9-Дәріс. C# тілінде тізбелер мен құрылымдарды өңдеу

Дәрістің мақсаты:

- Файлмен жұмыс істеудің негізгі ұғымдарын түсіндіру.
- System.IO атау кеңістігіндегі негізгі кластармен (File, StreamReader, StreamWriter) танысу.
- Файлдан оқу және файлға жазу тәсілдерін меңгеру.
- Ерекше жағдайларды өңдеу (исключения) және деректердің сақталуын қамтамасыз ету.

Негізгі тақырыптар

1. Файл және ағын (stream) ұғымдары.
2. File класы және оның статикалық әдістері (Exists, Delete, Copy, Move).
3. StreamReader және StreamWriter кластарын қолдану мысалдары.
4. Файлды ашу, оқу, жазу және жабу операциялары.
5. Буферлеу және деректер ағынын басқару.
6. Қате өңдеу (try...catch...finally) құрылымы.
7. using блогының артықшылығы және ресурс босату механизмі.

C# тілінде тізбелерді (enum) қолдану

Тізбелер логикалық түрде бір-бірімен байланысқан константалар, яғни тұрақты сөздер жиыны болып табылады.

Тізбе (Перечисление – Enumeration) – бұл бірнеше тұрақты атауларды біріктіріп тұратын, қолданушы анықтайтын бүтін сандық тип, оның жалпы атынан бөлек ішкі элементтерінің өзіндік аттары және соларға сәйкес нөмірлері болады. Тізбе **enum** түйінді сөзі арқылы төмендегідей түрде жарияланады:

```
enum <тізбе_аты>{<1-аты>, <2-аты>, …};
```

Enum түйінді сөзінен кейін тізбенің аты жазылып, жүйелі жақша ішінде тізім элементтері үтірмен бөлініп беріледі.

Мысалы, Bagyttar атты тізбе құрайық, ол әртүрлі қозғалыс бағыттарын жеке-жеке көрсететін болсын:

```
enum Bagyttar {Солға, Оңға, Алға, Артқа};
```

Осылай тізбені жариялаған кезде, оның ішкі әрбір константасына, яғни тізбе элементтеріне 0-ден басталған (Солға = 0, Оңға = 1, Алға = 2, Артқа = 3) бүтін мәндер меншіктеледі. Енді осы тізбе қолданылған мысалды қарастырайық:

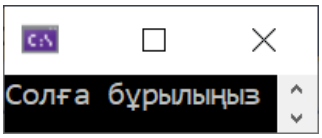
```
using System;
using System.Text;
namespace Mysal
{  enum Bagyttar { Солға, Оңға, Алға, Артқа };   // тізбені жариялау
   class Program
   {  public static void Buiryq(Bagyttar bagyt)  // тізбені қолданатын әдіс
      {  switch (bagyt)
         {  case Bagyttar.Алға:
                Console.WriteLine("Алға жүріңіз"); break;
            case Bagyttar.Артқа:
                Console.WriteLine("Артқа қайтыңыз"); break;
            case Bagyttar.Солға:
                Console.WriteLine("Солға бұрылыңыз"); break;
            case Bagyttar.Оңға:
                Console.WriteLine("Оңға бұрылыңыз"); break;

         }
      }

      static void Main(string[] args)
      {  Console.OutputEncoding = Encoding.Unicode;
```

```
            Bagyttar bagyt = Bagyttar.Солға;  // bagyt обьектісіне мән беру
            Buiryq(bagyt);                    // Buiryq әдісін шақыру
            Console.ReadKey();
        }
    }
}
```

Программаның орындалу нәтижесі:



Тізбелер сілтемелік типке жатады. Тізбелерді класс ішінде немесе сыртында құрып алып, класс ішінде осы тізбеге жататын айнымалыны жариялаймыз. Жоғарыдағы мысалда Bagyttar атты тізбесі құрылды, Program класында Buiryq атты әдіс жарияланып, bagyt параметрі Bagyttar тізбесіне жататыны анықталды. Әдісте switch операторы қолданылып, bagyt параметрінің тізбенің қай элементін алуына байланысты бұйрықтар орындалады. Main әдісінде Bagyttar тізбесіне жататын bagyt айнымалысы құрылып, оған мән берілді де, Buiryq әдісі шақырылды.

