18.2 Dosya Kullanımı

- C++ programlama dilinde girdi – çıktı işlemlerinin **veri yolu** (data stream) kullanılarak gerçekleştirildiğini daha önceki bölümlerde belirtmiştik. Girdi ve çıktıların bellekte saklanmaları ve tekrar tekrar kullanılmaları için veri yolu olarak dosyaları kullanabiliriz.
- Verilerin ikincil bellekte kalıcı olarak saklandığı yapıya **dosya** (file) adı verilir. Örneğin aşağıda kişilerin isim ve numara kayıtlarını tuttuğumuz bir dosyayı hard diskte veya flash diskte saklayabiliriz.

Şekil 18.1 Dosya ve Kayıtlar

18.2 Dosya Kullanımı

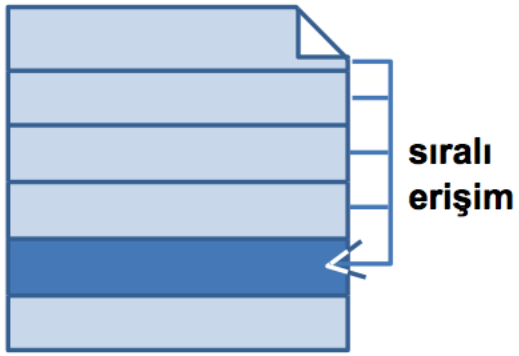
- Dosyalara erişim iki yöntemle yapılır:
  - Sıralı Erişim
  - Rastgele Erişim
- Dosya işlemleri için `fstream` kütüphanesini aşağıdaki komutla programa dahil etmemiz gerekir.

```
#include <fstream>
```

- Dosya girdi/çıktı işlemleri için üç sınıf tanımlanmıştır.
  - `ifstream` – Dosyadan okuma
  - `ofstream` – Dosyaya yazma
  - `fstream` – Dosyayı güncelleme

18.2.1 Sıralı Erişim

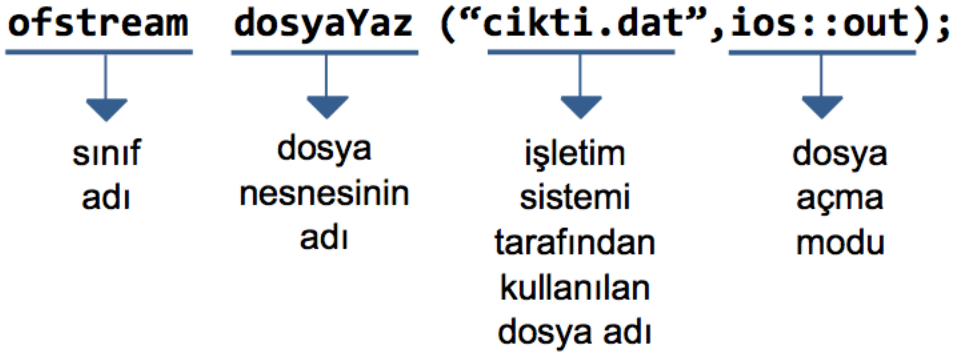
- *İng. Sequential Access*
- ***Sıralı erişim*** (sequential access) yöntemiyle okuma veya yazma işlemi aşağıda görüldüğü gibi dosyanın başından başlayarak sırayla yapılır.
- Dosyanın içinde saklanan herhangi bir veriye erişmek için o veriye kadar olan tüm kayıtlar sırayla okunur.



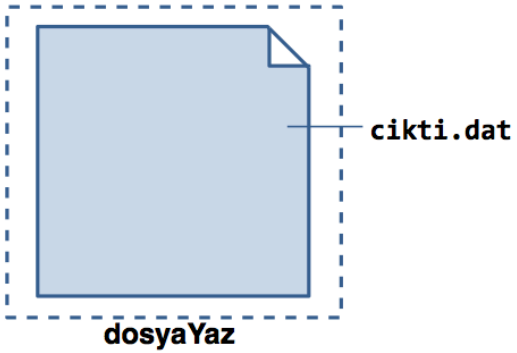
Şekil 18.3 Sıralı Erişim

18.2.1 Sıralı Erişim...

- **18.2.1.1 Dosya Çıktı İşlemleri**
- Dosyaya veri yazdırmak için `ofstream` sınıfını aşağıda gösterildiği şekilde kullanılır.



