

C++大学基础教程

第8章 类与对象

北京科技大学
信息基础科学系

◆ 类是实现C++面向对象程序设计的基础。面向对象程序设计的基本特点有：抽象、封装、继承和多态，类把数据和函数封装在一起，是C++封装的基本单元。

第8章 类与对象

- 8.1 类和对象的定义
- 8.2 对象的使用
- 8.3 构造函数
- 8.4 析构函数
- 8.5 拷贝构造函数
- 8.6 类的静态成员
- 8.7 类成员的保护和使用
- 8.8 类的组合（不要求）
- 8.9 面向对象分析和设计（不要求）

8.1 类和对象的定义

8.1 类和对象的定义

- ◆ 一个类表示现实生活中的一类事物，比如“学生”
 - 事物有相应的特征或属性，它们就是类的数据成员；
 - 事物可能有行为动作，也可能被某些行为动作所操作，这些都用函数来实现，这些函数和类有着不可分割的关系，是构成类的函数成员，或者叫成员函数。
- ◆ 在C++中，类实际上相当于一种用户自定义的数据类型。

8.1 类和对象的定义

- ◆ 对象是一类事物中的一个具体的个体。
- ◆ 在定义了类类型后，就可以定义该类型的变量，这个变量就称为类的对象（或实例）。所以，从程序设计的角度来看，对象就相当于变量。

8.1 类和对象的定义

◆ 举例：

```
class Student //类的定义
{private:
    int id;
    char name[20];
    int age;
    float score;
public:
    void getscore()
    void printstu()
};
```

1、使用类对象

- ◆ 如果已经知道某个类的功能和接口函数，就可以使用它了。先定义类对象，通过对象调用接口函数实现一定的功能。

```
class student stu01; //class也可以省略不写  
stu01.getscore();  
stu01.printstu();
```

- ◆ 这个过程的本质是：根据对象所需要完成的任务，向对象发送消息。对象收到消息后，调用接口中的相应的函数，完成它的功能。对象只需要知道“做什么”，具体工作由成员函数来完成。

2、类的声明

- ◆ 类的声明具体说明类的组成，声明类的语法形式为：

```
class 类名称
```

```
{ public:
```

```
    公有成员
```

```
protect:
```

```
    保护型成员
```

```
private:
```

```
    私有成员
```

```
};
```

2、类的声明

- ◆ 其中，“成员”既可以是数据成员，也可以是成员函数的原型。
 - 数据成员的声明方式与一般变量相同。
 - 函数成员是描述类的对象可以进行的操作，一般在类中声明原型，在类声明的外面定义函数的具体实现。

2、类的声明

- ◆ 关键字public、protect、private说明类成员的访问控制属性。
 - 私有（private）成员只允许本类的成员函数来访问；
 - 公有（public）成员是类对外的接口，在类声明和类（函数）实现之后，类的对象可以访问公有成员。
 - 保护型（protected）成员的可访问性和私有成员的性质相似。

2、类的声明

关于访问控制属性，注意：

- ◆ 在类声明中，三种访问控制属性，可以按任意次序出现，也可以不出现。public等关键字也可以多次出现，但是一个成员只能具有一种访问控制属性。
- ◆ 如果不写访问控制属性关键字，默认的是private。在书写时通常习惯将公有类型放在最前面，这样便于阅读，因为它们是外部访问时所要了解的。
- ◆ 一般情况下，一个类的数据成员应该声明为私有成员，这样封装性较好。一个类应该有一些公有的函数成员，作为对外的接口，否则别的代码无法访问类

3、类的成员函数

◆ 类的成员函数描述的是类的行为或操作。函数的原型声明要在类的主体中，原型说明了函数的参数表和返回值类型。而函数的具体实现一般是写在类声明之外的。

◆ 在类的外部定义成员函数的语法形式为：

返回值类型 类名::成员函数名(参数表)

