Chapter
04

程式的撰寫

學習大綱

- 4-1　高階程式指令應用
- 4-2　程式編寫

學習目標

- 1. 能熟練高階語言的資料型態。
- 2. 能熟練高階語言的運算子。
- 3. 能熟練高階語言的決策指令。
- 4. 能熟練高階語言的迴圈指令。
- 5. 能熟練高階語言的陣列型態。
- 6. 能熟練高階語言的自訂函式。
- 7. 能熟練高階語言的程式編寫。

4-1 高階程式指令應用

Arduino 的指令主要來自 C/C++ 語言的精華，但因其主要功能是控制 I/O，有些語法與資料型態有簡化，並增加一些 I/O 專用指令，以下簡介其資料型態、運算子、決策指令、陣列、自訂函式的精華。

4-1-1 資料型態

資料種類

Arduino 所能處理的資料種類分別有數值 (含整數、長整數及浮點數)、字元與字串等。

▮整數(Integers)

Arduino 可以處理的整數有三種進位方式，分別是十進位 (Decimal)、十六進位 (Hexadecimal) 及八進位 (Octal)。其中十進位則以我們平常書寫數字的方式即可。十六進位應以 0X 或 0x 開頭（數字的 0，不是字母的O），且以 0,1,2,3,4,5,7,8,9,a,b,c,d,e,f 代表 0 到 15，例如 0xa、或 0XB 均爲十六進制 (十六進位的 a,b,c,d,e,f 等字元大小寫都可以)，分別代表十進位的 10 與 11；八進位則應以 0 開頭（數字的 0），例如，072 則爲八進位，等於十進位的 58。

▮浮點實數(Floating-point literals)

數字中含有小數點或指數的稱爲浮點數、實數或浮點實數。浮點數可使用標準寫法或科學符號法表示，例如 321.123 即爲標準寫法，1.23e+4 即爲科學符號表示法。以指數爲例，E 或 e 表示 10 的次方，例如 0.0023、2.3E-3 及 2.3e-3 都是表示相同的浮點數;又例如 2.3E+2 則代表 230。但是，Arduino 的主要功能是 I/O 控制，不是數值運算的處理器，實數的表示與運算誤差很大。請自行鍵入以下程式，並觀察執行結果。

```
float a=2.3E-3;
Serial.begin(9600);
Serial.println(a);
```

▮字元(Character literals)

使用單引號『'』圍住的單一字元，稱爲字元，例如 'A' 或 'a' 等。

▮字串

Arduino 與 C 語言相同，字串以雙引號『" "』所圍住，例如 ,"Gwosheng"," 台灣 " 等。

▮布林值

C 語言使用 1 或 true 代表布林的 true，0 或 false 代表布林的 false。

資料型態

電腦爲了有效率的處理資料，就有資料型態（Data Types）的規劃，也就是大的資料用大盒子裝，小的資料用小盒子裝，如此才可節省記憶體，並加快處理效率。反過來說，若不分資料大小，通通用大盒子裝資料，那將會非常浪費記憶體，也拖垮執行效率。例如，所有的東西都用冰箱的盒子裝當然可以，但這樣非常浪費空間，還有，搬運時也很耗時。Arduino 所提供的資料型態、所佔用記憶體、所能代表的數值範圍如表 4-1：

▶ 表 4-1　Arduino 資料型態

資料型態	中文名稱	佔用記憶體的大小（位元）	所能代表的數值的範圍	備註
char	字元	8	-128 ～ 127	
byte	位元組	8	0 ～ 255	
int	整數	16	-32768 ～ 32767	
long	長整數	32	-2147483648 ～ 2147483647	
float	浮點數	32	+/-3.4E+-38	
double	倍精度浮點數	32	+/-3.4E+-38	Arduino 的 double 同 float
unsigned char	正字元	8	0 ～ 255	
unsigned int	正整數	16	0 ～ 65535	
unsigned long	正長整數	32	0 ～ 42949667295	

▶ 表 4-1 Arduino 資料型態（續）

資料型態	中文名稱	佔用記憶體的大小（位元）	所能代表的數值的範圍	備註
bool	布林	8	true or false	boolean 也可，但不鼓勵
String	字串			

自我練習

1. 請線上查詢 Arduino 資料型態。

變數宣告

變數的功能是用來輸入、處理及儲存外界的資料，而變數在使用以前則要事先宣告才可使用。Arduino 語言的變數宣告語法如下：

```
資料型態 變數名稱[=初值];
```

例如：

```
byte a;
```

即是宣告變數 a 爲 byte 型態，佔用 1 個 Byte，此種型態僅能儲存 0 到 255。變數的宣告亦可連同初值一起設定，如以下敘述：

```
float d = 30.2;
```

以下敘述，同時宣告兩個變數，且設定其初值。

```
int f1=3,f2=3;
```

以下敘述可宣告布林型態:（布林型態用來表示，運算結果的『眞』與『僞』）

```
bool g=true;
```

C/C++/Arduino 都可用字元陣列表示字串，以下程式可宣告 a 爲字串型態，請留意字元是單引號，字串是雙引號。

```
char a[]="ABC";//字串用雙引號
```

C++/Arduino 才有字串型態 String，以下程式可宣告變數 h 爲字串型態：

```
String h="123";
```

以下宣告 a 爲一維陣列：

```
byte a[8];
```

這樣就可使用 a[0]~a[7] 等 8 個變數。以下陣列宣告的同時，直接給予初值。

```
byte a[]={0x01,0x6,0x5b,0x4f,0x66,0x6d,0x7d,0x7,0x7f};
Serial.println(a[0]);
```

變數經過宣告之後，編譯器即會根據該變數的資料型態配置適當的記憶體儲存此變數，所以若要提高程式的執行效率，則應儘量依照資料性質，選擇佔用記憶體較小的資料型態。

變數的有效範圍

任一變數的宣告，若無特殊聲明，均屬於區域變數，其有效範圍僅止於該變數所在的程式區塊。所以，以下變數 i 的有效範圍僅在 setup()，無法在 loop(){} 函式內存取。

```
void setup() {
  Serial.begin(9600);
  byte i=1;
}
void loop() {
  Serial.println(i);
}
```

其次，請比較圖 4-1a 與圖 4-1b 的差別，圖 4-1a byte i=0; 放在 void loop() {} 裡面，則每次執行 void loop() {} 時，byte i=0 都被執行，所以其值永遠都相同，沒有累加效果。此時就要把此敘述放在外面，成爲全域變數，所以此 i，稱爲計數器，也可以想像成 loop() 被執行的次數，如圖 4-1b。

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  byte i=0;
  i=i+1;
  Serial.println(i);
}
```

圖 4-1a　i 為區域變數

```
void setup() {
  Serial.begin(9600);
}
byte i=0;
void loop() {
  i=i+1;
  Serial.println(i);
}
```

圖 4-1b　i 為全域變數

4-1-2 運算子

所謂運算子（Operator），指的是可以對運算元（Operand）執行特定功能的特殊符號。例如：3＋2的『3』與『2』稱爲運算元，『+』稱爲運算子。Arduino的運算子分爲五大類，分別是：算術（Arithmetic）運算子、比較（Comparison）運算子、布林（Boolean）運算子、位元操作（Bitwise）運算子及複合（Compound）運算子，分別說明如下：

算術運算子

下表是Arduino的算術運算子。（請開啓https://www.arduino.cc/reference/en/）

Arithmetic Operators

% (remainder)

* (multiplication)

+ (addition)

- (subtraction)

/ (division)

= (assignment operator)

圖 4-2　Arduino 算術運算子

以上算術運算子用來執行一般的算術運算，包括指派 (=)、取正負數 (+/-)、加 (+)、減 (-)、乘 (*)、除 (/)、取餘數 (%) 等，表 4-2 是以上算術運算子的功能說明：

▶ 表 4-2　Arduino 算術運算子

運算子	定義	優先順序	結合律
=	指派	15	由右至左
+/-	正負號，一元運算子	2	由右至左
*	乘法運算	4	由左至右
/	除法運算	4	由左至右
%	求餘數（Modulus）	4	由左至右
+/-	加法 / 減法運算	5	由左至右

四則運算

以下是一些簡單四則運算：

