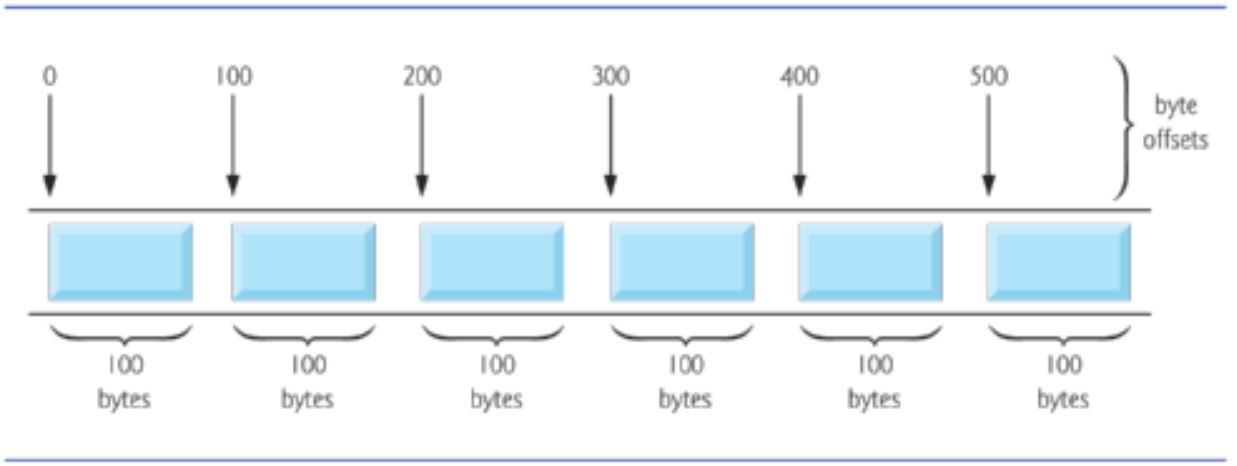
CME 112- Programming Languages II
Lecture 8: File Operations (Part-2)

Assist. Prof.Dr. Ümit ATİLA

RANDOM ACCESS FILES

- Random access files
  - Access individual records without searching through other records
  - Instant access to records in a file
  - Data can be inserted without destroying other data
  - Data previously stored can be updated or deleted without overwriting
- Implemented using fixed length records
  - Sequential files do not have fixed length records



CREATING A RANDOM ACCESS FILE

- Data in random access files
  - Unformatted (stored as "raw bytes")
    - All data of the same type (**int**s, for example) uses the same amount of memory
    - All records of the same type have a fixed length
    - Data not human readable

CREATING A RANDOM ACCESS FILE

- Unformatted I/O functions
  - **fwrite**
    - Transfer bytes from a location in memory to a file
  - **fread**
    - Transfer bytes from a file to a location in memory
  - Example:

    fwrite(&number, sizeof(int), 1, myPtr);
    - **&number** – Location to transfer bytes from
    - **sizeof(int)** – Number of bytes to transfer
    - **1** – For arrays, number of elements to transfer

      In this case, "one element" of an array is being transferred
    - **myPtr** – File to transfer to or from

CREATING A RANDOM ACCESS FILE

- Writing structs

  fwrite(&myObject, sizeof (struct myStruct), 1, myPtr);

  - **sizeof** – returns size in bytes of object in parentheses
- To write several array elements
  - Pointer to array as first argument
  - Number of elements to write as third argument

CREATING A RANDOM ACCESS FILE

```c
#include <stdio.h>

struct musteri{
    int hesapNo;
    char soyad[25];
    char ad[20];
    double bakiye;
};

int main(void)
{
    int i;
    struct musteri bosMusteri = {0,"","",0.0};
    FILE *myPtr;
    if((myPtr = fopen("musteri.dat","w"))== NULL)
        printf("Dosya olusturulamadi\n");
    else
    {
        for(i=1;i<=100;i++)
        {
            fwrite(&bosMusteri,sizeof(struct musteri),1,myPtr);
        }
        fclose(myPtr);
    }
    return 0;
}
```

WRITING DATA RANDOMLY TO A RANDOM ACCESS FILE

- **fseek**
  - Sets file position pointer to a specific position
  - **fseek(pointer, offset, symbolic_constant);**
  - **pointer** – pointer to file
    - **offset** – file position pointer (0 is first location)
    - **symbolic_constant** – specifies where in file we are reading from
      - **SEEK_SET** – seek starts at beginning of file
      - **SEEK_CUR** – seek starts at current location in file
      - **SEEK_END** – seek starts at end of file