{

函数体

}

3、类的成员函数

- ◆ 如果在类的内部定义成员函数的具体实现，则该成员函数为内联成员函数。
- ◆ 函数定义时没有任何的附加说明，所以称为隐式声明的内联成员。

4、对象

- ◆ 定义一个对象和定义一个一般变量相同。
- ◆ 定义变量时要分配存储空间，同样，定义一个对象时要分配存储空间，一个对象所占的内存空间是类的数据成员所占的空间总和。类的成员函数存放在代码区，不占内存空间。
- ◆ 类的成员是抽象的，对象的成员才是具体的。类的数据成员都不会有具体的属性值。只有对象的成员才会有具体的属性值。

4、对象

- ◆ 声明了类及其对象，在类的外部（指类定义和成员函数的实现代码之外），就可以访问对象的公有成员（包括数据成员和函数成员）了。
- ◆ 在类的外部，只能通过对象访问类的公有成员；在类的成员函数内部，可以直接访问类的所有成员，这就实现了对访问范围的有效控制。

5、类的作用域与可见性

- ◆ 类作用域是指类定义和相应的成员函数定义的范围，通俗地称为类的内部。C++认为一个类的全部成员都是一个整体的相关部分。一个类的所有成员位于这个类的作用域内，在该范围内，一个类的成员函数对本类的其它成员具有无限制的访问权。

8.2 对象的使用

8.2 对象的使用

- ◆ 对象是类的一个具体的实例，类和对象的关系相当于普遍与特殊的关系。在C++中，类是一个自定义的数据类型，对象是该数据类型的一个变量。
- ◆ 所以，可以定义一个全局的对象，也可以在函数体中定义一个局部的对象，或者动态地从堆中申请空间来创建一个对象，还可以定义对象数组，以及使用对象作为函数的参数与返回值。

1、对象指针

◆ 声明对象指针的一般语法形式为：

类名* 对象指针名；

◆ 使用对象指针访问对象的成员，语法形式为：

对象指针名->公有成员；

◆ 例如：

```
Clock c; //在栈中分配Clock型存储空间
```

```
Clock* pc1=new Clock; //在堆中分配Clock型存储空间
```

```
Clock* pc2=&c;
```

```
pc1->SetTime(12, 5, 0);
```

2、 this指针

问：一个类中所有对象调用的成员函数都执行同一段代码。那么，成员函数又是怎么识别当前是那个对象在访问呢？

答：this指针指出了成员函数当前所操作的数据所属的对象。不同的对象调用成员函数时，this指针将指向不同的对象，也就可以访问不同对象的数据成员。

3、对象数组

- ◆ 对象数组的元素是对象，不仅具有数据成员，而且还有函数成员，可以通过数组元素调用成员函数。

4、对象用做函数的参数和返回值

- ◆ 可以将对象作为参数传递给一个函数或从函数返回一个对象。
- ◆ 直接用对象作为参数，调用函数时，进行值传递，实参的值要复制给形参，如果类的数据成员较多时，需要一一复制，这种参数传递方式效率不高，可以使用对象指针或对象引用方式来传递函数参数。

8.3 构造函数

8.3 构造函数

- ◆ 每个对象区别于其他对象的地方主要有两个，外在的区别就是对象的标识符，即对象的名称，而内在的区别就是对象自身的属性值，即数据成员的值。
- ◆ 在定义一个对象的时候要给它分配存储空间，也可以同时给它的数据成员赋初值，称为对象的初始化。
- ◆ **C++**程序中的对象初始化工作由一个特殊的成员函数来完成，就是构造函数。
- ◆ 因为不同类型的对象的初始化工作是不一样的，因此构造函数从属于某个类的，即每个类都要定义它自己的构造函数，它是类的成员函数。

8.3 构造函数

◆ 定义构造函数的一般形式为:

```
class 类名
{ public:
    类名（形参表）;           //构造函数的原型
    //类的其它成员
};
类名::类名（形参表）       //构造函数的实现
{
    //函数体
}
```

8.3 构造函数

构造函数的特点是：

- ◆ 构造函数的函数名与类名相同；而且没有返回值。
- ◆ 构造函数一定是公有函数。
- ◆ 作为类的成员函数，构造函数可以直接访问类的所有数据成员。
- ◆ 在类的内部定义的构造函数是内联函数。构造函数可以带默认形参值，也可以重载。

