BÖLÜM 12: DOSYA İŞLEMLERİ

Birçok program uygulamasında, programa girilen veya program tarafından üretilen bilgilerin yardımcı bellekte (disket veya hard disk) depo edilip daha sonra tekrar kullanılması gerekmektedir. Program tarafından üretilen bilgilerin depo edildiği dosyalara VERİ DOSYALARI adı verilir. Dosyalar, bilgisayar programcılığında en önemli kavramların başında gelir. Program ile veri dosyası arasındaki bilgi alışverişi, programdan veri dosyasına bilgi yazılması veya veri dosyasından programa bilgi okunması şeklinde olabilir.

C programlama dilinde veri dosyaları içerdiği veriye göre TEXT (metin) dosyalar ve BINARY (ikili) dosyalar olmak üzere iki farklı türde olabilir. TEXT dosyalar, ASCII dosyalar gibi, kullanıldığı karakter kümesinin adıyla da anılabilir. Çalıştırılabilir (.EXE) dosyalar Binary dosya tipindedir. Veri dosyaları (hem Text hem de Binary dosyalar) bilgi yazma, bilgi okuma ve bilgi ekleme modunda açılabilir. Aşağıdaki listede veri dosyalarının çeşitli açılma modları görülmektedir.

Mod	*Anlamı*
`"w"`	Text dosyasını yazma modunda oluşturur. Aynı isimli dosya varsa siler ve boş bir dosya açar. Konum göstergeci dosya başındadır.
`"r"`	Text dosyayı okuma modunda açar. Göstergeç dosya başındadır.
`"a"`	Text dosyayı ekleme modunda açar. Dosya konum göstergeci dosya sonundadır. Dosya yoksa oluşturur.
`"w+"`	Text dosyayı yazma modunda açar, okuma da yapılabilir.
`"r+"`	Text dosyayı okuma modunda açar, yazma da yapılabilir.
`"a+"`	Text dosyayı ekleme modunda açar, okuma da yapılabilir.
`"wb"`	Binary dosyayı yazma modunda açar.
`"rb"`	Binary dosyayı okuma modunda açar.
`"ab"`	Binary dosyayı ekleme modunda açar.
`"w+b"`	Binary dosyayı yazma modunda açar, okuma da yapılabilir.
`"r+b"`	Binary dosyayı okuma modunda açar, yazma da yapılabilir.
`"a+b"`	Binary dosyayı ekleme modunda açar, okuma da yapılabilir.

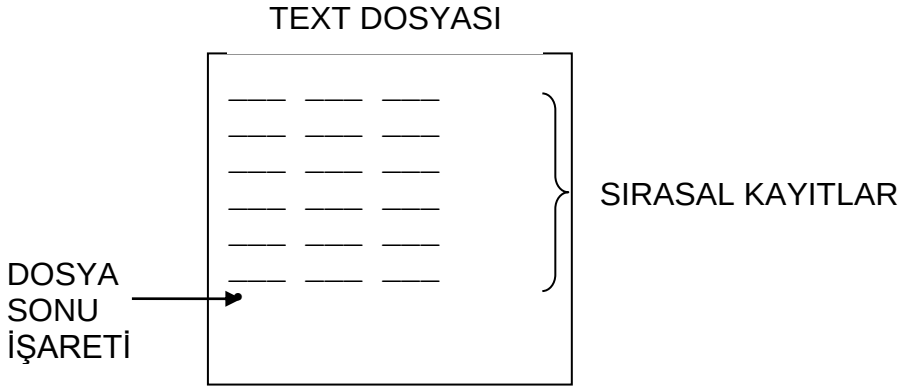
Dosya işlemleriyle ilgili fonksiyonlar `stdio.h` kütüphane dosyasında tanımlanmış olup bunlardan bazıları aşağıda verilmiştir;

Fonksiyon	*Görevi*
`fopen()`	Dosya oluşturur, açar
`fclose()`	Dosyayı kapatır
`putc()`	Dosyaya karakter yazar
`getc()`	Dosyadan karakter okur
`feof()`	Dosya sonuna gelindiğini sorgular
`fprintf()`	Dosyaya formatlı veri yazar
`fscanf()`	Dosyadan formatlı veri okur
`fputs()`	Dosyaya string yazar
`fgets()`	Dosyadan string okur
`fwrite()`	Dosyaya blok halinde veri yazar
`fread()`	Dosyadan blok halinde veri okur
`fseek()`	Verilere erişim için konum belirler
`ftell()`	Byte olarak dosya konum göstergecinin yerini döndürür.
`rename()`	Dosyayı yeniden isimlendirir
`remove()`	Dosyayı sabit bellekten siler
`rewind()`	Dosya konum göstergecini dosya başına döndürür.

I. TEXT DOSYALAR

Text dosyalar, ilk defa oluşturma ve bilgi yazma "w", bilgi okuma "r" ve bilgi ekleme "a" modlarında açılabilir. Bunlar temel modlardır. Bunlara ek olarak Text dosyalar "w+", "r+" ve "a+" modlarında da açılabilir. Bu modlar ve aralarındaki farklar sırasıyla incelenecektir.

Text dosyalar, BASIC programlama dilindeki sırasal (sequential) dosyalara benzetilebilir. Text dosyalara bilgiler sırasal olarak kaydedilir ve kaydedilen bilgilerin en sonuna dosya sonu işareti (EOF) konur. Dosya sonu işareti (EOF) ekranda görüntülenmeyen özel bir işarettir.



