習題九

作業請繳交至m943040078@student.nsysu.edu.tw 2005/12/29(四)繳交本次習題之流程圖 2006/01/05(四)繳交本次習題之程式碼和執行檔

Objective:

學習利用 Bubble Sort 以及 Merge Sort 將數字從小到大排序。

Exercise:

請用隨機亂數 srand()產生 30000 筆 data，將所產生的前 10000 筆資料存入 list1.dat 檔中，再將其餘的 20000 筆資料存入 list2.dat 檔中。接下來個別將兩檔案(list1.dat 及 list2.dat)使用 Bubble Sort 排序(從小到大)，最後將已排序過後的兩檔案 order1.dat 和 order2.dat 作 Merge Sort，且將結果存入 order3.dat 中。

格式如:

```
#include <stdio.h>
        :
FILE *finput，*foutput;
        :
printf(" 請輸入取樣個數 : ");
scanf("%d"，&total);
        :
finput=fopen("input.txt"，"r+");//開啓檔案，檔案在使用前是需先經過開啓動作
foutput=fopen("output.txt"，"w"); //開啓檔案，檔案在使用前是需先經過開啓動作
        :
        :
while(!feof(finput))
 {
fscanf(finput， "%s\n"， buff);     //從input.txt中讀取數字
num = atoi(buff);       //將字元轉換成整數型態
```

```
        :
fprintf(foutput，" %d\n"，num);     //將從input.txt中讀取的整數寫入output.txt中
 };
        :
 fclose(finput);        //關閉檔案
 fclose(foutput);       //關閉檔案
 }
```

Analysis:

1..使用 rand()

rand()用法：

rand( ) // 產生一個 0 ~ 32767 之間的亂數，必需使用標頭檔 stdlib.h

/* stdlib.h 宣告了一些數值轉換, 配置記憶體的函數 */

例：

```
#include< stdlib.h>
...
a = rand( );
```

完整的例子：產生 10 個 0 ~ 32767 之間的亂數

```
#include<stdio.h>
 #include<stdlib.h>
void main()
{ int i,a; for(i=1;i<=10;i++)
   { a=rand(); printf("%d\n",a); }
}
```

note:

光使用 rand()並無法符合本次習題之要求,

必須加入 srand()之使用, srand()目的在於使執行時產生的亂數不同

```
srand(time(NULL)); /*以系統時間做爲亂數種子值*/
```

srand()用法:

需 include stdlib.h

數值=rand();

例：

取 1-10 的亂數 a=(rand()%10)+1;
取 1-100 的亂數 a=(rand()%100)+1;
取 100-1000 的亂數 a=(rand()%901)+100;
由上幾例可以歸納出來
要取 a~b 的亂數可以這麼寫：
(rand()%(b-a+1))+a
如果單用 rand 取亂數會發現取多次後會出現相同的亂數
這個時候就可以用其他技巧來幫忙
利用 srand() (定義在 stdlib.h)
用 srand 取亂數需要一個參數作爲種子以產生新的亂數序列
而這個參數通常使用目前的時間傳入，這時候就需要用 time() (include <time.h>)
來幫忙。
利用 srand() (定義在 stdlib.h)
用 srand 取亂數需要一個參數作爲種子以產生新的亂數序列
而這個參數通常使用目前的時間傳入，這時候就需要用 time() (include <time.h>)
來幫忙。
使用方法
在使用 rand 的前一行加上
srand(time(NULL));

ex:
```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void main()
{
int a;
srand(time(NULL));
a=(rand()%100)+1;
printf("The Random Number is %d .n", a);
```

2.#include（檔 案 的 含 入）

此 敘 述 之 主 要 目 的 是 讓 我 們 將 某 個 程 式 檔 或 標 頭 檔 包 括 在 目 前 的 程 式 內 ， 使 目 前 的 程 式 可 引 用 該 檔 內 的 資 料 或 程 式 . 語 法 如 下 ：

<l> #include "檔案名稱" ： 此表示系統將會到目前的目錄（路徑 ）

下尋找所指定的檔案 ，如果找不到 ，則會去系統設定的目錄底下尋找 .

<2> #include <檔案名稱> : 此表示系統將會到系統設定的目錄底下尋找所指定的檔案 .

