# Лабораторная работа 9. Многопоточное программирование.

## Задание 1.

Организовать 3 потока, которые выполняют любые из указанных действий.

Задайте потокам различные приоритеты.

1. Выводит нечетные числа от 1 до 100.
2. Выводит четные числа от 1 до 100.
3. Выводит английские буквы от A до Z.
4. Выводит английские буквы в обратном порядке от Z до A.
5. Выводит числа Фибоначчи
6. Выводит строки от 1 до 30 звездочек (\*).
7. В задании 6 вместо звездочки используйте любой другой символ.

const int edge = 30;

static void Main(string[] args)

{

Thread threadEven = new Thread(new ThreadStart(EvenSeq));

threadEven.Priority = ThreadPriority.Lowest;

threadEven.Name = "Even";

threadEven.Start();

Thread threadOdd = new Thread(OddSeq);

threadOdd.Priority = ThreadPriority.Normal;

threadOdd.Name = "Odd";

threadOdd.Start();

Thread threadFibonachy = new Thread(FibonachiSeq);

threadFibonachy.Priority = ThreadPriority.Highest;

threadFibonachy.Name = "Fibonachy";

threadFibonachy.Start();

Console.ReadLine();

}

static void EvenSeq()

{

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

if (i % 2 == 0)

Console.WriteLine(Thread.CurrentThread.Name + ": " + i);

}

static void OddSeq()

{

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

if (i % 2 != 0)

Console.WriteLine(Thread.CurrentThread.Name + ": " + i);

}

static void FibonachiSeq()

{

int a = 1, b = 1, c;

Console.WriteLine(Thread.CurrentThread.Name + "\_1: 1\n" + Thread.CurrentThread.Name + "\_2: 1");

for (int i = 3; i <= edge; i++)

{

c = a + b;

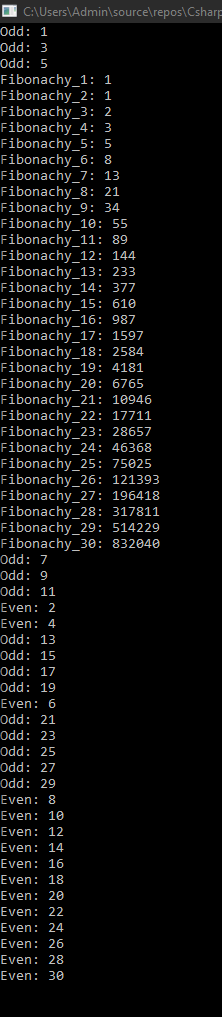
Console.WriteLine(Thread.CurrentThread.Name + $"\_{i}: " + c);

a = b;

b = c;

}

## }



## Задание 2.

Синхронизируйте потоки используя:

1. Критическую секцию

const int edge = 30;

static int i = 0;

static object oLock = new object();

static void Main(string[] args)

{

Thread threadEven = new Thread(new ThreadStart(EvenSeq));

threadEven.Priority = ThreadPriority.Lowest;

threadEven.Name = "Even";

threadEven.Start();

Thread threadOdd = new Thread(OddSeq);

threadOdd.Priority = ThreadPriority.Normal;

threadOdd.Name = "Odd";

threadOdd.Start();

Thread threadFibonachy = new Thread(FibonachiSeq);

threadFibonachy.Priority = ThreadPriority.Highest;

threadFibonachy.Name = "Fibonachy";

threadFibonachy.Start();

Console.ReadLine();

}

static void EvenSeq()

{

lock (oLock)

{

for (i = 1; i <= edge; i++)

if (i % 2 == 0)

Console.WriteLine(Thread.CurrentThread.Name + ": " + i);

}

}

static void OddSeq()

{

lock (oLock)

{

for (i = 1; i <= edge; i++)

if (i % 2 != 0)

Console.WriteLine(Thread.CurrentThread.Name + ": " + i);

}

}

static void FibonachiSeq()

{

lock (oLock)

{

int a = 1, b = 1, c;

Console.WriteLine(Thread.CurrentThread.Name + "\_1: 1\n" + Thread.CurrentThread.Name + "\_2: 1");

for (i = 3; i <= edge; i++)

{

c = a + b;

Console.WriteLine(Thread.CurrentThread.Name + $"\_{i}: " + c);

