

# NCKU Programming Contest Training Course

## Introduction & IO

### 2018/02/22

---

徐偉庭

*vtim99077@gmail.com*

Department of Computer Science and Information Engineering  
National Cheng Kung University  
Tainan, Taiwan



# Online Judge

---



# Online Judge

- POJ (PKU Online Judge)

- POJ : <http://poj.org>



- Uva Online Judge -> useful tool: uHunt

- UVa : <https://uva.onlinejudge.org>
- uHunt : <https://uhunt.onlinejudge.org>



# Problem format

## Description

問題描述

Calculate  $a+b$

## Input

輸入格式

Two integer  $a, b$  ( $0 \leq a, b \leq 10$ )

## Output

輸出格式

Output  $a+b$

## Sample Input

輸入範例

1 2

## Sample Output

輸出範例

3

## A+B Problem

Time Limit: 1000MS

Memory Limit: 10000K

Total Submissions: 342704

Accepted: 190305

時間、記憶體限制



# Result

Result	Memory	Time	Language	Code Length
Accepted	688K	0MS	G++	314B
Wrong Answer			C++	1849B
Compile Error			G++	319B
Accepted	4624K	94MS	G++	939B
Time Limit Exceeded			C++	2179B
Wrong Answer			C	300B
Wrong Answer			G++	1244B
Accepted	708K	0MS	G++	206B
Accepted	3528K	235MS	G++	1325B
Accepted	704K	0MS	G++	574B
Wrong Answer			C	389B
Wrong Answer			G++	1096B
Memory Limit Exceeded			G++	3073B
Accepted	732K	125MS	G++	1449B
Accepted	128K	0MS	C++	674B
Accepted	388K	79MS	G++	1288B
Presentation Error			C++	892B
Wrong Answer			G++	868B
Compile Error			G++	1285B
Accepted	664K	63MS	G++	428B



# Online Judge

- 常見 Online Judge (OJ):
  - POJ: 北京大學 - <http://poj.org/>
  - UVa: <https://uva.onlinejudge.org/>
  - ZOJ: 高中生程式解題 - <http://zerojudge.tw/>
  - SPOJ: <http://www.spoj.com/>



