ACPS C++ 教學講義 03

C++ 迴圈

有的時候，可能需要多次執行同一塊程式碼。

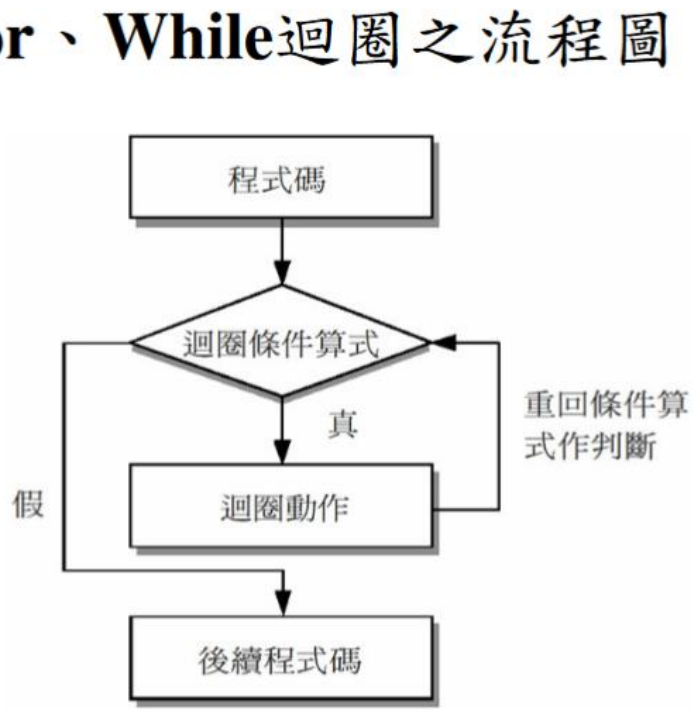
一般情況下，語句是順序執行的：

函數中的第一個語句先執行，接著是第二個語句，依此類推。

程式語言提供了允許更為複雜的執行路徑的多種控制結構。

迴圈語句允許我們多次執行一個語句或語句組。

下面是大多數程式語言中迴圈語句的一般形式：



教講者：華祖彬

迴圈類型

C++ 程式語言提供了以下幾種迴圈類型。

迴圈類型	描述
while 迴圈	當給定條件為真時，重複語句或語句組。 它會在執行迴圈主體之前測試條件。
for 迴圈	多次執行一個語句組，簡化管理迴圈變數的程式碼。
do...while 迴圈	除了它是在迴圈主體結尾測試條件外， 其他與 while 語句類似。
巢狀迴圈	您可以在 while、for 或 do..while 迴圈內 使用**一個或多個**迴圈。

迴圈控制語句

迴圈控制語句更改執行的正常順序。當執行離開一個範圍時，所有在該範圍中創建的自動對象都會被銷毀。

while 迴圈

只要給定的條件為真，while 迴圈語句會重複執行一個目標語句。

語法

```
while(condition)
{
   statement(s);
}
```

在這裡，statement(s) 可以是一個單獨的語句，也可以是幾個語句組成

的程式碼塊。condition 可以是任意的表達式，當為任意非零值時都為真。

當條件為眞時執行迴圈。

當條件為假時，程序流將繼續執行緊接著迴圈的下一條語句。

實例

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
using namespace std;

int main()
{
    // 區域變數宣告
    int a = 10;

    // while 迴圈執行
    while( a < 20 )
    {
        cout << "a 的值：" << a << endl;
        a++;
    }

     system("pause");
     return 0;
}
```


當上面的程式碼被編譯和執行時，它會產生下列結果：

```
a 的值： 10
a 的值： 11
a 的值： 12
a 的值： 13
a 的值： 14
a 的值： 15
```

```
a 的值： 16
a 的值： 17
a 的值： 18
a 的值： 19
```

for 迴圈

for 迴圈允許您編寫一個執行特定次數的迴圈的重複控制結構。

語法

```
for ( init; condition; increment )
{
   statement(s);
}
```

下面是 for 迴圈的控制流：

1. init 會首先被執行，且只會執行一次。
2. 這一步允許您宣告並初始化任何迴圈控制變數。您也可以不在這裡寫任何語句，只要有一個分號出現即可。
3. 接下來，會判斷 condition。如果為真，則執行迴圈主體。
4. 如果為假，則不執行迴圈主體，且控制流會跳轉到緊接著 for 迴圈的下一條語句。
5. 在執行完 for 迴圈主體後，控制流會跳回上面的 increment 語句。
6. 該語句允許您更新迴圈控制變數。該語句可以留空，只要在條件後有一個分號出現即可。
7. 條件再次被判斷。如果為真，則執行迴圈，這個過程會不斷重複（迴圈主體，然後增加步值，再然後重新判斷條件）。
8. 在條件變為假時，for 迴圈終止。

實例

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
using namespace std;

int main()
{
   // for 迴圈執行
    for( int a = 10; a < 20; a = a + 1 )
    {
        cout << "a 的值：" << a << endl;
    }

     system("pause");
     return 0;
}

