單元 5
霹靂燈

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;//零，不是字母 O
    int d=0x11;//零 x，不是字母 OX
    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，電腦是以 2 進位儲存數值，所以若以 8 位元儲存一個正整數，那

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』，那反過來說，若我們希望 LED 呈現

滅滅滅滅滅亮滅亮

那我們也可用 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。

霹靂燈

霹靂燈就是一串 LED 閃來閃去，可能右旋，也可能左旋，本章就是要來介紹霹靂燈的控制。

▶範例 5a

若有一霹靂燈的閃爍時序圖如下，請寫程式完成。(0 代表 LED 不亮，1 代表 LED 亮)

時序	LED7	LED6	LED5	LED4	LED3	LED2	LED1	LED0	值
0	0	0	0	0	0	0	0	1	0x01
1	0	0	0	0	0	0	1	1	0x03
2	0	0	0	0	0	1	1	1	0x07
3	0	0	0	0	1	1	1	1	0x0F
4	0	0	0	1	1	1	1	1	0x1F
5	0	0	1	1	1	1	1	1	0x3F
6	0	1	1	1	1	1	1	1	0x7F
7	1	1	1	1	1	1	1	1	0xFF

➲ 操作步驟

1 將以上資料數位化，得到 0x01, 0x03, 0x07, 0x0F, 0x1F, 0x3F, 0x7F, 0xFF，如上表最右一欄。

2 使用變數儲存以上資料。本例使用 a0, a1, a2, a3,a4, a5, a6, a7 表示如下：

```
    a0=0x01;
    a1=0x03;
    a2=0x07;
    a3=0x0F;
    a4=0x1F;
    a5=0x3F;
    a6=0x7F;
    a7=0xFF;
```

3 將以上資料寫入 Arduino，程式如下：

```
void setup() {
   DDRA=B11111111;
   Serial.begin(9600);
}
void loop() {
  byte a0,a1,a2,a3,a4,a5,a6,a7;
  //宣告 a0,a1...是 8 位元正整數，可表示 0~255
  a0=0x01;
  PORTA=a0;
```

```
  delay(1000);
  a1=0x03;
  PORTA=a1;
  delay(1000);
  a2=0x07;
  PORTA=a2;
  delay(1000);
  a3=0x0F;
  PORTA=a3;
  delay(1000);
  a4=0x1F;
  PORTA=a4;
  delay(1000);
  a5=0x3F;
  PORTA=a5;
  delay(1000);
  a6=0x7F;
  PORTA=a6;
  delay(1000);
  a7=0xFF;
  PORTA=a7;
  delay(1000);
}
```

➲ 自我練習

1. 若希望霹靂燈的變化如下，請寫程式完成。

時序	LED7	LED6	LED5	LED4	LED3	LED2	LED1	LED0	值
0	1	0	0	0	0	0	0	1	
1	0	1	0	0	0	0	1	0	
2	0	0	1	0	0	1	0	0	
3	0	0	0	1	1	0	0	0	
4	0	0	0	1	1	0	0	0	
5	0	0	1	1	1	1	0	0	
6	0	1	1	1	1	1	1	0	
7	0	1	1	1	1	1	1	1	
8	1	1	1	1	1	1	1	1	

9	1	1	1	1	1	1	1	0	
10	1	1	1	1	1	1	0	0	
11	1	1	1	1	1	0	0	0	
12	1	1	1	1	0	0	0	0	
13	1	1	1	0	0	0	0	0	
14	1	1	0	0	0	0	0	0	
15	1	0	0	0	0	0	0	0	
16	0	0	0	0	0	0	0	0	

2. 若有一個單向紅綠燈時序如下，請寫程式完成。

時序	LED2（紅燈）	LED1（黃燈）	LED0(綠燈)	值
0	0	0	1	
1	0	0	1	
2	0	0	1	
3	0	0	1	
4	0	1	0	
5	0	1	0	
6	1	0	0	
7	1	0	0	
8	1	0	0	

3. 請自行觀察路口的雙向紅綠燈，並寫程式完成。

單元 6
霹靂燈與陣列

上一單元的霹靂燈共 8 個時序，我們用了 8 個變數表示 8 個顯示方式，自我練習 1 共 17 個時序，那就要 17 個變數表示，這樣程式寫起來，眞是又臭又長，那如果更複雜的變化呢？本單元我們將要介紹一個很好用的工具，那就是陣列，陣列就是要解決儲存連續相同的大批資料。例如，上一單元範例，我們一共 8 筆連續資料，如果用陣列結構，那就可以宣告一個一維陣列如下：

```
int a[8];
```

那就表示一共可儲存 8 筆資料，分別是 a[0], a[1], a[2], a[3], a[4], a[5], a[6], a[7]，中括號內的數字稱爲索引，索引可用來指派資料來源，就如同我們常用座號來指派同學一樣，只是人類習慣從 1 號開始，但電腦因爲了方便配合取餘運算與重複循環控制，所以陣列從 0 號開始，但是 0 號可用，也可不用，只是重複控制從 0 號開始，若配合取餘『%』會較方便。上一範例的資料設定，若以陣列表示，則陣列指派如下：

```
a[0]=0x01;
a[1]=0x03;
a[2]=0x07;
a[3]=0x0F;
a[4]=0x1F;
a[5]=0x3F;
a[6]=0x7F;
a[7]=0xFF;
```

以上宣告陣列與陣列初值指派兩個步驟，亦可宣告陣列的同時，就指派

其值如下：

```
   byte a[]={0x01,0x03,0x07,0x0F,0x1F,0x3F,0x7F,0xFF};
```

往後就可以使用陣列索引存取陣列的值，程式如下：

```
   PORTA=a[0];
   delay(1000);
   PORTA=a[1];
   delay(1000);
   PORTA=a[2];
   delay(1000);
   PORTA=a[3];
   delay(1000);
   PORTA=a[4];
   delay(1000);
   PORTA=a[5];
   delay(1000);
   PORTA=a[6];
   delay(1000);
   PORTA=a[7];
   delay(1000);
```

以上全部程式如下：

```
void setup() {
   DDRA=B11111111;
   Serial.begin(9600);
}
void loop() {
  byte a[]={0x01,0x03,0x07,0x0F,0x1F,0x3F,0x7F,0xFF};
  PORTA=a[0];
  delay(1000);
  PORTA=a[1];
  delay(1000);
  PORTA=a[2];
  delay(1000);
  PORTA=a[3];
  delay(1000);
```

```
  PORTA=a[4];
  delay(1000);
  PORTA=a[5];
  delay(1000);
  PORTA=a[6];
  delay(1000);
  PORTA=a[7];
  delay(1000);
}
```

這樣程式還是很長，若配合迴圈與索引，程式還可簡化如下：

```
void setup() {
   DDRA=B11111111;
   Serial.begin(9600);
}
byte i=0;
void loop() {
  byte a[]={0x01,0x03,0x07,0x0F,0x1F,0x3F,0x7F,0xFF};
  PORTA=a[i];
  delay(1000);
  i=(i+1)%8;//i 每次遞增 1，但%8 可保障數字 i 在 0～7 之間
}
```

➲ 自我練習

1. 請將上一單元三個自我練習，以陣列重作。
2. 教學機。本單元的 LED 您可以拿來作爲順序教學的指示燈。例如，電子琴、跳舞機、CPR 急救順序、葉問拳法練習樁等，也就是您可以用 LED 來提醒使用者要按哪裡、踩哪裡、打哪裡等等等，請自己思考身邊的應用。

單元 7
七段顯示器

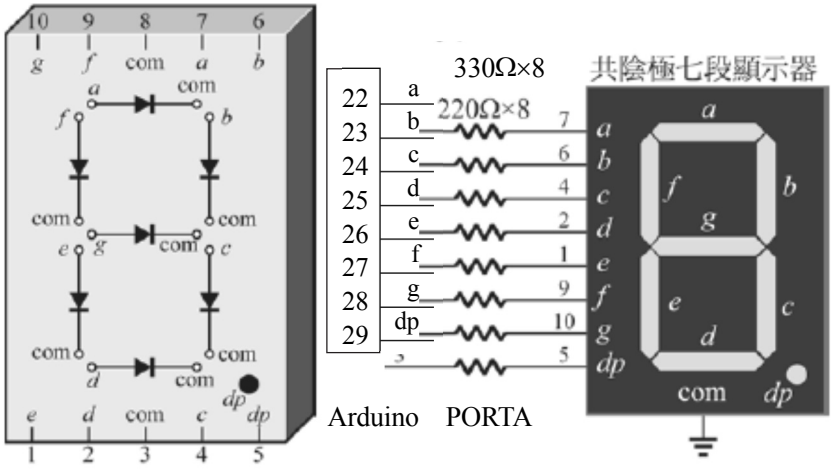
七段顯示器是由 8 個 LED 排列成「日」字，8 個 LED 照理應有 16 隻腳，但爲了簡化腳位與配合低或高電位驅動，廠商是將所有 LED 的陽極或陰極共接再拉出，所以分爲共陽極或共陰極七段顯示器，兩者原理相同，本書就介紹共陰極如下：

共陰極七段顯示器

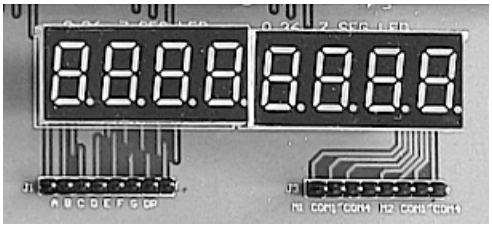
共陰極七段顯示器的內部電路與腳位圖如下圖左（大部分的共陰與共陽七段顯示器腳位排列都相同）。

➲ 測量檢驗方式

1. 請自行使用傳統指針型三用電表測量，先撥歐姆檔×10，例如黑棒(+)接 a，紅棒(–)接 com 點，則上面的 a 棒亮。
2. 或用附錄 A 自製的接點連通計，正極接 a，負極接 com，那 a 棒亮。



本書教學實驗板並未準備一位數七段顯示器，請將八位數七段顯示器的 com1 接地，那就可以當作一位數七段顯示器用，如下圖，其次，a,b,c,d,e,f,g,dp 的腳位我已經按順序由左而右排列。

