МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Направление специальности 1-40 01 01 10 Программное обеспечение информационных технологий (программирование интернет-приложений)

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ:

по дисциплине «Операционные системы»

Исполнитель

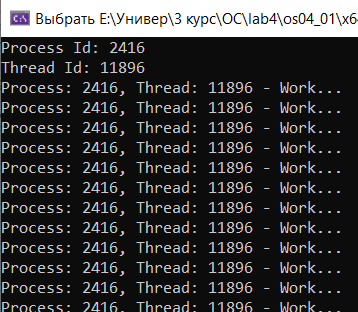
студент (ка) 3 курса группы 6 Розель Станислав Александрович

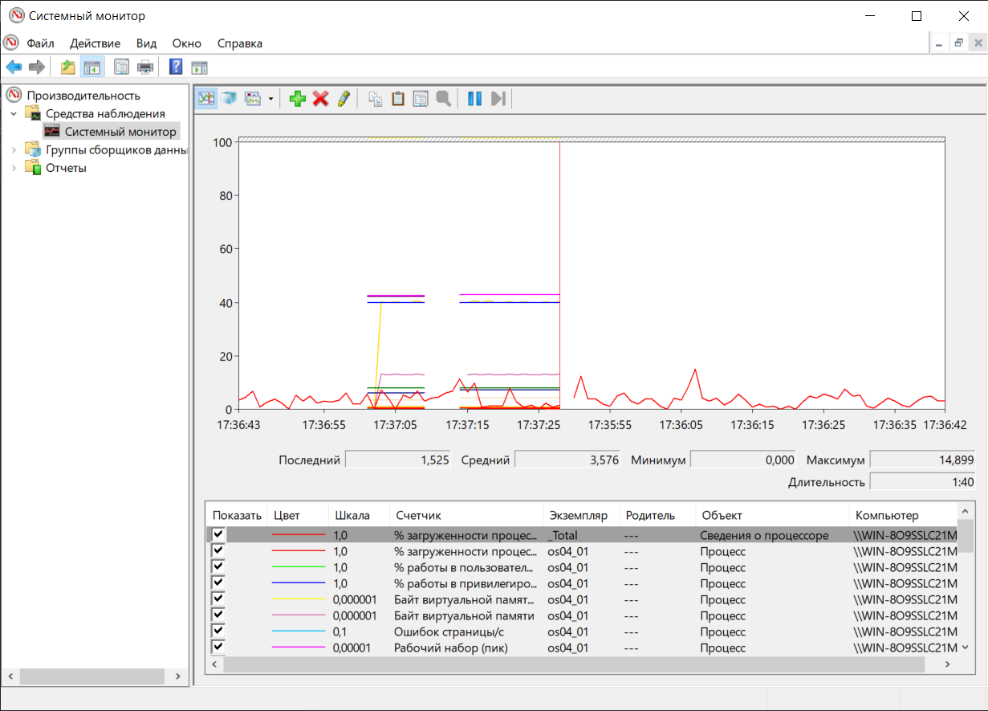
(Ф.И.О.)

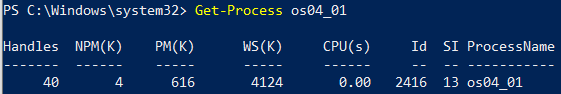
Минск 2024

Задание 1

|  |
| --- |
| #include <iostream>  #include <thread>  #include <chrono>  #include <windows.h>  int main()  {  DWORD processId = GetCurrentProcessId();  DWORD threadId = GetCurrentThreadId();  std::cout << "Process Id: " << processId << std::endl;  std::cout << "Thread Id: " << threadId << std::endl;  for (int i = 0; i < 1000; i++) {  std::cout << "Process: " << processId << ", Thread: " << threadId << " - Work..." << std::endl;  std::this\_thread::sleep\_for(std::chrono::seconds(1));  }  return 0;  } |

