const variable member

```
//const成员在被初始化之后就不可以被修改
//一般在private里面定义，在cons中初始化
#include<iostream>
using namespace std;
class A {
public:
    // 使用初始化列表来初始化const成员变量'a'
    A(int x) : a(x) {}
    //不可以用原来的写法
    //A(int x) {a = x;}

    void print() {
        cout << a << endl;
    }

    // 由于'a'是const，这个函数是无效的，不能编译
    // void change() {a = 3;}//不能被改变
private:
    const int a;
};

//int main() {
//    A obj(5);
//    obj.print();
//    // obj.change(); // 这会导致编译错误，因为'a'是const的。
//    return 0;
//}
```

一定要注意const成员一般在private中被定义了之后，一般在construction中初始化

```
// 使用初始化列表来初始化const成员变量'a'
    A(int x) : a(x) {}
    //不可以用原来的写法
    //A(int x) {a = x;}
```

注意const成员的初始化有特殊写法，一定要记住

`类名(数据类型 形参) : 变量名(传入形参) {}`

在平常的构造函数写法中，可以用这种方法，但是这里对于const成员的构造函数可不能用之前构造函数的写法，这里有它独有的写法

const function

```
#include<iostream>
using namespace std;
class B {
public:
    B(int x) { b = x; }
    void print() {
        cout << b << endl;
    }
    void change()const { cout << " " << endl; }
    //void change() const { b = 3; }//有const修饰的函数体内部不可以改变某个实参的值
private:
    int b;
};
```

那么有const变量，有没有const函数呢，有的

注意这里的const函数可不是专门来操作const变量的意思，不要搞混淆了

函数的常量属性一般加在函数名和传入参数括号后，有const修饰的函数体内部不可以改变某个实参的值

基本上加上了const之后这个函数体内部就不能改变参数的值了，做到“只读”的功能。