Şekil 18.4 `ofstream` Sınıfının Kullanımı



Şekil 18.5 dosyaYaz Nesnesi ve "cikti.dat" Dosyası

18.2.1 Sıralı Erişim...

- **18.2.1.1 Dosya Çıktı İşlemleri...**
- Çıktı dosyaları aşağıdaki farklı dosya açma modları kullanılarak açılabilir.
  - ios::out : Yazma işlemi dosyanın başından başlayarak yapılır ve daha önceden veri girilmişse yeni veriler bu verilerin üstüne yazılır.
  - ios::app : Yazma işlemi dosyanın en son verisinin olduğu yerden başlayarak yapılır ve daha önceden veri girilmişse herhangi bir veri kaybı yaşanmaz.
- Aşağıdaki tanımda olduğu gibi eğer dosya tanımında dosya açma modu kullanmazsak, varsayılan mod ios::out olacaktır.

18.2.1 Sıralı Erişim...

- **18.2.1.1 Dosya Çıktı İşlemleri...**
- Dosya açma işlemi open() fonksiyonu kullanılarak da yapılabilir.

```
ofstream dosyaYaz;
dosyaYaz.open("cikti.dat",ios::out);
```

- Dosya açmada hata oluşup oluşmadığını aşağıdaki şekilde kontrol edebiliriz.

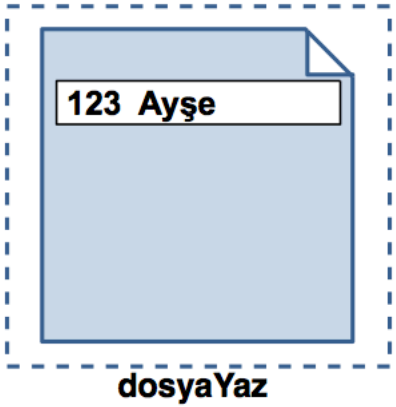
```
ofstream dosyaYaz ("cikti.dat",ios::out);
if (!dosyaYaz)
    cerr<<"dosya acilamadi";
else {...}
```

- dosyaYaz ismi eğer dosya başarılı bir şekilde açıldıysa `true`, açılamadıysa `false` değerini döndürür.

18.2.1 Sıralı Erişim...

- **18.2.1.1 Dosya Çıktı İşlemleri...**
- **Dosyaya Yazma:** Dosyayı yukarıdaki şekilde açtıktan sonra çıktılarımızı aynı cout veri yoluna yollar gibi dosya nesnesine göndermemiz gerekir.

```
int x=123;
string s="Ayse";
dosyaYaz <<x<<" "<<s<<endl;
```



Şekil 18.6 dosyaYaz Nesnesi ile Dosyaya Yazma

18.2.1 Sıralı Erişim...

- **18.2.1.1 Dosya Çıktı İşlemleri...**
- **Dosyayı Kapatma:** Program bitmeden dosyayı kapatmak için `close()` fonksiyonu kullanılır.

```
dosyaYaz.close();
```

- `close()` fonksiyonu işletim sisteminin dosya nesnesi ile dosya adı arasındaki ilişkiyi sonlandırmasını ve dosyayla ilgili son işlemlerin yapılmasını sağlar.
- `ifstream` ve `ofstream` sınıflarının içinde bulunan yıkıcı fonksiyonda `close()` fonksiyonu otomatik olarak çağırılır.
- Bu sebeple dosyalarınız için `close()` fonksiyonunu çağırmanız diğer programlama dillerinde olduğu gibi şart değildir.

