

**ГБОУ ВПО Нижегородский государственный технический
университет им. Р. Е. Алексеева
Институт радиоэлектроники и информационных технологий,
кафедра "Вычислительные системы и технологии"**

СОГЛАСОВАНО

Доцент каф. ВСТ

_____ Гай В. Е.

“ ____ ” _____

**ТЕХНОЛОГИИ РАСПРЕДЕЛЁННОЙ ОБРАБОТКИ ДАННЫХ
Отчет к лабораторной работе №3**

**РАСПАРАЛЛЕЛИВАНИЕ АЛГОРИТМА С ПОМОЩЬЮ
БИБЛИОТЕКИ CSR**

Инв. подл.	Подп. и дата	Инв. дубл.	Подп. и дата	Взам. инв.	Инв. дубл.	Подп. и дата	Инв. подл.

Студент гр. 13-В-2

_____ Домнина Н. А.

“ ____ ” _____

СОДЕРЖАНИЕ

1	Цель и порядок выполнения работы	3
2	Теоретические сведения	4
2.1	Библиотека Concurrent and Coordination Runtime	4
2.2	Создание проекта	5
2.3	Оценка времени выполнения	5
3	Выполнение лабораторной работы	6
3.1	Вариант задания	6
3.2	Листинг программы	6
3.3	Результат работы программы	9
4	Вывод	9

Инв. подл.	Подп. и дата	Инв. дубл.	Взам. инв.	Подп. и дата	Изм. Лист докум. Подп. Дата Разраб. Домнина Н. А. Пров. Гай В. Е. Н. контр. Утв.	Распараллеливание алгоритма с помощью библиотеки CCR Технологии распределённой обработки данных Отчет к лабораторной работе №3				Лит.	Лист	Листов
											2	9

1 ЦЕЛЬ И ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Цель работы: получить представления о возможности библиотеки Concurrency and Coordination Runtime для организации параллельных вычислений.

Порядок выполнения работы:

- а) Разработка последовательного алгоритма, решающего одну из приведённых задач в соответствии с выданным вариантом задания;
- б) Разработка параллельного алгоритма, соответствующий варианту последовательного алгоритма;
- в) Выполнение сравнения времени выполнения последовательного и параллельного алгоритмов обработки данных при различных размерностях исходных данных.

Инв. подл.	Подп. и дата	Взам. инв.	Инв. дубл.	Подп. и дата					
Изм.	Лист	докум.	Подп.	Дата	Распараллеливание алгоритма с помощью библиотеки CCR		Лист		
							3		

2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

2.1 Библиотека Concurrent and Coordination Runtime

Библиотека Concurrent and Coordination Runtime (CCR) предназначена для организации обработки данных с помощью параллельно и асинхронно выполняющихся методов. Взаимодействие между такими методами организуется на основе сообщений. Рассылка сообщений основана на использовании портов. Основные понятия CCR:

- а) Сообщение – экземпляр любого типа данных;
- б) Порт – очередь сообщений типа FIFO (First-In-First-Out), сообщение остаётся в порте пока не будет извлечено из очереди порта получателем. Определение порта:

```
Port<int> p = new Port<int>();
```

Отправка сообщения в порт:

```
p.Post(1);
```

- в) получатель – структура, которая выполняет обработку сообщений. Данная структура объединяет:

- один или несколько портов, в которые отправляются сообщения;
- метод (или методы), которые используются для обработки сообщений (такой метод называется задачей);
- логическое условие, определяющее ситуации, в которых активизируется тот или иной получатель.

Делегат, входящий в получатель, выполнится, когда в порт `intPort` придёт сообщение. Получатели сообщений бывают двух типов: временные и постоянные (в примере получатель – временный). Временный получатель, обработав сообщение (или несколько сообщений), удаляется из списка получателей сообщений данного порта.

Инов. подл.	Подп. и дата	Взам. инв.	Инов. дубл.	Подп. и дата	<i>Распараллеливание алгоритма с помощью библиотеки CCR</i>					Лист
										4
Изм.	Лист	докум.	Подп.	Дата						

г) процессом запуска задач управляет диспетчер. После выполнения условий активации задачи (одним из условий активации может быть получение портом сообщения) диспетчер назначает задаче поток из пула потоков, в котором она будет выполняться. Описание диспетчера с двумя потоками в пуле:

```
Dispatcher d = new Dispatcher(2, "MyPool");
```

Описание очереди диспетчера, в которую задачи ставятся на выполнение:

```
DispatcherQueue dq = new DispatcherQueue("MyQueue d);
```

2.2 Создание проекта

Нужно выполнить следующие действия:

- а) Установить библиотеку CCR (CCR входит в состав Microsoft Robotics Developer Studio);
- б) Создать проект консольного приложения и добавьте к проекту библиотеку Microsoft.Ccr.Core.dll.

