我要學會 C 語言

我要學會 C 語言(一)：Hello World!

如何用 Dev-C++ 寫出一個基本的小程式？

用 C 語言寫出一個 Hello, World!

好的，正文開始。

剛開始學習一個程式語言，我們都會先試著在螢幕上輸出 Hello, World! 這串字，所以一起來看看這個範例程式碼吧~

#include <stdio.h>

int main(void){

 printf("Hello, World!\n");

 return 0;

}

接著就是解釋這段程式碼，小小的五行程式碼到底蘊含了多少奧秘呢？

首先第一行是用來引用標頭檔，告訴編譯器我們需要用 stdio.h 這個標頭檔。標頭檔裡面記載了不同的函式(function，可以暫時把函式當作指令)的用法，一旦你用了某個函式，你就必須加入那個函式的標頭檔，例如這個程式碼會引用 stdio.h 就是因為用了 printf 這個函式。

#include <stdio.h>

再來是主程式 main 的部分。main 為程式的進入點，也就是說一支程式在執行時會先找 main 開始執行。而小括號裡面的 void 代表這個函式沒有傳入值，不需要傳參數進去。

int main(void){ }

printf 是 print 和 formatted 的縮寫，可以把東西以特定格式輸出到螢幕上。其中的 \n 是換行的意思。

printf("Hello, World!\n");

最後以 return 來結束程式，一般來說程式如果正常結束會回傳 0。

return 0;

在第一篇文章中，對 Hello, World! 有這樣的認識就足夠了。至於標頭檔在做什麼？函式該怎麼用？\n為什麼代表換行？有沒有其他的可以用？這些問題在後面的章節會再仔細講解。

剩下的篇幅先來討論一些剛開始學程式語言比較重要的事情。

大括號該怎麼放？大括號一定是倆倆對應的！

一個函數的開始，或是用到 if、for、while、switch 等等選擇結構或迴圈結構，通常會搭配大括號來使用。大括號代表一個區段，像上面的範例程式，printf 和 return 都在 main 這個函式裡面。

至於大括號放法通常有兩種，一種是左大括號緊接在那一行的最後面：

int main(void){

}

另外一種是左大括號放在新一行的最前面：

int main(void)

{

}

兩種做法都可以，習慣就好，這兩派的人都很多。

另外，由於大括號代表一個區段，所以大括號一定是有一個左大括號就有一個右大括號，如果怕很多會亂掉，可以習慣先打一個區段前先把 { } 都打出來再進去打裡面的程式碼。

縮排很重要，請統一使用 TAB 鍵或四個空格

縮排是什麼？就是該空白的地方要好好空白。而什麼時候應該縮排？在一層新的區段裡面就應該縮排一次。

有縮排的看起來是這樣



沒縮排的話看起來像這樣：



還看不出差別？那是因為現在程式碼只有五行，等到程式碼一多，看起來就會差很多。(當然這不是C啦)

int main 裡面的 void 可不可以省略？

如果要細說的話，那個 void 還是有它的意思。不過如果你真的不知道 void 在幹嘛，又很懶得打的話，把它省略是沒關係的。省略掉就會變成這樣：

int main(){

}

最後，寫程式很困難嗎？

其實寫程式沒那麼困難？只要搞懂每一行在做什麼，試著去思考電腦怎麼計算、執行那些過程，尤其是迴圈的部分，我會試著把迴圈的概念講清楚，跟著它的脈絡走，你就會覺得寫程式不是一件痛苦的事情，而是一種能夠把你所想的流程都實現的工具。

噢還有，把 C 紮實的學好，到後面修 Java 還是 PHP 都會很輕鬆 (#)

(二)：有小數點的數不是整數之資料型態與變數

對於 x^2+2x+1=0 這個方程式你想到什麼？一個方程式裡面最神奇的地方就是它有一個或多個未知數，有了未知數你就可以方便計算很多東西；而且對於一個函數 f(x) = x^2+2x+1，你還可以代入不同的 x 算出不同的值。寫程式也是一樣，你總不希望你的電腦沒有辦法做代數運算，也不能輸入其他的值讓他算出不同東西吧？

