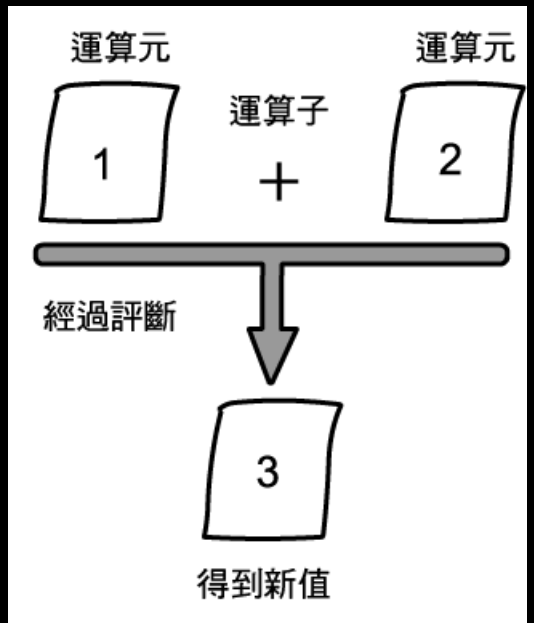
運算式和運算子

4-1 運算式與運算子

- 關於運算式
- 將運算式的新值輸出
- 進行各種運算
- 從鍵盤輸入值來進行加法運算

- 關於運算式
  - C語言運算式都是由下列2種元素所構成：
    - 運算子（operator）：負責演算的加減乘除等符號
    - 運算元（operand）：計算執行的對象（通常是數字或文字）
- 將運算式的新值輸出：
  - printf("1+2等於%d。\n", 1+2);
  - printf("3x4等於%d。\n", 3*4);



Sample1.c ▶ 將運算式的新值輸出

```
#include <stdio.h>

int main(void)
{
    printf("1+2 等於 %d。\n", 1+2);
    printf("3 × 4 等於 %d。\n", 3*4);

    return 0;
}
```

在此輸入 1+2 這樣的運算式

- 進行各種運算
  - 在運算式當中，運算元 (計算執行的對象)並非只侷限使用1或2這些數字，事實上運算元可以是其他各種型態的資料。
  - 範例：

```
…
int num1 = 2;
int num2 = 3;
int sum = num1+num2;
printf("變數num1的值是%d。\n", num1);
printf("變數num2的值是%d。\n", num2);
printf("num1+num2的值是%d。\n", sum);
num1 = num1+1;
printf("變數num1的值加1後是%d。\n", num1);
```

num1+num2之後，再指定給另一個變數**sum**

num1+1之後，再將結果指定給變數**num1**

Sample2.c ▶ 使用變數作為運算元

```
#include <stdio.h>

int main(void)
{
    int num1 = 2;
    int num2 = 3;
    int sum = num1+num2;

    printf("變數num1的值是%d。\n", num1);
    printf("變數num2的值是%d。\n", num2);
    printf("num1+num2的值是%d。\n", sum);

    num1 = num1+1;

    printf("變數num1的值加1後是%d。\n", num1);

    return 0;
}
```

① num1+num2之後，再指定給另一個變數sum

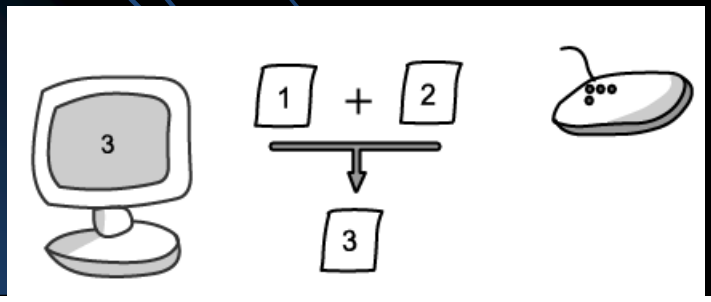
② num1+1之後，再將結果指定給變數num1

- 從鍵盤輸入值來進行加法運算
  - 利用變數進行運算時，只要適時變更變數所代表的值，則變數運算後得到的結果也會不同。

```
…
int num1, num2;
printf("請輸入第1個整數：\n");
scanf("%d", &num1);
printf("請輸入第2個整數：\n");
scanf("%d", &num2);
printf("相加之後的結果是%d。\n", num1+num2);
```

