# 11 Дәріс Арнайы блоктаушы конструкциялар

## 11.1 **Mutex** арнайы блоктаушы конструкция

Анықтама бойынша **Mutex** класы **lock** операторы сияқты ресурсқа немесе бір тізбектің код секциясына қатынауға рұқсат береді. Егерде тізбектерді синхрондау бір үдерісте орындалса, онда олардың арасында функционалдық айырмашылықтар жоқ. Бірақ, **Mutex** класы үшін объекті құру қажет. Сондай-ақ **Mutex** объектісі **lock** операторына қарағанда баяу жұмыс істейді деп саналады.

**Mutex** класының негізі артықшылығы – оның объектісі әртүрлі үдерістердің тізбектерін синхрондау немесе қосымшаның бірнеше даналарын қосуын тоқтату үшін пайдалану үшін қолданылады. Ол әртүрлі үдерістерде орындалатын параллельді тізбектер арасында «өзара шығарып тастау» проблемаларын шешеді деп айтылады.

Кез келген клас сияқты **Mutex** класы да қасиеттер мен әдістерге ие (дәрістің соңындағы қосымшаны қараңыз).

**Mutex**-ті келесі конструктордың көмегімен құруға болады:

static Mutex бағдарламадағы аты = new Mutex();
static Mutex бағдарламадағы аты = new Mutex(bool);
static Mutex бағдарламадағы аты = new Mutex(bool, "ОЖ аты");

Біріншісі атаусыз мьютексті құрайды және ағымдағы тізбекті оның иесі етіп жасайды, сондықтан да мьютекс ағымдағы тізбекпен блокталынады.

Екіншісі мьютексті құрайтын тізбек оған ие болуға (оны блоктауға)тырысатынын анықтайтын тек логикалық аргументті қабылдайды. Егерде bool типіндегі аргументке false-де орнатса, бұл мьютекс бастапқыдан блокталған – тізбекке жататынын білдіреді.

Алғашқы екі конструктор әдетте бір үдерістегі тізбектерді синхрондау үшін қолданылады. Әртүрлі үдерістердегі тізбектерді синхрондалау біз үшін қызығушылық тудыруда. Сондықтан да біз операциялық жүйе үшін **Mutex** атау беруге мүмкіндік беретін конструктордың үшінші нұсқасын пайдаланамыз. Мысалы:

static Mutex kl = new Mutex(false, "Kljsik_1");

Оқу бағдарламасы ретінде бағдарламаның консолдық терезесіне негізгі бағдарлама (сервермен) үшін 0-ден 9-ға дейінгі сандардың мәндер кестелерін және негізгі бағдарламадан қосылатын қосымша бағдарламаның (клиентпен) 10-ден 19-ға дейін сандардың кестелерін шығаруын қарастырамыз.

Негізгі бағдарламаның коды:

```
using System;
using System.Diagnostics;
using System.Threading;
using System.Runtime.InteropServices;

namespace ConsoleApplication1
{
 class soz_pro
 {
  static void Prin(int n)
```

