4.2 do...while 迴圈

猜數字遊戲

有時候事情要先做了，看狀況才知道要不要繼續下去。例如我們小時候玩的猜數字遊戲，A 心裡選定一個 1~100 之間的整數由 B 來猜，每次 B 猜了之後，A 就要回應他 (1)再大一點；(2)再小一點；(3)答對了。直到 B 猜中那個數字為止。目標是在最少的猜測次數中，命中正確答案。

把它寫成程式，大致如下。主要問題在於，B 要先猜一個數字，你才知道他猜的對不對，要不要繼續讓他猜下去。我們按下面程式這樣設計，while 迴圈的第一次條件判斷會遇到問題 - 「yourguess 的值還沒確定」，因為 B 根本還沒開始猜。

```
int answer = 32;  // A選定的數字
int yourguess;    // 你猜的數字
int count = 0;    // 記錄猜了幾次

while(answer!=yourguess)  // B 根本就還沒開始猜，yourguess 是多少？
{
    cout << "請猜一個數字(1~100):";
    cin >> yourguess;
    count++;

    if(yourguess<answer)
    {
        cout << "再大一點" << endl;
    }
    else if(yourguess>answer)
    {
        cout << "再小一點" << endl;
    }
}

cout << "答對了！你一共猜了 " << count << "次" << endl;
```

解決的方法大致有兩種。

方法一：先在迴圈外猜一次

```
int answer = 32;  // A選定的數字
int yourguess;    // 你猜的數字

cout << "請猜一個數字(1~100):";
```

```
cin >> yourguess;
int count = 1;    // 這裡猜了一次

while(answer!=yourguess)
{
    if(yourguess<answer)
    {
        cout << "再大一點" << endl;
    }
    else if(yourguess>answer)
    {
        cout << "再小一點" << endl;
    }
    else
    {
         cout << "請猜一個數字(1~100):";
         cin >> yourguess;
         count++;
    }
}

cout << "答對了！你一共猜了 " << count << "次" << endl;
```

這種作法會在外面重覆一段程式碼。

方法二：給定 yourguess 一個保證錯的數值

這個作法可以保證 while 第一圈的條件判斷式一定成立，但是若是規則包含可以使用負數、範圍可自定，那就比較麻煩了。

```
int answer = 32;     // A選定的數字
int yourguess = -1;  // -1 在可能的答案範圍之外
int count = 0;       // 記錄猜了幾次

while(answer!=yourguess)  // 第一圈保證是 false
{
    cout << "請猜一個數字(1~100):";
    cin >> yourguess;
    count++;

    if(yourguess<answer)
    {
        cout << "再大一點" << endl;
    }
    else if(yourguess>answer)
    {
        cout << "再小一點" << endl;
    }
```

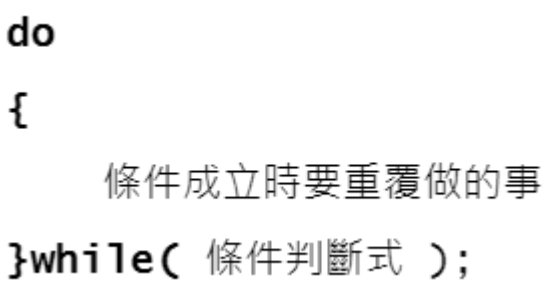
```
}

cout << "答對了！你一共猜了 " << count << "次" << endl;
```

do ... while

有別於 `while` 是先確定條件判斷式才進去執行一圈，我們還有一種 `do ... while` 敘述，可以在做完一圈工作後，再判斷要不要執行下一圈。

`do ... while` 的基本語法如下：



> 注意：do ... while(條件判斷式) 最後面有一個分號

使用 do ... while 就可以完美解決我們問題。

```
int answer = 32;  // A選定的數字
int yourguess;    // 你猜的數字
int count = 0;    // 記錄猜了幾次

do
{
    cout << "請猜一個數字(1~100):";
    cin >> yourguess;
    count++;

    if(yourguess<answer)
    {
        cout << "再大一點" << endl;
    }
    else if(yourguess>answer)
    {
        cout << "再小一點" << endl;
    }
}while(answer!=yourguess);
```

```
cout << "答對了！你一共猜了 " << count << "次" << endl;
```

比較 while 和 do ... while

絕大多數的情況下，只要用一點技巧，while 和 do ... while 可以互相取代。

以下的比較供大家判斷當下使用何者較恰當。

	判斷條件的時機	區塊被執行的次數
while	先檢查條件是否成立再做事	可能一次都不會被執行
do ... while	先做事再檢查條件是否成立	至少執行一次

Revision #11
★Created 2024-03-12 04:03:49 UTC by huihui
Updated 2024-03-25 12:50:54 UTC by huihui