12. Saat : Dosyalar I (Files)

Dosyalar (Files)

Programın çalışma esnasında her türlü değişken içinde tutulan ve işlenen bilgiler RAM (Read Access Memory) bellekte tutulur ve program sona erdiğinde RAM'dan ayrılmış olan bellek alanı tekrar işletim sistemine geri verilir. Dolayısı ile RAM'da kullanılan alan boşaltılacak ve aynı verilere erişmek imkansız olacaktır. Bu durumda uzun vadeli olarak kullanacağımız verilerin saklanması gerekir. Verilere bilgisayar kapatılıp açıldıktan sonra da ulaşabilmek istiyorsak dosyaları kullanmamız gerekir. Bu durumda veriler disket, CD, sabit disk gibi ortamlara kaydedilir ve gerektiğinde tekrar okutulabilir.

Dosya ve Akış (Stream) Nedir?

Peki, bu ortamlarda veriler nasıl tutulur? Yan bellekler (secondary storage device) denilen sabit disk, disket veya benzer ortamlara bilgiler kalıcı olarak yazılır. Yazılırken her veri kümesi bir isim altında toplanır. Bu veri kümeleri baştan başlayıp sona kadar sıralı `1` ve `0`'dan oluşan verilerdir.

C dilinde Akış (Stream) ve Dosya aslında aynı anlamda kullanılır. Bir önceki paragrafta da belirttiğimiz üzere sıralı veriler suyun akışına benzer şekilde kaplara dolar gibi değişkenlere dolarlar, yani atanırlar. C dili her dosyayı ardışık byte'ların akışı gibi algılar. Bu nedenle akış (stream) olarak adlandırılırlar.

Bütün C programlarında program çalıştığında standart olarak daha önceden aşina olduğumuz `stdin`, `stdout` ve `stderr` akışları (streams) açılır. Akışlar, çalışan program ile dosya arasında bir tampon bölge oluştururlar. Suyun geçici bir depoda toplanıp sonra başka bir yere aktarılması örneğinde olduğu gibi, bilgiler bu tampon bölgede biriktirilir ve ilgili depolama aygıtına gönderilir.

Her dosya sonunda bitişi gösteren bir dosya sonu işareti (end of file marker) vardır. RAM de çalışan programa veya aygıta bilgi akışı sırasında dosya (veya akış) sonunun geldiği bu işaretten anlaşılır.

Dosya Türleri

Dosyalar (akışlar/streams) metin ve ikili dosyalar (text and binary files) olarak ikiye ayrılır. Çalışılacak dosya türüne göre dosyalar açılır ve kayıt yapılır.

Metin dosyaları (Text Files) sıralı ve satırlar halinde metin bilgilerinden oluşmaktadır. Her satırda toplan 255 karakter bulunur. Her satırda sıfır veya daha fazla karakter içerir ve DOS işletim sisteminde, bir dizginin sonunda olduğu gibi "`\0`" işareti olmaz. Çünkü kaydedilen satırlar string değildir. Her satırın sonunu belirleyen bir *satır sonu* (end of line) işareti vardır. Kaynak kodu içinde satır sonlarına "`\n`" *yeni satır* (newline) konulur, fakat program dosyaya kayıt yaparken satır sonuna CR-LF (Cariage-Return – LineFeed) işareti koyar. Bu kombinasyon bahsettiğimiz satır sonu (end of line) işaretidir. Veri okunurken ise satır sonu işareti programa "`\n`" *yeni satır* karakteri olarak çevrilir.

İkili dosyalar (Binary Files), metin dosyalarında olduğu gibi dosya açılmadan önce fonksiyonda dosya tipi verilir. Bütün veriler byte byte ardışık olarak dosyaya yazılır. Kayıt içinde boşluk, satır sonu karakteri veya herhangi bir kaçış işaretinin özel bir anlamı yoktur.

Dosya İsimleri

Her disk dosyasını bir ismi vardır ve dosyalarla çalışırken bu isimler kullanılır. Dosya isimleri diğer veriler gibi string olarak saklanırlar. Dosya isimleri işletim sistemindeki kurallar dahilinde 256 karaktere kadar olabilir. Örneğin Windows işletim sistemlerinde (3.x hariç) `/ \ : * ? " < > |` işaretleri bulunamaz. Dosya isimlerinin genellikle son 3 hanesi nokta ile ayrılır ve dosya tipini gösterir.

