

# Lekcija 11 – Obrada datoteka

## Pregled

- 11.1 Uvod
- 11.2 Hijerarhija podataka
- 11.3 Datoteke i tokovi (Files and Streams)
- 11.4 Kreiranje sekvencijalne datoteke
- 11.5 Čitanje podataka iz sekvencijalne datoteke
- 11.6 Datoteke sa slučajanim pristupom (Random Access Files)
- 11.7 Kreiranje datoteke sa slučajnim pristupom
- 11.8 Upis podataka u datoteke sa slučajnim pristupom
- 11.9 Čitanje podataka iz datoteke sa slučajnim pristupom
- 11.10 Primjer: Transaction-Processing Program

# Ciljevi

- U ovoj lekciji:
  - Naučićete da krierate datoteke, upisujete u njih i čitatate podatke.
  - Upoznaćete se sa obradom sekvencijalnih datoteka.
  - Upoznaćete se sa obradom datoteka sa slučajnim pristupom.

## 11.1 Uvod

- Datoteke
  - Mogu se kreirati mijenjati i obrađivati u C programima
  - Koriste se permanentno čuvanje velikih količina podataka
    - Smještanje podataka u promjenljive i nizove je samo privremeno – po završetku programa podaci nestaju

## 11.2 Hijerarhija podataka

- Hijerarhija podataka:
  - Bit – najmanji objekat podataka (data item)
    - Vrijednosti 0 ili 1
  - Bajt (Byte) – 8 bitova
    - Koristi se za čuvanje karaktera
      - Decimalne cifre (digits), slova i specijalni simboli
  - Polje – grupa karaktera sa određenim značenjem
    - Primjer: vaše ime
  - Zapis (Record) – grupa povezanih polja
    - Reprezentovani pomoću `struct` (ili `class`)
    - Primjer: U evidenciji plata radnika (payroll system), zapis za radnika sadrži identifikacioni broj, ime, adresu, itd.

## 11.2 Hijerarhija podataka

- Hijerarhija podataka (nastavak):
  - Datoteka (File) – grupa povezanih zapisa
    - Primjer: Datoteka sa evidencijom radnika (payroll file)
  - Baza podataka (Database) – grupa povezanih datoteka

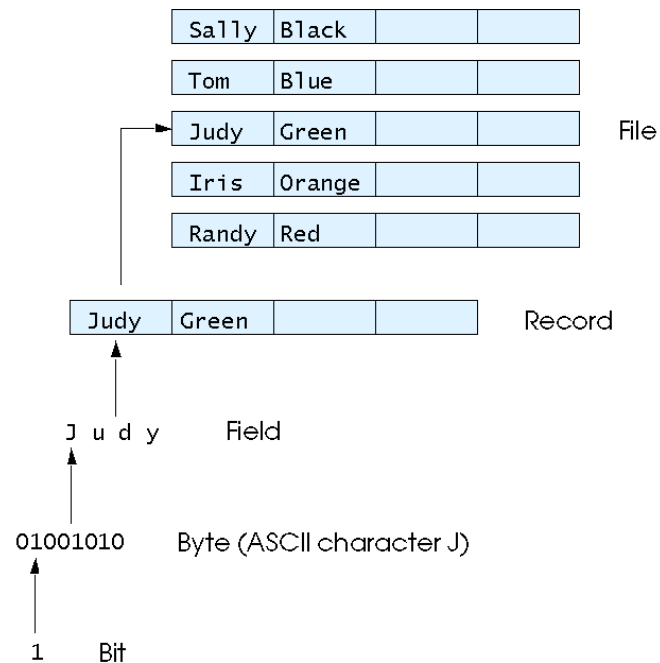


Fig. 11.1 The data hierarchy.

## 11.2 Hijerarhija podataka

- Datoteke
  - Ključ zapisa (Record key)
    - Identifikuje jednoznačno zapis da bi se lakše pronašao određeni zapis u datoteci (retrieval of specific records from a file)
  - Sekvencijalna datoteka
    - Zapisi su najčešće sortirani po ključu

## 11.3 Datoteke i tokovi (Files and Streams)

- C posmatra svaku datoteku kao niz bajtova
  - Kraj datoteke označava se sa *end-of-file (EOF) markerom*
    - Ili, datoteka završava na specifičnom bajtu
- Tok (stream) se kreira pri otvaranju datoteke
  - Tok obezbjeđuje komunikacioni kanal između datoteke i programa
  - Otvaranje datoteke vraća pokazivač na **FILE** strukturu
    - Primjer pokazivača na datoteku (file pointers):
    - `stdin` – standardni ulaz (tastatura-keyboard)
    - `stdout` – standardni izlaz (ekran - screen)
    - `stderr` – standardna greška (ekran - screen)

## 11.3 Datoteke i tokovi (Files and Streams)

- **FILE** struktura
  - Deskriptor datoteke (File descriptor)
    - Indeks u nizu koji čuva operativni sistem (tabela otvorenih datoteka - open file table)
  - File Control Block (FCB)
    - U svakom elementu čuva se kontrolni blok datoteke (FCB); sistem koristi FCB za administraciju datoteke



## 11.3 Datoteke i tokovi (Files and Streams)

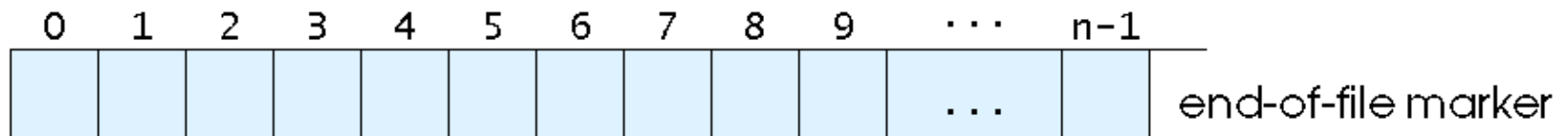


Fig. 11.2 C's view of a file of  $n$  bytes.

