**10-Дәріс. C# тілінде файлдарды пайдалану**

**Дәрістің мақсаты:**

C# тілінде файлдармен жұмыс істеуді үйрету: байттық, символдық және екілік ағындармен (streams) жұмыс, FileStream, StreamReader/Writer, BinaryReader/Writer, және File кластарын қолдану.

**Негізгі тақырыптар**

1. Файл және ағын (stream) ұғымдары.
2. FileStream класы: файл ашу режимдері (FileMode, FileAccess).
3. Байттық енгізу/шығару операциялары (WriteByte, ReadByte, Seek).
4. Символдық ағындар (StreamReader, StreamWriter).
5. Екілік ағындар (BinaryReader, BinaryWriter).
6. File класының негізгі әдістері (ReadAllText, WriteAllText, AppendAllText, т.б.).
7. Файлмен жұмыс істеу кезінде буферлеу және ерекше жағдайлар (исключения).
8. Практикалық мысалдар: файлға жазу, оқу, көшіру, өшіру, толықтыру.

**FileStream класы және файлға байттық енгізу-шығару**

Файл – белгілі бір атпен дискіге жазылған біркелкі (бір типтегі) мәліметтер жиыны. Көбінесе файл байттар тізбегінен тұрады. Файлдарға байттарды енгізіп шығару үшін соған сәйкес әдістер енгізілген және файлдарға байттық ағындар арқылы мәлімет енгізу-шығару әрекеттері іске асырылған.

Файлдарға мәлімет енгізіп шығаратын байттар ағыны символдық ағынның соларға сәйкес объектісінде орналасады. Файлдар символдарды, мәтіндерді өңдеу кездерінде, оларды енгізу-шығару операцияларында жиі қолданылады.

Мәліметтерді сыртқы құрылғылардан компьютер жедел жадына жүктеуді оқу немесе енгізу деп, ал оған кері процесті, яғни сыртқы құрылғыларға ақпарат жіберуді – жазу немесе (сыртқа) шығару деп атайды.

C# тіліндегі енгізу/шығару операциялары .NET кітапханасы кластары көмегімен ішкі енгізу/шығару жүйесі арқылы орындалады. Бұл дәрісте файлдармен мәлімет алмасу амалдары қарастырылады. Мұндағы мәлімет алмасу барысы ағындар арқылы жүзеге асырылады.

Ағын (stream) – бұл керекті ақпараттар жиынын мәліметтер көзінен оны қабылдайтын құрылғыға кез келген тәсілмен тасымалдауға, көшіруге қатысты айтылатын абстракциялық ұғым. Ағындар мәліметтердің стандартты типтерімен де, қолданушы анықтаған типтерімен де сенімді түрде жұмыс істейді және ол – жалпыға түсінікті бірыңғай синтаксистің де атауы.

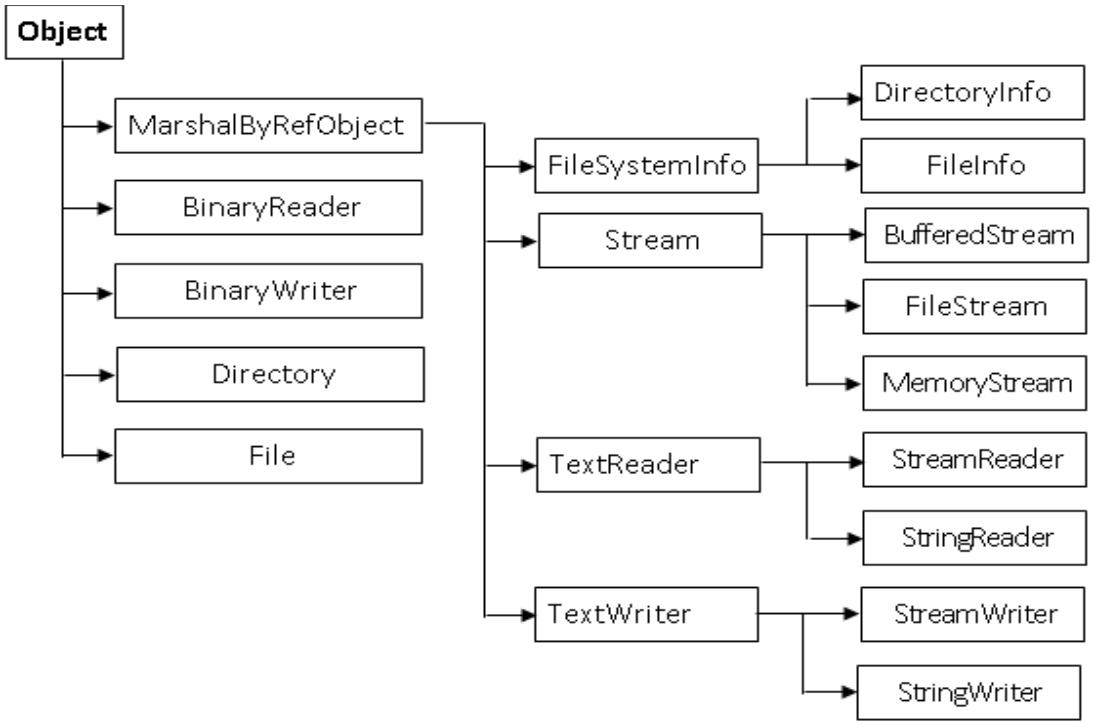
Ақпараттарды тасымалдау жылдамдығын арттыру үшін ағындармен мәлімет алмасу жедел жадының арнайы түрі – буфер арқылы орындалады. Ашылған әрбір файл үшін оны уақытша сақтайтын буфер бөлініп беріледі. Файлға мәлімет жазу кезінде тасымалданатын ақпарат алдымен буферге жазылып, ол сонда буфер көлемі толғанша сақталып тұрады. Мәліметтер тек сонан кейін барып арнайы команда көмегімен сыртқы құрылғыға жазылады. Файлдан ақпарат оқу кезінде де мәліметтер алдымен буферге оқылады да, ол толған кезде барып компьютер жадына көшіріледі.