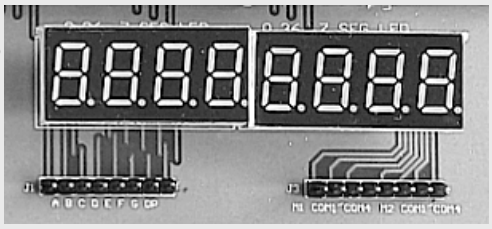


要完成上圖的接線，只要拿 8 條杜邦線，將一端插入 22～29，另一端插入電阻『J16』；再 8 條杜邦線，將一端插入電阻的『J17』，另一端插入上圖的『ABCDEFGDP』；再拿單 1 條杜邦線，一端插入上圖『com1』，另一端插入微控板『GND』。其次，共陰極要顯示 0 到 9，所要輸出的電壓如下表，1 代表亮，0 代表不亮。例如，要顯示 0，則 a, b, c, d, e, f 等腳位要給高電壓（亮），g 與 dp 要給低電位（不亮），所以資料要輸出 0x3F。

數字	dp	g	f	e	d	c	b	a	十六進制數值
0	0	0	1	1	1	1	1	1	0x3F
1	0	0	0	0	0	1	1	0	0x6
2	0	1	0	1	1	0	1	1	0x5B
3	0	1	0	0	1	1	1	1	0x4F
4	0	1	1	0	0	1	1	0	0x66
5	0	1	1	0	1	1	0	1	0x6D
6	0	1	1	1	1	1	0	1	0x7D
7	0	0	0	0	0	1	1	1	0x7
8	0	1	1	1	1	1	1	1	0x7F
9	0	1	1	0	1	1	1	1	0x6F

例如，以下程式可以重複循環顯示 0, 1, 2 各一秒鐘。

```
//PORTA 22,23,24,25,26,27,28,29
//連接七段顯示器的 a,b,c,d,e,f,g,dp
void setup() {
   DDRA=B11111111;
}
void loop() {
    PORTA=0x3f;
    delay(1000);
    PORTA=0x6;
    delay(1000);
    PORTA=0x5b;
    delay(1000);
}
```



▶範例 7a

示範查表顯示數字。

➲ 說明

前面我們要顯示 0，就輸出 0x3f；要顯示 1，就輸出 0x6；要顯示 2 就輸出 0x5b，爲了方便所有數字的顯示，我們通常以陣列儲存以上資料如下：

byte a[]={0x3f,0x6,0x5b,0x4f,0x66,0x6d,0x7d,0x7,0x7f,0x6f};

那只要指派陣列索引，就可以得到對應輸出碼，此即爲陣列索引查表功能。例如，要顯示 1，程式如下：

```
PORTA=a[1];
delay(1000);
```

要顯示 5，程式如下：

```
PORTA=a[5];
delay(1000);
```

以下程式，就可以重複顯示 0 到 9 各 1 秒。

```
int a[]={0x3f,0x6,0x5b,0x4f,0x66,0x6d,0x7d,0x7,0x7f,0x6f};
int i=0;
void setup() {
   DDRA=B11111111;
}
void loop() {
  PORTA=a[i];  //PORTA 本例使用 29,28,27,26,25,24,23,22
  delay(1000);
  i=(i+1) %10;  //共 10 個
}
```

➲ 自我練習

1. 同上題，那如何顯示一個空白呢？（提示：建議可將空白『 』以 10 存放）
2. 請寫一程式讓七段顯示器，逐一輪流顯示編號爲 a, b, c, d, e, f, g, dp 等 LED 各 1 秒與空白也 1 秒，且重複。

▶範例 7b

請用一個一維陣列儲存您的學號（例如 825710），並用一位數七段顯示器依序顯示。

➲ 思考步驟

1. 本例假設學號是 825710，那要分別送出 0x7f, 0x5b, 0x6d, 0x7, 0x6, 0x3f，所以將以上資料放在陣列如下：

```
   byte b[]={0x7f,0x5b,0x6d,0x7,0x6,0x3f};
```

2. 根據上一範例，將 b 陣列指派給 PORTA 即可，所以程式如下：

```
byte b[]={0x7f,0x5b,0x6d,0x7,0x6,0x3f};
byte len=6;
```

```
byte i=0;
void setup() {
   DDRA=B11111111;
}
void loop() {
  PORTA= b[i];
  delay(1000);
  i=(i+1) %len;
}
```

3. 前面程式是自己將 825710 查表，轉爲七段顯示器的輸出碼，其實也可讓電腦自己查表，例如，將 825710 放在陣列，如下：

```
   byte b[]={8,2,5,7,1,0};
```

4. 然後程式如下，此稱爲陣列中的陣列。

```
   PORTA= a[b[i]];
```

例如，i=0，b[0]=8，那 a[8]=0x7F，也就是送出 0x7f，那就顯示『8』。

5. 全部程式如下：

```
byte a[]={0x3f,0x6,0x5b,0x4f,0x66,0x6d,0x7d,0x7,0x7f,0x6f};
byte b[]={8,2,5,7,1,0};
byte len=6;
byte i=0;
void setup() {
   DDRA=B11111111;
}
void loop() {
  PORTA= a[b[i]];
  delay(1000);
  i=(i+1) %len;
}
```

➲ 補充說明

1. 程式設計的領域，很多都是重複循環，請多利用陣列索引從 0 開始

的特性，且配合取餘(%)運算，這樣才能簡化程式的撰寫。其次，陣列索引從 0 開始，只是給使用者方便，不用 0 開始也沒關係。

➲ 自我練習

1. 同上範例，如何也顯示一個空白 1 秒。
2. 請將今天的日期，以陣列儲存，並用一個七段顯示器逐一顯示。例如，今天的日期是"2019-04-26"，建議可將"-"以 10 存放，然後 a 陣列如下：

```
byte a[]={0x3f,0x6,0x5b,0x4f,0x66,0x6d,0x7d,0x7,0x7f,0x6f,0x40};
```

a[10]是 0x40，將顯示"-"，所以 b 陣列如下：

```
byte b[]={2,0,1,9,10,0,4,10,2,6};
```

3. 請將現在的時間，以陣列儲存，並用一個七段顯示器逐一顯示。

▶範例 7c

請寫一程式，可以輸出 1 個四位數整數。

➲ 思考步驟

1. 您要發揮國小數學能力，充分利用整數除法與取餘運算，將 1 個 4 位數，逐一分解成 4 個位數。例如，1028 如何得到 1, 0, 2, 8，程式如下：

```
 int c=1028;//1028 已經超過 255，所以要宣告為 int
d=c/1000;//千
c=(c-d*1000);
 d=c/100;//百
c=c-d*100;
 d=c/10;//十
d=c%10;//個位數
```

全部程式如下：

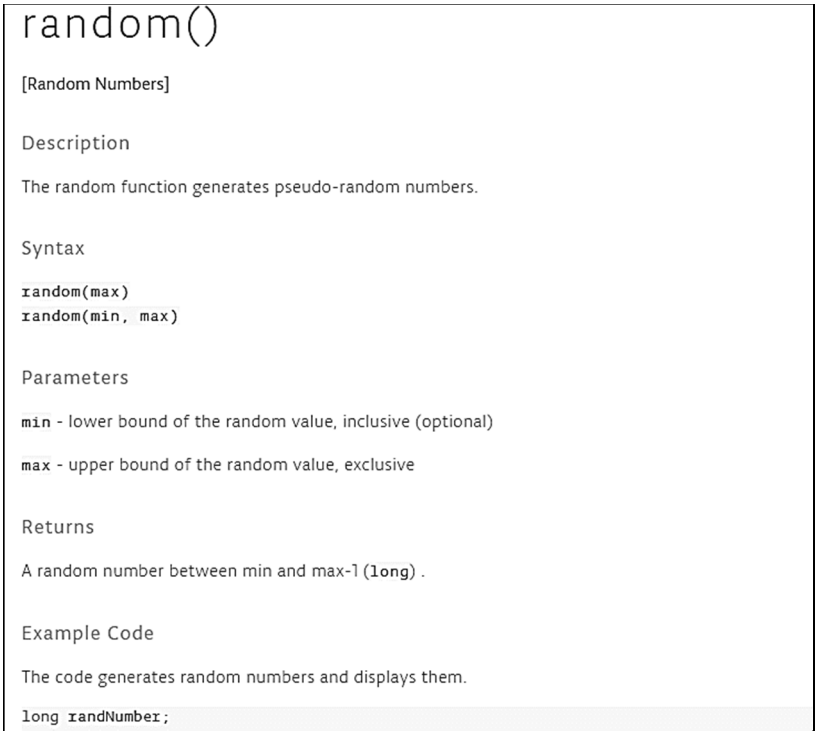
```
void setup() {
   DDRA=B11111111;
   Serial.begin(9600);
}
void loop() {
  byte a[]={0x3f,0x6,0x5b,0x4f,0x66,0x6d,0x7d,0x7,0x7f,0x6f,0x40};
  int c=1022;//1022 已經超過 255，不能宣告為 byte, 要宣告為 int
  byte d=1022/1000;//整數除法，得到千位數
  PORTA=a[d];//顯示千位數
  delay(1000);
  c=(c-d*1000);
  d=c/100;//百位數
  PORTA=a[d];//顯示百位數
  delay(1000);
  c=c-d*100;
  d=c/10;//十位數
  PORTA=a[d];//顯示十位數
  delay(1000);
  d=c%10;//個位數
  PORTA=a[d];//顯示個位數
  delay(1000);
}
```

➲ 自我練習

1. 同上題，數字出現後，請閃爍一下。
2. 同上題，可以顯示 3 位數，且含有 1 個小數點。例如，103.2。
3. 於單向紅綠燈的系統中（僅有紅黃綠三個燈，且秒數均小於 10），加上紅燈與綠燈倒數秒數。

單元 8
擲骰子

電腦常常需要模擬一些執行結果，例如，樂透開獎或擲骰子，此時就需要產生亂數，Arduino 產生亂數的方法是使用 random()函式，其語法如下圖：



random()

[Random Numbers]

Description

The random function generates pseudo-random numbers.

Syntax

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

Parameters

`min` - lower bound of the random value, inclusive (optional)

`max` - upper bound of the random value, exclusive

Returns

A random number between min and max-1 (`long`) .