於是有了變數。有了變數，你就可以暫時儲存一些資料在記憶體裡。

註1：^ 是次方的意思，10^2 代表 10 的平方、5^3 代表 5 的立方。

註2：未知數和變數在英文都稱為 variable。

只不過，在電腦裡面，變數分成好幾種，該種類的變數只能存放該種類的資料，例如一個整數的變數不能放入「9.2」，也不能放入「Noob」。

資料型態

於是有了資料型態，在 C 語言中資料型態大概有這幾種，後面是它的範圍(range)：

• char：-128 ~ 127

• short：-32768 ~ 32767

• int：-2147483648 ~ 2147483647

• unsigned：0 ~ 4294967296

• long：-2147483648 ~ 2147483647

• long long：-9223372036854775808 ~ 9223372036854775807

• float：1.17549×10^-38 ~ 3.40282×10^38

• double：2.22507×10^-308 ~ 1.79769×10^308

• long double：3.3621×10^-4932 ~ 1.18973×10^4932

其中粗體的資料型態是比較常用到的。

如果上面那個你覺得你看了霧煞煞，那你只要先記住幾件事情：

• char 叫做字元，可以用來存放符號、英文字母和數字 (詳見ASCII Table)

• int 是 integer 的縮寫，叫做整數。大概也是最常用到的，要放整數先選我就對了！

• long long 叫長長整數，是一種高級整數，當數字大過 2147483647 的時候就要選我~

• float 叫浮點數，要用小數點先選我，我大概可以存小數點以下 7 位數左右

• double 叫倍精度浮點數，是一種高級浮點數，可以存小數點以下 15 位數左右

其他剛開始大概用不太到。

不過這個時候又有問題出現了，你可能會想，既然 long long 那麼強大，可以放個九百京左右，幹嘛還要有 int 和 short int 的存在？那是因為，不同的資料型態是有不同的大小(size)的。當你使用了越大的資料型態，你就會在記憶體(RAM)佔據更多的空間。

資料型態的大小

這個部分沒有那麼重要，不過這裡可以告訴你為什麼不統一用 long long 取代 int，還有為什麼那些資料型態的範圍到底是哪裡來的。

先來回顧一下排列組合吧。電腦是由電路組成，所以只有通電與不通電兩種情況。(這也解釋了為什麼電腦以二進位計算)

• 有 1 條電路的情況下，可以組合出 2 種情況

• 有 2 條電路的情況下，可以組合出 4 種情況

• 有 3 條電路的情況下，可以組合出 8 種情況

• 有 n 條電路的情況下，可以組合出 2^n 種情況

在電腦裡面，我們把這種電路稱為位元(bit)，8 個位元稱作一個位元組(byte)。

一般來說：

• short 占了 2 個位元組

• int 占了 4 個位元組

• long long 占了 8 個位元組

• char 占了 1 個位元組

所以 int 占了 4 個位元組 = 32 個 bit，也就有 2^32 = 4294967296 種情況，所以可以存放 4294967296 個數字，其中一半要放負整數(-2147483648~-1)、另外一半要放零和正整數(0~2147483647)，這就是為什麼一個占了 n 個位元的變數，範圍是 −2n−1−2n−1 ~ 2n−1−12n−1−1 囉！

宣告變數

終於把資料型態大概說完了，接下來就是變數。剛剛說變數可以讓你儲存一些資料在記憶體裡，可是記憶體就是那麼大，程式怎麼知道要把變數放在記憶體的哪裡？所以我們使用每個變數之前都必須先宣告，一來是幫這個變數命名，二來是讓程式去告訴系統：請給我一個空間來存放資料。

再來，變數命名。雖然變數你要怎麼命名，只要你記得就可以，但是麻煩盡量幫變數想一個有關的名字，例如要存放香蕉數量的變數可以取名叫 banana\_count、存放年齡的變數可以叫 age，不然別人看了你的程式碼會不知道你在寫什麼，這個別人包括一個月後的自己。

那，讓我們來宣告一個變數，用來存放你的分數吧。

int score;

什麼？就這樣？

沒錯，宣告一個變數只需要這樣子就可以了。宣告完變數後才可以使用變數，例如給它一個值。

score = 80;

