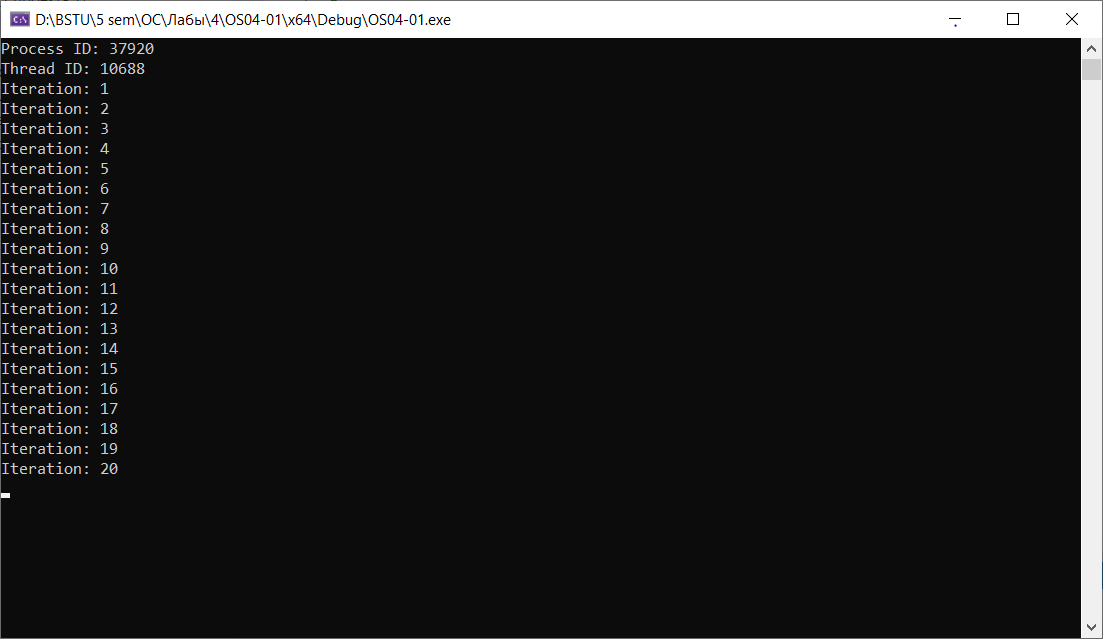
Преподаватель: Комкова А. В.

Минск 2024

**Лабораторная работа №4. Потоки**

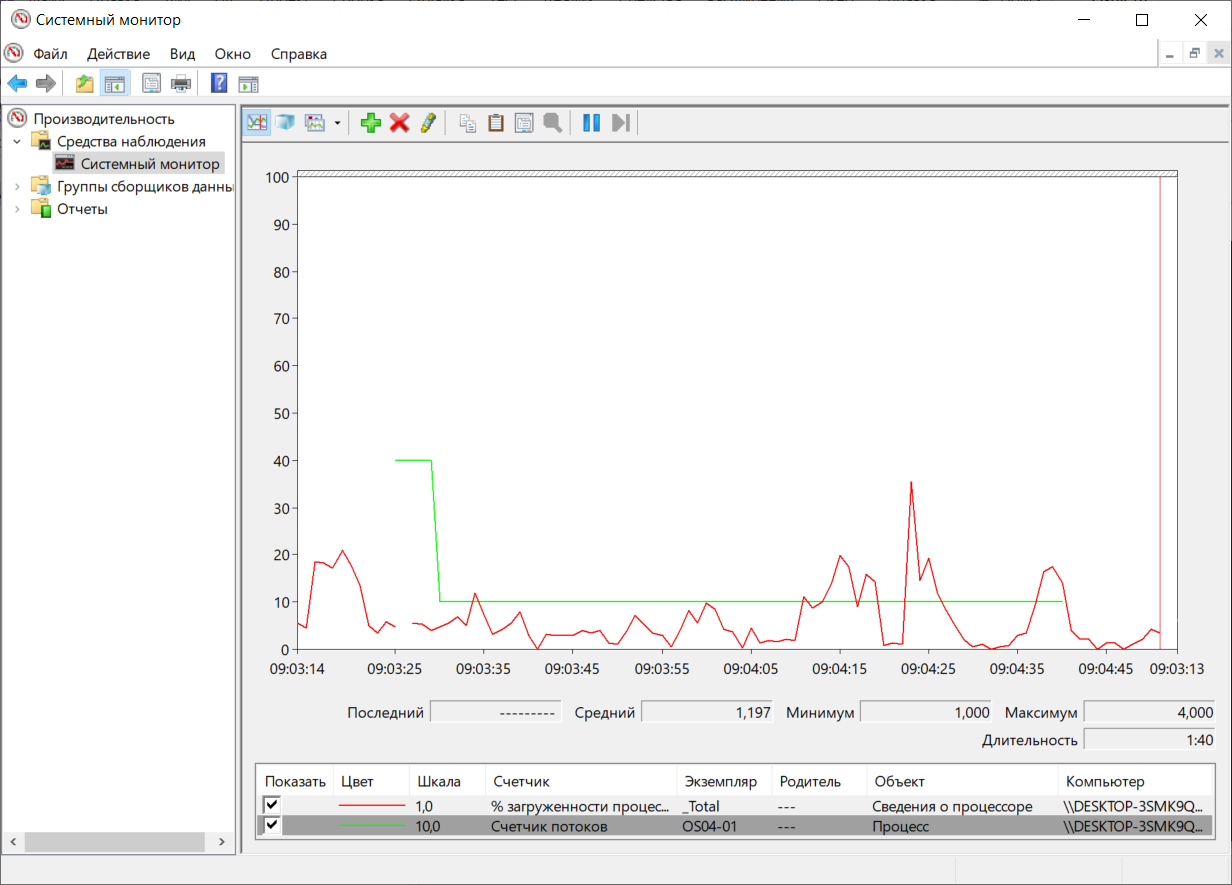
**Задание 1.** Разработайте консольное Windows-приложение OS04\_01 на языке С++, выполняющее длинный цикл с временной задержкой и с выводом на консоль идентификаторов текущего процесса и текущего потока.

Результат выполнения в консоли:

