

BM102

Algoritma ve Programlama II

**Girdi, Çıktı ve Dosya  
İşlemleri**

# İçerik



1. Girdi - Çıktı
  1. Girdi-Çıktı Fonksiyonları
  2. Formatlama Fonksiyonları
  3. Hizalama Fonksiyonları
2. Dosya Kullanımı
  1. Sıralı Erişim
  2. Rastgele Erişim

***Çözümlü Sorular***

# Hedefler



- Girdi çıktı fonksiyonlarını kullanma
- Verilen çıktı formatına uygun formatlama fonksiyonlarını kullanma
- Verilen çıktı hizalama özelliklerine uygun hizalama fonksiyonlarını kullanma
- İkincil bellekte saklanan bir dosyadan veri okuma
- Verileri çıktı olarak ikincil bellekte saklanmak üzere dosyaya yazma
- Var olan bir dosyanın sonuna yazma
- Sıralı erişim ve rastgele erişim yöntemleri arasındaki farkı anlatma
- Sıralı ve rastgele erişim yöntemlerinin hangi durumlarda tercih edileceğini örneklerle açıklama

# 1. Girdi - Çıktı

- C++ programlama dilinde girdi - çıktı işlemleri veri yolu (data stream) kullanılarak gerçekleştirilir.
- Veri yolunda, veri akışı baytlar halinde sağlanır.
- `<iostream.h>` başlık dosyasında yer alan bazı veri yolları:
  - `cout`: standart çıktı yolu, ostream sınıfının bir nesnesi
  - `cin`: standart girdi yolu, istream sınıfının bir nesnesi
  - `cerr`: standart hata yolu, ostream sınıfının bir nesnesi
- `cerr` hata mesajlarını ekranda görüntülemek için kullanılan nesnedir.

# 1.1. Girdi - Çıktı Fonksiyonları

- **put()**: Tek karakter yazdırmak için kullanılan bir fonksiyondur.

## Örnek

```
cout.put('?');  
cout.put('O').put('N');
```

## Örnek Çıktı

```
?  
ON
```

# 1.1. Girdi - Çıktı Fonksiyonları...

- **get():** Karakter veya dizgi okumak için kullanılan bir fonksiyondur.

## Örnek

```
char kar;  
kar = cin.get();           // Bir karakter okur
```

## Örnek

```
char kar;  
cout<<"Bir kelime giriniz:"<<endl;  
do {  
    kar = cin.get();           // Bir karakter okur  
    cout<<kar<<"harfini girdiniz."<<endl;  
} while (kar != EOF);        // Girdinin bittiğini kontrol eder
```

## Örnek Girdi

```
Ali ↵  
Ctrl/Z
```

## Örnek Çıktı

```
A harfini girdiniz.  
l harfini girdiniz.  
i harfini girdiniz.  
harfini girdiniz.
```

# 1.1. Girdi - Çıktı Fonksiyonları...

## Örnek

```
char diz[12];  
cin.get(diz,12);  
cout<<"Okunan: "<<diz;
```

*// diz'e 11 karakter okur*

## Örnek Girdi

MERHABA ↵

## Örnek Çıktı

Okunan: MERHABA

## Örnek

```
char diz[12];  
cin>>diz;  
cout<<"Okunan: "<<diz;
```

*// diz'i boşluğa kadar okur*

## Örnek Girdi

MERHABA DUNYA

## Örnek Çıktı

Okunan: MERHABA

# 1.1. Girdi - Çıktı Fonksiyonları...

- **getline():** get() fonksiyonu gibi çalışır ancak okunan dizginin sonuna “\n” ekler.

## Örnek

```
char diz[80];  
cin.getline(diz,80);  
cout<<"Okunan: "<<diz;
```

diz

"MERHABA\n"

## Örnek Girdi

MERHABA

## Örnek Çıktı

Okunan: MERHABA



# 1.1. Girdi - Çıktı Fonksiyonları...

- **gcount()**: en son kaç karakter okunduğunu döndürür.

## Örnek

```
char diz[10];  
cin.get(diz,10);  
cout<<diz<<"girdisinde "<<cin.gcount()<<"karakter var. ";
```

## Örnek Girdi

MERHABA

## Örnek Çıktı

MERHABA girdisinde 7 karakter var.