再強調一次，變數一定要先宣告才能使用。如果剛剛沒有宣告的話，這一行就會出現錯誤。

另外，數學上的 = 和 C 語言裡面的 =，意思不太一樣。數學上的 = 是指等號左右兩邊的值相等；但在 C 語言(及絕大部分程式語言裡面)，= 的意思是把等號右邊的值存放(assign)到等號左邊的變數裡，有←的味道存在。例如下面這段程式碼就是錯的：

int score;

80 = score;

而要在宣告變數的時候就給定一個值也是可以的，例如：

int score = 80;

再來宣告個油價看看，油價通常有小數點，所以應該使用浮點數：

float gas\_price = 27.3;

然而宣告完也給定初值的變數也可以再給別的值，這是沒有問題的。

float gas\_price = 27.3;

gas\_price = 28.4;

如果給值的時候給了一個超過範圍的，例如一個 int 給了 2147483648，就會發生溢位(overflow)，溢位會怎樣？你可能會得到一個非預期的值，或是程式炸掉、電腦炸掉、甚至導致程式被攻擊都是有可能的。

好啦資料型態與變數就先到這邊，不過變數可不只有這樣給值而已，它還有很多事情可以做的！下一篇來講變數的運算~

(三)：加加減減乘除的變數運算

第三篇要講的是運算，有了變數以後，總該拿幾個變數來運算看看吧？

C 語言除了有基本的五則運算以外，還有一些像是邏輯運算、遞增遞減運算、指定運算……等等。

五則運算

咦？不是四則運算嗎？怎麼變五則運算？

除了加(+)、減(-)、乘(\*)、除(/)以外，還有一個用來取餘數用的運算子 %，用法是 a%b 會得到 a 除以 b 的餘數。例如：

13 % 6 = 1

25 % 5 = 0

93 % 7 = 2

那麼，變數和變數之間的運算，在五則運算裡面比較簡單。例如：

#include <stdio.h>

int main(){

 int x=15, y=6;

 int a = x+y;

 int b = x-y;

 int c = x\*y;

 int d = x/y;

 int e = x%y;

 printf("%d\n", a);

 printf("%d\n", b);

 printf("%d\n", c);

 printf("%d\n", d);

 printf("%d\n", e);

 return 0;

}

輸出結果會是：

21

9

90

2

3

註：這裡使用了輸出用的 printf 函式，等到輸出入的時候會再詳細說明。

以上看起來好像沒有什麼問題，但是，咦？ 15/6 怎麼會是 2？

因為在 C 語言裡面，整數(int)運算過程會無條件捨去小數點以下的部分，所以 2.5 會被捨去，變成 2。

如果要正確的顯示小數點，應該用浮點數來運算。這個部分到下一篇也會再說一次。

遞增、遞減運算子

遞增運算子(++)和遞減運算子(–)可是 C/C++ 獨有的運算子，它們的功能就是：用來+1或用來-1，用法就是：a++、++a 或 a–、–a。由於遞增和遞減運算子的用法相同，以下皆以遞增運算子來做示範。

a++ 的結果和 a = a + 1 是一樣的，例如：

#include <stdio.h>

int main(){

 int a=7, b=7;

 a++;

 b++;

 printf("%d\n", a);

 printf("%d\n", b);

}

輸出結果無庸置疑的會是 8 和 8。

不過下面這段程式碼：

#include <stdio.h>

int main(){

 int a=7, b=7;

 printf("%d", ++a);

 printf("%d", b++);

}

輸出結果卻是 8 和 7。因為第 5 行會先把 a 做遞增後再輸出，第 6 行是先輸出 b 再做遞增，這是使用遞增/遞減運算子比較需要注意的部分。

如果要細講的話：

int a=3, int num;

num = a++;

num = ++a;

