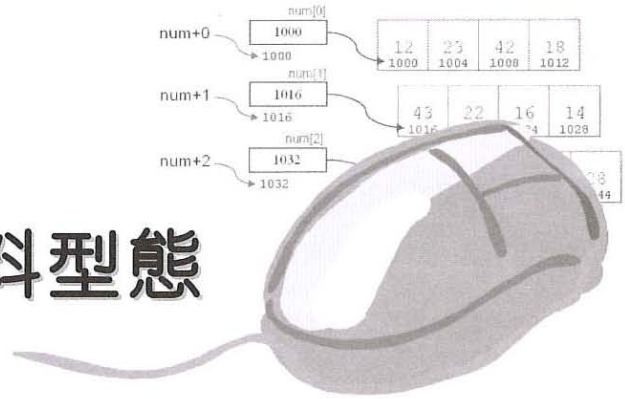
chapter

03

基本資料型態



在 C 語言中，變數是利用宣告的方式，將記憶體中的某個區塊保留下來以供程式使用。本章將就變數及各種資料型態做一個基礎的解說，同時也介紹溢位的發生與型態轉換等概念，學完本章，您將會對 C 語言所提供的資料型態有更深一層的認識。

本章學習目標

- 認識常數與變數的不同
- 學習 C 語言所提供的各種資料型態
- 了解溢位的發生
- 學習認識資料型態之間的轉換

3.1 變數與常數

不同類型的資料需要不同型態的變數來儲存。例如班級的人數一定是整數，此時便可利用整數型態的變數來儲存班級的人數；如果手機的重量為 62.4 克，因 62.4 這個數字帶有小數，因此利用浮點數型態的變數來儲存它較為適合。

C 語言提供了多種資料型態的變數，以因應各種資料的儲存所需。在介紹這些資料型態之前，我們先來看一個簡單的實例：

```
01  /* prog3_1,變數的使用 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     int num1=12400;      /* 宣告 num1 為整數變數，並設值為 12400 */
07     double num2=5.234;   /* 宣告 num2 為倍精度浮點數變數，並設值為 5.234 */
08
09     printf("%d is an integer\n",num1);   /* 呼叫 printf()函數 */
10     printf("%f is a double\n",num2);     /* 呼叫 printf()函數 */
11
12     system("pause");
13     return 0;
14  }
```

```
/* prog3_1 OUTPUT---

12400 is an integer
5.234000 is a double
----------------------*/
```

程式解說

在 prog3_1 中，6~7 行宣告了整數變數 num1 與倍精度浮點數變數 num2，並分別將整數常數 12400 與倍精度浮點數常數 5.234 設值給這兩個變數，9~10 行則是利用 printf() 函數將它們顯示在螢幕上。

當我們宣告一個變數（variable）時，編譯程式會在記憶體內配置一塊足以容納此變數大小的記憶體空間給它。不管變數的值如何改變，同一種型態的變數永遠佔用相同的記憶空間。常數（constant）則不同於變數，它的值是固定的，如整數常數 12400、倍精度浮點數常數 5.234 等。

以程式 prog3_1 為例，第 6 行宣告了一個整數變數 num1，並將整數常數 12400 設定給它，此時編譯器便會配置 4 個位元組（bytes）的記憶空間給變數 num1，並將 12400 寫入此記憶空間內，如下圖所示：

```
int num1=12400;
```

宣告整數變數 num1，並設值為 12400

圖 3.1.1
宣告整數變數 num1，此時編譯器會配置 4 bytes 的記憶空間給它

相同的，第 7 行宣告了一個倍精度浮點數變數 num2，並設定初值為 5.234。因倍精度浮點數佔了 8 個位元組，因此編譯器便會配置 8 個位元組的記憶空間給變數 num2，並將常數 5.234 寫入此記憶空間內，如下圖所示：

```
double num2=5.234;
```

宣告 double 型態的變數 num2，並設值為 5.234

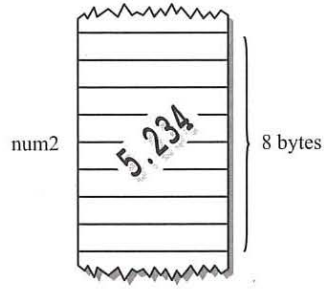


圖 3.1.2
宣告 double 型態的變數 num2，此時編譯器會配置 8 bytes 的記憶空間給它

prog3_1 說明了如何宣告變數，並設值給它。值得一提的是，最好能為變數取個有意義的名稱，如此可方便程式的撰寫，並減少開發時間，提升維護效率。 ❖

3.2 基本資料型態

在 C 語言裡，變數的型態可分為整數與浮點數兩大類型。下表中列出了各種基本資料型態所佔的記憶空間及範圍，您可以在使用時選擇適合的資料型態：

表 3.2.1 C 語言所提供的基本資料型態

資料型態		型態說明	位元組	表示範圍
整數類型	long int	長整數	4	-2147483648 到 2147483647
	int	整數	4	-2147483648 到 2147483647
	short int	短整數	2	-32768 到 32767
	char	字元	1	0 到 255 (256 個字元)
浮點數類型	float	浮點數	4	1.2e-38 到 3.4e38
	double	倍精度浮點數	8	2.2e-308 到 1.8e308