## 11.3 Datoteke i tokovi (Files and Streams)

- Read/Write funkcije u standardnoj biblioteci
  - `fgetc`
    - Čita jedan karakter iz datoteke
    - Ima argument tipa pokazivač na **FILE**
    - `fgetc( stdin )` je ekvivalentno sa `getchar()`
  - `fputc`
    - Upisuje jedan karakter u datoteku
    - Ima argument tipa pokazivač na **FILE** i karakter koji se upisuje kao argumente
    - `fputc( 'a', stdout )` ekvivalentno sa `putchar( 'a' )`
  - `fgets`
    - Učitava jedan red iz datoteke
  - `fputs`
    - Upisuje jedan red u datoteku
  - `fscanf` / `fprintf`
    - Rad sa datotekom ekvivalentan sa `scanf` i `printf`



```
1  /* Fig. 11.3: fig11_03.c
2      Create a sequential file */
3  #include <stdio.h>
4
5  int main()
6  {
7      int account;      /* account number */
8      char name[ 30 ]; /* account name */
9      double balance;   /* account balance */
10
11     FILE *cfPtr;       /* cfPtr = clients.dat file pointer */
12
13     /* fopen opens file. Exit program if unable to create file */
14     if ( ( cfPtr = fopen( "clients.dat", "w" ) ) == NULL ) {
15         printf( "File could not be opened\n" );
16     } /* end if */
17     else {
18         printf( "Enter the account, name, and balance.\n" );
19         printf( "Enter EOF to end input.\n" );
20         printf( "? " );
21         scanf( "%d%s%lf", &account, name, &balance );
22
```



```
23      /* write account, name and balance into file with fprintf */
24      while ( !feof( stdin ) ) {
25          fprintf( cfPtr, "%d %s %.2f\n", account, name, balance );
26          printf( "? " );
27          scanf( "%d%s%lf", &account, name, &balance );
28      } /* end while */
29
30      fclose( cfPtr ); /* fclose closes file */
31  } /* end else */
32
33      return 0; /* indicates successful termination */
34
35  } /* end main */
```

```
Enter the account, name, and balance.
Enter EOF to end input.
? 100 Jones 24.98
? 200 Doe 345.67
? 300 White 0.00
? 400 Stone -42.16
? 500 Rich 224.62
? ^Z
```

## Program Output

## 11.4 Kreiranje sekvencijalne datoteke

- C ne pretpostavlja ništa o strukturi sekvencijalne datoteke
  - Bez naznaka o zapisima u datoteci
  - Programer mora obezbijediti strukturu datoteke
- Kreiranje datoteke
  - `FILE *cfPtr;`
    - Kreira se FILE pointer cfPtr
  - `cfPtr = fopen("clients.dat", "w");`
    - Funkcija `fopen` vraća pokazivač na FILE za navedenu datoteku
    - Ima dva argumenta – datoteka koja se otvara i mod otvaranja datoteke (file open mode)
    - U slučaju neuspješnog otvaranja, vraća se NULL

## 11.4 Kreiranje sekvencijalne datoteke

Računarski sistem	Kombinacija tastera
UNIX sistemi	<i>&lt;return&gt; &lt;ctrl&gt; d</i>
IBM PC i kompatibilni sistemi	<i>&lt;ctrl&gt; z</i>
Macintosh	<i>&lt;ctrl&gt; d</i>
Fig. 11.4 End-of-file kombinacija tastera.	

## 11.4 Kreiranje sekvencijalne datoteke

- `fprintf`
  - Koristi se za “štampanje” u datoteku
  - Kao `printf`, osim što je prvi argument pokazivač na `FILE` (pokazivač na datoteku u koju se štampa)
- `feof( FILE pointer )`
  - Vraća true ako je end-of-file indikator dostignut u datoteci (nema više podataka za obradu)
- `fclose( FILE pointer )`
  - Zatvaranje datoteke
  - Pri završetku programa odrađuje se automatski
  - Dobra je praksa eksplicitnog zatvaranja datoteke
- Detalji
  - Programi mogu obrađivati više datoteka
  - Svaka datoteka mora imati jedinstveno ime i imati svoj pokazivač na `FILE`.

