Chapter

4

運算子與運算式

本章大綱

- ▶ 4-1　運算子
- ▶ 4-2　運算式
- ▶ 4-3　演算法基本概念與實例探討
- ▶ 4-4　APCS 觀念題

4-1　運算子

所謂運算子 (Operator)，指的是可以對運算元 (Operand) 執行特定功能的特殊符號。本書介紹以下四類常用運算子：算術 (Arithmetic) 運算子、邏輯 (Logical) 運算子、關係 (Relational) 運算子與複合指派運算子。除此之外，我們還會討論運算子的優先順序 (Precedence) 與結合律 (Associativity)。優先順序用來決定同一式子擁有多個運算子時，每一個運算子進行運算的優先順序；而結合律則決定了在同一敘述中，相鄰的運算子有相同優先順序的運算子執行的順序。

算術運算子(Arithmetic operators)

算術運算子用來執行一般的數學運算，包括指派 (=)、取正負數 (+/-)、加 (+)、減 (-)、乘 (*)、除 (/)、取餘數 (%)、遞增

(++) 及遞減 (--) 等，下表是 C 語言的算術運算子列表：

運算子	定義	優先順序	結合律
=	指派	14	由右至左
++,--	後置遞增 / 遞減	1	由左至右
++/--	前置遞增 / 遞減	2	由右至左
+/-	正負號，一元運算子	2	由右至左
*	乘法運算	3	由左至右
/	除法運算	3	由左至右
%	求餘數 (Modulus)	3	由左至右
+/-	加法 / 減法運算	4	由左至右

指派運算子（Assignment operators）

指派運算子的符號爲 (=)，其作用爲將運算符號右邊運算式的值指派給運算符號左邊的運算元。所以，以下敘述 sum=a+b 是將 a+b 的值指派給 sum。

```
int sum = 0,a = 3,b = 5;
sum = a + b;
```

上式與數學的等號是不同的，所以不要一直懷疑爲什麼 0 會等於 8。其次，你是不能將常數放在指派運算子的左邊，例如：

```
8 = x ;
```

此爲一個不合法的敘述，但以下敘述，將常數 8 指派給變數 x 是合法的。

```
x = 8 ;
```

五則運算

以下是一些簡單算術五則運算結果：

```
int a=5,b=4;
printf("%d\n",a+b);//9
printf("%d\n",a-b);//1
printf("%d\n",a*b);//20
printf("%d\n",a/b);//1
printf("%d\n",a%b);//1
```

整數除法或實數除法

C 語言的除法運算，只有被除數與除數的型態均爲整數，才是整數除法，商的型態爲整數。其次，只要被除數與除數有一型態爲實數即爲實數除法，得到實數商。例如：

```
int x=5, y=4;
double xf=5,yf=4;
printf("%d\n",x/y); //1
printf("%f\n",xf/y);//1.25
printf("%f\n",x/yf);//1.25
printf("%f\n",xf/yf);//1.25
```

整數除法與取餘的應用

整數除法與取餘可以將一個整數分解爲數個數字。例如，若要使用七段顯示器顯示 152，那就要將此數字分解爲 1,5,2 三個數字，其方法如下：

```
int a=152;
int a1=152/100;//1百位數
int a2=(152-a1*100)/10;//5十位數
int a3=a %10;//個位數
```

取餘

若要讓一個數字在累加的過程中，保持一定的循環，那也是用取餘。例如：

```
while(1)  {//無窮迴圈，請看第六章
    b=a[i];
    i=(i+1) %4;
}
```

那 i 就永遠在 0,1,2,3 循環。以上將數字分解的運算思維，在程式設計領域應用非常多，以下自我練習請多練習，本書後續的問題與 APCS 題目就能迎刃而解。

自我練習

1. 請輸入一個四位數，並將其分解輸出爲四個數字。例如，輸入 1234，那可以輸出 1,2,3,4。
2. 請輸入一個四位數，並將其反相輸出此四個數字。例如，輸入 1234，那可以輸出 4321。
3. 若有一個秒數是 12345，那如何分解爲幾時幾分幾秒。例如，本例要能得到 3 時 25 分 45 秒。
4. 若有一個 5 位數，請問如何分解爲兩個一組。例如，a=25678，那如何分別得到 25,56,67,78 等 4 個數字；其次，要如何得到 2,56,78 三組數字呢。
5. 若有一個 5 位數，那如何得到最左邊與最右邊數字的和。例如，輸入 12345，那結果是 6。
6. 若有一個 5 位數，那如何得到最左邊兩位數與最右邊兩位數字的和。例如，輸入 12345，那結果是 12+45=57。
7. 若有一個 6 位數，那如何得到最左邊三位數與最右邊三位數字的和。例如，輸入 123456，那結果是 123+456=579。

補充說明

以上方法不只一種，方法沒有優劣，只要自己想出來的都是最好的。

遞增(＋＋)及遞減(--)

遞增 (++) 及遞減 (--) 是 C 語言特有的運算子，此二個運算子又分爲前置與後置，前置是運算子在運算元之前，例如：

```
a=1;++a;printf("%d\n",a);  //2
```

後置是運算子在運算元之後，如

```
a=1;a++;printf("%d\n",a);;   //2
```

原則上不論 ++a 或 a++ 都是將 a 值加 1 並放回 a。

次方與根號

值得留意的是，C 語言的數學運算子只有以上 +,-,*,/,%，若所需運算沒有運算子，例如，要次方或根號運算，那要如何處理呢？答案是要使用數值處理函式。例如，以下 pow() 函式可計算 3 的 2 次方：

```
a = pow(3,2)；//include <math.h>
```

若要執行開根號運算，則應使用數值處理函式的 sqrt() 函式。如以下敘述是計算 9 的開根號。

```
x=sqrt(9);//include<math.h>
```

但以上數值函式都放在 math.h，所以都要在標頭檔使用

```
include <math.h>
```

載入函式，請留意以上敘述載入標頭檔，不用分號『;』結束。

自我練習

1. 請將以上運算式於 C 環境編譯、執行，觀察執行結果。

關係運算子(Relational Operators)

關係運算子又稱爲比較 (Comparison) 運算子，用於資料之間的大小比較，比較的結果可得到 int 型態的 1(true) 或 0(false)，下表是 C 語言中的關係運算子符號。其中等於與不等於的符號與其它語言有顯然的不同，請讀者留意，例如，下式用來比較 a 與 b 是否相等。

```
if (a == b) ;
```

