字串與IO

by sa072686
sa072686@gmail.com

字元與 ASCII 編碼

文字如何儲存？轉成整數不就解決啦

字串基礎IO與更多應用

將零或多個字元當作一個單位使用, 就是字串

Warning : getline() 必學

除非你會 fgets() 或者 getchar() 或者 scanf("%c", ... 或者 cin >> noskipws ...等
只會 cin 不會 getline 或 noskipws 你可能有些題目幾乎不可能做得出來

Warning: 接下來的東西對練習賽很重要

雖然不會全部用上，但全部不會的話，搞不好有四題左右不容易滿分，除非…

好吧可能不必太擔心，因為...

C++ 函式庫有 = C++ 實作得出來 = 可以自己實作 = 不用會也可以活得好好的

C++ IO優化

C++ 才做優化，C 的 printf() / scanf() 不需要優化就足夠快

C++ IO 優化

```
int main()
{
    // 放在 main() 大括號內最一開始
    ios::sync_with_stdio(0);
    cin.tie(0);
    // 之後再寫你原本的程式碼
    // ...
}
```

這會加速非常非常多

如果你打算用 cin / cout 打練習賽，請一定要加，否則 T★L★E 概不負責

為何可以加速

- cin / cout 為了遷就 printf() / scanf() 所以變慢了
  - 主要是 BufferedIO 的關係
  - 重要：優化後就不能和 printf() / scanf() 混用
  - 實際上所有 C 能存取 stdin / stdout 的函數都不能混用了，例如 getchar() 等
- cin 和 cout 會互相影響到彼此，這也會拖慢速度
  - 所以 cin.tie(0) 會影響手動輸出輸入時的邏輯
- 詳細可自行 google 它們的用途或IO優化，或者看看這篇或這篇

實際看看變動

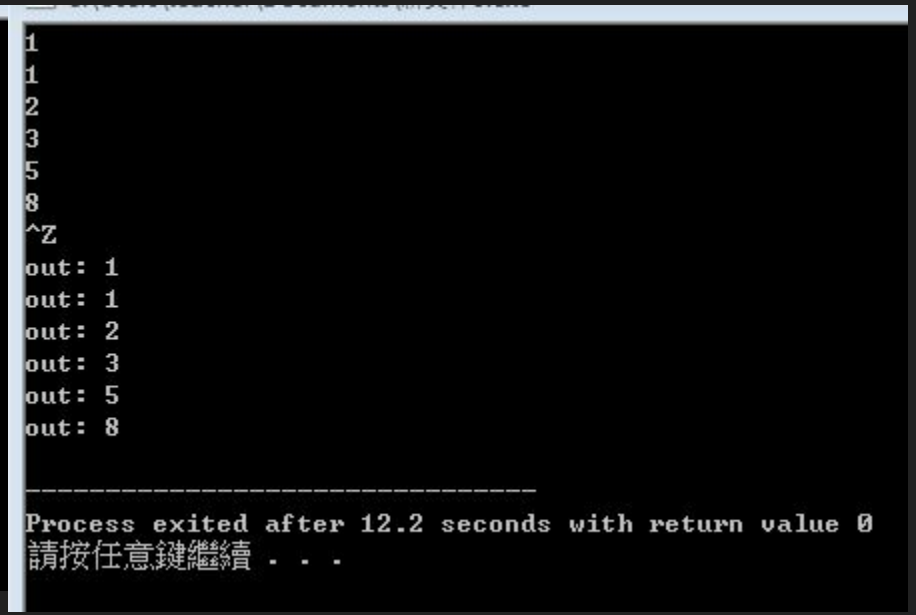
```
#include <iostream>
using namespace std;

int main()
{
    int a;
    while (cin >> a)
    {
        cout << "out: " << a << '\n';
    }
    return 0;
}
```

```
#include <iostream>
using namespace std;

int main()
{
    ios::sync_with_stdio(0);
    cin.tie(0);
    int a;
    while (cin >> a)
    {
        cout << "out: " << a << '\n';
    }
    return 0;
}
```