## 11.4 Kreiranje sekvencijalne datoteke

Mod	Opis
r	Otvaranje datoteke za čitanje.
w	Kreiranje datoteke za upis. Ako već postoji, odbacuje se prethodni sadržaj.
a	Nadovezivanje (append); otvaranje ili kreiranje za upis na kraj datoteke.
r+	Otvaranje datoteka za izmjene (čitanje ili upisivanje).
w+	Kreiranje datoteke za izmjene. Ako već postoji, odbacuje se prethodni sadržaj.
a+	Nadovezivanje (append); otvaranje ili kreiranje za izmjene; upisivanje se vrši na kraj datoteke.
rb	Otvaranje datoteke za čitanje u binarnom modu.
wb	Kreiranje datoteke za upisivanje u binarnom modu. Ako već postoji, odbacuje se prethodni sadržaj.
ab	Nadovezivanje (append); otvaranje ili kreiranje za upis na kraj datoteke u binarnom modu.
rb+	Otvaranje datoteka za izmjene (čitanje ili upisivanje) u binarnom modu.
wb+	Kreiranje datoteke za izmjene u binarnom modu. Ako već postoji, odbacuje se prethodni sadržaj.
ab+	Nadovezivanje (append) u binarnom modu; otvaranje ili kreiranje za izmjene; upisivanje se vrši na kraj datoteke.

Fig. 11.6 Načini otvaranja datoteka (File open modes).



## 11.5 Čitanje podataka iz sekvencijalne datoteke

- Čitanje iz sekvencijalne datoteke
  - Kreira se FILE pointer za datoteku iz koje se čita  
`cfPtr = fopen( "clients.dat", "r" );`
  - Koristi se `fscanf` za čitanje iz datoteke
    - Kao `scanf`, osim prvog argumenta koji je FILE pointer  
`fscanf( cfPtr, "%d%s%f", &account, name, &balance );`
  - Podaci se učitavaju od početka do kraja datoteke
  - Pokazivač pozicije u datoteci (file position pointer)
    - Daje broj sledećeg bajta koji treba da se učitava ili upiše
    - Nije pravi pokazivač već cio broj koji ukazuje na lokaciju bajta
    - Takođe se naziva "byte offset"
  - `rewind( cfPtr )`
    - Vraća pokazivač pozicije na početak datoteke (bajt 0)



```
1  /* Fig. 11.7: fig11_07.c
2      Reading and printing a sequential file */
3  #include <stdio.h>
4
5  int main()
6  {
7      int account;      /* account number */
8      char name[ 30 ]; /* account name */
9      double balance;   /* account balance */
10
11     FILE *cfPtr;       /* cfPtr = clients.dat file pointer */
12
13     /* fopen opens file; exits program if file cannot be opened */
14     if ( ( cfPtr = fopen( "clients.dat", "r" ) ) == NULL ) {
15         printf( "File could not be opened\n" );
16     } /* end if */
17     else { /* read account, name and balance from file */
18         printf( "%-10s%-13s\n", "Account", "Name", "Balance" );
19         fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
20
21         /* while not end of file */
22         while ( !feof( cfPtr ) ) {
23             printf( "%-10d%-13s%7.2f\n", account, name, balance );
24             fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
25         } /* end while */
26     }
```



## Outline



fig11\_07.c (2 of 2)

```
27      fclose( cfPtr ); /* fclose closes the file */
28  } /* end else */
29
30  return 0; /* indicates successful termination */
31
32 } /* end main */
```

Account	Name	Balance
100	Jones	24.98
200	Doe	345.67
300	White	0.00
400	Stone	-42.16
500	Rich	224.62



```
1  /* Fig. 11.8: fig11_08.c
2     Credit inquiry program */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main()
7  {
8      int request;    /* request number */
9      int account;    /* account number */
10     double balance; /* account balance */
11     char name[ 30 ]; /* account name */
12     FILE *cfPtr;    /* clients.dat file pointer */
13
14     /* fopen opens the file; exits program if file cannot be opened */
15     if ( ( cfPtr = fopen( "clients.dat", "r" ) ) == NULL ) {
16         printf( "File could not be opened\n" );
17     } /* end if */
18     else {
19
20         /* display request options */
21         printf( "Enter request\n"
22             " 1 - List accounts with zero balances\n"
23             " 2 - List accounts with credit balances\n"
24             " 3 - List accounts with debit balances\n"
25             " 4 - End of run\n? " );
```



```
26 scanf( "%d", &request );
27
28 /* process user's request */
29 while ( request != 4 ) {
30
31     /* read account, name and balance from file */
32     fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
33
34     switch ( request ) {
35
36         case 1:
37             printf( "\nAccounts with zero balances:\n" );
38
39             /* read file contents (until eof) */
40             while ( !feof( cfPtr ) ) {
41
42                 if ( balance == 0 ) {
43                     printf( "%-10d%-13s%7.2f\n",
44                             account, name, balance );
45                 } /* end if */
46
47                 /* read account, name and balance from file */
48                 fscanf( cfPtr, "%d%s%lf",
49                         &account, name, &balance );
50             } /* end while */
51
```



```
52     break;
53
54 case 2:
55     printf( "\nAccounts with credit balances:\n" );
56
57     /* read file contents (until eof) */
58     while ( !feof( cfPtr ) ) {
59
60         if ( balance < 0 ) {
61             printf( "%-10d%-13s%7.2f\n",
62                 account, name, balance );
63         } /* end if */
64
65         /* read account, name and balance from file */
66         fscanf( cfPtr, "%d%s%lf",
67             &account, name, &balance );
68     } /* end while */
69
70     break;
71
72 case 3:
73     printf( "\nAccounts with debit balances:\n" );
74
```



```
75      /* read file contents (until eof) */
76      while ( !feof( cfPtr ) ) {
77
78          if ( balance > 0 ) {
79              printf( "%-10d%-13s%7.2f\n",
80                  account, name, balance );
81          } /* end if */
82
83          /* read account, name and balance from file */
84          fscanf( cfPtr, "%d%s%lf",
85              &account, name, &balance );
86      } /* end while */
87
88      break;
89
90  } /* end switch */
91
92  rewind( cfPtr ); /* return cfPtr to beginning of file */
93
94  printf( "\n? " );
95  scanf( "%d", &request );
96  } /* end while */
97
```



```
98     printf( "End of run.\n" );
99     fclose( cfPtr ); /* fclose closes the file */
100 } /* end else */
101
102 return 0; /* indicates successful termination */
103
104 } /* end main */
```