8.3 构造函数

- ◆ 在声明类是如果没有定义类的构造函数，编译系统就会在编译时自动生成一个默认形式的构造函数，
- ◆ 默认构造函数是构造对象时不提供参数的构造函数。
- ◆ 除了无参数构造函数是默认构造函数外，带有全部默认参数值的构造函数也是默认构造函数。

8.4 析构函数

8.4 析构函数

- ◆ 与构造函数相反，当对象生存期结束时，需要调用析构函数，释放对象所占的内存空间。
- ◆ 与构造函数一样，析构函数也是类的一个公有成员函数，它的名称是在类名前加“~”构成，没有返回值，和构造函数不同的是析构函数不接受任何参数。
- ◆ 析构函数是在对象生存期即将结束的时刻由系统自动调用的。如果程序员没有定义析构函数，系统将自动生成和调用一个默认析构函数。
- ◆ 类的析构函数不能重载。

8.5 拷贝构造函数

8.5 拷贝构造函数

- ◆ 拷贝构造函数就是函数的形参是类的对象的引用的构造函数。
- ◆ 通过等于号复制对象时，系统会自动调用拷贝构造函数。
- ◆ 如果程序在类定义时没有显式定义拷贝构造函数，系统也会自动生成一个默认的拷贝构造函数，把成员值一一复制。
- ◆ 拷贝构造函数与原来的构造函数实现了函数的重载。

浅拷贝与深拷贝

- ◆ 完成简单的一一对应的复制的拷贝构造函数称为浅拷贝。
- ◆ 如果类的数据成员包括指针变量，类的构造函数用new运算符为这个指针动态申请空间。如果通过浅拷贝进行了对象的复制。最后，在退出运行时，程序会报错。这时需要用“深拷贝”的方式恰当定义类的拷贝构造函数。
- ◆ 一般来讲，如果一个类需要析构函数来释放资源，则它也需要定义一个显式拷贝构造函数来实现深拷贝。

8.5 拷贝构造函数

以下三种情况系统会自动调用拷贝构造函数：

- ◆ 当用类的一个对象去初始化该类的另一个对象时
- ◆ 如果函数的形参是类的对象，调用函数时，进行形参和实参结合时
- ◆ 如果函数的返回值是类的对象，函数执行完成返回调用者时

8.6 类的静态成员

1、静态数据成员

- ◆ 静态数据成员是类的数据成员的一种特例，采用static关键字来声明。
- ◆ 类的数据成员在类的每一个对象中分别存储不同的数值，但是静态数据成员则不同，它在整个类中只有一个拷贝，由该类的所有对象共同维护和使用，从而实现了同一类的不同对象之间的数据共享。
- ◆ 静态数据成员具有静态生存期。
- ◆ 在类的声明中只能声明静态数据成员的存在。由于类的声明是抽象的，静态数据成员的初始化需要在类的外部进行，通过类名对它进行访问。

2、静态成员函数

- ◆ 可以通过定义和使用静态成员函数来访问静态数据成员。
- ◆ 所谓静态成员函数就是使用static关键字声明函数成员。同静态数据成员一样，静态成员函数也属整个类，由同一个类的所有对象共同维护，为这些对象所共享。

2、静态成员函数

- ◆ 静态成员函数作为成员函数，它的访问属性可以受到类的严格控制。对公有静态成员函数，可以通过类名或对象名来调用；而一般的非静态公有成员函数只能通过对象名来调用。
- ◆ 静态成员函数可以直接访问该类的静态数据成员和函数成员；而访问非静态数据成员，必须通过参数传递方式得到对象名，然后通过对象名来访问。

8.7 类成员的保护和使用

1、类的封装

- ◆ 在程序设计中，将数据与操作数据的行为进行有机地结合，这就是封装。
- ◆ C++语言提供类这种语言成分来实现封装，类是属性和操作的结合体，并且在定义类的属性和操作时，规定了它们的可见性。
- ◆ 通过封装将一部分成员作为类与外部的接口，而将其它的成员隐藏起来，以防外界的干扰和误操作，使程序的不同模块之间的相互影响减小到最低限度。

2、友元

- ◆ 在一个类中，可以利用关键字friend将别的模块（一般函数、其它类的成员函数或其它类）声明为本类的友元，这样类中本来隐藏的信息（私有和保护成员）就可以被友元访问。
- ◆ 友元提供了不同类或对象的成员函数之间、类的成员函数与一般函数之间进行数据共享的机制。
- ◆ 友元并不是类的成员。