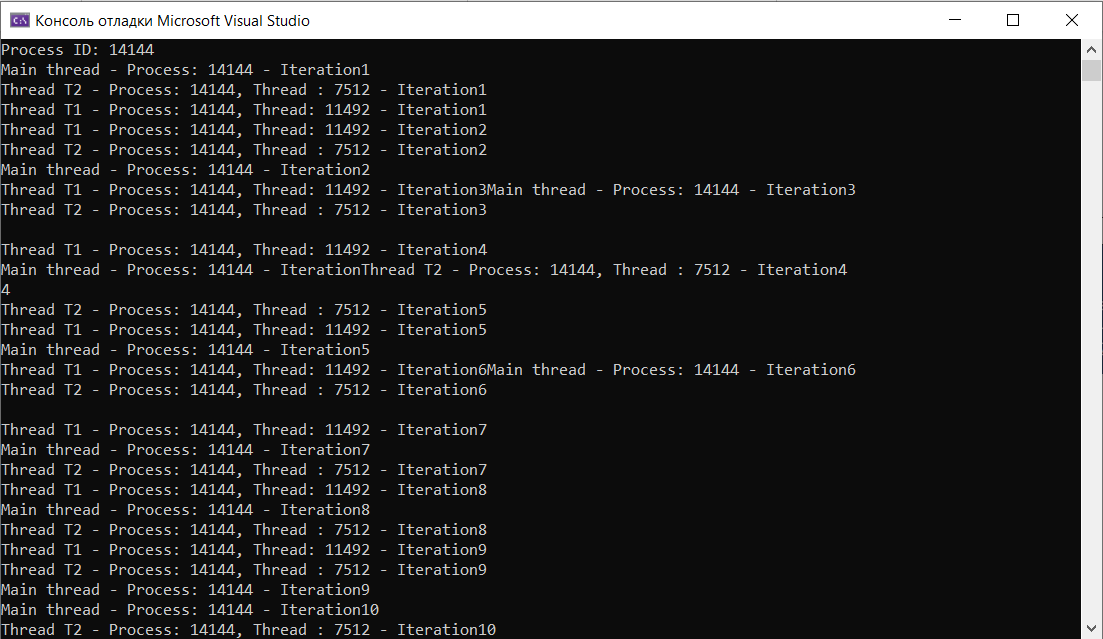


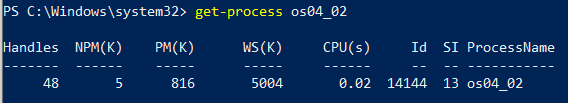




Задание 2

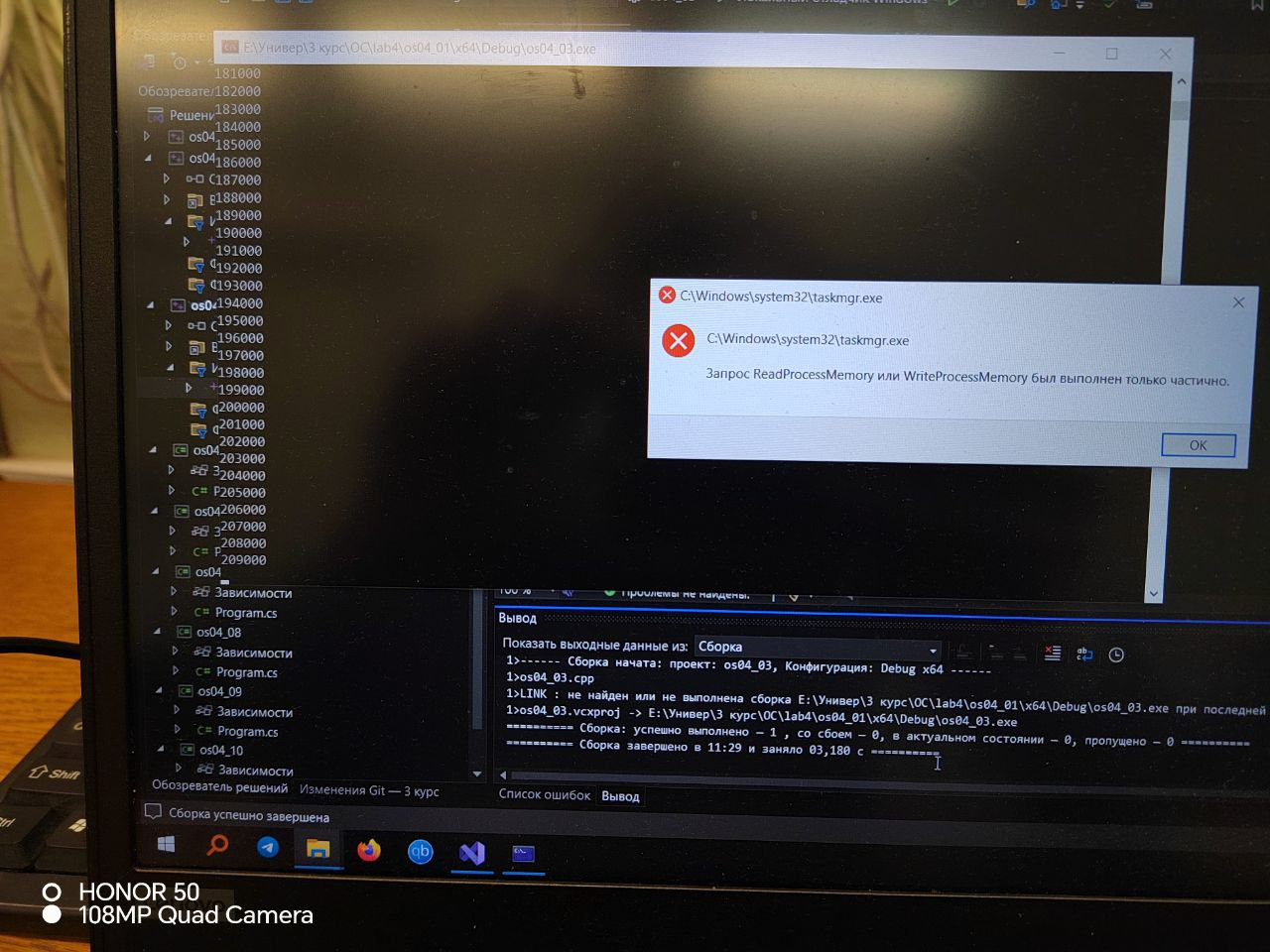
|  |
| --- |
| #include <iostream>  #include <thread>  #include <chrono>  #include <windows.h>  void OS04\_02\_T1(DWORD processId) {  for (int i = 0; i < 50; ++i) {  DWORD threadId = GetCurrentThreadId();  std::cout << "Thread T1 - Process: " << processId << ", Thread: " << threadId << " - Iteration" << i + 1 << std::endl;  std::this\_thread::sleep\_for(std::chrono::seconds(1));  }  }  void OS04\_02\_T2(DWORD processId) {  for (int i = 0; i < 125; ++i) {  DWORD threadId = GetCurrentThreadId();  std::cout << "Thread T2 - Process: " << processId << ", Thread : " << threadId << " - Iteration" << i + 1 << std::endl;  std::this\_thread::sleep\_for(std::chrono::seconds(1));  }  }  int main()  {  DWORD processId = GetCurrentProcessId();  std::cout << "Process ID: " << processId << std::endl;  std::thread thread1(OS04\_02\_T1, processId);  std::thread thread2(OS04\_02\_T2, processId);  for (int i = 0; i < 100; ++i) {  std::cout << "Main thread - Process: " << processId << " - Iteration" << i + 1 << std::endl;  std::this\_thread::sleep\_for(std::chrono::seconds(1));  }  thread1.join();  thread2.join();  return 0;  } |





