# C++面向对象编程思想探讨

蓝雯飞
(华中科技大学计算机学院,武汉 430074)
(中南民族大学计算机学院,武汉 430073)
E-mail:lanwenfei1@yahoo.com.cn

**摘 要** 论述了面向对象语言与面向对象编程思想的关系,详细讨论了作为面向对象程序设计三个基本原则封装、继承和多态性的思想精髓及特点,并用C++语言作为编程工具阐述了如何在面向对象程序设计中很好地运用它们。

**关键词** 面向对象 封装 继承 多态性

**文章编号** 1002-8331-(2004)22-0104-03 **文献标识码** A **中图分类号** TP311

# A Study on Thinking in C++ Object-oriented Programming

**Lan Wenfei**
(College of Computer Science, Huazhong University of Science and Technology, Wuhan 430074)
(College of Computer Science, South-Central University for Nationalities, Wuhan 430073)

**Abstract**: This paper discusses the relation between Object-oriented programming language and thinking in Object-oriented programming at first, then expatiates three rules, which are encapsulation, inheritance and polymorphism, involved in Object-oriented programming and their characteristic, and presents how to use them commendably in C++ Object-oriented programming.

**Keywords**: Object-Oriented, encapsulation, inheritance, polymorphism

## 1 引言

面向对象的方法和技术来源于面向对象语言[1],和传统的软件开发方法相比,具有自然性、简单性、高效性和费用低等多方面的特点,因此成为目前软件开发的主流方法。

面向对象方法包括面向对象分析、面向对象设计、面向对象编程和面向对象测试。面向对象编程也称面向对象程序设计,是将对象的数据及对数据的操作封装在一起,成为一个不可分割的整体,同时将具有相同特征的对象抽象为一种新的数据类型—类,通过对象间的消息传递使对象状态发生变化,最终完成计算的一种新颖的程序设计方法。这种方法符合人们的思维方式和表达方式,实现了从问题域到计算机域的直接映射,使程序设计从过分专业化的方法、规则和技巧中回到客观世界,回到人们通常的思维方式。

面向对象程序设计和传统的面向过程的结构化程序设计方法相比,能更好地适应在重用性、可扩展性、复杂性、可靠性和规模、质量、效率上对程序设计的种种需求,而被广泛推用,其方法本身也在这诸多的实践和磨练中日趋成熟、标准化和体系化,逐渐成为目前公认的主流程序设计方法[2]。

## 2 面向对象编程思想的重要性

开发程序离不开编程语言及编程方法即编程思想。流行的面向对象编程语言有C++、Java、C#等。编程思想中,又有面向对象和面向过程之分,它们既是世界观又是方法论。在软件开发的不断实践中,前者的优越性已经得到不断的体现和证实[3]。

掌握面向对象的编程思想如同获得练气的真谛,它的重要性往往胜过了对编程语言的选择。有人即使选择了很好的面向对象语言,也无法成为真正的编程高手。因为他看重的是语言的好坏,忽略的是对编程思想的领悟及运用。

实际上,一个系统或语言是不是面向对象的并不重要,重要的是怎样才能是面向对象的以及用什么办法实现相关的好处[4]。

编程高手的成长需要一个过程。初学编程需要选择好的语言,这样可以取得事半功倍的效果,同时激发兴趣,增强信心。一旦熟悉了一种语言之后,应以此为契机进而掌握面向对象编程思想。这时你熟悉的不再是语言本身的语法、函数、类库,而是封装、继承、绑定、多态和模式等思维方法,然后触类旁通,再学其他面向对象语言也不难。最终成功的真正的编程高手,是不受编程语言限制的。他们可能比较熟悉一种开发工具,但那也只是承载他们大道无形之气的器具。他们的思想、方法、模式甚至哲学,既超越了编程语言,又可以指导编程语言的实践。

以非面向对象的方法去使用面向对象编程工具是一个错误;同样使用面向对象编程工具,如果没有面向对象的编程思想,好比"不察其辞",最终仍然是"近而不可见",难以开发出优秀的系统。唯有潜心苦练,悉心总结,掌握面向对象编程思想的精髓,才能运用自如,最终达到"远而可知"的境界[3]。

在面向对象程序设计中,有三个基本的原则思想,它们是封装、继承和多态性编程思想,论文将着重进行讨论。

作者简介:蓝雯飞,女,华中科技大学博士研究生,中南民族大学副教授,研究方向:软件测试、面向对象程序设计方法与技术。

## 3 对象与封装

用面向对象程序设计方法求解现实世界的问题，是将问题域分割成具有相互关联的多个对象，这些对象是现实世界中的可以被感知或触摸的事物在面向对象程序空间中的直接映射。对象可以很简单，也可以很复杂，复杂的对象是由若干个简单的对象构成。对象具有两个共同的特点，一是都有自己的属性特征，二是都有自己的行为特征，例如，一个圆有半径、圆心坐标位置等属性，还有可施加在其上的放大、缩小、移动、绘制等行为。

