Lekcija 11 – Obrada datoteka

Pregled

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 kreiratim 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

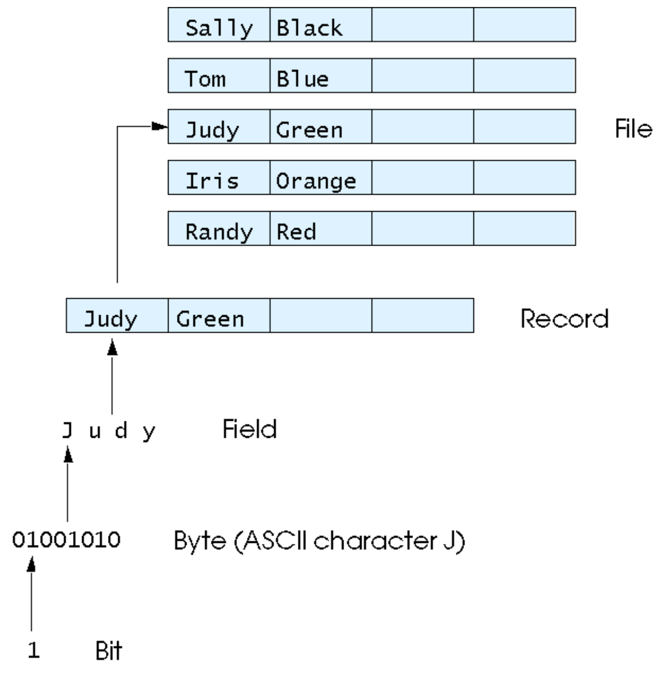


Fig. 11.1 The data hierarchy.

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 (tastaura-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)

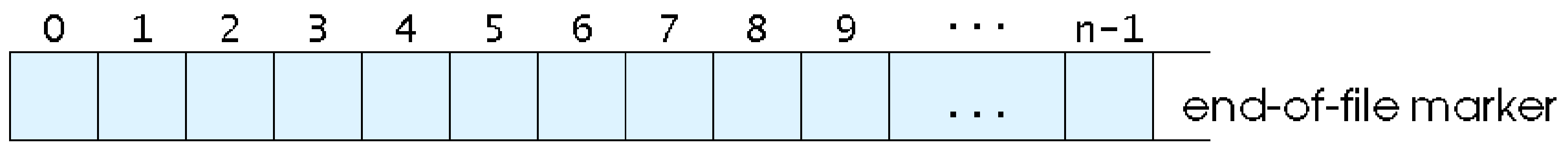


Fig. 11.2 C's view of a file of *n* bytes.

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 ekvivavlentan sa `scanf` i `printf`

Outline

fig11_03.c (1 of 2)

```c
/* Fig. 11.3: fig11_03.c
   Create a sequential file */
#include <stdio.h>

int main()
{
   int account;     /* account number */
   char name[ 30 ]; /* account name */
   double balance;  /* account balance */

    FILE *cfPtr;     /* cfPtr = clients.dat file pointer */

    /* fopen opens file. Exit program if unable to create file  */
    if ( ( cfPtr = fopen( "clients.dat", "w" ) ) == NULL ) {
       printf( "File could not be opened\n" );
    } /* end if */
    else {
       printf( "Enter the account, name, and balance.\n" );
       printf( "Enter EOF to end input.\n" );
       printf( "? " );
       scanf( "%d%s%lf", &account, name, &balance );

```

fig11_03.c (2 of 2)

```
      /* write account, name and balance into file with fprintf */
      while ( !feof( stdin ) ) {
         fprintf( cfPtr, "%d %s %.2f\n", account, name, balance );
         printf( "? " );
         scanf( "%d%s%lf", &account, name, &balance );
      } /* end while */

      fclose( cfPtr ); /* fclose closes file */
   } /* end else */

   return 0; /* indicates successful termination */

} /* end main */
```