```
int a=5,b=4;
Serial.println(a+b);//9
Serial.println(a-b);//1
Serial.println(a*b);//20
Serial.println(a%b);//1 取餘數
```

以上程式，請放在 setup() 裡面，就可觀察執行結果。

```
void setup() {
  Serial.begin(9600);
  //執行一次的放這裡
  int a=5,b=4;
  Serial.println(a+b);//9
  Serial.println(a-b);//1
  Serial.println(a*b);//20
  Serial.println(a%b);//1 取餘數
}
void loop() {}//重複執行的放loop()函式裡面，本例雖然沒用到，也不能省略。
```

整數除法或實數除法

Arduino/C/C++ 的除法運算，只有被除數與除數的型態均爲整數，才是整數除法，商的型態爲整數；否則即爲實數除法，得到實數商。例如：

```
int x=5, y=4,z;
float xf=5,yf=4;
Serial.println(x/y); // 1，被除數與除數的型態均爲整數
Serial.println(xf/y);//1.25
Serial.println(x/yf);//1.25
Serial.println(xf/yf);//1.25
```

比較運算子（Comparison Operators）

比較運算子又稱爲關係（Relational）運算子，用於資料之間的大小比較，比較的結果可得到 bool 型態的 1 (true) 或 0 (false)，以作爲以上決策『運算式』運算依據。表 4-3 是 Arduino 語言中的關係運算子符號，這些都和 C/C++ 相同。

▶ 表 4-3　Arduino 比較運算子

運算子	定義	優先順序	結合律
<	小於	7	由左至右
>	大於	7	由左至右
<=	小於等於	7	由左至右
>=	大於等於	7	由左至右
==	等於	8	由左至右
!=	不等於	8	由左至右

例如：

```
int a=5,b=3;
float c=5;
Serial.println(a>b);//1
Serial.println(a>=b);//1
Serial.println(a==b);//0
Serial.println(a!=b);//1
Serial.println(a=!b);//0 小心不要打錯，且沒有錯誤信息
Serial.println(a==c);//0 變數型態要相同才能比較
Serial.println(c>b);//1 變數型態要相同才能比較，但這次勉強正確
Serial.println(a=b);//3  單個等號是指派，請小心
```

布林運算子（Boolean Operators）

布林運算子又稱邏輯（Logical）運算子。當同一個運算式要同時存在兩個以上的比較運算時，則每兩個比較運算子之間必須使用布林運算子連結。例如，您要找『男生』且『年齡大於 40』，此一決策就同時含有兩個比較運算式，此時就要運用布林運算子連結。Arduino 布林運算子如表 4-4 所示：

▶ 表 4-4　Arduino 布林運算子

運算子	定義	優先順序	結合律
!	布林邏輯 not 運算	2	由右至左
&&	布林邏輯 and 運算	12	由左至右
||	布林邏輯 or 運算	13	由左至右

例如：

```
int a=5,b=3;
Serial.println(!(a>b));  //0  Arduino用0表示false
Serial.println((a>b) && (b>=4));//0 Arduino用0表示false
Serial.println((a>b) || (b>=4)); //1 Arduino用1表示true
```

又例如，要判斷 x 是否滿足 1<x<=6，則敘述如下：

```
int x=5;Serial.println((x>1) &&(x<=6));//1
int x=8;Serial.println((x>1) &&(x<=6));//0
```

自我練習

1. 請寫一程式，任意指派 -10 到 10 的整數 x，若此整數 x 滿足 -3<x<=6，請輸出『1』。
2. 請寫一程式，任意指派 2 到 12 的整數 x，若此整數 x 滿足 x>=8 或 x<4，請輸出『1』。
3. 請寫一程式，可以指派三角形三邊長 a, b, c，若能圍成三角形，則輸出『1』。提示：圍成三角形的條件是，任兩邊之和要大於第三邊，數學語言是：a+b > c and a+c > b and b+c > a，請寫程式完成以上判斷。

位元（Bitwise）運算子

位元（Bitwise）運算子是將一個 8 位元的整數逐位元進行運算，此類運算子還可以分爲兩類：位移（Shift）運算子與布林運算子。位移運算子可以用來將各個位元向左或是向右移；布林運算子則可以逐位元進行布林運算。表 4-5 是 Arduino 語言位元操作運算子的列表：

▶ 表 4-5　Arduino 位元運算子

運算子	定義	優先順序	結合律
~	位元 not 運算	2	由右至左
&	位元 and 運算	9	由左至右
^	位元 xor 運算	10	由左至右
|	位元 or 運算	11	由左至右
<<	逐位元向左位移	6	由左至右
>>	逐位元向右位移	6	由左至右

運算子 ~

『~』是位元 not 運算，not 是將位元 0 變 1，1 變 0，運算結果如表 4-6：

▶ 表 4-6　運算子 not 真值表

a	c=~a
1	0
0	1

例如：

```
byte a=1,b=0;
Serial.println(~a);// -2
```

a=1，分解爲 2 進位是 00000001

```
~a= not 00000001=11111110
```

首位元是 1 表示負數，那到底負多少，再取 2 補數。2 補數的步驟是先取 1 補數再加 1。所謂 1 補數是『逐位元將 1 變 0，0 變 1』，所以上面 11111110 的 1 補數是：

```
00000001
```

將 1 補數再加 1，就是 2 補數，上面 00000001 加 1 就是 00000010，其大小是 2。上面符號既然是『-』，那就代表此數是 -2，此即爲二補數的觀念。

運算子 &

『&』是位元 and 運算，and 運算眞值表如表 4-7，當位元 a 與 b 同時爲 1，c 才得到 1：

▶ 表 4-7　運算子 & 真值表

a	b	c= a & b
0	0	0
0	1	0
1	0	0
1	1	1

例如：

```
byte a=1,b=5;
Serial.println(a&b);//00000001 & 00000101=00000001=1
```

運算子 ^

『^』是位元 xor 運算，xor 運算眞值表如表 4-8，當位元 a 與 b 不相同時，c 才得到 1。

▶ 表 4-8　運算子 ^ 真值表

a	b	c=a^b
0	0	0
0	1	1
1	0	1
1	1	0

例如：

```
byte a=1,b=5;
Serial.println(a^b);//00000001 ^00000101=00000100=4
```

運算子 |

『|』是位元 or 運算，or 運算眞值表如表 4-9，當位元 a 與 b 有一爲 1，c 就得到 1。

▶ 表 4-9　運算子 | 真值表

a	b	c=a | b
0	0	0
0	1	1
1	0	1
1	1	1

例如：

```
byte a=1,b=5;
Serial.println(a|b);//00000001 | 00000101 =00000101=5
```

運算子 <<

『<<』是左移運算子，例如：

```
byte a=1;
Serial.println(a<<1);//2   1向左移爲1位元，結果是2
a=1;
Serial.println(a<<2);//4   1向左移爲2位元，結果是4
```

運算子 >>

『>>』是右移運算子，例如：

```
byte a=4;
a=a>>1;
Serial.println(a);//2    4向右移爲1位元，結果是2
```

運算子的優先順序（Precedence）

同一敘述，若同時含有多個運算子，此時即需定義運算子的優先順序。例如：

```
a=3+4*2;
```

『=』優先順序是12，『+』優先順序是5 ，『*』優先順序是4，所以運算順序是：

```
(a=(3+(4*2)));
```

運算子的結合律（Associativity）

當同一敘述，相鄰的運算子擁有相同優先順序的運算子，此時即需定義運算子是左結合或右結合。例如：