2.3 Оценка времени выполнения

Время выполнения вычислений будем определять с помощью класса

Stopwatch :

```
Stopwatch sWatch = new Stopwatch();
```

```
sWatch.Start();
```

```
<выполняемый код>
```

```
sWatch.Stop();
```

```
Console.WriteLine(sWatch.ElapsedMilliseconds.ToString());
```

Изм.	Лист	докум.	Подп.	Дата	<i>Распараллеливание алгоритма с помощью библиотеки CCR</i>	Лист 5
Изм.	Лист	докум.	Подп.	Дата		

3 ВЫПОЛНЕНИЕ ЛАБОРАТОРНОЙ РАБОТЫ

3.1 Вариант задания

Вариант 6:

- Разработка алгоритма поиска максимального и минимального значения массива.

3.2 Листинг программы

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Diagnostics;
using Microsoft.Ccr.Core;
using System.Threading;

namespace ConsoleApplication3
{
    public class InputData
    {
        public int start; // начало диапазона
        public int stop;  // конец диапазона
        public int i;
    }
    class Program
    {
        static int[] A;
        static int[] B;
        static int n;
        static int nc;
```

Изм.	Лист	докум.	Подп.	Дата	<p>using System.Linq; using System.Text; using System.Diagnostics; using Microsoft.Ccr.Core; using System.Threading;</p> <p>namespace ConsoleApplication3</p> <p>{</p> <p> public class InputData</p> <p> {</p> <p> public int start; // начало диапазона</p> <p> public int stop; // конец диапазона</p> <p> public int i;</p> <p> }</p> <p> class Program</p> <p> {</p> <p> static int[] A;</p> <p> static int[] B;</p> <p> static int n;</p> <p> static int nc;</p> <p> }</p>	Лист

```

static void Mul(InputData data, Port<int> resp)
{
    int min2 = A[0], max2 = A[0], i;
    System.Diagnostics.Stopwatch sWatch = new System.
        Diagnostics.Stopwatch();
    sWatch.Start(); // запуск секундомера
    for (i = data.start; i <= data.stop; i++)
    {
        if (A[i] < min2)
            min2 = A[i];
        if (A[i] > max2)
            max2 = A[i];
    }
    sWatch.Stop(); // отключение секундомера
    resp.Post(1);
    Console.WriteLine(min2);
    Console.WriteLine(max2);
    Console.WriteLine("Поток {0}: Параллельный алгоритм = {1} мс.", Thread.CurrentThread.
        ManagedThreadId, sWatch.ElapsedMilliseconds.
        ToString());
}

```

```

static void Main(string[] args)
{
    int i, min, max;
    nc = 2; // количество ядер
    n = 100000000; // количество элементов в массиве
    A = new int[n];
    B = new int[nc];

    Random r = new Random(); // генерируем массив
    for (int j = 0; j < n; j++)
        A[j] = r.Next(100);
    System.Diagnostics.Stopwatch sWatch = new System.
        Diagnostics.Stopwatch();
    sWatch.Start(); // запуск секундомера
    min = A[0];
    for (i = 1; i < n; i++)
    {

```

Инов. подл.	Подп. и дата	<div>Распараллеливание алгоритма с помощью библиотеки CCR</div>				Лист
Взам. инв.	Инов. дубл.					7
Подп. и дата						
Изм.	Лист	докум.	Подп.	Дата	Формат А4	

```

        if (A[i] < min)
            min = A[i];
    }
    max = A[0];
    for (i = 1; i < n; i++)
    {
        if (A[i] > max)
            max = A[i];
    }
    Console.WriteLine(min);
    Console.WriteLine(max);
    sWatch.Stop(); // отключение секундомера
    Console.WriteLine("Последовательный алгоритм = {0} мс.",
        sWatch.ElapsedMilliseconds.ToString());