.NET Framework ортасында файлдарға мәлімет енгізу-шығаруды ұйымдастыратын кластар қарастырылған. Файлға қатысты байттық ағын құру үшін, FileStream класы қолданылады. Бұл класс Stream класынан туынды болып саналады және оның барлық функцияларын пайдаланады.

Ағындар кластары, оның ішіндегі FileStream класы System.IO атаулар кеңістігінде анықталған. Сондықтан файлдарды пайдаланатын кез келген программаның ең басында төмендегі программалық код болуы тиіс:

```
using System.IO;
```

Ағындарды қолдану үшін .NET кітапханасында сатылы түрде жұмыс істейтін кластар иерархиясы қарастырылған, оның негізгі бөлігі 9.1-суретте көрсетілген. Бұл кластар System.IO атаулар кеңістігінде анықталған.




10.1-сурет. Ағындармен жұмыс істеуге арналған .NET кітапханасындағы кластардың иерархиялық құрылымы

Төмендегі 10.1-кестеде осы System.IO атаулар кеңістігіндегі кітапхана кластарының файлдармен, каталогтармен, жедел жады аймақтарымен әртүрлі режимдерде жұмыс істеу сипаттамалары берілген.

10.1-кестеде. System.IO атаулар кеңістігіндегі кітапхана кластары

| Класс | Сипаттамалары |
|---|---|
| BinaryReader, BinaryWriter | Қарапайым типтердің (бүтін сандық, логикалық, тіркестік және т.б.) мәндерін ішкі бейнелеу түрінде оқу мен жазу |
| BufferedStream | Байт ағындарын уақытша сақтау (мысалы, кейіннен оларды тұрақты сақтау орнына алып бару үшін) |
| Directory, Directorylnfo, File, Filelnfo | Каталогтармен немесе физикалық файлдармен жұмыс іс-теу: оларды құру, өшіру, қасиеттерін алу. File және Directory кластарының мүмкіндіктері статикалық әдістердің негізгі түрінде жасалған. Бірақ солар сияқты Directorylnfo және Filelnfo кластары кәдімгі қарапайым әдістерді қолданады |
| FileStream | Байттар ағыны түрінде берілген файлдармен кез келген (тікелей) бағытта мәлімет алмасу |
| MemoryStream | Компьютер жедел жадындағы байттар ағынымен кез келген бағытта мәлімет алмасу |
| StreamWriter, StreamReader | Мәтіндік ақпаратты файлдан оқу және файлға жазу (кез келген бағытта мәлімет алмасу сүйемелденбейді) |
| StringWriter, StringReader | Компьютер жедел жадындағы мәтіндік ақпаратпен жұмыс істеу |

Кестеден сыртқы құрылғылармен мәлімет алмасуды төмендегідей деңгейлерде:

- мәліметтердің екілік жүйедегі түрінде (BinaryReader, BinaryWriter);
- байттар түрінде (FileStream);
- мәтіндер, яғни символдар тізбегі түрінде (StreamWriter, StreamReader) орындауға болатынын көруге болады.

