

CPE 考試祕笈

2011 年 12 月 04 日

這份指南，提供給首次參加 CPE 程式檢定，或尚不熟悉 CPE 程式檢定的同學。
其內容將帶領同學熟悉考試環境、開發工具的使用，以及向大家分享考試解題時的實用技巧。

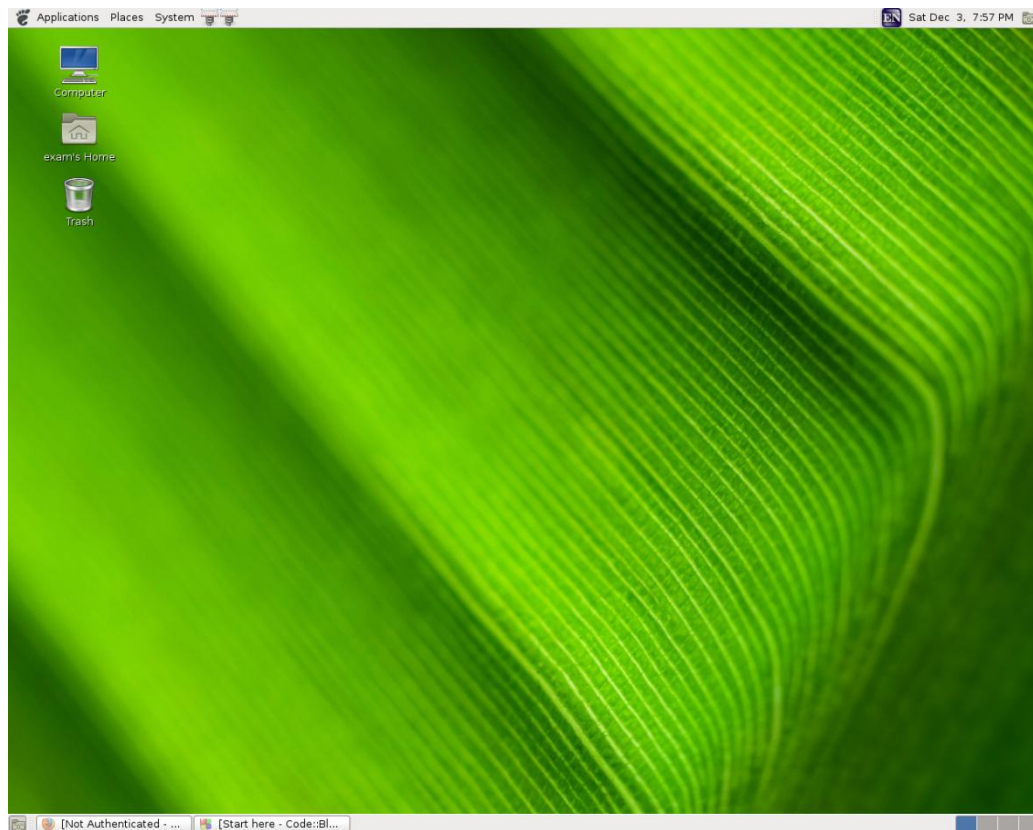
目次

CPE 考試環境	1
CPE 考試機	1
CPE 考試系統	2
利用所提供的資源	3
開發工具	4
使用 Code::Blocks	5
下載與安裝 Code::Blocks	9
處理測試資料	10
介紹	10
讀入 n 筆資料	12
讀至檔案結束	15
讀至 0 結束	17
測試程式	19
剪貼方式輸入測試資料	19
檔案方式導入測試資料	20

CPE 考試環境

CPE 考試機

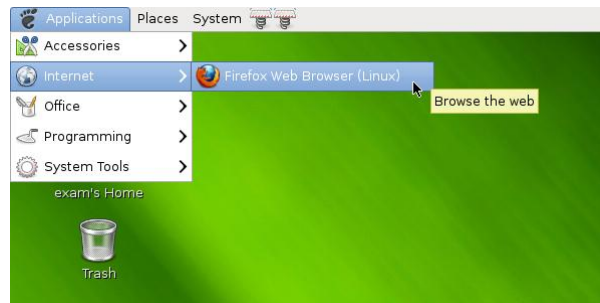
CPE 檢定的所有考場一律使用相同的 CentOS Linux 作業系統，它看起來會像這樣：



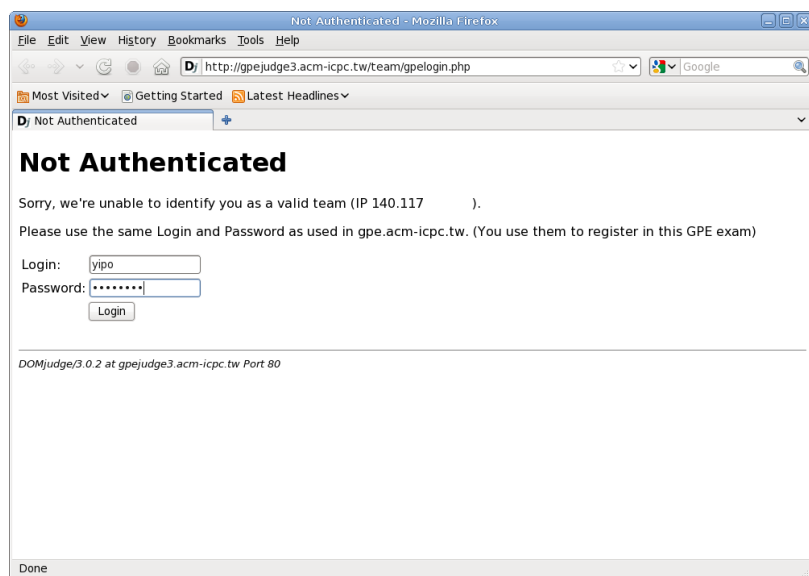
CPE 檢定的題目會以英文出題，考試機與登入系統也皆為英文介面。參加檢定考試前，記得要對英文有所準備喔。也因此，考試允許你自行攜帶一本字典進入考試使用。然而如果你覺得字典太過笨重，或手邊臨時找不到字典能帶來考場，也敬請利用考試系統所提供的線上字典查詢不懂的單字。

