# 面向对象程序设计讲义大纲

（2019 年 11 月 17 日学生学习发布版）

（适用于《新标准 C++程序设计》1—6 章）

注意：蓝色字体部分是暂时未详细讲述但需要提前了解的部分。

## 第 1 章　从 C 到 C++

**知识点：**

1、C++的历史、I / O 和头文件（教材 1.1—1.3）

2、引用（Reference）（教材 1.6）

3、函数重载（Overload）（教材 1.8）

4、函数的默认参数（Default Parameter）（教材 1.5）

5、new / delete 运算符（教材 1.9）

6、使用 string 对象处理字符串（教材 1.10）

7、函数模板（Function Template）（教材 9.1）

8、常量（Constant）

**重点：**

1、基本的数据类型、运算符与表达式、语句以及控制结构，C++与 C 基本兼容。

2、引用，引用相当于变量的别名，引用和被引用变量实际对应同一

个变量。引用作为函数形参是引用的重要用法，明确引用作为函数形参，则形参相当于实参的别名（一般情况下形参是实参的副本），因此修改引用形参会直接影响实参，如果要防止修改引用形参，可以使用 const T&这一用法。

3、C++对于函数原型（Prototype）的要求，要比 C 语言严格得多，只要源程序中函数放在调用者之后，就必须在调用者之前添加函数原型，称为函数原型声明或者简称声明（Declaration），函数定义又称为实现（Implementation）。

4、函数重载和函数默认参数，引入 C++“多态性”这一基本概念。

5、new / delete 运算符用于取代 C 语言中的 malloc() / free()函数进行动态内存分配 / 释放，还可以为数组动态分配 / 释放内存，即动态创建 / 删除数组。注意为数组动态分配 / 释放内存时，new 运算符的语法，以及 delete[]语法的使用。

6、初步了解 C++ STL，string 类是 C++ STL 中的一个类，可看作字符串类型，string 类的对象即 string 对象，用于在 C++中简单有效地处理字符串，相对 C 语言的字符串函数要方便得多。

7、函数模板（**提前讲述**），引入参数化类型或者泛型（Generics）的概念，注意模板形参、模板实参和模板的实例化概念，模板可以隐式实例化也可以显式实例化（明确指定模板实参），隐式实例化、显式实例化和显式具体化统称为具体化（Specialization）。函数模板是实现泛型算法的重要手段。

8、常量（**补充讲述**），C++中使用 const 关键字（限定符）定义的常

量是真正意义上的常量，相当于不可修改的变量，与变量一样有数据类型。必须注意区分 const T *与 T * const（T 表示数据类型，下同）。

## 第 2 章　类和对象初步

**知识点：**

1、C++类（Class）是 C 语言结构体（Structure）的扩展（教材 2.3）

2、成员变量和成员函数（教材 2.4—2.5）

3、访问控制权限（先学习 public / private，protected 留待后续讨论）（教材 2.6）

**重点：**

1、类是结构体的扩展，初学者可以先这样理解，C 语言结构体只能包含变量成员，而 C++类则可以包含成员变量和成员函数。

2、定义结构体类型后，还需要创建结构体变量才能使用，同样，定义类后，也需要创建对象才能使用，也就是说，对象是类的变量，或者说，类是对象数据类型。

3、一般面向对象理论中的定义：对象是一组特征（即属性或者成员变量）和一组针对特征的操作（即方法或者成员函数）组成的实体，类是对象的抽象。

4、C++中的结构体（使用 struct 关键字定义）和类（使用 class 关键字定义）基本是等价的，实际上都是类，都可以包含成员函数和成员变量，主要区别在于，结构体中的成员默认 public，类中的成员默认

private。

5、使用类（或者结构体）的语法，以及为指向对象（或者结构体变量）的指针动态分配／释放内存（同样使用 new／delete 运算符），C++都要比 C 语言简洁，因此 C++程序实现链表也比 C 语言程序要简洁一些，注意二者的差异。

## 第 3 章　类和对象进阶

**知识点：**

1、构造函数和析构函数初步（教材 3.1—3.2）

2、复制（拷贝）构造函数（教材 3.1）

3、构造函数和析构函数的执行时机（教材 3.1—3.3）

4、静态成员（教材 3.4）

5、const 成员函数（教材 3.5）

6、类的复合与对象的包容（**即成员对象和封闭类**）（教材 3.6）

7、友元（教材 3.8）

8、this 指针初步（教材 3.9）

**重点：**

1、构造函数（Constructor，可缩写为 ctor）在创建对象时自动执行，析构函数（Destructor，可缩写为 dtor）在对象删除（释放或者失效）时自动执行。全局对象的构造函数甚至先于 main 函数执行，全局对象的析构函数在 main 函数返回后执行（不同 C++编译器可能存在差