## Program Output

```
Enter request
 1 - List accounts with zero balances
 2 - List accounts with credit balances
 3 - List accounts with debit balances
 4 - End of run
? 1

Accounts with zero balances:
300      White      0.00

? 2

Accounts with credit balances:
400      Stone     -42.16

? 3

Accounts with debit balances:
100      Jones      24.98
200      Doe        345.67
500      Rich       224.62

? 4
End of run.
```

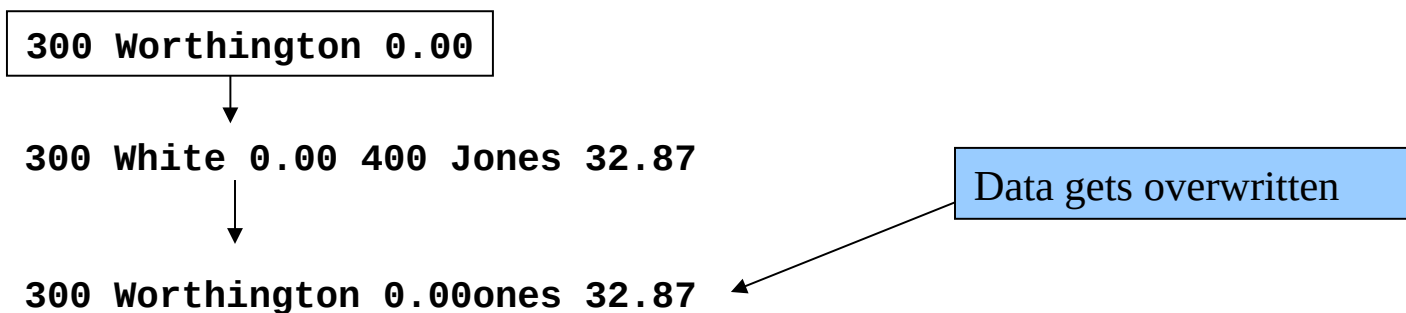


## 11.5 Čitanje podataka iz sekvencijalne datoteke

- Sekvencijalne datoteke
  - Ne mogu biti modifikovane bez rizika za uništenje drugih podataka
  - Polja mogu varirati u veličini
    - Različite reprezentacije za datoteku i ekran od unutrašnje reprezentcije (1, 34, -890 su tipa `int`, ali različitih veličina na disku)

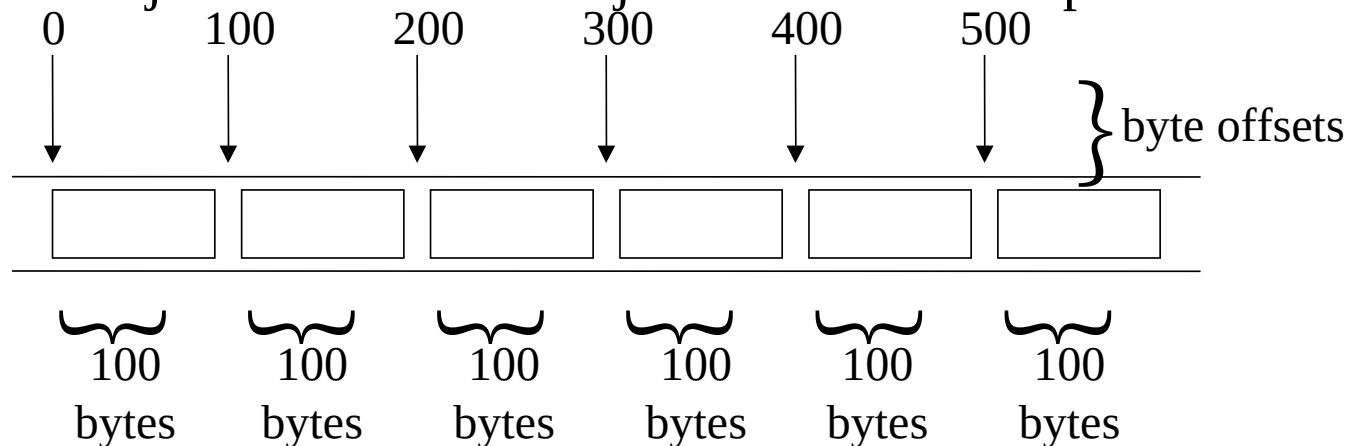
300 White 0.00 400 Jones 32.87 (old data in file)