Example Code

The code generates random numbers and displays them.

```
long randNumber;
```

例如，以下程式可產生 10 到 19 的整數亂數。

```
int r = random(10, 20);//10..19，不含 20
```

其中 min 若省略，則視爲 0，例如，以下程式可產生 0 到 299 的整數亂數。

```
    int r = random(300);//0..299
```

其次，因爲亂數的產生是一種查表的動作，爲了讓每次都能有不同的起始點，那就是要使用 randomSeed()函式。例如，以下程式將 A0 腳位空接，那就可隨機讀取一個雜訊，就將此雜訊作爲一個不同亂數起點的依據。

```
randomSeed(analogRead(A0));//用 0 或 A0 都表示 A0 腳位
```

▶範例 8a

示範以上程式。

➲ 執行結果

```
14
288
14
194
```

➲ 程式列印

```
void setup(){
  Serial.begin(9600);
  randomSeed(analogRead(A0));//以雜訊作為產生不同亂數起點的依據
}
void loop() {
  int r;
  // print a random number from 10 to 19
  r = random(10, 20);
  Serial.println(r);
  delay(1000);
  // print a random number from 0 to 299
  r = random(300);
  Serial.println(r);
}
```

➲ 自我練習

1. 請將以上範例的 randomSeed(analogRead(A0))去掉，並觀察執行結果，請留意每次執行程式時（『將序列埠監控視窗』關閉，再重新開啓序列埠視窗，即可重新執行程式），其亂數順序是否都相同。

擲骰子遊戲

有了 random()函式，就可完成擲骰子、樸克牌、樂透開獎等程式。例如，請寫一程式，可以模擬擲三顆骰子，於序列埠監控視窗輸出與七段顯示器同時輸出結果。

▶範例 8b

示範產生 3 個亂數。

➲ 執行結果

下圖是序列埠輸出視窗的結果

```
5
5
1
```

➲ 程式列印

```
//PORTA 22,23,24,25,26,27,28,29 連接到 seven-segment
a,b,c,d,e,f,g,dp
byte a[]={0x3f,0x6,0x5b,0x4f,0x66,0x6d,0x7d,0x7,0x7f,0x6f};
void setup(){
  Serial.begin(9600);
  randomSeed(analogRead(0));
}
void loop() {
  int r;
  // print a random number from 1 to 6
  r = random(1, 6);
  Serial.println(r);
  PORTA=a[r];
```

```
  delay(1000);
  r = random(1, 6);
  Serial.println(r);
  PORTA=a[r];
  delay(1000);
  r = random(1, 6);
  Serial.println(r);
  PORTA=a[r];
  delay(1000);
  delay (3000);
}
```

for 迴圈

上一範例，產生 3 次亂數，這 3 次亂數的程式其實相同，電腦有一個指令 for，它可以簡化程式的撰寫，例如，以上程式的

```
   r = random(1, 6);
   Serial.println(r);
   PORTA=a[r];
   delay(1000);
```

一共寫了 3 次，所以我們可以用一個 for 迴圈替代如下：

```
   for (int i=1;i<=3;i=i+1){// i=i+1 表示 i 先加 1，再放回 i，所以
每次遞增 1
    r = random(1, 6);
    Serial.println(r);
    PORTA=a[r];
    delay(1000);
   }
```

所以整個 loop()程式如下：

```
void loop() {
  int r;//本例 byte r;也可以
  // print a random number from 1 to 6
```

```
  for (int i=1;i<=3;i++){//因為 i=i+1 常用,所以可簡化為 i++
    r = random(1, 6);
    Serial.println(r);
    PORTA=a[r];
    delay(1000);
  }
  delay (2000);
}
```

▶範例 8c

示範以上程式。

```
byte a[]={0x3f,0x6,0x5b,0x4f,0x66,0x6d,0x7d,0x7,0x7f,0x6f};
void setup(){
  Serial.begin(9600);
  randomSeed(analogRead(0));
}
void loop() {
  int r;//本例 byte r;也可以
  // print a random number from 1 to 6
  for (int i=1;i<=3;i++){//for (byte i=1;i<=3;i=i+1)也可以
    r = random(1, 6);
    Serial.println(r);
    PORTA=a[r];
    delay(1000);
  }
  delay (2000);
}
```

其次，有時候亂數會相同，不容易觀察，所以以下程式可先清除七段內容，再顯示。

```
void loop() {
  int r;
  // print a random number from 1 to 6
  for (int i=1;i<=3;i++){
    r = random(1, 7);//1 to 6
```

```
    Serial.println(r);
    PORTA=a[r];
    delay(1000);
    PORTA=0;//滅
    delay(200);
  }
  delay (2000);
}
```

閃爍

連續明滅稱爲閃爍，以下程式，我們再用一個 j 迴圈讓它連續明滅 2 次。

```
void loop() {
  int r;
  for (int i=1;i<=3;i=i+1){
    r = random(1, 7);//1 to 6
    Serial.println(r);
    for (int j=1;j<=2;j++){//閃爍 2 次
      PORTA=a[r];
      delay(200);
      PORTA=0;//滅
      delay(200);
    }
    PORTA=a[r];
    delay(1500);
  }
```

➲ 自我練習

1. 請寫一程式，可以幫您產生 1 個 1 到 4 的亂數，然後依據亂數對應擲骰子次數。例如，產生亂數 3，那就連續擲 3 次骰子。

單元 9
for 迴圈

上一單元已經使用 for 迴圈簡化程式的撰寫，for 是程式設計裡最神奇好用的演算法，它可以用很簡短的指令讓電腦持續工作。例如，您想輸出 1 2 3 4 5 6 ，那可以撰寫程式如下：

```
void setup(){
  Serial.begin(9600);
  Serial.print(1);Serial.print(2);Serial.print(3);
  Serial.print(4); Serial.print(5); Serial.print(6);
}void loop() {}
```

以上 6 筆資料，還可一一鍵入程式，但如果是 100 筆資料呢？所以還是要慢慢熟悉電腦的 for 指令，因爲迴圈運算太重要了，所以本單元再深入介紹 for 指令的功能，等熟悉這個指令，那寫起程式來才能虎虎生風，所向披靡。例如，以上程式，也可以使用 for 迴圈如下：

```
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（i 在此是指派不是相等），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

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

3. 以下程式，『i=8』與『i==8』通通不行 ，這在單元十介紹『比較運算子』後，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. 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){//i<=8才有結果
    Serial.print(i);
}
```

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

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. 迴圈變數的宣告，若要遵守變數有效範圍的規定，那當然是在迴圈內宣告，如下圖左，但大部分人都便宜行事，含我在內，一律在迴圈外宣告一次後，就重複使用，如下圖右，但這樣您就要避免使用這些迴圈變數，這樣才不會互相干擾。

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

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

例如，以下圖左變數 i 就慘了，i 誤被拿去當迴圈用，i 值就被更動了，那結果就很慘。若養成迴圈內宣告迴圈變數 i，如下圖右，那迴圈變數的 i，和外面的 i 彼此獨立，那就互不干擾。

```
int i =2,sum=3;
for( i=1;i<=6;i++){
    Serial.print(i);
  }
sum=sum+i;
```

```
int i =2,sum=3;
for( int i=1;i<=6;i++){
    Serial.print(i);
  }
sum=sum+i;
```

請分別鍵入以下程式，並比較其執行結果。

```
void setup() {
    Serial.begin(9600);
    int i =2,sum=3;
    for( i=1;i<=6;i++){
      Serial.print(i);
    }
    Serial.println(i);
    sum=sum+i;
    Serial.println(sum);
}void loop() {}
```

```
void setup() {
    Serial.begin(9600);
    int i =2,sum=3;
    for( int i=1;i<=6;i++){
      Serial.print(i);
    }
    Serial.println(i);
    sum=sum+i;
    Serial.println(sum);
}void loop() {}
```

➲ 自我練習

1. 請寫一程式，可以輸出 -6 -4 -2 0 2 4
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“老師我愛您”20 次。

▶範例 9a

請輸出指定數字的九九乘法表，例如，指定 5，就輸出 5 的乘法表。

方法一：可以使用 9 個敘述一一輸出，程式如下：

```
void stup(){
  Serial.begin(9600);
  Serial.println("5*1=5");
  Serial.println("5*2=10");
  Serial.println("5*3=15");//共 9 列，以下略
}void loop() {}
```

方法二：可以善用迴圈，程式如下：

```
void setup(){
  Serial.begin(9600);
  int a=5;
  for(int i=1;i<=9;i++){
    Serial.print(a);//變數
    Serial.print("*");//字串
    Serial.print(i);
```

```
    Serial.print("=");
    Serial.println(a*i);
  }
}void loop() {}
```

```
5*1=5
5*2=10
5*3=15
5*4=20
5*5=25
5*6=30
5*7=35
5*8=40
5*9=45
```

以下使用雙迴圈，就可輸出完整九九乘法表。

```
void setup(){
  Serial.begin(9600);
  for(int i=1;i<=9;i++){
    for(int j=1;j<=9;j++){
      Serial.print(i);//變數
      Serial.print("*");//字串
      Serial.print(j);
      Serial.print("=");
      Serial.print(i*j);
      Serial.print(" ");
    }
    Serial.println();
  }
}void loop() {}
```

以上執行結果如下：

```
1*1=1 1*2=2 1*3=3 1*4=4 1*5=5 1*6=6 1*7=7 1*8=8 1*9=9
2*1=2 2*2=4 2*3=6 2*4=8 2*5=10 2*6=12 2*7=14 2*8=16 2*9=18
3*1=3 3*2=6 3*3=9 3*4=12 3*5=15 3*6=18 3*7=21 3*8=24 3*9=27
4*1=4 4*2=8 4*3=12 4*4=16 4*5=20 4*6=24 4*7=28 4*8=32 4*9=36
5*1=5 5*2=10 5*3=15 5*4=20 5*5=25 5*6=30 5*7=35 5*8=40 5*9=45
6*1=6 6*2=12 6*3=18 6*4=24 6*5=30 6*6=36 6*7=42 6*8=48 6*9=54
7*1=7 7*2=14 7*3=21 7*4=28 7*5=35 7*6=42 7*7=49 7*8=56 7*9=63
8*1=8 8*2=16 8*3=24 8*4=32 8*5=40 8*6=48 8*7=56 8*8=64 8*9=72
9*1=9 9*2=18 9*3=27 9*4=36 9*5=45 9*6=54 9*7=63 9*8=72 9*9=81
```

