計算機程式設計
C語言 Loop

郭忠義

jykuo@ntut.edu.tw

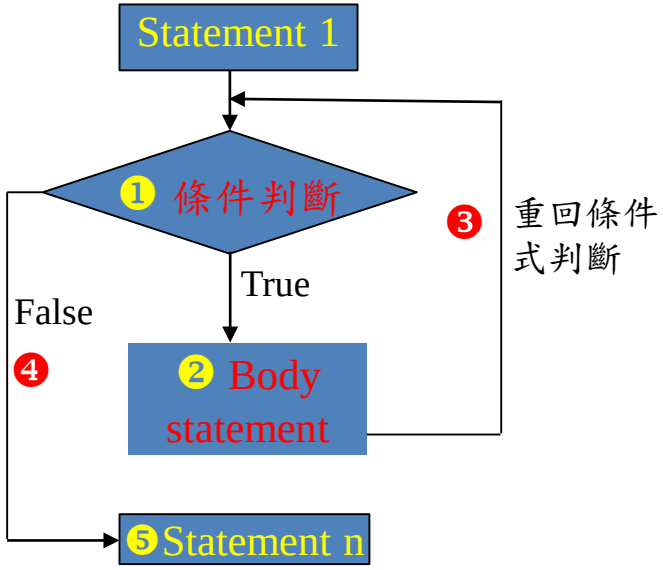
臺北科技大學資訊工程系

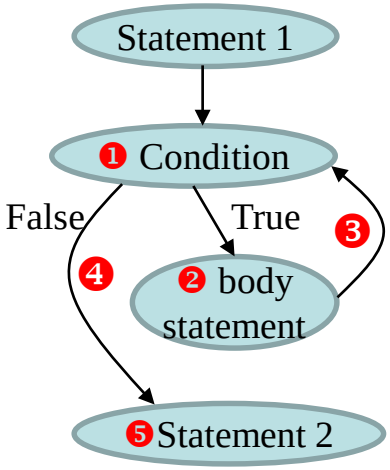
程式結構

- 計算機程式有三種結構
  - 循序結構
  - 選擇結構
    - 根據邏輯條件判斷執行正確指令。
    - 以邏輯條件判斷決定之後執行哪一段程式。
  - 重複結構
    - 程式不斷重複執行相同程式碼，利用迴圈完成重複計算。
    - 以邏輯條件判斷決定是否重複執行同一段程式碼。

迴圈原理

- 利用條件判斷真假，決定程式碼是否重複。
- 當條件判斷結果為真，程式會執行迴圈內容，之後重複回到條件判斷。
- 當條件算式結果為假，程式會跳出迴圈，繼續執行迴圈後的程式碼：

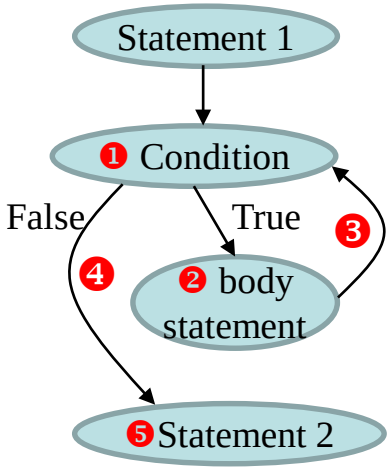




預先條件算式迴圈：while

- while 是預先條件式迴圈，檢查條件式結果是否真 (不為 0)。
  - 條件式：可以為任何運算式、變數或數值。如果結果為非 0 的數值，則表示為真; 否則為假
  - 若為真(非0)，則執行一次迴圈內動作，然後跳回條件式再檢查。
  - 如此一直執行到條件算式不成立為止 (0) 才離開迴圈。
  - 迴圈內動作：可以為任何合法程式指令。

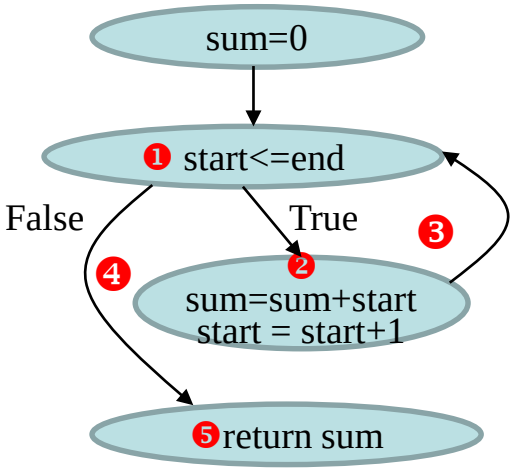
```
while (條件算式) {
   迴圈內動作
}
```



預先條件算式迴圈：while

- 適用於當迴圈內容不一定需執行時，因 while 條件式會先判斷，再決定是否執行迴圈動作。

```
#include <stdio.h>
int getSum(int start, int end) {
    int sum = 0;
    while (start<=end) {
        sum = sum + start;
        start = start + 1;
    }
    return sum;
}
int main() {
    printf("%d\n", getSum(1,10));
    return 0;
}
```



執行結果

- 迴圈條件算式，根據輸入的 start 跟 end 值判斷。如果 start 值小於或等於 end，則會執行迴圈動作，開始累加計算；如果 start 值大於 end，表示條件為假，不會執行迴圈動作。
  - 3~8 ?
  - 7~4 ?