```


當上面的程式碼被編譯和執行時，它會產生下列結果：

```
a 的值： 10
a 的值： 11
a 的值： 12
a 的值： 13
a 的值： 14
a 的值： 15
a 的值： 16
a 的值： 17
a 的值： 18
a 的值： 19
```

do...while 迴圈

不像 for 和 while 迴圈，它們是在迴圈頭部測試迴圈條件。

do...while 迴圈是在迴圈的尾部檢查它的條件。

do...while 迴圈與 while 迴圈類似，但是 do...while 迴圈會確保**至少執行一次迴圈**。

語法

```
do
{
   statement(s);

}while( condition );
```

請注意，條件表達式出現在迴圈的尾部，所以迴圈中的 statement(s) 會在條件被測試之前**至少執行一次**。

如果條件為真，控制流會跳**轉回上面的 do**，然後重新執行迴圈中的 statement(s)。

這個過程會不斷重複，直到給定條件變為假為止。

實例

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
using namespace std;

int main()
{
   // 區域變數宣告
    int a = 10;

    // do 迴圈執行
    do
    {
        cout << "a 的值：" << a << endl;
        a = a + 1;
    }while( a < 20 );

     system("pause");
     return 0;
}
```


當上面的程式碼被編譯和執行時，它會產生下列結果：

```
a 的值： 10
a 的值： 11
a 的值： 12
a 的值： 13
a 的值： 14
a 的值： 15
a 的值： 16
a 的值： 17
a 的值： 18
a 的值： 19
```

巢狀迴圈

一個迴圈內可以巢狀另一個迴圈。C++ 允許至少 **256** 個巢狀層次。

語法

```
for ( init; condition; increment )
{
   for ( init; condition; increment )
   {
      statement(s);
   }
   statement(s); // 可以放置更多的語句
}
```

巢狀 while 迴圈 語句的語法：

```
while(condition)
{
   while(condition)
   {
      statement(s);
   }
   statement(s); // 可以放置更多的語句
}
```

巢狀 do...while 迴圈 語句的語法：

```
do
{
   statement(s); // 可以放置更多的語句
   do
   {
      statement(s);
   }while( condition );

}while( condition );
```

關於巢狀迴圈有一點值得注意，您可以在任何類型的迴圈內巢狀其他任何類型的迴圈。比如，**一個 for 迴圈可以巢狀在一個 while 迴圈內**，反之亦然。

實例

下面的程序使用了一個巢狀的 for 迴圈來查找 2 到 100 中的質數：

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
using namespace std;

int main()
{
   int i, j;

   for(i=2; i<100; i++) {
      for(j=2; j <= (i/j); j++)
        if(!(i%j)) break; // 如果找到，則不是質數
        if(j > (i/j)) cout << i << " 是質數\n";
   }

    system("pause");
    return 0;
}
```


當上面的程式碼被編譯和執行時，它會產生下列結果：

```
2 是質數
3 是質數
5 是質數
7 是質數
11 是質數
13 是質數
17 是質數
19 是質數
23 是質數
29 是質數
31 是質數
```

```
37 是質數
41 是質數
43 是質數
47 是質數
53 是質數
59 是質數
61 是質數
67 是質數
71 是質數
73 是質數
79 是質數
83 是質數
89 是質數
97 是質數
```

C++ 提供了下列的控制語句，可中斷迴圈的執行。

控制語句	描述
break 語句	終止 loop 或 switch 語句，程序流將 繼續執行緊接著 loop 或 switch 的下一條語句。
continue 語句	引起迴圈跳過主體的剩餘部分，立即重新開始測試條件。
goto 語句	將控制轉移到被標記的語句。 但是不建議在程序中使用 goto 語句。

