Bölüm 11 – Dosya İşleme

İçerik

- Veri Dosyaları
  - C programı ile oluşturulabilir, yenilenebilir ve işlenebilir
  - Büyük çaplı verilerin sürekli saklanması için kullanılır
    - Değişken ve dizilerdeki veri depolamaları sadece geçicidir

Kayıt Anahtarı

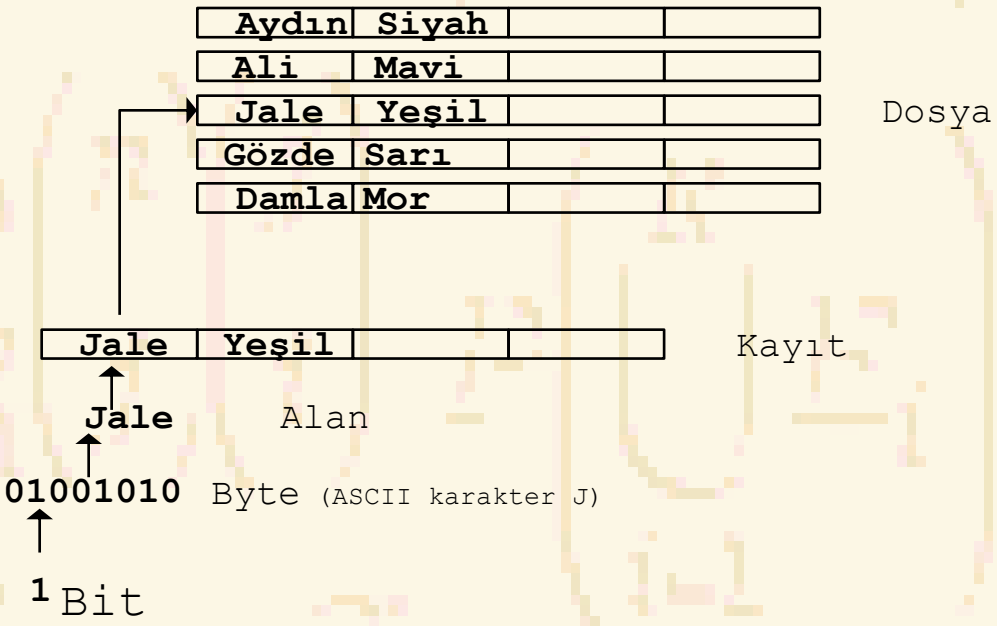
Bir dosyadan belli bir kayıdı almak için belirleyici

Dizisel dosya

Kayıtlar tipik olarak anahtar ile sıralanır

- Veri Düzeni:
  - Bit – en küçük veri parçası
    - Değeri `0` veya `1`
  - Byte – 8 bit
    - Karakter yüklemek için kullanılır
      - Desimal rakamlar, harfler, ve özel semboller
  - Alan– Anlam ifade eden karakter grubu
    - Örnek: sizin dersiniz
  - Kayıt– İlişkili alanlar grubu
    - `struct` veya `class` ile temsil edilir
    - Örnek: Bir ödeme sisteminde, belirli bir işçi için kimlik no, ad, adres v.s içeren bir kayıt

- Veri Düzeni (devam):
  - Dosya– ilişkili kayıtlar grubu
    - Örnek: ödeme dosyası
  - Veri tabanı (Database) – ilişkili dosyalar grubu



11.3 Dosyalar ve Kanallar (Streams)

- C her dosyayı bir byte dizisi olarak görür
  - Dosya *end-of-file belirteci* ile sonlanır
    - veya, belirtilen bir byte-da sonlanır
- Bir dosya açıldığında kanal oluşturulur
  - Dosya ve programlar arasında iletişim sağlar
  - Bir dosyayı açmak bir **`DOSYA (FILE )`** yapısına bir pointer gönderir
    - Örnek dosya pointer-ları:
    - **`stdin`** - standard input (klavye)
    - **`stdout`** - standard output (ekran)
    - **`stderr`** - standard hata (ekran)
- **`FILE`** yapısı
  - File tanımlayıcı
    - Açık dosya tablosu olarak adlandırılan, işletim sistemi dizisine, bir indeks
  - File Kontrol Bloğu (FCB)
    - Her dizi elemanında bulunur, sistem bunu dosyayı kaydetmek için kullanır