Dosya isimleri ayrıca bulundukları yerin yolunu da (path) ve sürücü ismini (`a:, c:` vb) içerebilirler. Eğer dosya yolu ve sürücü verilmemiş ise, dosyanın güncel (current) klasörde bulunduğu varsayılır. Örneğin Windows işletim sisteminde dosya ismi `c:\veri\liste.txt` olarak verilebilir. Bu isim c sürücüsünde veri isimli klasördeki liste.txt isimli dosyayı göstermektedir. Ters kesme (backslash) işaretinin (`\`) string içinde kaçış karakteri olduğunu biliyoruz. Dolayısı ile dosya ismini string olarak verecek olursak iki tane artarda ters kesme işareti koymamız gerekecektir. Örneğin: `char *dosyaismi="c:\\veri\\liste.txt";` Bunun aksine eğer isim klavyeden girilecekse tek ters kesme işareti konulmalıdır. Unix sistemlerde klasörleri ayırmak için backslash yerine (`/`) işareti kullanılmaktadır.

`FILE` Gösterici

FILE yapısı (structure) `stdio.h` da tanımlanmış dosya kontrol yapısıdır ve dosya göstericisi (file pointer) olarak adlandırılır. Bir dosya göstericisi aşağıdaki gibi tanımlanır

```
FILE *dosya_gostericisi;
```

Burada *dosya_gostericisi* bir gösterici değişken gibidir.

Dosyaların Açılması

Verilerin dosyalara yazılması veya okunması için dosyaların öncelikle açılması gerekir. Bir dosyanın açılması için fopen fonksiyonu kullanılır ve kullanımı aşağıdaki gibidir.

```
dosya_gostericisi = fopen(dosya_adı, "Açma_modu");
```

`fopen` fonksiyonu, verilen dosya adı ve açma modu ile *dosya_gostericisi*'ni ilişkilendirir, dolayısı ile artık *dosya_gostericisi* verilen dosya ismini temsil edecektir. Eğer `fopen` fonksiyonu başarılı bir şekilde işletilmez ise geriye NULL değeri döner. Bu gibi sorunlar tanımsız bir dosya ismi, aranılan dosyanın sabit disk üzerinde bulunamaması, sabit disk'in format edilmemiş olması veya mevcut olmayan klasörün ya da sürücünün verilmiş olması durumlarında meydana gelebilir. Böylece

dosya_gostericisi denetlenerek dosyanın açılıp açılmadığı da kontrol edilebilir. *dosya_gostericisi* NULL değerini almış ise dosya açılmamıştır.

`fopen` fonksiyonunda bulunan *dosya_adı* açılması istenen dosyanın kayıt ünitesi üzerindeki adıdır. *açma_modu* dosyanın ne amaçla açılacağını belirtmektedir. Örneğin dosyadan veri mi okunacaktır ? dosyaya veri mi yazılacaktır?. Ya da hem okuma hem de yazma mı yapılacaktır? Dosya açmanın tüm durumlarını aşağıdaki tabloda görebilirsiniz.

Açma Modu	***Anlamı***
r	Mevcut metin dosyasının okuma için açılması. Dosya mevcut değilse geriye NULL gönderilir.
w	Metin dosyasının yazılma için açılması. Dosya mevcut değilse yazma için yaratılır, mevcut ise bir uyarı yapılmadan silinir ve yeniden yaratılır.
a	Mevcut metin dosyasına veri ekleme için açılması. Dosya mevcut değilse yaratılır, mevcut ise veriler dosyanın sonuna eklenir.
r+	Mevcut metin dosyasının hem okuma hem de yazma için açılması. Dosya mevcut değilse geriye NULL gönderilir.
w+	Mevcut metin dosyasının hem okuma hem de yazma için açılması. Dosya mevcut değilse yazma için yaratılır.
a+	Mevcut metin dosyasının okuma ve yazma için açılması. Dosya mevcut değilse yaratılır, mevcut ise veriler dosyanın sonuna eklenir.
rb	Mevcut binary dosyanın okuma için açılması. Dosya mevcut değilse geriye NULL gönderilir.
wb	Metin binary dosyanın yazılma için açılması. Dosya mevcut değilse yazma için yaratılır, mevcut ise bir uyarı yapılmadan silinir ve yeniden yaratılır.
ab	Mevcut binary dosyaya veri ekleme için açılması. Dosya mevcut değilse yaratılır, mevcut ise veriler dosyanın sonuna eklenir.
r+b	Mevcut binary dosyanın hem okuma hem de yazma için açılması. Dosya mevcut değilse geriye NULL gönderilir.
w+b	Mevcut binary dosyanın hem okuma hem de yazma için açılması. Dosya mevcut değilse yazma için yaratılır.
a+b	Mevcut binary dosyanın okuma ve yazma için açılması. Dosya mevcut değilse yaratılır, mevcut ise veriler dosyanın sonuna eklenir.

