國立屏東大學 資訊工程學系 程式設計(一)

6 變數、常數與資料型態

- 資料型態Data Types
- 變數Variables
- 常數Constants
- 顯式型態轉換Explicit Conversions

程式設計主要的目的是為了解決問題，而大多數的問題又與資料處理相關。在C語言中，資料可以變數或常數的方式呈現，並加以運算或處理。另外，在C語言還將資料分類成不同的資料型態(Data Type)，例如我們可將數字分成整數、實數等。本章將就C語言的資料型態及變數(variable)與常數(constant)的宣告、初始化做一說明。在開始之前，先讓我們回顧一下在第5章所談到的變數與記憶體位址的概念：

摘錄第5章部份內容

事實上，電腦系統在執行程式時，所有的資料都是存放在記憶體當中。… 可是記憶體空間是由作業系統負責管理的，每當有程式要求執行時，作業系統的動態記憶體配置(或稱動態記憶體管理)模組會動態地分配一塊空間給該程式使用，但在絕大多數的情況下其所分配到的位置都不相同(當然也有相同的可能，只是機率比較低)。… 具體的做法是，先告訴電腦我們需要一個記憶體位置來存放一個整數，並且為了方便管理程式中還可能會需要的其它記憶體位址，我們必須為這個(需要空間來存放的)整數取個名字，例如以下的宣告：

```
int n;
```

上述的程式碼，稱為變數宣告(variable declaration)，表達了我們需要一個記憶體空間來存放一個稱為n的整數。在程式設計的術語裡，這個整數n被稱為變數(variable)。

換句話說，以上的宣告會要到一塊記憶體空間來存放變數n。當程式執行時，會產生一個稱為symbol table的表格，用來記錄所有的變數所分配到的記憶體空間，例如

symbol	type	address
n	int	100

當變數宣告完成後，我們可以使用&運算子來表達變數所在的記憶體位址，例如變數n所存放的位置，可以在程式碼中以&n來表示。在前述的例子中，變數n的記憶體位置為100，意即&n為100。所以當我們以scanf("%d", &n)來取得一個整數時，該整數的值(value)將會被存放到記憶體編號100的位置。假設使用者輸入的整數是3，那麼：從程式的角度來看，存放在記憶體位址100的變數n的值為3，也就是n=3，&n=100。

… 再強調一次，在程式碼中n代表一個變數，&n則代表這個變數的數值所存放的記憶體位址。

6.1 變數與記憶體位址

在第5章中，我們已經粗略地介紹了變數與記憶體的關係，本節將進一步詳細說明。由於程式設計的需要，我們必須在程式執行的階段利用記憶體空間來存放一些資料，並且可以對這些資料進行運算與處理。我們

將這些在程式執行階段所使用到的資料項目稱為變數(variable)。C語言要求所有的變數必須先宣告後才能使用。假設我們需要在程式中處理一個整數的運算，因此我們必須先進行整數變數的宣告：

```
int x;
```

上述的程式碼宣告了一個名稱為x的變數，在程式執行時，作業系統會分配足以放置整數的記憶體空間供它使用。所謂的足夠的空間指得是存放一個整數所需的空間，其大小視作業系統而定，通常是32bits，也就是4個bytes。

若要知道您所在的作業系統，使用多少空間來存放一個整數，可以使用sizeof(int)，它會傳回一個整數所佔的記憶體空間大小(其單位為byte)。

讓我們假設程式開始執行，並且變數x所需的記憶體空間被配置在0x7ffff34fff00記憶體位址，那麼程式的symbol table內容為：

symbol	type	address
x	int	0x7ffff34fff00

當中記載了一個名為x的符號，其型態為int，並且存放於0x7ffff34fff00位址起始的連續四個bytes(因為已經記載了其型態為int，所以系統可以知道其佔了四個bytes)。接著假設我們執行以下的程式碼：

```
x=38;
```