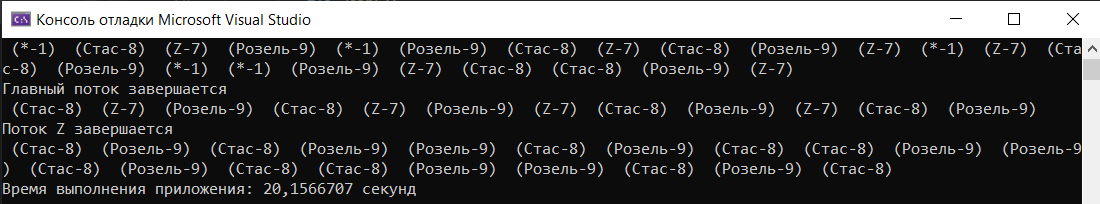
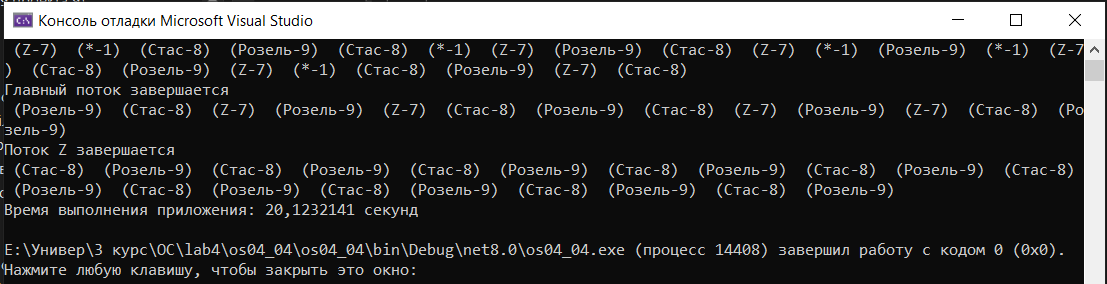
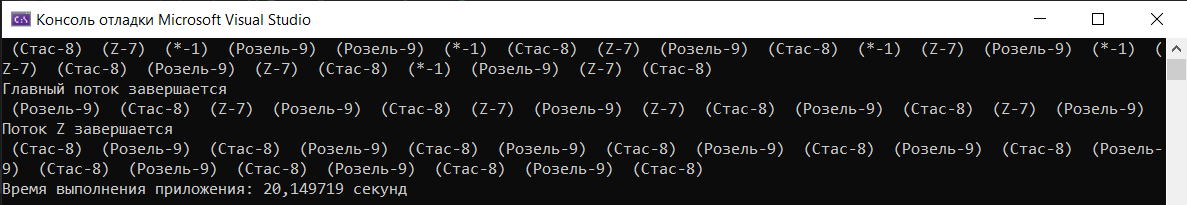
Задание 3

|  |
| --- |
| #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;  } |



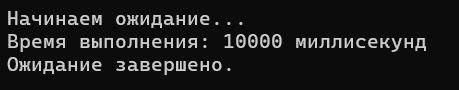
Задание 4

|  |
| --- |
| using System.Diagnostics;  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("\nПоток 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)  {  Stopwatch stopwatch = new Stopwatch(); // Создаем объект Stopwatch  stopwatch.Start(); // Запускаем таймер  var t1 = new Thread(ThreadZed);  var t1a = new Thread(ThreadWithParam);  var t1b = new Thread(ThreadWithParam);  t1.IsBackground = true; // false для 11 п.п  t1a.IsBackground = true; // false для 12 п.п  t1b.IsBackground = true; // Фоновый поток  t1.Start();  t1a.Start("Стас"); // Имя  t1b.Start("Розель"); // Фамилия  // Главный поток работает 5 секунд  for (int i = 0; i < 5; i++)  {  Console.Write(" (\*-{0}) ", Thread.CurrentThread.ManagedThreadId);  Thread.Sleep(1000);  }  Console.WriteLine("\nГлавный поток завершается");  // Ждем завершения не фонового потока  t1a.Join();  stopwatch.Stop(); // Останавливаем таймер  Console.WriteLine("\nВремя выполнения приложения: {0} секунд", stopwatch.Elapsed.TotalSeconds);  }  } |