```
x=a-b-c;
```

連續兩個減號『-』，優先順序相同，此時就要靠定義結合律，減法結合律是由左至右，所以以上同義於：

```
x=((a-b)-c);
```

而

```
x=y=z=2;
```

連續三個指派運算子『=』，指派運算子的結合律是由右至左，所以以上式子同義於：

```
(x=(y=(z=2)));
```

所以，以上敘述，x、y、z 的結果都是 2。

各種進位制

我們人類習慣使用 10 進制，逢 10 填 0 進 1，例如，$(242)_{10}$ 是表示 $2*10^2 + 4*10^1 + 2*10^0 = 242$；若是 8 進位，那就是逢 8 填 0 進 1，僅用 0, 1, 2, 3, 4, 5, 6, 7 等 8 個數字，所以 $(11)_8 = 1*8^1 + 1*8^0 = (9)_{10}$；若是 2 進位，那就是逢 2 填 0 進 1，僅用 0, 1 兩個數字，所以 $1011 = 1*2^3 + 0*2^2 + 1*2^1 + 1*2^0 = 11$；若是 16 進位，那就用 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F 表示 0 到 15，且逢 16 填 0 進 1，所以 $(A2E)_{16} = 10*16^2 + 2*16^1 + 14*16^0 = (302)_{10}$。

Arduino語言可以處理的整數有四種進位方式，分別是十進位（Decimal）、二進位（Binary）、八進位（Octal）及十六進位（Hexadecimal）。其中十進位則以我們平常書寫數字的方式即可，例如12；二進位則以B開頭，例如，B11則代表十進位的3；八進位則應以0開頭，例如011代表十進位的9(1*8+1)；十六進位應以0x開頭，例如，0x11代表十進位的17(1*16+1)。請鍵入以下程式，並觀察執行結果。

```
void setup() {
    Serial.begin(9600);
    int a=242;
    int b=B11;
    int c=011;//數字0，不是字母O
    int d=0x11;//數字0開頭，不是字母O
    Serial.println(a);
    Serial.println(b);
    Serial.println(c);
    Serial.println(d);
}void loop() {}
```

2進位與16進位

前面資料的數位化已經介紹，處理機是以2進位儲存數值，所以若以8位元儲存一個正整數，例如：

```
5
```

就會以

```
00000101
```

表示，我們若以LED觀察結果就是『滅滅滅滅滅亮滅亮』，反過來說，若有8顆LED呈現：

```
滅滅滅滅滅亮滅亮
```

要將此現象數位化，我們也可用2進位表示爲：

```
B00000101
```

其次，以上 2 進位有點長，所以我們習慣以 16 進位表示爲：

```
0x05
```

也就是在自動控制的領域裡，我們會習慣以 2 進位或 16 進位來表示一些燈號或控制結果，請讀者要慢慢習慣這種 2 或 16 進位表示方式。表 4-10 是一些常用數字的 16 進位書寫表示方式。

▶ 表 4-10　10 進位與 16 進位對照表

10進位	2進位	16進位	10進位	2進位	16進位
0	B0	0x0	11	B1011	0xb
1	B1	0x1	12	B1100	0xc
2	B10	0x2	13	B1101	0xd
3	B11	0x3	14	B1110	0xe
4	B100	0x4	15	B1111	0xf
5	B101	0x5	16	B10000	0x10
6	B110	0x6	17	B10001	0x11
7	B111	0x7	18	B10010	0x12
8	B1000	0x8	127	B01111111	0x7f
9	B1001	0x9	254	B11111110	0xfe
10	B1010	0xa	255	B11111111	0xff

自我練習

1. 若有 8 個燈號連續排列，且其燈號是『滅亮亮滅亮亮亮亮』，請問該如何以 16 進制數字回報。

亂數的產生

Arduino 產生亂數是使用 random() 方法，其語法如下：

```
random(min,max)
```

含 min，但不含 max，例如：以下程式產生 1 到 6 的整數亂數。

```
a=random(1,7);
```

min 可省略，若省略表示從 0 開始。例如：以下程式產生 0 到 6 的整數亂數。

```
a=random(7)
```

但使用前，要將 A0 腳位空接，使用以下程式起始亂數種子。（使用 A0 的雜訊得到不同的亂數起點）

```
randomSeed(analogRead(A0));
```

以下程式，可每秒產生 1 個 1 到 6 的整數亂數。

```
void setup() {
  randomSeed(analogRead(A0));
  Serial.begin(9600);
}
void loop() {
  int x=random(1,7);// 產生1個1到6的整數亂數
  Serial.println(x);
  delay(1000);
}
```

4-1-3 決策指令

人類的生活必須不斷面對決策問題，連一個不到三歲的小孩，也常要思考他手裡的十元是要坐電動車還是買棒棒糖。程式語言是協助人類解決問題的工具，當然也有決策流程敘述，Arduino 語言依決策流程點的多寡，分爲以下兩種決策流程敘述，第一是雙向分歧決策流程 if ~else~，第二是多向分歧決策流程的 switch...case，分別說明如下：

if...else

在日常生活領域中，常出現“假如～則～，否則～”時，此種決策流程模式有兩種解決問題的方案，故稱爲雙向分歧決策流程，此時可使用 if...else 敘述。if...else 敘述的語法如下：

```
if（運算式）
{
```

```
    敘述區塊1；
}
[else
{
    敘述區塊2；
}]//語法中的中括號[]，表示裏面內容可省略
```

以上語法流程圖如圖 4-3。

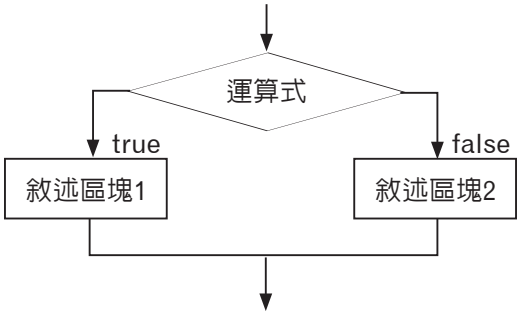


圖 4-3　if~else 流程圖

例如，若有流程圖如圖 4-4：

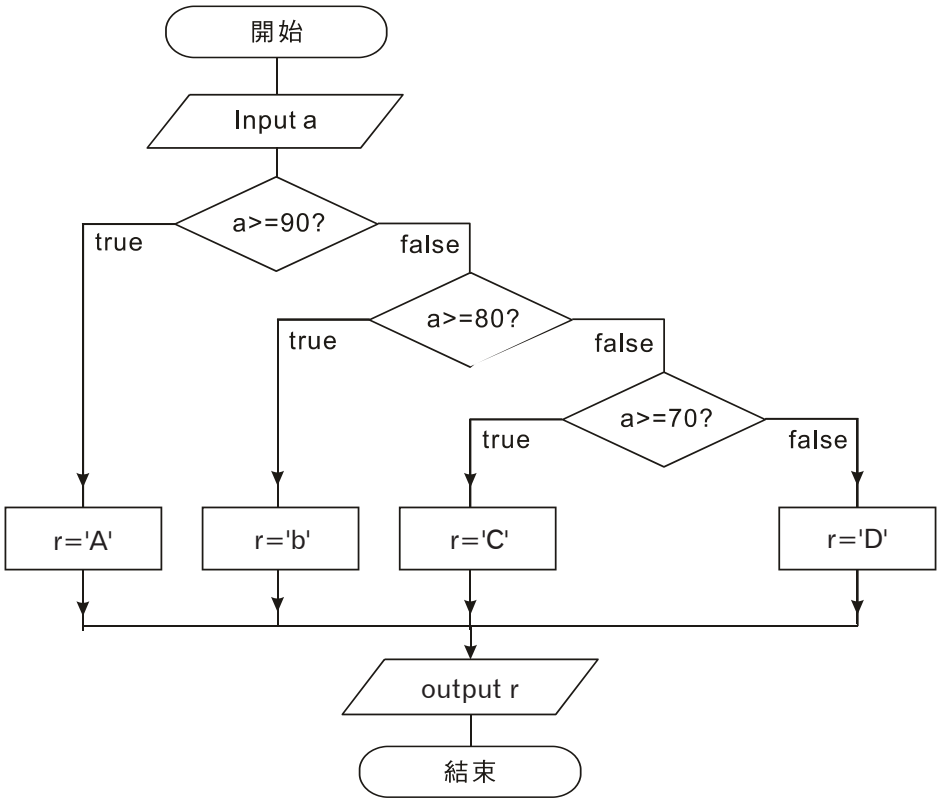


圖 4-4　成績判斷流程圖

請鍵入以下程式，寫出執行結果。