求兩數的最大公因數

- 兩數的最大公因數，可用輾轉相除法。
  - 兩數先相除一次後，用除數當新的被除數，餘數當新的除數。
  - 如此不斷相除，直到除數大於被除數為止，除數即為最大公因數。
  - 要避免除以 0 的情況，須判斷除數是否為 0 ，才能進行相除。

```
#include <stdio.h>
int gcd (int n1, int n2) {
  while ((n1!=0)&&(n2!=0)) {
    if (n1>n2) n1=n1%n2;
    else n2=n2%n1;
  }
  if (n1>n2) return (n1);
  else return (n2);
}
int main() {
   printf("%d\n", gcd(12,10));
   printf("%d\n", gcd(54,48));
   return 0;
}
```

程式執行說明

- 迴圈 while 的條件算式，在 n1!=0 和n2!=0為真的情況下，程式會執行while 底下大括號內的迴圈內容。
- 迴圈內容是輾轉相除法的程式碼。每次除完就以除數當成被除數，餘數當成除數一直除下去，直到除盡為止。
- 除盡後， n1 或 n2 值會為 0，使 while 的條件算式結果為假，會跳出迴圈執行迴圈後的程式碼。

Exercise

- 印出1, 2, 4, 9, 16, 25 ... 225 數字和

- 求三個數的最小公倍數

Exercise

- $f(k) = 1^1 + 2^2 + 3^3 + 4^4 + 5^5 + \ldots + K^K$ ，1<=K<=15
  - 輸入K
    - 若不是整數，輸出must be integer
    - 若不在範圍，輸出out of range
    - 若在合法範圍，輸出結果數值X、X的所有數字總和

Exercise – 判斷整數、浮點數、字串

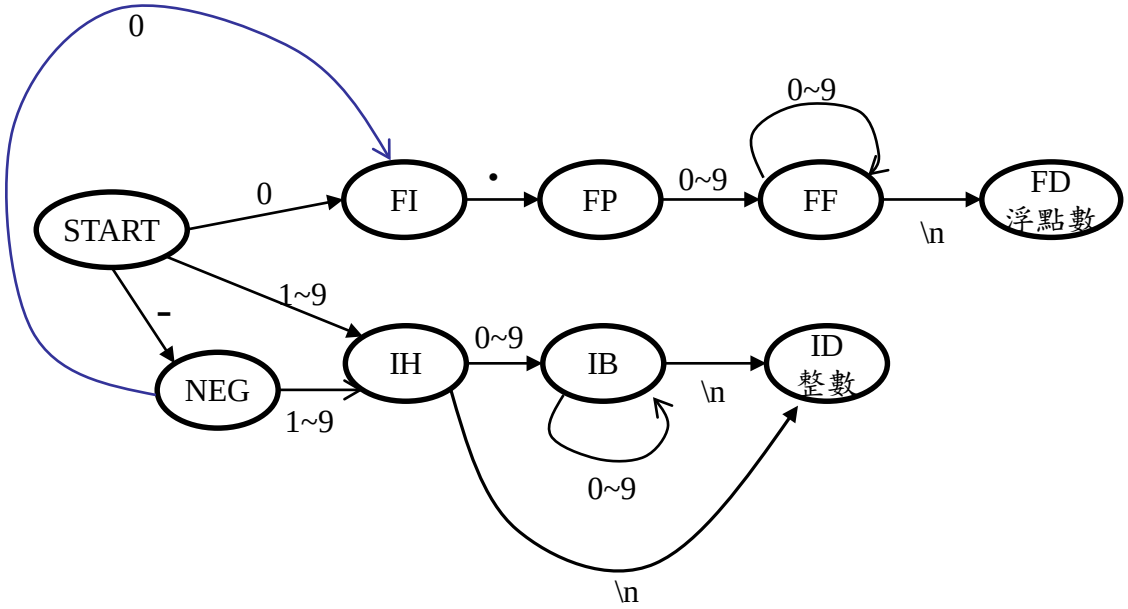
```
01  #include <stdio.h>
02  void f() {
03      int ret;
04      float n;
05      while (1) {
06          printf("Integer? ");
07          ret=scanf("%f", &n);
08          if (ret == 0) {
09              printf("char!\n");
10              while(getchar()!='\n');
11                  continue;
12          }
13          if (ret==1 && n != (int)n) {
14              printf("float!\n");
15              continue;
16          }
17          printf("OK\n");
18      }
19  }
20  int main() {
21      f();
22      return 0;
23  }
```

錯誤的程式?

```
Integer? 0.99999999
OK
Integer?
1.0000000001
OK
Integer? 1.00
OK
Integer? 4a
OK
Integer? char!
Integer?
```

Exercise – 判斷整數、浮點數、字串

//狀態轉換圖




```
>1.002356
float:1.002356
>10.214501
float:10.214501
>-100.2214
negative float:-100.221397
>10012
integer: 10012
>-1257
negative integer: -1257
>01245
string
```

