C/C++ const 的 3 種用法與範例

本篇 ShengYu 介紹 C/C++ const 的 3 種用法與範例，包含 C++ const 平常的一般基本用法以及 C++ const 在成員函式中的用法。

以下 C/C++ const 的用法介紹分別為這幾種，

- C/C++ const 加上變數前的用法
- C++ const 加在成員函式前面的用法
- C++ const 加在成員函式後面的用法

那我們開始吧！

C/C++ const 加上變數前的用法

這邊介紹 C/C++ const 加上變數前的用法，C/C++ const 加上變數前表示不能修改該變數，該變數為 read-only，

如下範例所示，宣告 `const int n = 5;` 之後如果嘗試對 n 進行修改的話會得到 `error: assignment of read-only variable ‘n’` 編譯錯誤訊息，

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     const int n = 5;
6
7     return 0;
8 }
```

這邊要另外舉個字串指標的例子，`const char *str` 雖然是不能修改其 str 指標指向的內容，但 str 指標本身卻是可以修改的，所以這部份在使用上需要特別注意，詳細說明如下，

如下例所示，宣告一個 `const char * name2 = "Amy";`，name2 表示指標指向的內容不可修改，如果嘗試對 name2 指標指向的內容進行修改的話會得到 `error: assignment of read-only location ‘*name2’` 編譯錯誤，例如下例中的 `name2[0] = 'B';` 就是對 name2 指向的內容進行修改，

但是 name2 指標本身是可以修改的，也就是可以修改 name2 指標指向別的地方，如下例中的 `name2 = name;` 將 name2 指向 name，這樣 name2 印出來的內容就會是 Tom 而不是 Amy，

如果要指標本身不可修改的話，可以像下中的 name3 前加上 const 變成 `const char * const name3`，這樣就是表示指標本身不可修改且指向的內容也不可修改，之後如果嘗試對 name3 的指標進行修改的話會得到 `error: assignment of read-only variable ‘name3’` 編譯錯誤訊息，

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main() {
6      char name[5] = "Tom";
7      cout << name << "\n";
8
9      const char * name2 = "Amy";
10
11     name2 = name;
```

```
12     cout << name2 << "\n";
13
14     const char * const name3 = "Amy";
15
16
17     return 0;
18 }
```

結果輸出如下，

```
1 Tom
2 Tom
```

如果換成整數指標的話就可能有這幾種情況，

```
1 const int * a = &b;
2 int const * a = &b;
3 int * const a = &b;
4 const int * const a = &b;
```

綜合上述指標加上 const 的用法大致分成兩種情況，一種就是不可修改的指標，另一種則是指標指向的內容(記憶體區塊)不可修改，
不可修改的指標：即指標不可修改，代表該指標永遠指向某塊記憶體位置
指標指向的內容(記憶體區塊)不可修改：即指標指向的記憶體區塊不能修改，只能讀取 read-only

C++ const 加在成員函式前面的用法

在 C++ 中有時候希望回傳的東西不能被修改的話，這時就可以使用 const 加在成員函式前面來達成這個目的，我們來看看下面這個例子，

在 main 函式裡要取得 `s.getName()` 成員函式回傳的變數話需要宣告一個 `const std::string& studentName`，由於 studentName 是 const 的關係所以之後我們就只能對這個變數作讀取不能修改值，如果嘗試對 studentName 進行修改的話會得到編譯錯誤，

因為 studentName 是 reference 參考的關係，所以之後使用 `s.setName("Tom")` 改變了 s 物件裡的 name 後，之後 studentName 裡面的值也會跟著改變，

```
1  #include <iostream>
2  #include <string>
3
4  class Student {
5  public:
6      Student() {}
7
8      const std::string& getName() {
9          return name;
10     }
11
12     void setName(std::string name) {
13         this->name = name;
14     }
15
16 private:
17     std::string name = "unknown";
18 };
19
20 int main() {
21     Student s;
```

```
22     const std::string& studentName = s.getName();
23     std::cout << studentName << "\n";
24
25     s.setName("Tom");
26     std::cout << studentName << "\n";
27     return 0;
28 }
```

結果輸出如下，

```
1 unknown
2 Tom
```

再舉個 STL 容器的例子，使用 std::queue 容器 將 1、2、3 元素推入後，之後使用 `front()` 取得頭部元素，這時我們只是需要把變數 n 印出來而已，所以不會對它進行修改，如果嘗試對 `const int &n` 修改的話會得到編譯錯誤的 `error: cannot assign to variable 'n' with const-qualified type 'const int &'` 訊息，

另外宣告 `int &n2 = q.front();` 參考的方式來修改 queue 頭部元素，之後 n 裡面的數值也會跟著改變，

```
1  #include <iostream>
2  #include <queue>
3
4  int main() {
5      std::queue<int> q;
6      q.push(1);
7      q.push(2);
8      q.push(3);
9      const int &n = q.front();
10
11     std::cout << n << "\n";
12
13     int &n2 = q.front();
14     n2 = 4;
15     std::cout << n << "\n";
16     return 0;
17 }
```

輸出結果如下，

```
1 1
2 4
```

C++ const 加在成員函式後面的用法

這邊介紹 C++ const 加在成員函式後面的用法，const 加在成員函式後面表示不能在該成員函式裡修改類別成員變數，因為該函式裡的存取類別成員都會是 read-only，範例如下，

如果在 `getCounter()` 成員函式裡嘗試對 counter 進行修改會得到編譯錯誤(`error: increment of member ‘Object::counter’ in read-only object`)，對其它類別成員 number 修改也是會得到編譯錯誤(`error: assignment of member ‘Object::number’ in read-only object`)，但是對 `getCounter()` 裡宣告的 number2 區域變數可以進行修改，

```
1  #include <iostream>
2  using namespace std;
3
```

```
4  class Object {
5  public:
6      Object() {}
7
8      int getCounter() const {
9
10
11         int number2;
12         number2 = 10;
13         return counter;
14     }
15
16     void addCount() {
17         counter++;
18     }
19
20 private:
21     int counter = 0;
22     int number = 0;
23 };
24
25 int main() {
26     Object o;
27     int counter = o.getCounter();
28     std::cout << counter << "\n";
29     o.addCount();
30     std::cout << counter << "\n";
31
32     int counter2 = o.getCounter();
33     std::cout << counter2 << "\n";
34     return 0;
35 }
```

以上就是 C/C++ const 的 3 種用法與範例介紹，
如果你覺得我的文章寫得不錯、對你有幫助的話記得 Facebook 按讚支持一下！

其它參考
https://welkinchen.pixnet.net/blog/post/48176548
https://docs.microsoft.com/zh-tw/cpp/cpp/const-cpp?view=msvc-170
https://blog.xuite.net/coolflame/code/16605512

其它相關文章推薦
如果你想學習 C++ 相關技術，可以參考看看下面的文章，
C/C++ 新手入門教學懶人包
C/C++ static 的 5 種用法
C/C++ extern 用法與範例
C/C++ call by value傳值, call by pointer傳址, call by reference傳參考 的差別