# I/O

---



# Standard Input & Output



標準輸入 → 由鍵盤輸入

標準輸出 → 由螢幕輸出

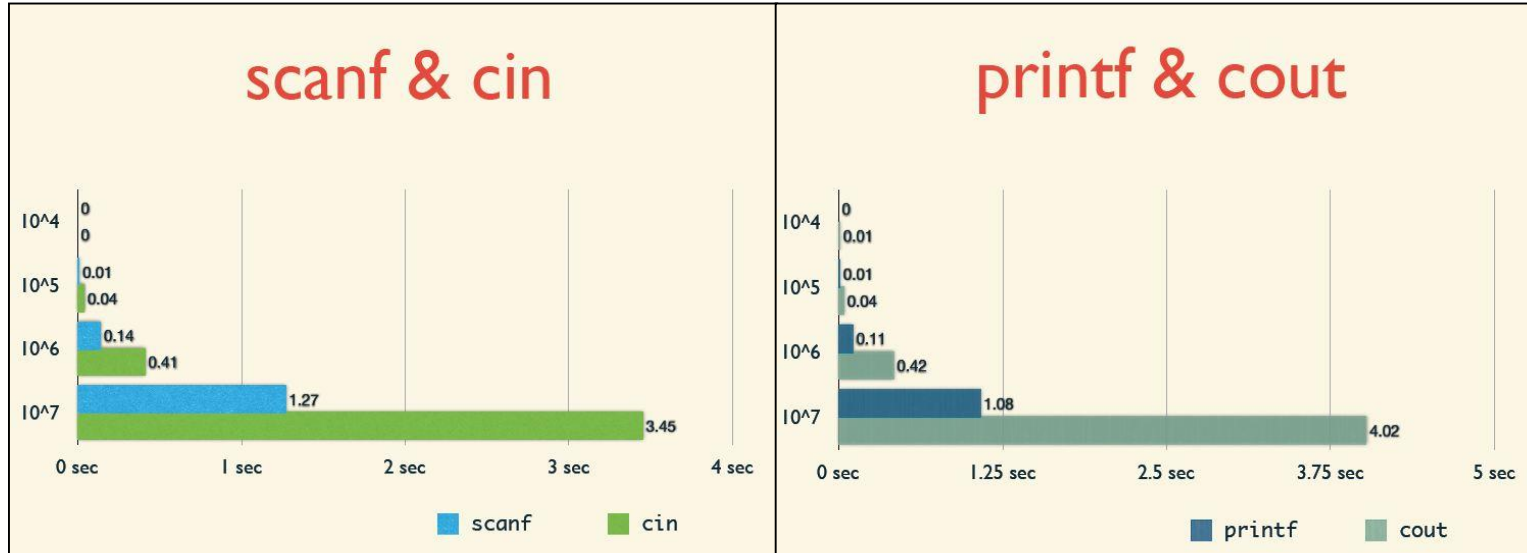




# I/O for contest

- Input  
scanf, gets, getchar, cin.....
- Output  
printf, puts, putchar, cout.....

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



# Single Test Case

- A + B problem

Sample Input

1 2

Sample Output

3



# Single Test Case

- A + B problem

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

int main() {
    int a,b;
    cin >> a >> b;
    cout << a+b << endl;
    return 0;
}
```



# Multiple Test Case

## Sample Input

```
95.123 12
0.4321 20
5.1234 15
6.7592 9
98.999 10
1.0100 12
```

## Sample Output

```
548815620517731830194541.899025343415715973535967221869852721
.00000005148554641076956121994511276767154838481760200726351203835429763013462401
43992025569.928573701266488041146654993318703707511666295476720493953024
29448126.764121021618164430206909037173276672
90429072743629540498.107596019456651774561044010001
1.126825030131969720661201
```



# Multiple Test Case (Input)

## 1. Given the number of test cases

A+B Problem

[Sample Input]

3

1 2

3 4

0 4

[Sample Output]

3

7

4



# Multiple Test Case (Input)

## 1. Given the number of test cases

```
/* Given Test cases # */
```

```
...
```

```
int tc,a,b;
```

```
scanf("%d",&tc);
```

```
while(tc--) {  
    scanf("%d%d",&a,&b);  
    printf("%d\n",a+b);  
}
```

```
...
```

A+B Problem

[Sample Input]

3

1 2

3 4

0 4

[Sample Output]

3

7

4



# Multiple Test Case (Input)

## 2. Terminated by special values

Hi, "input #".

[Sample Input]

30

10

25

0

[Sample Output]

Hi, 30.

Hi, 10.

Hi, 25.



# Multiple Test Case (Input)

## 2. Terminated by special values

```
/* Until zero */  
...  
int n;  
while(scanf("%d",&n)==1 && n) {  
    printf("Hi, %d.\n",n);  
}  
...
```

Hi, "input #".

[Sample Input]

30  
10  
25  
0

[Sample Output]

Hi, 30.  
Hi, 10.  
Hi, 25.





# Multiple Test Case (Input)

## 3. Terminated by EOF signal

- 若題目未指定終止條件，則為判斷 EOF 為終止條件

Hi, "input #".

[Sample Input]

30  
10  
25

[Sample Output]

Hi, 30.  
Hi, 10.  
Hi, 25.



# Multiple Test Case (Input)

## 3. Terminated by EOF signal

```
/* Until EOF */  
...  
int n;  
while(scanf("%d",&n) != EOF) {  
    printf("Hi, %d.\n",n);  
}  
...
```

Hi, "input #".

[Sample Input]

30  
10  
25

[Sample Output]

Hi, 30.  
Hi, 10.  
Hi, 25.



# Multiple Test Case (Input)

## 3. Terminated by EOF signal

scanf

```
while (scanf() != EOF)
{
    ...
}
```

fgets

```
while (fgets() != 0)
{
    ...
}
```

cin

```
while (cin >> x)
{
    ...
}
```



# Multiple Test Case (Output)

## 1. Blank line after all cases

A+B Problem

[Sample Input]

1 2

3 4

0 4

[Sample Output]

Case 1: 3

Case 2: 7

Case 3: 4



# Multiple Test Case (Output)

## 1. Blank line after all cases