Program Output

```
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
```

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	*<return> <ctrl> d*
IBM PC i kompatibilni sistemi	*<ctrl> z*
Macintosh	*<ctrl> d*

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 odrađuej 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", &accounnt, 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 učita ili upiše
    - Nije pravi pokazivač več cio broj koji ukazuje na lokaciju bajta
    - Takođe se naziva “byte offset”
  - `rewind( cfPtr )`
    - Vraća pokazivač pozicije na početak datoteke (bajt `0`)

Outline

fig11_07.c (1 of 2)

```
/* Fig. 11.7: fig11_07.c
   Reading and printing a sequential file */
#include <stdio.h>

int main()
{
   int account;     /* account number */
   char name[ 30 ]; /* account name */
   double balance;  /* account balance */

   FILE *cfPtr;     /* cfPtr = clients.dat file pointer */

   /* fopen opens file; exits program if file cannot be opened */
   if ( ( cfPtr = fopen( "clients.dat", "r" ) ) == NULL ) {
      printf( "File could not be opened\n" );
   } /* end if */
   else { /* read account, name and balance from file */
      printf( "%-10s%-13s%s\n", "Account", "Name", "Balance" );
      fscanf( cfPtr, "%d%s%lf", &account, name, &balance );

      /* while not end of file */
      while ( !feof( cfPtr ) ) {
         printf( "%-10d%-13s%7.2f\n", account, name, balance );
         fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
      } /* end while */

```

Outline

fig11_07.c (2 of 2)

```c
         fclose( cfPtr ); /* fclose closes the file */
      } /* end else */

      return 0; /* indicates successful termination */

} /* 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
```

Outline

fig11_08.c (1 of 5)

```
/* Fig. 11.8: fig11_08.c
   Credit inquiry program */
#include <stdio.h>

/* function main begins program execution */
int main()
{
   int request;     /* request number */
   int account;     /* account number */
   double balance;  /* account balance */
   char name[ 30 ]; /* account name */
   FILE *cfPtr;     /* clients.dat file pointer */

   /* fopen opens the file; exits program if file cannot be opened */
   if ( ( cfPtr = fopen( "clients.dat", "r" ) ) == NULL ) {
      printf( "File could not be opened\n" );
   } /* end if */
   else {

      /* display request options */
      printf( "Enter request\n"
             " 1 - List accounts with zero balances\n"
             " 2 - List accounts with credit balances\n"
             " 3 - List accounts with debit balances\n"
             " 4 - End of run\n? " );
```

Outline

fig11_08.c (2 of 5)

```c
      scanf( "%d", &request );

      /* process user's request */
      while ( request != 4 ) {

         /* read account, name and balance from file */
         fscanf( cfPtr, "%d%s%lf", &account, name, &balance );

         switch ( request ) {

            case 1:
               printf( "\nAccounts with zero balances:\n" );

               /* read file contents (until eof) */
               while ( !feof( cfPtr ) ) {

                  if ( balance == 0 ) {
                     printf( "%-10d%-13s%7.2f\n",
                             account, name, balance );
                  } /* end if */

                  /* read account, name and balance from file */
                  fscanf( cfPtr, "%d%s%lf",
                          &account, name, &balance );
               } /* end while */

```

Outline

fig11_08.c (3 of 5)

```
            break;

         case 2:
            printf( "\nAccounts with credit balances:\n" );

            /* read file contents (until eof) */
            while ( !feof( cfPtr ) ) {

               if ( balance < 0 ) {
                  printf( "%-10d%-13s%7.2f\n",
                          account, name, balance );
               } /* end if */

               /* read account, name and balance from file */
               fscanf( cfPtr, "%d%s%lf",
                       &account, name, &balance );
            } /* end while */

            break;

         case 3:
            printf( "\nAccounts with debit balances:\n" );