從鍵盤輸入的值經過轉換後，指定給變數**num1**和**num2**

以**num1**和**num2**所代表的數值進行加法運算並輸出結果



Sample3.c ▶ 加法運算程式

```
#include <stdio.h>

int main(void)
{
     int num1, num2;

     printf("請輸入第 1 個整數：\n");

     scanf("%d", &num1);

     printf("請輸入第 2 個整數：\n");

     scanf("%d", &num2);

     printf("相加之後的結果是 %d 。\n", num1+num2);

     return 0;
}
```

從鍵盤輸入的值經過轉換後，指定給變數 num1 和 num2

以 num1 和 num2 所代表的數值進行加法運算並輸出結果

4-2 運算子的種類

- 各種形式的運算子

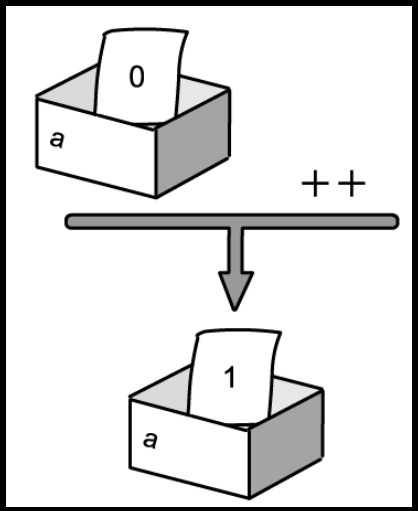
運算子	說明	運算子	說明
+	加法運算子	>=	關係運算子（大於等於）
-	減法運算子	<	關係運算子（小於）
*	乘法運算子	<=	關係運算子（小於等於）
/	除法運算子	==	相等運算子
%	餘數運算子	!=	不相等運算子
+	正號（單獨使用）	!	邏輯NOT運算子
-	負號（單獨使用）	&&	邏輯AND運算子
~	位元補數運算子	||	邏輯OR運算子
&	位元AND運算子	*	間接參照運算子
|	位元OR運算子	,	逗號運算子
^	位元XOR運算子	()	呼叫函數
=	指定	()	cast
<<	位元左移運算子	[]	陣列註標
>>	位元右移運算子	.	成員選擇運算子
++	前置遞增運算子	->	成員選擇運算子
--	前置遞減運算子	?:	條件運算子
>	關係運算子（大於）	sizeof	型別size運算子

Sample4.c ▶ 使用各種運算子

```
#include <stdio.h>

int main(void)
{

    int num1 = 10;
    int num2 = 5 ;

    printf("num1 與 num2 的各種運算：\n");
    printf("num1+num2 等於 %d 。\n", num1+num2);
    printf("num1-num2 等於 %d 。\n", num1-num2);
    printf("num1*num2 等於 %d 。\n", num1*num2);
    printf("num1/num2 等於 %d 。\n", num1/num2);
    printf("num1%%num2 等於 %d 。\n", num1%num2);

    return 0;
}
```

進行各種運算

- 運算子的種類
  - 運算子可依照搭配的運算元之數量，分爲一元運算子、二元運算子、三元運算子。
- 遞增運算子和遞減運算子
  - 遞增運算子(increment operator)，顧名思義就是會讓變數的值加1。
    - a++;
    - a = a+1;
  - 遞減運算子(decrement operator)則相反。



- 遞增（或遞減）運算子的「前置」與「後置」
  - ++a
  - a++
  - b = ++a;
    - 先將變數a的值指定給變數b後 → 變數a的值再加1
    - 變數a的值加1後 → 再指定給變數b

