

程式設計

Ch02. Variable and Arithmetic

Chuan-Chi Lai 賴傳淇

Department of Communications Engineering
National Chung Cheng University

Spring Semester, 2024

Outline

- 1 計算機中的數值表示方式
- 2 變數與宣告
- 3 基本資料型別
- 4 格式化輸入輸出
- 5 算術運算
- 6 型別轉換
- 7 `<math.h>` 數學函式庫
- 8 補充資料

計算機中的數值表示方式

- 在數學領域的實數中，數值大致分為整數、有理數與無理數。
- 在現代主流計算機的存儲中，全部都是以 0 與 1 的串列來表示。
- 在有限的空間中，我們並無法在計算機中呈現現實世界的所有數值，特別是對於絕大部分的有理數與無理數，我們只能在計算機中以近似值的方式重現。

計算機中的數值表示方式

在計算機中，基本的數值分為以下三種：

- 整數 (Integers)
- 實數 (Real Numbers)
- 字元 (Characters)

整數 (Integers)

- 在數學中，整數的集合有無限個元素。但是，計算機的儲存空間是有限的，無法表示所有整數。
- 整數表示有兩種基本類型：
 - 無號整數 (unsigned integer)：用以表示非負整數，在計算機中以二進位來表示。
 - 有號整數 (signed integer)：用以表示正整數、負整數以及零，其中非負整數以二進位來表示，負整數以二補數表示。

實數 (Real Numbers)

- 計算機用以儲存具有小數部分的實數有兩種方式：
 - 定點數 (fixed-point number)
 - 浮點數 (floating-point number)
- 但是我們不可能精確的在計算機中表示一個實數。例如， $1/3=0.3333\dots$ ，我們不可能精確地將其儲存在有限大小的記憶體中，只能儲存其近似值。
- 在相同位元數的情況下，浮點數可表現的數值範圍比定點數來得多，因此當今的計算機普遍以浮點數來表示實數。

定點數 (Fixed-Point Numbers)

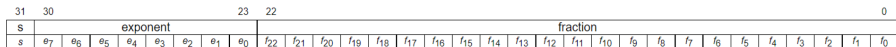
- 如同二進位的表示，每個位元都代表固定的值，但前半的位元代表整數部分，後半則代表小數部分。

	integer bits	fractional bits
$0.500 = \frac{1}{2}$	00000000	00000000.10000000 00000000
$1.250 = 1\frac{1}{4}$	00000000	00000001.01000000 00000000
$7.375 = 7\frac{3}{8}$	00000000	00000111.01100000 00000000

https://en.wikipedia.org/wiki/Computer_number_format

浮點數 (Floating-Point Numbers)

- 浮點數以科學記號的方式儲存數值，即“ \pm 分數 E 指數”。浮點數的位元包含三個部分：符號、指數和分數。而指數與分數部分所佔的位元數由 IEEE754 所規範。
- 下圖是 IEEE754 規範 32 位元浮點數中，符號 (S)、指數 (exponent) 和分數 (fraction) 部分所佔的位元。



https://zh.wikipedia.org/wiki/IEEE_754

字元 (Characters)

- 因為計算機的儲存是以 0 與 1 的序列組成，並無法直接表示人類語言的文字，因此我們使用字元編碼 (character encoding)，使整數對應到文字，例如繁體中文 Big-5、簡體中文 GBK、日文 cp932，以及萬國碼 UTF-8、UTF-16 等。
- 而最通用的英文字元編碼為 ASCII (American Standard Code for Information Interchange)。

ASCII

- ASCII 的每個字元均佔用 1 byte，但只使用了 7 bits，其最高位元始終為 0。
- ASCII 字元集分為兩類：
 - 控制字元 (0 ~ 31、127)
 - 可顯示字元 (32 ~ 126)，其中 32(16 進位的 20) 是空白鍵 (space)。
- 右圖是 16 進位對應的字元。