别）。构造函数可以重载，除了默认构造函数之外，还可以有带参数的构造函数。

2、带参构造函数与默认构造函数是重载关系，带参构造函数用于在创建对象时以特定参数调用构造函数，实现对象的特殊初始化。只有一个参数的带参构造函数一般称为类型转换构造函数。

3、复制（拷贝）构造函数（Copy constructor，可缩写为 cctor）是参数为 const T&的构造函数，在对象复制（拷贝）时自动执行。

4、作为函数局部变量的对象（包括 main 函数中的局部变量对象），一般按照定义顺序创建，依次自动执行构造函数，函数返回时按照与创建时顺序相反的顺序删除（释放或者失效），依次自动执行析构函数。

5、函数形参如果是对象，则调用函数时，形参对象是实参对象的副本，会创建形参对象，自动执行复制（拷贝）构造函数，函数返回时形参对象自动失效（删除），自动执行析构函数。函数形参如果是对象引用，则调用时不会执行复制构造函数，因为此时形参只是实参的别名。

6、函数返回值如果是对象，则函数返回时，会为返回的对象创建一个临时返回值对象副本，自动执行复制（拷贝）构造函数，如果函数返回值对象还要赋值给其它对象，还会执行相应的重载赋值运算符，临时函数返回值对象副本使用完成（例如赋值完成后）后即自动删除，自动执行析构函数。

7、从以上两点可以看出，函数的形参和返回值最好都不要是对象，

否则容易造成较大开销，使用对象引用较好，必要时可以使用常量对象引用。

8、函数的返回值如果是对象引用，注意不应该将局部变量对象直接返回，因为局部变量对象在函数返回时自动失效（删除），其引用无意义。可在局部变量对象定义时加 static 关键字，使之成为静态局部变量对象，再返回。

9、静态局部变量对象在函数第一次调用时创建，自动执行构造函数，直到程序结束时删除，自动执行析构函数。

10、C++中，构造函数、析构函数和重载赋值运算符的执行时机相当复杂，应该对照课堂教学中的实例，以及实验实例详细学习。

11、静态成员包括静态成员变量和静态成员函数，均使用 static 关键字。静态成员变量是类的所有对象共享的成员变量，只访问静态成员变量的函数可以定义为静态成员函数，实际上，静态成员变量和静态成员函数的意义是属于类本身的成员，而不是属于类的某一个具体对象的成员。

12、const 成员函数（常成员函数）加有 const 关键字，常量对象或者常量对象引用只能调用常成员函数，非常量对象或者非常量对象引用也可以调用常成员函数。重载运算符成员函数也可以加 const 关键字，使得常量对象或者常量对象引用可以使用重载运算符。

13、类的复合关系也称为对象（类）的包容关系，即在一个对象（类）中，以另一个类的对象作为成员变量，二者可分别称为包容器（Container）对象或者宿主（Host）对象，以及对象成员（或者子对

象、被包容对象），教材上称为**成员对象和封闭类**。包容关系表示一种整体—部分关系，较严格地说这是一种对象之间的关系。包容关系不同于类的继承，继承是一种类与类之间的关联，而类的复合（对象的包容）则是一种对象与对象之间的关联，或者说对象的包容是实体之间的关联。举例：不宜说一个具体的“灵长目人科动物”实体内部包含有一个“灵长目动物”实体，即二者不属于类的复合（对象的包容）关联，而“一条线段含有两个端点”则属于对象的包容关联。

14、包容器对象构造函数如果需要调用对象成员非默认构造函数，包括带参构造函数和复制（拷贝）构造函数，则需要使用专门语法，即在包容器对象构造函数实现时，在参数列表后加“:”的语法，即构造函数初始化列表语法。

15、包容器对象与对象成员，其构造函数、析构函数和重载赋值运算符的执行时机更为复杂，必须对照课堂教学中的实例，以及实验实例认真学习。

16、初步掌握友元关系和 friend 关键字的使用，重点是友元函数，并了解友元类和友元成员函数。友元容易破坏模块的内聚性，增大模块之间的耦合度，因此非必要时最好不要使用，但应该掌握。