Exercise – 判斷整數、浮點數、字串

```
01  #include <stdio.h>
02  #define START 0
03  #define IH 1 //integer head
04  #define IB 2 //integer body
05  #define ID 3 //integer defined
06  #define FI 4 //floating integer part
07  #define FP 5 //floating point
08  #define FF 6 //floating decimal
09  #define FD 7 //floating defined
10  #define STRING 8
11  #define NEG 9 //negative
13  //計算小數點的值
14  double getfValue(char key, int fPoint) {
15      int num = (key-'0');
16      double r=num;
17      for (int i=0; i<fPoint; i++)
18          r = r/10;
19      return r;
20  }
```

```
21  int getState(int state, char key) { //狀態轉換圖
22      if (state==START && key=='0') return FI;
23      else if (state==START && key>='1' && key<='9') return IH;
24      else if (state==START && key>='-') return NEG;
25      else if (state==NEG && key=='0') return FI;
26      else if (state==NEG && key>='1' && key<='9') return IH;
27      else if (state==FI && key=='.') return FP;
28      else if (state==IH && key=='.') return FP;
29      else if (state==IB && key=='.') return FP;
30      else if (state==FP && key>='0' && key<='9') return FF;
31      else if (state==FF && key>='0' && key<='9') return FF;
32      else if (state==FF && key=='\n') return FD;
33      else if (state==IH && key>='0' && key<='9') return IB;
34      else if (state==IB && key>='0' && key<='9') return IB;
35      else if (state==IB && key=='\n') return ID;
36      else return STRING;
37  }
```

Exercise – 判斷整數、浮點數、字串

```
38  int checkInput() {
39      char key;
40      int state=START, neg=1;
41      int iPart=0, fPoint=0;
42      float fPart=0;
43      printf(">");
44      while (1) {
45          key = getchar();
46          state = getState(state, key);
47          if (state==STRING) { //非整數、浮點數
48              printf("string");
49              break;
50          }
51          else if (state==FD) { //浮點數
52              if (neg==-1) printf("negative ");
53              printf("float:%f\n", (iPart+fPart)*neg);
54              break;
55          }
56          else if (state==ID) {//整數
57              if (neg==-1) printf("negative ");
58              printf("integer: %d\n", iPart*neg);
59              break;
60          }
```

```
61          else if (state==NEG) neg=-1;
62          else if (state==IH) iPart=key-'0';
63          //計算整數的值
64          else if (state==IB) iPart=iPart*10 + key-'0';
65          else if (state==FF) {
66              //小數點多一位
67              fPoint++;
68              //計算小數點的值
69              fPart = fPart + getfValue(key, fPoint);
70          }
71      }
72      return state;
73  }
74
75  int main() {
76      while (1) {
77          if (checkInput()==STRING) break;
78      }
79      return 0;
80  }
```

Exercise - while

- S(m, n) = {f(y), y∈[m..n]}，
- f(y) 產生 k個數字 {x_i, i∈[1..k]}
  - (1) if y is 1
    - x_i=1, end
  - (2) if y is odd, y= x_i = y/5+2, (or y=3*y+1) back to (1)
  - (3) if y is even, y= x_i = y/3+1, (or y=y/2) back to (1)
- 輸入m, n，1<=m<n<=10000，算出
  - 若m, n不是整數，輸出must be integer
  - 若m, n不在合法範圍，輸出out of range
  - 若在合法範圍，輸出
    - S(n) 中，每一個f(y_i)，都會產生k_i個數字，求最大k_i
    - S(n) 中，每一個f(y_i)，都會產生k_i個數字，將k_i個數字的每一位數的數字加總sum_i，求最大加總。例如 f(50)={17, 5, 3, 2, 1}，sum=1+7+5+3+2+1=19

Exercise - while

```
01  int sumX(int x) {
02      int value =0;
03      while (x>10) {
04          value = value + x%10;
05          x = x/10;
06      }
07      value = value + x;
08      return value;
09  }
```

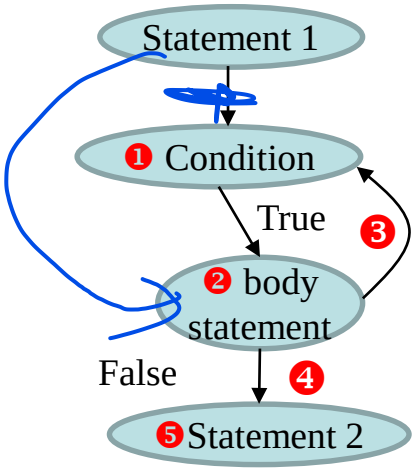
```
01  #include <stdio.h>
02  int f(int y) {
03      int count=0;
04      while (1) {
05          if (y<1) {
06              break;
07          }
08          else if (y==1) {
09              count++;
10              break;
11          }
12          else if (y%2==1) {
13              y = y/5+2;
14              count++;
15          }
16          else {
17              y = y/3+1;
18              count++;
19          }
20      }
21      return count;
22  }
```

```
23  int s(int m, int n) {
24      int value=0, i=m+1;
25      int maxF=f(m);
26      if (n==m) return maxF;
27      while (i<=n) {
28          value = f(i);
29          if (maxF<value) {
30              maxF = value;
31          }
32          i++;
33      }
34      return maxF;
35  }
36  int main() {
37      int value=0;
38      printf("%d\n", s(2,500));
39      return 0;
40  }
```