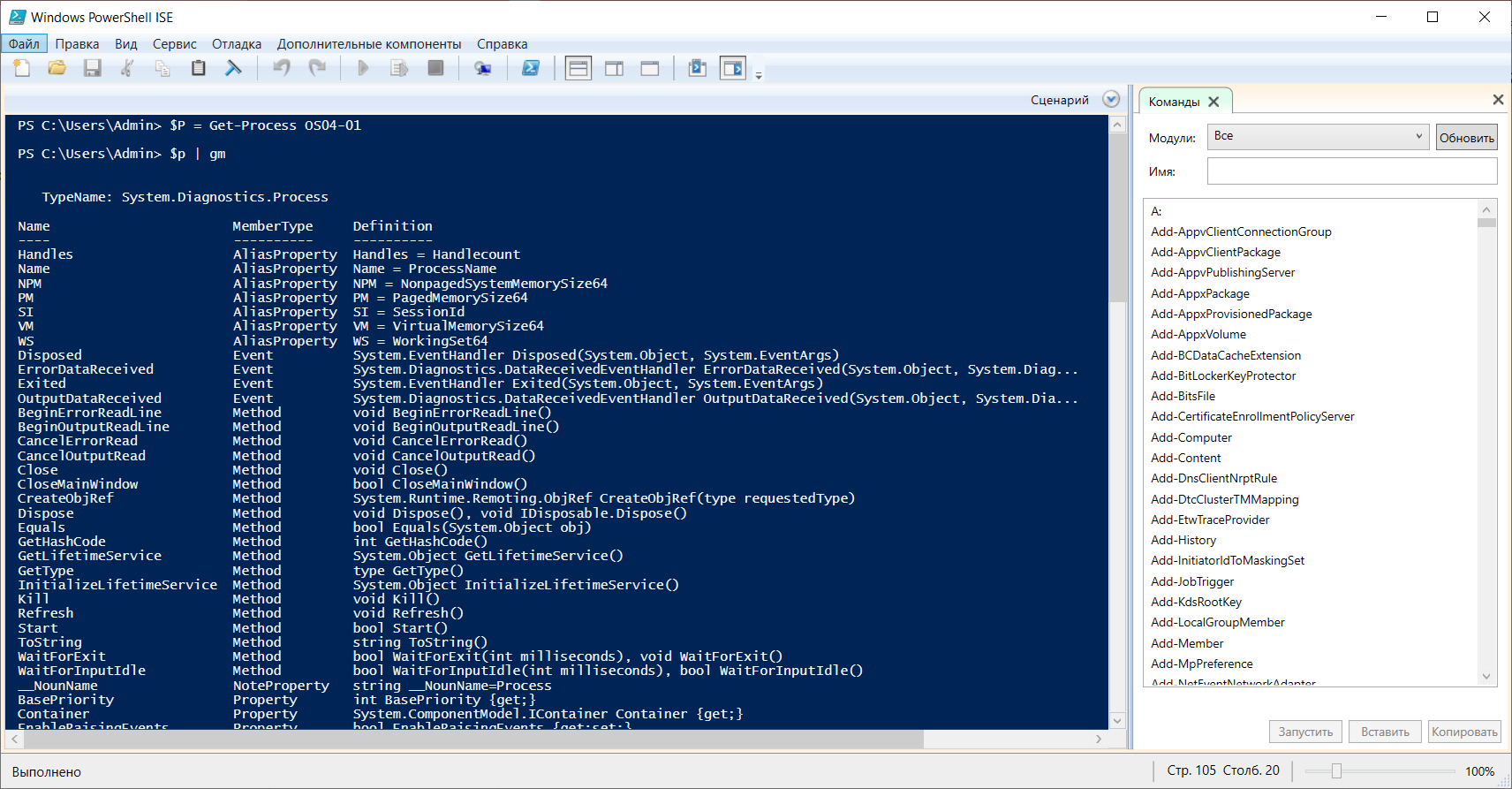


Продемонстрируйте информацию об потоках процесса OS04\_01 с помощью утилит PowerShell ISE и Performance Monitor.

Performance Monitor:

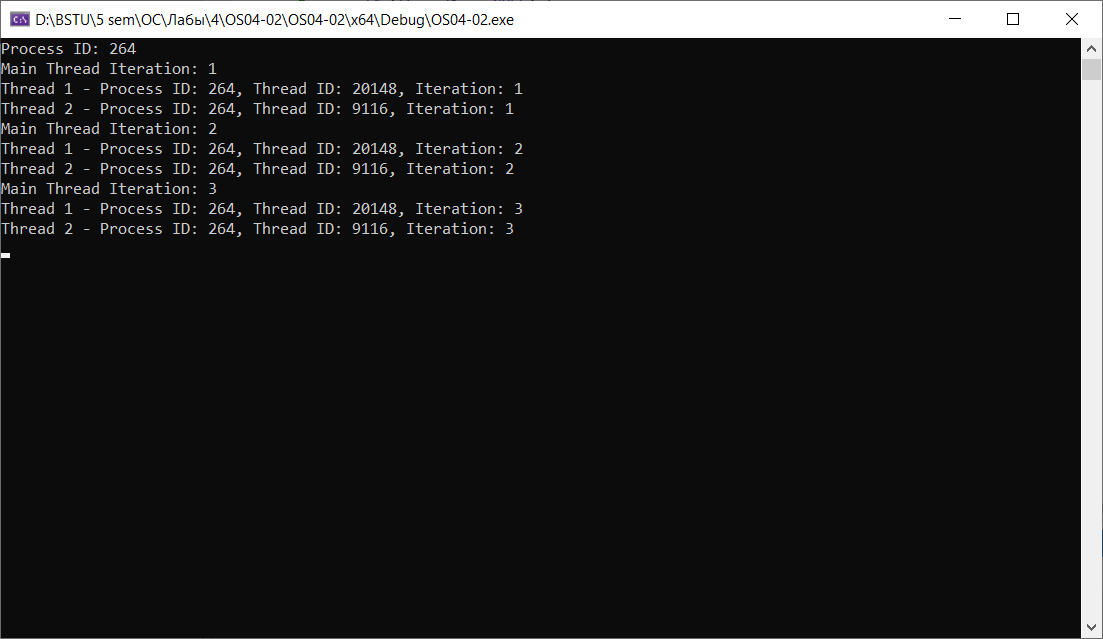


PowerShell ISE:



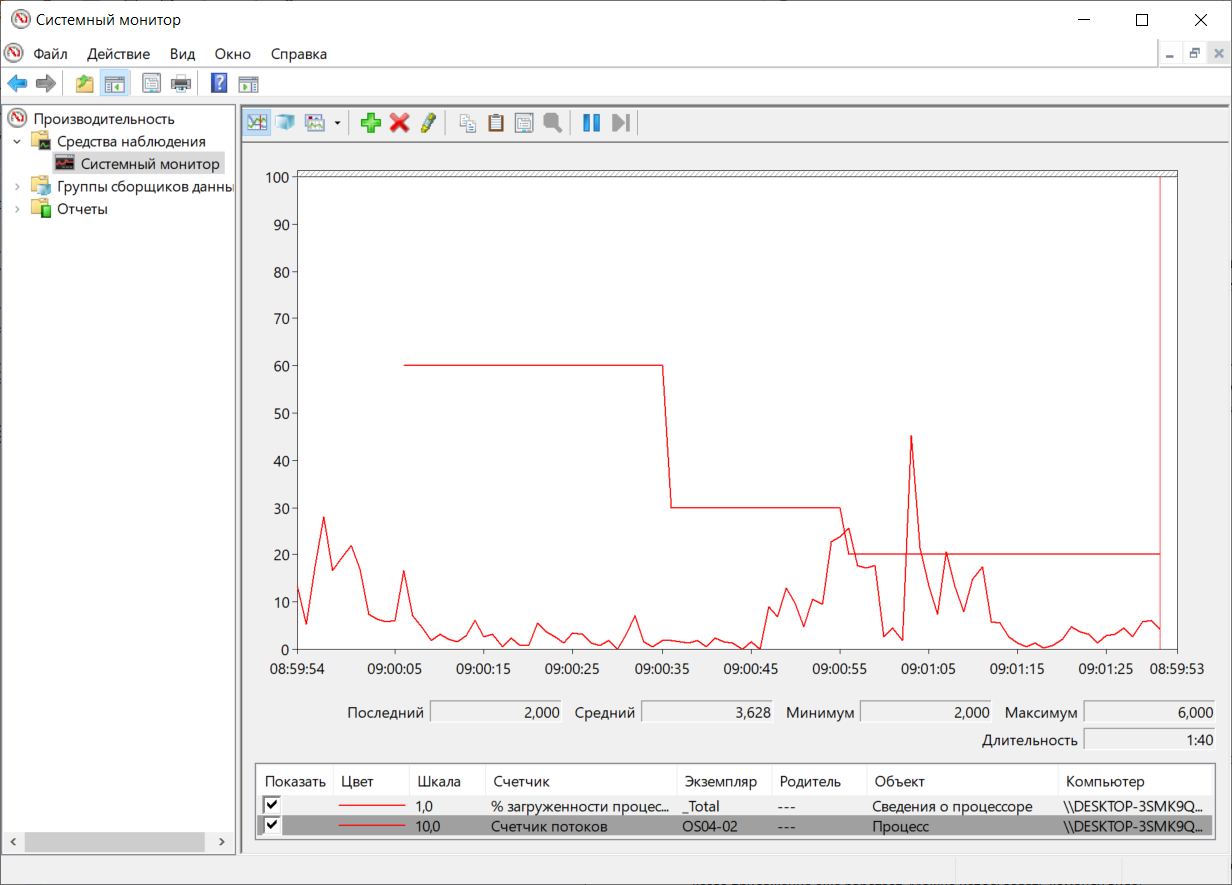
**Задание 2.** Разработайте консольное Windows-приложение OS04\_02 на языке С++, выполняющее цикл 100 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса. Процесс OS04\_02 должен создать два потока: потоковые функции OS04\_02\_T1, OS04\_02\_T2. Поток OS04\_02\_T1 - выполняет цикл 50 итераций с временной задержкой в 1 сек. с выводом на консоль идентификаторов процесса и потока. Поток OS04\_02\_T2 - выполняет цикл 125 итераций с временной задержкой в 1 сек. с выводом на консоль идентификаторов процесса и потока.

