Day2: 陣列與字串 07/07

3-1 逆序輸出

讀入不超過 100 個整數。
任務：逆序輸出，以空白隔開。
範例輸入：1 2 3 4 5 7 9
範例輸出：9 7 5 4 3 2 1
最後記得換行

解析：

1. 開一整數陣列 (思考大小 : 100? 10000?)
2. 連續輸入放入陣列中
3. 輸出格式

```
#include <stdio.h>
#define maxn 105
int a[maxn];
int main()
{
    int x, n=0;
    while (scanf("%d", &x)==1)
        a[n++]=x;              //連續讀入
    for (int i = n-1; i>0; i--)
        printf("%d ", a[i]);    //印出前 n-1 個
    printf("%d\n", a[0]);       //處理最後一個
    return 0;
}
```

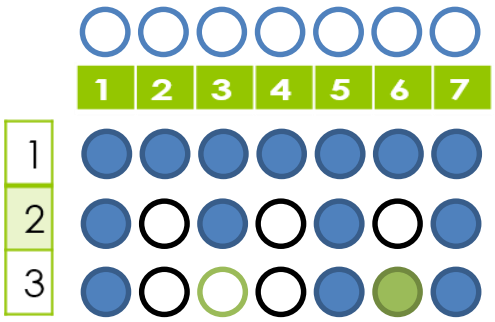
x	1	2	3	4	5	7	9	CtrlZ
n	0	1	2	3	4	5	6	7
A[i]	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	

3-2 開燈問題

有 n 盞燈，編號為 1~n。第 1 個人把所有燈打開，第 2 個人下所有編號為 2 的倍數的開關(這些燈將被關掉)第 3 個人按下所有編號為 3 的倍數的開關(關掉的燈將被打開，開著的燈將被關掉)依此類推。一共有 k 個人，問最後有哪些燈開著?

輸入：n 和 k，輸出：開著的燈的編號(k<= n <= 1000)。
範例輸入： 7 3
範例輸出： 1 5 6 7

解析：用 a[1], a[2], a[3], …, a[n] 表示編號為 1,2,3,…,n 的燈是否開著。




```
#include <stdio.h>
#include <string.h>
#define maxn 1010
int a[maxn];
int main()
{
    int n, k;
    memset(a, 0, sizeof(a));
    scanf("%d%d", &n, &k);
    for (int i=1; i<=k; i++)
        for (int j=1; j<=n; j++)
            if(j%i==0) a[j] = !a[j];
    for (int i=1; i<=n; i++)
        if (a[i]) printf("%d", i);
    printf("\n");
    return 0;
}
```

例題 3-1 TeX 中的引號(Tex Quotes, Uva 272)

TeX 是一種由 Donald Knuth 所發展出的一套文書排版軟體。這套軟體可以將原始文件檔加上一些像字型等型態後，轉成一份很漂亮的文件。而一份漂亮的文件是需要用 `` 和 " 來把別人說的話給「引」出來，而不是用大部份鍵盤上有的 " 。

如果原始文件檔內容是：

```
"To be or not to be," quoth the bard, "that is the question."
```

我們需要把原始文件變成這個樣子：

```
``To be or not to be,'' quoth the bard, ``that is the question.''
```

關鍵：

1. **如何判斷是左引號還是右引號**
2. **如何輸入字串**

```
// UVa272 Tex Quotes
// Rujia Liu
#include<stdio.h>
int main() {
  int c, q = 1;
  while((c = getchar()) != EOF) {
    if(c == '"') { printf("%s", q ? "``" : "''"); q = !q; }
    else printf("%c", c);
  }
  return 0;
}
```

解析：

```
if(c == '"') { printf("%s", q ? "``" : "''"); q = !q; }
    else printf("%c", c);
```

例題 3-2 WERTYU (UVa10082)

