2025/11/01 13:52 1/9 8. 迴圈Loop

國立屏東商業技術學院 資訊工程系 物件導向程式設計

8. 迴圈Loop

- · while loop
- · do while loop
- for loop
- · comma expression
- exiting from a loop
- break; continue; goto;
- · null statement;

在C/C++語言中,也有所謂的Loop□我們稱之為迴圈,用以重複地執行特定的程式碼。請參考figure 1,一個迴圈通常使用一組大括號將一些程式敘述包裹起來,並且反複地執行,直到特定的條件成立或不成立為止。判斷是否繼續執行的地方,可以在迴圈區塊的開頭處或是結束處,視所使用的loop敘述而定。本章將C語言所支援的loop敘述分別加以介紹。

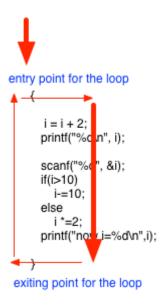


Fig. 1

8.1 while迴圈

while敘述的語法如下:

```
while (controlling_expression) statements
```

while敘述先判斷controlling_expression的值,若為true則一直執行後續的statement[]直

Last update: 2019/07/02 15:01

到controlling_expression的值為false時才結束。

<note important>

while迴圈可包含一行或一行以上的敘述

在語法上我們寫statements□指得是「敘述群」,意即一個或一個以上的敘述,我們可以使用一組大括號 將敘述包裹起來。但是當敘述超過一行時,該組大括號為必須。因此

```
while(i>0)
    i--;
```

與

```
while(i>0)
{
    i--;
}
```

以及

```
while(i>0)
{
    i--;
    cout << "i=" << i << endl;
}</pre>
```

都是正確的while敘述。以下我們不再重複statements的意義。

</note>

所以□while迴圈中的敘述可能會被執行0次到無限多次,視其controlling_expression的值而定。以下是一些例子:

```
// 計算1+2+3+...+10的結果
int i=1, sum=0;

//執行十次
while(i<=10)
{
    sum+=i;
    i++;
}
cout << "sum of 1 to 10 is " << sum << endl;
```

```
// 印出介於1到100間可以被7整除的數字
int i=1;
while(i<=100)
{
```

2025/11/01 13:52 3/9 8. 迴圈Loop

```
if(i%7==0)
    cout << i ;
i++;
}</pre>
```

<note important>C語言使用數值0表示false□其它非0的數值皆為true!</note>

```
// 反覆執行直到使用者輸入'q'為止
int quit=0;
char c;

while(!quit)
{
    // do something
    ...
    cout << "continue?(y/n)";
    scanf("%c", &c);
    if(c=='n')
        quit=1;
}</pre>
```

```
// 反覆執行直到使用者輸入'q'為止
int quit=0;
char c;

while(1)
{
    // do something
    ...
    if(expression)
        break;    // 直到特定條件成立時,使用break跳離
}
```

8.1.1 無窮迴圈(infinite loop)

有時,不小心設定while的controlling_expression□其結果會導致程式永遠無法結束,因為不論在各種情況下□controlling_expression永遠為true□我們將這種情況稱為無窮迴圈(infinite loop),例如:

```
// 反覆執行直到使用者輸入'q'為止 int quit=0; char c; while(i=100) // 這裡將i==100寫成了i=100 {
```

Last update: 2019/07/02 15:01

```
// do something
....
if(expression)
i=100; //直到特定條件成立時,將i設定為100
}
```

<note important> 若真的遇到此種情況,在Linux/Unix系統請使用Ctrl+C將程式跳離,然後使用ps aux指令查看程式的PID[]再以kill -9 PID指令將程式的執行中止掉[] </note>

8.2 do迴圈

do敘述的語法如下:

```
do statements while (controlling_expression);
```

do迴圈與while迴圈類似,不同之處在於其判斷是否繼續之處位於exiting point do敘述執行statements 然後才判斷接在while後面的controlling_expression的值,若為true則繼續回到do迴圈開頭處再次執行;若controlling_expression的值為false時,do迴圈才結束。相較於while迴圈 do迴圈內含的statements至少會被執行一次。以下是一些例子:

```
i=10;
do
{
    cout << i << endl;
    i--;
} while (i>0);
```

```
i=10;

do
{
    cout << i << endl;
} while (--i>0);
```

```
i=10;
do
{
```

2025/11/01 13:52 5/9 8. 迴圈Loop

```
cout << i << endl;
} while (i-- > 0 );
```

8.3 for迴圈

for敘述的語法如下:

```
for ( expression1; expression2; expression3 ) statements
```

其中expression1□expression2與expression3□分別是用以定義迴圈的初始條件、中止條件與更新(update)的處理,說明如下:

- expression1在迴圈初次執行前被執行,用以設定初始條件,例如i=0
- expression2在迴圈每次執行前加以檢查,視其值決定是否繼續執行,若其值為true則繼續,反之若 其值為false則結束。例如i<10
- expression3在迴圈每次執行完時加以執行,用以更新在expression1或expression2中的運算元的值,
 例如i++

我們可以用while的語法,來將for的語法改寫如下:

```
expression1;
while ( expression2)
{
   statements
   expression3;
}
```

