Arduino 程式語言介紹

Arduino 的語法為義大利籍工程師 David Mellis 所設計，用法與 C 語言相似，但在使用上更具有便利性，十分容易上手的程式。

3-1 基本架構（Structure）

一個 Arduino 程式碼（SKETCH）基本上是由 void setup（）、void loop（）兩部分組成， void setup（）在位於主程式 void loop（）前，只會在 Arduino 板子加上電源或 CPU 重置（reset）時執行 1 次，爾後就不會再執行，一般在函數內放置初始化 Arduino 板子的程式。一般稱 void loop（）為主程式，在此函數內放置你的 Arduino 腳本。這部份的程式會一直重複的被執行，直到 Arduino 板子的電源被關閉。圖 3-1 為 Arduino 程式基本架構，圖 3-2 為 Blink 程式碼。

```
void setup()     //初始設定區塊（只執行一次）

{



}

void loop()      //重複執行區塊（不斷地重複執行）

{



}
```

圖 3-1、Arduino 程式基本架構

```
int ledPin = 13;     //設定第13 pin為LED燈的接腳
void setup()
{
pinMode(ledPin, OUTPUT);   //設定pin腳模式為輸出
```

```
}
void loop()
{
digitalWrite(ledPin, HIGH);   //給pin腳高電壓 (LED 亮)
delay(1000);     //延遲1秒（1000毫秒）
digitalWrite(ledPin, LOW); //給pin腳低電壓 (LED 不亮)
delay(1000);     //延遲1秒（1000毫秒）

}
```

圖 3-2、Blink 程式碼

3-2 變數、常數與資料型態（Variables、Constants、Data Types）

常數（constant）與變數（variable）使用前都必須宣告它所欲儲存的資料型態，如此才能讓編譯器（compiler）配置適合的記憶體空間給它，常數是在指在分配的記憶體空間裡，放置固定不變的資料，而變數是在在分配的記憶體空間裡的資料是可改變的。宣告常數或變數的格式如下：

資料型態　常數/變數[=*預設值*];

其中**[=*預設值*]**並非必要項目，而分號(;)是結束符號；例如要宣告一個整數型態的 x 變數，其預設值為 100，如下：

```
int   x = 100 ;
```

若無預設值，則為：

```
int   x ;
```

若要同時宣告 x、y、z 三個整數型態的變數，則在變數名稱之間，以「,」分開，如下：

```
int   x ,y,z ;
```

arduino 程式語言內定之常數如表 3-1 所示；

表 3-1 arduino 程式語言內定之常數

名　　稱	說　　明
INPUT	定義數位接腳為輸入狀態
OUTPUT	定義數位接腳為輸出狀態
HIGH	電壓為 3V 或大於 3V
LOW	電壓為 2V 或小於 2V
true	定義邏輯準位為 true，時常將

	true 視為 1
false	定義邏輯準位為 false，時常將 false 視為 0

資料型態包括字元（char）、字串（string）、整數（integer）、浮點數（float）與無（void），其中字元與整數又分有符號（signed）和無符號（unsigned）兩類，如表 3-2 所示。

表 3-2 資料型態

型　　態	名　　稱	位　元　數	範　　圍
void	無	0	無
boolean	布林	8(1 byte)	true 或 false
char	字元	8(1 byte)	-128~+127
unsigned char	無號字元	8(1 byte)	0~255
byte	位元組	8(1 byte)	0~255
int	整數	16(2 bytes)	-32768~+32767
unsigned int	無號整數	16(2 bytes)	0~65535
word	2 個位元組	16(2 bytes)	0~65535
long	長整數	32(4 bytes)	-2,147,483,648~2,147,483,647.
unsigned long	無號長整數	32(4 bytes)	0~ 4,294,967,295
short	短整數	16(2 bytes)	-32768~+32767
unsigned short	無號短整數	16(2 bytes)	0~65535
float	符點數	32(4 bytes)	±10175494E-38~3.402823E+38
double	雙倍精度符點數	64(8 bytes)	±1.7E308

有些時候，我們可能需要轉換變數型態，例如把整數轉換成符點數，或把符點數轉換成整數等等，以符合程式所需。表 3-3 為 arduino 所提供的型態轉換函數。

表 3-3、為 arduino 所提供的型態轉換函數

函 數 名 稱	功　　能　　說　　明
char(x)	x 為任何資料型態之值，傳回 char 型態之值
byte(x)	x 為任何資料型態之值，傳回 byte 型態之值
int(x)	x 為任何資料型態之值，傳回 int 型態之值
word(x) word(h, l)	x 為任何資料型態之值，傳回 word 型態之值 h 為高位元組，l 為低位元組，傳回 word 型態之值
long(x)	x 為任何資料型態之值，傳回 long 型態之值

float(x)	x 為任何資料型態之值，傳回 float 型態之值

3-3 運算子（Operators）

程式是由許多敘述(statement)組成的，兒敘述的基本單位是運算式與運算子，運算子是程式敘述中運算的符號，可分為以下幾種：

- **算數運算子**（Arithmetic Operators）

顧名思義，算數運算子就是執行算數運算功能的操作符號，除了四則運算（加減乘除）外，還有取餘數，如表 3-4 所示。

表 3-4、算數運算子

符　號	功　　能	範　　例	說　　明
=	設定（assignment）	x=y	將 x 變數的內容設定給 y 變數
+	加（addition）	a=x+y	將 x 與 y 變數的值相加，其和放入 a 變數內
-	減（subtraction）	b=x-y	將 x 變數減 y 變數的值，其差放入 b 變數內
*	乘（multiplication）	c=x*y	將 x 與 y 變數的值相乘，其積放入 c 變數內
/	除（division）	d=x/y	將 x 變數除以 y 變數的值，其商放入 d 變數內
%	取餘數（modulo）	e=x%y	將 x 變數除以 y 變數的值，其餘數放入 e 變數內