```
1.  void setup() {
2.      Serial.begin(9600);
3.  }
4.  void loop() {
5.      int a;
6.      char b;
7.      Serial.print("Input a:");
8.      while(Serial.available() ==0) {}//等待使用者由鍵盤輸入資料
9.      a=Serial.parseInt(); //使用鍵盤輸入一個整數
10.     Serial.println(a);
11.     if(a>=90)                   /* 高於90分爲A */
12.         b='A';
13.     else
14.         if(a>=80)               /* 介於 80與90分爲B */
15.             b='B';
16.         else
17.             if(a>=70)           /* 介於 70與80分爲C */
18.                 b='C';
19.             else
20.                 b='D';          /* 不符合上述情況則爲D */
21.     Serial.print("The level is:");
22.     Serial.println(b);
23. }
```

程式說明

所有程式語言當需要使用者輸入資料時，都會原地等待使用者輸入，直到得到資料再往下執行，但是單晶微處理機不一樣，不能原地等待，因爲單晶微處理機要同時偵測與執行很多周邊設備，所以不能不做事而原地等待，若一定要單晶微處理機原地等待，則可使用 available()，也就是若沒有資料備妥，則持續等待。

自我練習

1. 請寫一個程式，每秒產生 1 個 -3 到 3 的亂數，且評判其爲負數、0、或正數。

2. 請寫一個程式，完成以下要求：
   (1) 可由鍵盤輸入一個 0 ～ 25 的整數。
   (2) 當此數大於等於 20 時，輸出五個燈。
   (3) 當此數是 16 ～ 19 時，輸出四個燈。
   (4) 當此數是 11 ～ 15 時，輸出三個燈。
   (5) 當此數是 0 ～ 10 時，輸出兩個燈。
3. 心算練習。請產生與輸出 2 個 1 位數亂數，由使用者輸入相加結果，微處理機判斷是否正確。

switch...case

一個決策點若同時擁有三個或三個以上的解決方案，則稱此爲多向分歧決策。多向分歧決策雖也可使用前面巢狀 if else 解決，但卻增加程式的複雜度及降低程式可讀性，若此一決策點能找到適當的運算式，能使問題同時找到分歧點，則可使用 switch case 敘述。switch case 語法如下：

```
switch（運算式）
  {
    case常數1：
         敘述區塊1；
         break;
    case常數2：
         敘述區塊2；
         break;
    case常數3：
         敘述區塊3；
         break;
     〔default:
         敘述區塊n；]//語法中的中括號[]，表示裏面可省略
  }
```

以上語法說明如下：

1. switch 的運算式值僅能爲整數或字元。
2. case 的常數僅能整數或字元，且其資料型態應與上面的 switch 運算式相同。

3. 處理機將會依 switch 的運算式值，逐一至常數 1、常數 2 尋找合乎條件的 case，並執行相對應的敘述區塊，直到遇到 break 敘述，才能離開 switch。
4. default 可放置特殊情況，也就是沒有適當的 case，則執行 default。若省略 default，且若沒有任何 case 滿足 switch 運算式，則程式會默默離開 switch 敘述。（備註：語法中，兩旁加中括號表示此敘述可省略。）
5. 敘述區塊可放置任何合法的敘述，當然也可放置 switch 或 if。
6. 以下敘述，可將 1、2、3、4 轉爲對應的季節。

```
1. void setup() {
2.     Serial.begin(9600);
3.     byte a=1;
4.     String b="";
5.     switch(a)
6.     {
7.         case 1:
8.           b="Spring";         /*春*/
9.           break;
10.        case 2:
11.          b="Summer";         /*夏*/
12.          break;
13.        case 3:
14.          b="Fall";           /*秋*/
15.          break;
16.        case 4:
17.          b="Winter";   /*冬*/
18.          break;
19.      default :
20.          b="input error";
21.    }
22.    Serial.println(a);
23.    Serial.println(b);
24. }void loop() {}
```

7. 有些語言可用逗號將兩種 case 放在一起，但在 C/C++、Arduino 語言中每一 case 僅能放置一個常數，所以若兩個或兩個以上 case，有相同的處理方法，則應將兩個 case 分成兩個敘述，請看以下範例。

8. 以下以 switch case 重作以上使用 if else 的成績判斷。

```
1. void setup() {
2.     Serial.begin(9600);
3.     byte a=98;
4.     String b="";
5.     //本例將分數除以10，得到0~10的整數，所以適用多重分歧
6.     switch(a/10) {
7.        case 10:
8.        case 9:
9.           b="A";
10.          break;//do not leave out
11.      case 8:
12.          b="B";
13.          break;
14.      case 7:
15.          b="C";
16.          break;
17.      case 6:
18.      case 5:
19.      case 4:
20.      case 3:
21.      case 2:
22.      case 1:
23.      case 0:
24.          b="D";
25.          break;
26.  }
27.    Serial.println(a);
28.    Serial.println(b);
29.}void loop() {}
```

自我練習

請以 switch case 重做 4-1-3 節的自我練習第 1 與第 2 題。

4-1-4 迴圈指令

Arduino 有 for 與 while 迴圈指令，用於解決重複的工作，其使用時機與差別，請看以下本單元說明。

for 迴圈

for 迴圈是用於程式設計階段就知道重複執行的次數。例如，您想輸出 1 2 3 4 5 6，可以撰寫程式如下：

```
void setup(){
  Serial.begin(9600);
  for(int i=1;i<=6;i=i+1){
    Serial.print(i);Serial.print(",");
  }
}void loop() {}
```

1. 第一個『1』稱爲起始值；第 2 個『i<=6』，稱爲迴圈執行的條件；第 3 個『i=i+1』，是每次遞增或遞減量，i 先加 1，再放回 i，所以每次遞增 1。其次，因爲 i=i+1 這種運算使用非常頻繁，所以也可以寫成 i++，例如，以上程式也可寫成：

```
for(int i=1;i<=6;i++){
    Serial.print(i);Serial.print(",");
}
```

2. 以下程式可以輸出 2 4 6 8 ，每次遞增 2。

```
for(int i=2;i<=8;i=i+2){//每次遞增2
    Serial.print(i);
}
```

3. 以下左邊程式，『i=8』不行 ，因爲「=」是指派運算子，此與比較運算子「==」不同，會產生編譯錯誤；後面「i==8」是迴圈結束條件不同，可以通過編譯，但結果與「i<=8」不同，請鍵入以下程式，就會明瞭。

```
for(int i=2;i=8;i=i+2){
    Serial.print(i);
}
```

```
for(int i=2;i==8;i=i+2){
    Serial.print(i);
}
```

4. 以下程式可以輸出 2 4 6，請留意『<』與『<=』都可以，只是範圍不同。

```
for(int i=2;i<8;i=i+2){
    Serial.print(i);
}
```

5. 以下左邊與右邊都是由大而小遞減輸出 6 5 4 3 2 1，『i--』是『i=i-1』的複合運算子。

```
for(int i=6;i>=1;i=i-1){
    Serial.print(i);
  }
```

```
for(int i=6;i>=1;i--){
    Serial.print(i);
  }
```

6. 以下程式每次遞減 2 輸出，請留意『遞減』時不等式的方向，『>=』表示資料是由大到小。

```
for(int i=8;i>=1;i=i-2){
    Serial.print(i);//8642
}
```

7. 遞增不等式的方向是『<=』，遞減不等式的方向是『>=』。以下程式不等式方向通通錯了，就通通沒有輸出資料。不等式的方向很好記，因爲方向就代表遞增或遞減。