  - 前置遞增運算子是先遞增之後才指定。
  - 後置遞增運算子則是先指定之後才遞增。

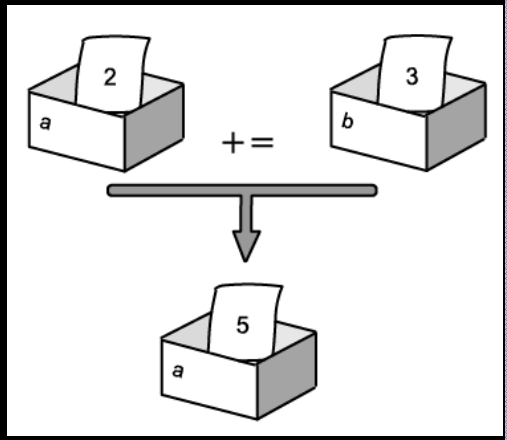
Sample5.c ▶ 前置與後置遞增運算子的應用

```
#include <stdio.h>

int main(void)
{

    int a = 0;
    int b = 0;

    b = a++;

    printf("指定後置遞增運算子到b之後的值為%d。\n", b);

    return 0;
}
```

使用後置遞增運算子

- 指定運算子
  - 指定運算子的功用是將右邊的值(不管文字或數字)指定給左邊的變數。
  - a += b;
    - 上述的「+=」運算子會將變數a與變數b進行加法運算，並將得到的新值再指定給變數a。

指定運算子	說明
+=	加法後指定數值給變數
-=	減法後指定數值給變數
*=	乘法後指定數值給變數
/=	除法後指定數值給變數
%=	求餘後指定數值給變數
&=	邏輯AND後指定數值給變數
^=	邏輯XOR後指定數值給變數
|=	邏輯OR後指定數值給變數
<<=	位元左移後指定數值給變數
>>=	位元右移後指定數值給變數



Sample6.c ▶ 使用複合指定運算子

```
#include <stdio.h>

int main(void)
{
    int sum = 0;
    int num = 0;

    printf("請輸入第 1 個整數：\n");
    scanf("%d", &num);
    sum += num;

    printf("請輸入第 2 個整數：\n");
    scanf("%d", &num);
    sum += num;

    printf("請輸入第 3 個整數：\n");
    scanf("%d", &num);
    sum += num;

    printf("3 個整數的合計是 %d 。\n", sum);

    return 0;
}
```

這裏使用複合指定運算子

- sizeof運算子
  - 使用sizeof運算子可以瞭解各種資料型態及運算式的值的大小。

- 位移運算子
  - 「位移運算」就是將10進制的數值轉換成2進制數值之後，根據指定的位移數目，在2進制數值的位數中向左或向右移動（位移），以求取新值。
  - 5<<2的運算過程如下：

	5	0000000000000101
<<	2	
	20	0000000000010100

原本的數字向左移動兩格，右邊則補上兩個**0**

  - 5>>2的運算過程如下：

	5	0000000000000101
>>	2	
	20	0000000000000001

原本的數字向右移動兩格，左邊則補上兩個**0**

Sample7.c ▶ 使用 sizeof 運算子

```
#include <stdio.h>

int main(void)
{
    int a = 1;
    int b = 0;

    printf("short int 型態的大小為 %dbyte。\n",
           sizeof(short int));
    printf("int 型態的大小為 %dbyte。\n", sizeof(int));
    printf("long int 型態的大小為 %dbyte。\n", sizeof(long int));
    printf("float 型態的大小為 %dbyte。\n", sizeof(float));
    printf("double 型態的大小為 %dbyte。\n", sizeof(double));
    printf("long double 型態的大小為 %dbyte。\n",
           sizeof(long double));
    printf("變數 a 的大小為 %dbyte。\n", sizeof(a));
    printf("運算式 a+b 的大小為 %dbyte。\n", sizeof(a+b));

    return 0;
}
```

