程式重複執行的方法

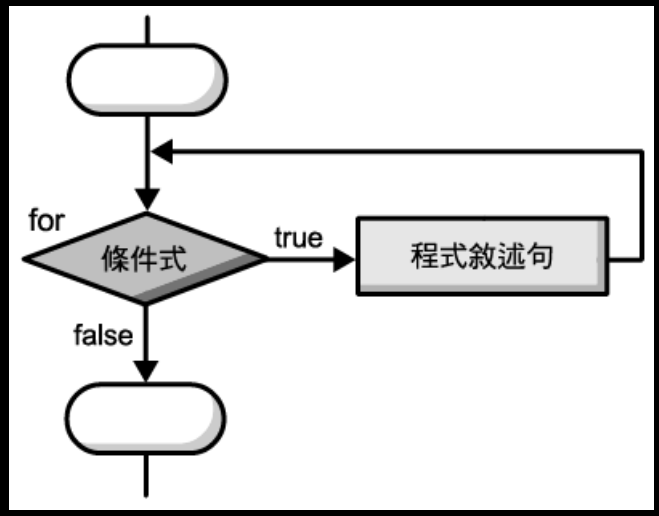
6-1 FOR敘述

- C語言用來處理重複事件的語法就稱爲迴圈敘述。
- 語法1：

```
for (❶起始值;    ❷判斷是否要重複執行的條件式;    ❸遞增或遞減運算)
    程式敘述句;
```

- 語法2：

```
for (❶起始值;    ❷判斷是否要重複執行的條件式;    ❸遞增或遞減運算)   {
    程式敘述句1;
    程式敘述句2;
    …
}
```



FOR 範例1

```
for(i=1; i<=5; i++){
    printf("執行迴圈。\n");
}
```

流程:

1. 在第1個式子的地方設定變數i的起始值。
2. 在第2個式子的地方，條件式如果爲true的話，緊接著會執行下方程式區塊的內容。
3. 在第2個式子的條件式變爲false之前，一直重複執行步驟二。

Sample1.c ▶ 使用 for 迴圈

```
#include <stdio.h>

int main(void)
{
    int i;

    for(i=1; i<=5; i++){
        printf("執行迴圈。\n");
    }

    printf("迴圈執行結束。\n");

    return 0;
}
```

變數 i 會從 1 開始一次一次累加，直到 i<=5 這個條件式變成 false 為止

這一行程式敘述句會重複執行

FOR 範例2

```
for(i=1; i<=5; i++){
    printf("第%d次的迴圈。\n", i);
}
```

- 透過上面這一段程式您應該可以清楚看到，迴圈在執行的過程中，用來計算執行次數的變數i的值也會輸出到螢幕上，如此一來，就可以看出程式總共重複執行多少次了。

Sample2.c ▶ 在螢幕上顯示迴圈的執行次數

```
#include <stdio.h>

int main(void)
{
    int i;

    for(i=1; i<=5; i++){
        printf("第 %d 次的迴圈。\n", i);
    }

    printf("迴圈結束。\n");

    return 0;
}
```

輸出文字的同時，遞增變數 i
遞增的值也會顯示在上面

FOR 範例3

- 根據從鍵盤輸入的值，印出相同數目的星號：

```
…
printf("請輸入要印出幾個*？\n");
scanf("%d", &num);
for(i=1; i<=num; i++){
   printf("*");
}
```

- 結果：

```
請輸入要印出幾個*？
10
**********
```

- 這種根據輸入的數字重複進行運算的方式，也會經常被用到。

Sample3.c ▶ 根據從鍵盤輸入的值，印出相同數目的星號

```
#include <stdio.h>

int main(void)
{
    int num;
    int i;

    printf("請問要輸出幾個*? \n");
    scanf("%d", &num);

    for(i=1; i<=num; i++){
        printf("*");
    }
    printf("\n");

    return 0;
}
```