- **比較運算子（Comparison Operators）**

比較運算子是比較兩個運算式（或變數）間的大小關係，其結果只有 1（true）或 0（false）一值，如表 3-5 所示。

表 3-5、關係運算子

符　號	功　　能	範　　例	說　　明
==	等於(equal to)	x==y	比較 x 與 y 變數是否相等，相等其結果為 1，不相等則為 0
!=	不等於(not equal to)	x!=y	比較 x 與 y 變數是否不相等，不相等其結果為 1，相等則為 0
<	小於(less than)	x<y	若 x 變數小於 y 變數，其結果為 1，否則為 0

>	大於(greater than)	x>y	若 x 變數大於 y 變數，其結果為 1，否則為 0
<=	小於或等於(less than or equal to)	x<=y	若 x 變數小於或等於 y 變數，其結果為 1，否則為 0
>=	大於或等於(greater than or equal to)	x>=y	若 x 變數大於或等於 y 變數，其結果為 1，否則為 0

- **布林運算子（Boolean Operators）**

布林運算子是執行邏輯運算功能，包括 AND（及）、OR（或）、NOT（反相），其運算結果為 1 或 0，如表 3-6 所示。

表 3-6、布林運算子

符　號	功　能	範　例	說　明
&&	及（and）	(x>y)&&(x>z)	若 x 大於 y 和 z 變數，其結果為 1，否則為 0
||	或（or）	(x>y)||(x>z)	若 x 大於 y 或大於 z 變數，其結果為 1，否則為 0
!	反相（not）	!(x>y)	若 x 大於 y，其結果為 0，否則為 1

- **位元運算子（Bitwise Operators）**

位元運算子是針對變數中每一個位元作邏輯運算，如表 3-7 所示，表中 x=0x26，y=0xe2。

表 3-7、位元運算子

符　號	功　能	範　例	說　明
&	位元及(bitwise and)	a=x&y	將 x、y 變數作位元及運算，其結果為 0x22 放入 a 變數中，x、y 內容不變
|	位元或(bitwise or)	a=x|y	將 x、y 變數作位元或運算，其結果為 0x6 放入 a 變數中，x、y 內容不變
^	位元互斥或(bitwise xor)	a=x^y	將 x、y 變數作位元互斥或運算，其結果為 0xc4 放入 a 變數中，x、y 內容不變
~	取 1's 補數(bitwise not)	a=~x	將 x 變數取 1's 補數運算，其結果為 0xd9 放入 a 變數中，x 內容不

			變
<<	位元左移(bitshift left)	a=x<<2	將 x 變數的值左移兩個位元，其結果為 0x98 放入 a 變數中，x 內容不變
>>	位元右移(bitshift right)	a=x>>1	將 x 變數的值左移一個位元，其結果為 0x13 放入 a 變數中，x 內容不變

- **合成運算子（Comparison Operators）**

類似 c 語言也提供一些簡潔方式，將算數運算子和設定運算子結合為新的運算子，如表 3-8 所示。

表 3-8、簡潔算數運算子

符號	功能	範例	說明
++	加 1 (increment)	a++	等於 a=a+1
--	減 1 (decrement)	a--	等於 a=a-1
+=	加入 (compound addition)	a+=b	等於 a=a+b
-=	減去 (compound subtraction)	a-=b	等於 a=a-b
*=	乘入 (compound multiplication)	a*=b	等於 a=a*b
/=	除 (compound division)	a/=b	等於 a=a/b
&=	位元及(compound bitwise and)	a&=b	等於 a=a&b
|=	位元或(compound bitwise or)	a|=b	等於 a=a|b
%=	取餘數(compound bitwise modulo)	a%=b	等於 a=a%b
<<=	位元左移(compound bitshift left)	a<<=b	等於 a=a<<b
>>=	位元右移(compound bitshift right)	a>>=b	等於 a=a>>b

- **運算子的優先順序**

表 3-7 列出各運算子的優先順序排列，優先順序的欄位內數字愈小者表示該運算子的優先順序愈高。

表 3-9、各運算子的優先順序

優先順序	運算子	說明
1	( )	小括號
2	~、!、+、-	補數、反相運算、正符號、負符號
3	++、--	遞增、遞減
4	*、/、%	乘、除、取餘數

5	+、-	加、減
6	<<、>>	左移、右移
7	>、<、>=、<=、==、!=	比較運算
8	&	位元運算－及
9	^	位元運算－互斥或
10	|	位元運算－或
11	&&	布林運算－及
12	||	布林運算－或
13	=	設定

3-4 控制流程

控制流程分成選擇性敘述和迴圈，其中選擇性敘述包括 if、if-else 及 switch-case 敘述，迴圈包括 for、while 及 do…while 等。

- **If 敘述**

If 敘述是我們想要根據判斷結果來執行不同的敘述，其格式如下：

```
If (判斷條件)
    {
      敘述 1;
      敘述 2;
       …..
      敘述 n;
    }
```

If 敘述的主體

當判斷條件為 true 時，就會逐一執行大括號所包含的敘述。若是在 if 敘述主

體中要處理的敘述只有 1 個，可以省略左、右大括號。if 敘述的流程圖如圖 3-3 所示：

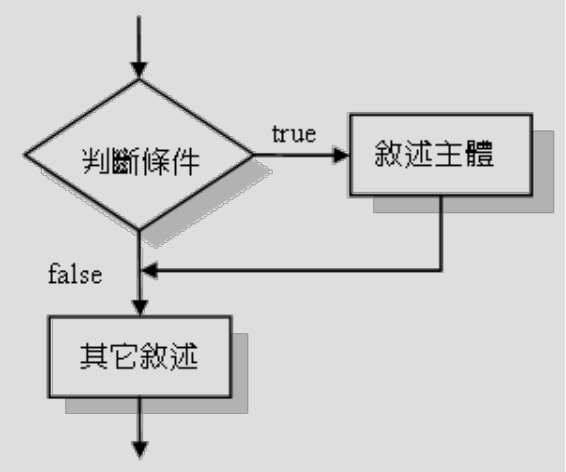


圖 3-3、if 敘述的流程圖

- **If-else 敘述**

當程式中有分岐的判斷敘述時，就可以用 if-else 敘述處理，當若判斷條件成立，及執行 if 敘述主體 1，判斷條件不成立，則執行 else 後面敘述主體 2，if-else 敘述格式如下：

```
If (判斷條件)
  {
   敘述主體 1;  //若判斷條件成立，則執行此部份
  }
else
  {
   敘述主體 2;  //若判斷條件不成立，則執行此部份
  }
```

