C++中const的用法详细总结  
作者：  
以下是对C++中const的用法进行了详细的总结分析，需要的朋友可以过来参考下，希望对大家有所帮助  
1. const修饰普通变量和指针  
const修饰变量，一般有两种写法：  
const TYPE value;  
TYPE const value;  
这两种写法在本质上是一样的。它的含义是：const修饰的类型为TYPE的变量value是不可变的。  
对于一个非指针的类型TYPE，无论怎么写，都是一个含义，即value只不可变。  
例如：  
const int nValue； //nValue是const  
int const nValue； // nValue是const  
但是对于指针类型的TYPE，不同的写法会有不同情况，例如：  
A. const char \*pContent;  
B. char \* const pContent;  
C. char const \*pContent;  
D. const char\* const pContent;  
对于前三种写法，我们可以换个方式，给其加上括号  
A. const (char) \*pContent;  
B. (char\*) const pContent;  
C. (char) const \*pContent;  
这样就一目了然。根据对于const修饰非指针变量的规则，很明显，A=C.  
  
- 对于A,C, const修饰的类型为char的变量\*pContent为常量，因此，pContent的内容为常量不可变.  
- 对于B, 其实还有一种写法： const (char\*) pContent;  
含义为：const修饰的类型为char\*的变量pContent为常量，因此，pContent指针本身为常量不可变.  
- 对于D, 其实是A和B的混合体，表示指针本身和指针内容两者皆为常量不可变  
总结:  
(1) 指针本身是常量不可变  
(char\*) const pContent;  
const (char\*) pContent;  
(2) 指针所指向的内容是常量不可变  
const (char) \*pContent;  
(char) const \*pContent;  
  
(3) 两者都不可变  
const char\* const pContent;  
还有其中区别方法：  
沿着\*号划一条线，  
如果const位于\*的左侧，则const就是用来修饰指针所指向的变量，即指针指向为常量；  
如果const位于\*的右侧，const就是修饰指针本身，即指针本身是常量。  
2. const修饰函数参数  
const修饰函数参数是它最广泛的一种用途，它表示函数体中不能修改参数的值(包括参数本身的值或者参数其中包含的值)。它可以很好  
void function(const int Var); //传递过来的参数在函数内不可以改变(无意义，因为Var本身就是形参)  
void function(const char\* Var); //参数指针所指内容为常量不可变  
void function(char\* const Var); //参数指针本身为常量不可变(也无意义， 因为char\* Var也是形参)  
参数为引用，为了增加效率同时防止修改。  
修饰引用参数时：  
void function(const Class& Var);//引用参数在函数内不可以改变  
void function(const TYPE& Var); //引用参数在函数内为常量不可变  
3. const 修饰函数返回值  
const修饰函数返回值其实用的并不是很多，它的含义和const修饰普通变量以及指针的含义基本相同。  
(1) const int fun1() 这个其实无意义，因为参数返回本身就是赋值。  
(2) const int \* fun2()  
调用时 const int \*pValue = fun2();  
我们可以把fun2()看作成一个变量，那么就是我们上面所说的1.(1)的写法，即指针内容不可变。  
(3) int\* const fun3()  
调用时 int \* const pValue = fun2();  
我们可以把fun2()看作成一个变量，那么就是我们上面所说的1.(2)的写法，即指针本身不可变。  
4. const修饰类对象/对象指针/对象引用  
const修饰类对象表示该对象为常量对象，其中的任何成员都不能被修改。对于对象指针和对象引用也是一样。  
const修饰的对象，该对象的任何非const成员函数都不能被调用，因为任何非const成员函数会有修改成员变量的企图。  
例如：  
复制代码 代码如下:  
  
class AAA  
{  
void func1();  
void func2() const;  
}  
const AAA aObj;  
aObj.func1(); ×  
aObj.func2(); 正确  
   
const AAA\* aObj = new AAA();  
aObj->func1(); ×  
aObj->func2(); 正确  
  
5. const修饰成员变量  
const修饰类的成员函数，表示成员常量，不能被修改，同时它只能在初始化列表中赋值。  
复制代码 代码如下:  
  
class A  
{  
…  
const int nValue; //成员常量不能被修改  
…  
A(int x): nValue(x) {}; //只能在初始化列表中赋值  
}  
  
6. const修饰成员函数  
const修饰类的成员函数，则该成员函数不能修改类中任何非const成员函数。一般写在函数的最后来修饰。  
复制代码 代码如下:  
  
class A  
{  
…  
void function()const; //常成员函数, 它不改变对象的成员变量. 也不能调用类中任何非const成员函数。  
}  
  
对于const类对象/指针/引用，只能调用类的const成员函数，因此，const修饰成员函数的最重要作用就是限制对于const对象的使用。  
7. const常量与define宏定义的区别  
(1) 编译器处理方式不同  
define宏是在预处理阶段展开。  
const常量是编译运行阶段使用。  
(2) 类型和安全检查不同  
define宏没有类型，不做任何类型检查，仅仅是展开。  
const常量有具体的类型，在编译阶段会执行类型检查。  
(3) 存储方式不同  
define宏仅仅是展开，有多少地方使用，就展开多少次，不会分配内存。  
const常量会在内存中分配(可以是堆中也可以是栈中)。