Chap04 继承与派生

程勇

北京化工大学

信息科学与技术学院计算机系

Sept. 2020

课程提纲

- Chap01 绪论
- Chap02 面向对象程序设计方法
- Chap03 重载与类型转换
- Chap04 继承与派生
- Chap05 多态性
- Chap06 输入输出流
- Chap07 容错与异常处理
- Chap08 模版

本章提纲

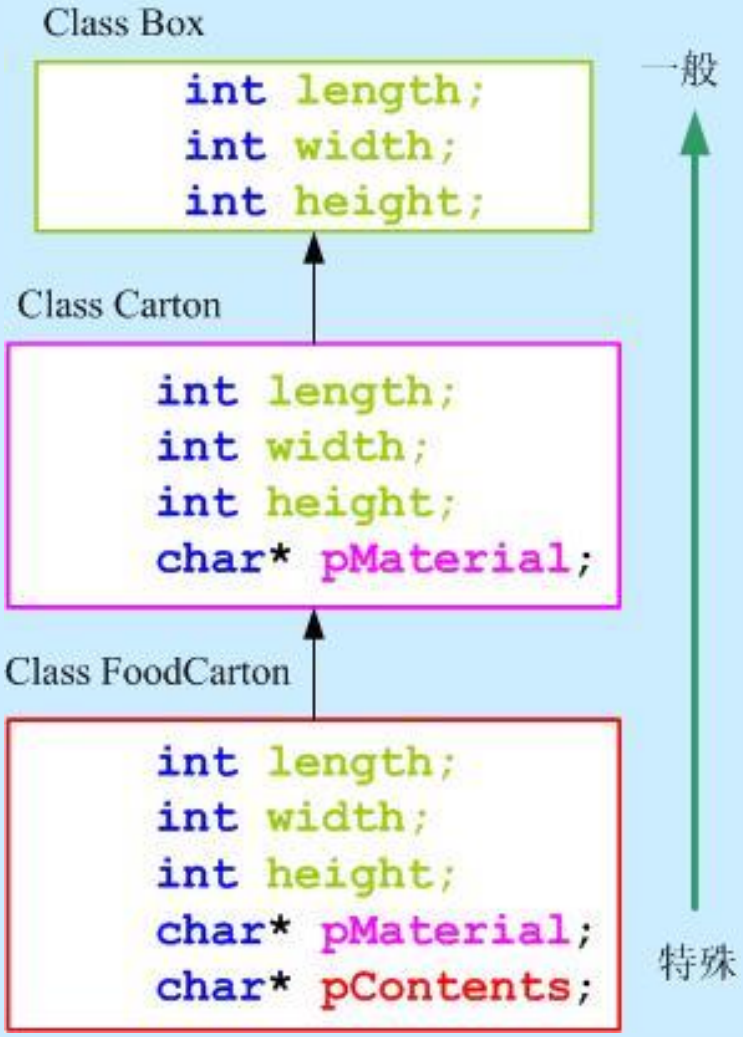
- 基本概念
  - 继承/基类/派生类
- 单继承
  - 共有/私有/保护成员
- 派生类初始化
  - 构造函数
  - 析构函数
- 访问基类成员
  - 同名/静态成员
- 多继承
  - 多继承概念
  - 二义性
  - 虚基类

基本概念

- 继承与派生
  - 继承是面向对象程序设计保证代码可复用性的最重要的手段，它允许程序员在保持原有类特性的基础上进行扩展，增加功能。继承体现了面向对象程序设计的层次结构以及由简单到复杂的认识过程；
  - 派生反映了事物之间的联系，事物的共性与个性之间的关系。派生与独立设计若干相关的类，前者工作量少，重复的部分可以从基类继承来，不需要单独编程；
- 基类和派生类
  - C++通过类派生(class derivation)的机制来支持继承。被继承的类称为基类(base class)或超类(superclass)，新的类为派生类(derived class)或子类(subclass)，基类和派生类的集合称作类继承层次结构(class hierarchy)；
  - 派生类中自动包含基类的所有数据成员和函数成员(但有一些限制)，还可拥有自己的数据成员和函数成员；

基本概念(续)

- 类层次结构
  - 基类通过表示更一般的概念和实体，而派生类则更为特殊；
- 聚合关系
  - 汽车由引擎、车厢等组成，使用Auto继承Engine无意义；
- 如何发现类层次结构
  - 首先进行is-a测试；
  - 如果通过is-a测试，则测试基类中是否有特性不适用于派生类，如果没有则继承是可行的；
  - 进行has测试，如果通过包含测试，则使用聚合；



从基类派生新类

- 一般格式

```
class 派生类名: 访问限定符 基类1 <,访问限定符 基类n>{
public:
        公有成员列表;
protected:
        受保护成员列表;
private:
        私有成员列表;
};
```

  - 其中基类1、…、基类n是已声明的类，在派生类定义的类体中给出的成员称为派生类成员，它们给派生类添加了不同于基类的新的属性和功能；
  - 当一个派生类有多个基类时称为多重继承(multiple-inheritance)，如果派生类只有一个直接基类则是单继承(single-inheritance)；