若是在 if-else 敘述主體中要處理的敘述只有 1 個，可以將左、右大括號去除。if-else 敘述的流程圖如圖 3-4 所示：

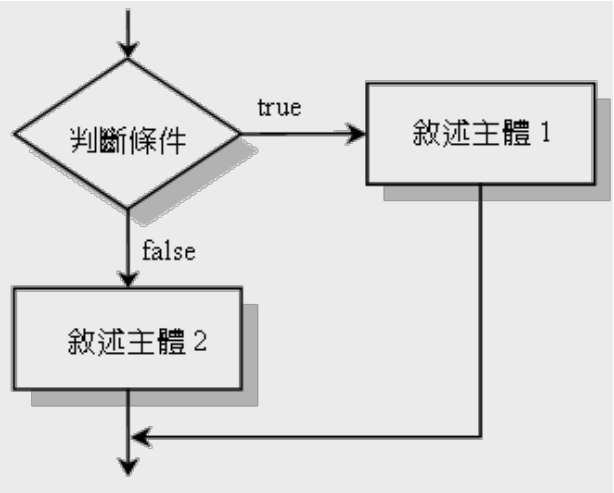


圖 3-4、if-else 敘述的流程圖

- **巢狀 If 敘述**

當 if 敘述中又包含了其他 if 敘述時，這種敘述稱為巢狀 If 敘述（nested if）。巢狀 If 敘述語法格式如下：

```
If (判斷條件 1)
  {
      If (判斷條件 2)
        {
          敘述主體;
        }
        .....
      其他敘述;
  }
```

若判斷條件 2 成立，則執行這個部份

若判斷條件 1 成立，則執行這個部份

巢狀 If 敘述的流程圖如圖 3-5 所示：

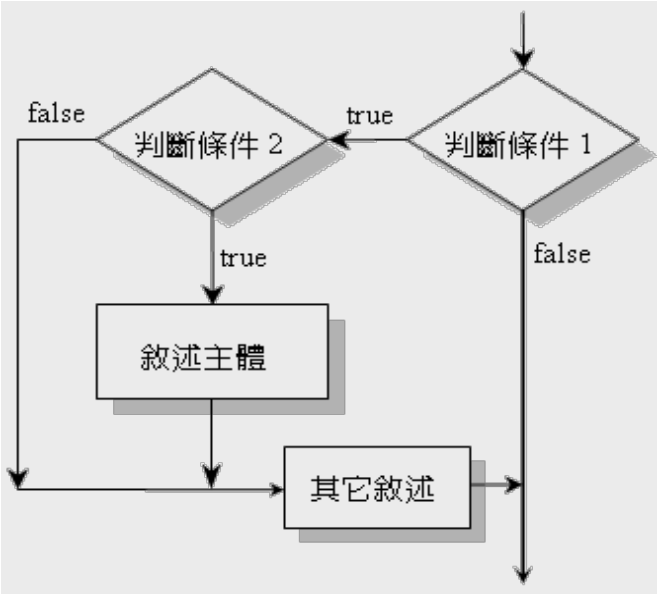


圖 3-5、巢狀 If 敘述的流程圖

- **If-else-if 敘述**

如果 else 敘述主體中緊接另一個 if 敘述，為了簡化程式碼的寫法，可以將 else 及下一個 if 敘述合寫在一起，其格式如下：

```
If (判斷條件 1)
  {                          若判斷條件 1 成立，
     敘述主體 1;              則執行這個部份
  }
else   If (判斷條件 2)
  {                          若判斷條件 2 成立，
     敘述主體 2;              則執行這個部份
  }
```

If-else-if 敘述的流程圖如圖 3-6 所示：

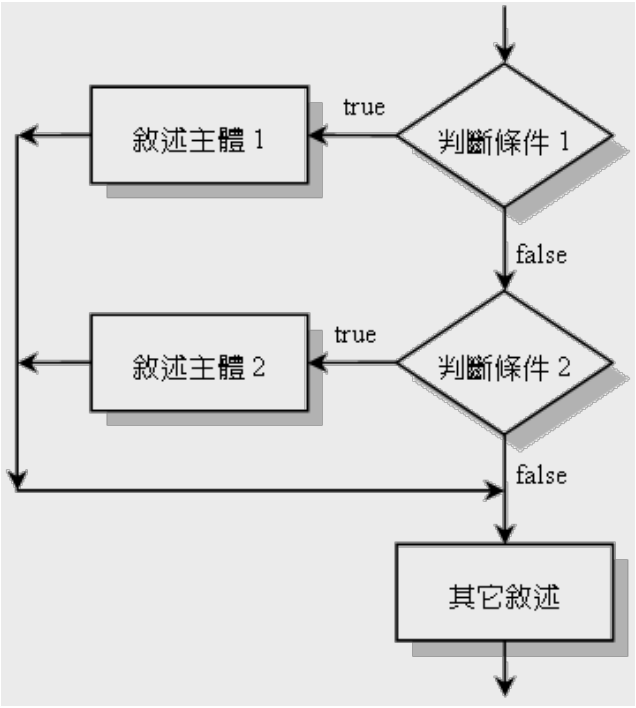


圖 3-6、If-else-if 敘述的流程圖

- **switch-case 敘述**

  switch 敘述可以將多選一的情況簡化，使程式簡潔易讀，其格式如下：

```
switch(運算式)
  {
   case 選擇值 1:
        敘述主體 1;          } 若運算式的值等於選擇值 1，則執行敘述主體 1
        break ;
   case 選擇值 2:
        敘述主體 2;          } 若運算式的值等於選擇值 2，則執行敘述主體 2
        break ;
        ..........
   case 選擇值 n:
        敘述主體 n;          } 若運算式的值等於選擇值 n，則執行敘述主體 n
        break ;
   default :
        敘述主體;            } 若運算式的值選不等於擇值 1~n，則執行敘述主體
  }
```