Результат выполнения в консоли:

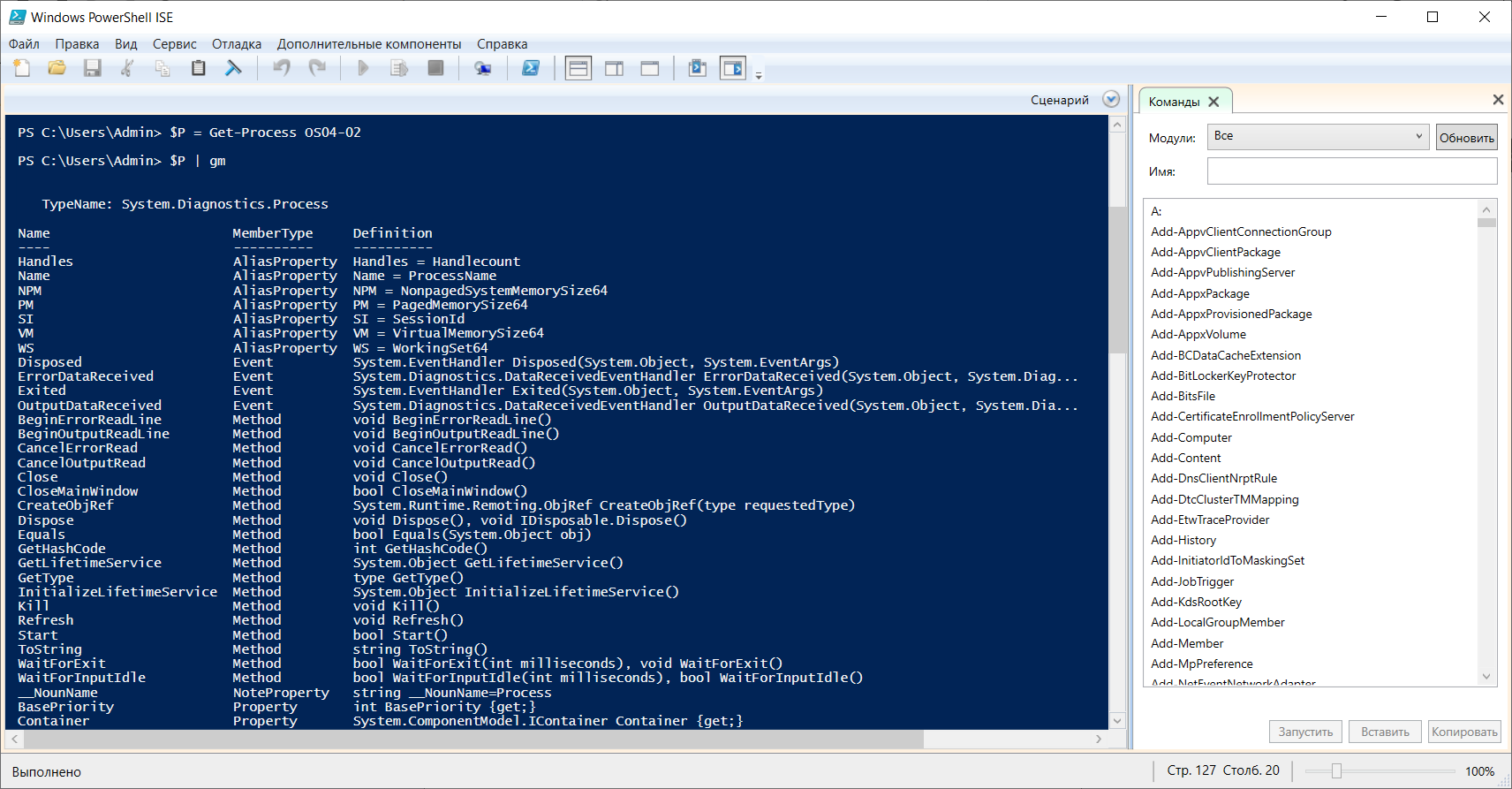


Продемонстрируйте информацию об потоках процесса OS04\_02 с помощью утилит PowerShell ISE и Performance Monitor.

Performance Monitor:

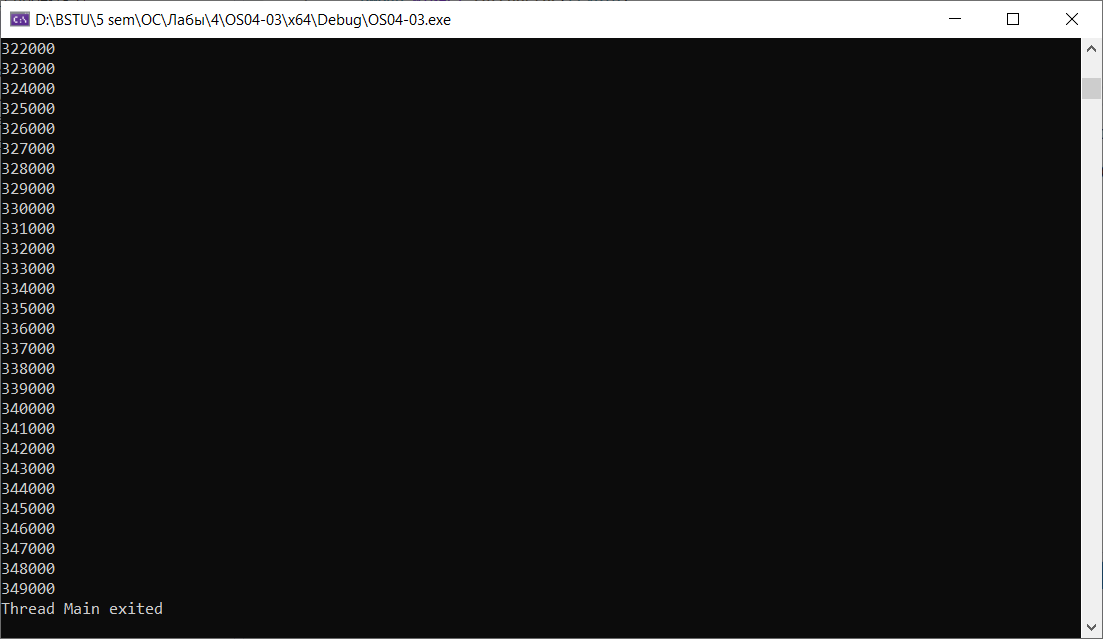


Windows PowerShell ISE:

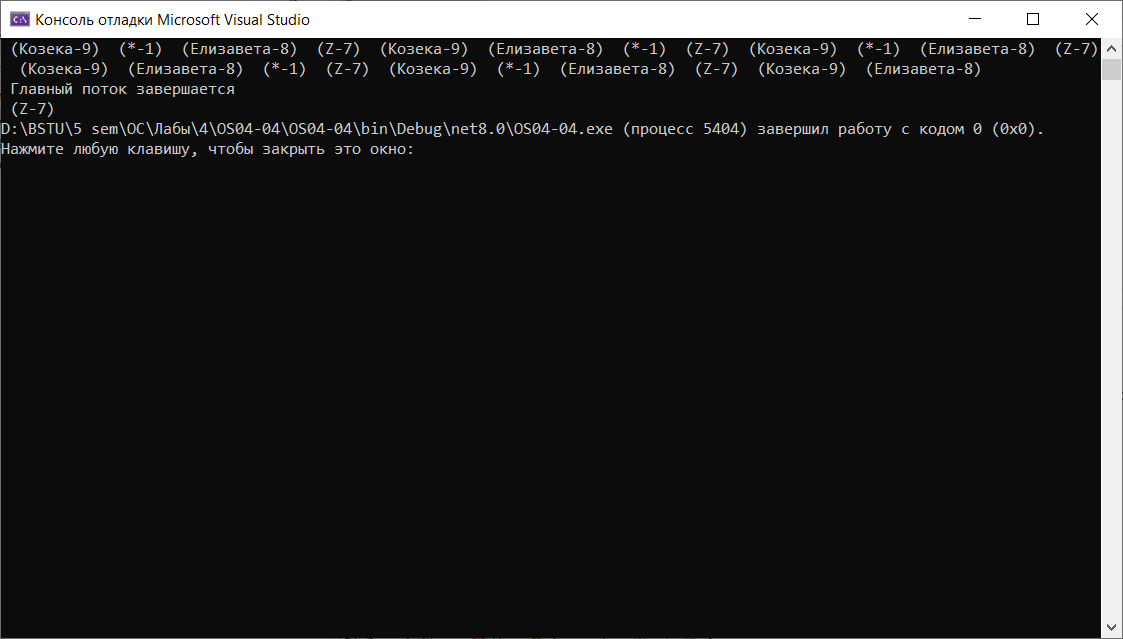


**Задание 3.** Определите, какое максимальное количество потоков можно создать в одном процессе на вашем компьютере, используя нижеприведенный программный код (или аналогичный). Сравните результаты между собой. От чего зависит максимальное количество потоков? Какое ограничение накладывает операционная система?

Результат выполнения в консоли:



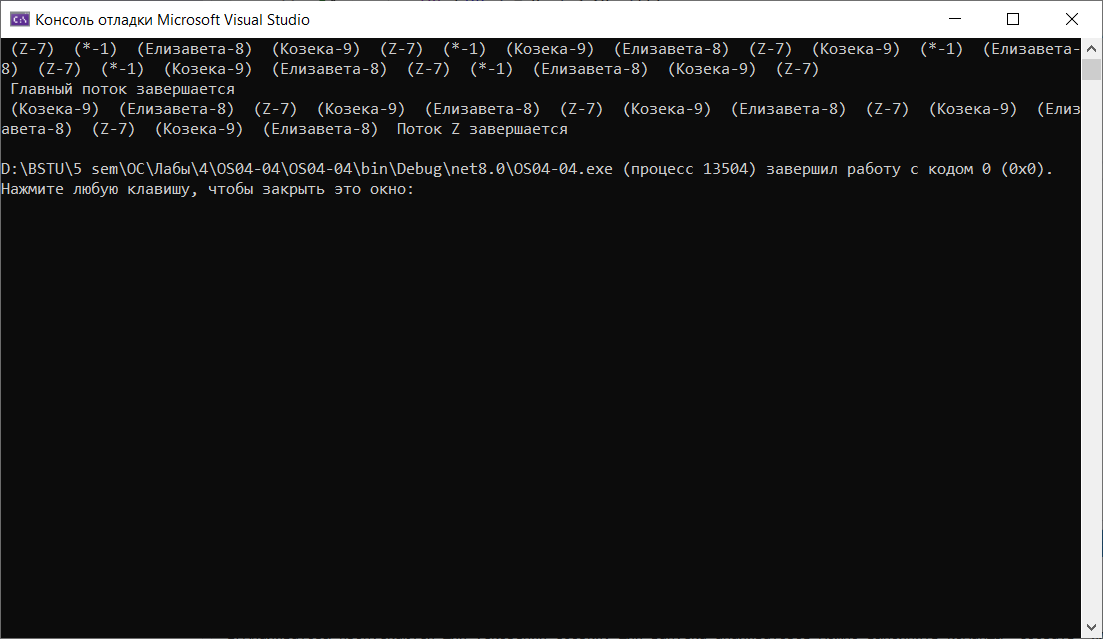
**Задание 4.** Создайте консольное приложение на C#, которое запускает три дополнительных потока и завершается через пять секунд. Один дополнительный поток завершается через 10 секунд, остальные два – через двадцать секунд. Фрагмент программного кода приведен ниже.



Общее время выполнения — 5 секунд.

Измените значение свойства IsBackground для первого дополнительного потока на false и снова выполните приложение.

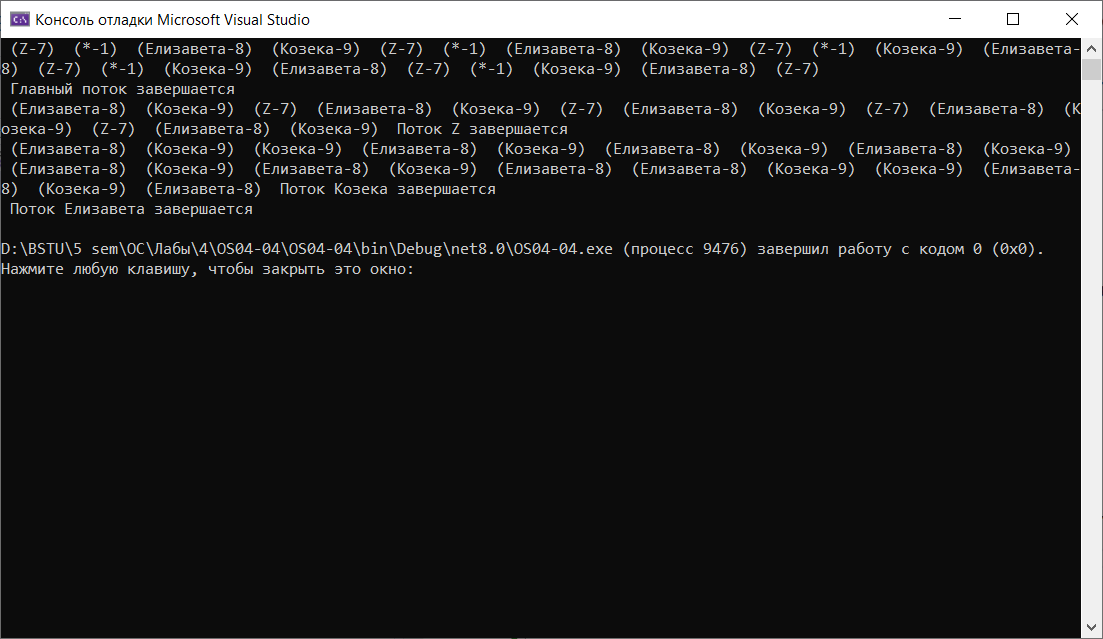
Результат выполнения в консоли:



Общее время выполнения — 10 секунд.

Измените значение свойства IsBackground для второго дополнительного потока на false и снова выполните приложение.