CPE 考試系統

如果在考場時，你不小心把在畫面上的 Firefox 瀏覽器關掉的話，請再次把它開啟。唯有透過瀏覽器登入 CPE 考試系統，才能進行考試。



登入系統所使用的帳號密碼，與報名 CPE 檢定時的一樣，到考場時可別忘了你的帳號與密碼了。



為了確保考試時會操作這個系統，你平時可以在家裡連到以下的網址，試著操作看看：

CPE 考前練習

<http://acm-icpc.tw/exam/>

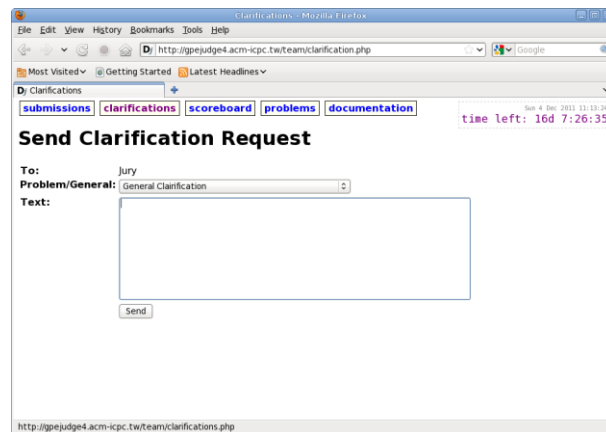
而在這裡有考試系統的使用說明可供參考：

CPE 伺服器簡易操作手冊

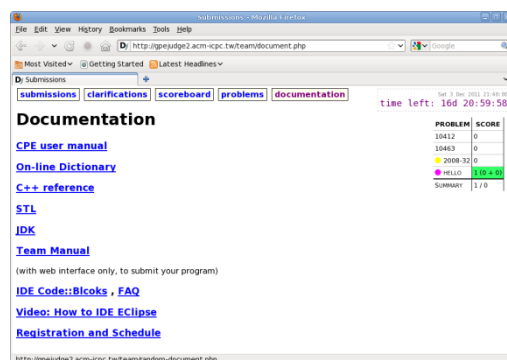
http://conf.acm-icpc.tw/CPE_user_manual/CPE_user_manual.html

利用所提供的資源

在考場時，如果覺得題目有地方不清楚，你可以利用系統發問，或是看看是否有人提出任何問題：



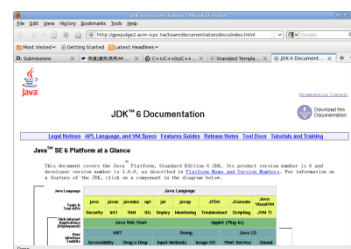
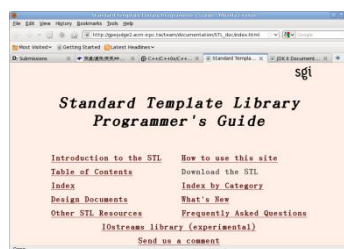
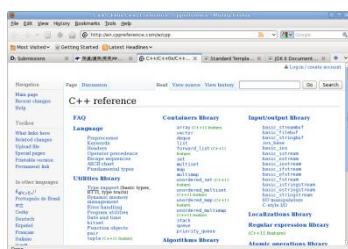
CPE 程式檢定測試你的程式設計能力，你不需要背誦單字，也不需要背函式庫。在系統上，我們提供了許多參考資料，敬請多多利用。



按下介面上的 documentation 按鈕

線上字典

函式庫查詢



C++ reference

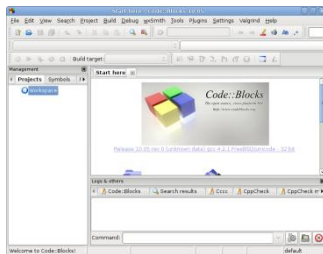
STL

JDK

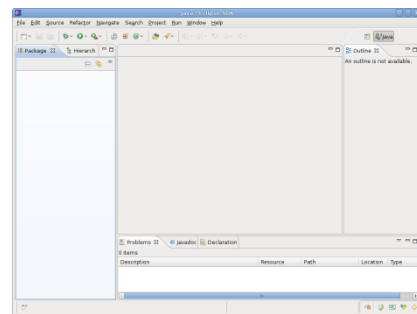
開發工具

CPE 程式檢定的考試機，提供以下的工具程式：

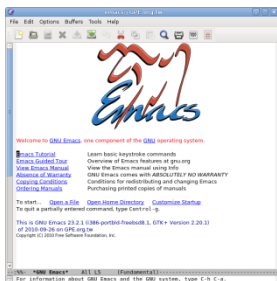
- Code::Blocks
- Eclipse
- Emacs
- gedit
- Vim