第 2 行可以拆成 num = a; a = a + 1;；第 3 行可以拆成 a = a + 1; num = a;。

另外，遞增遞減運算子只有加減法，並沒有 a\*\* 或 a// 之類的東西可以用唷，當然更沒有 a%% xD。

指定運算子

指定運算子則是大部分的運算子都有，有 +=、-=、=、/=、%=\*。可以讓撰寫程式的時候變得比較方便，不用再多寫一次變數名稱，例如：

a += 3 等同a = a+3`

a -= 3 等同 a = a-3

a \*= 3 等同 a = a\*3

a /= 3 等同 a = a/3

a %= 3 等同 a = a%3

當然還有別的，例如\*\*&=、|=、<<=\*\*、……等等。但是這些運算並不常見，在這裡省略不談。

那直接看例子吧：

#include <stdio.h>

int main(){

 int a=7, b=7, c=7, d=7, g=7;

 a += 3;

 b -= 3;

 c \*= 3;

 d /= 3;

 e %= 3;

 printf("%d\n", a);

 printf("%d\n", b);

 printf("%d\n", c);

 printf("%d\n", d);

 printf("%d\n", e);

 return 0;

}

輸出結果：

10

4

21

2

1

那變數運算的部分大概到這邊，這些運算子看起來很簡單，不過還是要試著練習一次才有效果。如果在設計程式的時候常常忘記某件事情的話，沒關係，總是要錯個幾次才記得起來 XD

(四)：輸入與輸出好好玩

要讓程式和使用者互動，不透過輸入和輸出怎麼行呢？

輸入和輸出其實有很多種，例如語音輸入、發出聲音、震動，甚至是讀取、寫入檔案(應該後面的章節會解釋)。不過一開始，我們先從最基本的開始玩：螢幕的輸入和輸出。

在 C 語言裡面，螢幕輸入和輸出是一種標準的輸入輸出：Standard Input and Output，我們幾乎每次都會引用的標頭檔 stdio.h 的 stdio 就是它的縮寫。

螢幕輸出

還記得 Hello World! 的時候，我們用了 printf 函式嗎？沒錯，printf 函式就是用來輸出的。基本用法：

printf("要輸出的文字");

然後你就會得到要輸出的文字了。

為什麼要用 " " 包起來？在程式語言來說，用雙引號包起來的代表字串、用單引號包起來的代表字元，字串就是多個字元的意思；而 printf 的功能其實是輸出字串。別擔心，字串的概念會在之後的章節再講一次。

不過 printf 沒那麼簡單，還記得我說過 printf 後面的 f 代表 formatted 嗎？printf 裡面還是可以填入各種格式的，比方說跳脫字元，例如 \n 代表換行，所以每輸出一個 \n 就會換行一次，例如：

printf("這是第一行\n\n這是第四行");

你會得到：

這是第一行

這是第四行

常見的跳脫字元有：

• \n：用來換行

• \t：用來縮排(等同於輸出一個 tab)

• \’：用來輸出一個單引號

• \”：用來輸出一個雙引號

• \：用來輸出一個反斜線

其他其實平常用不太到，就不多說以免造成困擾。而為了避免字元裡面含有單引號、字串裡面含有雙引號，或是含有反斜線，所以這三個會特別弄成跳脫字元。如果不懂我在說什麼，可以想想看如果你不用跳脫字元要怎麼用 printf 輸出一個雙引號，你就懂了 XD

另外，在變數運算的章節也提過 %d，%d 的用途就是來輸出變數的內容。例如：

#include <stdio.h>

int main(){

 int a=4;

 printf("%d", a);

 return 0;

}

你會得到一個 4。

如果要輸出多個變數的話，可以這麼做：

#include <stdio.h>

int main(){

 int a=4, b=7;

 printf("%d\n%d", a, b);

 return 0;

}

你會得到：

4

7

「噢噢，所以只要前面用 %d，後面用逗點跟變數名稱就好了嗎？」

如果這麼方便，就不用花一篇文章講資料型態了(笑)。要用 printf 的話，會因為不同的資料類型，前面要放的東西不一樣，常見的有：

• %d：顯示整數(十進位，d 是 decimal 的意思)

• %o：顯示整數(八進位，octal)

• %x：顯示整數(十六進位，heximal)

• %c：顯示字元

• %f：顯示浮點數

• %e：顯示浮點數(以科學記號方式表示)