系統會檢視符號x所在的記憶體空間與其型態，然後將數值38放入對應的記憶體空間內，請參考figure 1。

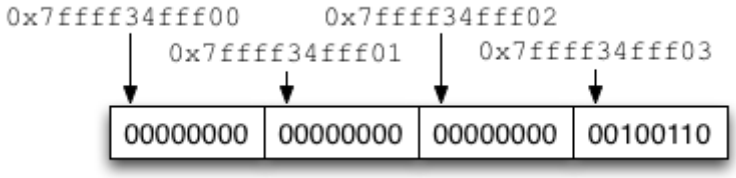

Fig. 1

由於一個整數假設為32bits，也就是四個bytes，所以我們會將0x7ffff34fff00~0x7ffff34fff03的記憶體空間，以38的二進位100110來儲存。通常描述記憶體空間的圖有兩種畫法，其一為figure 1的橫式，或是figure 2的直式，兩者的意義相同。

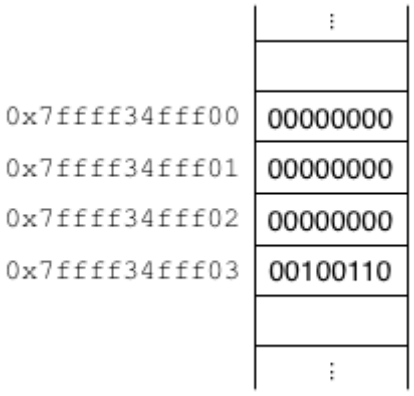

Fig. 2
記憶體位址的單位為byte，且通常以16進位來表示，0x7fff34fff00開頭處的0x就表示數值為16進位。

除卻這些細節，通常我們在設計C語言程式時，只需要抽像化地將變數x表達為記憶體中的某塊位置，不用特別去注意其所在的記憶體位置為何，figure 3就是我們常用的思考方式。

Fig. 3

如第5章所說明的，如果我們想要知道一個變數到底存放在哪個記憶體位址，可以使用&運算子來取得。請參考以下的程式，它宣告了一個整數變數，並將其值與記憶體位址加以輸出。

1 |h variableAndAddress.c

```c
#include <stdio.h>

int main()
{
  int x;

  x=38;

  printf("The value of x is %d.\n",x);
  printf("The memory address of x is %p\n", &x);
}
```

在printf()函式中，如要輸出記憶體位址，其format specifier為%p。

6.1.1 變數宣告(Variable Declaration)

現在，讓我們正式地介紹C語言變數宣告的語法(syntax)。請參考以下列的語法說明：

```
type variableName[=value]?[,variableName[=value]?]*
```

在上面的語法說明中，「[]」為選擇性的語法單元，其後接續「*」表示該語法單元可出現0次或多次；「?」表示出現0次或1次。另外還有「+」代表1次或多次。本書將使用這種表示法做為語法的說明。

其中type為型態，variableName為變數的名稱，中括號內的部份則是選擇性的(可以有，也可以忽略)，為該變數的初始數值。到目前為止，我們只介紹過int整數資料型態，其它可以使用的資料型態將在本章稍後加以介紹。

下面的程式碼片段宣告了一個名為x的整數變數，並且在後續設定其數值為38。

```c
int x;

...

x=38;
```

我們也可以將變數宣告與數值給定同時以一行程式碼來完成：

```
int x=38;
```

未設定初始值的變數，其數值是不可知的(有些系統會在配置記憶體空間時，同時將空間內所以的位元皆設定為0)。任何一個變數都不應該在未設定數值前就將它拿來運用，否則程式可能會遇到不可預期的錯誤。

我們也可以同時宣告有多個相同型態的變數，例如下面的程式碼，同時宣告了三個整數變數x、y與z、其中x不指定初始值、y與z的初始值則分別為3與6。要注意的是，我們在兩個變數宣告的中間，是以','加以隔開。

```
int x, y=3, z=6;
```

6.1.2 變數命名規則

程式可讀性(readability)

以下內容摘錄自skyang的網頁

為甚麼程式需要可讀性？因為你不會永遠自己一個人寫程式，你也不會永遠記得自己寫過的程式，請不要挑戰自己的記憶力。。如果其他人在使用你的程式的時候，你還必須隨時在側逐步解說，你大概也會瘋掉，所以怎麼樣把你的想法讓別人了解，又不用自己親自解釋，就成了團隊合作中非常重要的事情， 這部分除了好的文件(documentation)工夫以外，程式碼本身也有許多努力的空間。

