第三章

資料型態與運算式

前言

- 資料處理是C程式的核心，它包含了兩大部分：資料宣告與資料運算。
  - 資料以變數來加以儲存，並且需要經過宣告後才可以使用。
  - 資料運算則是利用C所提供的眾多運算子來處理資料。在這一章中，我們將說明這兩大重點。

大綱

- 3.1 基本資料型態
  - 3.1.1 int整數資料型態
  - 3.1.2 float單精準度浮點數資料型態
  - 3.1.3 double雙精準度浮點數資料型態
  - 3.1.4 char字元資料型態
  - 3.1.5 void資料型態
  - 3.1.6 bool資料型態【C++補充】
- 3.2 基本資料型態的修飾字
  - 3.2.1 short修飾字
  - 3.2.2 long修飾字
  - 3.2.3 unsigned修飾字
  - 3.2.4 signed修飾字

大綱（續）

- 3.3變數與常數
  - 3.3.1　變數的意義
  - 3.3.2　變數的命名
  - 3.3.3　變數的宣告
  - 3.3.4　常數的意義
  - 3.3.5　常數的種類

大綱（續）

- 3.4 運算子及運算元
  - 3.4.1 『=』設定運算子（Assignment Operator）
  - 3.4.2 算數運算子（Arithmetic Operator）
  - 3.4.3 比較運算子(Comparison Operator)
  - 3.4.4 邏輯運算子(Logical Operator)
  - 3.4.5 位元邏輯運算子(Bitwise Logical Operator)
  - 3.4.6 複合式指定運算子
  - 3.4.7 遞增與遞減運算子
  - 3.4.8 其他運算子
  - 3.4.9 運算子結合及優先權

大綱（續）

- 3.5資料型態的轉換
  - 3.5.1　運算式的型態轉換
  - 3.5.2　強制型態轉換
  - 3.5.3　使用函式轉換資料型態
- 3.6本章回顧

3.1 基本資料型態

- C語言的基本資料型態共有int、float、double、char、void等五種，隨著編譯器及作業系統的不同，每種資料型態所占的記憶體空間也有些許差異。由於變數宣告時，必須指定資料型態，因此在本節與下一節中，我們將一一加以討論與說明這些資料型態。

3.1.1 int整數資料型態

- 整數資料型態可用來代表帶有正負號的整數。在C程式中的整數資料型態可表示的數值範圍，隨著硬體、作業系統、編譯器而有所不同。

3.1.1 int整數資料型態（續）

- 在Windows與Linux 32位元作業系統中，使用4個位元組（32個位元）來儲存宣告爲int資料型態的變數，因此，在32位元的作業系統中int資料型態的範圍是-2147483648 ~ +2147483647，共有4294927696個整數值。計算方式如下：

$$-2^{32-1} \sim 2^{32-1}-1 \Rightarrow -2^{31} \sim 2^{31}-1$$

$$\Rightarrow -2147483648 \sim 2147483647$$

3.1.1　int整數資料型態（續）

- 【工具】：

  您可以編譯且執行datatype.c，該程式將顯示C程式的各種資料型態在該環境下，佔用記憶體多少個位元組。

  - 執行結果：（在Windows xp下使用Dev-C++編譯）

3.1.1 int整數資料型態（續）

```
1   /*datatype.c*/
2
3   #include <stdio.h>
4   #include <stdlib.h>
5
6   int main(void) {
7   printf("型態\t\t大小（bytes）\n");
8   printf("short\t\t%d\n", sizeof(short));
9   printf("int\t\t%d\n", sizeof(int));
10  printf("long\t\t%d\n", sizeof(long));
11  printf("float\t\t%d\n", sizeof(float));
12  printf("double\t\t%d\n", sizeof(double));
13  printf("long double\t%d\n", sizeof(long double));
14  printf("char\t\t%d\n", sizeof(char));
15  return 0;
    }
```

3.1.1 int整數資料型態

```
型態            大小（bytes）
short           2
int             4
long            4
float           4
double          8
long double     8
char            1
Press any key to continue
```

3.1.2 float單精準度浮點數資料型態

- 浮點數是可包含小數的正負數值，一般分爲單精準度與雙精準度兩種。
- 在C語言中float資料型態可用來存放宣告爲單精準度浮點數的變數。浮點數資料型態在IEEE協會已經制定了固定規格，而C語言也依照該規格來設計float資料型態。float資料型態的變數佔記憶體4個位元組（32位元），在正數方面可表達3.4E-38~3.4E+38（E後面的數字代表10的次方數）。

3.1.3 double雙精準度浮點數資料型態

- double則是用來表示雙精準度浮點數的資料型態，經由宣告爲double資料型態的變數將佔用記憶體8個位元組（64位元），在正數方面可表達1.7E-308~1.7E+308。double資料型態同樣可接受科學記號的表示方法。

3.1.4　char字元資料型態

- char稱爲字元資料型態，可以用來儲存單一字元。在C語言中，每個字元資料型態佔用記憶體1個位元組（8位元），因此可以表達256種不同的字元符號，表示法則可以使用0~256或-128~127，例如：'A'、'B'、'a'、'5'等等。每個字元符號都有相對應的字元碼，本書將之整理於附錄A的ASCII表，例如字元碼爲『65』就是對應ASCII字元的『A』。

3.1.4　char字元資料型態

- 在C++中，布林資料型態被納入考慮，而多定義了一種bool資料型態來加以對應，它只可以接受『真(true)』或『假(false)』兩種值，我們使用1來表示「真」、用0來表示「假」。當然，您必須使用能夠編譯C++的編譯器才會看得懂bool資料型態，並且在某些編譯器中，您還需要將檔案類型儲存爲C++的檔案類型（例如.cpp）才能使用bool資料型態。

3.1.5 void資料型態

- void資料型態可以用於函式與指標，使用於函式時，代表不傳入引數或該函式不回傳任何值，在前一章中，我們已經示範過了，而使用在指標宣告時，則留待往後的章節中再做介紹。

```
int f1(void)
{
  ..............................
}
```

```
void f2(int p1)
{
  ..............................
}
```

3.1.6　bool資料型態【C++補充】

- 對於C語言來說，它對於只有『真』與『假』的布林函數並未特別定義資料型態來加以對應，但布林型態的變數確時常需要在流程控制中充當判斷式的值或運算結果。雖然C語言未特別定義布林資料型態，但所幸我們可以使用一般的數值變數型態（例如整數變數型態）來當作布林變數。而C語言的編譯器會認定所有非0變數值的變數爲『真』，只有當變數值爲0時，才會被判斷爲『假』。

3.2 基本資料型態的修飾字

- C語言爲了使得基本資料型態更多樣化，以符合各種需求，特別定義了short、long、unsigned、signed等修飾字。透過這些修飾字與基本資料型態的結合就可以宣告更多種的資料型態，但並非每一種的基本資料型態都可以使用全部的修飾字，其中short與long爲同一類，unsigned與signed爲另一類，我們將在本節中詳加介紹這些修飾字的使用方法。