```
for(int i=1;i>=8;i=i+2){
    Serial.print(i);
}//i<=8才有結果
```

```
for(int i=8;i<=1;i=i-2){
    Serial.print(i);
}//i>=1才有結果
```

8. 迴圈的執行範圍是以兩個大括號『{}』表示，如以上程式都有大括號。若省略大括號，那就默認只執行後續一個敘述。例如，下圖左與右效果都相同。

```
for(int i=1;i<=6;i++){
    Serial.print(i);
  }
```

```
for(int i=1;i<=6;i++)
    Serial.print(i);
```

9. for 主要用來實現循序法。例如，人類計算 52*138 是使用直式乘法，但電腦可不用如此麻煩，因爲電腦有超強計算能力，所以就直接一個一個累加，程式如下：

```
void setup() {
  Serial.begin(9600);
  int s=0;
```

```
  int a=52, b=138;
  for (int i=1 ;i<=a;i++)
    s=s+b;
  Serial.println(s);
} void loop() {}
```

範例 4-1-4a

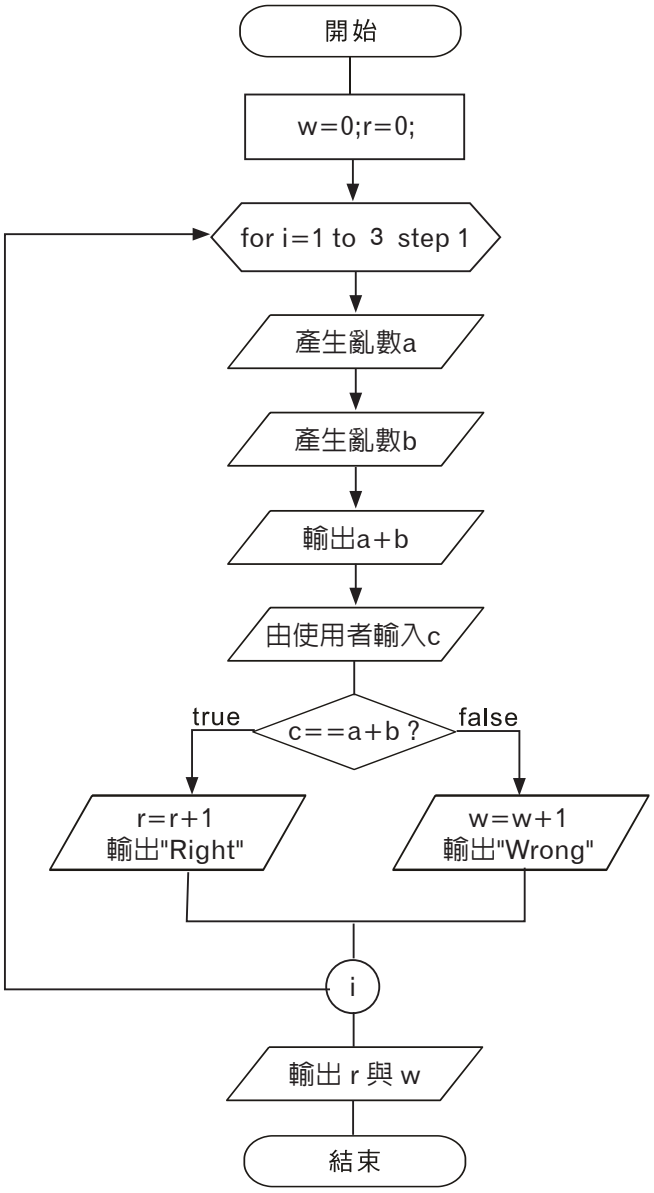
心算練習。請產生與輸出 2 個 1 位數亂數 3 次，由使用者輸入相加結果，微處理機判斷是否正確，且統計使用者答對的題數。

執行結果

```
COM5
1+2=
3
Right
7+3=
0
Wrong
6+5=
11
Right
Right=2
Wrong=1
```

程式列印

1. 以上程式的流程圖如右：



2. 根據以上流程圖，程式撰寫如下：

```
1.  void setup() {
2.    Serial.begin(9600);
3.    randomSeed(analogRead(A0));//播亂數種子，這樣每次才能得到不同亂數
4.    int a,b,c;
5.    int r=0;
6.    int w=0;
7.    for (int i=1;i<=3;i++){
8.        a=random(1,10);//產生1～9整數亂數
9.        b=random(1,10);
10.       //輸出 a+b=
11.       Serial.print(a);Serial.print("+");
                          Serial.print(b);Serial.println("=");
12.       while (Serial.available()==0){}//等待使用者輸入
13.       c=Serial.parseInt();  //取得使用者輸入
14.       Serial.println(c);
15.       if (c==(a+b)){//假如相同
16.         r=r+1;//答對題數累加1
17.         Serial.println("Right");
18.       }
19.       else {
20.         w=w+1;//答錯題數累加1
21.         Serial.println("Wrong");
22.       }
23.   }
24.   Serial.print("Right=");Serial.println(r);
25.   Serial.print("Wrong=");Serial.println(w);
26. }void loop() {}
```

自我練習

1. 請寫一程式，可以輸出“老師我愛您”20 次。
2. 請寫一程式，可以輸出 -4 -8 -12 -16 -20 -24
3. 請寫一程式，可以輸出所有英文小寫字元。
4. 請寫一個程式，可以計算 1+2+3…+100 的和。
5. 請寫一個程式，產生 10 個 -3 到 3 的亂數，並統計負數、0、正數的個數。

6. 認識鍵盤練習。請寫一個程式，電腦自動出現 1 個小寫字元十次，使用者輸入此字元，電腦評判是否正確，最後統計答對與答錯題數。
7. 請寫程式完成以下條件。
   (1) 產生 20 個 1 到 6 的亂數。
   (2) 輸出以上資料。
   (3) 計算並輸出共有多少個 3。
   (4) 統計所有數字出現次數。
8. 請寫程式完成以下條件。
   (1) 產生 50 個 0 到 100 的亂數。
   (2) 輸出以上資料。
   (3) 統計與輸出 0 ～ 9, 10 ～ 19, 20 ～ 29, 30 ～ 39, 40 ～ 49, 50 ～ 59, 60 ～ 69, 70 ～ 79, 80 ～ 89, 90 ～ 100 等區間的人數。

while指令

前面的 for 是用於程式設計階段已知迴圈次數，但有些情況，我們於程式設計階段並不知迴圈的執行次數，此時即可使用 while 指令。其次，有些迴圈可能一次都不執行，所以 while 指令又分爲前測試迴圈與後測試迴圈。while 的前測試迴圈語法如圖 4-6a，後測試迴圈如圖 4-6b。

```
while(運算式) {
    指令區塊;
}
```

圖 4-6a　前測試 while 語法

```
do {
   指令區塊;
}while (運算式);
```

圖 4-6b　後測試 while 語法

以上語法說明如下：

1. 不論是前測試或後測試迴圈，均是運算式值爲 1（true）時，繼續執行迴圈，運算式爲 0（false）時，離開迴圈。
2. 迴圈的範圍是指兩個大括號『{}』之間。
3. 前測試與後測試迴圈的差別爲，前測試迴圈有可能一次均不執行迴圈，但後測試迴圈至少執行一次。
4. 後測試迴圈的 while（運算式）後面要加分號（；），而前測試迴圈的 while 不用加分號。

範例 4-1-4a

探討計算機除法運算子，本例使用加減法，完成除法運算。

演算法則

兩數相乘時，程式設計階段就知道迴圈執行次數，所以使用 for。但除法就是不同了，只能說，當被除數大於除數時，就連續減去除數，能減去的次數，就是商，剩下的就是餘數。以 8/3 爲例，8 可以連續減 3 兩次，所以商就是 2，剩下餘數就是 2。以上說明的演算法如下：

1. a= 被除數。
2. b= 除數。
3. 商 q=0。
4. 所謂的商就是被除數 a 共有幾個除數 b，也就是只要 a 大於等於 b，就要執行以下指令：

   a=a-b；

   q=q+1；
5. 本例請以 8 除以 3，實際演練如下：

第1次	第2次	第3次
a=8;b=3 a>=b 成立 a=a-b=5 q=q+1=1	a=5;b=3 a>=b 成立 a=a-b=2 q=q+1=2	a=2;b=3 a>=b 不成立 q=2(商) r=a=2（餘數）

6. 以上演算法，以流程圖表示如圖 4-7。

程式列印

