4-1 流程指令

重點整理

1. C/C++語言提供 if 和 switch 兩種流程控制結構，控制程式的執行流程。
2. 條件式：流程指令依據條件式的成立與否，決定程式執行流程，條件式通常由關係運算子與邏輯運算子構成。
3. 關係運算子

運算子	說明
==	等於
!=	不等於
>	大於
<	小於
>=	大於等於
<=	小於等於

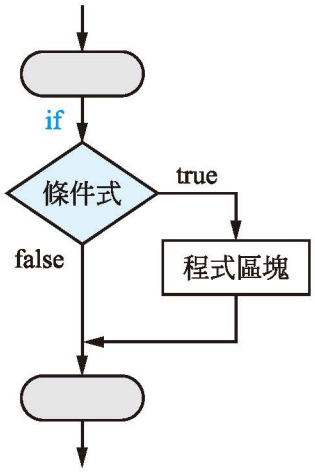
4. 邏輯運算子

運算子	說明
&&	且(AND)
||	或(OR)
!	非(NOT)

5. if 條件敘述

(1) 「if...」，當條件式成立(true)時，執行程式區塊，語法如下：

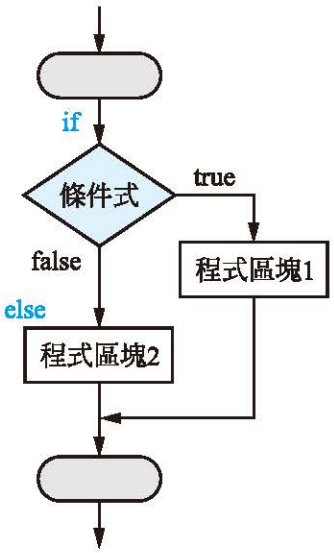
```
if (條件式) {
 程式區塊
}
```




```
1.  if (x>y){
2.      printf("條件式 x>y 成立\n");
3.  }
```

(2)「if … else …」，當條件式成立時，執行程式區塊 1，否則，執行程式區塊 2，語法如下：

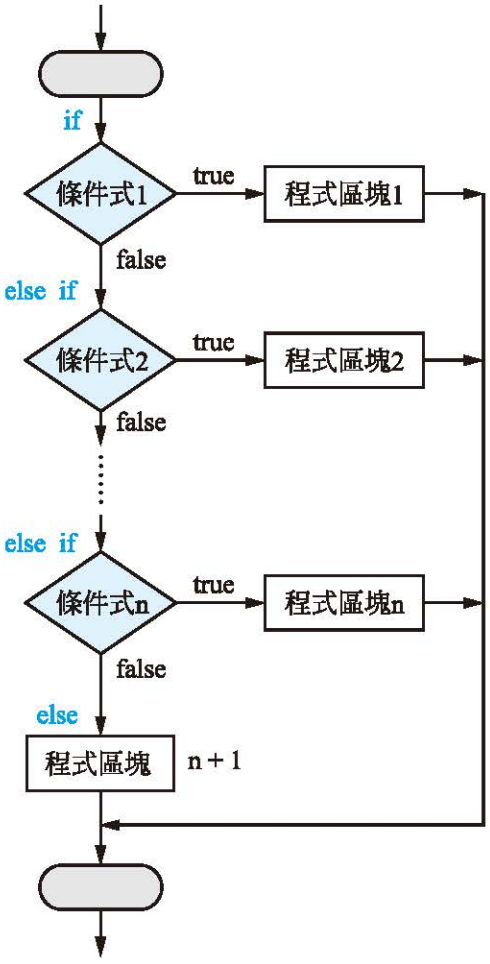
```
if (條件式) {
 程式區塊 1
}
else {
 程式區塊 2
}
```




```
1.  if (x>y){
2.      printf("條件式 x>y 成立\n");
3.  }
4.  else{
5.      printf("條件式 x>y 不成立\n");
6.  }
```

(3)「if … else if …」，當條件式 1 成立時，執行程式區塊 1，否則判斷條件式 2，當條件式 2 成立時，執行程式區塊 2，以此類推，最後若所有條件式皆不成立，則執行 else 內的程式區塊 n+1，語法如下：

```
if (條件式 1) {
 程式區塊 1
}
else if (條件式 2) {
 程式區塊 2
}
 :
else if (條件式 n) {
 程式區塊 n
}
else {
 程式區塊 n+1
}
```



```
1. if (x>y){
2.     printf("條件式 x>y 成立\n");
3. }
4. else if (x<y){
5.     printf("條件式 x<y 成立\n");
6. }
7. else
8.     printf("條件式 x>y 和 x<y 皆不成立\n");
9. }
```