打字時一個常見的錯誤就是沒有把手放在正確位置，而是偏右邊一個位置。所以會發生 Q 被打成 W，J 被打成 K 等等的情況。你的任務就是要把打錯的字修正回來。

範例輸入

```
S, GOMR YPFSU/
URD. ,U [JPMR MI,NRT OD 8346333
```

範例輸出

```
I AM FINE TODAY.
YES, MY PHONE NUMBER IS 7235222
```

解析：

```
#include<stdio.h>
char s[] = "`1234567890-=QWERTYUIOP[]\\ASDFGHJKL;'ZXCVBNM,./";
int main() {
  int i, c;
  while((c = getchar()) != EOF) {
     for (i=1; s[i] && s[i]!=c; i++);  // 找錯位之後的字元在常量表中的位置
     if (s[i]) putchar(s[i-1]);  // 如果找到，則輸出它的前一個字元
     else putchar(c);
  }
  return 0;
}
```

重點：

getchar()
putchar()
EOF

例題 3-3 迴文(Palindromes, UVa401)

迴文的定義為正向，反向讀到的字串均相同.
如：abba , abcba ... 等就是迴文
請判斷一個字串(長度 < 1000)是否是一個迴文？
範例輸入 ：
abba
Abcd
範例輸出 ：
yes
no

```
#include<stdio.h>
#include<string.h>
int main()
{
    char s[30];
    while(scanf("%s", s) == 1)
    {
        int len = strlen(s);
        int p = 1;
        for(int i = 0; i < (len+1)/2; i++)
            if (s[i] != s[len-1-i]) p = 0;  // 不是回文串
        if (p) printf("yes\n");
        else printf("no\n");
    }
    return 0;
}
```

```
char s[30];
scanf("%s", s);
strlen(s)
```

p 的功能是什麼？

例題 3-4 猜數字遊戲提示(Master-Mind Hints, Uva 340)

實現一個經典"猜數字"游戲。
給定答案序列和用戶猜的序列，統計有多少數字位置正確（A），有多少數字在兩個序列都出現過但位置不對（B）。
輸入包含多組數據。
每組輸入第一行爲序列長度 n，
第二行是**答案序列**，接下來是若干**猜測序列**。
猜測序列全 0 時該組數據結束。
n=0 時輸入結束。

範例輸入：

```
4
1 3 5 5
1 1 2 3
4 3 3 5
6 5 5 1
6 1 3 5
1 3 5 5
0 0 0 0
10
1 2 2 2 4 5 6 6 6 9
1 2 3 4 5 6 7 8 9 1
1 1 2 2 3 3 4 4 5 5
1 2 1 3 1 5 1 6 1 9
1 2 2 5 5 5 6 6 6 7
0 0 0 0 0 0 0 0 0 0
0
```

範例輸出：

```
Game 1:
(1,1)
(2,0)
(1,2)
(1,2)
(4,0)
Game 2:
(2,4)
(3,2)
(5,0)
(7,0)
```

```
1 3 5 5
1 1 2 3

a = 1  (數字相同位置相同個數)
b = 2  (數字相同個數)
xAxB = aA (b-a)B
```

```
#include<stdio.h>
#define maxn 1000 + 10
int main() {
  int n, a[maxn], b[maxn];
  int kase = 0;
  while(scanf("%d", &n) == 1 && n) {     // n=0 時輸入結束
    printf("Game %d:\n", ++kase);
    for(int i = 0; i < n; i++) scanf("%d", &a[i]);
    for(;;) {
      int A = 0, B = 0;
      for(int i = 0; i < n; i++) {
        scanf("%d", &b[i]);
        if(a[i] == b[i]) A++;
      }
   if(b[0] == 0) break;   // 正常的猜測序列不會有 0，所以只判斷第一個數是否爲 0 即可
      for(int d = 1; d <= 9; d++) {
        int c1 = 0, c2 = 0;    // 統計數字 d 在答案序列和猜測序列中各出現多少次
        for(int i = 0; i < n; i++) {
          if(a[i] == d) c1++;
          if(b[i] == d) c2++;
        }
        if(c1 < c2) B += c1; else B += c2;
      }
      printf("    (%d,%d)\n", A, B-A);
    }
  }
  return 0;
}
```

https://github.com/aoapc-book/aoapc-bac2nd/tree/master/ch3

字元與字串

資料來源：http://www.tcgs.tc.edu.tw/~sagit/cpp/q5.htm

一、字元與 ASCII 碼

一個英文字母、數字或其他的符號，我們稱它為字元。要表示一個字元，我們可以用一對單引號 ' 把該字元夾起來，例如：

```
char c='a';
```

而要輸入一個字元，有下列四種方式：

```
char c;
scanf("%c", &c);      // C
c=getc(stdin);        // C
cin >> c;             // C++
cin.get(c);           // C++
```