➲ 自我練習

1. 請寫一程式，嘗試使用兩層迴圈印出如下的九九乘法表。

```
  1  2  3  4  5  6  7  8  9
1 1  2  3  4  5  6  7  8  9
2 2  4  6  8 10 12 14 16 18
3 3  6  9 12 15 18 21 24 27
4 4  8 12 16 20 24 28 32 36
5 5 10 15 20 25 30 35 40 45
6 6 12 18 24 30 36 42 48 54
7 7 14 21 28 35 42 49 56 63
8 8 16 24 32 40 48 56 64 72
9 9 18 27 36 45 54 63 72 81
```

2. 請嘗試使用雙層迴圈輸出如下圖左與下圖右。

1 1 2 1 2 3 1 2 3 4 1 2 3 4 5	5 5 5 5 5 4 4 4 4 3 3 3 2 2 1

▶範例 9b

假如沒有乘法運算子，請問如何完成乘法運算。

您我都知道 5*9=45，但這是從小背誦而得，電腦就沒有這個能力，電腦其實只有加法指令（減法是取 2 補數，再相加）；乘法則要發揮連加的功能，因爲電腦計算能力很強，需要時可以馬上計算而得。例如 5*9 就 5 連加 9 次，程式如下：

```
void setup(){
  Serial.begin(9600);
  int a=5;
  int s=0;
  for(int i=1;i<=9;i++){
    s=s+a;
  }
  Serial.println(s);
```

```
}void loop() {}
```

➲ 自我練習

1. 請寫一個程式，可以計算 1+2+3+…+10 的和。
2. 請寫一個程式，可以計算 3+6+9+…+48 的和。

單元 10
變數宣告、運算子與算術運算

前面我們都把重點放在周邊裝置的控制，以周邊裝置學習電腦的程式設計，第二單元我們也曾說，Arduino 除了可作爲周邊裝置控制外，其實也是學習一般算術運算與資料處理的利器，本單元就要介紹算術運算，讓您除了進行以上周邊裝置控制外，也能學習一般程式語言的資料處理。

運算子與算術運算

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。本單元先介紹算術運算子，其他種類運算子將會陸續在後續單元介紹，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)

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，下表是以上算

術運算子的功能說明：(優先順序與結合律先不理會，後續單元 13 再介紹)

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（商的型態同被除數）	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

運算元與資料型態

一般的計算機進行算術運算，就直接運算，例如進行 3＊2，就直接按『3』、『＊』、『2』、『＝』就可得到答案，但是以上步驟當要重複計算時，都要重複按以上運算元與運算子，這樣非常耗時與不便，所以有電腦程式語言的發展。所謂電腦程式設計，是利用代號儲存以上運算元（這些代號在程式設計領域稱爲變數，意思是說，它所代表的值隨時可以改變），再將運算元與運算子結合，形成一個敘述，那此一敘述就可重複利用，例如，以上計算通常我們會利用

a=b*c;

來表示，b,c,a 就是運算元的代號，您也可以想像成分別代表長方形的長、寬與面積的代號，往後只要指派代號 b 與 c 的值，就可幫我們計算相乘的結果，並放到代號 a，且此公式可重複利用。例如，

```
int b=3;int c=2;int a;
a=b*c;
```

以上代號 a,b,c，在程式設計領域裡，我們稱爲『變數, variable』、『int b=3;』稱爲變數宣告與指派初值；『b*c』稱爲『運算式, Expression』；『a=b*c;』

稱爲『敘述, Statement』。其次，在第四單元的資料數位化單元，我們已經介紹不同的數字（整數、負整數與實數）與不同的數字的大小（數字的位數），其數位化的方式也不同，所佔用的記憶體大小也不同。電腦爲了有效率的處理這些資料，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也可以，但這樣非常浪費空間，還有，搬運時也很耗時。Arduino 所提供的資料型態如下表：

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
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	
char	字元	8	-128~127	
bool	布林	8	true or false	boolean 也可，但不鼓勵
String	字串			

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。在一些舊式的 Basic 語言中，變數使用前並不需要事先宣告，卻也帶來極大的困擾。例如，以下敘述即爲變數未宣

告的結果，編譯器便無法回應使用者在拼字上的錯誤，而造成除錯上的困難。

```
student=studend+1;
```

上式若事先宣告 student 如下：

```
int student;
```

則編譯器遇到 studend 時，便會出現 studend 未宣告的錯誤訊息，提醒使用者補宣告或注意拼字錯誤。其次，變數宣告的優點是可配置恰當的記憶體而提高資料的處理效率。例如，有些變數的值域僅爲整數，則不用宣告爲 float。此即爲小東西用小箱子裝，大東西用大箱子裝，才能有效運用空間與提升搬運效率。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。又例如，

int d;

那 d 就佔用 2 個 Byte,但可儲存-32768 到 32767。又例如，

```
char b,c;
```

則是宣告變數 b, c 爲 char 型態，此種型態可儲存單一字元，例如

b='A';c='*';

變數的宣告亦可連同初值一起設定，如以下敘述：

```
float d = 30.2;
```

宣告 d 是 float 型態，並設其初值是 30.2。以下敘述，宣告 e 是 char 型態，且其初值是字元'a'。

```
char e='a';
```

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：

```
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";
```

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。

『=』

『=』符號爲指派運算子，其作用爲將運算符號右邊運算式的值，指派給運算符號左邊的運算元。所以，以下敘述 sum=a+b 是先計算 a+b 的值，再將此值指派給 sum。

```
int sum = 0,a = 3,b = 5;
```

```
sum = a + b;
```

上式與數學的『相等』符號是不同的，請您一定要將它想成，右邊先運算，運算後再將結果指派給左邊的變數，所以不要一直懷疑爲什麼 0 會等於 8。其次，你是不能將常數放在指派運算子的左邊，例如：

```
8 = x ;
```

爲一個不合法的敘述，但以下敘述將常數 8 指派給變數 x 是合法的。

```
x = 8 ;
```

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
```

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為整數
Serial.println(xf/y);//1.25
Serial.println(x/yf);//1.25
Serial.println(xf/yf);//1.25
```

其次，若運算結果爲實數，但若指派給整數型態的變數，結果也是整數。例如，

```
   void setup() {
    Serial.begin(9600);
    int x=5, y=4,z;
    float xf=5,yf=4,zf;
    zf=xf/yf;//1.25
    Serial.println(zf);
    z=xf/yf;//1
    Serial.println(z); //原本運算結果是實數 1.25，但指派給整數，所
以會是 1
}void loop() {}
```

整數除法與取餘的應用

整數除法與取餘可以將一個整數分解爲數個數字。例如，若要使用七段顯示器顯示 152，那就要將此數字分解爲 1, 5, 2 三個數字，其方法如下：

```
   int a=152;
   int a1=152/100;//1 百位數
   int a2=(152-a1*100)/10;//5 十位數
   int a3=a %10;//個位數
```

取餘

若要讓一個數字在累加的過程中，保持一定的循環，那也是用取餘。例如，

```
   int i;
   void loop() {
      i=(i+1) %4;
   }
```

那 i 就永遠在 0,1,2,3 循環。

▶範例 10a

請寫一個程式，可以指派長方形的長與寬，並計算周長與面積。

➲ 題目分析

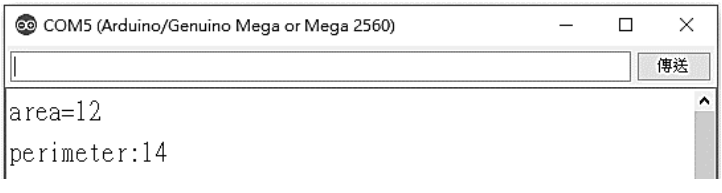
需要兩個變數儲存長與寬。請選擇此資料來源是整數還是浮點數，若是整數，還要分是 8 位元的 byte (0～255)、16 位元的 int (-32768～32767)或 32 位元的 long (-2147483648～2147483647)。本書整數若沒特別註明，就一律折衷，通通取 int。

2. 需要兩個變數儲存面積與周長。請選擇整數或是浮點數，本例也是選擇 int。

3. 先使用變數指派方式，指派變數的值，程式如下。這樣可以省略冗長的變數輸入，先專注於演算法的實現。

```
void setup() {
    Serial.begin(9600);
    int a,b,area,perimeter;
    a=3;
    b=4;
    area= a*b;
    perimeter=2*(a+b);
    Serial.print("area=");
    Serial.println(area);
    Serial.print("perimeter:");
    Serial.println(perimeter);
}void loop() {}
```

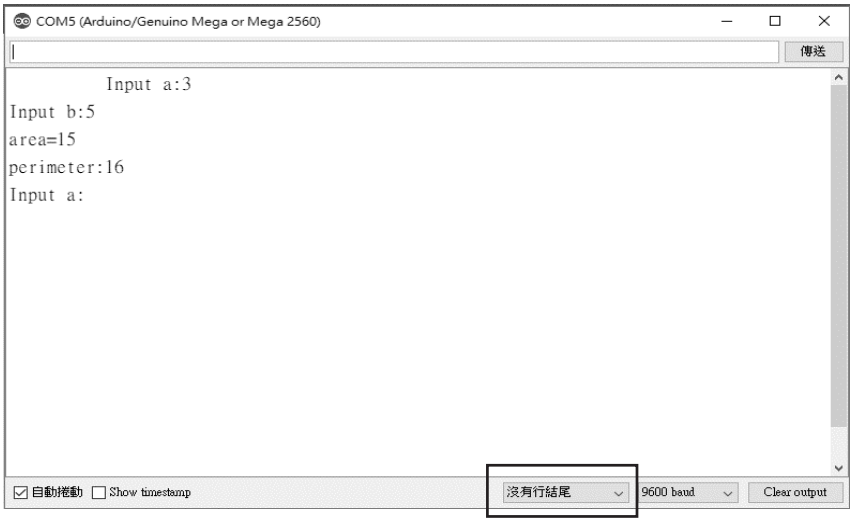
4. 以上程式執行結果如下：



5. 使用電腦的鍵盤輸入變數的數值，程式如下：

```
void setup() {
    Serial.begin(9600);
}
void loop() {
    int a,b,area,perimeter;
    Serial.print("Input a:");
    while(Serial.available() ==0) {}//等待使用者輸入資料
    a=Serial.parseInt();
    Serial.println(a);
    Serial.print("Input b:");
    while(Serial.available() ==0) {}//wait for user's input
data
    b=Serial.parseInt();
    Serial.println(b);
    area= a*b;
    perimeter=2*(a+b);
    Serial.print("area=");
    Serial.println(area);
    Serial.print("perimeter:");
    Serial.println(perimeter);
  }
```

