CHAPTER 11 – FILE PROCESSING

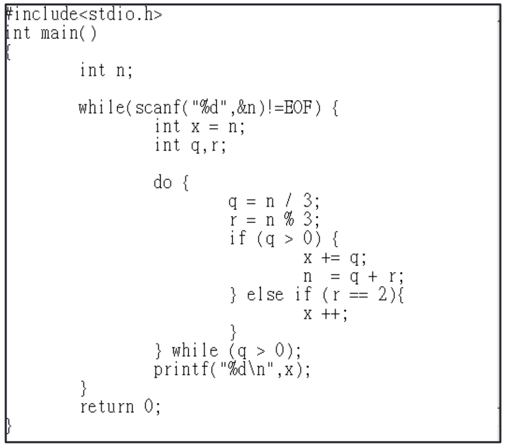
Stream IO

- File pointers
  - FILE*
- 開檔關檔
  - fopen
  - fclose
- 格式化輸入輸出(text files)
  - fscanf/fprintf
  - fgets/fputs
  - fgetc/fputc
- 非格式化(binary files) 輸入輸出
  - fseek/ftell
  - fread
  - fwrite

- Useful file operations
  - fflush(FILE* fp);
    - Flushes buffer for fp
    - fflush(NULL) will flush all buffers
  - int remove(const char* filename);
    - Delete a file
  - int rename(const char*old, const char* new);
    - Change the name of a file
  - setbuf/setvbuf
  - FILE* tmpfile(void);
  - char* tmpnam(char*s);

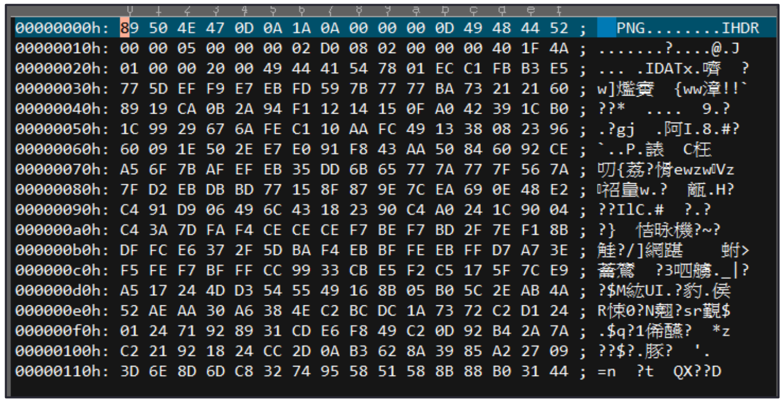
資料檔(File)-1/2

- 記憶體中存的資料只是暫時的。資料檔可以儲存大量資料，並可對它修改，新增，刪除資料等操作。
- 資料檔類別:
  - 資料檔若根據**讀取資料方式**可分成兩類:
    - 循序檔(Sequential files)
      - 資料檔必須從檔案頭至檔案尾循序讀/寫。
    - 隨機存取檔 (Random Access Files)
      - 資料檔可從檔案任意位置讀/寫。
  - 資料檔若根據**資料儲存格式**也可分成兩類:
    - 文字檔(Text files)
      - 資料以本文的方式儲存在檔案裡，通常可用一般文字編輯器開啟文字檔。
    - 二進制格式檔 (Binary Files)
      - 資料以二進制格式儲存在檔案裡。
  - 一般搭配的方式
    - 循序檔用文字檔格式
    - 隨機存取檔用二進制格式檔格式
- 紀錄的鍵值(Record key)
  - 可以由鍵值找到相要的那筆資料。



```
#include<stdio.h>
int main()
{
        int n;

        while(scanf("%d",&n)!=EOF) {
                int x = n;
                int q,r;

                do {
                        q = n / 3;
                        r = n % 3;
                        if (q > 0) {
                                x += q;
                                n  = q + r;
                        } else if (r == 2){
                                x ++;
                        }
                } while (q > 0);
                printf("%d\n",x);
        }
        return 0;
}
```

