**[const!指针!](http://www.cnblogs.com/biyeymyhjob/archive/2012/07/12/2588074.html)**

const的实现

const仅对编译器有效，严格来说，const不是“不可以修改”，而是告知编译器：我的变量不应该被修改，如果编译器发现程序企图修改这个变量，就会报错。有些情况，const变量有可能被编译器以外的东西修改，例如硬件中断等等。

1.指向const对象的指针   
　　可以这样理解指向const对象的指针：指向const对象的指针就是一个指针，不能通过它来修改它所指向的对象的值

      声明方法：const int \*p;

      const对象在初始化是不允许对其值进行修改的，因此，我们不能用一个普通指针指向一个const对象 下面的赋值会引起错误

      const int i =1；

      int \*p = &i;

      如果可以用普通指针修饰的话，就意味着可以改动其值 这const将失去意义

      正确使用方法

      const int i =1;

      const int \*p = &i; ( 或者 int const \*p=&i)

      这样，利用指向const对象的指针也是不能修改它所指向的const对象的值的。

      需要注意的两点：

      指向const对象的指针本身不是const类型（这也是它与const指针的主要不同点），所以它可以指向另一个const对象

      指向const对象的指针可以被赋予一个非const对象的地址，但是此时试图通过此指针来修改对象的值的操作是非法的

 2.const指针

   可以这样理解const指针：

    const指针就是一个指针，它本身就是const类型，所以将它初始化后不能改变它的指向，既不能让它指向新的对象。

    声明方法：

     int \*const p;//指向非const对象的const指针

    const int \*const p;//指向const对象的const指针   
  
　  由以上声明方法可以看出，const指针可以指向const对象和非const对象，但是两者的声明方法是不同的。   
  
　  使用const指针不可以修改其地址值，但是const指针指向非const对象，就可以利用它修改它所指向的对象的值。

PS:

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。

     总结：

     如果指针名前紧邻的关键字为const，那么它就是一个const指针；如果声明指针所指向的对象类型前有const关键字，那么它就是一个指向cosnt对象的指针。

      参考《Effective c++》 如果const位于星号的左侧，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位于星号的右侧，const就是修饰指针本身，即指针本身是常量