Тізбелер міндетті түрде бүтін сандар типінде (byte, int, short, long) болуы тиіс. Егер тип тікелей көрсетілмесе, онда келісім бойынша ол int типі болып есептеледі. Int типінен басқа типтер болса, тізбе атынан кейін қос нүкте қойылып типі көрсетіледі:

```
enum kunder
{Дүйсенбі, Сейсенбі, Сәрсенбі, Бейсенбі, Жұма, Сенбі, Жексенбі};
```

немесе

```
enum Uaqyt: byte { Таңертең, Түс, Кеш, Түн};
```

Бұл тізбенің әрбір тұрақты элементіне, келісім бойынша, бүтін сандық мән (нөмір) меншіктеледі, оның бірінші элементіне – 0 мәні, екіншісі – 1 мәні беріліп, ары қарай дәл сол сияқты түрде жалғаса береді. Kunder тізбесі int типінде болса, Uaqyt тізбесі byte типінде берілді.

Біз элементтер нөмірлерінің алғашқысын басқаша да тікелей көрсете аламыз, мысалы:

```
enum Operaciya
{
  Qosu = 1, // әрбір келесі элемент нөмірі бірге өсіп отырады
  Alu,      // бұл элемент нөмірі 2-ге тең
  Kobeitu,  // 3-ке тең
  Bolu      // 4-ке тең
}
```

Тек бірінші элементке ғана емес кез келген элементтің сандық мәнін беруімізге болады. Программа ішінде кез келген элементке мән тағайындасақ, одан кейінгі элементтерінің мәні 1-ге артып жалғаса береді. Егер біз бұл тізбенің Alu деген элементіне 5 деген индекстік мән беретін болсақ, Kobeitu 6, Bolu 7 мәніне тең болады.

Барлық элементтерге де тікелей түрде мән тағайындауға болады:

```
  enum Operaciya
{ Qosu = 1, Alu = 3, Kobeitu = 5, Bolu = 7 }
```

Мұндайда тізбе константаларының мәндері бірдей болуы да мүмкін, сондай-ақ бір константаға екінші бір константа мәнін меншіктей де аламыз:

```
enum Tus
{ Ақ=2, Қара, Көк=2, Қызыл = Ақ }
```

Мұнда “Aq” пен “Кок” константасының мәні 2-ге тең болса, “Qara” мен “Qyzyl” константасының мәні 3-ке тең.

Тізбелерді қолданғанда тізбе элементтерін меншіктеу үшін тізбеге сілтеме жасайтын айнымалыны қолданамыз. Программада осы айнымалыға тізбедегі бір мәнді бере аламыз, яғни айнымалы мәні ретінде тізбеде анықталған константалардың бірі қолданылуы тиіс. Әрбір константа тізбедегі орнына сәйкес белгілі бір нөмірге сәйкес келіп отырады, бірақ оған біз сандық мәнді тікелей меншіктей алмаймыз. Егерде біз экранға осы айнымалының мәнін шығаратын болсақ, онда сандық нөмір емес, константа мәнін аламыз. Ал сандық мән алғымыз келсе, онда оны сандық типке келтіру керек болады.