文字檔內容

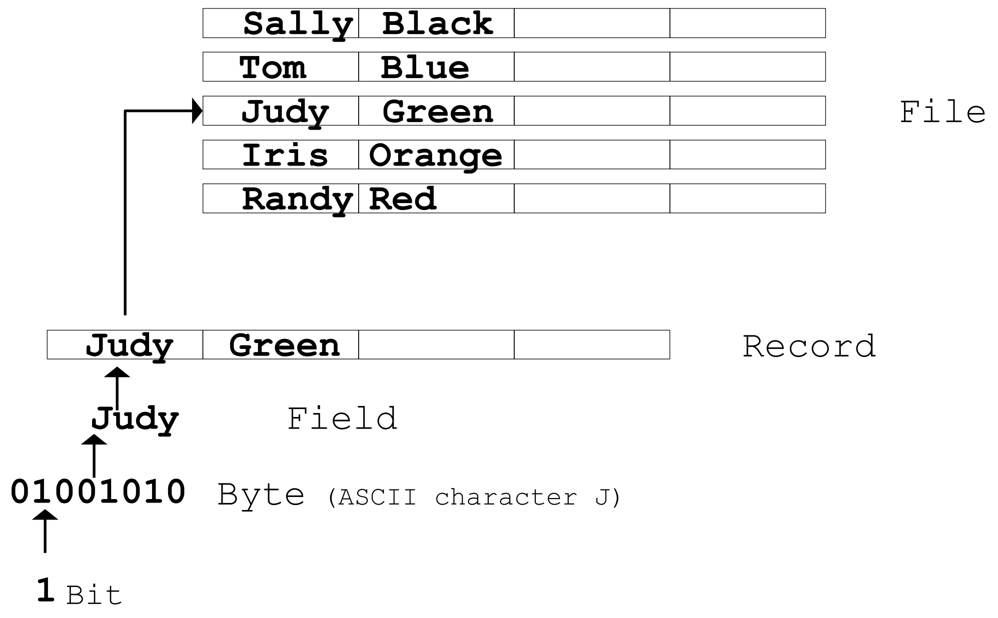


```
00000000h: 89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52 ;  PNG........IHDR
00000010h: 00 00 05 00 00 00 02 D0 08 02 00 00 00 40 1F 4A ; .......?....@.J
00000020h: 01 00 00 20 00 49 44 41 54 78 01 EC C1 FB B3 E5 ; ... .IDATx.�befehl ?
00000030h: 77 5D EF F9 E7 EB FD 59 7B 77 77 BA 73 21 21 60 ; w]爐賽  {ww漳!!`
00000040h: 89 19 CA 0B 2A 94 F1 12 14 15 0F A0 42 39 1C B0 ; ??*  ....  9.?
00000050h: 1C 99 29 67 6A FE C1 10 AA FC 49 13 38 08 23 96 ; .?gj  .阿I.8.#?
00000060h: 60 09 1E 50 2E E7 E0 91 F8 43 AA 50 84 60 92 CE ; `..P.誸  C枉
00000070h: A5 6F 7B AF EF EB 35 DD 6B 65 77 7A 77 7F 56 7A ; 叨{荔?愲ewzwVz
00000080h: 7F D2 EB DB BD 77 15 8F 87 9E 7C EA 69 0E 48 E2 ; 啗韶w.?  甂.H?
00000090h: C4 91 D9 06 49 6C 43 18 23 90 C4 A0 24 1C 90 04 ; ??IlC.#  ?.?
000000a0h: C4 3A 7D FA F4 CE CE CE F7 BE F7 BD 2F 7E F1 8B ; ?}  恄脉機?~?
000000b0h: DF FC E6 37 2F 5D BA F4 EB BF FE EB FF D7 A7 3E ; 觟?/]網諶    蚹>
000000c0h: F5 FE F7 BF FF CC 99 33 CB E5 F2 C5 17 5F 7C E9 ; 蕭鷟  ?3呬艜._|?
000000d0h: A5 17 24 4D D3 54 55 49 16 8B 05 B0 5C 2E AB 4A ; ?$M紘UI.?豹.侯
000000e0h: 52 AE AA 30 A6 38 4E C2 BC DC 1A 73 72 C2 D1 24 ; R悚0?N麴?sr覲$
000000f0h: 01 24 71 92 89 31 CD E6 F8 49 C2 0D 92 B4 2A 7A ; .$q?1俙醾?  *z
00000100h: C2 21 92 18 24 CC 2D 0A B3 62 8A 39 85 A2 27 09 ; ??$?.豚?  '.
00000110h: 3D 6E 8D 6D C8 32 74 95 58 51 58 8B 88 B0 31 44 ; =n  ?t  QX??D
```

二進制格式檔內容

資料檔(2/2)

- 檔案可由相關記錄組成



使用C語言處理檔案

- 將檔案視為一連串的位元組
  - 使用檔案前必須先開啟檔案:使用函式fopen開啟檔案
  - 使用完畢必須關閉檔案: 使用函式fclose關閉檔案

- 檔案開啟成功後就會有一個Stream關聯此檔案讀/寫相關訊息
  - 函式fopen開啟檔案並回傳一個指向結構FILE的指標，若開啟失敗傳回NULL。

```
FILE* fp = fopen(filename, "r"); //開啟文字檔
if (fp == NULL) {
        printf("%s開起失敗\n",filename);
} else {
       //檔案處理
       //.....
        fclose(fp); //關閉檔案
}
```

標準輸入/輸出

- 程式一啟動就有標準輸入/輸出。
  - stdin - standard input (keyboard)
  - stdout - standard output (screen)
  - stderr - standard error (screen)

- 可在命令列視窗下指令改變標準輸入/輸出。

```
C:\>Program.exe < infile
```
infile為Program.exe的標準輸入

```
C:\>Program.exe > outfile
```
outfile為Program.exe的標準輸出

```
C:>Program1.exe | Program2.exe
```
Program1.exe的標準輸出為Program2.exe的標準輸入