下面是關係運算子列表：

運算子	定義	優先順序	結合律
<	小於	6	由左至右
>	大於	6	由左至右
<=	小於等於	6	由左至右
>=	大於等於	6	由左至右
==	等於	7	由左至右
!=	不等於	7	由左至右

例如：

```
int a=8,b=5;
printf("%d\n",a>b);//1
printf("%d\n",a==b);//0
printf("%d\n",a>=b);//1
printf("%d\n",a<=b);//0
printf("%d\n",a!=b);//1
```

自我練習

1. 請將以上運算式於 C 環境編譯、執行，觀察執行結果。

2. 請鍵入以下程式，編譯與修正程式。

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
   int a=8,b=5;
   printf( “%d\n” ,a=b);
   printf( “%d\n” ,a=>b);
   printf( “%d\n” ,a=<b);
   printf( “%d\n” ,a=!b);
   return 0;
}
```

邏輯運算子(Logical Operators)

當同一個運算式要同時存在兩個以上的關係運算子時，每兩個關係運算子之間必須使用邏輯運算子連結，例如，您要找男生且年齡大於 40，此一選擇就同時含有兩個關係運算式，此時就要使用邏輯運算子。C 的邏輯運算子有否定 not『!』、邏輯 and『&&』、邏輯 or『 ||』，如下表所示：

運算子	定義	優先順序	結合律
!	邏輯否定運算	2	由右至左
&&	邏輯 and 運算	11	由左至右
||	邏輯 or 運算	12	由左至右

邏輯否定 not 是將『true』轉爲『false』,『false』轉爲『true』;邏輯 and 是兩件事都 true，結果才是 true，其餘都是 false，其眞值表如下：

事件1	事件2	結果
true	true	true
true	false	false
false	true	false
false	false	false

邏輯 or 是兩件事只要有一件爲 true，那就爲 true，只有兩件事全爲 false，才爲 false，其眞值表如下：

事件1	事件2	結果
true	true	true
true	false	true
false	true	true
false	false	false

例如：

```
int a=8,b=5,c=3;
printf("%d\n",!(a>b));//0
printf("%d\n",!(a==b));//1
printf("%d\n",a>b && b>c );//1
printf("%d\n",a>b && b==c );//0
printf("%d\n",a>b || b==c );//1
printf("%d\n",a==b || b==c );//0
```

自我練習

1. 若有一數學式，判斷 x 是否滿足 1<x<=6 , 請轉換爲 C 語言敘述。(提示：1<x<=6 是數學語言，C 語言是 int x=3;printf("%d",x>1 && x<=6); 或 x=9;printf("%d",x>1 && x<=6);)
2. 若有一數學式，判斷 x 是否滿足 x>3 or x<-2 , 請轉換爲 C 語言敘述。
3. 輸入三角形三邊長 a,b,c，圍成三角形的條件是，任兩邊之和要大於第三邊，那如何以 C 語言表示？
   (提示：任兩邊之和大於第三邊的數學語言是 :a+b> c and a+c> b and b+c> a，請將以上數學語言轉爲 C 語言)
4. 若有一數學式，同時判斷六個變數是否滿足 $\frac{a1}{a2}=\frac{b1}{b2}=\frac{c1}{c2}$，請轉換爲 C 敘述。那 $\frac{a1}{a2}=\frac{b1}{b2}\neq\frac{c1}{c2}$ 又如何表示呢？

複合指派運算子

結合指派與算術的運算子稱爲複合指派運算子，此乃 C 語言特有的運算子，例如程式設計者常會鍵入 sum=sum+5，程式語言乃制定此一複合指派運算子 +=，所以上述 sum=sum+5，即可寫成 sum + = 5，下表是 C 語言常用的複合指派運算子：

運算子	定義	優先順序	結合律
=	指派內容	14	由右至左
+=	相加之後再指派內容	14	由右至左
-=	相減之後再指派內容	14	由右至左
*=	相乘之後再指派內容	14	由右至左
/=	相除之後再指派內容	14	由右至左
%=	取餘之後再指派內容	14	由右至左

例如：

```
int sum=0,a=5;
sum+=5;
printf("%d\n",sum);//5
sum/=a;
printf("%d\n",sum);//1
```

運算子的優先順序(Precedence)

同一敘述，若同時含有多個運算子，此時即需定義運算子的優先順序。例如：

```
x=a+b*c;
```

在數學裡，我們定義先乘除再加減，程式語言也是相同，必須定義這些運算子的優先順序，C 語言的運算子優先順序如下表：

優先順序	運算子	說明	結合律
1	++,--	後置遞增與遞減。例如 a++,a--。	由左至右
2	++,-- +/- !	前置遞增與遞減。例如 ++a,--a。 取正或取負。 邏輯否定。	由右至左
3	*,/,%	算術乘、除、取餘。	由左至右
4	+,-	算術加、減	由左至右
6	>,>=,<,<=	關係運算的大小。	由左至右
7	==,!=	關係運算的相等或不相等。	由左至右
11	&&	邏輯的 and。	由左至右
12	||	邏輯的 or。	由左至右
14	=,+=,-=,*=,/=,%=	指派與複合指派。	由右至左

由以上各運算子的『優先順序』可知，乘號『*』的優先順序是第 3，而加號『+』則是第 4。所以，以上敘述：

```
x=a+b*c;
```

的結果與

```
x=(a+(b*c));
```

的效果相同。同理，

```
x=a>b && b>z;
```

的結果同義於

```
x=(a>b) && (b>z);
```

自我練習

1. 鍵入、執行以下程式，並寫出執行結果。

程式	執行結果

#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
  int a,b,c;
  a=1;b=2;b=3;
  a=a+a*2/2-3%2;
  printf("%d\n",a);
  a=5;b=4;
  c= a>2 && !(b>3) ||(a==3);
  printf("%d\n",c);
  return 0;
}	

運算子的結合律(Associativity)

當同一敘述，相鄰的運算子擁有相同優先順序的運算子，此時即需定義運算子是『由左至右』的左結合或『由右至左』的右結合。例如：

