第一章認識 C 語言

資訊科技系
林偉川

本章簡介

- 電腦並不瞭解人類的語言，當人類需要驅使電腦工作, 與其溝通時, 就必須以電腦所能理解的語言 (稱為電腦語言) 來對電腦下達命令。
- 程式語言的種類很多, 有些語言較接近人類的日常用語, 我們稱為高階語言, 例如C、C++(C#)、Java 等, 這種語言較易學習。
- 語言較接近電腦的運作方式, 稱之為低階語言, 例如 Assembly, 這種語言不易學習, 但寫出來的程式其執行效率較好。

2

C 語言的由來

- C 語言是最被廣泛使用的一種高階語言, 為避免各開發廠商所用的 C 語言語法產生差異, 由美國國家標準局 (American National Standard Institution) 為 C 語言訂定一套完整的國際標準語法, 稱為 ANSI C, 做為 C 語言的標準。
- Free Software:

http://www.bloodshed.net/download.html

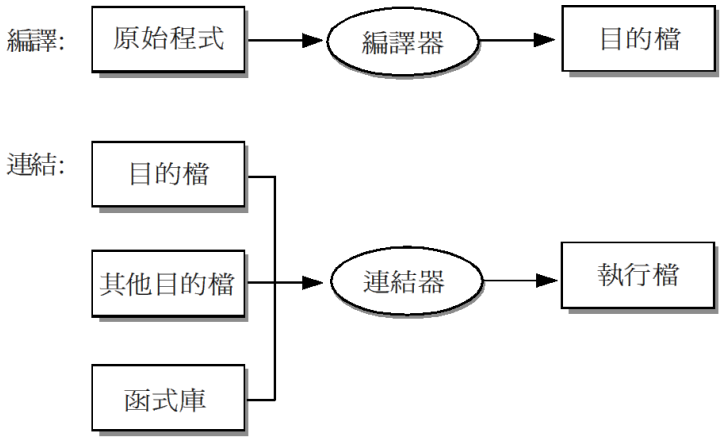
3

讓 C 語言程式變成可以執行

- 我們寫出來的 C 程式稱為原始程式, 將高階程式語言所寫的原始程式, 翻譯成機器語言言的方式有編譯與直譯兩種。
- C 語言採用的是編譯的方式 (因此可稱之為編譯式語言)。將 C 原始程式編譯為可執行檔, 需經過編譯和連結兩個動作：

4

編譯(.exe)和連結示意圖



5

開發步驟

- 編輯
- 前置處理
- 編譯➔.o
- 連結➔.exe
- 載入
- 執行

6

編譯

- 使用編譯器 (Compiler) 將人類看得懂的原始程式, 翻譯成電腦看得懂的目的檔。
- 在編譯的過程中, 編譯器就如同是語言的翻譯員會檢查原始程式中的語法是否有錯誤。如果有錯, 程式設計者必須修正原始程式的內容,再重新編譯。

7

連結

- 連結器 (linker) 的用處是將目的檔, 與程式中所使用到的函式庫做連結, 而產生完整的執行檔。如果開發的程式比較大, 或是參與開發的人比較多, 此時會將程式分成多個原始程式,每個原始程式都可個別編譯成目的檔, 然後於連結時, 再將所有的目的檔與函式庫連結成一個完整的執行檔。

8

編譯式程式語言列表：

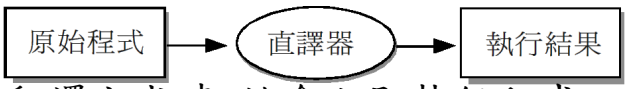
- 目前流行的程式開發工具, 例如 Borland C++ Builder、Microsoft Visual C++, 都已將編寫程式碼、編譯與連結等功能整合在一個操作介面中, 對開發程式來說, 相當的方便。

程式語言	編譯工具
C/C++	Visual C++, C++ Builder 等
Basic	Visual Basic
Java	JDK 虛擬機器, JBuilder