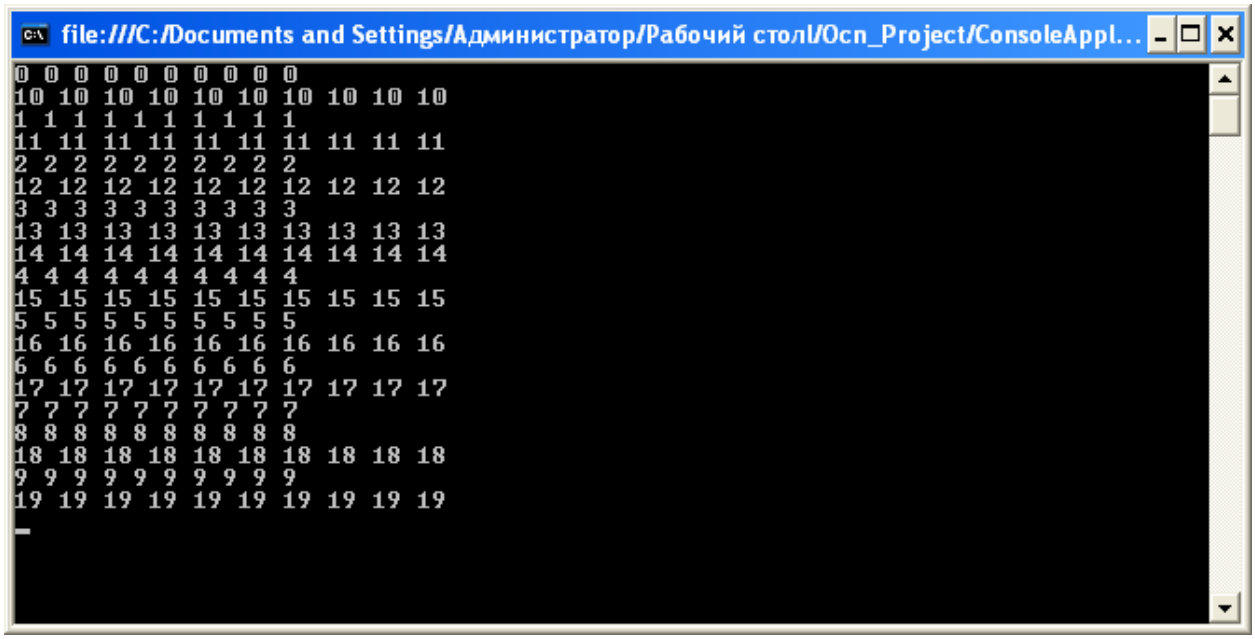
```
  {
   Console.Write(n + " ");
   Thread.Sleep(100);
  }
  static Mutex kl = new Mutex(false, "Kljsik_1");
  static void Main()
  {
   string clientExe = "Dop_pr.exe";
   var startInfo = new ProcessStartInfo(clientExe);
   startInfo.UseShellExecute = false;//в одно консольное окно
   Process p = Process.Start(startInfo);
   for (int j = 0; j < 10; j++)
   {
       kl.WaitOne();
       for (int i = 0; i < 10; i++)
       {
           Prin(j);
       }
       Console.WriteLine();
       kl.ReleaseMutex();
   }
   Console.ReadLine();
  }
 }
}
```

Қосымша бағдарламаның коды:

```
using System;
using System.Threading;

namespace ConsoleApplication1
{
 class Program
 {
  static void Prin(int n)
  {
   Console.Write(n + " ");
   Thread.Sleep(100);
  }
  static Mutex kl_2 = new Mutex(false, "Kljsik_1");
  static void Main()
  {
   int i, j;
   for (j = 10; j < 20; j++)
   {
    kl_2.WaitOne();
    for (i = 0; i < 10; i++)
         Prin(j);
    Console.WriteLine();
    kl_2.ReleaseMutex();
   }
   Console.ReadKey();
  }
 }
}
```

Негізгі және қосымша бағдарламалардың жұмысы:




11.1-сурет – Негізгі және қосымша бағдарламалардың жұмысы

Бағдарламалардың қалыпты жұмысы үшін `Dop_pr.exe` файлы бағдарламалардың қалыпты жұмысы үшін негізгі бағдарламаның `ConsoleApplication1.exe`. файлымен бір папкада болуға тиіс.

Әр бағдарламаның жұмысы консолдық терезеде көрсетіледі. Бағдарламалар атауы "Kljsik_1" мьютекстің көмегімен синхрондалған – бағдарламалар ақпаратты жолдар бойынша шығарады. Жолдың үзілуіне жол берілмейді (мьютекс осы үшін бақылау жүргізеді). Кезекті жолды шығарған соң екі бағдарлама да **Mutex**ті ала алады және келесі жолдың шығаруын жалғастыра алады.

Үдеріс атауын белгілеу кезінде var типі қолданылған:

```
var startInfo = new ProcessStartInfo(clientExe);.
```

Егерде тіл компиляторы автоматты түрде инициализациялау өрнегі бойынша типті тани алса, онда C# тілінде айнымалы типіндегі осындай тапсырма мүмкін, мысалы, жолдық айнымалының инициализациялауының келесі жазбалары эквивалентті:

string st1 = “Привет”; и

var st1 = “Привет”; .

