Учреждение образования

«Белорусский государственный технологический университет»

**Лабораторная работа №4**

Выполнил:

Студент 3 курса 4 группы ФИТ

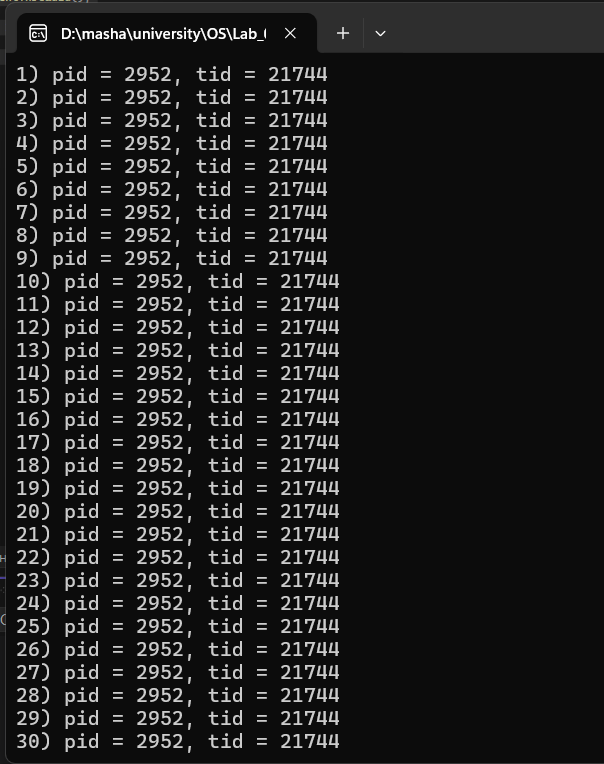
Сосновец Мария Игоревна

2024 г.

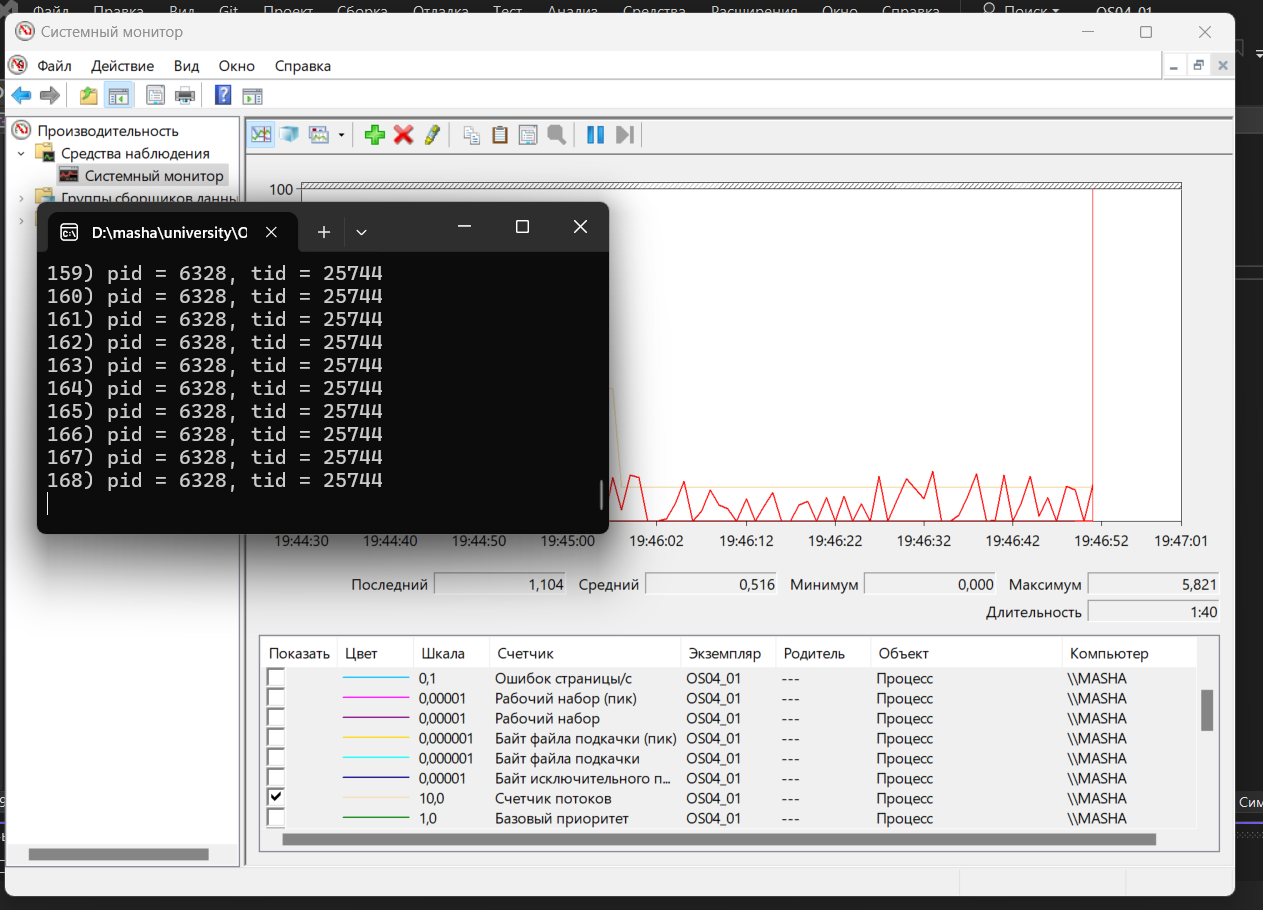
**Задание 01**

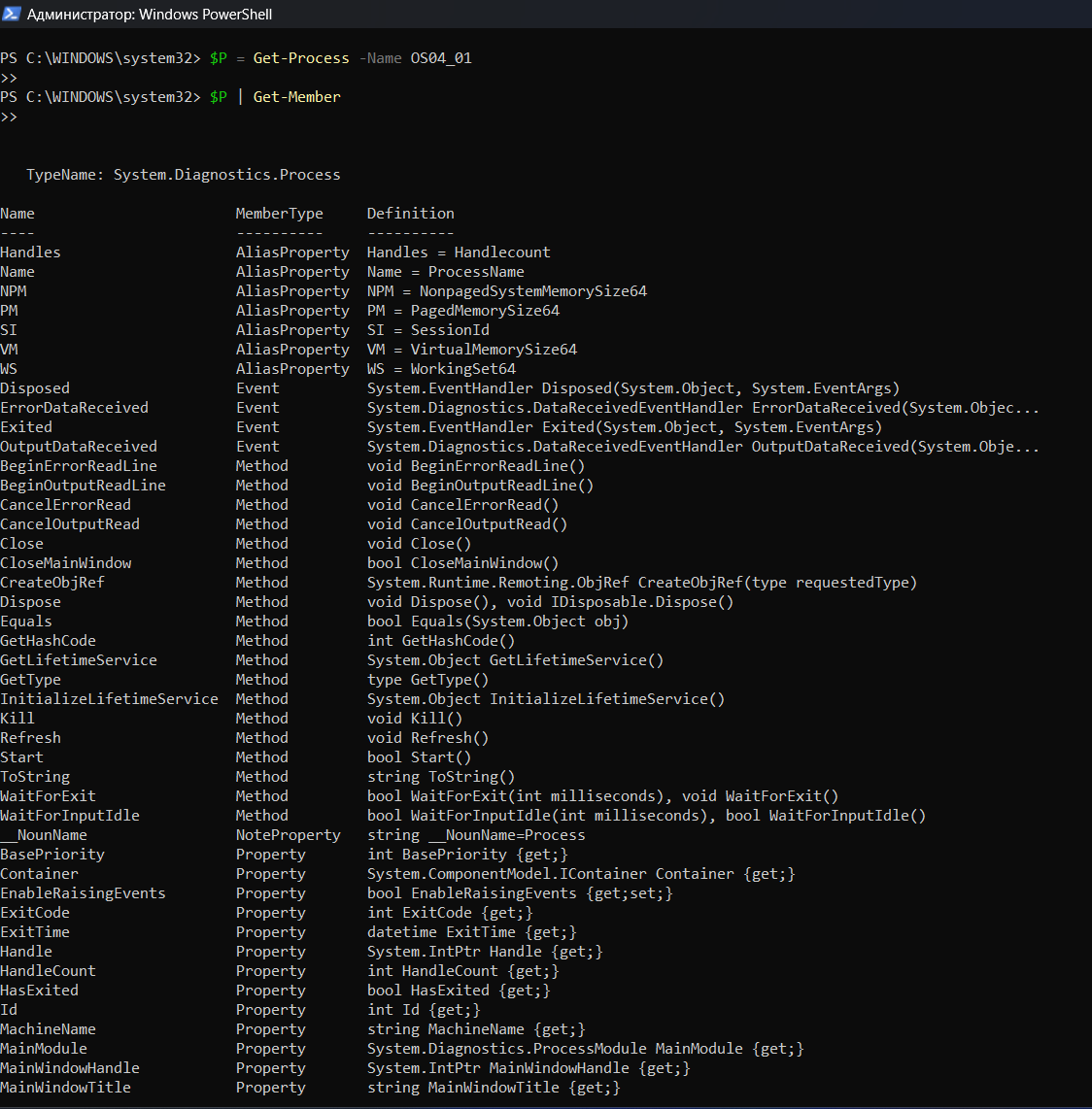
1. Разработайте консольное Windows-приложение OS04\_01 на языке С++, выполняющее длинный цикл с временной задержкой и с выводом на консоль идентификаторов текущего процесса и текущего потока.