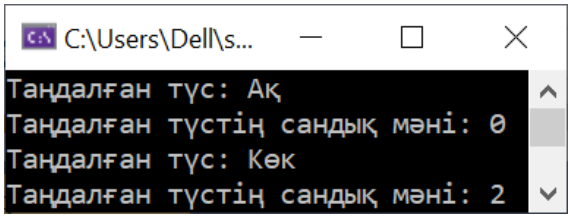
Тізім элементінің мәнін және сандық мәнін шығаруды келесі мысалдан көрулеріңізге болады:

```
using System;
```

```
using System.Text;
namespace Mysal
{ class Program
  { enum Tus
    { Ақ, Қара, Көк, Қызыл = Ақ }

    static void Main(string[] args)
    { Console.OutputEncoding = Encoding.UTF8;
      Tus t1, t2; //тізбе элементтерін меншіктейтін айнымалыларды сипаттау
      t1 = Tus.Қызыл; //Айнымалыға тізбе элементін меншіктеу
      Console.WriteLine("Таңдалған түс: " + t1); //тізбе элементі мәнін шығару
      Console.WriteLine("Таңдалған түстің сандық мәні: " + (int)t1);
      //тізбе элементінің сандық мәнін шығару
      t2 = Tus.Көк; //t2=2 – деп сандық мәнді меншіктей алмаймыз
      Console.WriteLine("Таңдалған түс: " + t2);
      Console.WriteLine("Таңдалған түстің сандық мәні: " + (int)t2);
      //тізбе элементінің сандық мәнін шығару үшін айнымалы типін көрсетеміз
      Console.ReadKey();
    }
  }
}
```

Программаның орындалу нәтижесі:



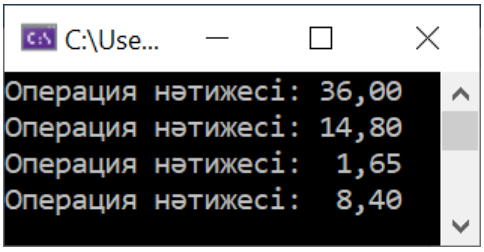
Мысалда Tus тізбесі құрылып, t1, t2 айнымалылары осы тізбеге жататыны анықталды және осы айнымалыларға тізбе элементтері меншіктелді. Айнымалылардың типін түрлендіру ((int)t1) арқылы тізбе элементінің сандық мәнін алдық.

Енді тізбелердің құрамында әдісі бар тағы бір мысал қарастырайық:

```
using System;
using System.Text;
namespace Mysal
{ class Program
  { enum Operaciya
    { Qosu, Alu, kobeitu, bolu}
    static void MathOp(double x, double y, Operaciya op) // Екі санға
                                  // математикалық операция орындайтын әдіс
    { double natije = 0.0;
      switch (op)
      {  case Operaciya.Qosu: natije = x + y; break;
         case Operaciya.Alu: natije = x - y; break;
         case Operaciya.kobeitu: natije = x * y; break;
         case Operaciya.bolu: natije = x / y; break;
      }
      Console.WriteLine("Операция нәтижесі: {0,5:f2}", natije);
    }

    static void Main(string[] args)
    { Console.OutputEncoding = Encoding.Unicode;
      MathOp(4.5, 8, Operaciya.kobeitu);
      MathOp(5.8, 9, Operaciya.Qosu);
      MathOp(18.2, 11, Operaciya.bolu);
```

```
        MathOp(21.8, 13.4, Operaciya.Alu);
        Console.ReadKey();
    }
  }
}
```



Программаның орындалу нәтижесі:

Мұнда арифметикалық операцияларды көрсететін Operaciya тізбесі (enum) бар. Оған қоса, параметрлер ретінде екі сан мен операция типін қабылдайтын MathOp әдісі анықталған. Негізгі Main әдісінде біз MathOp процедурасына екі сан мен операция типін көрсете отырып (калькулятор тәрізді), оны екі рет шақырамыз.

C# тілінде құрылымдарды қолдану

Базалық қарапайым типтерден басқа C# тілінде жиымдар тәрізді мәліметтердің күрделі типі болып саналатын *құрылым* деп аталатын тип қарастырылған.