後設條件算式迴圈：do-while

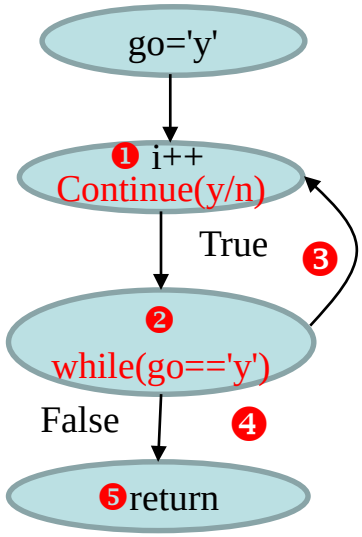
- do-while 是後設判斷式迴圈，先執行一次動作後，再判斷迴圈控制條件，若條件成立再回到前面執行 {} 內動作，如此重複直到條件算式結果為假。

```
do {
   迴圈內動作
} while (條件算式);
```



累計迴圈執行的次數

```
#include <stdio.h>
int main() {
    int i=0;
    char go='y';
    do {
        i++;
        printf("; Loop %d\n", i);
        printf("Continue(y/n):");
        go = getche();
    }  while (go=='y');
    printf("\nTotal Loop %d\n", i);
    return 0;
}
```




```
; Loop 1
Continue(y/n):y; Loop 2
Continue(y/n):y; Loop 3
Continue(y/n):y; Loop 4
Continue(y/n):y; Loop 5
Continue(y/n):y; Loop 6
Continue(y/n):y; Loop 7
Continue(y/n):n
Total Loop 7
```

Exercise累加值

- 輸入非0，印出目前累加值，輸入0，印出累加值並停止。

```
目前累加值:0
輸入下一個值: 25
目前累加值:25
輸入下一個值: 18
目前累加值:43
輸入下一個值: 33
目前累加值:76
輸入下一個值: 64
目前累加值:140
輸入下一個值: 0
累加值:140
```

執行結果

- 5~10 行是 do-while 迴圈程式碼，10 行條件算式會判斷，在第 8 行輸入的 number 值是否為 0。若不為 0，會跳回第 5 行，再將迴圈內容執行一次；若為 0，則跳出迴圈，執行第 11 行的程式碼。

```
01  #include <stdio.h>
02  int main() {
03      int sum=0;     //數值總和
04      int number=0;  //累加數值
05      do {
06          printf("目前累加值:%d\n", sum);
07          printf("輸入下一個值: ");
08          scanf("%d", &number);
09          sum = sum + number;
10      } while (number!=0);
11      printf("累加值:%d\n", sum);
12      return 0;
13  }
```

```
目前累加值:0
輸入下一個值: 25
目前累加值:25
輸入下一個值: 18
目前累加值:43
輸入下一個值: 33
目前累加值:76
輸入下一個值: 64
目前累加值:140
輸入下一個值: 0
累加值:140
```

無限迴圈/無窮迴圈

- 當迴圈的條件算式設定有誤，使迴圈的條件算式結果恆真，迴圈動作就會不斷執行，直到程式被強迫中止 (按下組合鍵 Ctrl + C) 或硬體停止回應 (電腦當機) 程式才會被終止。

```
while (1) {
    printf("無限迴圈\n");
}
```

```
do {
    printf("無限迴圈\n");
} while (1);
```

跳離迴圈：break

- break 跳出一層迴圈
  - 當 while、do-while 迴圈在執行中，想跳出迴圈有兩種方法
    - 一是使判斷條件不成立，另一是使用 break。
  - break 讓程式立即跳出迴圈，繼續執行迴圈之後的程式。
    - 將 break 放置於想要跳離的迴圈內即可：
  - 一個 break 只會跳出一層迴圈。
    - 如果是三層巢狀迴圈，就需三個 break 才能完全跳脫迴圈。

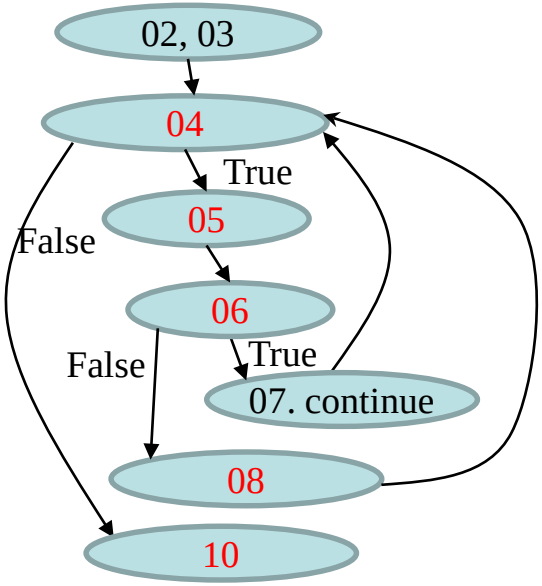
```
while (1) {
    printf("無限迴圈\n");
    break;
}
```

← 這訊息只出現一次

跳出一輪迴圈

- continue 跳出一輪迴圈，但未必跳出一層迴圈
  - while, for 都可以使用

```
01  #當number 沒超過20 不印@，超過印@
02  void test03() {
03      int i=0, number =0;
04      for (i=1; i<30; i++) {
05          number = number +i;
06          if (number<20)
07              continue
08          print("@ %d, %d", i, number);
09      }
10  }
```