"w" (Wrıte-Yazma) Modu

Bir text dosyasını diskette veya hard diskte ilk defa oluştururken "w" modunu kullanarak açmak gerekir. Daha önce oluşturulmuş ve içine bilgi yazılmış bir dosya tekrar "w" modunda açılırsa, yazıcı kafa dosyanın en başına konumlanacağı için, yeni girilen bilgiler önceki bilgilerin üzerine yazılır ve böylece önceki bilgiler silinmiş olur.

Text dosyalarla işlem yapacak olan programlarda öncelikle, değişkenlerin tanımlandığı bölümde bir dosya değişkeni tanımlanır. Bu tanımda pointer tipi bir değişken olan FILE tipi tanımlanır. Bu değişken program içinde dosyayı temsil eder. FILE kelimesinin büyük harfle yazılması zorunludur. Daha sonra fopen fonksiyonu ile dosya açılır ve geri dönüş değeri olarak bu dosya değişkenine FILE tipinde bir pointer döner. fopen fonksiyonunun iki parametresi bulunur ve bu parametreler parantez içinde verilir. Her iki parametre de çift tırnaklar içersindedir ve birbirinden virgülle ayrılmıştır. Bu parametrelerden birincisi açılacak dosyanın konumu ve adıdır. Dosya adı, DOS işletim sisteminde geçerli olacak bir ad olmalı, en çok sekiz karakterden meydana gelmelidir. Dosya adında en fazla üç karakterlik bir uzantı bulunabilir. Geleneksel olarak text veri dosyalarında TXT uzantısı Binary veri dosyalarında ise DAT uzantısı kullanılmaktadır. TXT İngilizcede metin anlamına gelen TEXT kelimesinin kısaltılmışı, DAT ise veri anlamına gelen DATA kelimesinin ilk üç harfidir. Böylece dosyanın ne tür bir veri dosyası olduğu vurgulanmaktadır. Ancak tekrar edelim ki, başka bir üç harfli ifade de uzantı olarak verilebilir ya da hiç uzantı yazılmayabilir. fopen fonksiyonunun ikinci parametresi dosyanın açılma modudur.

Aşağıdaki program, "rakam.txt" adlı bir dosyayı ilk defa oluşturmakta ve klavyeden girilen 5 adet sayıyı sırasal olarak bu dosyaya kaydetmektedir.

```
#include <stdio.h>
void main()
{    FILE *dd;
     int i, no;
     dd=fopen("d:\\rakam.txt","w");
                         /* dosya, yazma modunda açılıyor */
     if (dd == NULL)
          printf("Surucu bulunamadi.\n");
     else
     {    for(i=1; i<=5; i++)
          {    printf("%d. sayiyi giriniz:",i);
               scanf("%d",&no);
               fprintf(dd,"%d\n", no);
          }
          fclose(dd);
     }
}
```

Bu program çalıştırıldığında bilgisayarımızın D sürücüsünde `"rakam.txt"` adındaki bir veri dosyasını ilk defa oluşturur ve klavyeden girilen 5 adet sayıyı bu dosyaya sırasal olarak kaydeder. Değişkenlerin tanımlandığı bölümde `*dd` adlı değişken `FILE` (dosya) tipinde tanımlanmıştır. Program içinde `"rakam.txt"` adlı dosya `dd` değişkeni ile temsil edilir. Klavyeden girilen sayılar `fprintf` fonksiyonu ile dosyaya kaydedilir. `fprintf` fonksiyonu bilgileri `dd` ile gösterilen dosyaya formatlı olarak kaydeder. `fprintf` fonksiyonunun format string parametresi aynı `printf` fonksiyonunda olduğu gibidir.

Bu programdan sırasıyla 3, 5, 7, 4, 9 sayılarının girildiğini kabul edelim. Dosyayı da D sürücüsünde oluşturmuşsak işletim sistemindeki komut satırından `"rakam.txt"` dosyasının içeriğini `TYPE` komutu ile görüntüleyebiliriz. TEXT dosyaların içeriklerini herhangi bir TEXT editörü (Notepad gibi) kullanarak ta görebiliriz. Bu notlarda genel olarak aktif sürücü olarak D sürücüsünü kullanıyoruz. Sürücü belirtilmemesi durumunda ise dosya kaynak kodunuzun bulunduğu konumda açılır (`w`) veya okunmak için aranır (`r`).

```
D:\> TYPE RAKAM.TXT (ENTER tuşuna basıyoruz)
35
56
-78
49
91
```

Görüldüğü gibi, klavyeden girilen sayılar, `"rakam.txt"` isimli dosyaya sırasal olarak kaydedilmiştir. Programın sonunda da `fclose(dd)` fonksiyonu ile dosya kapatılmıştır.