17、初步了解 this 指针的使用，this 指针在类的成员函数中使用，表示指向当前对象的指针，*this 可以表示当前对象。在类的成员函数中需要访问成员变量时，在成员变量前加上“this->”有时能让程序更清晰。如果成员函数的形式参数或者局部变量与成员变量同名，访问成员变量时必须加上“this->”。

# 第 4 章　运算符重载

**知识点：**

1、运算符重载初步（教材 4.1）

2、重载赋值运算符（教材 4.2）

3、浅拷贝和深拷贝（教材 4.3）

4、运算符重载为友元函数（教材 4.4）

**重点：**

1、运算符重载需要重点掌握作为类成员函数形式的一般双目（二元）运算符重载，例如“=”、“+”、“[]”等。

2、以类成员函数形式重载双目（二元）运算符，一般运算符的第一操作数对应当前对象，运算符的第二操作数对应成员函数的参数，运算符表达式的返回值对应成员函数的返回值。

3、重载运算符如果返回引用，可以作为左值或者右值（可简单解释为左值可被赋值），重载运算符如果返回常量引用，则只能作为右值，一个典型的例子是访问数组元素的“[]”运算符。

4、重载赋值运算符成员函数的原型一般是 const T& T::operator = (const T&)，返回值类型是 const T&而不是 void，表示赋值表达式自身也有值，这对于 o3=o2=o1 形式的连续赋值是必需的。复合赋值运算符，例如“+=”、“-=”等，重载成员函数原型与之相似，赋值表达式自身有值这一点是需要特别注意的。

5、对象的复制（拷贝）与赋值是两个不同的概念，对象的复制是创建一个现有对象的复制品（副本）对象，复制对象之前复制品对象并不存在；对象的赋值则是对已经存在的对象赋新值，隐含有“对象原有值废除”的含义。

6、C++中，以下形式的代码均表示对象（或者变量）的复制（也称为复制初始化）：

```
T o2(o1);
T o2=o1;
T o2=T(o1);
```

而以下形式的代码则表示对象的赋值：

```
T o2;
o2=o1;
```

如果是以下形式的代码，则表示先复制创建一个临时对象，再将临时对象赋值给对象，最后删除临时对象：

```
T o2;
o2=T(o1);
```

7、对象的复制与赋值，默认都是逐个复制或者赋值对象的成员变量，教材上称为**浅拷贝**，但有时这种默认复制或者默认赋值不能满足要求，例如对象有分配了内存的指针成员变量，则仅仅复制或者赋值了指针本身，内存并未重新分配，导致两个对象的指针成员变量指向同一分配内存，这样有时是不允许的，因此需要特定的复制（拷贝）构造函数以及重载赋值运算符，教材上称为**深拷贝**。

8、初步掌握使用非成员函数形式重载运算符的基本方法，非成员函数形式重载双目（二元）运算符，运算符的第一操作数和第二操作数分别对应函数的两个参数，运算符表达式的返回值仍对应函数的返回值。让非成员函数成为类的友元函数，可使得非成员函数访问类的非public 成员，带来方便，因此非成员函数形式重载运算符的情况下，函数多作为类的友元函数。

## 第 5 章　继承与派生

**知识点：**

1、继承与派生初步（教材 5.1）

2、类的复合（对象的包容）关系和继承关系（教材 5.2）

3、protected 访问控制权限（教材 5.3）

4、派生类对象的构造与析构（教材 5.4）

5、赋值兼容规则（教材 5.7—5.8）

6、public 继承、protected 继承与 private 继承（教材 5.9）

**重点：**

1、继承（Inherit，Inheritance）是面向对象程序设计重要特性之一，继承是类与类的一种关联，派生类（Derived Class）继承自基类（Base Class），或者说基类是派生类的泛化。

2、继承是一种类与类之间的关联，或者说继承是概念之间的关联，举例（以电影《猩球崛起》为背景）：可以说“灵长目人科动物”概

念继承自“灵长目动物”概念，一个具体的“灵长目人科动物”实体具有“灵长目动物”实体的所有特征和操作，这是继承关联的一种表现，但不宜说一个具体的“灵长目人科动物”实体内部包含有一个“灵长目动物”实体。

3、再次强调：类的复合关系、对象的包容关系或者教材上的**成员对象和封闭类**（实际上是同一概念的不同说法）表示一种整体—部分关系，较严格地说这是一种对象之间的关系，或者说是对象与对象之间的关联（实体之间的关联），继承则是一种类与类之间的关联。再次举例：不宜说一个具体的“灵长目人科动物”实体内部包含有一个“灵长目动物”实体，即二者不属于类的复合（对象的包容）关联，而“一条线段含有两个端点”则属于对象的包容关联。

