1061 計 算 機 程 式 設 計　期 末 考 參 考 答 案

107/01/09 (二)

考試時間：14:10 – 16:10

1. 在一個應用程式裡我們希望在程式裡表示如右圖 m 列 n 行的矩陣 A，請回答下列問題：

3	2	-1	6
7	-4	-5	2
6	8	3	-2

   a. [5] 請在 main()函式裡定義一個 10x10 的區域整數陣列 A，並且在定義時初始化為右圖的資料

Sol:
```
int main() {
    int A[10][10] = {{3,2,-1,6},{7,-4,-5,2},{6,8,3,-2}};
    return 0;
}
```
不足的部份編譯器會設定為 0，但是需要有兩層的大括號

   b. [5] 這個應用程式裡需要以每一列為單位由小到大排順序，兩列的順序定義為：如果由左到右第一個不相等的元素 $A_{ik} < A_{jk}$, $0 \le i,j \le m-1, 0 \le k \le n-1$ 則第 i 列小於第 j 列，請撰寫一個函式 int compRow(int n, int row1[], int row2[])比對兩列的大小，其中 n 為陣列的行數，回傳 -1 代表 row1 比 row2 小，0 代表兩列資料相等，1 代表 row1 比 row2 大

Sol:
```
int compRow(int n, int row1[], int row2[]) {
    int i;
    for (i=0; i<n; i++)
        if (row1[i]<row2[i])
            return -1;
        else if (row1[i]>row2[i])
            return 1;
    return 0;
}
```

   c. [10] 接題 b，請撰寫一個函式 void bubbleSort(int m, int n, int A[][10]) 運用氣泡排序法將這個陣列的內容排好，其中 m 為陣列的列數，n 為陣列的行數

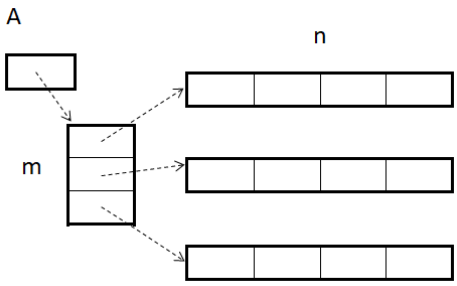
Sol:
```
void bubbleSort(int m, int n, int A[][10]) {
    int i, j, k, tmp;
    for (i=0; i<m-1; i++)
        for (j=m-2; j>=i; j--)
            if (compRow(n, A[j], A[j+1])>0)
                for (k=0; k<n; k++) {
                    tmp = A[j][k];
                    A[j][k] = A[j+1][k];
                    A[j+1] [k]= tmp;
                }
}
```
這個版本是把最小的搬到最前面，把最大的搬到最後面的寫法如下：
```
int bubbleSort(int m, int n, int A[][10]) {
    int i, j, k, tmp;
    for (i=m-1; i>=1; i--)
        for (j=0; j<i; j++)
```

```
            if (compRow(n, A[j], A[j+1])>0)
                for (k=0; k<n; k++) {
                    tmp = A[j][k];
                    A[j][k] = A[j+1][k];
                    A[j+1] [k]= tmp;
                }
    }
```

d. [10] 假設變數 n 為全域變數，請運用 stdlib 中的 qsort 將這個陣列各列由小到大排好 (請定義一個 int comp(const void *, const void *)函式來配合 qsort 運作)

Sol:

因為 compRow()函式需要三個參數, 其中第一個參數是每一列有幾個元素，也就是矩陣有幾個直行，qsort 沒有辦法知道這個參數，所以在這個版本裡面需要把 n 設定為全域變數；另外一種處理方法是一開始把整個 10x10 陣列的每一個元素都設為一個常數，然後呼叫 compRow(10, (int*)a, (int*)b);超過 n 的部份反正是相等的，如果兩列是相等的，會浪費一些時間比較，對結果則沒有影響，在 C++和 C++11 語法裡面就有其它封裝得比較好的方法處理這個 n 的數值的傳遞。




```
int comp(const void *a, const void *b) {
    return compRow(n, (int *)a, (int *)b);
}
…
qsort(A, m, sizeof(int[10]), comp);
```

e. [5] 如果這個應用裡需要的陣列範圍很大，例如 2≤m,n≤10000，像題 a 中在 main()函式裡直接定義固定維度 10000x10000 的整數陣列的話，編譯時不會有錯誤，但是執行時會發生錯誤，請問原因為何?

Sol:

堆疊的空間一般只有 32 Mbytes, 所以這個超級大的區域陣列會把堆疊空間用完

f. [10] 假設變數 m,n 存放的數值已知，範圍如題 e，請參考右上圖運用 malloc() 函式配合下列指標變數 A 的定義動態地配置 m 列 n 行的二維陣列 int **A;

Sol:

```
#include <malloc.h> //  或是  #include <stdlib.h>
…
int **A = (int **) malloc(m*sizeof(int*));
for (int i=0; i<m; i++)
    A[i] = (int *) malloc(n*sizeof(int));
```

g. [10] 假設題 f 中配置好的陣列裡面的資料已經設定好，請再運用 stdlib 中的 qsort 將這個陣列各列由小到大排好 (請再定義一個 int comp(const void *, const void *)函式來配合 qsort 運作)