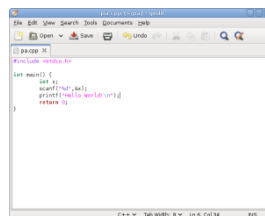
Code::Blocks



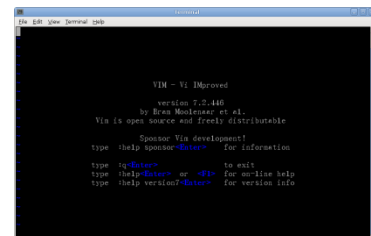
Eclipse



Emacs



gedit



Vim

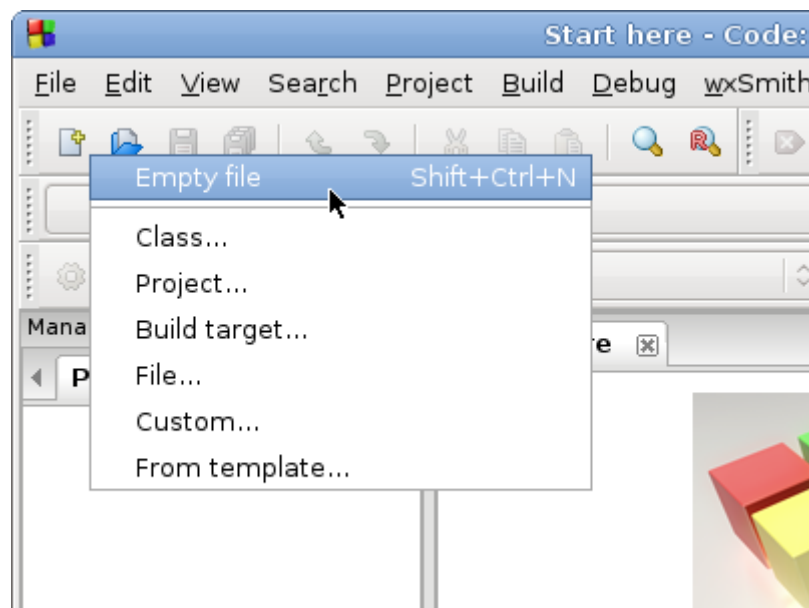
你可以自由選擇以你自己喜歡的方式進行解題。如果你尚不熟悉任何以上的工具，我們推薦你使用 Code::Blocks (也許你用過 Dev-C++，它們在介面上很相似)。底下我們提供 Code::Blocks 的簡易使用教學。

使用 Code::Blocks

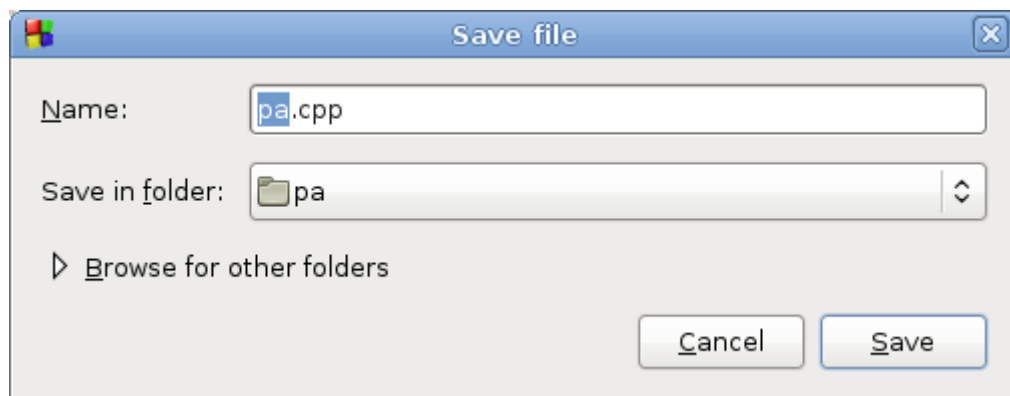
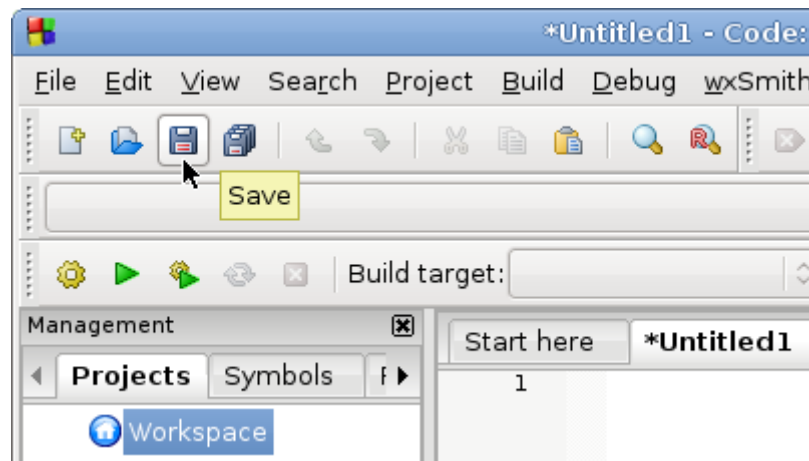
解題時，我們總是將原始碼寫在單一一份原始檔中。為此替每題題目皆開啟一個專案，也許有些麻煩。使用 Code::Blocks 我們可以不建立專案，開啟單一一份原始檔也能編譯與執行程式。推薦你使用這個方法：