Задание 5

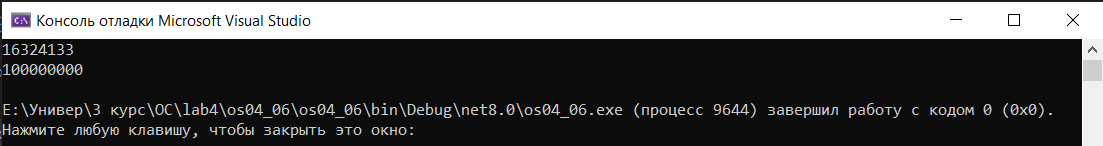
|  |
| --- |
| static Double MySleep(int ms)  {  Double Sum = 0, Temp;  Stopwatch stopwatch = Stopwatch.StartNew();  while (stopwatch.ElapsedMilliseconds < ms)  {  Temp = 0.711 + (Double)(stopwatch.ElapsedMilliseconds) / 10000.0;  Double a, b, c, d, e, nt;  for (int k = 0; k < 5500; ++k)  {  nt = Temp - 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;  } |





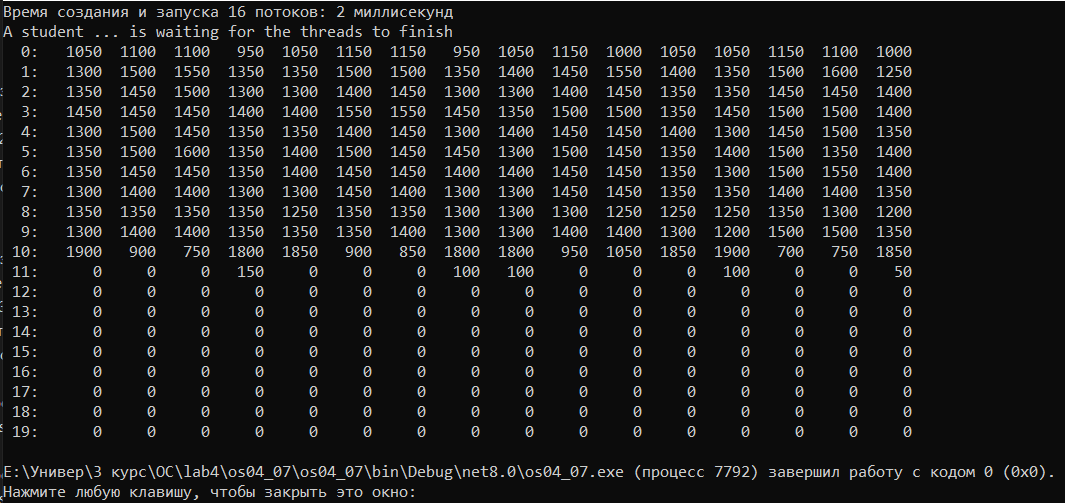
Задание 6

|  |
| --- |
| using System;  using System.Diagnostics;  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[] t = new Thread[20];  for (int i = 0; i < 20; ++i) {  t[i] = new Thread(WorkThread);  t[i].Start();  }  for (int i = 0; i < 20; ++i)  t[i].Join();  Console.WriteLine(Count);  Console.WriteLine(20 \* 5000000);  }    } |