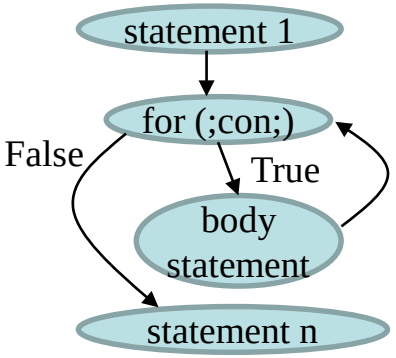
利用continue在任何時候略過迴圈
(略過本次迴圈剩餘的運算)

範圍設定式迴圈：for

- for 迴圈
  - 利用索引變數值的累加或累減控制迴圈執行次數，等該變數值達到設定標準，就會跳出迴圈。
  - for 迴圈最好不要用浮點數來控制

```
for (初始值設定;終止條件判斷;每次迴圈後的運算) {
    迴圈內的指令動作
}
```

```
初始值設定
while (終止條件判斷) {
    迴圈內的指令動作
    每次迴圈後的運算
}
```



範圍設定式迴圈：for

- 初始值設定：設定條件式中用到變數之初始值。
  - 例如 i=1
- 終止條件判斷：判斷是否執行迴圈中的程式。
  - 每次迴圈開始時檢查一次。
  - 例如設 i<3，表示只有在 i 小於 3 才會執行迴圈動作；若 i 大於等於 3，則不會執行迴圈。
- 每次迴圈後的運算：用於調整條件算式中的變數值
  - 例如條件算式為 i<3，用此運算來改變 i 的值，使終止條件成立
  - 例如 i++，使 i 的值最終大於等於 10，進而使迴圈結束。
  - 控制算式會在每次迴圈執行完畢時執行 1 次。

```
for (i=1; i<3;i++) {
    //迴圈內指令動作
}
```

for 迴圈執行步驟

1. 從初始算式開始，之後執行條件算式判斷是否執行迴圈。若結果為真，則執行迴圈動作。
2. 執行過一次迴圈動作後，執行每次迴圈後的運算，再次判斷條件算式中的結果。
3. 重複步驟 1、2，直到條件算式的判斷結果為假，跳出迴圈。

i = 0 → i<3 為True → 執行迴圈指令 → i++，i值變為 1
i = 1 → i<3 為True → 執行迴圈指令 → i++，i值變為 2
i = 2 → i<3 為True → 執行迴圈指令 → i++，i值變為 3
i = 3 → i<3 為False → 跳出迴圈

```
for (i=1; i<3;i++) {
    //迴圈內指令動作
}
```

Exercise for 迴圈 累加

- 計算 1~100 間所有奇數的和。
- 如何設定迴圈的執行條件，
  - 初始算式設定用於累加的變數 i 其初始值為 1、
  - 條件算式設定 i<100、
  - 控制算式設為每次將 i 的值加 2 (因為只計算奇數)，
  - 在迴圈中做累加的動作。

簡單的 for 迴圈累加

- 第 4、5 行，為 for 迴圈，程式碼只有一行，可省略大括號 {}。
  - 第47 行，初始算式 i=1 將 i 值設為 1；條件式 i 小於 100 下為真，持續執行迴圈動作；i+=2 表示每次迴圈執行後，將 i 加 2 。
  - 當 i 值變成 100 時，結束迴圈。
  - 1~100，2500

```
1  #include <stdio.h>
2  int main() {
3      int sum=0;    //計算總和
4      for (int i=0; i<100; i+=2)
5          sum = sum+i;   //迴圈每次跳 2
6      printf("\nTotal %d\n", sum);
7      return 0;
8  }
```

for 迴圈中可有兩組算式

- for 迴圈允許使用兩組以上的初始算式、條件算式及控制算式，每組間以逗號隔開：

```
for (初始值設定1, 2;終止條件判斷1, 2;每次迴圈後的運算1, 2) {
    迴圈內的指令動作
}
```

```
for (i=0, j=0; i<3, j<3; i++, j++)
```

  - 變數 i、j 初始值均為 0，每執行一次迴圈，i 會依控制算式 i++，將 i 加 1。
  - 同時，j 也會依控制算式 j++，將 j 加 1。
  - 一直到兩個條件算式 i<3 且 j<3 同時不成立，才會跳出迴圈。
  - 寫一個程式計算一個多項式 (1+2) + (2+4) + (3+6) ...+ (n+2*n) ，可使用此種 for 迴圈完成。

for 迴圈可用浮點數控制

- for 迴圈的迴圈變數可以每次加0.1。(不建議使用)
  - 迴圈有兩個迴圈變數，sum 與 i。每次迴圈執行後，將當時 i 值加到 sum ，且 i 加 0.1。

```
#include <stdio.h>
int main() {
    double sum=0;   //計算總和
    for (double i=0.1; i<1.05; sum+=i, i+=0.1)
        printf(" %.1f + ", i);
    printf(" = %.1f\n", sum);
    return 0;
}
```

```
0.1 +  0.2 +  0.3 +  0.4 +  0.5 +  0.6 +  0.7 +  0.8 +  0.9 +  1.0 +  = 5.5
```