`"r"` (Read-Okuma) Modu

Aşağıdaki program ise az önce oluşturmuş olduğumuz `"rakam.txt"` adlı dosyayı okuma modunda açarak, içindeki bilgileri programdaki değişkene okur ve görüntüler.

```
#include <stdio.h>
void main()
{    FILE *dd;
     int no;        /* dosya, okuma modunda açılıyor */
     if ((dd=fopen("d:\\rakam.txt","r")) == NULL)
          printf("Surucu veya dosya bulunamadi.\n");
     else
     {    while (fscanf(dd,"%d",&no)!=EOF)
               printf ("%d\n",no) ;
          fclose(dd);
     }
}
```

Program çıktısı:

```
35
56
-78
49
91
```

Bu program, `"rakam.txt"` adlı veri dosyasını `"r"` (read-okuma) modunda açmakta ve `fscanf` fonksiyonu ile içindeki bilgileri sırasıyla okumaktadır, `fscanf` fonksiyonu da `scanf` gibi formatlı bir giriş fonksiyonudur, `fscanf` komutunun ilk parametresi bilginin okunacağı dosyayı temsil eden dosya değişkenidir. Okuma işlemi dosya sonu işaretine kadar yapılır. Yukarıdaki programda bulunan `while` döngüsü, dosya sonuna `(EOF)` gelindiğinde sona erer. Bu programda dosya açıma işlemiyle `if` koşulunun beraber kullanıldığına dikkat edin.

`"a"` (Append-Ekleme) Modu

Bu mod, dosyayı bilgi ekleme amacıyla açmakta kullanılır. Diskette (veya hard diskte) mevcut olan bir dosya eğer tekrar `"w"` moduyla açılmış olsaydı, yazıcı kafa dosyanın başına konumlanacaktı. Yeni bilgiler önceki bilgilerin üzerine yazılacağından önceki bilgiler kaybolacaktı. Dosya `"a"` modunda açıldığında ise, yazıcı kafa dosya sonu `(EOF)` işaretinin bulunduğu yere konumlanır ve böylece yeni bilgiler önceki bilgilerin sonundan itibaren yazılmaya başlanır. Aşağıdaki programda, daha önce diskette oluşturulmuş olan `"rakam.txt"` adlı dosya, yeni bilgi eklemek amacıyla `"a"` modunda açılmıştır.

```
#include <stdio.h>
void main()
{    FILE *dd;
     int i, no;     /* dosya, ekleme modunda açılıyor */
     if ((dd=fopen("d:\\rakam.txt","a")) == NULL)
          printf("Surucu bulunamadi.\n");
     else
     {    for(i=1; i<=3; i++)
          {    printf("%d. sayıyı giriniz:",i);
               scanf("%d",&no);
               fprintf(dd,"%d\n", no);
          }
          fclose(dd);
     }
}
```

Bu program çalıştırıldığında, klavyeden girilen 3 sayı, daha önceki program tarafından "rakam.txt" adlı dosyaya kaydedilmiş olan 5 sayının altına sırayla eklenmektedir.

Aşağıdaki programda ise "adlar.txt" adlı bir Text dosya oluşturulmakta ve klavyeden girilen isim ve numaralar bu dosyaya sırasal olarak kaydedilmektedir. Programda while(1) yapısı kullanılarak sonsuz döngü oluşturulmuştur. Programcı devam etmek isteyip istemediği sorusuna karşılık klavyeden 'h' haricinde bir harf girildiği sürece döngü çalışır ve klavyeden girilen isim ve numaralar "adlar" dosyasına kaydedilir.

```
#include <stdio.h>
#include <conio.h>
void main()
{    FILE *dd;
     char ad[20], devam;
     int no;
     dd=fopen("d:\\adlar.txt","w");
     if (dd == NULL)
          printf("Surucu bulunamadi.\n");
     else
     {
          while(1)
          {    printf("Adi :");
               scanf("%s", ad);
               printf("Numarasi :");
               scanf("%d", &no);
               fprintf(dd,"%s\t%5d\n", ad, no);
               printf("\nDevam etmek istiyor musunuz:");
               devam = getche();
               if(devam == 'h') break;
               printf("\n");
          }
          fclose(dd);
          printf("\n");
     }
}
```

Programı çalıştıralım ve 3 kişiye ait isim ve numaraları klavyeden girelim. Daha sonra devam etmek istiyor musunuz sorusuna karşılık 'h' harfini girelim. Dosya kapanacak ve program sona erecektir. DOS ortamına dönüp TYPE komutu ile "adlar.txt" dosyasının konumunu bulup görüntülediğimizde, girilen isim ve numaraların bu dosyaya sırasal olarak kaydedilmiş olduğunu göreceğiz.

```
D:\> TYPE ADLAR.TXT (ENTER)
Kadir        55
Suat         32
Celal         9
```

Aşağıdaki program ise "adlar.txt" adlı veri dosyasını "r" (read-okuma) modunda açar ve içindeki bilgileri okuyarak görüntüler.

```
#include <stdio.h>
void main()
{    FILE *dd;
     char ad[20];
     int no;
     dd=fopen("d:\\adlar.txt","r");
     if (dd == NULL)
          printf("Surucu veya dosya bulunamadi.\n");
     else
     {    printf("Adi\tNumarasi\n");
          printf("================\n");
          while(fscanf(dd,"%s%d", ad, &no) !=EOF)
               printf("%s\t%5d\n", ad, no);
          fclose(dd);
     }
}
```

Program çıktısı:

```
Adi       Numarasi
==================
Kadir        55
Suat         32
Celal         9
```

Yukarıdaki program çalıştırıldığında, `"adlar.txt"` adındaki veri dosyasını okuma `"r"` modunda açar ve `while` döngüsü ile baştan itibaren dosya sonu işaretine `(EOF)` kadar bütün bilgileri okur ve görüntüler. Dosya sonu işaretine `(EOF)` gelindiğinde döngü sona erer. Dosyadaki bilgiler `fscanf` fonksiyonu ile okunmaktadır.