在不同的編譯器裡，整數類型的變數所佔的位元組可能會有些許的不同。例如早期的編譯器如 Turbo C 等，int 佔了 2 個位元組，long int 的長度則為 4 個位元組。而在 Dev C++中，int 與 long int 則是同樣佔了 4 個位元組。

3.2.1 整數型態 int

當資料不帶有小數時，即可用整數變數來存放它。於 Dev C++ 中，整數資料型態佔了 4 個位元組。若要宣告變數 num 為整數，並設值為 15，可利用下面的語法：

```
int num=15;          /* 宣告 num 為整數，並設值為 15 */
```

於 Dev C++裡，長整數也是佔了 4 個位元組，因此把變數宣告成 int 或者是 long int，使用起來並無差異。但在其它編譯器裡，int 可能只佔了 2 個位元組，此時如果需要用到比較大範圍的整數時，則必須把變數宣告成 long int，如下面的語法：

```
long int num=124000L;  /* 宣告 num 為長整數，並設值為 124000L */
```

其中大寫的 L 是代表此一常數是長整數常數。於上面的宣告中，int 是可以省略的，因此我們可以把這個宣告改寫成如下的敘述：

```
long num=124000L;      /* 宣告 num 為長整數，並設值為 124000L */
```

若是資料值很小，範圍在-32768 到 32767 之間時，便可利用短整數（short int）來存放，以節省記憶空間，因為它只佔了 2 個位元組。舉例來說，想宣告一個短整數變數 sum 時，可以於程式中做出如下的宣告：

```
short int sum;         /* 宣告 sum 為短整數 */
```

如此，編譯器即會配置一塊佔有 2 個位元組的記憶空間供 sum 變數使用。由於變數 sum 宣告成短整數，所以它的範圍只能在-32768 到 32767 之間。相同的，於上面的宣告語法中，關鍵字 int 是可以省略的。

無號整數

在宣告整數資料型態時，我們還可以加上 unsigned 這個關鍵字，使得它成為「無號」整數。當資料絕對不會出現負數的時候（例如班上學生的總人數），就可以用無號整數來儲存它。如此一來，這個無號整數變數的儲存範圍便只能是整數，且正數的表示範圍也會變成原先的兩倍。無號整數所佔記憶體空間及可表示的範圍如下所示：

表 3.2.2 無號整數的資料型態

資料型態	型態說明	位元組	表示範圍
unsigned long int	無號長整數	4	0 到 4294967295
unsigned int	無號整數	4	0 到 4294967295
unsigned short int	無號短整數	2	0 到 65535

舉例來說，想宣告一個無號的整數變數 num 時，可以於程式中做出如下的宣告：

```
unsigned int num;      /* 宣告 num 為無號整數 */
```

此時編譯器便會配置 4 個位元組的記憶空間供變數 num 使用。

溢位的發生

當數值的大小超過變數可以表示的範圍時，便會發生溢位（overflow）。下面的程式範例中，我們宣告了短整數 sum 與 s，並將 s 設值為短整數可以表示範圍的最大值，然後分別將 s 的值加 1 及加 2，再設給 sum 存放，用來了解溢位發生的情形：

```
01  /* prog3_2, 短整數資料型態的溢位*/
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     short sum,s=32767;        /* 宣告短整數變數 sum 與 s */
07
08     sum=s+1;
09     printf("s+1= %d\n",sum);     /* 列印出 sum 的值 */
10
11     sum=s+2;
12     printf("s+2= %d\n",sum);     /* 列印出 sum 的值 */
13
14     system("pause");
15     return 0;
16  }

/* prog3_2 OUTPUT---
s+1= -32768
s+2= -32767
--------------------*/
```

程式解說

於 prog3_2 中，第 6 行宣告了短整數變數 sum 與 s，並設定 s 的值為短整數所容許的最大值（32767）。當 s 的值加上 1，再設給 sum 變數存放，從第 9 行的輸出可以發現，sum 的值變成-32768，恰為短整數可以表示範圍的最小值，這就是資料型態的溢位（overflow）。相同的，第 11 行把 s 的值加上 2，再把結果設回給 sum 變數存放，由第 12 行的輸出可看出，其結果變成短整數可表示範圍中的次小值。

上述的情形就像是計數器的內容到最大值時，會自動歸零（零在計數器中是最小值）一樣，而在短整數中最小值為-32768，所以當短整數 s 的值最大時，加上 1 就會變成最小值-32768，這就是溢位的發生，如下面的圖例說明：

```
short sum,s=32767;
sum=s+1;
```