9

直譯式語言

- 直譯如同是現場口譯, 當程式需要執行時才開始翻譯。翻譯直譯式語言的工具叫做直譯器 (Interpreter), 其翻譯過程也與編譯式語言不同：



- 當直譯器翻譯完成時, 就會立即執行程式, 不會產生執行檔, 所以下次要再用到這個程式時, 又要翻譯一次。像JavaScript、VBScript、Perl 等都算是直譯式語言。

10

接觸 C 語言：由無到有

- 一個會輸出一行文字的程式：

```
#include <stdio.h>
int main(void) {
 printf("this is a test\n"); //單行註解
 /*
    多行註解
     comment
 */
 return 0;
}
```

11

接觸 C++ 語言：由無到有

- 一個會輸出一行文字的程式：

```
#include <iostream>
int main(void) {
 cout <<"this is a test" << endl; //單行註解
 /*
    多行註解
     comment
 */
 return 0;
}
```

12

印出一行字

1. #include <stdio.h>(C), #include <iostream>(C++),是個特別的程式碼, 稱為前置處理指令 (preprocessing directive)。主要的功用, 就是把 stdio.h 這個檔案 (此檔案會附於安裝編譯器的子資料夾下) 的內容放在原始程式的最開頭。

13

印出一行字

2. int main (void) 稱之為 main () 函式, 這是每個 C 程式都會有的部分, 稱為程式的進入點。main () 前面的 int 是整數的意思, 表示這個 C 程式在執行結束時, 會傳回一個整數值到作業系統。至於作業系統如何使用程式傳回值、void 的意思, 在此我們先不深入探討。請記得 "int main (void)" 是典型的 C 程式主體。

14

什麼都不做的程式

3. 在兩個大括號 { ... } 中所含的程式, 就是 main () 函式的內容。程式需要做哪些事情, 都是寫在這裡。
4. 在/* ... */ 裡面的字, 稱之為註解。當編譯器遇到註解時會跳過註解文字不做任何編譯, 因為這些註解是寫給人看的, 電腦不需理會。

15

印出一行字

5. printf () 是 C 語言裡最基本的輸出函式, 其功能是將括弧 () 中, 用引號包住的字串 (在本例中就是 “this is a test\n") 輸出到標準輸出裝置。由於預設的標準輸出裝置就是螢幕, 所以這行程式的功用, 就是將 " this is a test" 這段訊息, 輸出到螢幕上。可把任何想輸出的文字, 放在 "..." 之間。

16

什麼都不做的程式

6. 『return 0; 』就是這個範例中, main () 函式中唯一的一行敘述 (statement)。這行敘述的功能, 就是將數值 0 傳回給作業系統。通常在撰寫應用程式時, 會將程式執行的狀況以『傳回值』反應給作業系統。習慣上會用數值 0 表示程式執行順利, 無任何錯誤的意思；而用 1～255 之間的數值來表示執行過程發生錯誤, 至於要用什麼數字表示什麼錯誤, 則依程式設計人員自行應用, 並無特別的規定。

17

C 語言的基本語法說明

- main (void) 是什麼意思
- 含括檔 #include <...> 淺說
- 程式的分段：使用大括號 {} 與分號

18

main (void) 是什麼意思

- main (void) 是每個 C 程式都一定要有的函式。
- 函式 (function) 簡單的說, 就是一段程式的集合, 並能產生某項特定的功能。
- main (void) 函式中的 void 的意思是空白, 也就是說 main () 函式不接受任何參數。如果希望 main () 函式能接受參數, 讓程式每次都能依指定的參數做不同的運算, 則 main () 函式括弧中的寫法將有不同。