Ako želimo da promijenimo ime White u ime Worthington,



## 11.6 Datoteke sa slučajnim pristupom

- Datoteke sa slučajnim pristupom (random access files)
  - Pristup pojedinačnom zapisu bez pregledanja ostalih zapisa
  - Instantni pristup zapisima datoteke
  - Podaci se mogu umetati bez uništavanja postojećih podataka
  - Podaci koji su ranije smješteni u datoteku mogu se mijenjati ili brisati bez prepisivanja (overwriting)
- Implementiraju se pomoću zapisa fiksne dužine
  - Sekvencijalne datoteke nemaju fiksnu dužinu zapisa



## 11.7 Kreiranje datoteka sa slučajnim pristupom

- Podaci u datotekama sa slučajnim pristupom
  - Neformatirani (čuvaju se kao sirovi bajtovi - "raw bytes")
    - Svi podaci istog tipa (na primjer **int**) koriste istu količinu memorije
    - Svi zapisi imaju isti tip fiksne dužine
    - Podaci nisu čitljivi čovjeku

## 11.7 Kreiranje datoteka sa slučajnim pristupom

- Neformatirane I/O funkcije
  - `fwrite`
    - Transfer bajtova iz bafera u memoriji u datoteku
  - `fread`
    - Transfer bajtova iz datoteke u bafer u memorij
  - Primjer:

```
fwrite( &number, sizeof( int ), 1, myPtr );
```

    - `&number` – Bafer (lokacija) iz kojeg se prenose podaci
    - `sizeof( int )` – Broj bajtova za transfer
    - `1` – Za niyove, broj elemenata za transfer
      - U ovom slučaju, samo jedan element
    - `myPtr` – Datoteka u koju se upisuje ili se iz nje čita

## 11.7 Kreiranje datoteka sa slučajnim pristupom

- Upisivanje struktura (`struct`)  
`fwrite( &myObject, sizeof (struct myStruct), 1,  
myPtr );`
  - `sizeof` – vraća veličinu argumenta (objekta) u bajtovima
- Upisivanje više elemenata niza
  - Pokazivač na niz kao prvi argument
  - Broj elemenata ako treći argument



```
1  /* Fig. 11.11: fig11_11.c
2      Creating a randomly accessed file sequentially */
3  #include <stdio.h>
4
5  /* clientData structure definition */
6  struct clientData {
7      int acctNum;          /* account number */
8      char lastName[ 15 ]; /* account last name */
9      char firstName[ 10 ]; /* account first name */
10     double balance;       /* account balance */
11 }; /* end structure clientData */
12
13 int main()
14 {
15     int i; /* counter */
16
17     /* create clientData with no information */
18     struct clientData blankClient = { 0, "", "", 0.0 };
19
20     FILE *cfPtr; /* credit.dat file pointer */
21
22     /* fopen opens the file; exits if file cannot be opened */
23     if ( ( cfPtr = fopen( "credit.dat", "wb" ) ) == NULL ) {
24         printf( "File could not be opened.\n" );
25     } /* end if */
```



```
26 else {
27
28     /* output 100 blank records to file */
29     for ( i = 1; i <= 100; i++ ) {
30         fwrite( &blankClient, sizeof( struct clientData ), 1, cfPtr );
31     } /* end for */
32
33     fclose ( cfPtr ); /* fclose closes the file */
34 } /* end else */
35
36 return 0; /* indicates successful termination */
37
38 } /* end main */
```

## 11.8 Upisivanje podataka u datoteku sa slučajnim upisom

- `fseek`
  - Postavlja pokazivač pozicije na određenu poziciju u datoteci
  - `fseek( pointer, offset, symbolic_constant ) ;`
    - *pointer* – pokazivač na datoteku
    - *offset* – pokazivač pozicije (0 na prvoj lokaciji)
    - *symbolic\_constant* – specificira sa kog mjesta treba početi
    - `SEEK_SET` – počinje se od početka datoteke
    - `SEEK_CUR` – počinje se od trenutne pozicije u datoteci
    - `SEEK_END` – počinje se od kraja datoteke





```
1  /* Fig. 11.12: fig11_12.c
2     Writing to a random access file */
3  #include <stdio.h>
4
5  /* clientData structure definition */
6  struct clientData {
7      int acctNum;          /* account number */
8      char lastName[ 15 ]; /* account last name */
9      char firstName[ 10 ]; /* account first name */
10     double balance;       /* account balance */
11 }; /* end structure clientData */
12
13 int main()
14 {
15     FILE *cfPtr; /* credit.dat file pointer */
16
17     /* create clientData with no information */
18     struct clientData client = { 0, "", "", 0.0 };
19
20     /* fopen opens the file; exits if file cannot be opened */
21     if ( ( cfPtr = fopen( "credit.dat", "rb+" ) ) == NULL ) {
22         printf( "File could not be opened.\n" );
23     } /* end if */
24     else {
25
```



```
26  /* require user to specify account number */
27  printf( "Enter account number"
28          " ( 1 to 100, 0 to end input )\n? " );
29  scanf( "%d", &client.acctNum );
30
31  /* user enters information, which is copied into file */
32  while ( client.acctNum != 0 ) {
33
34      /* user enters last name, first name and balance */
35      printf( "Enter lastname, firstname, balance\n? " );
36
37      /* set record lastName, firstName and balance value */
38      fscanf( stdin, "%s%s%lf", client.lastName,
39              client.firstName, &client.balance );
40
41      /* seek position in file of user-specified record */
42      fseek( cfPtr, ( client.acctNum - 1 ) *
43              sizeof( struct clientData ), SEEK_SET );
44
45      /* write user-specified information in file */
46      fwrite( &client, sizeof( struct clientData ), 1, cfPtr );
47
48      /* enable user to specify another account number */
49      printf( "Enter account number\n? " );
50      scanf( "%d", &client.acctNum );
```



```
51     } /* end while */
52
53     fclose( cfPtr ); /* fclose closes the file */
54 } /* end else */
55
56 return 0; /* indicates successful termination */
57
58 } /* end main */
```

```
Enter account number ( 1 to 100, 0 to end input )
? 37
Enter lastname, firstname, balance
? Barker Doug 0.00
Enter account number
? 29
Enter lastname, firstname, balance
? Brown Nancy -24.54
Enter account number
? 96
Enter lastname, firstname, balance
? Stone Sam 34.98
Enter account number
? 88
Enter lastname, firstname, balance
? Smith Dave 258.34
Enter account number
? 33
Enter lastname, firstname, balance
? Dunn Stacey 314.33
Enter account number
? 0
```