18.2.1 Sıralı Erişim...

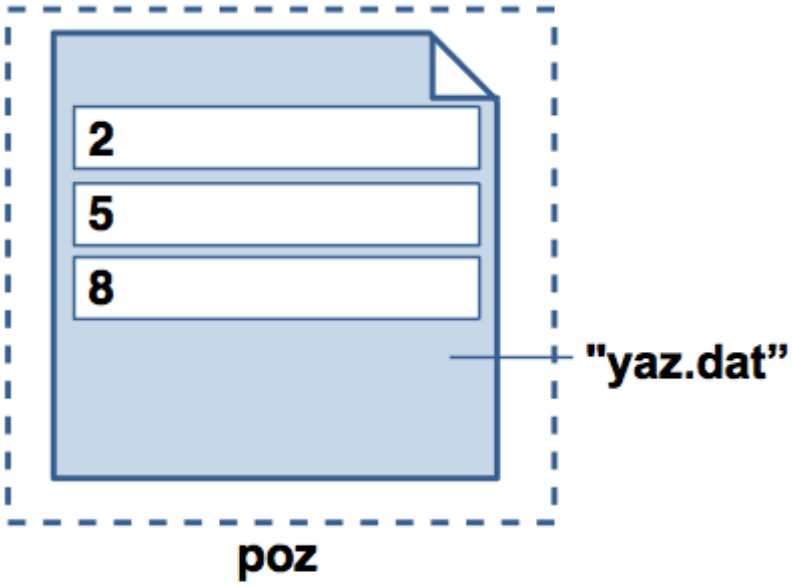
Örnek 18.2

```
#include <iostream>
#include <fstream>
int main()
{
    ofstream poz("yaz.txt",ios::out);          // Dosya tanımı
    int no;
    cout<<"50 sayi giriniz:";
    for (int i=1;i<=50;i++){
        cin>>no;                               // Kullanıcıdan sayılar okunur
        if (no>0)
            poz<<no<<endl;                     // Pozitif sayılar dosyaya yazdırılır
    }
    poz.close();
    return 0;
}
```

Örnek Girdi

```
2 -3 5 -7 8….
```

18.2.1 Sıralı Erişim...



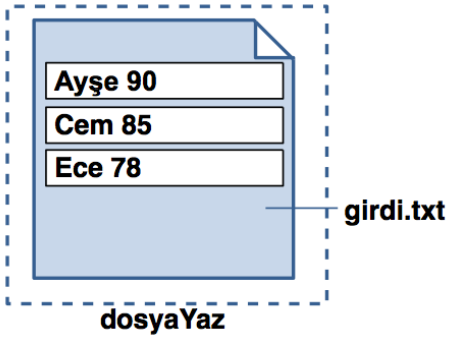
Şekil 18.6 poz Nesnesi ile "yaz.dat" Dosyası

18.2.1 Sıralı Erişim...

- **18.2.1.2 Dosya Girdi İşlemleri**
- Bir dosyadan veri okumak için daha önceden içinde veri olan bir dosyanın bulunması gerekir.
- Derleyicinin editörünü veya Word, NotePad, vs. gibi herhangi bir metin işleme yazılımını kullanıp dosyaya .txt, .doc, .dat gibi bir uzantı vererek bir metin dosyası hazırlayabiliriz

18.2.1 Sıralı Erişim...

- **18.2.1.2 Dosya Girdi İşlemleri...**
- Okunacak dosya programda bir nesne olarak tanımlanır.



Şekil 18.9 "girdi.txt" Dosyası

- Dosya açma modu aşağıdaki tanımda olduğu gibi yazılmazsa varsayılan mod olarak ios::in alınır.

```
ifstream  dosyaOku ("girdi.txt",ios::in);
```

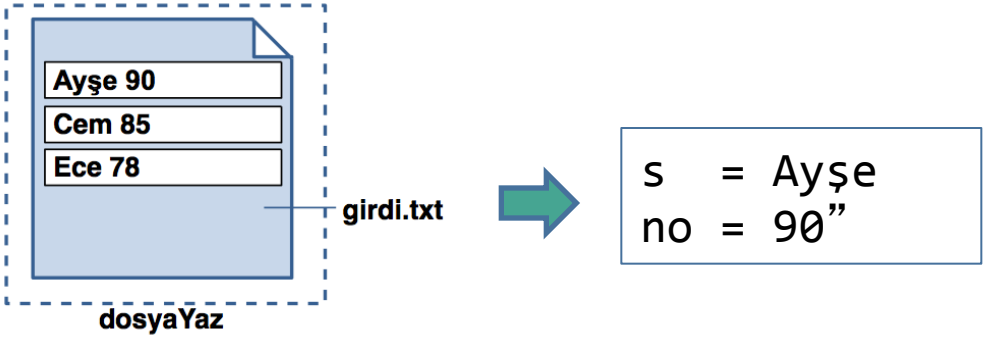
sınıf adı — dosya nesnesinin adı — işletim sistemi tarafından okunan dosya adı — dosya açma modu

Şekil 18.8 "girdi.txt" Dosyası

18.2.1 Sıralı Erişim...

- **18.2.1.2 Dosya Girdi İşlemleri...**
- Dosyayı açıldıktan sonra girdiler aynı `cin` veri yolundan okur gibi `dosyaOku` nesnesinden okunabilir.

```
string s;
int no;
dosyaOku >>s>>no;
```