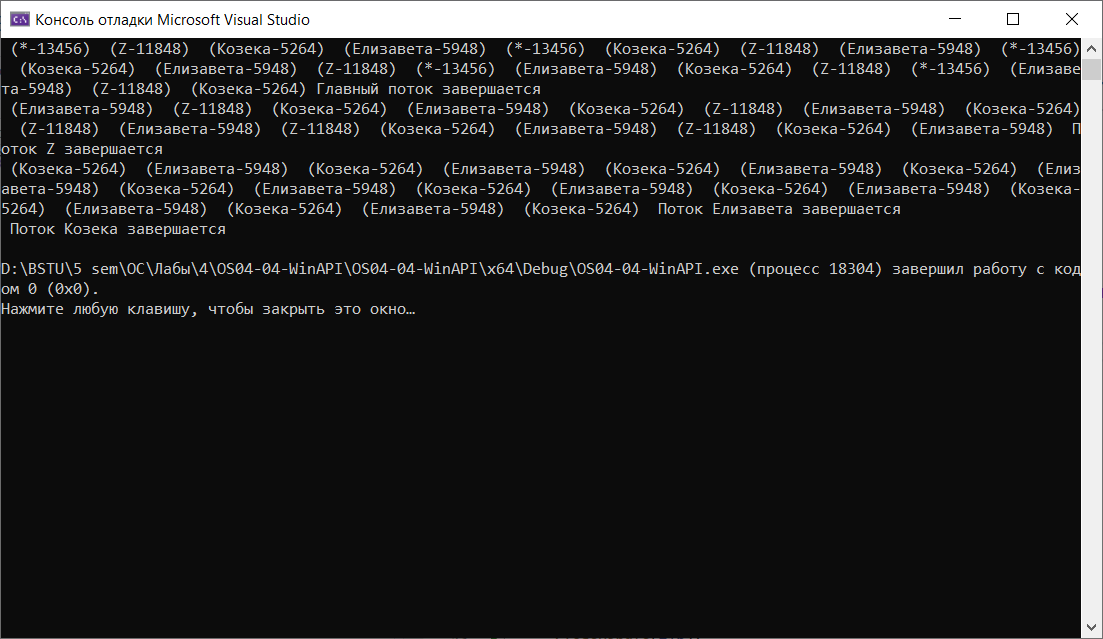
Результат выполнения в консоли:



Общее время выполнения — 20 секунд.

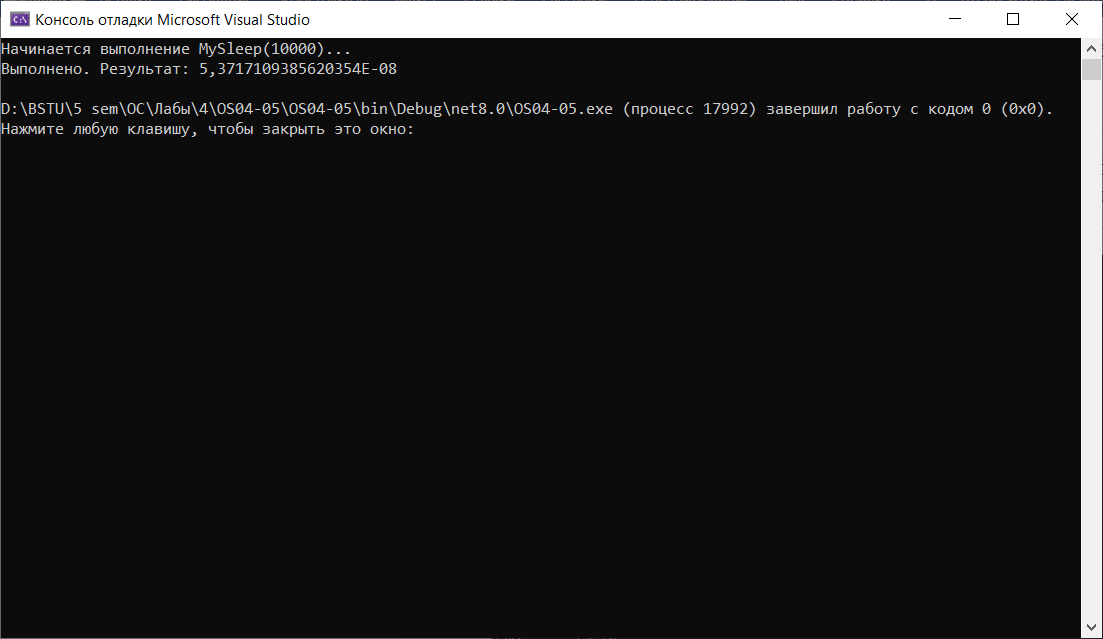
Как получить аналогичный результат в «чистом» Windows API?

Результат выполнения в консоли:

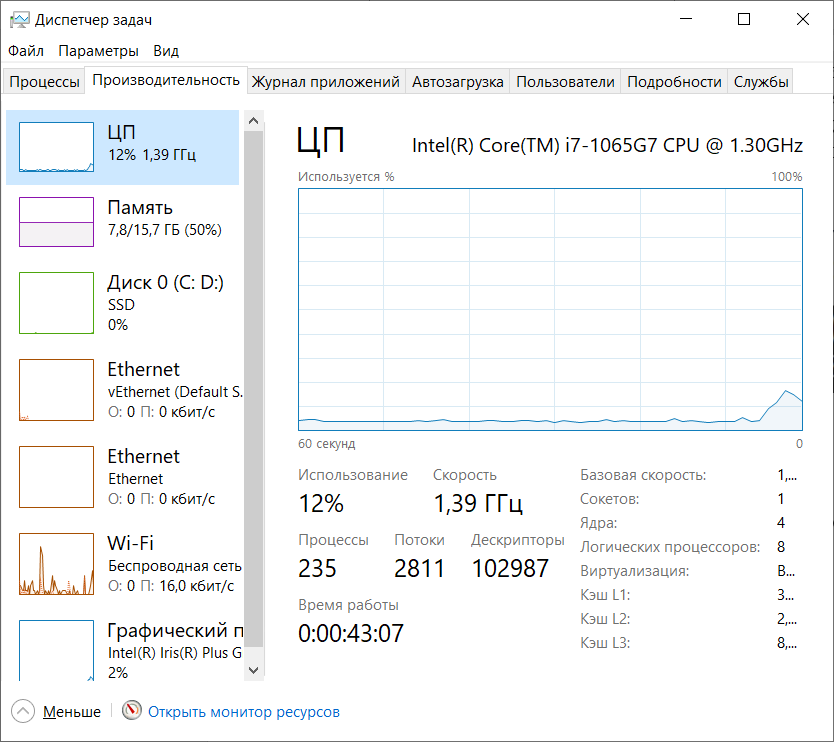


**Задание 5.** Создайте функцию, которая производит любые вычисления длительностью n миллисекунд на вашем компьютере (для последующих заданий метод Thread.Sleep(n) не подходит, так как он освобождает центральный процессор и ничего не делает). Убедитесь, что MySleep(10000) работает ровно 10 секунд.

Результат выполнения в консоли:

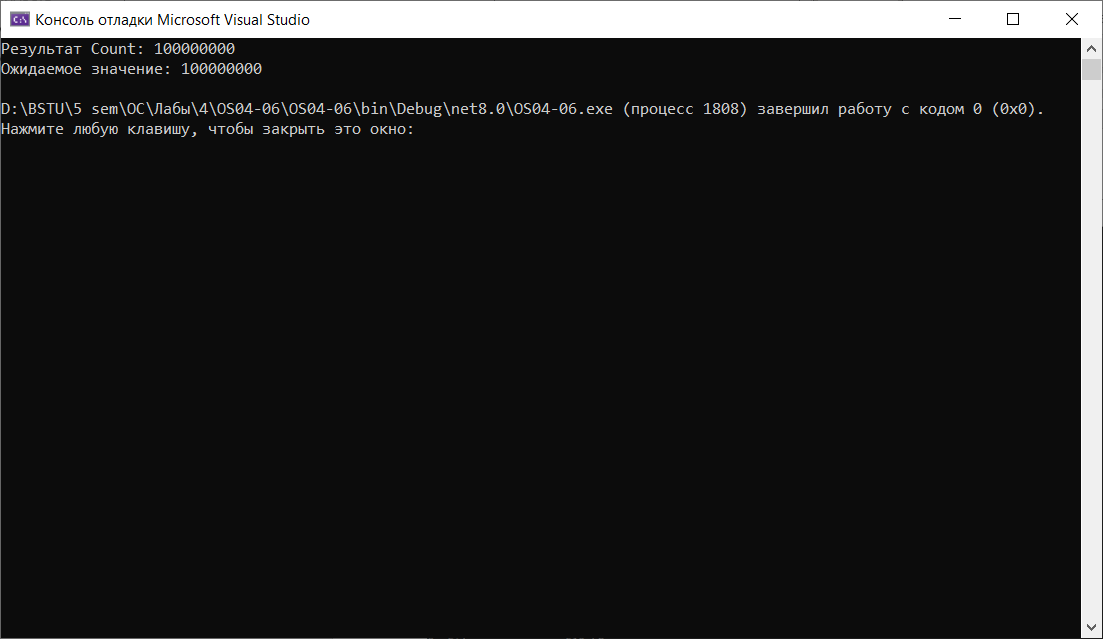


Узнайте количество ядер и логических процессоров в вашем компьютере.