## Program Output

## 11.8 Upisivanje podataka u datoteku sa slučajnim upisom

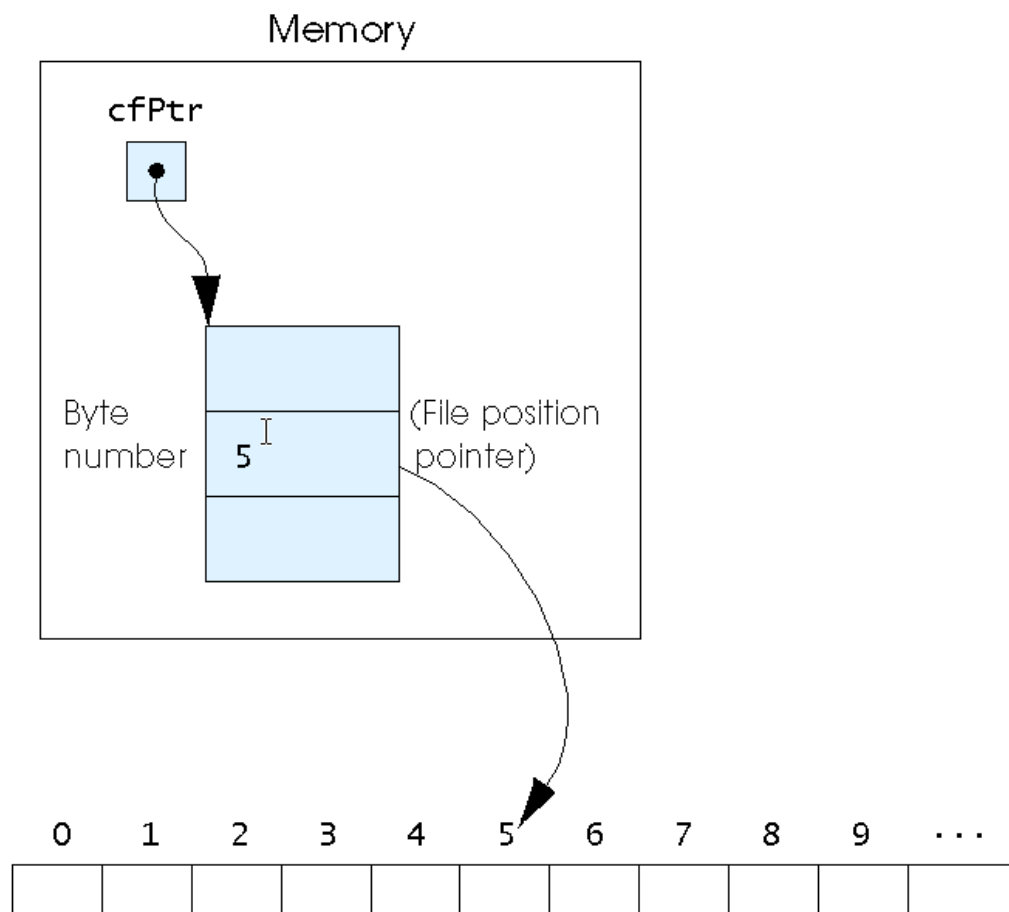


Fig. 11.14 The file position pointer indicating an offset of 5 bytes from the beginning of the file.

## 11.9 Čitanje podataka iz datoteke sa slučajnim pristupom

- `fread`
  - Učitava zadati broj bajtova iz datoteke u bafer u memoriji  
`fread( &client, sizeof (struct clientData), 1, myPtr );`
  - Može čitati više elemenata niza
    - Obezbijediti pokazivač na niz
    - Postaviti broj elemenata koje treba pročitati
  - Za čitanje više elemenata, zadati treći argument



```
1  /* Fig. 11.15: fig11_15.c
2     Reading a random access file sequentially */
3  #include <stdio.h>
4
5  /* clientData structure definition */
6  struct clientData {
7     int acctNum;          /* account number */
8     char lastName[ 15 ]; /* account last name */
9     char firstName[ 10 ]; /* account first name */
10    double balance;        /* account balance */
11 }; /* end structure clientData */
12
13 int main()
14 {
15     FILE *cfPtr; /* credit.dat file pointer */
16
17     /* create clientData with no information */
18     struct clientData client = { 0, "", "", 0.0 };
19
20     /* fopen opens the file; exits if file cannot be opened */
21     if ( ( cfPtr = fopen( "credit.dat", "rb" ) ) == NULL ) {
22         printf( "File could not be opened.\n" );
23     } /* end if */
```



```
24 else {
25     printf( "%-6s%-16s%-11s%10s\n", "Acct", "Last Name",
26         "First Name", "Balance" );
27
28     /* read all records from file (until eof) */
29     while ( !feof( cfPtr ) ) {
30         fread( &client, sizeof( struct clientData ), 1, cfPtr );
31
32         /* display record */
33         if ( client.acctNum != 0 ) {
34             printf( "%-6d%-16s%-11s%10.2f\n",
35                 client.acctNum, client.lastName,
36                 client.firstName, client.balance );
37         } /* end if */
38
39     } /* end while */
40
41     fclose( cfPtr ); /* fclose closes the file */
42 } /* end else */
43
44 return 0; /* indicates successful termination */
45
46 } /* end main */
```