11.2 Semaphore арнайы блоктаушы конструкциясы

Semaphore синхрондау объектісі үдерістерді синхрондау үшін арналған келесі арнайы блоктаушы конструкция болып табылады. Негізгі оның ерекшілігі – бір үдерістің де, бірнеше үдерістердің де бірнеше тізбектеріне ресурсты бір мезгілде пайдалануға рұқсат береді.

Semaphore келесі конструкторлардың көмегімен құрастыруға болады:

static Semaphore бағдарламадағы аты = new Semaphore(Сч, Max);

static Semaphore бағдарламадағы аты = new Semaphore(Сч, Max, "ОЖ-гі атау");

онда Сч – Semaphore қорғалатын ресурстарды қолданатын тізбектердің есептегіші, ал Max – Semaphore арқылы ресурстарға қатынауды алуға қабілетті

тізбектердің максималды саны. Әдетте бастапқы жағдайында тізбектер есептегіші семафор тізбектерінің максималды санына тең болады. Мысалы:

```
static Semaphore s = new Semaphore(3, 3);
```

Осы мысалда ресурстармен жұмысты бір мезгілде тек үш тізбекке рұқсат ететін Semaphore типіндегі s объектісі құрылады.

Semaphore-мен қорғалатын ресурстарға қатынауға рұқсат алу үшін WaitOne() әдісі қолданылады. Осы әдіс, егерде семафор сигналдық күйде болса – ресурстарға қатынауға рұқсат береді, тізбектер есептегішін бір бірлікке азайтады (егерде WaitOne() әдісінде басқа сан берілмесе). Егерде ресурстармен жұмыс істейтін тізбектердің саны семафор объектісінде белгіленген тізбектердің максималды санынан аз болса (тізбектер есептегіші нөлге тең немесе үлкен), онда семафор сигналдық күйде қалады. Ресурспен жұмыс істеуге қызығушылық танытқан тізбектер саны рұқсат берілгеннен көп бола бастаса, семафор ресурсқа қатынауды блоктайды. Release() әдісімен семафорды блоктаудан шығаруға болады. Осы әдіс семафор тізбектерінің есептегішін бір бірлікке ұлғайтады (егерде Release() әдісінде басқа сан берілмесе), ал бұл семафорды автоматты түрде сигналдық күйге ауыстырады және ол тізбектерге қол жетімді болып қалады.

Сонымен семафор объектісі бағдарлама ресурстарына көптеген тізбектердің қатынауын синхрондауға мүмкіндік береді.

Көптеген тізбектердің бағдарлама ресурстарына қатынауын синхрондау мысалы ретінде [Албахари б.748] алынған келесі бағдарламаның жұмысын қарастырамыз.

```
using System;
using System.Threading;

namespace ConsoleApplication1
{
 class Program
 {
  static Semaphore s = new Semaphore(3, 3);
  static void Main()
  {
   for (int i = 0; i < 10; i++)
       new Thread(Go).Start(i);
   Console.WriteLine("Работа цикла запуска нитей закончена!");
   Console.ReadKey();
  }
  static void Go(object id)
  {
   s.WaitOne();
   Console.WriteLine(id + " <- нить начинает работу!");
   Thread.Sleep(1000 * (int)id);
   Console.WriteLine(id + " -> нить закончила работу");
   s.Release();
  }
 }
}
```

Работа программы:

2 <- нить начинает работу!

4 <- нить начинает работу!

Работа цикла запуска нитей закончена!

```
5 <- нить начинает работу!
2 -> нить закончила работу
9 <- нить начинает работу!
4 -> нить закончила работу
0 <- нить начинает работу!
0 -> нить закончила работу
6 <- нить начинает работу!
5 -> нить закончила работу
1 <- нить начинает работу!
1 -> нить закончила работу
3 <- нить начинает работу!
3 -> нить закончила работу
8 <- нить начинает работу!
6 -> нить закончила работу
7 <- нить начинает работу!
9 -> нить закончила работу
7 -> нить закончила работу
8 -> нить закончила работу
```

Бағдарлама жұмысының нәтижесі бойынша бір мезетте бағдарлама ресурстарымен тек үш тізбек жұмыс істей алатынын көреміз. Егерде тізбектердің есептегіші және тізбектердің масималды санын бірге тең деп берсе, семафор объектісі мьютекске «ауысады». Қалған жағдайларда семафор объектісінде ұқсас объект жоқ.