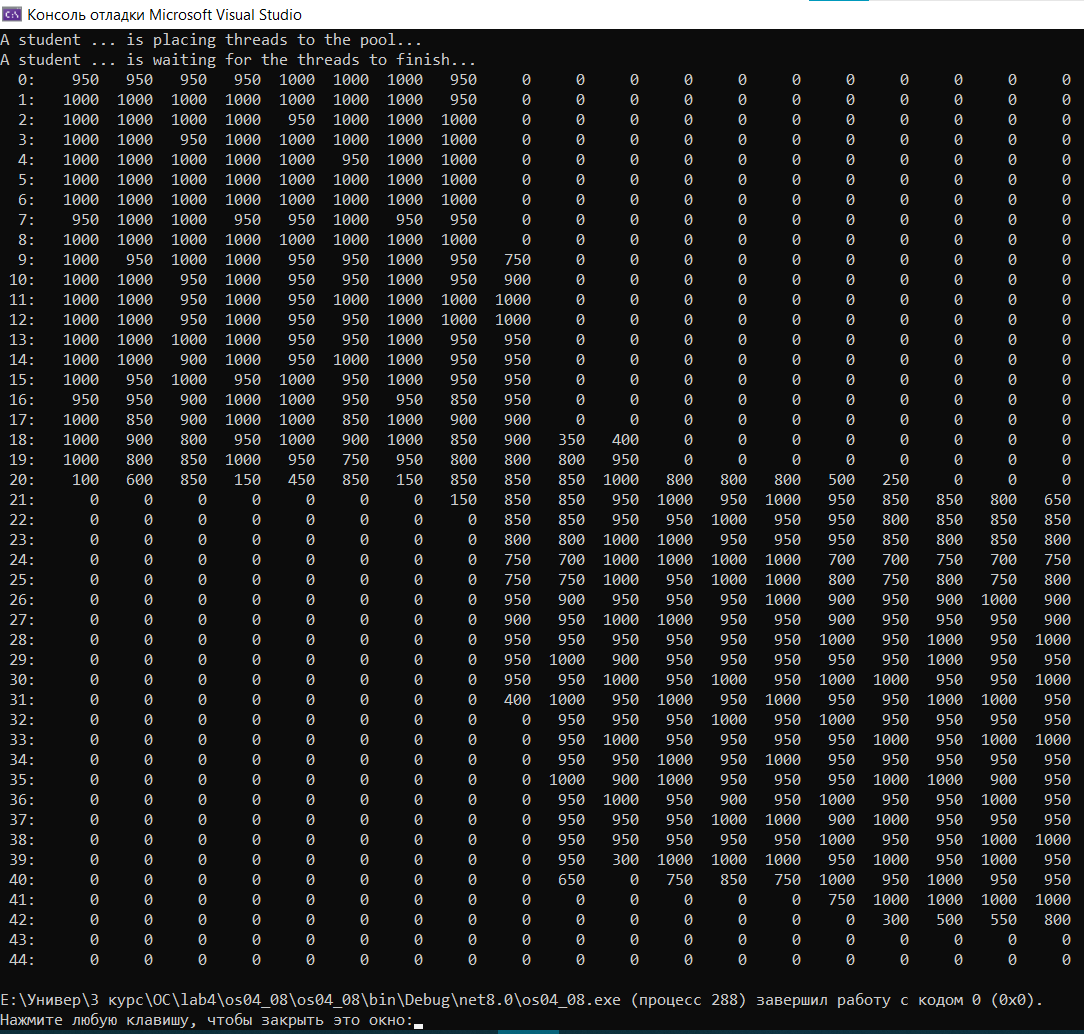


Задание 7

|  |
| --- |
| using System.Diagnostics;  class Program  {  const int ThreadCount = 20;  const int ThreadLifeTime = 20;  const int ObservationTime = 60;  static int[,] Matrix = new int[ThreadCount, ObservationTime];  static DateTime StartTime = DateTime.Now;  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;  }  static void WorkThread(object o)  {  int id = (int)o;  for (int i = 0; i < ThreadLifeTime \* 20; i++)  {  DateTime CurrentTime = DateTime.Now;  int ElapsedSeconds = (int)Math.Round(CurrentTime.Subtract(StartTime).TotalSeconds - 0.49);  if (ElapsedSeconds >= ObservationTime)  {  ElapsedSeconds = ObservationTime - 1;  }  Matrix[id, ElapsedSeconds] += 50;  MySleep(50); // из задания 5  }  }  static void Main(string[] args){  Thread[] t = new Thread[ThreadCount];  for (int i = 0; i < ThreadCount; ++i){  object o = i;  t[i] = new Thread(WorkThread);  t[i].Start(o);  }  Console.WriteLine("A student ... is waiting for the threads to finish");  for (int i = 0; i < ThreadCount; ++i)  t[i].Join();  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 8

