

**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.

“ \_\_\_\_ ” \_\_\_\_\_

**ТРОД**  
**Отчет к лабораторной работе №3**

**РАСПАРАЛЛЕЛИВАНИЕ АЛГОРИТМА С ПОМОЩЬЮ  
БИБЛИОТЕКИ CSR**

Студент гр. 13-В-2

\_\_\_\_\_ Молчанов М. Н.

“ \_\_\_\_ ” \_\_\_\_\_

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

# СОДЕРЖАНИЕ

<b>1</b>	<b>Цель и порядок выполнения работы</b>	<b>3</b>
<b>2</b>	<b>Теоретические сведения</b>	<b>4</b>
2.1	Библиотека Concurrent and Coordination Runtime . . . . .	4
2.2	Создание проекта . . . . .	5
2.3	Оценка времени выполнения . . . . .	5
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
3.1	Вариант задания . . . . .	6
3.2	Листинг программы . . . . .	6
3.3	Результат работы программы . . . . .	8
<b>4</b>	<b>Вывод</b>	<b>9</b>

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

# 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 будут 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	<div>Распараллеливание алгоритма с помощью библиотеки CCR</div>					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:

- Установить библиотеку CCR (CCR входит в состав Microsoft Robotics Developer Studio);
- 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	<div>Распараллеливание алгоритма с помощью библиотеки CCR</div>	<div>Лист 5</div>

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

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());
```



```
void Mul(InputData data , Port<int> resp)
{
    Stopwatch sWatch = new Stopwatch();
    sWatch.Start();
    for (int i = data.start; i < data.stop; i++)
    {
        C[i] = 0;
        for (int j = 0; j < n; j++)
            C[i , j] += A[i , j] * B[i , j];
    }
    sWatch.Stop();
    Console.WriteLine("Поток {0}: Параллельный алгоритм =
    {1} мс.", Thread.CurrentThread.ManagedThreadId ,
    sWatch.ElapsedMilliseconds.ToString());
    resp.Post(1);
}
```

```
//Далее, задаются исходные данные для каждого экземпляра вычислительного метода:
// делим количество строк в матрице на nc частей
    int step = (Int32)(m / nc);
// заполняем массив параметров
    int c = -1;
```

## Распараллеливание алгоритма с помощью библиотеки CCR

```

        for (int i = 0; i < nc; i++)
        {
            ClArr[i].start = c + 1;
            ClArr[i].stop = c + step;
            c = c + step;
        }
//Создаётся диспетчер с пулом из двух потоков:
Dispatcher d = new Dispatcher(nc, "Test Pool");
DispatcherQueue dq = new DispatcherQueue("Test Queue", d);
//Описывается порт, в который каждый экземпляр метода Mul() отправляет
    сообщение после завершения вычислений:
    Port<int> p = new Port<int>();
//Метод Arbiter.Activate помещает в очередь диспетчера две задачи (два
    экземпляра метода Mul):
    for (int i = 0; i < nc; i++)
        Arbiter.Activate(dq, new Task<InputData, Port<int>>(
            ClArr[i], p, Mul));
//С помощью метода Arbiter.MultipleItemReceive запускается задача (при
    ёмник), которая обрабатывает получение двух сообщений портом p:
    Arbiter.Activate(Environment.TaskQueue, Arbiter.
        MultipleItemReceive(true, p, nc, delegate(int[] array)
        {
            Console.WriteLine("Вычисления завершены");
        }));
    }
}
}

```

### 3.3 Результат работы программы

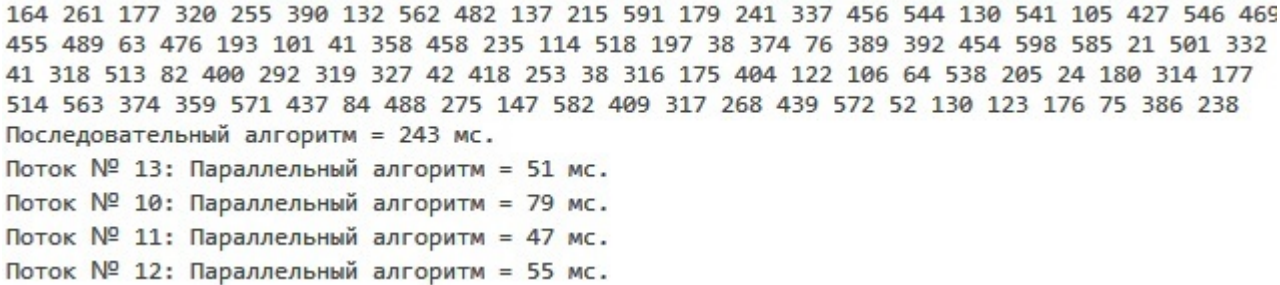
Скриншот работы программы представлен на Рис.1.

```

164 261 177 320 255 390 132 562 482 137 215 591 179 241 337 456 544 130 541 105 427 546 469
455 489 63 476 193 101 41 358 458 235 114 518 197 38 374 76 389 392 454 598 585 21 501 332
41 318 513 82 400 292 319 327 42 418 253 38 316 175 404 122 106 64 538 205 24 180 314 177
514 563 374 359 571 437 84 488 275 147 582 409 317 268 439 572 52 130 123 176 75 386 238
Последовательный алгоритм = 243 мс.
Поток № 13: Параллельный алгоритм = 51 мс.
Поток № 10: Параллельный алгоритм = 79 мс.
Поток № 11: Параллельный алгоритм = 47 мс.
Поток № 12: Параллельный алгоритм = 55 мс.

```

Рисунок 1

Изм.	Лист	докум.	Подп.	Дата	<pre>{     Console.WriteLine("Вычисления завершены"); })); }</pre>
Инв. подл.	Подп. и дата	Взам. инв.	Инв. дубл.	Подп. и дата	<h3>3.3 Результат работы программы</h3> <p>Скриншот работы программы представлен на Рис.1.</p>  <p>164 261 177 320 255 390 132 562 482 137 215 591 179 241 337 456 544 130 541 105 427 546 469 455 489 63 476 193 101 41 358 458 235 114 518 197 38 374 76 389 392 454 598 585 21 501 332 41 318 513 82 400 292 319 327 42 418 253 38 316 175 404 122 106 64 538 205 24 180 314 177 514 563 374 359 571 437 84 488 275 147 582 409 317 268 439 572 52 130 123 176 75 386 238 Последовательный алгоритм = 243 мс. Поток № 13: Параллельный алгоритм = 51 мс. Поток № 10: Параллельный алгоритм = 79 мс. Поток № 11: Параллельный алгоритм = 47 мс. Поток № 12: Параллельный алгоритм = 55 мс.</p> <p>Рисунок 1</p>
					<h2>Распараллеливание алгоритма с помощью библиотеки CCR</h2>



## 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. Быстродействие параллельного алгоритма напрямую зависит от числа используемых ядер.

Инв. подл.	Подп. и дата	Взам. инв.	Инв. дубл.	Подп. и дата					
Изм.	Лист	докум.	Подп.	Дата	Распараллеливание алгоритма с помощью библиотеки CCR	Лист			
						9			