		most significant nibble							
		0_	1_	2_	3_	4_	5_	6_	7_
least significant nibble	_0	NUL	DLE	SP	0	@	P	'	p
	_1	SOH	DC1	!	1	A	Q	a	q
	_2	STX	DC2	"	2	B	R	b	r
	_3	ETX	DC3	#	3	C	S	c	s
	_4	EOT	DC4	\$	4	D	T	d	t
	_5	ENQ	NAK	%	5	E	U	e	u
	_6	ACK	SYN	&	6	F	V	f	v
	_7	BEL	ETB	'	7	G	W	g	w
	_8	BS	CAN	(8	H	X	h	x
	_9	HT	EM)	9	I	Y	i	y
	_A	LF	SUB	*	:	J	Z	j	z
	_B	VT	ESC	+	;	K	[^	}
	_C	FF	FS	,	<	L	\		
	_D	CR	GS	=	=	M]	_	}
	_E	SO	RS	.	>	N	^	n	~
	_F	SI	US	/	?	O	_	o	DEL

https:
//learn.microsoft.com/zh-tw/
azure/rtoS/netx-duo/appendix-e

變數與宣告 Variables and Declaration

變數

- 當我們需要在記憶體中儲存一個數值時，我們需要先宣告變數，隨後才能把數值存入其中。程式語言的變數具有以下 6 種屬性：
 - 名稱 (Name)：變數的識別字 (identifier)
 - 位址 (Address)：儲存在記憶體的位址 (address)
 - 型別 (Type)：決定變數所佔的記憶體大小與資料型別 (整數、浮點數、字元)
 - 值 (Value)：變數所儲存的數值
 - 作用域 (Scope)：表示變數在程式碼中能被取用的範圍
 - 生命周期 (Lifetime)：表示變數具有實際意義的時間，即從宣告到被釋放的時間

- 宣告一個整數變數 book 並印出其值與位址：

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main() //主程式
5 {
6     int book = 500; //宣告變數
7     printf("%d\n", book); //印出book的value
8     printf("%p\n", &book); //印出book的地址
9     return 0; //結束程式
10 }
```

C:\Projects\HelloWorld\test.e

500

000000000061FE1C

識別字 (Identifier)

- 程式中所用到的各種名稱、如變數名稱、常數名稱、函數名稱等。
- 識別字要自己取名字，最好是和其相關的有意義的名字。

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main() //主程式
5 {
6     int book = 500; //宣告變數
7     printf("%d\n", book); //印出book的value
8     printf("%p\n", &book); //印出book的地址
9     return 0; //結束程式
10 }
```

C:\Projects\HelloWorld\testLe x + v

500
000000000061FE1C

識別字命名規則

- 第一個字一定是英文字母或底線字元 “_”。
- 其他字元可以是英文字母、數字與底線 “_” 或 “\$” 符號組成。
- 長度不可超過 255 個字。
- 大小寫字母所代表意義不同 (case sensitive)。
- C 語言的保留字 (reserved words) 或稱關鍵字 (key words) 不可作為識別字。

正確	錯誤	說明
abc	3abc	第一個字母不可為數字
_ab	for	不可使用關鍵字
A12	A#12	不可使用特殊符號

保留字 (Reserved Words)

- 具有特殊意義的識別字。
- 程式中不可以使用保留字當作變數名稱、常數名稱或函數名稱。
- 保留字在不同的 Compiler 或不同的版本中可能有差異。
- C 語言的保留字：

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

變數的命名

- 在同個區塊 (block) 中，無法宣告兩個相同的名字的變數。
- 避免使用無意義的變數名稱，例如：a、b、c、a123、b001 等等。
- 變數名稱要與存放的資料相符，以免在閱讀上理解錯誤。
- 採用駝峰式命名法，若需要使用 2 個以上的單字，第二個單字以後開頭大寫，例如：testScore。
- 不建議使用縮寫。
- 如果要使用縮寫，一般通則可省略母音，如 button 可縮寫為 btn，count 可縮寫為 cnt，也可保留第一音節，如 number 可縮寫為 num。不過請注意在縮寫之後要確定能不能讓其他人看得懂。

位址 (Address)

- C 語言中宣告的每個變數都實際占用於記憶體中的某段空間。而記憶體中的每個位元組 (byte) 都具有一個編號，稱之為記憶體位址 (memory address)。
- 變數的記憶體位址表示其佔用於記憶體區段的起點位置。
- 我們可以使用「&」取址運算子來取得變數的位址，61FE1C (16 進位)。