然而，在进行面向对象程序设计时，一一去描述同类对象既不可能也是极大的工作重复，例如，为了设计一个学生成绩管理系统，在设计程序系统时不可能一一描述每个学生对象。为了克服类似的复杂性，在程序设计时，采用了人们在处理复杂问题时常用的一种方法—抽象，它是将做什么和怎么做分开，从而隐藏了问题的复杂性。

在面向对象程序设计中，抽象是最基本的原则之一，包括数据抽象和行为抽象。数据抽象是抽象出某类对象的公共属性，行为抽象是抽象出某类对象的公共行为。这样，用抽象的方法抓住了编程者所关心的重要信息，而忽略掉一些不重要的细节部分，从而找到了一类对象的抽象数据类型。然后，用面向对象编程工具的类机制把一类对象共有属性的数据结构和操作该数据结构的行为捆绑在一起，封装在一个程序实体内定义成一种类类型，这一过程就是数据封装，简称封装。单从这点上来考虑，作者认为面向对象程序设计也可称类程序设计。

类的使用者只需要知道怎么使用接口，而不需要知道操作是怎么实现的，因此，封装是对抽象的实现，封装的结果最终隐藏了问题的复杂性，并提供了代码的重用性，因为类是允许重复生成同类对象的可重用的程序代码，而且还可以被继承，从而减轻了开发一个软件系统的难度和周期。

下面是对平面图形系统中的矩形对象抽象封装后得到的C++类[5]。

```
class Rectangle
{ public:
    Rectangle(int x,int y,int w,int h){ X=x;Y=y;W=w;H=h;}
    void Move(int,int);//移动矩形的行为
    void Size(int,int);//修改矩形大小的行为
    void Where(int&,int&);//查询矩形大小的行为
  private:
    int X,Y,W,H;//四个属性构成的矩形对象的数据结构
};
//每个行为的实现细节
void Rectangle::Move(int x,int y){ X=x,Y=y;}
void Rectangle::Size(int w,int h){ W=w,H=h;}
void Rectangle::Where(int& x,int& y){ x=X,y=Y;}
```

当有了对矩形对象抽象封装而得到的类 Rectangle 后，在程序中可根据需要生成任意个矩形对象，如下的 C++代码段可生成一千个矩形对象，它们具有相同的属性结构和行为特征，但属性的当前值可以不同。

```
for(int i=1;i<=1000;i++)
{ cin>>x>>y>>w>>h;
  r[i-1]=new Rectangle(x,y,w,h);
}
```

哪怕是生成一万个矩形对象，仍然用上述代码段，只需要将循环次数由 1000 改成 10000 即可。由此可见，对象是由样板—类来产生的，在程序运行时，不同的对象分配不同的存储，因而能区分不同的对象。

从上面的例子中看出，编程工具提供的类机制把对象的数据结构和使用数据结构的操作封装在一起，并且通常将数据结构部分声明为 private 访问属性而使其不能被外界直接访问，或者说对外界隐藏，而只能通过在这个数据结构上定义的 public 访问属性的操作接口间接地访问数据结构，使数据结构被隐藏在操作接口背后，操作的实现细节也一样被隐藏在操作接口的背后 达到了封装数据结构的目的，这种编程思想使得类的使用者只需知道操作的接口，而没有必要了解数据结构及在数据结构上的操作的实现细节，大大减轻了使用者的负担。另外，当类的数据结构或操作的实现被修改时，只要操作的接口不变，使用类的程序的其余部分则不必修改，有利于程序的维护。

一些优秀的面向对象语言本身也使用抽象封装方法，例如，在 Java 语言中将作为图形界面元素的菜单、按钮、对话框、复选框、单选按钮组等对象都抽象封装成一些基本组件类或容器类，设计 Java 程序图形用户界面，就是使用标准组件类和容器类的过程，从而大大降低了开发程序图形用户界面的难度和周期。实践证明，用 Java 语言开发一个图形用户界面的程序量比用 C 语言少得多，原因很简单，C 语言不是面向对象语言，因此它没有对象的概念，更不具有数据抽象、数据封装的思想。