Құрылымдар (`structure` – структура) өзара логикалық түрде байланысқан әртүрлі типтегі мәліметтерді байланыстырады. Оның элементтері компьютер жадында бірінен кейін бірі тізбектеліп орналасады.

Құрылымдар struct түйінді сөзінің көмегімен жарияланады және синтаксис тұрғысынан қарағанда, кластарға ұқсас болып келеді. Төменде құрылымдарды жариялаудың жалпы формасы келтірілген:

```
struct аты
{
   // мүшелерді жариялау
}
```

мұндағы аты құрылымға берілген нақты идентификаторды (құрылым атын) білдіреді. Құрылым мүшелерін жариялағанда әр мүшесін жеке-жеке типтерін көрсетіп жариялайды.

Мысалы, студент туралы құрылымды жариялау керек болса, төмендегідей түрде жазылады:

```
struct Student
{ public string aty;       // аты-жөні
  public int jasy;              // жасы
  public float stipendia;   // стипендиясы
}
```

мұндағы құрылым аты `Student` идентификаторы, ал құрылым элементтерін құрылым өрістері (`field` – өріс) дейді, олар кез келген типте бола береді. Мұнда үш айнымалы – `aty`, `jasy` және `stipendia` анықталған, олар студенттің атын, жасын және стипендия мөлшерін сақтайды.

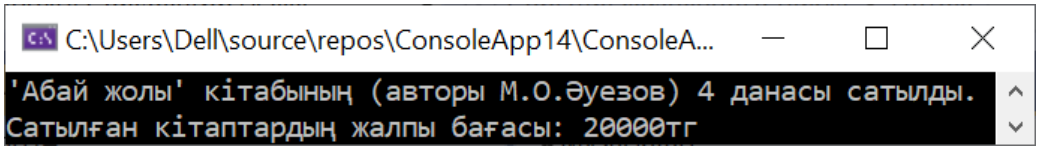
Құрылымдар ішіне қарапайым айнымалылар және әдістер кіреді. Айнымалылар мен әдістерді программаның кез келген жерінен пайдалану мүмкіндігін беру үшін олардың алдына `public` (ашық) қатынасу модификаторын қосамыз.

Келесі мысалда Kitap құрылымын құрайық, онда кітап аты, авторы және бағасы, сатылған кітап саны туралы мәліметтер сақталатын болады. Оған қоса, кітап туралы экранға мәлімет шығаратын және сатылған кітаптардың жалпы бағасын шығататын әдістерді де қосып жазып шығарайық:

```
using System;
using System.Text;
namespace Mysal
{ struct Kitap
  { public string aty;       // кітаптың аты
    public string avtory;    // кітаптың авторы
    public int bagasy;       // кітаптың бағасы
    public int sany;         // сатылған кітап саны
    public int JBaga(int s, int b)  // сатылған кітаптың жалпы бағасын
                                    // шығаратын әдіс
```

```
    { return (s * b); }
    public void Shygaru()   //кітап туралы ақпаратты шығаратын әдіс
    { Console.WriteLine($"'{aty}' кітабының (авторы {avtory})" +
                        $" {sany} данасы сатылды.");
    }
 }
 class Program
 { static void Main(string[] args)
   { Console.OutputEncoding = Encoding.UTF8;
     Kitap kitap1;  // Kitap құрылымындағы kitap1 айнымалысын сипаттау
                    // kitap1 құрылымының мүшелерінің мәнін беру
     kitap1.aty = "Абай жолы";
     kitap1.avtory = "М.О.Әуезов";
     kitap1.bagasy = 5000;
     kitap1.sany = 4;
     kitap1.Shygaru();   // kitap1 құрылымынына Shygaru әдісін қолдану
     Console.WriteLine("Сатылған кітаптардың жалпы бағасы: " +
             kitap1.JBaga(kitap1.sany, kitap1.bagasy) + "тг");
                        // kitap1 құрылымынына JBaga әдісін қолдану
     Console.ReadKey();
   }
 }
}
```

Программа жұмысының нәтижесі:



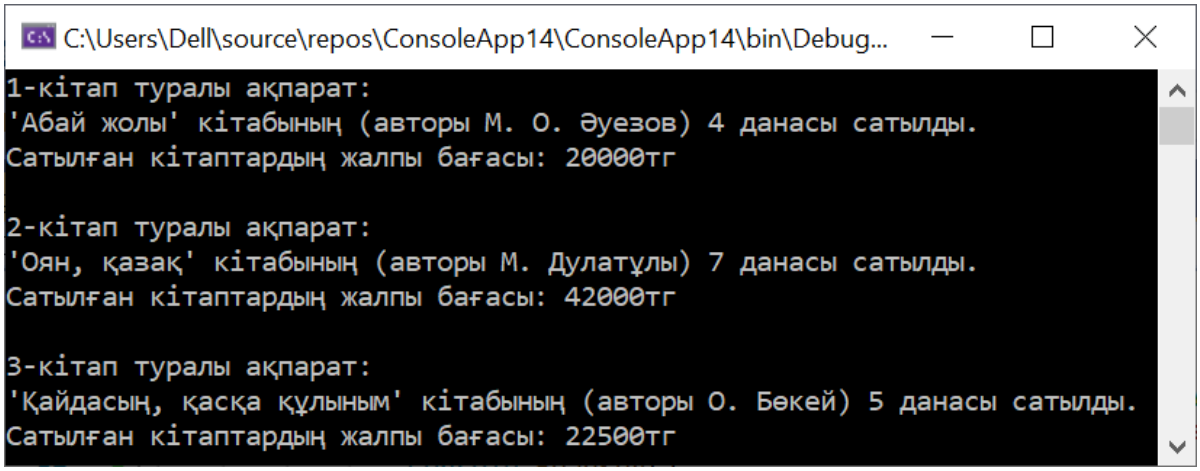
Құрылымды атаулар кеңістігі ішінде немесе класс ішінде де құра аламыз. Бұл мысалда құрылым атаулар кеңістігі ішінде сипатталды. Оны осы атаулар кеңістігіндегі барлық кластардың ішінде қолдану үшін құрылым өрістерін `public` қатынасу түрінде алдық. `Kitap` құрылымы құрылымында кітаптың аты, авторын, бағасын, сатқан санын білдіретін 4 өріс және сатылған кітап туралы ақпаратты шығаратын `Shygaru` әдісі, сатылған кітаптардың жалпы бағасын шығаратын `Jbaga` әдіс сияқты 2 әдіс осы құрылымның құрамына кіреді. `Program` класында `kitap` құрылмына жататын `kitap1` айнымалысын алып және `kitap1` құрылымының мәндерін енгізеді және аталған 2 әдісті `kitap1` құрылымына қолданып программаның нәтижесін алады. Мұнда `kitap` құрылымы бір кітапқа ғана қолданылып, `kitap1` айнымалысы алынды. Егер бізде бірнеше кітап болатын болса, оларға осы құрылымға жататын жеке-жеке айнымалылар алуымызға немесе құрылымдардың жиымы ретінде сипаттауымызға болады.