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main() //主程式
5 {
6     int book = 500; //宣告變數
7     printf("%d\n", book); //印出book的value
8     printf("%p\n", &book); //印出book的地址
9     return 0; //結束程式
10 }
```



C:\Projects\HelloWorld\test.e x + v

500

000000000061FE1C

資料型別 (Data Type)

- C 語言的變數宣告必須指定該變數的資料型別，將資料型別後方寫下欲宣告的變數名稱即可宣告該變數。
- C 語言的基本資料型別為 int (整數)、char(字元)、float(浮點數)、double(雙精度浮點數)，於本章下一節會詳細介紹。

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main() //主程式
5 {
6     int book = 500; //宣告變數
7     printf("%d\n", book); //印出book的value
8     printf("%p\n", &book); //印出book的地址
9     return 0; //結束程式
10 }
```

C:\Projects\HelloWorld\testLe x + -

500
000000000061FE1C

- C 語言中，每個資料型別都有其固定的大小，因此變數占用於記憶體空間的大小是由變數的資料型別所決定的。例如 int 的大小是 4 bytes，因此 int 變數會占用 4 個 bytes 長度的連續記憶體空間。
- 在整個程式執行期間，變數名稱是不變的，但其所儲存的值可能會一直改變。
- 所佔用之記憶體位置由作業系統 (Operating System) 決定。
- 於前述的程式碼中，int 變數 book 的位址為 61FE1C (16 進位)，因為其大小為 4 bytes，因此變數 book 實際占用了記憶體中的 61FE1C、61FE1D、61FE1E、61FE1F (16 進位) 這四個位址。

變數的宣告、初始化與指派

- 若要在同一個敘述句宣告多個同樣型別的變數，可以使用逗號分隔變數名稱。
- 變數的值可選擇在宣告變數時同時初始化 (initial)，或是之後再指派 (assign) 數值。

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main() //主程式
5  {
6      int a = 100, b = 200; //宣告變數並初始化為指定值
7      int c, d; //宣告並初始化變數，此變數具有不定值
8      c = 300; //指派數值
9      d = 400; //指派數值
10 }
```

作用域與生命週期

- 作用域 (scope) 表示變數在程式碼中能被取用的範圍，而生命週期 (lifetime) 表示變數具有實際意義的時間，即從宣告到被釋放的時間。
- 一般而言，變數的作用域與生命週期為從該變數的宣告敘述開始，直到該區間 (block) 的結束為止。因此我們不可存取尚未宣告的變數。

基本資料型別 Basic Data Types

資料型別 (Data Types)

- 變數的資料型別決定了變數儲存的資料種類、資料的格式，以及該變數所佔的記憶體空間的大小。
- C 語言的資料型別包含整數 (int)、浮點數 (float)、字元 (char)，另外還有陣列 (array)、指標 (pointer)、結構 (struct)、聯集 (union) 等。
- 此章節我們只討論整數、浮點數、字元這 3 種最基本的資料型別。

變數、字面常量與資料型別

- 變數與字面常量 (literal constant) 均有屬於其的資料型別，變數的資料型別於宣告時決定，字面常量的資料型別則由字面的格式以及字尾決定。

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main() //主程式
5 {
6     int book = 500; //宣告變數
7     printf("%d\n", book); //印出book的value
8     printf("%d\n", &book); //印出book的地址
9     return 0; //結束程式
10 }
```

```
C:\Projects\HelloWorld\test.c  +  v
500
0000000000061FE1C
```

整數型別

- 整數分為有號整數 (signed) 以及無號整數 (unsigned)，並且包含基本的 int 在內，總共有 5 種大小變形，分別為 char、short int、int、long int、long long int (short int、long int、long long int 可省略“int”)。
- long int 的大小為 4 或 8 bytes，取決於編譯器。(註：MinGW 定義 long 為 4 bytes，其值的範圍與 int 相同)。

型態	值的範圍	型態	值的範圍	位元組
char	-128~127	unsigned char	0~255	1
short	-32,768~32,767	unsigned short	0~65,535	2
int	-2,147,483,648~2,147,483,647	unsigned int	0~4,294,967,295	4
long	相同於 int 或 long long， 取決於編譯器	unsigned long	相同於 unsigned int 或 unsigned long long	4 or 8
long long	-9,223,372,036,854,775,808~ 9,223,372,036,854,775,807	unsigned long long	0~ 18,446,744,073,709,551,615	8

