Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

Специальность 1-40 01 01 Программное обеспечение информационной безопасности мобильных систем

**Отчёт по лабораторной работе №4**

Дисциплина: Операционные системы

Тема: Потоки

Выполнила:

студентка 3 курса 5 группы

Городилина Анастасия Сергеевна

Минск 2024

**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.
2. int main()

{

DWORD pid = GetCurrentProcessId();

DWORD tid = GetCurrentThreadId();

for (int i = 0; i < 1000; i++)

{

cout << "PID = " << pid << ", TID = " << tid << endl;

Sleep(2000);

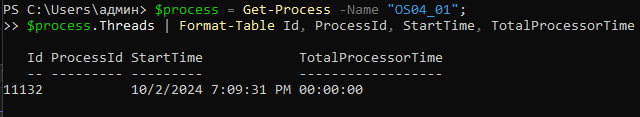
}

}

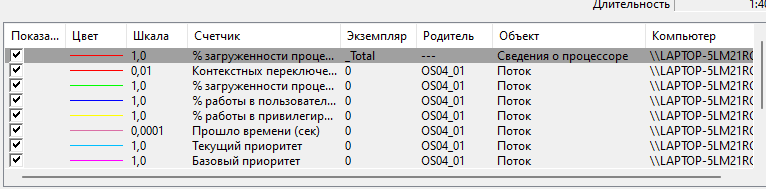
1. powershell ise

$process = Get-Process -Name "OS04\_01";

$process.Threads | Format-Table Id, ProcessId, StartTime, TotalProcessorTime;



Performance monitor



**Задание 02**

1. 3-6) Разработайте на языке консольное Windows-приложение **OS04\_02** на языке С++, выполняющее цикл 100 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.

DWORD WINAPI Thread1()

{

DWORD tid = GetCurrentThreadId();

for (short i = 0; i < 50; i++)

{

cout << "PID = " << pid << ", ID of thread1 = " << tid << endl;

Sleep(1000);

}

return 0;

}

DWORD WINAPI Thread2()

{

DWORD tid = GetCurrentThreadId();

for (short i = 0; i < 125; i++)

{

cout << "PID = " << pid << ", ID of thread2 = " << tid << endl;

Sleep(1000);

}

return 0;

}

int main()

{

pid = GetCurrentProcessId();

DWORD os04\_02ID = GetCurrentThreadId();

DWORD os04\_02\_T1ID = NULL;

DWORD os04\_02\_T2ID = NULL;

HANDLE os04\_02\_T1 = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)Thread1, NULL, 0, &os04\_02\_T1ID);

HANDLE os04\_02\_T2 = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)Thread2, NULL, 0, &os04\_02\_T2ID);

for (short i = 0; i < 100; ++i)

{

cout << "PID = " << pid << ", Thread main ID = " << os04\_02ID << endl;

Sleep(1000);

}

WaitForSingleObject(os04\_02\_T1, INFINITE);

WaitForSingleObject(os04\_02\_T2, INFINITE);

CloseHandle(os04\_02\_T1);

CloseHandle(os04\_02\_T2);

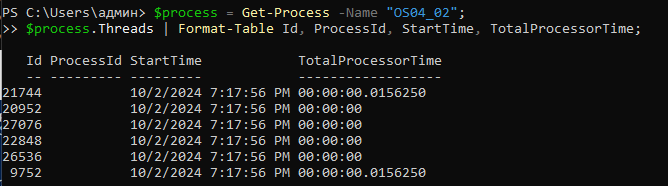
system("pause");

return 0;

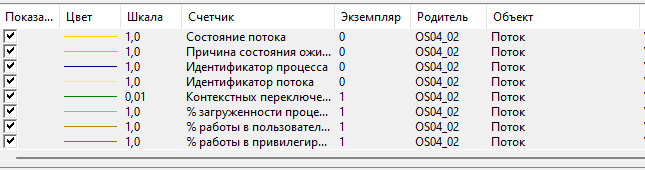
}

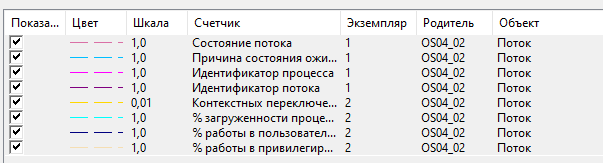


1. powershell ise



Performance monitor





**Задание 03**

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.

