



SAKARYA ÜNİVERSİTESİ
Bilgisayar ve Bilişim Bilimleri Fakültesi
Bilgisayar Mühendisliği Bölümü

DOSYALAR **(Files)**



Prof. Dr. Cemil ÖZ
Prof. Dr. Celal ÇEKEN
Doç. Dr. Cüneyt BAYILMIŞ

Konular

- ✓ Dosyalama İşlemleri
- ✓ Dosya Modları
- ✓ Metin Dosyaları - İkili Dosyalar
- ✓ Dosya Veri Yazma İşlemi
- ✓ Dosya Veri Okuma İşlemi
- ✓ Örnek
- ✓ Dosya Konum İşaretçileri
- ✓ Rasgele Erişimli Dosya İşlemleri
- ✓ Veritabanı Bağlantısı
- ✓ Örnek
- ✓ Kaynaklar

Dosyalama İşlemleri

- ✓ Verilerin sürekli saklanması/depolanması için dosyalardan yararlanılır.
- ✓ Bilgisayarın işlediği tüm verilerin sonuçta 1 ve 0'lardan oluştuğu unutulmamalıdır.
- ✓ Bir dosya (file) birden çok kayıt (record) alanından oluşur.
- ✓ Kayıt alanları karakter grupları ile karakterler de bit grupları ile gösterilir.
- ✓ Programlar dosyalara iki farklı şekilde ulaşırlar; sıralı ve rasgele.
- ✓ Yazdığınız program sizin ulaşım seçiminize göre değişiklik gösterecektir.
- ✓ Dosyaya erişim modunuz, sizin dosyadan veriyi nasıl okuyacağınızı, dosyaya veriyi nasıl yazacağınızı, dosyadaki verileri nasıl değiştireceğinizi veriyi dosyadan nasıl sileceğinizi ve benzeri durumları belirlemenizi sağlar.
- ✓ Bazı dosyalara iki şekilde de ulaşılabilir.

Dosyalama İşlemleri

Dosyalamayla ilgili 3 temel işlem

- ✓ Dosya açılır
- ✓ Veri yazılır, ya da okunur
- ✓ Dosya kapatılır

Dosyalarla ilgili işlemlerde aşağıda bulunan ve istream ile ostream sınıflarından türetilen sınıflar kullanılır (cin cout bu sınıflardan oluşturulan nesnelerdir.).

- ✓ **ofstream**: Dosyalara yazmak için kullanılan stream sınıfı
- ✓ **ifstream**: Dosyalardan okumak için kullanılan stream sınıfı
- ✓ **fstream**: Dosyalara yazmak için kullanılan stream sınıfı

Dosya açılırken dosya adı ve mod belirtilir

```
fstream dataFile("example.txt", ios::out | ios::app);
```

```
1 ofstream myfile;  
2 myfile.open ("example.bin", ios::out | ios::app | ios::binary);
```

class	default mode parameter
ofstream	ios::out
ifstream	ios::in
fstream	ios::in ios::out

Dosya Modları

Mod Bayrağı	Görevi
<code>ios::in</code>	Varolan dosyanın okunmak üzere açılması. Dosya yok ise hata döndürülür
<code>ios::app</code>	Dosyaya yazmak için kullanılır. Dosya yok ise oluşturulur. Tüm ekleme işlemleri dosya sonuna yapılır.
<code>ios::ate</code>	<code>ios::app</code> gibidir. Farklı olarak yazma işlemi herhangi bir yere yapılabilir.
<code>ios::binary</code>	Dosyanın ikili modda açılmasını sağlar. Varsayılan mod text dir
<code>ios::out</code>	Yazma modu. Dosya var ise içeriği boşaltılır.
<code>ios::trunc</code>	Dosya var ise içeriği temizlenir. <code>ios::out</code> için varsayılan moddur.
<code>ios::noreplace</code>	Dosya var ise hata verir.
<code>ios::nocreate</code>	Dosya yok ise oluşturulmasın. Open fonksiyonu dosya yok ise hata verecektir.