```
x=a-b-c;
```

同樣是減號 (-)，優先順序相同，此時就要靠定義結合律，減法結合律是由左至右，所以以上同義於

```
x=((a-b)-c);
```

而

```
a=b=c=2;
```

指派運算子的結合律是由右至左，所以以上式子同義於：

```
(a=(b=(c=2)));
```

以上式子，a、b、c 的結果都是 2。

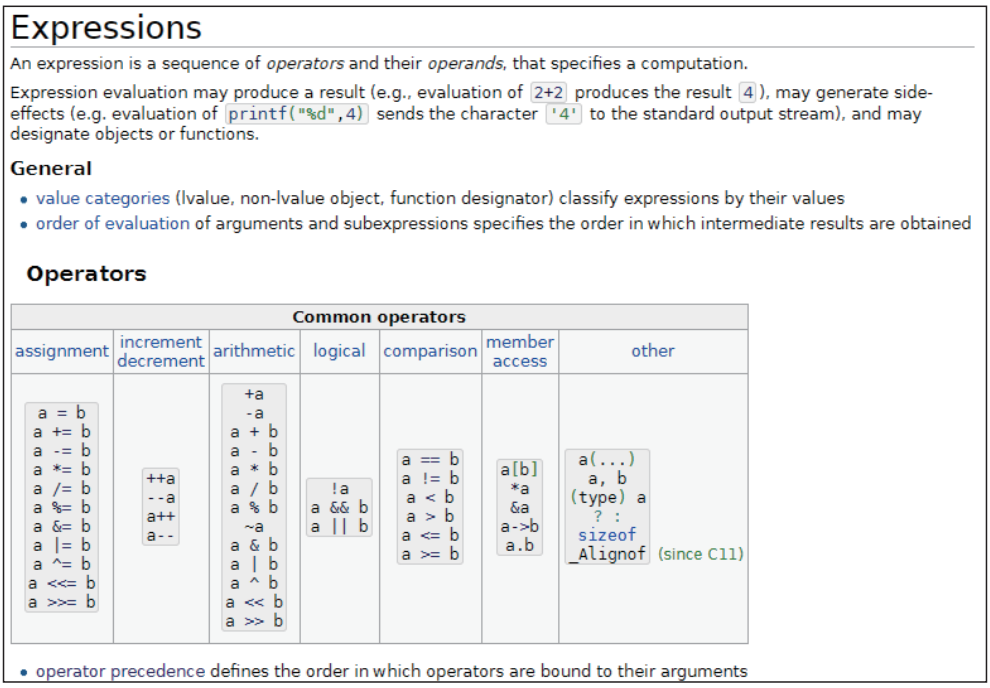
自我練習

1. 鍵入、執行以下程式，並寫出執行結果。

程式	執行結果

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
   int a=3,b=2,c=1;
   c=a-b-c;
   printf("%d\n",c);
   a=b=c=2;
   printf("a=%d,b=%d,c=%d\n",a,b,c);
   return 0;
}
```

關於以上運算子的詳細說明可點選 https://en.cppreference.com/w/c 裡面的『Expressions』，如下圖：



Expressions

An expression is a sequence of *operators* and their *operands*, that specifies a computation.

Expression evaluation may produce a result (e.g., evaluation of `2+2` produces the result `4`), may generate side-effects (e.g. evaluation of `printf("%d",4)` sends the character `'4'` to the standard output stream), and may designate objects or functions.

General

- value categories (lvalue, non-lvalue object, function designator) classify expressions by their values
- order of evaluation of arguments and subexpressions specifies the order in which intermediate results are obtained

Operators

Common operators						
assignment	increment decrement	arithmetic	logical	comparison	member access	other

`a = b`
`a += b`
`a -= b`
`a *= b`
`a /= b`
`a %= b`
`a &= b`
`a |= b`
`a ^= b`
`a <<= b`
`a >>= b`
	`++a`
	`--a`
	`a++`
	`a--`
		`+a`
		`-a`
		`a + b`
		`a - b`
		`a * b`
		`a / b`
		`a % b`
		`~a`
		`a & b`
		`a | b`
		`a ^ b`
		`a << b`
		`a >> b`
			`!a`
			`a && b`
			`a || b`
				`a == b`
				`a != b`
				`a < b`
				`a > b`
				`a <= b`
				`a >= b`
					`a[b]`
					`*a`
					`&a`
					`a->b`
					`a.b`
						`a(...)`
						`a, b`
						`(type) a`
						`? :`
						`sizeof`
						`_Alignof` (since C11)

- operator precedence defines the order in which operators are bound to their arguments

繼續點選上圖的『operator precedence 』，還可得運算子的優先順序，如下圖：

C Operator Precedence

The following table lists the precedence and associativity of C operators. Operators are listed top to bottom, in descending precedence.

Precedence	Operator	Description	Associativity
1	++ -- () [] . -> (*type*){*list*}	Suffix/postfix increment and decrement Function call Array subscripting Structure and union member access Structure and union member access through pointer Compound literal(C99)	Left-to-right
2	++ -- + - ! ~ (*type*) * & sizeof _Alignof	Prefix increment and decrement Unary plus and minus Logical NOT and bitwise NOT Type cast Indirection (dereference) Address-of Size-of[note 1] Alignment requirement(C11)	Right-to-left
3	* / %	Multiplication, division, and remainder	Left-to-right
4	+ -	Addition and subtraction	
5	<< >>	Bitwise left shift and right shift	
6	< <= > >=	For relational operators < and ≤ respectively For relational operators > and ≥ respectively	
7	== !=	For relational = and ≠ respectively	

4-2　運算式

運算式 (Expression)

任何一個可求得一個值的式子，都稱爲一個運算式，例如，5+3 會傳回一個數值，所以 5+3 是一個運算式，以下是一些合法的運算式，這些運算式通常放在 printf(“%d”, 運算式)、第五章的 if(運算式)、第六章的 for(; 運算式 ;) 及 while(運算式)。一般而言，可以放在等號右邊的東西，都可以稱爲運算式。

```
6 ;          // 傳回6
4+5 ;        // 傳回9
sum > 3 ;    // 傳回true or false
a==b;        // 傳回true or false
a + b ;      // 傳回a + b的結果
```

敘述 (Statement)

凡是控制執行的順序、對運算式取值或不作任何事，均可稱爲敘述。所有的敘述都要以分號 (;) 作結束，例如，以下式子即是一個敘述。於 C 語言中，若前一個敘述未以分號結束，則錯誤訊息通常出現在下一個敘述的開頭。

```
sum = sum + 1 ;
```

敘述區塊 (Block Statement) 或複合敘述 (Compound Statement)