整數的字面常量

- 整數可使用 10 進位、8 進位 (0 前綴)、16 進位 (0x 或 0X 前綴) 表示，並且可使用字尾 u、l 等來代表不同的整數型別。
- long int 的大小為 4 或 8 bytes，取決於編譯器。(註：MinGW 定義 long 為 4 bytes，其值的範圍與 int 相同)。

型態	字尾	字面常量範例
int	無	10, -5, 0, 0x5ac, 0X4B, 0123
unsigned int	u 或 U	10u, 0u, 0x5acu, 0123U
long	l 或 L	10l, -100L, 0x3a2L, 0246l
unsigned long	ul 或 UL	100UL, 0X12ABul, 0234ul
long long	ll 或 LL	100ll, -999LL, 0xabcLL, 0123ll
unsigned long long	ull 或 ULL	2300ull, 1000ULL, 0x1BAull, 0234ULL

浮點數型別

- 浮點數於 C 語言有兩種型別，float (單精度浮點數) 以及 double (雙精度浮點數)。其差別在於可儲存的值的大小以及能確保精準的位數。

型態	值的範圍	精確度	位元組
float	0 與 $\pm 2.939 \times 10^{-38} \sim \pm 3.403 \times 10^{+38}$	7 位數	4
double	0 與 $\pm 5.563 \times 10^{-309} \sim \pm 1.798 \times 10^{+308}$	15 位數	8

浮點數的字面常量

- 浮點數可使用小數或科學記號 (E 或 e) 格式表示。無字尾的浮點數預設為 double 型別，可使用字尾 f 或 F 指定為 float 型別。

型態	字尾	字面常量範例
double	無	10.0, -0.5, .123, 1.3E20, 2e10, -3.1E-2
float	f 或 F	10.0F, -3.14f, .25f, 1.45e30F, -7E-10f

字元型別

- char 不只屬於整數，本身也代表字元型別。於變數的運算 (如加減乘除) 時，會以整數的方式進行，並且於輸出 (如 printf) 時，會依據所指定的格式印出整數或該整數所對應的字元。

型態	值的範圍	位元組
char	-128~127	1

字元的字面常量


- 我們使用單引號括起一個或多個字作為字元的字面常量。並且可使用反斜線表示跳脫序列。

類別	範例	備註
普通字元字面常量	'a', 'b', 'c'	其值分別為 0x61、0x62、0x63。
簡單跳脫序列	'\t', '\n'	其值分別為 0x9、0xA，另外還有 \' \" \? \\ \a \b \f \r \v 總共 11 個字元。
八進位跳脫序列	'\0', '\10', '\123'	其值分別為 0、010、0123，最多可使用 3 位 8 進位數字。
16 進位跳脫序列	'\xab'	其值為 0xAB，最多可使用 2 位 16 進位數字。
多字元字面常量	'ABCD'	其值為 0x61626364，多字元字面常量型態為 int，最多可放入 4 個字元。

sizeof 運算子

- 使用 sizeof 運算子可以取得資料型別或數值所佔的位元組 (bytes)。

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main()
5 {
6     printf("Size of char: %d byte\n", sizeof(char));
7     printf("Size of short: %d byte\n", sizeof(short));
8     printf("Size of int: %d byte\n", sizeof(int));
9     printf("Size of long: %d byte\n", sizeof(long));
10    printf("Size of long long: %d byte\n", sizeof(long long));
11    printf("Size of float: %d byte\n", sizeof(float));
12    printf("Size of double: %d byte\n", sizeof(double));
13    system("pause");
14 }
```



C:\Projects\sizeoftype.exe x + v

```
Size of char: 1 byte
Size of short: 2 byte
Size of int: 4 byte
Size of long: 4 byte
Size of long long: 8 byte
Size of float: 4 byte
Size of double: 8 byte
請按任意鍵繼續 . . . |
```


格式化輸入輸出 Formatted Input / Output

printf 與 scanf

- printf 與 scanf 是 `<stdio.h>` 提供的函式。
- printf 用來將格式化字串輸出至標準輸出流 (stdout, 預設是終端)。
- scanf 用來將標準輸入流 (stdin, 預設是鍵盤) 中的資料以格式化形式讀取。

輸出函式 printf

- 單純輸出一個字串：

