國立屏東商業技術學院 資訊工程系 物件導向程式設計

4. 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。要注意的是C++語言大部份的資料型態皆與C語言一致，僅增加了bool資料型態。

4.0.1 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]?]*
```

<note tip> 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 </note>

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。

```
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;
```

<note important> 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 </note>

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;
```

還要注意的是，C++提供我們一種新的寫法，可以將變數的初始值以其它變數的運算加以表達，例如：

```
int a=5;

int b=a*2;
```

4.0.2 變數命名規則

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
- 大寫與小寫字元將視為不同字元
- 不能與C++語言的保留字相同
- 在C++的實作中，以一個底線開頭後接一個大寫字母，或是以兩個底線開頭的變數，被保留做為C++的實作用途(供編譯器及其相關資源使用)。因此，使用這樣的命名並不會造成compiler的錯誤，但有可能造成執行上的錯誤。

<note> C語言共有以下84個保留字:

alignas	alignof	and	and_eq	asm	auto	bitand
bitor	bool	break	case	catch	char	char16_t
char32_t	class	compl	const	constexpr	const_cast	continue
decltype	default	delete	do	double	dynamic_cast	else
enum	explicit	export	extern	false	float	for
friend	goto	if	inline	int	long	mutable
namespace	new	noexcept	not	not_eq	nullptr	operator
or	or_eq	private	protected	public	register	reinterpret_cast
return	short	signed	sizeof	static	static_assert	static_cast
struct	switch	template	this	thread_local	throw	true
try	typedef	typeid	typename	union	unsigned	
using	virtual	void	volatile	wchar_t	while	xor
xor_eq						

</note>

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。

<note tip>

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

</note>

4.1 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)。

4.1.1 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。

4.1.2 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。

4.2 基本資料型態

C++語言提供多種資料型態，包含基本型態(fundamental type)與複合資料型態(compound type)兩類。本章僅就基本型態做一說明，複合型態請參閱後續章節。

4.2.1 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<latex>(0111 1111 1111 1111 1111 1111 1111 1111)_2=2,147,483,647</latex>，也就是<latex>2^{31}-1</latex>，至於最小的負數並不是<latex>(1111 1111 1111 1111 1111 1111 1111

1111 1111)$_2$= -2,147,483,647</latex>而是<latex>(1000 0000 0000 0000 0000 0000 0000 0000)$_2$</latex>其值為<latex>-2,147,483,648=- 2^{32}</latex>

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<latex>4,294,967,295</latex>也就是<latex>$2^{32} - 1$</latex>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除此之外C++還提供了一種更長位數的型態long long int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8種整數型態：

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

<note important>

型態也可以縮寫?

我們在宣告整數型態的變數時，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 long long代表long long int以及unsigned short代表unsigned short int unsigned long代表unsigned long int unsigned long long代表unsigned long long int

</note>

4.2.1.1 整數型態數值範圍

C++與C語言一樣，在整數型態的數值範圍方面，都是取決於其所使用的記憶體空間。由於不同平台上可能會有差異C++語言僅提供規範，實際情形由各平台上的實作決定。因此，在一個平台上撰寫程式時，我們通常會使用以下的程式，先行瞭解各型態所佔的空間：

```
#include <iostream>
#include <climits> // use limits.h

int main()
{
  using namespace std;
  int n_int = INT_MAX;       // initialize n_int to max int value
  short int n_short = SHRT_MAX;  // symbols defined in climits file
```

```
  long int n_long = LONG_MAX;
  long long int n_llong = LLONG_MAX;

  // sizeof operator yields size of type or of variable
  cout << "int is " << sizeof (int) << " bytes." << endl;
  cout << "short is " << sizeof n_short << " bytes." << endl;
  cout << "long is " << sizeof n_long << " bytes." << endl;
  cout << "long long is " << sizeof n_llong << " bytes." << endl;
  cout << endl;

  cout << "Maximum values:" << endl;
  cout << "int: " << n_int << endl;
  cout << "short: " << n_short << endl;
  cout << "long: " << n_long << endl;
  cout << "long long: " << n_llong << endl << endl;

  cout << "Minimum int value = " << INT_MIN << endl;
  cout << "Bits per byte = " << CHAR_BIT << endl;

  return 0;
}
```

以我們的ws.csie.npic.edu.tw工作站為例，limits.cpp的執行結果如下：

```
[09:52 junwu@ws ch4]$ g++ limits.cpp
[09:52 junwu@ws ch4]$ ./a.out
int is 4 bytes.
short is 2 bytes.
long is 8 bytes.
long long is 8 bytes.

Maximum values:
int: 2147483647
short: 32767
long: 9223372036854775807
long long: 9223372036854775807

Minimum int value = -2147483648
Bits per byte = 8
[09:52 junwu@ws ch4]$
```