建立原始檔

1. 按下工具列上的頭一個按鈕 New File，並於選單選擇 Empty file。



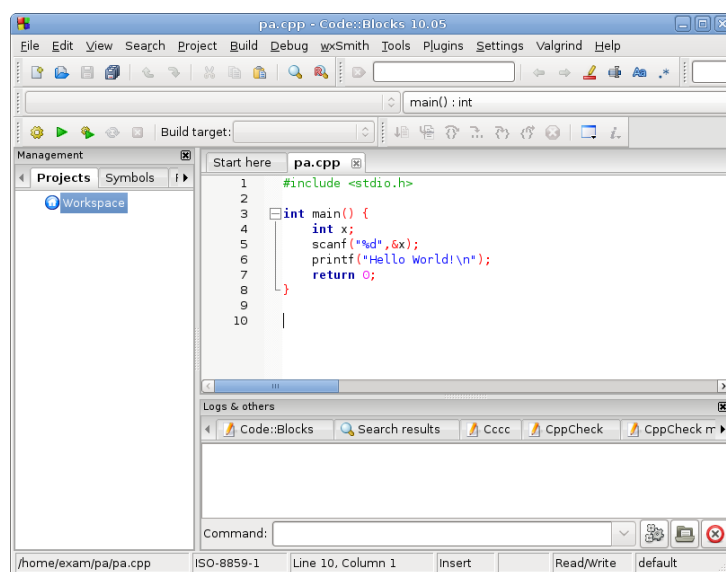
2. 接著按下存檔按鈕，為你的檔案命名，並記得加上所使用語言的附檔名。



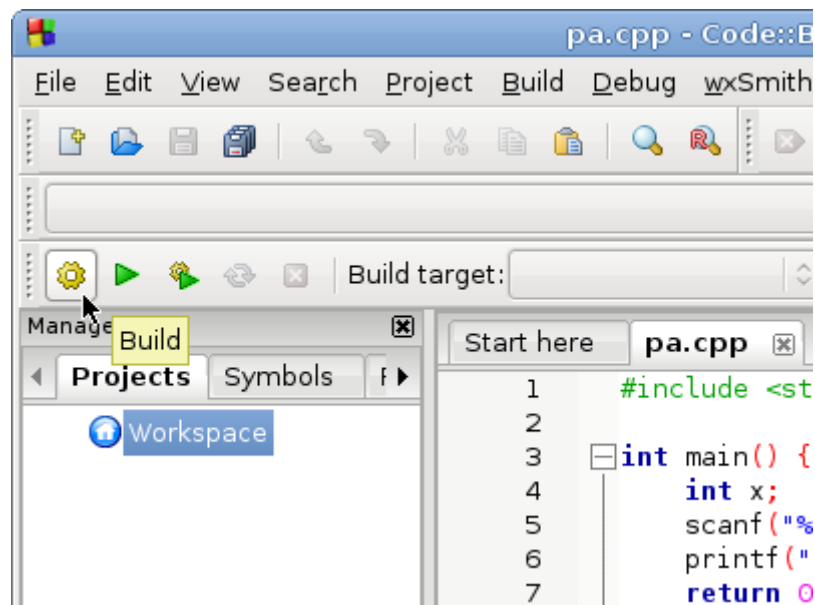
若使用 C 語言即為.c；C++則為.cpp。

Code::Blocks 根據附檔名來判斷你使用何種語言撰寫程式。

3. 就這麼簡單，可以開始撰寫程式了。



編譯與執行

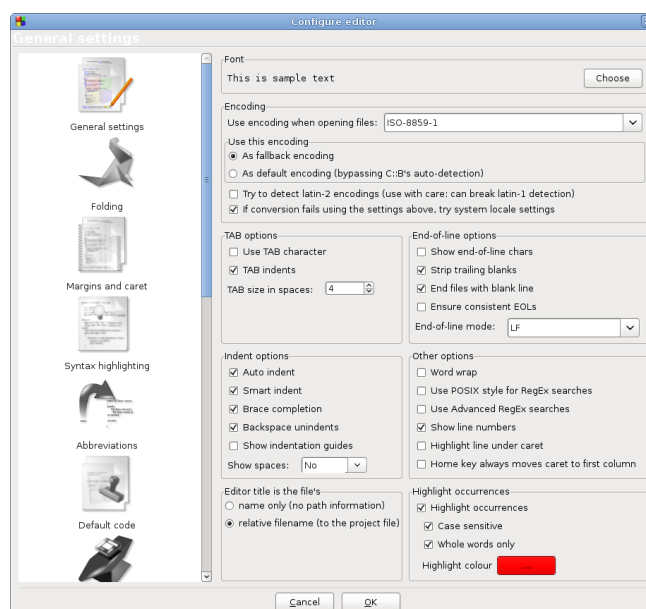
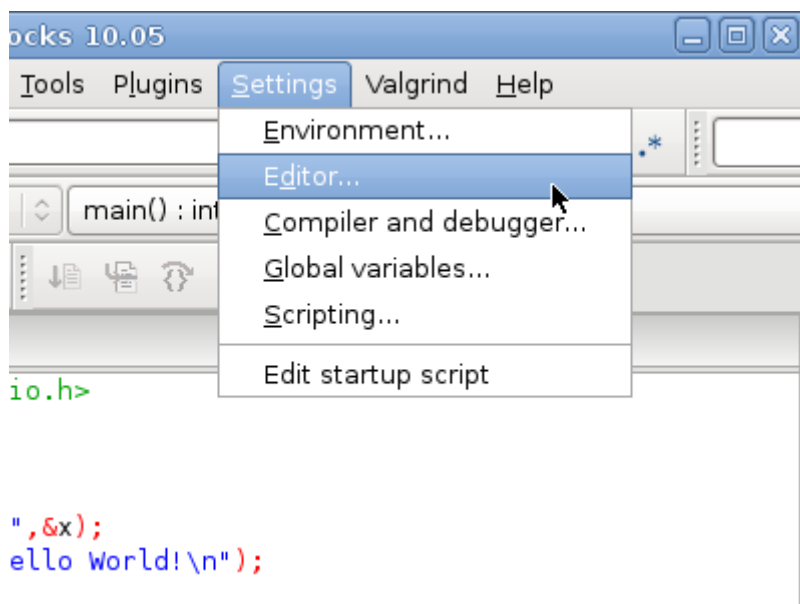