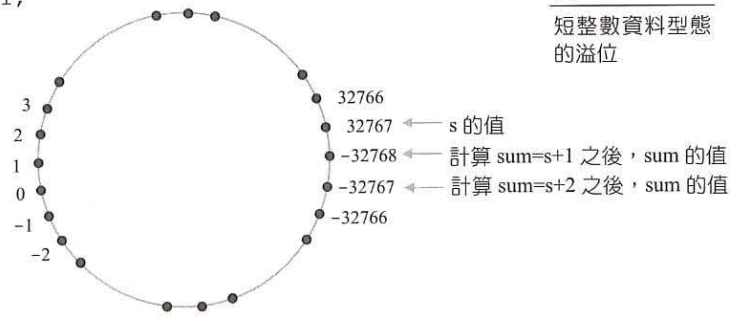


圖 3.2.1
短整數資料型態的溢位

於本例中，如果修改第 6 行，把 sum 宣告成 int，s 宣告成 short，則溢位就不會發生，這是因為變數 sum 可儲存的範圍，可達 4 個位元組之故（short 只用兩個位元組來儲存資料），由此可知，要避免溢位的發生，慎選變數的資料型態是很重要的。 ❖

3.2.2 字元型態 char

字元型態佔有 1 個位元組（byte），可以用來儲存字元。通常字元會被編碼，亦即把每一個字元均編上一個整數碼，以方便處理這些字元，其中 ASCII 是較為人知的編碼系統（請參閱附 D）。

ASCII 是 American Standard Code for Information Interchange 的縮寫，用來制訂電腦中每個符號對應的整數代碼，這個代碼也叫做電腦的內碼（code）。每個 ASCII 碼是以 1 個位元組儲存，數字 0 到 127 代表不同的常用符號，例如大寫 A 的 ASCII 碼是 65，小寫 a 則是 97。

由於每個 ASCII 碼佔了一個位元組，每個位元組有 8 個位元，所以每個位元組可以表示 $2^8 = 256$ 個字元，但數字 0 到 255 中，較高的位元（即 128~255）並沒有被使用到，所以後來又將這些位元也編入 ASCII 碼中，成為八個位元的「延伸 ASCII 碼」（extended ASCII）。延伸的 ASCII 碼裡加上了許多數學與表格框線等特殊符號，成為目前最常用的內碼。

想在程式裡宣告某個字元變數，並設值給它，可利用下面的語法：

```
char ch;          /* 宣告字元變數 ch */
ch='A';           /* 將字元常數'A'設值給字元變數 ch */
```

當然，您也可以在宣告的同時便設定初值：

```
char ch='A';      /* 宣告字元變數 ch，並將字元常數'A'設值給它 */
```

請您注意，字元常數必須放在單引號裡面，而不是雙引號。此外，我們也可以把整數設給字元變數，這種方式是利用 ASCII 碼來設定字元變數。例如小寫字母 a 的 ASCII 碼是 97，我們便可直接把整數 97 設定給字元變數 ch 存放，如下面的範例：

```
char ch=97;       /* 宣告字元變數 ch，並設值為 ASCII 碼為 97 的字元 */
```

有了上面的認知之後，現在您可以知道下面兩個敘述的不同了：

```
char ch='7';      /* 將字元常數'7'設給字元變數 ch */
char ch=7;        /* 將 ASCII 碼為 7 的字元給字元變數 ch */
```

如果要在 printf() 函數裡印出字元，列印格式碼可用「%c」，如要印出字元的 ASCII 碼，則可用「%d」。於下面的程式中，我們分別以字元格式碼「%c」與整數格式碼「%d」來列印字元 a，用以驗證 a 的 ASCII 碼是 97：

```
01  /* prog3_3, 字元的列印*/
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     char ch='a';                  /* 宣告字元變數 ch，並設值為'a' */
07     printf("ch= %c\n",ch);        /* 印出 ch 的值 */
08     printf("ASCII of ch= %d\n",ch);      /* 印出 ch 的十進位值 */
09
10     system("pause");
11     return 0;
12  }

/* prog3_3 OUTPUT---

ch= a
ASCII of ch= 97
--------------------*/
```

程式解說

於本例中，第 6 行宣告了字元變數 ch，並設值為 'a'，第 7 行利用字元格式碼「%c」印出 ch 的值，因此程式的輸出為小寫英文字母 a。第 8 行則是以整數格式碼「%d」來列印 ch 的值，因此輸出為字元 'a' 的 ASCII 碼 97。 ❖

除了直接設定字元變數為某個特定的字元之外，我們也可以把字元變數設值為某個字元的 ASCII 碼，如此一來，這個字元變數就等同於 ASCII 碼所對應到的字元，於如下面的程式範例：

```
01  /* prog3_4, 以 ASCII 碼設定字元 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     char ch=90;                  /* 將整數 90 設給字元變數 ch */
07     printf("ch=%c\n",ch);        /* 印出 ch 的值 */
08
09     system("pause");
10     return 0;
11  }

/* prog3_4 OUTPUT---
ch=Z
--------------------*/
```