2、友元

关于友元类，要注意：

- ◆ 友元关系是不能传递的。B类是A类的友元，C类是B类的友元，C类和A类之间，如果没有声明，就没有任何友元关系，不能进行数据共享。
- ◆ 友元关系是单向的。如果声明B类是A类的友元，B类的成员函数就可以访问A类的私有和保护数据，但A类的成员函数却不能访问B类的私有和保护数据。

3、常对象和常成员

- ◆ 定义对象时用const进行修饰，称为常对象，它的数据成员值在对象的整个生存期间内不能被改变。也就是说，常对象在定义时必须进行初始化，而且不能被更新。
- ◆ 使用常量，既实现了数据共享、又可以保证数据不会被改变。

3、常对象和常成员

- ◆ 对于类的某个成员函数，使用const关键字修饰，称为常成员函数。常对象只能调用常成员函数，保证常对象的数据成员不能被修改。
- ◆ 使用const说明的数据成员称为常数据成员。如果在一个类中说明了常数据成员，那么任何函数中都不能对该成员赋值。构造函数对该数据成员进行初始化，就只能通过初始化列表。

3、常对象和常成员

常成员函数的特点：

- ◆ `const`是函数类型的一个组成部分，在函数实现时也要带`const`关键字。
- ◆ 常成员函数不能更新对象的数据成员，也不能调用该类中的非常成员函数。
- ◆ 常对象只能调用常成员函数，但是常成员函数也可以被普通对象来调用。
- ◆ `const`关键字可以被用于参与对重载函数的区分。

8.8 类的组合

8.8 类的组合

- ◆ 类的组合（也称类的聚集），描述的就是一个类内嵌其它类的对象作为数据成员的情况，它们之间的关系是一种包含与被包含的关系。
- ◆ 在面向对象程序设计中，可以对复杂对象进行分解、抽象，把一个复杂对象分解为简单对象的组合。

8.8 类的组合

- ◆ 当创建组合类的对象时，各个内嵌对象也将被自动创建。因此，在创建组合类对象时既要对本类的基本数据成员进行初始化，又要对内嵌对象成员进行初始化。
- ◆ 在声明一个组合类的对象时，不仅它自身的构造函数将被调用，而且还将调用其内嵌对象的构造函数。

8.8 类的组合

◆ 组合类构造函数定义的一般形式为：

类名::类名(形参表):内嵌对象1(形参表), 内嵌对象2(形参表),...

{

//类的初始化

}

◆ 其中，“内嵌对象1(形参表), 内嵌对象2(形参表),...”为初始化列表，用来完成对内嵌对象的初始化。

8.8 类的组合

◆ 组合类的构造函数的执行顺序是：

- (1) （如果有多个内嵌对象）按照内嵌对象在组合类的声明中出现的次序，依次调用其内嵌对象的构造函数。（注意：并不是按照初始化列表中给出的顺序）。
- (2) 再执行本类的构造函数的函数体。

8.9 面向对象分析和设计

8.9 面向对象分析与设计

建立一个大型的软件系统可不是一件简单的事情，它是一项大工程，人们称它为软件工程，它研究如何建立大型软件系统：

- ◆ 可靠性；
- ◆ 成本效益好；
- ◆ 可理解性；
- ◆ 可维护性。

8.9 面向对象分析与设计

- ◆ 面向对象程序设计是面向对象思想在软件工程领域的全面应用，包括：面向对象分析（OOA），面向对象设计（OOD），面向对象编程（OOP），等。
- ◆ 面向对象分析阶段要把问题的范围定义清楚，分析系统需求，把实际中不重要的东西忽略，对所关心的问题建立一个模型。
- ◆ 设计是在分析的基础上进一步加工，OOD阶段是对OOA模型的修改和补充，采用一致的概念、原则和表示法，二者没有严格的阶段划分，但设计阶段侧重于考虑与实现有关的细节。
- ◆ 到了OOP阶段，OOD阶段设计的类被具体的程序设计语言的类所实现。

作业

◆ 第8章习题：15，16，20