|  |
| --- |
| using System.Diagnostics;  class Program  {  const int ThreadCount = 20;  const int ThreadLifeTime = 20;  const int ObservationTime = 60;  static int[,] Matrix = new int[ThreadCount, ObservationTime];  static DateTime StartTime = DateTime.Now;  private static object matrixLock = new object();  static void WorkThread(object o)  {  int id = (int)o;  for (int i = 0; i < ThreadLifeTime \* 20; i++)  {  DateTime CurrentTime = DateTime.Now;  int ElapsedSeconds = (int)Math.Round(CurrentTime.Subtract(StartTime).TotalSeconds - 0.49);  if (ElapsedSeconds >= 0 && ElapsedSeconds < ObservationTime)  {  lock (matrixLock)  {  Matrix[id, ElapsedSeconds] += 50;  }  }  MySleep(50);  }  }  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);  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 Double MySleep(int ms)  {  Double Sum = 0, Temp;  Stopwatch stopwatch = Stopwatch.StartNew();  while (stopwatch.ElapsedMilliseconds < ms)  {  Temp = 0.711 + (Double)(stopwatch.ElapsedMilliseconds) / 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;  }  } |

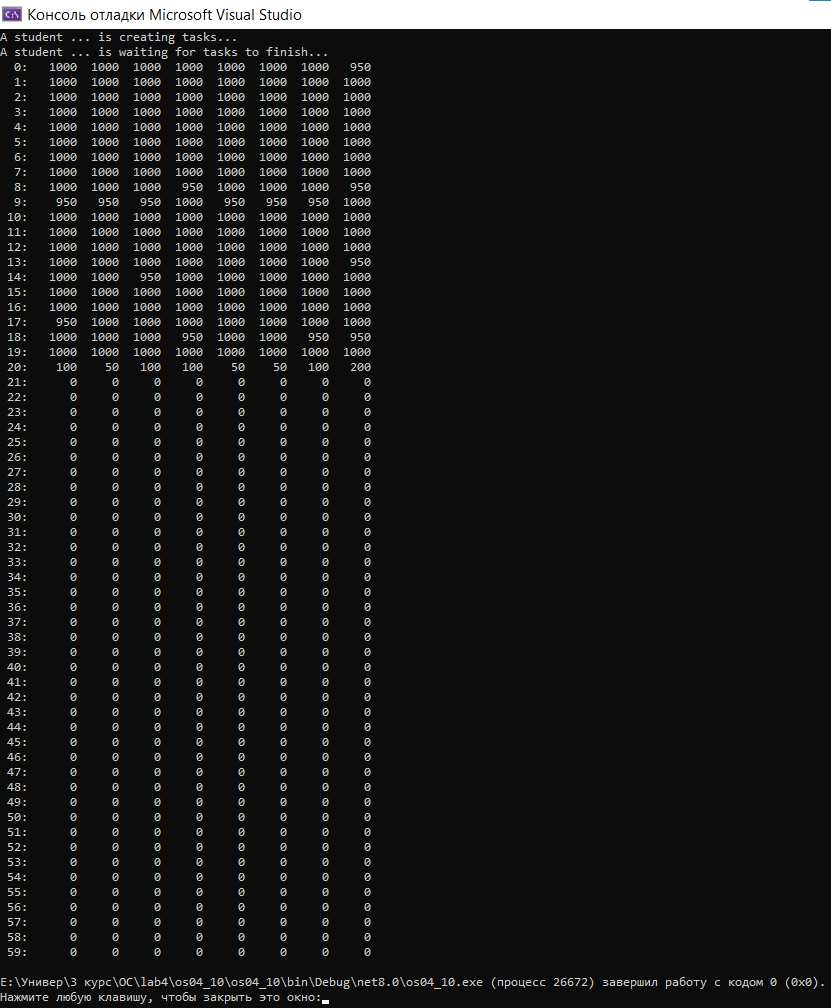


Задание 9

|  |
| --- |
| using System;  using System.Diagnostics;  using System.Threading.Tasks;  class Program  {  const int ThreadCount = 20;  const int ThreadLifeTime = 20; // Время жизни потоков в секундах  const int ObservationTime = 60; // Общее время наблюдения в секундах  static int[,] Matrix = new int[ThreadCount, ObservationTime];  static DateTime StartTime = DateTime.Now;  private static object matrixLock = new object();  static void WorkThread(object o)  {  int id = (int)o;  for (int i = 0; i < ThreadLifeTime \* 20; i++)  {  DateTime CurrentTime = DateTime.Now;  int ElapsedSeconds = (int)Math.Round(CurrentTime.Subtract(StartTime).TotalSeconds - 0.49);  if (ElapsedSeconds >= 0 && ElapsedSeconds < ObservationTime){  lock (matrixLock)  { Matrix[id, ElapsedSeconds] += 50;}  }  MySleep(50);  }  }  static void Main(string[] args){  Task[] tasks = new Task[ThreadCount];  for (int i = 0; i < ThreadCount; i++){  int threadId = i;  tasks[i] = Task.Run(() => WorkThread(threadId));  }  Console.WriteLine("A student ... is waiting for tasks to finish...");  Task.WaitAll(tasks);  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 Double MySleep(int ms){  Double Sum = 0, Temp;  Stopwatch stopwatch = Stopwatch.StartNew();  while (stopwatch.ElapsedMilliseconds < ms){  Temp = 0.711 + (Double)(stopwatch.ElapsedMilliseconds) / 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 10

|  |