結構FILE

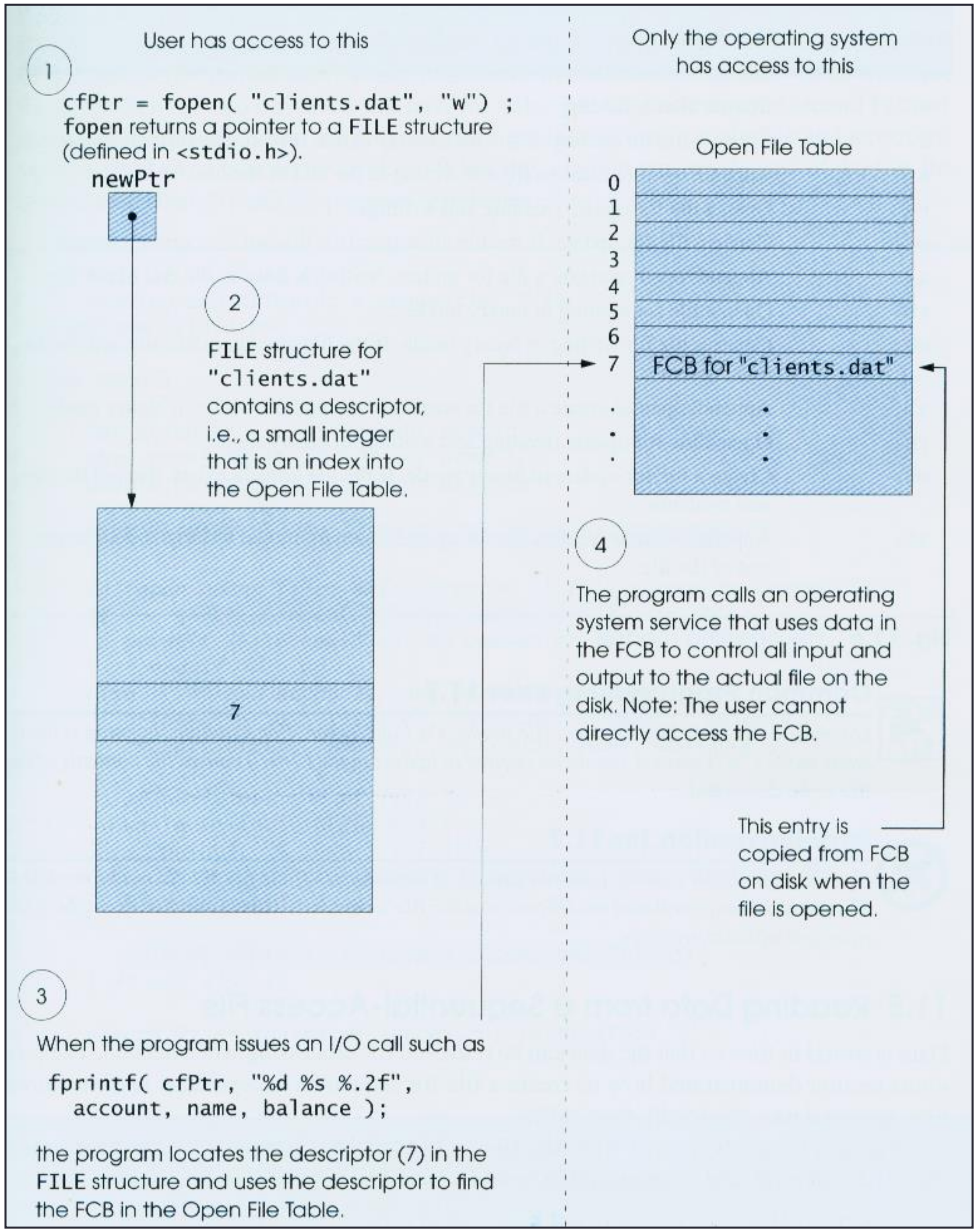
File descriptor

- Index into operating system array called the open file table

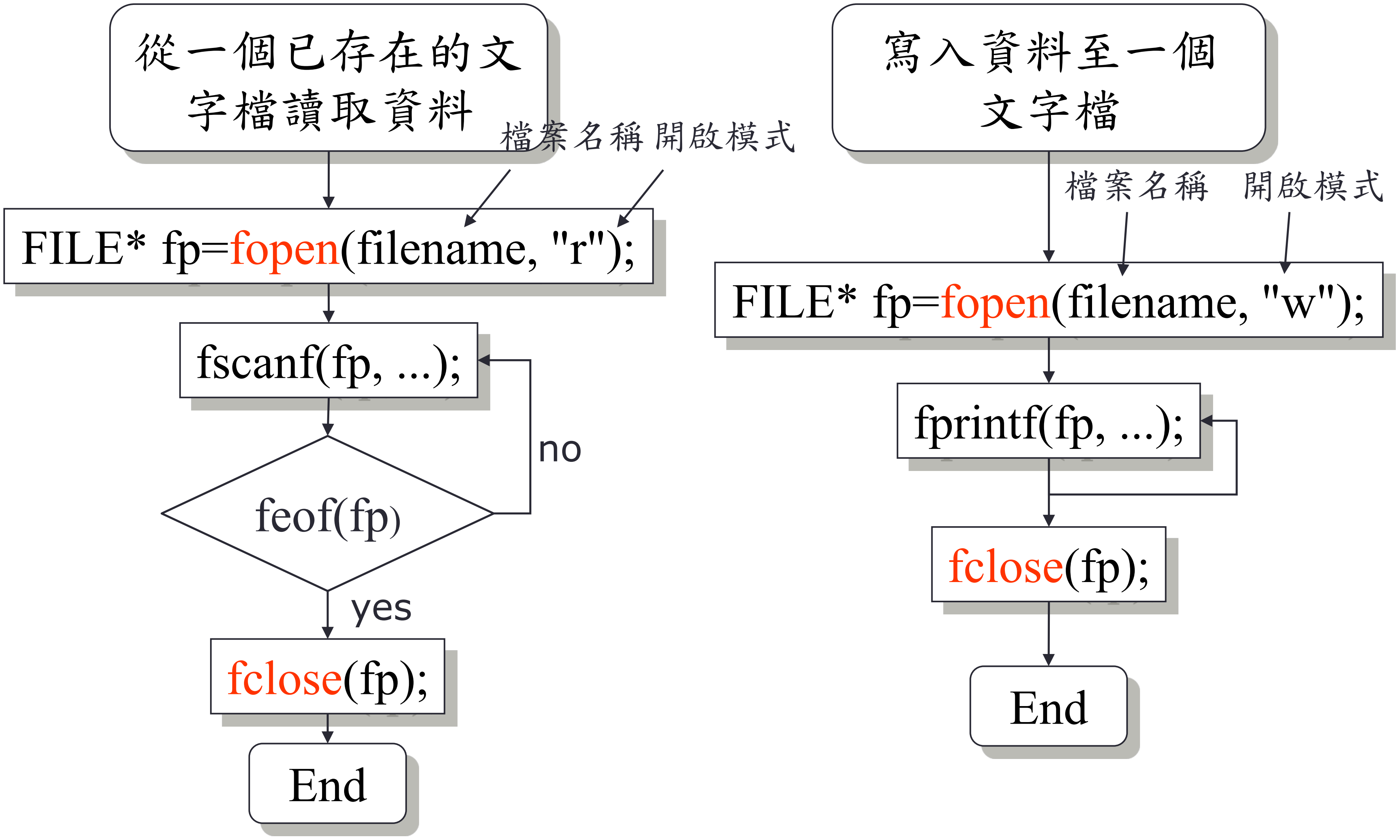
File Control Block (FCB)