11.3 Dosyalar ve Kanallar (Streams)

- Standard kütüphanedeki Read/Write fonksiyonları
  - **`fgetc`**
    - Bir dosyadan bir karakter okur
    - Bir **`FILE`** pointer-ını argüment olarak alır
    - **`fgetc( stdin )`**, **`getchar()`** –a denktir
  - **`fputc`**
    - Bir dosyaya bir karakter yazar
    - Bir **`FILE`** pointer ve bir karakteri bir argüment olarak yazmak için alır
    - **`fputc( 'a', stdout )`**, **`putchar( 'a' )`** –a denktir
  - **`fgets`**
    - Bir dosyadan bir satır okur
  - **`fputs`**
    - Bir dosyaya bir satır yazar
  - **`fscanf`** / **`fprintf`**
    - **`scanf`** ve **`printf`** –e denk dosya işlemciler

11.4 Dizisel Erişim Dosyası Oluşturma

- C bir dosya yapısına zorlamaz
  - Bir dosyadaki kayıt yapısı tipi yoktur
  - Dosya yapısını programcı hazırlamalıdır
- Bir dosya oluşturma
  - **`FILE *fPtr;`**
    - **`fPtr`** adlı bir **`FILE`** pointer oluşturur
  - **`fPtr = fopen("ilkFile.dat", açmamodu);`**
    - **`fopen`** fonksiyonu belirlenen dosyaya bir **`FILE`** pointer gönderir
    - İki argüment alır– açılacak dosya ve açma modu
    - Açılış gerçekleşmezse, **`NULL`** gönderir
  - **`fprintf`**
    - Dosyaya yazmak için kullanılır
    - Printf –e benzerdir, sadece ilk argüment bir **`FILE`** pointerdır (yazılacak dosyaya pointer)

- **`feof(`** ***`FILE`*** *pointer* **`)`**
  - Eğer end-of-file belirteci (işlenecek daha fazla veri yok) belirtilen dosya için kuruluysa doğru gönderir
  - **`fclose(`** ***`FILE`*** *pointer* **`)`** doğru gönderir
  - Belirtilen dosyayı kapatır
  - Program sonunda otomatik olarak işlenir
  - Dosya kapatmada iyi bir pratiktir

- Detaylar
  - Programlar tek yada çok dosyayı işleyebilir veya hiç bir dosyayı işlemeyebilir
  - Her dosya tek bir özel ada ve pointer-a sahip olmalıdır

- Dosya açma modları:

Mod	Açıklama
`r`	Okumak için dosya aç.
`w`	Yazmak için dosya oluştur. Halihazırda dosya mevcut ise, içeriğini yok eder.
`a`	Ekleme(append); varolan dosyanın sonuna ekleme yapar veya yeni dosya oluşturur
`r+`	Yenilemek için dosya açar (okuma ve yazma).
`w+`	Yenilemek için dosya oluşturur. Halihazırda dosya mevcut ise, içeriğini yok eder.
`a+`	Ekleme; Yenilemek için dosya açar veya oluşturur; Dosyanın sonuna yazılır.

```c
/* Fig. 11.3: fig11_03.c
   Dizisel dosya oluşturma */
#include <stdio.h>

int main()
{
   int hesap;
   char isim[ 30 ];
   double balans;
   FILE *mfPtr;   /* mfPtr = musteri.dat file pointer */

   if ( ( mfPtr = fopen( "musteri.dat", "w" ) ) == NULL )
      printf( "Dosya açılamıyor!\n" );
   else {
      printf( "Hesap, isim, ve balansı giriniz.\n" );
      printf( "Girdiyi bitirmek için  EOF  giriniz.\n" );
      printf( "? " );
      scanf( "%d%s%lf", &hesap, isim, &balans );

      while ( !feof( stdin ) ) {
         fprintf( mfPtr, "%d %s %.2f\n",
                  hesap, isim, balans);
         printf( "? " );
         scanf( "%d%s%lf", &hesap, isim, &balans);
      }

      fclose( mfPtr );
   }

   return 0;
}
```