上面的 getc() 函數可以從檔案讀取一個字元，而 stdin 則是標準輸入(Standard Input)，也就是從鍵盤讀取。(可被重新導向)

而要印出一個字元，則可以用下面的方式：

```
char c='a';
printf("%c", c); // C
putc(c, stdout); // C
cout << c; // C++
cout.put(c); // C++
```

上面的 stdout 同樣是標準輸出(Standard Output)，也就是從螢幕輸出。(可被重新導向)

事實上，字元在電腦中是以一個八位元的整數來儲存(即 1 Byte)，而這個符號與數字的對應關係我們稱為 ASCII 碼 (American Standard Code for Information Interchange)，例如：

數字	符號	數字	符號	數字	符號
48	0	65	A	97	a
49	1	66	B	98	b
50	2	67	C	99	c
51	3	68	D	100	d
52	4	69	E	101	e
…	…	…	…	…	…

也就是說，其實字元也是一個數字，而大小寫字母相差 32 (小寫字母比較大)。而我們也可以把字元拿來做加減乘除等四則運算。例如

```
char c='a';
```

```
c+=3;
cout << c;
```

上面的例子會印出字元 d。至於下面的程式可以印出 A 到 Z 及它們的 ASCII 碼值：

```
char c;
for(c='A'; c<='Z'; c++) {
  cout << c << " " << int(c) << endl; // C++
  printf("%c %d\n", c, c); // C
}
```

除此之外，C/C++ 裡還有一些特殊字元，通常以反斜線 \ 開頭：

'\0' – 空字元，用於字串的結束，它的值也就是 0

'\n' – New Line，換行符號

'\r' – Carriage Return，回歸鍵(即 Enter 鍵)

'\t' – Tab，跳格

'\b' – Backspace，倒退鍵

'\a' – Bell，嗶一聲

'\\' – 反斜線 \

'\'' – 單引號 '

'\"' – 雙引號 "

至於要判斷一個字元是數字或大小寫字母，除了用 c>='A' && c<='Z' 之類的寫法外，C/C++ 提供了一些字元處理的函數，不過要使用它們要先 #include <ctype.h>，這些函數如下：

isdigit(c) – 判斷 c 是否為數字

islower(c) – 判斷 c 是否為小寫字母

isupper(c) – 判斷 c 是否為大寫字母

isalpha(c) – 判斷 c 是否為字母

isalnum(c) – 判斷 c 是否為字母或數字

二、字串

接下來，我們討論到字串，字串就是一段文字，我們可以用一對雙引號 " 把該段文字夾起來。由於 C 中並沒有字串的變數型態(PS. C++ 中有 string 的字串型態)，而是用字元的陣列來儲存一個字串，我們先宣告字串的長度：

```
#define LEN 80
```

再來宣告字串變數：

```
char s[LEN], t[LEN]="TCGS";
```

上面的例子宣告了 s、t 為字串，長度為 LEN 個字元。我們在宣告字串的時候要注意它的長度，以免位數不夠造成程式錯誤。事實上，每個字串後面都有一個 '\0' 的字元，也就是說，上面的 "TCGS" 字串，事實上總共用了 5 Bytes，這一點要特別注意。我們用下面的表格說明：

t[0]	t[1]	t[2]	t[3]	t[4]	t[5]	t[6]	t[7]	t[8]	t[9]
'T'	'C'	'G'	'S'	'\0'	?	?	?	?	?

而我們要輸入和輸出一個字串可以用下面的方式：

```
char s[LEN];
scanf("%s", s); // C
printf("%s\n", s); // C
cin >> s; // C++
cout << s << endl; // C++
```

比較特別的是，因為 s 本身就是字串的位址，所以使用 scanf() 讀取字串時在 s 前面不用加上 & 符號。

要注意的是，上面的方式讀到空白就會停止，例如我們輸入的是 "Hello, TCGS!" 則只有 "Hello," 被讀進字串 s 裡。而要避免這種情況，我們可以使用整行讀取的方式：

```
fgets(s, LEN, stdin); // C
cin.getline(s, LEN); // C++
```