- Found in every array element, system uses it to administer the file

FILE*	int fd
stdin	0
stdout	1
stderr	2


User has access to this
Only the operating system has access to this
1
cfPtr = fopen("clients.dat", "w") ;
fopen returns a pointer to a FILE structure (defined in <stdio.h>).
newPtr
2
FILE structure for "clients.dat" contains a descriptor, i.e., a small integer that is an index into the Open File Table.
7
Open File Table
0
1
2
3
4
5
6
7
FCB for "clients.dat"
4
The program calls an operating system service that uses data in the FCB to control all input and output to the actual file on the disk. Note: The user cannot directly access the FCB.
This entry is copied from FCB on disk when the file is opened.
3
When the program issues an I/O call such as
fprintf(cfPtr, "%d %s %.2f", account, name, balance);
the program locates the descriptor (7) in the FILE structure and uses the descriptor to find the FCB in the Open File Table.

處理文字檔步驟


從一個已存在的文字檔讀取資料
檔案名稱 開啟模式
FILE* fp=fopen(filename, "r");
fscanf(fp, ...);
feof(fp)
no
yes
fclose(fp);
End
寫入資料至一個文字檔
檔案名稱 開啟模式
FILE* fp=fopen(filename, "w");
fprintf(fp, ...);
fclose(fp);
End

文字檔案讀/寫函式

- **fgetc(FILE*)**
  - 從檔案讀一個字元
  - **fgetc(stdin)** equivalent to **getchar()**
- **fputc(int ch,FILE*)**
  - 寫入一個字元到檔案
  - **fputc('a', stdout)** equivalent to **putchar('a')**
- **fgets/fputs**
  - 從檔案讀/寫一行
- **fscanf** / **fprintf**
  - 檔案版本之**scanf** and **printf**
  - fscanf(stdin,   )與scanf效果一樣
  - fprintf(stdout,)與printf效果一樣

輸出至文字檔

- **宣告FILE pointer, 將關聯到一個你欲開啟的檔案**

例如、 FILE* myPtr = fopen("myFile.dat", “w”);

- **使用fprintf輸出資料至檔案**

例如、fprintf(myPtr, “%d%s%f\n”, myInt, myString, myFloat);

- **最後關閉檔案**

例如、 fclose(myPtr);

```
//輸出1000整數至data.txt
#include<stdio.h>
int main()
{
     FILE *fp;
     int  data;

     fp = fopen("data.txt","w");
     if (fp == NULL) {
          printf("data.txt 建立失敗\n");
          return 0;
     }
     for(data = 0; data < 1000; ++data) {
          fprintf(fp,"%d\n",data);
     }

     fclose(fp) ;
     return 0;
}
```

從文字檔輸入

- **宣告FILE pointer, 將關聯到一個你欲開啟的檔案**

例如、 `FILE* myPtr = fopen( "myFile.dat", "r" );`

- **使用fscanf從檔案讀資料**

例如、 `fscanf( myPtr, "%d%s%f", &myInt, &myString, &myFloat );`

- **最後關閉檔案**

例如、 `fclose(myPtr);`

- **File position pointer**

是一個數字記錄目前讀寫在檔案第幾個byte

- **rewind(myPtr)**

將file position pointer指到檔案開頭(byte 0)

