

ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»
(СПбГУТ)**

Кафедра Автоматизации предприятий связи

МНОГОПОТОЧНОЕ И АСИНХРОННОЕ ПРОГРАММИРОВАНИЕ

2020

Делегаты

```
public delegate int OpDelegate(int a, int b);

class Program
{
    static void Main(string[] args)
    {
        Calc calc = new Calc();
        OpDelegate plusDelegate = new OpDelegate(calc.Plus);
        OpDelegate minusDelegate = new OpDelegate(Calc.Minus);

        int r1 = plusDelegate(5, 10);
        int r2 = minusDelegate(20, 18);
        Console.WriteLine("r1={0}, r2={1}", r1, r2);

        r1 = Calc.Op(plusDelegate, 100, 200);
        r2 = Calc.Op(minusDelegate, 500, 25);
        Console.WriteLine("r1={0}, r2={1}", r1, r2);

        r1 = Calc.Op((a, b) => { return a * b; }, 5, 7);
        r2 = Calc.Op((a, b) => { return a * 5 + b * 10; }, 5, 10);
        Console.WriteLine("r1={0}, r2={1}", r1, r2);
    }
}
```

```
class Calc
{
    public int Plus(int a, int b)
    {
        return a + b;
    }

    public static int Minus(int a, int b)
    {
        return a - b;
    }

    public static int Op(OpDelegate op, int a, int b)
    {
        return op(a, b);
    }
}
```

```
class Program
{
    delegate int Plus(int a, int b);
    delegate void Hello(string s);

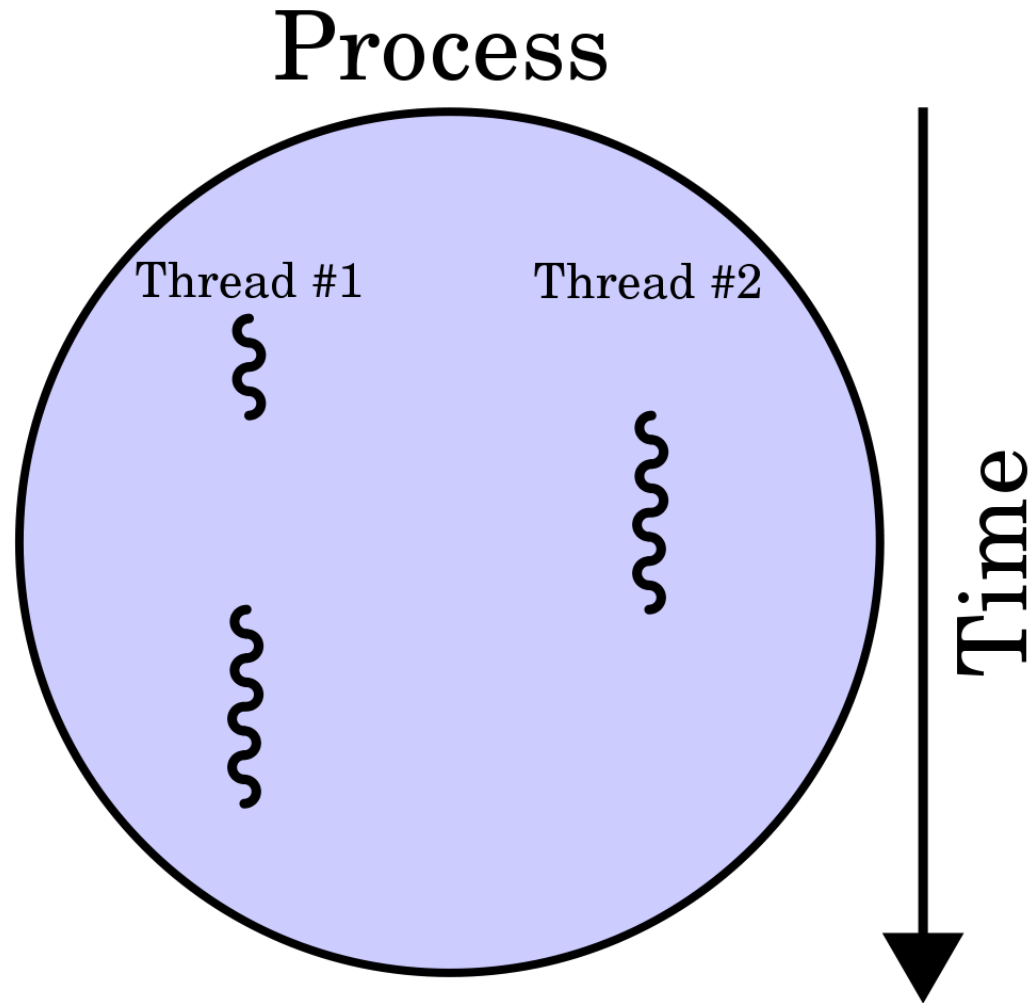
    static void Main(string[] args)
    {
        Plus plus = (int a, int b) => a + b;
        int r = plus(5, 10);
        Console.WriteLine("Result = {0}", r);

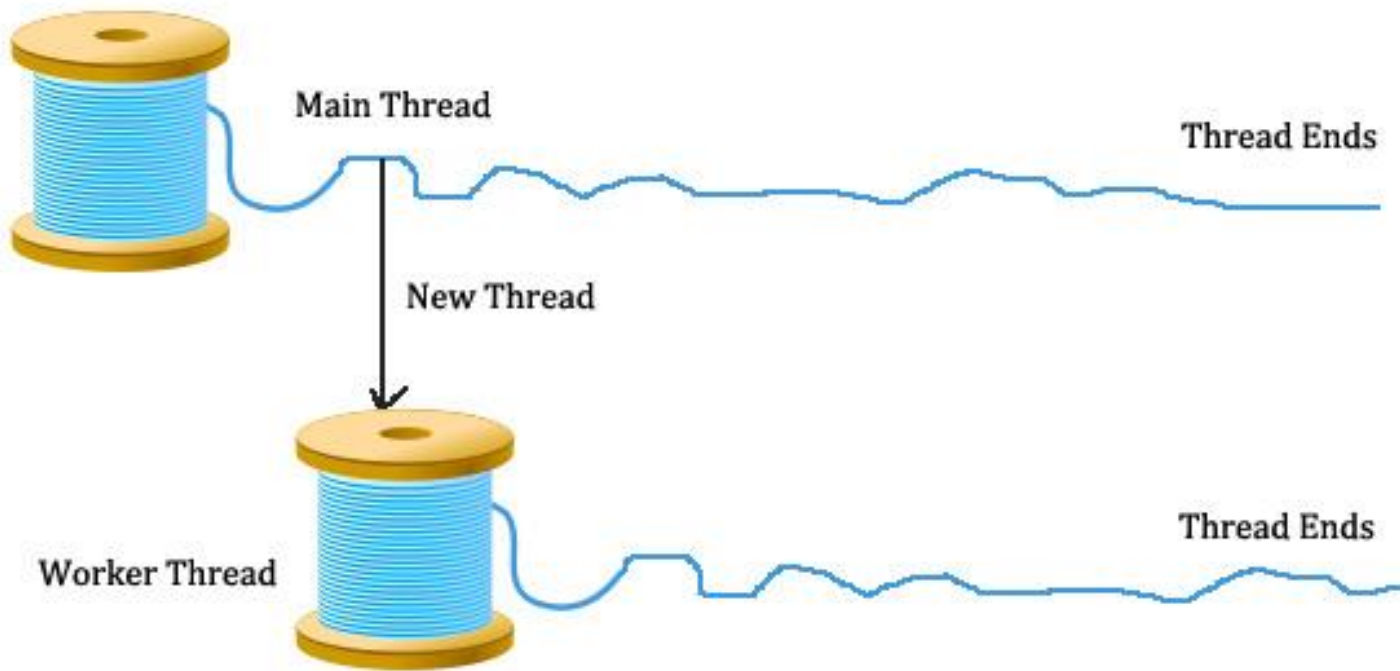
        Func<int, int, int> minus = (a, b) => a - b;
        r = minus(10, 8);
        Console.WriteLine("Result = {0}", r);

        Hello hello = n => { string s = "Hello, " + n + "!"; Console.WriteLine(s); };
        hello("World");
    }
}
```