```
/* \n\n */  
...  
int a,b,cs=1;  
while(scanf("%d%d",&a,&b)!=EOF){  
    printf("Case %d: %d\n\n",cs++,a+b);  
}  
...
```

### A+B Problem

[Sample Input]

1 2  
3 4  
0 4

[Sample Output]

Case 1: 3  
  
Case 2: 7  
  
Case 3: 4



# Multiple Test Case (Output)

## 2. Separated by blank line

A+B Problem

[Sample Input]

1 2  
3 4  
0 4

[Sample Output]

Case 1: 3

Case 2: 7

Case 3: 4



# Multiple Test Case (Output)

## 2. Separated by blank line

```
/* Separated */  
...  
int a,b,cs=1;  
while(scanf("%d%d",&a,&b)!=EOF){  
    if(cs>1) putchar("\n");  
    printf("Case %d: %d\n",cs++,a+b);  
}  
...
```

### A+B Problem

#### [Sample Input]

```
1 2  
3 4  
0 4
```

#### [Sample Output]

```
Case 1: 3  
Case 2: 7  
Case 3: 4
```



# Cutting Skill

- String Token

[Sample Input]

Electron ICPC  
kk free999 kevinx6000

[Sample Output]

2: Electron & ICPC  
3: kk & free999 & kevinx6000





# Cutting Skill

- strtok

```
char* strtok( char *str, const char *delimiters );
```



欲切割字串



分隔字符字串

return value : 指向當前切割字串之指標，若切割完畢  
則回傳 NULL。



# Cutting Skill

- String Token

```
#include<cstring>
```

```
...
```

```
gets(str);
```

```
char token[]=" ",*ptr;
```

```
for( ptr=strtok(str,token); ptr; ptr=strtok(NULL,token) )
```

```
{
```

```
    /* ptr is one token */
```

```
}
```

```
...
```

[Sample Input]

Electron ICPC

kk free999 kevinx6000

[Sample Output]

2: Electron & ICPC

3: kk & free999 & kevinx6000



# File I/O

- File I/O: freopen

```
/* freopen */  
...  
freopen("f1.in", "r", stdin);  
freopen("f1.out", "w", stdout);  
while(scanf(...)!=EOF){  
    printf(...);  
}  
...
```



# Vim

---





- 常用指令

- /<string> – 搜尋
  - i 找下一個
  - I 找上一個
- u – undo
- v – 選取文字
- y – 複製
- p – 貼上
- d – 刪除
- :new – 新視窗(水平分割)
- :vnew – 新視窗(垂直分割)
- :sp – 水平分割開啟現有或指定檔案
- :vsp – 垂直分割開啟現有或指定檔案



The image displays a Java code editor with a complex flow diagram overlaid. The code is as follows:

```

6 private boolean foo;
private boolean ReallyLongName;
private boolean aReallyShortName;
private int bar1, bar2;