Şimdi de, `"adlar.txt"` dosyasına yeni bilgiler ekleyen programı inceleyelim. Bu program `"a"` (append-ekleme) modunda açılır ve kullanıcı `"Devam etmek istiyor musunuz"` sorusuna karşılık klavyeden `'h'` dışında herhangi bir harf girdiği sürece klavyeden girilen isim ve numaralar `"adlar.txt"` dosyasındaki mevcut bilgilerin sonuna eklenir.

```
#include <stdio.h>
#include <conio.h>
void main()
{    FILE *dd;
     char ad[20], devam;
     int no;
     dd=fopen("d:\\adlar.txt","a");
     if (dd == NULL)
          printf("Surucu bulunamadi.\n");
     else
     {
          while(1)
          {    printf("Adi :"); scanf("%s", ad);
               printf("Numarasi :"); scanf("%d", &no);
               fprintf(dd,"%s\t%5d\n", ad, no);
               printf("\nDevam etmek istiyor musunuz:");
```

```
                devam = getche();
                if(devam == 'h') break;
                printf("\n");
           }
           fclose(dd);
           printf("\n");
     }
}
```

Programı çalıştıralım ve iki kişiye ait isim ve numaraları girdikten sonra programı sona erdirelim. DOS ortamında `TYPE` komutu ile "`adlar.txt`" dosyasının içeriğini görüntülediğimizde yeni girilen bilgilerin önceki bilgilere eklendiği görülecektir.

```
D:\> TYPE ADLAR.TXT (ENTER)
Kadir        55
Suat         32
Celal         9
Hakan        54
Ahmet        43
```

`fscanf` fonksiyonu, `scanf` fonksiyonunda olduğu gibi boşluk içeren karakter dizilerin (string) okutulmasında kullanılmamalıdır. Bunun yerine `gets` fonksiyonunda olduğu gibi işlev gören `fgets` fonksiyonu kullanılabilir. Aynı şekilde `fprintf` yerine de `fputs` fonksiyonu kullanılabilir. Aşağıdaki örnekte 20 karakter uzunluğuna kadar olan ve boşlukta içeren 5 adet isim okutulmakta ve "`isimler.txt`" dosyasına yazdırılmaktadır.

```
#include <stdio.h>
void main()
{    FILE *dd;
     int i;
     char ad[20];
     if ((dd=fopen("d:\\isimler.txt","w")) == NULL)
          /* dosya, yazma modunda açılıyor */
          printf("Sürücü bulunamadı\n");
     else
     {    for(i=1; i<=5; i++)
          {    printf("%d. adi giriniz:",i);
               fgets(ad, 20, stdin);
               fputs(ad, dd);
          }
          fclose(dd);
     }
}
```

`fgets` fonksiyonu, `gets` fonksiyonunda olduğu gibi string sonu karakterini de kullanır. Bu yüzden aşağıdaki program `ad` karakter dizi değişkenine 19 karakter okur ve 20. karakter olarak string sonu karakterini belleğe yerleştirilir. `fgets` fonksiyonu belirtilen sayıda karakter okuduktan sonra değil dosya sonu ile veya yeni satır karakteri ile karşılaştığında da okuma işlemini sona erdirir. Aşağıdaki program az önce yazılan isimleri ekranda görüntülemektedir.

```
#include <stdio.h>
void main()
{    FILE *dd;
     char ad[20];
     dd=fopen("d:\\isimler.txt","r");
     if (dd == NULL)
          printf("Sürücü veya dosya bulunamadı\n");
               /* dosya, okuma modunda açılıyor */
     else
     {    while (!feof(dd))
          {    fgets(ad, 20, dd);
               printf("%s", ad);
          }
          fclose(dd);
     }
}
```

Şimdi de basit bir muhasebe programını inceleyelim. Bu programda, buraya kadar görmüş olduğumuz hemen hemen bütün konular bulunmaktadır.

```
#include <stdio.h>  /* Muhasebe programı */
#include <conio.h>
#include <string.h>
#include <stdlib.h>
char menu();
void ekle();
void ara();

void main()
{    char secenek;
     do
     {    secenek = menu() ;
          switch(secenek)
          {    case 'e':
                    ekle(); break;
               case 'a':
                    ara(); break;
          }
     }
     while (secenek != 'c');
}

/* Menüyü görüntüleyen fonksiyon */
char menu()
{    char ch;
     do
     {    printf("(E)kle, (A)ra, veya (C)ikis :");
          ch = tolower(getche());
          printf("\n");
     }
     while(ch != 'e' && ch != 'a' && ch !='c');
     return ch;
}
```

```
/* isim ve hesap özetinin girildiği fonksiyon */
void ekle()
{    FILE *dd;
     char isim[80];
     float hesap;
     dd = fopen("d:\\hesap.txt","a");
     if (dd == NULL)
          printf("Surucu bulunamadi.\n");
     else
     {    printf("Adı giriniz :");
          scanf("%s",isim);
          printf("Hesap miktarını giriniz (YTL):");
          scanf("%f",&hesap);
          fprintf(dd,"%s\t%8.2f\n",isim, hesap);
          fclose(dd);
     }
}

/* Girilen isme ait hesap özetini veren fonksiyon */
void ara()
{    FILE *dd;
     char isim[80], isim2[80];
     float hesap;
     dd = fopen("d:\\hesap.txt","r");
     if (dd == NULL)
          printf("Dosya bulunamadi.\n");
     else
     {    printf("ismi giriniz : ");
          gets(isim2);
          while(fscanf(dd,"%s",isim) != EOF)
          {    fscanf (dd,"%f", &hesap);
               if(!strcmp(isim, isim2))
               {    printf("%s:\t%8.2f YTL.\n",isim, hesap);
                    break;
               }
          }
          fclose(dd);
     }

}
```