19

main (void) 是什麼意思

- C 語言提供許多不同的函式, 這些函式都已事先編譯好存於函式庫中, 若在程式中用到這類函式, 在連結的階段, 就會將函式庫中編譯好的函式內容, 連結到我們的程式中。
- main () 函式比較特別, 它不像函式庫中的函式, 都有事先設計好的功用。main () 函式的內容, 完全由寫程式的人自行決定, 換言之, 我們希望程式能完成某項工作, 就要將完成該工作的 C 語言敘述放在 main () 函式中。

20

解 x=123 代入 2x+99 的結果

3 程式 Ch01_03.c 解 x=123 代入 2x+99 的結果

```
01  #include <stdio.h>
02
03  int main(void)
04  {
05    int x,y;                  /* 宣告變數 x,y */
06    x = 123;                  /* 設定 x 的值爲 123  */
07    y = 2 * x + 99;           /* 算式相當於 2x+99 */
08    printf("答案 = %d",y);  /* 印出運算結果 */
09    return 0;                /* 傳回 0 表示程式執行無誤 */
10  }
```

執行結果

```
答案 = 345
```

21

解 x=123 代入 2x+99 的結果

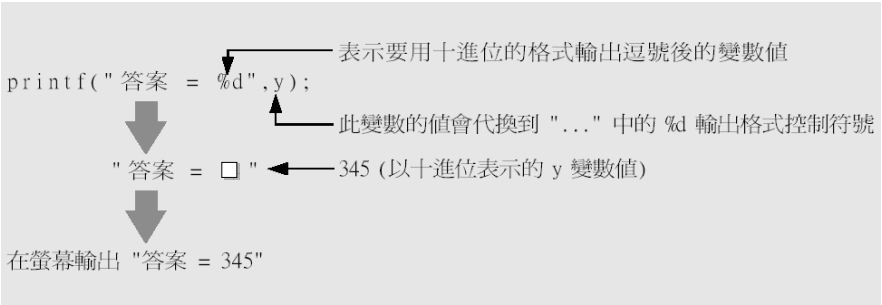
1. 第 5 行為宣告變數 x 與 y 都是整數 (int)。
2. 第 6 行是將 x 值以 123 代入, 另一種講法則是：將 x 的值設定為 123。
3. 第 7 行是用 y 來接受 2x+99 的運算結果, 在 C 語言中所有的算式都要寫成像本行的模式, 也就是說等號的左邊只能有一個變數, 右邊則是算式。

22

解 x=123 代入 2x+99 的結果

4. 第 8 行是印出 y 的值, 也就是印出 2x+99 的結果。對照一下本行內容和輸出結果, 您會發現我們要 printf () 輸出的是 "答案 = %d", 但執行結果則是輸出 "答案 = 345", 也就是說 printf () 函式自動將 %d 的部分代換成 y 的值 345。其中 %d 稱為 printf () 的輸出格式控制符號, 其作用就是將逗號後的變數 y 的值, 以十進位的格式輸出在 %d 的位置：

解 x=123 代入 2x+99 的結果



- 在接下來兩章, 還會介紹 printf () 函式其它的輸出格式控制符號。

含括檔 #include <...> 淺說

- #include, 稱為前置處理指令 (preprocessing directive)。編譯器在編譯 C 程式前, 會先由前置處理器 (preprocessor) 來處理程式中的前置處理指令。
- 在原始程式第 1 行加上 #include<stdio.h>, 會把 stdio.h 這個檔案的內容放在原始程式的最開頭。這個動作稱為含括 (include), 而被含括到我們程式中的 stdio.h 則稱為含括檔, 其特色為副檔名為 .h。

25

含括檔 #include <...> 淺說

- 使用函式前, 必須先宣告 (declare) 函式, 為方便我們直接使用函式庫中的函式, C 語言預先將標準函式庫中的函式宣告, 分門別類放在各個含括檔中。
- 程式中用到了 printf () 函式, 而其宣告就存放 stdio.h 中。若不在程式開頭加上 #include<stdio.h> 這一行, 在程式的連結階段將會出現 "printf () 函式未定義" 之類的錯誤訊息。