Aşağıdaki deyimler `test.txt` isimli bir metin dosyasını açmaktadır.

```
FILE *fptr;
fptr = fopen("test.txt", "r")
if (fptr == NULL)
   {
     printf("test.txt dosyasi acilamadi...\n");
     exit(1);
   }
```

yada

```
FILE *fptr;
if ((fptr = fopen("test.txt", "r")) == NULL)
   {
     printf("test.txt dosyasi acilamadi...\n");
     exit(1);
   }
```

Burada "`r`" dosyanın sadece okuma için açılacağını göstermektedir. Herhangi bir hata meydana geldiğinde `fopen` fonksiyonu geriye NULL değer gönderir ve program ekrana "`test.txt dosyasi acilamadi...`" mesajını ekrana yazar ve `exit(1)` fonksiyonunun çağrılması ile program sona erer. Dosyanın açılması durumunda program devam edecektir.

Dosyaların Kapatılması

Dosya okunduktan, yazıldıktan veya ekleme yapıldıktan sonra kapatılması gerekmektedir. Dosyanın kapatılması, dosya adı ile *`dosya_gostericisi`*'nin ilişkilendirmesinin sona erdirilmesidir ve `fclose()` fonksiyonu ile yapılır. `fclose()` fonksiyonunun yazım kuralı,

```
int fclose(dosya_gostericisi);
```

olarak yazılır. Yukarıda örnekte verilen `text.txt` dosyasının kapatılması için aşağıdaki deyim yazılır.

```
fclose(fptr);
```

`fptr` ile ilişkilendirilen `test.txt` dosyasının yukarıdaki deyimle bu ilişkisi bitirilir, diğer bir deyişle dosya kapatılır.

Program 12.1'de dosya açma ve kapa ile ilgili bir örnek program verilmiştir. 7'inci satırda `fp` değişkeni dosya göstericisi olarak tanımlanmıştır. 9'uncu satırda `fopen()` fonksiyonu ile `text.doc` isimli dosya yalnız okuma ("`r`") modunda açılmış ve `fp` dosya göstericisi ile ilişkilendirilmiştir. Aynı satırda dosya göstericisinin NULL değeri alıp almadığının kontrolü if ile yapılmıştır. Çıktıda görüldüğü gibi dosya göstericisi NULL değerini aldığından yani açılmak istenilen dosya bulunamadığından ilk if bloğu (10-14 satırları arası) işletilerek mesajlar ekrana yazılmıştır.

KOD 12.1. Dosyanın açılması ve kapatılması.

```
 1: /* p1201.c: Dosya ac/kapat */
 2: #include <stdio.h>
 3:
 4: main()
 5: {
 6:
 7:   FILE *fp;
 8:
 9:   if((fp=fopen("text.doc", "r"))==NULL)
10:     {
11:       printf("text.doc dosyasi acilamadi...\n");
12:       printf("program kapatiliyor...\n");
13:       exit(0);
14:     } else {
15:       printf("text.doc dosyasi acildi...\n");
16:       printf("fp nin degeri : 0x%p\n", fp);
```

```
17:       printf("Dosya kapatiliyor...\n");
18:       fclose(fp);
19:     }
20: }
```

ÇIKTI P1201.exe

```
text.doc dosyasi acilamadi...
program kapatiliyor...
```

Dosya Okuma ve Yazma

Bir dosyaya veri kaydetmek için çeşitli yöntemlerden biri seçilebilir. Veri kaydetmek için C dilinde birçok fonksiyon bulunmaktadır. Bu fonksiyonlardan bazıları aşağıdaki gibi sınıflandırılabilir.