## 1.2. Formatlama Fonksiyonları

- **precision()** ve **setprecision()**: yazdırılacak basamak sayısını belirtmek için kullanılırlar.

### Örnek

```
cout.precision(2);  
cout<<1.23456789 ;
```

### Örnek Çıktı

1.2

### Örnek

```
#include <iomanip.h>  
...  
cout<<setprecision(2)<<1.23456789 ;
```

### Örnek Çıktı

1.2

### Örnek

```
#include <iomanip.h>  
...  
cout<<setprecision(4)<<1.23456789 ;
```

### Örnek Çıktı

1.235

## 1.2. Formatlama Fonksiyonları...

- **width()** ve **setw()**: Bir değerin yazdırılacağı alan boyunu belirtmek için kullanılırlar. Alan boyu bir değerin yazdırılacağı alandaki karakter sayısını belirtir.

### Örnek

```
int x=5;
cout<<x;
cout.width(10);
cout<<x<<x;
```

### Örnek Çıktı

```
5
-----
                    5 5
-----
```

### Örnek

```
#include <iomanip.h>
...
int x=5;
cout<<x;
cout<<setw(10)<<x<<x ;
```

### Örnek Çıktı

```
5
-----
                    5 5
-----
```

## 1.2. Formatlama Fonksiyonları...

- `fill()`: boşlukları verilen karakterle doldurur

### Örnek

```
cout.width(5);  
cout.fill('0');  
cout<<1 ;
```

### Örnek Çıktı

```
0 0 0 0 1  
_ _ _ _ _
```

### Örnek

```
cout.width(8);  
cout.fill('*');  
cout<<23 ;
```

### Örnek Çıktı

```
* * * * *  
2 3  
_ _ _ _ _
```

# 1.3. Hizalama Fonksiyonları

- **setf()** ve **setiosflags()**: Hizalanacak yönü belirtmek için kullanılırlar

## Örnek

```
int x=12345;  
cout.width(10);  
cout.setf(ios::left,ios::adjustfield);  
cout<<x ;
```

## Örnek Çıktı

```
1 2 3 4 5  
-----
```

## Örnek

```
#include <iomanip.h>  
...  
int x=12345;  
cout.setiosflags(ios::left)<<x;
```

## Örnek Çıktı

```
1 2 3 4 5  
-----
```

## Örnek

```
float x=1.23;  
cout<<x<<endl;  
cout.setf(ios::scientific,ios::floatfield);  
cout<<x ;
```

## Örnek Çıktı

```
1 . 2 3  
-----
```

```
1 . 2 3 0 0 0 0 e + 0 0  
-----
```

# 1.3. Hizalama Fonksiyonları...

```
#include <iostream.h>
using namespace std;
class S
{
    protected:
        int a;
    public:
        S(int x){a=x;}
        void r(){cout.width(5); cout.fill('&'); cout<<a<<endl;}
};
class M: public S
{
    int a;
    public:
        M(int x, int y):S(x){a=y;}
        void r(){cout.width(4); cout.fill('?');
            cout<<a<<endl; cout.setf(ios::left,ios::adjustfield);
            cout.width(6); cout.fill('%'); cout<<S::a<<endl;}
};
```



## 1.3. Hizalama Fonksiyonları...

```
int main()
{
    S * t[2];
    t[0]=new S(10);
    t[0]->r();
    t[1]=new M(15,18);
    t[1]->r();
    delete [] t;
    return 0;
}
```