在任何可以放上單一敘述的地方，你都能放上敘述區塊，敘述區塊亦稱複合敘述，一個複合敘述是由兩個大括號組合而成，如下所示，但大括號之後不可再加分號。

```
{
    t = a ;
    a = b ;
    b = t ;
}
```

註解 (Comments)

適當的程式註解才能增加程式的可讀性，C/C++ 語言的註解方式如下：

```
/* 我是註解 */
//我是註解
```

上式 /* 符號以後的字串視爲註解，編譯程式不予處理，直到遇到 */ 爲止。還有，同一列中，雙斜線 (//) 後面的也視爲註解，編譯器均不予處理。例如，以下二式分號右邊均爲註解：

```
x= x+ 3;            /* x值加3 */
sum = sum + y;      // 將y之值加至sum
```

前者，因爲可超過兩列，較適合編寫較長的註解，後者，則僅能寫在同一列。

縮排與空行

除了適當的註解，程式設計應善用縮排與空行，才能使你的程式讓人易於閱讀。因爲有了縮排 (請使用 Tab 鍵，而不是自行按空白鍵，因爲空白鍵很費力，且無法完全對齊)，程式才有層次感。其次，空行則可使段落更加明顯。所以，本書範例則都遵守這些規定，請讀者自行觀察。

程式架構介紹

C 的程式架構如下：大致分爲四大區塊，分別是函式引用區、全域變數宣告區、函式實作區、主程式區，如下表所示：

標題	程式
函式引用區（載入所需標頭檔）	`#include <stdio.h>` `#include <stdlib.h>`
全域變數宣告區	`int s;`

函式實作區	`int  add (int a,  int b ){`
	`  int  c;`
	`  c = a + b;`
	`  return (c);`
	`}`

主程式區 main()	`int main(int argc, char *argv[]) {`
	`  s = add (6, 2);`
	`  printf("%d",s);`
	`  return 0;`
	`}`

自我練習

請鍵入以下程式，並觀察執行結果，雙斜線『//』後面爲註解，可以不用輸入。

```
//函式引用區（載入所需標頭檔）
#include <stdio.h>
#include <stdlib.h>
//全域變數宣告區
int s=0;
//函式實作區
int  add (int a,  int b ){
   int  c;
   c = a + b;
   return (c);
}
//主程式 main()區
int main(int argc, char *argv[]) {
   s = add (6, 2);
   printf("%d",s);
   return 0;
}
```

4-3 演算法基本概念與實例探討

演算法基本概念

演算法是指電腦程式完成一項工作所需要有限『步驟』的集合。若從計算的角度，演算法嚴謹定義如下：

> 由有限 (finite) 的步驟 (step) 所構成的集合，依照給定輸入 (input) 依序執行每個明確 (definite) 且有效 (effective) 的步驟，以便能夠解決特定的問題；而步驟的執行必定會終止 (terminate)，並產生輸出 (output)。

演算法的流程控制

常見的演算法流程控制有三種，分別是自然語言、虛擬碼與流程圖。分別說明如下：

自然語言

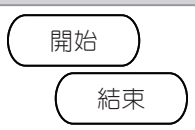
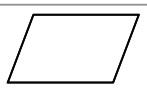
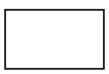
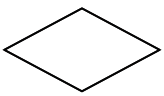
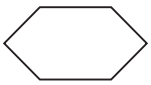
自然語言就是使用我們日常生活的文字表示或已經熟悉的數學語言。例如，範例 4_3c 解一元二次方程式的演算法。

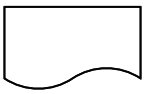
虛擬碼

在自然語言中，有時嵌入一些程式敘述，那會縮短文字長度，也會讓敘述更加清楚易懂。例如，範例 4-3c 的兩數交換演算法與範例 7_4a 的排序演算法。這些演算法裡面除了有文字敘述外，也包括了簡單的迴圈重複指令，那會使得演算步驟更加簡潔又明瞭。

流程圖

流程圖 (flow chart) 是利用各種方塊圖形、線條及箭頭等符號來表達解決問題的步驟及進行的順序，常用的流程符號如下表：

編號	符號	意義
1		起迄符號。 代表流程圖的開始或結束，此符號若是開始，則僅有一出口，若是結束則僅有一入口。
2		輸入與輸出符號。 用來填入輸入與輸出的符號，此符號有一入口與一出口。
3		處理符號。 用來填入處理程序，此符號有一入口與一出口。
4		決策符號。 用來表示決策分歧點，此符號的出口至少有兩個，分別是 true 或 false。
5		重複符號。 聲明重複指令的起點與離開條件，通常配合下面的連結符號表示重複的範圍。
6		連結符號。 可配合上面的重複符號或移至另一頁的起點。

編號	符號	意義
7		副程式。(副程式又稱函式) 用來表示另一個副程式。
8		程式流向符號。 用來連結以上符號，說明程式的流向。
9		代表輸出至螢幕。 將資料由螢幕輸出。
10		代表輸出至印表機。 將資料由印表機輸出。

使用流程圖表示演算法的有範例 5_1a。其次，以下本書的所有範例的演算法，也都會依照範例性質，選用最適當的方法來表示。

範例4-3a

請寫一個程式，可以輸入長方形的長與寬，並計算周長與面積。

演算分析

1. 需要兩個數值輸入。請選擇輸入整數還是浮點數，本例選擇整數。
2. 需要兩個輸出。請選擇整數或是浮點數，本例選擇整數。
3. 變數先用指派的方式輸入，先集中精力在演算法的實現，程式如下：

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
   int a=3,b=4;
   int c,d;
   c=a*b;
   d=2*(a+b);
```

```
   printf("%d",c);
   printf("%d",d);
   return 0;
}
```

4. 套上變數的輸入及輸出提示，程式如下：

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
   int a=3,b=4;
   int c,d;
   printf("input a:");
   scanf("%d",&a);//請留意&
   printf("input b:");
   scanf("%d",&b);//請留意&
   c=a*b;
   d=2*(a+b);
   printf("面積=%d\n",c);
   printf("周長=%d\n",d);
   return 0;
}
```