實際看看變動~~秒數不要在意那些都花在等待輸入~~



在程式結束前不會顯示輸出的東西

除非你主動要求…它存在，只是沒顯示給你看

多重輸入時可用 CTRL+Z 表示結束

而且就不會輸出輸入交錯在一起了！

輸出輸入格式控制

不是必備，但沒有它們的話自力實作會非常繁複~~雖然是不錯的練習和經驗~~

前置：載入控制 IO 用的函式庫

```
#include <iomanip>
```

浮點數篇

如何控制浮點數的表達

浮點數位數控制

- `cout << fixed << setprecision( 2 ) << 3.14159 << '\n';`
  - 四捨五入至小數點下第 2 位，2 可代換成其它整數常數、變數或「任意結果為整數的運算式」
- `cout << fixed << setprecision( 0 ) << 2.49 << '\n';`
- `cout << fixed << setprecision( 0 ) << 2.50 << '\n';`
- 設定後永久有效，直到更改或消除為止；只影響浮點數

科學記號表示

- `cout << scientific << 16.80 << ": " << 12 << '\n';`
- `cout << scientific << setprecision( 2 ) << 16.80 << '\n';`
- 設定後永久生效，會與 fixed 互相覆蓋，只對浮點數生效

自動補齊長度

太長的不會理你

輸出長度 k 靠右對齊

- `cout << setw( 5 ) << 123 << '\n';`
- `cout << setw( 5 ) << "ABC" << '\n';`
  - 設定下個東西輸出長度 5 靠右對齊，不足時以空白補齊，可對應各種不同型別
- `cout << setw( 2 ) << "ABC" << '\n';`
  - 當輸出長度超過設定長度時不會發生任何作用，也不會做出長度限制
- 設定後只對下個東西有效

輸出長度 k 靠右對齊，不足補份以 c 字元補齊

- `cout << setw( 5 ) << setfill( '0' ) << 123 << '\n';`
- `cout << setw( 5 ) << setfill( '?' ) << "ABC" << '\n';`
  - 設定下個東西長度 5 靠右對齊，不足時 以指定字元補齊
- 設定後永久有效，往後 setw() 不足時一律改以設定的字元補齊

應用：時間（例如 08:07）

- `cout << setw( 2 ) << setfill( '0' ) << 8 << ":";`
- `cout << setw( 2 ) << 7 << '\n';`

裏技. 製造重覆的字元

- `string s;`
- `s.resize(32, '#');`
- `cout << s << '\n';`
  - 注意 `resize` 只會對多出來的部份補 `'#'` 所以如果你一開始就有東西 ...
- `s = "ABC";`
- `s.resize(32, '!');`
- `cout << s << '\n';`
  - 星星樹表示

裏技. 輸出 k 個指定字元

- `cout << setw( 8 ) << setfill( '#' ) << "" << '\n';`
  - 因為空字串長度是 0, 所以會被補滿 8 個 '#' ......

裏技 v.s. 星星樹

```
int n;
cin >> n;
for (int space=n-1, star=1; space>=0; space--, star+=2)
{
    cout << setw( space ) << setfill( ' ' ) << "";
    cout << setw( star ) << setfill( '*' ) << "";
    cout << '\n';
}
```

八進位與十六進位

以為 C++ 願意幫你提供各種進位的轉換嗎？那你就錯了

讓輸出數字以八進位或十六進位表示

- `cout << setbase(8) << 12 << ": " << 12 << '\n';`
- `cout << setbase(10) << 12 << ": " << 12 << '\n';`
- `cout << setbase(16) << 12 << ": " << 12 << '\n';`
- `cout << setbase(2) << 12 << ": " << 12 << '\n';`
- 設定後永久有效，只對整數生效
- 只對 8, 10, 16 生效，其餘整數只會重設回預設值的十進位

讓輸出數字以八進位或十六進位表示

- `cout << oct << 12 << ": " << 12 << '\n';`
- `cout << dec << 12 << ": " << 12 << '\n';`
- `cout << hex << 12 << ": " << 12 << '\n';`
- 設定後永久有效，只對整數生效
- 如果你記得八、十、十六進位的英文，這組也許比較好記
  - 效果同 `setbase()`

輸入八進位或十六進位格式的數字

- `cin >> oct >> n;`
- `cin >> hex >> n;`
- 設定後永久有效，可用 `setbase()` 或 `dec` 設回十進位
- 十六進位可輸入 `0x` 開頭的格式，大小寫皆可，無需特別設定