Example:

```
#include < stdio.h >
#include "def.h"

main()
{
        printf("PI=%2.5f\n",PI);
        printf("a+b=%d\n",a+b);
}
```

-- def.h 的檔案內容 --

```
#define PI   3.14159
#define a    10
#define b    20
```

執行結果 :

```
PI=3.14159
a+b=30
```

3.fopen():

```
FILE *fp;
fp=fopen("檔案名稱","存取模式");
```

在這裡，fp 是一個指標，它指向 FILE 結構變數。FILE 結構內存著一些關於檔案的資訊，如：檔案位置指示，資料傳輸緩衝區的長度及其在記憶體中的位址等等，通常我們可以不用去理會這些數值，只要會使用 C 所提供的檔案處理函式，就可以了。FILE *fp; 中的 fp 通常稱爲檔案指標。在使用 C 所提供的檔案處理函式時，只要指定好檔案指標，就是對那個已開啓的檔案做相對應的處理。 fopen() 函式的兩個參數，第一個是"檔案名稱"，也就是要處理的檔案名稱，如果不在所執行的目錄，就要指定檔案的全名，也就是要包含路徑。第二個參數是"存取模式"，有下列字串可供選擇：

"r"	開啓一個文字檔(text)，供程式讀取。
"w"	開啓一個文字檔(text)，供程式將資料寫入此檔案內。如果磁碟內不包含這個檔案，則系統會自行建立這個檔案。如果磁碟內包含這個檔案，則此檔案內容會被蓋過而消失。
"a"	開啓一個文字檔(text)，供程式將資料寫入此檔案的末端。如果此檔案不存在，則系統會自行建立此檔案。

"rb"	開啓一個二元檔(binary)，供程式讀取。
"wb"	開啓一個二元檔，供程式將資料寫入此檔案內。如果磁碟內不包含這個檔案，則系統會自行建立這個檔案。如果磁碟內包含這個檔案，此檔案內容會被蓋過而消失。
"ab"	開啓一個二元檔(binary)，供程式將資料寫入此檔案末端，如果此檔案不存在，則系統會自行建立此檔案。

4.fprintf():

fprintf( ) 主要目的是供你將資料，以格式化方式寫入某檔案內

使用格式如下：

```
fprintf( fp ， "…….", ……);
```

此函數控制列印區和列印和列印變數區的使用，格式和printf( )使用格式相同. fprintf( )和printf( )兩者唯一的差別是，printf( )會將資料列印在螢幕上，而 fprintf( )會將資料列印在某個檔案內。

5.fscanf():

要讀取檔案內的資料時，可以用 fscanf 。其格式如下：

```
fscanf( fp， "控制字串"， &變數 1， &變數 2， ...);
```

fscanf() 的格式如同 scanf() ，只是輸入參數中的第一個參數必須是檔案指標，也就是指定出 fscanf() 要讀取那一個資料檔案。

以下範例：讀取檔案內容，將各個字元與ASCII值一起顯示。

ftoascii.c

```
1 #include <stdio.h>        /* 宣告 printf， scanf， fopen，fprintf...*/
```

```
 2 |                                                                    |
 3 | void main(void)                                                    |
 4 | {                                                                  |
 5 |     FILE *fp;                  /*  檔案指標          */            |
 6 |     char filename[20];   /*  讀取的檔案名稱  */                    |
 7 |     char ch=0;                                                     |
 8 |                                                                    |
 9 |     printf("File name : ");                                        |
10 |     scanf("%19s",  filename );          /*  讀取輸入檔名  */       |
11 |     fp = fopen( filename,  "r");                                   |
12 |     if( fp == NULL )                    /*  判斷是否開啓成功  */   |
13 |         printf("\aCannot open %s !\n",filename);                   |
14 |     else                                                           |
15 |         for( ; ch != EOF ; )           /* ch == EOF  時，迴圈結束  */ |
16 |         {                                                          |
17 |             fscanf( fp,  "%c",  &ch);      /*  讀取一個字元  */    |
18 |             printf("%c = %d\n",  ch,  ch);                        |
19 |         }                                                          |
20 |     fclose(fp);                                                   |
21 | }                                                                  |
```

```
| File name : ftoascii.c                                             |
| # = 35                                                             |
| i = 105                                                            |
| n = 110                                                            |
| ... ...   (省略)                                                   |
| } = 125                                                            |
| = -1                                                               |
```

第 15 行，迴圈以 ch != EOF 作爲結束的條件判斷。其中，EOF 是一個字元，表示檔案結束(End Of File)的字元。所以當 fscanf 讀取到檔案結束字元時，迴圈就會結束執行。