WRITING DATA RANDOMLY TO A RANDOM ACCESS FILE

```c
#include <stdio.h>

struct musteri{
    int hesapNo;
    char soyad[25];
    char ad[20];
    double bakiye;
};

int main(void)
{
    struct musteri hesapBilgi = {0,"","",0.0};
    FILE *myPtr;
    if((myPtr = fopen("musteri.dat","r+"))== NULL)
        printf("Dosya acilamadi\n");
    else
    {
        printf("Hesap no gir (1-100 arasi deger)\n"
            "Veri girisini bitirmek icin 0 gir");
        scanf("%d",&hesapBilgi.hesapNo);
        while(hesapBilgi.hesapNo!=0)
        {
            printf("Soyad Ad ve Bakiye gir\n?");
            fscanf(stdin,"%s%s%lf",hesapBilgi.soyad,
                hesapBilgi.ad,&hesapBilgi.bakiye);
```

WRITING DATA RANDOMLY TO A RANDOM ACCESS FILE

```
26
27              fseek(myPtr,(hesapBilgi.hesapNo-1)*
28                  sizeof(struct musteri),SEEK_SET);
29
30              fwrite(&hesapBilgi,sizeof(struct musteri),1,myPtr);
31
32              printf("Hesap no gir\n?");
33              scanf("%d",&hesapBilgi.hesapNo);
34          }
35          fclose(myPtr);
36      }
37      return 0;
38  }
```

READING DATA SEQUENTIALLY FROM A RANDOM ACCESS FILE

- fread
  - Reads a specified number of bytes from a file into memory

  fread(&client, sizeof (struct clientData), 1,myPtr);
  - Can read several fixed-size array elements
    - Provide pointer to array
    - Indicate number of elements to read
  - To read multiple elements, specify in third argument

READING DATA SEQUENTIALLY FROM A RANDOM ACCESS FILE

```c
#include <stdio.h>

struct musteri{
    int hesapNo;
    char soyad[25];
    char ad[20];
    double bakiye;
};

int main(void)
{
    struct musteri hesapBilgi = {0,"","",0.0};
    FILE *myPtr;
    if((myPtr = fopen("musteri.dat","r"))== NULL)
        printf("Dosya acilamadi\n");
    else
    {
        printf("%-10s%-16s%-11s%10s\n","HesapNo","Soyad","Ad","Bakiye");
        while(!feof(myPtr))
        {
            fread(&hesapBilgi,sizeof(struct musteri),1,myPtr);
            if(hesapBilgi.hesapNo!=0)
            printf("%-10d%-16s%-11s%10.2f\n",hesapBilgi.hesapNo,
                hesapBilgi.soyad,hesapBilgi.ad,hesapBilgi.bakiye);
        }
        fclose(myPtr);
    }
    getchar();
    return 0;
}
```

CASE STUDY: A TRANSACTION PROCESSING PROGRAM

- This program
  - Demonstrates using random access files to achieve instant access processing of a bank's account information
- We will
  - Update existing accounts
  - Add new accounts
  - Delete accounts
  - Store a formatted listing of all accounts in a text file

CASE STUDY: A TRANSACTION PROCESSING PROGRAM

```c
#include <stdio.h>

struct musteri{
    int hesapNo;
    char soyad[25];
    char ad[20];
    double bakiye;
};
int secimGir(void);
void textDosya(FILE *);
void kayitGuncelle(FILE *);
void yeniKayit(FILE *);
void kayitSil(FILE *);
void listele(FILE *);

int main(void)
{
    FILE *myPtr;
    int secim;
    if((myPtr = fopen("musteri.dat","r+"))== NULL)
        printf("Dosya acilamadi\n");
    else
    {
        while((secim = secimGir()) != 6)
        {
```

CASE STUDY: A TRANSACTION PROCESSING PROGRAM

```
26                  switch(secim)
27                  {
28                      case 1:textDosya(myPtr);break;
29                      case 2:kayitGuncelle(myPtr);break;
30                      case 3:yeniKayit(myPtr);break;
31                      case 4:kayitSil(myPtr);break;
32                      case 5:listele(myPtr);break;
33                  }
34              }
35              fclose(myPtr);
36          }
37  }
```

CASE STUDY: A TRANSACTION PROCESSING PROGRAM

```
39  int secimGir()
40  {
41      int menuSecim;
42      printf("\n Secimini yap\n"
43          "1-musteri.dat dosyasinin icerigini\n"
44          "  formatli olarak \"hesaplar.dat\" dosyasina yaz\n"
45          "2-hesap guncelle\n"
46          "3-yeni hesap ekle\n"
47          "4-hesap sil\n"
48          "5-musteri.dat dosyasinin icerigini listele\n"
49          "6-cikis\n?");
50      scanf("%d",&menuSecim);
51      return menuSecim;
52  }
```

CASE STUDY: A TRANSACTION PROCESSING PROGRAM

```
54 void textDosya(FILE *okuPtr)
55 {
56     FILE *yazPtr;
57     struct musteri hesapBilgi = {0,"","",0.0};
58     if((yazPtr = fopen("hesaplar.dat","w"))== NULL)
59         printf("Dosya acilamadi\n");
60     else
61     {
62         rewind(okuPtr);
63         fprintf(yazPtr,"%-10s%-16s%-11s%10s\n","HesapNo","Soyad","Ad","Bakiye");
64         while(!feof(okuPtr))
65         {
66             fread(&hesapBilgi,sizeof(struct musteri),1,okuPtr);
67             if(hesapBilgi.hesapNo!=0)
68             fprintf(yazPtr,"%-10d%-16s%-11s%10.2f\n",hesapBilgi.hesapNo,
69                 hesapBilgi.soyad,hesapBilgi.ad,hesapBilgi.bakiye);
70         }
71         fclose(yazPtr);
72     }
73 }
```