Потоки

Запуск двух потоков на одноядерном процессоре






```
using System;  
using System.Threading;
```

```
namespace Thread01
```

```
{
```

Ссылки: 0

```
class Program
```

```
{
```

Ссылки: 0

```
static void Main(string[] args)
```

```
{
```

```
    Thread t = Thread.CurrentThread;
```

```
    Console.WriteLine("Имя потока: {0}", t.Name);
```

```
    t.Name = "Основной поток";
```

```
    Console.WriteLine("Имя потока: {0}", t.Name);
```

```
    Console.WriteLine("Поток запущен? {0}", t.IsAlive);
```

```
    Console.WriteLine("Статус потока: {0}", t.ThreadState);
```

```
    Console.WriteLine("Приоритет потока: {0}", t.Priority);
```

```
}
```

```
}
```

```
}
```

Консоль отладки Microsoft Visual Studio

Имя потока:

Имя потока: Основной поток

Поток запущен? True

Статус потока: Running

Приоритет потока: Normal

Приоритеты потоков

- **Lowest**
- **BelowNormal**
- **Normal**
- **AboveNormal**
- **Highest**

```

using System;
using System.Threading;

namespace Thread02
{
    Ссылка: 0
    class Program
    {
        Ссылка: 0
        static void Main(string[] args)
        {
            Thread thread2 = new Thread(Op2);
            thread2.Start();
            for(int i = 1; i < 6; i++)
            {
                Console.WriteLine("Главный поток {0}", i);
                Thread.Sleep(1000);
            }
        }
    }

    ссылка: 1
    public static void Op2()
    {
        for (int i = 1; i < 11; i++)
        {
            Console.WriteLine("Поток 2 {0}", 1000 * i);
            Thread.Sleep(1000);
        }
    }
}

```

```

Поток 2 1000
Главный поток 1
Поток 2 2000
Главный поток 2
Поток 2 3000
Главный поток 3
Поток 2 4000
Главный поток 4
Поток 2 5000
Главный поток 5
Поток 2 6000
Поток 2 7000
Поток 2 8000
Поток 2 9000
Поток 2 10000

```

```

static void Main(string[] args)
{
    Thread thread2 = new Thread(Op2);
    thread2.Name = "Поток 2";
    thread2.Start();
    for(int i = 1; i < 6; i++)
    {
        Console.WriteLine("Главный поток {0}", i);
        Thread.Sleep(1000);
    }
    Thread thread3 = new Thread(Op2);
    thread3.Name = "Поток 3";
    thread3.Start();
}

```

Ссылка: 2

```

public static void Op2()
{
    for (int i = 1; i < 11; i++)
    {
        Console.WriteLine("{0}, {1}", Thread.CurrentThread.Name, i);
        Thread.Sleep(1000);
    }
}

```

```

1  Главный поток 1
2  Поток 2, 1000
3  Главный поток 2
4  Поток 2, 2000
5  Главный поток 3
6  Поток 2, 3000
7  Главный поток 4
8  Поток 2, 4000
9  Главный поток 5
10 Поток 2, 5000
11 Поток 3, 1000
12 Поток 2, 6000
13 Поток 3, 2000
14 Поток 2, 7000
15 Поток 3, 3000
16 Поток 2, 8000
17 Поток 3, 4000
18 Поток 2, 9000
19 Поток 3, 5000
20 Поток 2, 10000
21 Поток 3, 6000
22 Поток 3, 7000
23 Поток 3, 8000
24 Поток 3, 9000
25 Поток 3, 10000

```

```

static void Main(string[] args)
{
    Thread thread2 = new Thread(Op2);
    thread2.Name = "Поток 2";
    thread2.Start();
    Thread thread3 = new Thread(Op2);
    thread3.Name = "Поток 3";
    thread3.Start();
    for (int i = 1; i < 6; i++)
    {
        Console.WriteLine("Главный поток {0}", i);
        Thread.Sleep(1000);
    }
}

