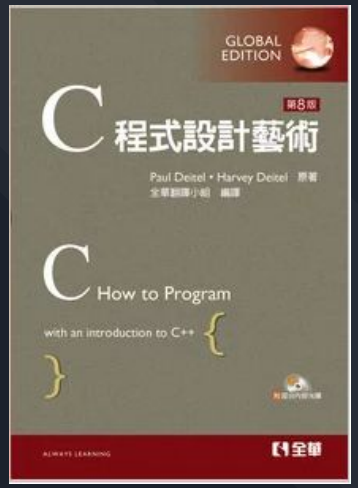
基礎程式設計
C How to Program
Simple C Programs



Yung-Chen Chou Ph.D.
iSchool, Feng Chia University
Aug. 15, 2021

一個簡單的C程式:列印一行文字

hello.c > ...

```c
// A first program in C
#include <stdio.h>

// function main() begin program execution
int main(void){
    printf("Hello world!!\n");
    return 0;
}// end function main()
```

// single line comment

#include Preprocessor Directive

An Output Statement

註解 (Comment)

- 雖然這個程式非常的簡單，不過它包含了C的幾個重要的特性

```
1  // A first program in C
```

- 以 // 開頭的兩行表示這是**註解 (comment)**。
- 你也可以使用 /* */ 多行註解 (multi-line comments)，從 /* 開始且以 */ 結尾之間的任何內容都被會視為註解

#include 前置處理器指令

```
2   #include <stdio.h>
```

- C 前置處理器 （**C preprocessor**） 指令
- 在程式編譯之前，前置處理器會處理以 **#** 開頭的每一行
- 第 2 行告訴前置處理器將**標準輸入／輸出標頭檔**（**standard input/output header**）（**<stdio.h>**）引入到程式裡面

空行與空白

- 第 3 行只有一個空行
- 使用空行、空格及定位字元 （tabs） 使程式較容易閱讀
- 這些字元統稱為空白 (white space)，空白通常會被編譯器略過

main 函數

```
5  int main(void){
6      printf("Hello world!!\n");
7      return 0;
8  }// end function main()
```

- 是每個C程式都有的部分
- 在 main之後的小括號表示main是程式的一個建構區塊，稱為函式 (function)
- 每個函式本體 (body) 以左大括號 (left brace) { (第5行) 開始。右大括號 (right brace) } (第8行) 表示每個函式的結束位置。這兩個括號以及之間的程式稱為一個區塊 (block)

輸出敘述

```
6       printf("Hello world!!\n");
```

- 指示電腦執行一個**動作** (action)，將雙引號內的字串 (string) 顯示在螢幕上。
- 字串有時稱做**字元字串** (character string)、**訊息** (message)，或是**字面常數** (literal)
- 這一整行，包括 printf (字母 **f** 表示「**格式化的**」)、**括號**、括號裡的**引數** (argument) 及**分號** (semicolon 「；」) 等，稱為一道**敘述式** (statement)。
- 每行敘述須以**分號** (也叫**敘述結束符號**，statement terminator) 做結束

跳脫序列

- 要特別注意的是，字元 **\n** 並不會顯示在螢幕上
- 反斜線符號 （****） 稱為跳脫字元 （**escape character**）
  它表示 `printf` 應印出一些特殊的字元
- 跳脫序列 **\n** 表示新增一行 （**newline**）

Escape sequence	Description
\n	Newline. Position the cursor at the beginning of the next line.
\t	Horizontal tab. Move the cursor to the next tab stop.
\a	Alert. Produces a sound or visible alert without changing the current cursor position.
\\	Backslash. Insert a backslash character in a string.
\"	Double quote. Insert a double-quote character in a string.