- Dosyaya bir karakterin kaydedilmesi/okunması
- Dosyaya bir karakter dizisinin kaydedilmesi/okunması
- Dosyaya blok bilgi kaydedilmesi/okunması

Dosyaya Bir Karakterin Kaydedilmesi/Okunması

`fgetc()` ve `fputc()` fonksiyonları dosyalara veri kayıt edilmesi ve okunması için kullanılır. `fgetc()` fonksiyonu dosyadan her seferinde sadece bir karakter okuyabilir. Bu fonksiyon aşağıdaki şekilde kullanılmaktadır.

```
int fgetc(dosya_gosterici);
```

Dosyadan bir karakter okunduktan sonra `dosya_gosterici` bir artacağı için ikinci karakter olarak okunan karakterden sonraki karakter okunur. Aynı karakter ardı ardına iki kez veya dosyadaki mevcut karakterler sayısından fazla karakter okunamaz.

`fgetc()` fonksiyonu dosyaya her seferinde sadece bir karakter yazar. Bu fonksiyonun yazım kuralı aşağıdaki şekildedir:

```
int fputc(karakter, dosya_gosterici);
```

Dosyaya veri yazdırılırken aynı karakter istenildiği kadar art arda yazdırılabilir.

12.2 kodu hem dosyaya karakter yazdırabilir hem de dosyadan karakter okuyabilir.

KOD **12.2. Dosyaya bir karakterin kaydedilmesi/okunması**

```
 1: /* p1202.c: fgetc() ve fputc() */
 2: /* bir karakterin dosyaya yazdirilmasi ve okunmasi*/
 3: #include <stdio.h>
 4:
 5: main()
 6: {
 7:   char *chr = "Mehmet Zeki Coskun";
 8:   char ch;
 9:   FILE *fp;
10:
11:   /* Dosyaya yazdirma */
```

```
12:   if((fp=fopen("test.doc", "w"))==NULL)
13:   {
14:     printf("Dosya yazdirma icin acilamadi...");
15:     exit(0);
16:   } else {
17:     while(*chr) {
18:       fputc(*chr, fp);
19:       chr++;
20:     }
21:     fclose(fp);
22:     printf("Dosyaya yazdirildi...\n");
23:   }
24:
25:   /* Dosyadan okuma */
26:   if((fp=fopen("test.doc", "r"))==NULL)
27:   {
28:     printf("Dosya okuma icin acilamadi...");
29:     exit(0);
30:   } else {
31:     while(ch!=EOF) {
32:       ch=fgetc(fp);
33:       printf("%c", ch);
34:     }
35:     fclose(fp);
36:     printf("\nDosya okundu...\n");
37:   }
38: }
```

ÇIKTI P1202.exe

```
Dosyaya yazdirildi...
Mehmet Zeki Coskun
Dosya okundu...
```

P1201 programında, `chr` ve `ch` isimli bir karakter dizileri tanımlanmış ve `ch`r'ye ilk değer olarak bir isim atanmıştır. 10-22 satırları arasında dosyaya tek tek karakter yazdırma için kodlar yazılmıştır. 12'inci satırda `test.doc` isimli dosya yazdırma modu (`"w"`) ile açılmış, `fp` dosya göstericisi ile ilişkilendirilmiş ve dosyanın açılıp açılmadığının kontrolü yapılmıştır. 13-16 bloğu dosyanın açılamaması durumunda çalıştırılacaktır. Dosya açıldığından dolayı 16-23 arasındaki blok çalıştırılmıştır. 18'inci satırda `fp` dosya göstericisinin belirlediği dosyaya `chr` isimli karakter dizisinin birer elemanı `fputc()` fonksiyonu ile kaydedilmiş, daha sonraki satırda chr karakter dizisinin elemanı bir arttırılmış, karakter sonu kontrolü ise `while()` döngüsü ile 18'ıncı satırda yapılmıştır. Döngü bittikten sonra 21'inci satırda dosya kapatılmış ve bir sonraki satırda `printf()` fonksiyonu ile mesaj yazdırılmıştır.

25-37 satırlar arasında daha önce yazdırılan dosyanın içeriği birer karakter okutularak ekrana yazdırılmıştır. 26'inci satırda daha önce olduğu gibi dosya açılmış, dosya mevcut olduğundan 30-37 satırları çalıştırılmıştır. `ch` değişkenine `fgetc()` fonksiyonu ile okunan karakterler atanmış 32'inci satırda ise ekrana yazdırılmıştır. Ardı ardına okunan karakterler `printf()` fonksiyonu ile yazdırılırken satır atlama kontrol işareti (`\n`) kullanılmadığından ekranda karakterler yan yana yazdırılmıştır. Dosyanın sonu ise 31'inci satırda `ch!=EOF` ile kontrol ettirilmiştir.