```

Outline

fig11_08.c (4 of 5)

```
            /* read file contents (until eof) */
            while ( !feof( cfPtr ) ) {

               if ( balance > 0 ) {
                  printf( "%-10d%-13s%7.2f\n",
                          account, name, balance );
               } /* end if */

               /* read account, name and balance from file */
               fscanf( cfPtr, "%d%s%lf",
                       &account, name, &balance );
            } /* end while */

            break;

      } /* end switch */

      rewind( cfPtr ); /* return cfPtr to beginning of file */

      printf( "\n? " );
      scanf( "%d", &request );
   } /* end while */

```

Outline

fig11_08.c (5 of 5)

```
         printf( "End of run.\n" );
         fclose( cfPtr ); /* fclose closes the file */
      } /* end else */

      return 0; /* indicates successful termination */

   } /* 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 drugh 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)
```

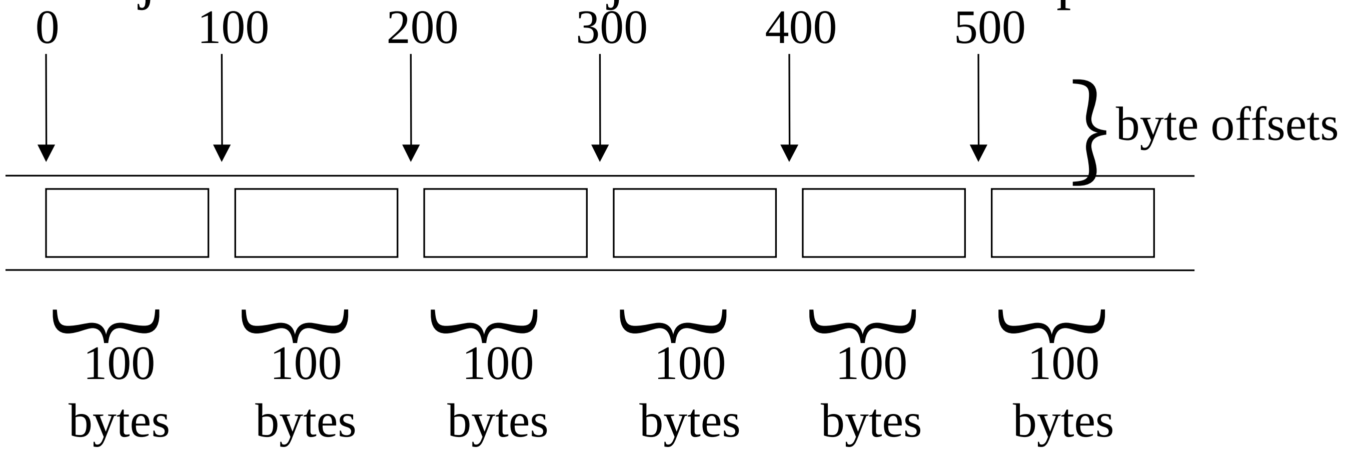
Ako želimo da promijenimo ime White u ime Worthington,

```
300 Worthington 0.00

300 White 0.00 400 Jones 32.87

300 Worthington 0.00ones 32.87
```

Data gets overwritten

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`**) korsite 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

Outline

fig11_11.c (1 of 2)

```c
/* Fig. 11.11: fig11_11.c
   Creating a randomly accessed file sequentially */
#include <stdio.h>

/* clientData structure definition */
struct clientData {
   int acctNum;          /* account number */
   char lastName[ 15 ];  /* account last name */
   char firstName[ 10 ]; /* account first name */
    double balance;       /* account balance */
 }; /* end structure clientData */

 int main()
 {
    int i; /* counter */

    /* create clientData with no information */
    struct clientData blankClient = { 0, "", "", 0.0 };

    FILE *cfPtr; /* credit.dat file pointer */

    /* fopen opens the file; exits if file cannot be opened */
    if ( ( cfPtr = fopen( "credit.dat", "wb" ) ) == NULL ) {
       printf( "File could not be opened.\n" );
    } /* end if */
```