6.fclose():

fclose( ) 用於關閉檔案，如果 fclose( )執行失敗，它的傳回值是非零值

在 C 語言中關閉檔案主要有兩個目的：

1. 檔案在關閉前會將檔案緩衝區資料寫入磁碟檔案內，否則檔案緩衝區資料會遺失。

2. 一個 C 語言程式，在同一時間可開啓的檔案數量有限，一般是 20 個，如果你的程式很 大，要開啓超過 20 個檔案時，你必須將暫時不用的檔案關閉。

7.feof()

檔案末端測試函數—feof( )的使用格式如下:

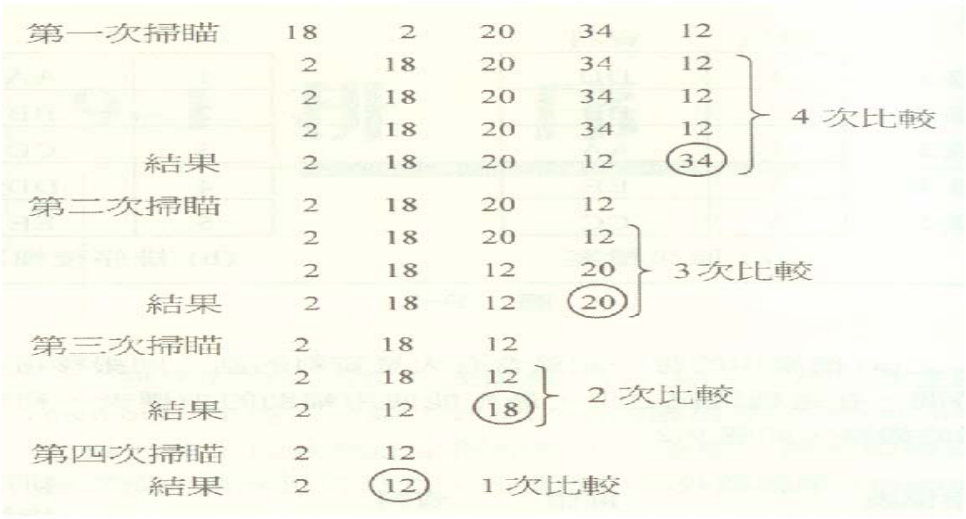
```
| feof(檔案指標);                                    |
```

利用 feof( )，測試檔案指標所指之處，若於末端系統將傳回 0，否則傳回 1。

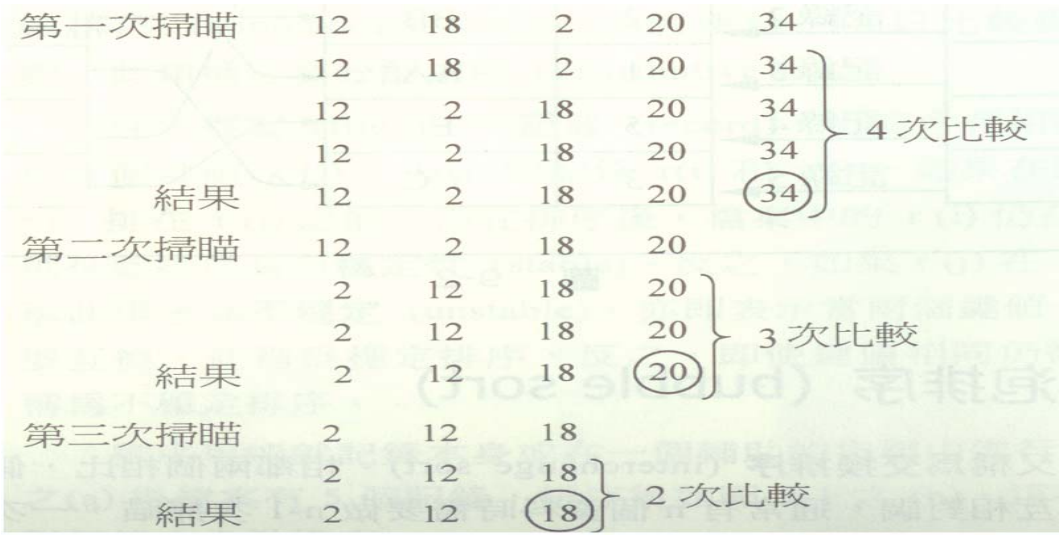
8(1).氣泡排序(bubble sort)

氣泡排序又稱爲**交換排序**(interchange sort),相鄰兩個比,假使前一個比後一個大時,則互相對調。通常有 n 個資料時需要做 n-1 次掃描,一次掃描完後,資料量減少 1,當沒有對調時,就表示已排序好了。