程式解說

於本例中，第 6 行宣告了字元型態的變數 ch，並將它設值為 90，這個設定代表把變數 ch 的值設定為 ASCII 碼為 90 的字元，也就是 Z 這個大寫的英文字母。因此，第 7 行由 printf() 函數可輸出 Z。

附帶一提，字元常數與字串常數是有區別的。字元常數是以一對單引號包圍字元，而字串常數則是以一對雙引號包圍。例如 'a' 是一個字元，而 "holiday" 即為一字串常數。當然 "a" 可看成是只包含了一個字元的字串，但字串 "a" 和字元 'a' 所代表的意義並不相同，C 語言裡處理字元和字串的方式也不一樣。關於這兩者真正的區別，在第九章中會有詳細的討論。 ❖

在使用 ASCII 碼時，要注意的是，數字字元（如字元 '2'）和它相對應的 ASCII 碼是不同的，舉例來說，字元 '2' 的 ASCII 碼為 50，並不是整數 2 這個值，如下面的範例：

```
01  /* prog3_5, 數字字元與其相對應的 ASCII 碼 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06      char ch='2';                    /* 宣告字元變數 ch，並設值為'2' */
07      printf("ch=%c\n",ch);           /* 印出字元變數 ch */
08      printf("the ASCII of ch is %d\n",ch);  /* 印出 ch 的 ASCII 碼 */
09
10      system("pause");
11      return 0;
12  }

/* prog3_5 OUTPUT----
ch=2
the ASCII of ch is 50
-----------------------*/
```

程式解說

於本例中，第 6 行宣告了字元變數 ch，並設值為字元常數 '2'，第 7 行印出了字元變數 ch 的值，第 8 行則是印出了 ch 的 ASCII 碼。從本例的輸出中，您可以看到字元 '2' 的 ASCII 碼為 50，而不是整數 2。 ❖

稍早曾提到，字元型態也算是整數型態的一種，因此我們可以將字元變數設值為整數。但是字元型態的表示範圍只有 0~255，如果我們將大於 255 的整數以字元格式碼「%c」印出，會發生什麼問題呢？

於下面的程式中，我們宣告一個整數變數 i 並設值為 298（大於字元型態的表示範圍 255），可以想像會發生什麼事嗎？

```
01  /* prog3_6, 字元型態的列印問題*/
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     int i=298;
07     printf("ASCII(%d)=%c\n",i,i);   /* 印出 ASCII 碼為 i 的字元 */
08
09     system("pause");
10     return 0;
11  }
```

```
/* prog3_6 OUTPUT---
ASCII(298)=*
--------------------*/
```

程式解說

當我們將大於 255 的整數以字元格式碼「%c」印出時，結果出現了星形符號「*」，這是因為字元只佔有 1 個位元組，而整數有 4 個位元組，所以當「%c」遇到超過 255 的數值時，就只會截取後面 1 個位元組的資料，以上面的例子來說，298 的二進位為 100101010，被「%c」截取後面 8 個 bits（1 個位元組）後變成 00101010，剛好是十進位的 42，而 ASCII 碼 42 就是「*」符號，如下圖所示：

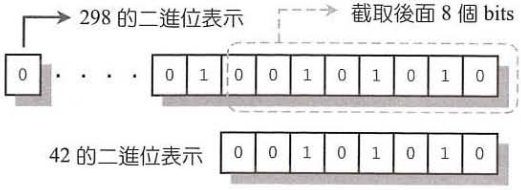


圖 3.2.2
字元型態的變數，若設值超過 255，則只會截取後面 8 個 bits

這種截取後面 1 個位元組的方式，相當於數值除以 256 後的餘數。您可以試著將 298 除以 256 後取其餘數，確認餘數是否為 42。 ❖

跳脫字元與跳脫序列

對於某些無法直接用鍵盤輸入的字元，C 語言是以反斜線字元「\」，加上一個控制碼做為一個完整的特殊字元，以便和正常的字元有所區別。當程式遇到「\」這個字元與控制碼時，便會依照控制碼所代表的意義來執行程式。例如，「\a」代表警告音，其中字元「a」代表一個控制碼。當編譯式編譯到「\a」時，便知道它是一個警告音。

由於反斜線字元「\」之後緊接一個字元時，這個字元會被解譯成控制碼，它已經跳脫了原來的涵意，因此反斜線「\」稱為跳脫字元（escape character），而反斜線「\」加上後面的控制碼，則稱為跳脫序列（escape sequence）。例如「\a」就是一個代表警告音的跳脫序列。

