14 дәріс. Анонимдік функциялар. Анонимдік әдістер және лямбда-өрнектер.

Дәрістің мақсаты: студенттерде анонимдік функцияларды пайдалану ерекшеліктері туралы түсініктерін көрсетуге қабілет қалыптастыру.

Осы дәрісті меңгеру нәтижесінде студенттер келесі қабілеттерге ие болады:

- Анонимдік әдістерді пайдалану ерекшеліктері туралы түсініктерін көрсету;
- Лямбда-өрнектерді пайдалану ерекшеліктері туралы түсініктерін көрсету.

Анонимдік функциялар

Делегат сілтеме жасап тұратын әдіс көбінесе осы мақсат үшін қолданылады. Басқаша айтқанда, әдістің қолданылуының негізгі себебі ол тек қана делегат арқылы шақырылады, ал әдістің өзі еш уақытта шақырылмайды. Осындай жағдайларда, жеке әдіс құрмай-ақ, *анонимдік* (жасырын) *функцияны* қолдануға болады. Анонимдік функция, шын мәнінде, делегат конструкторына берілетін атаусыз кодтық блок болып табылады. Ашып айтар болсақ, анонимдік функцияның артықшылығы оның қарапайымдылығында. Соның арқасында, ең басты қызметі делегатқа берілуі болып табылатын жеке әдісті жариялауға болады.

C# тілінің 3.0 нұсқасын бастап, анонимдік функцияның екі түрі қарастырылды: *анонимдік әдістер* және *лямбда-өрнектер*. Анонимдік әдістер C# тілінің 2.0 нұсқасына енгізілсе, лямбда-өрнектер оның 3.0 нұсқасында пайда болған еді. Жалпы айтқанда, лямбда-өрнек анонимдік әдістің жұмыс істеу қағидасын жақсартады және қазіргі уақытта анонимдік функциясын құру үшін ол қолайлырақ болып табылады. Бірақ, анонимдік әдістер C# тілі кодтарында кең қолданылады, сондықтан да олар C# тілінің маңызды құрамдас бөлігі болып табылады.

Анонимдік әдістер лямбда-өрнектерден бұрын пайда болғандықтан, оларды қарастыру лямбда-өрнектердің ерекшелігін дұрыс түсінуге мүмкіндік береді. Сонымен қатар, анонимдік әдістерді, лямбда-өрнектерді қолдануға болмайтын көптеген жағдайларда, пайдалануға болады. Сондықтан да осы тарауда анонимдік әдістер де және лямбда-өрнектер де қатар қарастырылады.

Анонимдік әдістер

Анонимдік әдіс — делегаттың нақты экземплярымен байланысты кодтың атаусыз блогын құрудың бір тәсілі.

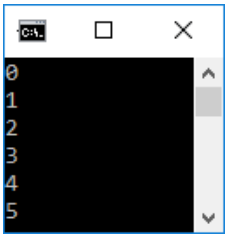
Анонимдік әдісті құру үшін delegate түйінді сөзінен кейін код блогын көрсету жеткілікті. Мұның қалай құрылатынын нақты мысалмен көрсетейік. Төменде көрсетілген программада анонимдік әдіс 0-ден 5-ке дейін санауды ұйымдастыру үшін қолданылады.

```
// Анонимдік әдістің қолданылуы

using System;

// Делегат типін жариялау.
delegate void CountIt();

class AnonMethDemo
{
    static void Main()
    {

        // Ары қарай анонимдік әдіс ретінде делегатқа берілетін
```

```
        // сандарды санайтын код орналасады.
        CountIt count = delegate
        {
            // Бұл кодтық блок делегатқа беріледі.
            for (int i = 0; i <= 5; i++)
                Console.WriteLine(i);
        };  // нүктелі үтірге назар аударыңыз

        count();
    }
}
```