6. 若程式區塊內只有一行程式敘述時，可以省略左右大括號{ }。
7. **條件運算子「? :」**，可以代替「if … else …」敘述，當條件式 1 成立時，執行程式區塊 1，否則，執行程式區塊 2，語法如下：

   條件式 1 ? 程式區塊 1 : 程式區塊 2

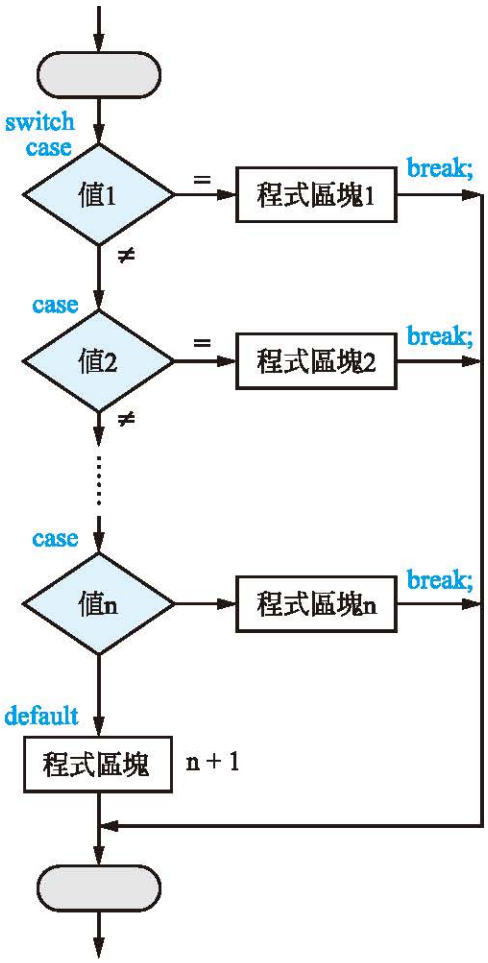
   以下 2 個程式片段的執行結果相同。

```
1. x = (x>99) ? 0 : x+1;      /* 當 x 大於 99，將 x 指定為 0，否則將 x 加 1 */
```

```
1. if (x>99){                 /* 當 x 大於 99，將 x 指定為 0 */
2.     x=0;
3. }
4. else{                      /* 否則，將 x 加 1 */
5.     x=x+1;
6. }
```

8. 「switch … case …」條件敘述，可以代替「if … else if …」敘述，運算式會由上而下依序和每一個 case 的值比對，當運算式的結果和某一個 case 的值相等時，就執行該 case 內的程式區塊，直到遇到 `break` 指令才結束；若運算式和所有 case 的值皆不相等，則執行 `default` 內的程式區塊，語法如下：

```
switch (運算式) {
 case  值 1:
       程式區塊 1
       break;
 case  值 2:
       程式區塊 2
       break;
       :
 case  值 n:
       程式區塊 n
       break;
 default:
       程式區塊 n+1
}
```



下列 2 個程式片段的執行結果相同。

程式片段 1：

```
1.  switch (x){
2.      case 0:
3.          printf("x 等於 0\n");
4.          break;
5.      case 1:
6.          printf("x 等於 1\n");
7.          break;
8.      case 2:
9.          printf("x 等於 2\n");
10.         break;
11.     default:
12.         printf("x 不等於 0、1、2\n");
13. }
```

程式片段 2：

```
1.  if (x==0){
2.      printf("x 等於 0\n");
3.  }
4.  else if (x==1){
```

```
5.      printf("x 等於 1\n");
6.  }
7.  else if (x==2){
8.      printf("x 等於 2\n");
9.  }
10. else{
11.     printf("x 不等於 0、1、2\n");
12. }
```