3.2.1　short修飾字

- short修飾字是爲了節省記憶體空間而設計的基本資料型態修飾字，可以用來修飾int資料型態，也就是short int（也可以直接縮寫爲short）。使用short修飾int時，資料只會佔用2個位元組（16個位元），數值範圍從-32768 ~ +32767。

3.2.2 long修飾字

- long修飾字則是爲了擴充更大的數值表示範圍而設計的，可以用來修飾int、double資料型態，使得表達更大的數值或更精確的數字。
- 當使用在int時，請撰寫爲long int（也可以直接縮寫爲long），此時資料會佔用2倍的記憶體空間，也就是8個位元組（64位元），數值範圍從$-2^{63} \sim +2^{63}-1$。

3.2.2 long修飾字

- 使用long來修飾double時，必須撰寫成long double（不可省略double）並且會使用12個位元組來存放資料以便增加精確度。

3.2.3 unsigned修飾字

- 在3.2.1節中，很明顯地可以得知，不論是int、float、double、char都可以包含正值與負值，不過C語言允許使用者透過unsigned修飾字強迫這些資料型態只能爲正值，由於unsigned並不影響資料所佔用的位元數，因此，可表達的數字範圍可以增加1倍。例如：unsigned int可以表達的數字範圍是0~2^{32}-1。

3.2.4　signed修飾字

- signed則是允許出現正數與負數，因此，除了配合char資料型態之外，對於其他資料型態而言並無影響。signed char可表達的範圍則是-128~127。（signed也可以與short、long聯合使用）

3.3 變數與常數

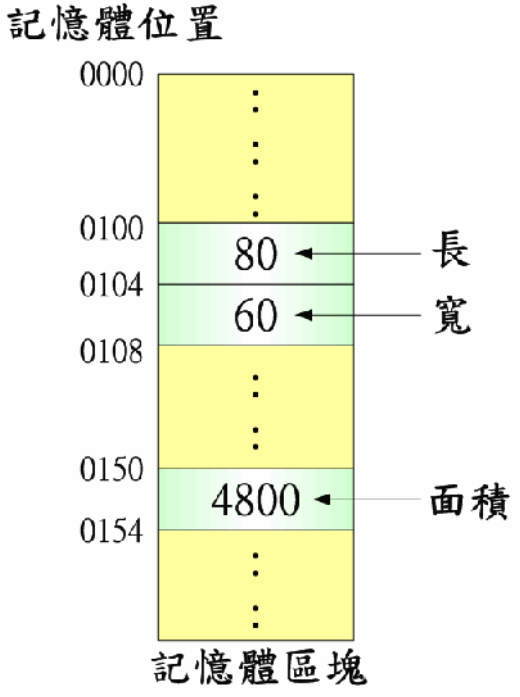
- C程式的資料處理基本單位即爲「變數」與「常數」，想要撰寫一個好的程式，必須先建立一些有關於變數與常數的正確觀念。

3.3.1 變數的意義

- 變數代表在程式執行過程中可能會被改變的某個數值。以現實的環境來說，我們所身處的世界就是一個多變數的世界，例如：全球人口數每天都不同，受出生人口、死亡人口、意外事件發生率、甚至於季節條件等而變化。而其他的條件也都是一個不斷改變的變數。

3.3.1 變數的意義

- 在程式的運作過程中，事實上也是靠眾多變數的變化來完成工作的，例如：一個計算長方形面積的程式，至少就必須包含3個變數：長、寬、面積。
- 如果長與寬都不能改變數值的話，這個程式將只能解決某一個固定的小問題，例如：只能固定計算長為3、寬為2的長方形面積。
- 因此，以上面的範例來說，在記憶體中，就必須儲存長、寬、面積等3個變數如右圖。



變數與記憶體

3.3.2 變數的命名

- 變數在記憶體中佔用了某一小塊記憶體空間，程式可以由記憶體位址取得這些變數內容，但是，對於人來說，記憶體位址是非常難以記憶與了解的，因此，所有的高階語言都提供了以名稱來代替變數在記憶體中的位置，換句話說，我們只要給賦予該變數一個名稱，就可以直接透過名稱來取得變數值，而不用煩惱該變數究竟是在記憶體的是哪一個位置。

3.3.2 變數的命名

- 不同的程式語言對於變數名稱的規定也不相同，在C語言中，變數的名稱必須符合下列規定：
- (1) 變數名稱必須由英文字母、阿拉伯數字、_（底線符號）三種字元構成，但變數的第一個字元不可以爲『阿拉伯數字』。
  - 合法的變數名稱：

```
Length
width
Length2
WIDTH2
_length
```

  - 不合法的變數名稱：

```
2length     不可以用數字做為起始字元
NT$         不可以包含$符號
X,Y,Z       不可包含,符號
int         不可使用關鍵字
_asm        不可使用系統保留字
```

3.3.2 變數的命名

- (2) 變數名稱不可與C語言內定的關鍵字或保留字相同，因爲這些關鍵字或保留字對於編譯器而言，具有特殊意義。C語言的關鍵字與保留字如下：（ANSI C明確定義了如下的關鍵字，至於fortran、asm、__asm等等保留字則是編譯器爲了提供 Fortran及組合語言等語言指令所保留的特殊句元）