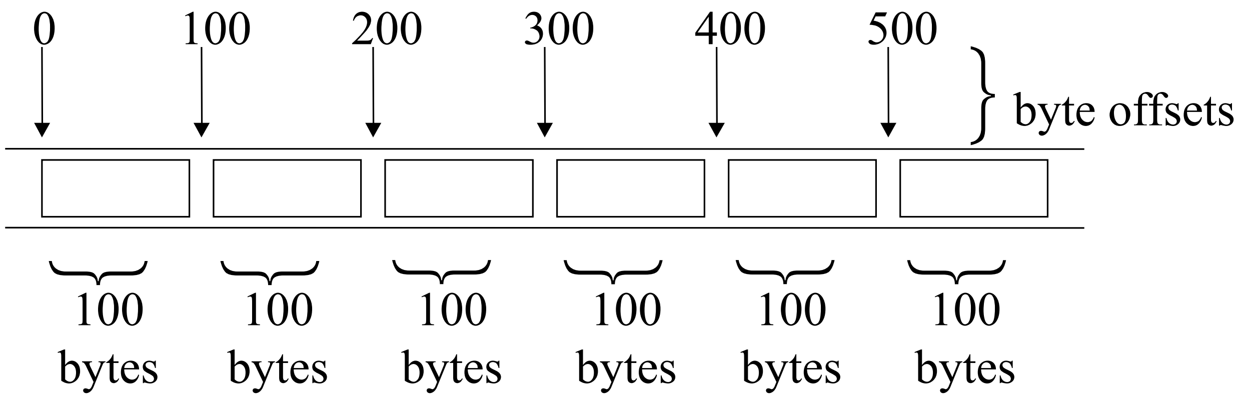
```
//從檔案data.txt讀一串整數
#include<stdio.h>
int main()
{
	FILE *fp;
	int  data;
	char yes;

	fp = fopen("data.txt","r");
	if (fp == NULL) {
		printf("data.txt not found\n");
		return 0;
	}

	do {
		fscanf(fp,"%d",&data);
		while(!feof(fp)) {
			printf("%d\n",data);
			fscanf(fp,"%d",&data);
		}
		printf("read again (y/n)?");
		yes = getchar();
		if (yes== 'y' || yes == 'Y') {
			rewind(fp); //將file position pointer移至檔案開頭
		}
	} while(yes== 'y' || yes == 'Y');

	fclose(fp) ;
	return 0;
}
```

文字檔案開啟模式

Mode	Description
r	開啟只可供輸入的文字檔。
w	開啟只可供輸出的文字檔。若檔案已存在其內容會被清除。
a	開啟只可供輸出的文字檔，會以加在檔尾方式輸出資料。
r+	開啟可供輸入/輸出的文字檔。
w+	開啟可供輸入/輸出的文字檔。若檔案已存在其內容會被清除。
a+	開啟可供輸入/輸出的文字檔，會以加在檔尾方式輸出資料。

隨機存取檔

- 當你需要在檔案上有效率的讀/寫任一筆記錄時，文字檔格式就顯的不適用。
  - 主要問題是每一筆資料可能不一樣大，因此不容易移動file position pointer至任一筆記錄。
  - 例如、1, 34, -890 都是整數,但是在文字檔裡若沒控制其輸出格式，其佔的檔案空間會不一樣大(分別佔1, 2, 4位元組)。
- 若每筆記錄長度一致則可以由資料推算出此筆資料在檔案的第幾個位元組，進而輕易讀/寫任意一筆記錄。



- **二進制格式檔非常適合這種情況。**
- 索引檔(Indexed file)

二進制格式檔

- 資料存放在二進制格式檔裡
  - 非格式化
  - 就是他在記憶體的內容(stored as "raw bytes")
    - 例如每個整數都佔一樣大的空間
    - 不容易直接讀懂
- **輸出/輸入函式**
  - **fwrite**
    - 將記憶體裡一連串的byte寫入檔案