基类访问限定符

- 基类访问限定符对继承成员的影响

派生方式	基类限定符	派生类访问控制	派生类外部访问控制
public	public	public	可直接访问
	protected	protected	不可直接访问
	private	继承但不可直接访问	不可直接访问
protected	public	protected	不可直接访问
	protected	protected	不可直接访问
	private	继承但不可直接访问	不可直接访问
private	public	private	不可直接访问
	protected	private	不可直接访问
	private	继承但不可直接访问	不可直接访问

基类访问限定符(续)

- 注意事项
  - 受保护成员和私有成员仅在派生类中有区别：
    - 基类的私有成员只能有基类的函数来访问；
    - 基类的受保护成员可以由派生类中的函数来访问；
  - 默认的基类访问指定符为private，即如果省略访问指定符，就等于使用private作为访问指定符；
  - 在程序设计中，绝大多数都使用public作为访问指定符；
  - 可以使用using声明改变继承成员的访问指定符，例如

```
class Truck : private Auto{
public:
    using Auto::mileage;  // contents成员在基类中为共有或受保护的
// rest of the class definition
}
```

  - 不能将using声明应用于基类的私有成员，因为私有成员在派生来中是不能访问的；

派生类构造函数

- 派生类构造函数
  - 一般在创建派生类对象时都自动调用了基类的默认构造函数；
  - 也可以在派生类的构造函数中指定调用某个基类的构造函数；
- 执行次序
  - 调用基类构造函数，按它们在派生类定义的先后顺序调用；
  - 调用成员对象的构造函数，按它们在类定义中声明的先后顺序调用；
  - 派生类的构造函数体中的操作；
- 注意事项
  - 调用基类构造函数的表示法与在构造函数中初始化数据成员的表达法完全相同；
  - 基类中非私有的数据成员可以在派生类中访问，但它们不能在派生类构造函数的初始化列表中进行初始化；

副本构造函数和析构函数

- 副本构造函数
  - 必须将副本构造函数的参数指定为引用；
  - 一定要确保正确初始化派生类对象的成员，包括所有继承的成员和派生类中特有的成员；

```
Carton::Carton(const Carton& aCarton) : Box(aCarton){…};
```

- 析构函数
  - 在派生来中必须把新增一般成员处理好，而对新增的成员对象和基类的善后工作，系统会自己调用成员对象和基类的析构函数来完成；
  - 派生类析构函数的执行顺序与构造函数相反
    - 首先对派生类新增一般成员析构；
    - 然后对新增对象成员析构；
    - 最后对基类成员析构；

重复的成员名

- 基类和派生类出现同名的成员
  - 名称的重复并不会阻碍继承；
- 同名的数据成员
  - 如果同名的数据成员在基类中为private，则在派生类中不可直接访问，否则可以使用<基类名::数据成员>来访问基类数据成员；
- 同名的函数成员

  由于重载的函数必须在同一作用域内，而基类和派生类定义了不同的作用域，故这种情况下不是重载。具体分两种情况考虑：
  - 函数名称相同，参数列表不同

    必须使用using声明，在派生类中引入基类成员函数的限定名，然后派生类函数可以调用则两个函数；
  - 函数名称相同，参数列表也相同

    在函数签名都相同的情况下，使用<基类名::成员函数名>来区分基类函数和派生类函数；

多继承

- 基本概念
  - 由多个基类共同派生出新的派生类，这样的继承结构被称为多重继承或多继承(multiple-inheritance)；

```
class CerealPack : public Carton, public Contents{
}
```

  - 基类在冒号后指定，用逗号隔开，每个基类都有自己的访问类型指定符，如省略则视为private；
- 继承成员模糊性
  - 即由继承而得到的函数同名问题；
  - 理想的解决方案就是重新编写类，从而避免重复的成员名；
  - 另一个解决方法就是使用<基类名::>来确定，如

```
CerealPack packofflakes;
packofflakes.Carton::volume();
packofflakes.Contents::getWeidht();
```

虚基类

- 重复的继承
  - 在使用多继承时，不能将一个类多次用作直接基类，但可能间接基类重复的现象；
  - 派生类中包括一个基类的多个子对象版本；
  - 基类重复出现是可行的，但必须限定对重复基类成员的引用，而且还必须用限定符来指定初始化那个重复基类的对象；
- 虚基类
  - 为避免基类的重复，即在任何派生类中，基类只重复一次，为此可使用关键字virtual，把类声明为虚基类；

```
class Box : public virtual Common{…}
class Contents : public virtual Common{…}
class Freebie : public <virtual> Common{…}
class CerealPack : public Box, public Contents, public Freebie{…}
```

  - 如果把Common声明为Freebie的虚基类，则CerealPack就只继承Common的一个子对象，否则有两个子对象；

本章小结

- 基本概念
- 单继承
- 派生类初始化
- 访问基类成员
- 多继承

- 课程网站
  - http://www.jiaowu.buct.edu.cn