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

C語言的關鍵字

3.3.2 變數的命名

C/C++編譯器所規定的保留字

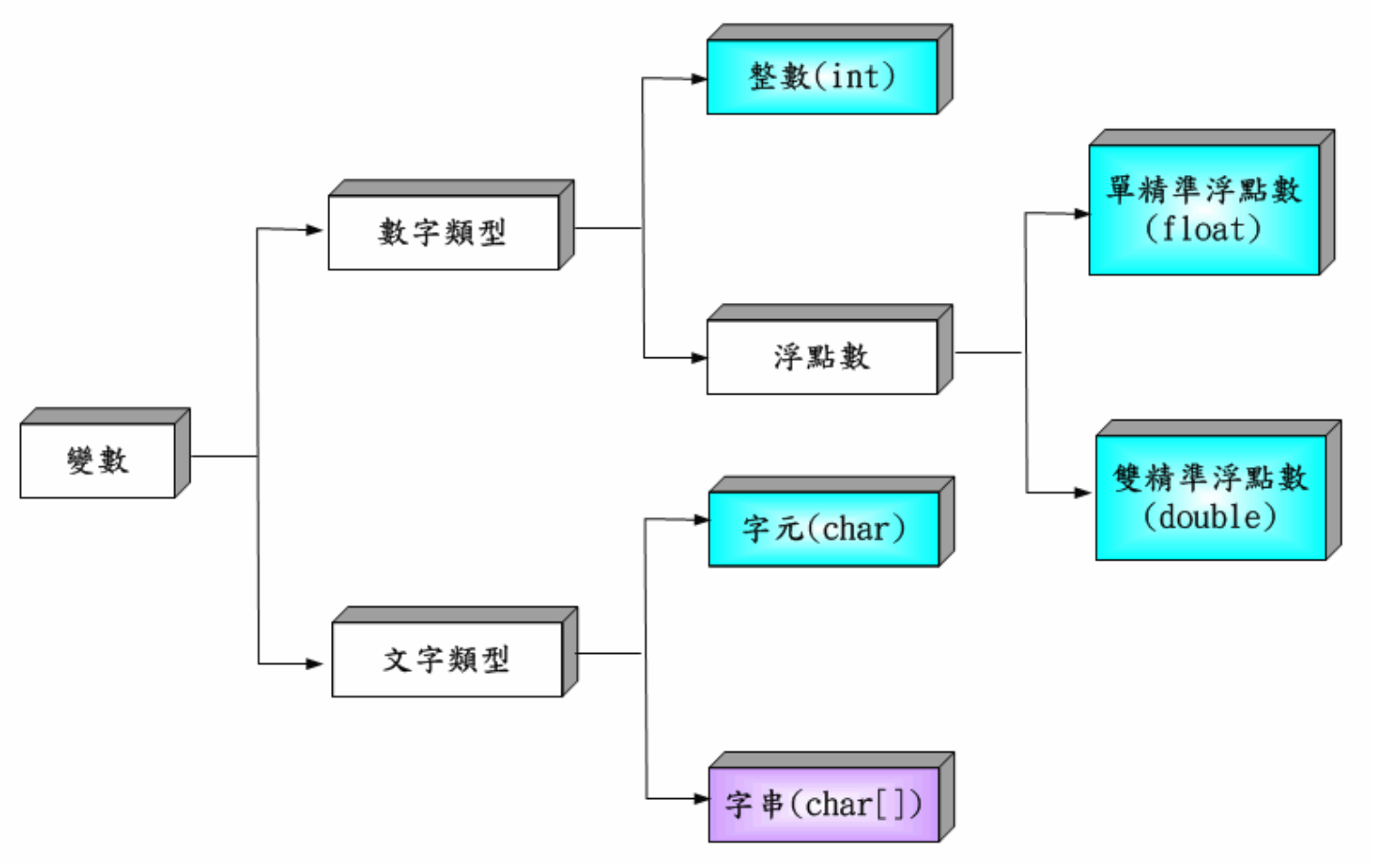
__cdecl	_cdecl	__interrupt
__pascal	_pascal	__near,__far,__huge
__cs,__ds,__es,__seg,__ss	__near,__far,__huge	__near,__far
__export	__import	__loadds
__saveregs	__fastcall	__stdcall

3.3.3 變數的宣告

- C語言規定，任何變數使用前必須先加以宣告。而變數的宣告應該以最適當的資料型態爲主，宣告不適當的資料型態可能無法負擔未來變數的變化，或者浪費記憶體空間。

3.3.3 變數的宣告

C語言的基本資料型態



3.3.3 變數的宣告

- 此外，不適當的變數宣告，可能導致資料處理結果並非預期想要的結果。舉例來說，若X爲int資料型態，Y爲float資料型態，經過以下運算後，會得到不同的結果。這是因爲在做數學運算時，整數變數無法儲存小數的數值，因此X計算結果與預期有所差距。
  - X = 20/5 = 4　　Y = 20/5 = 4
  - X = 20/6 = 3　　Y = 20/6 = 3.333333

3.3.3 變數的宣告

- 變數宣告（不設定初始值）
  - 語法：

[修飾字] 基本資料型態　　變數名稱1, 變數名稱2, 變數名稱3.....；

  - 範例：
    - 宣告x,y爲整數變數，z爲正整數變數，p爲單精準度浮點數變數，q爲長度加大的雙精準度浮點數變數：

```
int x,y;
unsigned int z;
float p;
long double q;
```

3.3.3 變數的宣告

- 變數宣告（設定初始值）
  - 語法：

```
[修飾字] 基本資料型態      變數名稱 = 初始值;
```

  - 範例：
    - 宣告x爲整數變數，並同時指定初始值100；宣告y爲字元變數，並同時指定初始值爲『H』：

```
int   x=100;
char  y='H';
```

3.3.3 變數的宣告

- 變數宣告（設定動態資料初始值）
  - C語言允許將變數的初始值指定爲『動態資料』，所謂動態資料代表使用運算式來設定變數的初始值並且運算式中包含了變數。
  - 範例：
    - 假設在程式之中，已經宣告了length與width兩個變數。則我們可以宣告area變數的動態資料初始值：

```
int length=10,width=20;
int area=length*width;
```

3.3.4 常數的意義

- 常數在日常生活中無所不見，例如：圓周率π、自然指數e、光速...等等，代表的意思是在宇宙間永遠不會變動的數值。
- 但是對於程式而言，常數則是一個「在程式執行過程中不會被改變的某個數值、字元或字串」。換句話說，常數在執行過程中，同樣會佔用某些記憶體，但是該記憶體內容卻不會改變。

3.3.5 常數的種類

- 對於C語言而言，常數一共分爲6種：整數常數、浮點數常數、字元常數、列舉常數、字串常數、變數常數。其細節如下：

  - 整數常數(integer-constant)
    - 整數常數非常簡單，就是一般的正整數、0、負整數，例如：3、15、-52、0、-23。若是八進位則前面必須加上0，例如0571。若是十六進位則前面必須將上0x或0X，例如0x4e、0X5F等等（x的大小寫代表a,b,c,d,e,f的大小寫）。

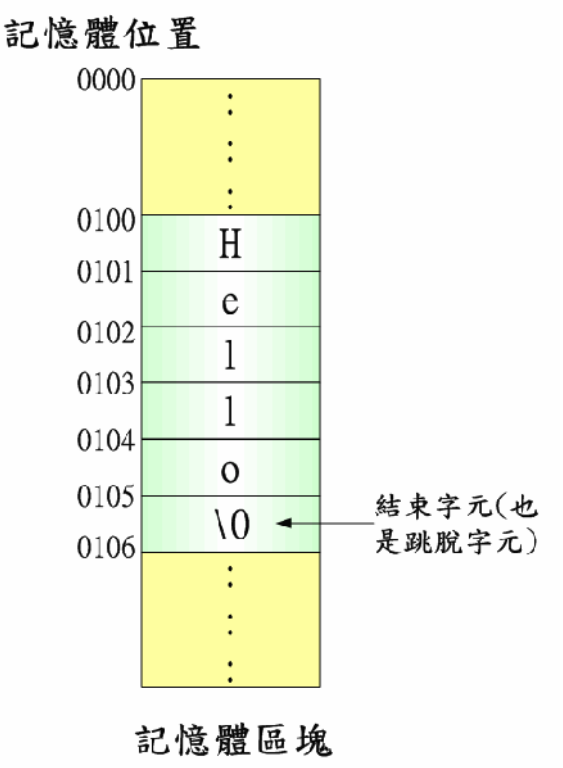
  - 浮點數常數(floating-constant)
    - 浮點數常數則是可包含小數的數值，例如：3.156、-4.567。另一種表示浮點數常數的方式則是科學記號表示法，您可以使用e或E來代表10的次方數，例如：3.000067e+004就是30000.67。