Metin Dosyaları - İkili Dosyalar

- ✓ İkili (binary) modta açılan dosya akışları giriş ve çıkış işlemlerini formattan bağımsız olarak gerçekleştirirler.
- ✓ Metin (text) dosyalarda ise bazı özel karakterlerin (satırbaşı ve satırbaşına dönüş karakterleri) formatından dolayı bazı dönüşüm işlemleri gerçekleşir.
- ✓ İkili dosyalar formatsızdır ve ASCII formatında saklanmazlar.

1	2	9	7	<EOF>
---	---	---	---	-------

1297 expressed in ASCII

49	55	57	55	<EOF>
----	----	----	----	-------

1297 as an integer, in binary

00000101	00010001
----------	----------

1297 as an integer, in hexadecimal

05	11
----	----

Dosya Veri Yazma İşlemi

✓ Dosya yazma işleminden önce dosya açılmalıdır.

① `ofstream dosyaYaz ("kayit.txt", ios :: out) ;`

<code>ofstream</code>	<code>dosyaYaz</code>	<code>("kayit.txt", ios :: out) ;</code>
sınıf adı	dosya nesnesinin adı	kullanılan dosya dosya açma modu

② `ofstream dosyaYaz ;`

`dosyaYaz . open ("kayit.txt") ;`

veya

`dosyaYaz . open ("kayit.txt", ios :: out);`

✓ Dosyaya veri yazma

`dosyaYaz << "Dosyaya yazıyorum \n";`

✓ Dosya yazma işleminden sonra dosya kapatılmalıdır. Herhangi bir kapatma işlemi yapılmazsa işletim sistemi dosyayı program bitince kapatır.

`dosyaYaz . close () ;`

Dosyadan Veri Okuma İşlemi

✓ Dosya okuma modunda açılmalıdır.

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

sınıf adı

dosya
nesnesinin
adı

kullanılan
dosya

dosya açma
modu
(default)

❷ `ifstream dosyaOku ;`

`dosyaOku . open ("kayit.txt") ;`

veya

`dosyaOku . open ("kayit.txt", ios :: out) ;`

✓ Dosyadan veri okuma

`dosyaOku >> değişken; veya dosyaOku . get (değişken);`

Dosyadan Veri Okuma İşlemi

- ✓ String ifadeler gibi dosya sonunda bir işaretçi ile sonlanır.
- ✓ Dosya sonuna gelinip gelinmediği kontrol edilebilir.

```
while ( ! dosyaOku . eof ( ) )    // end of file
```

veya

```
while ( dosyaOku . good ( ) )
```

Dosya İşlemi

- ✓ Dosyanın açılıp açılmadığını kontrol etme işlemi,

```
if (dosyaYaz . is_open ( ) )
```

```
    cout << "Dosya açıldı";
```

Örnek

```
#include <iostream>
#include <fstream>
using namespace std;

int main(){
    // dosya veri yazma işlemi
    ofstream dosyaYaz;
    dosyaYaz.open("kayit.txt", ios::out);
    dosyaYaz<<"Dosya Veri Yazma \n";
    dosyaYaz.close();

    // dosyadan okuma işlemi
    ifstream dosyaOku("kayit.txt", ios::in);
    if (dosyaOku.is_open())
        cout<<"Dosya Acma Basarili\n";
    char oku;
    while (!dosyaOku.eof()) {
        dosyaOku>>oku;
        cout<<oku;
    }
    dosyaOku.close();
    return 0;
}
```

Örnek 2

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
struct rehber {
    string ad;
    string tel;
};
int main()
{
    rehber rehberim;
    char devam='e';
    ofstream dosyaYaz;
    dosyaYaz.open("Rehber.txt",ios::app);
    do {
        cout<<"ad ve telefon bilgileri giriniz\n";
        cin>>rehberim.ad>>rehberim.tel;
        dosyaYaz<<rehberim.ad<<"\t"<<rehberim.tel<<"\n";
        cout<<"\n Yeni kayıt yapacak misiniz (e/h) ";
        cin>>devam;
    } while(!(devam=='h'));
    dosyaYaz.close();
```