Файлдармен мәлімет алмасу тізбекті түрде орындалса, онда файлдан келесі элементті оқу (жазу) тек оның алдындағы элеметті оқығаннан (жазғаннан) кейін ғана жүргізіледі.

Ал мәлімет алмасу байттың көрсетілген адресі бойынша кез келген орыннан алынған элементпен орындала беретін болса, онда ол тікелей түрде мәлімет алмасу болып саналады.

Файлмен байланысқан байттық ағынды қалыптастыру үшін FileStream класының объектісі құрылады. Бұл класта бірнеше конструкторлар анықталған.

Төменде солардың ең жиі кездесетіні келтірілген:

FileStream(string жол, FileMode режим)

мұндағы жол ашылатын файлдың адресі мен аты, яғни оған баратын маршрутты толық көрсетуге болады; ал режим – файлды ашу тәртібі, ол 9.2-кестеде көрсетілген FileMode – режимдерінің біріне

сәйкес келуі керек. FileStream класындағы бұл режимдер файлды одан мәлімет оқу немесе жазу үшін ашады. Олардан өзгешелеу бір тәсіл – файлды тек оған мәлімет қосып жазу үшін ашатын FileMode.Append режимі болып табылады.

10.2-кесте. FileMode режиміндегі мәндер

| Мәні | Сипаттамасы |
|---|---|
| FileMode.Append | Файл соңына жаңа мәліметтерді қосады. |
| FileMode.Create | Жаңа файл ашады. Егер құрылған файлдың атымен бірдей басқа файл болса, ол өшеді. |
| FileMode.Create-New | Бұрын құратын файлдың атымен бірдей файл болмаған жағдайда жаңа файл ашады. |
| FileMode.Open | Бұрын құрылған файлды ашады |
| FileMode.Open-OrCreate | Егер бұрыннан осындай аты бар файл болса, соны ашады, әйтпесе жаңа файл құрып, соны ашады. |
| FileMode.Truncate | Бұрын құрылған файлды ашып, ондағы мәліметті өшіреді, яғни ұзындығын нөлге теңейді |

Егер файлды ашу мүмкін болмаса, онда аластама туындайды. Дискіде файл жоқ болғандықтан, оны ашу мүмкін болмаса, онда FileNotFoundException аластамасы туындайды. Ал егер файлды белгілі бір енгізу-шығару әрекетінің қате кетуіне байланысты аша алмайтын болсақ, онда IOException аластамасы туындайды.

FileMode режимімен бірге файлмен қатынас құру үшін System.IO атаулар кеңістігінде анықталған FileAccess тізбелері қолданылады. Файлдарды ашқан кезде FileAccess тізбелері арқылы файлдарды тек оқу үшін, тек жазу үшін немесе оқу әрі жазу үшін ашуға мүмкіндік аламыз. Файлдарды қай мақсатта ашқанымызды білдіретін FileAccess тізбелері 10.3-кесте көрсетілген.

10.3-кесте. FileAccess тізбелерінің мәндері

| *Мәні* | *Сипаттамасы* |
|---|---|
| Read | Файлды тек оқу үшін ашу |
| ReadWrite | Файлды оқу және жазу үшін ашу |
| Write | Файлды жазу үшін ашу |

**Байттар ағындары.** Байттар деңгейінде файлға мәлімет енгізу/шығару әрекеттері FileStream класы көмегімен орындалады, ол ағындармен стандартты операциялар жиынын орындауды анықтайтын Stream класынан туындаған оның мұрагері болып табылады. Stream класының элементтері 10.4-кестеде сипатталған.