Құрылымдарды жиым ретінде берген кезде жиымның типі ретінде құрылымның аты көрсетіледі. Бұл жиымның элементтерінің осы құрылымға жататын білдіреді. Енді алдыңғы мысалдағы `Kitap` құрылымына жататын `kitap` объектісін алып, оны жиым ретінде сипаттап, 3 кітапқа қолданайық. Осыған сәйкес төменде `Kitap` құрылымынан кейінгі `Program` класынан ары қарай орналасқан программа бөлігін кеңейтіп жазып шығайық:

```
. . .
class Program
{ static void Main(string[] args)
  {
    Console.OutputEncoding = Encoding.UTF8;
    Kitap[] kitap = new Kitap[3];     // Kitap құрылымына жататын
                                      // 3 элементтен тұратын kitap жиымы
                                  // 1-кітап мәліметтерін енгізу
    kitap[0].aty = "Абай жолы";
    kitap[0].avtory = "М. О. Әуезов";
    kitap[0].bagasy = 5000;
    kitap[0].sany = 4;
    Console.WriteLine("1-кітап туралы ақпарат:");
    kitap[0].Shygaru();          // 1-кітапқа әдістерді қолдану
```

```
        Console.WriteLine("Сатылған кітаптардың жалпы бағасы: " +
                kitap[0].JBaga(kitap[0].sany, kitap[0].bagasy) + "тг");
                                    // 2-кітап мәліметтерін енгізу
        kitap[1].aty = "Оян, қазақ";
        kitap[1].avtory = "М. Дулатұлы";
        kitap[1].bagasy = 6000;
        kitap[1].sany = 7;
        Console.WriteLine("\n2-кітап туралы ақпарат:");
        kitap[1].Shygaru();         // 2-кітапқа әдістерді қолдану
        Console.WriteLine("Сатылған кітаптардың жалпы бағасы: " +
                kitap[1].JBaga(kitap[1].sany, kitap[1].bagasy) + "тг");
                                    // 3-кітап мәліметтерін енгізу
        kitap[2].aty = "Қайдасың, қасқа құлыным";
        kitap[2].avtory = "О. Бөкей";
        kitap[2].bagasy = 4500;
        kitap[2].sany = 5;
        Console.WriteLine("\n3-кітап туралы ақпарат:");
        kitap[2].Shygaru();         // 3-кітапқа әдістерді қолдану
        Console.WriteLine("Сатылған кітаптардың жалпы бағасы: " +
                kitap[2].JBaga(kitap[2].sany, kitap[2].bagasy) + "тг");
        Console.ReadKey();
    }
}
. . .
```

Программаның орындалу нәтижесі:



Мұнда тек `Program` класындағы жазылған программа коды ғана алынды. Өйткені алдыңғы мысалдағы атаулар кеңістігіндегі сипатталған `Kitap` құрылымы еш өзгеріссіз қалады. Тек `Program` класында `Kitap` құрылымына жататын `kitap` объектісі жиым ретінде беріліп, `kitap`[0] жиымына 1-кітап туралы ақпаратты, `kitap`[1] жиымына 2-кітап туралы, `kitap`[2] жиымына 3-кітап туралы ақпарат меншіктеледі.

Құрылымдарда конструкторларды пайдалану. Құрылымда қарапайым әдістерден бөлек арнайы әдіс – конструкторды пайдалануға болады, ол белгілі бір объектіні бастапқы мәндермен инициалдайды, мысалы, барлық өрістерге, келісім бойынша берілген мәндерді тағайындайды.

Негізінде, құрылымдарда конструкторларды қолдану міндетті болып саналмайды, мысалы,

```
Kitap kitap1;
```

Дегенмен, мұндай кездерде құрылымды пайдалану үшін алдымен оның барлық өрістерін инициалдап алу керек.

```
kitap1.name = "Abai joly";
kitap1.avtor = "M.Auezov";
kitap1.sany = 5;
```

Келісім бойынша конструкторды шақыру құрылым өрістерін автоматты түрде инициалдау мүмкіндігін береді (мысалы, сандық мәліметтер үшін – бұл 0 мәні):

```
Kitap kitap2 = new Kitap(); // конструкторды пайдалану
```

Конструкторды шақыру синтаксисі мынадай болады:

```
new құрылым_аты ([параметрлер_тізімі])
```