Aşağıdaki programda ise tek bir karakterin dosyaya yazılması veya dosyadan okutulması durumlarında kullanılan `getc()` ve `putc()` fonksiyonlarının kullanımları gösterilmektedir. Bu programda dosya konum göstergeci her karakter okunduğunda bir ilerler. Bu ilerleme dosya sonuna gelene kadar devam eder.

```
#include <stdio.h>
void main()
{    FILE *dd;
     char cumle[BUFSIZ], ch;
     if ((dd=fopen("d:\\cumle.txt","w")) == NULL)
          printf("Sürücü bulunamadı\n");
```

```
     else
     {    printf("Bir cumle giriniz:");
          gets(cumle);
          for (int i=0; cumle[i]; i++)
               putc(cumle[i], dd);
          fclose(dd);
     }

     if ((dd=fopen("d:\\cumle.txt","r")) == NULL)
          printf("Surucu veya dosya bulunamadi\n");
     else
     {    while((ch = getc(dd)) != EOF)
               printf("%c", ch);
          fclose(dd);
     }
}
```

Yukarıdaki `while` döngüsü aşağıdaki şekilde de yazılabilir. `feof()` fonksiyonu `EOF` sabitini (-1) değil pozitif bir değer döndürür.

```
while (!feof(dd))
{    ch = getc(dd);
     printf("%c", ch);
}
```

Oluşturmuş olduğunuz herhangi bir dosyayı (TEXT veya BINARY) gerektiğinde yeniden isimlendirmek için `rename()` fonksiyonunu silmek için ise `remove()` fonksiyonunu kullanabilirsiniz.

`rename()` fonksiyonunu dosyayı yeniden isimlendirmek amacıyla kullanılır.
Formatı:
```
rename(eski_dosya_ismi, yeni_dosya_ismi);
```
Örnek:
```
rename("d:\\cumle.txt", "d:\\cumle2.txt");
```

`remove()` fonksiyonunu ise dosyayı sabit bellekten silmek amacıyla kullanılır.
Formatı:
```
remove(dosya_adı);
```
Örnek:
```
remove("d:\\cumle.txt");
```

`rewind()` fonksiyonunu dosya konum göstergecini dosya başına döndürmek amacıyla kullanılır.
Formatı:
```
rewind(dosya_değişkeni);
```

Örnek:
Aşağıdaki uygulamada "`w+`" modunda açılan `rakam.txt` dosyasına önce 5 adet sayı yazdırılıyor daha sonra dosya kapatılmadan dosya konum göstergeci dosya başına döndürülmektedir. Bu sayede yazılan 5 adet sayı dosyadan okunarak ekranda görüntülenmektedir.

```
#include <stdio.h>
void main()
{    FILE *dd;
     int i, no;
     dd=fopen("d:\\rakam.txt","w+");
                         /* dosya, yazma modunda açılıyor */
     if (dd == NULL)
          printf("Surucu bulunamadi.\n");
     else
          for(i=1; i<=5; i++)
          {    printf("%d. sayiyi giriniz:",i);
               scanf("%d",&no);
               fprintf(dd,"%d\n", no);
          }

     rewind(dd);
     while (fscanf(dd,"%d",&no)!=EOF)
          printf ("%d\n",no) ;
     fclose(dd);
}
```

II. BINARY DOSYALAR

Binary dosyalara BYTE düzeyinde kayıt yapılır ve okunur. Bu dosyalarda kayıt yazma işleminde `fwrite()`, kayıt okuma işleminde ise `fread()` fonksiyonları kullanılır.

`fwrite()` fonksiyonunun formatı:
`fwrite(değişken, byte sayısı, bilgi sayısı, dosya değişkeni)`

Bu fonksiyonun parametreleri:

`değişken:` Kaydedilecek bilgiyi tutan değişkenin adresi.
`byte sayısı:` Kaydedilecek bilginin byte olarak uzunluğu (bu uzunluk daha önce görmüş olduğumuz `sizeof` fonksiyonu ile bulunur).
`bilgi sayısı:` Verinin peş peşe kaç defa yazılacağı. Genellikle bu sayı 1'dir.
`dosya değişkeni:` Diskteki dosyayı programda temsil eden değişken.

`fread()` fonksiyonunun formatı:
`fread(değişken, byte sayısı, bilgi sayısı, dosya değişkeni)`

Bu fonksiyonun parametreleri:

`değişken:` Okunacak olan bilgiyi tutacak olan değişkenin adresi.
`byte sayısı:` Okunacak bilginin byte olarak uzunluğu (sizeof komutu kullanılarak bulunur).
`bilgi sayısı:` Her seferde okunacak bilgi sayısı. Genellikle bu sayı 1 olur.
`dosya değişkeni:` Diskteki dosyayı programda temsil eden değişken.