```
1.  void setup() {
2.      Serial.begin(9600);
3.      int a=8,b=3,q=0;
4.      while(a>=b) {//只要a>=b，則重覆執行迴圈
5.          a-=b;//a=a-b
6.          q++;//q=q+1
7.      }
```

```
8.      Serial.print("quotient = ");Serial.println(q);  /* 商數 */
9.      Serial.print("remainder= ");Serial.println(a);  /* 餘數 */
10. }void loop() {}
```

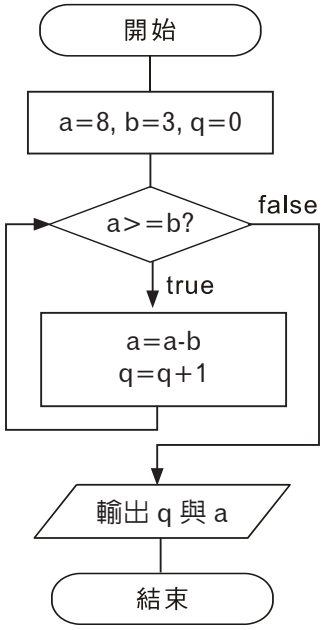


圖 4-7　範例流程圖

1. 本例必須重複很多次，所以適用迴圈，程式才能精簡，迴圈有 for 與 while，本例設計階段不知重複幾次，而是一面減、一面判斷，所以適用 while 迴圈。
2. 本程式僅能使用前測試 while 迴圈，而不能使用後測試迴圈，因爲有可能一開始被除數就小於除數。

※ 範例 4-1-4b

請寫一程式，可以輸入一個整數 (0 到 65535)，然後反向輸出其數字、數字和、幾位數。例如：輸入 2345，則輸出 5,4,3,2,sum=14,4 位數。

執行結果

程式列印

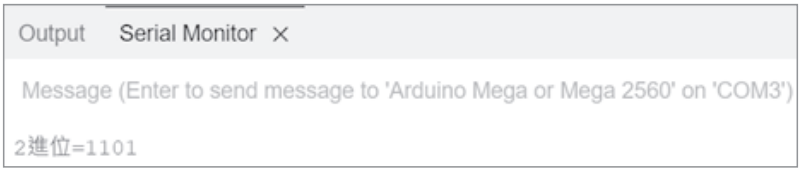
1. 只要每次取除以 10 的餘數，就可以從個位數取到每一個數字。
2. 因爲數字長度不限，設計階段不知迴圈次數，所以使用 while 重複迴圈，程式如下：

```
1.  void setup() {
2.    Serial.begin(9600);
3.    int a=2345;
4.    int sum=0;
5.    int b=0;
6.    int n=0;
7.    while (a>0){
8.      b=a%10;//求餘數
9.      sum=sum+b;//累加餘數和
10.     n=n+1;//計數器，統計迴圈執行次數
11.     Serial.print(b);Serial.print(",");
12.     a=a/10;//除以10
13.   }
14.   Serial.print("sum=");
15.   Serial.print(sum);
16.   Serial.print(",");
17.   Serial.print(n);
18. }void loop() {}
```

※ 範例 4-1-4c

請寫一個程式，可以將 10 進位轉爲 2 進位。

執行結果



程式列印

1. 將 10 進位整數 a 連續除以 2 的餘數串接就是 2 進位，只要 a>0 就要繼續求餘數。例如：

13/2=6..1

6/2=3..0

3/2=1..1

1/2=0..1 // 商已經是 0，迴圈結束

2. 先出爐的餘數放右邊，所以 2 進位是 1101。
3. 以上演算，也是設計階段不知迴圈重複次數，也是要使用 while 迴圈，所以程式如下：

```
1.  void setup() {
2.      Serial.begin(9600);
3.      int a=13;
4.      int b=0;
5.      String s="";
6.      while (a>0){
7.        b=a%2;//求餘數
8.        s=String(b)+s;//將b轉爲字串，向前進行字串串接
9.        a=a/2;//除以2
10.     }
11.     Serial.print("2進位=");
12.     Serial.print(s);
13.   }void loop() {}
```

自我練習

1. 同範例 4-1-4b，但是反相兩個 1 組輸出且輸出其和。例如，指派 12345，則輸出 45,23,1 與和是 69。
2. 心算練習。請連續產生與輸出 2 個 1 位數的亂數，由使用者輸入相加結果，微處理機判斷是否正確，直到使用者答錯爲止，且統計連續正確題數。
3. 請寫一個程式，滿足以下條件：（擲骰子遊戲）
   (1) 可以產生兩個 1 至 6 的亂數。
   (2) 累加以上亂數。
   (3) 輸出此亂數與其和。
   (4) 若亂數和大於 8，則重複 (1) ～ (3)，直到亂數和小於等於 8，則程式結束。

4. 請寫一程式，可以連續產生兩個 1 ～ 6 的亂數，並輸出此兩個亂數，直到後面的亂數大於前面的亂數。

※ 5. 請寫一程式，可以連續產生 3 個 1 ～ 6 的亂數，並輸出此 3 個亂數，直到有其中兩個亂數相等爲止。

※ 6. 請寫一程式，可以連續產生 4 個 1 ～ 6 的亂數，並輸出此 4 個亂數，直到有其中兩個亂數相等爲止，並輸出此不相等的數字與和。例如，產生 6, 4, 5, 1 則繼續產生亂數，若亂數爲 6, 2, 1, 6 則其和爲 3。

4-1-5 陣列

處理少量的資料，可以爲每一筆資料設定一個變數，但若資料數量很多，例如，若有資料如下：

```
3,8,4,7,2,9
```

要求其和、極大、極小呢？是不是指派 6 個變數，當然也可以，但是程式會非常冗長，爲了改善此一問題，就要使用本節的陣列型態了。因爲陣列型態，可以使用『陣列索引』配合 for 與 while 迴圈而簡化程式，此即爲本章重點。例如：以上資料就可宣告陣列如下：

```
int a[7];
```

int 是陣列資料型態，陣列型態可爲 4-1-1 節任一型態，a 是陣列變數名稱，變數名稱也應該遵循 4-1-1 節變數命名規則。然後就可依序指派這些資料到陣列，程式如下：

```
a[1]=3;a[2]=8;a[3]=4;a[4]=7;a[5]=2;a[6]=9;
```

中括號內的 1,2,3 稱爲『陣列索引』，簡稱索引。人類索引編號通常從 1 號開始使用，但電腦卻是從 0 號開始，所以宣告如下：

```
int a[7];
```

其實是使用 0,1,2,3,4,5,6 等 7 個位置。索引 0 號可用可不用，陣列索引從 0 開始，有其獨到用法，若配合迴圈與取餘，可以執行一些無限循環的功能。例如：以下程式，可以讓陣列索引在 0 到 6 之間無限循環。

```
void loop() {
    PORTA=a[i];
    delay(1000);
    i=(i+1) %7;//保障i在0,1,2,3,4,5,6循環
}
```

陣列的存取

陣列的存取都要靠索引，例如：以下程式可重新指派 a 陣列索引 3 的值。

```
a[3]=1;
```

以下程式可將 a 陣列索引 3 的值指派給變數 b，但請留意兩者資料型態要相符。

```
int b=a[3];
```

▮陣列宣告與初值設定

單一變數可宣告變數的同時就指派初值，陣列也可以。例如：前面陣列的宣告與初值指派，程式可以簡化如下：

```
int a[7]={0,3,8,4,7,2,9};
```

本例索引 0 不用，也要給予 1 個 0，3 才會從索引 1 開始放。事實上，陣列宣告時，您可以不用指派陣列的個數，所以以下程式就可以。