甚麼是可讀性？簡單來說就是程式碼「有多麼容易被看懂」的程度，可讀性的來源有三：

約定成俗的寫作規格，例如命名規則, 段落縮排規格。 適時出現的註解。 程式邏輯的簡明精煉，除了這一項需要時間鍛鍊，前兩項是可以速成的。 換句話說，可讀性也是一種讓程式碼力求簡單明瞭的工夫。你可能會問，為甚麼程式要力求簡明？別人看不懂又怎麼樣？不管黑貓白貓，只要會抓老鼠就是好貓不是嗎？是啊，如果這隻貓不老不死，還是你馬子或兒子，永遠不會離開你。

試想，寫得不好懂的程式碼，如果500行的程式就已經很難了解，開發到20,000行的時候怎麼辦？這時候如果程式碼原作老鳥沒空教菜鳥？或是老鳥已經跳槽到別公司的時候怎麼辦？事實上台灣有許多的軟體公司(或團隊)的時間就這麼靜止在高手離職的那一天，達數年之久。

軟體公司老闆共通的痛苦有: 1. 招不到高手，2. 高手招來了沒空帶菜鳥，3.高手稍有不爽就跑, 4. 菜鳥學會了也跑，5. 陳年舊 code 沒人看得懂…那還開公司幹嘛？所以會投資在程式風格的教育訓練的老闆是有遠見的，最通常的做法就是頒布全公司(或全團隊)的程式風格規範書，並成為傳承的一部分。

變數名稱在程式語言中又稱為識別字(identifier)、其代表在程式執行階段的某個資料項目，因此建議使用較具意義的變數名稱，才容易理解、提升程式碼的可讀性(readability)、C語言變數命名具備以下規定：

- 只能使用英文大小寫字母、數字與底線(_)
- 不能使用數字開頭
- 不能與C語言的保留字相同

C語言共有以下34個保留字:

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

while							

C語言是case-sensitive的語言，意即大小寫會被視為不同的字元，因此以下的宣告其變數名稱皆是正確且不相同的：

```
int JUN, jun, Jun, JUn, JuN, jUn, jUN, juN;
```

為了讓程式碼的可讀性提升，使用有意義的變數名稱是相當重要的，有時我們甚至會使用一個以上的英文單字為變數命名，此時可以適當地調整大小寫或加上底線，例如下面是正確的宣告：

```
int bestStudent, BestStudent, best_student;
```

建議使用良好的命名規則，例如Camel Case或Hungarian Notation。目前C語言程式設計師通常以lower camel case法為變數命名，使用Hungarian Notation的程式市設計師也不再少數。

Camel Case命名規則

以英文命名，可由多個英文單字組成，每個單字除首字母外一律以小寫表示。任何兩個單字連接時，第二個單字的首字母必須使用大寫。第一個單字的首字母若以小寫表示，則稱為lower camel case反之若第一個單字的首字母以大寫表示時，稱為upper camel case例如：

- lower camel case
  - amy, userName, happyStory, setData, getUserInput等皆屬之
- upper camel case
  - Student, FulltimeStudent, CourseTime等皆屬之

6.2 常數

在程式碼中，經宣告並給定初始值後，就不再(也不允許)變更其數值的資料，就稱為常數(constant)

6.2.1 常數宣告

C語言的常數宣告語法如下：

```
const type variableName=value[,variableName=value]*
```

其實，常數的宣告就如同變數宣告一樣，只要在最前面加上const這個保留字即可，同時所有常數的宣告都必須給定初始值。請參考下面的程式碼片段：

```
const int x=3, y=5;
...

x=6;
```

```
...
```

上面的程式碼正確地宣告了兩個整數常數，但後續我們又改變了其中一個常數的數值，這樣會導致在編譯時的錯誤。您會得到“error: read-only variable is not assignable”的錯誤訊息。

6.2.2 常數定義

除了前述的常數宣告外，我們還可以使用#define這個preprocessor directive來定義常數。例如：

```
#define  PI      3.1415926

int main()
{
   int radius=5;
   float area;

   area = PI * radius * radius;

   ...
}
```

這個程式以#define定義了一個符號“PI”其值為3.1415926。當程式被編譯時，preprocessor會先將程式碼進行掃描，將其中所有出現PI之處，都改以3.1415926代替，然後再將代換後的程式碼交由compiler進行編譯。

6.3 基本資料型態

C語言提供多種資料型態，包含基本資料型態(basic data type)與自定資料型態(user-defined data type)兩類。本章僅就基本資料型態做一說明，自定資料型態請參閱後續章節。