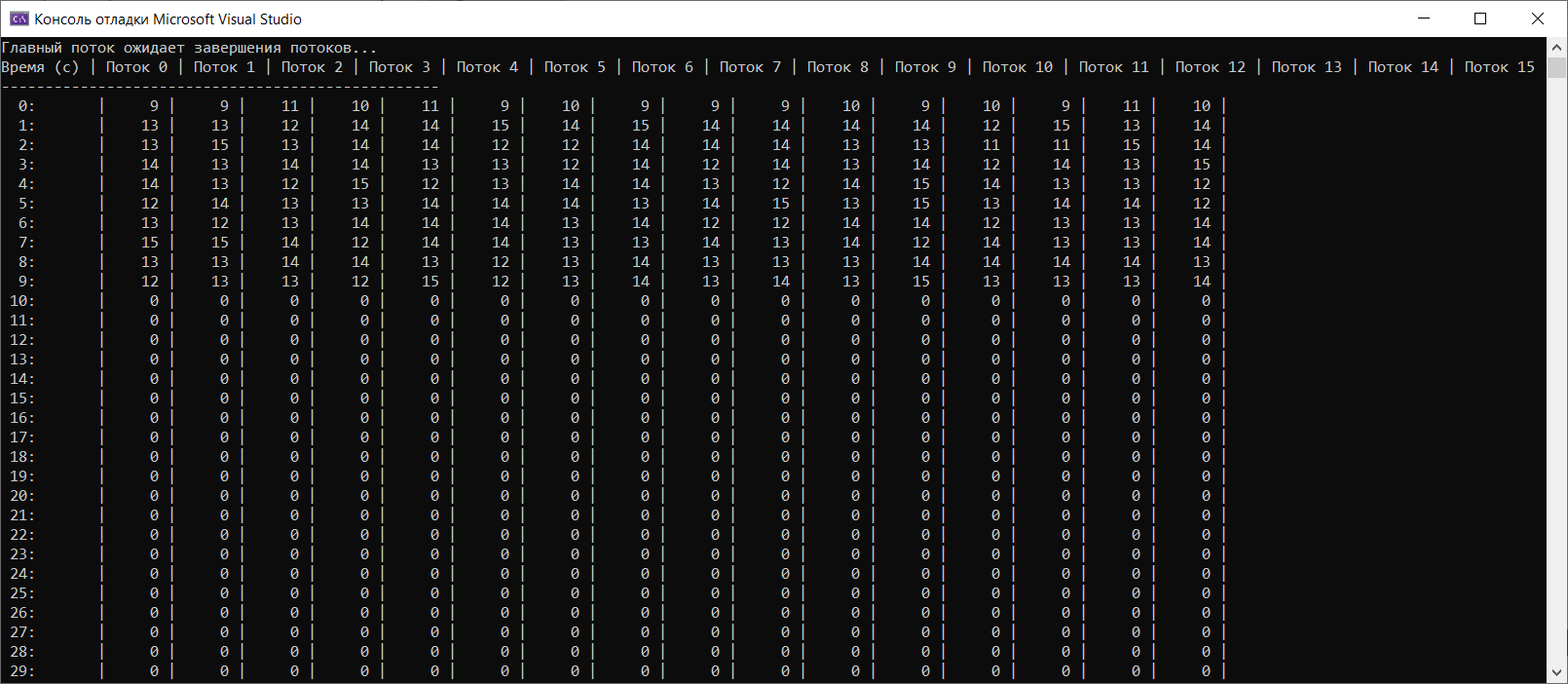
**Задание 6.** Разработайте консольное приложение OS04\_06 на языке С#, запускающее 20 потоков, каждый из которых в цикле 5000000 раз увеличивает на единицу значение общей для всех потоков переменной. Исходное значение переменной — ноль. Выведите результат и сравните с произведением 20\*5000000.

Результат выполнения в консоли:



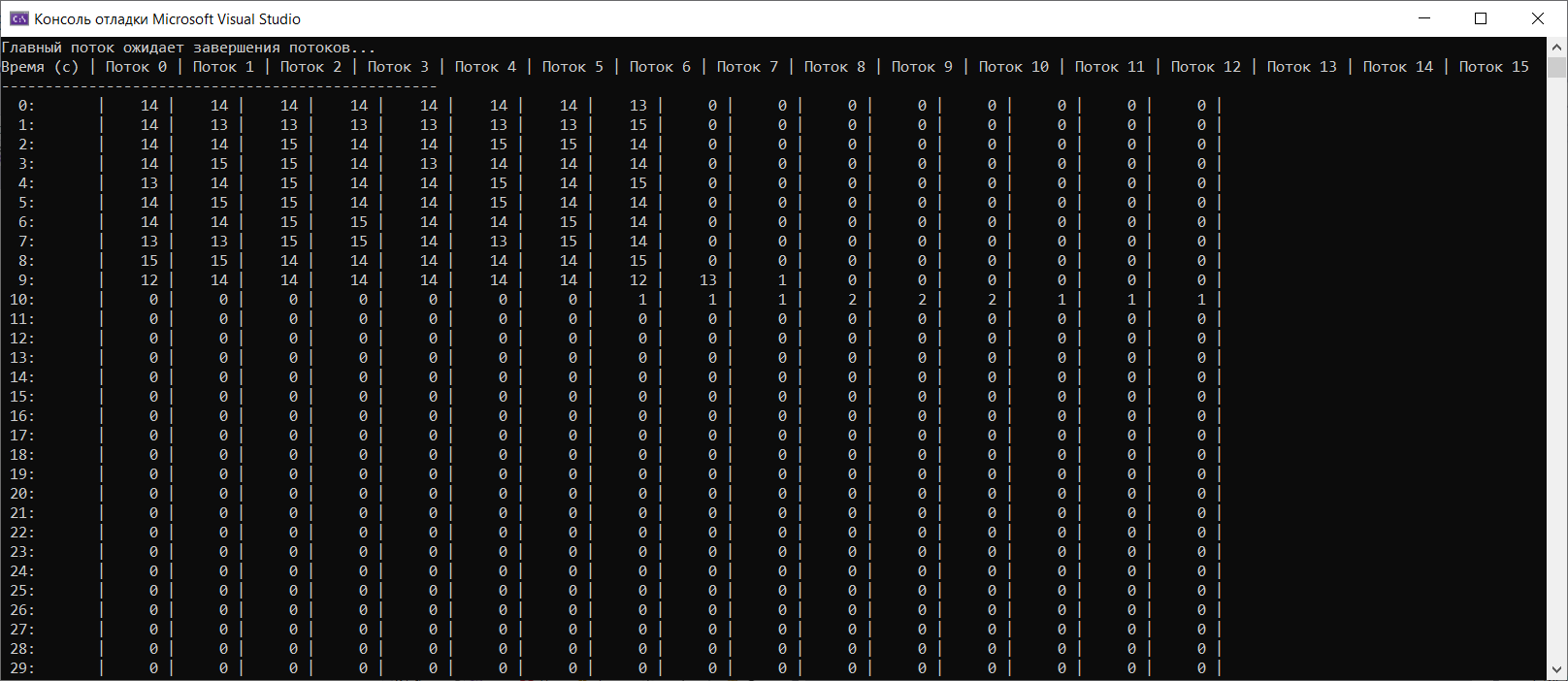
**Задание 7.** Разработайте консольное приложение OS04\_07 на языке С#, запускающее N потоков, каждый из которых будет производить вычисления t секунд (использовать разработанный в задании 5 метод), используя класс System.Threading.Thread. Сохраните информацию о работе потоков в течение T секунд и выведите на экран в виде таблицы. Подберите подходящие параметры в зависимости от количества логических процессоров в вашем компьютере (например, для четырех логических процессоров N = 10, t = 10, T=30).