9. 使用「switch … case …」時，在每個 case 的最後，需使用 break 指令來中斷 switch 流程，否則程式將會繼續往下執行，如下所列之程式片段，若將第 7 行註解，則當 x 爲 1 時，會連續輸出「x 等於 1」「x 等於 2」，直到遇到第 10 行的 break 指令，才離開 switch 結構。

```
1.  switch (x){
2.      case 0:
3.          printf("x 等於 0\n");
4.          break;
5.      case 1:
6.          printf("x 等於 1\n");
7.          //break;
8.      case 2:
9.          printf("x 等於 2\n");
10.         break;
11.     default:
12.         printf("x 不等於 0、1、2\n");
13. }
```

10. 在某些編譯器，可以允許比對一個範圍的值，如下，

```
1.  switch(x){
2.      case 1…3:   /* 1≤x≤3 */
3.          :
4.          break;
5.      case 4…6:   /* 4≤x≤6 */
6.          :
7.          break;
8.      default:
9.          :
10. }
```

01 範例

某購物網站依會員年度消費金額，將會員區分爲不同等級，如下表，請幫該網站設計一程式，輸入年度消費金額，能判斷該會員等級。

消費金額(元)	會員等級
0 ~ 3,000	普通
3,001 ~ 10,000	黃金
10,001 ~ 50,000	白金
50,001 ~	鑽石

解

程式碼：

```
1.  #include <stdio.h>
2.
3.  int main(){
4.      unsigned int m;
5.
6.      printf("請輸入年度消費金額：");
7.      scanf("%d", &m);
8.
9.      if (m<=3000){
10.         printf("普通會員\n");
11.     }
12.     else if (m>3000 && m<=10000){
13.         printf("黃金會員\n");
14.     }
15.     else if (m>10000 && m<=50000){
16.         printf("白金會員\n");
17.     }
18.     else{
19.         printf("鑽石會員\n");
20.     }
21. }
```

程式解說：

行數	說明
6, 7	輸入年度消費金額。
9 ~ 20	利用「if … else if …」條件敘述，控制程式流程。

02 範例

陽曆規定每 4 年就有一年的 2 月必須多一天，因此在 2 月 29 日出生的同學，每 4 年只能過一次生日，而有 2 月 29 日的這一年，稱之爲「閏年」，判斷西元年是否爲閏年的規則如下，請依據此規則設計一程式，判斷輸入的西元年是否爲閏年。

1. 西元年可被 100 整除
   1-1. 可被 400 整除，是閏年
   1-2. 不可被 400 整除，不是閏年
2. 西元年不可被 100 整除
   2-1. 可被 4 整除，是閏年
   2-2. 不可被 4 整除，不是閏年

解

程式碼：

```
1.  #include <stdio.h>
2.
3.  int main(){
4.      int year;
5.
6.      printf("請輸入一西元年：");
7.      scanf("%d", &year);
8.
9.      if (year % 100 == 0){               /* 1.可以被 100 整除 */
10.         if (year % 400 == 0){           /* 1-1.可以被 400 整除 */
11.             printf("是閏年\n");
12.         }
13.         else{                           /* 1-2.無法被 400 整除 */
14.             printf("不是閏年\n");
15.         }
16.     }
17.     else{                               /* 2.無法被 100 整除 */
18.         if (year % 4 == 0){             /* 2-1.可以被 4 整除 */
19.             printf("是閏年\n");
20.         }
21.         else{                           /* 2-2.無法被 4 整除 */
22.             printf("不是閏年\n");
23.         }
24.     }
25. }
```

程式解說：

行數	說明
6, 7	使用者輸入西元年。
9, 10, 18	判斷餘數是否爲 0，餘數爲 0 表示能被整數，則條件式成立。
9 ~ 24	依閏年規則控制程式流程，以西元 2020 年爲例，「if...else...」結構的執行流程如下： (1) 第 9 行，條件不成立(2020 除 100，餘數不爲 0)，跳到第 17 行，進入 else 的程式區塊。 (2) 第 18 行，條件成立(2020 除 4，餘數爲 0)，執行第 19 行，輸出 "是閏年"。

03 範例

眞好吃速食店的菜單如下，爲節省人力成本，同時加快廚房出餐速度，請你開發一自助點餐系統，使用者只要輸入 1~5 的數字代碼，即可選擇不同的套餐，同時廚房人員馬上可以料理餐點。

