C++语言程序设计

第六章：多态性与虚函数

宋霜

哈尔滨工业大学（深圳）

机电工程与自动化学院

邮箱：songshuang@hit.edu.cn

第六章：多态性与虚函数

- 多态性
- 虚函数

第六章：多态性与虚函数

多态性

- 向不同的对象发送同一个消息，不同的对象在接收时会产生不同的行为。
- C++中的表现形式：具有不同功能的函数可以用同一个函数名，实现调用不用内容的函数。

第六章：多态性与虚函数

多态性

- 向不同的对象发送同一个消息，不同的对象在接收时会产生不同的行为。
- C++中的表现形式：具有不同功能的函数可以用同一个函数名，实现调用不用内容的函数。
- 静态多态性

第六章：多态性与虚函数

多态性

- 向不同的对象发送同一个消息，不同的对象在接收时会产生不同的行为。
- C++中的表现形式：具有不同功能的函数可以用同一个函数名，实现调用不用内容的函数。
- 静态多态性
  - 函数重载，编译时决定

第六章：多态性与虚函数

多态性

- 向不同的对象发送同一个消息，不同的对象在接收时会产生不同的行为。
- C++中的表现形式：具有不同功能的函数可以用同一个函数名，实现调用不用内容的函数。
- 静态多态性
  - 函数重载，编译时决定
- 动态多态性
  - 运行时决定，通过虚函数实现

第六章：多态性与虚函数

虚函数

- 允许派生类中重新定义与基类同名的函数
- 通过基类的指针或引用访问基类和派生类中的同名函数

第六章：多态性与虚函数

虚函数

- 允许派生类中重新定义与基类同名的函数
- 通过基类的指针或引用访问基类和派生类中的同名函数
- 基类中通过virtual修饰
- 派生类中重新定义函数体

第六章：多态性与虚函数

虚函数

- 允许派生类中重新定义与基类同名的函数
- 通过基类的指针或引用访问基类和派生类中的同名函数
- 基类中通过virtual修饰
- 派生类中重新定义函数体
- 静态关联
- 动态关联

第六章：多态性与虚函数

纯虚函数

- virtual 函数类型 函数名（参数表）=0；

抽象类

- 定义了纯虚函数的类
- 目的时用来作为基类去建立派生类
- 为一个类族提供一个公共接口

类的使用小结

- 构造函数
- 继承与派生
- 数据存储

默认构造函数

- 如果类中定义了成员变量，没有提供其他构造函数，你需要定义一个默认构造函数（没有参数）。
- 默认构造函数更适合于初始化对象，使对象内部状态一致、有效
- 如果没有提供其他构造函数，又没有定义默认构造函数，编译器将为你自动生成一个，编译器生成的构造函数并不会对对象进行初始化。

拷贝构造函数

- 为需要动态分配内存的类声明一个拷贝构造函数和一个赋值操作符
- 尽量使用初始化而不要在构造函数里赋值
- 初始化列表中成员列出的顺序和它们在类中声明的顺序相同
- 让operator=返回*this 的引用
- 在operator=中对所有数据成员赋值
- 在operator=中检查给自己赋值的情况

拷贝构造函数

- 仅在代码中需要拷贝一个类对象的时候使用拷贝构造函数；
- 不需要拷贝时应使用DISALLOW_COPY_AND_ASSIGN。

```
// 禁止使用拷贝构造函数和赋值操作的宏，应在类的private:中使用
#define DISALLOW_COPY_AND_ASSIGN(TypeName) \
TypeName(const TypeName&); void operator=(const TypeName&)
class Foo
{
public:
    Foo(int f);
    ~Foo();
private:
    DISALLOW_COPY_AND_ASSIGN(Foo);
};
```

存取操作

- 将数据成员私有化，并提供相关存取函数
- 存取函数的定义一般内联在头文件中

```
class Foo
{
public:
    Foo(int f);
    ~Foo();
    int GetNumber( ){return number};
    void SetNumber(int a){number=a;};
private:
    int number;
};
```

类

声明次序

- 定义次序如下：public:、protected:、private:
- 每一块中声明次序一般如下：
  - 1) typedefs 和enums；
  - 2) 常量；
  - 3) 构造函数；
  - 4) 析构函数；
  - 5) 成员函数，含静态成员函数；
  - 6) 数据成员，含静态数据成员。
  - 宏DISALLOW_COPY_AND_ASSIGN 置于private:块之后，作为类的最后部分。
  - .cc 文件中函数的定义应尽可能和声明次序一致。

类

存取操作

- 避免public 接口出现数据成员
- 尽可能使用const
- 尽量用“传引用”而不用“传值”
- 必须返回一个对象时不要试图返回一个引用

成员函数

- 争取使类的接口完整并且最小。
- 分清成员函数，非成员函数和友元函数
  - 虚函数必须是成员函数。
  - operator>>和operator<<决不能是成员函数
  - 只有非成员函数对最左边的参数进行类型转换
  - 其它情况下都声明为成员函数

类

编写短小的函数

- 倾向于选择短小、凝练的函数。
- 如果函数超过40 行，可以考虑在不影响程序结构的情况下将其分割一下。
- 即使一个长函数现在工作的非常好，一旦有人对其修改，有可能出现新的问题，甚至导致难以发现的bugs。
- 使函数尽量短小、简单，便于他人阅读和修改代码。
- 在处理代码时，考虑将其分割为更加短小、易于管理的若干函数。

类

继承 Inheritance

- 使用组合（composition）通常比使用继承更适宜
- 如果使用继承的话，只使用公共继承。
- 继承用于两种场合
  - 实现继承（implementation inheritance），子类继承父类的实现代码
  - 接口继承（interface inheritance），子类仅继承父类的方法名称
- is-a：使用继承
- has-a：使用组合

类

继承 Inheritance

- 使公有继承体现 "是一个" 的含义
- 通过分层来体现 "有一个" 或 "用...来实现"
- 区分接口继承和实现继承
  - 定义纯虚函数的目的在于，使派生类仅仅只是继承函数的接口。
  - 声明简单虚函数的目的在于，使派生类继承函数的接口和缺省实现。
  - 声明非虚函数的目的在于，使派生类继承函数的接口和强制性实现。
- 决不要重新定义继承而来的非虚函数