Her dosyanın sonunda, dosyanın bittiğine gösteren bir karakter kayıt edilir. Bu karakter okunan `ch` karakteri ile karşılaştırılır ve dosyanın sonu belirlenebilir. EOF, End Of File (dosya sonu) anlamındadır.

Dosyaya Bir Karakter Dizisinin Kaydedilmesi/Okunması

Bir önceki konuda karakterlerin tek tek kaydedilmesi ve okunması için iki fonksiyon vardı. Bu fonksiyonları kullanarak dosyadan karakter dizisini okuyabiliyor ya da dosyaya karakter dizisini döngüler ve kontrol komutlarıyla yazdırabiliyorduk. Bunlara gerek kalmadan karakter dizilerini (dizgi/stirng) bir fonksiyonla okutabilir veya yazdırabiliriz.

Karakter dizisi yazdırmak için `fputs()` ve karakter dizisi okutmak için `fgets()` fonksiyonları kullanılır. `fputs()` fonksiyonu kullanılarak, bir karakter dizisinin yani bir dizginin (string) bir defada yazdırılması sağlanabilir. Bu fonksiyon aşağıdaki şekilde kullanılmaktadır.

```
int fputs(karakter_dizisi, dosya_gosterici);
```

Dosyadan bir karakter dizisinin bir seferde okunması isteniyorsa `gets()` fonksiyonu kullanılabilir.

```
int fgets(karakter_dizisi, dizi_uzunluğu, dosya_gosterici);
```

Kod 12.3 dosyaya bir karakter dizisinin kaydedilmesi ve okunması amacıyla yazılmıştır. Bir önceki uygulamada (kod 11.2) yaratılan dosya (`test.doc`) bu örnekte kullanılmıştır. Programda iki farklı dosya aynı anda açılmış ve işlemler yapılmıştır. İki dosya için farklı dosya göstericisi 8'inci satırda tanımlanmıştır. 10-14 satırlarında `test.doc` dosyası okuma için açılmış, 15-19 satırlarında `test1.doc` dosyası yazma için açılmıştır. Her iki dosya için sırasıyla `fp1`ve `fp2` dosya göstericileri tanımlanmıştır. Herhangi ikisinden birinin açılmaması durumunda ilk dosya için 13'üncü satırda, ikinci dosya için 18'inci satırda program sona erecektir.

21'inci satırda `fp1` dosya göstericisinin işaret ettiği dosyadan 80 karakterlik veri okunmuş ve `ch` karakter dizisine atanmıştır. 25'inci satırda ise `fp2` dosya göstericisinin işaret ettiği dosyaya `ch` karakter dizisi yazdırılmıştır. 22 ve 26'ıncı satırlarda mesajlar yazdırılmıştır. İlk dosya 23'üncü, ikinci dosya 27'inci satırda kapatılmıştır.

KOD **12.2. Dosyaya bir karakterin kaydedilmesi/okunması**

```
 1: /* p1203.c: fgets() ve fputs() */
 2: /* bir karakter dizisinin dosyaya yazdirilmasi ve okunmasi*/
 3: #include <stdio.h>
 4:
 5: main()
 6: {
 7:   char ch[80];
 8:   FILE *fp1, *fp2;
 9:
10:   if((fp1=fopen("test.doc", "r"))==NULL)
11:   {
12:     printf("test.doc okuma icin acilamadi...");
13:     exit(0);
14:   }
15:   if((fp2=fopen("test1.doc", "w"))==NULL)
16:   {
17:     printf("test1.doc yazdirma icin acilamadi...");
18:     exit(0);
```

```
19:   }
20:
21:   fgets(ch, 80, fp1);
22:   printf("okunan karakter dizi : %s\n", ch);
23:   fclose(fp1);
24:
25:   fputs(ch, fp2);
26:   printf("yazilan karakter dizi : %s\n", ch);
27:   fclose(fp2);
28: }
29:
```

ÇIKTI P1203.exe

```
okunan karakter dizi : Mehmet Zeki Coskun
yazilan karakter dizi : Mehmet Zeki Coskun
```

