**14. Многопоточные приложения**

Задание 1. Создать консольное приложение, в котором реализовано 3 потока. Первый выводит числа от 0 до 9, второй – от 10 до 19, третий – от 20 до 29. Вывод значений на экран происходит следующим образом: сначала своё число выводит первый поток, а затем второй и третий. Организовать возможность использования методов Start и Sleep, а также изменение свойства Priority, позволяющее изменить приоритет потока. Можно воспользоваться инструментами синхронизации потоков lock, AutoResetEvent и т.д.

namespace task1

{

using System;

using System.Threading;

class Program

{

static AutoResetEvent firstDone = new AutoResetEvent(false);

static AutoResetEvent secondDone = new AutoResetEvent(false);

static void Main(string[] args)

{

Thread t1 = new Thread(FirstThread);

Thread t2 = new Thread(SecondThread);

Thread t3 = new Thread(ThirdThread);

t1.Priority = ThreadPriority.Highest;

t2.Priority = ThreadPriority.Normal;

t3.Priority = ThreadPriority.Lowest;

t1.Start();

t2.Start();

t3.Start();

t1.Join();

t2.Join();

t3.Join();

}

static void FirstThread()

{

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

{

Console.WriteLine("First thread: " + i);

firstDone.Set();

secondDone.WaitOne();

}

}

static void SecondThread()

{

for (int i = 10; i < 20; i++)

{

firstDone.WaitOne();

Console.WriteLine("Second thread: " + i);

secondDone.Set();

}

}

static void ThirdThread()

{

for (int i = 20; i < 30; i++)

{

secondDone.WaitOne();

Console.WriteLine("Third thread: " + i);

firstDone.Set();

}

}

}

}

Анализ результатов:

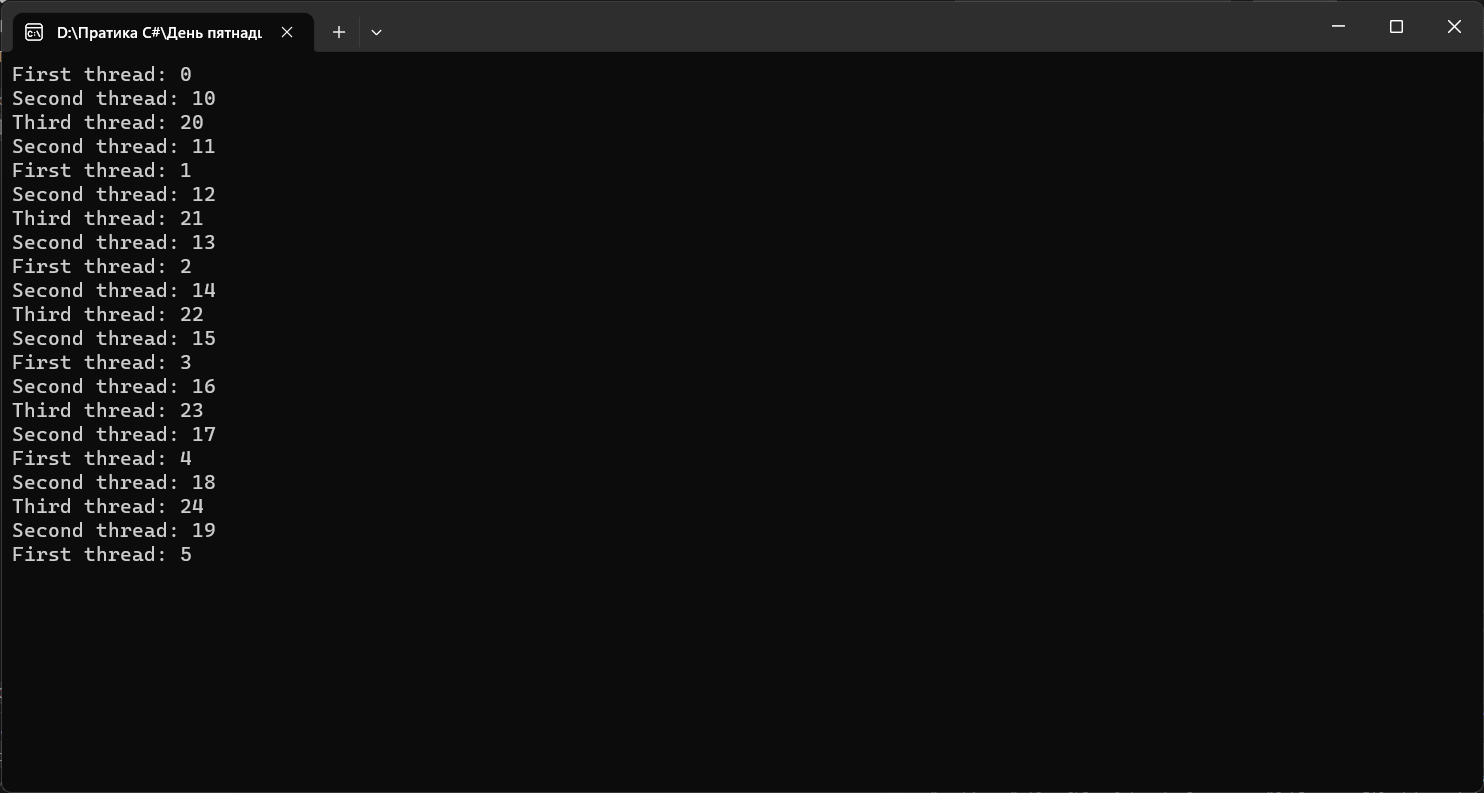


Рисунок 1.1 –работы программы

Задание 2. Создать консольное приложение, в котором будут два метода, один метод будет выполняться двумя потоками одновременно, а другой метод в каждый момент времени будет выполняться одним потоком. (Количество потоков: 2, Метод: 1. A+A 1 +A 2 +A 3 +..+A N , А и N вводятся с клавиатуры; 2. A\*A 1 \*A 2 \*A 3 \*…\*A N , А и N вводятся с клавиатуры ).