11.4 Dizisel Erişim Dosyası Oluşturma

```
Hesap isim ve balansı giriniz.
Girdiyi bitirmek için EOF giriniz.
? 100 Aydın 24.98
? 200 Yılmaz 345.67
? 300 Koç 0.00
? 400 Kaya -42.16
? 500 Demir 224.62
?
```

- Dizisel erişim dosyası okuma
  - Bir **`FILE`** pointer oluştur, okunacak dosyaya bağla:

```
oPtr = fopen("dosyam.dat", "r" );
```

  - Dosyadan okumak için **`fscanf`** kullan
    - **`scanf`** gibi, sadece ilk argüment bir **`FILE`** pointer

```
fscanf( oPtr, "%d%s%f", &birInt, &birString,
   &birFloat );
```

  - Veri baştan sona okunur
  - Dosya pozisyon pointerı (file position pointer)
    - Okunacak/yazılacak bir sonraki byte sayısını belirtir
    - Gerçek anlamda pointer olmayıp bir tamsayı değerdir (byte yerini belirler)
    - byte offset olarak da adlandırılır
  - **`rewind( oPtr )`**
    - Dosya pozisyon pointerını dosya başlangıcına getirir

```c
/* Fig. 11.7: fig11_07.c
   Dizisel dosyayı okuma yazma */
#include <stdio.h>

int main()
{
   int hesap;
   char isim[ 30 ];
   double balans;
   FILE *mfPtr;   /* mfPtr = musteri.dat file pointer */

   if ( ( mfPtr = fopen("musteri.dat", "r" ) ) == NULL )
      printf( "File could not be opened\n" );
   else {
      printf( "%-10s%-13s%s\n", "Hesap", "İsim", "Balans" );
      fscanf( cfPtr, "%d%s%lf", &hesap, isim, &balans );

      while ( !feof( mfPtr ) ) {
         printf( "%-10d%-13s%7.2f\n", hesap, isim, balans );
         fscanf( mfPtr, "%d%s%lf", &hesap, isim, &balans );
      }

      fclose( mfPtr );
   }

   return 0;
}
```

Hesap	İsim	Balans
100	Aydın	24.98
200	Yılmaz	345.67
300	Koç	0.00
400	Kaya	-42.16
500	Demir	224.62

```
1  /* Fig. 11.8: fig11_08.c
2     Kredi sorgulama programı */
3  #include <stdio.h>
4
5  int main()
6  {
7     int istem, hesap;
8     double balance;
9     char isim[ 30 ];
10    FILE *mfPtr;
11
12    if ( ( mfPtr = fopen( "musteri.dat", "r" ) ) == NULL )
13       printf( " Dosya açılamadı\n" );
14    else {
15       printf(" İsteğinizi giriniz\n "
16              " 1 - Sıfır balanslı hesapları listele\n"
17              " 2 - Kredi balanslı hesapları listele\n "
18              " 3 - Açık balanslı hesapları listele\n "
19              " 4 - İşlemi sonlandır? " );
20       scanf( "%d", &istem );
21
22       while ( istem != 4 ) {
23          fscanf( mfPtr, "%d%s%lf", &hesap, isim,
24                  &balans );
25
26          switch ( istem ) {
27             case 1:
28                printf( "\nSıfır balanslı "
29                        "hesaplar:\n" );
30
31                while ( !feof( mfPtr ) ) {
32
```

```
33                if ( balanS == 0 )
34                   printf( "%-10d%-13s%7.2f\n",
35                          hesap, isim, balanS );
36
37                fscanf( mfPtr, "%d%s%lf",
38                       &hesap, isim, &balans );
39             }
40
41             break;
42          case 2:
43             printf( "\nKredi balanslı "
44                     "hesaplar:\n" );
45
46             while ( !feof( mfPtr ) ) {
47
48                if ( hesap < 0 )
49                   printf( "%-10d%-13s%7.2f\n",
50                          hesap, isim, balans);
51
52                fscanf( mfPtr, "%d%s%lf",
53                       &hesap, isim, &balans );
54             }
55
56             break;
57          case 3:
58             printf( "\n Açık balanslı "
59                     "hesaplar:\n" );
60
61             while ( !feof( mfPtr ) ) {
62
63                if ( balanS > 0 )
64                   printf( "%-10d%-13s%7.2f\n",
```