### Çıktı

```
&&&10
??18
15%%%
```

## 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.



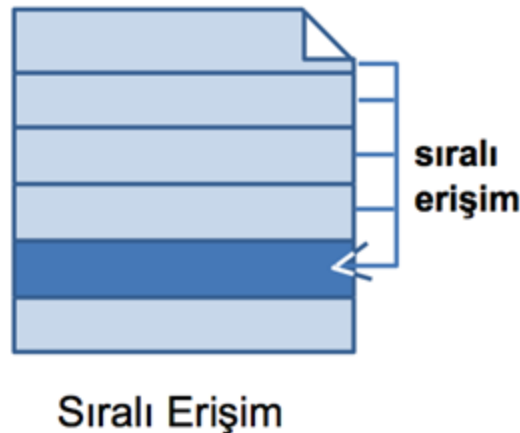


## 2. Dosya Kullanımı

- 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

## 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.



# 2.1. Sıralı Erişim...

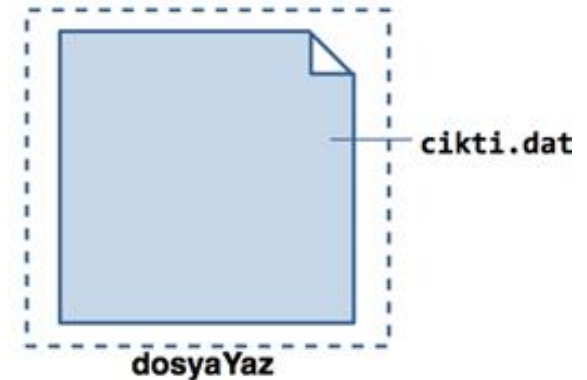
- 2.1.1. Dosya Çıktı İşlemleri
- Dosyaya veri yazdırmak için ofstream sınıfını aşağıda gösterildiği şekilde kullanılır.

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

↓                    ↓                    ↓                    ↓

sınıf                dosya                işletim                dosya  
adı                nesnesinin                sistemi                açma  
                     adı                                tarafından                modu  
    kullanılan  
    dosya adı

ofstream Sınıfının Kullanımı



dosyaYaz Nesnesi ve "cikti.dat" Dosyası

## 2.1. Sıralı Erişim...

- 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.

# 2.1. Sıralı Erişim...

- 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.

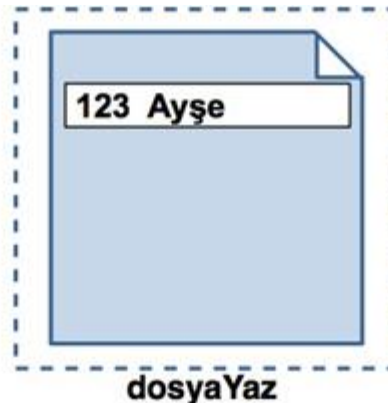
```
ofstream dosyaYaz ("cikti.dat",ios::out);  
if (!dosyaYaz)  
    cerr<<"dosya acilamadi";  
else { ... }
```

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

# 2.1. Sıralı Erişim...

- 2.1.1 Dosya Çıktı İşlemleri...
- **Dosyaya Yazma:** Dosyayı açıldıktan sonra çıktılarımızı aynı cout veri yoluna yollar gibi dosya nesnesine göndermemiz gerekir.

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



**dosyaYaz Nesnesi ile Dosyaya Yazma**

# 2.1. Sıralı Erişim...

- **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.