Результат выполнения в консоли:



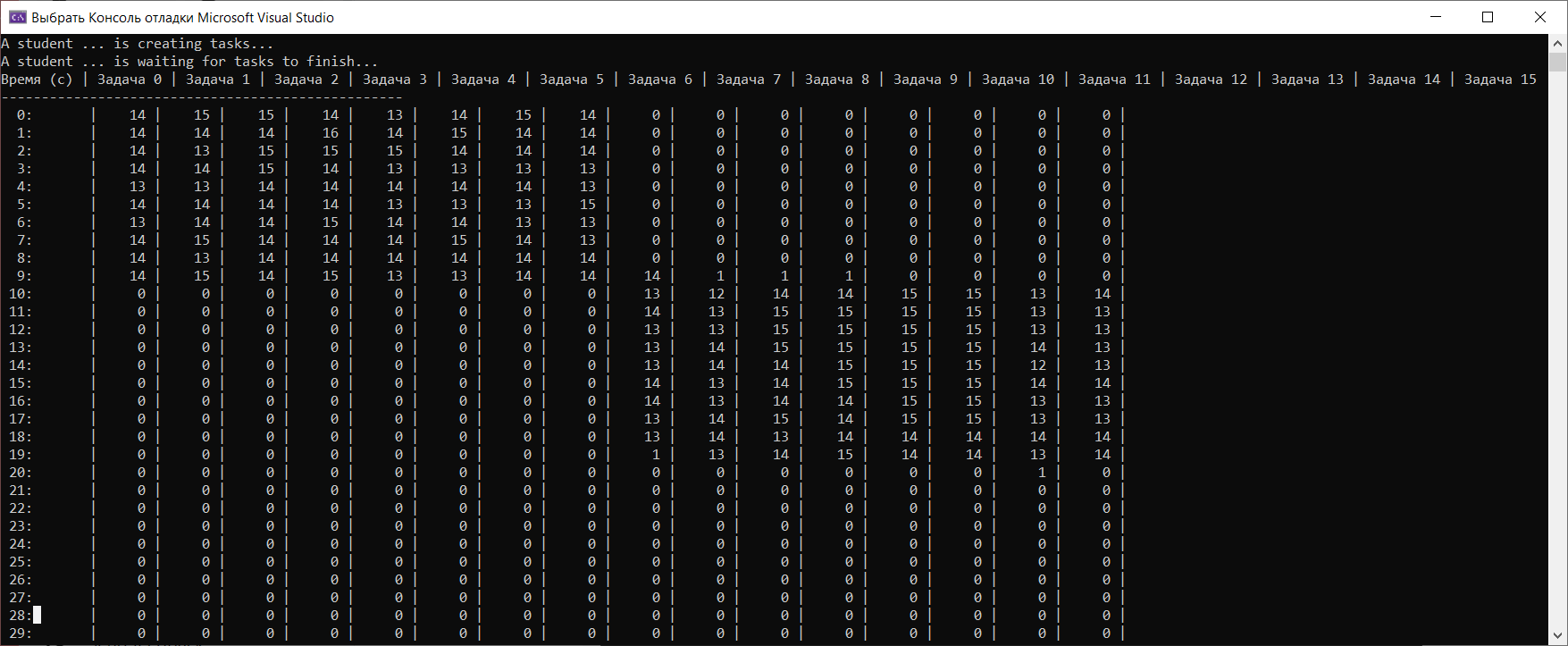
**Задание 8.** Скопируйте консольное приложение OS04\_07 как OS04\_08. Теперь используйте пул потоков. Выведите статистику работы потоков на экран в виде таблицы.

Результат выполнения в консоли:



**Задание 9.** Скопируйте консольное приложение OS04\_07 как OS04\_09. На этот раз используйте System.Threading.Tasks.Task. Выведите статистику работы потоков на экран в виде таблицы. Сравните результаты заданий 7-9 и запишите вывод в отчет.

Результат выполнения в консоли:



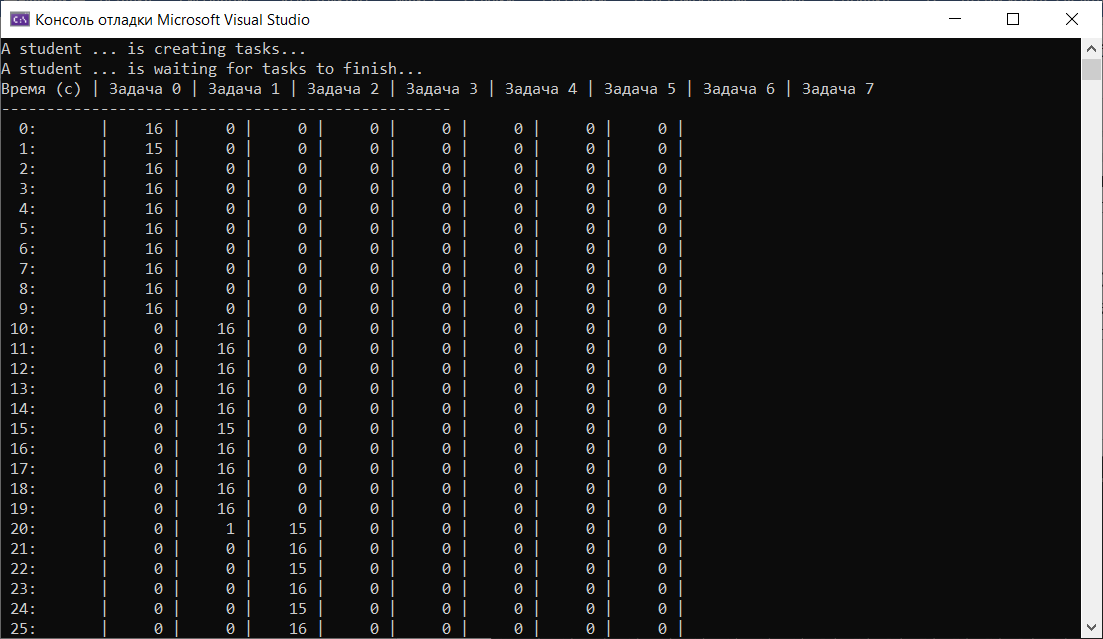
Задание 7 (Потоки): используется явное управление потоками, что дает возможность точно задавать количество и продолжительность работы каждого потока, но может приводить к сложностям с синхронизацией.

Задание 8 (Пул потоков): использует ThreadPool, что упрощает управление потоками и повышает производительность за счет переиспользования потоков. Это также снижает накладные расходы на создание и уничтожение потоков.

Задание 9 (Задачи): использует Task, что позволяет еще больше упростить код и улучшить читаемость. Task предоставляет более высокоуровневую абстракцию по сравнению с потоками и пулом потоков, позволяя легко работать с асинхронными операциями и параллельными вычислениями.