Outline

fig11_11.c (2 of 2)

```
   else {

      /* output 100 blank records to file */
      for ( i = 1; i <= 100; i++ ) {
         fwrite( &blankClient, sizeof( struct clientData ), 1, cfPtr );
      } /* end for */

      fclose ( cfPtr ); /* fclose closes the file */
   } /* end else */

   return 0; /* indicates successful termination */

} /* 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

Outline

fig11_12.c (1 of 3)

```
/* Fig. 11.12: fig11_12.c
   Writing to a random access file */
#include <stdio.h>

/* clientData structure definition */
struct clientData {
   int acctNum;           /* account number */
   char lastName[ 15 ];   /* account last name */
   char firstName[ 10 ]; /* account first name */
    double balance;        /* account balance */
 }; /* end structure clientData */

 int main()
 {
    FILE *cfPtr; /* credit.dat file pointer */

    /* create clientData with no information */
    struct clientData client = { 0, "", "", 0.0 };

    /* fopen opens the file; exits if file cannot be opened */
    if ( ( cfPtr = fopen( "credit.dat", "rb+" ) ) == NULL ) {
       printf( "File could not be opened.\n" );
    } /* end if */
    else {

```

Outline

fig11_12.c (2 of 3)

```
   /* require user to specify account number */
   printf( "Enter account number"
           " ( 1 to 100, 0 to end input )\n? " );
   scanf( "%d", &client.acctNum );

   /* user enters information, which is copied into file */
   while ( client.acctNum != 0 ) {

      /* user enters last name, first name and balance */
      printf( "Enter lastname, firstname, balance\n? " );

      /* set record lastName, firstName and balance value */
      fscanf( stdin, "%s%s%lf", client.lastName,
              client.firstName, &client.balance );

      /* seek position in file of user-specified record */
      fseek( cfPtr, ( client.acctNum - 1 ) *
             sizeof( struct clientData ), SEEK_SET );

      /* write user-specified information in file */
      fwrite( &client, sizeof( struct clientData ), 1, cfPtr );

      /* enable user to specify another account number */
      printf( "Enter account number\n? " );
      scanf( "%d", &client.acctNum );
```

Outline

fig11_12.c (3 of 3)