```
1 printf("字串");
```

- 依指定的格式輸出：

```
1 printf("格式化字串", 引數2, 引數3, ...);
```

- printf 可放入多個引數，第 1 個引數是格式化字串 (formatted string)，而第 2 個以後的引數是格式化字串中，格式指定詞 (format specifier) 所對應的數值。

格式化輸入輸出

- 當輸出的字串帶有數值時，可利用格式指定詞指定數值輸出的格式。如下方程式碼中的%d 代表 10 進位整數，%f 代表小數格式。

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 void main(void)
4 {
5     int x = 10; //宣告變數並指定初值
6     int y = 3;
7     float z;
8     z = (float)x / y; //依序將引數交給格式字串顯示
9     printf("%d / %d = %f\n", x, y, z);
10 }
```

- 輸出結果：



A screenshot of a Windows command prompt window. The title bar shows the file path "C:\Projects\test.exe". The window content displays the output of the program: "10 / 3 = 3.333333" followed by a prompt "請按任意鍵繼續 . . . |".

輸入函式 scanf

- 依格式字串指定的格式取得輸入，並依序放入變數：

1

```
scanf("格式化字串",&變數1,&變數2, ...);
```

- scanf 也可放入多個引數，第 1 個引數是格式化字串，而第 2 個以後的引數是格式化字串中，格式指定詞所對應的記憶體位址 (使用取址運算子 & 取得)。
- 使用者會輸入與第 1 個引數相同格式的字串，scanf 會將引數中格式指定詞所對應到使用者輸入的字串中的數值儲存到相對應的變數記憶體位址。

格式化輸入輸出

- 使用者依照指定的格式輸入後，scanf 將輸入的字串依指定格式解讀，並將值存入變數所在的位址中。

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 void main(void)
4 {
5     int x;
6     float y, z;
7     printf("請輸入一個整數及一個浮點數:"); //提示訊息
8     scanf("%d %f\n",&x, &y);
9     z = x / y;
10    printf("%d / %.2f = %f", x, y, z);
11 }
```

- 輸出結果：



```
C:\Projects\test.exe
請輸入一個整數及一個浮點數:3 3.14
3 / 3.14 = 0.955414
請按任意鍵繼續 . . . |
```

格式指定詞 (Format Specifier)

- 格式指定詞在格式化字串中，以% 百分比符號開頭。於 printf 指定數值的資料型別以及輸出時的格式。於 scanf 指定輸入的格式以及儲存至記憶體的资料型別 (攸關記憶體大小)。
- 格式指定詞會由 d、o、x、f、e、c、p 等轉換符號 (conversion specifier) 代表格式以及型別種類 (整數、浮點數、字元)，並且以 hh、h、l、ll 長度符號代表確切的型別，如 %hd 代表 10 進位的 short、%lld 代表 10 進位的 long long。

格式化輸入輸出

轉換符號	意義	範例
d	有號整數，10 進位	110
u	無號整數，10 進位	110
o	無號整數，8 進位	156
x、X	無號整數，16 進位 (大、小寫)	6e、6E
i	有號整數，printf 時：同 d， scanf 時：讀取 10、8、16 進位格式	
f	浮點數，小數格式	0.000314
e、E	浮點數，科學記號格式 (大、小寫)	3.141592e-004 3.141592E-004
g、G	浮點數，printf 時：由數值的科學記號指數部分大小決定 以 f 或 e、E 表示，scanf 時：讀取小數或科學記號格式	
c	字元	n
p	記憶體位址，16 進位	
%	百分比符號	%

格式指定詞整理表格

printf 的格式指定詞

	d, i	u, o, x, X	f, e, E, g, G	c	p
hh	char	unsigned char			
h	short	unsigned short			
(none)	int	unsigned int	double	char	void*
l	long	unsigned long			
ll	long long	unsigned long long			

- 其中，float 與 double 共用格式指定詞；%p 對應的 void* 代表所有型別的記憶體位址。

溢位 (overflow)

- 格式指定詞會在輸出前，將欲輸出的數值轉型成指定的型別，並依照指定的格式輸出。
- 請使用正確的格式指定詞，以免在輸出時發生溢位。

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 void main(void)
4 {
5     char c = 'A';
6     int neg = -100;
7     unsigned long long bigNumber = 1000000000000000000ull;
8     // char and int format
9     printf("int formant A : %hhd\n", c);
10    printf("char formant A: %c\n\n", c);
11    // using unsigned specifier int when negative number
12    printf("signed int -100 : %d\n", neg);
13    printf("unsigned int -100: %u\n\n", neg);
14    // using specifier too small
15    printf("unsigned LL: %llu\n", bigNumber);
16    printf("int          : %d\n", bigNumber);
17    system("pause");
18 }
```

```
C:\Projects\test.exe x + v
int formant A : 65
char formant A: A

signed int -100 : -100
unsigned int -100: 4294967196

unsigned LL: 1000000000000000000
int          : -1981284352
請按任意鍵繼續 . . . |
```