Dosyaya Blok Bilgi Kaydedilmesi/Okunması

İçeriğini herhangi bir metin editörü ile doğrudan göremediğimiz dosyalar binary (ikili) dosyalardır. Bu dosyaların okunması ve yazılması farklı fonksiyonlarla yapılmaktadır. Dosyaya yazmak için kullanılan fonksiyon `fwrite()` fonksiyonudur:

```
int fwrite(bellek, boyutu, sayısı, dosya_gosterici);
```

Burada,

- *bellek* : dosyaya yazdırılacak verinin geçici olarak bellekte saklandığı alan (buffer) (bir dizi veya bir yapı (struct) ile tanımlanmış bir veri seti),
- *boyutu* : yazdırılacak alanın uzunluğu (burada boyut sizeof() fonksiyonu ile verilebilir),
- *sayısı* : bu verinin tekrarlamam sayısı,
- *Dosya_gosterici*: dosya göstericisi

Aşağıda `struct` ile tanımlanmış bir veri setinin blok kayıt eden bir program yazılmıştır. Program kayıt yapısındaki üyeleri (numara, notu, isim) tek seferde dosyaya kayıt eder. Dosya modu ("`wb`") binary (ikili) kayıt modunda açılmıştır. Kayıt isimli yapının (struct) 20'inci satırda adresi verilmiştir, adres aynı zamanda bellekte saklandığı alandır.

KOD **12.4. Dosyaya blok kayıt edilmesi (ikili/binary kayıt) – `fwrite()`**

```
 1: /* p1204.c: fwrite()() */
 2: /* Binary verinin dosyaya yazdirilmasi, fwrite() */
 3: #include <stdio.h>
 4:
 5: struct ogrenci {
 6:   int numara;
 7:   float notu;
 8:   char isim[50];
 9: } kayit = {10822, 75.0, "Mehmet"};
10:
11: main()
```

```
12: {
13:   FILE *fp;
14:
15:   if((fp=fopen("binary.doc", "wb"))==NULL)
16:   {
17:     printf("Dosya acilamadi...\n");
18:     exit(0);
19:   } else {
20:     fwrite(&kayit, sizeof(kayit), 1, fp);
21:     fclose(fp);
22:   }
23: }
```

ÇIKTI `P1204.exe`

Binary dosyalardaki verileri okumak için `fread()` fonksiyonu kullanılmaktadır:

```
int fread(bellek, boyutu, sayısı, dosya_gosterici);
```

Burada,

- *bellek*: dosyadan okutulacak yazdırılacak verinin geçici olarak bellekte saklanacağı alan (buffer)
- *boyutu* : okutulacak alanın uzunluğu,
- *sayısı*: bu verinin tekrarlamam sayısı,
- *dosya_gosterici*: dosya göstericisi.

Kod 12.5'de bir önceki programda (kod 11.4) yazdırılan verileri okuyan bir program vardır. 15'inci satırda dosya sadece binary okuma modunda ("`rb`") açılmıştır. 20'inci satırda binary veri okuma fonksiyonu ile kayıt ismindeki yapı okunmuş ve ekrana yazdırılmıştır.

KOD **12.5. Dosyadan blok veri okunması (ikili/binary kayıt) – fread()**

```
 1: /* p1205.c: fwrite()() */
 2: /* Binary verinin dosyadan okutulmasi, fread() */
 3: #include <stdio.h>
 4:
 5: struct ogrenci {
 6:   int numara;
 7:   float notu;
 8:   char isim[50];
 9: } kayit;
10:
11: main()
12: {
13:   FILE *fp;
14:
15:   if((fp=fopen("binary.doc", "rb"))==NULL)
16:   {
17:     printf("Dosya acilamadi...\n");
18:     exit(0);
19:   } else {
20:     fread(&kayit, sizeof(kayit), 1, fp);
21:     fclose(fp);
22:
23:     printf("Ögrenci numarasi : %d\n", kayit.numara);
```

```
24:     printf("Ögrenci notu     : %f\n", kayit.notu);
25:     printf("Ögrenci ismi     : %s\n", kayit.isim);
26:   }
27: }
```

ÇIKTI P1205.exe

```
Ogrenci numarasi : 10822
Ogrenci notu     : 75.000000
Ogrenci ismi     : Mehmet
```

Bir dosyadan binary okuma yapılırken verilen alan kadar olan tüm bilgi okunur. Bir dizi okutulacaksa dizi eleman sayısı verilir, okunan tüm dizi okunma esnasında tek tek elemanlara atanır.