連結器與可執行程式

- 標準函式庫的函式（如 **printf** 或 **scanf**） 並不是C語言的一部分
- 編譯器無法發現他們是否拼字錯誤
- 當編譯器在編譯 printf 敘述式時，它只是在目的碼中空出一塊空間來「呼叫」函式庫函式
- 編譯器不知道函式庫函式在哪裡—但是連結器知道
- 一直要到連結器進行連結的時候，才會找出函式庫函式的位置，然後在目的碼中插入對函式庫函式正確的呼叫
- 此時，目的碼便是完整且可執行的了
- 因此，已經連結的程式通常稱為可執行（**executable**）的程式

Sample Code: 兩整數相加

- 使用標準函式庫scanf函式，來讀進使用者輸入的兩個整數，然後計算這兩個值的和，再以 printf 將結果印出來

```
int v1; // first number to be entered by user
int v2; // second number to be entered by user
int sum; // variable in which sum will be stored
```

- 都是定義 (**definitions**)
- v1、v2 和 sum 是變數 (**variables**) 名稱
- 變數就是電腦記憶體中，存放數值供程式使用的位置。

```
int v1, v2, sum;
```

你也可以把多個同資料型態的變數一起宣告。

C demo1.c > ...

```c
// addition program
#include <stdio.h>

// function main begins program execution
int main(void){
    int v1; // first number to be entered by user
    int v2; // second number to be entered by user

    printf( "Enter first integer\n" ); // prompt
    scanf( "%d", &v1 ); // read an integer

    printf( "Enter second integer\n" ); // prompt
    scanf( "%d", &v2 ); // read an integer

    int sum; // variable in which sum will be stored
    sum = v1 + v2; // assign total to sum

    printf( "Sum is %d\n", sum ); // print sum
}  // end function main
```

```
➜  C_Workspace gcc demo1.c -o demo1
➜  C_Workspace ./demo1
Enter first integer
8
Enter second integer
9
Sum is 17
```

Image source:
https://ahmadnaser.com/top-5-computer-programmers-in-the-world/

識別字和大小寫區別

- C 語言當中變數的名稱可以是任意合法的識別字（**identifier**）
- 識別字是由字母、數字和底線 （ **_** ） 組成的一連串字元，但第一個字元不可以是數字
- 識別字的長度沒有限制，不過 C標準 中規定編譯器只需辨認前31個字元
- C會區分大小寫 （**case sensitive**），所以 **a1** 與 **A1**是不同的識別字

語法

- 提示訊息

```
printf( "Enter first integer\n" ); // prompt
```

  - 字面印出 Enter first integer, 並將游標移至下一行的開頭
  - 這個訊息就叫作**提示** (**prompt**), 因為它指示使用者進行某特定動作
- Scanf函式與格式化輸入

```
scanf( "%d", &v1 ); // read an integer
```

  - 使用scanf從使用者處讀取一個數值。scanf函式由標準輸入 (通常是鍵盤) 讀取輸入值

指定敘述句 (assignment statement)

```
sum = v1 + v2; // assign total to sum
```

- 計算了變數 **v1** 和 **v2** 的和，並使用**指定運算子 =**（**assignment operator**）將結果設定給變數 **sum**
- 大部分的計算都是用**指定敘述式**來執行
- 運算子 **=** 和 **+** 稱為**二元運算子**（**binary operator**），因為它們需要有**二個運算元**（**operands**）
- 使用格式控制字串列印

```
printf( "Sum is %d\n", sum ); // print sum
```

  - 呼叫函式 printf，在螢幕上印出字面常數 Sum is 以及 變數 sum 的值

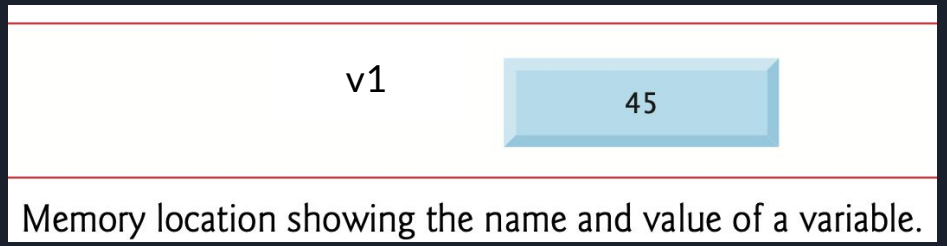
在 printf 敘述式裡執行計算

- 可以將計算的執行放在 `printf` 敘述式裡
- 上面兩道敘述式可以合併成

```
printf( "Sum is %d\n", v1 + v2 ); // print sum
```

- 第 19 行的右大括號 （ } ） 表示 main函式 結束

記憶體觀念



Memory location showing the name and value of a variable.

