14. DOSYALAR

Giriş

Verilerin elektronik ortamda saklanması ile ilgili çalışmaların başlama tarihi, programlama macerasının başlangıcına kadar geriye gider. Yirminci Yüzyılın başlarında kullanılmaya başlayan delikli kartları ve 1960'ların başlarında kullanılmaya başlanan manyetik bantları bir tarafa bırakarak, 1970'lerin başılarını bu konun başlangıç tarihi olarak sayabiliriz. Çünkü 1970'lerin başlarında harddiskler hayatımıza girmiş ve bugünkü anlmada veriler elektronik ortamda saklanmaya başlamıştı. Daha sonra 1980'lerde ilişkisel veritananı çalışmalarının başlaması, 1990'larda SQL dilinin ortaya çıkması, aynı dönemlerde object oriented veritabanı çalışmalarının yapılması, günümüzde çok büyük miktarlardaki verinin elektronik ortamlara kolay ve güvenli şekilde taşınmasını ve bu ortamlarda işlenmesini sağlayan teknolojilerin hayatımıza girmesini sağladı.

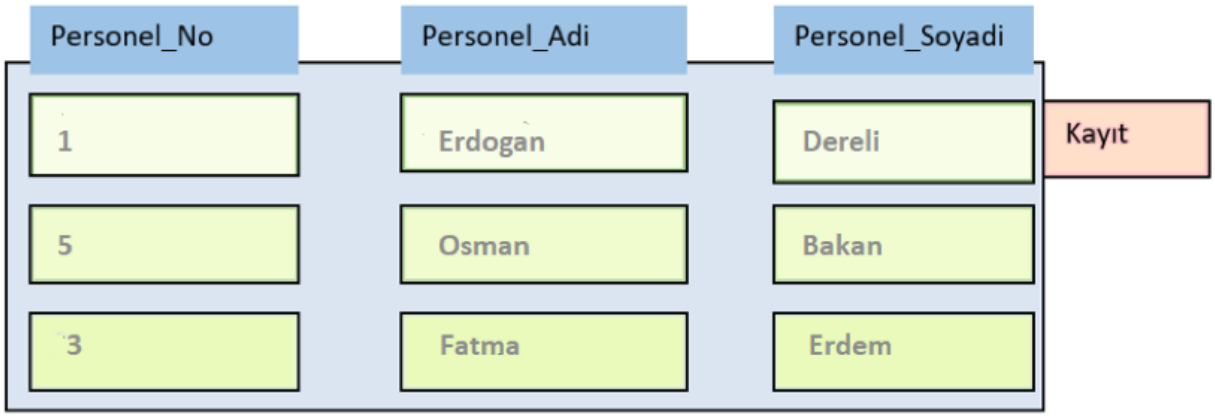
1970 'li yıllarda bilgisayar ortamında verilerin saklanması için dosya sistemleri kullanılmaktaydı (Bugün bazı özel alanlarda dosya sistemlerinin kullanılmasına az da olsa devam edilmektedir). Dosya kavramı için, "verinin ikinci seviye bellek ünitelerinde(Harddisk, flashdisk vb.) saklandığı yapıdır" şeklinde bir tanım yapabiliriz. Ya da dosya, veri saklama birimlerinde depolanan veri topluluklarına verilen isimdir de denebilir.

Yukarıdaki paragrafta birden çok sayıda *veri* kavramı geçti. Dosya sistemleri veya veri tabanlarından söz edilirken bu kavrama sık sık başvurulur ve çoğu kere de *veri* kavramının *bilgi* kavramı ile karıştığı görülür. Bunun önüne geçebilmek için aşağıda veri ve bilgi kavramlarının tanımını verelim.

Veri (data), Bilgisayarın alabildiği, işleyebildiği, sonuç üretebildiği ve saklayabildiği her şey veridir. Örneğin işlenmemiş deney sonuçları, gözlem sonuçları ve izlenimlerimiz birer veridir. Veri için anahtar kelime işlenmemiş olma durumudur diyebiliriz.

