Лабораториялық жұмыс 4. Ағындарды синхронизациялау құралдарын пайдалану.

Лабораториялық жұмыстың мақсаты: Студенттерде ағындарды синхронизациялау құралдарын таңдау және пайдалану дағдыларын қалыптастыру.

Лабораториялық жұмысты орындау нәтижесінде студенттер келесі қабілеттерге ие болады:

- Есептің берілгеніне байланысты синхронизациялаудың құралын таңдау;
- Синхронизациялау құралын пайдалану.

Тапсырма: Төменде берілген мысалдармен танысып, нұсқа бойынша берілген тапсырмаларда ағындар арасында синхронизация орнату үшін екі синхронизациялау тәсілін таңдап, оларды пайдаланыңыз. Синхронизациялаудың екі тәсілін пайдалану нәтижелеріне баға беріңіз.

C# тілінің кірістірілген бұғаттау тәсілі

Мысал 1. Lock көмегімен бұғаттау.

```
using System;
using System.Threading;
class SumArray {
int sum;
object lockOn = new object(); // бұғаттауға қажетті жабық объект
public int SumIt(int[] nums) {
lock(lockOn) { // әдісті бұғаттау
sum = 0;
for(int i=0; i < nums.Length; i++) {
sum += nums[i];
Console.WriteLine(Thread.CurrentThread.Name + " ағыны үшін аралық қосынды = " + sum);
Thread.Sleep(10); // тапсырмаларды ауыстыруға рұқсат беру
}
return sum;
}
}
}
class MyThread {
public Thread Thrd;
int[] a;
int answer;
// MyThread класының барлық экземплярлары үшін SumArray типті жалғыз объект құру
static SumArray sa = new SumArray();
// Жаңа ағын құру
public MyThread(string name, int[] nums) {
a = nums;
Thrd = new Thread(this.Run);
Thrd.Name = name;
Thrd.Start(); // ағынды іске қосу
}
// Жаңа ағынның атқарылуын бастау.
void Run() {
```

```
Console.WriteLine(Thrd.Name + " басталды.");
answer = sa.SumIt(a);
Console.WriteLine(Thrd.Name + " ағыны үшін қосынды: " + answer);
Console.WriteLine(Thrd.Name + " аяқталды.");
}
}
class Sync {
static void Main() {
int[] a = {1, 2, 3, 4, 5);
MyThread mt1 = new MyThread("#1 буын", a);
MyThread mt2 = new MyThread("#2 буын", a);
mt1.Thrd.Join();
mt2.Thrd.Join();
}
}
```

Monitor класын пайдаланып бұғаттауда хабарламалармен алмасу тәсілі

Мысал 2. Monitor класын бұғаттау барысында пайдалану.

```
using System;
using System.Threading;
class TickTock {
object lockOn = new object();  // бұғаттауға қажетті жабық объект
public void Tick(bool running) {
lock(lockOn) { // әдісті бұғаттау
if(!running) { // сағатты тоқтату
Monitor.Pulse(lockOn); // күтудегі ағындарға хабарлау
return;
}
Console.Write("тик ");
Monitor.Pulse(lockOn); // Tock() әдісінің орындалуына рұқсат беру
Monitor.Wait(lockOn); // Tock() әдісінің аяқталуын күту
}
}
public void Tock(bool running) {
lock(lockOn) { // әдісті бұғаттау
if(!running) { // сағатты тоқтату
Monitor.Pulse(lockOn); // күтудегі ағындарға хабарлау
return;
}
Console.WriteLine("так");
Monitor.Pulse(lockOn); // Tick()әдісінің орындалуына рұқсат беру
Monitor.Wait(lockOn); // Tick()әдісінің аяқталуын күту
}
}
}
class MyThread {
public Thread Thrd;
TickTock ttOb;
// Жаңа ағын құру
public MyThread(string name, TickTock tt) {
Thrd = new Thread(this.Run);
```

```
ttOb = tt;
Thrd.Name = name;
Thrd.Start();
}
// Жаңа ағын жұмысын бастау.
void Run() {
if(Thrd.Name == "Tick") {
for(int i=0; i<5; i++) ttOb.Tick(true);
ttOb.Tick(false);
}
else {
for(int i=0; i<5; i++) ttOb.Tock(true);
ttOb.Tock(false);
}
}
}
class TickingClock {
static void Main() {
TickTock tt = new TickTock();
MyThread mt1 = new MyThread("Tick", tt);
MyThread mt2 = new MyThread("Tock", tt);
mt1.Thrd.Join();
mt2.Thrd.Join();
Console.WriteLine("Сағат тоқтатылды");
}
}
```