## 2.1. Sıralı Erişim...

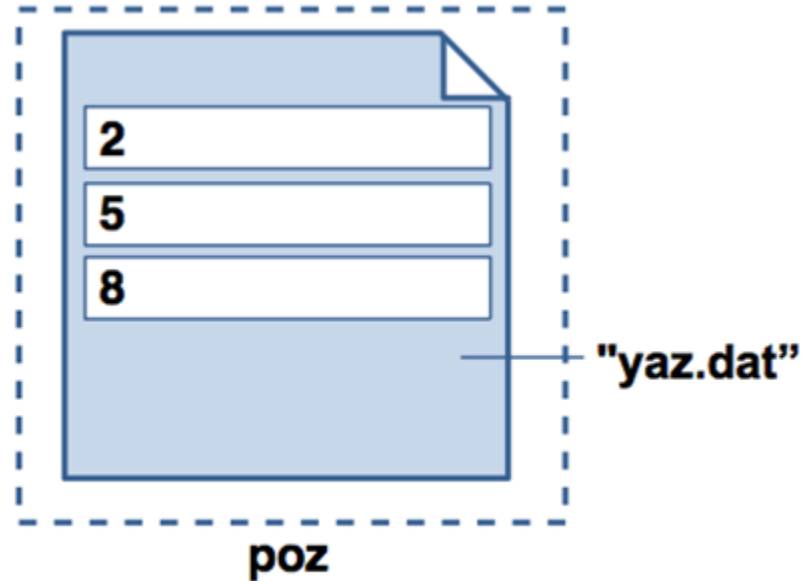
```
#include <iostream>
#include <fstream>
int main()
{
    ofstream poz("yaz.txt",ios::out);           // Dosya tanımı
    int no;
    cout<<"50 sayi giriniz:";
    for (int i=1;i<=50;i++){
        cin>>no;                                // Kullanıcıdan sayılar okunur
        if (no>0)             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

**Örnek Girdi**

2 -3 5 -7 8...



## 2.1. Sıralı Erişim...



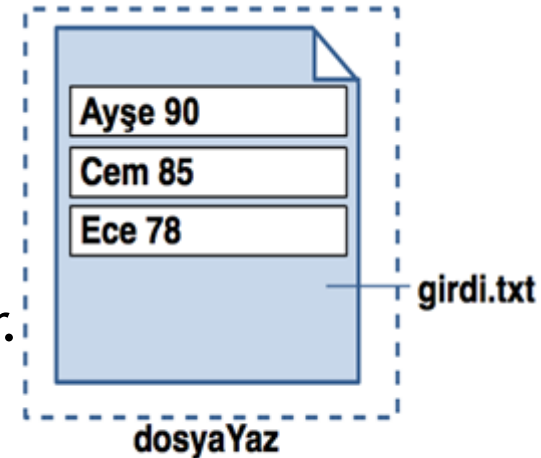
poz Nesnesi ile "yaz.dat" Dosyası

## 2.1. Sıralı Erişim...

- 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.

## 2.1. Sıralı Erişim...

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



"girdi.txt" Dosyası

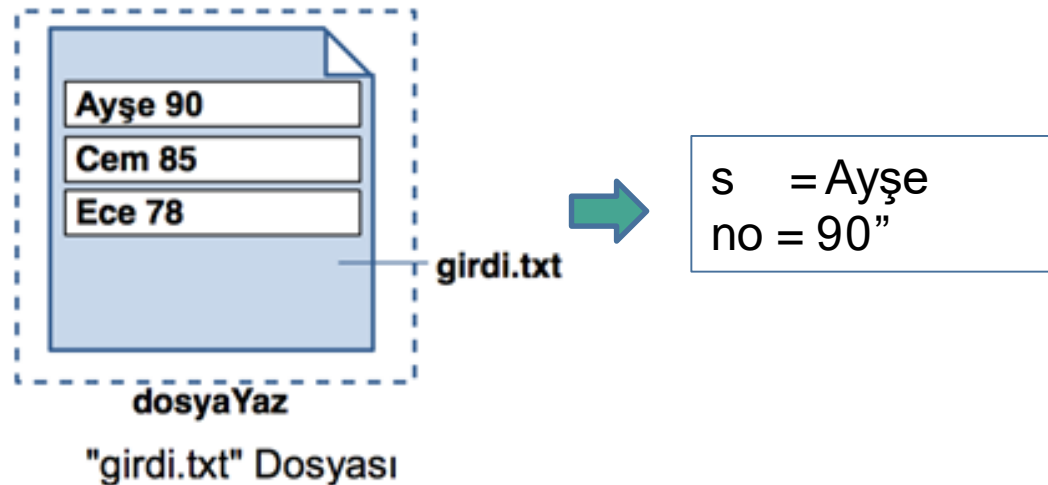
- Dosya açma modu aşağıdaki tanımda olduğu gibi yazılmazsa varsayılan mod olarak `ios::in` alınır.



# 2.1. Sıralı Erişim...

- 2.1.2. Dosya Girdi İşlemleri...
- **Dosyadan Okuma:** Dosyayı açıldıktan sonra girdiler aynı c i n veri yolundan okur gibi dosyaOku nesnesinden okunabilir.

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



## 2.1. Sıralı Erişim...

- 2.1.3. Dosya Sonu
- eof( ):
  - dosya sonu (**end off line**) fonksiyonu
  - dosyanın sonuna gelindiye true, daha okunacak veriler varsa ise false döndürür

# 2.1. Sıralı Erişim...

- 2.1.3. Dosya Sonu...