```
      } /* end while */

      fclose( cfPtr ); /* fclose closes the file */
   } /* end else */

   return 0; /* indicates successful termination */

} /* end main */
```

Program Output

```
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
```

11.8 Upisivanje podataka u datoteku sa slučajnim upisom

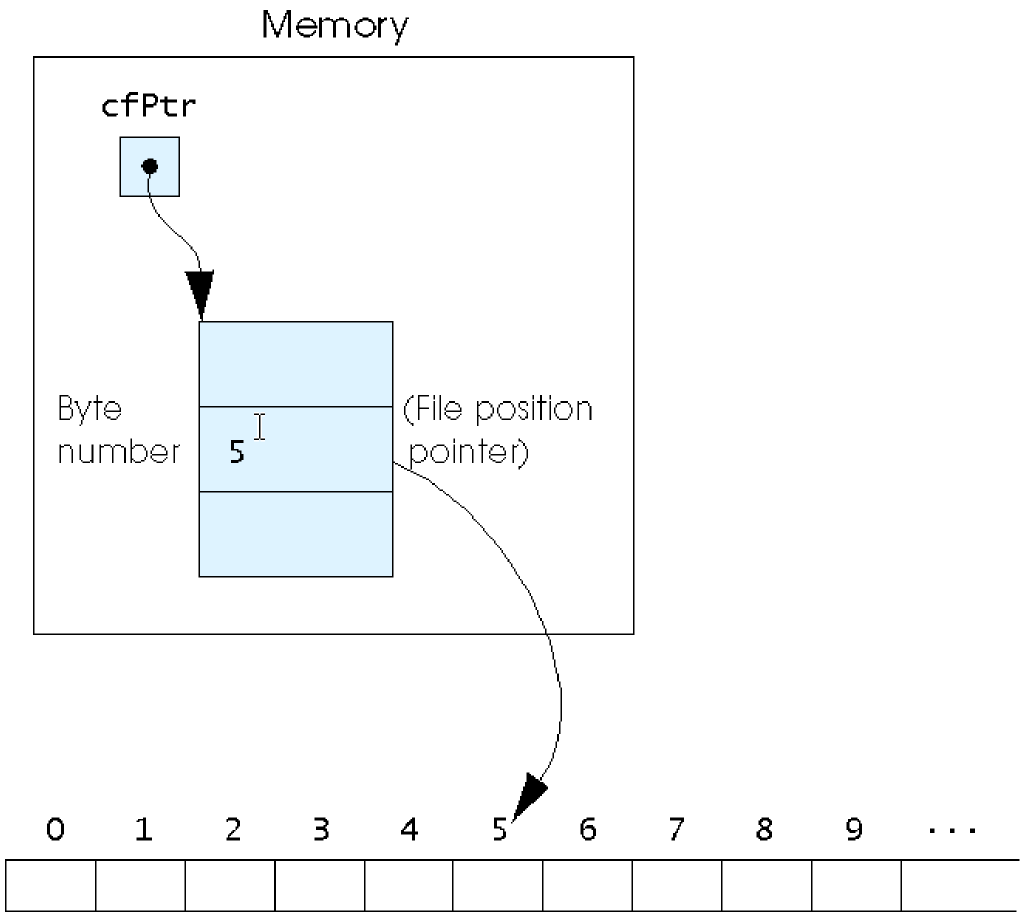


Fig. 11.14 The file position pointer indicating an offset of 5 bytes from the beginning of the file.

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 elemanata, zadati treći argument

Outline

fig11_15.c (1 of 2)

```
/* Fig. 11.15: fig11_15.c
   Reading a random access file sequentially */
#include <stdio.h>

/* clientData structure definition */
struct clientData {
   int acctNum;           /* account number */
   char lastName[ 15 ];   /* account last name */
   char firstName[ 10 ]; /* account first name */
    double balance;        /* account balance */
}; /* end structure clientData */

int main()
{
   FILE *cfPtr; /* credit.dat file pointer */

   /* create clientData with no information */
   struct clientData client = { 0, "", "", 0.0 };

   /* fopen opens the file; exits if file cannot be opened */
   if ( ( cfPtr = fopen( "credit.dat", "rb" ) ) == NULL ) {
      printf( "File could not be opened.\n" );
   } /* end if */
```

Outline

fig11_15.c (2 of 2)

```c
   else {
      printf( "%-6s%-16s%-11s%10s\n", "Acct", "Last Name",
              "First Name", "Balance" );

      /* read all records from file (until eof) */
      while ( !feof( cfPtr ) ) {
         fread( &client, sizeof( struct clientData ), 1, cfPtr );

         /* display record */
         if ( client.acctNum != 0 ) {
            printf( "%-6d%-16s%-11s%10.2f\n",
                    client.acctNum, client.lastName,
                    client.firstName, client.balance );
         } /* end if */

      } /* end while */

      fclose( cfPtr ); /* fclose closes the file */
   } /* end else */

   return 0; /* indicates successful termination */

} /* end main */
```

```
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
```

Outline

Program Output

11.10 Primjer: A Transaction Processing Program

- Program za obradu bankarskih transkacija
  - 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

Outline

fig11_16.c (1 of 11)

```
/* Fig. 11.16: fig11_16.c
   This program reads a random access file sequentially, updates data
   already written to the file, creates new data to be placed in the
   file, and deletes data previously in the file. */
#include <stdio.h>