## 4 类与继承

继承也是面向对象程序设计的一个重要思想，对继承的更多的使用场合包含在多态性中，所以这节只简要介绍继承思想，多态性在下节讨论。

现实生活中到处都有继承的例子，如每个人都从自己父母身上继承了人种、外貌、性格和举止行为等特征，还会有不同的特征如年龄、姓名及学历等。类似地，在面向对象程序设计中，继承表达的是一种类与类之间的被继承与继承的关系，是类与类之间的一种抽象与具体的关系，是对象与对象之间的一般与特殊的关系。这种关系使得一个类可以继承另一个类的属性和操作，还可以有新的属性和操作，这就为面向对象程序设计的代码重用提供了很好的方法支持。在继承关系中，被继承的类称基类或父类，由继承方式得到的类称派生类或子类，子类还可以被继承得到子类，如此下去，使得类与类间的继承关系形成多层次结构，从而能很好地刻划现实世界中存在的多层次关系，如一个大学下面有各级学院，每个学院包括若干个专业系，一个系又是由几个教研室组成，每个教研室有若干个教师。在类层次结构中，上层类所具有的属性和方法全部被下层类自动继承。因此，越在上层的类越具有普遍性和共性，越在下层的类越细化、具体化和专门化[6]。

总的来说，继承具有如下特点：

(1)继承能方便地将现实世界中对象的一般与特殊的关系模型化成类层次结构。例如，由于正方形是矩形的特例。所以正方形类可从矩形类继承快速得到，下面是正方形类 Square 的C++定义：

```
class Square:public Rectangle
{ public:
    Square(int x,int w,int h):Rectangle(x,x,w,h){}
};
```

由于正方形的操作及属性和矩形相同，所以正方形不需要

再定义新的操作和属性，只是初始化属性时，将长和宽的值置为相同即可。

(2)通过继承提高了代码的可重用性。这里的代码重用是指子类自动拥有父类的属性和方法。

(3)通过继承可以减少程序的冗余信息。因为利用继承使得父类中的属性和方法不必在派生类乃至间接派生类中重复定义。

(4)使用继承使得扩充程序时不必修改已有的代码，从而极大地减少了软件的维护工作量。

从上面的讨论可看出，面向对象程序设计的继承思想充分体现了抽象与具体、一般与特殊关系的自然现象。

## 5 公有继承与多态性

在面向对象程序设计中，多态性是指不同但相似的对象收到同一消息将导致完全不同的结果[7]。它的主要内容是指子类型和公有继承[7]。例如，整数类型中的子集构成了整数类的一个子类型，整数类型的有关操作如加、减、乘、除、取余、比较等都可用于整数子集类型的对象上。从这个例子中可看出，每一个子类型的对象可以被用在高一级的类型中，即高一级类型中的所有操作可用于下一级类型的对象。

当类 D 公有直接或间接继承类 B 时，就称类 D 是类 B 的子类型。根据类型兼容规则，类 D 的任何对象 Od 就可以出现在基类 B 的对象 Ob 可出现的地方。这就意味着一个公共的消息集既可以被送到类 B 的对象，也可以被送到类 D 的对象上，即用一组相同的方法和逻辑来使用这个类层次中不同类中的同名虚函数版本，或者说使用一段代码来处理一组相关的对象，以达到代码重用的目的，这就是多态性的实质。

举一个例子，希望求平面系统中圆的面积和矩形的面积[8]，为了能统一处理圆对象和矩形对象，需要为圆类和矩形类抽象一个形状类，该形状类不代表具体的平面图形，是为了使用多态性而引入的高层类。下面是对该问题的 C++代码描述：

```
class shape//形状类
{ public:
    virtual double area(){ return 0.0 ;}
};
class circle :public shape//圆类公有继承形状类
{ public:
    virtual double area(){ return 3.14*radius*radius ;}
};
class rectangle :public shape//矩形类公有继承形状类
{ public:
    virtual double area(){ return w*h ;}
};
```

对上述三个类构成的简单的类层次结构，可以用如下的代码统一求圆的面积和矩形面积，类似这样的代码被称为多态程序段。

```
void showarea(const shape &r){ cout<<r.area()<<endl ;}//多态程序段
```

并且，可按下面的方式使用多态程序段：

```
void main()
{ shape s1 ;circle c1(1.0) ;rectangle r1(4,6) ;
  showarea(s1) ;showarea(c1) ;showarea(r1) ;
}
```

上述 3 种使用方式都是正确的。

程序运行结果为：

0.0

3.14

24

这组数据正好是满足要求的。现在来分析一下多态程序段的执行，当引用形参 $r$ 是对圆对象的引用时，根据程序执行时的动态绑定原则，此时将调用求圆面积的虚函数版本，从而求得圆的面积，同理当 $r$ 引用矩形对象时，能求得矩形的面积。