    例如、將連續100整數輸出到檔案裡。

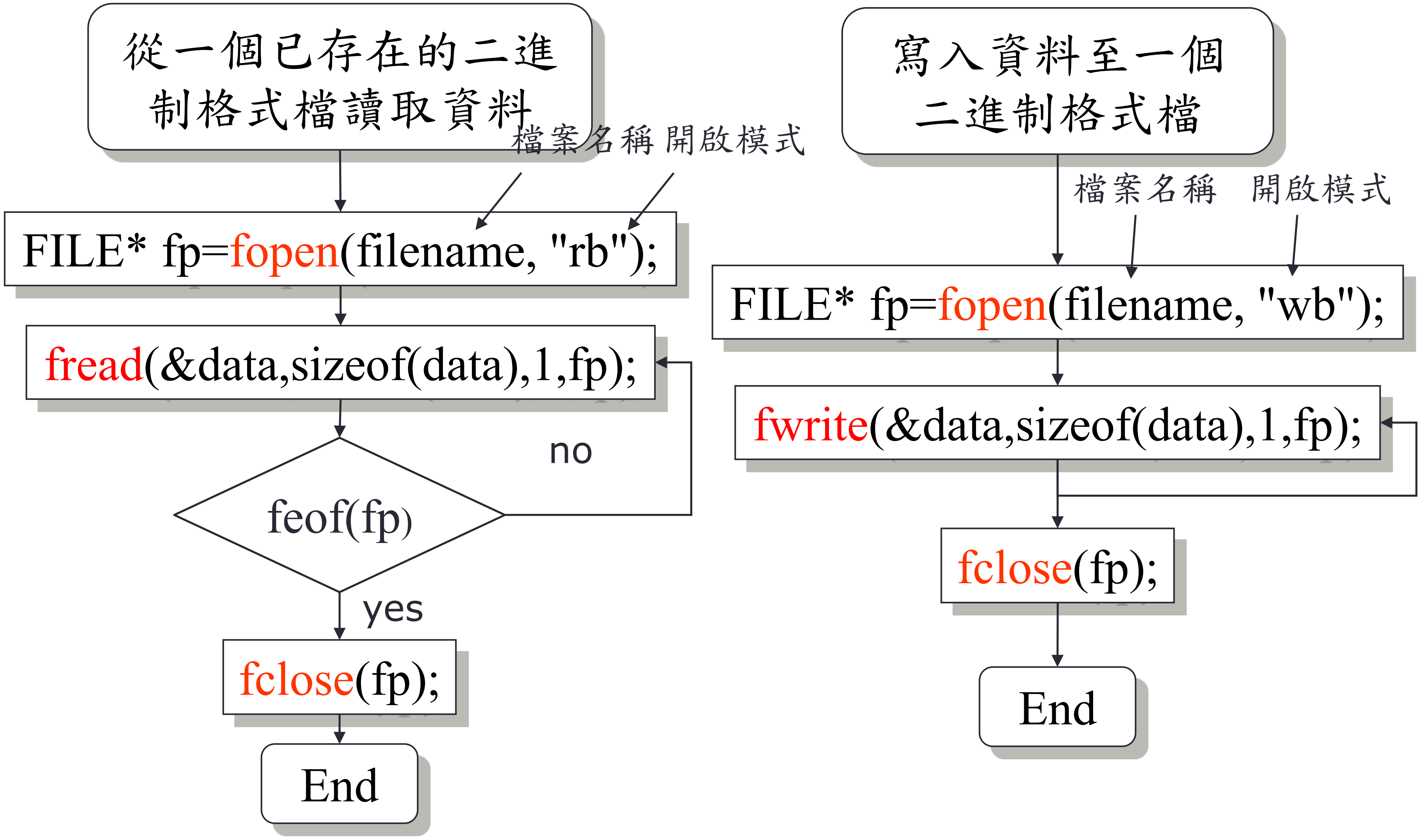
    ```
    int number[100];
    fwrite(number, sizeof( int ), 100, fp );
    ```

    - **number** – 資料起始位置
    - **sizeof(int)** – 元素大小
    - 總共**100**元素
    - **fp** – File to transfer to or from
  - **fread**
    - 將檔案裡一連串的byte寫入記憶體

    例如、從檔案裡讀入100整數到陣列number。

    ```
    fread(number, sizeof( int ), 100, fp );
    ```

處理二進制格式檔步驟



從一個已存在的二進制格式檔讀取資料
檔案名稱
開啟模式
FILE* fp=fopen(filename, "rb");
fread(&data,sizeof(data),1,fp);
feof(fp)
no
yes
fclose(fp);
End
寫入資料至一個二進制格式檔
檔案名稱
開啟模式
FILE* fp=fopen(filename, "wb");
fwrite(&data,sizeof(data),1,fp);
fclose(fp);
End

二進制格式檔模式

Mode	Description
rb	開啟只可供輸入的二進制檔。
wb	開啟只可供輸出的二進制檔。若檔案已存在其內容會被清除。
ab	開啟只可供輸出的二進制檔，會以加在檔尾方式輸出資料。
rb+	開啟可供輸入/輸出的二進制檔。
wb+	開啟可供輸入/輸出的二進制檔。若檔案已存在其內容會被清除。
ab+	開啟可供輸入/輸出的二進制檔，會以加在檔尾方式輸出資料。

輸出/輸入結構變數

- 二進制格式檔非常適合結構變數

例如、

```
struct myStruct {
		char name[100];
		int grade;
} myObject, myObjects[100];

fwrite( &myObject, sizeof (struct myStruct), 1, fp); //將myObject存入檔案
fread( &myObject, sizeof (struct myStruct), 1, fp); //從檔案讀入一筆struct myStruct資料
```

- 將連續100筆struct myStruct資料寫入檔案

```
fwrite(myObjects, sizeof (struct myStruct), 100, fp);
```

- 從檔案讀入連續100筆struct myStruct資料

```
fread(myObjects, sizeof (struct myStruct), 100, fp);
```

- **使用sizeof計算每個結構元素大小**

移動/查詢file position pointer

- 移動file position pointer
  - **`fseek(FILE*`** *pointer,unsigned offset,int symbolic_constant*`);`
    - *pointer* – pointer to file
    - *offset* – file position要移到哪裡 (從0算起)
    - 從檔案何處開始算位移*symbolic_constant*可以為下列其中一個
      - **`SEEK_SET`** – 從檔案開始算起
      - **`SEEK_CUR`** – 目前file position pointer算起
      - **`SEEK_END`** – 從檔案尾算起
- 查詢file position pointer
  - **`ftell(FILE*`** *pointer*`)`

範例:儲存100整數於二進制格式檔

```
#include<stdio.h>
int main()
{
        FILE*      fp;
        int        data, i;
        fp = fopen("data.bin","wb");
        if (fp == NULL) {
                printf("data.bin開啟失敗\n");
                return 0;
        }
        for(i = 0; i < 100; ++i) {
                data = rand();
                fwrite(&data,sizeof(data),1,fp);
        }
        fclose(fp);
        return 0;
}
```