namespace task3

{

using System;

using System.Threading;

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Enter A:");

int A = int.Parse(Console.ReadLine());

Console.WriteLine("Enter N:");

int N = int.Parse(Console.ReadLine());

Thread t1 = new Thread(() => Method1(A, N));

Thread t2 = new Thread(() => Method1(A, N));

Thread t3 = new Thread(() => Method2(A, N));

t1.Start();

t2.Start();

t3.Start();

t1.Join();

t2.Join();

t3.Join();

}

static void Method1(int A, int N)

{

int sum = A;

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

{

sum += A + i;

}

Console.WriteLine("Method1: Sum = " + sum + " (Thread ID: " + Thread.CurrentThread.ManagedThreadId + ")");

}

static void Method2(int A, int N)

{

int prod = A;

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

{

prod \*= A + i;

}

Console.WriteLine("Method2: Product = " + prod + " (Thread ID: " + Thread.CurrentThread.ManagedThreadId + ")");

}

}

}Таблица 1.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 5, 8 | 81 |

Анализ результатов:

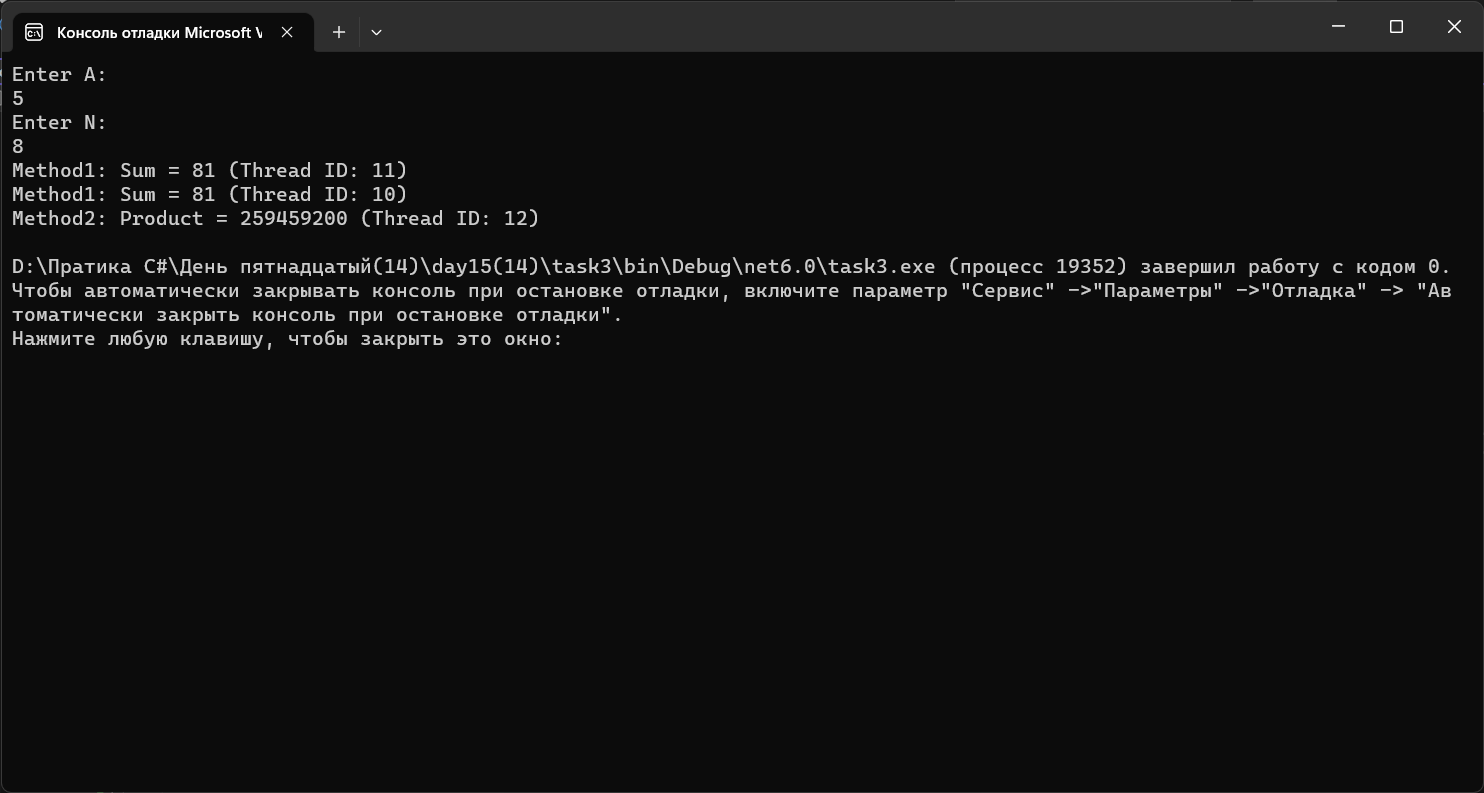
**

Рисунок 1.2 – Результат работы программы

работы программы