以泰勒展開式估算 e^x 數值

丁培毅

實習目標：

1. 練習迴圈敘述
2. 思考**浮點數**的表示方法與表示誤差
3. 熟悉程式處理**數值運算**需要注意的事項
4. 練習 math.h 函式庫的使用

   http://www.cplusplus.com/reference/cmath/?kw=math.h

以泰勒展開式估算 e^x 數值

請撰寫一個程式

1. 要求使用者由鍵盤鍵入一實數 x，以泰勒展開式

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$$

計算 e^x 的數值到誤差小於 10^{-9} 爲止 (e=2.7182818….. 是所謂的 Euler number)

2. 執行範例如下：

範例 1 (藍字部份是使用者自己輸入的部份)：
請輸入 x：**3.5**
e^3.500000 = 33.115451958545
範例 2 (藍字部份是使用者自己輸入的部份)：
請輸入 x：**1**
e^1.000000 = 2.718281828447
範例 3 (藍字部份是使用者自己輸入的部份)：
請輸入 x：**0**
e^0.000000 = 1.000000000000

分析

1. 由題目的描述中可以看到應該要使用迴圈來計算數列的數值，第一個想法就是計算等差級數 0 + 1 + 2 + … + (n-1) 的迴圈

```
int sum=0, i, n=10;
for (i=0; i<n; i++)
    sum += i;
```

不過有兩個明顯的不同，一是上面都是整數變數，這個題目應該要用到被精準的浮點數 double，二是上面這個迴圈是事先知道要執行 n 次的迴圈，但是這個題目似乎不能先知道迴圈需要執行多少次，所以這個迴圈一般來說就會使用 while 迴圈

2. 對於任意的 x 數值來說，由於 x/i 的絕對值隨著 i 變大會趨近於 0 ，所以這個數列的第 i 項 $x^i / i!$ 的絕對值一定隨著 i 增加而趨近於 0，一般來說這樣並不代表第 i+1 項到無窮大項的和會趨近於 0，不過我們只希望透過這個題目練習迴圈的控制，並且瞭解浮點數運算的一些性質，我們先假設前 i 項的部份和會越來越接近 e^x，所以所謂的誤差就是 $| e^x$ – 前 i 項部份和 $|$ ，如果這個數值小於 10^{-9}，計算部份和的迴圈就可以結束了

3. 接下來看到的問題就是 e^x 到底是多少? 如果不知道的話，前面這個迴圈結束條件是算不出來的，我們先做一個假設，就是 C 的函式庫 math.h 裡的函式 exp(x) 可以計算 e^x，而且誤差遠小於 10^{-9}，如此我們就可以得到一個參考值來計算誤差，從而可以評估迴圈是不是該結束了，迴圈長得像下面這樣

```
#include <math.h>
…
double term, etox = 1, expofx = exp(x);
while (fabs(expofx - etox) >= 1e-9) {
    計算 term = x^i / i!
    etox += term;
 }
```

4. 接下來該思考的問題是如何計算第 i 項 $x^i / i!$，當然最直接的方法就是先算出 x^i，再計算出 i!，然後相除得到第 i 項：
   a. 計算 x^i 也就是計算 (…((x * x) * x) * … * x)，看起來可以用計算 0 + 1 + 2 + … + n-1 的迴圈來完成
   b. 計算 i! 也就是計算 (…((1 * 2) * 3) * … * i) ，看起來也可以用計算 0 + 1 + 2 + … + n-1 的迴圈來完成

5. 但是這會有兩層的問題
   a. 效率問題：計算 x^i 其實應該只需要由前一次所算的 x^{i-1} 再乘上 x 就可以了，不需要重複計算 x^{i-1}，計算 i! 也應該由 (i-1)! 的結果乘上 i 就可以，不需要重複計算 (i-1)!
   b. 精確度問題：倍精準浮點數只能精確表示 20! 或是 9^{15} ，再大就開始有誤差，所以先算個別 i! 和 x^i 的誤差會很大，需要先計算 x/i， x^i / i! = (x^{i-1} / (i-1)!) * (x/i)，如此才能在算到比較大的 i 值時，還不會引入太多的誤差，程式修改爲

```
#include <math.h>
…
double term = 1, etox = 1, expofx = exp(x);
while (fabs(expofx - etox) >= 1e-9) {
    term = term * x / i;
    i = i + 1;
    etox += term;
 }
```

6. 最後處理一下輸出的格式：

```
printf("e^%f = %.12f\n", x, etox);
```

7. 注意測試的時候也需要測試 x 是比較大的數，例如 20，也需要測試負數
8. 其實因爲這個練習目的是迴圈的練習，目的並不是真的要非常精確地計算 e^x ，所以我們用了 exp(x) 這個函式來找到適當地迴圈結束條件，如果真的目的是用數列來計算 e^x，那麼應該是在沒有辦法使用 exp(x) 的情況下才有意義，那麼在決定迴圈執行到什麼時候停止時，可能只能用 $| x^i / i! |$ 小於 10^{-9} 作爲結束條件，這時雖然沒有辦法保證與 e^x 的誤差小於 10^{-9} ，但是其實也相當接近了，例如：

```
#include <math.h>
…
int i = 1;
double x, term, etox = 1;
由鍵盤讀入 x
term = x;
while (fabs(term) >= 1e-9) {
    etox += term;
    i = i + 1;
    term = term * x / i;
}
```