[Outline](#)**Program Output**

Acct	Last Name	First Name	Balance
29	Brown	Nancy	-24.54
33	Dunn	Stacey	314.33
37	Barker	Doug	0.00
88	Smith	Dave	258.34
96	Stone	Sam	34.98



## 11.10 Primjer: A Transaction Processing Program

- Program za obradu bankarskih transakcija
  - Demonstracija datoteke sa slučajnim pristupom za dostup informaciji o bankarskim računima
- Program može da
  - Mijenja iznose na postojećim računima
  - Dodaje nove račune
  - Briše račune
  - Čuva formatiranu listu svih računa u tekstualnoj datoteci



```
1  /* Fig. 11.16: fig11_16.c
2      This program reads a random access file sequentially, updates data
3      already written to the file, creates new data to be placed in the
4      file, and deletes data previously in the file. */
5  #include <stdio.h>
6
7  /* clientData structure definition */
8  struct clientData {
9      int acctNum;          /* account number */
10     char lastName[ 15 ]; /* account last name */
11     char firstName[ 10 ]; /* account first name */
12     double balance;        /* account balance */
13 }; /* end structure clientData */
14
15 /* prototypes */
16 int enterChoice( void );
17 void textFile( FILE *readPtr );
18 void updateRecord( FILE *fPtr );
19 void newRecord( FILE *fPtr );
20 void deleteRecord( FILE *fPtr );
21
22 int main()
23 {
24     FILE *cfPtr; /* credit.dat file pointer */
25     int choice; /* user's choice */
26
```



```
27  /* fopen opens the file; exits if file cannot be opened */
28  if ( ( cfPtr = fopen( "credit.dat", "rb+" ) ) == NULL ) {
29      printf( "File could not be opened.\n" );
30  } /* end if */
31  else {
32
33      /* enable user to specify action */
34      while ( ( choice = enterChoice() ) != 5 ) {
35
36          switch ( choice ) {
37
38              /* create text file from record file */
39              case 1:
40                  textFile( cfPtr );
41                  break;
42
43              /* update record */
44              case 2:
45                  updateRecord( cfPtr );
46                  break;
47
```



```
48      /* create record */
49      case 3:
50          newRecord( cfPtr );
51          break;
52
53      /* delete existing record */
54      case 4:
55          deleteRecord( cfPtr );
56          break;
57
58      /* display message if user does not select valid choice */
59      default:
60          printf( "Incorrect choice\n" );
61          break;
62
63      } /* end switch */
64
65      } /* end while */
66
67      fclose( cfPtr ); /* fclose closes the file */
68  } /* end else */
69
70      return 0; /* indicates successful termination */
71
72  } /* end main */
73
```



```
74 /* create formatted text file for printing */
75 void textFile( FILE *readPtr )
76 {
77     FILE *writePtr; /* accounts.txt file pointer */
78
79     /* create clientData with no information */
80     struct clientData client = { 0, "", "", 0.0 };
81
82     /* fopen opens the file; exits if file cannot be opened */
83     if ( ( writePtr = fopen( "accounts.txt", "w" ) ) == NULL ) {
84         printf( "File could not be opened.\n" );
85     } /* end if */
86     else {
87         rewind( readPtr ); /* sets pointer to beginning of record file */
88         fprintf( writePtr, "%-6s%-16s%-11s%10s\n",
89                 "Acct", "Last Name", "First Name", "Balance" );
90
91         /* copy all records from record file into text file */
92         while ( !feof( readPtr ) ) {
93             fread( &client, sizeof( struct clientData ), 1, readPtr );
94         }
```



```
95      /* write single record to text file */
96      if ( client.acctNum != 0 ) {
97          fprintf( writePtr, "%-6d%-16s%-11s%10.2f\n",
98                  client.acctNum, client.lastName,
99                  client.firstName, client.balance );
100      } /* end if */
101
102  } /* end while */
103
104      fclose( writePtr ); /* fclose closes the file */
105  } /* end else */
106
107 } /* end function textFile */
108
109 /* update balance in record */
110 void updateRecord( FILE *fPtr )
111 {
112     int account;          /* account number */
113     double transaction; /* account transaction */
114
115     /* create clientData with no information */
116     struct clientData client = { 0, "", "", 0.0 };
117
```



```
118  /* obtain number of account to update */
119  printf( "Enter account to update ( 1 - 100 ): " );
120  scanf( "%d", &account );
121
122  /* move file pointer to correct record in file */
123  fseek( fPtr, ( account - 1 ) * sizeof( struct clientData ),
124         SEEK_SET );
125
126  /* read record from file */
127  fread( &client, sizeof( struct clientData ), 1, fPtr );
128
129  /* display error if account does not exist */
130  if ( client.acctNum == 0 ) {
131      printf( "Account #%d has no information.\n", account );
132  } /* end if */
133  else { /* update record */
134      printf( "%-6d%-16s%-11s%10.2f\n\n",