```
int a[]={0,3,8,4,7,2,9};
```

但是陣列的大小給的太小也不行。例如，以下程式就不行：

```
int a[3]={0,3,8,4,7,2,9};
```

其次，給大一點當然可以，以下陣列大小給 10，初值僅給 7 個，其索引分別是 0~6，索引 7,8,9 沒給初值，其值由編譯器指派爲 0。

```
int a[10]={0,3,8,4,7,2,9};
```

陣列的資料結構的最大優點是，陣列結構可配合迴圈（前面單一變數不行），例如，以上陣列 a[] 求平均、極大值、與極小值的流程圖如圖 4-8。

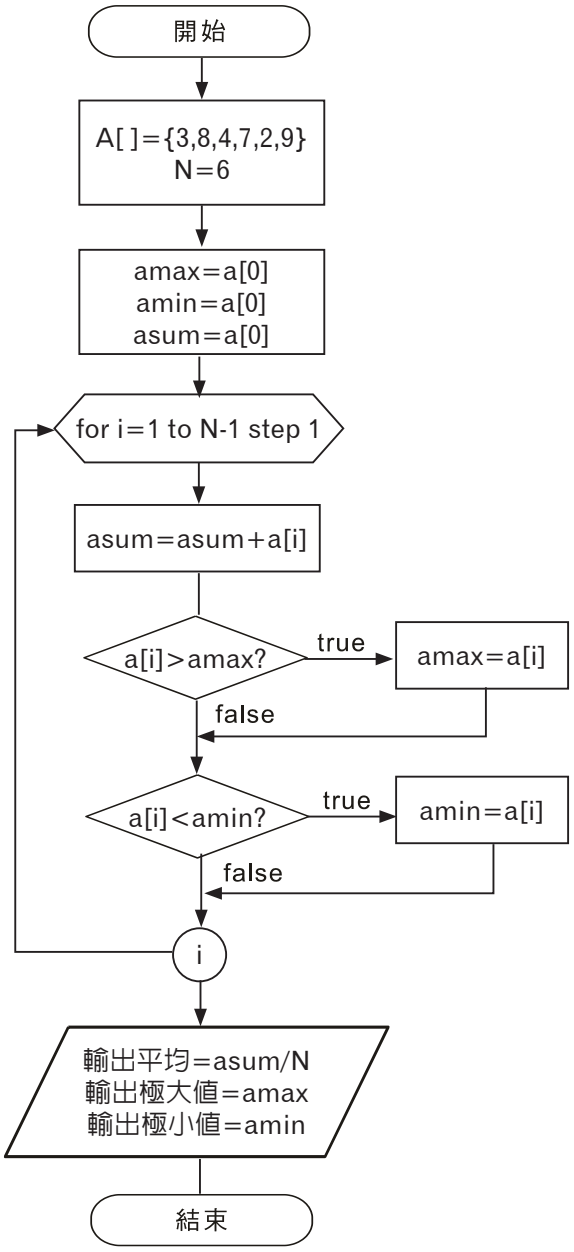


圖 4-8　搜尋極大值流程圖

根據以上流程圖，Arduino 程式如下：

```
1.  void setup() {
2.    Serial.begin(9600);
3.    int a[]={3,8,4,7,2,9};  //指派陣列初值a[0]=3,a[1]=8,a[2]=4…
4.    //有陣列就有索引，有索引就可用迴圈，有了迴圈，程式才能精簡
5.    int  amax, amin, asum;
6.    int N=6;
7.    Serial.println(N);
8.    amax = a[0];  //先指派極大值爲a[0]
9.    amin = a[0];  //先指派極小值爲a[0]
10.   asum = a[0];  //先指派和爲a[0]
11.   for (int i = 1; i <= N-1; i++) {
12.     asum += a[i]; //逐一累加所有陣列值
13.     if (a[i]>amax) //逐一探訪所有陣列值，若大於極大值，
14.       amax = a[i];//極大值被取代
15.     if (a[i]<amin) //逐一探訪所有陣列值，若小於極小值，
16.       amin = a[i]; //極小值被取代
17.   }
18.   Serial.print("average=");//輸出平均
19.   Serial.println( asum / N);
20.   Serial.print("amax=");//輸出極大值
21.   Serial.println(amax );
22.   Serial.print("amin=");//輸出極小值
23.   Serial.println(amin );
24. }
25. void loop() {
```

自我練習

1. 假設有資料如下：

   3,2,1,5,3,2,1,4,5,6

   請統計所有數字出現的次數。

2. 假設有資料如下：

   100,8,9,90,88,4,6,9,11

   請統計其中位數。（若有奇數筆資料，排序後中間那筆資料即爲中位數，若有偶數筆資料，則中間兩筆的平均爲中位數。）

4-1-6 自訂函式

在程式設計時，常會遇到某些程式片段需要在同一個程式或不同程式重複出現多次，如果這些程式片段都分別在每個地方寫一次，那是一件非常浪費時間的事，且會使程式變得冗長而不易閱讀；更糟糕的是，萬一要調整此程式片段的部分功能，還要至不同的地方修改，造成程式維護困難，此時可透過函式解決以上困難。

函式的使用方式是將此常用的程式片段賦予一個名稱，當程式設計者需要使用此程式片段時，只要使用此名稱即可呼叫此程式片段，此程式片段即稱爲『函式』，又稱爲『副程式』（物件導向的程式設計則改稱爲『方法』）。其次，函式完成之後可交由不同的程式呼叫使用，以節省程式設計者的時間。自訂函式的使用步驟如下：

步驟一：實作函式本體。
步驟二：呼叫函式。

實作函式本體

實作函式本體的簡要語法如下：

```
傳回值型態    函式名稱 ( [ 參數 1, 參數 2...] ){
    [ 敘述區塊；]
    [ return (運算式)；]
    [ 敘述區塊；]
}
```

以上語法說明如下：

1. 每個函式可放在 setup() 與 loop() 函式前面或後面均可，且每個函式的地位均相同，所以不可以在函式中再定義其它函式。
2. 函式原則上從左大括號『{』執行到右大括號『}』，但若中途有特殊原因要離開函式，可用 return 提早離開。
3. 參數（Parameter）有些書亦稱爲引數（Argument）。函式呼叫的參數名稱與實作函式本體的參數名稱可不同，但其資料具有傳遞性，也就是

函式呼叫的參數將會傳遞給函式本體，至於函式本體的運算結果是否回傳至主程式，那就得依參數的傳遞方式了，傳遞的方式有傳值、傳址與傳參考，請複習一年級的程式設計實習。

4. 以下程式片段，其函式名稱是『add』，功能是將所傳來的兩個參數相加並傳回。

```
int add (int a, int b){
    int c;
    c=a+b;
    return (c);
}
```

呼叫函式

呼叫函式的簡要語法如下：

```
[ 傳回值 =]    函式名稱 ([ 參數 1, 參數 2, ...]);
```

以上語法說明如下：

1. 函式若無傳回值，那函式會以『void』表示，則呼叫函式名稱前就不用『傳回值』接收。
2. 主程式函式的呼叫參數稱爲實際參數（Actual Parameter），函式本體的參數稱爲形式參數（Formal Parameter），且兩者的名稱可以不同。實際參數會將參數值傳給形式參數，而形式參數是否將值傳回，則要看參數傳遞方式，參數的傳遞方式有傳值、傳址及傳參考，這部分本書不予介紹。
3. 以下程式片段可呼叫 add 函式，傳回值皆爲 8：