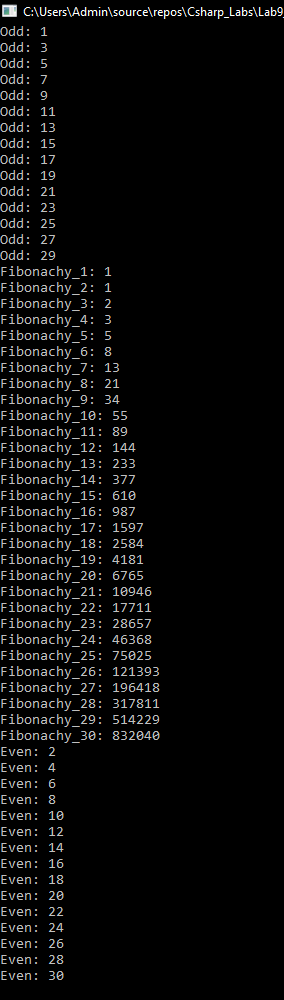
a = b;

b = c;

}

}

}



1. Монитор

static void EvenSeq()

{

try

{

Monitor.Enter(oLock);

for (i = 1; i <= edge; i++)

if (i % 2 == 0)

Console.WriteLine(Thread.CurrentThread.Name + ": " + i);

}

finally

{

Monitor.Exit(oLock);

}

}

static void OddSeq()

{

try

{

Monitor.Enter(oLock);

for (i = 1; i <= edge; i++)

if (i % 2 != 0)

Console.WriteLine(Thread.CurrentThread.Name + ": " + i);

}

finally

{

Monitor.Exit(oLock);

}

}

static void FibonachiSeq()

{

try

{

Monitor.Enter(oLock);

int a = 1, b = 1, c;

Console.WriteLine(Thread.CurrentThread.Name + "\_1: 1\n" + Thread.CurrentThread.Name + "\_2: 1");

for (i = 3; i <= edge; i++)

{

c = a + b;

Console.WriteLine(Thread.CurrentThread.Name + $"\_{i}: " + c);

a = b;

b = c;

}

}

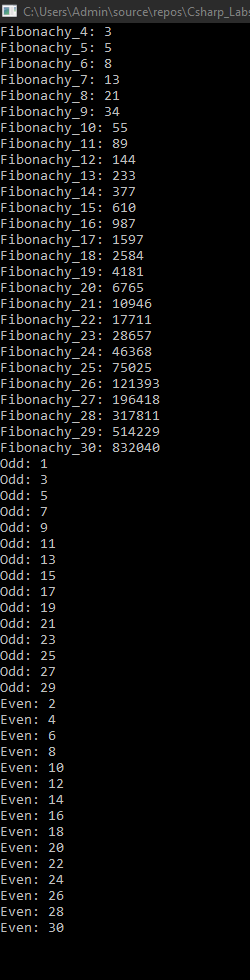
finally

{

Monitor.Exit(oLock);

}

}



1. Семафор

const int edge = 30;

static int i = 0;

static Semaphore sem = new Semaphore(1, 1);

static void Main(string[] args)

{

Thread threadEven = new Thread(new ThreadStart(EvenSeq));

threadEven.Priority = ThreadPriority.Lowest;

threadEven.Name = "Even";

threadEven.Start();

Thread threadOdd = new Thread(OddSeq);

threadOdd.Priority = ThreadPriority.Normal;

threadOdd.Name = "Odd";

threadOdd.Start();

Thread threadFibonachy = new Thread(FibonachiSeq);

threadFibonachy.Priority = ThreadPriority.Highest;

threadFibonachy.Name = "Fibonachy";

threadFibonachy.Start();

Console.ReadLine();

}

static void EvenSeq()

{

sem.WaitOne();

for (i = 1; i <= edge; i++)

if (i % 2 == 0)

Console.WriteLine(Thread.CurrentThread.Name + ": " + i);

sem.Release();

}

static void OddSeq()

{

sem.WaitOne();

for (i = 1; i <= edge; i++)

if (i % 2 != 0)

Console.WriteLine(Thread.CurrentThread.Name + ": " + i);

sem.Release();

}

static void FibonachiSeq()

{

sem.WaitOne();

int a = 1, b = 1, c;

Console.WriteLine(Thread.CurrentThread.Name + "\_1: 1\n" + Thread.CurrentThread.Name + "\_2: 1");

for (i = 3; i <= edge; i++)

{

c = a + b;

Console.WriteLine(Thread.CurrentThread.Name + $"\_{i}: " + c);

a = b;

b = c;

}

sem.Release();

}

1. Таймер

const int edge = 30;

static int i = 0;

static void Main(string[] args)