Бұл программада, алдымен, void типін қайтаратын, параметрсіз Countlt делегат типі жарияланады. Содан соң Main() әдісінде Countlt делегатының count экземпляры құрылады, оған delegate түйінді сөзінен кейін орналасқан код блогы беріледі. Дәл осы код блогы count делагатын қолданған кезде орындалатын анонимдік әдіс болып табылады. Код блогынан кейін тұрған нүктелі үтірге назар аударыңыз, ол жариялау операторын нақты түрде аяқтайтын белгі болып табылады. Төменде жоғарыдағы программа нәтижесі келтірілген.

```
0
1
2
3
4
5
```



Аргументтердің анонимдік әдіске берілуі

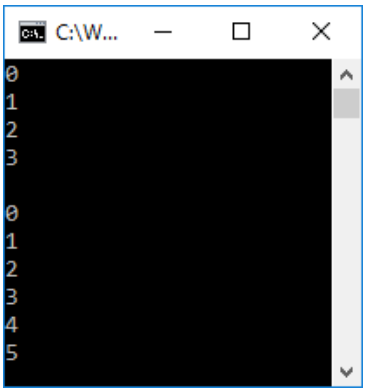
Анонимдік әдіске бір немесе бірнеше аргумент беруге болады. Ол үшін delegate түйінді сөзінен кейін жақша ішінде параметрлер тізімін көрсету жеткілікті, ал мұнан кейін делегат экземплярын қолданғанда, оған тізімге сәйкес аргументтер беріледі. Төменде мысал ретінде алдыңғы программаның өзгертілген нұсқасы келтірілген, онда аргумент ретінде саналуға тиіс мәндер берілген.

```
//Аргумент қабылдайтын анонимдік әдістің қолданылуын көрсету

using System;

// Енді CountIt делегатының параметрі бар екеніне назар аударыңыз
delegate void Countlt(int end);

class AnonMethDemo2 {

   static void Main() {

        // Есептің соңғы шешімі анонимді әдісге беріледі.
        Countlt count = delegate(int end)
        {
            for (int i = 0; i <= end; i++)
                Console.WriteLine(i);
        };

        count(3);
        Console.WriteLine();
        count(5);
    }
```

```
}
```

Бұл программа нұсқасында CountIt делегаты бүтін сан түріндегі аргументті қабылдайды. Анонимдік әдісті құрғанда параметрлер тізімі delegate түйінді сөзінен кейін көрсетілетініне назар аудару керек. End параметрі аты бар әдісті құрғандағы сияқты анонимді әдісте де код үшін қолжетімді болады.Төменде программаның орындалу нәтижесі көрсетілген.

```
0
1
2
3

0
1
2
3
4
5
```



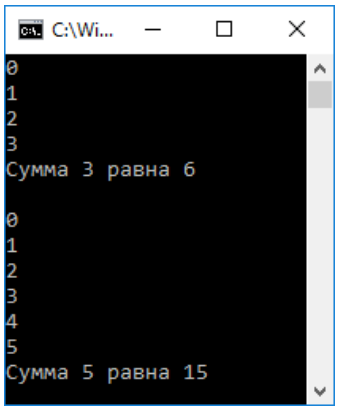
Анонимдік әдістен мән қайтару

Анонимдік әдіс мән қайтара алады. Ол үшін аты бар әдістердегі сияқты return операторы қолданылады. Қайтарылған мәннің типі делегатты жариялағанда көрсетілген қайтарылатын типпен сәйкес келуі керек. Мысал ретінде төменде қосындылау арқылы санау жүргізіп, нәтиже қайтаратын программа коды көрсетілген.

```
//Мән қайтаратын анонимдік әдістің қолданылуы.

using System;

//  Бұл делегат мән қайтарады
delegate int Countlt(int end);

class AnonMethDemo3 {

   static void Main() {
     int result;

     // Санауға арналған мәндер анонимдік әдіске беріледі.
     // Саналған мәндер қосындысы қайтарылады
     Countlt count = delegate (int end) {
     int sum = 0;

     for(int i=0; i <= end; i++) {
         Console.WriteLine (i);
      sum += i;
     }
     return sum;  // анонимдік әдістен мән қайтарылады
     };

     result = count(3);
     Console.WriteLine("Сумма 3 равна " + result);
     Console.WriteLine ();

     result = count(5);
     Console.WriteLine("Сумма 5 равна " + result);
  }
}
```