讓十六進位以大寫顯示

- `cout << hex << 255 << '\n';`
- `cout << hex << uppercase << 255 << '\n';`
- `cout << nouppercase;`
- 設定後永久有效，只對系統自動轉換(例如十六進位)的字母有效
  - 直接命令它輸出一個含小寫字母的字串，是不會有任何改變的
- 用 `nouppercase` 可以取消此設定

字串整理

面對很難處理的東西，除了寫更複雜的規則，搞不好可以找到很簡潔的整理方式

運算式間可以有任意數量空白，也可以沒有

- `2+ 3   * ( 6 -8)  /(3+ 2)`
- 主要問題出在數量不固定，如果固定是 1 或 0 事情會簡單不少

把空白變成固定 1 很難，但全部消滅似乎...？

- string s, t;
- getline(cin, s);
- for (int i=0; i<s.size(); i++)
  - if (!isspace(s[i]))
    - t += s[i];
- cout << t << '\n';

萬惡的 stringstream

應對以行為單位、行內東西數量不固定但又不告訴你數量時，超強

stringstream 所在函式庫

```
#include <sstream>
```

用起來和 cin 幾乎一模一樣

- 對象從標準輸入（stdin）變成字串而已

如何使用

- stringstream ss;
- ss.str("8 7 6 3");
- int ary[128], i = 0;
- while (ss >> ary[i])
- {
  - cout << "ary[" << i << "] = " << ary[i] << '\n';
  - i++;
- }

重覆利用時

- 必須把 EOF 的 flag 清掉
- `ss.clear();`
  - 注意:這不會把 ss 內清空
- `ss.str("");`
  - 塞空字串重置內容

也可以用 cout 的方式餵東西

- `ss << 1 << " ABC";`
- `cout << ss.str() << '\n';`
  - `.str()` 不加參數時可取得 `ss` 內的文字, 不會改變內容

裏技. 拿來四捨五入至小數點下第 k 位

- `ss << fixed << setprecision( 3 ) << 3.14159;`
- `double f;`
- `ss >> f;`
  - `f = 3.14200`

scanf() && printf()

只放 C++ 似乎並不是很公平~~而且格式化字串那麼香~~

逐行輸入

- 由於 C++11 以後砍了超方便的 gets() ..
- fgets(buf, sizeof(buf), stdin);
  - 實際輸入字數會被限制為傳入的 size - 1，留一格放 '\0'
  - size 給不夠大的話，到換行前就會停下
  - 為了辨識是否有完整讀完一整行，換行字元會被存進來
- 特別要小心不能以 strlen(buf) == 0 判斷空字串
  - windows 還會以 \r\n 作為換行 ..

常見輸出格式

- `printf("%5d", a);`
  - 輸出長度不足 5 時以空白補齊，靠右對齊；同樣對長度 > 5 不起作用、不加限制
  - 也可用於 d 以外的東西，例如 %5s
- `printf("%02d", a);`
  - 輸出長度不足 2 時以 0 補齊，靠右對齊；同樣對長度 > 2 不起作用、不加限制
- `printf("%5.3f", f);`
  - 輸出長度 5 小數點下四捨五入至第 3 位的浮點數
  - 5 是總長度，不足時靠右對齊，超過時不加限制
- `printf("%.5s", "0123456789");`
  - 輸出字串前 5 個字；若字串長度不足 5 則直接輸出，不補齊
- `printf("%-5s]", "123");`
  - 輸出不足長度 5 時以空白補齊，靠左對齊；對長度 > 5 不起作用、不加限制
  -] 只是方便讓你看得到補齊的空白所佔的空間，並不是上述格式的一部份

進位制或浮點數相關

- `printf("%x", 255);`
- `printf("%X", 255);`
  - 輸出以十六進制表達，x 大小寫會影響輸出字母大小寫
- `printf("%o", 255);`
  - 輸出以八進制表達
- `printf("%e", 3.14159);`
  - 輸出以科學記號表達

浮動長度指定

- `printf("%*d", 5, 123);`
- `printf("%0*d", 2, 8);`
- `printf("%.*s", 3, "ABCDEFG");`
- `printf("%.*f", 3, 3.14159);`
- 在指定長度的地方不放數字改放 * 則可用參數指定長度
  - 因為 * 比起 d 或 f 這些要來得早出現，因此指定它們的參數也必須放前面