跳出一輪迴圈

- break 是跳脫整個迴圈
- continue是跳脫 "這一輪" 迴圈。
  - 第 3~7 行迴圈，輸出從 1 到 10 除了 5 以外的數值。
  - 當迴圈進行到第 5 圈，會使第 4 行 if 條件判斷式的判斷為真。而執行第 5 行的 contiune，跳過 i=5 這一輪，繼續執行下一輪。

```
1  #include <stdio.h>
2  int main() {
3      for (int i=0; i<=10; i++){
4          if (i==5)      // i=5 時成立
5              continue;  //跳脫第5次迴圈
6          printf("%d ", i);
7      }
8      return 0;
9  }
```

```
0 1 2 3 4 6 7 8 9 10
```

for 的無限迴圈

- for 迴圈中有三個設定項目，若缺少條件算式，或程式邏輯錯誤使條件算式永遠為真時，會導致一直執行不停的無限迴圈：

```
#include <stdio.h>
int main() {
    for (int i=1;;)
        printf("cannot stop\n");
    return 0;
}
```

使用迴圈的注意事項

- 條件算式的設定要合理
  - 不當的條件算式設定會產生無限迴圈，或者根本未執行到迴圈的內容。所以在設定迴圈的條件式時，請仔細檢查條件算式的推演結果，以下是一些條件算式不合理的例子：

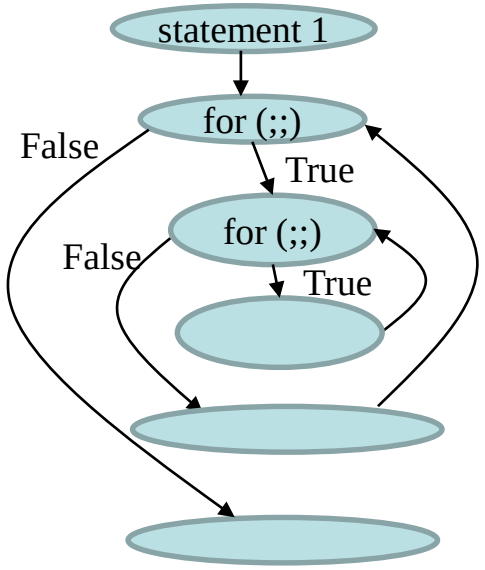
```
while(1)                條件算式為非 0 數值，導致無窮迴圈
for (i=0; i<100;)       缺項導致 i<100 永遠為真，形成無窮迴圈
for (i=0; i<0; i++)     條件算式不可能為真，永遠不會執行迴圈
```

- 依照程式需求，選擇使用特性適合的迴圈
  - 先判斷再決定是否執行時，使用 while 迴圈
  - 先執行一次再決定是否繼續時，使用 do-while 迴圈
  - 準確控制迴圈內容的執行次數，使用 for 迴圈

巢狀迴圈

- 巢狀迴圈是在迴圈的條件算式為真時，所執行的動作內還有其他迴圈。

```
for (初始運算式1;條件算式1;控制算式1) {

   第一個迴圈動作指令1

   for (初始運算式2;條件算式2;控制算式2) {

      第二個迴圈動作指令

   }

   第一個迴圈動作指令2
}
```



巢狀迴圈

- 2 層巢狀迴圈為例，外迴圈每執行一圈，就把所有內迴圈執行一次。
  - 每次執行時都會先執行外迴圈的第一圈，
  - 然後把內迴圈所有迴圈執行完後，
  - 再執行外迴圈的第二圈。

```
1  #include <stdio.h>
2  int main() {
3      for (int i=2; i<10; i++) {
4          for (int j=2; j<10; j++)
5              printf(" %d*%d=%2d ", i, j, i*j);
6          printf("\n");
7      }
8      return 0;
9  }
```

程式執行說明

- 第 3~7 行，為巢狀迴圈。第 4、5行，為內迴圈。
  - 第 3 行，外迴圈變數 i=2 ，進入內迴圈連續執行 8 次。分別為 i=2 j=2，i=2 j=3，... ，i=2 j=9。
  - 執行完第 6 行，輸出換行字元後，回到第 3 行，以 i=3 再次進入內迴圈，執行 i=3 j=2，i=3 j=3 ... i=3 j=9 8次後，再以 i=4 帶入。
  - 以此類推，直到 i 的值不小於 10 為止。

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

Homework I

- 迴圈與 if 條件判斷式混合應用：求 1~N 之間的質數
  - 質數是除了 1 與本身之外，無法被其他數整除的數。
  - 可利用巢狀迴圈計算 1~100 間任一個數字，是否會被比它小的數字 (從 2 開始) 整除，會整除的就不是質數。找不到可整除的，就是質數。
  - 輸出，質數每三個輸出一個。

輸出等腰三角形圖案

- 巢狀迴圈，可將外迴圈的迴圈變數，放入內迴圈的條件變數。隨著外迴圈執行次數增加，內迴圈的執行次數也跟著改變。
- 利用內外迴圈變數間的關係，輸出星號等腰三角形。

```
1  #include <stdio.h>
2  int main() {
3      int n; // 三角形底的星號數
4      do {
5          scanf("%d", &n);
6      } while (n%2==0);
7      for (int i=0; i<=n/2; i++){
8          for (int j=n/2; j>i; j--) //控制輸出空白
9              printf(" ");
10         for (int k=0; k<=2*i; k++) //控制輸出*
11             printf("*");
12         printf("\n");
13     }
14     return 0;
15 }
```

```
4
6
9
    *
   ***
  *****
 *******
*********
```