可见，多态性为统一地处理(如 showarea()函数)一组接口相同但实现不同的操作(如 area())提供了极好的方法支持，是一种形而上学的方法，它使程序逻辑简单明了、可读性强。

此外，多态性还能使程序扩充特别容易，例如，如果我们还希望求三角形的面积和正方形的面积，只需要从 shape 类公有派生三角形类 triangle，从矩形类公有正方形类 square，因为正方形是矩形的特化。而多态程序段保持不变，照样可求得三角形和正方形的面积。

在使用多态性时关键要理解好最顶层类的设计和作用。它通过继承为下层的类提供了代码的共享，因而减少了重复性开发、提高了软件的开发效率。另一方面，通过多态性统一了处理类层次结构中同名的虚函数，简化了处理这些同名虚函数的处理逻辑，同样也提高了软件的开发效率。正是因为这一点，即为了能统一地处理一个类层次结构中同名的虚函数，在 C++语言中，增加了如下两种语法机制，使程序员可以最大程度地使用多态性。

(1)空的虚函数

在很多应用中，类层次的顶层类并不具备下层类的一些功能。可在基类中设计一个什么也不做、空的虚函数。这样，就可通过顶层类提供统一处理该类层次的方法。下面是使用空的虚函数定义形状类 shape 的更好形式：

```
class shape//形状类
{ public:
    virtual double area(){} //空的虚函数
};
```

一般地，通过抽象，在顶层的基类中，总有很多这种空的虚函数，它为其子孙类用虚函数机制实现多态性提供了一个统一的界面。这是在多态性程序设计时经常用到的方法。

(2)纯的虚函数

在许多情况下，在基类中不能为虚函数给出一个有意义的实现，这时可以将它声明为纯虚函数。它的实现留给派生类来做。下面是使用纯虚函数定义形状类 shape 的最好形式：

```
class shape//形状类
{ public:
    virtual double area()=0 ;//纯虚函数
};
```

含纯虚函数的类在 C++中叫抽象类，由于抽象类是对事物的高度抽象，所以这样的类不能产生对象，尽管抽象类是用继承语法来表达的，但它的主要目的不是为代码共享而设计的，而是为了使用多态性而设计的，它是另一个维度的抽象[9]。

在一些面向对象语言中都有抽象类的概念，只不过表示的语法不同而已，如在 Java 语言中用关键字 abstract 定义的类才是抽象类，以示和普通类的区别。

由此可见，面向对象编程语言与面向对象编程思想是相辅

(下转 140 页)

FSM_EVENT_NULL 结束。在下面的转移列表中，当 State0 收到事件 Event1 时，将在该列表中搜索该状态下该事件相应的动作 Action1，转移到 State1，依此类推直到收到FSM_EVENT_NULL事件，结束本次状态转移。

| 当前状态 | 收到的事件 | 引发的动作函数 | 下一状态 |
|---|---|---|---|
| State0 | Event1 | Action1 | State1 |
| State1 | Event2 | Action2 | State2 |
| … | … | … | … |
| StateN | FSM_EVENT_NULL | AC_NULL | ST_IDLE |

### 3.3.2 状态转移图

远程教育中蓝牙设备间，连接建立完成后，以一个课件文件的传输为例，图 5 展示了该文件传输过程的状态转移情况[2]。数据帧的连续发送是利用有限状态机、调度机机制实现。每发送一个数据帧包，向 RFCOMM 接口队列发送空信息，调度机根据此信息调用相应的处理程序处理，处理程序中发送数据“输出”事件—RF_DATA_PUMP 到有限状态机，激发相应的动作—RF_SEND，该动作将一个新的数据帧装载好并发送出去，同时再次发送一个空消息到 RFCOMM 接口队列。如此周而复始直到课件发送工作完成。程序设计流程图如图 6 所示。

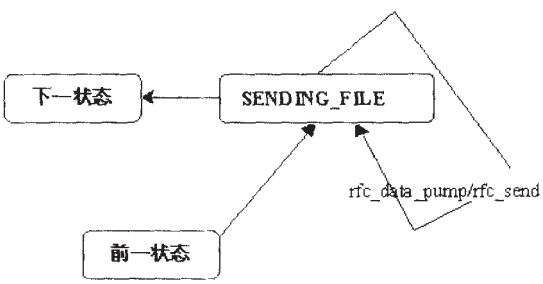


图 5 数据通信时消息时序图

## 3.4 采用 BlueStack 的实现描述

蓝牙设备间发送的原语、消息的定义，及这种原语、消息与操作系统的兼容，正是由 BlueStack 协议栈完成的。远程教育蓝牙设备间的通信设计中，利用 BlueStack 中的 RFCOMM Lib、SDP Lib、DM Lib 三个库函数与协议栈建立联系。