135             client.acctNum, client.lastName,
136             client.firstName, client.balance );
137
138      /* request user to specify transaction */
139      printf( "Enter charge ( + ) or payment ( - ): " );
140      scanf( "%lf", &transaction );
141      client.balance += transaction; /* update record balance */
142  }
```



```
143     printf( "%-6d%-16s%-11s%10.2f\n",
144             client.acctNum, client.lastName,
145             client.firstName, client.balance );
146
147     /* move file pointer to correct record in file */
148     fseek( fPtr, ( account - 1 ) * sizeof( struct clientData ),
149           SEEK_SET );
150
151     /* write updated record over old record in file */
152     fwrite( &client, sizeof( struct clientData ), 1, fPtr );
153 } /* end else */
154
155 } /* end function updateRecord */
156
157 /* delete an existing record */
158 void deleteRecord( FILE *fPtr )
159 {
160     /* create two clientDatas and initialize blankClient */
161     struct clientData client;
162     struct clientData blankClient = { 0, "", "", 0 };
163
164     int accountNum; /* account number */
165
```





```
166  /* obtain number of account to delete */
167  printf( "Enter account number to delete ( 1 - 100 ): " );
168  scanf( "%d", &accountNum );
169
170  /* move file pointer to correct record in file */
171  fseek( fPtr, ( accountNum - 1 ) * sizeof( struct clientData ),
172         SEEK_SET );
173
174  /* read record from file */
175  fread( &client, sizeof( struct clientData ), 1, fPtr );
176
177  /* display error if record does not exist */
178  if ( client.acctNum == 0 ) {
179      printf( "Account %d does not exist.\n", accountNum );
180  } /* end if */
181  else { /* delete record */
182
183      /* move file pointer to correct record in file */
184      fseek( fPtr, ( accountNum - 1 ) * sizeof( struct clientData ),
185             SEEK_SET );
186
187      /* replace existing record with blank record */
188      fwrite( &blankClient,
189             sizeof( struct clientData ), 1, fPtr );
190  } /* end else */
191
```



```
192 } /* end function deleteRecord */
193
194 /* create and insert record */
195 void newRecord( FILE *fPtr )
196 {
197     /* create clientData with no information */
198     struct clientData client = { 0, "", "", 0.0 };
199
200     int accountNum; /* account number */
201
202     /* obtain number of account to create */
203     printf( "Enter new account number ( 1 - 100 ): " );
204     scanf( "%d", &accountNum );
205
206     /* move file pointer to correct record in file */
207     fseek( fPtr, ( accountNum - 1 ) * sizeof( struct clientData ),
208           SEEK_SET );
209
210     /* read record from file */
211     fread( &client, sizeof( struct clientData ), 1, fPtr );
212
```



```
213  /* display error if account previously exists */
214  if ( client.acctNum != 0 ) {
215      printf( "Account #%d already contains information.\n",
216             client.acctNum );
217  } /* end if */
218  else { /* create record */
219
220      /* user enters last name, first name and balance */
221      printf( "Enter lastname, firstname, balance\n? " );
222      scanf( "%s%s%lf", &client.lastName, &client.firstName,
223            &client.balance );
224
225      client.acctNum = accountNum;
226
227      /* move file pointer to correct record in file */
228      fseek( fPtr, ( client.acctNum - 1 ) *
229            sizeof( struct clientData ), SEEK_SET );
230
231      /* insert record in file */
232      fwrite( &client,
233            sizeof( struct clientData ), 1, fPtr );
234  } /* end else */
235
236 } /* end function newRecord */
237
```



```
238 /* enable user to input menu choice */
239 int enterChoice( void )
240 {
241     int menuChoice; /* variable to store user's choice */
242
243     /* display available options */
244     printf( "\nEnter your choice\n"
245            "1 - store a formatted text file of accounts called\n"
246            "    \"accounts.txt\" for printing\n"
247            "2 - update an account\n"
248            "3 - add a new account\n"
249            "4 - delete an account\n"
250            "5 - end program\n? " );
251
252     scanf( "%d", &menuChoice ); /* receive choice from user */
253
254     return menuChoice;
255
256 } /* end function enterChoice */
```



After choosing option 1 accounts.txt contains:

Acct	Last Name	First Name	Balance
29	Brown	Nancy	-24.54
33	Dunn	Stacey	314.33
37	Barker	Doug	0.00
88	Smith	Dave	258.34
96	Stone	Sam	34.98

After choosing option 2 accounts.txt contains:

Enter account to update ( 1 - 100 ): 37

37	Barker	Doug	0.00
----	--------	------	------

Enter charge ( + ) or payment ( - ): +87.99

37	Barker	Doug	87.99
----	--------	------	-------

After choosing option 3 accounts.txt contains:

Enter new account number ( 1 - 100 ): 22

Enter lastname, firstname, balance

? Johnston Sarah 247.45