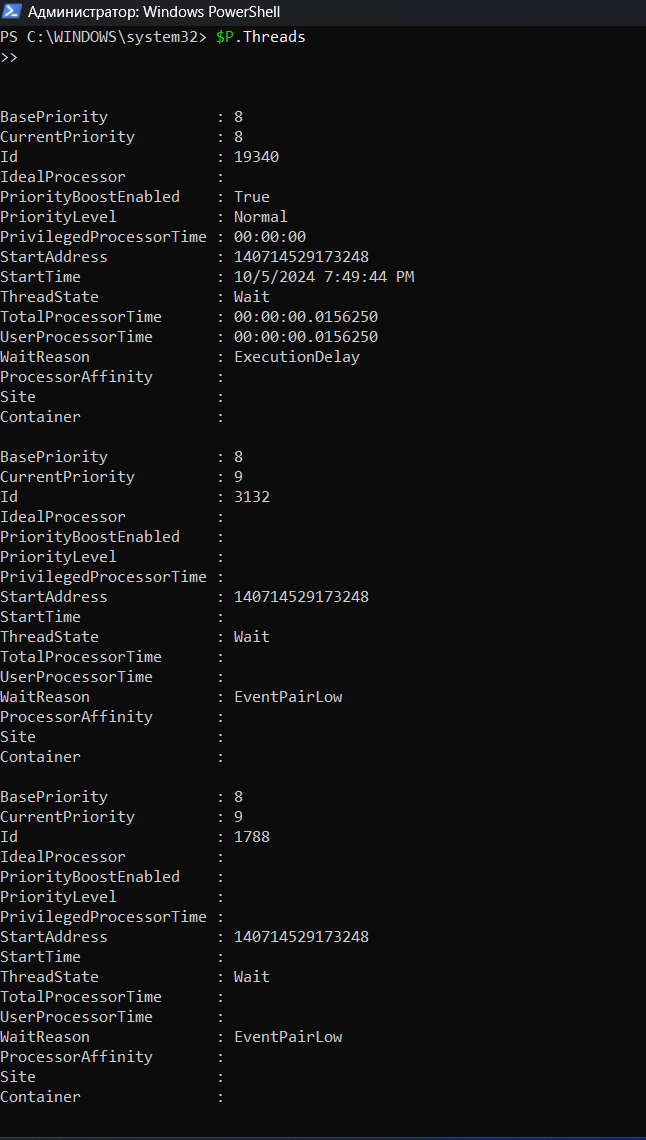
|  |
| --- |
| #include <iostream>  #include <Windows.h>  using namespace std;  int main() {  DWORD pid = GetCurrentProcessId();  DWORD tid = GetCurrentThreadId();  for (int i = 1; i < 2000; i++) {  Sleep(800);  cout << i << ") pid = " << pid << ", tid = " << tid << endl;  }  } |

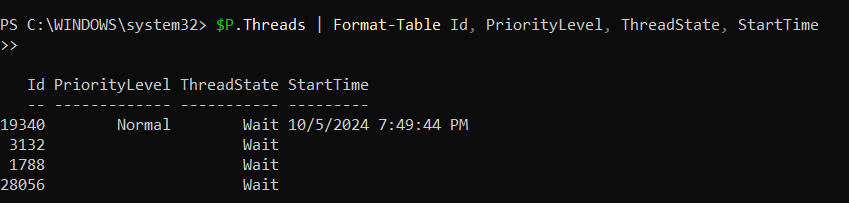
****

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





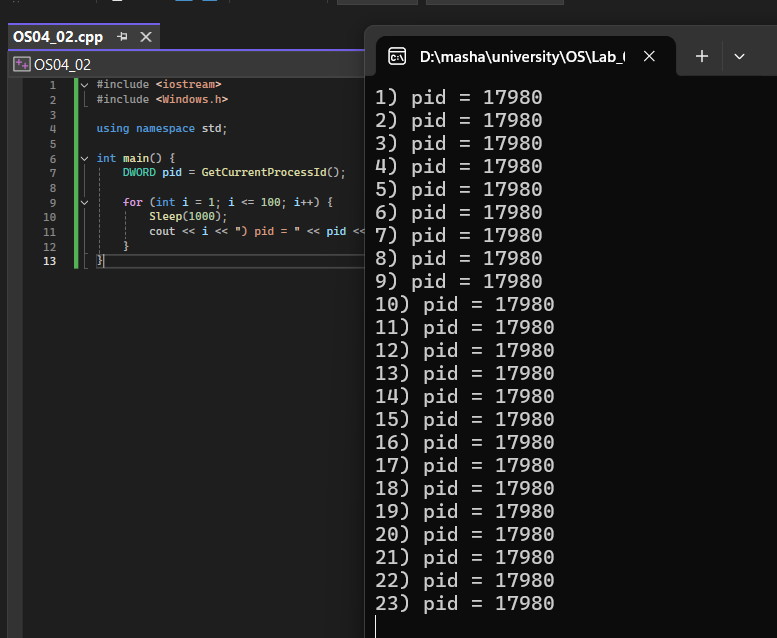




**Задание 02**

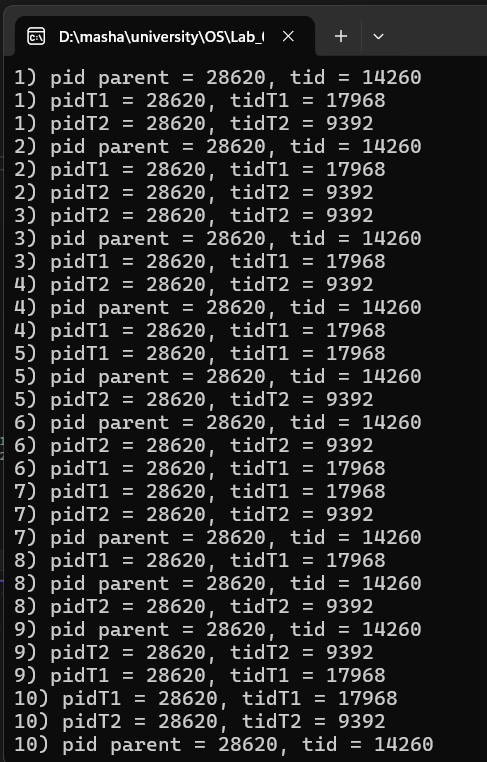
3. Разработайте консольное Windows-приложение OS04\_02 на языке С++, выполняющее цикл 100 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.