break 語句

C++ 中 break 語句有以下兩種用法：

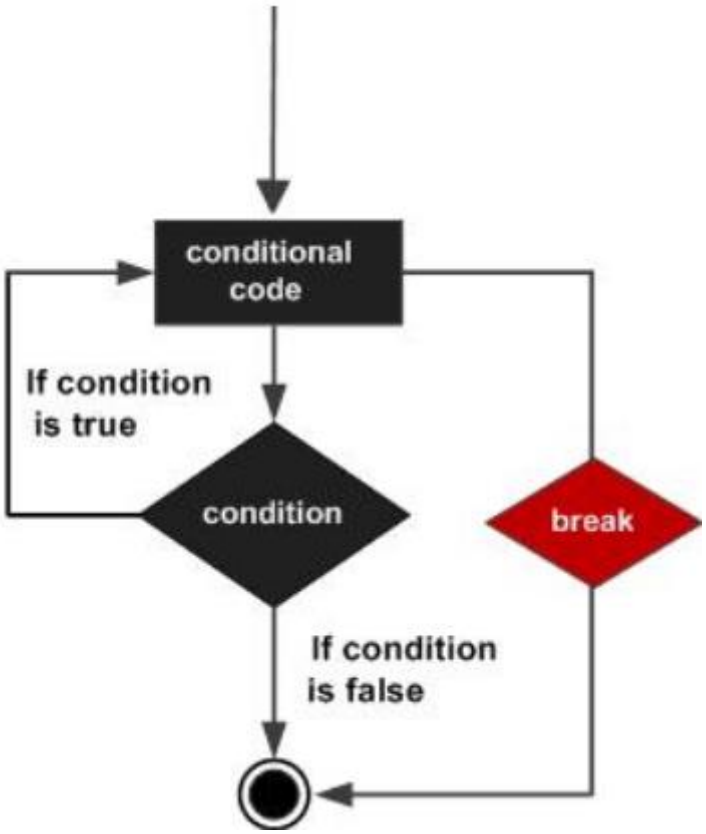
1. 當 break 語句出現在一個迴圈內時，迴圈會立即終止，且程序流將繼續執行緊接著迴圈的下一條語句。
2. 它可用於終止 switch 語句中的一個 case。

如果您使用的是巢狀迴圈（即一個迴圈內巢狀另一個迴圈），break 語句會停止執行最內層的迴圈，然後開始執行該塊之後的下一行程式碼。

語法

```
break;
```

流程圖



實例

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
using namespace std;

int main()
{
    int a = 10;

    do
    {
        cout << "a 的值：" << a << endl;
        a = a + 1;
        if( a > 15)
        {
            // 終止迴圈
            break;
        }
    }while( a < 20 );

     system("pause");
     return 0;
}
```


當上面的程式碼被編譯和執行時，它會產生下列結果：

```
a 的值： 10
a 的值： 11
a 的值： 12
a 的值： 13
a 的值： 14
a 的值： 15
```

continue 語句

continue 語句有點像 break 語句。但它**不是強迫終止**。

continue 會跳過當前迴圈中的程式碼，**強迫開始下一次迴圈**。

對於 FOR 迴圈，

continue 語句會導致執行條件測試和迴圈增量部分。

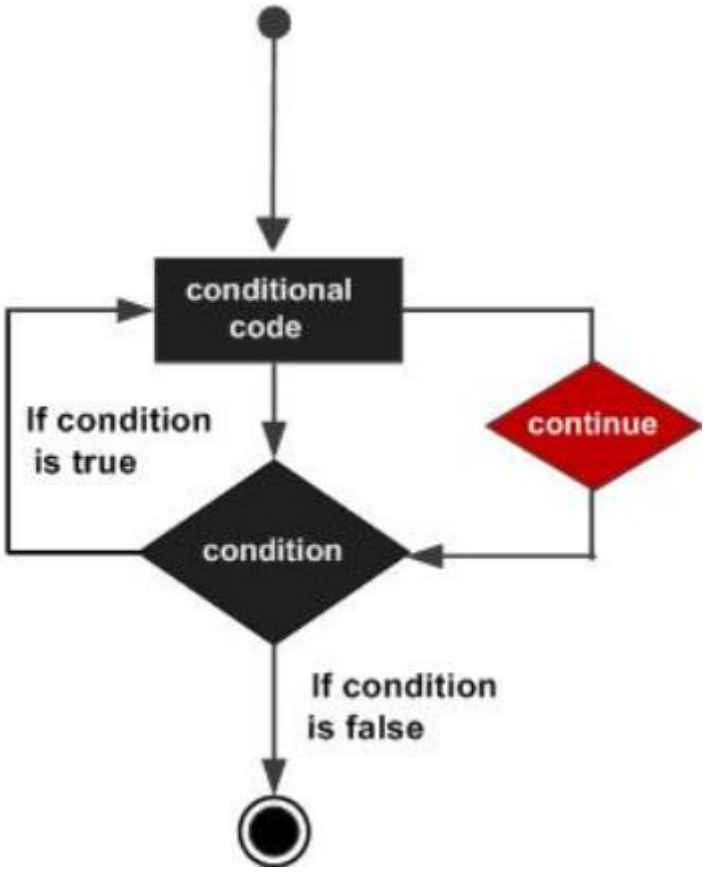
對於 WHILE 和 DO...WHILE 迴圈，

continue 語句會導致程序控制回到條件測試上。

語法