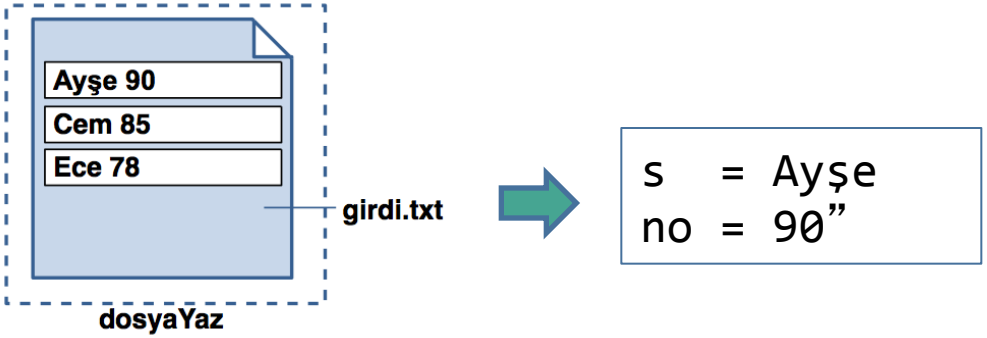


Şekil 18.9 "girdi.txt" Dosyası

18.2.1 Sıralı Erişim...

- **18.2.1.2 Dosya Girdi İşlemleri...**
- **Dosyadan Okuma**: Dosyayı açıldıktan sonra girdiler aynı `cin` veri yolundan okur gibi `dosyaOku` nesnesinden okunabilir.

```
string s;
int no;
dosyaOku >>s>>no;
```



Şekil 18.9 "girdi.txt" Dosyası

18.2.1 Sıralı Erişim...

- **18.2.1.3 Dosya Sonu**
- eof():
  - dosya sonu (**e**nd **o**f **f**ile) fonksiyonu
  - dosyanın sonuna gelindiyse `true`, daha okunacak veriler varsa ise `false` döndürür

18.2.1 Sıralı Erişim...

- **18.2.1.3 Dosya Sonu...**

Örnek 18.3

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main()
{   ifstream ogrenciler("girdi.txt");  // Girdi dosyası tanımı
    string isim, yuksekIsim;
    int n, yuksekNot=0;
    while (!ogrenciler.eof()){         // Dosyanın sonuna kadar döner
        ogrenciler>>isim>>n;           // Dosyadan isim ve numarayı okur
        if (n>yuksekNot){              // En yüksek notu alan ismi bulur
            yuksekNot=n;
            yuksekIsim=isim;
        }
    }
    cout<<"En yuksek not: "<<yuksekNot;
    cout<<" Isim:"<<yuksekIsim<<endl;
    return 0;
}
```

Çıktı