11.5 Dizisel Erişim Dosyasından Veri Okuma

```
65                              hesap, isim, balans );
66
67                      fscanf( mfPtr, "%d%s%lf",
68                              &hesap, isim, &balans );
69                   }
70
71                   break;
72             }
73
74             rewind( mfPtr );
75             printf( "\n? " );
76             scanf( "%d", &istem );
77          }
78
79          printf( "İşlem sonu.\n" );
80          fclose( mfPtr );
81       }
82
83       return 0;
84    }
```

11.5 Dizisel Erişim Dosyasından Veri Okuma

```
İsteğinizi giriniz
  1 - Sıfır balanslı hesapları listele
  2 - Kredi balanslı hesapları listele
  3 - Açık balanslı hesapları listele
  4 - İşlem sonu
? 1

Sıfır balanslı hesaplar:
300         Koç              0.00

? 2

Kredi balanslı hesaplar:
400         Kaya            -42.16

? 3

Açık balanslı hesaplar:
100         Aydın             24.98
200         Yılmaz            345.67
500         Demir             224.62
? 4
İşlem sonu.
```

- Dizisel erişim dosyası
  - Düzenleme sırasında diğer verileri yok etme riski vardır
  - Alan boyutları değişebilir
    - Dosya ve ekran gösterimi iç(makine) gösterimden farklı
    - **1**, **34**, **-890** **`int`** değerlerdir, fakat diskde farklı boyuta sahiptirler

```
300 Koç 0.00 400 Kaya 32.87     (Dosyadaki eski veriler)
```