1. 程式寫妥要編譯程式，按「齒輪」按鈕。
2. 通過編譯欲執行程式，按「三角形」按鈕。
3. 若想一併執行以上兩步驟，按第三個「齒輪與三角形」的按鈕。
4. 以上也都有對應的快速鍵可供使用：

編譯程式	Ctrl+F9
執行程式	Ctrl+F10
編譯並執行	F9

依照個人喜好調整

如果你也和我一樣覺得編輯器的字有點小，或是不喜歡 Code::Blocks 預設的縮排方式。點選功能表上的「Settings > Editor...」，可以叫出控制設定的視窗以調整設定。



點 Font 部份的 Choose 按鈕，可以調整字型大小。

TAB options 與 Indent options 的部份是有關縮排的設定。

其他選項也可自行摸索調校。

下載與安裝 Code::Blocks

文件這裡所述也僅是概略的介紹。若要在檢定時得心應手，你還需要多多熟悉你的工具。不如現在就打開 Code::Blocks，試著寫寫幾份題目吧（或練習後面章節所提供的練習）。如果你的電腦尚未安裝 Code::Blocks，可以透過以下的方式安裝：

Windows 作業系統

前往官方網站下載安裝程式，執行並遵照其指示完成安裝。

Code::Blocks 下載頁面

<http://www.codeblocks.org/downloads>

• Home
• Features
• Screenshots
• Downloads

- Binaries
- Source
- SVN

• Plugins
• User manual
• Licensing
• Donations

Quick links

• FAQ
• Forums
• Wiki
• Nightlies
• BugTracker
• PatchTracker
• Browse SVN
• Browse SVN log

• Windows 2000/XP/Vista/7
• Linux 32-bit
• Linux 64-bit
• Mac OS X

NOTE: There are also more recent *nightly builds* available in the **forums** or (for Debian users) in **Jens' repository**. Please note that we consider nightly builds to be *stable*, usually.

IMPORTANT NOTE: If you try to download from BerliOS and get a "Too many clients" - error, you should retry to download the file. According to a BerliOS - admin, this can happen several times, before the download starts. If it still does not work, search our **forum** for "alternate mirror", you will find at least alternatives for the windows and debian downloads.

Windows 2000 / XP / Vista / 7:

File	Date	Size	Download from
codeblocks-10.05-setup.exe	27 May 2010	23.3 MB	BerliOS or Sourceforge.net
codeblocks-10.05mingw-setup.exe	27 May 2010	74.0 MB	BerliOS or Sourceforge.net

[SF.NET download](#)

NOTE: The codeblocks-10.05mingw-setup.exe file *includes* the GCC compiler and GDB debugger from **MinGW**.

點選 Download the binary release，下載含有 MinGW 的版本。

Ubuntu 作業系統

打開終端機，輸入以下指令：

```
sudo apt-get update
sudo apt-get install codeblocks -y
```

通常系統會提示你輸入密碼，待其完成下載與安裝即可。

處理測試資料

如果因為不知如何讀取資料，而無法體會程式解題的樂趣，那樣是很可惜的。因此，文件在這裡介紹一些常見的測試資料型式，以及該如何讀取它，並附上實際题目的範例，希望能夠帶領你快速進入解題的狀況。每個部份後面還有相關的練習題，試著作些練習，對檢定考試更有幫助。

介紹

C language

談到輸入與輸出，對 C 語言來說便是 scanf 與 printf。無論如何，我們還是再介紹一次 scanf 的用法。

```
int scanf(const char *format, ...);
```

函式第一個要填的參數是個字串，裡面你敘述要如何讀取接下來的輸入。對於你所打算讀取的資料格式與型態，依序填入對應的符號，如以下的表格：

字元	char *	%c	字串	char *	%s
		整數	正整數	八進位	十六進位
(unsigned) char *		%hhd	%hhu	%hho	%hhx
(unsigned) short *		%hd	%hu	%ho	%hx
(unsigned) int *		%d	%u	%o	%x
(unsigned) long *		%ld	%lu	%lo	%lx
(unsigned) long long *		%lld	%llu	%llo	%llx
浮點數	float *	%f	浮點數	double *	%lf

接下來的參數便依序填入存放對應資料變數的位址，即變數名稱前面加上 & 符號。特別注意字串變數，其名稱本身已代表是位址，所以唯有字串不必再加上 & 符號。

舉個例子，像是這樣：

```
char ch,str[64];
int num;
float value;

scanf("%c%s%d%f",&ch,str,&num,&value);
```

除了 %c 以外，各種輸入方式皆會忽略前方多餘的空白字元，即 space、tab 與 enter。因此資料中，只要東西的順序一樣，其間怎麼空隔、有多少空隔，其實都是無所謂的。像是下面這些情況，其實對我們來說都是一樣的：(都能以 scanf("%d%f%f%f", ...) 的方式正確讀取。)

3 ↓ 1.10 ↓ 2.20 ↓ 3.30	3.1.10.2.20.3.30	3 ↓ 1.10.2.20.3.30	3..1.10 ↓ ↓ .2.20...3.30
---------------------------------	------------------	-----------------------	--------------------------------