補充說明

1. 本題也可以使用掌上型計算機，但每計算一個面積，就要按完所有運算元與運算子，但若寫成本例程式，則每次只要輸入運算元就可以，以下各個範例也都是這樣。

自我練習

1. 輸入長方體的長、寬、高，計算其表面積與體積。

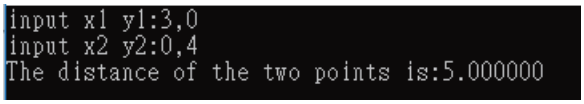
範例4-3b

請寫一程式，滿足以下條件。

1. 可以輸入兩個座標。
2. 計算此兩點座標距離。
3. 輸出此兩點距離。

輸出結果

1. 本例使用 scanf("%d,%d",&x1,&y1) 連續輸入兩個數字，數入數字時要加上逗號『,』隔開，如下圖：



2. 若是使用 scanf("%d%d",&x1,&y1) 連續輸入兩個數字，輸入數字時要加上空白『』隔開。
3. 若是使用 scanf("%d%d",x1,y1) 連續輸入兩個數字，漏掉『&』，那就錯了。

演算法

1. 已知兩點座標分別是 (x1,y1),(x2,y2)，則其距離的公式的數學語言是：

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

2. 以上數學語言的 C 語言如下：

```
d=sqrt(pow((x1-x2),2)+pow((y1-y2),2));
```

程式列印

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>//本例使用sqrt,pow數學函式
int main(int argc, char *argv[]) {
   int x1,y1,x2,y2;
   double d;
   //input
   printf("input x1 y1:");
   scanf("%d,%d",&x1,&y1);//留意&
   printf("input x2 y2:");
   scanf("%d,%d",&x2,&y2);
   //process
   d=sqrt(pow((x1-x2),2)+pow((y1-y2),2));
   //output
   printf("The distance of the two points is:%f\n",d);
   return 0;
}
```

補充說明

1. 數學運算子只有 +,-,*,/,%(取餘數)，其餘數學運算都要使用數學函數，關於數學函數可先翻閱本書 8_1 節，本例 pow() 是次方，sqrt() 是開根號。

自我練習

1. 輸入三角形三邊長 a、b、c，求其面積。(提示：先計算 d=(a+b+c)/2，則三角形面積 $=\sqrt{d(d-a)(d-b)(d-c)}$，本例假設所輸入的三角形三邊長可圍成三角形，例如輸入 3，4，5 則得三角形面積 6，但並不是任意三條線都可圍成三角形，若要判斷是否可圍成三角形，請繼續研讀下一章)

資料的抽象化

若有一直線方程式是 ax+by+c=0，點 p(m,n) 到直線的距離是 $d=\frac{|am+bn+c|}{\sqrt{a^2+b^2}}$（提示：傳回整數絕對值函式是 abs()，傳回實數是 fabs()）這件事要由電腦作，那就是要將資料從人類約定的書寫習慣 ax+by+c=0 獨立抽象出來。例如，直線方程式是 3x+4y+5=0，求點 p(1,2) 到直線的距離，首先，以

```
a=3;b=4;c=5;
```

約定此爲直線

```
3x+4y+5=0
```

然後以

```
m=1;n=2;
```

約定此爲 p(1,2)，所以全部程式如下：

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>//使用fabs()
int main(int argc, char *argv[]) {
   int a,b,c,m,n;
   double d;
   a=3;b=4;c=5;
   m=1;n=2;
   d=fabs(a*m+b*n+c)/(sqrt(a*a+b*b));
   printf("%f",d);
   return 0;
}
```

又例如，您要判斷 q(1,-2) 是否在直線上，也是將資料抽象出來，程式如下：（if 請看下一章）

```
a=3;b=4;c=5;
m=1;n=-2;
if ((a*m+b*n+c)==0)
   printf("yes");
else
   printf("no");
```

以上 3x+4y+5=0 是人類在數學上書寫的約定，電腦只要給 a,b,c，我們就約定此爲 ax+by+c=0，此稱爲資料的抽象化或資料的數位化。

範例4-3c

1. 寫一個程式，可以輸入一個一元二次方程式，並求其解（本例假設所輸入的方程式恰有二解）。
   提示，解一元二次方程式的演算法如下：
   (1) 設有一元二次方程式如下：
   $ax^2+bx+c=0$
   (2) 輸入 a,b,c 三個整數。（本例假設整係數方程式）
   (3) 令 d= $\sqrt{b^2-4ac}$。提示：此爲數學語言，C 語言是 d=sqrt(b*b – 4*a*c);，括號、乘號均不能省。）
   (4) 則其二解分別爲 $x_1=\frac{-b+d}{2a}$，$x_2=\frac{-b-d}{2a}$。（提示：此爲數學語言，C 語言是 x1=(-b+d)/(2*a)，括號、乘號均不能省。）
   (5) 例如，$2x^2-7x+3=0$。其解爲 $x_1=3.0$,$x_2=0.5$。

輸出結果

```
input a,b,c :2,-7,3
x1=3.000000
x2=0.500000
```

程式列印

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main(int argc, char *argv[]) {
   int a,b,c;//係數
   double d,x1,x2;
   printf("input a,b,c :");
   scanf("%d,%d,%d",&a,&b,&c);//留意&
   d=sqrt(b*b-4*a*c);
   x1=(-b+d)/(2*a);//留意所有括號不能漏掉
   x2=(-b-d)/(2*a);//留意所有括號不能漏掉
   printf("x1=%f\n",x1);
   printf("x2=%f\n",x2);
   return 0;
}
```

補充說明

1. 本例假設輸入的係數有實數解，待下一章再判斷所輸入係數有沒有實數解。
2. 國中解一元二次方程式是先用配方法，配方法需要一點判斷，可減少計算，但是電腦可不用如此麻煩，電腦可使用公式法，公式法雖計算較多，但完全不用判斷，最適合由電腦完成。

自我練習

1. 寫一個程式，可以輸入一個二元一次方程式，並求其解（本例假設所輸入的方程式恰有一解）。提示，解二元一次方程式的演算法如下：（此稱爲克拉瑪公式）

   (1) 設二元一次方程式如下：

   $$a_1x + b_1y = c_1$$

   $$a_2x + b_2y = c_2$$

(2) 輸入 a1,b1,c1,a2,b2,c2 六個整數。(本例假設整係數方程式)

(3) 令 $d=\begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix} = a_1b_2 - a_2b_1$。

(4) 則其解分別是 $x=\dfrac{\begin{vmatrix} c_1 & b_1 \\ c_2 & b_2 \end{vmatrix}}{d} = (c_1b_2 - c_2b_1)/d$

(5) 例如，$y=\dfrac{\begin{vmatrix} a_1 & c_1 \\ a_2 & c_2 \end{vmatrix}}{d} = (a_1c_2 - a_2c_1)/d$

3x+y=5

x − 2y=-3

則其解爲 x=1 y=2

補充說明

解二元一次方程式，國中會先教代入消去法與加減消去法，這都需要一點判斷，但可以簡化計算。但是電腦可就神了，電腦可使用公式法，可完全不用判斷，直接計算而得，這樣最適合計算機了。

範例4-3d

請寫一程式，滿足以下條件。

1. 可以輸入兩個數字。
2. 交換此兩個數字。
3. 輸出交換的結果。

演算法則

兩個變數的內容要交換，就如同兩隻手的東西要交換位置。所以假設你有 a,b 兩隻手，各拿一樣東西，那要交換其位置的方法如下：

1. 先找來第 3 隻手 t。
2. 將 a 的東西交給 t 暫存。