#include <Windows.h>

#include <iostream>

DWORD WINAPI threadProc(LPVOID);

int main()

{

HANDLE hThread;

DWORD dwThreadId;

for (int Count = 0; Count < 1000000; Count++)

{

hThread = CreateThread(

NULL, 0, threadProc, NULL, CREATE\_SUSPENDED, &dwThreadId);

if (hThread == INVALID\_HANDLE\_VALUE || hThread == NULL) {

printf("CreateThread failed (error %d) after %d threads\n",

GetLastError(), Count);

break;

}

ResumeThread(hThread);

CloseHandle(hThread);

if (Count % 1000 == 0)

printf("%d\n", Count);

}

printf("Thread Main exited\n");

return 0;

}

DWORD WINAPI threadProc(LPVOID args)

{

Sleep(600000);

return 0;

}

В ходе выполнения кода ноутбук отключился(180000-194000 потоков).

**Задание 04**

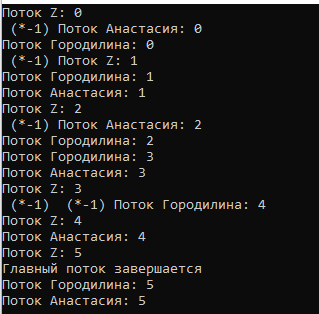
Создайте консольное приложение на C#, которое запускает

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 Имя Фамилию)



**Task 2**

using System;

using System.Threading;

class Program

{

static void ThreadZed()

{

for (int i = 0; i < 10; i++)

{

Console.WriteLine("Поток Z: " + i);

Thread.Sleep(1000);

}

Console.WriteLine("Поток Z завершается");

}

static void ThreadWithParam(object o)

{

for (int i = 0; i < 20; i++)

{

Console.WriteLine($"Поток {o}: {i}");

Thread.Sleep(1000);

}

Console.WriteLine($"Поток {o} завершается");

}

static void Main(string[] args)

{

Thread t1 = new Thread(ThreadZed);

Thread t2 = new Thread(ThreadWithParam);

Thread t3 = new Thread(ThreadWithParam);

t1.IsBackground = false; // false для п.11

t2.IsBackground = true; // false для п.12

t3.IsBackground = true;

t1.Start();

t2.Start("Настя"); // Имя

t3.Start("Лагун"); // 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");

//// Ожидание завершения всех потоков

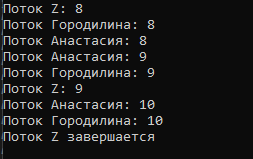
//t1.Join();

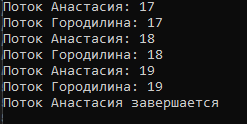
//t2.Join();

//t3.Join();

}

}



**Task 3**

**Задание 05**

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 void Main(string[] args)

{

// Пример использования MySleep для ожидания 10 секунд

Stopwatch stopwatch = Stopwatch.StartNew();

MySleep(10000);

stopwatch.Stop();

Console.WriteLine($"Функция MySleep завершилась за: {stopwatch.ElapsedMilliseconds} миллисекунд");

// Получение количества ядер и логических процессоров

int coreCount = Environment.ProcessorCount;

Console.WriteLine($"Количество логических процессоров: {coreCount}");

}

static double MySleep(int ms)

{

double sum = 0, temp;

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 < 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);

e = Math.Abs(1.0 - a \* a - b \* b) + Math.Abs(1.0 - c \* c - d \* d);

sum += e;

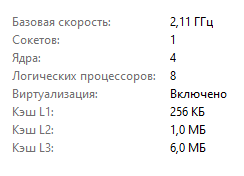
}

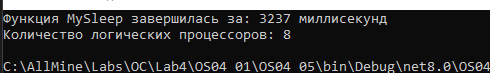
}

return sum;