於 switch 敘述裡，運算式會計算出一個值，此值可能是選擇值 1~n 裡面的任何一個，此時 switch 會根據運算式所計算出的值核對各 case 的選擇值，當相等時在執行相對應之敘述主體。要特別注意的是，於 switch 敘述的選擇值只能是字元或整數。其執行流程：

1. switch 敘述先計算括號中的運算式。
2. 根據運算式之值，檢查是否符合 case 後面的選擇值。如果某個 case 後面的選擇值符合運算式的結果，就會執行該 case 所包含的敘述，直到執行到 break 敘述後，才跳離整個 switch 敘述。
3. 若是所有 case 後面的選擇值街不符合，則執行 default 之後所包含的敘述，執行完畢即離開 switch 敘述。如果沒有定義 default 的敘述，則直接跳離 switch 敘述。

需要注意的，如果忘了在 case 敘述結尾處加 break，則會一直執行到 switch 敘述的尾端，才會離開 switch 敘述，如此可能會造成執行結果的錯誤。switch 敘述的流程圖如圖 3-7 所示：

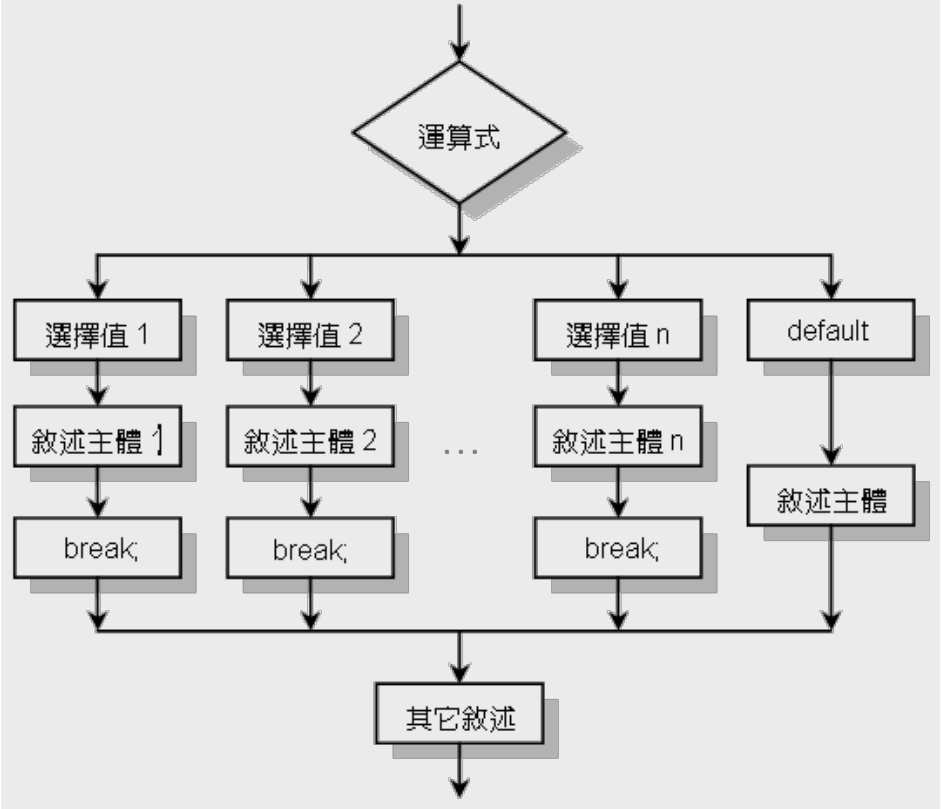


圖 3-7、switch 敘述的流程圖

- **for** 迴圈

  for 迴圈是明確知道某段程式要執行的次數，其敘述格式如下：

```
for(設定迴圈初值;判斷條件;設定增減量)
  {
     迴圈主體 ;
  }
```

若是在 for 迴圈主體中的敘述只有 1 個，則可以將左、右大括號省略，for 迴圈的執行流程如下：

1. 第一次進入 for 迴圈時，便會先執行設定迴圈初值，也就是設定迴圈控制變數的起始值。
2. 根據判斷條件的內容，檢查是否要繼續執行迴圈，當判斷條件值為真（true）時，繼續執行迴圈主體；判斷條件值為假（false）時，則跳離迴圈，執行之後的敘述。
3. 執行完迴圈主體內之敘述後，迴圈控制變數會根據設定增減量執行，更改迴圈控制變數的值，再回到步驟 2 重新判斷是否繼續執行迴圈。