`feof()` Fonksiyonu

`feof()` fonksiyonu dosyadaki bilgilerin tümümün okunup okunmadığını gösterir. Bir dosya açıldığında bilgiler sırayla okunur. Dosya göstericisi her okumada yeni okuyacağı adresi alır. Herhangi bir kontrol yapılmaz ise disk üzerinde bu işlem kayıt ünitesi üzerindeki adreslerin sonuna kadar sürer. Dosyanın sonuna gelinip gelinmediği bu fonksiyon ile kontrol edilebilir. Fonksiyonun kullanımı aşağıdaki gibidir.

```
int feof(dosya_gosterici);
```

Aşağıdaki while döngüsü `fp` (dosya göstericisi) ile temsil edilen dosya sonuna kadar devam eder.

```
while (!feof(fp)){
  fread(bellek, boyutu, sayısı, fp);
  diger deyim(ler)
}
```

Alıştırmalar

1. "d:" sürücüsünde "dosyalar" klasöründeki "`metin.txt`" isimli dosyayı bir file göstericisi sadece okuma modu ile açan deyimi yazınız.

   FILE *p;

   p=fopen("c:\\dosyalar\\metin.txt", "r");

2. Aşağıdaki deyimlerin anlamlarını yazınız.

```
a) fopen("dosya.doc", "r")
b) fopen("dosya.doc", "w")
c) fopen("dosya.doc", "a")
```

3. HATA BUL : `fp` bir dosya göstericisi, kayit bir yapı değişkeni ise aşağıdaki deyimlerden hangileri yanlıştır?

```
a) fclose(fp);
b) fputc(fp, *chr);
c) ch=fgetc();
d) fgets(ch, 80, fp1);
e) fputs(fp2, ch);
f) fwrite(kayit, sizeof(kayit), 1, fp);
g) fread(kayit, sizeof(kayit), 1, fp);
```

4. Klavyeden girilen bir ismi karakter karakter bir dosyaya yazan, okuyan ve okuduğunu ekrana alt alta karakter karakter yazan bir program yazınız.

5. Klavyeden girilen bir ismi dizgi şeklinde bir dosyaya yazan, okuyan ve okuduğunu ekrana yazan bir program yazınız.

6. 5'inci soruda yaratılan dosya içeriğinin ilk 3 karakterini blok olarak okuyan bir program yazınız.

12. DERS CEVAPLAR

1. HATA BUL:

```
FILE *fp;
*fp=open(("d:\\dosyalar\\metin.txt", "r");
```

2.

```
d) fopen("dosya.doc", "r") -> dosya.doc dosyasını yalnız okuma modunda açar.
e) fopen("dosya.doc", "w") -> dosya.doc dosyasını yalnız yazma modunda açar.
f) fopen("dosya.doc", "a") -> dosya.doc dosyasını yalnız ekleme modunda açar.
```

3.

```
h) fclose(fp); -> DOĞRU
i) fputc(fp, *chr); -> fputc(*chr, fp);
j) ch=fgetc();
k) fgets(ch, 80, fp1); -> DOĞRU
l) fputs(fp2, ch); -> fputs(ch, fp2);
m) fwrite(kayit, sizeof(kayit), 1, fp); ->
          fwrite(&kayit, sizeof(kayit), 1, fp);
n) fread(kayit, sizeof(kayit), 1, fp); ->
          fread(&kayit, sizeof(kayit), 1, fp);
```

4.

```
/* Soru 12.4'ün cevabidir. */
#include <stdio.h>

int main()
{
  int i;
  FILE *fp;
  char *isim, ch;

  if((fp=fopen("dosya.doc", "w"))==NULL)
  {
    printf("dosya.doc yazma icin acilamadi...");
    exit(0);
  }

  printf("Bir isim giriniz : ");
  scanf("%s", isim);

  while(*isim) {
    fputc(*isim, fp);
    isim++;
  }

  fclose(fp);

  if((fp=fopen("dosya.doc", "r"))==NULL)
  {
    printf("dosya.doc okuma icin acilamadi...");
    exit(0);
  }

  while(ch!=EOF) {
    ch=fgetc(fp);
```

```
    printf("%c\n", ch);
  }

  fclose(fp);

  return 0;
}
```