下表列出了常用的跳脫序列與其相對應的 ASCII 碼。您可以把字元變數直接設值為跳脫序列，或者是它的 ASCII 碼，然後放在 printf() 函數裡使用：

表 3.2.3 常用的跳脫序列

跳脫序列	所代表的意義	十進位 ASCII
\a	警告音(alert)	7
\b	倒退一格(backspace)	8
\n	換行(new line)	10
\r	歸位(carriage return)	13
\0	字串結束字元(null character)	0
\t	跳格(tab)	9
\\	反斜線(backslash)	92
\'	單引號(single quote)	39
\"	雙引號(double quote)	34

以下面的程式為例，我們將 beep 設值為 '\a'，並將字元變數 beep 所代表的十進位值列印於螢幕上，當程式執行到 printf() 這行指令時，您還會聽到 "嗶" 一聲警告音呢！

```
01  /* prog3_7, 跳脫序列的列印*/
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     char beep='\a';          /* 宣告字元變數 beep，並設定其值為'\a' */
07     printf("%c", beep);      /* 響一聲警告音 */
08     printf("ASCII of beep=%d", beep);  /* 印出beep的ASCII值 */
09
10     system("pause");
11     return 0;
12  }
```

```
/* prog3_7 OUTPUT---
ASCII of beep=7
--------------------*/
```

程式解說

本例執行後，電腦會發出 "嗶" 一聲的警告音。此外於第 6 行中，不管您是設定

```
char beep='\a';    /* 宣告字元變數 beep，並設定值為'\a' */
```

或是

```
char beep=7;       /* 宣告字元變數 beep，並設值為 ASCII 碼為 7 的字元 */
```

皆可以聽到警告音，但是建議使用第一種方式；因為它不但好記（跳脫序列「\a」的 a 即為英文 alarm 的開頭字母），同時並不是每種編譯程式都使用 ASCII 碼，若是使用跳脫序列，將可以提高程式的可攜性。 ❖

我們再舉一個例子來說明跳脫序列的應用，若是想印出

```
"We are the World"
```

字串，由於雙引號在 C 語言另有其代表的意義，因此在 printf() 函數中不能直接將它列印出來，此時利用跳脫序列「\"」即可順利解決這個問題，如下面的程式：

```
01  /* prog3_8, 跳脫序列「\"」的列印 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06      char ch='\"';      /* 宣告字元變數 ch，並設值為'\"' */
07      printf("%cWe are the World%c\n",ch,ch);      /* 印出字串 */
08
09      system("pause");
10      return 0;
11  }

/* prog3_8 OUTPUT---
"We are the World"
----------------------*/
```

程式解說

於本例中，第 6 行宣告了 ch 變數，並設值為跳脫序列「\"」，第 7 行則以「%c」格式碼印出了 ch 的值。從程式的輸出可看出，雙引號現在已經可以由 printf() 函數正確的列印出來了。 ❖

3.2.3 浮點數型態 float

在日常生活中經常會使用到小數型態的數值，如里程數、身高、體重等需要更精確的數值時，整數就不敷使用。在數學中，這些帶有小數點的數值稱為實數（real numbers），在 C 語言裡，這種資料型態稱為浮點數（floating point），例如，2.7、4.98 與 3.14159 等皆為浮點數。於 Dev C++中，浮點數型態的長度為 4 個位元組，有效範圍為 $1.2\times10^{-38} \sim 3.4\times10^{38}$。

想要宣告一個浮點數變數 num，可利用下面的語法：

```
float num;    /* 宣告浮點數變數 num */
```

如要在宣告浮點數變數 num 時便一併設定初值，可用下面的語法：

```
float num=5.46F;      /* 宣告浮點數變數 num，並設值為 5.46F */
```

於上例中，我們在數字 5.46 的後面加上一個字母 F，用來表示 5.46 是一個浮點數常數。如果沒有在數字之後加上字母 F，則編譯器把它看成是「倍精度浮點數」（double）型態的常數。此時如果把 double 型態的常數設給 float 型態的變數，便會有型態轉換上的問題。

在某些編譯程式裡，把 double 型態的常數設給 float 型態的變數，在編譯時並不會發生錯誤，但有些編譯程式對於語法的檢查較嚴，則會有警告訊產生。建議您在浮點數常數的結尾加上英文字母 F（或小寫的 f），用以明示此常數為 float 型態的常數，而非 double。

浮點數的表示方式，除了一般帶有小數點的形式外，還可用指數的型態表示。舉例來說，245.32 可以表示成 2.4532E2 （2.4532×10^{2}）、0.07652 可以表示成 7.652E-2（7.652×10^{-2}）等。在使用 printf() 函數時，如要印出浮點數，可用「%f」格式碼，如要以指數的型式印出，可用「%e」格式碼，如下面的範例：

```
01 /* prog3_9, 浮點數的列印 */
02 #include <stdio.h>
03 #include <stdlib.h>
04 int main(void)
05 {
06    float num1=123.45F;     /* 宣告 num1 為浮點數，並設值為 123.45F */
07    float num2=4.56E-3F;    /* 宣告 num2 為浮點數，並設值為 4.56E-3F */
08
09    printf("num1=%e\n",num1);    /* 以指數的型態印出 num1 的值 */
```

```
10     printf("num2=%f\n",num2);    /* 以浮點數的型態印出 num2 的值 */
11
12     system("pause");
13     return 0;
14  }

/* prog3_9 OUTPUT---
num1=1.234500e+002
num2=0.004560
----------------------*/
```