|  |
| --- |
| #include <iostream>  #include <Windows.h>  using namespace std;  int main() {  DWORD pid = GetCurrentProcessId();  for (int i = 1; i <= 100; i++) {  Sleep(1000);  cout << i << ") pid = " << pid << endl;  }  } |



4. Процесс OS04\_02 должен создать два потока: потоковые функции OS04\_02\_T1, OS04\_02\_T2.

|  |
| --- |
| #include <iostream>  #include <Windows.h>  using namespace std;  DWORD OS04\_02\_T1() {  DWORD pid = GetCurrentProcessId();  DWORD tid = GetCurrentThreadId();  for (int i = 0; i <= 50; i++) {  Sleep(1000);  cout << i << ") pidT1 = " << pid << ", tidT1 = " << tid << endl;  }  return 0;  }  DWORD OS04\_02\_T2() {  DWORD pid = GetCurrentProcessId();  DWORD tid = GetCurrentThreadId();  for (int i = 0; i <= 125; i++) {  Sleep(1000);  cout << i << ") pidT2 = " << pid << ", tidT2 = " << tid << endl;  }  return 0;  }  void main() {  DWORD pid = GetCurrentProcessId();  DWORD tid = GetCurrentThreadId();  DWORD T1\_CHILD = NULL;  DWORD T2\_CHILD = NULL;  HANDLE HCHILDT1 = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)OS04\_02\_T1, NULL, 0, &T1\_CHILD);  HANDLE HCHILDT2 = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)OS04\_02\_T2, NULL, 0, &T2\_CHILD);  for (int i = 1; i <= 100; i++) {  Sleep(1000);  cout << i << ") pid parent = " << pid << ", tid = " << tid << endl;  }  WaitForSingleObject(HCHILDT1, INFINITE);  WaitForSingleObject(HCHILDT2, INFINITE);  CloseHandle(HCHILDT1);  CloseHandle(HCHILDT2);  } |



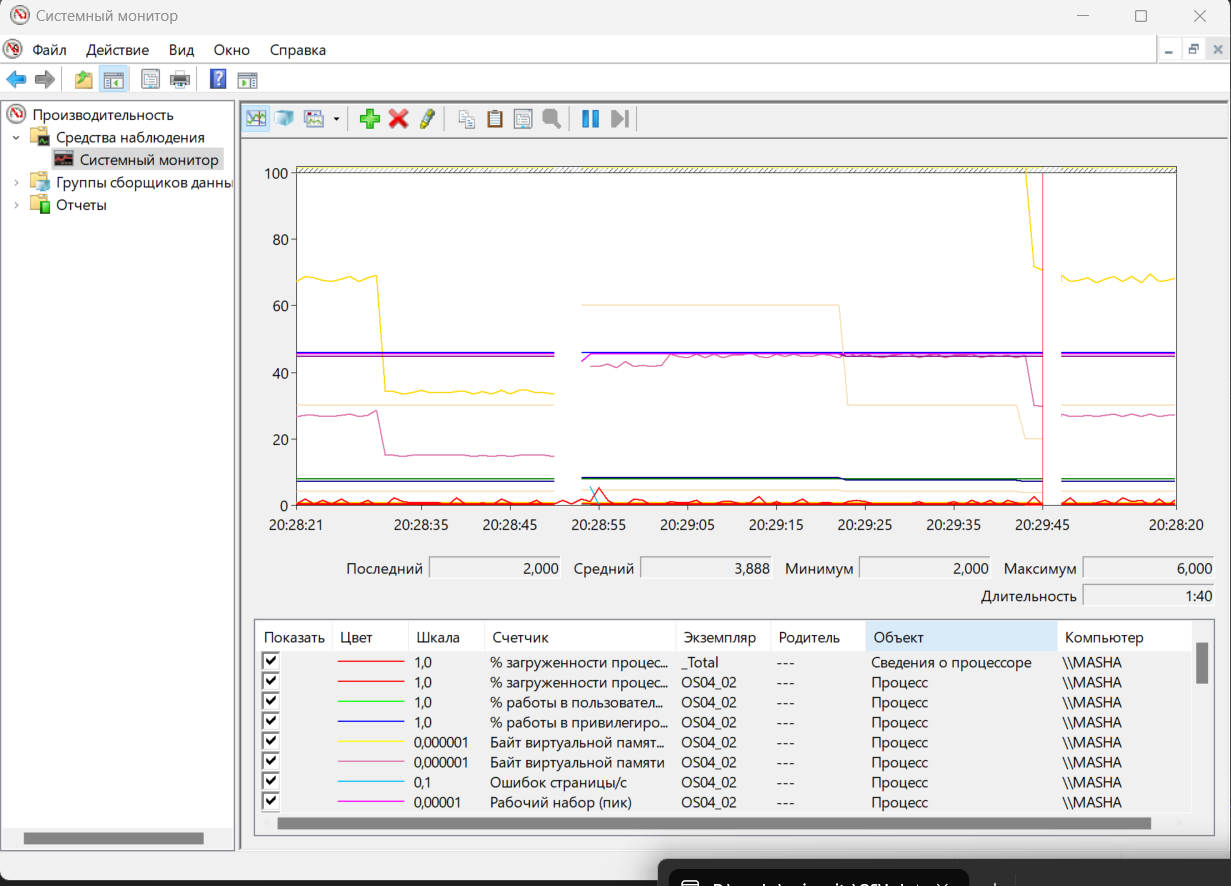
5. Поток OS04\_02\_T1 - выполняет цикл 50 итераций с временной задержкой в 1 сек. с выводом на консоль идентификаторов процесса и потока.

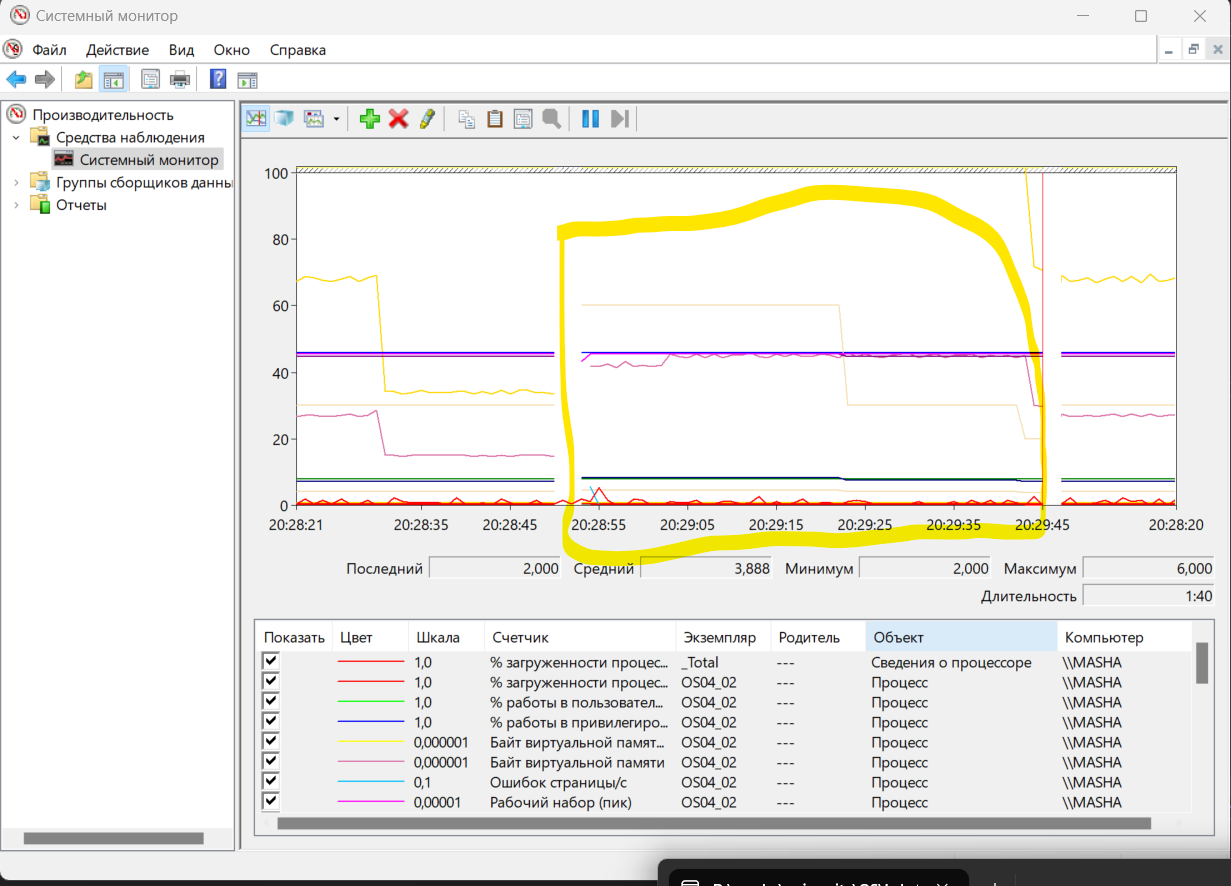
|  |
| --- |
| DWORD OS04\_02\_T1() {  DWORD pid = GetCurrentProcessId();  DWORD tid = GetCurrentThreadId();  for (int i = 0; i <= 50; i++) {  Sleep(1000);  cout << i << ") pidT1 = " << pid << ", tidT1 = " << tid << endl;  }  return 0;  } |

