

程式設計與實習(二)

BY 孫茂勛

EMAIL:JOHN85051232@GMAIL.COM

3/28(二) CPE



10409 Die Game

Life is not easy. Sometimes it is beyond your control. Now, as contestants of ACM ICPC, you might be just tasting the bitter of life. But don't worry! Do not look only on the dark side of life, but look also on the bright side. Life may be an enjoyable game of chance, like throwing dice. Do or die! Then, at last, you might be able to find the route to victory.



This problem comes from a game using a die. By the way, do you know a die? It has nothing to do with "death." A die is a cubic object with six faces, each of which represents a different number from one to six and is marked with the corresponding number of spots. Since it is usually used in pair, "a die" is a rarely used word. You might have heard a famous phrase "the die is cast," though.

When a game starts, a die stands still on a flat table. During the game, the die is tumbled in all directions by the dealer. You will win the game if you can predict the number seen on the top face at the time when the die stops tumbling.

Now you are requested to write a program that simulates the rolling of a die. For simplicity, we assume that the die neither slips nor jumps but just rolls on the table in four directions, that is, north, east, south, and west. At the beginning of every game, the dealer puts the die at the center of the table and adjusts its direction so that the numbers one, two, and three are seen on the top, north, and west faces, respectively. For the other three faces, we do not explicitly specify anything but tell you the golden rule: the sum of the numbers on any pair of opposite faces is always seven.

Your program should accept a sequence of commands, each of which is either "north", "east", "south", or "west". A "north" command tumbles the die down to north, that is, the top face becomes the new north, the north becomes the new bottom, and so on. More precisely, the die is rotated around its north bottom edge to the north direction and the rotation angle is 90 degrees. Other commands also tumble the die accordingly to their own directions. Your program should calculate the number finally

光是題目敘述就滿滿的英文

誰想看阿!!!!



<https://uva.onlinejudge.org/external/104/10409.pdf>

3/28(二) CPE

直接看input/output和測資比較快
判斷題目想要做什麼，真的看不懂再回去看敘述

Input

The input consists of one or more command sequences, each of which corresponds to a single game. The first line of a command sequence contains a positive integer, representing the number of the following command lines in the sequence. You may assume that this number is less than or equal to 1024. A line containing a zero indicates the end of the input. Each command line includes a command that is one of 'north', 'east', 'south', and 'west'. You may assume that no white space occurs in any lines.

Output

For each command sequence, output one line containing solely the number on the top face at the time when the game is finished.

Sample Input

```
1
north
3
north
```

不負責任翻譯：
第一行的數字代表接下來有幾行字
每個字代表...#\$\$%^&*

...

..

.

最後0代表輸入結束

3/28(二) CPE

把程式分成數個小部分慢慢寫

Ex：程式可以分成測資輸入、輸出、撰寫題目要求.....

先確認程式碼輸入部分是正確的再來寫其他的部分

```
while(scanf("%d",&num) != EOF)
{
    if(num == 0) break;
    ...
}
```

← 輸入寫完把測資先打進去看看能不能正常執行

3/28(二) CPE

善用Debug技巧

- 有事沒事就printf看一下你的變數看正不正確
- 不要全部寫完送出後才在找問題錯在哪一個地方
 - 通常是找不到

3/28(二) CPE

送出前記得把測試的printf都註解掉

```
char str1[LEN] = {0};
char str2[LEN] = {0};
int num1[LEN] = {0};
int num2[LEN] = {0};
char op = 0;
while(scanf("%s %c %s", str1, &op, str2) != EOF)
{
    //printf("%s %c %s = \n", str1, op, str2); ←
    for(int i = 0 ; i < strlen(str1) ; ++i)
    {
        num1[strlen(str1) - 1 - i] = str1[i] - '0';
        //printf("%d", num1[i]); ←
    }
    for(int i = 0 ; i < strlen(str2) ; ++i)
    {
        num2[strlen(str2) - 1 - i] = str2[i] - '0';
        //printf("%d", num1[i]); ←
    }
    switch(op)
    {
        case '+':
```

3/28(二) CPE

- 不要加system("pause");
- 注意輸入如果同時有字元又有其他型態的時候
換行也是個字元->記得拿掉

最後

如果你是在學校的考場...
練習時請務必完成第一題的hello world
確定你的電腦設備是正常的...