private boolean isFooAndBar(){
    foo = false;
}

{
    public void main(String args[]){
        foo = true;
        if(foo){
            bar1 = bar2 + 1 + fooClass.invokeRandomMethod();
            bar1 = bar2 + 2;
            bar1++;
            bar2++;
            if( aReallyLongName
                aReallyLongName
                aReallyShortName
            )
        }
    }
}

```

The flow diagram consists of various colored arrows and labels indicating execution paths and control flow. Key labels include:

- h, k, l, j**: A small graph at the top right.
- W**: A yellow arrow pointing from the `bar1` variable to the `isFooAndBar` method call.
- %**: Multiple orange circular labels indicating branching points.
- S**: A yellow label pointing to the `fooClass.invokeRandomMethod()` call.
- fi**: A blue label pointing to the `bar1 = bar2 + 2;` line.
- b, w**: Yellow arrows pointing from the `bar1++` and `bar2++` lines.
- Ctrl-N**: A pink label pointing to the `if` statement.
- gd**: A blue label pointing to the `aReallyLongName` variable.
- Ctrl-W p, Ctrl-W j, Ctrl-W k**: Red labels pointing to the `if` statement and its branches.
- Ctrl-F, Ctrl-B**: Green labels pointing to the `if` statement and the `bar1` variable.
- M, zz**: Green labels pointing to the `if` statement and the `bar1` variable.
- L, zb**: Green labels pointing to the `if` statement and the `bar1` variable.

The bottom of the image shows a file explorer with the following files:

- `Sample.java` [23,29] 83%
- `AnotherNew.java` 3,0-1
- `Another.java` 3,0-1

字元(character)

h j k l

← ↓ ↑ →

單字(word)

w b 前/後個單字

W B 前/後個單字(跳過符號)

e 單字尾端

行(line)

0 行頭 \$ 行尾

^ 行頭(非空白字元)

段落(paragraph)、區塊(block)

{ 上一段 } 下一段

[{ 區塊頭 ]} 區塊尾

% 對應括號

螢幕(screen)、檔案(file)

H 螢幕頂端 Zt 捲至頂端

M 螢幕中間 ZZ 捲至中間

L 螢幕底部 zb 捲至底部

C-B 上一頁 C-F 下一頁

gg 檔頭 G 檔尾

mx 標記x 'x 跳至標記x

搜尋(search)

\* # 向後/向前搜尋目前單字

fx 向後搜尋字元x

gd 跳至目前單字的定義位置

/xxx 搜尋xxx

n N 下/上一個搜尋結果

<b>[ESC]</b>	<b>[C-]</b>	進入normal mode
<b>v</b>		進入visual mode
<b>V</b>		進入visual line mode
<b>[C-v]</b>		進入visual block mode
<b>i</b>		進入insert mode
<b>R</b>		進入replace mode
<b>a</b>		在游標後附加
<b>A</b>		在行末附加
		動作指令
<b>y</b>		複製(範圍)
<b>d</b>		刪除/剪下(範圍)
<b>c</b>		修改(範圍)
<b>x</b>		刪除/剪下(字元)
<b>D</b>		刪除至行末
<b>C</b>		修改至行末
<b>p</b>		貼上
<b>J</b>		和下一行合併
<b>r</b>		替換(字元)
<b>&gt;</b>		縮排
<b>&lt;</b>		反縮排
<b>.</b>		重複上一命令
<b>u</b>		回復上一命令
		EX指令
<b>:w</b>		儲存(:wq 儲存並退出)
<b>:q</b>		退出(:q! 強制退出)
<b>:e x</b>		編輯檔案x
<b>:n</b>		開新文件
<b>:h</b>		呼叫vim help
<b>:xx</b>		跳至xx行

<b>自動補齊 [insert mode]</b>		
<b>C-N</b>	<b>C-P</b>	自動補齊下/上個可能字
<b>C-X</b>	<b>C-F</b>	自動補齊可能檔名
<b>分割視窗(split window)</b>		
<b>vsp</b>	<b>sp</b>	垂直/水平分割視窗
<b>diffs</b>		分割視窗並比較(diff)檔案
<b>C-W p</b>		(來回)跳至前一個分割視窗
<b>C-W w</b>		跳至下一個分割視窗

# Vim

- vimrc 設定教學
  - <http://wiki.csie.ncku.edu.tw/vim/vimrc>

```
57     size_t length = 0;
58     ssize_t read_length;
59     FILE *fp = fopen(listFileName, "r");
60
61     assert(fp != NULL && "Cannot open file list");
62
63     while((read_length = getline(&line, &length, fp)) != -1) {
64         current = (struct filelist *)malloc(sizeof(struct filelist));
65         if(current != NULL) {
66             current->next = NULL;
67             current->filename = strdup(line);
68
69             // Assume all source code from project is *.c
70             if(current->filename[read_length-1] == '\n') {
71                 current->filename[read_length-2] = 'o';
72                 current->filename[read_length-1] = '\0';
73             }
74             else
75                 current->filename[read_length-1] = 'o';
76             printf("%s\n", current->filename);
77         }
78
79         if(head == NULL) head = current;
```

61,1-4 38%





- vimrc 常用設定
  - :set nu
    - 顯示行號
  - :set ai
    - 自動對其縮排
  - :set tabstop=4 (default 8)
    - 縮排間隔數
  - :set bg=dark (default light)
    - 上色模式



# Thanks for your listening!



# Practice

- Uva (5)  
100, 579, 10424, 11727, 11984
- POJ (5)  
1000, 1004, 1298, 1450, 2159