6. 以上程式執行結果如下：(請將輸入方式點選『沒有行結尾』，如下圖，不然無法輸入第二個變數)



➲ 自我練習

1. 指派長方體的長、寬、高，計算其表面積與體積。
2. 使用電腦的鍵盤輸入長方體的長、寬、高，計算其表面積與體積。
3. 請寫一個程式，可以輸入一個台斤數，且轉爲公斤數輸出。
4. 請寫一個程式，可以任意輸入一個 0～86399 的整數，且轉爲『時：分：秒』的格式輸出。

▶範例 10b

假設有 5 筆資料如下，

55,66,77,88,99

請您寫程式計算總和與平均。

➲ 思考步驟

1. 選擇適當的資料結構。本例可使用單一變數，也可使用一維陣列。
2. 選擇資料型態。本例資料來源都在 0～255 之間，嚴格來說選用 byte 就可以。所以若資料結構使用單一變數，那變數命名與宣告如下：（但一般我們都便宜行事，整數一律選用 int）

```
byte a1=55,a2=66,a3=77,a4=88,a5=100;
```

3. 5 個數字的總和一定超過 255，所以選用 int，程式如下。

```
int sum=a1+a2+a3+a4+a5;
```

以上亦可分開撰寫如下：

```
int sum;
sum=a1+a2+a3+a4+a5;
```

4. 平均通常取小數一位，所以選用 float，程式如下：

```
float avg=sum/5.0;//實數相除運算，5.0 是實數
```

上式若寫成

```
float avg=sum/5;//整數相除運算，5 是整數
```

那會將 sum/5 先取整數，再指派給 avg，請自行比較其差別。

5. 以上全部程式如下：

```
void setup() {
    Serial.begin(9600);
    byte a1=55,a2=66,a3=77,a4=88,a5=100;
    int sum=a1+a2+a3+a4+a5;
    float avg=sum/5.0;
    //float avg=sum/5;
    Serial.println(sum);
    Serial.println(avg);
}void loop() {}
```

6. 以上 5 筆資料，還可使用 5 個單一變數儲存，但若資料數量一多，那就要選用陣列結構了，不然程式會非常冗長。本例使用一維陣列宣告與初值指派如下：

```
byte a[]={0,55,66,77,88,100};
```

陣列索引從 0 開始，但是索引 0 可用可不用，因爲很多人不習慣從 0 號開始，所以索引 0，可放一個 0，如以上敘述，那就可以跳掉索引 0，資料就從索引 1 開始存放，本例資料 55，就放在 a[1]，資料 66 就放在 a[2]...依此類推。

7. 以上使用陣列儲存資料，那程式就可使用迴圈，且配合陣列索引存取資料，那資料多時，程式才會簡潔，所以程式如下，

```
void setup() {
    Serial.begin(9600);
    byte a[]={0,55,66,77,88,100};
```

```
    int sum=0;
    float avg;
    for(int i=1;i<=5;i++){
      sum=sum+a[i];
    }
    avg=sum/5.0;
    Serial.println(sum);
    Serial.println(avg);
}void loop() {}
```

識別字（Identifiers）命名規則

眞實的世界裏，每個人、事及物都有一個名稱，程式設計亦不例外，於程式設計時我們必須爲每一個變數、常數、函式命名，以上所有變數、常數、函式等名稱，統稱爲程式語言的識別字(Identifiers)。先以數學爲例，數學的變數命名規則是，已知數用 a, b, c，未知數用 x, y, z；物理則習慣用 v 代表速度，t 代表時間；而 C/C++/Arduino 語言的識別字命名規則如下：

1. 識別字必須是以字母（大小寫的 A 至 Z）或底線(_)開頭。例如，以下是一些合法的識別字。

   a
   i
   sum
   _sum
   Income

   以下是一些非法的識別字。

   ```
   7eleven        //  不能由數字開頭
   %as            //  不能由符號開頭
   ```

2. 識別字由字母開頭後，僅可由字母、底線及數字組合而成，且不得包含空白與符號。例如，以下是一些合法的識別字。

```
a123
a123b
_a_b
```

以下是一些非法的識別字。

```
A=              //  不能含有 = 號
sum!            //  不能含有 ! 號
Age#3           // 不能含有 # 號
a c             //  不能含空白
c+3             //  不得含有加號
```

3. 識別字的大小寫均視爲不同，例如 Score、score 及 SCORE 皆代表不同的識別字。
4. 識別字不得使用保留字，如 if、for 等。但是，if1、fora 等則可使用。
5. 識別字要用有意義的單字，例如，sum、avg、StudentNumber 或 AverageIncome。除非有效範圍很小（或稱生命週期極短）的變數才用 x、i 或 a 等當識別字，也千萬不要用 k23erp 等這種沒意義又難記的識別字。
6. 識別字有多個單字時，中間可以加上底線（_），例如上例的 StudentNumber 可寫成 student_Number，若擔心打字不靈光亦可寫成 Stu_Num、stu_num、stuNum 或 stunum，其中 stuNum 又稱駝峰表示法，因爲大寫字母看起來像駱駝駝峰一樣，這樣可以避免鍵入底線的困擾、且提昇閱讀效率，Arduino 的函數命名則採用此種命名習慣。

保留字（Keywords）

保留字（又稱關鍵字）是任一程式語言已事先賦予某一識別字（可識別的文字或字串，稱爲識別字）一個特別意義，所以程式設計者不得再重

複賦予不同的用途，就像現實生活中，電視、冰箱已經有特別意義了，所以沒有人取名字爲電視或冰箱。Arduino 也是一樣，例如 if 已賦予決策功能，程式設計者當然不得再定義 if 爲另外的用途，以下是 Arduino 語言的保留字，例如表內的 loop、setup、break、coutinue、do 等等，大部分擷取 C/C++語言精華。

STRUCTURE

The elements of Arduino (C++) code.

Sketch	Arithmetic Operators	Pointer Access Operators
loop()	% (remainder)	& (reference operator)
setup()	* (multiplication)	* (dereference operator)
	+ (addition)	
Control Structure	- (subtraction)	**Bitwise Operators**
break	/ (division)	& (bitwise and)
continue	= (assignment operator)	<< (bitshift left)
do...while		>> (bitshift right)
else	**Comparison Operators**	^ (bitwise xor)
for	!= (not equal to)	| (bitwise or)
goto	< (less than)	~ (bitwise not)
if...else	<= (less than or equal to)	
return	== (equal to)	**Compound Operators**
switch...case	> (greater than)	&= (compound bitwise and)
while	>= (greater than or equal to)	*= (compound multiplication)
		++ (increment)
Further Syntax	**Boolean Operators**	+= (compound addition)
#define (define)	! (logical not)	-- (decrement)
#include (include)	&& (logical and)	-= (compound subtraction)
/* */ (block comment)	|| (logical or)	/= (compound division)
// (single line comment)		^= (compound bitwise xor)
; (semicolon)		|= (compound bitwise or)

➲ 自我練習

1. 請寫一程式，可以計算全班程式設計考試的總分與平均。
2. 請寫一程式，可以連續輸入 5 個數字，並輸出此 5 個數字、求其和與平均。
3. 同上題，輸出完畢，還可輸入座號，而得其分數。

單元 11
決策指令與比較運算子

人類的生活必須不斷面對決策問題，連我家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~，例如天色暗了就點燈，否則不理會；第二是多向分歧決策流程的 switch...case，例如你身上有 5000 元，走進一家五星級的大飯店用餐，你的分歧點就很多，有自助餐、中式套餐、日本料理、泰國餐點等等分歧點。本章的重點即是探討 Arduino 語言的決策流程敘述。其次，現代正夯的機器人、自駕車更要時時刻刻不斷判斷，這都要使用決策敘述。

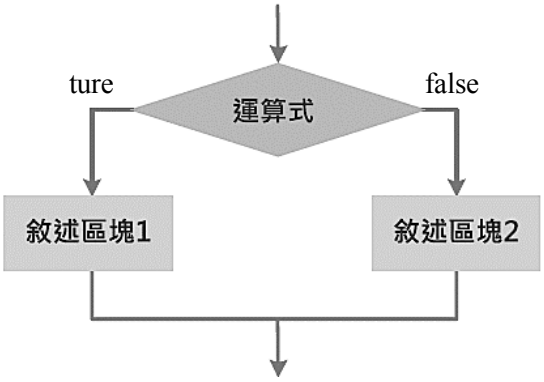
if...else

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；
}]
```

以上語法說明如下：

1. 運算式的值若爲 1 (true)，則執行敘述區塊 1；運算式的值若爲 0 (false)，則執行敘述區塊 2，其流程圖如下：



2. 敘述區塊的範圍請使用大括號({})包圍。例如，以下敘述可依 a 的大小評量其及格與否。請善用程式的縮排效果，這樣才方便辨識與閱讀。(縮排是按 TAB 鍵，不是逐一按空白鍵。)

```
if (a>=60){
   b=1;
}
else {
   b=0;
}
```

3. 敘述區塊內的敘述若只有一個，則敘述區塊上下兩個大括號可予省略。例如，以上程式可簡化如下：

```
if (a>=60)
   b=1;
else
   b=0;
```

4. 有時爲了簡化程式的撰寫，可將否則的部分寫在 if 前面，並省略 else。例如，以下程式同義於上面程式（語法的中括號，代表此部分可省略）。

```
b=0;
if(a>=60)
   b=1;
```

5. 若省略敘述區塊上下的大括號，則條件成立時，僅執行敘述區塊的第一個敘述。例如，以下敘述執行之後，b=0，c=1。

```
a=55;b=0;c=0;
if(a>=60)
   b=1;
   c=1;   //此敘述並不會因為縮排而屬於 if 敘述的一部份
Serial.println(b);    //  b=0, c=1
Serial.println(c);
```

6. if()後面不要緊接分號，例如，