}

}





**Задание 06**

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 void WorkThread()

{

for (int i = 0; i < 5000000; ++i)

{

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 переменной Count: {Count}");

Console.WriteLine($"Ожидаемый результат: {20 \* 5000000}");

}

}

****

**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).

Разработайте на языке консольное Linux-приложение **OS04\_06** на языке С, выполняющее длинный цикл с временной задержкой и с выводом на консоль идентификатора процесса.

class Program

{

// Настройки

const int ThreadCount = 10; // Количество потоков

const int ThreadLifeTime = 10; // Время жизни потока в секундах

const int ObservationTime = 30; // Время наблюдения в секундах

static int[,] Matrix = new int[ThreadCount, ObservationTime];

static DateTime StartTime = DateTime.Now;

static void Main(string[] args)

{

Thread[] threads = new Thread[ThreadCount];

// Запуск потоков

for (int i = 0; i < ThreadCount; ++i)

{

int threadId = i; // локальная копия переменной

threads[i] = new Thread(WorkThread);

threads[i].Start(threadId);

}

Console.WriteLine("Ожидание завершения потоков...");

// Ожидание завершения потоков

for (int i = 0; i < ThreadCount; ++i)

{

threads[i].Join();

}

// Вывод результата в виде таблицы

Console.WriteLine("Время (с) | Поток 0 | Поток 1 | Поток 2 | Поток 3 | Поток 4 | Поток 5 | Поток 6 | Поток 7 | Поток 8 | Поток 9 |");

Console.WriteLine("-------------------------------------------------------------");

for (int s = 0; s < ObservationTime; s++)

{

Console.Write($"{s,3}: |");

for (int th = 0; th < ThreadCount; th++)

{

Console.Write($" {Matrix[th, s],5} |");

}

Console.WriteLine();

}

}

static void WorkThread(object o)

{

int id = (int)o;

DateTime threadStartTime = DateTime.Now;

while ((DateTime.Now - threadStartTime).TotalSeconds < ThreadLifeTime)

{

DateTime currentTime = DateTime.Now;

int elapsedSeconds = (int)(currentTime - StartTime).TotalSeconds;

if (elapsedSeconds < ObservationTime)

{

Matrix[id, elapsedSeconds] += 50; // Увеличиваем значение в матрице

}

MySleep(50); // Вызов метода, который занимает время

}

}

static Double MySleep(int ms)

{

Double sum = 0, temp;

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 < 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);

e = Math.Abs(1.0 - a \* a - b \* b) + Math.Abs(1.0 - c \* c - d \* d);

sum += e;

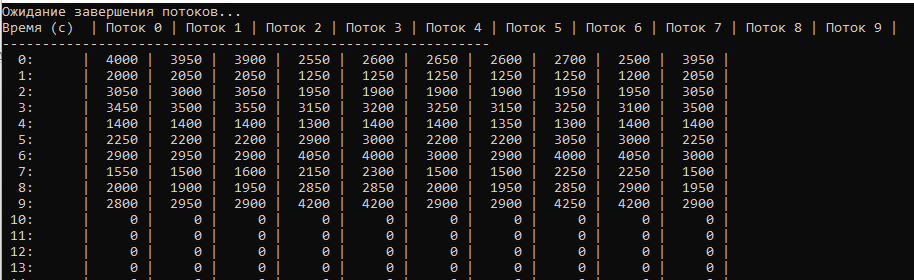
}

}

return sum;

}

}

****

**Задание 8**

1. 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;

class Program

{

// Настройки

const int ThreadCount = 10; // Количество потоков

const int ThreadLifeTime = 10; // Время жизни потока в секундах

const int ObservationTime = 30; // Время наблюдения в секундах

static int[,] Matrix = new int[ThreadCount, ObservationTime];

static DateTime StartTime = DateTime.Now;

static void Main(string[] args)

{

Console.WriteLine("Студент Городилина помещает потоки в пул...");