Программаның бұл нұсқасында қосындының мәні count делегатының экземплярымен байланысты код блогымен қайтарылады. Мұндағы анонимдік әдісте return операторының аты бар әдістегідей қолданылатынына назар аударыңыз.Төменде осы кодтың нәтижесі көрсетілген.

```
0
1
2
3
Сумма 3 равна 6

0
1
2
```



3
4
5
Сумма 5 равна 15

Лямбда-өрнектер

Анонимдік әдістердің құндылығына қарамастан, олардың орнына мүмкіндіктері мол лямбда-өрнектер келді. Лямбда-өрнек C# тілінің 1.0 нұсқасынан бастап, енгізілген мәні зор жаңалықтардың бірі болды. Ол жаңа синтаксистік элементке негізделіп жасалған және анонимдік әдіске қарағанда, анағұрлым тиімді балама (альтернатива) болып табылады. Лямбда-өрнек артықшылықтары LINQ-мен (ол туралы 19-тарауда) жұмыс жасағанда жақсы білінгенмен, олар делегаттармен де және оқиғалармен де бірге жиі қолданылады. Енді осы туралы айтатын боламыз.

Лямбда-өрнек – бұл анонимдік функцияларды құрудың жаңа түрі. (Оның анонимдік әдіс ретінде қолданылуы бұдан бұрын айтылды.) Сол себепті ол делегатқа меншіктеле алады. Лямбда-өрнек оған ұқсас анонимдік әдіске қарағанда, тиімді болғандықтан, көбінесе осы өрнекті қолдану жиі ұсынылып жүр.

Лямбда-оператор

Барлық лямбда-өрнектерде лямбда-өрнекті екі бөлікке бөлетін жаңа => лямбда-оператор қолданылады. Оның сол жақ бөлігінде кіріс параметрі (немесе бірнеше параметр), ал оң жақ бөлігінде – лямбда-өрнектің тұлғасы (өз денесі) көрсетіледі. Кейде => операторы «ауысады» немесе «болып шығады» деген сөздерімен сипатталады.

C# тілінде лямбда-өрнектің, өз ішкі мазмұнына байланысты, екі түрі сүйемелденеді. Егер лямбда-өрнектің денесі бір өрнектен ғана тұратын болса, *жеке лямбда-өрнек* құрылады. Мұндай жағдайда өрнек жүйелі (фигуралы) жақшаға алынбайды. Ал егер лямбда-өрнектің денесі жүйелі жақшаларға алынған операторлар блогынан тұрса, онда *блоктық лямбда-өрнек* құрылады. Мұндайда блоктық лямбда-өрнек бірнеше операторлар тізбегінен: циклдерден, if шартты операторларынан және әдістерді шақырулардан да тұратын болды. Енді лямбда-өрнектердің осы екі түрін жеке-жеке қарастырып шығамыз.

Жеке лямбда-өрнектер

Жеке лямбда-өрнектегі => операторының оң жағындағы бөлігі сол жағында көрсетілген параметрге (бірнеше параметрлерге) әсер етеді. Мұндай өрнектің есептеліп қайтарылатын нәтижесі лямбда-операторды орындалу нәтижесі болып табылады.

Төменде жалғыз параметрді қабылдайтын жеке лямбда-өрнектің жалпы жазылу формасы көрсетілген.

параметр => өрнек

Егер бірнеше параметрді көрсету керек болса келесі форма қолданылады:

(параметрлер_тізімі) => өрнек

Осылайша екі немесе одан көп параметрді көрсету үшін, оларды жақшаға алу қажет. Егер өрнек параметрді қажет етпесе, іші бос жақшаларды қолдану керек.

