

大數問題

一、基本觀念

在某些情況下，我們必須處理位數相當多的一個整數，例如 100 位數，系統內建的資料型態不管是 int、long int、long long int 等，位數顯然都不夠用。要解決這個問題，我們必須自己用程式來處理，最簡單的方法，就是模仿人工處理數字的方式，也就是把數字拆成一個位數一個位數，這個時候我們可以用一個陣列來完成這個動作。不過，這樣一來，所有對於這種超大整數的處理，包括輸入一個大數、兩個大數相加、印出一個大數等等，都需要我們自己寫一個函數來處理。至於這個陣列要用什麼樣的資料型態，因為兩個一位數相乘也不過才 81，所以我們可以用 char 的陣列來記錄一個超長整數，如：

```
char n[300];
```

就可以記錄 300 位數的資料。前面提到大數的輸入及輸出，都需要我們自己寫函數來處理，我們分別把它們定名為 Input() 及 Print()：

```
#include <stdio.h>
#include <string.h>
#define LEN 300
int Input(char n[])
{
}
void Print(char n[])
{
}
void main()
{
    char a[LEN];
    Input(a);
    Print(a);
}
```

上面第三行 #define LEN 300 的意思，即是定義一個常數叫做 LEN，而它的值是 300，下面用到 LEN 的地方，在執行的時候便會自動變成 300。這樣寫的好處是，如果我們有好個地方都用到 300 這個數字，但是後來發現不夠用了，要將它改成 1000，這個時候要一行一行改，還可能不小心漏掉，如果用 #define 定義一個常數，則只要改一個地方就行了。接下來我們看到 Input() 函數要做的步驟：

1. 把該陣列的每一格數字歸零。
2. 將使用者輸入的數字以字串方式存到另一個字串。
3. 計算該字串的長度。
4. 從該字串的尾端(即個位數)開始一個一個位數往左，將它轉換成數字後，存到陣列對應的格子中。

我們把上面提到的字串及超長整數的陣列做一個比較：

字串	s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]	s[8]	s[9]
	'1'	'2'	'3'	'4'	'5'	'6'	0	×	×	×

陣列	n[0]	n[1]	n[2]	n[3]	n[4]	n[5]	n[6]	n[7]	n[8]	n[9]
	6	5	4	3	2	1	0	0	0	0

上面提到的 Input() 的程式碼可以寫成：

```
int Input(char n[])
{
    char s[LEN];
    int i, l;
    for(i=0; i<LEN; i++)
        n[i]=0;
    if(scanf("%s", s)<1) return -1;
    l=strlen(s);
    for(i=0; i<l; i++)
        n[i]=s[l-i-1]-'0';
    return 0;
}
```

上面的 Input() 函數為了要得知是否有資料輸入，所以我們利用傳回值 0 代表輸入成功，傳回 -1 代表傳入失敗。至於 Print() 函數要做的步驟為：

1. 從該陣列最後面往回找到第一個不是 0 的數字。
2. 從該位數字往左邊依序把它們一個一個印出來。
3. 印出換行符號。

它的程式碼可以寫成：

```
void Print(char n[])
{
    int i;
    for(i=LEN-1; i>0; i--)
        if(n[i]!=0) break;
    for(; i>=0; i--)
        printf("%d", n[i]);
    printf("\n");
}
```

寫成之後，我們可以先輸入一個約 60 位數的數字，看是不是可以正常印出結果。

二、大數相加

接下來我們來寫一個處理兩個大數相加的題目，接著上一段的程式，我們接下來要寫一個處理相加的函數 **Add()**，它的做法就跟小學時的加法運算一樣，從個位數開始，一個位數一個位數相加，如果有超過 10 的部分，就把它往前進位。例如：

大數 A	0	1	2	3	4	5	6	7	8	9
大數 B	0	1	2	3	4	5	6	7	8	9
相加	0	2	4	6	8	10	12	14	16	18
進位	0	2	4	6	9	1	3	5	7	8

下面的程式碼可以參考看看：

```
void Add(char a[], char b[], char c[])
{
    int i;
    for(i=0; i<LEN; i++)
        c[i]=a[i]+b[i];
    for(i=0; i<LEN-1; i++) {
        if(c[i]>=10) {
            c[i+1]+=c[i]/10;
            c[i]=c[i]%10;
        }
    }
}
```

接下來把主程式 main() 改寫成：

```
void main()
{
    char a[LEN], b[LEN], c[LEN];
    Input(a);
    Input(b);
    Add(a, b, c);
    Print(c);
}
```

三、#10106 — Product

題目：[按這裡](#)。

說明：每組測試資料有 2 列，分別代表 2 個大數 X、Y ($0 \leq X、Y < 10^{250}$)，輸出 X*Y 的結果。

這一題的大數位數有 250 位，我們原先宣告的 LEN 有 300 所以已經夠用了。因為乘出來的積的最大位數為原來的兩倍，所以那一行 `C[LEN]` 要改成 `C[LEN*2]`，而 `Print()` 函數裡面的 `i=LEN-1` 也要改成 `i=LEN*2-1`。接下來，我們要寫一個乘(Multiply)的函數 `Mul()`，它的做法就跟小學的乘法原理是一樣的，下面我們來看到它的程式碼：

```
void Mul(char a[], char b[], char c[])
{
    int i, j;
    for(i=0; i<LEN*2; i++)
        c[i]=0;
    for(i=0; i<LEN; i++) {
        for(j=0; j<LEN; j++) {
            c[i+j]+=a[j]*b[i];
            if(c[i+j]>=10) {
                c[i+j+1]+=c[i+j]/10;
                c[i+j]=c[i+j]%10;
            }
        }
    }
}
```

再把主程式改成：

```
void main()
{
    char a[LEN], b[LEN], c[LEN*2];
    Input(a);
    Input(b);
    Mul(a, b, c);
    Print(c);
}
```

最後再改成連續輸入的寫法：

```
void main()
{
    char a[LEN], b[LEN], c[LEN*2];
    while(1) {
        if(Input(a)) return;
        Input(b);
        Mul(a, b, c);
        Print(c);
    }
}
```

[上一頁](#)

[首頁](#)

[下一頁](#)