```
#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

## 2.1. Sıralı Erişim...

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

# 2.1. Sıralı Erişim...

- 2.1.4. Dosya Sonuna Ekleme...

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
void dosyadanYaz (ifstream &g, ofstream &c)
{
    char dizgi[80];
    g.getline(dizgi,80);
    while (!g.eof()){
        c<<dizgi;
        g.getline(dizgi,80);
        c<<endl;
    }
}
int main()
{
    ifstream f1("girdi1.txt");
    ifstream f2("girdi2.txt");
    ofstream f3("birlesik.doc",ios::app);
    dosyadanYaz(f1,f3);
    dosyadanYaz(f2,f3);
    f3.close();
    return 0;
}
```

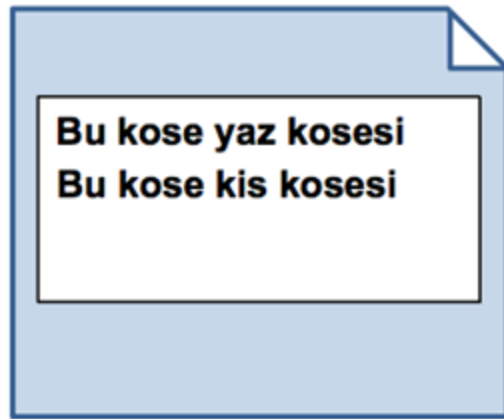
*// Dosyadan bir satır okunur*  
*// Dosyanın sonuna kadar döner*  
*// Satır dosyaya yazdırılır*

*// İlk girdi dosyası*  
*// İkinci girdi dosyası*  
*// Çıktı dosyası*  
*// İlk dosya yazdırılır*  
*// İkinci dosya yazdırılır*

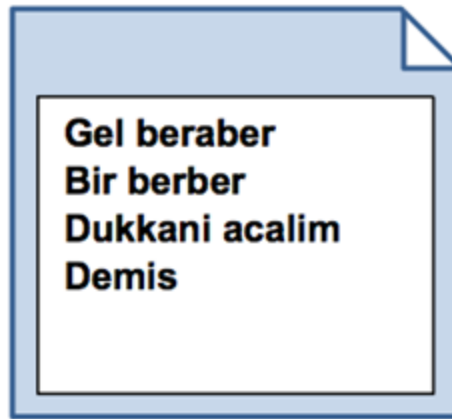


## 2.1. Sıralı Erişim...

- 2.1.4. Dosya Sonuna Ekleme...



**girdi1.txt**



**girdi2.txt**

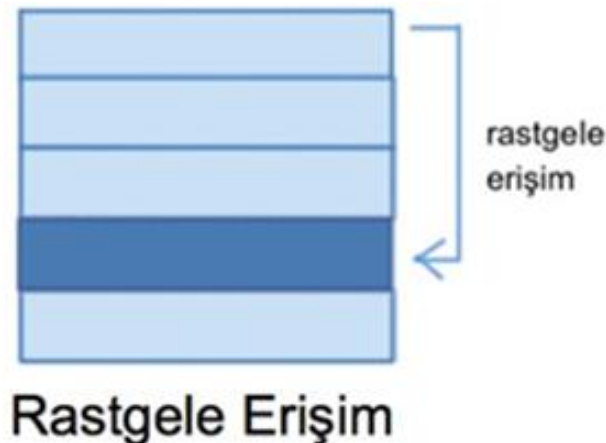


**birlesik.txt**

"birlesik.txt" Dosyası ve Dosyanın Sonuna Ekleme

## 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.



## 2.2. Rastgele Erişim...

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



### fstream Sınıfının Kullanımı

- `seekp()`: dosyanın herhangi bir yerine gitmek için kullanılır.
- Örnek:
  - dosyanın 500. byte'ında yer alan değeri okur  
`dosya1.seekp(500);`  
`dosya1>>x;`

# Çözümlü Sorular

## Soru

2. Aşağıda verilen örnek çıktıyı dikkate alarak main() fonksiyonunda eksik bırakılan kısımları, gerekli girdi çıktı fonksiyonlarını kullanarak doldurunuz.