Төменде жеке лямбда-өрнектің қарапайым мысалдары көрсетілген:

```
count => count+2
```

Бұл өрнекте count айнымалысы count + 2 өрнегі әсер ететін параметр болып табылады, осының нәтижесінде count операторының мәні екіге артады. Төменде жеке лямбда-өрнектің тағы бір мысалы көрсетілген.

```
n => n % 2 == 0
```

Мұндағы жағдайда егер n параметрінің сандық мәні жұп сан болса, өрнек true логикалық мәнін қайтарады, кері жағдайда false логикалық мәнін қайтарады.

Лямбда-өрнек екі кезеңде қолданылады. Алдымен лямбда-өрнекпен үйлесімді болып келетін делегат типі жарияланады, сонан соң лямбда-өрнек меншіктелетін делегат экземпляры жарияланады. Оның есептелген нәтижесі қайтарылатын мән болып табылады.

Төмендегі программа мысалында екі жеке лямбда-өрнектің қолданылуы көрсетілген. Алдымен программада делегаттың екі типі жарияланады. Олардың біріншісі int типті аргументті қабылдап алып, сол типтегі нәтижені қайтаратын Incr делегаты болып табылады. Ал, екіншісі int типті аргументті қабылдайтын, бірақ bool типті нәтиже қайтаратын IsEven делегаты болып келеді. Сонан кейін осы делегаттар экземплярларына жеке лямбда-өрнектер меншіктеледі. Соңында лямбда-өрнек делегаттардың соларға сәйкес экземпляларының көмегімен есептеледі.

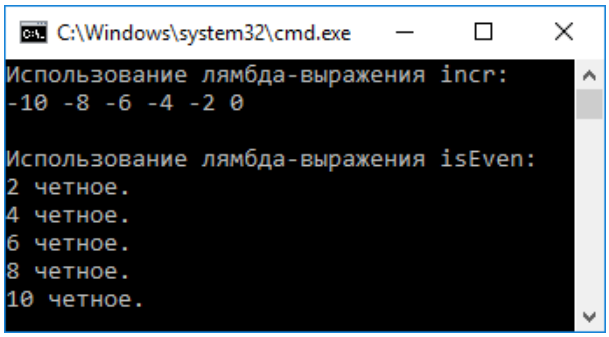
```
// Екі жеке лямбда-өрнекті пайдалану.

using System;

// int типін қабылдайтын және int типін қайтаратын делегат жариялау.
delegate int Incr(int v);

// int типін қабылдайтын және bool типін қайтаратын делегат жариялау.
delegate bool IsEven(int v);

class SimpleLambdaDemo {

   static void Main()  {

    // Лямбда-өрнегіне сілтеме жасап, өз параметрін 2-ге арттыратын
    // Incr делегатын құру.
    Incr incr = count => count + 2;

    // Ал енді incr лямбда-өрнегіне пайдалану.
    Console.WriteLine("Использование лямбда-выражения incr: ");
    int x = -10;
    while (x <= 0)
    {
        Console.Write(x + " ");
        x = incr(x); // x мәнін 2-ге арттыру
    }

    Console.WriteLine("\n");

    // Лямбда-өрнекке сілтеме жасайтын IsEven делегат экземплярын құру,
    // егер оның параметрі жұп сан болса, true мәнін қайтарады,
    // әйтпесе false мәнін қайтарады
    IsEven isEven = n => n % 2 == 0;
```

```
    //Ал енді isEven лямбда-өрнегін пайдаланамыз.
    Console.WriteLine("Использование лямбда-выражения isEven: ");

    for (int i = 1; i <= 10; i++)
        if (isEven(i)) Console.WriteLine(i + " четное.");
  }
}
```

Программаның орындалуы төмендегідей нәтиже береді.

Использование лямбда-выражения incr:
-10 -8 -6 -4 -2 0

Использование лямбда-выражения isEven:
2 четное.
4 четное.
6 четное.
8 четное.
10 четное.



Программадағы келесі жарияланған жолдарға назар аударыңыз.

Incr incr = count => count +2;
IsEven isEven = n => n % 2 == 0;