代碼	套餐
1 號餐	陽光牛肉麵
2 號餐	熱情雞肉捲
3 號餐	酷炫打拋豬
4 號餐	活力海鮮粥
5 號餐	清心素食鍋

解

程式碼：

```
1.  #include <stdio.h>
2.
3.  int main(){
4.      int s=0;
5.
6.      printf("請輸入套餐代碼(1~5)：");
7.      scanf("%d", &s);
8.
9.      switch(s){
10.         case 1:
11.             printf("1 號餐，陽光牛肉麵\n");
12.             break;
13.         case 2:
14.             printf("2 號餐，熱情雞肉捲\n");
15.             break;
```

```
16.         case 3:
17.             printf("3 號餐，酷炫打拋豬\n");
18.             break;
19.         case 4:
20.             printf("4 號餐，活力海鮮粥\n");
21.             break;
22.         case 5:
23.             printf("5 號餐，清心素食鍋\n");
24.             break;
25.         default:
26.             printf("請重新輸入套餐代碼(1~5)：");
27.     }
28. }
```

程式解說：

行數	說明
6, 7	輸入 1 ~ 5 之間的數字。
9 ~ 27	使用「switch … case …」敘述，當使用者輸入的值等於其中一個 case 的值時，執行該 case 的程式區塊。
25, 26	使用者輸入的值不在 1~5 範圍內，進入 default 程式區塊。

•立即練習

(D) 1. 已知 x=3，y=4，z=5，下列哪一個條件式的結果爲 false？
(A)x<y (B)z>y (C)x>z || y>x (D)z>x && y<x。

(A) 2. 下列哪一個指令可以中斷程式的執行？
(A)break (B)continue (C)goto (D)return。

(D) 3. 以下哪一個指令不是 C 語言的決策指令？
(A)if (B)switch (C)if…else… (D)return。

(B) 4. 已知 A 爲 true，B 爲 false，下列哪一個運算式的結果爲 true？
(A)A && B (B)A && B || A (C)A && B || B (D)B || A && B。

(C) 5. 下列程式片段執行後，x 爲何？ (A)2 (B)1 (C)10 (D)9。

```
int x=1;
switch(x){
    case 1:
        x++;
    case 3:
        x=x+3;
    case 5:
        x=x+5;
}
```

(A) 6.　下列程式片段執行後的輸出爲何？
(A)Yes　(B)No　(C)YesNo　(D)YesNo??。

```
char ans='Y';
switch(ans){
    case 'Y':
        printf("Yes");
        break;
    case 'N':
        printf("No");
        break;
    defalut:
        printf("??");
}
```

(C) 7.　下列程式片段執行後，z 的值爲何？　(A)2　(B)3　(C)4　(D)5。

```
int x, y, z;
x=1;y=3;z=5;
if (x>y){
    z++;
}
else{
    z--;
}
```

(A) 8.　下列程式片段執行後，a 的值爲何？　(A)1　(B)-1　(C)15　(D)8。

```
int a=8, b=7;
if (a == b)
    a=a+b;
else if (a<b)
    a=b-a;
else
    a=a-b;
```

(C) 9.　下列程式片段執行後，x 的值爲何？　(A)1　(B)0　(C)2　(D)3。

```
int x=0;
if (++x == 1)
    printf("%d", x+1);
else
    printf("%d", x-1);
```

(B) 10. 下列程式片段執行後，x 的值爲何？ (A)1 (B)0 (C)2 (D)3。

```
int x=0;
if (x++ == 1)
    printf("%d", x+1);
else
    printf("%d", x-1);
```

(A) 11. 下列程式片段執行後，x 與 y 的值分別爲何？

(A)x=2, y=2 (B)x=2, y=3 (C)x=1, y=2 (D)x=1, y=3。

```
int x=1, y=2;
if ((++x>1) || (++y>2)){
    printf("x=%i, y=%i", x, y);
}
```

(B) 12. 下列程式片段執行後，x 與 y 的值分別爲何？

(A)x=2, y=2 (B)x=2, y=3 (C)x=1, y=2 (D)x=1, y=3。

```
int x=1, y=2;
if ((++x<1) || (++y>2)){
    printf("x=%i, y=%i", x, y);
}
```

(A) 13. 下列程式片段執行後，x 與 y 的值分別爲何？

(A)x=2, y=2 (B)x=2, y=3 (C)x=1, y=2 (D)x=1, y=3。

```
int x=1, y=2;
if ((++x<1) && (++y>2)){
    :
}
```