Bilgi ise, verinin işlenmesi sonucu ortaya çıkan her türlü gerçek, malumat ve kavrayıştır. Bugün teknolojinin geldiği noktada her türlü veri çok hızlı ve doğru bir biçimde işlenip bilgiye dönüştürülüyor ve karar verme noktasında olanlara sunuluyor. Yakın gelecekte bu konuda yeterli farkındalık geliştiremeyen bütün organizasyonlar ne yazık ki rekabet avantajlarını süratle kaybedeceklerdir.

Veri ve bilgi tanımından sonra şimdi de dosya nedir sorusunun cevabını netleştirmeye çalışalım. Bunun için, bir işyerinde çalışanların bilgilerinin tutulduğu bir dosya düşünelim. Vereceğimiz örnek oldukça basit ama konuya açıklık getirecek özellikte olsun. Dosyada her çalışanın personel numarası, adı ve soyadı yer alsın. Şekil 14.1 'de dosyanın yapısı gösterilmiştir.



Şekil 14.1 Dosya Örneği

Dosyada yer alan her satıra **kayıt** denir. Şekil 14.1'den de görülebileceği gibi kayıtlar birbirleri ile ilişkili alanlardan (field) oluşur. Ayrıca her bir kaydın, işyerinde çalışan bir kişiye ait olduğuna da dikkat ediniz.

Dosya sistemlerinin kullanılması sırasında bazı dezavantajlar ortaya çıkmıştır. Yeri gelmişken bu dezavantajların önemlilerini de burada sıralayalım:

Veri tekrarı: Aynı veri çeşitli dosyalarda birden fazla yer alabilmektedir buda sistemin hantallaşmasına neden olur. Mesela bir stok dosyasında stok numarası verisinin malzeme dosyasında, fatura dosyasında ve ambar girişi dosyasında yer alması gibi.

Verinin birkaç dosyada güncellemesi: Veri birden fazla dosyada tekrar edilebildiği için, verinin bir dosyada güncellenip diğerlerinde güncellenmemesi Veri Bütünlüğünün (Data Integrity) bozulmasına neden olabilir. Buna bağlı olarak birbiri ile çelişen raporlar üretilmektedir.

Belleğin tekrarlı bilgi nedeniyle israfı: Aynı verinin birden fazla dosya içinde bulunması nedeniyle kullanılan veri hard diskte fazla yer işgal edecek. Yani hard disk tekrarlı veriler için kullanılmış olacaktır.

Sadece belirli bir dilin kullanılması: Her programlama dilinin kendisine ait bir dosyalama sistemi vardır. Aynı işyerinde farklı diller kullanılarak geliştirilen programlar aracılığı ile oluşturulan dosyalardan ortak rapor üretilemeyeceği için birçok bilginin tekrar tekrar farklı dosyalara girilmesi gerekebilir.

Dosya sistemlerinin yukarıda sayılanlar ve benzeri dezavantajlarının ortaya çıkmasından dolayı verilerin saklanmasıyla ilgili teknolojiler bugün gelişerek ve dönüşerek çok farklı noktalara taşınmıştır. Ancak yine de dosya sistemleri yukarıda da belirtildiği gibi kullanılmaya devam edilmektedir.

C++ iki temel dosya tipini desteklemektedir. Bunlardan birincisi sıralı erişimli dosyalar, ikincisi ise doğrudan erişimli dosyalardır. Biz bu bölümde C++'ın desteklediği dosya tiplerinden sıralı erişimli dosyaları ele alacağız ve sıralı erişimli dosyalara kayıt yapılması, önceden dosyaya yapılmış kayıtların okunması, kayıt silinmesi ve kayıtların güncellenmesi konularını ele alacağız ve bu başlıkları birer örnek çözerek açıklayacağız.

