4.3 遞增、遞減與複合指定運算子

遞增與遞減運算子

我們很常在迴圈裡用到 i = i+1 這樣的遞增敘述。

```
int i=1;

while(i<=10)
(
    cout << i << " ";
    i = i+1;  // 遞增 1
}
cout << endl;
```

```
1 2 3 4 5 6 7 8 9 10
```

這種情況可以使用 **遞增(increment)運算子** `++` 來處理。

```
int i=1;

while(i<10)
(
    cout << i << " ";
    i++;  // 遞增 1
}
cout << endl;
```

```
1 2 3 4 5 6 7 8 9 10
```

`i++` 就相當於 `i=i+1`。

同樣的 `i=i-1;` 可以用 **遞減(decrement)運算子** `--` 來處理。

```
int i=10;

while(i>0)
(
    cout << i << " ";
    i--;  // 遞減 1
}
cout << endl;
```

```
10 9 8 7 6 5 4 3 2 1
```

複合指定運算子

如果是增減 1 之外的數值，如 `i = i+2`，則可以用 **複合指定(compound assignment)運算子**。

```
int i=1;

while(i<10)
(
    cout << i << endl;
    i+=2;  // 遞增 2
}
```

`i+=2` 就相當於 `i=i+2`。

常用的複合指定運算子

運算子	範例	相當於
`+=`	`i += 2`	`i = i+2`
`-=`	`i -= 2`	`i = i-2`
`*=`	`i *= 2`	`i = i*2`
`/=`	`i /= 2`	`i = i/2`
`%=`	`i %= 2`	`i = i%2`

遞增、遞減運算子的評估時機

遞增運算子有兩種使用方式，如果我們要將 `變數i` 遞增 1。

- `i++`
- `++i`

以下兩個程式的執行結果相同。

使用 i++

```
int i=1;

while(i<10)
(
    cout << i << " ";
    i++;  // 遞增 1
}
cout << endl;
```

使用 ++i

```
int i=1;

while(i<10)
(
    cout << i << " ";
    ++i;  // 也是遞增 1
}
cout << endl;
```

那麼 `++` 放在變數的前面和後面有什麼差別呢？主要在於 **先遞增再評估其值** 還是 **先評估其值再遞增**。

看了以下這個實例應該就很清楚了。

```
int i=1;

cout << i++ << endl;  // 1
cout << i << endl;    // 2
cout << ++i << endl;  // 3
cout << i << endl;    // 3
```

執行到第3行時，cout 要輸出 i++ 的值，到底是 **要先輸出i的值，再遞增i** 還是要 **先遞增i，再輸出i的值**？

因為我們把 `++` 寫在 `i` 後面，所以當下是 **先評估 i 的值給 cout，之後再遞增 i**。

而在第5行，因為因為我們把 `++` 寫在 `i` 前面，所以當下是 **先遞增 i，再評估 i 的值給 cout**。

如果牽涉到指定(assign)運算時也是一樣。

```
int i=1;
int a;
```

```
a = i++;
cout << a << endl;  // 1
cout << i << endl;  // 2
a = ++i;
cout << a << endl;  // 3
cout << i << endl;  // 3
```

遞減運算子的運作方式相同就不再贅述。

⚠ 由於遞增遞減運算子使用在複雜的指定敘述中，很容易讓人在閱讀時搞錯評估時機和實際指定過去的值。所以建議只在很單純，絕對不會搞錯的地方使用。否則寧可用 (i+1) 或 (i-1) 這樣明確的寫法。

Revision #7
★Created 17 March 2024 04:43:10 by huihui
Updated 25 March 2024 12:49:03 by huihui