浮點數的精確度

- 我們可以在% 與浮點數的轉換符號之間加入「.」(點) 與非負整數來決定浮點數應四捨五入至小數點後第幾位 (預設為 6)。
- 另外，因為浮點數在數值的儲存上具有精準位數的限制，若輸出的位數太多，較小的位數會出現失真。

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 void main(void)
4 {
5     float value = 314.159;
6     printf("default %f: %f\n", value);
7     printf("%.2f      : %.2f\n", value);
8     printf("%.10f     : %.10f\n\n", value);
9     printf("default %e: %e\n", value);
10    printf("%.2e      : %.2e\n", value);
11    printf("%.10e     : %.10e\n\n", value);
12 }
```

```
C:\Projects\test.exe
default %f: 314.158997
%.2f      : 314.16
%.10f     : 314.1589965820

default %e: 3.141590e+002
%.2e      : 3.14e+002
%.10e     : 3.1415899658e+002
```

最小欄位寬度

- 我們可以在% 與轉換符號之間，表示精確度的「.」（點）之前加入正整數來決定欄位並靠右對齊。
- 在正整數之前加上負號 (-) 可變更為靠左對齊，加上零 (0) 可改為補零直到填滿欄位。

```
1 ~ #include <stdio.h>
2 ~ #include <stdlib.h>
3 ~ void main(void)
4 {
5     int value = 100;
6     double neg_pi = -3.14159;
7     printf("| %8d | %8.2f |\n", value, neg_pi);
8     printf("| %-8d | %-8.2f |\n", value, neg_pi);
9     printf("| %08d | %08.2f |\n", value, neg_pi);
10 }
```



100	-3.14
100	-3.14
00000100	-0003.14

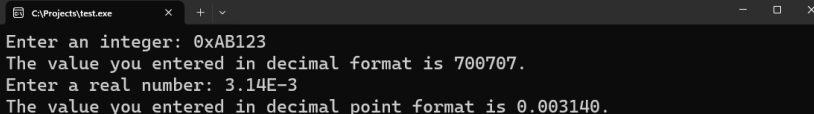
scanf 的格式指定詞

	d, i	u, o, x, X	f, e, E, g, G	c
hh	char*	unsigned char*		
h	short*	unsigned short*		
(none)	int*	unsigned int*	float*	char*
l	long*	unsigned long*	double*	
ll	long long*	unsigned long long*		

- 其中，*(星號) 代表該資料型別變數的記憶體位址。

格式化輸入輸出範例

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 void main(void)
4 {
5     int value1;
6     double value2;
7     printf("Enter an integer: ");
8     scanf("%i", &value1);
9     printf("The value you entered in decimal format is %d.\n", value1);
10    printf("Enter a real number: ");
11    scanf("%lg", &value2);
12    printf("The value you entered in decimal point format is %f.\n", value2);
13 }
```



Enter an integer: 0xAB123
The value you entered in decimal format is 700707.
Enter a real number: 3.14E-3
The value you entered in decimal point format is 0.003140.