14.1 C++'ta Dosya Giriş/Çıkış İşlemleri

C++ 'ta dosyalara giriş çıkış işlemleri üç adet sınıftan oluşturulan nesnelerle tanımlanır.

ofstream: Dosya oluşturmak ve dosyaya Yazmak için kullanılır

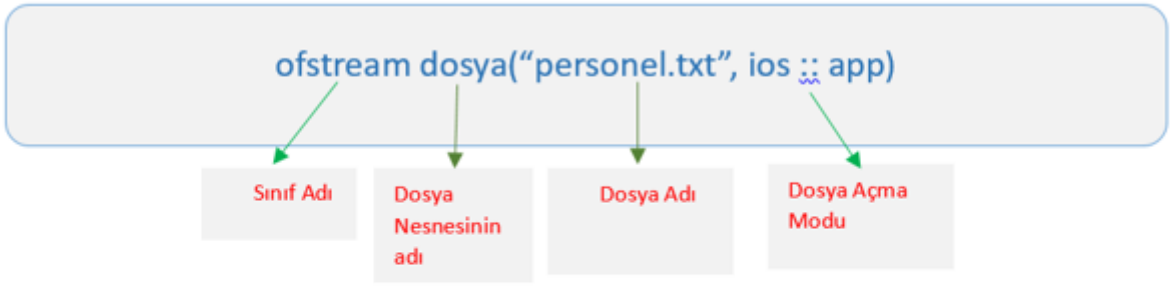
ifstream: Dosyadan okuma yapmak için kullanılır.

fstream: Dosya oluşturmak, dosyaya veri yazmak ve dosyadan veri okumak için kullanılır.

C++'ta program içerisinde dosya işlemleri yapabilmek için **fstream** dosyasının programa eklenmesi gerekmektedir.

14.2 C++' ta Sıralı Erişimli Dosyalara veri Yazmak

Verilerin dosyaya yazılabilmesi için önce dosyanın oluşturulması gerekmektedir. Bunun için *ofstream* sınıfı kullanılmaktadır.



- Dosyayı açtıktan sonra çıktılar **cout** veri yoluna gönderiliyor gibi dosya nesnesine gönderilmesi gerekir.

- *dosya<<prsNo<<prs.Ad<<prs.Soyad;*

Dosya Açma Modları

C++ 'ta programcıların tercihine göre dosyalar üzerinde farklı işlemlerin yapılması için özel komutlar vardır. Bu komutlara ***mod*** denir. Tablo 14.1'de dosya açma modları verilmiştir.

Sıra No	Dosya Açma Modu
1	**ios::out** ofstream sınıfı için varsayılan mod. Dosyayı oluşturur ve yazma modunda açar
2	**ios::app** Dosya yoksa dosyayı oluşturur ve ekleme modunda açar. Dosya varsa dosyayı ekleme modunda açar.
3	**ios::in** ifstream için varsayılan mod. Dosyayı okuma modunda açar
6	**ios::trunc** Dosya varsa dosyayı siler
7	**ios::ate** Başlangıç konumunun dosyanın sonuna set edilmesini sağlar.
8	**ios::binary** Dosyayı binary modda açar

open() Fonksiyonu Kullanarak Dosya açmak

```
ofstream dosya;
dosya.open("personel.txt", ios :: app);
```

open() fonksiyonu ile dosya açılırken önce bir dosya ismi, sonra da isteğe bağlı olarak dosya açma modu kullanılmaktadır.

C++ 'ta Dosya Açma İşlemlerinde Hata Kontrolü

C++'ta Dört adet önceden tanımlı akış nesnesi vardır.

cin: Standart giriş
cout: Standart çıkış
cerr: Standart hata
clog: Bellekte saklanan cerr

```
İf(!dosya)
{
  cerr<<"dosya açılamadı";
  exit(1)
                    }
```

C++ 'ta Dosya Kapamak

C++ 'ta Program bitmeden dosyayı kapatmak için **close()** fonksiyonu kullanılır.

dosya.close();

