Capitolo 11 – Elaborazione di file

Outline

Obiettivi

- In questo capitolo, apprenderemo a:
  - Creare, leggere, scrivere e modificare i file.
  - Prendere familiarità con i file ad accesso sequenziale.
  - Prendere familiarità con i file ad accesso casuale.

11.1 Introduzione

- I file
  - Possono essere creati, modificati, ed elaborati da programmi scritti in C
  - Sono utilizzati per la memorizzazione permanente dei dati
    - La memorizzazione di dati in variabili ed array è solo temporanea

11.2 La gerarchia dei dati

- Gerarchia dei dati:
  - Bit – il più piccolo
    - Valore `0` o `1`
  - Byte – 8 bits
    - Utilizzato per memorizzare un carattere
      - Cifre decimali, lettere, e simboli speciali
  - Campi – gruppo di caratteri
    - Esempio: your name
  - Record – gruppo di campi
    - Rappresentato da `struct` o `class`
    - Esempio: In a payroll system, a record for a particular employee that contained his/her identification number, name, address, etc.

11.2 La gerarchia dei dati

- Gerarchia dei dati:
  - File – gruppo di record
    - Esempio: payroll file
  - Database – gruppo di files

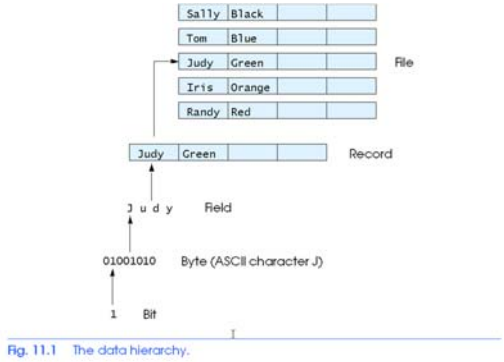


Fig. 11.1 The data hierarchy.

11.2 La gerarchia dei dati

- File
  - Record chiave
    - Identifica un record per facilitare il ritrovamento di uno specifico record da un file
  - File sequenziale
    - I record sono tipicamente ordinati in base alla chiave

11.3 File e Stream

- Il C vede un file come una sequenza di byte
  - I file terminano con un marcatore di fine file
    - O, terminano con uno specifico byte
- Quando si apre un file si crea uno stream
  - Fornisce il canale di comunicazione tra file e programma
  - L'apertura di un file restituisce un puntatore ad una struttura `FILE`
    - Esempi:
    - `stdin` - standard input (keyboard)
    - `stdout` - standard output (screen)
    - `stderr` - standard error (screen)

11.3 File e Stream

- Struttura `FILE`
  - Descrittore di file
    - Index into operating system array called the open file table
  - File Control Block (FCB)
    - Found in every array element, system uses it to administer the file

11.3 File e Stream

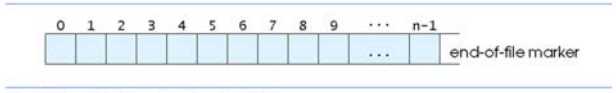


Fig. 11.2 C's view of a file of *n* bytes.

11.3 File e Stream

- Funzioni di lettura/scrittura
  - `fgetc`
    - Legge un carattere da un file
    - Puntatore `FILE` come argomento
    - `fgetc( stdin )` è equivalente a `getchar()`
  - `fputc`
    - Scrive un carattere su un file
    - Puntatore `FILE` e un carattere come argomento
    - `fputc( 'a', stdout )` è equivalente a `putchar( 'a' )`
  - `fgets`
    - Legge una linea da un file
  - `fputs`
    - Scrive una linea su un file
  - `fscanf` / `fprintf`
    - Versioni per file equivalenti di `scanf` e `printf`

Outline

fig11_03.c (1 of 2)

```
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
```


Outline

fig11_03.c (2 of 2)

```
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 */
```

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 Creazione di un file ad accesso sequenziale

- Il C non impone una struttura file
  - Non esiste la nozione di record in un file
  - Il programmatore deve fornire la struttura
- Creare un file
  - `FILE *cfPtr;`
    - Crea un puntatore `FILE` chiamato `cfPtr`
  - `cfPtr = fopen("clients.dat", "w");`
    - La funzione `fopen` restituisce un puntatore `FILE` al file specificato
    - Ha due argomenti – il file da aprire e la modalità d'apertura
    - Se l'apertura fallisce viene restituito `NULL`

11.4 Creazione di un file ad accesso sequenziale

Computer system	Key combination
UNIX systems	*<return> <ctrl> d*
IBM PC and compatibles	*<ctrl> z*
Macintosh	*<ctrl> d*

Fig. 11.4 End-of-file key combinations for various popular computer systems.