```
c=add (6, 2);
```

或

```
m=6; n=2;
c=add (m, n);
```

以下是全部程式列印。

```
1.  int add (int a, int b){
2.    int c=a+b;
3.    return (c);
4.  }
5.  void setup() {
6.    Serial.begin(9600);
7.    int a1=3,b1=4;
8.    int c1;
9.    c1=add(a1,b1);
10.   Serial.println(c1);//7
11. }void loop() {}//雖沒用，但也不可以去掉
```

自我練習

1. 請寫一程式，請先定義一個函式，他可以將所輸入的數取絕對值。例如，函式名稱可以是 myabs()，那 a=myabs(3)，會傳回 3，a=myabs(-2)，會傳回 2。

2. 加、減、乘、除、取餘都有運算子，可否另以函式完成呢。例如，以上範例 add(3,4) 傳回 7，請製作 sub(7,2) 傳回 5，mul(3,8) 傳回 24 等等。

3. 試寫一程式，計算 C_n^m 的值。

   提示：$C_n^m = \frac{m!}{n!(m-n)!}$，m! = 1*2*3…*m。例如，$C_2^5 = \frac{5!}{2!3!} = 10$

4-2 程式編寫

一個 Arduino 程式碼基本上是由 setup() 與 loop() 函式組成，如圖 4-9 所示：

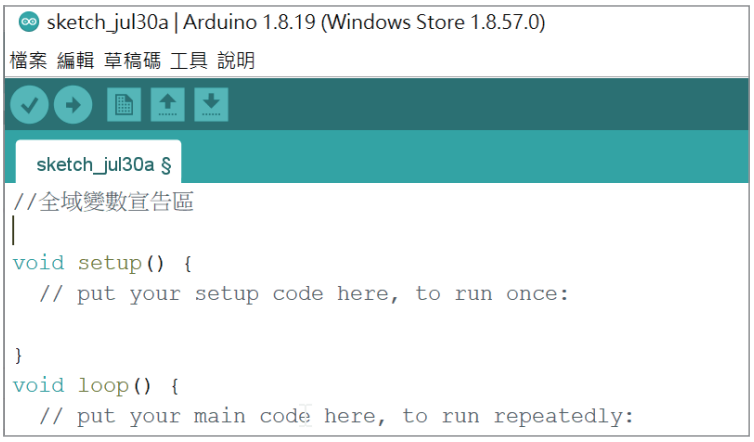


圖 4-9 Arduino 程式架構圖

微處理機啓動後，setup() 函式僅執行 1 次，loop() 函式則重複一直被執行，所以一些基本設定就放在 setup(){} 裡面，需要反覆執行的就放在 loop(){} 函式內。請鍵入以下程式，並比較執行結果。

```
void setup() {
  randomSeed(analogRead(A0));
  Serial.begin(9600);
}
void loop() {
  int x=random(1,7);
  Serial.println(x);
  delay(1000);
}
```

```
void setup() {
  randomSeed(analogRead(A0));
  Serial.begin(9600);
  int x=random(1,7);
  Serial.println(x);
  delay(1000);
}
void loop() {}
```

若是全域變數，也應該放在全域變數區，請鍵入以下程式，並比較其差別。

```
int x=0;
void setup() {
  Serial.begin(9600);
}
void loop() {
  Serial.println(x);
  x=x+1;
  delay(1000);
}
```

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  int x=0;
  Serial.println(x);
  x=x+1;
  delay(1000);
}
```

範例 4-2a

請寫一個程式，可以指派長方形的長與寬，並計算面積。

程式設計解題步驟

1. 資料的數位化
   (1) 本題若使用掌上型計算機，其方法是直接將鍵入「長 * 寬 =」，就可得到答案。例如，鍵入「15*8=」，就可得到答案「120」。
   (2) 但使用 Arduino 語言，必須先將所要計算的值先以一個英文代號儲存，例如 a,b 或 score，以上代號在程式設計的領域，稱爲「變數」。例如：

```
a=15
b=8
```

   設定變數的目的是將資料與程式分開，這樣當資料改變時，還可重複計算。計算過程都是以變數與程式指令描述，此稱爲程式設計。程式設計的優點是有一致性，只要第一次對，往後都對。掌上型計算機就沒有一致性了，每次都要按兩次，才能確認計算結果是否正確，且無法重複使用。
   (3) 選用資料型態。本例需要兩個變數儲存長與寬。請選擇此資料來源是整數還是浮點數，若是整數，還要分是 8 位元的 byte (0 ～ 255)、16 位元的 int (-32768 ～ 32767) 或 32 位元的 long (-2147483648 ～ 2147483647)。本書整數若沒特別註明，就一律折衷，通通取 int，且每一個敘述都要以分號 (;) 結束，所以程式如下：

```
int a=15;
int b=8;
```

2. 寫出演算法。本例是輸入長方形的長與寬來計算面積，所以演算法如下：

```
面積=a*b
```

3. 使用變數與程式指令，完成以上演算法。本例是計算面積，面積也要使用變數儲存，也要選資料型態，本例使用 c，資料型態選 int ，所以是：

```
int c=a*b
```

4. 輸出結果。

```
Serial.print("面積=");
Serial.println(c);
```

5. 以上全部程式如下：

```
1.  void setup() {
2.    Serial.begin(9600);//啓動序列埠
3.    int a=15;
4.    int b=8;
5.    int c=a*b;
6.    Serial.print("面積=");
7.    Serial.println(c);
8.  }
9.  void loop() {}
```

自我練習

1. 指派長方體的長、寬、高，計算其表面積與體積。
2. 請寫一個程式，可以指派一個台斤數，且轉爲公斤數輸出。
3. 請寫一個電子時鐘程式，可以使用序列埠視窗輸出，輸出格式爲『時：分：秒』。
4. 請寫一個倒數計時器，可以使用序列埠視窗輸出，輸出格式爲『分：秒』。

學後測驗

題號	題目	結果

1	`void loop() {`
	`  byte i=0;`
	`  i=i+1;`
	`  Serial.println(i);`
	`}`	

2	`byte i=0;`
	`void loop() {`
	`  i=i+1;`
	`  Serial.println(i);`
	`}`	

3	`byte i=0;`
	`void loop() {`
	`  i=i-1;`
	`  Serial.println(i);`
	`}`	

4	`int i=0;`
	`void setup() {`
	`  Serial.begin(9600);`
	`  i=i-1 ;`
	`  Serial.println(i);`
	`}`	

5	`int a=5,b=4;`
	`float c=4;`
	`Serial.println(a%b);`
	`Serial.println(a/b);`
	`Serial.println(a/c);`	

6	`int a=5,b=3;`
	`Serial.println(!(a>b));`
	`Serial.println((a>b) && (b>=4));`
	`Serial.println((a>b) || (b>=4));`	

7	byte a=3; Serial.println(~a); Serial.println(a^4); Serial.println(a&3); Serial.println(a|5); Serial.println(a<<3); Serial.println(a>>2);	

8	int d=0x23;
	int e=023;
	Serial.println(d);
	Serial.println(e);	

9	oid setup() {
	  Serial.begin(9600);
	  int i=3 ;
	  String b="";
	  if (i%2==0)
	  b="偶數" ;
	  else
	  b="奇數";
	  Serial.println(b);
	}void loop() {}	

10	for(int i=2;i<8;i=i+2){
	  Serial.print(i);
	  }	

11	for(int i=8;i>=1;i=i-2){
	  Serial.print(i);//8642
	  }	

12	int a=52, b=138,s=0;
	for (int i=1 ;i<=a;i++)
	  s=s+b;
	Serial.println(s);	

13	int a=18,b=5,q=0;
	while(a>=b) {
	  a-=b;//a=a-b
	  q++;//q=q+1
	  }
	Serial.print("quotient = ");Serial.
	println(q);
	Serial.print("remainder= ");Serial.
	println(a);	

14	void setup() {
	  Serial.begin(9600);
	  int a[]={13,8,4,37,2,19};
	  int amax, amin, asum; N=6;
	  amax = a[0]; amin=a[0];
	  for (int i = 1; i <= N-1; i++) {
	  if (a[i]>amax)
	  amax = a[i];
	  if (a[i]<amin)
	  amin = a[i];
	  }
	  Serial.println(amax);
	  Serial.println(amin);
	}
	void loop() {	

15	int add (int a, int b){
	  int c=a+b; return (c);
	}
	void setup() {
	  Serial.begin(9600);
	  Serial.println(add(4,6));
	}void loop() {}	

MEMO