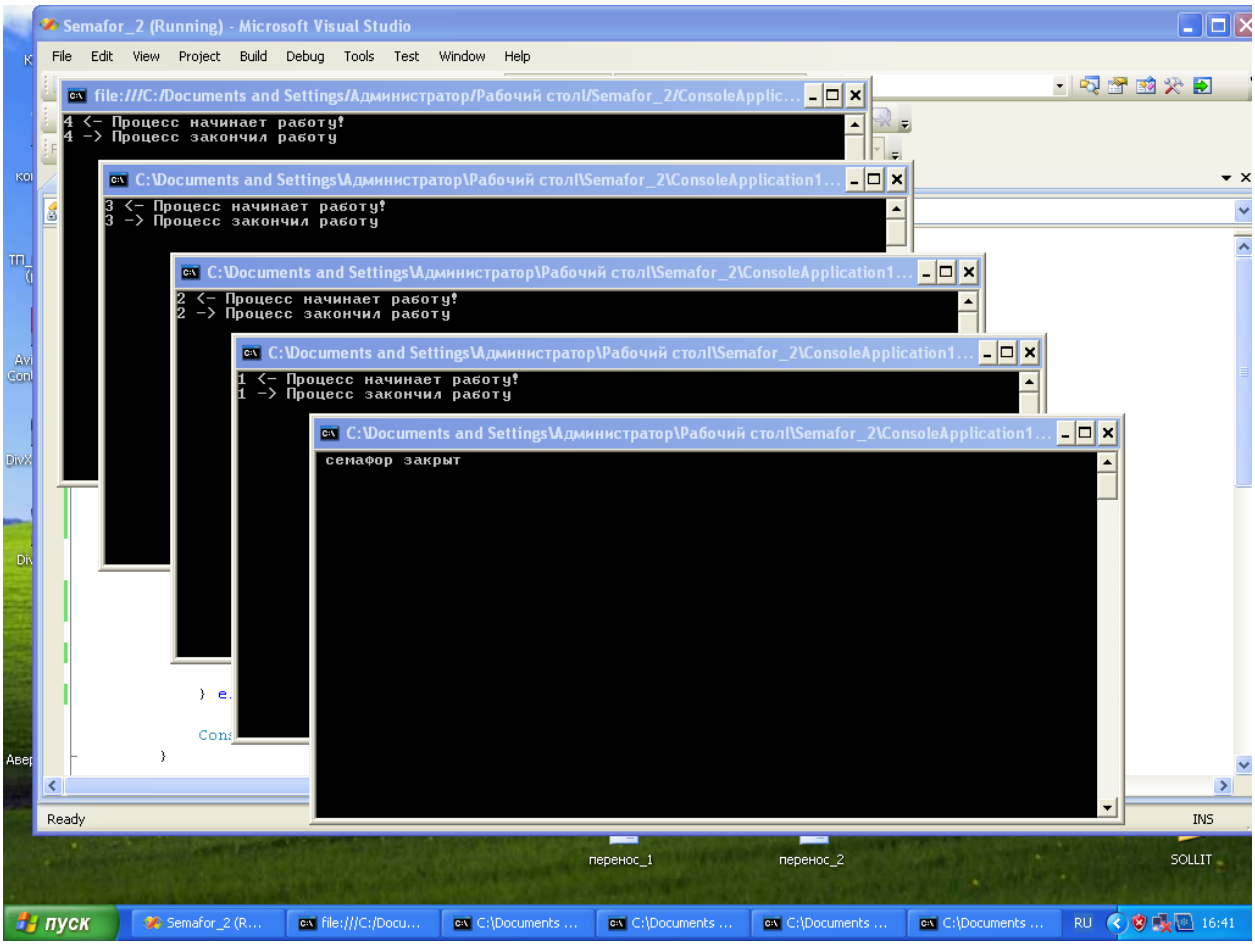
Семафор объектісінің әртүрлі үдерістермен жұмысын көрсету үшін бағдарлама өзін-өзі семафор мүмкіндіктері таусылғанға дейін қосатын таза оқу бағдарламасын пайдаланамыз.