Aşağıdaki program, 100 elemanlı bir dizi değişkeni sadece bir tek `fwrite()` fonksiyonu ile `"ornek.dat"` adlı bir veri dosyasına kaydetmektedir. Bu dizi değişkenin elemanlarında 0'dan 99'a kadar olan sayılar vardır, `"ornek.dat"` dosyası `"wb"` yani binary yazma modunda açılmıştır.

```
#include <stdio.h>
void main ()
{    FILE *fp;
     int sayilar[100];
     int i;
     fp=fopen("d:\\ornek.dat","wb");
     for (i=0; i<100; i++)
          sayilar[i] = i;
     fwrite(sayilar, sizeof(sayilar), 1, fp);
     fclose(fp);
}
```

Bu program çalıştırıldığında, `sayilar[100]` dizi değişkeninin elemanlarına 0 ile 99 arasındaki sayılar aktarılır. Daha sonra da `fwrite()` fonksiyonuyla `sayilar[100]` dizi değişkeni `"ornek.dat"` dosyasına kaydedilir.

Şimdi inceleyeceğimiz program ise `"ornek.dat"` dosyasındaki bilgileri okur. Bütün bilgiler (yani 0 ile 99 arasındaki sayılar) bir tek fread komutu tarafından okunmuştur. Bu program okuma işlemi yapacağı için `"ornek.dat"` dosyası `"rb"` yani bilgi okuma modunda açılmıştır.

```
#include <stdio.h>
void main ()
{    FILE *fp;
     int sayilar[100];
     int i;
     fp=fopen("d:\\ornek.dat","rb");
     fread(sayilar, sizeof(sayilar), 1, fp);
     for (i=0; i<100; i++)
          printf("%2d\n", sayilar[i]);
     fclose(fp);
}
```

Program çalıştırıldığında `"ornek.dat"` dosyasındaki bilgiler okunup ekranda görüntülenecektir.

III. RASTGELE (RANDOM) ERİŞİMLİ DOSYALAR

Rastgele erişimli dosyalarda `fseek` fonksiyonu, yazıcı/okuyucu kafa olarak da kullanılan dosya konum göstergecini belirli bir kayıt alanına konumlandırır. Böylece her kaydın bir kayıt numarası olmakta, bu numara kullanılarak kayda doğrudan doğruya ulaşma imkanı bulunmaktadır. Bu durum özellikle sıralı olarak oluşturulmuş dosyalarda arama yapan programlarda büyük kolaylık sağlar. Aşağıda `fseek()` fonksiyonunun formatı verilmiştir.

```
fseek(dosya değişkeni, kayıt pozisyonu, referans noktası)
```

`dosya değişkeni`: Diskteki dosyayı, programda temsil eden değişkenin adı.
`kayıt pozisyonu`: Kayıtların başlangıcından, okuyucu veya yazıcı kafanın bulunduğu noktadan ya da kayıtların sonundan itibaren, byte olarak, kayıt yapılacak veya kayıt okunacak alan.

`referans noktsı`: Kayıt pozisyonunun nereden itibaren hesaplanacağını gösterir. `0`, `1` veya `2` değerlerini alabilir.
`0`: Kayıt pozisyonu, dosyanın başından itibaren byte cinsinden hesaplanır. `0` yerine `SEEK_SET`'te yazılabilir.
`1`: Kayıt pozisyonunun, yazıcı veya okuyucu kafanın bulunduğu noktadan itibaren hesaplanacağını gösterir. `1` yerine `SEEK_CUR`'da yazılabilir.
`2`: Kayıt pozisyonu dosyanın sonundan itibaren byte cinsinden hesaplanır. `2` yerine `SEEK_END`'de yazılabilir.

Binary ve text dosyalarda kayıtlara numara da verilebilir. Fiziki (yani diskteki) kayıt alanı numarası sıfırdan başlar. Ancak her kayıt alanına kayıt yapılması zorunluluğu yoktur. Kayıtlar rasgele alanlara yazılıp okunabilir. Bu durumda içinde kayıt bulunan alanlarla, kayıt bulunmayan alanların birbirinden ayırt edilmesi gerekir. Uygulamada içinde kayıt bulunan alanlara ilk kayıt elemanı olarak özel bir işaret (örneğin *) konulabilir ve böylece bir alanda kayıt bulunup bulunmadığı kontrol edilebilir.

Aşağıdaki program `"rehber.txt"` adlı bir text dosya oluşturarak girilen kayıt numaralarına göre belirli kayıt alanlarına kişilerin ad ve soyadlarıyla telefon numaralarını yazmaktadır. Bu text dosyası `"r+"` (okuma/yazma) modunda açılmıştır. Buna göre daha önce diskette bulunan bir dosyada kayıt mevcutsa bu kayıtlar okunabilecek, eğer istenirse yeni kayıt yapılabilecektir. Bu kayıtlar dosyada mevcut bulunan kayıtlara eklenir. Dosyadaki kayıtlarda silme işlemi yapılamaz.

```
#include <stdio.h>
#include <conio.h>
#include <errno.h>

struct kaytip
{    char iz;
     char ad[10];
     char soyad[15];
     char tel[10];
};

struct kaytip kayit;
FILE *dd;
int kayuz, kayno;
long kaypos;
char cevap;

void main()
{    dd=fopen("d:\\rehber.txt","r+");
     if(errno == ENOENT)
          dd = fopen("d:\\rehber.txt", "w+");
     kayuz = sizeof(kayit);
     do
     {    printf ("\nKayit no :");
          scanf("%d", &kayno);
          kayit.iz=' ';
```

```
          kaypos = (kayno-1) * kayuz;
          fseek(dd, kaypos, SEEK_SET);
          fread(&kayit, sizeof(kayit), 1, dd);
          if(kayit.iz != '*')
          {    printf("Adi :");
               scanf("%s", kayit.ad);
               printf("Soyadi :");
               scanf("%s", kayit.soyad);
               printf("Telefonu :");
               scanf("%s", kayit.tel);
               kayit.iz = '*';
               fseek(dd, kaypos, SEEK_SET);
               fwrite(&kayit, sizeof(kayit), 1, dd);
          }
          else
               printf("Bu bölgede bir kayıt var..\n");
          printf("\nDevam etmek istiyormusunuz (E/H) :");
          cevap = getche () ;
     }
     while(cevap != 'h' && cevap != 'H');
     fclose(dd);
}
```