(B) 14. 下列程式片段執行後，x 與 y 的值分別爲何？

(A)x=2, y=2 (B)x=2, y=3 (C)x=1, y=2 (D)x=1, y=3。

```
int x=1, y=2;
if ((++x>1) && (++y>2)){
    :
}
```

(D) 15. 下列程式碼片段執行後，變數 y 的值爲何？

(A)1 (B)3 (C)4 (D)5。

```
x = 1;
switch(x){
    case 1:
        y = 3;
    case 2:
        y = 4;
    default:
        y = 5;
    }
```

答案 打★表示有詳解

1.(D)　2.(A)　3.(D)　★4.(B)　★5.(C)　6.(A)　7.(C)　8.(A)　★9.(C)　★10.(B)　★11.(A)
★12.(B)　★13.(A)　★14.(B)　★15.(D)

解析

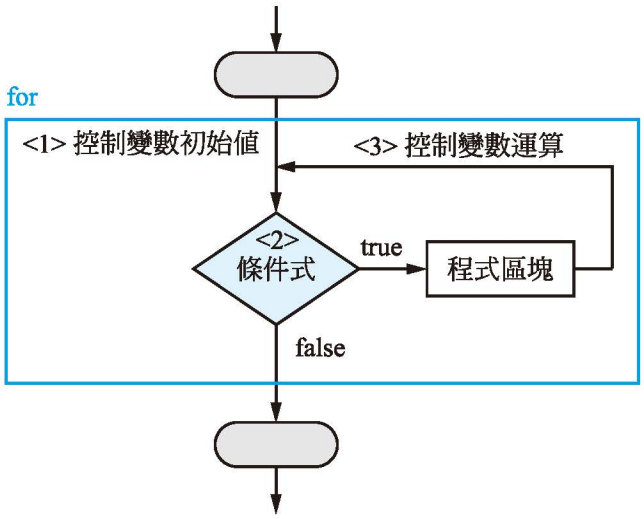
4. 「&&」的優先權高於「||」。
5. 因爲 case 內沒有 break 指令，所以每一個 case 的程式區塊都會執行。
9. 「++x」爲前置運算，條件式(++x==1)會先將 x 加 1，再判斷 x 是否等於 1。
10. 「x++」爲後置運算，條件式(x++==1)會先判斷 x 是否等於 1，再將 x 加 1。
11. 邏輯運算子「||」，當左邊的運算元爲 true，則整個條件式就爲 true，將不再處理右邊的運算元，因此(++y>2)不會被執行。
12. 邏輯運算子「||」，當左邊的運算元爲 false 時，必須再判斷右邊的運算元，才能判斷整個條件式爲 true 或 false。
13. 邏輯運算子「&&」，當左邊的運算元爲 false，則整個條件式就爲 false，將不再處理右邊的運算元，因此(++y>2)不會被執行。
14. 邏輯運算子「&&」，當左邊的運算元爲 true 時，必須再判斷右邊的運算元，才能判斷整個條件式爲 true 或 false。
15. 因爲 case 內沒有 break 指令，故會依序執行「y=3;」、「y=4;」、「y=5;」程式碼，最後 y 的值爲 5。

4-2 迴圈

重點整理

1. 迴圈指令可以讓程式重覆執行，C/C++語言提供 for 和 while 兩種迴圈結構。
2. 「for」指令，可以控制程式重覆執行的次數，語法如下：

```
for (控制變數初始值；條件式；控制變數運算式) {
        <1>          <2>          <3>
    程式區塊
}
```



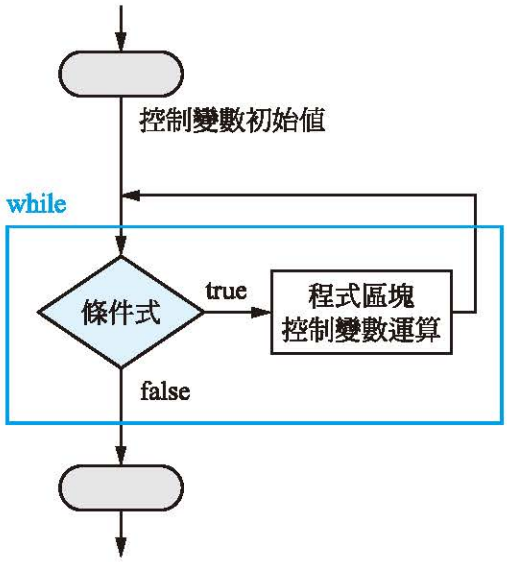
```
1.  int i;
2.  /* 控制變數 i，初始值為 0 */
3.  /* 條件式：i<10 */
4.  /* 迴圈每重覆一次，i 的值加 1，此迴圈結構共重覆 10 次 */
5.  for (i=0; i<10; i=i+1){
6.      printf("%d", i);     /* 執行 10 次，輸出 0123456789 */
7.  }
8.  printf("%d", i);         /* 離開迴圈後，i 的值為 10 */
```