```

Ссылка: 2

```

public static void Op2()
{
    for (int i = 1; i < 11; i++)
    {
        Console.WriteLine("{0}, {1}", Thread.CurrentThread.
        Thread.Sleep(1000);
    }
}

```

```

Поток 3, 1000
Поток 2, 1000
Главный поток 1
Поток 3, 2000
Поток 2, 2000
Главный поток 2
Поток 2, 3000
Поток 3, 3000
Главный поток 3
Поток 2, 4000
Поток 3, 4000
Главный поток 4
Поток 2, 5000
Поток 3, 5000
Главный поток 5
Поток 2, 6000
Поток 3, 6000
Поток 2, 7000
Поток 3, 7000
Поток 2, 8000
Поток 3, 8000
Поток 2, 9000
Поток 3, 9000
Поток 2, 10000
Поток 3, 10000

```

```

static void Main(string[] args)
{
    Thread thread2 = new Thread(new ParameterizedThreadStart(Op2));
    thread2.Name = "Поток 1";
    thread2.Start(10);
    Thread thread3 = new Thread(new ParameterizedThreadStart(Op2));
    thread3.Name = "Поток 2";
    thread3.Start(1000);
}

```

Ссылка: 2

```

public static void Op2(Object param)
{
    int a = (int)param;
    for (int i = 1; i < 6; i++)
    {
        Console.WriteLine("{0}, {1}", Thread.CurrentThread.Name, a * i);
        Thread.Sleep(1000);
    }
}

```

```

Поток 2, 1000
Поток 1, 10
Поток 1, 20
Поток 2, 2000
Поток 1, 30
Поток 2, 3000
Поток 1, 40
Поток 2, 4000
Поток 1, 50
Поток 2, 5000

```

```

static void Main(string[] args)
{
    Thread thread2 = new Thread(new ParameterizedThreadStart(Op2));
    thread2.Name = "Поток 1";
    thread2.Start(new Param { IterationNumber = 5, Multiplier = 10 });
    Thread thread3 = new Thread(new ParameterizedThreadStart(Op2));
    thread3.Name = "Поток 2";
    thread3.Start(new Param { IterationNumber = 7, Multiplier = 1000 });
}

```

```

Поток 1, 10
Поток 2, 1000
Поток 1, 20
Поток 2, 2000
Поток 1, 30
Поток 2, 3000
Поток 1, 40
Поток 2, 4000
Поток 1, 50
Поток 2, 5000
Поток 2, 6000
Поток 2, 7000

```

Ссылка: 2

```

public static void Op2(Object p)
{
    Param param = (Param)p;
    for (int i = 1; i <= param.IterationNumber; i++)
    {
        Console.WriteLine("{0}, {1}", Thread.CurrentThread.Name, param.Multiplier * i);
        Thread.Sleep(1000);
    }
}