根據上述的程序，其流程圖如圖 3-8 所示。

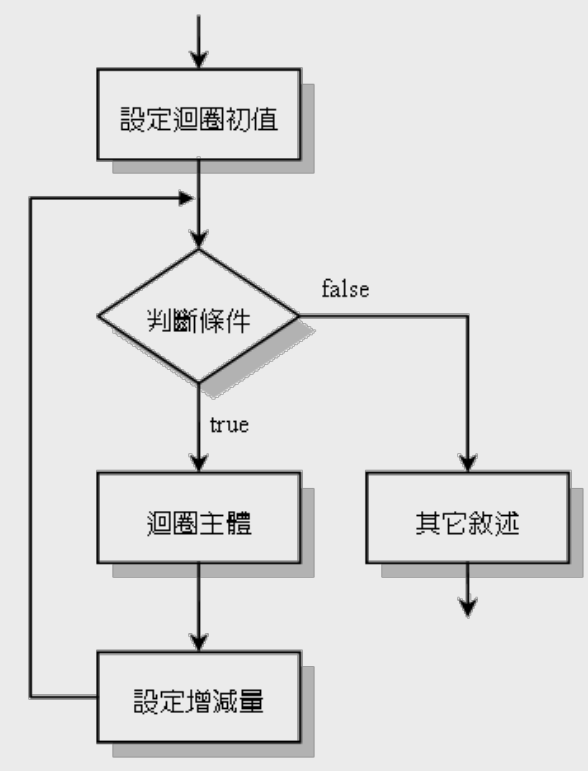


圖 3-8、for 流程圖

範例 1：

```
for(x=0 ; x<10 ; x++)
```

說明：

設定迴圈初值：x=0

判斷條件：x<10

設定增減量：x++

迴圈會執行 10 次

範例 2：

```
for( ; ; )
```

說明：

設定迴圈初值、判斷條件及設定增減量都是空白時，是一無窮盡迴圈

- **while 迴圈**

當迴圈重複執行的次數很確定時，會使用 for 迴圈。但對於有些問題，無法先知道迴圈要執行幾次時，就要考慮使用 while 迴圈或 do...while 迴圈。下面是 while 迴圈的使用格式：

```
設定迴圈控制變數初值;
while(判斷條件)
  {
    迴圈主體 ;
    執行迴圈控制變數的增減量;
  }
```

當 while 迴圈主體中的敘述只有 1 個，則可以將左、右大括號省略。while 迴圈的執行流程如下：

1. 第一次進入 while 迴圈之前，必須先設定迴圈控制變數的起始值。
2. 根據判斷條件的內容，檢查是否要繼續執行迴圈，當判斷條件值為真（true）時，繼續執行迴圈主體；判斷條件值為假（false）時，則跳離迴圈，執行之後的敘述。
3. 執行完迴圈主體內之敘述後，執行迴圈控制變數的增減量，重新設定迴圈控制變數的值，由於 while 迴圈不會主動更改迴圈控制變數的內容，所以在 while 迴圈中，設定迴圈控制變數的工作要由我們自己來作，再回到步驟 2 重新判斷是否繼續執行迴圈。

根據上述的程序，其流程圖如圖 3-9 所示。

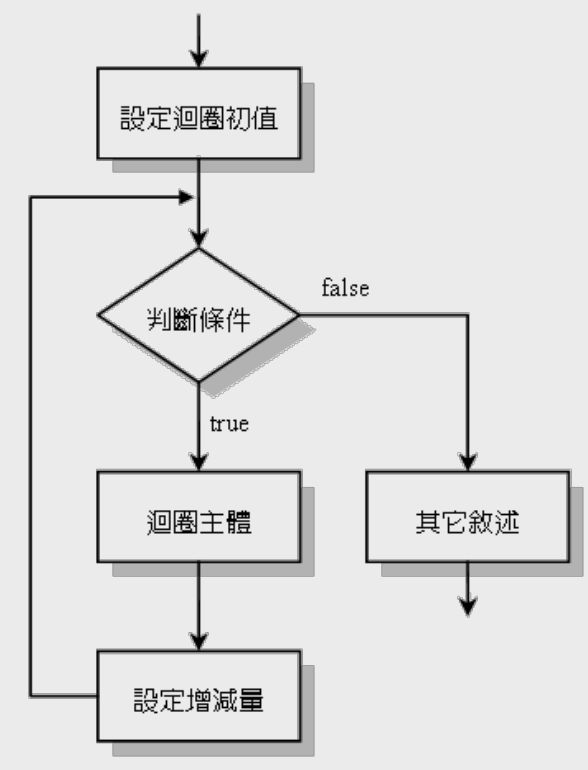


圖 3-9、while 迴圈流程圖

- **for 迴圈與 while 迴圈比較**

如果迴圈的執行次數為已知，則 for 和 while 兩種迴圈都可以使用，表 3-10 為 for 迴圈與 while 迴圈比較。

表 3-10、 for 迴圈與 while 迴圈比較

for 迴圈	while 迴圈

for(設定初值 ; 判斷條件 ; 設定增減量)
{
敘述 1;
敘述 2;
……
敘述 n;
}
	設定初值;
	while(判斷條件)
	{
	敘述 1;
	敘述 2;
	……
	敘述 n;
	設定增減量
	}

- **do....while 迴圈**

  do while 迴圈也是用於迴圈使用次數未知時。至於 while 迴圈及 do while 迴圈最大不同的地方，就是要進入 while 迴圈前，會先執行判斷條件的真假，再決定是否執行迴圈主體，而 do while 迴圈則會先執行迴圈主體一次後，再執行判斷條件的真假，所以不管判斷條件的真假為何，do while 迴圈至少會執行一次迴圈主體，而 while 迴圈的迴圈主體可能一次都不會執行。下面是 do while 迴圈的使用格式：

```
設定迴圈控制變數初值;
do
 {
    迴圈主體 ;
    執行迴圈控制變數的增減量;
  }  while(判斷條件) ;
```

do while 迴圈的執行流程如下：

1. 進入 while 迴圈之前，必須先設定迴圈控制變數的起始值。
2. 直接執行迴圈主體，迴圈主體執行完畢，才開始根據判斷條件的內容，檢查是否要繼續執行迴圈，當判斷條件值為真（true）時，繼續執行迴圈主體；判斷條件值為假（false）時，則跳離迴圈，執行後續的敘述。
3. 執行完迴圈主體內之敘述後，執行迴圈控制變數的增減量，重新設定迴圈控制變數的值，由於 do while 迴圈不會主動更改迴圈控制變數的內容，所以在 do while 迴圈中，設定迴圈控制變數的工作要由我們自己來作，再回到步驟 2 重新判斷是否繼續執行迴圈。