Program 14.1 : Bir iş yerinde çalışan personelin personel numarası, adı ve soyadının kaydedileceği sıralı erişimli dosya oluşturan ve bu dosyaya işyerinde çalışan personelin bilgilerini kaydeden C++ programını yazınız.

```
2  #include<fstream>
3  #include<string>
4  #include<conio.h>
5  #include<locale.h>
6  using namespace std;
7  struct pers{
8          string prsNo;
9          string prsAd;
10         string prsSoyad;
11             };
12 int main() {
13     setlocale(LC_ALL,"Turkish");
14     pers prs;
15     char ch='e';
16     ofstream dosya("personel.txt",ios::app);
17     if(!dosya)
18     {
19             cerr<<"dosya acilmadi"<<endl;
20             exit(1);
21             }
22     do
23     {
24     cout<<"Personel No       :"; cin>>prs.prsNo;
25     cout<<"Personel Adı       :"; cin>>prs.prsAd;
26     cout<<"Personel Soyadı    :"; cin>>prs.prsSoyad;
27     dosya<<prs.prsNo<<" "<<prs.prsAd<<" "<<prs.prsSoyad<<endl;
28     cout<<"veri girişine devam edecek misiniz  : "; ch=getche();cout<<endl;
29     }while(ch=='e');
30     dosya.close();
31     return 0;
32 }
```

Program 14.1 Sıralı erşimli dosyaya veri yazılması

Program 14.1'de önce programın çalışması için gerekli olan başlık dosyaları programa dâhil edildi. Arkasından string tipinde üç elemanı(prsNo, prsAd, prsSoyad) olan bir yapı tanımlandı. main() fonksiyonu içerisinde ***prs*** isimli bir yapı değişkeni tanımlandı. Daha sonra ***ofstream*** sınıfından ***dosya*** isimli bir nesne oluşturuldu ve parametre olarak ***personel.txt*** dosya ismi ve dosya açma modu olarak ta **ios::*app*** belirlendi.

ofstream dosya("personel.txt",ios::app);

program ilk defa çalıştırıldığında çalışma sırası bu satıra geldiği zaman disk üzerinde(Bir yol tanımlaması yapılmadığı için programın çalıştığı klasörde) personel.txt dosyası oluşturulacak ve ekleme modunda açılacaktır. Eğer program daha önceden çalıştırılmış ve personel.txt dosyası oluşmuş ise bu ifade daha önce oluşturulmuş olan dosyayı ekleme modunda açacaktır.

Programda 17-21. Satırlar arasındaki ifadelerle disk üzerinde dosya kontrolü yapılacak ve hata dönmez ise program 22. Satırdan çalışmaya devam edecektir. Eğer bu kontrolden hata dönerse programın çalışması sona erecektir.

22. ve 29. satırlar arasındaki do….while döngüsü içerisinde klavyeden ‘e’ tuşuna basıldığı sürece personel bilgisi girilecek ve her bir personele ait veri girildikten sonra;

dosya<<prs.prsNo<<" "<<prs.prsAd<<" "<<prs.prsSoyad<<endl;

ifadesi verilerin personel.txt dosyasına yazılmasını sağlayacaktır.

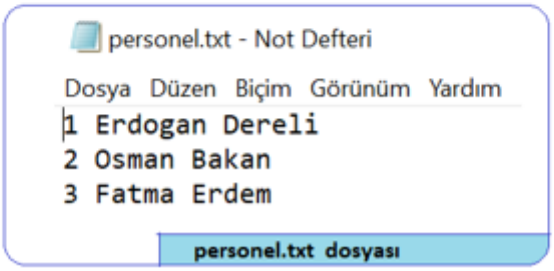
Veri girişi tamamlandığında kullanıcı ‘e’ tuşu dışında bir tuşa basarak programın çalışmasını sona erdirecektir.

Aşağıda, programın çalışması sırasındaki ekran görüntüsü ve programın çalışması tamamlandıktan sonra oluşan personel. txt dosyası verilmiştir.

```
Personel No       :1
Personel Adı      :Erdogan
Personel Soyadı   :Dereli
veri girişine devam edecek misiniz  : e
Personel No       :2
Personel Adı      :Osman
Personel Soyadı   :Bakan
veri girişine devam edecek misiniz  : e
Personel No       :3
Personel Adı      :Fatma
Personel Soyadı   :Erdem
veri girişine devam edecek misiniz  : h

--------------------------------
Process exited after 70.72 seconds with return value 0
Press any key to continue . . .
```