```

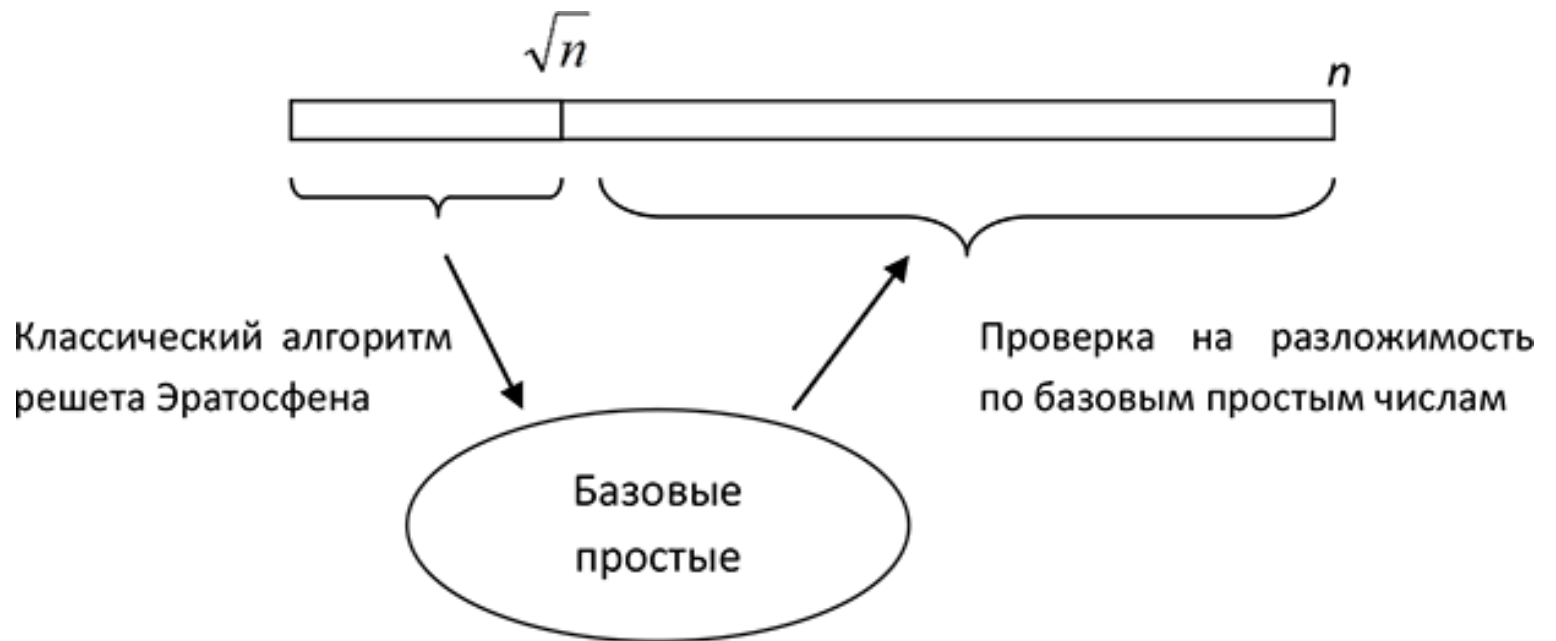
Ссылка: 4

```

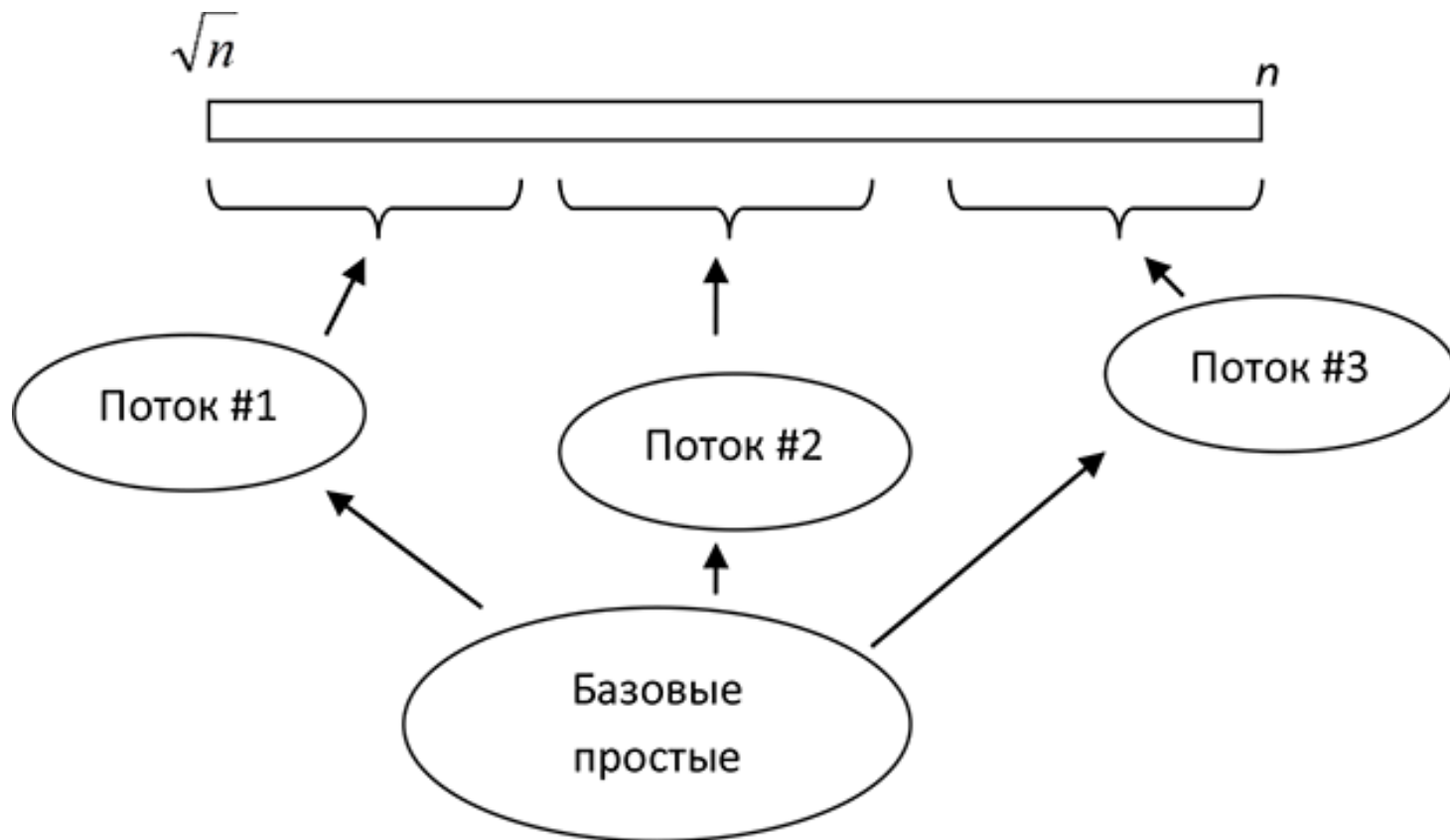
class Param
{
    Ссылка: 3
    public int IterationNumber { get; set; }
    Ссылка: 3
    public int Multiplier { get; set; }
}