    // создание массива объектов для хранения параметров
    InputData[] ClArr = new InputData[nc];
    for (i = 0; i < nc; i++)
        ClArr[i] = new InputData();
    // делим количество элементов в массиве на nc частей
    int step = (Int32)(n / nc);
    // заполняем массив параметров
    int c = -1;
    for (i = 0; i < nc; i++)
    {
        ClArr[i].start = c + 1;
        ClArr[i].stop = c + step;
        ClArr[i].i = i;
        c = c + step;
    }
    Dispatcher d = new Dispatcher(nc, "Test Pool"); // диспетч
        ер с nc потоками в пуле
    DispatcherQueue dq = new DispatcherQueue("Test Queue", d);
    Port<int> p = new Port<int>();

    for (i = 0; i < nc; i++)
        Arbiter.Activate(dq, new Task<InputData, Port<int>>()
            { ClArr[i], p, Mul });
    Arbiter.Activate(dq, Arbiter.MultipleItemReceive(true, p,
        nc, delegate(int[] array)
        { }));
}
}
}
}
}

```

Изм.	Лист	докум.	Подп.	Дата	Инв. подл.	Подп. и дата	Взам. инв.	Инв. дубл.	Подп. и дата
Распараллеливание алгоритма с помощью библиотеки CCR					Лист 8				

3.3 Результат работы программы

Скриншот работы программы представлен на Рис.1.

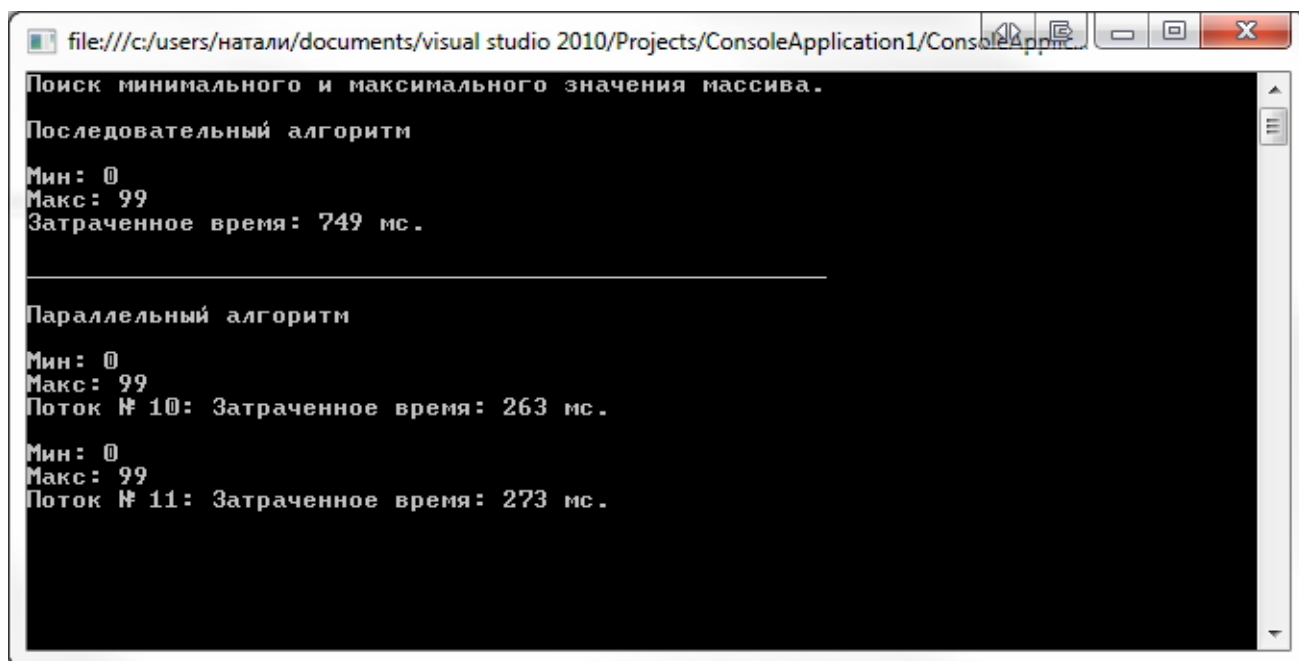


Рисунок 1

4 ВЫВОД

В результате выполнения лабораторной работы мы получили представление о возможности библиотеки Concurrent and Coordination Runtime для организации параллельных вычислений. Мы выяснили, что скорость работы параллельного алгоритма превосходит скорость работы последовательного алгоритма более чем в 2 раза. Быстродействие параллельного алгоритма напрямую зависит от числа используемых ядер.

Изм.	Лист	докум.	Подп.	Дата	<i>Распараллеливание алгоритма с помощью библиотеки CCR</i>	Лист
						9