```
ifstream dosyaOku;
dosyaOku.open ("Rehber.txt");
string isim;
cout<<"Aranilacak Adi Giriniz: ";
cin>>isim;
while(!dosyaOku.eof()){
    // while (dosyaOku>>rehberim.ad>>rehberim.tel)
    dosyaOku>>rehberim.ad>>rehberim.tel;
    if (isim == rehberim.ad)
        cout<<rehberim.ad<<"\n"<<rehberim.tel;
}
dosyaOku.close();
return 0;
}
```

- ✓ C++'da dosyaya herhangi bir yapı zorlanmaz.
- ✓ Dosyaları programcı formatlı veri girişi ile yapılandırır. Örneğin ad, tel bilgilerinin tek bir kişiye ait olması

[dosyarehber.cpp](#)

Dosya Konum İşaretçileri

- ✓ Ardışık olarak bir dosyadan okuma işleminde, programlar işleme dosyanın başından başlar ve istenilen veri bulunana kadar ard arda tüm verileri okur.
- ✓ Bir dosyanın her seferinde başından itibaren işleme sokulması gereksiz işlem ve zaman kaybına neden olur.
- ✓ Bu problem, hem okuma hem de yazma işlemi için dosya konum işaretçileri ile çözülür.
- ✓ Dosya konum işaretçileri işlem yapılacak (okuma/yazma) bir sonraki baytın numarasını tutar.
- ✓ `ifstream` \Rightarrow `seekg` (seek get) `ofstream` \Rightarrow `seekp` (seek put)
- ✓ Ayrıca, dosyada konumlandırmanın yönü belirtilmelidir.
 - ✓ `beg` : başlangıca göre
 - ✓ `cur` : geçerli/mevcut konuma göre
 - ✓ `end` : sonuna göre

Dosya Konum İşaretçi Modları

Komut	Okuma/Yazma Pozisyonunu Etkileme Durumu
<code>dosyaNesnesi.seekg (n);</code>	Dosyanın n. baytına konumlandır. n=0 ise dosya başına konumlandır.
<code>dosyaNesnesi.seekp(n, ios::beg);</code> <code>dosyaNesnesi.seekp(32L, ios::beg);</code>	Yazma pozisyonunu dosya başından 33. bayta kur.
<code>dosyaNesnesi.seekp(-10L, ios::end);</code>	Dosya sonundan 11. bayta yazma pozisyonunu kur.
<code>dosyaNesnesi.seekp(120L, ios::cur);</code>	Geçerli pozisyondan 121. bayta yazma pozisyonunu kur.
<code>dosyaNesnesi.seekg (2L, ios::beg);</code>	Dosya başından 3. bayta okuma pozisyonunu kur.
<code>dosyaNesnesi.seekg (-100L, ios::end);</code>	Dosya sonundan 101. bayta okuma pozisyonunu kur.
<code>dosyaNesnesi.seekg (40L, ios::cur);</code>	Geçerli pozisyondan 41. bayta okuma pozisyonunu kur.
<code>dosyaNesnesi.seekg(0L, ios::end);</code>	Dosya sonuna okuma pozisyonunu kur.

Dosya Konum İşaretçileri

- ✓ Dosya üzerinde bulunan mevcut konumu öğrenme;

`ifstream` \Rightarrow `tellg` (tell get)

`ofstream` \Rightarrow `tellp` (tell put)

- ✓ Dosyadaki mevcut konumu öğrenme

`long konum;`

`konum = dosyaNesnesi.tellg();`

Rasgele Erişimli Dosya İşlemleri

- ✓ Rasgele erişimli dosya ile ayrı kayıtlara, diğer kayıtları aramaya gerek kalmadan doğrudan ve hızlı bir şekilde erişilebilir.
- ✓ C++ ta bir dosya için belirli bir yapı zorunluluğu yoktur.
- ✓ Rasgele erişim için en kolay yöntemlerden biri dosyadaki tüm kayıtların/kayıt alanlarının aynı büyüklükte gerçekleştirilmesidir. Böylelikle bir kaydın konumu hızlı bir şekilde tespit edilebilir.
- ✓ Sabit kayıt alanı kullanımı ile dosyada değişiklikler/güncellemeler, diğer verilere zarar vermeden daha kolay gerçekleştirilebilir.

- ✓ Dosyaya veri yazma