6. Поток OS04\_02\_T2 - выполняет цикл 125 итераций с временной задержкой в 1 сек. с выводом на консоль идентификаторов процесса и потока.

|  |
| --- |
| DWORD OS04\_02\_T2() {  DWORD pid = GetCurrentProcessId();  DWORD tid = GetCurrentThreadId();  for (int i = 0; i <= 125; i++) {  Sleep(1000);  cout << i << ") pidT2 = " << pid << ", tidT2 = " << tid << endl;  }  return 0;  } |

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

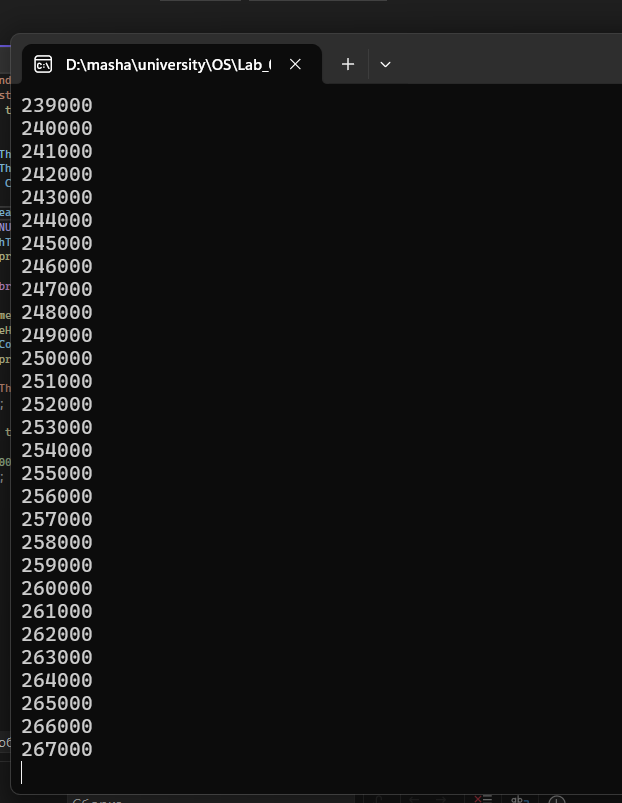




**Задание 03.**

8. 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 3 максимальное количество потоков? Какое ограничение накладывает операционная система.

Около 30 тысяч. Максимальное количество потоков зависят от от памяти (размер стека на поток), архитектуры системы (32 или 64 бита) и ограничений ОС. В Windows каждый поток по умолчанию использует 1 МБ памяти для стека.



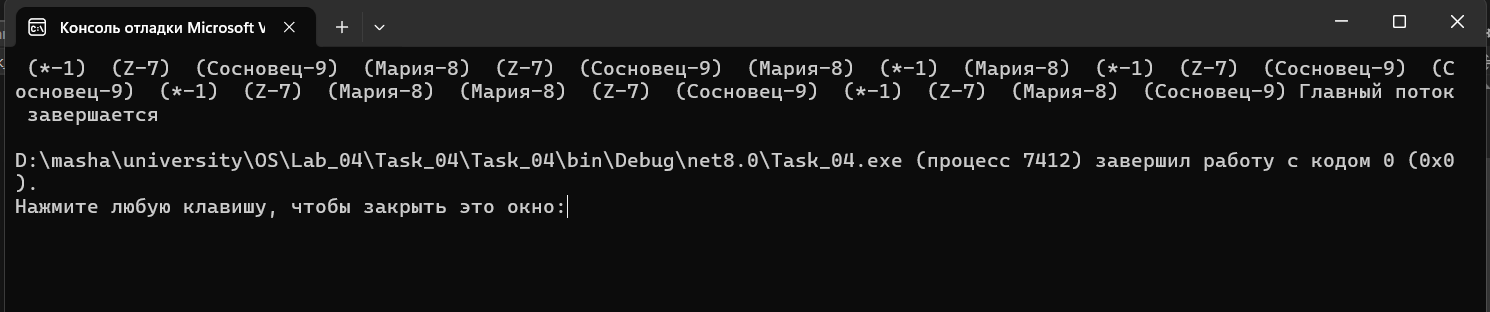
**Задание 04**

9. 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. (Вставьте Свои ИмяФамилию).

|  |
| --- |
| using System;  using System.Threading;  class Program  {  // Поток Z работает 10 секунд  static void ThreadZed()  {  for (int i = 0; i < 10; i++)  {  Console.Write(" (Z-{0}) ", Thread.CurrentThread.ManagedThreadId);    Thread.Sleep(1000);  }  Console.WriteLine(" Поток Z завершается ");  }  // Поток работает 20 секунд, параметр - строка-идентификатор  static void ThreadWithParam(object o)  {  for (int i = 0; i < 20; i++)  {  Console.Write(" ({0}-{1}) ", o.ToString(),  Thread.CurrentThread.ManagedThreadId);  Thread.Sleep(1000);  }  }  static void Main(string[] args)  {  var t1 = new Thread(ThreadZed);  var t1a = new Thread(ThreadWithParam);  var t1b = new Thread(ThreadWithParam);  Фоновые потоки не блокируют завершение приложения  t1.IsBackground = true; // false для п.11  t1a.IsBackground = true; // false для п.12  t1b.IsBackground = true;  t1.Start();  t1a.Start("Мария"); // Имя  t1b.Start("Сосновец"); // Фамилия  // Главный поток работает 5 секунд  for (int i = 0; i < 5; i++)  {  Console.Write(" (\*-{0}) ", Thread.CurrentThread.ManagedThreadId);  Thread.Sleep(1000);  }  Console.WriteLine("Главный поток завершается");  }  } |