```

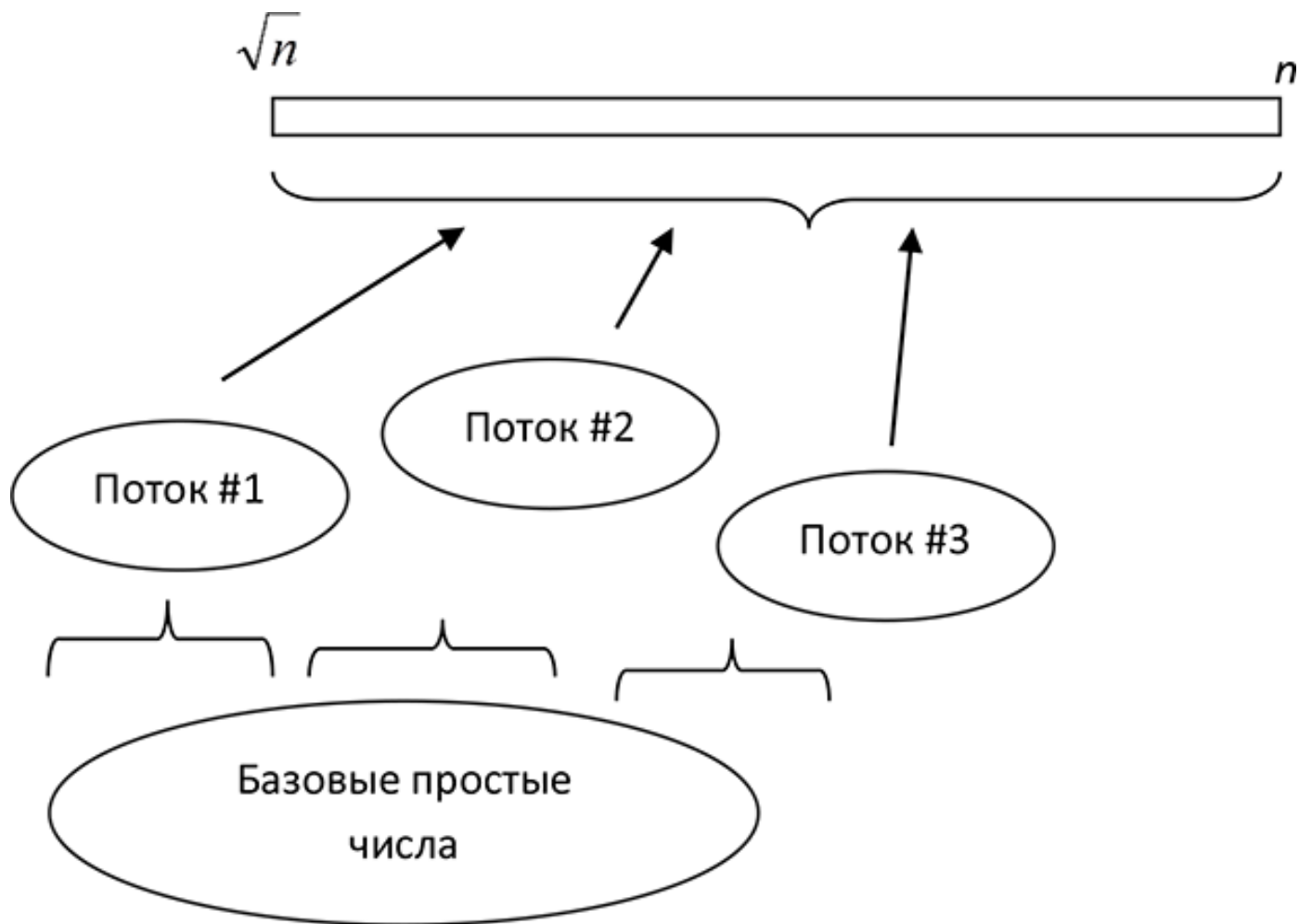
Модифицированный последовательный алгоритм поиска