程式解說

於本例中，第 6 行宣告了浮點數變數 num1，並設值為 123.45F；第 7 行宣告了浮點數變數 num2，並設值為 4.56E-3F（相當於 0.00456）。於這兩行變數的設定中，您可以發現數字之後都加上了一個英文字母 F，用以表示它是浮點數常數。

第 9 行以指數的型式印出 num1 的值，得到 1.234500e+002（相當於 1.2345×10^{2}），第 10 行則是以浮點數的格式來印出指數 4.56×10^{-3}。 ❖

3.2.4 倍精度浮點數型態 double

當浮點數的表示範圍不夠大的時候，則可以使用倍精度浮點數（double）。於 Dev C++ 中，double 型態的長度為 8 個位元組，有效範圍為 $2.2\times10^{-308}\sim1.8\times10^{308}$。利用 printf() 函數來列印 double 型態的變數時，列印字元也是用「%f」格式碼。

除了 double 可表示範圍遠大於 float 之外，它可表示的數值精度也遠大於 float 的精度。float 型態約略只有 7~8 個位數的精度，而 double 型態則可達 15~16 個位數。

下面的範例分別宣告了一個浮點數變數 num1 與一個倍精度浮點數變數 num2，並設值為一個具有 15 個位數的浮點數變數常數，用以觀察 float 與 double 型態可表示的數字精度。程式的撰寫如下：

```
01 /* prog3_10, float 與 double 精度的比較 */
02 #include <stdio.h>
03 #include <stdlib.h>
04 int main(void)
05 {
06    float  num1=123.456789012345F; /* 宣告 num1 為 float，並設定初值 */
07    double num2=123.456789012345;  /* 宣告 num2 為 double，並設定初值 */
08
09    printf("num1=%16.12f\n",num1); /* 列印出浮點數 num1 的值 */
10    printf("num2=%16.12f\n",num2); /* 列印出倍精度浮點數 num2 的值 */
11
12    system("pause");
13    return 0;
14 }
```

```
/* prog3_10 OUTPUT-----
num1=123.456787109375
num2=123.456789012345
-------------------------*/
```

程式解說

於本例中，第 6 行宣告了 float 型態的變數 num1，並設值為 123.456789012345F。由於 float 的精度只到 7~8 個數字，因此雖然設定了具有 15 個位數的浮點數給 num1，但從第 9 行的輸出中可以看出，num1 還是只能容納 8 個位數的精度，如下圖所示：

圖 3.2.3
float 型態只有 7~8 個數字的精度

第 7 行則是宣告了 double 型態的變數 num2，相同的，我們也設定了擁有 15 個數字精度的數值給它。因 double 的精度可達 15~16 個有效數字，因而第 10 行的 printf() 函數可列印出完整的 15 個數字，如下圖所示：

```
double num2=123.456789012345;
```

圖 3.2.4
double 型態的變數可達 15~16 個數字的精度

於本例中，讀者可以發現 9~10 行使用了「%16.12f」這個列印格式。這個格式裡，16.12f 代表了以 16 個字元的欄寬來列印浮點數，其中小數點之後的位數為 12 個。關於這種格式化的輸出技巧，於下一章中會有更詳盡的介紹。 ❖

3.3 查詢常數、變數或資料型態所佔位元組

如果想知道某個常數、變數，或某種資料型態佔了多少個位元組，可利用 C 語言所提供的關鍵字 sizeof 來查詢。下表列出了查詢變數佔了多少個位元組的語法：

```
sizeof 變數或常數名稱;
或
sizeof(變數或常數名稱);
```

格式 3.3.1
查詢變數所佔的位元組

由上面的格式可知，如要查詢某個變數或常數佔了多少個位元組，sizeof 後面可以直接加上變數或常數的名稱，或者是把變數或常數放在括號內。如果是利用 sizeof 查詢某種資料型態所佔的位元組，則資料型態的名稱必須放在括號內，如下面的語法：

```
sizeof(資料型態名稱);
```