26

含括檔 #include <...> 淺說

- 在程式一開始先以 #include 的方式, 將程式中要使用的函式宣告含括進來。
- 單一程式中所能含括的檔案數並無限制, 不過只要使用至某個函式庫的函式時,一定要先用 #include 將相關 .h檔含括進來, 否則會出現連結錯誤：

27

含括檔 #include <...> 淺說

```
#include <xxx>◄──將使用宣告於 xxx 檔案中的函式
#include <ooo>◄──將使用宣告於 ooo 檔案中的函式
...
int main(void)
{
   ...
}
```

28

程式的分段：用大括號{}與分號

- C語言對其程式寫作格式中有許多嚴謹的規定,這是為了避免編譯器在編譯時無法解讀程式的含意而發生編譯錯誤。在寫程式時就好像在寫作文一樣,需要用標點符號來達到"段落分明"。
- C語言使用以下兩種符號來標示段落：
  - 大括號{}
  - 分號；

大括號{ }括住程式

```
int main(void)
{                ←用大括號括住函式內容
 ...
  if (...)
 {               ←用大括號括住 if 的條件判斷式
    ...
 }

  for(...)
 {               ←用大括號括住迴圈內容
    ...
 }
  ...
}

func(...)
{                ←用大括號括住其他函式內容
  ...
}
```

- 大括號{ }中間的程式表示是一小段函式的結束,或是一小段的敘述結束,我們稱之為程式區塊(block)：

分號表示敘述的結尾

- 每一句完整的程式後要加上分號, 表示該敘述結束：

```
printf("C 語言的世界, 你好啊!!");←呼叫 printf()函式的敘述結束,用分號結尾

int x,y;                          ←宣告變數完畢,用分號結尾

y = 2 * x + 1;                    ←運算式敘述完畢,用分號結尾
```

31

分號表示敘述的結尾

- 每個由分號結尾的一個程式片段, 可稱為一段敘述 (Statement)。只要依規定在每個敘述結尾加上分號,編譯器就能依分號出現的位置, 自動判讀出某幾個字是一個敘述, 某幾個字又是另一個敘述。
- 只有少數幾個敘述不必在結尾加上分號 (if、else、while 等流程控制敘述), 其它的敘述都一定要用分號做結尾。

32

適時的分行：便於程式的閱讀

- 在 C 語言中並未限制單行的長度, 也就是說就算把整個程式寫成非常長的一行, 但只要將每個敘述正確的以分號標開, 這樣在編譯過程中也不會發生錯誤。
- 這樣的程式讀起來, 感覺會像讀一篇沒有標點符號的文章一樣, 不知道哪裡該停頓, 哪裡段落結束：

```
int main(void) { int x,y,z; x=10; y=5; z=20; }
```

33

適時的分行：便於程式的閱讀

- 如果我們依照分號將每行分開, 看起來就容易閱讀多了：

```
int main(void) {
 // int x, y, z;
  int x;
  int y;
  int z;
  x=10; y=5;   z=20;
}
```

34

適時的分行：便於程式的閱讀

- a=b+c+d+e+f+g+h+i+j+k+l+m;
- 當一行敘述太長時, 只要是變數和運算符號(簡稱算符)、逗號、括號之間, 都可依需要斷行, 在換行處不需要加任何符號, 只需在運算式結束處加分號即可, 如此並不會影響編譯器的判讀：

```
a=b+c+d+e+
   f+g+h+i+
   j+k+l+m;
```

35

適時的分行：便於程式的閱讀

- 斷行時切記不可將原本完整的變數名稱、函式名稱、關鍵字分斷, 因為這都將造成編譯器解讀錯誤：

```
main()    斷行成 ⟶  mai
                      n()

printf()  斷行成 ⟶  print
                      f("Hello!\n");
```