Bu programda, önce kayıt yapılmak istenen kayıt alanında mevcut bir kayıt olup olmadığı kontrol edilmektedir. Eğer `kayit.iz` değişkeninin okuduğu alana "`*`" karakteri varsa o alanda daha önceden yapılmış bir kayıt var demektir, bu nedenle o alana yeni kayıt yapılmaz, `kayuz` değişkeni, bir kaydın byte olarak uzunluğunu vermektedir. Aşağıdaki program ise, "`rehber.txt`" dosyasındaki kayıtları okur.

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

struct kaytip
{    char iz;
     char ad[10];
     char soyad[15];
     char tel [10];
};

struct kaytip kayit;
FILE *dd;
int kayuz,kayno;
long kaypos;
char cevap;

void main()
{    dd = fopen("d:\\rehber.txt","r+");
     kayuz = sizeof(kayit);
     do
     {    printf("\n");
          printf ("\nKayit numarasi giriniz :");
          scanf ("%d", &kayno);
          kayit.iz = ' ';
```

```
            kaypos = (kayno-1) * kayuz;
            fseek(dd, kaypos, SEEK_SET);
            fread(&kayit, sizeof(kayit), 1, dd);
            if(kayit.iz == '*')
            {    printf("Adi :%s\n", kayit.ad) ;
                 printf("Soyadi :%s\n", kayit.soyad);
                 printf("Telefonu :%s\n", kayit.tel);
            }
            else
                 printf("Boyle bir kayit yok.\n");

            printf("\nDevam etmek istiyor musunuz (E/H) :");
            cevap = getche() ;
       }
       while(cevap != 'h' && cevap != 'H');
       fclose(dd);
}
```

Bu program çalıştırıldığında, okumak istediğimiz kaydın numarasını sorar ve girilen numaraya göre okuyucu kafa belli bir alana konumlanır. Bundan sonra belirli bir uzunluktaki alanı okuyarak o kayda ait bütün elemanları elde etmiş olur.

Dikkat edilmesi gereken nokta şudur: Bu tip uygulamalarda belirli bir alandaki bilgiler aynı anda okunmaktadır. Bu okunan alanda `struct` tipi tanımlanan değişkenin tüm elemanları vardır. Yukarıdaki iki program birbiriyle bağlantılı olup menü seçeneği eklenerek birleştirilebilir. Birinci program `"rehber.txt"` dosyasına kayıt yapmakta, ikinci program ise bu kayıtları okumaktadır.

`kayit`	değişkeni `struct` olarak tanımlanmış olup, elemanları şunlardır:
`iz`	belirli bir alanda kayıt olup olmadığını belirler. Alanın başında (yani o alanın ilk byte'ında '*' işareti varsa o alanda kayıt var demektir.
`ad[10]`	kişinin adı
`soyad[10]`	kişinin soyadı
`tel[10]`	kişinin telefon numarası

Yukarıdaki iki program birleştirilerek, menü sistemli daha gelişmiş bir uygulama haline getirilebilir.

Aşağıdaki Müşteri Hesap Programları incelemeniz açısından Deitel & Deitel'ın C How to Program (3rd Edition, Chapter 11) adlı kitabınından alınmıştır.

```
/*fig11_12.c Writing to a random access file */

#include <stdio.h>

struct clientData {
   int acctNum;
   char lastName[15];
   char firstName[10];
   double balance;
};
```

```
int main()
{
   FILE *cfPtr;
   struct clientData client = { 0, "", "", 0.0 };
   if ( ( cfPtr = fopen( "credit.dat", "w+" ) ) == NULL )
      printf( "File could not be opened.\n" );
   else {
      printf( "Enter account number"
              " ( 1 to 100, 0 to end input )\n? " );
      scanf( "%d", &client.acctNum );
      while ( client.acctNum != 0 ) {
         printf( "Enter lastname, firstname, balance\n? " );
         fscanf( stdin, "%s%s%lf", client.lastName,
                 client.firstName, &client.balance );
         fseek( cfPtr, ( client.acctNum - 1 ) *
                 sizeof( struct clientData ), SEEK_SET );
         fwrite( &client, sizeof( struct clientData ), 1,
                  cfPtr );
         printf( "Enter account number\n? " );
         scanf( "%d", &client.acctNum );
      }
      fclose( cfPtr );
   }
   return 0;
}