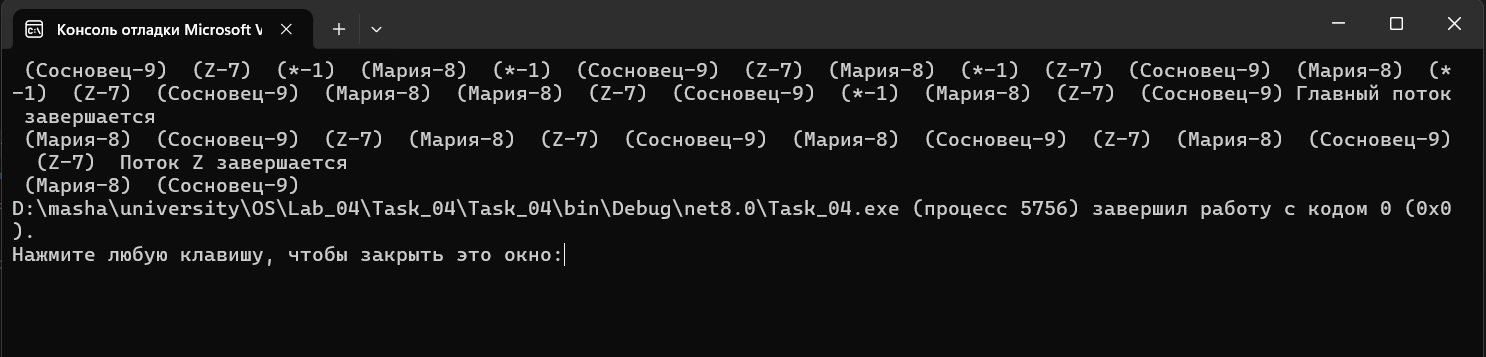
10. Выполните приложение. Обратите внимание на общее время работы приложения.(5 секунд)

Общее время выполнения приложения будет 5 секунд, поскольку основной поток завершится через 5 секунд. Фоновые потоки продолжат свою работу, но так как они фоновые, они также завершатся автоматически после завершения основного потока.



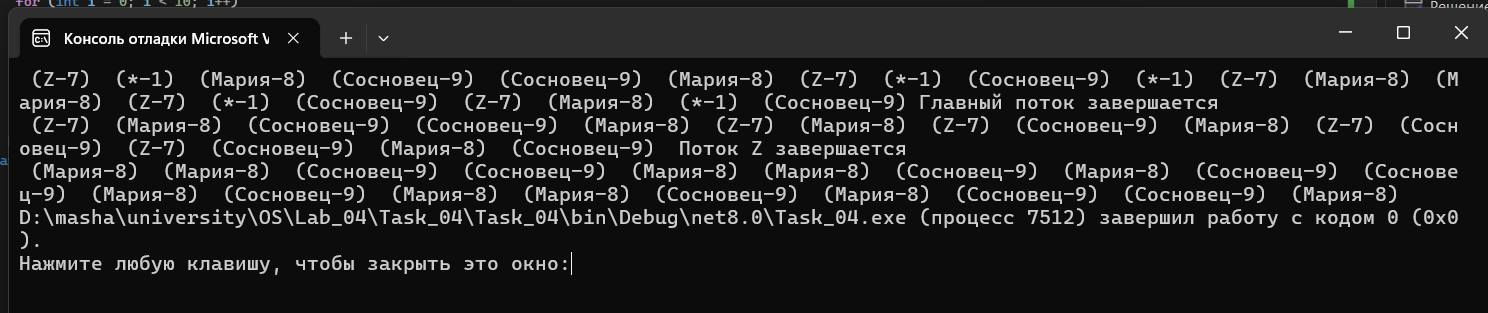
11. Измените значение свойства IsBackground для первого дополнительного потока на false и снова выполните приложение. Обратите внимание на общее время работы приложения.

завершится через **10 секунд**, потому что поток t1 (поток Z) работает 10 секунд, а теперь он не является фоновым и не завершится при завершении основного потока.



12. Измените значение свойства IsBackground для второго дополнительного потока на false и снова выполните приложение. Обратите внимание на общее время работы приложения.

Приложение завершится через **20 секунд**, так как второй поток t1a теперь нефоновый и будет продолжать работать, пока не завершит выполнение всей своей задачи (20 секунд).



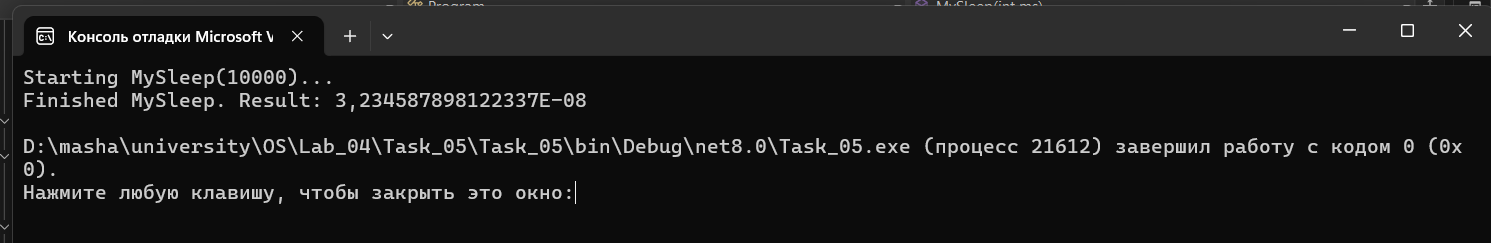
13. Вставьте три скриншота в отчет и объясните результат.

14. [For nerds only] Как получить аналогичный результат в «чистом» Windows API?

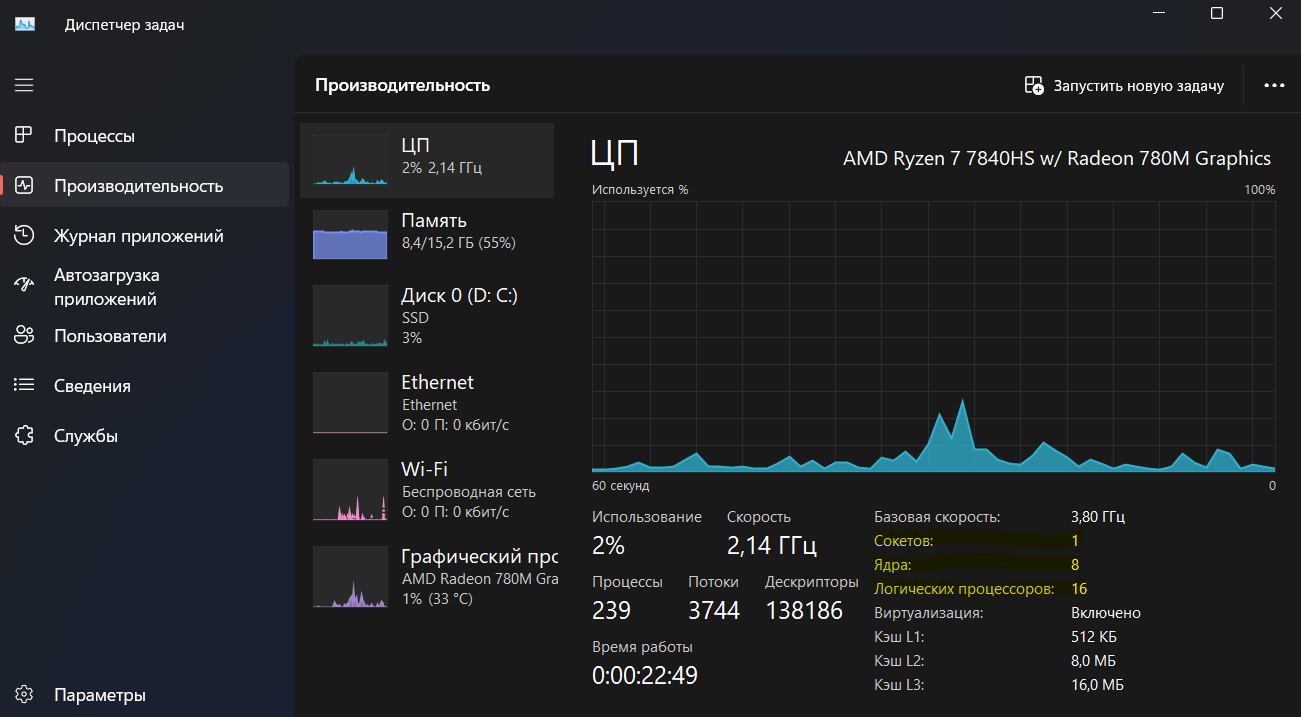
**Задание 05 (вспомогательное)**

15. 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.