根據上述的程序，其流程圖如圖 3-10 所示。

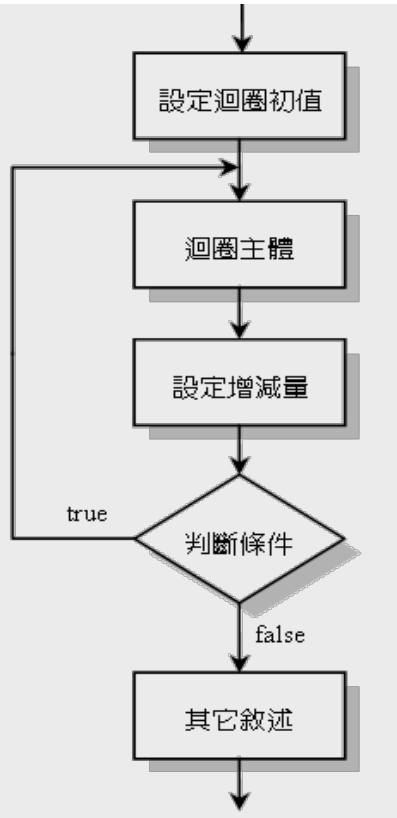


圖 3-10、do while 迴圈流程圖

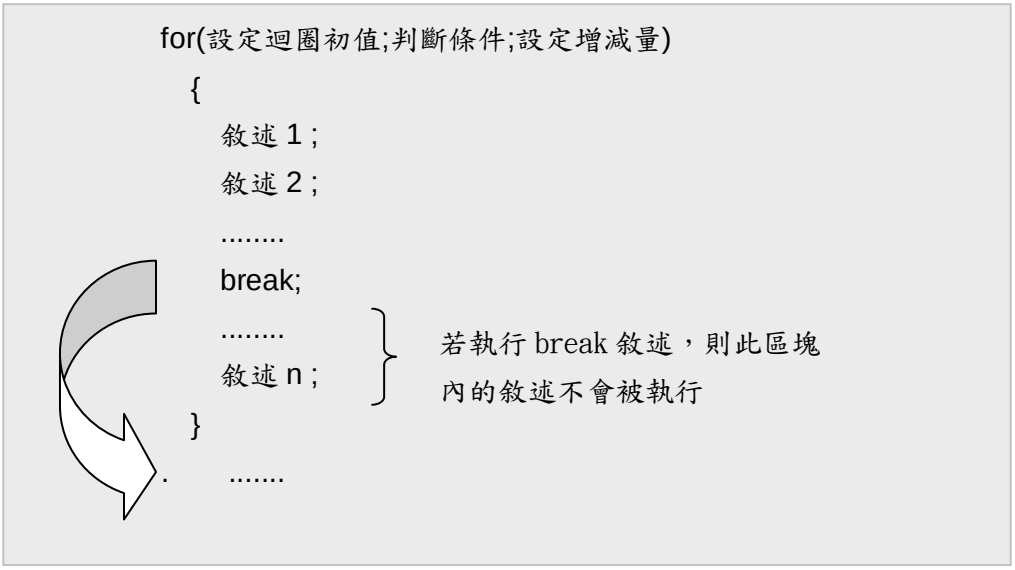
- **使用那一種迴圈**

for 迴圈、while 迴圈及 do while 迴圈這三種迴圈到底使用那一種？這沒有一定的答案，完全視程式的需求而定，表 3-11 為三ㄨ種迴圈的比較，讀者比較使用。

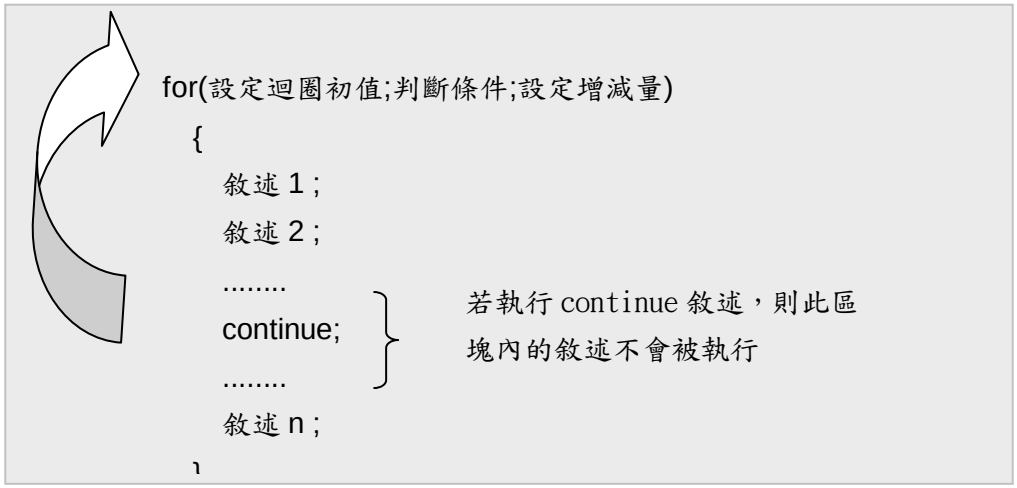
迴 圈 特 性	迴 圈 種 類		
	for	while	do while
前端測試判斷條件	是	是	否
後端測試判斷條件	否	否	是
於迴圈主體中需要更改控制變數之值	否	是	是
迴圈控制變數自動變更	是	否	否
迴圈重複的次數	已知	未知	未知
至少執行迴圈主體的次數	0 次	0 次	1 次
何時重複執行迴圈	條件成立	條件成立	條件成立

