時間長度加法器

丁培毅

實習目標：

1. 透過函式參數傳出函式的執行結果
2. 練習傳遞記憶體位址, 練習透過記憶體位址存取資料
3. 練習時間計算的進位處理方式

時間長度加法器

1. 請撰寫函式 timeAdd() 來執行時間長度的加法，這個函式的原型爲
   void timeAdd(int *hour1, int *minute1, int *second1, int hour2, int minute2, int second2);
   請配合下面的 main() 函式測試

```
int main() {
   int h1, m1, s1, h2, m2, s2;
   h1 = 3; m1 = 18; s1 = 22; h2 = 5; m2 = 22; s2 = 7;
   printf("%d 小時 %d 分鐘 %d 秒鐘 加上 %d 小時 %d 分鐘 %d 秒鐘\n", h1, m1, s1, h2, m2, s2);
   timeAdd(...);
   printf("\t結果等於 %d 小時 %d 分鐘 %d 秒鐘。\n", h1, m1, s1);
   h1 = 3; m1 = 48; s1 = 32; h2 = 5; m2 = 22; s2 = 57;
   printf("%d 小時 %d 分鐘 %d 秒鐘 加上 %d 小時 %d 分鐘 %d 秒鐘\n", h1, m1, s1, h2, m2, s2);
   timeAdd(...);
   printf("\t結果等於 %d 小時 %d 分鐘 %d 秒鐘。\n", h1, m1, s1);
   h1 = 3; m1 = 88; s1 = 92; h2 = 5; m2 = 122; s2 = 257;
   printf("%d 小時 %d 分鐘 %d 秒鐘 加上 %d 小時 %d 分鐘 %d 秒鐘\n", h1, m1, s1, h2, m2, s2);
   timeAdd(...);
   printf("\t結果等於 %d 小時 %d 分鐘 %d 秒鐘。\n", h1, m1, s1);
   return 0;
}
```

2. 程式執行範例輸出如右：

```
3 小時 18 分鐘 22 秒鐘 加上 5 小時 22 分鐘 7 秒鐘
        結果等於 8 小時 40 分鐘 29 秒鐘。
3 小時 48 分鐘 32 秒鐘 加上 5 小時 22 分鐘 57 秒鐘
        結果等於 9 小時 11 分鐘 29 秒鐘。
3 小時 88 分鐘 92 秒鐘 加上 5 小時 122 分鐘 257 秒鐘
        結果等於 11 小時 35 分鐘 49 秒鐘。
```


分析

1. 首先先研究一下範例
   03:18:22 + 05:22:07 = 08:40:29　看起來時分秒個自相加即可
   03:48:32 + 05:22:57 = 08:70:89 => 09:11:29　由秒進位到分再到時
   03:88:92 + 05:122:257 = 08:210:349 => 11:35:49　同上的進位法則
   但是每次進位不見得只有 1
2. 函式參數傳遞方法
   a. **傳值**　void fun(int in, const double inArray[])
      呼叫端 int x=10; double y[]={1.1, 2.2, 3.3}; **fun(x, y);**
      函式裡變數 **in** 的數值一開始就是 x 的數值, 可以修改但是不影響呼叫端的變數 x 的數值, 如果改成 const int in 就不能修改了
      陣列 **inArray** 就是陣列 y, 但是不能修改 (inArray[i] 就是 y[i])
   b. **傳位址** void fun(**int ***inout, double inoutArray[])
      呼叫端 int x=10; double y[]={1.1, 2.2, 3.3}; **fun(&x, y);**
      函式裡變數 ***inout** 就是變數 x, 內容可以讀取也可以修改
      陣列 **inArray** 就是陣列 y (inArray[i] 就是 y[i]), 可以讀取/修改
      請把宣告時的 **int*** 看成是『整數的記憶體位址』型態

3. 題目要求的函式
   void timeAdd(int *hour1, int *minute1, int *second1,
   int hour2, int minute2, int second2);
   其中變數組 (*hour1, *minute1, *second1) 存放第一個時間長度值
   (hour2, minute2, second2) 存放第二個時間長度值
   第一個參數傳遞變數的記憶體位址是因爲希望讓兩個時間值加總以後的時間可以藉由這一組參數傳回呼叫端 (事實上是一直在直接使用呼叫端的記憶體變數)
   a. 請實作前面分析的時分秒各自相加，結果直接放到第一個時間長度變數中
   b. 請運用 if 敘述計算進位 (請注意第三個範例是提醒你每次進位不見得都是 1，也有可能是 2, 3,….，計算時要一起考慮), 計算時請運用整數的除法與整數的餘數運算 (/, %)
4. 要傳遞多個資料數值回到呼叫端時，如果要運用函式的回傳值來做就需要使用結構變數 struct，否則就是像這樣傳遞變數的位址，或是陣列，在運用陣列時也特別注意 const 的使用，如果陣列裡面的資料不希望函式裡更動的話，就可以運用 const 讓編譯器幫你檢查

4. 整個函式裡使用到這個陣列的所有敘述，沒有任何敘述可能會修改陣列的內容，使得這個函式對於這個陣列沒有 side effect，只能看內容而不會改其內容，在程序化的程式設計裡，函式代表的是茲廖處理的動作，你會不斷地製用函式來隱藏底部的運作細節，呼叫一個函式時你必須很清楚地知道它會將什麼資料處理成爲什麼資料，會修改那些變數，如此才能夠用已經設計好的函式去組合更大更複雜的功能。例如一段程式裡把所有學生的成績資料放在陣列裡

```
int scores[100];
```

也許呼叫計算平均值的函式,

```
double average;
average = calculateAverage(scores, nScores);
```

計算完了以後也許呼叫存檔的函式

```
writeFile(scores, nScores);
```

上面這樣的程序我們有假設 calculateAverage() 函式不會修改 scores 陣列裡面的資料值，這時候如果 calculateAverage() 是你自己寫的函式，自己可以去檢查，但是如果是團隊裡其他成員寫的，你就很希望這個函式是定義成

```
double calculateAverage(const int scores[], int nScores);
```


延伸問題

1. 兩個時間長度大小的比較
2. 時間長度的減法
3. 時間乘上常數
4. 以結構 struct 語法來整合時分秒
5. 整合『日』的表示方法，加減乘法
6. 程式執行時間長短的計算 (55 毫秒精確度，10ms 精確度，CPU clock level 精確度)