輸入常見格式

- `scanf("%*d");`
  - 輸入一個 %d 但是不做任何記錄、不保存
- `scanf("%5d", &a);`
  - 輸入一個整數，最長只讀入 5 個數字，剩下保留不會被讀掉
- `scanf("%d:%d", &minute, &second);`
  - 輸入一個整數、一個 : 、一個整數，任何一個地方比對失敗都會中途結束掉
- `scanf("%[ABC]", &s);`
  - 輸入只能包含 ABC 三種字元的字串，遇到不是的就會停下
- `scanf("%[0-9]", &s);`
  - 輸入只能包含 0 ~ 9 的字串，0-9代表介於 0 和 9 之間
- `scanf("%[^ABC]", &s);`
  - 在 [] 內第一個字放 ^ 代表 [] 中的字元是被排除的字元
  - 所以這樣意思是:輸入一個不包含 ABC 三種字元的字串，遇到 ABC 其中之一就會停下

不常見但意外實用的 %n

- %n 不會從輸入讀取任何東西，只會儲存 %n 位置離這次 scanf() 起點有多遠
- scanf("%d%n", &a, &b);
  - 輸入 "123" 則 b = 3
  - 輸入 " 123" 則 b = 5
- 注意它會把跳過的空白和換行也算在內，而通常離上次輸入會隔空白或換行
- 不會被計入「讀取到且被成功儲存」的計數內
  - 上述的 scanf() 最多回傳 1，因為 %n 不算

scanf() 回傳值

- 意義為成功讀入並成功儲存多少個東西
- `int res = scanf("%d %d", &a, &b);`
  - 輸入為 "A" 讀不到任何整數, 回傳 0
  - 輸入為 "1" 只能讀入一個整數, 回傳 1
  - 輸入為 "1 A" 只能讀入一個整數, 回傳 1
  - 輸入為 "2 3" 可讀入兩個整數, 回傳 2
  - 輸入為 "4 5 6" 只會讀入前兩個整數, 回傳 2
  - 如果 EOF 回傳 -1

sprintf() && sscanf()

以字串為目標做輸出或輸入，和 stringstream 微妙地不同

輸出至字串上

- `char buf[128];`
- `sprintf(buf, "%d => %d", a, b);`
- 常用在給好樣板，想照輸入填上時；例如
- `sprintf(msg, "get item [%s] x %d!!", item_name, item_count);`

從字串輸入

- `sscanf(buf, "%d %d", &a, &b);`
- 多用於 `getline` 讀入一整行後，從裡面輸入內容

混合技:輸入單行空白分隔的未知個數的整數

- sscanf() **不會記憶上次讀到哪**, 所以用 char* 加上 %n 手動推進
  - 畢竟 s.c_str() 或者 char[] 是不能推進的
- getline(cin, s);
- const char *ptr = s.c_str();
- int pos, res, i=0;
- while (sscanf(ptr, "%d%n", &ary[i], &pos) == 1)
- {
  - ptr += pos;
  - i++;
- }

注意 : 應用時先輸入整行再 sscanf()

比較不會影響後續, 東西沒全讀完也不用在意輸入的遊標停在哪

混用的時候怎麼辦

s.c_str() 可從字串取得 C 型態的字串（不可修改內容）來使用

各種奇形怪狀的應用例

應用：C++ 難以處理的樣板置換

- `const char *pat = "get item [%s] x %d!"`
- `sprintf(msg, pat, "potion", 3);`
  - `get item [potion] x 3!`
- `pat = "[%s] x %d got!!";`
- `sprintf(msg, pat, "potion", 3);`
  - `[potion] x 3 got!!`

應用：C++ 難以處理的樣板置換

- `char pat[128];`
- `while (scanf("%s", &pat) == 1)`
- `{`
  - `sprintf(msg, pat, "potion", 3);`
  - `puts(msg);`
- `}`

混合技：動態指定 scanf() 讀入長度

- 自己的格式化字串自己造
- `sprintf(pat, "%%%dd", k);`
  - 如果 `k = 5` 則 `pat = "%5d"`
- `scanf(pat, &a);`
- 同理可以用來生成 `sprintf()` 用的格式化字串