Мьютексті пайдалану

Мысал 3. Мьютекс көмегімен бұғаттау.

```
using System;
using System.Threading;
class SharedRes {
public static int Count = 0; // ортақ қолданылатын ресурс
public static Mutex Mtx = new Mutex(); // ортақ қолданылатын ресурсқа қол жеткізуді
                                       // басқаратын мьютекс
}
// Бұл ағында SharedRes.Count айнымалысы инкременттеледі
class IncThread {
int num;
public Thread Thrd;
public IncThread(string name, int n) {
Thrd = new Thread(this.Run);
num = n;
Thrd.Name = name;
Thrd.Start();
}
// ағынға кіру нүктесі.
void Run() {
Console.WriteLine(Thrd.Name + " мьютексті күтуде.");
// Получить мьютекс.
SharedRes.Mtx.WaitOne();
```

```
Console.WriteLine(Thrd.Name + " мьютексті қабылдады.");
do {
Thread.Sleep(500);
SharedRes.Count++;
Console.WriteLine(Thrd.Name + " ағынында, SharedRes.Count = " + SharedRes.Count);
num--;
} while(num > 0);
Console.WriteLine(Thrd.Name + " мьютексті босатты.");
// Мьютексті босату.
SharedRes.Mtx.ReleaseMutex();
}
}
//Бұл ағында SharedRes.Count айнымалысы декременттеледі.
class DecThread {
int num;
public Thread Thrd;
public DecThread(string name, int n) {
Thrd = new Thread(new ThreadStart(this.Run));
num = n;
Thrd.Name = name;
Thrd.Start();
}
// ағынға кіру нүктесі.
void Run() {
Console.WriteLine(Thrd.Name + " мьютексті күтуде.");
// Получить мьютекс.
SharedRes.Mtx.WaitOne();
Console.WriteLine(Thrd.Name + " мьютексті қабылдады.");
do {
Thread.Sleep(500);
SharedRes.Count--;
Console.WriteLine(Thrd.Name + " ағынында, SharedRes.Count = " + SharedRes.Count);
num--;
} while(num > 0);
Console.WriteLine(Thrd.Name + " мьютексті босатты.");
// Мьютексті босату.
SharedRes.Mtx.ReleaseMutex();
}
}
class MutexDemo {
static void Main() {
// Екі ағынды құру
IncThread mt1 = new IncThread("Инкременттеуші ағын", 5);
Thread.Sleep(1); // ағынды іске қосу
DecThread mt2 = new DecThread("Декременттеуші ағын", 5);
mt1.Thrd.Join();
mt2.Thrd.Join();
}
}
```

Семафорды пайдалану

Мысал 4. Семафор көмегімен бұғаттау.

```
using System;
using System.Threading;
// Келесі ағын өзінің тек екі экземплярын қатар орындауға рұқсат береді
class MyThread {
public Thread Thrd;
// Семафор құрылады, ол бастапқыда бар 2 рұқсаттың екеуін береді
static Semaphore sem = new Semaphore(2, 2);
public MyThread(string name) {
Thrd = new Thread(this.Run);
Thrd.Name = name;
Thrd.Start();
}
// Ағынға кіру нүктесі.
void Run() {
Console.WriteLine(Thrd.Name + " рұқсатты күтуде.");
sem.WaitOne();
Console.WriteLine(Thrd.Name + " рұқсатты алды.");
for(char ch='A'; ch < 'D'; ch++) {
Console.WriteLine(Thrd.Name + " : " + ch + " ");
Thread.Sleep(500);
}
Console.WriteLine(Thrd.Name + " рұқсатты босатты.");
// Семафорды босату.
sem.Release();
}
}
class SemaphoreDemo {
static void Main() {
// Үш ағын құру.
MyThread mt1 = new MyThread("#1 ағын");
MyThread mt2 = new MyThread("#2 ағын");
MyThread mt3 = new MyThread("#3 ағын");
mt1.Thrd.Join();
mt2.Thrd.Join();
mt3.Thrd.Join();
}
}
```

Тапсырмалар

1-нұсқа

Матрицаны векторға көбейтуді орындаңыз. Матрицаның әрбір жолы жеке ағында өңделуі тиіс.

2-нұсқа

Берілген аралыққа жататын барлық жай сандарды табыңыз. Тапсырманы орындау үшін классикалық Евклид алгоритмін қолданыңыз. Берілген аралықты кіші аралықтарға бөліп, әрбір ішкі аралықты жеке ағында өңдеңіз.