格式 3.3.2
查詢資料型態所佔的位元組

下面的程式是利用 sizeof 指令查詢 Dev C++中，各種基本資料型態所佔用的位元組：

```
01 /* prog3_11, 列印出各種資料型態的長度 */
02 #include <stdio.h>
03 #include <stdlib.h>
04 int main(void)
05 {
06    char ch;              /* 宣告字元變數 ch */
07    float num;            /* 宣告浮點數變數 num */
08
09    printf("sizeof(2L)=%d\n",sizeof(2L));    /* 查詢常數 2L 所佔位元組 */
10
11    printf("sizeof(ch)=%d\n",sizeof(ch));    /* 查詢字元變數ch所佔位元組 */
12    printf("sizeof(num)=%d\n",sizeof(num));  /* 查詢變數 num 所佔位元組 */
13
14    printf("sizeof(int)=%d\n",sizeof(int));  /* 查詢 int 型態所佔位元組 */
15    printf("sizeof(long)=%d\n",sizeof(long));  /* 查詢 long 型態所佔位元組 */
16    printf("sizeof(short)=%d\n",sizeof(short));/* 查詢 short 所佔位元組 */
17
18    system("pause");
19    return 0;
20 }
```

```
/* prog3_11 OUTPUT---
sizeof(2L)=4
sizeof(ch)=1
sizeof(num)=4
sizeof(int)=4
sizeof(long)=4
sizeof(short)=2
-----------------------*/
```

程式解說

於本例中，我們分別以 sizeof 指令查詢常數、變數與資料型態所佔的位元組，藉以學習 sizeof 指令的用法。建議您可以把本範例的結果與表 3.2.1 做一個比較，用以驗證每一種資料型態所佔的位元組。 ❖

3.4 資料型態的轉換

有些時候，我們可能會想把變數的型態做轉換，例如把整數轉換成浮點數，或把浮點數轉換成整數等等，以符合程式所需。這個時候便可利用型態的轉換技巧來達成轉換的目的。將資料型態轉換成另一種型態的語法如下：

```
(欲轉換的資料型態) 變數名稱;
```

格式 3.4.1
資料型態的強制性轉換語法

值得一提的是，把浮點數強制轉換成整數時，編譯器並不會做四捨五入的動作，而是直接將小數部份捨棄，只留下整數的部份。下面是把浮點數轉換成整數的範例：

```
01 /* prog3_12, 資料型態的轉換*/
02 #include <stdio.h>
03 #include <stdlib.h>
04 int main(void)
05 {
06    int n1,n2;
07    float num1=3.002F,num2=3.988F;
08
09    n1=(int) num1;       /* 將浮點數 num1 轉換成整數 */
10    n2=(int) num2;       /* 將浮點數 num2 轉換成整數 */
11
12    printf("num1=%f, num2=%f\n",num1,num2);  /* 印出浮點數的值 */
13    printf("n1=%d, n2=%d\n",n1,n2);  /* 印出浮點數轉成整數後的值 */
```

```
14
15     system("pause");
16     return 0;
17  }

/* prog3_12 OUTPUT---------
num1=3.002000, num2=3.988000
n1=3, n2=3
-----------------------------*/
```

程式解說

於本例中，第 6 行宣告了兩個整數 n1 與 n2，第 7 行則宣告了兩個浮點數變數 num1 與 num2，並設定初值為 3.002F 與 3.988F。第 9 與第 10 行分別將浮點數 num1 與 num2 轉換成整數，並設定給整數變數 n1 與 n2。從輸出中可看出，強制將浮點數轉換成整數時，浮點數之後的小數均會被捨棄。 ❖

在處理一些基本的數學運算時，也常會使用到強制型態轉換。例如，在 C 語言裡進行除法運算時，如果是整數相除，則運算結果只取其商（為一整數），並捨棄掉所有的小數。因此，若是想讓整數相除時，也能得到小數位數時，可將整數轉換成浮點數，再進行除法運算，如下面的範例：

```
01  /* prog3_13, 資料型態的轉換*/
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     int num=5;
07
08     printf("num/2=%d\n",num/2);              /* 整數相除 */
09     printf("(float)num/2=%f\n",(float)num/2); /* 將整數轉成浮點數，再做除法 */
10
11     system("pause");
12     return 0;
13  }
```

```
/* prog3_13 OUTPUT--------
num/2=2
(float)num/2=2.500000
-----------------------------*/
```

程式解說

於本例中，第 8 行計算 num/2，整數變數 num 的值為 5，5/2=2.5，由於變數 num 與常數 2 都是整數，因此計算的結果取其商，也就是整數 2。第 9 行則是把整數 num 轉換成浮點數後，再與 2 相除，則可得到 2.5 這個結果。 ❖

事實上，當運算式中變數的型態不同時，C 語言會自動將可表示範圍較小的資料型態轉換成可表示範圍較大的資料型態（如 short 轉成 int，float 轉成 double 或 int 轉成 double 等），再進行運算。也就是說，假設有一個整數和倍精度浮點數作運算時，C 語言會把整數轉換成倍精度浮點數後再作運算，運算結果也會變成倍精度浮點數。關於運算式的資料型態轉換，於第五章裡會有更詳盡的介紹。