/* clientData structure definition */
struct clientData {
   int acctNum;          /* account number */
   char lastName[ 15 ];  /* account last name */
   char firstName[ 10 ]; /* account first name */
   double balance;       /* account balance */
}; /* end structure clientData */

/* prototypes */
int enterChoice( void );
void textFile( FILE *readPtr );
void updateRecord( FILE *fPtr );
void newRecord( FILE *fPtr );
void deleteRecord( FILE *fPtr );

int main()
{
   FILE *cfPtr; /* credit.dat file pointer */
   int choice;  /* user's choice */

```

Outline

fig11_16.c (2 of 11)

```
   /* fopen opens the file; exits if file cannot be opened */
   if ( ( cfPtr = fopen( "credit.dat", "rb+" ) ) == NULL ) {
      printf( "File could not be opened.\n" );
   } /* end if */
   else {

      /* enable user to specify action */
      while ( ( choice = enterChoice() ) != 5 ) {

         switch ( choice ) {

            /* create text file from record file */
            case 1:
               textFile( cfPtr );
               break;

            /* update record */
            case 2:
               updateRecord( cfPtr );
               break;

```

Outline

fig11_16.c (3 of 11)

```
            /* create record */
            case 3:
               newRecord( cfPtr );
               break;

            /* delete existing record */
            case 4:
               deleteRecord( cfPtr );
               break;

            /* display message if user does not select valid choice */
            default:
               printf( "Incorrect choice\n" );
               break;

         } /* end switch */

      } /* end while */

      fclose( cfPtr ); /* fclose closes the file */
   } /* end else */

   return 0; /* indicates successful termination */

} /* end main */

```

Outline

fig11_16.c (4 of 11)

```
/* create formatted text file for printing */
void textFile( FILE *readPtr )
{
   FILE *writePtr; /* accounts.txt file pointer */

   /* create clientData with no information */
   struct clientData client = { 0, "", "", 0.0 };

   /* fopen opens the file; exits if file cannot be opened */
   if ( ( writePtr = fopen( "accounts.txt", "w" ) ) == NULL ) {
      printf( "File could not be opened.\n" );
   } /* end if */
   else {
      rewind( readPtr ); /* sets pointer to beginning of record file */
      fprintf( writePtr, "%-6s%-16s%-11s%10s\n",
               "Acct", "Last Name", "First Name","Balance" );

      /* copy all records from record file into text file */
      while ( !feof( readPtr ) ) {
         fread( &client, sizeof( struct clientData ), 1, readPtr );

```

Outline

fig11_16.c (5 of 11)

```
         /* write single record to text file */
         if ( client.acctNum != 0 ) {
            fprintf( writePtr, "%-6d%-16s%-11s%10.2f\n",
                     client.acctNum, client.lastName,
                     client.firstName, client.balance );
          } /* end if */

      } /* end while */

      fclose( writePtr ); /* fclose closes the file */
   } /* end else */

} /* end function textFile */

/* update balance in record */
void updateRecord( FILE *fPtr )
{
   int account;        /* account number */
   double transaction; /* account transaction */

   /* create clientData with no information */
   struct clientData client = { 0, "", "", 0.0 };

```

fig11_16.c (6 of 11)

```
   /* obtain number of account to update */
   printf( "Enter account to update ( 1 - 100 ): " );
   scanf( "%d", &account );

   /* move file pointer to correct record in file */
   fseek( fPtr, ( account - 1 ) * sizeof( struct clientData ),
          SEEK_SET );

   /* read record from file */
   fread( &client, sizeof( struct clientData ), 1, fPtr );

   /* display error if account does not exist */
   if ( client.acctNum == 0 ) {
      printf( "Acount #%d has no information.\n", account );
   } /* end if */
   else { /* update record */
      printf( "%-6d%-16s%-11s%10.2f\n\n",
              client.acctNum, client.lastName,
              client.firstName, client.balance );