- **迴圈跳離－break、continue**

跟 c 程式語言一樣，有一些跳離敘述，如 break、continue、goto 等，在結構化程式設計上，並不鼓勵使用者運用，因為這些跳離敘述會增加除錯及閱讀上的困難，除非在某些不得已的情況才使用。底下介紹 break、continue 兩個敘述。



以上是以 for 迴圈為例，在迴圈主體中有一 break 敘述時，當程式執行到 break，及會離開迴圈主體，到迴圈後的敘述執行。



以上同樣以 for 迴圈為例，在迴圈主體中有一 continue 敘述時，當程式執行到 break，即會回到迴圈的起點，繼續執行迴圈主體。

3-5 陣列與指標

陣列是由一群相同型態的變數所組成的一種資料結構，以一個相同的變數名稱來表示，陣列中各別的元素（element）是以「索引值」（或稱為註標，index），來標示存放的位置。陣列依存放元素的複雜程序，分為一維、二維與二維以上的多維陣列。

- **一維陣列**

一維陣列的宣告格式如下：

資料型態　陣列名稱**[**陣列大小**];**

以下是一維陣列宣告的範例：

```
int led[10] ;       // 宣告整數陣列 led，可存放 10 個元素
float temp[7];      // 宣告浮點數陣列 temp，可存放 7 個元素
char name[12]; // 宣告字元陣列 name，可存放 12 個元素
```

宣告好陣列之後，如果想要使用陣列裡的元素，可以利用陣列的索引值完成。整個陣列好比整個旅館房間，而索引就好像房間的編號，只要根據房間編號（索引值），就能夠找到住宿的客人（儲存於陣列的元素）。需注意是陣列的索引值的編號必須由 0 開始。

如果想直接在宣告時就設定陣列初值，只要陣列的宣告格式後面加上初值設定即可，其宣告格式如下：

資料型態　陣列名稱**[n]={初值 1, 初值 2, … , 初值 n} ;**

設定陣列初值範例如下:

```
int led[4]={0x01,0x02,0x04,0x08} ;  // 宣告整數陣列 led，並設定陣列初值
```

上面範例中宣告一個整數陣列 led，陣列元素有 4 個，大括號裡的初值會分別依序指定給各元素存放，led[0]=0x01、led[1]=0x02、led[2]=0x04 及 led[3]=0x08。

若宣告時沒有將陣列元素的個數列出，編譯器會視所給予的初值個數來決定陣列的大小，下面的設定陣列初值範例：

```
int led[]={0x01,0x02,0x04} ;  // 有 3 個初值，所以陣列 led 的大小為 3
```

- **二維陣列或多維陣列**

多維陣列的宣告格式如下：

資料型態　陣列名稱**[**陣列大小 1**] [**陣列大小 2**].... [**陣列大小 n**];**

一個二維 3x2 整數陣列範例，如下：

```
int num[3][2]={{10,11} , {20,22} , {30,33}};
```

代表 num[0][0]的預設值為 10，num[0][1] 的預設值為 11、....num[2][1] 的預設值為 33。完成宣告後，就可像一般變數一樣操作，

```
x=num[0][1]+5;
```

執行後，a 的內容為 16。

- **指標**

指標是用來存放記憶體位址的變數，其宣告格式如下：

資料型態　*變數名稱 ;

範例如下：

```
int *ptr ;
```

也可以把同型態的變數與指標放在一起宣告：

```
int *ptr1, *ptr2, a, b, c ;
```

指標常用的運算有兩種，一是取出變數的地址，然後存放在指標裡；二是取

出指標變數所指之變數的內容，這兩種工作由位址運算子「&」及依址取值運算子「*」完成。其範例如下：

```
ptr1=&a;
b=*ptr1;
```

第一行中 a 變數的位址被放入 ptr1 指標變數內，

第二行中是將 ptr1 指標所指的位址 a 變數的內容指定給 b，

以上兩行執行的結果等同 b=a 的執行結果。

3-6 特殊符號

以下介紹幾個特殊符號：。

；(分號)：用於結束敘述，忘記以分號結束一行將導致編譯錯誤。例如：a=14;

{}(大括號)：左右括號必須成雙配對，否則會有編譯錯誤發生，主要用於函數、迴圈或條件等敘述，例如：

函數：

```
void myfunction(datatype argument){
   statements(s)
}
```

迴圈：

```
while (boolean expression)
{
    statement(s)
}

do
{
    statement(s)
} while (boolean expression);

for (initialisation; termination condition; incrementing expr)
{
    statement(s)
}
```

條件敘述：

```
if (boolean expression)
{
```

```
    statement(s)
}
else if (boolean expression)
{
    statement(s)
}
else
{
    statement(s)
}
```

//(單行註釋)、/* .... */(單行註釋）：註釋是是用來說明程式的意義或功能。例如：

```
x = 5;   // This is a single line comment. Anything after the slashes is a
comment
             // to the end of the line

/* this is multiline comment - use it to comment out whole blocks of code

if (gwb == 0){     // single line comment is OK inside a multiline comment
x = 3;                  /* but not another multiline comment - this is invalid */
}
// don't forget the "closing" comment - they have to be balanced!
*/
```

#define：使用#define 前置處理器方便將常用的常數、字串替換成一個自定的識別明稱，例如：#define PI 3.1416。

#include：語法為#include <檔頭檔>，編譯器會把這行敘述以整個檔頭檔的內容取代。例如：# include <stdio.h>

3-7 函數

函數是 C 語言的基本模組，函數可以簡化主程式的結構，ardunio 程式把函數分成設計者自訂與系統提供兩類。

- **設計者自訂函數**

函數的架構與主程式的架構類似，不過，函數能傳入引數，也能傳出引數，或將執行結果傳回呼叫程式。函數名稱可由設計者自訂，其基本架構如下所示：

```
void Sub_name(int x)      // Sub_name：函數名稱、int x：傳入引數
 {                        // 函數起始符號
   unsigned char led ;    } 宣告區
   int a, b ;             }
   ....
   led=0xff ;   // led 亮  } 程式區
   ....                   }
}                         // 函數結束符號
```