3.3.5 常數的種類

- 字元常數(character-constant)
  - 'a'、'A'、'0'、'#'等都是字元常數，字元常數必須使用單引號『'』括起來。這些字元常數都具有相對應的ASCII碼，例如：'A'的ASCII碼爲65。除了這些可以列印在螢幕上的圖形字元之外，還有另一類則稱爲跳脫序列字元（Escape sequence character）簡稱跳脫字元，這些字元必須使用『\』做爲開頭來加以表示，例如『\n』代表換行的跳脫字元，能夠使得螢幕游標跳到下一行。跳脫字元可以使用8進位或16進位表示其ASCII碼，例如：'\007'代表響鈴（bell），'\0x0A'代表換行（NewLine，也就是\n）。
  - 列舉常數(enumeration-constant)「之後章節介紹」

3.3.5 常數的種類

- 字串常數(string constant)
  - 字串常數(string constant或string literal)又稱爲字元字串(character string)，它代表「由0個以上的字元所組合而成的常數」，例如：“”（空字串）、“Hello”、“您好”等等都是常數，
  - 在C語言中，字串常數必須使用雙引號『"』括起來。。



字串常數的記憶體配置

3.3.5 常數的種類

- 變數常數(qualifier const)
  - 變數常數這個名詞很容易讓人困惑，其實這個名詞看似矛盾，卻只是一個特例的稱呼。C語言的變數必須經由宣告才能使用，而在宣告時，我們可以指定它爲一個常數，也就是該變數的變數值一開始就被指定，並且不允許在程式中被更改，然後就能夠像變數一樣，以名稱在程式中加以運用（但不可以更動其值）。
  - 另一種比較容易接受的說法是，變數常數如同變數一樣，也可以使用某一個符號來當做名稱，不同的是，變數名稱所代表的變數值，可以在程式執行中改變，而變數常數所代表的常數則不允許改變。
  - 變數常數必須經由const關鍵字來加以宣告，並且指定起始值（也就是固定值）。自此之外，不論在程式的哪個地方，都不允許重新指定變數常數的值。

3.3.5 常數的種類

- 語法：

```
const [修飾字 基本資料型態]        符號常數名稱=常數值；
```

- 說明：
  - 變數常數同樣可以依照資料特性區分爲多種資料型態，但若省略宣告資料型態時，則內定爲int資料型態，例如：『const int a=100;』與『const a=100;』具有相同效果。
- 範例：
  - 宣告pi爲符號常數，資料型態爲float：

```
const float pi=3.1416;
```

3.3.5 常數的種類

- 範例3-1：ch3_01.c。

```
1  /*****************************
2      檔名:ch3_01.c
3      功能:求圓面積
4   *****************************/
5
6  #include <stdio.h>
7  #include <stdlib.h>
8
9  void main(void)
10 {
11  int r=3;
12  const float pi=3.1416;
13  float area=pi*r*r;
14  printf("半徑爲%d",r);
15  printf("的圓面積爲%f\n",area);
16  system("pause");
17 }
```

3.3.5 常數的種類

- 執行結果：

```
半徑爲3的圓面積爲28.274399
請按任意鍵繼續 . . .
```

- 範例說明：
  - (1) 範例中，一共宣告了r與area等2個變數，其中r爲整數變數，初始值爲3。area爲單精準度浮點數變數，使用動態資料做爲初始值，也就是area的值隨著r的值而改變。
  - (2) pi爲單精準度浮點數變數常數，使用const宣告爲3.1416。
  - (3) printf是C語言的輸出函式，需要引用stdio.h標頭檔。第14行的%d代表印出整數資料，其資料來源爲後面的變數r。第15行的%f代表印出浮點數資料，其資料來源爲後面的變數area。至於『\n』則是換行符號，它也是一種屬於跳脫字元的字元常數。而我們將在下一章詳細介紹printf。
  - (4) 『"半徑爲"』及『"的圓面積爲"』都是字串常數。

3.4 運算子及運算元

- 程式的目的是解決問題，而手段則是依靠各個變數值的改變，想要變更變數值就必須透過運算式加以完成。運算式(Expression)也是敘述的一種，它是由運算元(Operand)與運算子(Operator)共同組成，就像是常見的數學公式就是最基本的運算式，例如：area=r*r*3.14。其中的『r』、『3.14』就是運算元，而『=』、『*』就是運算子。

- C語言提供了許多的運算子，並且允許運算元爲一個或多個的『常數』、『變數』、『函式呼叫敘述』或甚至於是『其他運算式』的組合，在本節中，我們將針對這些C語言所提供的運算子詳加說明。

3.4.1 『=』設定運算子

- 『=』運算子可以說是最常見的運算符號，『=』的作用是將符號右邊的運算式計算後的值指定給左邊的變數。因此又稱爲設定運算子或指定運算子。
  - 格式：

```
變數 = 運算式;
```

  - 範例：

```
a = 10;
b = 'w';
c = p+q;
```

3.4.1 『=』設定運算子

- 說明：
  - 『=』運算子的左邊只能有也必須有唯一的一個變數，不能是數值、函次、其他的複合運算式，例如下列都是錯誤的使用範例：

```
10 = x+40;    /*左邊不可以是數值*/
f(h) = 15;    /*左邊不可以是函式*/
x+y =z;                    /*左邊不可以是複合運算式*/
```

3.4.1 『=』設定運算子

- 【觀念】：
  - 在C語言中，『=』的意義爲指定或設定，與數學上的相等有些許不同，對於C語言而言，相等代表一種比較，因此必須使用比較運算子的『==』符號。
- 【解析】：
  - 雖然『=』運算子爲運算式中最常見的運算子，但並非所有的運算式都必須包含『=』運算子，
  - 例如：a++也是一個運算式。