```
dosyaNesnesi . write ( reinterpret_cast < const char * > (&kayit), sizeof (kayit) ) ;
```

- ✓ Dosyadan veri okuma

```
dosyaNesnesi . write ( reinterpret_cast < char * > (&kayit), sizeof (kayit) ) ;
```


Rasgele Erişimli Dosya İşlemleri

✓ Rasgele erişimli dosyalarda konum işaretçisini konumlandırma işlemi için ardışık dosyalarda da kullanılan **seekg** ve **seekp** komutları kullanılır.

```
dosyaNesnesi . seekp ( sayi * sizeof (SinifAdi) ) ;
```

```
dosyaNesnesi . seekp ( nesneAdi. fonksiyon () - 1) * sizeof (SinifAdi) ) ;
```

```
dosyaYaz . seekp ( rehberim. al () - 1) * sizeof (rehber) ) ;
```

Rasgele Erişimli Dosya İşlemleri Örneği

```
// Bu program boş kayıt alanlarına sahip bir dosya oluşturur.

#include <iostream>

#include <fstream>

using namespace std;

// Invtry yapısı

struct Invtry
{
    char desc[31];
    int qty;
    float price;
};

void main(void)
{
    fstream inventory("invtry.dat", ios::out | ios::binary);

    Invtry record = { "", 0, 0.0 };    // {"sakarya", 54, 54.0 } sabit kayıt

    // Boş kayıtlar yazılıyor
    for (int count = 0; count < 5; count++)
    {
        cout << "Now writing record " << count << endl;
        inventory.write((char *)&record, sizeof(record));
    }
    inventory.close();
}
```

Rasgele Erişimli Dosya İşlemleri Örneği

```
// Bu program envanter dosyasının içeriğini gösterir.
#include <iostream>
#include <fstream>

// Invtry yapısı
struct Invtry
{
    char desc[31];
    int qty;
    float price;
};

void main(void)
{
    fstream inventory("invtry.dat", ios::in | ios::binary);
    Invtry record = { "", 0, 0.0 };
    // kayıtlar okunuyor ve gösteriliyor
    inventory.read((char *)&record, sizeof(record));
    while (!inventory.eof())
    {
        cout << "Description: ";
        cout << record.desc << endl;
        cout << "Quantity: ";
        cout << record.qty << endl;
        cout << "Price: ";
        cout << record.price << endl << endl;
        inventory.read((char *)&record, sizeof(record));
    }
    inventory.close();
}
```

Veritabanı Bağlantısı

Gereksinimler

- ✓ Mysql sunucusu
- ✓ Mysql sunucuda çalışan bir veritabanı uygulaması
- ✓ Mysql C++ sürücüsü (kitaplık)

```
#include <mysql.h>
#include <stdio.h>

main() {
    MYSQL *conn;
    MYSQL_RES *res;
    MYSQL_ROW row;

    char *server = "localhost";
    char *user = "root";
    char *password = "PASSWORD"; /* set me first */
    char *database = "mysql";

    conn = mysql_init(NULL);

    /* Connect to database */
    if (!mysql_real_connect(conn, server,
        user, password, database, 0, NULL, 0)) {
        fprintf(stderr, "%s\n", mysql_error(conn));
        exit(1);
    }

    /* send SQL query */
    if (mysql_query(conn, "show tables")) {
        fprintf(stderr, "%s\n", mysql_error(conn));
        exit(1);
    }

    res = mysql_use_result(conn);

    /* output table name */
    printf("MySQL Tables in mysql database:\n");
    while ((row = mysql_fetch_row(res)) != NULL)
        printf("%s \n", row[0]);

    /* close connection */
    mysql_free_result(res);
    mysql_close(conn);
}
```

Kaynaklar

- ✓ Robert Lafore, Object Oriented Programming in C++, Macmillan Computer Publishing
- ✓ Deitel, C++ How To Program, Prentice Hall
- ✓ Prof. Dr. Celal ÇEKEN, Programlamaya Giriş Ders Notları