11.4 Creazione di un file ad accesso sequenziale

  - `fprintf`
    - Utilizzato per stampare in un file
    - Come la `printf`, eccetto che il primo argomento è un puntatore `FILE` (puntatore al file in cui si vuole stampare)
  - `feof( FILE pointer )`
    - Restituisce l'indicatore end-of-file
  - `fclose( FILE pointer )`
    - Chiude il file specificato
    - Eseguito automaticamente quando termina il programma
    - È buona pratica chiudere il file esplicitamente
- Dettagli
  - I programmi possono operare su un file, su più file o su nessun file
  - Ogni file deve avere un nome unico ed il suo rispettivo puntatore

11.4 Creazione di un file ad accesso sequenziale

Mode	Description
`r`	Open a file for reading.
`w`	Create a file for writing. If the file already exists, discard the current contents.
`a`	Append; open or create a file for writing at end of file.
`r+`	Open a file for update (reading and writing).
`w+`	Create a file for update. If the file already exists, discard the current contents.
`a+`	Append; open or create a file for update; writing is done at the end of the file.
`rb`	Open a file for reading in binary mode.
`wb`	Create a file for writing in binary mode. If the file already exists, discard the current contents.
`ab`	Append; open or create a file for writing at end of file in binary mode.
`rb+`	Open a file for update (reading and writing) in binary mode.
`wb+`	Create a file for update in binary mode. If the file already exists, discard the current contents.
`ab+`	Append; open or create a file for update in binary mode; writing is done at the end of the file.

Fig. 11.6 File open modes.

11.5 Lettura da un file ad accesso sequenziale

- Lettura
  - Crea un puntatore `FILE`, collegarlo al file da leggere
    `cfPtr = fopen( “clients.dat", "r" );`
  - Utilizzare la `fscanf` per leggere dal file
    - Come la `scanf`, eccetto che il primo argomento è un puntatore `FILE`

    `fscanf( cfPtr, "%d%s%f", &accounnt, name, &balance );`
  - Dati letti dall’inizio alla fine
  - Punatatore di posizione del file
    - Indica il numero del prossimo byte da leggere/scivere
    - Non è veramente un puntatore, ma un valore intero (specifica la locazione in byte)
    - Detto anche byte offset
  - `rewind( cfPtr )`
    - Riposizionamento del punatatore posizione all’inizio del file (byte `0`)

Outline

fig11_07.c (1 of 2)

```
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%s\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
```



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

Outline

fig11_07.c (2 of 2)

```
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)

```
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
```

Outline

fig11_08.c (2 of 5)


```
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
```

Outline

fig11_08.c (3 of 5)


```
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
```

Outline

fig11_08.c (4 of 5)


```
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 */
```

Outline

fig11_08.c (5 of 5)

Program Output

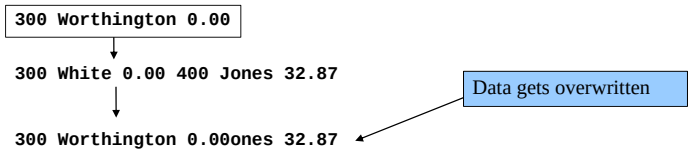
```
Enter request
 1 - List accounts with zero balances
 2 - List accounts with credit balances
 3 - List accounts with debit balances
 4 - End of run
? 1

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 Lettura da un file ad accesso sequenziale

- File ad accesso sequenziale
  - Non può essere modificato senza il rischio di distruggere i dati
  - I campi possono variare in dimensione
    - Diverse rappresentazioni nei file
    - `1`, `34`, `-890` sono tutti `int`, ma hanno diverse dimensioni su disco

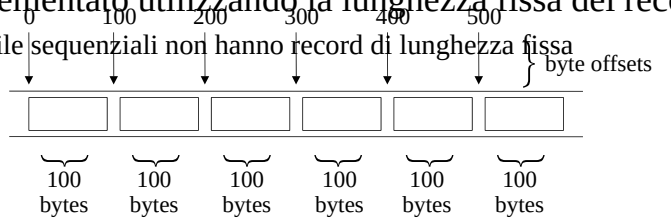
```
300 White 0.00 400 Jones 32.87   (old data in file)
```

