|  |  |
| --- | --- |
| 1. 定义派生类 2. 派生类中的虚函数 3. 派生类对象及派生类向基类的类型转换 4. 派生类构造函数 5. 派生类使用基类的成员 6. 继承与静态成员 7. 派生类的声明 8. 被用作基类的类 9. 防止继承的发生 | 类派生列表  单继承  override  class Bulk\_qquote:public Quote;错误的声明  派生类到基类的  静态成员  Static void statemem();  直接基类，间接基类  Class NoDerivd Final |
| 1. 定义派生类 2. 必须通过使用**类派生类列表**明确指出它是从哪个基类继承而来的 3. 访问说明符：public protected private 4. 访问说明符作用：控制派生类从基类继承而来的成员是否对派生类的用户可见 5. 单继承,继承一个类 6. 派生类中的虚函数 7. 派生类经常覆盖它继承的虚函数（可以不覆盖）。那么这个虚函数就是基类的版本，类似其它普通成员 8. 可以在它覆盖的函数前使用virtual，不是必须 9. 显式的注明它使用某个成员函数覆盖了它继承的虚函数，加关键字override 10. 派生类对象及派生类向基类的类型转换 11. 一个派生类对象包含多个组成部分：自己定义的（非静态）成员的子对象（成员函数或成员变量），以及一个与该派生类继承的基类对应的子对象（成员函数或者变量） 12. 因为派生类对象中含有与其基类对应的组成部分，所以可以把派生类对象当做基类对象来使用， 13. 也能把基类的指针或引用绑定到派生类对象中的基类部分上 14. 派生类到基类的类型转换，编译器：隐式地执行   note：派生类对象中含有与其基类对应的组成部分   1. 派生类构造函数 2. 派生类并不能直接初始化基类部分的成员，需要使用基类的构造函数来初始化它的基类部分   note：每个类控制它自己的成员初始化过程   1. 因为类可能有私有成员，派生类不能访问私有成员给他进行值初始化，而构造函数则不会。 2. 若不指定基类初始化函数，则派生类的基类部分会像数据成员一样执行默认初始化,就是执行基类的构造函数，即对数据成员默认初始化 3. 必须使用构造函数初始化列表来将实参传递给基类构造函数   note：首先初始化基类的部分，然后按照声明的顺序依次初始化派生类的成员   1. 派生类使用基类的成员 2. 派生类可以使用基类的公有 受保护的成员 3. 关键概念：遵循基类的接口   每个类负责定义各自的接口。要想与类的对象交互必须使用该类的接口，即使这个对象是派生类的基类部分也是如此   1. 派生类应该遵循基类的接口，并且通过调用基类的构造函数来初始化那些从基类中继承而来的成员 2. 继承与静态成员 3. 如果基类定义了一个静态成员，则再整个继承体系中只存在该成员的唯一定义。 4. 基类的静态成员，派生类可不可以访问依照访问限定符 5. 派生类的声明 6. 派生类的声明不包含派生列表 7. 声明语句：是让程序知晓某个名字的存在以及该名字表示一个什么样的实体 8. 各种细节必须与类的主体一起出现 9. 被用作基类的类 10. 想让一个类作为基类，必须定义而非仅声明。因为派生类继承基类的成员，为了使用这些成员，派生类要知道他们是什么 11. 隐含意思是派生类不能派生它本身 12. 一个类是基类，也可以是一个派生类   class A  class B : A  class C : B  直接基类: A对于B  间接基类：A对于C  C拥有 B 与A的所有的成员  最终的派生类将包含它的直接基类的子对象以及每个间接基类的子对象   1. 防止继承的发生 2. 不希望其它类继承他，不想考虑它是否适合作为一个基类 3. 关键字final | |