習 題 （題號前標示有 ✚ 符號者，代表附錄 E 裡附有詳細的參考答案）

3.1 變數與常數

✚ 1. 於下面的敘述中，試指出何者為變數，何者為常數：

(a) `int num=134;`

(b) `int sum=76844;`

(c) `double value=0.44632;`

2. 試修改 prog3_1，使得第 9 行與第 10 行可分別印出 num1 與 num2 的平方值。第 9 行與第 10 行的輸出結果應如下所示：

```
num1 的平方為 153760000
num2 的平方為 27.394756
```

3.2 基本資料型態

✚ 3. 下列何者是錯誤的常數？試指出其錯誤之所在。

(a) `134.45L`

(b) `10km24`

(c) `a2048`

(d) `1.3453F`

4. 試指出下列常數各是屬於哪一種型態：

(a) `124.23`

(b) `3.23E12F`

(c) `2.436F`

(d) `311980L`

(e) `1024`

5. 試將下列各數以 C 語言的浮點數指數寫法來表示：

   (a) -96.43　　(b) 1974.56

   (c) 0.01234　　(d) 0.000432

6. 試將下列各指數改寫成 C 語言的浮點數表示方式：

   (a) -9.5e-4　　(b) 3.78e+5

   (c) 5.12e-2　　(d) 6.1732e+12

7. 試說明下列字元的意義。

   (a) \b　　(b) \n

   (c) \t　　(d) \a

8. int、char、float 與 double 資料型態的變數，各佔有多少個位元組？它們能夠表示的數值範圍是多少？

9. unsigned 型態適用在何種資料型態？它有什麼特點？

10. 下列宣告變數的敘述中，哪一些可為 C 的編譯器所接受？

    (a) `bool flag=true;`

    (b) `int num=40;`

    (c) `double float sum=5.04;`

    (d) `long value=47828L;`

11. 下列的敘述中，試問應該用什麼型態的變數來描述下列各項較為恰當？

    (a) 一個班級學生人數的總數。

    (b) 紐約帝國大廈的樓層數。

    (c) 月球到地球的距離。

    (d) 手機的重量。

    (e) 您的身高與體重。

    (f) 這本 C 語言教學手冊的總頁數。

    (g) 一個離子（ion）的重量。

(h) 硬碟每秒的轉速。

(i) 地球上人口的總數。

12. 試寫一程式，利用設定字元變數 ch 為 ASCII 碼的方式讓電腦發出一個警告音（警告音的 ASCII 碼為 7）。

13. 下面的兩行敘述是程式碼的片斷：

```
01  char ch=312;
02  printf("%c\n",ch);
```

(a) 試問第一行代表什麼意義？

(b) 試說明執行第 2 行所得的結果，並說明為什麼會得到這個結果？

14. 請參閱下面的程式碼，然後回答接續的問題：

```
01   /* hw3_14, 數字溢位的練習 */
02   #include <stdio.h>
03   #include <stdlib.h>
04   int main(void)
05   {
06      unsigned short num=80000;
07      printf("%d\n",num);
08
09      system("pause");
10      return 0;
11   }
```

(a) 試說明執行此程式的結果，為什麼是 14464，而不是 80000 這個數字？

(b) 如果想讓本題第 7 行的執行結果恰好為 80000，應如何修改程式碼？

15. 請參閱下面的程式碼，然後回答接續的問題：

```
01   /* hw3_15, 數字精度的問題 */
02   #include <stdio.h>
03   #include <stdlib.h>
04   int main(void)
05   {
06      float num1=30000.1F;
07      float num2=0.0004F;
08      printf("%f\n",num1+num2);
```

```
09
10      system("pause");
11      return 0;
12  }
```

(a) 試執行此程式碼，您會得到什麼結果？

(b) 於數學上，30000.1+0.0004=30000.1004，試說明執行此程式碼後，為什麼得不到這個結果？

(c) 如果想讓本題的執行結果恰好為 30000.1004，應如何改進？試撰寫一個完整的程式碼來改進之。

3.3 查詢常數、變數或資料型態所佔位元組

16. 試撰寫一程式，利用 sizeof 關鍵字查詢下列各種資料型態所佔的位元組：

(a) unsigned int

(b) double

(c) unsigned short int

17. 試撰寫一程式，利用 sizeof 關鍵字查詢下列各常數所佔的位元組：

(a) 578

(b) 784000000

(c) 6.78f

(d) 718.26

(e) 6.42e127

3.4 資料型態的轉換

18. 假設浮點數變數 num1 與 num2 的值分別為 123.39f 與 3.8e5f，試撰寫一程式，將這兩個變數值轉換成整數。

19. 請參閱下面的程式碼，然後回答接續的問題：

```
01  /* hw3_19, 型態轉換的練習 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     int num1=5,num2=8;
07     printf("%d\n",num1/num2);
08
09     system("pause");
10     return 0;
11  }
```

(a) 試解釋第 7 行的輸出結果為何是 0？

(b) 試修改程式碼，利用型態轉換的方式，使得第 7 行的輸出結果為 0.625000。