If we want to change White's name to Worthington,



11.6 File ad accesso casuale

- File ad accesso casuale
  - L'accesso ai singoli record senza effettuare la ricerca attraverso gli altri record
  - Accesso istantaneo ai record nel file
  - I dati possono essere inseriti senza distruggere gli altri dati
  - I dati precedentemente memorizzati possono essere aggiornati o cancellati senza sovrascrivere
- Implementato utilizzando la lunghezza fissa dei record
  - I file sequenziali non hanno record di lunghezza fissa



11.7 Creazione di un file ad accesso casuale

- Dati in un file ad accesso casuale
  - Non formattati (memorizzati come "raw bytes")
    - Tutti i dati delle stesso tipo (**`int`**, per esempio) utilizzano la stessa quantità di memoria
    - Tutti i record dello stesso tipo hanno una lunghezza fissa
    - I dati non sono human readable

11.7 Creazione di un file ad accesso casuale

- Funzioni I/O non formattate
  - `fwrite`
    - Trasferisce i byte da una locazione di memoria al file
  - `fread`
    - Trasferisce i byte da file ad una locazione di memoria
  - Esempio:

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

    - `&number` – Locazione dei byte da trasferire
    - `sizeof( int )` – Numero di byte da trasferire
    - `1` – per gli array, numero di elementi da trasferire
      - In questo caso, "un elemento" di un array da trasferire
    - `myPtr` – file da trasferire da o a

11.7 Creazione di un file ad accesso casuale

- Scrivere `struct`

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

  - `sizeof` – restituisce la dimensione in byte dell'oggetto in parentesi
- Per scrivere più elelmenti di un array
  - Puntatore all'array come primo argomento
  - Numero di elementi da scrivere come terzo argomento

Outline

fig11_11.c (1 of 2)

```
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 */
```


Outline

fig11_11.c (2 of 2)

```
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 Scrittura su un file ad accesso casuale

- `fseek`
  - Setta il puntatore della posizione del file ad una specifica posizione
  - `fseek(` *pointer*, *offset*, *symbolic_constant* `);`
    - *pointer* – puntatore al file
    - *offset* – puntatore della posizione sul file (0 è la prima locazione)
    - *symbolic_constant* – specifica da dove leggere nel file
    - `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

Outline

fig11_12.c (1 of 3)

```c
/* 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)

```c
      /* 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)

```c
      } /* 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 Scrittura su un file ad accesso casuale

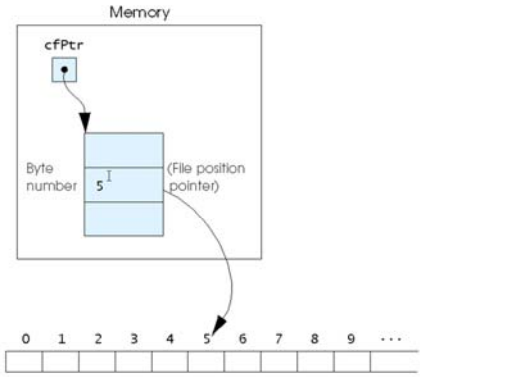


Fig. 11.14 The file position pointer indicating an offset of 5 bytes from the beginning of the file.

11.9 Lettura da un file ad accesso casuale

- `fread`
  - Legge un numero specificato di byte dal file in memoria

```
fread( &client, sizeof (struct clientData), 1,
  myPtr );
```

  - Può leggere più elementi di lunghezza-fissa di un array
    - Fornisce il puntatore all' array
    - Indica il numero di elementi da leggere
  - Per leggere più elementi, si specifica il terzo elemento

Outline

fig11_15.c (1 of 2)

```
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 */
```


Outline

fig11_15.c (2 of 2)

```
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 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

Outline

fig11_16.c (1 of 11)

```
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
```


Outline

fig11_16.c (2 of 11)

```
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
```


Outline

fig11_16.c (3 of 11)

```
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
```

Outline

fig11_16.c (4 of 11)


```
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
```

Outline

fig11_16.c (5 of 11)


```
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( "Acount #%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
```

Outline

fig11_16.c (6 of 11)


```
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
```

Outline

fig11_16.c (7 of 11)

```
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
```

Outline

fig11_16.c (8 of 11)


```
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
```

Outline

fig11_16.c (9 of 11)


```
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
```

Outline

fig11_16.c (10 of 11)


```
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 acounts 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 */
```

Outline

fig11_16.c (11 of 11)

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
```