|  |
| --- |
| using System;  using System.Diagnostics;  class Program  {  static double MySleep(int ms)  {  Stopwatch stopwatch = new Stopwatch();  stopwatch.Start();  double sum = 0, temp;  int iterations = 5500;  while (stopwatch.ElapsedMilliseconds < ms)  {  for (int t = 0; t < ms; ++t)  {  temp = 0.711 + (double)t / 10000.0;  double a, b, c, d, e, nt;  for (int k = 0; k < iterations; ++k)  {  nt = temp - k / 27000.0;  a = Math.Sin(nt);  b = Math.Cos(nt);  c = Math.Cos(nt / 2.0);  d = Math.Sin(nt / 2.0);  e = Math.Abs(1.0 - a \* a - b \* b) + Math.Abs(1.0 - c \* c - d \* d);  sum += e;  }  }  }  stopwatch.Stop();  return sum;  }  static void Main()  {  int milliseconds = 10000;  Console.WriteLine($"Starting MySleep({milliseconds})...");  double result = MySleep(milliseconds);  Console.WriteLine($"Finished MySleep. Result: {result}");  }  } |



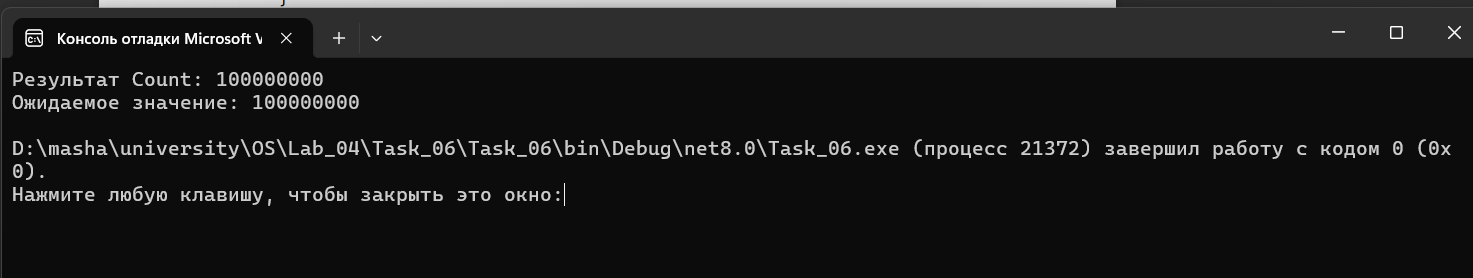
16. Узнайте количество ядер и логических процессоров в вашем компьютере (приложение 1).



**Задание 06**

17. 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 20х5000000.

|  |
| --- |
| using System;  using System.Threading;  class Program  {  static int Count = 0;  static object lockObject = new object();  static void WorkThread()  {  for (int i = 0; i < 5000000; ++i)  {  lock (lockObject)  {  Count = Count + 1;  }  }  }  static void Main(string[] args)  {  Thread[] threads = new Thread[20];  for (int i = 0; i < 20; ++i)  {  threads[i] = new Thread(WorkThread);  threads[i].Start();  }  for (int i = 0; i < 20; ++i)  {  threads[i].Join();  }  Console.WriteLine($"Результат Count: {Count}");  Console.WriteLine($"Ожидаемое значение: {20 \* 5000000}");  }  } |

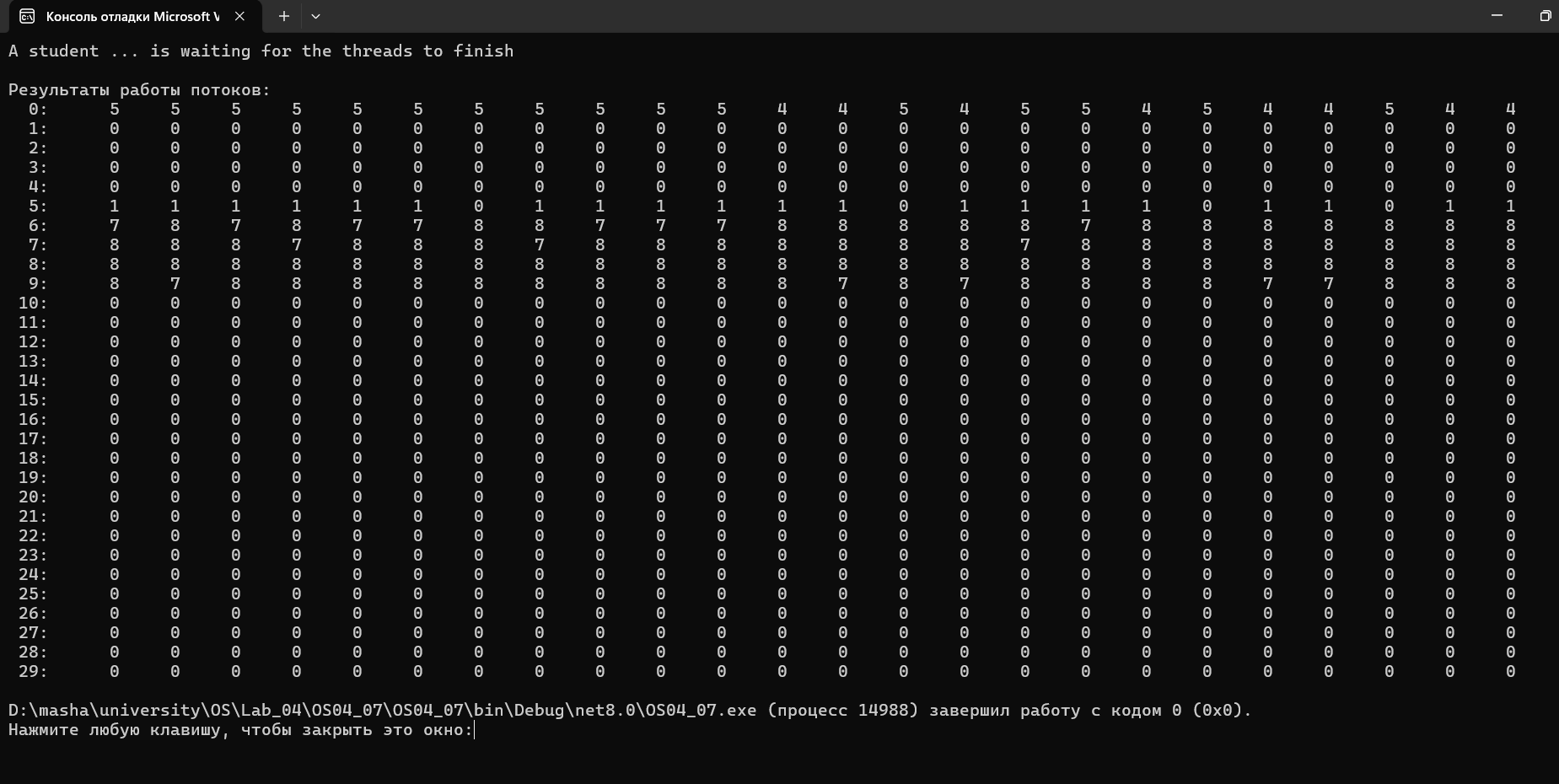


18. Сравните результаты в группе. Есть ли какая закономерность?

**Задание 07**

19. 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).