• %lld：顯示長整數

• %s：顯示字串

• %%：顯示一個 % 符號

下面這個例子示範了如何顯示字元、整數、浮點數，以及一個整數的八進位和十六進位如何表示。

#include <stdio.h>

int main(){

 int a=23;

 float b=9.2;

 char c='A';

 printf("%d\n", a);

 printf("%o\n", a);

 printf("%x\n", a);

 printf("%f\n", b);

 printf("%c\n", c);

 return 0;

}

輸出結果：

23

27

17

9.2

C

另外，printf 還可以調整一些輸出的細節。

如果要指定小數點輸出位數，可使用 %.2f (輸出到小數點第二位)。

如果要指定輸出的總位數(不含小數點)，可使用 %3d (總共輸出三位，不足會以空白補齊)。

也可以指定輸出總位數，並在前面補零，使用 %08d (總共輸出八位，不足會以零補齊)。

總位數和小數點也能搭配使用，例如 %5.3f (小數點以下三位，小數點以上兩位)。

螢幕輸入

螢幕輸入的時候使用 scanf 函式。例如：

#include <stdio.h>

int main(){

 int input;

 scanf("%d", &input);

 return 0;

}

就可以把使用者輸入的值存到變數 input 裡面。

scanf 也是 scan 和 formatted 的縮寫，所以 scanf 的用法和 printf 幾乎一模一樣。但比較需要注意的是：

• scanf 裡面絕對不要包含 \n

• scanf 後面不是直接放變數，而是要加上一個 &

這兩點都很重要，因為做錯這兩點在編譯的時候不會出錯，但在執行上卻不是自己想要的結果，所以常常會因為做錯這兩件事情而找錯誤找半天。不如剛開始就養成好習慣，不要犯這兩個錯誤！

至於為什麼 scanf 要搭配 &，等到後面指標的章節會再詳細講解。

如果要一次輸入多個變數的話，通常中間會以空白分隔：

#include <stdio.h>

int main(){

 int a;

 int b;

 float c;

 scanf("%d %d %f", &a, &b, &c);

 return 0;,

}

這樣就能把多個值放到多個變數裡面，記得整數要搭配 %d、浮點數要搭配 %f …… 等等。

輸出與輸入

學會輸出和輸入後，就可以來做一點互動的事情啦！試著從零開始寫個簡易乘法計算機吧！

首先，我們一個程式裡一定會有主程式，所以先把 main 函式的部分補齊吧：

#include <stdio.h>

int main(){

 return 0;,

}

接著乘法需要一個被乘數、一個乘數，所以需要兩個變數來存放。至於資料型態呢？可以先使用整數就好。

#include <stdio.h>

int main(){

 int multiplicand, multiplier;

 return 0;,

}

然後要讓使用者輸入兩個整數，放到兩個變數 multiplicand、multiplier 裡面，使用 scanf 函式。

#include <stdio.h>

int main(){

 int multiplicand, multiplier;

 scanf("%d %d", &multiplicand, &multiplier);

 return 0;,

}

把值存到變數以後，接下來就是把兩個值乘起來並輸出囉。

#include <stdio.h>

int main(){

 int multiplicand, multiplier;

 scanf("%d %d", &multiplicand, &multiplier);

 printf("%d", multiplicand \* multiplier);

 return 0;,

}

這樣看起來已經沒有什麼問題了。不過還有沒有辦法讓它更完善呢？如果你真的有跟著編譯的話，你會發現，程式一打開是空白的，會等待使用者輸入，這時不妨提示使用者應該輸入什麼。

可以在 scanf 之前加入一個 printf ，請使用者輸入一個被乘數和乘數：

#include <stdio.h>

int main(){

 int multiplicand, multiplier;

 printf("請輸入被乘數和乘數，中間以空白分隔：");

 scanf("%d %d", &multiplicand, &multiplier);

 printf("%d", multiplicand \* multiplier);

 return 0;,

}

到這邊，我們已經一步一步建構出一個簡單的程式了。等到下一章，搭配 if 來使用，讓你的程式在不同情況做不同事情，會更有成就感的！

(五)：如果這樣會怎樣之選擇結構

能夠互動以後，你會不會希望程式在不同時候能夠做不同的事情呢？這時候就要用到選擇結構；選擇結構是什麼？講白話一點，就是 if 啦！

if 的用法就是，如果達到某種條件，就做這件事情；反之，就做另外一件事情。不過，為了描述達到某種條件，我們必須先學會關係運算子。

關係運算子

關係運算子是什麼呢？別看它名字很困難一樣，其實關係運算子就是數學上數字的比較，有大於(>)、小於(<)、大於等於(>=)、小於等於(<=)、等於(==)、及不等於(!=)。