| --- |
| using System;  using System.Diagnostics;  using System.Threading.Tasks;  class Program{  const int ThreadCount = 8;  const int ThreadLifeTime = 20; // Время жизни потоков в секундах  const int ObservationTime = 60; // Общее время наблюдения в секундах  static int[,] Matrix = new int[ThreadCount, ObservationTime];  static DateTime StartTime = DateTime.Now;  private static object matrixLock = new object();  static void WorkThread(object o){  int id = (int)o;  for (int i = 0; i < ThreadLifeTime \* 20; i++){  DateTime CurrentTime = DateTime.Now;  int ElapsedSeconds = (int)Math.Round(CurrentTime.Subtract(StartTime).TotalSeconds - 0.49);  if (ElapsedSeconds >= 0 && ElapsedSeconds < ObservationTime){  lock (matrixLock){ Matrix[id, ElapsedSeconds] += 50;}  }  MySleep(50);  }  }  static void Main(string[] args){  Task[] tasks = new Task[ThreadCount];  Console.WriteLine("A student ... is creating tasks...");  for (int i = 0; i < ThreadCount; i++){  int threadId = i;  tasks[i] = Task.Run(() => WorkThread(threadId));  }  Console.WriteLine("A student ... is waiting for tasks to finish...");  Task.WaitAll(tasks);  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 Double MySleep(int ms){  Double Sum = 0, Temp;  Stopwatch stopwatch = Stopwatch.StartNew();  while (stopwatch.ElapsedMilliseconds < ms){  Temp = 0.711 + (Double)(stopwatch.ElapsedMilliseconds) / 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

|  |
| --- |
| #include <stdio.h>  #include <pthread.h>  #include <unistd.h> // Для функции sleep()  #include <sys/types.h> // Для получения идентификатора процесса (PID)  // Потоковая функция OS04\_10\_T1  void\* OS04\_10\_T1(void\* arg) {      pid\_t tid = getpid(); // Получение идентификатора процесса      for (int i = 0; i < 75; i++) {          printf("Thread OS04\_10\_T1: Process ID = %d, iteration = %d\n", tid, i+1);          sleep(1); // Задержка 1 секунда      }      return NULL;  }  int main() {      pthread\_t thread; // Идентификатор потока      pid\_t pid = getpid(); // Получение идентификатора процесса      // Создание потока      if (pthread\_create(&thread, NULL, OS04\_10\_T1, NULL) != 0) {          perror("Error creating thread");          return 1;      }      // Основной процесс выполняет 100 итераций      for (int i = 0; i < 100; i++) {          printf("Process OS04\_10: Process ID = %d, iteration = %d\n", pid, i+1);          sleep(1); // Задержка 1 секунда      }      // Ожидание завершения потока      pthread\_join(thread, NULL);      return 0;  } |