```
a=55;b=0;c=0;
if(a>=0);
   b=1;
   c=1;
Serial.println(b);   //  b=1, c=1 */
Serial.println(c);
```

這是初學者常犯的錯誤，因爲 if 遇到分號（；），表示此指令已經結束。

7. 敘述區塊內可以放置任何合法敘述，當然也可以再放置 if；if 中有 if，稱爲巢狀 if，請看以下範例 11a。

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依據。下表是 Arduino 語言中的關係運算子符號，這些都和 C/C++相同。例如，以下敘述用來比較 a 與 b 是否相等。

```
if (a == b)
    Serial.println("Equal");
```

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
```

➲ 自我練習

1. 請將以上運算式於 Arduino 環境鍵入、編譯、執行，觀察執行結果。

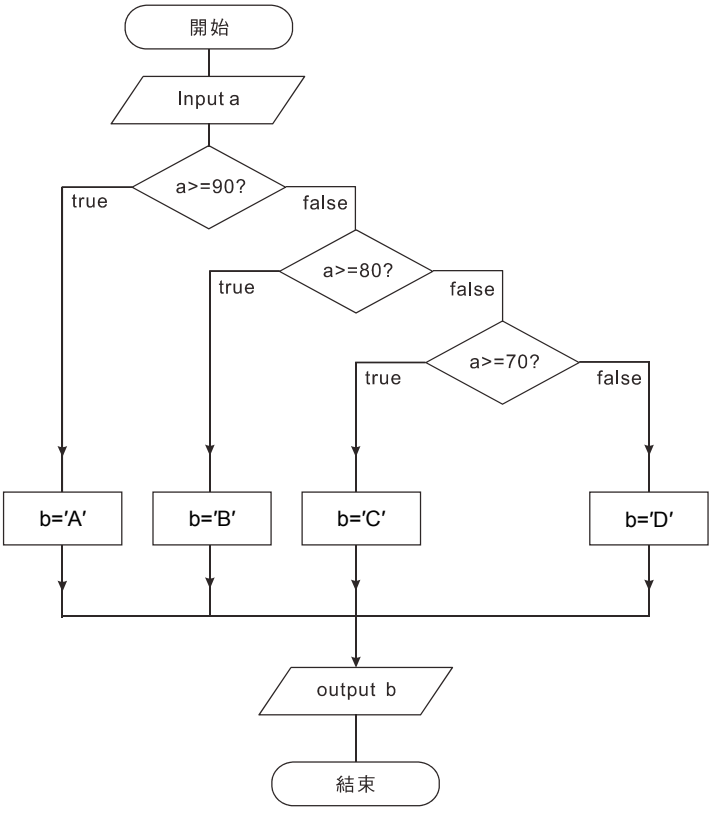
▶範例 11a

試寫一程式，完成以下要求：

1. 輸入一個 0～100 的分數。
2. 當分數大於等於 90 分時，輸出 A。
3. 當分數介於 80～89 時，輸出 B。
4. 當分數介於 70～79 時，輸出 C。
5. 當分數介於 0～69 時，輸出 D。

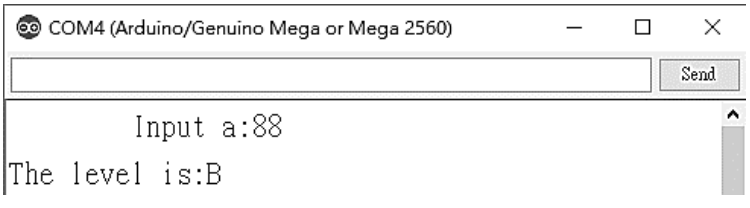
➲ 流程分析

1. 使用流程圖分析如下：



2. 以上每一個決策流程點，都有兩個分歧點，所以適用 if~else。
3. 每一個 else 後面均需進一步決策流程，所以可在每一 else 後面再放置 if。

➲ 執行結果



➲ 程式列印

```
void setup() {
   Serial.begin(9600);
}
void loop() {
   int a;
   char b;
   Serial.print("Input a:");
   while(Serial.available() ==0) {}
   a=Serial.parseInt();
   Serial.println(a);
   if(a>=90)              /* 高於 90 分為 A */
      b='A';
   else
      if(a>=80)           /* 介於 80 與 90 分為 B */
         b='B';
      else
         if(a>=70)        /* 介於 70 與 80 分為 C */
            b='C';
         else
            b='D';        /* 不符合上述情況則為 D */
   Serial.print("The level is:");
   Serial.println(b);
}
```

➲ 補充說明

1. 此題有人會寫成 if (90<a<100)，但 Arduino 並沒有這種運算式，因為 90<a<100 是數學語言，不是程式語言。
2. 有人會寫成

```
if (a<100 & a>=90) b='A';
if (a<90  & a>=80)  b='B';
if (a<80  & b>=70)  b='C';
if (a<70  & b>=0)   b='D';
```

這樣雖然也可以，但是其執行效率非常差，因為不管分數為何，都

要執行四次判斷，所以這樣效率很低。

3. 這題有人會寫成如下：

```
if(a>=90)           /* 高於 90 分為 A */
    b='A';
else if(a>=80)      /* 介於 80 與 90 分為 B */
    b='B';
else if(a>=70)      /* 介於 70 與 80 分為 C */
    b='C';
else
    b='D';      /* 不符合上述情況則為 D */
```

這樣當然可以，而且這樣不用一再縮排，因爲程式一再縮排，程式會右移，那程式很容易被分割爲兩列，因爲分割兩列後的程式，其可讀性會降低。

➲ 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，產生 10 個-3 到 3 的亂數，並統計負數、0、正數的個數。
3. 請寫一個程式，完成以下要求：
   (1) 產生一個 0～25 的亂數。
   (2) 當亂數大於等於 20 時，輸出五個燈。
   (3) 當亂數是 16～19 時，輸出四個燈。
   (4) 當亂數是 11～15 時，輸出三個燈。
   (5) 當亂數是 0～10 時，輸出兩個燈。
4. 假設自來水費率如下：
   100 度以下，每度 3 元。
   100～300 度，超過 100 度的部分，每度 5 元。

300 度以上，超過 300 度的部分，每度 6 元。

根據以上條件，寫一程式，可輸入用水度數，得到水費。例如，測資如下表：

度數	水費
50	50＊3＝150
200	100＊3＋100＊5＝800
400	100*3+200*5+100*6=1900

5. 假設電話通話費每 5 秒 1 元，不足 5 秒以 5 秒計，請產生 1 個 0 到 100 的亂數，並計算其通話費。例如，通話 12 秒，3 元；28 秒，6 元。
6. 假設所得稅稅率累進法則如下：
   (1) 淨所得 30 萬以下繳納 6%。
   (2) 淨所得 30～80 萬之間，則前面 30 萬繳納 6%，超過 30 萬的部分繳納 13%。
   (3) 淨所得在 80～200 萬之間繳納 21%。(前面 30 萬繳 6%，30～80 萬之間繳 13%)
   (4) 淨所得超過 200 萬，超過的部分繳納 30%。

   試寫一程式可以輸入淨所得，並計算應繳納稅額。測資如下：

淨所得	納稅額
20 萬	20 萬*6%=12000
40 萬	30 萬×6%+10 萬×13%= 31000
100 萬	30 萬×6%+50 萬×13%+20 萬*21% =125000
220 萬	30 萬×6%+50 萬×13%+120 萬*21% +20 萬＊30%=395000

7. 假設計程車的計費方式如下：

   1000 公尺以內 60 元。超過 1000 公尺時，

   (1) 日間以每 500 公尺加收 6 元，不足 500 公尺時，以 500 公尺計

算。

(2) 夜間以每 300 公尺加收 8 元，不足 300 公尺，以 300 公尺計算

請根據以上條件，寫一程式完成車資的計算。本例，日間輸入 1，夜間輸入 0，再輸入一個里程數，並計算車資。測資如下表：

模式	里程	車資	模式	里程	車資
1	500	60	0	500	60
1	1800	60+6*2=72	0	1800	60+8*3=84

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；〕
  }
```

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. 電腦將會依 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。

```
void setup() {
   Serial.begin(9600);
   byte a=1;
   String b="";
   switch(a)
   {
      case 1:
        b="Spring";     /*春*/
        break;
      case 2:
        b="Summer";   /*夏*/
        break;
      case 3:
        b="Fall";         /*秋*/
        break;
      case 4:
        b="Winter";     /*冬*/
        break;
    default :
      b="input error";
   }
   Serial.println(a);
```

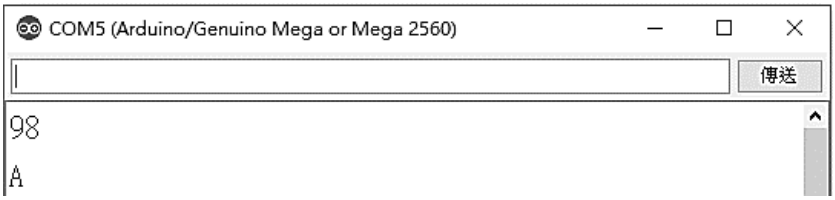
```
    Serial.println(b);
}void loop() {}
```

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。

▶範例 11b

試以 switch case 重作範例 11a。

➲ 執行結果



➲ 程式列印

```
void setup() {
    Serial.begin(9600);
    byte a=98;
    String b="";
    switch(a/10) {
       case 10:
       case 9:
          b="A";
          break;//do not leave out
      case 8:
          b="B";
          break;
      case 7:
          b="C";
          break;
      case 6:
      case 5:
```