Sol:

n 的處理方式和題 d 相同

```
int comp(const void *a, const void *b) {
    return compRow(n, *(int **)a, *(int **)b);
}
…
qsort(A, m, sizeof(int*), comp);
```

h. [10] 請問題 c, 題 d, 題 g 中排順序時平均資料對調的次數各為何? (請以 m,n 的函數表示，留下最高次項即可，常數倍數及其它項可以省略，例如 0.25 x^2 + 0.5 x + 10 只要寫 x^2 即可)

Sol:

題 c: $O(m^2 n)$

題 d: O(n m log m)

題 g: O(m log m)

i. [5] 請問題 a 的區域陣列和題 f 的動態陣列在程式裡使用時都可以用 A[i][j]的語法來存取矩陣中的資料 A_{ij}，編譯器其實都是解釋成*(*(A+i)+j)，兩個陣列的組織方式明明有很大的差別，請解釋為什麼同樣的 A[i][j]或是*(*(A+i)+j)語法對於兩種陣列都可以正常運作?

Sol:

關鍵在於 A 的兩種定義下 *(A+i) 的運算是不一樣的，在 int A[10][10]; 的定義下，A+i 位址會增加 i*sizeof(int[10])，*(A+i) 是一個 10 個整數的陣列，也就是它的第一個元素的位址；而在 int **A; 的定義下 A+i 位址會增加 i*sizeof(int*)，*(A+i) 是一個整數變數的位址

2. 在一個應用程式裡我們希望依序產生如右圖的資料輸出：

a. [6] 請完成下列程式

```
01. #include <stdio.h>
02. void print(int m, int num[]) {
03.         for (int i=m-1; i>=0; i--)
04.                 printf("%d ", num[i]);
05.         printf("\n");
06. }
```

```
07. int main() {
08.         int i, m=3, n=5, num[10];
09.         for (i=0; i<=m; i++) // 設定 num 陣列的初始值
10.                 num[i] = n-1-i;
11.         while (num[m]==n-1-m) {
12.                 print(m, num);
13.                 num[i=0]--;
14.                 while (i<m&&num[i]< 0 ) {
15.                         num[i]= n-1-i;
16.                         num[++i]--;
17.                 }
18.         }
19.         return 0;
20. }
```

請注意最後一位數字由 0 到 n-1, 倒數第二位數字由 0 到 n-2, …, 依此類推, 看成 n 位數的話數字不是連續的

b. [4] 請問題 a 中如果去除第 13 列和第 16 列的 --,第 14 列該改成什麼才會得到相同的結果?

Sol:

1. **如果寫 i<m&&--num[i]<0 的話，當 i>=m 時因為條件已經失敗了，&&的右半完全不會執行 (short circuit evaluation)，--的動作沒有執行，外層的 while 迴圈不會結束**
2. **如果改成--num[i]<0&&i<m 的話，測試起來會對，但是要小心原來 i<m 是希望保證 num[m]這個可能超過陣列範圍的元素不要使用到，改到後面就沒有效果了，不過這裡原來程式第 16 列本來就會將 num[m]減 1**
3. **如果改成(num[i]--, i<m&&num[i]<0) 和原本的程式是完全一樣的**

c. [10] 請完成下列產生相同輸出的遞迴函式

```
2 3 4
2 3 3
2 3 2
2 3 1
2 3 0
2 2 4
2 2 3
2 2 2
2 2 1
2 2 0
2 1 4
2 1 3
2 1 2
2 1 1
2 1 0
2 0 4
2 0 3
2 0 2
2 0 1
2 0 0
1 3 4
1 3 3
1 3 2
1 3 1
1 3 0
1 2 4
1 2 3
1 2 2
1 2 1
1 2 0
1 1 4
1 1 3
1 1 2
1 1 1
1 1 0
1 0 4
1 0 3
1 0 2
1 0 1
1 0 0
0 3 4
0 3 3
0 3 2
0 3 1
0 3 0
0 2 4
0 2 3
0 2 2
0 2 1
0 2 0
0 1 4
0 1 3
0 1 2
0 1 1
0 1 0
0 0 4
0 0 3
0 0 2
0 0 1
0 0 0
```

```
01. #include <stdio.h>
02.
03. void countDown(int n, int data[], int istart, int iend);
04. void print(int num[], int m) {
05.         for (int i=0; i<m; i++)
06.                 printf("%d ", num[i]);
07.         printf("\n");
08. }
09.
10. int main() {
11.         int data[8], n=5, m=3;
12.         countDown(n, data, 0, m-1);
13.         return 0;
14. }
15.
16. void countDown(int n, int data[], int istart, int iend) {
17.         int i;
18.         if (istart>iend)
19.                 print(data, iend+1);
20.         else
21.                 for (data[istart]=n-1-(iend-istart); data[istart]>=0; data[istart]--)
22.                         countDown(n, data, istart+1, iend);
23. }
```

3. **[10] 請問右側程式執行後會印出幾個整數？(請將程式的輸出寫出來)**

Sol:

12 個

印出的數字依序是 7, 11, 17, 26, 13, 20, 10, 5, 8, 4, 2, 1

```
n = 14;
while (n != 1) {
        if ((n%2)==1) {
                n = 3 * n + 1;
        }
        else {
                n = n / 2;
                printf("%d ", n);
        }
}
```