上面的方式， fgets() 會保留後面的換行符號 ('\n')，而 cin.getline() 則不會，這點要注意。

因為字串是字元的陣列，所以如果我們要取得某一個字元，可以用 s[i] 的方式取得，例如下面的程式可以輸入一行文字後把它從後面反過來輸出：

```
char s[LEN];
int i, len;
cin.getline(s, LEN);
len=strlen(s);
for(i=len-1; i>=0; i--) cout << s[i];
cout << endl;
```

上面我們用了一個 strlen() 的函數，它的功能是傳回該字串的長度。要使用這個函數要先 #include <string.h>。而 string.h 裡還有許多常用的字串函數，如下：

```
strlen(s1) – 傳回 s1 的長度(不含 '\0' 字元)
strcpy(s1, s2) – 將 s2 的內容複製到 s1
strcmp(s1, s2) – 比較 s1、s2 的內容，如果相等傳回 0
strcat(s1, s2) – 將 s2 串接到 s1 後面
strstr(s1, s2) – 傳回 s2 字串在 s1 字串中第一次出現的位置
strrev(s1) – 將 s1 字串倒置 (非標準)
```

三、編碼問題

在第二次世界大戰中，德軍的通訊編碼被美國破解，以致於機密被美國竊聽而慘敗。德軍的編碼規則(假)如下：將訊息每個字母往後推兩位再傳出去，例如 A→C、B→D，而後面的 Y→A、Z→B，所有的訊息都是大寫字母，而空白以及其他的符號則維持不變。例如 "I LOVE TCGS." 則會變成 "K NQXG VEIU."。

假設你是美軍的情報員，收到德軍編碼過的訊息，要快速地將它解碼回原來的訊息，你可以完成這個任務嗎？

輸入 1：K NQXG VEIU.
輸出 1：I LOVE TCGS.
輸入 2：PRUE 2005 KU EQOKPI UQQP.
輸出 2：NPSC 2005 IS COMING SOON.

我們可以一個字元一個字元讀進來，如果是大寫字母，則把它減 2，如果減 2 之後比 A 還小，則加上 26，然後把它印出來。如果不是字母，則直接輸出即可。程式如下：

```
char c;
while (1) {
  cin.get(c);
  if( cin.fail() ) break;
  if( (c>='A') && (c<='Z') ) {
    c-=2;
    if( c<'A' ) c+=26;
  }
  cout << c;
}
```

輸入結束請按 Ctrl+Z 再按 Enter。上面使用 cin.fail() 判斷是不是讀到結束了。如果是使用 C 的話，可以改成：

```
c = getc(stdin);
if( c==EOF ) break;
```

你也可以把上面的程式改成輸入原本的文字，而輸出編碼過的文字。

四、練習題

1. 基礎：Q458 – The Decoder
2. 基礎：Q10082 – WERTYU
3. 基礎：Q10222 – Decode the Mad man
4. 基礎：Q272 – TeX Quotes
5. 綜合：Q494 – Kindergarten Counting Game
6. 綜合：Q10340 – All in All

string 與 stringstream

資料來源：http://www.tcgs.tc.edu.tw/~sagit/cpp/q6.htm

一、string

string 是 C++ 提供的字串型態，和 C 的字串相比，除了有不限長度的優點外，還有其他許多方便的功能。要使用 string， 必須先加入這一行：

```
#include <string>
```

接下來要宣告一個字串變數，可以寫成：

```
string s;
```

我們也可以在宣告的同時讓它設成某個字串：

```
string s="TCGS";
```

而要取得其中某一個字元，和傳統 C 的字串一樣是用 s[i] 的方式取得。比較不一樣的是如果 s 有三個字元，傳統 C 的字串的 s[3] 是 0 字元，但是 C++ 的 string 則是只到 s[2] 這個字元而已。

下面我們把 string 與 字元陣列的語法做一個對照：

操作	string	字元陣列
宣告字串	`string s;`	`char s[100];`
取得第 i 個字元	`s[i]`	`s[i]`
字串長度	`s.length()` 或 `s.size()`	`strlen(s)`
讀取一行	`getline(cin, s);`	`gets(s);`
設成某字串	`s="TCGS";`	`strcpy(s, "TCGS");`
字串相加	`s=s+"TCGS";` `s+="TCGS"`	`strcat(s, "TCGS");`
字串比較	`s=="TCGS"`	`strcmp(s, "TCGS")`