3.4.2 算數運算子

- 算數運算子可用來做數學的運算，C語言提供的算數運算子共有5種：『+』、『-』、『*』、『/』、『%』，如下表所列：

算數運算子	使用範例	說明
+	a + b	a加b
-	a - b	a減b
*	a * b	a乘b
/	a / b	a除以b
%	a % b	取a除以b的餘數

3.4.2 算數運算子

```
1  /*************************
2      檔名:ch3_02.c
3      功能:算數運算子
4   *************************/
5
6  #include <stdio.h>
7  #include <stdlib.h>
8
9  void main(void)
10 {
11  int x,y;
12  x = 20;
13  y = 7;
14  printf("當x=%d",x);
15  printf(",y=%d時\n",y);
16  printf("x + y = %d\n",x+y);
17  printf("x - y = %d\n",x-y);
18  printf("x * y = %d\n",x*y);
19  printf("x / y = %d\n",x/y);   /* y不可爲0 */
20  printf("x %% y = %d\n",x%y);   /* y不可爲0 */
21  system("pause");
22 }
```

3.4.2 算數運算子

- 執行結果：

```
當x=20,y=7時
x + y = 27
x - y = 13
x * y = 140
x / y = 2
x % y = 6
請按任意鍵繼續 . . .
```

3.4.2 算數運算子

- 範例說明：
  - (1) 第16行的%d代表要印出整數資料，其資料來源爲x+y的結果。其餘第17~20行亦同理。
  - (2) 第20行爲何需要用『%%』來代表印出「%」符號呢？這是因爲%在printf內有特殊意義（用來帶領%d、%f等等），因此必須重複兩次。
  - (3) 由於x、y都是整數資料型態，因此x/y會被轉換成整數資料型態（只能記錄整數部分）。（印出整數2和%d並無關係，即使改用%f也無法印出2.857…的浮點數資料，因爲x、y都是整數資料型態）

$$\frac{x}{y} = \frac{20}{7} = 2.857\cdots \xrightarrow{\text{只取整數部分}} 2$$

  - (4) 不論是做除法或取餘數時，分母都不可爲0。而20%7的結果是餘數6。
  - (5) x*y不可以簡寫爲xy，這一點與數學不太一樣。

3.4.2 算數運算子

- 範例3-3：ch3_03.c。

```
1  /*****************************
2      檔名:ch3_03.c
3      功能:複雜的算數運算
4   *****************************/
5
6  #include <stdio.h>
7  #include <stdlib.h>
8
9  void main(void)
10 {
11  float   answer;
12  float   a=2.1,b=3.5,c=4;
13  printf("a=%f",a);
14  printf("b=%f",b);
15  printf("c=%f\n",c);
16  answer = b*b-4*a*c;
17  printf("b^2-4ac=%f\n",answer);
18  system("pause");
19 }
```

3.4.2 算數運算子

- 執行結果：

```
a=2.100000      b=3.500000      c=4.000000
b^2-4ac=-21.349998
請按任意鍵繼續 . . .
```

- 範例說明：
  - 第16行的運算式『answer=b*b-4*a*c』當中，會先做『*』乘法，再做『-』減法，最後才是『=』指定。這是因爲『*』號的運算子優先權比『-』號還高，並且『-』號運算子優先權比『=』號還高的緣故，詳細的各種運算子優先權，請見3.4.7節，
  - 如果您記不住所有的優先權，又沒有資料或書籍可以查閱時，可以將想要先做的部分運算式，使用『()』小括號括起來即可。

3.4.3 比較運算子

- C語言提供的比較運算子有『==』、『!=』、『>』、『<』、『>=』、『<=』等六種。其運算結果爲布林值，也就是真（True）或假（False），真會以整數1來表示，假會以整數0來表示。所以比較運算子常用在條件判斷之用，其餘比較運算子說明如下表所列：

比較運算子	意義	使用範例	說明
= =	等於	x==y	比較x是否等於y
!=	不等於	x!=y	比較x是否不等於y
>	大於	x>y	比較x是否大於y
<	小於	x<y	比較x是否小於y
>=	大於等於	x>=y	比較x是否大於等於y
<=	小於等於	x<=y	比較x是否小於等於y

3.4.3 比較運算子

- 範例3-4：ch3_04.c

```
1  /*****************************
2      檔名:ch3_04.c
3      功能:比較運算子
4   *****************************/
5  #include <stdio.h>
6  #include <stdlib.h>
7
8  void main(void)
9  {
10  int x=10,y=20;
11
12  printf(" x=%d",x);
13  printf(" y=%d\n",y);
14  printf("1代表真,0代表假\n");
15  printf("x==y ==> %d\n",(x==y));
16  printf("x!=y ==> %d\n",(x!=y));
17  printf("x>y  ==> %d\n",(x>y));
18  printf("x<y  ==> %d\n",(x<y));
19  printf("x>=y ==> %d\n",(x>=y));
20  printf("x<=y ==> %d\n",(x<=y));
21  system("pause");
22 }
```

3.4.3 比較運算子

- 執行結果：

```
x=10  y=20
1代表真,0代表假
x==y  ==>  0
x!=y  ==>  1
x>y   ==>  0
x<y   ==>  1
x>=y  ==>  0
x<=y  ==>  1
請按任意鍵繼續 . . .
```

- 範例說明：
  - 比較運算子的執行結果會回傳一個布林值。以1代表真、0代表假。所以比較運算子常做爲條件判斷之用。

3.4.4　邏輯運算子

- 邏輯運算子可以將布林資料1（True）或0（False）做某些運算，若將邏輯運算子與其他比較運算子搭配使用，運算結果將仍然是邏輯資料（布林值），因此常被用來當做條件判斷之用。在C語言中，邏輯運算子共有NOT、AND、OR等3種，其符號分別爲『!』、『&&』、『||』，我們以真值表方式來加以介紹。

3.4.4 邏輯運算子

- (1) !（NOT）
  - 『!』邏輯運算子會將緊接著的運算元的值給反相，若輸入的值爲1（True），輸出的值將爲0（False）。反之，輸入的值爲0（False），輸出的值將爲1（True）。

X	!X
0	1
1	0

真值表：!X

3.4.4 邏輯運算子

- (2) &&（AND）
  - 當前後兩個運算元都是1（True），輸出才會是1（True），其餘的各種情況都是輸出0（False）。

X	Y	X && Y
0	0	0
0	1	0
1	0	0
1	1	1

真值表：X && Y

3.4.4 邏輯運算子

- (3) ||（OR）
  - 當前後兩個運算元中只要有一個是1（True），輸出就會是1（True），只有當兩個運算元都是0（False）時，輸出才會是0（False）。

X	Y	X || Y
0	0	0
0	1	1
1	0	1
1	1	1

真值表：X || Y

- 【註】：
  事實上C語言規定，所有非0的值，都被視爲真（不只有1才會視爲True）。只有0才會被視爲False。但一般我們想要指定某個值爲真時，通常會將它指定爲1，但其實指定爲其他非0值也可以。