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

namespace ConsoleApplication1
{
 class Program
 {
  static Semaphore s = new Semaphore(5, 5,"sem_1");
  static void Main()
  {
   int kol;
   s.WaitOne(); kol = s.Release();
   if(kol>=1)
   {
    Process.Start("ConsoleApplication1");
    s.WaitOne();
    Console.WriteLine(kol + " <- Процесс начинает работу!");
    Thread.Sleep(1000*5);
    Console.WriteLine(kol + " -> Процесс закончил работу");
    s.Release();
   } else Console.WriteLine(" семафор закрыт ");
   Console.ReadKey();
```

```
    }
  }
}
```

Бағдарлама жұмысы:




11.1-сурет –Semaphore көмегімен ресурстарды үлестіру

[Албахари б.747] кітабында семафор объектісінің жұмысы бейнеленген түрде түнгі клуб жұмысымен салыстырылады.

«Semaphore түнгі клубқа ұқсас – ол дөкей /вышибала/ қамтамасыз ететін белгілі сыйымдылққа ие. Толған соң ешкім түнгі клубқа кіре алмайды, сыртта кезек жинақталады. Әрі қарай, бір адам клубтан кетсе, кезек басындағы біреу клубқа кіре алады. Semaphore конструкторы минимум екі параметрді қабылдайды - әлі қол жетімді орындар саны және түнгі клубтың жалпы сыйымдылығы».

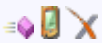
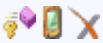
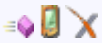
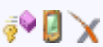
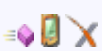
Қосымша

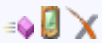
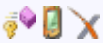
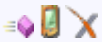
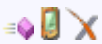
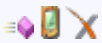
**Мьютекс**

**Конструкторы**

| | Имя | Описание |
|---|---|---|
| | Mutex | Перегружен. Инициализирует новый экземпляр класса Mutex. |

**Методы**

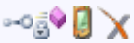
| | Имя | Описание |
|---|---|---|
|  | Close | При переопределении в производном классе освобождает все ресурсы, занимаемые текущим объектом WaitHandle. (Унаследовано от WaitHandle.)<br><br>В .NET Compact Framework этот член переопределяется Close(). <br><br>В XNA Framework этот член переопределяется Close(). |
|  | CreateObjRef | Создает объект, который содержит всю необходимую информацию для создания прокси-сервера, используемого для взаимодействия с удаленным объектом. (Унаследовано от MarshalByRefObject.) |
|  | Dispose | При переопределении в производном классе освобождает неуправляемые ресурсы, используемые объектом WaitHandle, и (необязательно) освобождает управляемые ресурсы. (Унаследовано от WaitHandle.) |
|  | Equals | Определяет, равен ли заданный объект Object текущему объекту Object. (Унаследовано от Object.) |
|  | Finalize | Позволяет объекту Object попытаться освободить ресурсы и выполнить другие операции очистки, перед тем как объект Object будет утилизирован в процессе сборки мусора. (Унаследовано от Object.) |
|  | GetAccessControl | Получает объект MutexSecurity, представляющий безопасность управления доступом для именованного мьютекса. |
|  | GetHashCode | Играет роль хэш-функции для |

| | | |
|---|---|---|
| | | определенного типа. (Унаследовано от Object.) |
|  | GetLifetimeService | Извлекает объект обслуживания во время существования, который управляет политикой времени существования данного экземпляра. (Унаследовано от MarshalByRefObject.) |
|  | GetType | Возвращает объект Type для текущего экземпляра. (Унаследовано от Object.) |
|  | InitializeLifetimeService | Возвращает объект обслуживания во время существования для управления политикой времени существования данного экземпляра. (Унаследовано от MarshalByRefObject.) |
|  | MemberwiseClone | Перегружен. |
|  | OpenExisting | Перегружен. Открывает существующий именованный мьютекс. |
