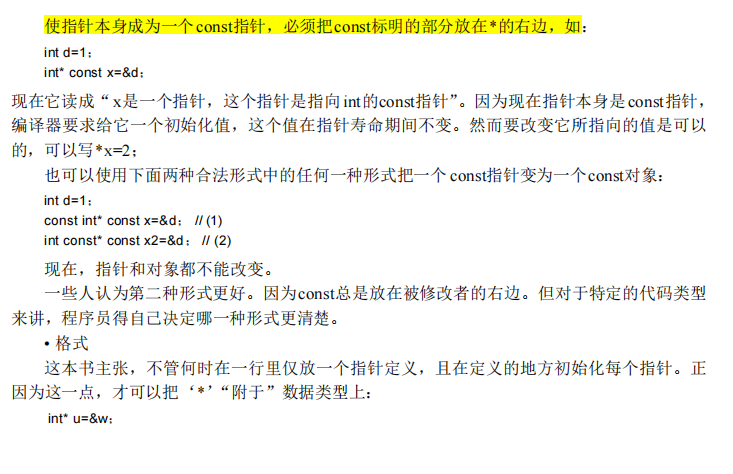
1. 关于const指针那些事
2. 面向对象的三个基本特征是封装、继承、和多态
3. (A) Inheritance 继承

(B) Polymorphism 多态

(C) overloading 重载

(D) Encapsulation 封装

4. 类定义是，未声明private和public，默认是private

5. const和引用都必须在定义的时候赋值

6. 友元函数是非成员函数，

**友元函数的声明位置首先必须是在类中**，至于是在public,protected,private无所谓,**外界都可以直接调用**

友元函数的实现位置,也就是具体的定义(非声明)所在位置第1种方式(推荐):在宿主类中声明的同时直接提供定义。第二种：将宿主类用一个新的namespace框起来,然后在其他cpp文件中提供友元的定义,但是需要使用类似 myNameSpace::operator<<() 这样的命名空间限定符

7. Const可修饰：函数参数不变、返回值不变、函数本身不修改类的数据成员

8. 在以下三种情况下需要使用初始化成员列表：

情况一、需要初始化的数据成员是对象的情况(这里包含了继承情况下，通过显示调用父类的构造函数对父类数据成员进行初始化)；

情况二、需要初始化**const**修饰的类成员；

情况三、需要初始化**引用成员**数据；

9.

1.类里面的任何成员变量在定义时是不能初始化的。

2.一般的数据成员可以在构造函数中初始化。

3.**const数据成员必须在构造函数的初始化列表中初始化。**

4.**static要在类的定义外面初始化**。

5.数组成员是不能在初始化列表里初始化的。

6.不能给数组指定明显的初始化。

这6条一起，说明了一个问题：C++里面是不能定义常量数组的！因为3和5的矛盾。这个事情似乎说不过去啊？没有办法，我只好转而求助于静态数据成员。

到此，我的问题解决。但是我还想趁机复习一下C++类的初始化：

1.初始化列表：CSomeClass::CSomeClass() : x(0), y(1){}

2.类外初始化：int CSomeClass::myVar=3;

3.const常量定义必须初始化，C++类里面使用初始化列表;

4.C++类不能定义常量数组。

10.指针和引用的区别：

针和引用都是地址的概念，指针指向一块内存，它的内容是所指内存的地址；引用是某块内存的别名。

* 程序为指针变量分配内存区域，而不为引用分配内存区域。
* 指针使用时要在前加 \* ，引用可以直接使用。
* 引用在定义时就被初始化，之后无法改变；指针可以发生改变。 即引用的对象不能改变，指针的对象可以改变。
* 没有空引用，但有空指针。这使得使用引用的代码效率比使用指针的更高。因为在使用引用之前不需要测试它的合法性。相反，指针则应该总是被测试，防止其为空。
* 对引用使用“sizeof”得到的是变量的大小，对指针使用“sizeof”得到的是变量的地址的大小。
* 理论上指针的级数没有限制，但引用只有一级。即不存在引用的引用，但可以有指针的指针。
* int \*\*p //合法
* int &&p //非法
* ++引用与++指针的效果不一样。
* 例如就++操作而言，对引用的操作直接反应到所指向的对象，而不是改变指向；而对指针的操作，会使指针指向下一个对象，而不是改变所指对象的内容。

11.抽象类不可new子类可new（有纯虚函数的是抽象类）

12. C++用虚函数实现动态多态性

13.所有类共用一个静态数据成员，静态成员函数只能访问静态数据成员

14. 友元关系不能被继承，基类的友元对派生类没有特殊的访问权限

15. 不能继承的东西：

①构造函数和析构函数不能被继承

②operator= 也不能被继承，因为它完成类似于构造函数的活动。这就是 说，尽管我们知道如何由等号右边的对象初始化左边的对象的所有成员，但这并不意味着这个初始化在继承后仍有意义。

16.函数决定要素：参数类型表，返回类型，函数名称

17.父类可以引用子类，而子类不可以引用父类

18.虚继承才会有一个子类，否则有多个子类

19.“=”必须是成员函数

20.命名空间：

namespace加上作用域：

: : 全局作用域访问符

名字空间名称: : 名字空间作用域访问符

类: : 类作用域访问符

21.友元函数不能用const修饰