```
t=a;
```

3. 將 b 的東西交給 a。

   a=b;

4. 將 t 的東西交給 b，而完成兩隻手東西的交換。

   b=t;

5. 其次，若未先找來第三之手 t 幫忙，電腦並沒有這個能力，同時拋出雙手東西，再同時接住另一手的東西，如以下敘述：

   a=b;

   b=a;

6. 這就是一種運算思維，電腦運算要很明確。

輸出結果

```
input a,b: 3,4
after swap a=4 , b=3
```

程式列印

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
   //declare
   int a, b, t;
   //input
   printf("input a,b: ");
   scanf("%d, %d", &a,&b);
   /* swap */
   t=a;
   a=b;
   b=t;
   //output
   printf("after swap a=%d , b=%d", a, b);
   return 0;
}
```

自我練習

1. 輸入三個數，並將 1 交給 2，2 交給 3 ，3 交給 1，並輸出。

範例4-3e

時間差的計算。假設某一汽車從 h1=3,m1=20 進入，h2=5,m2=10 離開，請寫一程式可以計算其停留時間。

執行結果

```
t=110 分,等於 1:50
```

演算法

關於兩個時間差的計算，小學是用直式算法，若分鐘不夠要借位，但電腦就強了，直接通通化爲分鐘，就可相減。若是兩個日期差則可通通轉爲日，關於兩個日期差請看第 7 章。所以學電腦程式設計，就要慢慢調整成電腦的運算思維。

其次，甚至您還要往前調整整個運算思維，因爲時間的儲存與表示，要用一個長整數『秒』表示就好，不要分別用年、月、日表示，因爲『年月日時分秒』僅是方便給人看的，電腦要用一個變數『秒』儲存與運算就好，請繼續看以下各章就會明瞭。

程式列印

```
int main(int argc, char *argv[]) {
   int h1=3,m1=20;
   int h2=5,m2=10;
   int t1,t2,t,h,m;
   t1=h1*60+m1;
   t2=h2*60+m2;
   t=t2-t1;
   h=t/60;
   m=t %60;
```

```
   printf("t=%d 分,等於 %d:%d",t,h,m);
   return 0;
}
```

範例4-3f

假設某次考試成績資料如下：

55、66、77、88、99

(1) 請寫一程式輸入以上資料。

(2) 計算總和。

(3) 輸出總和與平均。

輸出結果

```
input number:55
input number:66
input number:77
input number:88
input number:99
The sum is 385,The average is77
```

程式列印

```
int main(int argc, char *argv[]) {
   int a,sum=0;
   printf("input number:");
   scanf("%d",&a);
   sum=sum+a;

   printf("input number:");
   scanf("%d",&a);
   sum=sum+a;

   printf("input number:");
   scanf("%d",&a);
   sum=sum+a;

   printf("input number:");
```

```
    scanf("%d",&a);
    sum=sum+a;

    printf("input number:");
    scanf("%d",&a);
    sum=sum+a;

    printf("The sum is %d,The average is%d",sum,sum/5);
    return 0;
}
```

補充說明

1. 本例只有 5 筆資料，程式就落落長，那如果有 50 筆，甚至 10000 筆，那不就操死程式設計師，請大家放心，待學習第六、七兩章的迴圈與另外一種資料結構陣列，就會豁然開朗。

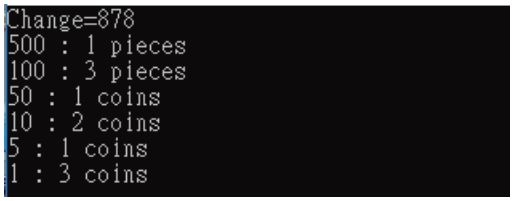
自我練習

1. 每一個人請挑一個物理、數學或基本電學公式，設計一個程式，計算其結果。例如，自由落體的距離與時間關係，公式爲 $d=v_0t+gt^2/2$，那就要能輸入 v_0,t，設定常數 g，並輸出距離 d。
2. 請分四次輸入 1 個 0 到 9 的整數，並將它合併爲 1 個整數。例如，輸入 1，輸入 2，輸入 3，輸入 4，則輸出 1234。
3. 同上習題，請先輸入 3 個 0 到 9 的整數，再輸入兩個 0 到 9 的整數，並將其合併爲 1 個浮點數。例如，輸入 1，輸入 2，輸入 3，輸入 4，輸入 5，則輸出 123.45。
4. 請輸入 1 個 4 位數，並將其分解輸出如 $a1*10^3+a2*10^2+a3*10^1+a4$。例如，輸入 1234，則輸出 $1*10^3 + 2*10^2 + 3*10^1 + 4$。

範例4-3g

找零問題。請寫一程式，可以協助付款 1000 元時，計算所需找回的錢，並以最節省鈔票與硬幣數量的方式找錢。（APCS105 試題）

執行結果



程式列印

1. 以下是考試試題的寫法：

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
   int Total, Paid, Change;
   Total=122;
   Paid=1000;
   Change=Paid-Total;
   printf("Change=%d\n",Change);
   printf ("500 : %d pieces\n", (Change-Change%500)/500);
   Change=Change%500;
   printf ("100 : %d pieces\n", (Change-Change%100)/100);
   Change=Change%100;
   // A 區
   printf ("50 : %d coins\n", (Change-Change%50)/50);
   Change=Change%50;
   // B 區
   printf ("10 : %d coins\n", (Change-Change%10)/10);
   Change=Change%10;
   // C 區
   printf ("5 : %d coins\n", (Change-Change%5)/5);
   Change=Change%5;
   // D 區
   //printf ("1 : %d coins\n", (Change-Change%1)/1);
   printf ("1 : %d coins\n", Change);
```