- 像 **v1**、**v2**和 **sum** 這樣的變數名稱都會對應到電腦裡的記憶體位置 （**locations**）
- 每個**變數**都具有一個名稱 （**name**）、一個型別 （type）、及一個值 （**value**）

```
scanf( "%d", &v1 ); // read an integer
```

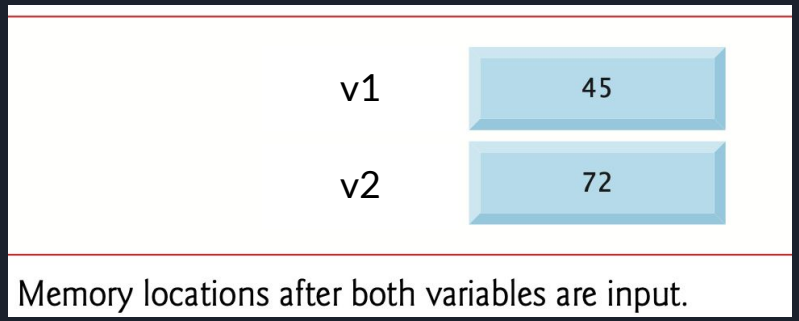
- 執行時，使用者輸入的數值會被放在 **v1** 所被指定的記憶體位置
- 假設使用者輸入數字45作為變數 v1 的值

記憶體觀念

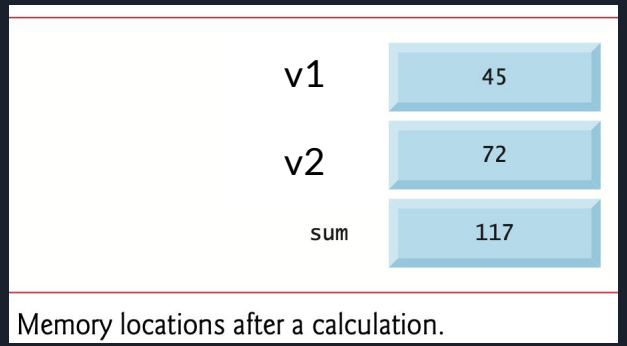
```
scanf( "%d", &v2 ); // read an integer
```

- 執行時，假設使用者輸入72，這個數值就會存到v2的位置
- 在程式讀取 v1 和 v2 的值之後，它會將這兩個值相加，然後將其和放到變數 sum

```
sum = v1 + v2; // assign total to sum
```



Memory locations after both variables are input.



Memory locations after a calculation.

C的算術運算

C operation	Arithmetic operator	Algebraic expression	C expression
Addition	+	$f + 7$	`f + 7`
Subtraction	-	$p - c$	`p - c`
Multiplication	*	bm	`b * m`
Division	/	x / y or $\frac{x}{y}$ or $x \div y$	`x / y`
Remainder	%	$r \bmod s$	`r % s`

Fig. 2.9 | Arithmetic operators.

整數除法與模數運算子

- 整數除法（Integer division）得到的結果會是個整數
  - 如 7/4 會等於 1，而 17/5 會等於 3。
- C 還提供了模數運算子 %（remainder operator），它的運算結果是整數相除後的餘數
- 模數運算子是個整數運算子，它的運算元必須是整數
- 運算式 x % y 會產生 x 除以 y 的餘數。因此 7%4 等於 3，而 17%5 等於 2

橫行形式的算術運算式

- C 的算術運算式須以橫行形式 (**straight-line form**) 輸入電腦，方便將程式輸入電腦
- 因此，像是「a除以b」的運算式就必須寫成**a/b**，如此所有的運算子和運算元都會排成橫行
- 用小括號將子運算式分群
  - 在 C運算式中，小括號的用法跟代數運算的用法很像
  - 例如，要將 **a** 乘以 **b + c** 的值，就寫成 **a * (b + c)**

運算子優先順序規則

- 在同一對小括號中的運算子先進行計算
- 小括號具有「**最高優先權**」
- 若為**巢狀** （**nested**）， 或稱**嵌入** （**embedded**） 的小括號 （**parentheses**）， 如

```
( ( a + b ) + c )
```