以下是一些例子:

```
int i,sum=0;
for(i=1;i<=10;i++)
{
    sum+=i;
}
cout << "sum=" << sum << endl;</pre>
```

在expression1與expression3中也可以用','同時指定多個運算式,例如:

```
int i,sum;
for(i=1, sum=0;i<=10;i++)
{
    sum+=i;
}
cout << "sum=" << sum << endl;</pre>
```

expression1-3也可以被省略,例如:

```
int i=0;
for( ; i<10;i++)
   cout << "i=" << i << endl;</pre>
```

8.4 巢狀迴圈(nested loop)

一個迴圈內如含有另一個迴圈,則稱之為,<mark>巢狀迴圈(nested loop)</mark>。每一層的迴圈可以是for□while或do其中一個,以下我們僅以for迴圈為例,其它的組合您可以自行學習。

```
//印出1!+2!+3! + ... + 10!
int i,j,temp,sum=0;

for(i=1;i<=10;i++)
{
    temp=0;

    for(j=1; j<=i; j++)
    {
        temp*=j;
    }

    sum += temp;
}

cout << "sum=" << sum << endl;</pre>
```

請思考以下問題:

- 第6行的temp=0可不可以省略?
- 可以併入第4行,寫做□for(i=1, temp=0; i←10; i++□嗎?
- 同理,第2行的sum=0也可以併入嗎?

```
//印出1!+2!+3! + ... + 10!
int i,j,temp=1,sum=0;
```

2025/11/01 13:52 7/9 8. 迴圈Loop

```
for(i=1;i<=10;i++)
{
    temp*=i;
    sum += temp;
}

cout << "sum=" << sum << endl;</pre>
```

8.5 從迴圈中跳離

除了使用迴圈的controlling_expression來控制迴圈的執行外,我們還可以使用break continue與goto敘述來改變程式的動線,使其可以跳離迴圈所屬的程式區塊。

8.5.1 break敘述

break在迴圈中一旦被執行,則在此次迴圈執行過程中剩下還未執行的敘述就會被跳過不執行,並且結束 迴圈的執行。當迴圈的中止條件不在開頭或結尾時[]break敘述就便得很有用處,例如:

```
|| 「反覆要求使用者輸入一個整數 , 並且將其累加 , 直到使用者輸入0為止

int n, sum=0;

for(;;)
{
    printf("Please input a number (0 for quit):");
    scanf("%d", &n);
    if(n==0)
        break;
    sum+=n;
}
printf("sum=%d.\n", sum);
```

8.5.2 continue敘述

continue則和break相反,它並不會結束迴圈的執行,而是省略當次執行時未完成的程式碼,直接執行迴圈的下一回合。

```
//  反覆要求使用者輸入一個整數,並且將其累加,直到使用者輸入<math>0 為止,但輸入值若為負數則加以忽略 int n, sum=0; for(;;)
```

```
{
    printf("Please input a number (0 for quit):");
    scanf("%d", &n);
    if(n==0)
        break;
    if(n<0)
        continue;
    sum+=n;
    // continue敘述使程式碼跳到了這裡
}
printf("sum=%d.\n", sum);
```

8.5.3 goto敘述

C/C++語言還提供另一種無條件的跳躍敘述 - goto敘述。我們可以在程式碼中的特定位置標記一些label 其方法為在某行以標記名稱後接冒號的方式來定義,爾後需要改變程式碼執行動線時,則使用goto 標記名 稱;的方式即可完成。請參考以下的範例:

```
//反覆要求使用者輸入一個整數,並且將其累加,直到使用者輸入0為止
int n, sum=0;

for(;;)
{
    printf("Please input a number (0 for quit):");
    scanf("%d", &n);
    if(n==0)
        goto done;
    sum+=n;
}

done:
    printf("sum=%d.\n", sum);
```

goto敘述不一定要配合迴圈的使用,例如:

h goto.c

```
#include <stdio.h>
int main()
{
char cmd;
begin:
```

2025/11/01 13:52 9/9 8. 迴圈Loop

```
scanf("%c", &cmd);

if(cmd != 'q')
    goto begin;

printf("exit\n");
}
```

這個程式讓使用者不斷地輸入一個字元,直到其輸入字元為'q'時才結束程式。其中第7行定義了一個名為begin的標記,在第11行的if敘述若條件成立時則使用goto敘述跳躍到begin標記處。

<note tip> 許多程式設計師一直在爭論是否該在程式碼中使用goto□正反兩面的意見都值得參考。我覺得如果您覺得好用就用吧?只是每次使用時也順便想一想:同樣的功能如果不使用goto可以做到嗎?以免以後你不用goto就不會寫程式!我所認識的程式設計師裡面,兩種人都有,不過反對使用goto的人,通常完全容不下在程式中使用goto□如果你擔心以後工作上的主管或同事不喜歡你寫的含有goto的程式,那你最好用與不用都會寫,這樣就沒問題了! ◎ 不要使用goto </note>

From:

https://junwu.nptu.edu.tw/dokuwiki/ - Jun Wu的教學網頁

國立屏東大學資訊工程學系

CSIE, NPTU

Total: 230351

Permanent link:

https://junwu.nptu.edu.tw/dokuwiki/doku.php?id=cpp:loops

Last update: 2019/07/02 15:01

