[1、const修饰一般常量及数组](#58dqpi1501034769863)

[2.const修饰指针变量\*及引用变量&](#16daeq1500464665314)

[const修饰指针（\*）：](#84qcao1500464728993)

[const修饰引用（&）：](#19neqk1500464902704)

[int const &a=x;](#80voaa1500464949896)

[3、函数中使用CONST](#33aevk1500464979567)

[(1)const修饰函数参数](#3bnsm1500465463077)

[(2)const 修饰函数返回值](#11vtwv1500465678234)

[(3)类中的成员函数：A fun4()const; 其意义上是不能修改所在类的的任何变量。](#29ewsf1500464677315)

[4、常用：](#35wdmm1500465980281)

**1、const修饰一般常量及数组**

const int a=10;               等价的书写方式：     int const a=10;

const int arr[3]={1,2,3};                        int const arr[3]={1,2,3};

对于类似这些基本数据类型，修饰符const可以用在类型说明符前，也可以用在类型说明符后，其结果是一样的。在使用这些常量的时候，只要不改变这些常量的值便好。

**2.const修饰指针变量\*及引用变量&**

**const修饰指针（\*）：**

const int\* a = & [1]

int const \*a = & [2]

int\* const a = & [3]

const int\* const a = & [4]

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。

因此，[1]和[2]的情况相同，都是指针所指向的内容为常量，这种情况下不允许对内容进行更改操作，如不能\*a = 3 ；

[3]为指针本身是常量，而指针所指向的内容不是常量，这种情况下不能对指针本身进行更改操作，如a++是错误的；

[4]为指针本身和指向的内容均为常量。

**const修饰引用（&）：**

int const &a=x;

const int &a=x;

int &const a=x;//这种方式定义是C、C++编译器未定义，虽然不会报错，但是该句效果和int &a一样。

这两种定义方式是等价的，此时的引用a不能被更新。如：a++ 这是错误的。

**3、函数中使用CONST**

**(1)const修饰函数参数**

eg：参数指针所指内容为常量不可变

void function(const char\* Var);

即将函数参数 作为const修饰符

**(2)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。

    a.const int fun1() //这个其实无意义，因为参数返回本身就是赋值。

    b. const int \* fun2()  //调用时 const int \*pValue = fun2();

                          //我们可以把fun2()看作成一个变量，即指针内容不可变。

    c.int\* const fun3()   //调用时 int \* const pValue = fun2();

                          //我们可以把fun2()看作成一个变量，即指针本身不可变。

**(3)类中的成员函数：A fun4()const; 其意义上是不能修改所在类的的任何变量。**

**4、常用：**

const string & get\_meet\_id() const { return current\_meet\_.getMeetId(); }

void set\_room(const string & value);