C++

對於 C++ 來說輸入與輸出就是 cin 與 cout。cin 會根據你給它的變數型態，讀取相應的資料，使用的運算元是指向變數的 >>。相對來說是簡單直覺許多。

讀入 n 筆資料

測試資料最常見的就是這種型式了。處理方式就是讀進這個 n，然後跑 n 次的迴圈。

C language

```
int main() {  
    int n;  
    scanf("%d",&n);  
    while (n--) {  
        /* 讀取每筆資料 */  
    }  
    return 0;  
}
```

C++

```
int main() {  
    int n;  
    cin>>n;  
    while (n--) {  
        // 讀取每筆資料  
    }  
    return 0;  
}
```

範例

10406: [Vito's family](#)

黑道老大 Vito 搬來街上住，希望與每位親戚家的距離總合要最短。

C Language

```
#include <stdio.h>
#define MAX_R 100

int num[MAX_R];

int main() {
    int n,r,s,i;
    int sum;

    scanf("%d",&n);
    while (n-->0) {
        scanf("%d",&r);
        for (i=0;i<r;i++) {
            scanf("%d",&s);
            num[i]=s;
        }

        /* 剩下的留給你完成，提示：中位數。 */

        printf("%d\n",sum);
    }

    return 0;
}
```

C++

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

vector<int> num;

int main() {
    int n,r,s;

    cin>>n;
    while (n-->0) {
        cin>>r;
        num.clear();
        for (int i=0;i<r;i++) {
            cin>>s;
            num.push_back(s);
        }

        // 剩下的留給你完成，提示：中位數。

        cout<<endl;
    }

    return 0;
}
```

從這邊我們知道，可以有很多層的「讀入 n 筆資料」。

練習

10401: [Fibonacci Base](#)

10403: [Funny Encryption Method](#)

10408: [What is the Probability?](#)

讀至檔案結束

這種測試資料不會告訴你有幾筆資料，你要一直處理到沒有資料為止。

C language

檢查 scanf 函式的回傳值可以知道資料結束了沒有。平常 scanf 會回傳它成功讀進了多少個元素；當讀到檔案結束時，scanf 則回傳 EOF。

把它整合在 while 的判斷條件裡就成了這樣子：

```
int main() {
    int x;
    while (scanf("%d",&x)!=EOF) {
        /* 處理目前這筆資料 */
    }
    return 0;
}
```

C++

若把 cin 放進 while 的判斷條件裡，cin 會自動轉型成 void*。而當檔案結束時，其值會變成 NULL，也就是 0 代表 false。

所以我們寫做這樣：

```
int main() {
    int x;
    while (cin>>x) {
        // 處理目前這筆資料
    }
    return 0;
}
```

範例

10407: [Hashmat the brave warrior](#)

算出敵方軍隊與我方人數的差距。

C Language

```
#include <stdio.h>

int main() {
    long long a,b; /* 可能等於 2^32，所以 unsigned int 還不夠。 */
    while (scanf("%lld%lld",&a,&b)!=EOF) {
        /* 試著完成它吧！ */
    }
    return 0;
}
```

注意到 scanf 對於 long long 要使用 %lld 喔。

C++

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

int main() {
    long long a,b; // 可能等於 2^32，所以 unsigned int 還不夠。
    while (cin>>a>>b) {
        // 試著完成它吧！
    }
    return 0;
}
```

練習

10400: [The 3n + 1 problem](#)

10405: [Jolly Jumpers](#) (混合兩種方式)

10411: [Back to High School Physics](#)

讀至 0 結束

這種類型也是時常會出現的型式，只要簡單地加個判斷跳出迴圈即可。

C language

```
int main() {
    int n;
    while (scanf("%d",&n)!=EOF) {
        if (n==0) break;

        /* ... */
    }
    return 0;
}
```

C++

```
int main() {
    int n;
    while (cin>>n) {
        if (n==0) break;

        // ...
    }
    return 0;
}
```

範例

10404: [Primary Arithmetic](#)

看看做加法時會進位幾次。

C Language

```
#include <stdio.h>

int main() {
    int a,b;
    while (scanf("%d%d",&a,&b)!=EOF) {
        if (a==0&&b==0) break;

        /* 留給你完成 */
    }
    return 0;
}
```

C++

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

int main() {
    int a,b;
    while (cin>>a>>b) {
        if (a==0&&b==0) break;