Q : $123456789987654321 + 987654321123456789 = ?$



複習一下整數可使用的範圍

<u>int</u>	4bytes	-2147483648 to 2147483647
unsigned <u>int</u>	4bytes	0 to 4294967295
signed int	4bytes	-2147483648 to 2147483647
long <u>int</u>	4bytes	-2,147,483,648 to 2,147,483,647

大數運算

- 用來做很大很大的數的運算
- 陣列
- 手算怎麼算的程式就怎麼寫
- CPE熱門考題(另外一個是質數表)

數字的表示法

- 陣列
- ASCII
- 順序顛倒

index	0	1	2	3	4
Char[]	1	2	3	4	5
Int[]	5	4	3	2	1

數字的表示法

- 陣列
- ASCII
- 順序顛倒

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define LEN 100
int main()
{
    char str1[LEN] = {0};
    char str2[LEN] = {0};
    char op = 0;
    while(scanf("%s %c %s",str1,&op,str2)!= EOF)
    {
        printf("%s %c %s = \n",str1,op,str2);
    }
    return 0;
}
```

說到字元又想到ASCII

- $9 * 9 = 81$
- $'9' * '9' = ?$

數字的表示法

- 陣列
- **ASCII**
 - 我們不要 '9' 而是要9
 - How? 減掉 '0'
- 順序顛倒

數字的表示法

- 陣列
- ASCII
- 順序顛倒
 - 原本的最高位數在陣列第0個位置不好操作
 - WHY?
 - 多創一個int陣列把順序顛倒過來

index	0	1	2	3	4
Char[]	1	2	3	4	5
Int[]	5	4	3	2	1

數字的表示法

```
char str1[LEN] = {0};
char str2[LEN] = {0};
int num1[LEN] = {0};
int num2[LEN] = {0};
char op = 0;
while(scanf("%s %c %s",str1,&op,str2)!= EOF)
{
    printf("%s %c %s = \n",str1,op,str2);
    for(int i = 0 ; i < strlen(str1) ; ++i)
    {
        num1[strlen(str1) - 1 - i] = str1[i] - '0';
    }
    for(int i = 0 ; i < strlen(str2) ; ++i)
    {
        num2[strlen(str2) - 1 - i] = str2[i] - '0';
    }
}
```

輸出

前面顛倒過了，所以要在顛倒回來

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

```
int main()
{
    ...
    print(num1);
    print(num2);
    ...
}
```

開始講運算之前

先把運算的判斷寫好

```
switch(op)
{
    case '+':
        add(num1,num2);
        break;
    case '-':
        subtraction(num1,num2);
        break;
    case '*':
        multiple (num1,num2);
        break;
}
```

Created by free version of DocuFreezer

$$\begin{array}{r}
 1111111 \\
 9999999 \\
 + 9999999 \\
 \hline
 1999998
 \end{array}$$

大數運算 - 加法

從最低位元開始算，這樣才能算進位

```
void add(int num1[], int num2[])
{
    int num3[LEN] = {0};
    for(int i = 0 ; i < LEN ; ++i)
    {
        num3[i] += num1[i] + num2[i];
        num3[i+1] += num3[i] / 10;
        num3[i] %= 10 ;
    }
    print(num3);
}
```

	1	1	1	1	1	1	1	
	9	9	9	9	9	9	9	9
	9	9	9	9	9	9	9	9
+								
	1	9	9	9	9	9	9	8

Created by free version of DocuFreezer

[illegible]

大數運算 - 減法

與加法相同，只是要考慮借位的情形

```
void subtraction(int num1[], int num2[])
{
    int num3[LEN] = {0};
    for(int i = 0 ; i < LEN ; ++i)
    {
        num3[i] += num1[i] - num2[i];
        if(num3[i] < 0)
        {
            num3[i] += 10;
            num3[i+1] -= 1;
        }
    }
    print(num3);
}
```

$$\begin{array}{r} 10000000 \\ - 9999999 \\ \hline 1 \end{array}$$

大數運算 – 減法

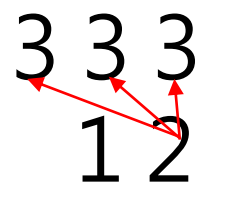
這段程式碼只能在 $\text{num1} > \text{num2}$ 的時候正常執行

Ex : $1 - 2 = ?$

小數減大數怎麼寫？

- 比較兩數長度和最大次方的係數，找出最大數
- 變成大數 – 小數
- 輸出的時候自己補上負號

大數運算 - 乘法

$$\begin{array}{r} \text{X} \\ 333 \\ \times 12 \\ \hline 666 \\ 333 \end{array}$$


大數運算 — 乘法

```
void multiple(int num1[],int num2[])
{
    int num3[LEN] = {0};
    for(int i = 0 ; i < LEN ; ++i)
    {
        if(num1[i] == 0)continue; //0就不用做乘法
        for(int j = 0 ; i+j < LEN ; ++j)
        {
            num3[i+j] += num1[i] * num2[j];
        }
    }
    for(int i = 0 ; i < LEN ; ++i)
    {
        num3[i+1] += num3[i] / 10;
        num3[i] %= 10 ;
    }
    print(num3);
}
```

X

3 3 3
1 2

6 6 6
3 3 3

大數運算 – 比大小

- 比較最高次方的係數
- 如果最高次方的係數相同，比較第二高的
- Ex : 12345 ? 67899
- Ex : 3456 ? 10000

大數運算 - 比大小

```
int compare( int num1[], int num2[])
{
    int i = LEN - 1;
    while( i > 0 && num1[i] == num2[i])
    {
        i--;
    }
    return num1[i] - num2[i];
}
```