- **系統提供函數**

Arduino 提供了很多的函數，縮短了使用者於程式設計的時間，底下介紹幾類常用的函數。未介紹的部份請參考 http://arduino.cc/en/Reference/HomePage。

1. 數位 I / O（DIGITAL I/O）

數位 I / O 函數包括 pinMode()、digitalWrite()及 digitalRead()等，

a. pinMode()：指定的引腳(pin) 的配置行為，為輸入或輸出。

語法（Syntax）：

```
pinMode(pin, mode)
```

參數（Parameters）：

pin：你想設置的模式的引腳數

Mode：有 INPUT, OUTPUT, or INPUT_PULLUP 三種。

範例（Example）：

```
int ledPin = 13;                // LED connected to digital pin 13
void setup()
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
```

```
}

void loop()
{
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

b. **digitalWrite**()：寫一個高(HIGH)或低(LOW)的值到數位引腳。

語法：

digitalWrite (pin, value)

參數：

pin：數位輸出引腳數

value：HIGH 或 LOW。

範例如上例。

c. digitalRead()：從指定的引腳數讀取值，其值為高或低。。

語法：

digitalRead (pin)

參數：

pin：數位輸入引腳數

傳回值：

HIGH 或 LOW

範例：

```
// Sets pin 13 to the same value as pin 7, declared as an input.
int ledPin = 13; // LED connected to digital pin 13
int inPin = 7;   // pushbutton connected to digital pin 7
int val = 0;     // variable to store the read value
```

```
void setup()
{
  pinMode(ledPin, OUTPUT);       // sets the digital pin 13 as output
  pinMode(inPin, INPUT);       // sets the digital pin 7 as input
}

void loop()
{
   val = digitalRead(inPin);   // read the input pin
  digitalWrite(ledPin, val);    // sets the LED to the button's value
}
```

2. 類比 I / O（ANALOG I/O）

本節介紹 analogRead ()及 analogWrite() – PWM 類比 I / O 函數。

a. analogRead()：讀取類比輸入引腳數(0 到 5 為大多數實驗部分板、0 到 7 為 Mini 和 Nano, 0 to 15 為 Mega)。

語法：

analogRead (pin)

參數：

pin：類比輸入引腳數

傳回值：

整數(0~1023)

範例：

```
int analogPin = 3;        // potentiometer wiper (middle terminal)
                           //connected to analog pin 3
                           // outside leads to ground and +5V
int val = 0;              // variable to store the value read

void setup()
{
   Serial.begin(9600);              //   setup serial
```

```
}

void loop()
{
   val = analogRead(analogPin);      // read the input pin
   Serial.println(val);              // debug value
}
```

b. analogWrite()：寫入類比值（PWM 波）到引腳。

語法：

analogWrite (pin,value)

參數：

pin：類比輸入引腳數整數(0~1023)

value：工作週期（duty cycle）0（always off）到 255（always on）

傳回值：

無

範例：

```
//Sets the output to the LED proportional to the value read from the potentiometer.

int ledPin = 9;          // LED connected to digital pin 9
int analogPin = 3;    //potentiometer connected to analog pin 3
int val = 0;             // variable to store the read value

void setup()
{
  pinMode(ledPin, OUTPUT);    // sets the pin as output
}

void loop()
{
  val = analogRead(analogPin);    // read the input pin
```

```
  analogWrite(ledPin, val / 4);    // analogRead values go from 0 to 1023,
                                   //analogWrite values from 0 to 255
}
```

3. 延遲時間函數

本節介紹 delay()及 delaymicroseconds()等兩個延時函數。

a. delay()：以參數量為的延遲時間量（以毫秒為單位，1000 等於 1 秒）。

語法：

delay (ms)

參數：

ms：延遲的毫秒數（無符號長整數，unsigned long）

傳回值：

無

範例：

```
int ledPin = 13;                  // LED connected to digital pin 13

void setup()
{
  pinMode(ledPin, OUTPUT);       // sets the digital pin as output
}

void loop()
{
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

b. delaymicroseconds()：以參數量為的延遲時間量（以微秒為單位，1000 等於 1 毫秒）。

語法：

delaymicroseconds (us)

參數：

us：延遲的微秒數（無符號整數，unsigned int），最大值為 16383。

傳回值：

無

範例：

```
int outPin = 8;                 // digital pin 8

void setup()
{
  pinMode(outPin, OUTPUT);      // sets the digital pin as output
}

void loop()
{
  digitalWrite(outPin, HIGH);   // sets the pin on
  delayMicroseconds(50);        // pauses for 50 microseconds
  digitalWrite(outPin, LOW);    // sets the pin off
  delayMicroseconds(50);        // pauses for 50 microseconds
}
```

3-8 習題

1.請說明 Arduino 程式碼的基本架構？

2. arduino 程式語言內定之常數有那些？

3.何謂變數？其資料型態包括那些？

4.請寫出 if 敘述的格式及流程圖?

5. 請寫出 if - else 敘述的格式及流程圖?

6. 請寫出 if - else - if 敘述的格式及流程圖?

7.請寫出 switch - case 敘述的格式及流程圖?

8.請寫出 for 迴圈敘述的格式及流程圖?

9.請寫出 while 敘述的格式及流程圖?

10.請寫出 do...while 敘述的格式及流程圖?

11.比較 for 及 while 之差異？

12. 請比較 for 迴圈、while 迴圈及 do while 迴圈這三種迴圈使用的使用時機?

13.請說明 delay()函數的用法？

參考資料

1. http://arduino.cc/en/Reference/HomePage