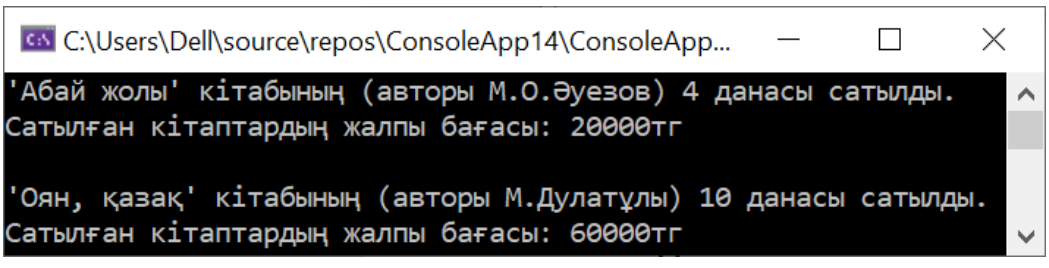
Конструктордың аты құрылымның атымен бірдей болады. Ал параметрлер тізімінде құрылымның өрістері бойынша мәндері жазылады. Конструкторды қолдануды келесі мысалда көрсетейік:

```
using System;
using System.Text;
namespace Mysal
{ struct Kitap
  { public string aty;       // кітаптың аты
    public string avtory;    // кітаптың авторы
    public int bagasy;       // кітаптың бағасы
    public int sany;         // сатылған кітап саны
    public Kitap(string a,string av,int b,int s) // конструкторды құру
    { aty = a; avtory = av; bagasy = b; sany = s; }
    public int JBaga(int s, int b)   // сатылған кітаптың
                             // жалпы бағасын шығаратын әдіс
    { return (s * b); }
    public void Shygaru()    // кітап туралы ақпаратты шығаратын әдіс
    {  Console.WriteLine($"'{aty}' кітабының (авторы {avtory})" +
              $" {sany} данасы сатылды.");
    }
  }

  class Program
  { static void Main(string[] args)
    { Console.OutputEncoding = Encoding.UTF8;
      Kitap kitap1; //Kitap құрылымындағы kitap1 айнымалысын сипаттау
                    // kitap1 құрылымының мүшелерінің мәнін беру
      kitap1.aty = "Абай жолы";
      kitap1.avtory = "М.О.Әуезов";
      kitap1.bagasy = 5000;
      kitap1.sany = 4;
      kitap1.Shygaru();// kitap1 құрылымынына Shygaru әдісін қолдану
      Console.WriteLine("Сатылған кітаптардың жалпы бағасы: " +
              kitap1.JBaga(kitap1.sany, kitap1.bagasy) + "тг");
      Console.WriteLine();

            // 2-кітап туралы ақпаратты конструктор арқылы беру
      Kitap kitap2 = new Kitap("Оян, қазақ", "М.Дулатұлы", 6000,10);
      kitap2.Shygaru();// kitap2 құрылымынына Shygaru әдісін қолдану
      Console.WriteLine("Сатылған кітаптардың жалпы бағасы: " +
              kitap2.JBaga(kitap2.sany, kitap2.bagasy) + "тг");
      Console.WriteLine();
      Console.ReadKey();
    }
  }
}
```

Программаның орындалу нәтижесі:



Мұнда Kitap құрылымын жарияланғанда `Kitap` конструкторы құрылады:

```
public Kitap(string a, string av, int b, int s)
{ aty = a; avtory =av; bagasy = b; sany = s; }
```

Құрылым элементтерінің мәнін беру барысында `kitap1` объектісін тікелей инициалдасақ, `kitap2` объектісіне конструкторды (`Kitap kitap2 = new Kitap("Оян, қазақ", "М. Дулатұлы", 6000, 10))` шақырады.

Бұл мысалда көрсетілгендей, құрылым `new` операторының көмегімен конструкторды шақыру арқылы немесе объектіні қарапайым жариялау жолымен де инициалдануы мүмкін. Осылай, егер `new` операторы пайдаланылса, онда құрылым өрістерін келісім бойынша шақырылатын конструктор (мұндайда барлық өрістерде келісім бойынша берілетін мәндер орнатылады) немесе қолданушы анықтаған конструктор инициалдайды.