|  | ReleaseMutex | Освобождает объект Mutex один раз. |
|  | SetAccessControl | Задает безопасность управления доступом для именованного системного мьютекса. |
|  | ToString | Возвращает объект String, который представляет текущий объект Object. (Унаследовано от Object.) |
|  | WaitOne | Перегружен. |

**Свойства**

| | Имя | Описание |
|---|---|---|
|  | Handle | **Устаревшее.** Получает или задает собственный дескриптор операционной системы. (Унаследовано от WaitHandle.) |
|  | SafeWaitHandle | Получает или задает собственный дескриптор операционной системы. (Унаследовано от WaitHandle.) |

**Явные реализации интерфейса**

| | Имя | Описание |
|---|---|---|
|  | IDisposable.Dispose | Инфраструктура. Освобождает все ресурсы, используемые объектом WaitHandle. (Унаследовано от WaitHandle.) |

Универсальный тип Semaphore предоставляет следующий члены.

**Конструкторы**

| | Имя | Описание |
|---|---|---|
|  | Semaphore | Перегружен. Инициализирует новый экземпляр класса Semaphore. |

**Методы**

| | Имя | Описание |
|---|---|---|
|  | Close | При переопределении в производном классе освобождает все ресурсы, занимаемые текущим объектом WaitHandle. (Унаследовано от WaitHandle.) |
|  | CreateObjRef | Создает объект, который содержит всю необходимую информацию для создания прокси-сервера, используемого для взаимодействия с удаленным объектом. (Унаследовано от MarshalByRefObject.) |
|  | Dispose | При переопределении в производном классе освобождает неуправляемые ресурсы, используемые объектом WaitHandle, и (необязательно) освобождает управляемые ресурсы. (Унаследовано от WaitHandle.) |
|  | Equals | Определяет, равен ли заданный объект Object текущему объекту Object. (Унаследовано от Object.) |
|  | Finalize | Позволяет объекту Object попытаться освободить ресурсы и выполнить другие операции очистки, перед тем как объект Object будет утилизирован в процессе |

| | | |
|---|---|---|
| | | сборки мусора. (Унаследовано от Object.) |
| | GetAccessControl | Возвращает настройки управления доступом для именованного системного семафора. |
| | GetHashCode | Играет роль хэш-функции для определенного типа. (Унаследовано от Object.) |
| | GetLifetimeService | Извлекает объект обслуживания во время существования, который управляет политикой времени существования данного экземпляра. (Унаследовано от MarshalByRefObject.) |
| | GetType | Возвращает объект Type для текущего экземпляра. (Унаследовано от Object.) |
| | InitializeLifetimeService | Возвращает объект обслуживания во время существования для управления политикой времени существования данного экземпляра. (Унаследовано от MarshalByRefObject.) |
| | MemberwiseClone | Перегружен. |
| | OpenExisting | Перегружен. Открытие существующего именованного семафора. |
| | Release | Перегружен. Выполняет выход из семафора. |
| | SetAccessControl | Задает настройки управления доступом для именованного системного семафора. |
| | ToString | Возвращает объект String, который представляет текущий объект Object. (Унаследовано от Object.) |
| | WaitOne | Перегружен. |

**Свойства**

| | Имя | Описание |
|---|---|---|
| | Handle | **Устаревшее.** Получает или задает собственный дескриптор операционной системы. |

| | | |
|---|---|---|
| | | (Унаследовано от WaitHandle.) |
|  | SafeWaitHandle | Получает или задает собственный дескриптор операционной системы. (Унаследовано от WaitHandle.) |

**Явные реализации интерфейса**

| | Имя | Описание |
|---|---|---|
|  | IDisposable.Dispose | Инфраструктура. Освобождает все ресурсы, используемые объектом WaitHandle. (Унаследовано от WaitHandle.) |