Koç adını Kerimoğlu ile değiştirmek istersek,

```
300 Kerimoğlu 0.00
```

↓

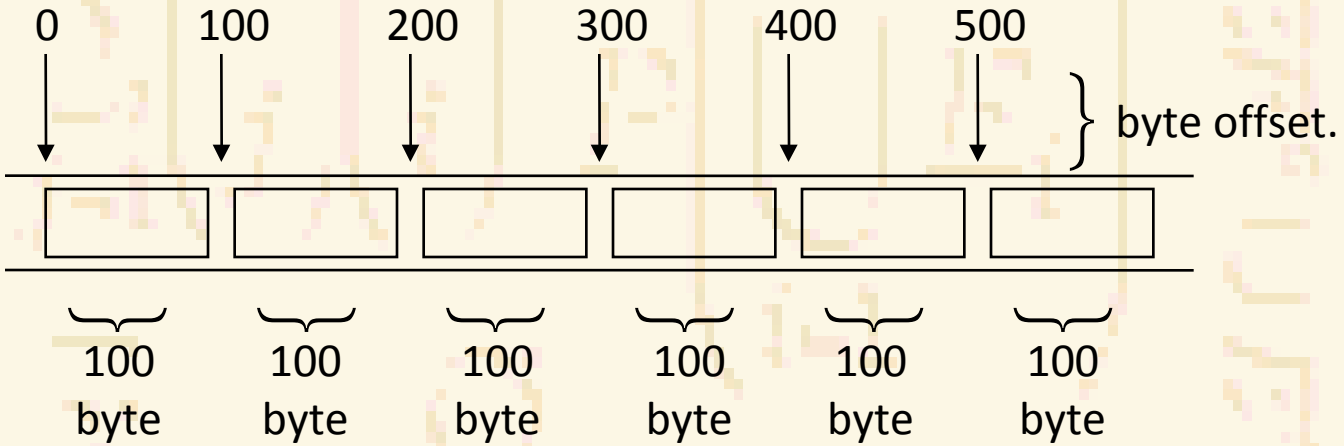
```
300 Koç 0.00 400 Kaya 32.87
```

↓

```
300 Kerimoğlu 0.00aya 32.87
```

Verinin üzerine yazılır

- Rasgele erişim dosyaları
  - Diğer kayıtları incelemeden belli bir kayıta bağlantı sağlar
  - Bir dosyadaki kayıtlara doğrudan bağlantı
  - Diğer verileri silmeden veri girişi yapılabilir
  - Eski veriler üzerine yazılmadan yenilenebilir veya silinebilir
- Sabit uzunluklu kayıtlar kullanarak düzenleme
  - Dizisel dosyalar sabit uzunluklu kayıtlara sahip değildir



11.7 Rasgele Erişim Dosyası Oluşturma

- Rasgele erişim dosyasında veriler
  - Formatsız (“ham byte" olarak yüklenir)
    - Aynı tipten veriler (örneğin **`int`**) aynı miktarda bellek kullanır
    - Aynı tipten veriler sabit uzunluğa sahiptir
    - Veri insan okuyabilir değildir

- Formatsız I/O fonksiyonları
  - **`fwrite`**
    - Bellekteki bir yerdeki byte-ları bir dosyaya aktarır
  - **`fread`**
    - Bir dosyadaki byte-ları bellekte bir yere aktarır
  - Örnek:

```
fwrite( &sayi, sizeof( int ), 1, mPtr );
```

    - **`&sayi`** – Aktarılacak byte-ların bellekteki yeri
    - **`sizeof( int )`** – Aktarılacak byte sayısı
    - **`1`** – Diziler için, aktarılacak eleman sayısı
      - Bu durumda, dizinin "bir elemanı" aktarılacak
    - **`mPtr`** – Aktarılacak dosya

11.7 Rasgele Erişim Dosyası Oluşturma

- Yazım yapısı

```
fwrite( &nesne, sizeof (struct yapi), 1, mPtr );
```

  - **`sizeof`** – parantez içindeki nesnenin byte sayısını gönderir
- Bir çok dizi elemanını yazmak için
  - İlk argüment diziye pointer
  - Üçüncü argüment yazılacak eleman sayısı

```c
/* Fig. 11.11: fig11 11.c
   Rasgele erişim dosyasını dizisel olarak oluşturma */
#include <stdio.h>

struct musteriData {
   int musNum;
   char soyad[ 15 ];
   char ad[ 10 ];
   double balans;
};

int main()
{
   int i;
   struct musteriData bosMus = { 0, "", "", 0.0 };
   FILE *mfPtr;

   if ( ( mfPtr = fopen( "kredi.dat", "w" ) ) == NULL )
      printf( "Dosya açılamadı.\n" );
   else {

      for ( i = 1; i <= 100; i++ )
         fwrite( &bosMus,
                sizeof( struct musteriData ), 1, mfPtr );

      fclose( mfPtr );
   }

   return 0;
}
```