程式執行說明

- 第 7~13 行，外迴圈控制換行。依輸入三角形的底，設定行數。
- 第 8、9 行，第 1 個內迴圈，用來控制每行輸出空白字元數。
- 第 10、11 行，第 2 個內迴圈，用來控制每行輸出的星號數。輸出的星號，會印在第 9 行迴圈所輸出的空白字元後。

Exercise等腰三角形圖案

- 使用一層迴圈與function，輸出等腰三角形圖案

```
void printStar(int n) {
    for (int i=1; i<=n; i++)
        printf("*");
}
```

```
void printStar(int n, char mark) {
    for (int i=1; i<=n; i++)
        printf("%c", mark);
}
```

```
void printTriangle(int n) { // n 是高度



}
```

n=5

```
####*
###***
##*****
#*******
*********
```

巢狀迴圈應用：輸出字母圖形

- 輸出以字母組成的直角三角形
  - 第 3~7 行是外迴圈，迴圈變數 i 用來控制內迴圈的輸出數。
  - 第4、5 行是內迴圈，用來輸出英文字母。
  - 第 5 行的 printf () 會將數字轉換成字元從螢幕輸出。
  - 外迴圈每執行一次，將 i 值代入內迴圈。內迴圈就會執行 i 次，並輸出 i 順序的字母。

```
1  #include <stdio.h>
2  int main() {
3      for (int i=1; i<7; i++){
4          for (int j=0; j<i; j++)
5              printf("%c", j+65);
6          printf("\n");
7      }
8      return 0;
9  }
```

```
A
AB
ABC
ABCD
ABCDE
ABCDEF
```

巢狀迴圈應用：輸出字母圖形

- 輸出以字母組成的直角三角形

```
1  void printLine(int n);
2
3  void print(int n) {
4      for (int i=1; i<n; i++) {
5          printLine();
6          printf("\n");
7      }
8  }
9
```

```
A
AB
ABC
ABCD
ABCDE
ABCDEF
```

Exercise

- 以下程式輸出?

```
#include <stdio.h>
int main() {
    int a,b;
    for(b=1;b<=3;b=b+2 ) {
        for( a=6;a>=2;a=a-2 )
            printf("%d,%d,%d\n",a,a+b,a*b);
        printf("\n");
    }
}
```

Homework II

- 將Code寫成 二個function，每一個function使用一層迴圈
  - 輸入圖案編號與層數，輸出各種圖形

```
1
12
123
1234
12345

54321
4321
321
21
1

1
22
333
4444
55555
```

```
1
121
12321
1234321
123454321

___1___
__212__
_32123_
4321234

4321234
_32123_
__212__
___1___
```

Homework

- 輸入兩個整數N, M。
  - N代表要輸出的圖形種類
    - N = 1代表英文字母與星號的複合三角形
    - N = 2代表數字與星號的複合圖形
    - N 不為 1或2，輸出ERROR並直接結束程式
  - M代表圖形的高度
    - 若N = 1時，M的範圍為2 <= M <= 29
      - 第二層開始的圖形為英文字母與星號相間輸出
      - 第二層的英文字母為A
      - 第三層的英文字母為B
      - 第四層的英文字母為C
      - 第五層的英文字母為A，以此類推
    - 若N = 2時，M的範圍為1 <= M <= 9
    - 若M不在範圍，輸出ERROR並結束程式

Homework

- 當N = 1, M = 5時 ，輸出圖形為：

```
####*####
###*A*###
##*B*B*##
#*C*C*C*#
*A*A*A*A*
```

- 當N = 2, M = 5 時，輸出圖形為：

```
1**********1
12********21
123******321
1234****4321
12345**54321
```

- 輸入2, 10，輸出ERROR
- 輸入3，輸出ERROR

Exercise

- APCS 2019
  - A, B 兩隊比賽籃球，每場籃球賽有四節，輸入A, B雙方每一節的比分，求最終比賽結果，其規則為若A兩場全贏：Win；兩場全敗：Lose；一勝一敗：Draw。
  - 輸入皆為正整。
  - 每場比賽雙方比分不同。
  - 輸入 A:B 二場、每場四節的比分
  - 使用function改善

11 2
22 12
13 17
16 18
22 33
41 15
32 19
26 28
輸出
Win

```
#include <stdio.h>
int main(){
    int i=0, a = 0, b = 0, ans = 0, ai=0, bi=0;
    for (i = 0 ; i < 4 ; i++ ){
        scanf("%d %d",&ai, &bi);
        a += ai;          b += bi;
    }
    ans += ( a > b ? 1 : -1 );
    a = b = 0;
    for (i = 0 ; i < 4 ; i++ ){
        scanf("%d %d",&ai, &bi);
        a += ai;       b += bi;
    }
    ans += ( a > b ? 1 : -1 );
    if (!ans )        printf("Draw");
    else if ( ans > 0 )        printf("Win");
    else        printf("Lose");
    printf("\n");
    return 0; }
```