從上面的表格，我們可以發現 string 的用法比較直覺，因此如果沒有特別的需要，儘量使用 string 會比較方便。

二、stringstream

stringstream 是 C++ 提供的另一個字串型的串流(stream)物件，和之前學過的 iostream、fstream 有類似的操作方式。要使用 stringstream， 必須先加入這一行：

```
#include <sstream>
```

stringstream 主要是用在將一個字串分割，可以先用 clear( ) 以及 str( ) 將指定字串設定成一開始的內容，再用 >> 把個別的資料輸出，例如：

```
string s;
stringstream ss;
int a, b, c;
getline(cin, s);
ss.clear();
ss.str(s);
ss >> a >> b >> c;
```

下面我們看到一個使用 stringstream 的例子：

題目：輸入的第一行有一個數字 N 代表接下來有 N 行資料，每一行資料裡有不固定個數的整數(最多 20 個，每行最大 200 個字元)，請你寫一個程式將每行的總和印出來。

輸入：

```
3
1 2 3
20 17 23 54 77 60
111 222 333 444 555 666 777 888 999
```

輸出：

```
6
251
4995
```

程式如下：

```
string s;
stringstream ss;
int n, i, sum, a;
cin >> n;
getline(cin, s); // 讀取換行
for (i=0; i<n; i++)
{
    getline(cin, s);
    ss.clear();
    ss.str(s);
    sum=0;
    while (1)
    {
```

```
        ss >> a;
        if ( ss.fail() ) break;
        sum+=a;
    }
    cout << sum << endl;
}
```

三、C 字串分割函數 — strtok( )

strtok( ) 函數是另一個字串分割的選擇，它可以將原本的 C 傳統字串依指定的字元把它們分割成兩個子字串，我們可以用 atoi(s) 、 atol(s) 、 atof(s) 等函數將切出來的字串轉換成整數、長整數、浮點數(double)等 (PS.要 #include <stdlib.h>)。

我們先宣告字串長度：

```
#define LEN 200
```

主程式：

```
int n, i, sum;
char s[LEN], *p;
cin.getline(s, LEN);
n=atoi(s);
for(i=0; i<n; i++) {
  cin.getline(s, LEN);
  for (sum=0, p=strtok(s, " "); p; p=strtok(0, " "))
      sum+=atoi(p);
  cout << sum << endl;

}
```

strtok( ) 函數的第一個參數是要切割的字串，第二個參數則是分割字元。如果成功，則傳回切割出來的第一個字串的位址，如果失敗，則傳回 0。而第二次使用 strtok() 時，則第一個參數要用 NULL (0) ，這樣它就會繼續上一次的字串切割，說明如下：

1.呼叫 strtok() 之前 (s="I LOVE U")：

s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]	s[8]
'I'	' '	'L'	'O'	'V'	'E'	' '	'U'	'\0'

2.執行 p=strtok(s, " ")：

s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]	s[8]
'I'	'\0'	'L'	'O'	'V'	'E'	' '	'U'	'\0'

p 為 s[0] 的位址，strtok() 記錄 s[2] 的位址

3.再執行 p=strtok(0, " ")：

s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]	s[8]
'I'	'\0'	'L'	'O'	'V'	'E'	'\0'	'U'	'\0'

p 為 s[2] 的位址，strtok() 記錄 s[7] 的位址

4.再執行 p=strtok(0, " ")：

s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]	s[8]
'I'	'\0'	'L'	'O'	'V'	'E'	'\0'	'U'	'\0'

p 為 s[7] 的位址，strtok() 記錄 s[8] 的位址

從上面的過程中，我們可以知道原本的字串 s 會被切割成數個子字串，而這些子字串中間以 '\0' 分隔開。

四、練習題：Q482 – Permutation Arrays

輸入：

```
3 1 2
32.0 54.7 -2
```

輸出：

```
54.7
-2
32.0
```

說明：輸入兩行文字，第一行有 N 個整數，代表第二行中 N 個浮點數的順序，請把第二行的浮點數依序印出來。