11.8 Rasgele Erişim Dosyasına Rasgele Veri Yazma

- **`fseek (dosya arama)`**
  - Dosya pozisyon pointer-ını (offset) belli bir pozisyona getirir
  - **`fseek(`** *pointer*, *offset*, *sembolik_sabit* **`);`**
    - *pointer* – dosyaya pointer
    - *offset* – dosya pozisyon pointer-ı (0 ilk pozisyon (yer))
    - *sembolik_sabit* – dosyanın okunan yerini belirler
    - **`SEEK_SET`** – arama dosyanın başından başlar
    - **`SEEK_CUR`** – arama dosyadaki o anki yerden başlar
    - **`SEEK_END`** – dosyanın sonundan başlar

```c
/* Fig. 11.12: fig11 12.c
   Rasgele erişim dosyasına yazma */
#include <stdio.h>

struct musteriData {
   int musNum;
   char soyad[ 15 ];
   char ad[ 10 ];
   double balans;
};

int main()
{
   FILE *mfPtr;
   struct musteriData musteri = { 0, "", "", 0.0 };

   if ( ( mfPtr = fopen( "kredi.dat", "r+" ) ) == NULL )
      printf( "Dosya açılamadı.\n" );
   else {
      printf( "Hesap numarasını giriniz"
              " ( 1 den 100 e, girdi sonu için 0 )\n? " );
      scanf( "%d", &musteri.musNum );

      while ( musteri.musNum != 0 ) {
         printf( "Soyadı, adı ve balansı gir\n? " );
         fscanf( stdin, "%s%s%lf", musteri.soyad,
                 musteri.ad, &musteri.balans );
         fseek( mfPtr, ( musteri.musNum - 1 ) *
                sizeof( struct musteriData ), SEEK SET );
         fwrite( &musteri, sizeof( struct musteriData ), 1,
                 mfPtr );
         printf( "Hesap numarasını gir \n? " );
```

11.8 Rasgele Erişim Dosyasına Rasgele Veri Yazma

```
33            scanf( "%d", &musteri.musNum );
34         }
35
36         fclose( mfPtr );
37      }
38
39      return 0;
40   }
```

```
Hesap numarasını giriniz (1 den 100 e, girdi sonu için 0 )
? 37
Soyadı, adı ve balansı gir
? Koç Ali 0.00
Hesap numarasını gir
? 29
Soyadı, adı ve balansı gir
? Demir Ahmet -24.54
Hesap numarasını gir
? 96
Soyadı, adı ve balansı gir
? Kaya Sami 34.98
```

```
Hesap numarasını gir
? 88
Soyadı, adı ve balansı gir
? Aydın Tuncay 258.34
Hesap numarasını gir
? 33
Soyadı, adı ve balansı gir
? Yılmaz Mehmet 314.33
Hesap numarasını gir
? 0
```

11.9 Rasgele Erişim Dosyasından Dizisel (Ardışık) Veri Okuma

- **`fread (dosya oku)`**
  - Bir dosyadan belleğe belli sayıda byte okur

```
fread( &musteri, sizeof (struct
  musteriData), 1, mPtr );
```

  - Bir çok sabit boyutlu dizi elemanı okuyabilir
    - Diziye pointer sağla
    - Okunacak eleman sayısını belirt
  - Çoklu eleman okumak için, üçüncü argümentte belirt

...ık) Veri Okuma

```c
/* Fig. 11.15: fig11 15.c
   Dizisel rasgele erişim dosyası okuma */
#include <stdio.h>

struct musteriData {
   int musNum;
   char soyad[ 15 ];
   char ad[ 10 ];
   double balans;
};

int main()
{
   FILE *mfPtr;
   struct musteriData musteri = { 0, "", "", 0.0 };

   if ( ( mfPtr = fopen( "kredi.dat", "r" ) ) == NULL )
      printf( "Dosya açılamadı.\n" );
   else {
      printf( "%-6s%-16s%-11s%10s\n", "Hesap", "Soyadı",
              "Adı", "Balans" );

      while ( !feof( mfPtr ) ) {
         fread( &musteri, sizeof( struct musteriData ), 1,
                mfPtr );

         if ( musteri.musNum != 0 )
            printf( "%-6d%-16s%-11s%10.2f\n",
                    musteri.musNum, musteri.soyad,
                    musteri.ad, musteri.balans );
      }
```