註 最後離開迴圈時，i 的值為 10。

3. 「while」指令，適合在無法事先確定迴圈重覆次數時使用，與 for 指令不同，**控制變數的初始值設定與運算不包括在 while 結構內**：

(1) 「while」迴圈敘述，先進行條件式判斷，條件式成立才執行程式區塊，語法如下：

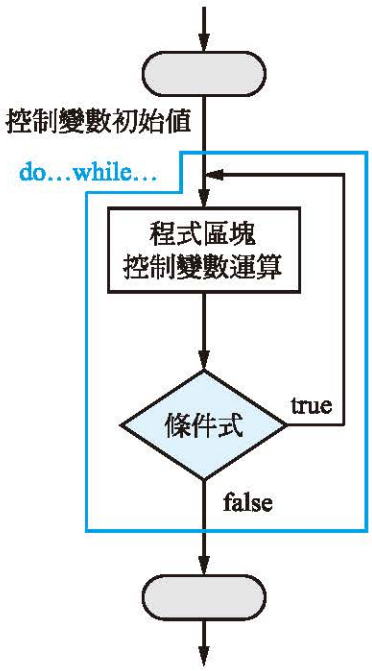
```
while (條件式) {
 程式區塊
}
```




```
1.  int i;
2.  i=0;                 /* 自行設定控制變數 i，初始值為 0 */
3.  while(i<10){         /* 條件式：i<10 */
4.      printf("%d", i); /* 執行 10 次，輸出 0123456789 */
5.      i=i+1;           /* 自行在程式區塊內將 i 加 1 */
6.  }
7.  printf("%d", i);     /* 離開迴圈後，i 的值為 10 */
```

(2) 「do…while…」迴圈敘述，先執行程式區塊，再進行條件式判斷，**程式區塊至少會執行 1 次**，語法如下：

```
do {
 程式區塊
} while (條件式);
```




```
1.  int i;
2.  i=0;                  /* 自行設定控制變數 i，初始值為 0 */
3.  do{
4.      printf("%d", i); /* 執行 10 次，輸出 0123456789 */
5.      i=i+1;            /* 自行在程式區塊內將 i 加 1 */
6.  } while(i<10);        /* 條件式：i<10 */
7.  printf("%d", i);      /* 離開迴圈後，i 的值為 10 */
```

4. 若程式區塊內只有一行程式敘述時，可以省略左右大括號{ }。

5. 巢狀迴圈：在迴圈內部又包含另一個迴圈，巢狀迴圈可以有 2 層以上，依據內外層迴圈的控制變數是否相關，可區分成以下兩種情況：

(1) 控制變數不相關

```
1.  for (i=1;i<=3;i++){
2.      for (j=1;j<=3;j++){                  /* j 的變化與 i 無關 */
3.          printf("i=%d, j=%d\n", i, j);    /* 執行 9 次*/
4.      }
5.  }
```

控制變數 i 和 j 的變化如下，

i	j
1	1
	2
	3
2	1
	2
	3
3	1
	2
	3

(2) 控制變數相關

```
1.  for (i=1;i<=3;i++){
2.      for (j=1;j<=i;j++){                  /* j 的變化與 i 有關 */
3.          printf("i=%d, j=%d\n", i, j);    /* 執行 6 次*/
4.      }
5.  }
```

控制變數 i 和 j 的變化如下，

i	j
1	1
2	1
	2
3	1
	2
	3

6. **break** 指令：強制中止迴圈執行，通常會搭配 if 條件使用。

```
1.  int i;
2.  for (i=0; i<10; i=i+1){
3.      if (i>=5){
4.          break;          /* 當 i>=5 時，中止迴圈流程，跳至第 8 行接續執行 */
5.      }
6.      printf("%d", i);    /* 執行 5 次，輸出 01234 */
7.  }
8.  printf("%d", i);        /* 離開迴圈後，i 的值為 5 */
```