```
   //Change=Change%1;
   return 0;
}
```

2. 寫程式變數盡量要有意義，且全部處理完了，再一起輸出，這樣比較好讀，所以程式我重寫如下：

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   int a=1000;
   int b=22;
   int c=a-b;
   int a500=c/500;
   c=c-500*a500;
   int a100=c/100;
   c=c-100*a100;
   int a50=c/50;
   c=c-50*a50;
   int a10=c/10;
   c=c-10*a10;
   int a5=c/5;
   c=c-5*a5;
   int a1=c;//剩下的一定小於5
   printf("the change is %d\n",a-b);
   printf("500:%d\n",a500);
   printf("100:%d\n",a100);
   printf("50:%d\n",a50);
   printf("10:%d\n",a10);
   printf("5:%d\n",a5);
   printf("1:%d\n",a1);
   return 0;
}
```

3. 以上輸出結果如下：

```
the change is 978
500:1
100:4
50:1
10:2
5:1
1:3
```

4. 找零的方法很多，以上又稱爲貪婪法，因爲每一刀下去，都得到一個最佳的結果。

4-4 APCS觀念題

1. 右側程式碼執行後輸出結果爲何？ 105/10/04

(A)3

(B)4

(C)5

(D)6

```
int a=2, b=3;
int c=4, d=5;
int val;

val = b/a + c/b + d/b;
printf ("%d\n", val);
```

參考解答

此題重點是被除數與除數都是整數，所以商一定是取整數，所以答案是（A）。

---

2. 假設x,y,z爲布林(boolean)變數，且x=TRUE, y=TRUE, z=FALSE。請問下面各布林運算式的眞假值依序爲何？(TRUE 表眞，FALSE 表假) 105/10/14

!(y || z) || x

!y || (z || !x)

z || (x && (y || z))

(x || x) && z

(A) TRUE FALSE TRUE FALSE

(B) FALSE FALSE TRUE FALSE

(C) FALSE TRUE TRUE FALSE

(D) TRUE TRUE FALSE TRUE

參考解答

這題沒有問題，請看 4_1 節，依照關係與邏輯運算子運算規則與優先順序與結合律運算，答案是（A）。

---

3. 下列程式碼是自動計算找零程式的一部分，程式碼中三個主要變數分別爲 Total (購買總額)，Paid (實際支付金額)，Change (找零金額)。但是此程式片段有冗餘的程式碼，請找出冗餘程式碼的區塊。　105/10/19

(A) 冗餘程式碼在 A 區　(B) 冗餘程式碼在 B 區

(C) 冗餘程式碼在 C 區　(D) 冗餘程式碼在 D 區

```
int Total, Paid, Change;
   …
Change = Paid - Total;
printf ("500 : %d pieces\n", (Change-Change%500)/500);
Change = Change % 500;

printf ("100 : %d coins\n", (Change-Change%100)/100);
Change = Change % 100;

// A 區
printf ("50 : %d coins\n", (Change-Change%50)/50);
Change = Change % 50;

// B 區
printf ("10 : %d coins\n", (Change-Change%10)/10);
Change = Change % 10;

// C 區
printf ("5 : %d coins\n", (Change-Change%5)/5);
Change = Change % 5;

// D 區
printf ("1 : %d coins\n", (Change-Change%1)/1);
Change = Change % 1;
```

參考解答

扣掉 5 元硬幣以後，最後只剩小於等於 4 了，那都是一元硬幣的數量了，所以『D 區』是多餘。

---

4. 如果 X_n 代表 X 這個數字是 n 進位，請問 $D02A_{16} + 5487_{10}$ 等於多少？　105/10/22

(A) $1100\ 0101\ 1001\ 1001_2$　(B) 162631_8

(C) 58787_{16}　(D) $F599_{16}$

參考解答

本題把所有數字都轉爲十進位，就可找出解答。

進位轉換，十六進位用 ABCDEF 分別表示 10,11,12,13,14,15，

所以 $D02A_{16}+5487_{10}$

$=13*16^3+0*16^2+2*16^1+10*16^0+5487=53290+5487=58777$

162631_8

$=1*8^5+6*8^4+2*8^3+6*8^2+3*8^1+1=58777$

---

5. 若 A[1]、A[2]， 和 A[3] 分 別 爲 陣 列 A[] 的 三 個 元 素 (element)，下列那個程式片段可以將 A[1] 和 A[2] 的內容交換？ 106/03/11

(A) A[1] = A[2]; A[2] = A[1];

(B) A[3] = A[1]; A[1] = A[2]; A[2] = A[3];

(C) A[2] = A[1]; A[3] = A[2]; A[1] = A[3];

(D) 以上皆可

參考解答

此題純粹兩個變數的交換，請看範例 4-3d。

---

6. 若函式 rand() 的回傳值爲一介於 0 和 10000 之間的亂數，下列那個運算式可產生介於 100 和 1000 之間的任意數 (包含 100 和 1000) ？ 106/03/12

(A) rand() % 900 + 100

(B) rand() % 1000 + 1

(C) rand() % 899 + 101

(D) rand() % 901 + 100

參考解答

此題爲數學問題。因爲

0<=rand()<=10000

0%901<=rand()_%901<=10000%901// 大家都 %901

0<=rand()%901<=900

0+100<=rand()%901+100<=900+100// 大家都加 100

100<=rand()%901+100<=1000

---

7. 若要邏輯判斷式 !(X_1 || X_2) 計算結果爲眞 (True)，則 X_1 與 X_2 的值分別應爲何？ 106/03/21

(A) X_1 爲 False，X_2 爲 False　(B) X_1 爲 True，X_2 爲 True

(C) X_1 爲 True，X_2 爲 False　(D) X_1 爲 False，X_2 爲 True

參考解答

本題建議將解答一一代入題目，請看 4_1 節就可找到（A）

---

8. 若 a, b, c, d, e 均爲整數變數，下列哪個算式計算結果與 a+b*c-e 計算結果相同？ 106/03/25

(A) (((a+b)*c)-e)　(B) ((a+b)*(c-e))