// Запуск потоков из пула

for (int i = 0; i < ThreadCount; ++i)

{

int threadId = i; // локальная копия переменной

ThreadPool.QueueUserWorkItem(WorkThread, threadId);

}

Console.WriteLine("Студент Городилина ожидает завершения потоков...");

Thread.Sleep(1000 \* ObservationTime); // Ожидание завершения потоков

// Вывод результата в виде таблицы

Console.WriteLine("Время (с) | Поток 0 | Поток 1 | Поток 2 | Поток 3 | Поток 4 | Поток 5 | Поток 6 | Поток 7 | Поток 8 | Поток 9 |");

Console.WriteLine("-------------------------------------------------------------");

for (int s = 0; s < ObservationTime; s++)

{

Console.Write($"{s,3}: |");

for (int th = 0; th < ThreadCount; th++)

{

Console.Write($" {Matrix[th, s],5} |");

}

Console.WriteLine();

}

}

static void WorkThread(object o)

{

int id = (int)o;

DateTime threadStartTime = DateTime.Now;

while ((DateTime.Now - threadStartTime).TotalSeconds < ThreadLifeTime)

{

DateTime currentTime = DateTime.Now;

int elapsedSeconds = (int)(currentTime - StartTime).TotalSeconds;

if (elapsedSeconds < ObservationTime)

{

Matrix[id, elapsedSeconds] += 50; // Увеличиваем значение в матрице

}

MySleep(50); // Вызов метода, который занимает время

}

}

static Double MySleep(int ms)

{

Double sum = 0, temp;

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 < 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);

e = Math.Abs(1.0 - a \* a - b \* b) + Math.Abs(1.0 - c \* c - d \* d);

sum += e;

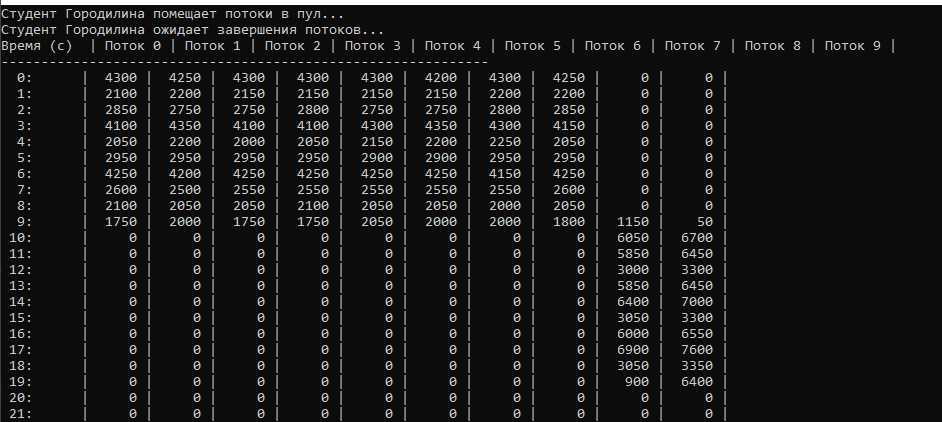
}

}

return sum;

}

}



**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.

class Program

{

// Настройки

const int TaskCount = 10; // Количество задач

const int TaskLifeTime = 10; // Время жизни задачи в секундах

const int ObservationTime = 30; // Время наблюдения в секундах

static int[,] Matrix = new int[TaskCount, ObservationTime];

static DateTime StartTime = DateTime.Now;

static void Main(string[] args)

{

Console.WriteLine("Настя создает задачи...");

Task[] tasks = new Task[TaskCount];

// Запуск задач

for (int i = 0; i < TaskCount; i++)

{

int taskId = i; // локальная копия переменной

tasks[i] = Task.Run(() => Work(taskId));

}

Console.WriteLine("Настя ожидает завершения задач...");

Task.WaitAll(tasks); // Ожидание завершения всех задач

// Вывод результата в виде таблицы

Console.WriteLine("Время (с) | Задача 0 | Задача 1 | Задача 2 | Задача 3 | Задача 4 | Задача 5 | Задача 6 | Задача 7 | Задача 8 | Задача 9 |");