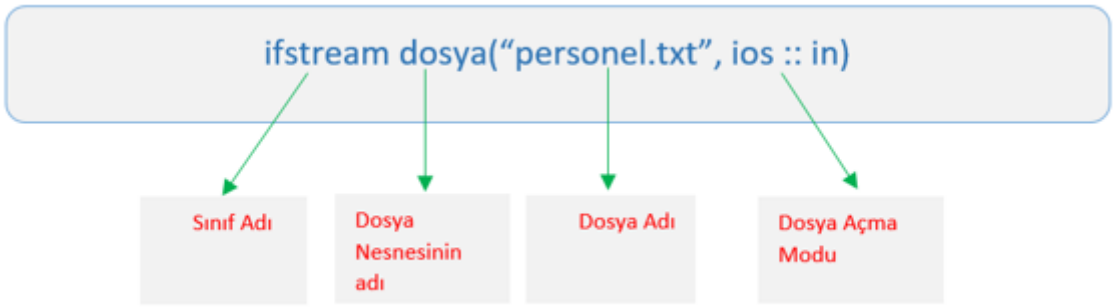
Program 14.1'in Veri Giriş Ekranı



personel.txt dosyası

14.3 C++’ ta Sıralı Erişimli Dosyalardan Veri Okumak

C++’ta sıralı erişimli dosyalardan veri okumak için ***ifstream*** sınıfı kullanılır. Daha önce oluşturulmuş olan dosyadan verileri okumak için *ifstream* sınıfından bir nesne oluşturulur. Dosya, oluşturulan *ifstream nesnesinin* varsayılanı modunda (ios::in) açılır. Dosya açıldıktan sonra girdiler cin veriyolundan okunuyormuş gibi oluşturulan nesneden okunur. Aşağıda ifstream sınıfından nesne oluşturulması ve dosyanın okumada modunda açılması gösterimiştir.



Program 14.2 : Program 14.1'de oluşturulan ve içerisine bir işyerinde çalışan personelin bilgilerinin kaydedildiği *personel.txt* dosyasını okuma modunda açan ve verilerin bu dosyadan okuyarak ekrana yazılmasını sağlayan programı yazınız?

```
1  #include <iostream>
2  #include<fstream>
3  using namespace std;
4  struct pers{
5        string prsNo;
6        string prsAd;
7        string prsSoyad;
8              };
9  int main() {
10     pers prs;
11     ifstream dosya("personel.txt",ios::in);
12     if(!dosya)
13     {
14         cerr<<"dosya acilmadi"<<endl;
15         exit(1);
16     }
17     while(!dosya.eof())
18      {
19         dosya>>prs.prsNo>>prs.prsAd>>prs.prsSoyad;
20         if(!dosya.eof())
21            cout<<prs.prsNo<<" "<<prs.prsAd<<" "<<prs.prsSoyad<<endl;
22               }
23     dosya.close();
24     return 0;
25 }
```

Program 14.2 Sıralı erişimli Dosyadaki verileri listelemek

Program 14.2'de önce programın çalışması için gerekli olan başlık dosyaları programa dâhil edildi. Arkasından string tipinde üç elemanı(prsNo, prsAd, prsSoyad) olan bir yapı tanımlandı. main() fonksiyonu içerisinde ***prs*** isimli bir yapı değişkeni tanımlandı. Daha sonra ***ifstream*** sınıfından ***dosya*** isimli bir nesne oluşturuldu ve parametre olarak ***personel.txt*** dosya ismi ve dosya açma modu olarak ta **ios::*in*** belirlendi.

ifstream dosya("personel.txt",ios::in);

program ilk defa çalıştırıldığında çalışma sırası bu satıra geldiği zaman disk üzerinde bulunan *personel.txt* dosyası okuma modunda açılacaktır.

Eğer program çalıştığı sırada *personel.txt* dosyası disk üzerinde bulunmuyorsa, programda 12. ve 16. satırlar arasında yazılan ifadelerden dolayı ekrana "*dosya açılmadı*" mesajını yazacak ve programdan çıkılacaktır.