5.

```
/* Soru 12.5'in cevabidir. */
#include <stdio.h>

int main()
{
  int i;
  FILE *fp;
  char *isim, ch;

  if((fp=fopen("dosya.doc", "w"))==NULL)
  {
    printf("dosya.doc yazma icin acilamadi...");
    exit(0);
  }

  printf("Bir isim giriniz : ");
  scanf("%s", isim);

  fputs(isim, fp);

  fclose(fp);

  if((fp=fopen("dosya.doc", "r"))==NULL)
  {
    printf("dosya.doc okuma icin acilamadi...");
    exit(0);
  }

  fgets(isim, 50, fp);
  printf("%s", isim);

  fclose(fp);

  return 0;
}
```

6.

```
/* Soru 12.6'nin cevabidir. */
#include <stdio.h>

int main()
{
  int i;
  FILE *fp;
  char *isim, ch;

  if((fp=fopen("dosya.doc", "rb"))==NULL)
  {
    printf("dosya.doc okuma icin acilamadi...");
    exit(0);
  }

  fread(isim, 3, 1, fp);
  printf("%s", isim);
```

```
  fclose(fp);

  return 0;
}
```

12. SAAT

1. HATA BUL:

```
FILE *fp;
*fp=open(("d:\\dosyalar\\metin.txt", "r");
```

2.

```
g) fopen("dosya.doc", "r") -> dosya.doc dosyasını yalnız okuma modunda açar.
h) fopen("dosya.doc", "w") -> dosya.doc dosyasını yalnız yazma modunda açar.
i) fopen("dosya.doc", "a") -> dosya.doc dosyasını yalnız ekleme modunda açar.
```

3.

```
o) fclose(fp); -> DOĞRU
p) fputc(fp, *chr); -> fputc(*chr, fp);
q) ch=fgetc();
r) fgets(ch, 80, fp1); -> DOĞRU
s) fputs(fp2, ch); -> fputs(ch, fp2);
t) fwrite(kayit, sizeof(kayit), 1, fp); ->
          fwrite(&kayit, sizeof(kayit), 1, fp);
u) fread(kayit, sizeof(kayit), 1, fp); ->
          fread(&kayit, sizeof(kayit), 1, fp);
```

4.

```
/* Soru 12.4'ün cevabidir. */
#include <stdio.h>

int main()
{
  int i;
  FILE *fp;
  char *isim, ch;

  if((fp=fopen("dosya.doc", "w"))==NULL)
  {
    printf("dosya.doc yazma icin acilamadi...");
    exit(0);
  }

  printf("Bir isim giriniz : ");
  scanf("%s", isim);

  while(*isim) {
    fputc(*isim, fp);
    isim++;
  }

  fclose(fp);

  if((fp=fopen("dosya.doc", "r"))==NULL)
  {
    printf("dosya.doc okuma icin acilamadi...");
    exit(0);
  }

  while(ch!=EOF) {
```

```
    ch=fgetc(fp);
    printf("%c\n", ch);
  }

  fclose(fp);

  return 0;
}
```

5.

```
/* Soru 12.5'in cevabidir. */
#include <stdio.h>

int main()
{
  int i;
  FILE *fp;
  char *isim, ch;

  if((fp=fopen("dosya.doc", "w"))==NULL)
  {
    printf("dosya.doc yazma icin acilamadi...");
    exit(0);
  }

  printf("Bir isim giriniz : ");
  scanf("%s", isim);

  fputs(isim, fp);

  fclose(fp);

  if((fp=fopen("dosya.doc", "r"))==NULL)
  {
    printf("dosya.doc okuma icin acilamadi...");
    exit(0);
  }

  fgets(isim, 50, fp);
  printf("%s", isim);

  fclose(fp);

  return 0;
}
```

6.

```
/* Soru 12.6'nin cevabidir. */
#include <stdio.h>

int main()
{
  int i;
  FILE *fp;
  char *isim, ch;

  if((fp=fopen("dosya.doc", "rb"))==NULL)
  {
    printf("dosya.doc okuma icin acilamadi...");
    exit(0);
  }

  fread(isim, 3, 1, fp);
  printf("%s", isim);
```

```
  fclose(fp);

  return 0;
}
```