## 4 小结

对于蓝牙这一新兴的技术而言，市场上的 Windows 内还没有直接的蓝牙的接口函数，可以最直接、准确地在自己的产品中采用蓝牙设备的方式，就是在主机的系统设计中采用蓝牙协议栈，蓝牙协议栈将架起带有蓝牙设备的主机间的通信桥梁。这也就要求设计人员应对国际上比较成熟的一流协议栈有深刻了解，并由此触类旁通，将蓝牙技术灵活应用到工业、国防等众多场合。

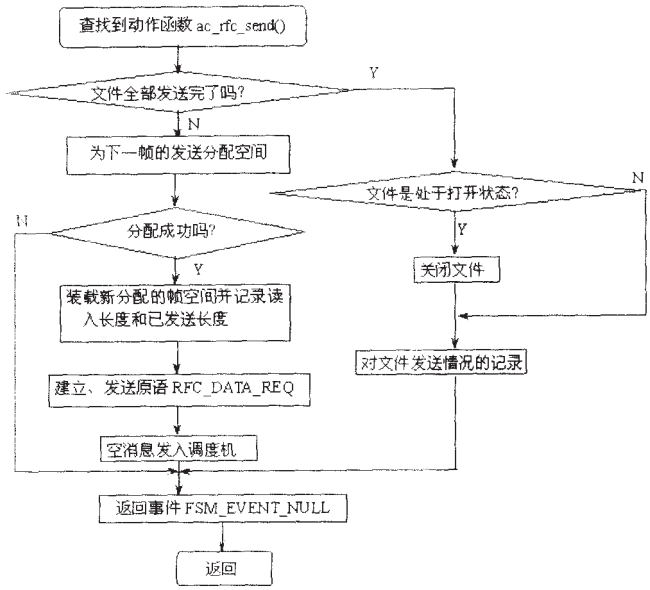


图 6 数据通信程序设计流程图

该研究课题系教育部 2001 年中央财政专项“现代远程教育关键技术与支撑服务系统和天地网结合”项目—无线接入网络教育应用部分。文中所给出的典型的蓝牙设备间连接建立和通信的过程作为该项目的核心设计实现部分，已于今年 7 月顺利通过教育部验收。（收稿日期：2004 年 1 月）

## 参考文献

1.马建仓，罗亚军，赵玉亭编著.蓝牙核心技术及应用[M].科学出版社，2003-01
2.张禄林，雷春娟，郎晓虹编著.蓝牙协议栈及其实现[M].人民邮电出版社，2001-10
3.BlueStack User Manual.http://www.mezoe.com，2001-11
4.Bluetooth specifications & profiles.http://www.bluetooth.com，2001-01

（上接 106 页）

相存的，面向对象语言提供的抽象类是为了在程序设计时能更多地、更方便地使用多态性编程思想，但是，不管使用哪种编程语言，领悟并能熟练运用多态性编程思想才是最重要的，否则，即便使用最优秀的语言，也难以开发出高效的、真正的面向对象程序。

## 6 结语

正是由于封装、继承和多态性编程思想，使面向对象的程序设计方法能自然、准确地模拟现实世界的问题，这种解决问题的新方法不仅能科学、自然地模拟现实世界的各种复杂问题，而且编程者可以方便地共享已有的软件资源和程序代码，从而大幅度提高软件的开发效率和减少今后的软件维护工作量。（收稿日期：2004 年 5 月）

## 参考文献

1.杨芙清.面向对象的系统分析[M].北京：清华大学出版社，1998-05
2.印 .Java 语言与面向对象程序设计[M].北京：清华大学出版社，2000-02
3.刘艺.Delphi 面向对象编程思想[M].北京：机械工业出版社，2003.2
4.Ian Graham.袁兆山译.面向对象方法原理与实践[M].北京：机械工业出版社，2003-01
5.蓝雯飞.C++语言中的面向对象特征探讨[J].计算机工程与应用，2000，36(9)：91~92
6.刘正林.Java 技术基础[M].武汉：华中科技大学出版社，2002
7.蓝雯飞.C++面向对象程序设计中的多态性研究[J].计算机工程与应用，2000，36(8)：97~98
8.蓝雯飞.用 C++中的子类型实现软件再用之探讨[J].计算机系统应用，1999，(6)：26
9.蓝雯飞.面向对象程序设计语言 C++中的多态性[J].微型机与应用，2000(6)：11