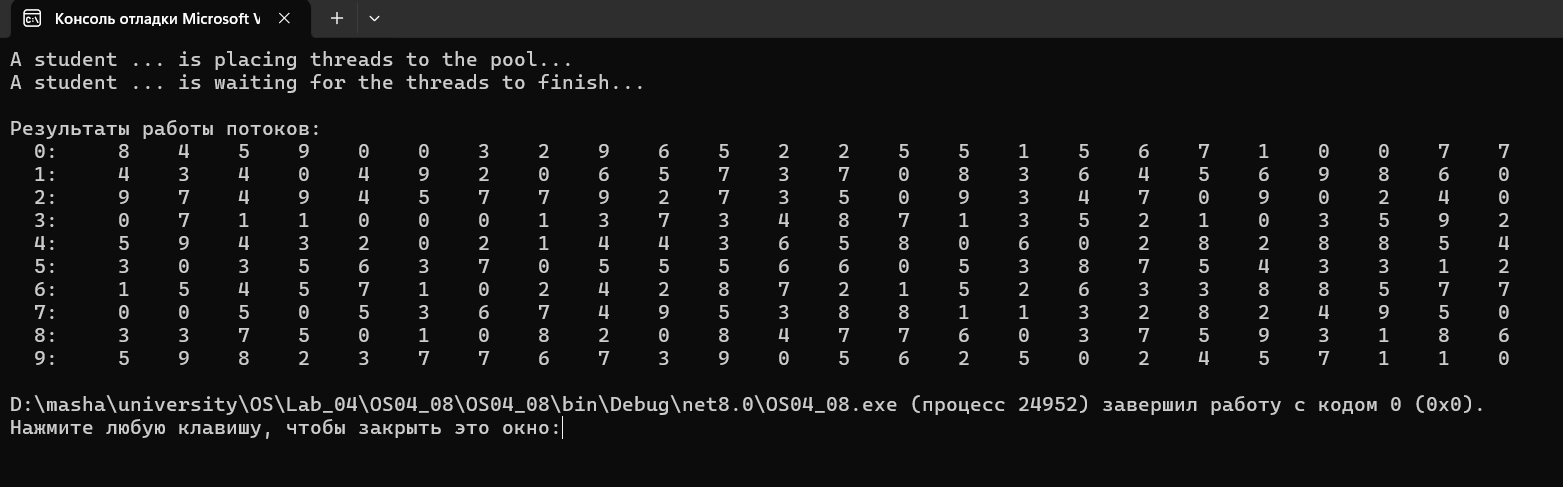
|  |
| --- |
| using System;  using System.Threading;  class Program  {  const int ThreadCount = 24;  const int ThreadLifeTime = 10;  const int ObservationTime = 30;  static int[,] Matrix = new int[ThreadCount, ObservationTime];  static DateTime StartTime = DateTime.Now;  static void MySleep(int ms)  {  double sum = 0;  for (int t = 0; t < ms; ++t)  {  double temp = 0.711 + (double)t / 10000.0;  double a, b, c, d, e, nt;  for (int k = 0; k < 5500; ++k)  {  nt = temp - k / 27000.0;  a = Math.Sin(nt);  b = Math.Cos(nt);  c = Math.Cos(nt / 2.0);  d = Math.Sin(nt / 2.0);  e = Math.Abs(1.0 - a \* a - b \* b) + Math.Abs(1.0 - c \* c - d \* d);  sum += e;  }  }  Thread.Sleep(ms);  }  static void WorkThread(object o)  {  int id = (int)o;  DateTime threadStartTime = DateTime.Now;  while ((DateTime.Now - threadStartTime).TotalSeconds < ThreadLifeTime)  {  int elapsedSeconds = (int)(DateTime.Now - StartTime).TotalSeconds;  if (elapsedSeconds >= ObservationTime || elapsedSeconds < 0)  break;  Matrix[id, elapsedSeconds] += 1;  MySleep(100);  }  }  static void Main(string[] args)  {  Thread[] t = new Thread[ThreadCount];  for (int i = 0; i < ThreadCount; ++i)  {  t[i] = new Thread(WorkThread);  t[i].Start(i);  }  Console.WriteLine("A student ... is waiting for the threads to finish");  for (int i = 0; i < ThreadCount; ++i)  {  t[i].Join();  }  Console.WriteLine("\nРезультаты работы потоков:");  for (int s = 0; s < ObservationTime; s++)  {  Console.Write("{0,3}: ", s);  for (int th = 0; th < ThreadCount; th++)  {  Console.Write(" {0,5}", Matrix[th, s]);  }  Console.WriteLine();  }  }  } |



**Задание 08**

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

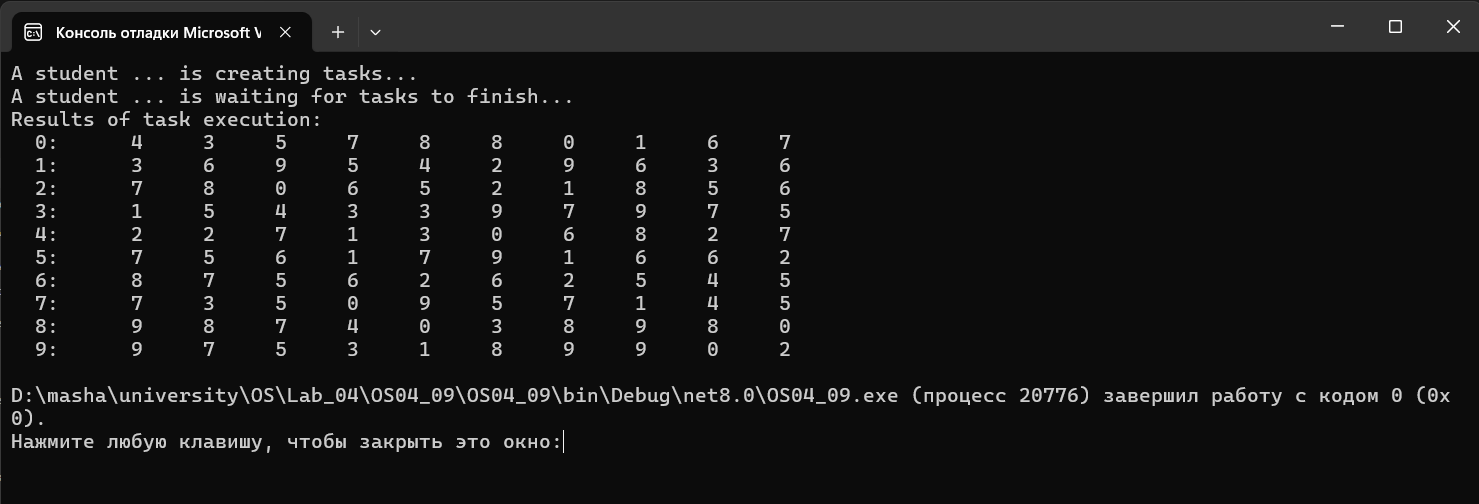
|  |
| --- |
| using System;  using System.Threading;  namespace OS04\_08  {  class Program  {  const int ThreadCount = 24;  const int ObservationTime = 10;  static int[,] Matrix = new int[ThreadCount, ObservationTime];  static void Main(string[] args)  {  Console.WriteLine("A student ... is placing threads to the pool...");  for (int i = 0; i < ThreadCount; ++i)  {  object o = i;  ThreadPool.QueueUserWorkItem(WorkThread, o);  }  Console.WriteLine("A student ... is waiting for the threads to finish...");  Thread.Sleep(1000 \* ObservationTime);  Console.WriteLine("\nРезультаты работы потоков:");  for (int s = 0; s < ObservationTime; s++)  {  Console.Write("{0,3}: ", s);  for (int th = 0; th < ThreadCount; th++)  {  Console.Write("{0,5}", Matrix[th, s]);  }  Console.WriteLine();  }  }  static void WorkThread(object state)  {  int threadIndex = (int)state;  Random random = new Random();  for (int s = 0; s < ObservationTime; s++)  {  Matrix[threadIndex, s] = random.Next(0, 10);  Thread.Sleep(random.Next(100));  }  }  }  } |