11.9 Rasgele Erişim Dosyasından Dizisel (Ardışık) Veri Okuma

```
33          fclose( mfPtr );
34       }
35
36       return 0;
37    }
```

Hesap	Soyadı	Adı	Balans
29	Demir	Nadir	-24.54
33	Tosun	Mustafa	314.33
37	Berke	Ayşe	0.00
88	Koç	Ahmet	258.34
96	Deniz	Salim	34.98

11.10 Örnek: Bir Veri İşlem Programı

- Bu program
  - rasgele erişim dosyası kullanarak bir banka müşteri sistemine erişim sağlamak içindir
- Yapılacak işlemler
  - Varolan hesapları yenileme
  - Yeni hesap açma
  - Hesap silme
  - Tüm hesapları bir formatlı teks dosyasına yükleme

```
1  /* Fig. 11.16: fig11 16.c
2     Bu program dizisel rasgele erişim dosyası okur,
3     dosyada olan verileri yeniler, dosyaya eklemek için
4     yeni veri oluşturur, ve dosyadan
5     veri siler.                                    */
6  #include <stdio.h>
7
8  struct musData {
9     int musNum;
10    char soyad[ 15 ];
11    char ad[ 10 ];
12    double balanS;
13 };
14
15 int secimGir( void );
16 void teksFile( FILE * );
17 void yenileKayit( FILE * );
18 void yeniKayit( FILE * );
19 void silKayit( FILE * );
20
21 int main()
22 {
23    FILE *mfPtr;
24    int secim;
25
26    if ( ( mfPtr = fopen( "kredi.dat", "r+" ) ) == NULL )
27       printf( "Dosya açılamadı.\n" );
28    else {
29
30       while ( ( secim = secimGir() ) != 5 ) {
31
32          switch ( secim ) {
```

```
            case 1:
               teksFile( mfPtr );
               break;
            case 2:
               yenileKayit( mfPtr );
               break;
            case 3:
               yeniKayit( mfPtr );
               break;
            case 4:
               silKayit( mfPtr );
               break;
         }
      }

      fclose( mfPtr );
   }

   return 0;
}

void teksFile( FILE *okuPtr )
{
   FILE *yazPtr;
   struct musData musteri = { 0, "", "", 0.0 };

   if ( ( yazPtr = fopen( "hesaplar.txt", "w" ) ) == NULL )
      printf( "Dosya açılamadı.\n" );
   else {
      rewind( okuPtr );
      fprintf( yazPtr, "%-6s%-16s%-11s%10s\n",
               "Hesap", "Soyadı", "Adı","Balans" );
```

```
65
66          while ( !feof( okuPtr ) ) {
67             fread( &musteri, sizeof( struct musData ), 1,
68                    okuPtr );
69
70             if ( musteri.musNum != 0 )
71                fprintf( yazPtr, "%-6d%-16s%-11s%10.2f\n",
72                         musteri.musNum, musteri.soyad,
73                         musteri.ad, musteri.balans);
74          }
75
76          fclose( yazPtr );
77       }
78
79    }
80
81    void yenileKayit( FILE *fPtr )
82    {
83       int hesap;
84       double islem;
85       struct musData musteri = { 0, "", "", 0.0 };
86
87       printf( "Yenilenecek hesap no gir ( 1 - 100 ): " );
88       scanf( "%d", &hesap );
89       fseek( fPtr,
90              ( hesap - 1 ) * sizeof( struct musData ),
91              SEEK SET );
92       fread( &musteri, sizeof( struct musteriData ), 1, fPtr );
93
94       if ( musteri.musNum == 0 )
95          printf( "Hesap #%d hakkında bilgi yok.\n", hesap );
96       else {
```

```
97        printf( "%-6d%-16s%-11s%10.2f\n\n",
98                musteri.musNum, musteri.soyad,