(C) ((a+(b*c))-e)　(D)(a+((b*c)-e))

參考解答

此題重點是運算子的優先順序，乘法比加減法優先，請看 4_1 節。

自我練習

以下是我參加 APCS 考試，聯想的一些 APCS 模擬試題，其實寫程式沒人這樣寫，但這樣才能測驗學生對本章的瞭解程度。請先紙上寫出答案，再在電腦執行比對答案，這樣才會進步。其次，以下是 C++ 的程式，輸出入較簡潔，cout 是輸出，需要配合

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

endl 是跳一列並歸位到最左邊。若要實習，那也請於開新專案時，點選『C++』，其餘都相同。

題號	題目	結果

1.	//Arithmetic Operator
	//The Precedence of Arithmetic
	//The Associativity of Arithmetic
	#include <iostream>
	using namespace std;
	int main(int argc, char** argv) {
	int a=1,b=2,c=3,d=4,e=5;
	cout<<a*b-c*-d<<endl;
	cout<<a/b*c+d+ ++e<<endl;
	a=(b=2)+(c=3);
	cout<<a<<endl; //上式括號可否去掉
	
	a=1;b=2;c=3;
	a+=b+=c+=5;
	cout<<"a="<<a<<endl;
	cout<<"b="<<b<<endl;
	cout<<"c="<<c<<endl;
	
	a=1;b=2;c=3;	

1.

```
   a=++b + ++c;
   cout<<a<<","<<b<<","<<c<<endl;
   a=b++ +c++;
   cout<<a<<","<<b<<","<<c<<endl;
   a=++b + c++;
   cout<<a<<","<<b<<","<<c<<endl;
   a=b-- + --c;
   cout<<a<<","<<b<<","<<c<<endl;

   a=1;b=2;c=3;
   d+=b+c;
   cout<<"d="<<d<<endl;
   d*=a=b+5;
   cout<<"d="<<d<<endl;
   return 0;
}
```

2.

```
//Relational Operator
#include <iostream>
using namespace std;
int main(int argc, char** argv) {
   int a=1;
   cout<<(a>4)<<endl;
   cout<<(a>=0)<<endl;
   cout<<(a==1)<<endl;
   cout<<(a=5)<<endl;
   cout<<(a!=1)<<endl;
   //cout<<(a=>4)<<endl;
   //cout<<(a=!5)<<endl;
   int a=1,b=2,c=3,d=4,e=5;
   c=(a+3)==b;
   cout<<c<<endl;
   c=(a+1)==b;
   cout<<c<<endl;
   return 0;
}
```

3.

```
#include <iostream>
using namespace std;
//Logital Operator
//The Precedence of Logical
//The Associativity of Logical
int main(int argc, char** argv) {
   int a=5,b=4,c=3,d=2,e=1;
   cout<<((a!=b) & (b>=3))<<endl;
   cout<<((a!=b) ^ !(b>=3))<<endl;
   cout<<(!(a!=b) | !(b>=3))<<endl;
   cout<<((a==1 )& (a>0) && (b>=2)<<endl;
   cout<<((a==1 )| (a>0) & (b>2) ^(c==3))<<endl;
   cout<<((a==1 ) || (a>0) && (b>2)
^(c==3))<<endl;
   return 0;
}
```

4.

```
//The Bitwise Operator，逐位元運算
#include <iostream>
using namespace std;
int main(int argc, char** argv) {
   int a=3,b=-2,c=3,d=4;
   cout<<~a<<endl;
   cout<<~b<<endl;
   cout<<(a<<2)<<endl;//左移2位元
   cout<<(a>>2)<<endl;//右移2位元
   cout<<(b<<2)<<endl;
   cout<<(b>>2)<<endl;
   cout<<(a & b)<<endl;//逐位元and運算
   cout<<(a | b)<<endl; //逐位元or運算
   cout<<(a ^ b)<<endl; 逐位元互斥或運算
   cout<<(a & b | c ^ d)<<endl;
   a<<=3; //左移3位元，並放回
   cout<<a<<endl;
   b>>=2;
   cout<<a<<endl;
   return 0;
}
```

5.

```
#include <iostream>
using namespace std;
int main(int argc, char** argv) {
   int a=4;b=5,t;
   t=a;a=b;b=t;
   cout<<a<<endl;
   cout<<b<<endl;
   return 0;
}
```

習 題

複數的表示法有直角座標的標準式與極座標的極式，兩者的關如下：

$$\vec{A} = A\angle\theta = a + bi$$

$$=> A = \sqrt{a^2 + b^2}$$

$$\theta = \tan^{-1}\frac{b}{a}$$

$$a = A\cos\theta \quad , b = A\sin\theta$$

1. 請設計一個程式，可以將直角座標轉為極座標。例如，輸入 3+4*i* 可得到 5 ∠ 60。（三角函數請看 8-1 節）
2. 請設計一個程式，可以將極座標轉為直角座標。例如，輸入 5 ∠ 30 可得到 4+3i。
3. 請設計一程式，可以計算兩個以標準式 (3+4*i*) 表示的複數的加、減、乘、除等運算。假設 $\vec{P} = a + bi, \vec{Q} = c + di,$ 則以上運算方法如下：
   (1) $\vec{P} + \vec{Q}$ =(*a*+*c*)+(*b*+*d*)*i*
   (2) $\vec{P} - \vec{Q}$ =(*a*-*c*)+(*b*-*d*)*i*
   (3) $\vec{P} * \vec{Q}$ =(*ac*-*bd*)+(*ad*+*bc*)*i*
   (4) $\vec{P} / \vec{Q}$ =((*ac*+*bd*)+(*bc*-*ad*))*i*/(*a***a*+*d***d*)
4. 請設計一程式，可以計算兩個以極式 (5 ∠ 30) 表示的複數的加、減、乘、除等運算。假設 $\vec{P} = r1\angle\theta_1, \vec{Q} = r2\angle\theta_2$ 則以上運算方法如下：
   (1) $\vec{P} + \vec{Q}$ 與 $\vec{P} - \vec{Q}$ 請先將極式轉為標準式，再相加減，最後再轉為極式。
   (2) $\vec{P} * \vec{Q}$ =(r_1*r_2) ∠ (θ_1+ θ_2)
   (3) $\vec{P} / \vec{Q}$ =(r_1/r_2) ∠ (θ_1- θ_2)

MEMO