上述程式，主要使用了sizeof()函式以及位在climits標頭檔中的定義，有關climits標頭檔，可至/usr/include/c++目錄下查詢。下表為climits中所定義的部份常數：

Symbolic Constant	**Represents**
CHAR_BIT	Number of bits in a char
CHAR_MAX	Maximum char value
CHAR_MIN	Minimum char value

Symbolic Constant	Represents
SCHAR_MAX	Maximum signed char value
SCHAR_MIN	Minimum signed char value
UCHAR_MAX	Maximum unsigned char value
SHRT_MAX	Maximum short value
SHRT_MIN	Minimum short value
USHRT_MAX	Maximum unsigned short value
INT_MAX	Maximum int value
INT_MIN	Minimum int value
UINT_MAX	Maximum unsigned int value
LONG_MAX	Maximum long value
LONG_MIN	Minimum long value
ULONG_MAX	Maximum unsigned long value
LLONG_MAX	Maximum long long value
LLONG_MIN	Minimum long long value
ULLONG_MAX	Maximum unsigned long long value

Tab. 1: 在climits中定義的常數

4.2.1.2 整數數值的表達

<本節內容與C語言一致>

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)、LL(或ll)、U(或u)強制該數值為long型態、long long型態或是unsigned型態，也可以混用表示unsigned long型態或unsigned long long。例如：13L, 376l, 0374ULL, 0x3ab3L, 0xffffffUL, 03273LU等皆屬之。下面的程式，顯示如何將整數型態的數值以不同的數字系統輸出：

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

int main()
{
  int x = 0x7f;

  cout << "dec x=" << x << endl;
  cout << hex;
  cout << "hex x=" << x << endl;
  cout << oct;
  cout << "oct x=" << x << endl;

  return 0;
}
```

我們可以試著將程式開頭處的using namespace std;移除，看看會發生什麼事？原來像是endl, hex, oct, ...等都是定義在std這個namespace中，如果沒有先載入該名稱領域，你也可以std::endl, std::hex等方式撰寫程式碼，只是對於時常會用到的變數定義還是先載入會比較方便。

另外C++並未預先定義有關二進位數字系統的處理，下面這個程式參考自DANIWEB，可用以處理二進位的轉換，雖然使用到許多還未教到的技巧，但同學們可以先參考，待日後再回頭詳讀。

```
#include <iostream>

template < typename T >
inline T highbit(T& t)
{
  return t = (((T)(-1)) >> 1) + 1;
}

template < typename T >
std::ostream& bin(T& value, std::ostream &o)
{
  for ( T bit = highbit(bit); bit; bit >>= 1 )
  {
    o << ( ( value & bit ) ? '1' : '0' );
  }
  return o;
}


int main()
{
  unsigned long value = 0x12345678;
  std::cout << "hex: " << std::hex << value << std::endl;
  std::cout << "dec: " << std::dec << value << std::endl;
  std::cout << "oct: " << std::oct << value << std::endl;
  std::cout << "bin: ";
  bin(value, std::cout);
```

```
  std::cout << std::endl;
  return 0;
}
```

4.2.1.3 整數型態數值的輸入與輸出

除了使用C++的cout與cin來進行資料的輸出與輸入外，原本在C語言中所使用的各種輸出入方法仍可在C++中使用，下表彙整了原本在C語言中，有整數相關的format specifier外：

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. 2: 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：

```
#include <iostream>

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);
}
```

4.2.2 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。

4.2.2.1 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/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 3。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	<latex>$1.17549x10^{-38}$</latex>	<latex>$3.40282x10^{38}$</latex>	6 digits		
double/double-precision	<latex>2.22507×10	**{-308}</latex>**	<latex>1.79769×10	**{308}</latex>**	15 digits

Tab. 3: IEEE 754標準的浮點數型態數值範圍與精確度

4.2.2.2 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/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。

4.2.2.3 數值的輸入與輸出

如果要使用原本C語言中的輸出入方式，下表為適用於浮點數型態的format specifier。

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. 4: 浮點數型態的Format Specifiers

4.2.3 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/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。

4.2.3.1 數值範圍與運算

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。

4.2.3.2 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/C++語言針對一些特殊字元，提供一組escape sequence如table 5

Escape Sequence	意義
\a	Alert(bell)
\b	Backspace
\f	form feed
\n	new line
\r	carriage return
\t	Horizontal tab
\v	Vertical tab
\\	backslash
\?	?
\'	'
\“	”

Tab. 5: 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

4.2.3.3 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
```

4.2.4 布林型態/bool

布林型態為C++所新增的資料型態，其名稱為bool一個bool型態的資料只可能有true或false兩種可能的數值。與傳統的C語言一樣，若你要以整數來表達bool型態的值，則以0表示false其它非0的值皆視為true

```
bool isQuit = false;

int continueProcess = true; //將true轉換為1
```

4.3 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);
```