Program 14.2'de 17. ve 22. Satırlar arasında while döngüsü oluşturulmuş ve bu döngünün dosya sonuna kadar çalışması sağlanmıştır. Program 14.2'de yazılan while döngüsü aşağıda gösterilmiştir.

```
17      while(!dosya.eof())
18      {
19          dosya>>prs.prsNo>>prs.prsAd>>prs.prsSoyad;
20          if(!dosya.eof())
21              cout<<prs.prsNo<<" "<<prs.prsAd<<" "<<prs.prsSoyad<<endl;
22              }
```

Döngü içerisinde 19. Satırda yazılan ifade ile dosyadaki kayıtların sıra ile okunması, 21 satırda yazılan ifadeyle de okunan kaydın ekrana yazılması sağlanmıştır.

Aşağıda, program 14.1'in ekran çıktısı gösterilmiştir.

14.4. C++' ta Sıralı Erişimli Dosyalardan Kayıt Silmek

Program 14.3 'te, *personel txt* dosyasından, klavyeden personel numarası girilen kaydın silinebilmesi için önce *personel.txt* dosyası okuma modunda açılmış ve geçici olan bir dosya (*gpersonel.txt*) yazma modunda açılmıştır. *personel.txt* dosyasından okunan kayıtlar, silinecek kayıt hariç *gpersonel.txt* dosyasına yazılmıştır. İşlem tamamlandıktan sonra *personel.txt* dosyası silinmiş ve *gpersonel.txt*'nin ismi *personel.txt* olarak değiştirilmiştir. Bu işlemlerin sonunda dosyadan klavyeden girilen kaydın silinmesi sağlanmıştır.

14.5. C++' ta Sıralı Erişimli Dosyalarda Kayıt Güncellemek

Program 14.4, *personel.txt* dosyasındaki kayıtların güncellenmesi için yazılmıştır. Bu programda da program 14.3'e benzer şekilde kayıtlar *personel.txt* 'den sırayla okunarak gpersonel.txt 'ye yazılması sağlanmış ve sıra güncellenmesi gereken kayda geldiğinde güncellenecek kayıtla ilgili bilgiler kullanıcıya ekranda gösterilmiş, programda kullanıcının güncel bilgileri klavyeden giriş yapabilmesi için düzenleme yapılmıştır. Kullanıcının güncellenecek kayıtla ilgili yeni bilgilerin girişini yapmasından sonra bu bilgilerin de *gpersonel.txt* 'ye yazılması sağlanmıştır. *personel.txt* 'deki tüm kayıtlar, güncellenen kayıt dahil *gpersonel.txt* 'ye yazılması tamamlandıktan sonra *personel.txt* silinmiş ve *gpersonel.txt* 'nin ismi *personel.txt* olarak değiştirilerek güncelleme işlemi tamamlanmıştır.

Program 14.5: Bu ünitede 14.2'de text dosyaya kayıt ekleme, 14.3'te text dosyadan okunan kayıtların listelenmesi, 14.4'te text dosyadan kayıtların silinmesi ve 14.5'te text dosyadaki kayıtların güncellenmesi ile ilgili ayrı ayrı programlar yazılmıştır. Program 14.5'te yukarıda yazılan programları birer fonksiyon haline getiriniz, ek olarak menü isimli bir fonksiyon oluşturunuz ve programda switc –case yapısı kullanarak yukarıda sayılan bütün bu özelliklere kullanıcının tek ekrandan ulaşmasını sağlayınız?

Bölüm Özeti

Bu bölümde C++ akışlarını kullanarak dosya bağlantılı faaliyetlerin nasıl yürütüldüğü öğrendik. Bu faaliyetler dosyaların oluşturulması, dosyalara veri kaydedilmesi, hataların kontrol edilmesi, Dosyalardan kayıt silinmesi ve dosyalardaki kayıtların güncellenmesi başlıkları altında incelendi. Son olarak ta bütün bu operasyonların bir menü altın da nasıl toplanacağı, programın fonksiyonlar halinde nasıl yazılacağı gösterildi.