查看資料型態的大小

查看運算式的大小

4-3 運算子的執行優先順序

- 運算子的優先順序爲何？
  - 撰寫程式時，要特別注意運算子執行時的優先順序。使用括弧可以改變執行時的優先順序。
  - 優先順序的相關說明可參考下頁。
- 當運算子的優先順序相同時
  - 假設是同一個運算式中同時有加減乘除，那麼就應該遵循由左至右（left associative）的原則。
  - 由右而左（right associative）執行的運算式，大多是指定(=)運算式。

表 4-3：運算子的執行優先順序（2 條實線之間的運算子，彼此的優先順序相同）

運算子	名稱	結合規則
()	呼叫函數	左
[]	陣列標註	左
.	成員選擇	左
->	成員選擇	左
++	後置遞增	左
--	後置遞減	左
!	邏輯 NOT	右
~	位元補數	右
+	正號	右
-	負號	右
sizeof	資料長度	右
++	前置遞增	右
--	前置遞減	右
&	位址	右
*	間接參照	右
()	成員（cast）	右
%	餘數	左
*	乘法	左
/	除法	左
+	加法	左
-	減法	左
<<	位元左移	左
>>	位元右移	左
>	大於	左
>=	大於等於	左
<	小於	左
<=	小於等於	左
==	相等	左
!=	不等於	左
&	AND 位元運算	左
|	OR 位元運算	左
^	XOR 位元運算	左
&&	邏輯 AND	左
||	邏輯 OR	左
? :	條件式	右
=	指定	右
,	逗號	左

4-4 資料型態的轉換

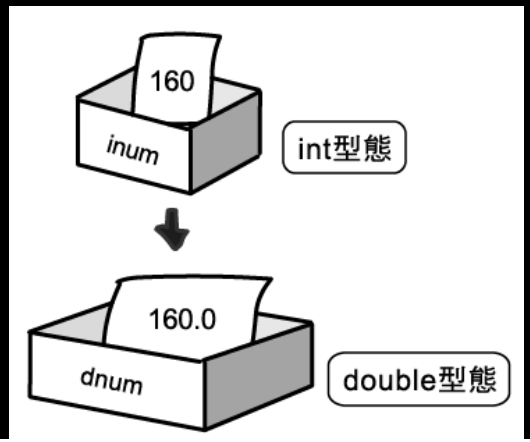
- 指定給資料型態較大的變數
- 指定給資料型態較小的變數
- cast運算子
- 不同類型的資料進行運算
- 同類型的資料進行運算

- 指定給資料型態較大的變數之範例：

```
…
int inum;
double dnum;
inum = 160;
printf("身高是%d公分。\n", inum);
printf("指定成double型態的變數。\n"
dnum = inum;
printf("身高是%f公分。\n", dnum);
```

- 結果:

身高是160公分。
指定成double型態的變數。
身高是**160.000000**公分。



Sample8.c ▶ 將原來的數字指定給資料型態較大的變數

```
#include <stdio.h>

int main(void)
{
    int inum;
    double dnum;

    inum = 160;

    printf("身高是 %d 公分。\n", inum);

    printf("指定成 double 型態的變數。\n");

    dnum = inum;

    printf("身高是 %f 公分。\n", dnum);

    return 0;
}
```

指定給資料型態較大的變數

- 指定給資料型態較小的變數之範例：

```
…
double dnum;
int inum;
dnum = 160.5;
printf("身高是%f公分。\n", dnum);
printf("指定成int型態的變數。\n");
inum = dnum;
printf("身高是%d公分。\n", inum);
```

- 結果:

身高是160.500000公分。
指定成int型態的變數。
身高是**160**公分。

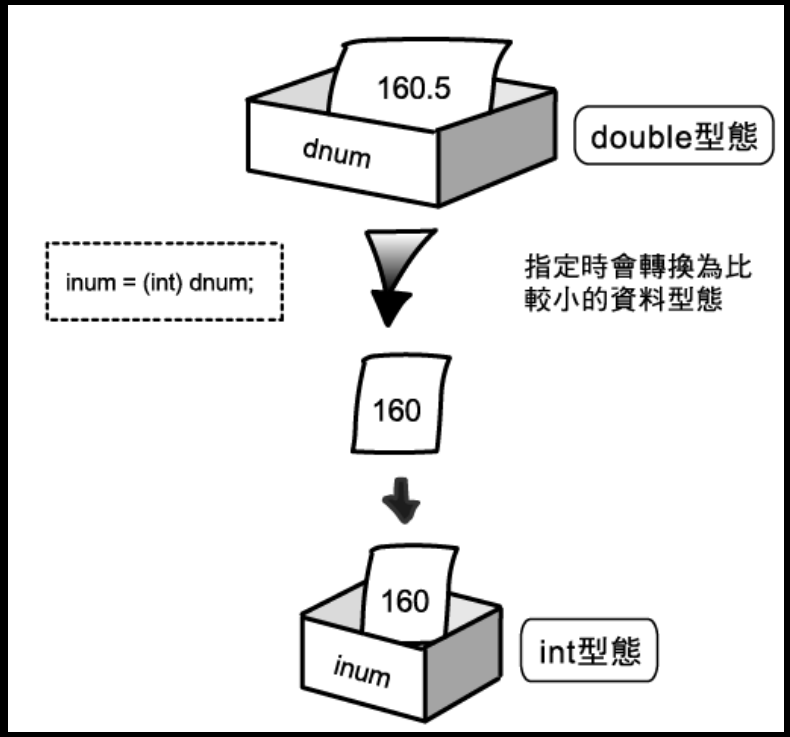
Sample9.c ▶ 指定給資料型態較小的變數

```
#include <stdio.h>

int main(void)
{
    double dnum;
    int inum;

    dnum = 160.5;

    printf("身高是%f公分。\n", dnum);

    printf("指定成int型態的變數。\n");

    inum = dnum;

    printf("身高是%d公分。\n", inum);

    return 0;
}
```

將原來的數值指定給資料型態較小的變數後…

- cast運算子
  - 當要轉換資料型態的時候，必須清楚的明示即將轉換的資料型態，所以也叫強制轉型運算子
  - 語法：

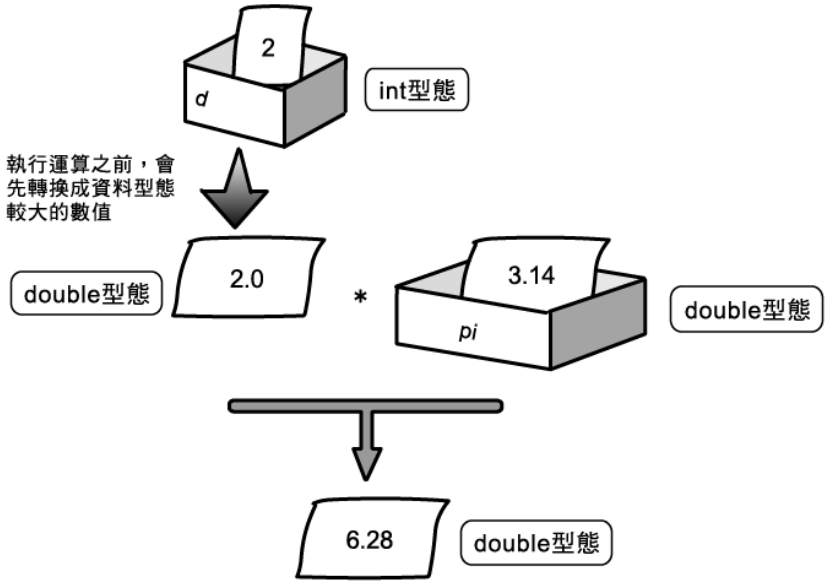
    (資料型態) 運算式



- 不同型態的資料如何進行運算？
  - 運算元的資料型態不同時，通常會先轉換成資料型態較大的數值（由小轉大）。運算後得到的結果也是屬於較大資料型態的類型。
  - 範例：

```
…
int d = 2;
double pi = 3.14;
printf("直徑是%d公分的圓。\n", d);
printf("其圓周爲%f公分。\n", d*pi);
```