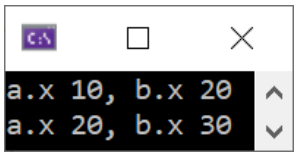
Ал бір құрылымның айнымалысы басқа құрылымның айнымалысына меншіктелгенде, меншіктеу операторының оң жақ бөлігіндегі құрылым объектісінің *көшірмесі* құрылады.

Мысал ретінде келесі программаны қарастырайық.

```
using System;
namespace Mysal
{ struct Struct
  { public int x; }

  class Program
  { static void Main(string[] args)
    { Struct a; Struct b;
      a.x = 10; b.x = 20;
      Console.WriteLine("a.x {0}, b.x {1}", a.x, b.x);
      a = b; b.x = 30;
      Console.WriteLine("a.x {0}, b.x {1}", a.x, b.x);
      Console.ReadKey();
    }
  }
}
```

Программа жұмысының нәтижесі:



Жоғарыда келтірілген нәтиже көрсеткендей,

```
a = b;
```

меншіктеуінен кейін құрылымның айнымалылары `a` мен `b` бұрынғыдай мүлде жеке түрде (оқшауланып) қала береді, яғни, `a` айнымалысы `b` айнымалысына сілтеме жасамайды және онымен ешқандай байланыста емес (тек оның `b` айнымалысы мәнінің көшірмесінен тұратыны болмаса). Егер `a` мен `b` айнымалылары белгілі бір кластың объектілеріне қарай бағыттайтын сілтемелі типте болса, онда жағдай тіпті басқаша болар еді.

Бақылау сұрақтары

1. Ағын (stream) дегеніміз не және не үшін қолданылады?
2. Файлмен жұмыс жасау кезінде File класының қандай әдістері жиі қолданылады?
3. StreamReader және StreamWriter кластарын қандай мақсатта қолданамыз?
4. Файлды оқу мен жазу кезінде буфердің рөлі қандай?
5. try...catch блогының маңызы неде?
6. using блогын қолданудың артықшылығы неде?
7. File.Exists() және File.Delete() әдістерінің қызметін түсіндіріңіз.
8. Файлдан жолдарды оқу үшін қандай әдіс пайдаланылады?
9. Файлмен жұмыс кезінде ресурстарды босатпаса не болады?
10. Файл жазу кезінде қандай ерекше жағдайлар туындауы мүмкін?

Ұсынылатын әдебиеттер

1. Колдаев, В.Д. Основы алгоритмизации и программирования: Уч. пособие/ под ред. проф. Л.Г.Гагариной. -М.: ИД «ФОРУМ»: ИНФРА-М,2015.-416 с.
2. Бөрібаев Б. Компьютердің арифметикалық негіздері: Жоғары оқу орындары студенттеріне арналған оқу құралы. -Алматы: Қазақ университеті, 2009. -80 б.

3. Бөрібаев Б. Алгоритмдеу, мәліметтер құрылымы және программалау тілдері: оқулық. – Алматы: Қазақ университеті, 2012. -235 б.
4. Вирт Н. Алгоритмы и структуры данных: Пер. с англ. -М.: Мир, 2011.-360с.
5. Бөрібаев Б., Абдрахманова М. C# тілінде программалау (практикалық курс): оқу құралы –Алматы: Қазақ университеті, 2018. -258 б.
6. Troy Dimes. C# Programming for Beginners. Paperback – January 25, 2015.
7. Шилдт Г. C# 4.0: полное руководство. М.: ООО ”И.Д.Вильямс”, 2017. -1056 с.
8. Джуст В. Разработка обслуживаемых программ на языке C#. СПб.: Изд. дом «Питер», 2017.
9. Шарп Д. Microsoft Visual C#. Подроб. руководство. 8-е изд.С-Петербург: Изд. дом «Питер», 2016.
10. A. Troelsen, P. Japikse. Pro C# 7: With .NET and .NET Core. – Apress, 2017. -1372 p.