        /* 留給你完成 */
    }
    return 0;
}
```

練習

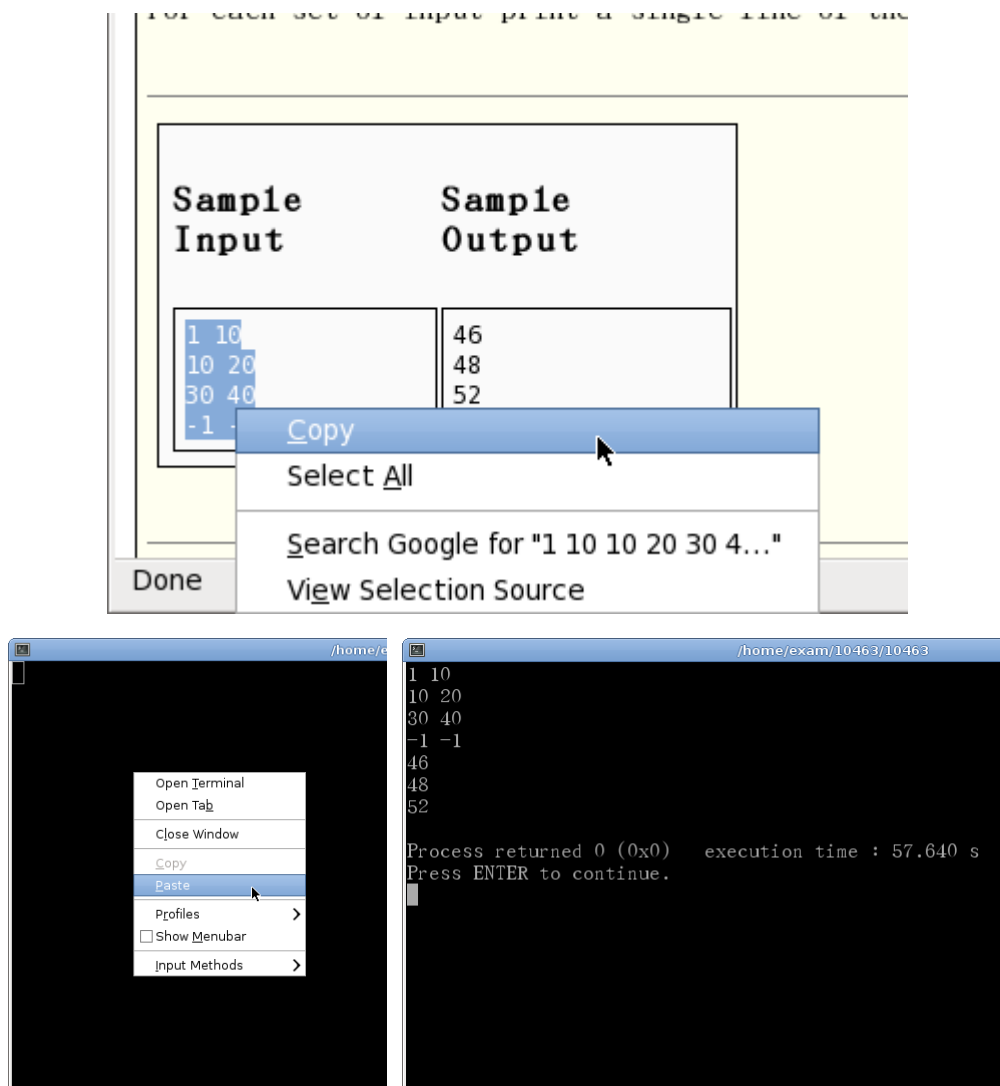
10416: [Last Digit](#)

10418: [Minesweeper](#)

測試程式

測試你的程式，除了手動輸入測試資料以外，其實也可以用剪下、貼上的方式，直接從題目網頁上剪下測試資料，貼進你的程式做為輸入。或者是將測試資料存成檔案，利用終端機執行程式，透過指令將檔案導入做為輸入。後者這樣的好處是，修改測試資料，再重新測試將十分方便；而且這個方式，程式輸出的結果也不會受到輸入的資料干擾。

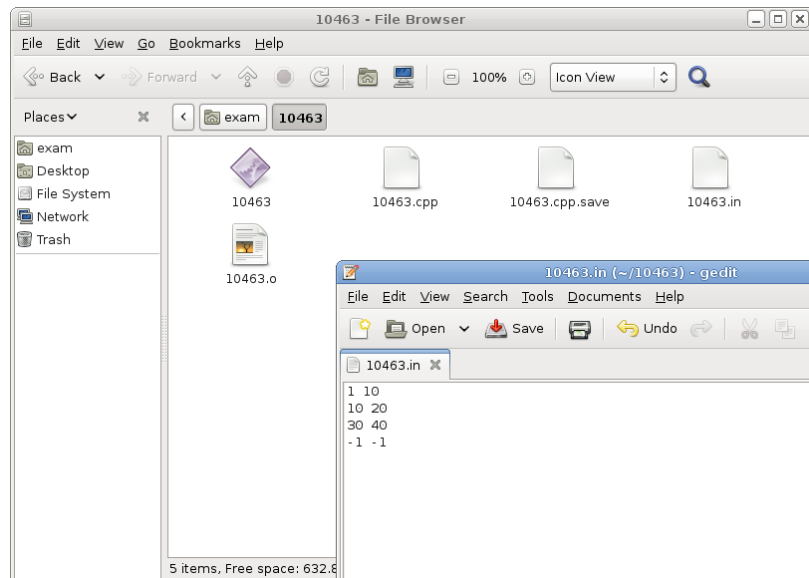
剪貼方式輸入測試資料



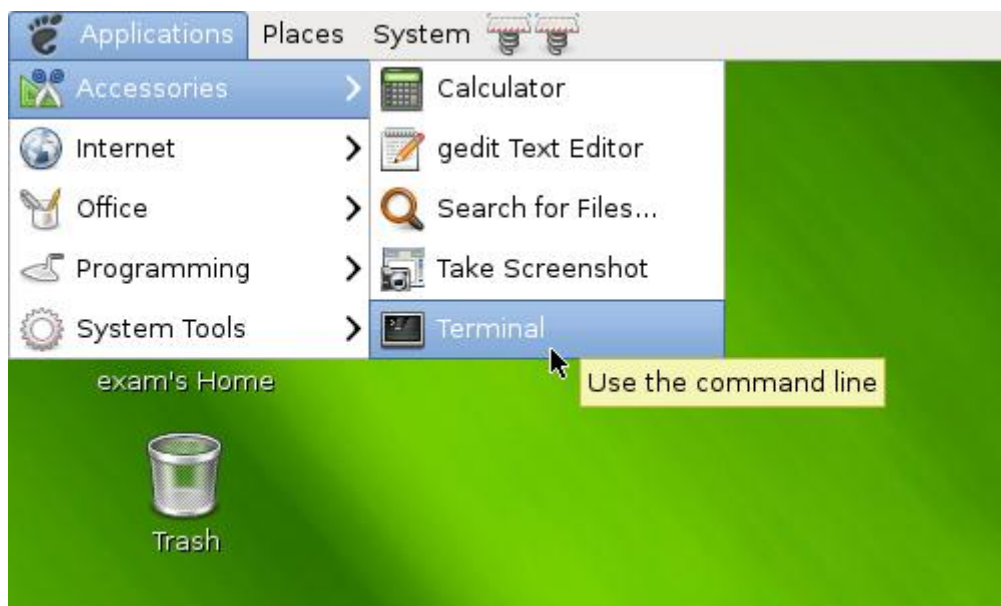
就這麼輕鬆容易。(有時候最後一個換行會沒複製到，記得再按一次 enter。)

檔案方式導入測試資料

先把測試資料打好，存檔於程式旁邊。(大家習慣上是題目名稱加上附檔名 .in)



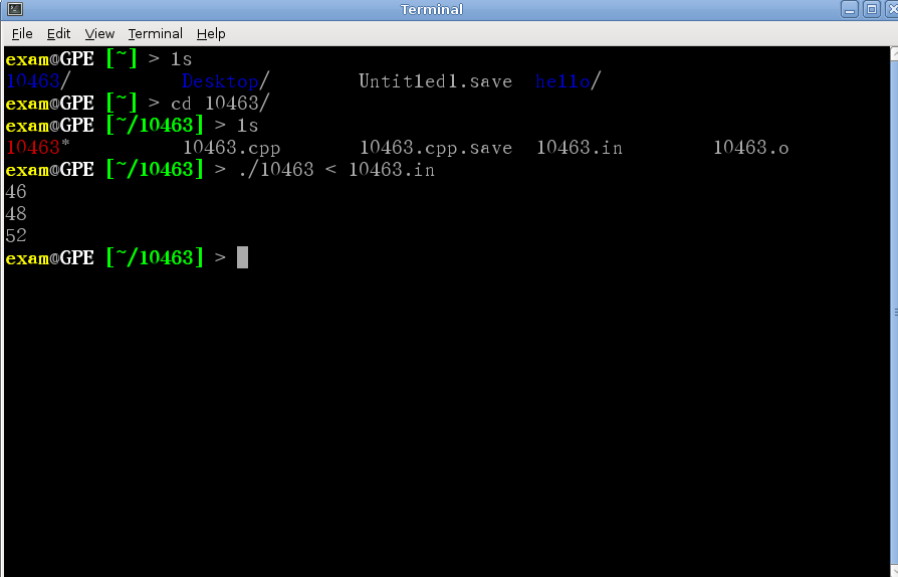
這個方式需要使用終端機，請打開終端機 (Terminal)。



輸入指令切換到程式碼所在的資料夾。(以 cd 切換資料夾；以 ls 檢視資料夾中有哪些檔案。)

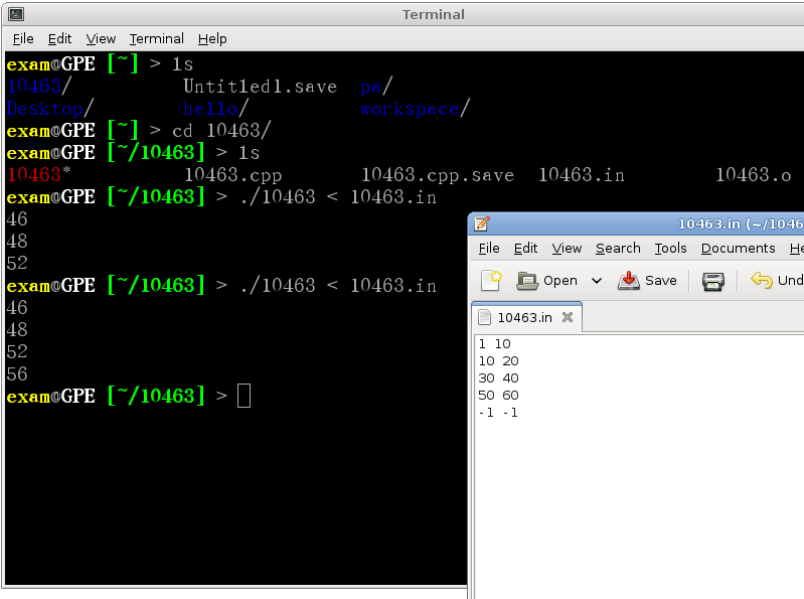
使用 < 符號來將檔案導入欲執行的程式：

`./你的程式 < 測試資料檔案`



```
exam@GPE [~] > ls
10463/ Desktop/ Untitled1.save hello/
exam@GPE [~] > cd 10463/
exam@GPE [~/10463] > ls
10463* 10463.cpp 10463.cpp.save 10463.in 10463.o
exam@GPE [~/10463] > ./10463 < 10463.in
46
48
52
exam@GPE [~/10463] >
```

要試著以不同的測試資料測試程式，只要修改一下檔案內容，再重新執行以上指令即可。(按方向鍵↑，可以叫出之前曾輸入過的指令。)



```
exam@GPE [~] > ls
10463/ Desktop/ Untitled1.save pa/ workspace/
exam@GPE [~] > cd 10463/
exam@GPE [~/10463] > ls
10463* 10463.cpp 10463.cpp.save 10463.in 10463.o
exam@GPE [~/10463] > ./10463 < 10463.in
46
48
52
exam@GPE [~/10463] > ./10463 < 10463.in
46
48
52
56
exam@GPE [~/10463] >
```

10463.in (~/10463)

```
1 10
10 20
30 40
50 60
-1 -1
```