3-нұсқа

Аю мен аралардың өзара байланысу процесін бейнелейтін программа құрыңыз. N араның әрқайсысы бал жинауға қатысады: бір рет бал жинауға шыққанда барлығы бірдей мөлшерде (M1) бал жинайды және оған кездейсоқ уақыт жұмсайды. Аю X уақыт ішінде M2 мөлшерінде балмен қоректенеді және келесі X уақытта азықсыз тіршілік ете алады. Әрбір араның жұмысын жеке ағында жүзеге асырыңыз.

4-нұсқа

Шарлардың қозғалысын бейнелейтін программа құрыңыз. N шар берілген. Олардың тік және көлденең координаталары кездейсоқ шамаларға өзгереді. Егер шар берілген аймақ шекарасынан төмен түссе, жоғалып кетеді. Әрбір шардың координатасының өзгеруін жеке ағында жүзеге асырыңыз.

5-нұсқа

Топтардың қарсыласуын бейнелейтін программа құрыңыз. Ойыншылардың N тобы бар. Әрбір топтың ойыншылар саны кездейсоқ шамаға артады және қарсылас топтың кездейсоқ ойыншылар санын жояды. Әрбір топтар жұбының өзара қарсыластығы жеке ағында жүзеге асырылуы тиіс.

6-нұсқа

Бақылау суммасы. Өлшемдері әртүрлі N файл берілген. Әрбір файл үшін бақылау суммасын (файлдың барлық символдары кодтарының қосындысын) анықтау керек. Әрбір файл жеке ағында өңделуі тиіс.

7-нұсқа

Кедергілі жүгірісті бейнелейтін программа құрыңыз. Жүгіруге арналған трассаның матрица түріндегі шартты картасы жасалады. Матрицаның ені жүгірушілер санына тең, ал биіктігі бекітілген, кездейсоқ ұяшықтарға орналастырылған кездейсоқ кедергілер саны бейнеленеді. Жүгірушілер трасса бойымен жылжи отырып, кедергіге кезіккен жағдайда нақты анықталған уақытқа бөгеледі. Мәреге жеткен жүгірушілер өздерінің нөмірлерін хабарлайды. Әрбір жүгірушінің жұмысын жеке ағында жүзеге асырыңыз.

8-нұсқа

Қойлар мен қасқыр ойынын бейнелейтін программа құрыңыз. Бірнеше қой мен қасқырдың қозғалыстарын бейнелейтін программа құру қажет. Қасқыр мен қойдың координаталары сәйкес келген жағдайда, қой жоғалады. Егер екі қойдың координаталары сәйкес келсе, жаңа қой пайда болады. Қасқыр мен қойлар кездейсоқ қозғалады. Әрбір қойдың қозғалысын жеке ағында жүзеге асырыңыз.

9-нұсқа

$Y=23*x^2-33$ функциясының мәндерін x=0.01 қадамымен есептеуді орындаңыз. Есептелген мәндер x мәндерімен қатар жиымға жазылып отыруы тиіс. Жиымға жазылған x және y мәндерін экранға шығару керек. Мәндерді есептеу және жиымға жазу бір ағында, мәндерді жиымнан оқып, экранға шығару жеке ағында орындалуы тиіс.

10-нұсқа

Мәліметтер жиымын сұрыптау және сұрыптау күйін экранда бейнелеу. Бірінші ағында жиымды өсу реті бойынша, екінші ағында кему реті бойынша сұрыптау орындалуы тиіс. Әрбір элементтің орны ауысқанда экранда жиымның ағымдағы күйі бейнеленеді.

11-нұсқа

0 мен 9 аралығынан кездейсоқ сандарды генерациялайтын 3 ағын құрыңыз. Белгілі бір батырманы басқанда генерациялау тоқтатылып, генерацияланған сандар тізбектерінен келесідей ішкі тізбектерді іздеу қажет: қатар орналасқан үш бірдей сан, қатар орналасқан екі бірдей сан. Әрбір ағындық тізбек үшін осы ішкі тізбектер санын анықтаңыз.

12-нұсқа

Берілген аралықтан Фибоначчи сандарын іздейтін ағынды және жай сандарды іздейтін ағынды құру қажет. Сандар тізбектері екі жеке файлға жазылады, экранға сандар тізбектері және олардың мөлшері шығарылады.

13-нұсқа

Берілген файл құрамынан берілген тіркесті іздеу программасын құрыңыз. Әрбір файлдың қатары жеке ағымда өңделуі тиіс.

14-нұсқа

Файлға кездейсоқ мәліметтер жазып, оларды оқып, экранға шығару программасын құрыңыз. Файлға мәліметтерді жазу мен оларды оқып, экранға шығару екі жеке ағында орындалуы тиіс.