6.3.1 整數型態/Integer Types

顧名思義，整數型態就是用以表示整數的資料，C語言中的整數型態，以integer的前三個字母int表示，唸做int或是integer都可。在現在的系統中，int通常為32位元(但在一些較舊的PC上，int也可能只是16位元)，其中最左邊的bit代表正負數 → 0代表正整數或0，1代表負整數。以32bits為例，最大的正整數為 $(0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111)_2 = 2,147,483,647$，也就是$2^{31}$-1；至於最小的負數並不是$(1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111)_2$= -2,147,483,647，而是 $(1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000)_2$，其值為-2,147,483,648=- 2^{32}。

若不想用最左邊的bit來表達正負號，可以使用unsigned這個保留字加在整數型態的前面。例如unsigned int可表達的範圍為0到$4,294,967,295$，也就是$2^{32} - 1$。unsigned除了是保留字外，我們也稱它為修飾字(modifier)。因為它可以加在其它保留字的前面，用以限縮或拓展其可表達的數值範圍。

除了unsigned修飾字外，整數int型態還可以搭配short與long兩個修飾字，將其表達空間加以調整。假設int為32bits，那麼short int則為16bits，long int則為64bits。您也可以再搭配unsigned修飾字一起使用，因此C語言一共有以下6種整數型態：

- Standard signed integer types (標準符號整數型態)

    - short int
    - int
    - long int
- Standard unsigned integer types (標準無符號整數型態)
    - unsigned short int
    - unsigned int
    - unsigned long int

型態也可以縮寫?

我們在宣告unsigned, short或long的整數變數時，可以將int省略，例如:可以short來代表short int；以unsigned代表unsigned int；以long代表long int；unsigned short代表unsigned short int；unsigned long代表unsigned long int。

6.3.1.1 整數型態數值範圍

如果想要知道您所使用的系統上，各種整數資料型態的最小值與最大值，可以使用在limits.h標頭檔中的巨集(macro)定義(關於巨集請參考本書第X章)。limits.h標頭檔共定義了9個與整數資料型態相關的巨集定義，詳如table 1。

巨集名稱	說明
SHRT_MIN	short int型態的最小值
SHRT_MAX	short int型態的最大值
INT_MIN	int型態的最小值
INT_MAX	int型態的最大值
LONG_MIN	long int型態的最小值
LONG_MAX	long int型態的最大值
USHRT_MAX	unsigned short int型態的最大值
UINT_MAX	unsigned int型態的最大值
ULONG_MAX	unsigned long int型態的最大值

Tab. 1: limits.h標頭檔定義的整數型態相關巨集

您可以使用上述9個巨集數值，印出各整數資料型態的最大值與最小值，請參考下面這個程式：

1 |h printIntLimits.c

```c
#include <stdio.h>
#include <limits.h>


int main()
{
  printf("short int [%d, %d]\n",SHRT_MIN, SHRT_MAX);
  printf("int [%d, %d\n", INT_MIN, INT_MAX);
  printf("long int [%ld, %ld]\n", LONG_MIN, LONG_MAX);
  printf("unsigned short int [0, %u]\n", USHRT_MAX);
  printf("unsigned int [0, %u]\n", UINT_MAX);
  printf("unsigned long int [0, %lu]\n", ULONG_MAX);
```

```
}
```

unsigned的整數其format specifier為%u，long型態的整數可在其format specifier前加上l，如%ld, %lu。

以64位元的Mac OS X 10.8.4為例，各整數型態的數值範圍如table 2。

Type	minimal value	maximal value
short int	-32768	32767
int	-2147483648	2147483647
long int	-9223372036854775808	9223372036854775807
unsigned short int	0	65535
unsigned int	0	4294967295
unsigned long int	0	18446744073709551615

Tab. 2: 64位元系統上的整數型態數值範圍

6.3.1.2 整數數值的表達

除了宣告變數為某種型態外，我們也可以直接在程式碼中使用整數數值，本節將說明各種型態的整數數值的表示方法。

在C語言中，整數數值可依其所使用的進位系統分成十進制(decimal, base 10)、二進制(binary, base 2)、八進制(octal, base 8)與十六進制(hexadecimal, base 16)等四種表示法。