從鍵盤輸入數字

根據輸入的數值，重複輸出星號

FOR 範例4

- 讓使用者輸入一個值 (假設是10)，然後程式會自動算出由1加總到10的結果，並輸出到螢幕上：

```
…
printf("請問要求從1加到那個數字爲止的和呢？\n");
scanf("%d", &num);
for(i=1; i<=num; i++){
    sum += i;
}
printf("從1加到%d爲止的和爲%d。\n", num, sum);
```

- 結果：

請問要求從1加到那個數字爲止的和呢？
10
從1加到**10**爲止的和爲**55**。

Sample4.c ▶ 根據輸入的數值求取總和

```
#include <stdio.h>

int main(void)
{
    int num, sum;
    int i;

    num = 0;
    sum = 0;

    printf("請問要求從 1 加到那個數字為止的和呢？ \n");
    scanf("%d", &num);

    for(i=1; i<=num; i++){
        sum += i;
    }

    printf("從 1 加到 %d 為止的和為 %d 。 \n", num, sum);

    return 0;
}
```

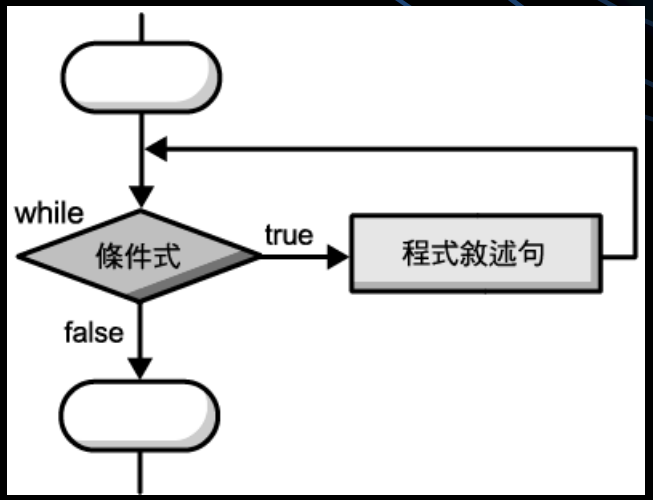
這一行用來讀取輸入的數值

變數 i 從 1 開始進行累加運算，直到等於輸入的數字為止

6-2 WHILE敘述

- 只要條件式爲真，就會重複執行敘述。
- 語法：

```
while (條件式) {
  程式敘述;
  …
}
```



- 範例：

```
while(i <= 5){
    printf("第%d次的迴圈。\n", i);
    i++;
}
```

- 必須小心設定while敘述的條件式，不要變成無窮迴圈了。

Sample5.c ▶ 使用 while 敘述

```
#include <stdio.h>

int main(void)
{
    int i = 1;

    while(i <= 5){
        printf("第 %d 次的迴圈。\n", i);
        i++;
    }
    printf("迴圈結束。\n");

    return 0;
}
```

當條件式等於 true 時…

會在程式區塊內依序重複執行

變數 i 會持續累加，直到 i<=5 為 false 時就跳離迴圈

Sample6.c ▶ 使用慣用的條件式

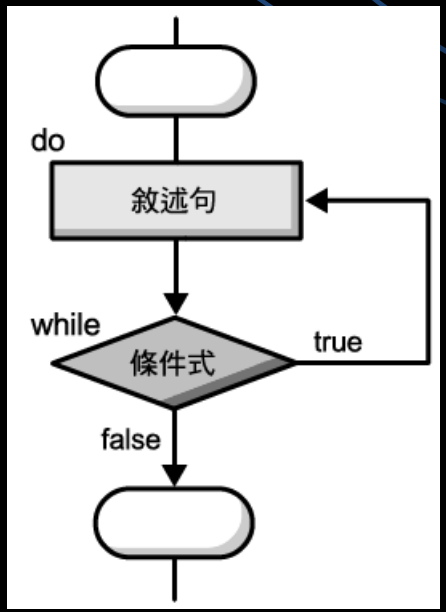
```
#include <stdio.h>

int main(void)
{
    int num = 1;

    while(num){
        printf("請輸入1個整數：(以0結束)\n");
        scanf("%d", &num);
        printf("輸入的是%d。\n", num);
    }
    printf("迴圈結束。\n");

    return 0;
}
```