Console.WriteLine("-------------------------------------------------------------");

for (int s = 0; s < ObservationTime; s++)

{

Console.Write($"{s,3}: |");

for (int th = 0; th < TaskCount; th++)

{

Console.Write($" {Matrix[th, s],5} |");

}

Console.WriteLine();

}

}

static void Work(int id)

{

DateTime taskStartTime = DateTime.Now;

while ((DateTime.Now - taskStartTime).TotalSeconds < TaskLifeTime)

{

DateTime currentTime = DateTime.Now;

int elapsedSeconds = (int)(currentTime - StartTime).TotalSeconds;

if (elapsedSeconds < ObservationTime)

{

Matrix[id, elapsedSeconds] += 50; // Увеличиваем значение в матрице

}

MySleep(50); // Вызов метода, который занимает время

}

}

static Double MySleep(int ms)

{

Double sum = 0, temp;

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 < 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);

e = Math.Abs(1.0 - a \* a - b \* b) + Math.Abs(1.0 - c \* c - d \* d);

sum += e;

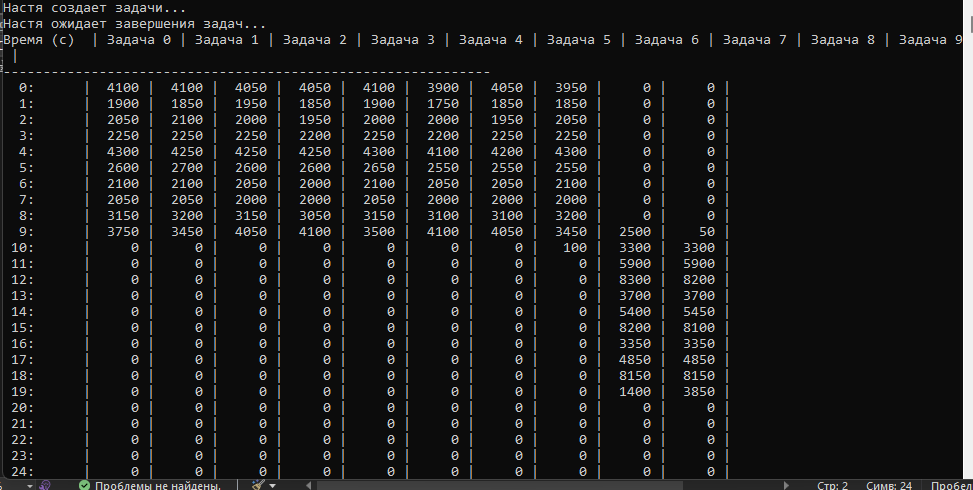
}

}

return sum;

}

}



OS04\_07: Каждый поток создается вручную с помощью класса Thread. Можно самостоятельно управлять жизненным циклом потоков, включая их создание и завершение.

OS04\_08: Пул потоков автоматически управляет созданием и уничтожением потоков, что снижает накладные расходы на создание новых потоков, так как переиспользует уже существующие.

OS04\_09: Использует задачи, что упрощает управление потоками и улучшает читаемость кода. Результаты могут быть более стабильными, так как Task управляет потоками более эффективно и безопасно по сравнению с ручным управлением потоками

**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

class Program

{

// Настройки

static readonly int TaskCount = Environment.ProcessorCount; // Количество задач = количеству логических процессоров

const int TaskLifeTime = 10; // Время жизни задачи в секундах

const int ObservationTime = 30; // Время наблюдения в секундах

static int[,] Matrix = new int[TaskCount, ObservationTime];

static DateTime StartTime = DateTime.Now;

static void Main(string[] args)

{

Task[] t = new Task[TaskCount];

int[] numbers = new int[TaskCount];

// Инициализация массива чисел

for (int i = 0; i < TaskCount; i++)

numbers[i] = i;

Console.WriteLine("Настя Городилина создает задачи...");