/* fig11_15.c Reading a random access file sequentially */
#include <stdio.h>

struct clientData {
   int acctNum;
   char lastName[ 15 ];
   char firstName[ 10 ];
   double balance;
};

int main()
{
   FILE *cfPtr;
   struct clientData client = { 0, "", "", 0.0 };

   if ( ( cfPtr = fopen( "credit.dat", "r" ) ) == NULL )
      printf( "File could not be opened.\n" );
   else {
      printf( "%-6s%-16s%-11s%10s\n", "Acct", "Last Name",
              "First Name", "Balance" );

      while ( !feof( cfPtr ) ) {
         fread( &client, sizeof( struct clientData ), 1,
                cfPtr );

         if ( client.acctNum != 0 )
            printf( "%-6d%-16s%-11s%10.2f\n",
```

```
                        client.acctNum, client.lastName,
                        client.firstName, client.balance );
            }
            fclose( cfPtr );
         }
         return 0;
      }


/* fig11_16.c (menü Türkçeleştirilmiştir)
Müşteri takip programı. Bu program random access dosyayı sıralı
erişim olarak okur, dosyayı güncelleştirir, dosyaya yeni kayıt ekler
ve kayıt siler.
*/

#include <stdio.h>

void display( FILE * );
void updateRecord( FILE * );
void newRecord( FILE * );
void deleteRecord( FILE * );
int enterChoice( void );

struct clientData
{  int acctNum;
   char lastName[ 15 ];
   char firstName[ 10 ];
   double balance;
};

void main()
{
   FILE *cfPtr;
   int choice;
   if ( ( cfPtr = fopen( "d:\\credit.dat", "r+" ) ) == NULL )
      printf( "File could not be opened.\n" );
   else
   {  while ( ( choice = enterChoice() ) != 5 )
      {  switch ( choice )
          {
            case 1:
               display( cfPtr ); break;
            case 2:
               updateRecord( cfPtr ); break;
            case 3:
               newRecord( cfPtr ); break;
            case 4:
               deleteRecord( cfPtr ); break;
         }
      }
      fclose( cfPtr );
   }
}
```

```
void display( FILE *readPtr )
{  struct clientData client = { 0, "", "", 0.0 };
   rewind( readPtr );
   printf( "%-6s%-16s%-11s%10s\n",
     "Hes#", "Soyad", "Ad","Alacak" );

      while ( !feof( readPtr ) )
      {  fread( &client, sizeof( struct clientData ), 1, readPtr );
         if ( client.acctNum != 0 && !feof( readPtr ))
            printf( "%-6d%-16s%-11s%10.2f\n",
              client.acctNum, client.lastName,
              client.firstName, client.balance );
      }
}

void updateRecord( FILE *fPtr )
{  int account;
   double transaction;
   struct clientData client = { 0, "", "", 0.0 };

   printf( "Enter account to update ( 1 - 100 ): " );
   scanf( "%d", &account );
   fseek( fPtr, ( account - 1 ) * sizeof( struct clientData ),
         SEEK_SET );
   fread( &client, sizeof( struct clientData ), 1, fPtr );

   if ( client.acctNum == 0 )
      printf( "Acount #%d has no information.\n", account );
   else
   {  printf( "%-6d%-16s%-11s%10.2f\n\n", client.acctNum,
         client.lastName, client.firstName, client.balance );
      printf( "Enter charge ( + ) or payment ( - ): " );
      scanf( "%lf", &transaction );
      client.balance += transaction;
      printf( "%-6d%-16s%-11s%10.2f\n",
         client.acctNum, client.lastName,
         client.firstName, client.balance );
      fseek( fPtr, ( account - 1 ) * sizeof( struct clientData ),
         SEEK_SET );
      fwrite( &client, sizeof( struct clientData ), 1, fPtr );
   }
}

void deleteRecord( FILE *fPtr )
{  struct clientData client, blankClient = { 0, "", "", 0.0 };
   int accountNum;

   printf( "Enter account number to delete ( 1 - 100 ): " );
   scanf( "%d", &accountNum );
   fseek( fPtr, ( accountNum - 1 ) * sizeof( struct clientData ),
         SEEK_SET );
   fread( &client, sizeof( struct clientData ), 1, fPtr );
```

```
   if ( client.acctNum == 0 )
      printf( "Account %d does not exist.\n", accountNum );
   else
   {  fseek( fPtr, ( accountNum - 1 ) * sizeof( struct clientData ),
         SEEK_SET );
      fwrite( &blankClient, sizeof( struct clientData ), 1, fPtr );
   }
}

void newRecord( FILE *fPtr )
{  struct clientData client = { 0, "", "", 0.0 };
   int accountNum;
   printf( "Enter new account number ( 1 - 100 ): " );
   scanf( "%d", &accountNum );
   fseek( fPtr, ( accountNum - 1 ) * sizeof( struct clientData ),
         SEEK_SET );
   fread( &client, sizeof( struct clientData ), 1, fPtr );

   if ( client.acctNum != 0 )
      printf( "Account #%d already contains information.\n",
         client.acctNum );
   else
   {  printf( "Enter lastname, firstname, balance\n? " );
      scanf( "%s%s%lf", &client.lastName, &client.firstName,
         &client.balance );
      client.acctNum = accountNum;
      fseek( fPtr, ( client.acctNum - 1 )*sizeof( struct clientData),
         SEEK_SET );
      fwrite( &client, sizeof( struct clientData ), 1, fPtr );
   }
}

int enterChoice( void )
{  int menuChoice;
   printf( "\nSecenekler\n"
      "1 - Hesap Goruntuleme\n"
      "2 - Hesap Guncelleme\n"
      "3 - Yeni Hesap\n"
      "4 - Hesap Sil\n"
      "5 - Cikis\n? " );
   scanf( "%d", &menuChoice );
   return menuChoice;
}
```