例如有 5 個資料,分別是 18, 2, 20, 34, 12 以氣泡排序的步驟如下:



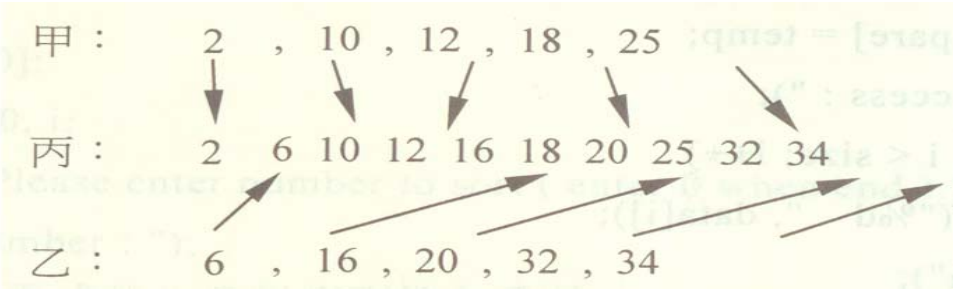
假設鍵值是 12, 18, 2, 20, 34 ,則需要幾次掃描呢?



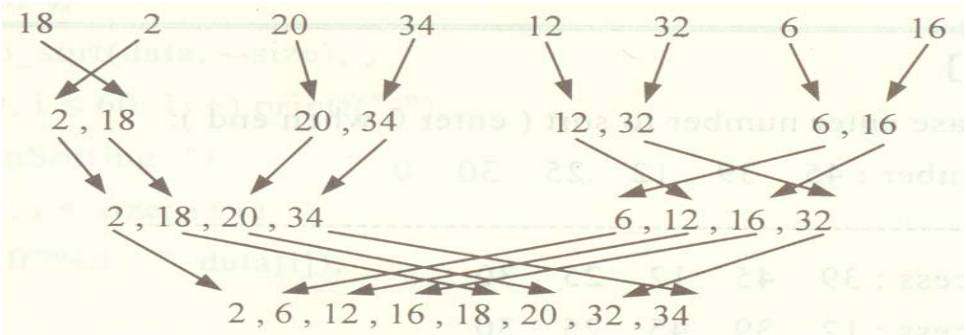
由於在第 3 次掃瞄,沒有做互換的動作,因此可之資料已排序好,不用再比較了。氣泡排序是stable,最壞時間與平均時間均爲$O(n^2)$,所需要額外空間也很少。

8(2).合併排序(merge sort)

合併排序乃是將兩個或兩個以上已排序好的檔案,合併成一個大的已排序好的檔案。例如有兩個已排序好的檔案分別甲={2, 10, 12, 18, 25},乙={6, 16, 20, 32, 34}。合併排序過程如下:甲檔案的第一個資料是 2,而乙第一個資料是 6,由於 2 小於 6,故將 2 寫入丙檔案的第一個資料;甲檔案的第二個資料是 10,10 比 6 大,故 6 寫入丙檔案;乙檔案的第二個資料爲 16,16 比 10 大,故 10 寫入丙檔案;以此類推,最後丙檔案爲{2, 6, 10, 12, 16, 18, 20, 25, 32, 34}。



上述的合併排序的前題是將兩個已經排序好的資料合併在一個檔案中。假使在一堆無排序的資料,我們可以將其分割成二部份,一直分割到每一群組只有一個資料時,再將它們兩兩合併,如下圖所示,假設有下列 8 個鍵值 18, 2, 20, 34, 12, 32, 6, 16。



從上圖..大致可以發現最後合併的動作乃在應用上對兩個排序好的資料加以合併的方法。

合併分類是stable,最壞時間與平均時間均爲O(nlog10n)。所需的額外空間與檔案大小成正比。

}

9.把字元寫入檔案

fputc()函數能將一個字元送往檔案，並儲存之。其格式爲：

```
fputc ( 字元，檔案指標)
```

字元：可爲字元常數或字元變數

範例如下:

```
#include <stdio.h>
void main( )
{
    FILE *fp;
     fp= fopen(“kdata.dat” ，” w”);
     if( fp != NULL)
        fputc( ‘A’ ，fp);
     fclose(fp);
    }
```