在第二章的時候提過：數學上的 = 和 C 語言裡面的 =，意思不太一樣。數學上的 = 是指等號左右兩邊的值相等；但在 C 語言(及絕大部分程式語言裡面)，= 的意思是把等號右邊的值存放(assign)到等號左邊的變數裡，有←的味道存在。

而如果要比較等號兩邊是不是相等，用的是 ==，很重要！\*\*if 裡面幾乎只會用到 ==！\*\*如果你寫 = 不會出錯，但意思不一樣。

邏輯運算子

邏輯運算子其實也不是太難的東西，就是且(AND)、或(OR)和非(NOT)而已。若要在同時達到條件 a 和條件 b 才做某件事，可以使用 if (a && b)；若要達到條件 a 或 條件 b 就做某件事，可以使用 if (a || b)。注意：關係運算子和等於(==)一樣，且(&&)與或(||) 都必須重複。

而非則是用 ! 來表達，例如當沒有達成 a 的時候做某件事可以寫成 if(!a)。

選擇結構：if

下面這段程式碼可以判斷輸入的整數是否大於 5：

#include <stdio.h>

int main(){

 int a;

 scanf("%d", &a);

 if(a>5){

 printf("%d 大於 5\n", a);

 }else{

 printf("%d 沒有大於 5\n", a);

 }

 return 0;

}

把條件放在小括號裡面，然後用大括號開啟一個新的區段，並把達成條件要做的事情(敘述)放在該區段裡面；也可以使用 else，指定沒有達成條件才做的事情。

然而不使用 else 也是可以的：

#include <stdio.h>

int main(){

 int a;

 scanf("%d", &a);

 if(a>5){

 printf("a>5才會輸出這行");

 }

 printf("不管怎樣都會輸出這行\n");

 return 0;

}

也可以在使用 else if，例如經典的是否為閏年題目：

#include <stdio.h>

int main(){

 int y=0;

 printf("請輸入年份：");

 scanf("%d", &y);

 if(y%4000==0){

 printf("%d 年不是閏年\n", y);

 }else if(y%100==0 && y%400!=0){

 printf("%d 年不是閏年\n", y);

 }else if(y%4==0){

 printf("%d 年是閏年\n", y);

 }else{

 printf("%d 年不是閏年\n", y);

 }

 return 0;

}

if 裡面也可以再包 if (所謂的巢狀 if)：

#include <stdio.h>

int main(){

 int a=0;

 scanf("%d", &a);

 if(a%7==0 && a%3==0){

 printf("%d 是 21 的倍數\n", a);

 }else{

 if(a%7==0){

 printf("%d 是 7 的倍數\n", a);

 }else if(a%3==0){

 printf("%d 是 3 的倍數\n", a);

 }else{

 printf("%d 不是 3 或 7 的倍數\n", a);

 }

 }

 return 0;

}

if 的用法大概是這樣，如果把多個 if 拿來做搭配，就能夠做出決策樹的效果。基本上選擇結構學到這裡就可以了。但如果你嫌 if 很麻煩的話，可以試著學學看 switch 結構。

(六)：繞啊繞轉啊轉的迴圈結構

到目前為止，我們的程式差不多都是由上往下走，很循序漸進的執行。如果想要讓重複性的執行某些動作，就必須使用迴圈結構。

迴圈是什麼？迴圈就是重複性的執行某個區段裡的程式碼。但要重複幾次？什麼情況下要重複？迴圈有幾種？慢慢把它看完你就懂了！

在 C 語言裡面，迴圈主要有三種：for、while、do while。

for 迴圈

for 迴圈的基本架構是這樣的：

for(初始化動作; 執行條件 ; 增加動作){

 敘述

}

通常我們會使用一個變數來當作計數器(counter)，初始化動作會給這個計數器一個起始值；第二個是執行條件，只有當符合執行條件的時候才會進到 for 迴圈裡執行敘述；第三個則是每次做完敘述要增加的動作。