- 最裡面那對小括號中的運算子會先計算
- 接著會處理乘法、除法和模數運算
- 若運算式有多個乘法、除法和模數運算， 則從左到右進行計算
- 因此乘法、除法和模數運算具有相同的優先權層級

運算子優先順序規則

- 接下來進行加和減的運算
- 若運算式有多個加減法運算，則從左到右進行計算
- 加和減的優先順序等級是相同的，但是優先次序低於乘法、除法以及模數運算子
- 最後處理的是賦值運算子

Operator(s)	Operation(s)	Order of evaluation (precedence)
( )	Parentheses	Evaluated first. If the parentheses are nested, the expression in the *innermost* pair is evaluated first. If there are several pairs of parentheses "on the same level" (i.e., not nested), they're evaluated left to right.
* / %	Multiplication Division Remainder	Evaluated second. If there are several, they're evaluated left to right.
+ -	Addition Subtraction	Evaluated third. If there are several, they're evaluated left to right.
=	Assignment	Evaluated last.

Fig. 2.10 | Precedence of arithmetic operators.

簡單的代數與C運算式

- 下面的算式計算了5個數的算術平均數：

Algebra: $m = \frac{a+b+c+d+e}{5}$

```
C:      m = ( a + b + c + d + e ) / 5;
```

- 此處需要用小括號將加法圍起來，假如遺漏了其中的括號，便會得到`a + b + c + d + e/5`，運算式就算錯了

$$a+b+c+d+\frac{e}{5}$$

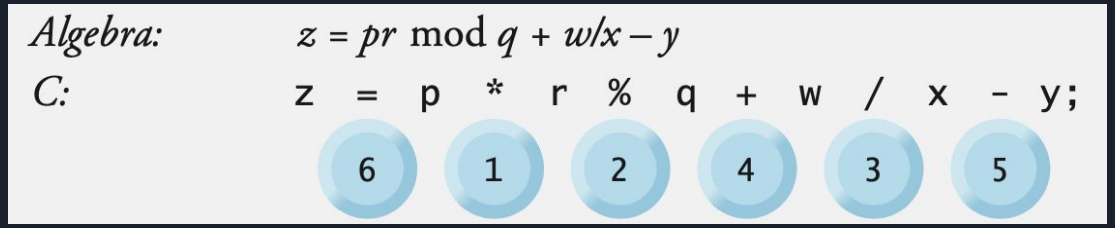
Algebra: $y = mx + b$

```
C:      y = m * x + b;
```

- 右方的算式是直線的方程式：
- 這裡不需要小括號。程式會先計算乘法運算，因為乘法的優先權高於加法的優先權

簡單的代數與C運算式

- 以下算式包括模數運算（%）、乘法、除法、加法、減法和賦值等運算：



- 在運算式底下圈起來的數字，表示C執行這些運算子的順序
- 乘法、模數運算和除法首先會依據從左到右的順序進行計算，因為它們的優先權比加法和減法更高
- 接著才計算加法和減法。它們也是從左到右進行計算
- 最後，將計算結果賦值於變數z

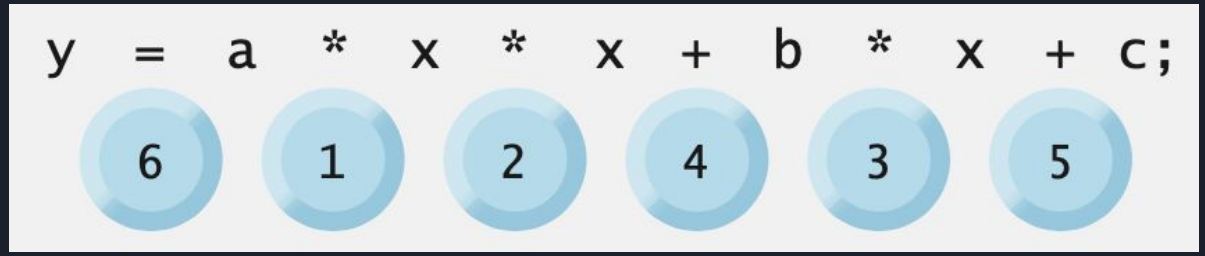
簡單的代數與C運算式

- 並不是擁有一對括號以上的運算式均含有多層的括號
- 例如，底下的運算式就不包含多層括號，這兩對括號是屬於「同層級的」

```
a * (b + c) + c * (d + e)
```