3.4.4 邏輯運算子

- 範例3-5：ch3_05.c

```
1  /***************************
2      檔名:ch3_05.c
3      功能:邏輯運算子
4   ***************************/
5  #include <stdio.h>
6  #include <stdlib.h>
7   void main(void)
8  {
9    int x=1,y=0;        /*  x爲真,y爲假  */
10   printf("1代表真,0代表假\n");
11   printf(" x=%d",x);
12   printf(" y=%d\n",y);
13   printf("--------------------\n");
14   printf("not x    ==> %d\n",!x);
15   printf("x and y  ==> %d\n",(x && y));
16   printf("x or y   ==> %d\n",(x || y));
17   printf("x xor y  ==> %d\n",((x && !y)||(!x
18  && y)));
19   printf("x nand y ==> %d\n",!(x && y));
20   printf("x nor y  ==> %d\n",!(x ||y));
21   system("pause");
   }
```

3.4.4 邏輯運算子

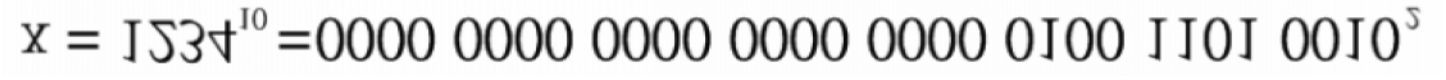
- 執行結果：

```
1代表真,0代表假
 x=1  y=0
--------------------
not  x      ==> 0
x  and  y   ==> 0
x  or  y    ==> 1
x  xor  y   ==> 1
x  nand  y  ==> 1
x  nor  y   ==> 0
請按任意鍵繼續 . . .
```

3.4.5 位元邏輯運算子

- C語言爲何被視爲最接近低階語言的高階語言呢？主要是因爲C語言提供了對位元直接運算的能力以及可嵌入組合語言等兩大特色。

  - 在位元運算方面，C語言提供六種位元邏輯運算子，透過這些運算子，我們就可以只針對某些位元（bit）做運算處理，不必像前面所介紹的運算子必須針對變數整體做運算。例如：整數變數x在記憶體佔用32個位元（4個位元組）長度，若x爲1234，則實際在記憶體中的內容如下：



3.4.5 位元邏輯運算子

- 我們只要透過位元邏輯運算子，就可以針對某一些位元單獨做運算，進行更低階的資料處理。
- C語言提供的位元邏輯運算子如下表。

位元邏輯運算子	意義	範例	說明
~	反相	~x	將x的所有位元做NOT運算
&	且	x & y	將x與y相對應的位元做AND運算
|	或	x | y	將x與y相對應的位元做OR運算
^	互斥	x^y	將x與y相對應的位元做XOR（互斥）運算
>>	右移	x >> p	將x的內容往右移動p個bit（左邊補0）
<<	左移	x << p	將x的內容往左移動p個bit（右邊補0）

3.4.5 位元邏輯運算子

- 【註】：
  - XOR（互斥）的真值表如下，當前後兩個運算元的值不相同時，才會輸出1（True）。

真值表：X ^ Y

X	Y	X ^ Y
0	0	0
0	1	1
1	0	1
1	1	0

3.4.5 位元邏輯運算子

- 範例3-6：ch3_06.c

```
1  /*****************************
2      檔名:ch3_06.c
3      功能:位元邏輯運算子
4   *****************************/
5  #include <stdio.h>
6  #include <stdlib.h>
7
8  void main(void)
9  {
10  unsigned short int x=100,y=50,p=3,xx;
11  xx=~x;
12  printf("p=3\n");
13  printf("x=01100100\n");
14  printf("y=00110010\n");
15  printf("--------------------\n");
16  printf("not x     ==> %d\n",xx);
17  printf("x and y   ==> %d\n",(x & y));
18  printf("x or y    ==> %d\n",(x | y));
19  printf("x xor y   ==> %d\n",(x ^ y));
20  printf("x >> p    ==> %d\n",(x >> p));
21  printf("x << p    ==> %d\n",(x << p));
22  system("pause");
23 }
```

3.4.5 位元邏輯運算子

- 執行結果：

```
p=3
x=01100100
y=00110010
----------------------
not x     ==> 65435
x and y   ==> 32
x or y    ==> 118
x xor y   ==> 86
x >> p    ==> 12
x << p    ==> 800
請按任意鍵繼續 . . .
```

3.4.6 複合式指定運算子

- C語言除了簡單的指定運算子『=』之外，還提供另一種同時具有運算及指定功能的運算子，此類運算子有很多，如下表所列。

複合式指定運算子	使用方法	功能等同於
+=	i += j;	i = i + j;
-=	i -= j;	i = i - j;
*=	i *= j;	i = i * j;
.=	i /= j;	i = i / j;
%=	i %= j;	i = i % j;
|=	i |= j;	i = i | j;
&=	i &= j;	i = i & j;
^=	i ^= j;	i = i ^ j;
>>=	i >>= j;	i = i >> j;
<<=	i <<= j;	i = i << j;

3.4.6 複合式指定運算子

- 【解析】：
  既然使用i+=j與i=i+j功能相同，那C語言爲何要發展複合式運算子呢？這是因爲機器特性的緣故。
- 組合語言中，常常會出現ADD AX,BX這類指令，所代表的涵義是AX=AX+BX。
- 爲了使得編譯器能夠將C語言程式轉換爲最有效率的機器碼，因此，i+=j這類複合式指定運算子被發明出來對應硬體結構特性，不過，現在的編譯器最佳化技術已經很成熟了，因此，使用i+=j與i=i+j在效能上常常是相同的，究竟要使用哪一種表示法，則視使用者的習慣而定。

3.4.7 遞增與遞減運算子

- C語言提供兩個特別的運算符號：遞增符號「++」、遞減符號「--」。遞增符號會將運算元的內容加1，遞減符號的內容會將運算元減1。

複合式指定運算子	使用方法	功能等同於
++	a++; ++a;	a = a + 1;
--	a--; --a;	a = a - 1;

3.4.7 遞增與遞減運算子

- 『++』和『--』可以放在運算元的後面，也可以放在運算元的前面。放在運算元後面時，代表要做前置運算，如：i ++。放在運算元前面時，代表要做後置運算，例如：++i。兩種運算的意義不同，分述如下：
  - 前置運算（例如：i++）：
    - 在使用這個運算元之前，先進行加一或減一的動作。
  - 後置運算（例如：++i）：
    - 在使用這個運算元之後，再進行加一或減一的動作。

- 範例3-7：ch3_07.c

```
1  /*************************
2      檔名:ch3_07.c
3      功能:遞增/遞減運算子
4   *************************/
5  #include <stdio.h>
6  #include <stdlib.h>
7
8  void main(void)
9  {
10  int i=5,j=10,a,b;
11  int x=5,y=10,c,d;
12
13  a = 1+ i++;
14  b = 1+ j--;
15  c=  1+ ++x;
16  d=  1+ --y;
17
18  printf("i = %d\n",i);
19  printf("j = %d\n",j);
20  printf("x = %d\n",x);
21  printf("y = %d\n",y);
22  printf("a = %d\n",a);
23  printf("b = %d\n",b);
24  printf("c = %d\n",c);
25  printf("d = %d\n",d);
26  system("pause");
27 }
```