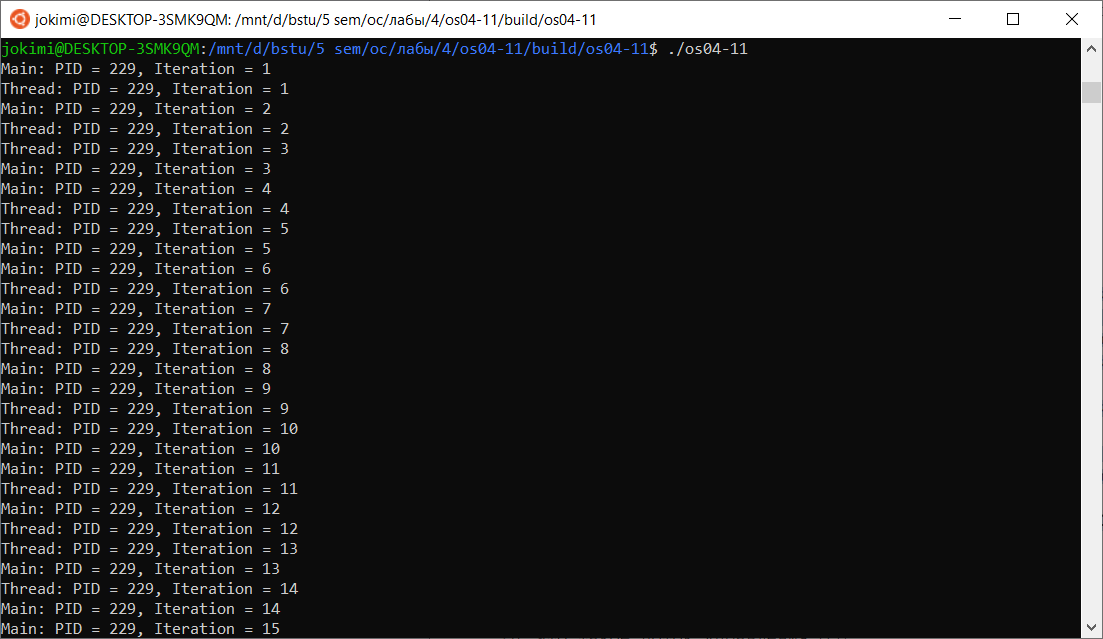
**Задание 10.** Скопируйте консольное приложение OS04\_09 как OS04\_10. Уменьшите количество задач до количества логических процессоров. Организуйте выполнение задач по очереди. Выведите статистику работы потоков на экран в виде таблицы.

Результат выполнения в консоли:

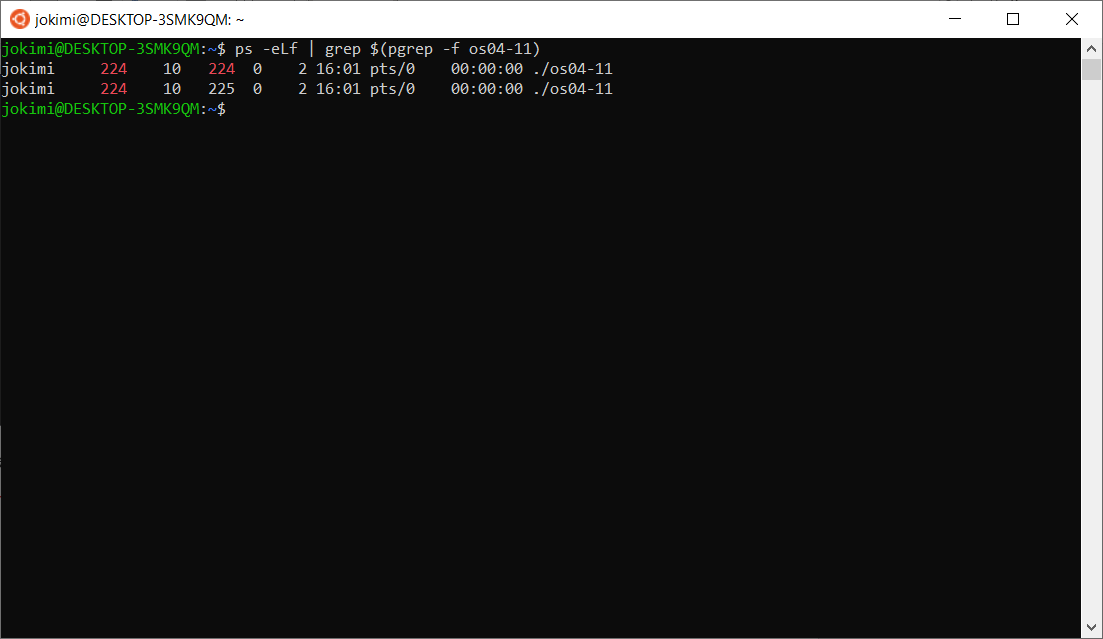


**Задание 11.** Разработайте консольное Linux-приложение OS04\_11 на языке С, выполняющее цикл 100 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса (использовать функции из pthread.h). Процесс OS04\_11 должен создать поток: потоковая функция OS04\_11\_T1. Поток OS04\_11\_T1 выполняет цикл 75 итераций с временной задержкой в 1 сек. с выводом на консоль идентификаторов процесса.

Результат выполнения в консоли:



Продемонстрируйте информацию о потоках процесса OS04\_11 с помощью утилиты ps.



Ответы на вопросы:

**1. Что такое поток управления OS?**

Поток управления — это наименьшая единица выполнения в операционной системе. Он представляет собой последовательность команд, которые могут выполняться независимо от других потоков в рамках одного процесса. Потоки позволяют программе выполнять несколько задач одновременно, улучшая производительность и отзывчивость.

**2. С помощью каких системных вызовов создаются потоки в Windows и Linux?**

Windows: CreateThread() (создание нового потока) и ExitThread() (завершение выполнения потока).

Linux: pthread\_create() (создание потоков) и pthread\_exit() (завершение выполнения потока).

**3. Что такое системные и пользовательские потоки?**

Системные потоки создаются и управляются операционной системой. Они обеспечивают базовую функциональность многопоточности и могут включать в себя такие потоки, как потоки ядра.

Пользовательские потоки создаются и управляются приложением. Они предоставляют более высокоуровневый интерфейс и могут быть реализованы с использованием библиотек, таких как pthreads в Linux.

**4. Что такое многопоточность?**

Многопоточность — это способность программы выполнять несколько потоков одновременно. Это позволяет улучшить производительность, так как разные потоки могут выполнять разные задачи, используя ресурсы процессора более эффективно.

**5. Что такое контекст потока и для чего он нужен?**

Контекст потока — это набор данных, который включает в себя состояние (регистры, стек и т.д.) потока. Он необходим для сохранения информации о текущем состоянии потока, чтобы система могла приостановить и возобновить его выполнение.

**6. Перечислите состояния, в которых может быть поток, и поясните их назначение.**

Созданный (New): поток создан, но ещё не запущен.

Готовый (Ready): поток готов к выполнению и ожидает выделения процессорного времени.

Запущенный (Running): поток выполняется на процессоре.

Заблокированный (Blocked): поток не может продолжать выполнение, ожидая какого-либо события (например, завершения ввода-вывода).

Завершённый (Terminated): поток завершил выполнение.

**7. Что такое LWP?**

LWP (Lightweight Process) — это легковесный процесс, который управляется операционной системой и может быть использован для реализации многопоточности. Это отдельный поток выполнения, который может работать параллельно с другими LWPs в рамках одного процесса.

**8. Что такое потокобезопасность программного кода?**

Потокобезопасность — это свойство кода, при котором он может корректно выполняться в многопоточной среде без риска возникновения гонок данных или других проблем, связанных с конкурентным доступом к общим ресурсам.

**9. Что такое реентерабельность кода?**

Реентерабельный код — это код, который может быть вызван несколькими потоками одновременно без риска повреждения данных или состояния. Он использует локальные переменные и не зависит от глобального состояния.

**10. Что такое Fiber?**

Fiber (волокно) — это легковесный поток, который управляется приложением, а не операционной системой. Волокна позволяют управлять выполнением контекста более гибко, чем традиционные потоки, но требуют от разработчика большей ответственности за переключение контекстов.