**14 Многопоточные приложения**

Задания №1. Дана последовательность натуральных чисел {a 0 …a n–1}. Создать многопоточное приложение для поиска суммы ∑a i, где a i – четные числа. Примерный алгоритм реализации: в качестве примера рассмотрим случай, когда в массиве хранятся номера элементов. Основной поток создает нужное число потоков, число которых задается параметром, каждому из которых передается параметр – номер процесса (или номер части массива, обрабатываемой потоком). Функция потока возвращает частичную сумму. Основной поток ожидает завершения всех потоков и находит общую сумму.

Листинг программы:

using System;

using System.Threading;

class Program

{

static int[] numbers;

static int totalSum = 0;

static object lockObj = new object();

static void CalculatePartialSum(int startIndex, int endIndex)

{

int partialSum = 0;

for (int i = startIndex; i < endIndex; i++)

{

if (numbers[i] % 2 == 0)

{

partialSum += numbers[i];

}

}

lock (lockObj)

{

totalSum += partialSum;

}

}

static void Main()

{

Console.Write("Введите количество элементов массива: ");

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

numbers = new int[n];

Console.WriteLine("Введите элементы массива:");

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

{

numbers[i] = int.Parse(Console.ReadLine());

}

Console.Write("Введите количество потоков: ");

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

Thread[] threads = new Thread[threadCount];

int chunkSize = n / threadCount;

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

{

int startIndex = i \* chunkSize;

int endIndex = (i == threadCount - 1) ? n : startIndex + chunkSize;

threads[i] = new Thread(() => CalculatePartialSum(startIndex, endIndex));

threads[i].Start();

}

foreach (Thread thread in threads)

{

thread.Join();

}

Console.WriteLine($"Общая сумма четных чисел: {totalSum}");

}

}

Анализ результатов:

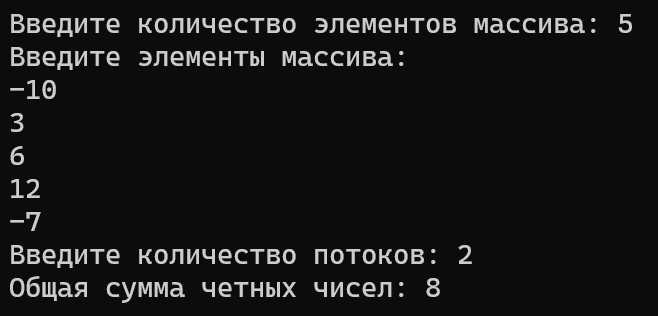


Рисунок 14.1 – Результаты работы программы