10.4-кесте. Stream класының элементтері

| ***Элементі*** | ***Сипаттамасы*** |
|---|---|
| BeginRead, BeginWrite | Асинхронды енгізу/шығаруды бастау |
| CanRead, CanSeek, CanWrite | Ағынның қандай операцияларды сүйемелдейтінін анықтайтын қасиеттері: мәліметтерді оқу және/немесе жазу, солармен тікелей қатынас құру |
| Close | Орындалып жатқан ағынды жабу және сонымен байланысты ресурстарды (сокеттер, файлдарға нұсқауыштар және т.б.) босату |
| EndRead, EndWrite | Асинхронды енгізу операциясының аяқталуын күту; асинхронды шығару операциясын аяқтау |
| Flush | Буферден алынған мәліметтерді ағынмен байланысқан деректер көзіне жазып, буферді тазалау. Егер бұл ағын үшін буфер пайдаланылмайтын болса, онда бұл әдіс ештеңе істемейді |
| Length | Байтпен берілген ағын ұзындығын қайтару |
| Position | Ағындағы мәлімет оқылуға/жазылуға (өңделуге) тиіс позицияны қайтару |
| Read, ReadByte | Өңделіп жатқан ағыннан байттар тізбегін (немесе бір байтты) оқу және нұсқауышты ары қарай оқылған байттар санына сәйкес орынға көшіру (жылжыту) |
| Seek | Ағын нұсқауышын ол тұрған орыннан көрсетілген позицияға алып бару |
| SetLength | Өңделіп жатқан ағын ұзындығын орнату |
| Write, WriteByte | Өңделіп жатқан ағынға тізбегін (немесе бір байтты) жазу және нұсқауышты ары қарай жазылған байттар санына сәйкес орынға көшіру (жылжыту) |

FileStream класы бұл элементтерді дискілік файлдармен жұмыс істеу үшін қолданады. Файлмен жұмыс істеу режимдерін анықтау үшін стандартты FileMode және FileAccess тізбелері пайдаланылады.

Енді мысал арқылы байттарды файлға жазу және соларды символдар түрінде файлдан оқуды көрсетейік (латынның кіші әріптерінің кодтары - 97+123):

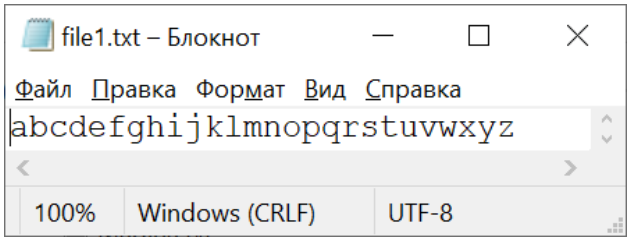
```
using System;
using System.Text;
using System.IO; // System.IO кітапханасын қосу
namespace Mysal
{ class Program
  { static void Main(string[] args)
    { Console.OutputEncoding = Encoding.UTF8;
      FileStream f = new FileStream("d:\\file1.txt",
      FileMode.Create, FileAccess.ReadWrite);
      byte a;
      // Файлға латынның 26 кіші әріптерін жазу
      for (int i = 0; i < 26; ++i)
            { a = (byte)(i + 97);
              f.WriteByte(a);
      }
            // ішкі нұсқауышты файл басына алып бару
            f.Seek(0, SeekOrigin.Begin);
      long j = f.Length; // файл ұзындығы (байтпен)

            // файл мәліметін оқу және экранға шығару
      do
      {
               a = (byte)f.ReadByte(); // символ кодын оқу
               Console.Write(" "+(char)a); // символды шығару
               j = j - 1;
      } while (j != 0); // while (f.Length != 0);
            Console.WriteLine();
            f.Close(); // файлды жабу
            Console.ReadKey();
    }
  }
}
```

Программаның орындалу нәтижесі:




Құрылған файл және ондағы жазылған символдар:




Бұл мысалда FileMode.Create режимі арқылы d: дискісінде file1.txt файлы құрылды. FileAccess.ReadWrite қатынасу режимі іске қосылғандықтан, бұл файлды мәлімет жазу үшін де, оқу үшін де қолдана аламыз. WriteByte әдісі арқылы құрылған файлға байттарды жаздық. ReadByte әдісі арқылы файлдағы байттарды оқып, символдық типке айналдырып, экранға шығардық. Программа соңында Close әдісі арқылы файл жабылды.

**Символдық ағындар мен екілік ағындар**

StreamReader және StreamWriter символдық ағындары Unicode символдарымен жұмыс істейді, сондықтан олар көбінесе адамдар оқуға ыңғайлы файлдармен жұмыс істеу кезінде қолданылады. Бұл ағындар сәйкесінше TextReader және TextWriter кластарының туындылары (мұрагерлері) болып табылады, сол себепті көбінесе олардың функционалдық деңгейі дұрыс қамтамасыз етіледі. Мәтіндік файлдар үшін олардан кез келген бағытта мәлімет оқу/жазу, яғни қатынас құру әрекеті орындалмайды. TextReader және TextWriter кластарының негізгі әдістері 10.5,10.6- кестелерде көрсетілген.