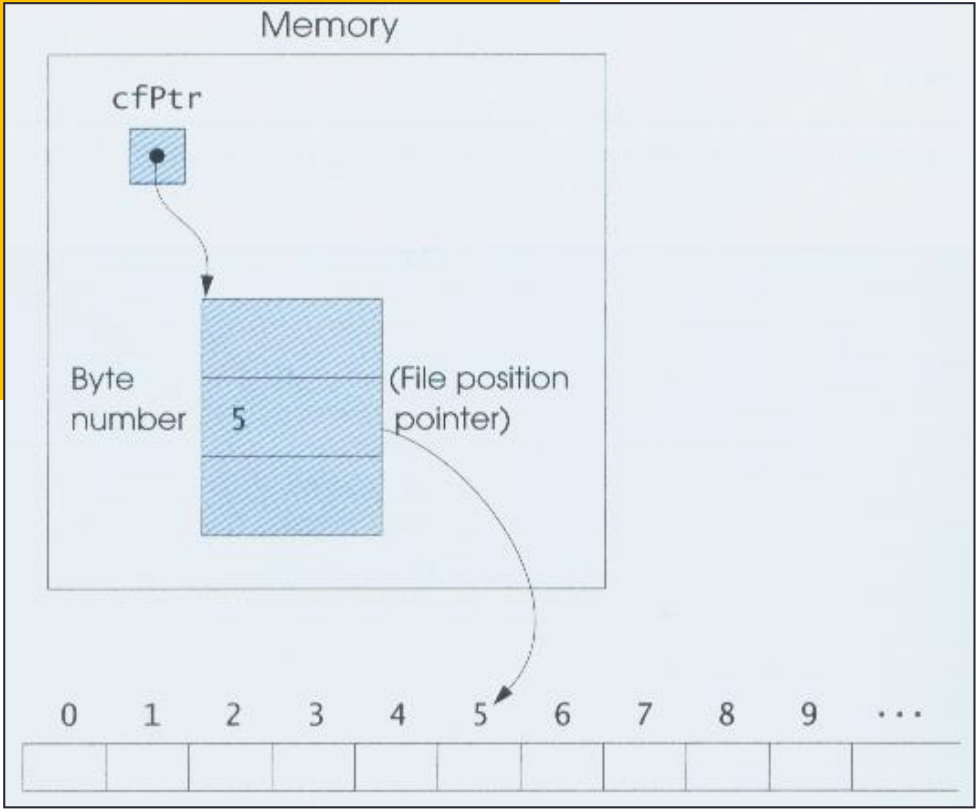
範例:從於二進制格式檔讀取整數資料

```
#include<stdio.h>
int main()
{
	FILE*	fp;
	unsigned data;
	int	totalNumber,id, fileSize;
	fp = fopen("data.bin","rb");
	if (fp == NULL) {
		printf("data.bin開啟失敗\n");
		return 0;
	}

	fseek(fp,0,SEEK_END);
	fileSize	= ftell(fp);
	totalNumber = fileSize/sizeof(int);
	printf("檔案大小共%d位元組，有%d整數\n",fileSize,totalNumber);

	printf("查詢第幾個整數?");
	scanf("%d",&id);
	while(id >=0 && id < totalNumber) {
		fseek(fp,sizeof(int)*id,SEEK_SET);
		fread(&data,sizeof(int),1,fp);
		printf("第%d個整數內容是%d\n",id,data);
		printf("查詢第幾個整數?");
		scanf("%d",&id);
	}
	fclose(fp);
	return 0;
}
```

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

struct clientData {
   int acctNum;
   char lastName[ 15 ];
   char firstName[ 10 ];
   double balance;
};

int main()
{
   FILE *cfPtr;
   struct clientData client = { 0, "", "", 0.0 };

   if ( ( cfPtr = fopen( "credit.dat", "rb+" ) ) == NULL )
      printf( "File could not be opened.\n" );
   else {
      printf( "Enter account number"
              " ( 1 to 100, 0 to end input )\n? " );
      scanf( "%d", &client.acctNum );

      while ( client.acctNum != 0 ) {
         printf( "Enter lastname, firstname, balance\n? " );
         fscanf( stdin, "%s%s%lf", client.lastName,
                client.firstName, &client.balance );
         fseek( cfPtr, ( client.acctNum - 1 ) *
                sizeof( struct clientData ), SEEK_SET );
         fwrite( &client, sizeof( struct clientData ), 1,
                 cfPtr );
         printf( "Enter account number\n? " );
```

```
            scanf( "%d", &client.acctNum );
         }

         fclose( cfPtr );
      }

   return 0;
}
```

```
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
```

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

struct clientData {
   int acctNum;
   char lastName[ 15 ];
   char firstName[ 10 ];
   double balance;
};

int main()
{
   FILE *cfPtr;
   struct clientData client = { 0, "", "", 0.0 };

   if ( ( cfPtr = fopen( "credit.dat", "rb" ) ) == NULL )
      printf( "File could not be opened.\n" );
   else {
      printf( "%-6s%-16s%-11s%10s\n", "Acct", "Last Name",
              "First Name", "Balance" );

      while ( !feof( cfPtr ) ) {
         fread( &client, sizeof( struct clientData ), 1,
                cfPtr );

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

      fclose( cfPtr );
   }

   return 0;
}
```

```
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

- 此程式示範

  使用隨機存取檔作資料處理，提供的操作有
  - 更新帳號資料
  - 加入新帳號資料
  - 刪除帳號資料
  - 將資料存入文字檔