算術運算 Arithmetic

簡單範例：兩個整數相加

- 輸入整數 a 與 b ，相加後存入 c ，最後印出。

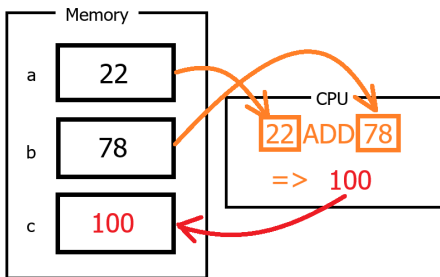
```
1  #include <stdio.h>
2  #include <stdlib.h>
3  void main(void)
4  {
5      int a, b, c;
6      printf("Enter two numbers: ");
7      scanf("%d %d", &a, &b);
8      c = a + b;
9      printf("%d + %d = %d\n", a, b, c);
10 }
```

C:\Projects\test.exe

Enter two numbers: 23 77
23 + 77 = 100

$c=a+b$; 背後的記憶體觀念

- 首先，計算機會先做 $a+b$ 的動作：a 與 b 的內容會被拷貝至 CPU 的暫存器中，在 CPU 內進行相加，得到 100。(橘色部分)
- 接著，做 $c=100$ 的動作：將 CPU 內中的 100 拷貝回變數 c 的記憶體空間。(紅色部分)



算術運算

- 除了加法 (+), C 語言也提供了減法 (-)、乘法 (*)、除法 (/)、模數 (%) 等算術運算子。
- 這些運算子亦符合數學上的先乘除後加減, 並且也可以使用小括號調整欲優先計算的部分。

運算子	運算	優先度	關聯性
()	小括號	最高	由左至右
*	乘法		
/	除法		由左至右
%	模 (餘) 數		
+	加法		由左至右
-	減法		
=	指派	最低	由右至左

除法、模數運算

- 在除法方面，因為 C 語言的算術運算具有整數的封閉性，若兩個運算元均為整數，則運算的結果亦為整數。因此，整數/整數的結果為整數，並且實際數值為數學上運算的結果取整數部分，例如：
 - 數學上 $1/2=0.5$ ，取整數部分得 0。
 - 數學上 $-7/2=-3.5$ ，取整數部分得 -3。
- 在模數 (mod，也就是取餘數) 方面，C 語言規定其運算元只能為整數型別。若 a 與 b 為整數，則 $a \% b$ 的值為 $a - (a / b) * b$ 。
- 另需注意除法與模數的第二運算元不可為 0，否則會產生執行階段錯誤。

複合指派運算子

- 在程式設計中，常會使用類似 $a=a+b$ 等將數值累加運算，可使用複合指派運算子 $+=$ 代替，即寫成 $a+=b$ 。
- 除了加法，其他算術運算子也有其對應的複合指派運算子，其執行的優先度均同等於簡單指派運算子 “=”。

運算子	範例	等價於
$+=$	$a += b;$	$a = a + b;$
$-=$	$a -= b;$	$a = a - b;$
$*=$	$a *= b;$	$a = a * b;$
$/=$	$a /= b;$	$a = a / b;$
$\%=$	$a \% = b;$	$a = a \% b;$

型別轉換

- 當數值進行運算時，在某些情況會出現資料型別轉換的狀況。以下將會介紹下列 C 語言型別轉換的類型：
 - 整數提升 (Integral promotion)
 - 算數轉換 (Arithmetic conversion)
 - 指派轉換 (Assignment conversion)
 - 明確型別轉換 (Explicit type conversion，或稱 cast)

整數提升

- C 語言在被設計時有個規定，當運算元為無論是有號或無號的 char 或 short，在運算時會提升為 int 型別，若 int 型別無法保留原本的數值，則會提升為 unsigned int 型別。
- 也就是說，若 a 與 b 為 short，則 $a + b$ 的結果為 int。這個設計的目的是為了有利於 CPU 的運算。

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 void main(void)
4 {
5     short a = 100, b = 200;
6     printf("a = %hd\tsize: %d\n", a, sizeof(a));
7     printf("b = %hd\tsize: %d\n", a, sizeof(b));
8     printf("a + b = %d\tsize: %d\n", a + b, sizeof(a + b));
9 }
```

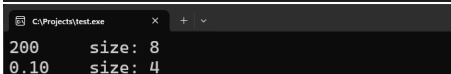
C:\Projects\test.exe

a = 100 size: 2
b = 100 size: 2
a + b = 300 size: 4

算數轉換

- 資料型別具有大小順序，由小到大依序為：char < short < int < long < long long < float < double。此順序是根據**可儲存的數值範圍**，而並非佔用的記憶體大小。
- 當不同資料型別的兩個運算元進行算術運算時，資料型別較小的一方會被轉換為**可儲存的數值範圍較大**一方的資料型別後再進行運算。

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 void main(void)
4 {
5     int iVal = 100;
6     unsigned long long ullVal = 100;
7     float fVal = 1000.0f;
8     // iVal 轉換成 unsigned long long
9     // 相加結果為 unsigned long long
10    printf("%llu\tsize: %d\n",
11           iVal + ullVal, sizeof(iVal + ullVal));
12
13    // ullVal 轉換成 float
14    // 相除結果為 float
15    printf("%.2f\tsize: %d\n",
16           ullVal / fVal, sizeof(ullVal / fVal));
17 }
```



```
C:\Projects\test.exe
200      size: 8
0.10     size: 4
```