10.5-кесте. TextWriter базалық класының негізгі әдістері

| Элементі | Сипаттамасы |
|---|---|
| Close | Файлды жабу және онымен байланысты ресурстарды босату. Егер жазу кезінде буфер пайдаланылатын болса, ол автоматты түрде тазаланады |
| Flush | Ағымдағы файл үшін буферлердің барлығын тазарту және оларда жинақталған мәліметтерді өздерінің тұрақты сақталатын орнына жазып қою. Мұнда файлдың өзін жабылмайды |
| NewLine | Жаңа жолдың басын білдіретін символдар тізбегін беру үшін пайдаланылады. Келісім бойынша, "күймешені (каретканы) қайтару" - "келесі жолға көшу " (\r\n) тізбегі қолданылады |
| Write | Мәтін фрагментін ағынға жазу |
| WriteLine | Жолды ағынға жазу және басқа жолға көшу |

10.6 – кесте. TextReader класының аса маңызды әдістері

| Элементі | Сипаттамасы |
|---|---|
| Peek | Файлдағы нұсқауыш позициясына өзгертпей, келесі символды қайтару |
| Read | Кіріс ағынынан мәліметтер оқу |
| ReadBlock | Кіріс ағынынан пайдаланушы көрсеткен символдар санын оқып, соларды буферге көрсетілген позициядан (орыннан) бастап жазып шығу |
| ReadLine | Өңделіп жатқан ағыннан сөз тіркесін оқып, оны string типіндегі мән ретінде қайтару. null бос тіркесі файлдың соңын (EOF) білдіреді |
| ReadToEnd | Өңделіп жатқан позициядан бастап, ағынның соңына дейінгі символдардың барлығын оқып, оқылған мәліметтерді string типіндегі бір жол ретінде қайтару |

**Екілік ағындар.** Екілік файлдар мәліметтерді компьютер жадында қалай сақталса, дәл сондай түрде, яғни ішкі бейнелену формасында сақтайды. Екілік файлдар мәліметтерді адамдардың оқуы үшін емес, оларды программаларда пайдалану үшін қолданады.

BinaryWriter шығыс (нәтижелік) ағыны мәліметтерді кез келген бағытта жазуға арналған, яғни оның мәліметтерді екілік файлдың кез келген жеріне жазу мүмкіндігі бар. Екілік файл көбінесе FileStream ағыны ретінде қарастырылатын базалық ағын негізінде ашылады. Кіріс мәліметі бар екілік ағынның мәліметтердің құрамдас қарапайым типтері үшін асыра жүктелген оқу әдістері болады. Екілік ағындардың негізгі әдістері 10.7, 10.8- кестелерде көрсетілген.

10.7 – кесте. BinaryWriter класының аса маңызды элементтері

| Элементі | Сипаттамасы |
|---|---|
| BaseStream | Базалық ағын, онымен BinaryWriter объектісі жұмыс істейді |
| Close | Ағынды жабу |
| Flush | Буферді тазалау |
| Seek | Өңделіп жатқан ағындағы позицияны (орынды) орнату |
| Write | Өңделіп жатқан ағынға мән жазу |

10.8 – кесте. BinaryReader класының аса маңызды элементтері

| Элементі | Сипаттамасы |
|---|---|
| BaseStream | Базалық ағын, онымен BinaryReader объектісі жұмыс істейді |
| Close | Ағынды жабу |
| PeekChar | Ағындағы ішкі нұсқауышты жылжытпай, келесі символды қайтару |
| Read | Байттар немесе символдар ағынын оқып, соларды кіріс параметрі ретінде берілетін жиымда сақтау |
| ReadXXX | Ағыннан белгілі бір типтегі (мысалы, ReadBoolean, ReadByte, Readlnt32 және т. б.) мәліметтер оқу |

BinaryWriter және BinaryReader кластарының қолдануын файлға құрылымдар жиынын жазу және оқуды жүзеге асыратын программаны құрып көрейік:

```
using System;
using System.Text;
using System.IO; // System.IO кітапханасын қосу
namespace Mysal
{ struct Kitap
  {
      public string aty;
      public string avtory;
      public int bagasy;
      public int sany;
```