int型態的變數d會轉換成double型態後再進行運算



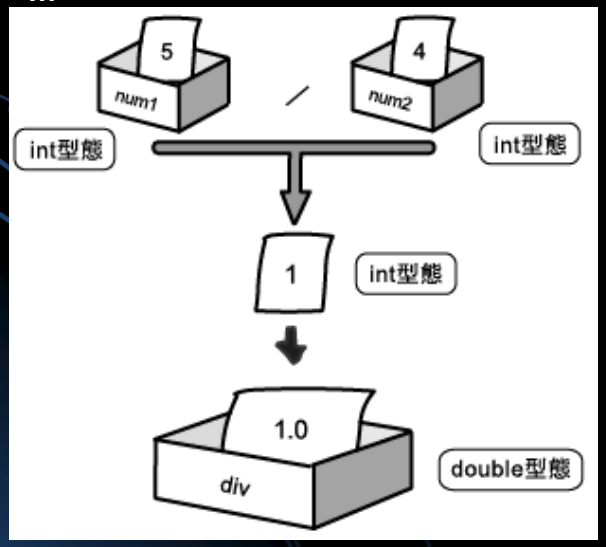
Sample10.c ▶ 不同型態資料的運算

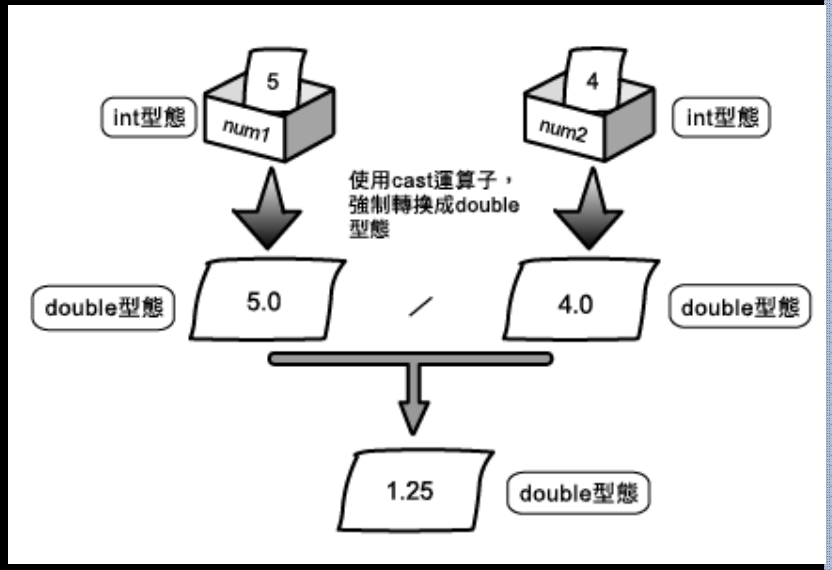
```
#include <stdio.h>

int main(void)
{
    int d = 2;
    double pi = 3.14;

    printf("直徑是 %d 公分的圓。\n", d);
    printf("其圓周為 %f 公分。\n", d*pi);

    return 0;
}
```

int 型態的變數 d 會轉換成 double 型態後再進行運算

- 同類型的資料進行運算
  - 範例：

```
...
int num1, num2;
double div;
num1 = 5;
num2 = 4;

div = num1 / num2;                    //div = 1
…
div = (double)num1 / (double)num2;      //div =
 1.250000
…
```





Sample11.c ▶ 同類型的資料運算

```
#include <stdio.h>

int main(void)
{
    int num1, num2;
    double div;

    num1 = 5;
    num2 = 4;

    div = num1 / num2;

    printf("5/4 為 %f 。\n", div);

    return 0;
}
```

原本是想求 5 ÷ 4 的結果，但是…