```
int main()
{   float fiyat1 = 20.4087;
    float fiyat2 = 30.550 + 4.0 / 2.0;
    cout._____( _____);
    cout << "Fiyat(TL):";
    cout. _____ ( _____);
    cout<<fiyat1 <<"TL"<<endl;
    cout << "Fiyat(TL):";
    cout. _____ ( _____);
    cout<<fiyat2 <<"TL"<<endl;
    _____;
    cout. _____ ( _____);
    cout<< "Fiyat(TL):";
    cout. _____ ( _____);
    cout<< fiyat1 <<"TL"<<endl;
    cout<< "Fiyat(TL):";
    cout. _____ ( _____);
    cout<< fiyat2 <<"TL"<<endl;
    return 0;
}
```




# Çözümlü Sorular

## Cevap

2.

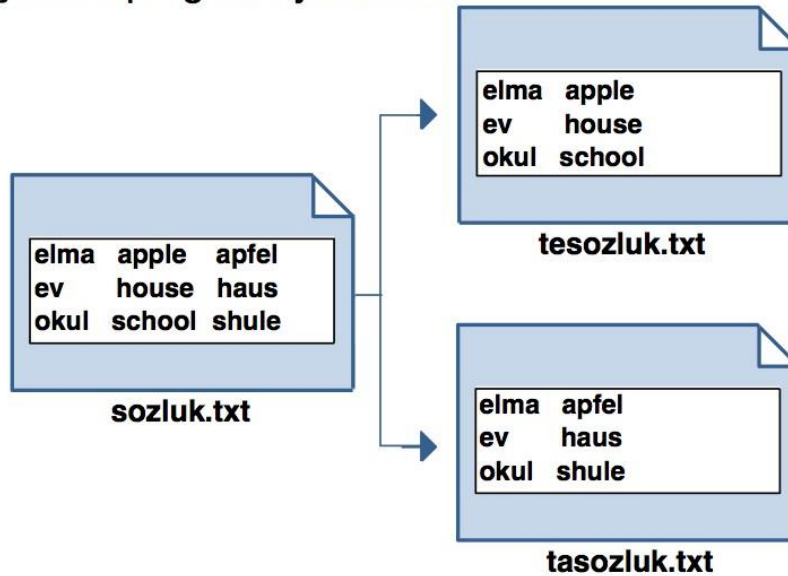
```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    float fiyat1 = 20.4087;
    float fiyat2 = 30.550 + 4.0 / 2.0;
    cout.fill('-');
    cout << "Fiyat(TL):";
    cout.width(22);
    cout<<fiyat1 <<"TL"<<endl;
    cout << "Fiyat(TL):";
    cout.width(20);
    cout<<fiyat2 <<"TL"<<endl;
    cout<<endl;
    cout.precision(3);
    cout<< "Fiyat(TL):";
    cout.width(19);
    cout<< fiyat1 <<"TL"<<endl;
    cout<< "Fiyat(TL):";
    cout.width(19);
    cout<< fiyat2 <<"TL"<<endl;
    return 0;
}
```



# Çözümlü Sorular

## Soru

4. Türkçe kelimelerin İngilizce ve Almanca karşılıkları aşağıda gösterildiği gibi sozluk.txt dosyasında yer almaktadır. Bu dosyadaki bilgileri kullanarak Türkçe-İngilizce ve Türkçe-Almanca olmak üzere iki ayrı sözlük dosyası yaratacak şekilde programı yazınız.



# Çözümlü Sorular

## Cevap

```
4. #include <iostream>
    #include <fstream>
    using namespace std;
    int main(void)
    {
        ifstream okuyucu("sozluk.txt");
        ofstream teyazici("tesozluk.txt");
        ofstream tayazici("tasozluk.txt");
        string turkce,ingilizce,almanca;
        while (!okuyucu.eof()){
            okuyucu>>turkce>>ingilizce>>almanca;
            teyazici<<turkce<<" "<<ingilizce<<"\n";
            tayazici<<turkce<<" "<<almanca<<"\n";
        }
        teyazici.close();
        tayazici.close();
        okuyucu.close();
        return 0;
    }
```