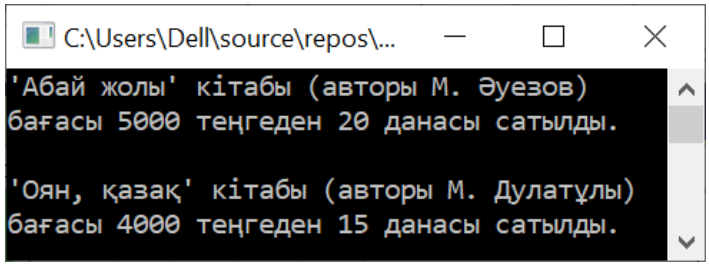
```
    public Kitap(string a, string av, int b, int s)
    { aty = a; avtory = av; bagasy = b; sany = s; }
}

class Program
{
 static void Main(string[] args)
    {
        Console.OutputEncoding = Encoding.UTF8;
        Kitap[] kitap = new Kitap[2];
        kitap[0] = new Kitap("Абай жолы", "М. Әуезов", 5000, 20);
        kitap[1] = new Kitap("Оян, қазақ", "М. Дулатұлы", 4000, 15);

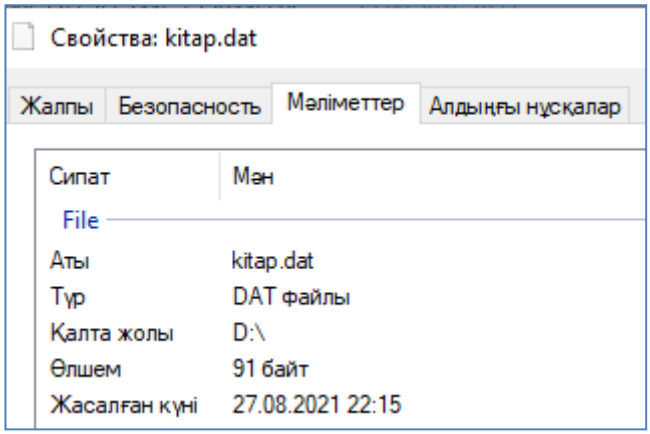
        try
        { // BinaryWriter обьектісін құру
          using (BinaryWriter writer = new  BinaryWriter(File.Open("d:\\kitap.dat",
                        FileMode.OpenOrCreate)))
            { // файлға құрылымның әр өрісінің мәнін жазу
              foreach (Kitap k in kitap)
              { writer.Write(k.aty);
                writer.Write(k.avtory);
                writer.Write(k.bagasy);
                writer.Write(k.sany);
              }
            }
        // BinaryReader обьектісін құру
        using (BinaryReader reader = new BinaryReader(File.Open("d:\\kitap.dat", FileMode.Open)))
        { // файлдың соңына жеткенше әр мәнді файлдан оқу
            while (reader.PeekChar() > -1)
            {
                string aty= reader.ReadString();
                string avtory = reader.ReadString();
                int bagasy = reader.ReadInt32();
                int sany = reader.ReadInt32();

            Console.WriteLine("'{0}' кітабы (авторы {1}) \nбағасы {2} теңгеден {3} данасы сатылды.",
                    aty, avtory, bagasy, sany);
                Console.WriteLine();
          }
         }
       }
        catch (Exception e)
        {
         Console.WriteLine(e.Message);
        }
        Console.ReadLine();
    }
 }
}
```

Программаның нәтижесі:



Құрылған файлдың сипаттамасы:




Мысалда өткен дәрісте қарастырылған Kitap құрлымы жасалды. Main әдісінде Кітар құрлымының жиымын құрдық және конструктор арқылы өрістерін толтырған соң оны BinaryWriter класының обьектісі құрылып, File.Open ("d:\\kitap.dat", FileMode. OpenOrCreate) жолындағы параметрлерді writer обьектісіне меншіктейді. Содан кейін, цикл бойынша құрылымның әр өрісін ағынға жазып, файлға орналастырамыз.

Ары қарай BinaryReader класы d: дискідегі kitap.dat файлынан құрылымды оқиды. Ағынды соңына дейін оқу үшін PeekChar() әдісін қолданамыз. Бұл әдіс келесі символды оқиды және оның сандық мәнін қайтарады. Егер символ болмаса, онда әдіс -1 қайтарады, яғни файлдың соңына жеткенімізді білдіреді. BinaryReader класы арқылы оқыған файлдағы құрлымды Console.WriteLine әдісі арқылы экранға шығарды. Бұдан BinaryWriter және BinaryReader кластары екілік файлдармен жұмыс істеуге өте ыңғайлы екенін көруге болады.