7. **continue** 指令：停止往下執行，重新開始下一個迴圈。

```
1.  int i;
2.  for (i=0; i<10; i=i+1){
3.      if (i>=5){
4.          continue;       /* 當 i>=5 時，不執行後面的程式碼，跳至第 2 行繼續下一圈 */
5.      }
6.      printf("%d", i);    /* 執行 5 次，輸出 01234 */
7.  }
8.  printf("%d", i);        /* 離開迴圈後，i 的值為 10 */
```

8. 無窮迴圈：條件式永遠成立，迴圈會一直重覆執行，下列是兩個無窮迴圈的例子，

例 1：

```
1.  while(1){  /* C 語言中，非零的數值表示 true */
2.      printf("hola\n");
3.  }
```

例 2：

```
1.  for (i=0; i<=0; i=i-1){  /* 每重覆一次迴圈，控制變數 i 的值減 1，使得條件式 i<=0 永遠為 true */
2.      printf("amigo\n");
3.  }
```

04 範例

請設計一程式，輸入一正整數 n，輸出 $1^2+3^2+5^2+\cdots+n^2$ 的計算結果。

解

程式碼：

```
1.  #include<stdio.h>
2.
3.  int main(void)
4.  {
5.      int n, sum=0;
6.
7.      printf("請輸入正整數 n：");
8.      scanf("%d", &n);
9.
10.     for (int i=1; i<=n; i=i+2){
11.         sum = sum + i*i;
12.     }
13.
14.     printf("%d", sum);
15. }
```

程式解說：

行數	說明
7,8	輸入一正整數 n。
10 ~ 12	i 從 1 開始，迴圈每重覆 1 次，i 會加 2，迴圈共重覆 n/2 次。 假設 n=10，則 i 和 sum 的變化如下， i / sum=0 / 1 / 1 / 0+1*1=1 3 / 10 / 1+3*3=10 5 / 35 / 10+5*5=35 7 / 84 / 35+7*7=84 9 / 165 / 84+9*9=165 11 / / 條件式(i<=n)不成立，離開迴圈

05 範例

請設計一程式，輸入一正整數 N，輸出 N!(1×2×3×…×N) 的結果。

解

程式碼：

```
1.  #include<stdio.h>
2.
3.  int main(void)
4.  {
5.      int N, sum=1;
6.
7.      printf("請輸入正整數 N：");
8.      scanf("%d", &N);
9.
10.     for (int i=N; i>1; i--){
11.         sum = sum * i;
12.     }
13.
14.     printf("%d", sum);
15. }
```

程式解說：

行數	說明
7,8	輸入一正整數 N。
10 ~ 12	i 從 N 開始，迴圈每重覆 1 次，i 會減 1。 假設 N=5，則 i 和 sum 的變化如下， i | sum=1 | 5 | 5 | 1*5=5 4 | 20 | 5*4=20 3 | 60 | 20*3=60 2 | 120 | 60*2=120 1 | | 條件式(i>1)不成立，離開迴圈

06 範例 素養題

就讀國小五年級的堂妹正在學習因數，請利用 while 迴圈結構設計一程式，堂妹只要輸入一個任意正整數 n，該程式會列出 n 的所有因數。

解

程式碼：

```
1.  #include<stdio.h>
2.
3.  int main(void)
4.  {
5.      int i, n;
6.      printf("請輸入正整數：");
7.      scanf("%d", &n);
8.
9.      i=1;
10.     while (i<=n){
11.         if (n%i == 0){
12.             printf("%d ", i);
13.         }
14.         i=i+1;
15.     }
16. }
```

程式解說：

行數	說明
6, 7	輸入一正整數 n。
9	自行指定控制變數 i 的初始值。
10 ~ 15	控制變數 i 從 1 至 n，while 迴圈共重覆 n 次。
11 ~ 13	檢查 n 除 i 的餘數是否爲 0，餘數爲 0 表示 i 爲 n 的因數。
14	在程式區塊內，自行改變控制變數 i 的值。

• 立即練習

(C) 1.　哪一個是不論條件成立與否，至少會執行一次的迴圈？
(A)while 迴圈　(B)巢狀式 while 迴圈　(C)do while 迴圈　(D)for 迴圈。