例如下面這段程式碼，可以輸出 1~5 的數字：

#include <stdio.h>

int main(){

 int i;

 for(i=1;i<=5;i++){

 printf("%d ", i);

 }

}

輸出結果會是：

1 2 3 4 5

看似困難的 for 迴圈，其實只要搞清楚他的脈絡是怎麼走的，就不會亂掉了。

同樣的，也可以用巢狀 for 迴圈來印出直角三角形。

#include <stdio.h> int main(){ int i, j; for(i=1;i

輸出結果會是：

＊

＊＊

＊＊＊

＊＊＊＊

＊＊＊＊＊

原則上，使用 for 迴圈的時機為需要一個計數器從小到大(或從大到小)慢慢做某件事情。

while 迴圈

while 迴圈的基本架構則是這樣，其實比 for 還要簡單：

while(條件){

 敘述

}

只要條件成立，就會一直執行敘述。所以 while 迴圈的使用時機為：達成某個條件時就一直做某件事情的時候。

例如我一樣要輸出 1~5 的數字：

#include <stdio.h>

int main(){

 int i=1;

 while(i<=5);

 printf("%d ", i);

 i++;

 }

}

輸出結果：

1 2 3 4 5

當然，while 也可以弄成巢狀迴圈。

do-while 迴圈

do while 迴圈其實不太重要？畢竟他就是 while 迴圈，只是迴圈開始之前再做一件事情而已，架構如下：

do {

 敘述

} while (條件);

一開始會先執行一次敘述，接著去判斷條件是否成立，成立的話再回去執行敘述，直到條件不成立為止。

值得注意的是 while (條件); 最後面有個分號。

不過 do while 迴圈還是有它的用途，例如我要計算正整數的位數：

#include <stdio.h>

int main(){

 int digits = 0, n;

 printf("輸入一個正整數：");

 scanf("%d", &n)

 do {

 n /= 10;

 digits++;

 } while (n>0);

 printf("這個整數是 %d 位數\n", digits);

 return 0;

}

如果我們單純使用 while(n>0) 的迴圈的話，那輸入 0 會顯示 0 位數，但答案應該是 1 位數，這時候使用 do-while 迴圈就可以解決這個問題了。

以上就是 C 語言中的三種迴圈。從 while 和 for 的範例中可以發現，同樣的結果其實用三種迴圈都可以寫得出來，不過通常我們會選擇看起來最簡潔的，或是自己最習慣的方式。可別小看 C 語言只有三種迴圈，這三種迴圈搭配起來就可以做到很多不同的事情唷！

再者，巢狀迴圈中需要注意：通常當你有 n 層迴圈，你就會至少用到 n 個變數。

(七)：寫一遍就能用好幾遍好方便的函式

C 語言中，可以把重複的程式提出來寫成函式(function)，而需要的時候直接呼叫這個函式就好，也就是模組化的概念。

什麼叫模組化？簡單來說，就是把特定功能分出來當成一個模組，需要的時候只要呼叫這個模組就好；而需要修正的時候也只要修正模組即可。

比方說，有一個可以算 1+2+…+100 的程式如下：

#include <stdio.h>

int main(){

 int i, sum=0;

 for(i=1;i<=100;i++){

 sum += i;

 }

 printf("%d", i);

 return 0;

}

而其中第 4 行到第 8 行是用來計算和輸出的程式碼。我們可以把它獨立出來，寫成一個函式，必且在 main 主程式中呼叫它。例如我們把這個部分命名為 Count1To100：

#include <stdio.h>

void Count1To100(){

 int i, sum=0;

 for(i=1;i<=100;i++){

 sum += i;

 }

 printf("%d", i);

}

int main(){

 Count1To100();

 return 0;