**File класының әдістері.** Файлдарды өңдеу үшін қолданылатын кластың бірі File класы. File класының жиі қолданылатын келесідей әдістері бар, олар статикалық болып табылады және оларды объект құрмай-ақ пайдалануға болады:

10.9-кесте. File класының жиі қолданылатын әдістері

| | |
|---|---|
| AppendAllText() | көрсетілген мәтінді файлға қосу. Егер көрсетілген файл жоқ болатын болса, ол құрылады; |
| AppendText() | мәтінді файлға қосу, ол файл StreamWriter объектісін қайтара алады және оны қосымша ақпарат қосып жазуда пайдалануға болады; |
| Copy() | көрсетілген файлды жаңа орынға көшіру; |
| Create() | көрсетілген жол (маршрут) бойынша файл ашу; |
| CreateText() | UTF-8 форматындағы ақпаратты жазу үшін файл ашу; |
| Decrypt() | файл мәліметін шифрлаусыз бастапқы қалпына келтіру; |
| Delete() | көрсетілген файлды өшіру; |
| OpenText() | файлды UTF-8 форматындағы мәтінмен жұмыс істейтіндей түрде ашу; |
| ReadAllLines() | файлдың барлық жолдарын оқу; |
| ReadAllText() | файлды толық оқып шығу. |
| SetAccessControl() | файлмен қатынасу құқығын тағайындау; |
| SetCreationTime() | құрылған уақыт мерзімін өзгерту; |
| SetLastAccessTime() | файлды қолданылған уақытын өзгерту; |
| SetLastwriteTime() | файлға соңғы мәлімет жазу уақытын өзгерту; |
| WriteAllLines() | файл құрып, оған жолдар жиымын жазу. |
| WriteAllText() | файлға сөз тіркесі жіберіліп, ол бұрынғы мәліметтің үстіне жазылады. |
| Encrypt() | көрсетілген файлды шифрлау; |
| Exists() | көрсетілген файлдың бар екенін тексеру; |
| GetAccessControl() | файлға онымен қатынасу құқығын қайтару; |
| GetAttributes() | файл атрибуттарын қайтару; |
| GetCreationTime() | көрсетілген файлдың құрылған уақыты; |
| GetLastAccessTime() | файлды соңғы пайдаланған уақыт мерзімі; |
| GetLastWriteTime() | файлға соңғы мәлімет жазылған уақыт мерзімі; |
| Move() | файлды жаңа бумаға жылжыту (алып бару); |
| Open() | көрсетілген файлды ондағы мәліметті оқу және өзгерту үшін ашу; |
| OpenRead() | көрсетілген файлды оқу үшін ашу; |
| OpenWrite() | көрсетілген файлды оған мәлімет жазу үшін ашу. |

Жоғарыдағы файлдармен жұмыс істеуге арналған кейбір әдістерді пайдалану мысалдарын қарастырайық.

10.10-кесте. Файлдармен жұмыс істеуге арналған әдістер

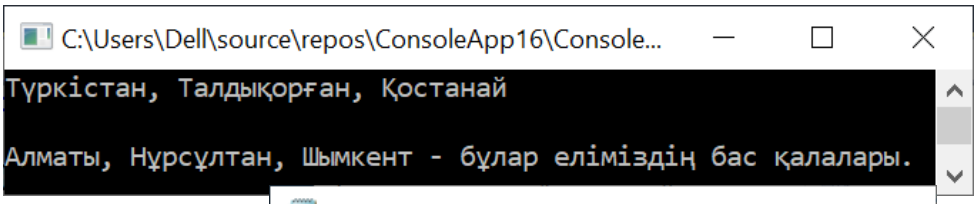
| | |
|---|---|
| ReadAllText() | статикалық әдіс – файлды толық оқып шығу. Мұнда параметр ретінде файлдың толық адресін және атын көрсетуге болады. Файл адресін жазу кезінде "\\" символдары арқылы оны экрандау керек (D:\\bax). |
| ReadAllLines() | бұл статикалық әдіс символдардан тұратын мәтін жолдары жиымын қайтарады, яғни файлдағы мәліметті оның жолдары бойынша оқу жүзеге асырылады. |
| WriteAllText() | мұнда параметр ретінде сөз тіркесін береміз, сол тіркес толығымен файлға жіберіліп, бұрынғы мәтіннің орнына жазылады. |
| AppendAllText() | мұнда жаңа мәлімет бұрынғы файлдағы мәлімет соңына қосылып жазылады. |