二次多項式計算

- C如何計算一個二次多項式



- 在運算式底下圈起來的數字，表示C執行這些運算的順序
- 跟代數運算一樣，在運算式中加入非必要的小括號，可讓運算式的計算順序更清楚
- 這些括號稱為**多餘括號**（**redundant parentheses**）
- 例如，前述的敘述可加上以下的小括號：

```
y = ( a * x * x ) + ( b * x ) + c;
```

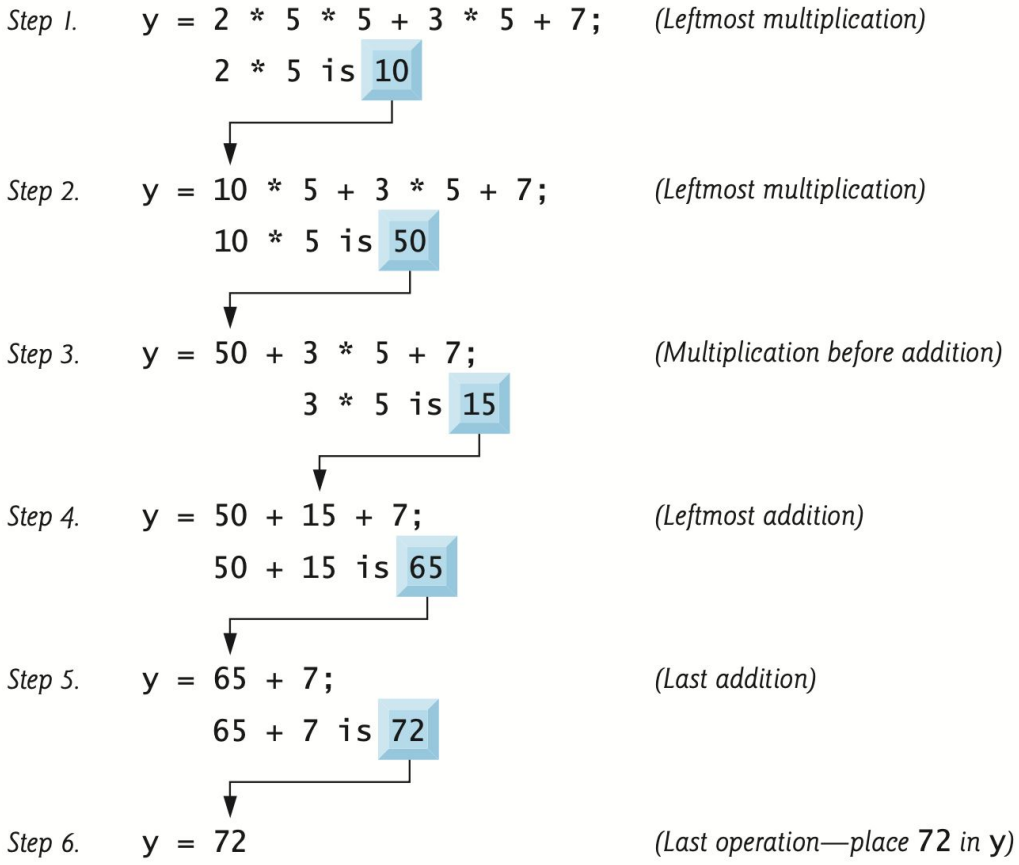


Fig. 2.11 | Order in which a second-degree polynomial is evaluated.

判斷:等號運算子和關係運算子

(Decision Making: Equality and Relational Operators)

Algebraic equality or relational operator	C equality or relational operator	Example of C condition	Meaning of C condition
Relational operators			
>	>	x > y	x is greater than y
<	<	x < y	x is less than y
≥	>=	x >= y	x is greater than or equal to y
≤	<=	x <= y	x is less than or equal to y
Equality operators			
=	==	x == y	x is equal to y
≠	!=	x != y	x is not equal to y

Fig. 2.12 | Equality and relational operators.