CASE STUDY: A TRANSACTION PROCESSING PROGRAM

```
75 void kayitGuncelle(FILE *fPtr)
76 {
77     int hesapID;
78     double islemMiktari;
79     struct musteri hesapBilgi = {0,"","",0.0};
80     printf("Guncellenecek hesap no gir[1-100]:");
81     scanf("%d",&hesapID);
82     fseek(fPtr,(hesapID-1)*sizeof(struct musteri),SEEK_SET);
83     fread(&hesapBilgi,sizeof(struct musteri),1,fPtr);
84     if(hesapBilgi.hesapNo==0)
85         printf("%d nolu hesap için bilgi girilmemiş\n",hesapID);
86     else
87     {
88         printf("%-10d%-16s%-11s%10.2f\n\n",hesapBilgi.hesapNo,
89                 hesapBilgi.soyad,hesapBilgi.ad,hesapBilgi.bakiye);
90         printf("Hesaba yatacak (+) veya hesaptan cekilecek (-) tutari gir:");
91         scanf("%lf",&islemMiktari);
92         hesapBilgi.bakiye += islemMiktari;
93         printf("%-10d%-16s%-11s%10.2f\n\n",hesapBilgi.hesapNo,
94                 hesapBilgi.soyad,hesapBilgi.ad,hesapBilgi.bakiye);
95         fseek(fPtr,(hesapID-1)*sizeof(struct musteri),SEEK_SET);
96         fwrite(&hesapBilgi,sizeof(struct musteri),1,fPtr);
97     }
98 }
```

CASE STUDY: A TRANSACTION PROCESSING PROGRAM

```
100 void kayitSil(FILE *fPtr)
101 {
102     struct musteri hesapBilgi, bosHesap = {0,"","",0.0};
103     int hesapID;
104     printf("Silinecek hesap no gir[1-100]:");
105     scanf("%d",&hesapID);
106     fseek(fPtr,(hesapID-1)*sizeof(struct musteri),SEEK_SET);
107     fread(&hesapBilgi,sizeof(struct musteri),1,fPtr);
108     if(hesapBilgi.hesapNo==0)
109         printf("Silinecek %d nolu hesap yok",hesapID);
110     else
111     {
112         fseek(fPtr,(hesapID-1)*sizeof(struct musteri),SEEK_SET);
113         fwrite(&bosHesap,sizeof(struct musteri),1,fPtr);
114     }
115 }
```

CASE STUDY: A TRANSACTION PROCESSING PROGRAM

```
117 void yeniKayit(FILE *fPtr)
118 {
119     int hesapID;
120     struct musteri hesapBilgi = {0,"","",0.0};
121     printf("Yeni hesap no gir[1-100]:");
122     scanf("%d",&hesapID);
123     fseek(fPtr,(hesapID-1)*sizeof(struct musteri),SEEK_SET);
124     fread(&hesapBilgi,sizeof(struct musteri),1,fPtr);
125     if(hesapBilgi.hesapNo!=0)
126         printf("%d nolu hesap zaten mevcut\n",hesapID);
127     else
128     {
129         printf("Soyad, Ad ve bakiye gir:");
130         scanf("%s%s%lf",hesapBilgi.soyad,hesapBilgi.ad,&hesapBilgi.bakiye);
131         hesapBilgi.hesapNo = hesapID;
132         fseek(fPtr,(hesapID-1)*sizeof(struct musteri),SEEK_SET);
133         fwrite(&hesapBilgi,sizeof(struct musteri),1,fPtr);
134     }
135 }
```

CASE STUDY: A TRANSACTION PROCESSING PROGRAM

```
137 void listele(FILE *fPtr)
138 {
139     struct musteri hesapBilgi = {0,"","",0.0};
140 
141     printf("%-10s%-16s%-11s%10s\n","HesapNo","Soyad","Ad","Bakiye");
142     while(!feof(fPtr))
143     {
144         fread(&hesapBilgi,sizeof(struct musteri),1,fPtr);
145         if(hesapBilgi.hesapNo!=0)
146         printf("%-10d%-16s%-11s%10.2f\n",hesapBilgi.hesapNo,
147             hesapBilgi.soyad,hesapBilgi.ad,hesapBilgi.bakiye);
148     }
149     fclose(fPtr);
150     getchar();
151 }
```

LAB WORKS

- Patient Following System
  - Define a struct included patient name, age, and a set of illness information.
  - Insert a number of patient.
  - Find any patient who has got some key data.
  - Delete a patient record.
  - Modify a patient record.