- Decimal
  - 除正負號外，以數字0到9組成，除了數值0之外，不可以0開頭。
  - 例如：0, 34, -99393皆屬之
- Binary
  - 除正負號外，僅由數字0與1組成，必須以0b開頭。
  - 例如：0b0, 0b101, 0b111皆屬之
- Octal
  - 除正負號外，僅由數字0到7組成，必須以0開頭。
  - 例如：00, 034, 07777皆屬之
- Hexadecimal
  - 除正負號外，由數字0到9以及字母(大小寫皆可)a到f組成，必須以0x或0X開頭。
  - 例如：0xf, 0xff, 0X34A5, 0X3F2B01皆屬之

我們還可以在數值後面加上L(或l)、U(或u)強制該數值為long型態或是unsinged型態，兩者也可以混用表示unsigned long型態，例如：13L, 376l, 0374L, 0x3ab3L, 0xffffffUL, 03273LU等皆屬之。

6.3.1.3 整數型態數值的輸入與輸出

除了本書已經介紹過的%d、%u、%ld、%lu等適用於整數型態的format specifier外，C語言還有%o與%x用以表示八進制與十六進制的整數數值，全部彙整於table 3。

Format Specifier	意義	
%d	十進制的整數	
%o	八進制的整數	
%x	十六進制的整數	
%u	unsigned整數	
%h	short型態的整數	可在%d, %u, %o與%x前加上h搭配使用
%l	long型態的整數	可在%d, %u, %o與%x前加上l搭配使用

Tab. 3: 整數型態的Format Specifiers

我們可以使用scanf()與printf()函式，配合format specifier來取得或輸出特定的整數型態的數值。請參考intIO.c程式範例，示範如何取得各種型態的整數，並且加以輸出：

1|h intIO.c

```
#include <stdio.h>

int main()
{
  int x;
  short int y;
  long int z;

  printf("Please input an int:");
  scanf("%d",&x);

  printf("%d_decimal =  %o_octal = %x_hexadecimal.\n", x, x, x);

  printf("Please input a short int in octal:");
  scanf("%ho",&y);
  printf("%hd_decimal =  %ho_octal = %hx_hexadecimal.\n", y, y, y);

  printf("Please input a long int in hexadecimal:");
  scanf("%lx",&z);
  printf("%ld_decimal =  %lo_octal = %lx_hexadecimal.\n", z, z, z);
}
```

6.3.2 浮點數型態/Floating Types

顧名思義，浮點數型態就是用以表示小數的資料。C語言中有3種符點數的型態。float, double與long double。分別實作了 IEEE 754當中的單精確度、倍精確度與擴充精確度：

- float: 單精確度浮點數(single-precision floating-point)
- double: 倍精確度浮點數(double-precision floating-point)
- long double: 擴充精確度符點數(extended-precision floating-point)

一般而言，float型態適用於對小數的精確度不特別要求的情況，例如體重計算至小數點後兩位、學期成績計算至小數點後一位等情況。而double則用在重視小數的精確度的場合，例如台幣對美金的匯率、工程或

科學方面的應用等。至於long double則更進一步提供精確度，但非常少機會使用到。

6.3.2.1 精確度與數值範圍

由於在不同平台上，浮點數的實作差異甚大，所以C語言的標準並沒有提到float, double與long double該提供多少的精確度。本節以IEEE 754標準為參考，將浮點數的數值範圍與精確度做一整理，請參考table 4。如果還需要更詳細的資訊，請參考定義在float.h標頭檔中的巨集。

Type	smallest positive value	largest value	precision
float/single-precision	$1.17549x10^{-38}$	$3.40282x10^{38}$	6 digits
double/double-precision	$2.22507x10^{-308}$	$1.79769x10^{308}$	15 digits

Tab. 4: IEEE 754標準的浮點數型態數值範圍與精確度

6.3.2.2 數值的表達

浮點數數值的表達有兩種方式:

- Decimal
  - 除正負號外，以數字0到9以及一個小數點組成。
  - 例如：0.0, 34.3948, 3.1415926, -99.393皆屬之。
- scientific notation
  - 由一個decimal數字與exponent組成
  - decimal數字前可包含一個正負號，數字中可包含一個小數
  - exponent表示10的若干次方，以一個E或e後接次方數表達
  - 在E或e的後面可接一個正負號，表示該次方數為正或負
  - 例如：345E0, 3.45e+1, 3.45E-5皆屬之