判斷:等號運算子和關係運算子

(Decision Making: Equality and Relational Operators)

C demo3.c > ...

```c
// Using if statements, relational
// operators, and equality operators.
#include <stdio.h>

// function main begins program execution
int main( void ){
    printf( "Enter two integers, and I will tell you\n" );
    printf( "the relationships they satisfy: " );

    int num1; // first number to be read from user
    int num2; // second number to be read from user

    scanf( "%d %d", &num1, &num2 ); // read two integers 16
    if ( num1 == num2 ) {
        printf( "%d is equal to %d\n", num1, num2 );
    } // end if
```

判斷:等號運算子和關係運算子

(Decision Making: Equality and Relational Operators)

C demo3.c > ...

```c
    if(num1!=num2){
        printf( "%d is not equal to %d\n", num1, num2 );
    }//endif
    if(num1<num2){
        printf( "%d is less than %d\n", num1, num2 );
    }//endif
    if(num1>num2){
        printf( "%d is greater than %d\n", num1, num2 );
    }//endif
    if(num1<=num2){
        printf( "%d is less than or equal to %d\n", num1, num2 );
    }//endif
    if(num1>=num2){
        printf( "%d is greater than or equal to %d\n", num1, num2 );
    }//endif
} // end function main
```

比對兩個變數**num1**和**num2**的值是否相等

判斷:等號運算子和關係運算子

(Decision Making: Equality and Relational Operators)

```
Enter two integers, and I will tell you
the relationships they satisfy: 3 7
3 is not equal to 7
3 is less than 7
3 is less than or equal to 7
```

```
Enter two integers, and I will tell you
the relationships they satisfy: 22 12
22 is not equal to 12
22 is greater than 12
22 is greater than or equal to 12
```

```
Enter two integers, and I will tell you
the relationships they satisfy: 7 7
7 is equal to 7
7 is less than or equal to 7
7 is greater than or equal to 7
```

判斷:等號運算子和關係運算子
(Decision Making: Equality and Relational Operators)

- 由高至低列出運算子的運算優先順序。運算子的優先權是從上到下遞減。注意：等號也是個運算子。除了指定運算子 = 之外，其他所有運算子的結合性都是由左至右。指定運算子 (=) 是從右至左結合。

Operators	Associativity
()	left to right
* / %	left to right
+ -	left to right
< <= > >=	left to right
== !=	left to right
=	right to left

Fig. 2.14 | Precedence and associativity of the operators discussed so far.

C語言的關鍵字 (keywords) 或保留字 (reserved words)

- 這些字對C編譯器來說都具有特殊的意義，因此程式設計師必須小心，不可以將他們用來做為識別字（如變數名稱）

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

Keywords added in C99 standard

_Bool _Complex _Imaginary inline restrict

Keywords added in C11 standard

_Alignas _Alignof _Atomic _Generic _Noreturn _Static_assert _Thread_local

Fig. 2.15 | C's keywords.

安全程式設計 (Secure C Programming)

- C安全程式標準 (The CERT C Secure coding Standard) ，其中提出了一些指導原則，幫助你免於在程式實作時，使系統暴露在攻擊之下
- **避免單一參數的 printf 函式**
  - 指導原則的其中一項，是避免在使用 **printf** 時，只用一個字串當作參數
  - 如果你要顯示一個以換行 (**newline**) 結束的字串，**請使用 puts** 函式(**puts function**)，它將參數裡的字串後加上換行字元並顯示

```
printf( "Welcome to C!\n" );
```

```
puts( "Welcome to C!" );
```

安全程式設計 (Secure C Programming)

- 我們字串後面沒有\n ， 因為 puts 會幫自動加上
- 如果你要顯示一個字串且不使用換行符號結尾， 請在使用 printf 時帶入兩個參數
- 轉換指定詞（conversion specifier） %s 可用於顯示一個字串

```
printf( "Welcome " );
```

```
printf( "%s", "Welcome " );
```