指派轉換

- 當數值指派 (assign) 給予該數值的資料型別不同的變數時，會進行指派轉換。
- 指派轉換可能會因為資料型別的數值範圍不同而在過程中丟失數值。

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  void main(void)
4  {
5      // double 轉換成 float
6      float fVal = -1.5;
7      printf("fVal = %f\n", fVal);
8      // float 轉換成 int
9      int iVal = fVal;
10     printf("iVal = %d\n", iVal);
11     // int 轉換成 unsigned short
12     unsigned short usVal = iVal;
13     printf("usVal = %hu\n", usVal);
14 }
```

C:\Projects\test.Exe

fVal = -1.500000
iVal = -1
usVal = 65535

明確型別轉換

- 欲使數值轉型為指定的資料型別，可使用轉換運算子 “()”，語法為：

```
1 ( type-name ) cast-expression
```

- 型別轉換常用來將整數變數暫時轉換為浮點數型別，使計算除法時可得到小數部分的結果。

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 void main(void)
4 {
5     int a = 5, b = 3;
6     double tmp;
7     tmp = a / b;
8     printf("Didn't cast: %.2f\n", tmp);
9     tmp = (double)a / b;
10    printf("Casted : %.2f\n", tmp);
11 }
```

C:\Projects\test.exe

Didn't cast: 1.00
Casted : 1.67

運算子與優先度順序整理

- 運算優先度由高而低排序，以分隔線表示不同優先度：

運算子	運算	關聯性
sizeof()	取變數或型態的大小	由右至左
&	取位址	由右至左
(type)	型態轉換	由右至左
*, /, %	乘法、除法、模 (餘) 數	由左至右
+, -	加法、減法	由左至右
=, +=, -=, *=, /=, %=	簡單指派、複合指派	由右至左

<math.h> 數學函式庫

- 數學函式庫裡提供了許多函式以方便的進行浮點數的數學運算，例如開平方根、取對數、次方數、三角函數等。
- 數學函式庫的函式，其引數 (argument) 與回傳值 (return value) 均為 double，若引數放入 float 或整數型別，會進行型別轉換。

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <math.h>
4
5  int main()
6  {
7      int n = 10;
8      double e = 2.7182818284, tmp;
9      tmp = pow(e, n);
10     printf("%f ^ %d = %f\n", e, n, tmp);
11     printf("ln(%f) = %f\n", tmp, log(tmp));
12 }
```

C:\Projects\HelloWorld\math x + v

2.718282 ^ 10 = 22026.465790
ln(22026.465790) = 10.000000

<math.h> 數學函式庫

- 常用的數學函式：

函式宣告	說明
<code>double pow (double base, double exponent);</code>	次方
<code>double sqrt (double x);</code>	開平方根
<code>double log (double x);</code>	取自然對數 (ln)
<code>double log10 (double x);</code>	取對數
<code>double ceil (double x);</code>	取不小於 x 的最小整數
<code>double floor (double x);</code>	取不大於 x 的最大整數
<code>double fmod (double numer, double denom);</code>	取浮點餘數
<code>double sin(double x);</code>	取正弦值
<code>double cos (double x);</code>	取餘弦值
<code>double tan(double x);</code>	取正切值

- IEEE-754 與浮點數運算：
 - <https://ithelp.ithome.com.tw/articles/10266532>
- IEEE 754 Tutorial (Python and C) :
 - https://github.com/lovelessless99/IEEE_754_Tutorial
- IEEE-754 Floating Point Converter :
 - <https://www.h-schmidt.net/FloatConverter/IEEE754.html>

Q & A