3.4.7 遞增與遞減運算子

- 執行結果：

```
i = 6
j = 9
x = 6
y = 9
a = 6
b = 11
c = 7
d = 10
請按任意鍵繼續 . . .
```

3.4.7　遞增與遞減運算子

- 範例說明：

  - (1) 雖然i,j,x,y的執行結果都各自遞增或遞減1了，但由於將『++』與『--』放在不同位置，所以a,b,c,d的執行結果並不相同。第14行是先將執行a=1+i然後再執行i=i+1，所以a=6、i=6。而第16行程式，則是先執行x=x+1再執行c=c+x，所以x=6、c=7。第15、17行也是相同的道理。

  - (2) 另外，讀者會發現c=1+ ++x若刪除空白並且將1也用變數取代時，讀者可能會對於運算結果產生疑惑，例如：c=p+++q，到底是c=p+(++q)還是c=(p++)+q呢？這其實和運算子的優先權有關，『++』的運算子優先權比『+』優先權還高，因此，c=p+++q相當於c=(p++)+q。

3.4.8 其他運算子

- 除了上述運算子之外，C語言還提供另外一些運算子，分述如下：
- sizeof()運算子
  - sizeof( )運算子可以用來計算任何資料型態所佔的記憶體大小(以位元組爲單位)。
  - 格式：

```
sizeof 運算式;                    或
sizeof(運算式);                   或
sizeof(資料型態);                 或
sizeof(常數);                     或
sizeof(變數);
```

3.4.8 其他運算子

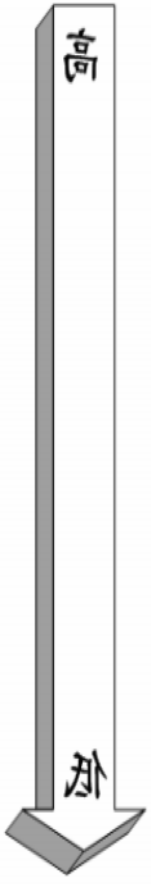
- 條件運算子
  - C語言提供了條件運算子『?:』，可以用來當做簡單的if-eles判別指令，如下列範例，我們會在條件判斷一節中詳加說明。

條件運算子	使用範例	說明
?	a ? b : c	若a爲真，則取b，否則取c

3.4.8 其他運算子

運算子	說明
()	應用廣泛，不同位置有不同涵義，例如：函式宣告的引數串列、提高部分運算式的優先權、控制流程的判斷式等。
{}	區塊的起始與結束。
[]	指定陣列維度及存取陣列元素。
,	分隔運算子，例如：變數宣告的間隔。
.	直接存取自訂結構的成員
->	存取指標結構體運算子以及間接存取自訂結構的成員
*	在數學運算式中扮演乘法，在定義變數或函式時扮演指標變數或函式指標。
&	在位元運算式中扮演AND功能，應用於變數時扮演取得變數位址或將變數設爲參考。

3.4.9 運算子結合性及優先權

優先權	運算子	結合性
	() [] -> . ++（後置） --（後置）	由左而右
	! ~ ++（前置） --（前置） + - *(type) sizeof	由右而左
	* / %	由左而右
	+ -	由左而右
	<< >>	由左而右
	< <= > >=	由左而右
	== !=	由左而右
	&	由左而右
	^	由左而右
	|	由左而右
	&&	由左而右
	||	由左而右
	?:	由右而左
	= += -= *= /= %= &= ^= |= <<= >>=	由右而左
	,	由左而右

運算子優先權順序及結合性

3.5 資料型態的轉換

- 在C語言中，在同一個運算式允許出現不同資料型態的運算，但是可能會發生資料型態轉換的問題，本節將就資料型態轉換之觀念與方法加以詳加討論。
- 3.5.1　運算式的型態轉換
  - 若某一個運算式中包含不同的資料型態，則編譯器會自動對資料做最適當的轉換。我們以範例來加以說明：

3.5.1 運算式的型態轉換

- 範例3-9：ch3_09.c

```
1  /*****************************
2       檔名:ch3_09.c
3       功能:自動型態轉換
4   *****************************/
5
6  #include <stdio.h>
7  #include <stdlib.h>
8
9  void main(void)
10 {
11  int a=8,b;
12  double c=3.1416;
13  b=a+c;
14
15  printf("b = %d\n",b);
16  system("pause");
17 }
```

3.5.1 運算式的型態轉換

```
b = 11
請按任意鍵繼續 . . .
```

- 範例說明：
  - 本範例在編譯過程中，可能會出現warning警告訊息，但仍可編譯與執行。當編譯器處理b=a+c時，會先將a的記憶體配置提升爲8個位元組（即double所佔記憶體大小），經由計算a+c完畢後，再依據b的資料型態來指定（=）數值。所以b=11。

3.5.1 運算式的型態轉換

- 事實上，編譯器在處理不同資料型態時，有一定的步驟與規則必須遵守，如下所述：
  - Step1：進行實際運算前，將所有的基本資料型態按照下表，初步轉換爲int或double型態：

轉換前	轉換後
char、unsigned char、(unsigned) short int、int	int
float、double	double

3.5.1 運算式的型態轉換

- Step2：經由初步轉換之後，運算式中只剩下int、double、long int、unsigned long等四種資料型態，然後再依照如下的優先權高低，選擇優先權最高的資料型態加以再次轉換：