      /* request user to specify transaction */
      printf( "Enter charge ( + ) or payment ( - ): " );
      scanf( "%lf", &transaction );
      client.balance += transaction; /* update record balance */

```

Outline

fig11_16.c (7 of 11)

```
      printf( "%-6d%-16s%-11s%10.2f\n",
              client.acctNum, client.lastName,
              client.firstName, client.balance );

      /* move file pointer to correct record in file */
      fseek( fPtr, ( account - 1 ) * sizeof( struct clientData ),
             SEEK_SET );

      /* write updated record over old record in file */
      fwrite( &client, sizeof( struct clientData ), 1, fPtr );
   } /* end else */

} /* end function updateRecord */

/* delete an existing record */
void deleteRecord( FILE *fPtr )
{
   /* create two clientDatas and initialize blankClient */
   struct clientData client;
   struct clientData blankClient = { 0, "", "", 0 };

   int accountNum; /* account number */

```

fig11_16.c (8 of 11)

```
   /* obtain number of account to delete */
   printf( "Enter account number to delete ( 1 - 100 ): " );
   scanf( "%d", &accountNum );

   /* move file pointer to correct record in file */
   fseek( fPtr, ( accountNum - 1 ) * sizeof( struct clientData ),
          SEEK_SET );

   /* read record from file */
   fread( &client, sizeof( struct clientData ), 1, fPtr );

   /* display error if record does not exist */
   if ( client.acctNum == 0 ) {
      printf( "Account %d does not exist.\n", accountNum );
   } /* end if */
   else { /* delete record */

      /* move file pointer to correct record in file */
      fseek( fPtr, ( accountNum - 1 ) * sizeof( struct clientData ),
         SEEK_SET );

      /* replace existing record with blank record */
      fwrite( &blankClient,
              sizeof( struct clientData ), 1, fPtr );
   } /* end else */

```

Outline

fig11_16.c (9 of 11)

```
} /* end function deleteRecord */

/* create and insert record */
void newRecord( FILE *fPtr )
{
   /* create clientData with no information */
   struct clientData client = { 0, "", "", 0.0 };

   int accountNum; /* account number */

   /* obtain number of account to create */
   printf( "Enter new account number ( 1 - 100 ): " );
   scanf( "%d", &accountNum );

   /* move file pointer to correct record in file */
   fseek( fPtr, ( accountNum - 1 ) * sizeof( struct clientData ),
          SEEK_SET );

   /* read record from file */
   fread( &client, sizeof( struct clientData ), 1, fPtr );

```

Outline

fig11_16.c (10 of 11)

```
   /* display error if account previously exists */
   if ( client.acctNum != 0 ) {
      printf( "Account #%d already contains information.\n",
              client.acctNum );
   } /* end if */
   else { /* create record */

      /* user enters last name, first name and balance */
      printf( "Enter lastname, firstname, balance\n? " );
      scanf( "%s%s%lf", &client.lastName, &client.firstName,
             &client.balance );

      client.acctNum = accountNum;

      /* move file pointer to correct record in file */
      fseek( fPtr, ( client.acctNum - 1 ) *
             sizeof( struct clientData ), SEEK_SET );

      /* insert record in file */
      fwrite( &client,
              sizeof( struct clientData ), 1, fPtr );
   } /* end else */

} /* end function newRecord */

```

fig11_16.c (11 of 11)

```
/* enable user to input menu choice */
int enterChoice( void )
{
   int menuChoice; /* variable to store user's choice */

   /* display available options */
   printf( "\nEnter your choice\n"
           "1 - store a formatted text file of acounts called\n"
           "    \"accounts.txt\" for printing\n"
           "2 - update an account\n"
           "3 - add a new account\n"
           "4 - delete an account\n"
           "5 - end program\n? " );

   scanf( "%d", &menuChoice ); /* receive choice from user */

   return menuChoice;

} /* end function enterChoice */
```

Outline

Program Output

```
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
```