當num為0時（false），迴圈結束

6-3 DO～WHILE敘述

- 語法：

```
do {
  程式敘述1;
  …
}while (條件式) ;
```



- 範例：

```
do{
    printf("第%d次的迴圈。\n", i);
    i++;
}while(i <= 5);
```

- While敘述是在執行程式區塊前判斷條件，
  do~while敘述則是在執行完程式區塊後判斷條件。

Sample7.c ▶ 使用 do ～ while 迴圈

```
#include <stdio.h>

int main(void)
{
    int i = 1;

    do{
        printf("第 %d 次的迴圈。\n", i);
        i++;
    }while(i <= 5);

    printf("迴圈結束。\n");

    return 0;
}
```

這個部分會重複執行

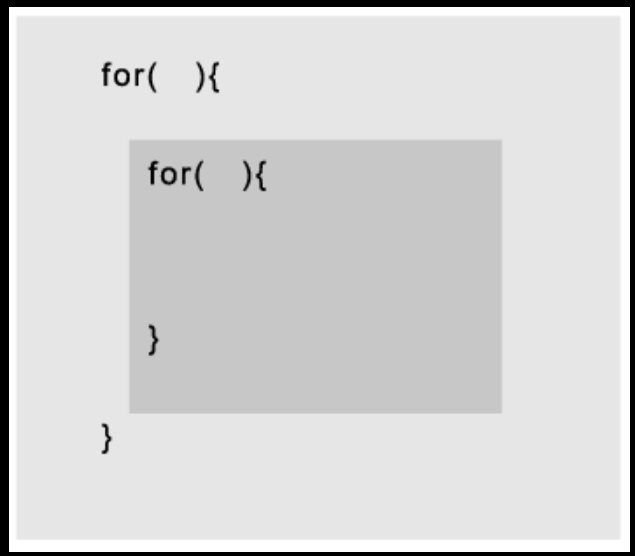
一旦 i<=5 這個條件式為偽（false），程式就會馬上跳開迴圈

6-4 巢狀迴圈

- 迴圈中還有其他迴圈就叫做巢狀迴圈，for敘述或while敘述都可以形成巢狀迴圈。

- 語法：

```
for (式子1-1; 式子2-1; 式子3-1) {
  …
  for (式子1-2; 式子2-2; 式子3-2) {
    …
  }
}
```



- 範例：

```
…
for (int i=0; i<3; i++) {
    for (int j=0; j<3; j++) {
        cont << “i是” << i << “:j是” << j << ‘\n’;
    }
}
```

- 執行畫面：
  - i是0:j是0
  - i是0:j是1
  - i是0:j是2
  - i是1:j是0
  - i是1:j是1
  - i是1:j是2
  - i是2:j是0
  - i是2:j是1
  - i是2:j是2

外圍的迴圈每執行一次，內側的迴圈就必須執行三次

外圍的迴圈總共會執行三次

Sample8.c ▶ 利用 2 個 for 敘述形成巢狀迴圈

```
#include <stdio.h>

int main(void)
{
    int i, j;

    for(i=0; i<5; i++){
        for(j=0; j<3; j++){
            printf("i 是 %d；j 是 %d。\n", i, j);
        }
    }

    return 0;
}
```

由 2 個 for 敘述所形成的巢狀迴圈

- 巢狀迴圈可搭配其他敘述來使用，例如if敘述。

- 範例：

```
    for(i=0; i<5; i++){
       for(j=0; j<5; j++){
          if(ch == 0){
             printf("*");
             ch = 1;
           }
          else{
             printf("-");
             ch = 0;
          }
       }
       printf("\n");
    }
```