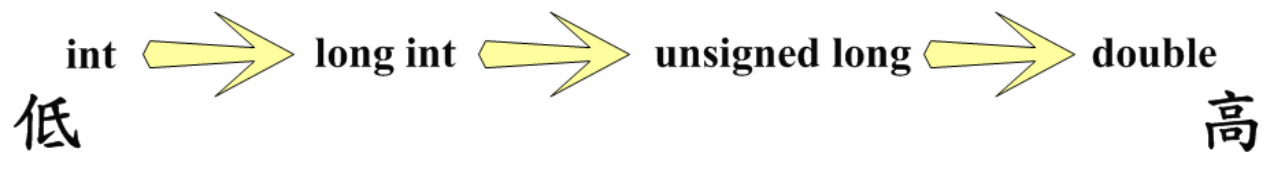
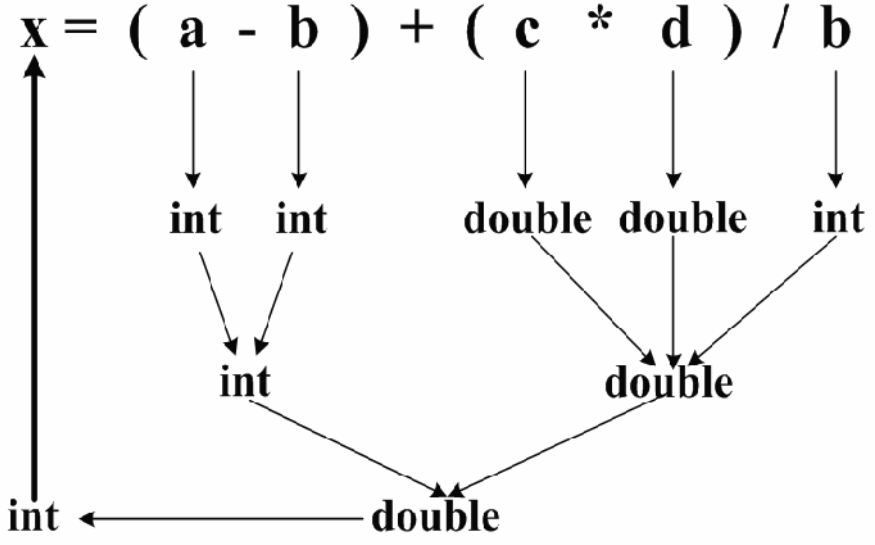


圖3-5 資料型態的轉型優先權

- Step3：轉換完畢後就進行實際運算，並且將運算結果轉換爲等號左邊變數所屬的資料型態，最後將運算結果回存到左邊的變數中。

3.5.1 運算式的型態轉換

- 範例：假設我們共有5個變數，其資料類型如下：
  - int a;
    char b;
    float c;
    double d;
    int x;
  - 當出現運算式爲x=(a-b)+(c*d)/b時，會以下圖步驟來轉換資料型態



3.5.2 強制型態轉換

- 自動資料型態轉換非常複雜，使得我們常常難以控制，不過C語言也提供了另外一個方法來解決資料型態轉換的問題，也就是允許使用者強制指定要將變數轉換爲哪一種資料型態，格式如下。
  - 格式：

```
(強制轉換型態) 運算式或變數;
```

3.5.2　強制型態轉換

- 範例：若x=15，y=4皆爲整數變數，
  - z爲浮點數變數，則運算式『z = x / y』，會得到『z=3』（因爲x與y都是整數，而15/4=3.75，取整數爲3，因此z=3）。
  - 但若是以強制型態轉換來重寫運算式『z = (float)x / (float)y』，則運算前會先將x、y都轉換爲浮點數，因此15.0/4.0=3.75，因此z=3.75。

3.5.3 使用函式轉換資料型態

- 除了上述轉換資料型態的方法外，在C函式庫<stdlib.h>中也提供了多個函式可以用來轉換資料型態（大多是數字與字串的轉換），整理如下：
- 字串轉數值（<stdlib.h>）

函式	函式原型	輸入值	回傳值	說明
atof	double atof(const char *string);	要被轉型的字串	double資料型態	將字串轉成double資料型態
atoi	int atoi(const char *string);	要被轉型的字串	int資料型態	將字串轉成int資料型態
atol	long atol(const char *string);	要被轉型的字串	long資料型態	將字串轉成long資料型態
其他	strtod()、strtol()	strtod(s, (char**)NULL)同等於atof(const char *string)、(int)strtol(s, (char**)NULL, 10)同等於int atoi(const char *string)，詳見C++函式庫精華錄一書		

```
/*********************************
    檔名:ch3_10.c
    功能:atoi、atof、atol型態轉換
 *********************************/
#include <stdio.h>
#include <stdlib.h>
void main(void)
{
   char *string; double d; int i; long l;

   string = "1234567";
   l = atol( string );
   printf("%s \t轉換爲long=>%d\n",string,l);

   string = "12345jhchen";
   l = atol( string );
   printf("%s \t轉換爲long=>%d\n",string,l);

   string = "1234567persons";
   i = atoi( string );
   printf("%s \t轉換爲int=>%d\n",string,i);

   string = "-1234.56E-7";
   d = atof( string );
   printf("%s \t轉換爲double=>%.9f\n",string,d);
   system("pause");
}
```

3.5.3 使用函式轉換資料型態

- 執行結果：

```
1234567          轉換爲long=>1234567
12345jhchen      轉換爲long=>12345
1234567persons   轉換爲int=>1234567
-1234.56E-7      轉換爲double=>-0.000123456
請按任意鍵繼續 . . .
```

- 範例說明：
  - 第20行的string字串變數爲『12345jhchen』，因此轉換時只能轉換數字部分，所以轉換結果爲『12345』。（本範例中使用了非常多有關於printf的符號，例如%s、%.9f、\t，別擔心，我們馬上將於下一章中說明這些符號）

3.5.3 使用函式轉換資料型態

- 數值轉字串（<stdlib.h>）

函式	函式原型	輸入值	回傳值	說明
_itoa	char *_itoa(int value, char *string, int radix);	value爲欲轉換的數值。 string 存放轉出來的結果。 radix 爲數值的基底，範圍必須是在2~36。	指標字串	將整數型態轉爲字串
_fcvt	char *_fcvt(double value, int count, int *dec, int *sign);	value爲要被轉換的數值。 count爲小數點後的位數。 dec爲小數點所在的位置。 sign爲正負號，0代表正，非0整數代表負。	指標字串	將浮點數轉爲字串

3.6 本章回顧

- 在本章中，我們學習到C語言的變數與常數宣告以及運算式、運算子、運算元的表示方法。並且瞭解了C語言對於不同資料型態之間的轉換規則與方法。本章重點整理如下：
  - (1) C語言的基本資料型態有：int、float、double、char、void等五種。
  - (2) short、long、unsigned、signed可用來修飾資料型態的長度及資料的表示方式（是否含正負表示法）。
  - (3) C語言的任何變數在使用前必須先經由宣告。

3.6 本章回顧

- (4) 變數的命名原則：
  - 變數名稱必須由英文字母、阿拉伯數字、_（底線符號）三種字元構成，但變數的第一個字元不可以爲『阿拉伯數字』。
  - 變數名稱不可以是C語言的關鍵字或保留字。
  - 變數名稱大小寫不同。
- (5) 變數常數使用const宣告，在程式執行過程中，數值不可更改。
- (6) 運算式由運算子與運算元組成，可以用來改變各個變數的數值。
- (7) C語言的各種運算子，其中最特殊的是複合式運算子與遞增／遞減運算子。

3.6 本章回顧

- (8) 同一個運算式中，若出現不同資料型態的變數，則編譯器會自動將資料做最適當的型態轉換。
- (9) 在C語言中，強制型態轉換如下：

```
(強制轉換型態)  運算式或變數;
```

- (10) 我們可以透過<stdlib.h>函式庫所提供的多個函式來執行數值與字串的轉換。

本章習題