應用：輸入 xxx: ddd 的類型

- 例如 "money: 50" 或者 "str: 99" 之類的
- 核心：讀入非 : 後遊標會停在 : 上面，把 : 讀掉就只剩整數部份了
  - `sscanf(s, "%*[^:]:%d", &a);`

進階應用:數字字串中撈出位置 i 開始長度 k 的數字

```
int get(string &s, int i, int k)
{
    int res;
    char buf[128];
    sprintf(buf, "%%%dd", k);
    sscanf(s.c_str()+i, buf, &res);
    return res;
}
```

混合技：ungetc()

- 當你想看下一個字，又怕看了會後悔時...
  - 例：2+3*5 or 2+(3*5)
  - 需要先往下看是不是括號，但如果是數字則可能是 1 位數可能更多
    - 1 位數的情形，往下看又會拉到 * 會很頭痛
- 如果會後悔~~一開始就別做~~何不反悔呢
  - `int ch = getchar();`
  - `if (ch == '(')`
    - `// ...`
  - `else // 偷看完後悔了，再塞回去`
    - `ungetc(ch, stdin);`

進階應用：判斷數字個數 UVa598

- 核心：整數可能是 0 或 1 或 2 個
  - 那就命令讀 2 個，至於成功幾個丟給 sscanf() 去判斷，坐等結果就好
- getline(cin, s);
- int cnt = sscanf(s.c_str(), "%d%d", &a, &b);
- if (cnt == 0)
  - p = 1, q = n;
- else if (cnt == 1)
  - p = 1, q = a;
- else if (cnt == 2)
  - p = a, q = b;

進階應用：文字判別 UVa10473

- 核心：0x 的有無可供判別進位制種類
  - 那就規定 scanf() 開頭必須是 0x 看讀不讀得到，比對失敗就會直接停下回傳 0 了
  - 非空白、非 % 開頭特殊記號，會和當前位置的文字進行比對
- while (cin >> s)
- {
  - int cnt = sscanf(s.c_str(), "0x%X", &num);
  - if (cnt == 1)
  - {
    - printf("%d\n", num);
    - continue;
  - }
  - sscanf(s.c_str(), "%d", &num);
  - printf("0x%X\n", num);
- }

進階應用：字串判讀 UVa 10194

- 核心：非 # 隊名 => # => 整數得分 => @ => ...
  - 小心 fgets() 最後會有換行，也必須排除掉；getline() 則無此問題
- while (fgets(s, sizeof(s), stdin))
- {
  - sscanf(s, "%[^#]#%d@%d#%[^#\n]", s0, &a, &b, s1);
- }

進階應用:字串判讀 UVa11148

- 核心:想要的東西必為數字開頭、空白結尾
  - 忽略掉非數字部份,取非空白部份即為所求;為防數字開頭因此在開頭塞空白
- `s[0] = ' ';`
- `while (fgets(s+1, sizeof(s)-1, stdin)) {`
  - `int pos;`
  - `char *ptr = s;`
  - `while ( true ) {`
    - `int res = sscanf(ptr, "%*[^0-9]%[^ ]%n", buf, &pos);`
    - `if (res != 1)`
      - `break;`
    - `ptr += pos;`
  - `}`
- `}`

進階應用：字串判讀 UVa11148

- 核心二：必要部份有三種
  - 一、12
  - 二、12/34
  - 三、12-34/56
  - 由於二、三的第一個非數字字元不同，乾脆直接用 %c忽略掉，再判斷整數成功讀入個數
- `res = sscanf(buf, "%d%*c%d%*c%d", &a, &b, &c);`
- `if (res == 1) // 一、`
- `else if (res == 2) // 二、`
- `else if (res == 3) // 三、`

進階應用：連續數字分段輸入 TOJ 460

- 核心：連續十或十六進位整數，反正每位數是一個單位
  與其輸入完再用除法拆開，不如每次限制最大位數 1
- for (k=0; k<4; k++)
  - scanf("%1x", &chess[i][k]);
- 日期時間也可應用，例如日期 20200123
  - scanf("%4d%2d%2d", &year, &month, &day);

歡樂練習時間

- TOJ 104
- UVa 10473
- UVa 598
- UVa 825
- TOJ 460
- UVa 10194
- UVa 11148
- UVa 587

Q & A