{

Thread.CurrentThread.Name = "Main";

TimerCallback tcEven = new TimerCallback(EvenSeq);

Timer tEven = new Timer(tcEven, null, 1000, 0);

TimerCallback tcOdd = new TimerCallback(OddSeq);

Timer tOdd = new Timer(tcOdd, null, 2000, 0);

TimerCallback tcFibonachi = new TimerCallback(FibonachiSeq);

Timer tFibonachi = new Timer(tcFibonachi, null, 3000, 0);

Console.ReadLine();

}

static void EvenSeq(object o)

{

for (i = 1; i <= edge; i++)

if (i % 2 == 0)

Console.WriteLine(Thread.CurrentThread.Name + ": " + i);

}

static void OddSeq(object o)

{

for (i = 1; i <= edge; i++)

if (i % 2 != 0)

Console.WriteLine(Thread.CurrentThread.Name + ": " + i);

}

static void FibonachiSeq(object o)

{

int a = 1, b = 1, c;

Console.WriteLine(Thread.CurrentThread.Name + "\_1: 1\n" + Thread.CurrentThread.Name + "\_2: 1");

for (i = 3; i <= edge; i++)

{

c = a + b;

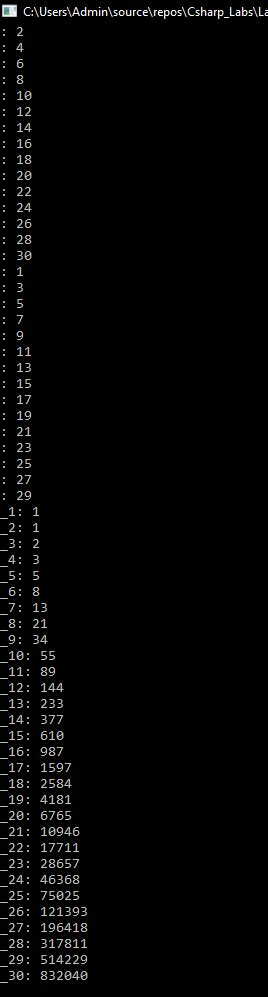
Console.WriteLine(Thread.CurrentThread.Name + $"\_{i}: " + c);

a = b;

b = c;

}

}



1. Задачи

const int edge = 30;

static int i = 0;

static void Main(string[] args)

{

Task taskEven = new Task(EvenSeq);

Task taskOdd = taskEven.ContinueWith(OddSeq);

Task taskFibonachi = taskOdd.ContinueWith(FibonachiSeq);

taskEven.Start();

Console.ReadLine();

}

static void EvenSeq()

{

for (i = 1; i <= edge; i++)

if (i % 2 == 0)

Console.WriteLine(Thread.CurrentThread.Name + ": " + i);

}

static void OddSeq(Task t)

{

for (i = 1; i <= edge; i++)

if (i % 2 != 0)

Console.WriteLine(Thread.CurrentThread.Name + ": " + i);

}

static void FibonachiSeq(Task t)

{

int a = 1, b = 1, c;

Console.WriteLine(Thread.CurrentThread.Name + "\_1: 1\n" + Thread.CurrentThread.Name + "\_2: 1");

for (i = 3; i <= edge; i++)

{

c = a + b;

Console.WriteLine(Thread.CurrentThread.Name + $"\_{i}: " + c);

a = b;

b = c;

}

}

1. Параллельное программирование

const int edge = 30;

static int i = 0;

static object oLock = new object();

static void Main(string[] args)

{

Parallel.Invoke(EvenSeq, OddSeq, FibonachiSeq);

Console.ReadLine();

}

static void EvenSeq()

{

lock (oLock)

{

for (i = 1; i <= edge; i++)

if (i % 2 == 0)

Console.WriteLine(Thread.CurrentThread.Name + ": " + i);

}

}

static void OddSeq()

{

lock (oLock)

{

for (i = 1; i <= edge; i++)

if (i % 2 != 0)

Console.WriteLine(Thread.CurrentThread.Name + ": " + i);

}

}

static void FibonachiSeq()

{

lock (oLock)

{

int a = 1, b = 1, c;

Console.WriteLine(Thread.CurrentThread.Name + "\_1: 1\n" + Thread.CurrentThread.Name + "\_2: 1");

for (i = 3; i <= edge; i++)

{

c = a + b;

Console.WriteLine(Thread.CurrentThread.Name + $"\_{i}: " + c);

a = b;

b = c;

}

}

}