執行畫面

```
*-*-*
-*-*-
*-*-*
-*-*-
*-*-*
```

Sample9.c ▶ 巢狀迴圈與 if 敘述

```
#include <stdio.h>

int main(void)
{
    int i, j, ch;

    ch = 0;

    for(i=0; i<5; i++){
        for(j=0; j<5; j++){
            if(ch == 0){
                printf("*");
                ch = 1;
            }
            else{
                printf("-");
                ch = 0;
            }
        }
        printf("\n");
    }

    return 0;
}
```

裡外 2 個 for 敘述形成的巢狀迴圈

印出 * 之後，讓 ch=1，因此接下來會印出 -

印出 - 之後，接下來讓 ch=0，因此會印出 *

裡面的迴圈結束之後便換行

6-5 改變程式執行的流程

- break指令
  - 主要用意是強制結束程式並跳出迴圈。
  - 語法：**break;**
  - 範例：

```
for ( i=1; i<=10; i++ ){

   printf ( "第%d次的處理。\n" , i ) ;

   if ( i == res )
      break;

}
```

- 在switch敘述中使用break指令。
- 範例：

```
switch(res){
    case 1:
    case 2:
        printf("還要再加強唷！\n");
        break;
    case 3:
    case 4:
        printf("就照這個樣子保持下去。\n");
        break;
    case 5:
        printf("相當優秀唷！\n");
        break;
    default:
        printf("要輸入1～5的成績。\n");
        break;
}
```

變數res的值等於1或2時會執行這一部分

變數res的值等於3或4時會執行這一部分

請注意break指令擺放的位置

Sample10.c ▶ 利用 break 指令跳離迴圈

```
#include <stdio.h>

int main(void)
{
    int res;
    int i;

    printf("請問要在第幾次終止迴圈呢？（1～10）\n");

    scanf("%d", &res);
    for(i=1; i<=10; i++){
        printf("第 %d 次的處理。\n", i);
        if(i == res)
            break;
    }

    return 0;
}
```

本來這裡的 for 迴圈會執行 10 次

加上 if 的判斷之後，當迴圈執行次數等於鍵盤輸入數目時，就會跳離迴圈

Sampel11.c ▶ 在 switch 敘述中使用 break 指令

```
#include <stdio.h>

int main(void)
{
    int res;

    printf("請輸入成績。(1～5)\n");
    scanf("%d", &res);

    switch(res){
        case 1:
        case 2:
            printf("還要再加強唷！\n");
            break;
        case 3:
        case 4:
            printf("就照這個樣子保持下去。\n");
            break;
        case 5:
            printf("相當優秀唷！\n");
            break;
        default:
            printf("要輸入1～5的成績。\n");
            break;
    }

    return 0;
}
```

變數 res 的值等於 1 或 2 時會執行這一部分

請注意 break 指令擺放的位置

請注意 break 指令擺放的位置

變數 res 的值等於 3 或 4 時會執行這一部分

- continue指令
  - 讓目前執行中的迴圈暫時停住不往下執行，而是回到迴圈繼續下一個執行
  - 語法：**continue;**
  - 範例：

```
for ( i=1; i<=10; i++ ){

   printf ( "%d次的處理。\n" , i ) ;

   if ( i == res )
      break;

}
```

Sample12.c ▶ continue 指令會讓程式再度回到迴圈繼續下一個執行

```
#include <stdio.h>

int main(void)
{
    int res;
    int i;

    printf("要跳過第幾次的處理？（1～10）\n");
    scanf("%d", &res);

    for(i=1; i<=10; i++){
        if(i == res)
            continue;
        printf("第%d次的處理。\n", i);
    }

    return 0;
}
```

當迴圈執行的次數與輸入的數字相同時（i==res），會執行 continue 指令

執行 continue 指令之後，程式不會往下執行，而會回到迴圈重新執行