```
En yuksek not: 90 Isim:Ayse
```

18.2.1 Sıralı Erişim...

- **18.2.1.4 Dosya Sonuna Ekleme**
- Eğer dosyayı `ios::app` moduyla açılırsa, dosyadaki eski bilgiler silinmez ve dosyanın sonuna veriler eklenir.

18.2.1 Sıralı Erişim...

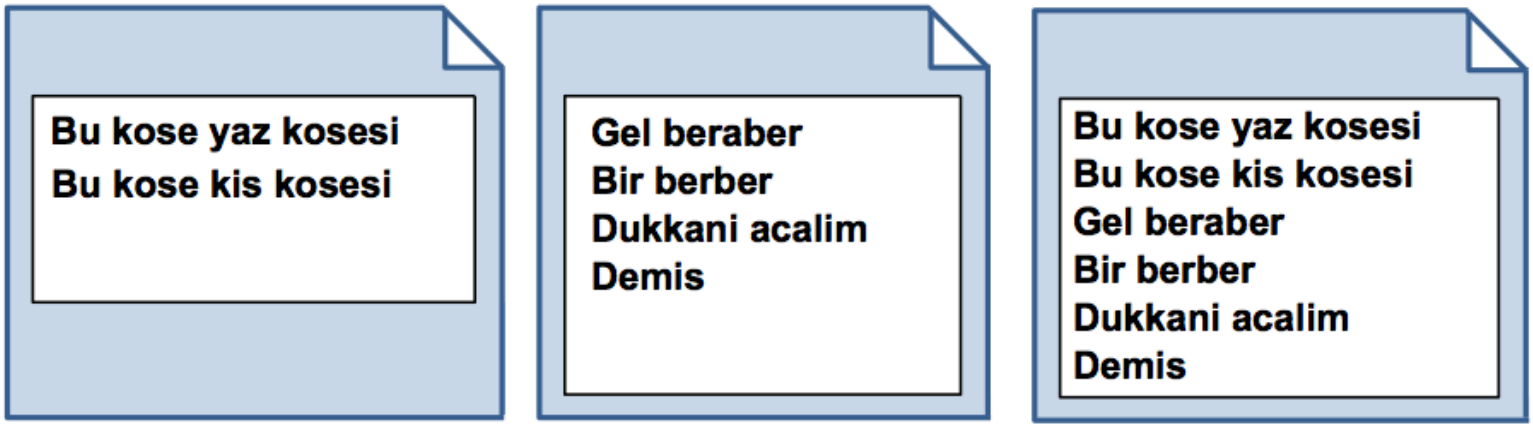
- **18.2.1.4 Dosya Sonuna Ekleme...**

Örnek 18.5

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
void dosyadanYaz (ifstream &g, ofstream &c)
{
    char dizgi[80];
    g.getline(dizgi,80);                    // Dosyadan bir satır okunur
    while (!g.eof()){                       // Dosyanın sonuna kadar döner
        c<<dizgi;                           // Satır dosyaya yazdırılır
        g.getline(dizgi,80);
        c<<endl;
    }
}
int main()
{
    ifstream f1("girdi1.txt");              // İlk girdi dosyası
    ifstream f2("girdi2.txt");              // İkinci girdi dosyası
    ofstream f3("birlesik.doc",ios::app);   // Çıktı dosyası
    dosyadanYaz(f1,f3);                     // İlk dosya yazdırılır
    dosyadanYaz(f2,f3);                     // İkinci dosya yazdırılır
    f3.close();
    return 0;
}
```

18.2.1 Sıralı Erişim...

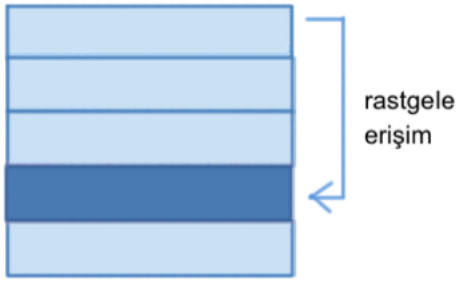
- **18.2.1.4 Dosya Sonuna Ekleme...**



Şekil 18.13 "`birlesik.txt`" Dosyası ve Dosyanın Sonuna Ekleme

18.2.2 RastgeleErişim...

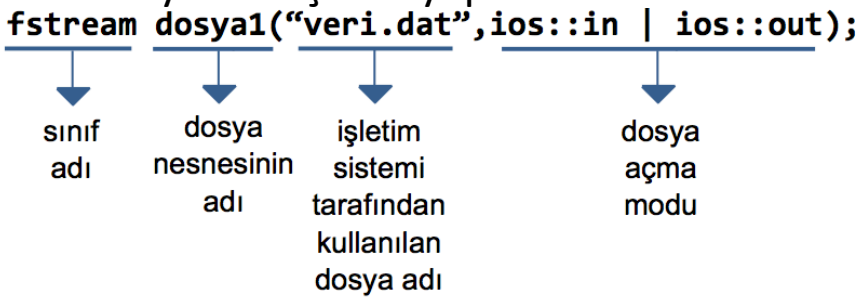
- *İng. Random access*
- Bazı durumlarda dosyanın herhangi bir yerindeki bilgi doğrudan okunmak istenebilir
- Örneğin seyrettiğimiz bir video filminde doğrudan 10. dakikaya gitmek isteyebiliriz.
- Bu durumlarda *rastgele erişim* dosya erişim yöntemi kullanılır.



Şekil 18.14 Rastgele Erişim

18.2.2 RastgeleErişim...

- Rastgele erişim için aşağıda görüldüğü gibi `fstream` tipinde bir nesne yaratılır.
- `dosya1` yazma ve okuma olmak üzere iki mod'da açılmıştır. `dosya1` üzerinde hem okuma hem de yazma işlemi yapılabilir.



Şekil 18.15 `fstream` Sınıfının Kullanımı

- `seekp()`: dosyanın herhangi bir yerine gitmek için kullanılır.
- **Örnek**:
  - dosyanın 500. byte'ında yer alan degeri okur

```
dosya1.seekp(500);
dosya1>>x;
```