36

適時的分行：便於程式的閱讀

- 用雙引號 (“) 括住的一段字串, 也不能隨意斷行：
  printf(“字串太長時，可以用此種寫法”);
- 如果想將 “...” 的內容分成多行, 必需在斷行的位置加上 “\” 符號：
  printf(“字串太長時， \
  可以用此種寫法”);

37

適時的分行：便於程式的閱讀

- 或者乾脆將這行呼叫分成 2 個敘述 (呼叫 printf () 2 次)：
  printf(“字串太長時”);
  printf(“，可以用此種寫法”);

38

程式碼內縮：表現出程式的層次感

- 除了行的排列外，寫程式也要注意到層次感，段落明顯，章節分明：

```
int main(void)
{
   int x,y;          ←main() 函式的大括號內容要內縮
   ...
   for (...)
   {
      x = ...        ←迴圈的大括號內容要內縮
      ...
      if (...)
      {
         y= ...      ←條件式的大括號內容要內縮
      }
   }

}
```

39

善用程式碼的註解：幫助了解程式

- 寫程式要注意多加註解：
- 您看懂這個程式的目的了嗎?

4 程式 Ch01_04.c 毫無註解的程式

```
01  #include <stdio.h>
02
03  int main(void)
04  {
05     int sum,difference;
06     int bignumber, smallnumber;
07     sum = 10;
08     difference = 4;
09     bignumber = (sum + difference)/ 2;
10     smallnumber = (sum - difference)/2;
11
12     printf("大數: %d\n",bignumber);
13     printf("小數: %d\n",smallnumber);
14
15     return 0;
16  }
```

執行結果

```
大數: 7
小數: 3
```

善用程式碼的註解：幫助了解程式

- 我們只看到程式中宣告了一堆變數, 經過簡單的運算後, 再將結果印出。
- 除此之外, 也只能從所宣告的變數名稱來猜測了。

41

善用程式碼的註解：幫助了解程式

- 將同一個程式加上註解：

5 程式 Ch01_05.c 和差問題

```
01 /* 程式目的：已知兩個數的和與差, 求出兩個數的值 */
02  /* 檔案名稱：Ch01_05.c */
03  /* 程式設計：阿哲 */
04  /* 完成日期：2003/04/30 */
05
06  #include <stdio.h>
07
08  int main(void)
09  {
10    int sum,difference; /* 宣告變數 sum 爲兩數和, difference 爲兩數差 */
11    int bignumber, smallnumber;      /* 宣告變數 bignumber 和 */
12                                     /* smallnumber ,分別用於存放大數和小數的值 */
13    sum = 10;                        /* 指定數值 10 給變數 sum */
14    difference = 4;                  /* 指定數值 4 給變數 difference */
15    bignumber = (sum + difference)/ 2;    /* 大數 = (和+差)/2 */
16    smallnumber = (sum - difference)/2;   /* 小數 = (和-差)/2 */
17
18    printf("大數: %d\n",bignumber);        /* 將大數的值輸出 */
19    printf("小數: %d\n",smallnumber);      /* 將小數的值輸出 */
20
21    return 0;
22  }
```

42

善用程式碼的註解：幫助了解程式

- 原來這個程式是在解數學上有名的『和差問題』, 就是知道兩個數的和跟差後, 再反算出原來兩個數的值。
- 加了註解的程式還是比較容易了解。
- 程式第 17、18 行呼叫 printf () 函式輸出變數值時, 在 "..." 中使用到一個特別的控制字元 "\n", 它代表的意思是請 printf () 函式在此換行, 而且下次輸出時是從下一行的開頭開始。
- 所以上列程式的輸出會分別輸出在 2 行：

43

善用程式碼的註解：幫助了解程式

```
大數： 7
小數： 3
```

- 請不要以為程式是分別在第 17、18 行呼叫 printf () 函式, 所以輸出結果也是 2 行。
- 若您將 "\n" 刪除並重新編譯執行程式, 就會發現第 2 個 printf () 的輸出會緊接在前一行輸出結果的後面, 也就是 2 次輸出都擠到同一行了。

44