```
continue;
```

流程圖



實例

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
using namespace std;

int main()
{
    int a = 10;

    do {
        if( a == 15) {
            // 跳過重複
            a = a + 1;
            continue;
        }
        cout << "a 的值：" << a << endl;
        a = a + 1;
    }while( a < 20 );

     system("pause");
     return 0;
}

```


當上面的程式碼被編譯和執行時，它會產生下列結果：

```
a 的值： 10
a 的值： 11
a 的值： 12
a 的值： 13
a 的值： 14
a 的值： 16
a 的值： 17
a 的值： 18
a 的值： 19
```

goto 語句

goto 語句允許把控制無條件轉移到同一函數內的被標記的語句。

注意：在任何編程語言中，都不建議使用 goto 語句。

因為它使得程序的控制流難以跟蹤，使程序難以理解和難以修改。

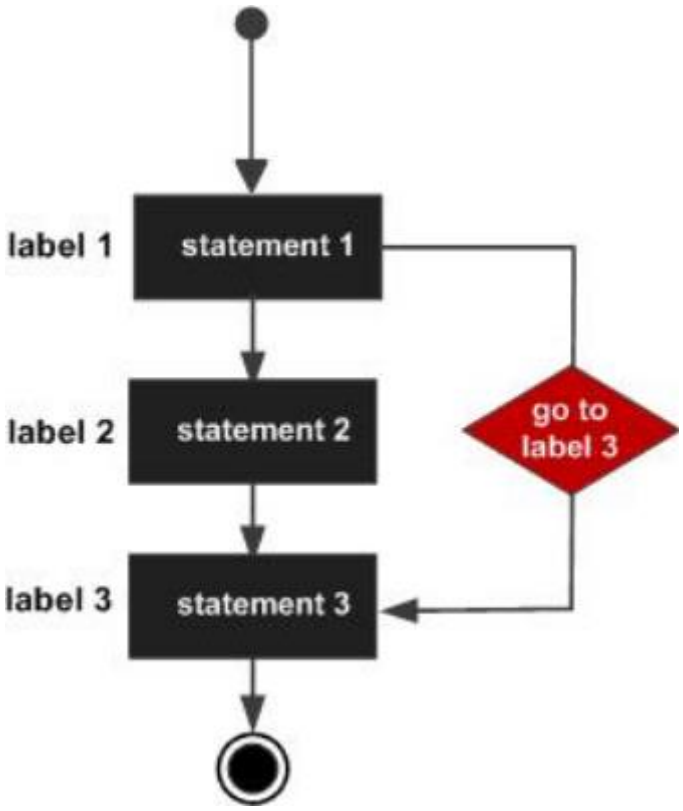
任何使用 goto 語句的程序可以改寫成不需要使用 goto 語句的寫法。

語法

```
goto label;
..
.
label: statement;
```

在這裡，label 是識別被標記語句的標識符，可以是任何除 C++ 關鍵字以外的純文字。標記語句可以是任何語句，放置在標識符和冒號（:）後邊。

流程圖



實例

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
using namespace std;

int main()
{
    int a = 10;

    LOOP:do {
        if( a == 15) {
            // 跳過重複
            a = a + 1;
            goto LOOP;
        }
        cout << "a 的值：" << a << endl;
        a = a + 1;
    }while( a < 20 );

     system("pause");
     return 0;
}
```


當上面的程式碼被編譯和執行時，它會產生下列結果：

```
a 的值： 10
a 的值： 11
a 的值： 12
a 的值： 13
a 的值： 14
a 的值： 16
a 的值： 17
a 的值： 18
a 的值： 19
```

goto 語句一個很好的作用是退出深巢狀程式結構。

例如，下面的程式碼片段：

```
for(...) {
  for(...) {
    while(...) {
      if(...) goto stop;
      .
      .
      .
    }
  }
}
stop:
cout << "Error in program.\n";
```

一個簡單的 break 語句在這裡不會起到作用，

因為它只會使程序退出最內層迴圈。

無限迴圈

如果條件永遠不為假，則迴圈將變成無限迴圈。

for 迴圈在傳統意義上可用於實現無限迴圈。

由於構成迴圈的三個表達式中任何一個都不是必需的，您可以將某些條件表達式留空來構成一個無限迴圈。

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <iostream>
4   using namespace std;
5
6   int main()
7   {
8
9       for( ; ; ) {
10        printf("This loop will run forever.\n");
11      }
12
13      system("pause");
14      return 0;
15  }
16
```


當條件表達式不存在時，它被假設為真。

您也可以設置一個初始值和增量表達式，但是一般情況下，C++ 程式設計師偏向於使用 for(;;) 結構來表示一個無限迴圈。

注意：您可以按 Ctrl + C 鍵終止一個無限迴圈。