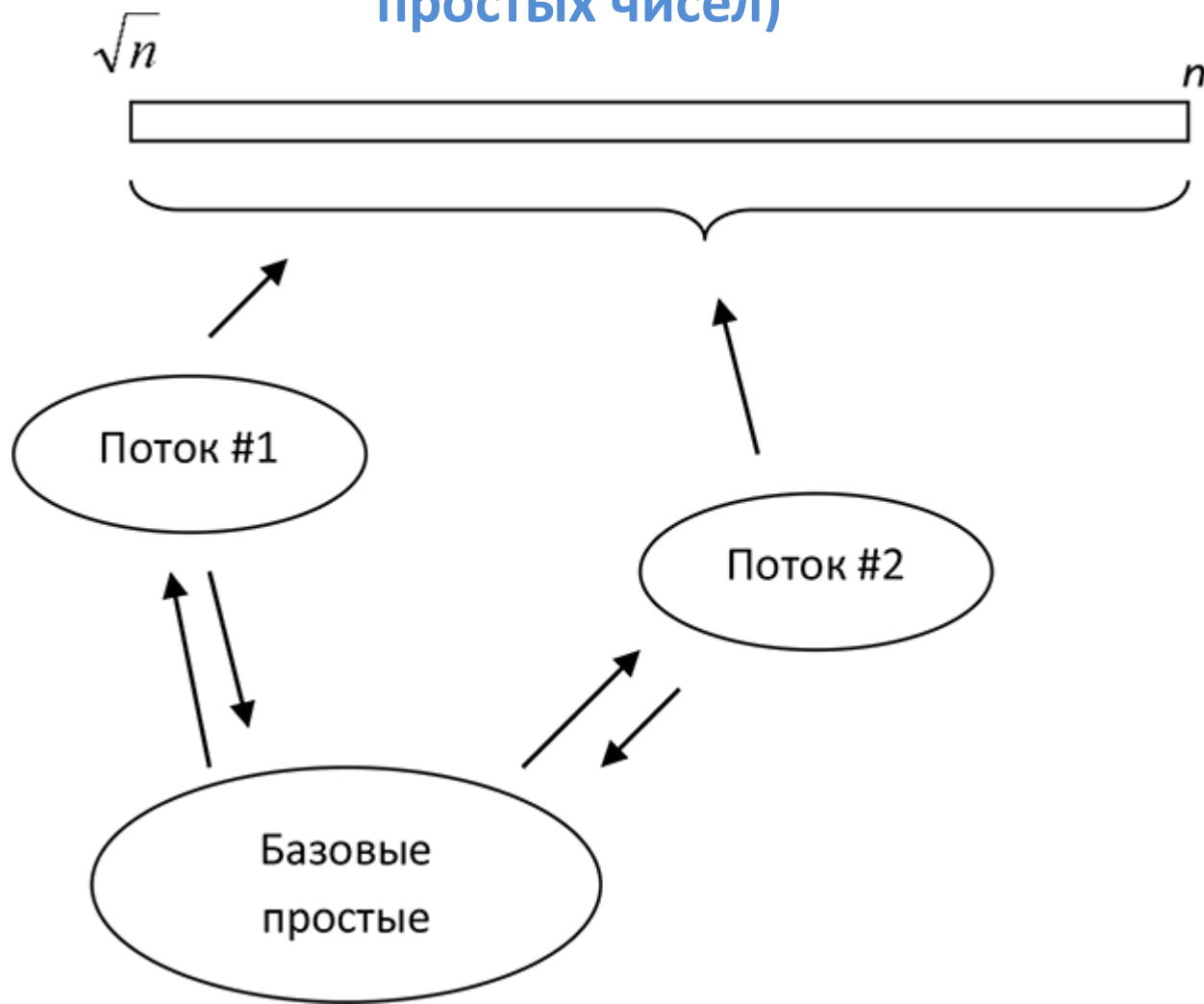
Параллельный алгоритм (декомпозиция по данным)