99                musteri.ad, musteri.balans );
100       printf( "Ekleme ( + ) veya ödemeyi ( - ) gir: " );
101       scanf( "%lf", &islem );
102       musteri.balans += islem;
103       printf( "%-6d%-16s%-11s%10.2f\n",
104               musteri.musNum, musteri.soyad,
105               musteri.ad, musteri.balans );
106       fseek( fPtr,
107              ( hrsap - 1 ) * sizeof( struct musData ),
108              SEEK_SET );
109       fwrite( &musteri, sizeof( struct musData ), 1,
110               fPtr );
111    }
112 }
113
114 void silKayit( FILE *fPtr )
115 {
116    struct musData musteri,
117                         bosmusteri = { 0, "", "", 0 };
118    int hesapNum;
119
120    printf( "Silinecek hesap no yu "
121            "gir ( 1 - 100 ): " );
122    scanf( "%d", &hesapNum );
123    fseek( fPtr,
124           ( hesapNum - 1 ) * sizeof( struct musData ),
125           SEEK_SET );
126    fread( &musteri, sizeof( struct musData ), 1, fPtr );
```

```
127
128    if ( musteri.musNum == 0 )
129       printf( "Hesap %d yoktur.\n", hesapNum );
130    else {
131       fseek( fPtr,
132          ( hesapNum - 1 ) * sizeof( struct musData ),
133          SEEK_SET );
134       fwrite( &bosMusteri,
135               sizeof( struct musData ), 1, fPtr );
136    }
137 }
138
139 void yeniKayit( FILE *fPtr )
140 {
141    struct musData musteri = { 0, "", "", 0.0 };
142    int hesapNum;
143    printf( "Yeni hesap no gir ( 1 - 100 ): " );
144    scanf( "%d", &hesapNum );
145    fseek( fPtr,
146           ( hesapNum - 1 ) * sizeof( struct musData ),
147           SEEK_SET );
148    fread( &musteri, sizeof( struct musData ), 1, fPtr );
149
150    if ( musteri.musNum != 0 )
151       printf( "Hesap #%d bir hesaba sahiptir. Yeni no gir \n",
152               musteri.musNum );
153    else {
154       printf( "Soyadı adı ve balansı gir\n? " );
155       scanf( "%s%s%lf", &musteri.soyad, &musteri.ad,
156              &musteri.balans );
```

```
157       musteri.musNum = hesapNum;
158       fseek( fPtr, ( musteri.musNum - 1 ) *
159             sizeof( struct musData ), SEEK_SET );
160       fwrite( &musteri,
161               sizeof( struct musData ), 1, fPtr );
162    }
163 }
164
165 int secimGir( void )
166 {
167    int menuSecim;
168
169    printf( "\n Seçiminiz ?\n"
170       "1 - Çağrılan hesabın formatlı teks dosyasını yükle\n"
171       "     \"hesaplar.txt\" ye yazılacak \n"
172       "2 - Hesabı yenile\n"
173       "3 - Yeni hesap ekle\n"
174       "4 - bir hesabı sil\n"
175       "5 - Program sonu\n? " );
176    scanf( "%d", &menuSecim );
177    return menuSecim;
178 }
```

11.10 Örnek: Bir Veri İşlem Programı

Seçim1 yapıldığında hesaplar.txt nin içeriği :

Hesap	Soyadı	Adı	Balans
29	Kara	Ahmet	-24.54
33	Duman	Mehmet	314.33
37	Kaya	Ali	0.00
88	sever	Hale	258.34
96	Demir	Sami	34.98

```
Yenilenecek hesap no gir (1 - 100): 37
37      Kaya                Ali                 0.00

Ekleme ( + ) veya ödemeyi ( - ) gir:  +87.99
37      Kaya                Ali                 87.99
```

```
Yeni hesap no gir (1 - 100): 22
Soyad ad ve balansı gir
? Kandemir Serap 247.45
```