// Создание и запуск задач по очереди с использованием ContinueWith

t[0] = new Task(() => { Work(0); });

t[0].Start(); // Запускаем первую задачу

for (int i = 1; i < TaskCount; i++)

{

int taskId = numbers[i];

t[i] = t[i - 1].ContinueWith(\_ => Work(taskId));

}

Console.WriteLine("Настя Городилина ожидает завершения задач...");

Task.WaitAll(t); // Ожидание завершения всех задач

// Вывод результата в виде таблицы

Console.WriteLine("Время (с) | Задача 0 | Задача 1 | Задача 2 | Задача 3 |");

Console.WriteLine("-------------------------------------------------------------");

for (int s = 0; s < ObservationTime; s++)

{

Console.Write($"{s,3}: |");

for (int th = 0; th < TaskCount; th++)

{

Console.Write($" {Matrix[th, s],5} |");

}

Console.WriteLine();

}

}

static void Work(int id)

{

DateTime taskStartTime = DateTime.Now;

while ((DateTime.Now - taskStartTime).TotalSeconds < TaskLifeTime)

{

DateTime currentTime = DateTime.Now;

int elapsedSeconds = (int)(currentTime - StartTime).TotalSeconds;

if (elapsedSeconds < ObservationTime)

{

Matrix[id, elapsedSeconds] += 50; // Увеличиваем значение в матрице

}

MySleep(50); // Вызов метода, который занимает время

}

}

static Double MySleep(int ms)

{

Double sum = 0, temp;

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 < 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);

e = Math.Abs(1.0 - a \* a - b \* b) + Math.Abs(1.0 - c \* c - d \* d);

sum += e;

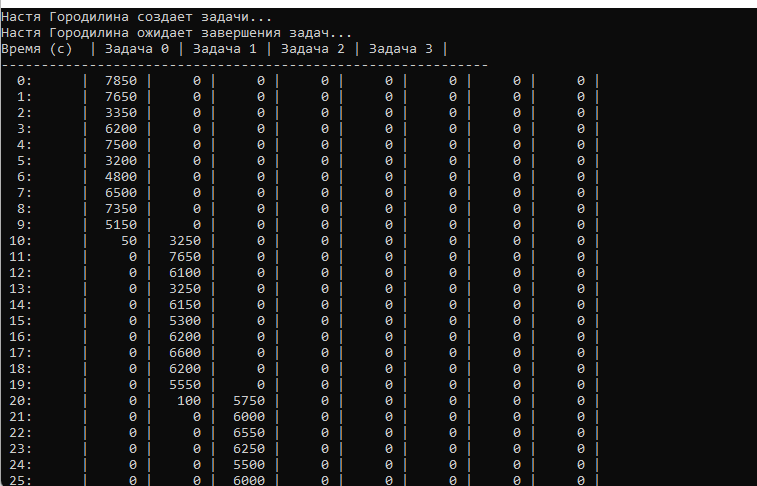
}

}

return sum;

}

}



**Задание 11**

Разработайте на языке консольное Linux-приложение **OS04\_07** на языке С, выполняющее цикл 100 итераций с временной задержкой в 1 сек. с выводом на консоль идентификатора процесса.

void\* Thread(void\* arg)

{

pid\_t pid = getpid();

pid\_t tid = syscall(SYS\_gettid);

for (int i = 1; i <= 75; ++i)

{

printf("%d. PID = %d, ID of thread = %d\n", i, pid, tid);

sleep(1);

}

pthread\_exit("Child thread");

}

int main()

{

pid\_t pid = getpid();

pid\_t tid = syscall(SYS\_gettid);

pthread\_t a\_thread;

void\* thread\_result;

int res = pthread\_create(&a\_thread, NULL, Thread, NULL);

for (int i = 1; i <= 100; ++i)

{

printf("%d. PID = %d, Thread main ID = %d\n", i, pid, tid);

sleep(1);

}

int status = pthread\_join(a\_thread, (void\*\*)&thread\_result);

exit(0);

}