```
      case 4:
      case 3:
      case 2:
      case 1:
      case 0:
         b="D";
         break;
  }
    Serial.println(a);
    Serial.println(b);
}void loop() {}
```

➲ 自我練習

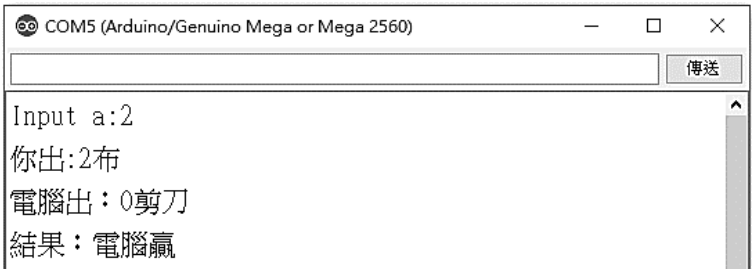
1. 請嘗試將 b=″A″;下一列的 break;去掉，並輸入 92 分，觀察執行結果。
2. 請將範例 11a 的自我練習 1～5，以 select case 重作。
3. 請產生 0 到 8 的亂數，並依照亂數點亮對應 LED。例如，產生 3，那就亮第 3 個 LED。
4. 請產生 0 到 9 的亂數，並由七段顯示器顯示。
5. 請寫程式完成以下條件。
   (1) 產生 20 個 1 到 6 的亂數。
   (2) 以陣列儲存以上資料。
   (3) 輸出以上資料。
   (4) 計算並輸出共有多少個 3。
6. 請寫程式完成以下條件。
   (1) 產生 20 個 1 到 6 的亂數。
   (2) 以陣列儲存以上資料。
   (3) 輸出以上資料。
   (4) 統計與輸出所有數字出現的次數。

7. 請寫程式完成以下條件。
   (1) 產生 50 個 0 到 100 的亂數。
   (2) 以陣列儲存以上資料。
   (3) 輸出以上資料。
   (4) 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。

▶範例 11c

猜拳遊戲。請寫一個程式，可以由人和電腦猜拳。

➲ 執行結果



➲ 程式列印

```
void setup(){
  Serial.begin(9600);
  randomSeed(analogRead(0));
}
void loop() {
    int a,b;
    String a1,b1,r="";
    Serial.print("Input a:");
    while(Serial.available() ==0) {}  //在此等待使用者輸入
    a=Serial.parseInt();
    Serial.println(a);
    b = random(0, 2);// get a random number from 0 to 2
    switch (a){
       case 0:
```

```
          a1="剪刀";
          switch(b){
             case 0:
                b1="剪刀";
                r ="平手" ;
              break;
             case 1:
                b1="石頭";
                r ="電腦贏" ;
                break;
             case 2:
                b1="布";
                r ="你贏";
                break;
             }
          break;
       case 1:
          a1="石頭";
          switch(b){
             case 0:
                b1="剪刀";
                r ="你贏" ;
                break;
             case 1:
                b1="石頭";
                r ="平手" ;
                break;
             case 2:
                b1="布";
                r ="電腦贏";
                break;
          }
          break;
       case 2:
          a1="布";
          switch(b){
             case 0:
                b1="剪刀";
                r ="電腦贏";
                break;
```

```
                case 1:
                    b1="石頭";
                    r ="你贏";
                    break;
                case 2:
                    b1="布";
                    r ="平手";
                    break;
            }
            break;
        }
        Serial.print("你出:");
        Serial.print(a);
        Serial.println(a1);
        Serial.print("電腦出：");
        Serial.print(b);
        Serial.println(b1);
        Serial.print("結果：");
        Serial.println(r);
        delay(2000);
}
```

➲ 自我練習

1. 請寫一程式，可以執行三個人的猜拳遊戲，並評定勝負（可讓電腦產生兩個亂數，使用者輸入一個，且由序列埠監控視窗輸出結果。）

單元 12
指撥開關

前面的 LED 與七段顯示器都是輸出設備，本單元起要開始介紹一些輸入設備，例如。指撥開關與按壓開關。

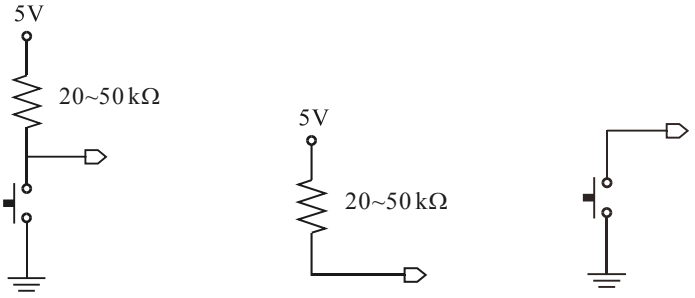
數位輸入

Arduino 當輸入時，有兩種可能，一種是接收其他 IC 的輸出，這時只要設定其爲 INPUT 即可，但若要接收按壓開關或指撥開關等開關裝置，則可設定其具有上拉電阻(INPUT_PULLUP)，這樣可以簡化外部電路，Arduino 說明文件如下圖。

> Properties of Pins Configured as INPUT_PULLUP
>
> There are 20K pullup resistors built into the Atmega chip that can be accessed from software. These built-in pullup resistors are accessed by setting the pinMode() as INPUT_PULLUP. This effectively inverts the behavior of the INPUT mode, where HIGH means the sensor is off, and LOW means the sensor is on.
>
> The value of this pullup depends on the microcontroller used. On most AVR-based boards, the value is guaranteed to be between 20kΩ and 50kΩ. On the Arduino Due, it is between 50kΩ and 150kΩ. For the exact value, consult the datasheet of the microcontroller on your board.
>
> When connecting a sensor to a pin configured with INPUT_PULLUP, the other end should be connected to ground. In the case of a simple switch, this causes the pin to read HIGH when the switch is open, and LOW when the switch is pressed.
>
> The pullup resistors provide enough current to dimly light an LED connected to a pin that has been configured as an input. If LEDs in a project seem to be working, but very dimly, this is likely what is going on.
>
> The pullup resistors are controlled by the same registers (internal chip memory locations) that control whether a pin is HIGH or LOW. Consequently, a pin that is configured to have pullup resistors turned on when the pin is an INPUT, will have the pin configured as HIGH if the pin is then switched to an OUTPUT with pinMode(). This works in the other direction as well, and an output pin that is left in a HIGH state will have the pullup resistors set if switched to an input with pinMode().

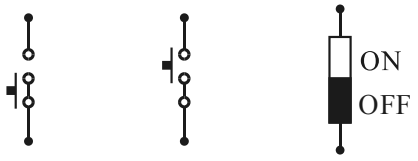
開關的標準電路如下圖左，按鍵沒按壓是高電位，壓下去是低電位，這樣就可以判斷開關有沒有被按。其次，Arduino 爲了讓使用者簡化電路，

此上拉電阻(下圖中)可用軟體指派,所以使用者只要接一個開關就好,如下圖右。(補充說明,沒上拉電阻可以嗎?當然不可以,因爲開關沒按壓時,Arduino 輸入腳位浮接(懸空未接稱爲浮接),此時就無法判定其電壓的高與低了。



➲ 指撥開關

指撥開關的內部結構如下圖,開關可上下滑動,下圖左是下滑,電路處於斷路,下圖中往上滑動,則電路接通,往後爲了簡化電路,均以下圖右表示指撥開關。



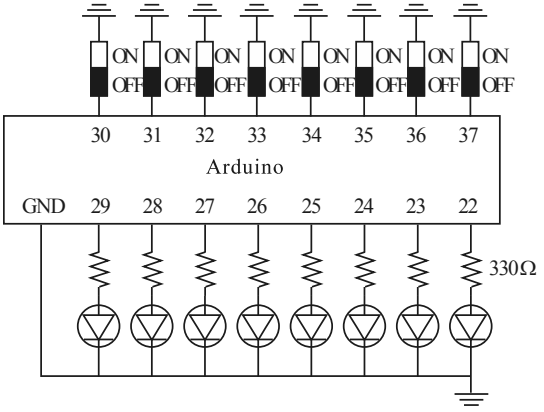
▶範例 12a

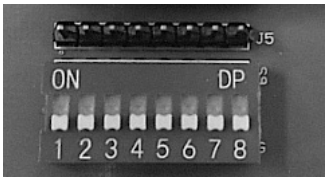
示範使用指撥開關。

➲ 操作步驟

1 完成以下電路,其中 LED 同上範例 4a。

2 指撥開關的實驗板實體圖如下圖右,杜邦線從 J5 連接到 Arduino 微控板對應腳位即可,Gnd 內部已經連接。





3 digitalRead()一次僅讀取指定腳位電壓，請鍵入以下程式，並觀察執行結果，請留意開關 ON，其值是 LOW，LED 滅掉。

```
void setup() {
    Serial.begin(9600);
    pinMode(37,INPUT_PULLUP);//PC0
    DDRA=B11111111;
    PORTA=0;
}
void loop() {
    byte b=digitalRead(37);//讀一個位元
    Serial.println(b);
    digitalWrite(22,b);   //寫入一個位元
}
```

4 PINC 可一次讀取 PORTC 暫存器 8 隻腳全部電壓，請鍵入以下程式，並觀察執行結果，請留意開關 ON，其值是 LOW，LED 滅掉。

```
void setup() {
    Serial.begin(9600);
    pinMode(30,INPUT_PULLUP);//PC7
    pinMode(31,INPUT_PULLUP);//PC6
    pinMode(32,INPUT_PULLUP);//PC5
    pinMode(33,INPUT_PULLUP);//PC4
    pinMode(34,INPUT_PULLUP);//PC3
    pinMode(35,INPUT_PULLUP);//PC2
    pinMode(36,INPUT_PULLUP);//PC1
```

```
    pinMode(37,INPUT_PULLUP);//PC0
    DDRA=B11111111;
    PORTA=0x00;
}
void loop() {
    byte b=PINC;//讀 8 個位元
    Serial.println(b);
    PORTA=b;
}
```

5 請將以上程式的 8 個 INPUT_PULLUP 通通改爲 INPUT，程式如下，並觀察執行結果。

```
    pinMode(37,INPUT);//PC0
```

指撥開關通常可用來作開關，或運作模式的選擇，請看以下範例說明。

▶範例 12b

請用 3 位元指撥開關，分別控制 3 個 LED 的明滅，且 ON 時才亮。

➲ 操作步驟

1 使用 digitalRead()函式，每次讀取 1 位元，且逐一控制對應 LED，程式如下：

```
void setup() {
    Serial.begin(9600);
    pinMode(35,INPUT_PULLUP);//PC2
    pinMode(36,INPUT_PULLUP);//PC1
    pinMode(37,INPUT_PULLUP);//PC0
    DDRA=B11111111;
    PORTA=0;
}
void loop() {
    byte b;
    b=digitalRead(35);//讀一個位元
```