```c
/* 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
   already in the file.                                */
#include <stdio.h>

struct clientData {
   int acctNum;
   char lastName[ 15 ];
   char firstName[ 10 ];
   double balance;
};

int enterChoice( void );
void textFile( FILE * );
void updateRecord( FILE * );
void newRecord( FILE * );
void deleteRecord( FILE * );

int main()
{
   FILE *cfPtr;
   int choice;

   if ( ( cfPtr = fopen( "credit.dat", "rb+" ) ) == NULL )
      printf( "File could not be opened.\n" );
   else {

      while ( ( choice = enterChoice() ) != 5 ) {

         switch ( choice ) {
            case 1:
               textFile( cfPtr );
               break;
            case 2:
               updateRecord( cfPtr );
               break;
            case 3:
               newRecord( cfPtr );
               break;
            case 4:
               deleteRecord( cfPtr );
               break;
         }
      }

      fclose( cfPtr );
   }

   return 0;
}

void textFile( FILE *readPtr )
{
   FILE *writePtr;
   struct clientData client = { 0, "", "", 0.0 };

   if ( ( writePtr = fopen( "accounts.txt", "w" ) ) == NULL )
      printf( "File could not be opened.\n" );
   else {
      rewind( readPtr );
      fprintf( writePtr, "%-6s%-16s%-11s%10s\n",
               "Acct", "Last Name", "First Name","Balance" );

      while ( !feof( readPtr ) ) {
         fread( &client, sizeof( struct clientData ), 1,
                readPtr );

         if ( client.acctNum != 0 )
            fprintf( writePtr, "%-6d%-16s%-11s%10.2f\n",
                     client.acctNum, client.lastName,
                     client.firstName, client.balance );
      }

      fclose( writePtr );
   }

}

void updateRecord( FILE *fPtr )
{
   int account;
   double transaction;
   struct clientData client = { 0, "", "", 0.0 };

   printf( "Enter account to update ( 1 - 100 ): " );
   scanf( "%d", &account );
   fseek( fPtr,
          ( account - 1 ) * sizeof( struct clientData ),
          SEEK_SET );
   fread( &client, sizeof( struct clientData ), 1, fPtr );

   if ( client.acctNum == 0 )
      printf( "Acount #%d has no information.\n", account );
   else {
      printf( "%-6d%-16s%-11s%10.2f\n\n",
              client.acctNum, client.lastName,
              client.firstName, client.balance );
      printf( "Enter charge ( + ) or payment ( - ): " );
      scanf( "%lf", &transaction );
      client.balance += transaction;
      printf( "%-6d%-16s%-11s%10.2f\n",
              client.acctNum, client.lastName,
              client.firstName, client.balance );
      fseek( fPtr,
              ( account - 1 ) * sizeof( struct clientData ),
              SEEK_SET );
      fwrite( &client, sizeof( struct clientData ), 1,
               fPtr );
   }
}

void deleteRecord( FILE *fPtr )
{
   struct clientData client,
                     blankClient = { 0, "", "", 0 };
   int accountNum;

   printf( "Enter account number to "
           "delete ( 1 - 100 ): " );
   scanf( "%d", &accountNum );
   fseek( fPtr,
          ( accountNum - 1 ) * sizeof( struct clientData ),
          SEEK_SET );
   fread( &client, sizeof( struct clientData ), 1, fPtr );
```

```c

   if ( client.acctNum == 0 )
      printf( "Account %d does not exist.\n", accountNum );
   else {
      fseek( fPtr,
         ( accountNum - 1 ) * sizeof( struct clientData ),
         SEEK_SET );
      fwrite( &blankClient,
              sizeof( struct clientData ), 1, fPtr );
   }
}

void newRecord( FILE *fPtr )
{
   struct clientData client = { 0, "", "", 0.0 };
   int accountNum;
   printf( "Enter new account number ( 1 - 100 ): " );
   scanf( "%d", &accountNum );
   fseek( fPtr,
          ( accountNum - 1 ) * sizeof( struct clientData ),
          SEEK_SET );
   fread( &client, sizeof( struct clientData ), 1, fPtr );

   if ( client.acctNum != 0 )
      printf( "Account #%d already contains information.\n",
              client.acctNum );
   else {
      printf( "Enter lastname, firstname, balance\n? " );
      scanf( "%s%s%lf", &client.lastName, &client.firstName,
             &client.balance );
      client.acctNum = accountNum;
      fseek( fPtr, ( client.acctNum - 1 ) *
            sizeof( struct clientData ), SEEK_SET );
      fwrite( &client,
              sizeof( struct clientData ), 1, fPtr );
   }
}

int enterChoice( void )
{
   int menuChoice;

   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 );
   return menuChoice;
}
```

```
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
```

```
Enter account to update (1 - 100): 37
37    Barker            Doug              0.00

Enter charge (+) or payment (-): +87.99
37    Barker            Doug             87.99
```

```
Enter new account number (1 - 100): 22
Enter lastname, firstname, balance
? Johnston Sarah 247.45
```