說明：fp 爲檔案指標，fopen()函數「開啓」(Open)新檔—kdata.dat，第 2 參數—“w”表示欲將資料存入檔案中。若 fopen()成功地開啓該檔，就傳回指標值，存入 fp 指標，使 fp 指向 kdata.dat 檔。反之，若 fopen()無法開啓 kdata.dat，就傳回「虛」(NULL)指標。因此，可比較 fp 值是否爲 NULL，來得知 fopen()是否圓滿地開啓檔案。一旦開啓 kdata.dat 檔案，就執行 putc(fp，’ A’)，將字元—‘A’，送往 fp 所指之檔案—kdata.dat 儲存起來。最後，fclose()函數將 kdata.dat 檔案「關」起來。

10.從檔案中讀出字元

想從檔案讀取一個字元，可用 fgetc()函數，其格式爲：

```
fgetc(檔案指標)
        ↑
        |__.....(將傳回整數)
```

爲何傳回整數而非字元呢？請想一想，檔案內含有兩種資料—

⊙一般字元

⊙表示 EOF 的特殊字元—Ctrl-Z

fgetc()函數的處理如下：

⊙如果讀到一般字元，就轉換爲整數值，再傳回來。

⊙如果讀到 EOF 字元，就將 EOF 字元 Ctrl-Z 轉爲整數值-1，再傳回來。

假若 fgetc()傳回字元，就無法傳回-1 了。

請看個程式，它從 kdata.dat 檔讀出字元—

```
#include <stdio.h>
#include<stdlib.h>
void main( )
{
FILE *fptr;
Int x;
fptr= fopen(“kdata.dat” ,” r”);
if( fptr == NULL)
{
puts(“can’ t open file!!!”);
exit(0);
}
while( (x=fgetc(fptr) ) != -1)
printf(“%c” , x);
fclose(fptr);
}
```

指標 fptr 指向 kdata.dat 檔，整數變數 x，準備接受 fgetc()傳回來的值。fopen()
函數的第 2 參數—“r”表示：欲從 kdata.dat 讀出(Read)資料。
While 迴圈內之 fgetc()從 kdata.dat 檔讀取一個字元，傳給 x 變數，再拿 x 與-1 比較，判斷是否已讀 EOF 記號(Ctrl-Z)了。如果讀到一般字元，printf()就顯示之，
直到 Ctrl-Z 字元爲止。
由於 stdio.h 標頭檔定義了 EOF 常數，代表整數值-1，所以上述指令—
while((x=fgetc(fptr)) != -1)　　相當於　　while((x=fgetc(fptr)) != EOF)

11.for loop:

語法如下:

```
for (<算式 1>; <算式 2>; <算式 3>)
  <循環主體>;
```

如果循環主體包含多於一個語句時，它應該被一對大括號包圍著。

Ex:

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

```
  int n， product;
  printf(“n? ”);
  scanf(“%d”， &n);
   for (product = n; product <= 1000; product *= n)
     printf(“%d\n”， &product);
  return 0;
}
```

12.宣告:

資料型態 變數名稱[， 變數名稱];

ex:

int i， j， k， sum = 0;

char c ='0'， C ='c'， ch;

float x， y， z， area = 0.0， radius;

13.printf("輸入字串"):

printf 是一個列印指令，其結果是在電腦螢幕上列印"輸入實數"這一串字。

printf 最後的字母是"f"，所謂"f"，是指"format"的意思，也就是我們的列印是根據一種規格的，以後有一個例子會將這一點解釋清楚。

(1) printf("平均值=%f"， sum/5.0);

printf 是一個列印指令，首先我們注意到在" "裡有幾個字是和變數無關的，那就是"平均值="，在=的後面，我們要列印一個浮點數字，這個浮點數字是什麼呢？這個數字就是 sum/5.0。

(2) printf("\n");

這個指令是換行指令，每次執行這個指令，螢幕游標就會換行(跳到下一行)。

ex: printf("hello!\n");

14.scanf("%f"， &data)

scanf 是一個讀取指令，我們可以想像使用者會從鍵盤鍵入一個數字，每次鍵入以後，我們的程式就會將這個數字讀進去。讀到那裡去呢？讀到 data 這個變數裡去。如果你鍵入 6.4，data 就變成了 6.4，如果鍵入的數字是 5.1，data 就是 5.1。

至於"%f"是指什麼呢？"%f"中的 f 是 floating point(浮點數字)，這個指令是將 data 的值變成所讀進來的浮點數字。爲什麼會在 data 前面加上&呢？這點很難解釋，我們不妨記下這個規則，反正 scanf("%f"， data)是不對的，一定要 scanf("%f"， &data) 才對。