Параллельный алгоритм (декомпозиция по множеству базовых простых чисел)



Параллельный алгоритм (последовательный перебор множества базовых простых чисел)



```
1 class Program
2 {
3     static int x=0;
4     static void Main(string[] args)
5     {
6         for (int i = 0; i < 5; i++)
7         {
8             Thread myThread = new Thread(Count);
9             myThread.Name = "Поток " + i.ToString();
10            myThread.Start();
11        }
12
13        Console.ReadLine();
14    }
```

```
class ThreadSafe
{
    static bool done;
    static object locker = new object();

    static void Main()
    {
        new Thread(Go).Start();
        Go();
    }

    static void Go()
    {
        lock (locker)
        {
            if (!done)
            {
                Console.WriteLine("Done");
                done = true;
            }
        }
    }
}
```

```
class ThreadUnsafe
{
    static int val1, val2;

    static void Go()
    {
        if (val2 != 0)
            Console.WriteLine(val1 / val2);
        val2 = 0;
    }
}
```

```
class ThreadSafe
{
    static object locker = new object();
    static int val1, val2;

    static void Go()
    {
        lock (locker)
        {
            if (val2 != 0)
                Console.WriteLine(val1 / val2);

            val2 = 0;
        }
    }
}
```

```
class ThreadUnsafe
{
    static int x;
    static void Increment() { x++; }
    static void Assign()    { x = 123; }
}
```

```
class ThreadUnsafe
{
    static object locker = new object();
    static int x;

    static void Increment()
    {
        lock (locker)
            x++;
    }

    static void Assign()
    {
        lock (locker)
            x = 123;
    }
}
```



```
class ThreadTest
{
    static void Main()
    {
        Thread t = new Thread(Go);
        t.Start(true);           // == Go(true)
        Go(false);
    }

    static void Go(object upperCase)
    {
        bool upper = (bool)upperCase;
        Console.WriteLine(upper ? "HELLO!" : "hello!");
    }
}
```

```
class JoinDemo
{
    static void Main()
    {
        Thread t = new Thread(delegate() { Console.ReadLine(); });
        t.Start();
        t.Join();    // ожидать, пока поток не завершится
        Console.WriteLine("Thread t's ReadLine complete!");
    }
}
```