Енді осы әдістердің қолданылуын мысал арқылы көрсетейік. Алдын ала құрылған "d:\\qalalar.txt" файлында «Түркістан, Талдықорған, Қостанай» деген мәтін жазылған. Файлдағы мәтінді оқып, экранға шығарып, сосын мәтінді басқа мәтінмен ауыстырып, артынан олардың соңына тағы мәлімет қосатын программа құрайық:

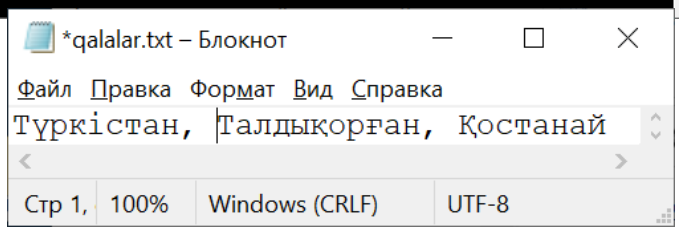
```
using System;
using System.Text;
using System.IO; // System.IO кітапханасын қосу
namespace Mysal
{ class Program
  { static void Main(string[] args)
    { Console.OutputEncoding = Encoding.UTF8;
      string File2 = "d:\\qalalar.txt";
      // былай көрсетуге де болады File2 = @"D:\qalalar.txt";
      // файлдағы мәліметті оқып, экранға шығару
      string readAllFile = File.ReadAllText(File2);
      Console.WriteLine(readAllFile); // мәтінді экранға шығару
      // файлдағы мәтінді басқа мәтінмен ауыстыру
      File.WriteAllText(File2, "Алматы, Нұрсұлтан, Шымкент");
      Console.WriteLine();
      // Келесі команда файл соңына бірнеше сөз қосады
      File.AppendAllText(File2, " -бұлар еліміздің бас қалалары.");
      readAllFile = File.ReadAllText(File2);
      // толықтырылған файл мәтінін экранға шығару
      Console.WriteLine(readAllFile);

    }
  }
}
```

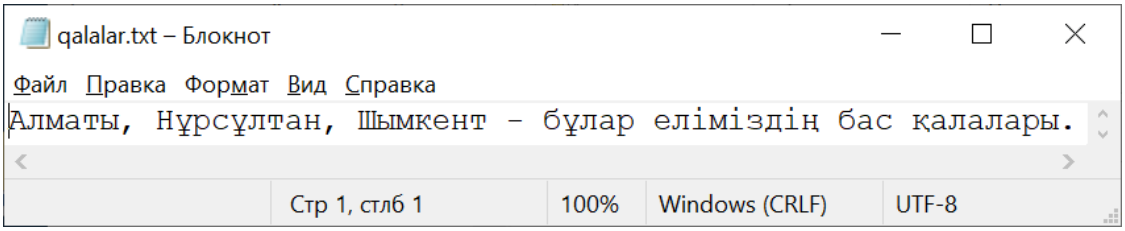
Программаның нәтижесі:




Файлдағы алғашқы мәтін:




Файлдағы соңғы мәтін:




Мұнда File.ReadAllText әдісі арқылы файлдағы мәтінді толық оқып, ReadAllText айнымалысына меншіктедік. File.WriteAllText әдісі арқылы файлдағы мәтінді "Алматы, Нұрсұлтан, Шымкент", - деген мәтінге ауыстырдық. Соңынан File.AppendAllText әдісі арқылы файлдығы мәтіннің соңына " - бұлар еліміздің бас қалалары.",- деген мәтінді қостық.

**Бақылау сұрақтары**

1. Ағын (stream) дегеніміз не және ол не үшін қажет?

2. FileStream класының негізгі міндеті қандай?
3. FileMode және FileAccess тізбелерінің айырмашылығы неде?
4. Байттық енгізу мен символдық енгізудің айырмашылығын түсіндіріңіз.
5. BinaryReader және BinaryWriter кластарын қандай жағдайларда қолданамыз?
6. File.ReadAllText() және File.AppendAllText() әдістерінің айырмашылығын көрсетіңіз.
7. Файлмен жұмыс кезінде қандай аластамалар (исключения) пайда болуы мүмкін?
8. Файлдарды оқу мен жазу кезінде буфердің рөлі қандай?
9. using операторын файлмен жұмыс кезінде не үшін қолданамыз?
10. Файлдарға мәліметтерді екілік және мәтіндік түрде жазудың айырмашылығы неде?

**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.