4、一般来说，派生类继承了基类所有的属性和方法，即成员变量和成员函数，派生类对象也有基类对象的所有成员变量和成员函数，但基类 private 成员不能被派生类添加的成员函数访问，也就是说派生类内部并不能访问基类的 private 成员。

5、如果名称和参数表都相同（以下简称同名）的成员函数，同时在基类和派生类中定义和实现，则二者在派生类中都是同时存在的，派生类内部如果需要调用基类的同名成员函数可以在成员函数名前面加上基类名和“::”运算符。

6、protected 访问控制权限用于类的继承，protected 成员可在类内部，以及本类的派生类内部访问，但仍然不能通过类的对象在类外部访问。

7、一般来说，创建派生类对象时，先执行基类的构造函数，再执行

派生类的构造函数；删除（释放）派生类对象时，先执行派生类的析构函数，再执行基类的析构函数。

8、派生类构造函数如果需要调用基类非默认构造函数，包括带参构造函数，则与类的复合（对象的包容）相似，也需要使用构造函数初始化列表语法，即在派生类构造函数实现时，在参数列表后加“:”的语法，但要注意“:”之后使用基类名而不是成员变量对象名，因为继承是类与类之间的关联。

9、public 继承（公有派生）的情况下，派生类对象可以直接看作基类对象，基类对象的引用可以直接引用派生类对象，指向基类对象的指针可以直接指向派生类对象。

10、private 继承与 protected 继承（即**私有派生和保护派生**）参见教材 5.9。

## 第 6 章　多态与虚函数

**知识点：**

1、多态性、重写（Override）与虚函数（教材 6.1—6.4）

2、虚析构函数（教材 6.5）

3、纯虚函数和抽象类（教材 6.6）

**重点：**

1、C++实现多态性的两种重要方法分别是重载（Overload）和重写（Override），实际上“Overload”的原意是“超载”，而“Override”

有“覆盖”之意。实现重写的核心是虚函数（Virtual Function），也称为可重写函数（Overridable Function）或者可重写方法（Overridable Method）。

2、如果基类中的虚函数在派生类中被重写（Override），则默认情况下，只要当前对象是派生类的，必然调用派生类中重写的虚函数，相当于基类中的虚函数被覆盖（Override）了。

3、指向基类对象的指针可以直接指向派生类对象，但如果基类和派生类中有同名成员函数，只会调用基类的成员函数；如果同名成员函数是虚函数，则调用派生类的成员函数（虚函数根据实际对象调用）。

4、虚函数只需要在最初的基类中加 virtual 关键字，无论继承多少层次仍然是虚函数，当然在派生类中给虚函数也加上 virtual 关键字能使得程序更清晰。

5、即使成员函数是虚函数，派生类内部如果需要调用基类中虚函数的实现，仍然可以在成员函数名前面明确加上基类名和“::”运算符，换而言之，虚函数在类内部并未完全被覆盖（Override）消失。

6、构造函数不能是虚函数，但析构函数可以是虚函数。如果用 new 关键字创建派生类对象，并用指向基类对象的指针去指向派生类对象，则基类的析构函数应该是虚析构函数，否则用 delete 关键字删除派生类对象时无法正确调用派生类的析构函数。

7、纯虚函数是只有定义没有实现的虚函数，又称为抽象方法（Abstract Method），起“占位”作用，有纯虚函数的类称为抽象类，抽象类不能创建对象，但可以作为基类，也可以定义指向抽象类对象的指针，

用于指向抽象类派生类的对象。只有纯虚函数成员的抽象类，某些资料也称为接口（Interface）。

8、面向对象程序设计（OOP）的基本特性是封装、继承和多态。常见面向对象编程语言对 OOP 的支持能力各有不同，简介如下：

●Java：支持封装、继承和多态；一般只支持单线继承；多重继承仅在从接口继承时支持，称为实现（Implement）多个接口；不支持运算符重载；有限支持泛型（模板），需要 JDK 1.5 以上版本。

●C#：与 Java 相似，但有限支持运算符重载；泛型（模板）的支持需要 C# 2.0 以上版本。

●Visual Basic 5.0—6.0（VB5、VB6）：支持封装；有限支持继承和多态，只能继承接口，通过实现（Implement）接口支持继承和多态，支持实现多个接口，即有限的多重继承。

●Visual Basic .NET（VB.NET，VB2002 以上版本）：与 C#相似，支持封装、继承和多态，但不支持运算符重载；泛型（模板）的支持需要 VB 2005 以上版本。