```
    Serial.println(b);//輸出看看，確認是否正確
    if (b==0)//==  比較是否相等運算子
      digitalWrite(24,HIGH); //用常數符號表示高電位 5V
    else
      digitalWrite(24,LOW);   //表示低電位 0V
    b=digitalRead(36);//讀一個位元
    Serial.println(b);//輸出看看，確認是否正確
    if (b==0)//== 是比較是否相等運算子
      digitalWrite(23,HIGH); //用常數符號表示高電位 5V
    else
      digitalWrite(23,LOW);   //表示低電位 0V
    b=digitalRead(37);//讀一個位元
    Serial.println(b);//輸出看看，確認是否正確
    if (b==0)//== 是比較是否相等運算子
      digitalWrite(22,HIGH); //用常數符號表示高電位 5V
    else
      digitalWrite(22,LOW);   //表示低電位 0V
}
```

2 以上我們用腳位編號寫程式，但有可能您要改變腳位，此時您就要到處修改，所以我們可將這些腳位，通通在程式最前面用常數符號（以下簡稱常數）代替，這樣當您要改變腳位時，只要在最前面修改就行，不用到處修改，若沒有使用常數，那就要到處修改，萬一沒有完全修改，那就造成程式執行不一致。將以上腳位定義常數的程式如下：

```
const byte s2=35;//switch2
const byte s1=36;
const byte s0=37;
const byte l2=24;//led2
const byte l1=23;//led1
const byte l0=22;//led0
void setup() {
    Serial.begin(9600);
    pinMode(s2,INPUT_PULLUP);//PC2
    pinMode(s1,INPUT_PULLUP);//PC1
    pinMode(s0,INPUT_PULLUP);//PC0
```

```
    DDRA=B11111111;
    PORTA=0;
}
void loop() {
    byte b;
    b=digitalRead(s2);//讀一個位元
    Serial.println(b);//輸出看看，確認是否正確
    if (b==0)//== 是比較是否相等運算子
      digitalWrite(l2,HIGH); //用常數符號表示高電位 5V
    else
      digitalWrite(l2,LOW);   //表示低電位 0V
    b=digitalRead(s1);//讀一個位元
    Serial.println(b);//輸出看看，確認是否正確
    if (b==0)//== 是比較是否相等運算子
      digitalWrite(l1,HIGH); //用常數符號表示高電位 5V
    else
      digitalWrite(l1,LOW);   //表示低電位 0V
    b=digitalRead(s0);//讀一個位元
    Serial.println(b);//輸出看看，確認是否正確
    if (b==0)//== 是比較是否相等運算子
      digitalWrite(l0,HIGH); //用常數符號表示高電位 5V
    else
      digitalWrite(l0,LOW);   //表示低電位 0V
}
```

3 使用 PINC 讀取 3 位元。本例一次讀 8 位元，但是我們僅用 3 個位元，沒有用到的 bit7, 6, 5, 4, 3 等 5 位元，請用位元運算子的 and 運算子『&』強制遮罩掉。

```
const byte s2=35;//switch2
const byte s1=36;
const byte s0=37;
const byte l2=24;//led2
const byte l1=23;//led1
const byte l0=22;//led0
void setup() {
    Serial.begin(9600);
    pinMode(s2,INPUT_PULLUP);//PC2
    pinMode(s1,INPUT_PULLUP);//PC1
```

```
    pinMode(s0,INPUT_PULLUP);//PC0
    DDRA=B11111111;
    PORTA=0;
}
void loop() {
    byte b;
    b=PINC;//讀 8 個位元
    Serial.print(b);//輸出看看，確認是否正確
    Serial.print(',');
    b=b & B00000111;//遮罩沒用到的位元 7,6,5,4,3
    Serial.println(b);//輸出看看，確認是否正確
    switch (b){
      case (B000):
        digitalWrite(l2,HIGH);
        digitalWrite(l1,HIGH);
        digitalWrite(l0,HIGH);
        break;//語法，不可漏掉
      case (B001):
        digitalWrite(l2,HIGH);
        digitalWrite(l1,HIGH);
        digitalWrite(l0,LOW);
        break;//不可漏掉
      case (B010):
        digitalWrite(l2,HIGH);
        digitalWrite(l1,LOW);
        digitalWrite(l0,HIGH);
        break;//不可漏掉
      case (B011):
        digitalWrite(l2,HIGH);
        digitalWrite(l1,LOW);
        digitalWrite(l0,LOW);
        break;//不可漏掉
      case (B100):
        digitalWrite(l2,LOW);
        digitalWrite(l1,HIGH);
        digitalWrite(l0,HIGH);
        break;//不可漏掉
      case (B101):
        digitalWrite(l2,LOW);
        digitalWrite(l1,HIGH);
```

```
        digitalWrite(l0,LOW);
        break;//不可漏掉
      case (B110):
        digitalWrite(l2,LOW);
        digitalWrite(l1,LOW);
        digitalWrite(l0,HIGH);
        break;//不可漏掉
      case (B111):
        digitalWrite(l2,LOW);
        digitalWrite(l1,LOW);
        digitalWrite(l0,LOW);
        break;//不可漏掉
    }
}
```

4 本例若不習慣指撥開關 ON 是低電位，那也可以先使用位元反相運算子『～』，將每一位元都反相，那程式如下：

```
const byte s2=35;//switch2
const byte s1=36;
const byte s0=37;
const byte l2=24;//led2
const byte l1=23;//led1
const byte l0=22;//led0
void setup() {
    Serial.begin(9600);
    pinMode(s2,INPUT_PULLUP);//PC2
    pinMode(s1,INPUT_PULLUP);//PC1
    pinMode(s0,INPUT_PULLUP);//PC0
    DDRA=B11111111;
    PORTA=0;
}
void loop() {
    byte b;
    b=PINC;//讀 8 個位元
    Serial.print(b);//輸出看看，確認是否正確
    Serial.print(',');
    b=~b;//每一位元皆反相
    Serial.print(b);//輸出看看，確認是否正確
```

```
  Serial.print(',');
  b=b & B00000111;//遮罩沒用到的位元 7,6,5,4,3
  Serial.println(b);//輸出看看，確認是否正確
  switch (b){
    case (B000):
      digitalWrite(l2,LOW);
      digitalWrite(l1,LOW);
      digitalWrite(l0,LOW);
      break;//語法，不可漏掉
    case (B001):
      digitalWrite(l2,LOW);
      digitalWrite(l1,LOW);
      digitalWrite(l0,HIGH);
      break;//不可漏掉
    case (B010):
      digitalWrite(l2,LOW);
      digitalWrite(l1,HIGH);
      digitalWrite(l0,LOW);
      break;//不可漏掉
    case (B011):
      digitalWrite(l2,LOW);
      digitalWrite(l1,HIGH);
      digitalWrite(l0,HIGH);
      break;//不可漏掉
    case (B100):
      digitalWrite(l2,HIGH);
      digitalWrite(l1,LOW);
      digitalWrite(l0,LOW);
      break;//不可漏掉
    case (B101):
      digitalWrite(l2,HIGH);
      digitalWrite(l1,LOW);
      digitalWrite(l0,HIGH);
      break;//不可漏掉
    case (B110):
      digitalWrite(l2,HIGH);
      digitalWrite(l1,HIGH);
      digitalWrite(l0,LOW);
      break;//不可漏掉
    case (B111):
```

```
        digitalWrite(l2,HIGH);
        digitalWrite(l1,HIGH);
        digitalWrite(l0,HIGH);
        break;//不可漏掉
    }

}
```

➲ 自我練習

1. 請用 3 位元指撥開關，當作 2 進位的輸入，OFF 代表 0，ON 代表 1，且用七段顯示器輸出其值。例如，000，輸出 0；110，輸出 6。
2. 運作模式的選擇。請寫一個程式，讓您的單向紅綠燈可以用 1 位元指撥開關設定 2 種運作模式，2 種運作模式如下：

模式	綠燈時間	紅燈時間
0	5	5
1	閃黃燈	閃紅燈

3. 運作模式的選擇。請寫一個程式，讓您的單向紅綠燈可以用 2 位元指撥開關設定 4 種運作模式，4 種運作模式如下：

模式	綠燈時間	紅燈時間
0	5	5
1	7	3
2	4	6
3	閃黃燈	閃紅燈

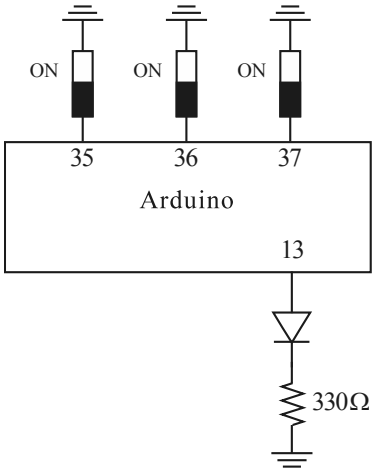
4. 同自我練習 2，但擴充爲雙向控制系統。

▶範例 12c

表決器。假設有一項評審工作，有 3 位評審，當其中兩人或以上同意，則表示過關且燈亮，請設計此電路。

➲ 電路設計

1. 本例先用指撥開關當作評審按鈕，當有 2 個或以上評審 on 時，表示通過，LED 亮，電路設計如下：(亦可用按壓開關較方便，請看第 15 單元)



2. 以上想法的程式列印如下：

```
const byte sw1=35;
const byte sw2=36;
const byte sw3=37;
const byte led=13;  //預植的 LED
byte a;
byte b;
void setup() {
    pinMode(sw1,INPUT_PULLUP);
    pinMode(sw2,INPUT_PULLUP);
    pinMode(sw3,INPUT_PULLUP);
    pinMode(led,OUTPUT);
    digitalWrite(led,LOW);
    DDRA=B111;//24,23,22
    Serial.begin(9600);
}
void loop() {
    a=PINC;
    a=~a;//逐位元反相
    a=a & B00000111;//將不要的位元遮罩
```

```
    PORTA=a;
    b=0;
    b=b+bitRead(a,0);
    b=b+bitRead(a,1);
    b=b+bitRead(a,2);
    Serial.println(b);
    if (b>=2)
       digitalWrite(led,HIGH);
    else
       digitalWrite(led,LOW);
}
```

3. 以上作法，程式較簡單，但是評審完，評審還要自己將指撥開關撥回來，有點不方便。待第 15 單元介紹按壓開關，再用按壓開關重做本範例。

➲ 自我練習

1. 同上範例，但每個人增加一個燈號，顯示自己的指撥狀態。
2. 假如有 5 位評審，三位以上通過就通過，請設計電路與程式。