迴圈混合應用：計算階乘

- 從鍵盤輸入正整數，輸出該數字的階乘，再詢問是否要繼續。

```
1  #include <stdio.h>
2  int factorial(int n) {
3      for (int i=n-1; i>0; i--)
4          n = n*i;
5      return n;
6  }
7  int main() {
8      int number=0; //計算階層
9      char go='y';  //判斷迴圈是否繼續
10     do {
11         printf("\n計算階乘，請輸入一個值:");
12         scanf("%d", &number);
13         printf("Answer=%d\n", factorial(number));
14         printf("是否繼續(y/n):");
15         go = getche();
16     } while (go=='y');
17     return 0;
18 }
```

```
計算階乘，請輸入一個值:12
Answer=479001600
是否繼續(y/n):y
計算階乘，請輸入一個值:23
Answer=862453760
是否繼續(y/n):n
```

程式執行說明

- 2~6，factorial，計算階乘。範圍由迴圈判斷條件設定為輸入的數值到 1。
- 10~16，後設條件判斷式迴圈 do-while，用來判斷是否繼續輸入數字。
- 15，輸入判斷字元。再由16迴圈的條件算式判斷，決定是否繼續執行迴圈。

Exercise

- 計算BMI值的function
  - BMI值計算公式:　BMI = 體重(公斤) / 身高2(公尺2)
  - 例如：一個52公斤的人，身高是155公分，則BMI為：
  - 52(公斤)/1.552 (公尺^2)= 21.6
  - 正常範圍為 BMI=18.5～24
  - 輸入身高、體重，輸出BMI值。
  - 身高正常範圍 0.5~2.50 公尺，體重正常20~300 公斤，若輸入不在正常範圍，輸出"Input Error (0.5~2.50)"/ "Input Error (20~300)" ，請重新輸入。
  - 若BMI值太高，輸出"BMI too hight"，太低輸出"BMI too low"。
  - 可以接受不斷輸入計算，直到輸入-1停止。

Exercise

- 猜數字，隨機產生一個介於1~10的答案，使用者猜中則停止輸入，根據使用者輸入提示以下訊息:
  - 1.猜太大
  - 2.猜太小
  - 3.猜中了

```
#include <stdio.h>
void myFunction() {
    int input=0;
    int ans = random.randint(1,10)
    while (true) {
        printf("Guess 1~10: ");
        scanf("%d", &input);
        if (inputData == ans) {
            print("Right")
            break;
        }
}
int main() {
    myFunction();
}
```

Homework III

- 撲克牌
  - A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K
  - A~10 點數為 1~10，J, K, Q 為 0.5。
- 電腦與玩家各隨機(測資輸入)發撲克牌，加總點數接近 10.5 則贏。
  - 超過 10.5 爆掉分數為 0 。
- 程式
  - 隨機(測資)發一張撲克牌給玩家，玩家選擇要牌或不要牌。
  - 隨機(測資)發一張撲克牌給電腦，電腦判斷是否停發牌。
    - 牌面比玩家小，要牌
    - 小於8點(含)，要牌
    - 其餘情況不要牌
  - 輸出電腦與玩家的點數，以及電腦贏或玩家贏或平手。

Homework III

A 先發一張給給玩家
J 再發一張給電腦
Y 玩家選擇要牌
9 發一張給玩家
8 電腦牌面0.5點，未超過8點，再發一張給電腦
N 玩家選擇不要牌
3電腦牌面小於玩家，要牌

10.0 vs. 0.0
player win

4 先發一張給給玩家
6 再發一張給電腦
Y 玩家選擇要牌
1 發一張給玩家
2 電腦牌面 6 點，未超過 8 點，要牌
Y 玩家選擇要牌
5 發一張給玩家
J 電腦牌面比玩家少，要牌
2 電腦牌面比玩家少，要牌

10.0 vs. 10.5
computer win

9 先發一張給給玩家
8 再發一張給電腦
N 玩家選擇不要牌
9 電腦牌面比玩家少，要牌

9.0 vs. 0.0
player win

Homework III

```
#include <stdio.h>
double input(char x, char y) {
    if (x=='1') return 10;
    else if ((x>='2')&&(x<='9')) return (x-'0');
    else if (x=='A') return (1);
    else return 0.5;
}
int isDeal(int sum) {
    if (sum<=8) return 1;
    return 0;
}
int main() {
    char x, y;
    double in=0, scoreA=0, scoreB=0;
    scanf("%c%c", &x, &y);
    in = input(x, y);
    scoreA = scoreA + in;
    printf("%.1f, %.1f\n", input(x,y), scoreA);
```

```
    if (isDeal(scoreA)) {
        scanf("%c%c", &x, &y);
        in = input(x, y);
        scoreA = scoreA + in;
        printf("%.1f, %.1f\n", input(x,y), scoreA);
    }
    if (isDeal(scoreA)) {
        scanf("%c%c", &x, &y);
        in = input(x, y);
        scoreA = scoreA + in;
        printf("%.1f, %.1f\n", input(x,y), scoreA);
    }
    return 0;
}
```

Homework IV

- 輸入10進位整數，轉成二進位
- 輸入二進位，轉成10進位
- 輸入 b1 進位 x，轉成b2進位 y