C語言默認的浮點數型態為double，如果您要特別強制一個數值之型態為float或long double，可以在數值後接上一個F或L(大小寫皆可)。例如：3.45L, 3.45f等皆屬之。

6.3.2.3 數值的輸入與輸出

適用於浮點數型態的format specifier，彙整於table 5：

Format Specifier	意義
%f	float型態
%e	以scientific notation表示
%g	在%f或%e的結果中選擇較短者
%l	double型態，必須與%f, %e, %g搭配，例如%lf
%L	long double型態，必須與%f, %e, %g搭配，例如%Le

Tab. 5: 浮點數型態的Format Specifiers

6.3.3 字元型態/Character Types

所謂的字元型態就是用以表示文字、符號等資料，在C語言中只有一種字元型態：

- char

在不同的系統中，字元的數值可能會代表不同意義，視其所採用的字元集(character set)而定。現行最常見的字元集為ASCII(American Standard Code for Information Interchange)，請參考Wikipedia關於ASCII的說明。

6.3.3.1 數值範圍與運算

在C語言中，一個char型態的數值就是一個整數。具體來說，C語言使用8 bits的整數，使用從00000000到11111111共256種可能的數值來對映到ASCII的字元。例如'A'的ASCII數值為65，'0'為48等。

因此，我們可以把char型態的數值當成整數來進行運算，例如：

```
char c;
int i;

i ='a';    // i的值為97
c = 65;    // c的值為'A'
c = c + 1; // c的值為'B'
```

既然char型態就是整數，那可不可以再配合unsigned使用呢？因為char型態的整數數值是用以對應特定的字元集(如ASCII)，而每個字元集都有其可表達的字元個數要求，C語言會自動將char定義為singed或unsigned以符合字元集的需求。因此我們通常不會特別在char前加上unsigned，但是，如果您有某些較小的整數資料要處理，就可以考慮使用char來代替int，因為int為32 bits，甚至short int也要使用到16 bits，若您只需要處理一些介於-128到127之間的數值，那您就可以考慮改用char來代替int，或是宣告為unsigned char來處理那些介於0到255的正整數資料。

6.3.3.2 字元數值的表達

字元數值的表達方法有兩種:

- 字元值
  - 以一對''單引號將字元放置其中。
  - 例如，'A', '4', ' ', '&'皆屬之。
- 整數值
  - 對應在字元集中的整數值
  - 例如：65, 97等皆屬之

C語言針對一些特殊字元，提供一組escape sequence，如table 6

Escape Sequence	意義
\a	Alert(bell)
\b	Backspace
\f	form feed

Escape Sequence	意義
\n	new line
\r	carriage return
\t	Horizontal tab
\v	Vertical tab
\\	backslash
\?	?
\'	'
\“	”

Tab. 6: Escape Sequence

除此之外，還可以使用八進制或十六進制來表達字元：

- 八進制的escape sequence
  - 以\開頭，後面接八進制數值
  - 八進制數值在此處可不用0開頭
  - 例如： \33 或 \033皆屬之
- 十六進制的escape sequence
  - 以\x開頭，後面接十六進制數值
  - 十六進制的數值不可超過FF
  - 十六進制的數值不需以0x開頭
  - 十六進制的數值可以使用大寫或小寫的英文字母
  - 例如 \x1B 或 \x1b 皆屬之

6.3.3.3 數值的輸入與輸出

適用於字元型態的format specifier只有一個 %c您可以搭配%c於scanf()與printf()函式使用，以取得或輸出字元資料。此外，您還可以使用getchar()與putchar()函式來取得或輸出一個字元，例如：

```
char c; //宣告一個字元變數c

c = getchar(); //以getchar()取得使用者輸入的字元, 並放置於變數c

putchar(c);     //將字元變數c輸出
```

6.4 資料型態轉換

如果在程式碼中，我們想要把某個數值之型態加以轉換，可以使用顯示型態轉換(explicit conversion)來對數值進行強制的轉型(casting)使用的方法很簡單，只要在想要轉型的數值前加上一組()其中指定欲轉換的型態即可，例如:

```
int x;
long int y;

y=(long)x;
y = (long)(x+837);
x = (int)sizeof(int);
```

6.5 課後練習

作業4