}

這樣就算寫好一個簡單的函式囉！

不過你也許會說，程式好好的幹嘛沒事寫成一個函式？這樣看起來沒感覺，當你程式很多地方都需要重複寫它的時候，你就有感覺了。另外，函式還有更好用的一個地方，就是它可以傳入和回傳值。

傳入值和回傳值是什麼？

這大概要先從數學上的函數開始講起了。 f(x)=x+1f(x)=x+1，看到這個函數你想到什麼？

在數學的函數中，可以把 xx 代入不同的數，得到不同的對應值，例如 f(2)=3f(2)=3、f(3)=4f(3)=4、f(4)=5f(4)=5……等等。以 f(2)=3f(2)=3 來說，22 就是傳入值，33 就是回傳值。

而函數的代入值也不見得只有一個，例如 f(x,y)=x+y+2f(x,y)=x+y+2。那就會得到 f(1,1)=4f(1,1)=4、f(2,3)=7f(2,3)=7…… 等等。

函式、函數，傻傻分不清楚？

函式和函數在英文都是 function，只是翻譯問題而已。原則上是一樣的東西。

C 語言的函式呢？

C 語言的函式和數學上的函數用法基本上是一樣的。要先看看函式有幾個傳入值，把相對應的傳入值傳進去後，就會做指定動作，然後回傳一個回傳值。

好饒舌啊 QQ，更饒舌的是 C 語言的函數不一定要有傳入值和回傳值。

在開始之前，來回顧一下資料型態吧。變數有不同的形態，常見的有 int、double 等等。另外，如果空的話會寫成 void。

那，來看看一個簡易的加法函式好了：

int plus(int a, int b){

 return a+b;

}

首先一開始宣告這個函式的資料型態，也就是要回傳的資料型態，接著打函式的名字，小括號內放的是傳入值的資料型態和名稱。接著就和一般寫程式沒有兩樣了，只是最後要把結果用 return 來回傳。

例如這個函式會回傳 int 型態，名字叫做 plus，使用的時候需要傳入兩個 int，一個叫 a、一個叫 b，最後把 a+b 的結果回傳回去。

那要怎麼用它呢？以下這個範例程式：

#include <stdio.h>

int plus(int a, int b){

 return a+b;

}

int main(){

 int m, n;

 scanf("%d %d", &m, &n);

 int result = plus(m, n);

 printf("%d", result);

}

一開始把 plus 函式寫在前面，main 主程式裡面做的事情是：開兩個整數 m、n，用 scanf 得到兩個使用者輸入的數，分別存到 m、n 變數裡面。

接著我再開一個變數叫 result，存放 plus(m, n) 的計算結果。為什麼 result 要開 int 而不是 double 或其他資料型態？因為 plus 函式的開頭是寫 int，代表回傳值會是 int 類型。

這時，result 裡面存放的已經是 m 和 n 相加結果的值了，可以直接用 printf 把它輸出。

另外，為什麼函式的傳入值是 a、b，但 main 丟的是 m、n？這樣也可以？

在 C 語言裡面，每個變數在自己的函式裡面都是獨立的，main 裡面宣告的變數和其他函式宣告的變數不會互相影響。例如這個例子中 main 在呼叫 plus 函式時，丟的其實是 m、n 的值，而 plus 函式也只是把這個值接過來而已，它們並不在意變數到底怎麼命名的。

要是能夠理解這裡在說什麼的話，其實也可以把函式當作一個變數使用。例如剛剛那個範例的 main 裡面可以改成這樣：

int main(){

 int m, n;

 scanf("%d %d", &m, %n);

 printf("%d", plus(m, n));

 return 0;

}

就不用另外去開一個變數來儲存。

對了，值得注意的是：函式一遇到 return，就會把後面的值傳出去，然後函式裡剩下的東西都不會執行了唷！

OK 到這裡還是不能理解的話，建議你先多看幾遍。\*\*函式之間是怎麼運作的比怎麼寫函式還重要！\*\*畢竟函式並沒有一定的寫法，你是要把 plus 當成 int 回傳，還是要把 plus 當 void 在裡面直接 printf，這些都是可以視情況而定的，更何況等你熟練了以後還可以直接去套別人寫好的函式來用，根本不用自己寫函式。