**Задание 09**

21. 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

|  |
| --- |
| using System;  using System.Threading.Tasks;  class Program  {  const int TaskCount = 10;  const int ObservationTime = 10;  static int[,] Matrix = new int[TaskCount, ObservationTime];  static void Main(string[] args)  {  Task[] tasks = new Task[TaskCount];  int[] numbers = new int[TaskCount];  for (int i = 0; i < TaskCount; i++)  numbers[i] = i;  Console.WriteLine("A student ... is creating tasks...");  for (int i = 0; i < TaskCount; i++)  {  int taskIndex = i;  tasks[i] = Task.Run(() => Work(taskIndex));  }  Console.WriteLine("A student ... is waiting for tasks to finish...");  Task.WaitAll(tasks);  Console.WriteLine("Results of task execution:");  for (int s = 0; s < ObservationTime; s++)  {  Console.Write("{0,3}: ", s);  for (int th = 0; th < TaskCount; th++)  Console.Write(" {0,5}", Matrix[th, s]);  Console.WriteLine();  }  }  static void Work(int index)  {  Random rand = new Random();  for (int s = 0; s < ObservationTime; s++)  {  Matrix[index, s] = rand.Next(0, 10);  Task.Delay(rand.Next(100, 500)).Wait();  }  }  } |



OS04\_07(потоки): тяжелее, так как каждый поток управляется вручную, время переключения контекста между потоками плохо сказывается на производительности.

OS04\_08 (пул потоков): эффективнее, так как повторно использует потоки, упрощение управления потоками.

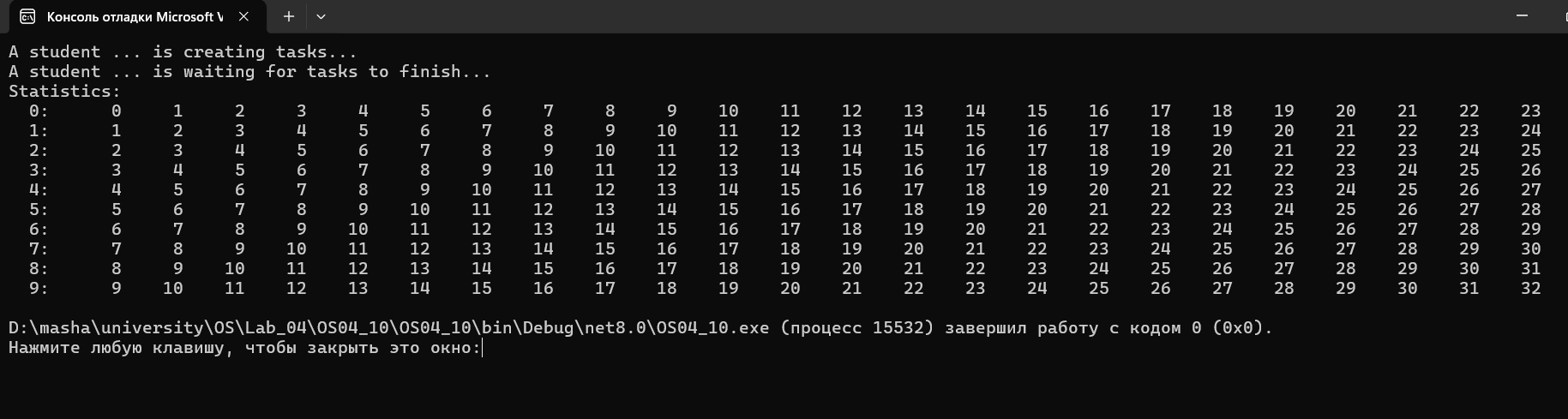
OS04\_09 (задачи): наилучший подход, который обеспечивает лучшее управление асинхронными операциями, лучше читаемость и обработка ошибок.

22. [For nerds only] Реализуйте запуск задач в цикле.

**Задание 10**

23. 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 < >.

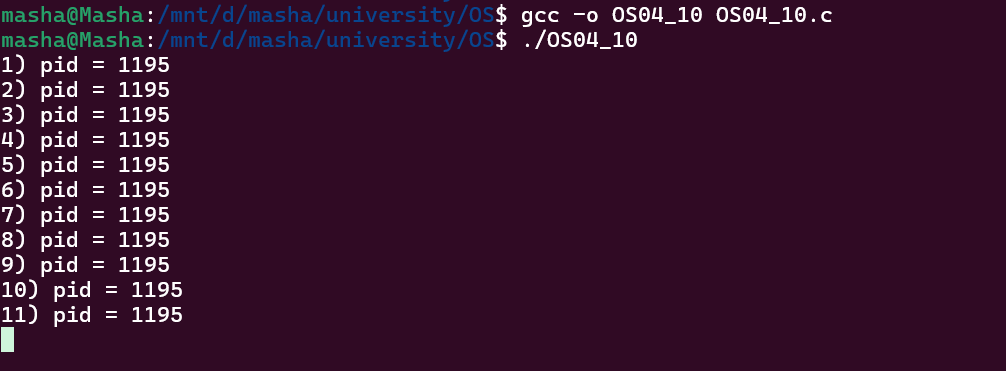
|  |
| --- |
| using System;  using System.Threading.Tasks;  class Program  {  const int ObservationTime = 10;  static int TaskCount = 24;  static int[,] Matrix = new int[TaskCount, ObservationTime];  static void Main(string[] args)  {  Task[] t = new Task[TaskCount];  int[] numbers = new int[TaskCount];  for (int i = 0; i < TaskCount; i++)  numbers[i] = i;  Console.WriteLine("A student ... is creating tasks...");  for (int i = 0; i < TaskCount; i++)  {  int index = i;  if (i == 0)  {  t[i] = new Task(() => Work(index));  t[i].Start();  }  else  {  t[i] = t[i - 1].ContinueWith(delegate { Work(index); });  }  }  Console.WriteLine("A student ... is waiting for tasks to finish...");  Task.WaitAll(t);  Console.WriteLine("Statistics:");  for (int s = 0; s < ObservationTime; s++)  {  Console.Write("{0,3}: ", s);  for (int th = 0; th < TaskCount; th++)  Console.Write(" {0,5}", Matrix[th, s]);  Console.WriteLine();  }  }  static void Work(int id)  {  for (int s = 0; s < ObservationTime; s++)  {  Matrix[id, s] = id + s;  System.Threading.Thread.Sleep(100);  }  }  } |



**Задание 11**

24. Разработайте на языке консольное Linux-приложение OS04\_10 на языке С, выполняющее цикл 100 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса (использовать функции из pthread.h).

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <unistd.h>  int main() {  for (int i = 0; i < 100; i++) {  pid\_t pid = getpid();  sleep(1);  printf("%d) pid = %d\n", i + 1, pid);  }  return 0;  } |



25. Процесс OS04\_10 должен создать поток: потоковая функция OS04\_10\_T1.

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <pthread.h>  #include <unistd.h>  void\* OS04\_10\_T1(void\* arg) {  for (int i = 0; i < 75; i++) {  printf("%d) child thread pid = %d\n", i + 1, getpid());  sleep(1);  }  return NULL;  }  int main() {  pthread\_t thread;  printf("main pid = %d\n", getpid());  if (pthread\_create(&thread, NULL, OS04\_10\_T1, NULL) != 0) {  fprintf(stderr, "Error creating thread\n");  return 1;  }  for (int i = 0; i < 100; i++) {  printf("%d) main thread pid = %d\n", i + 1, getpid());  sleep(1);  }  pthread\_join(thread, NULL);  return 0;  } |

