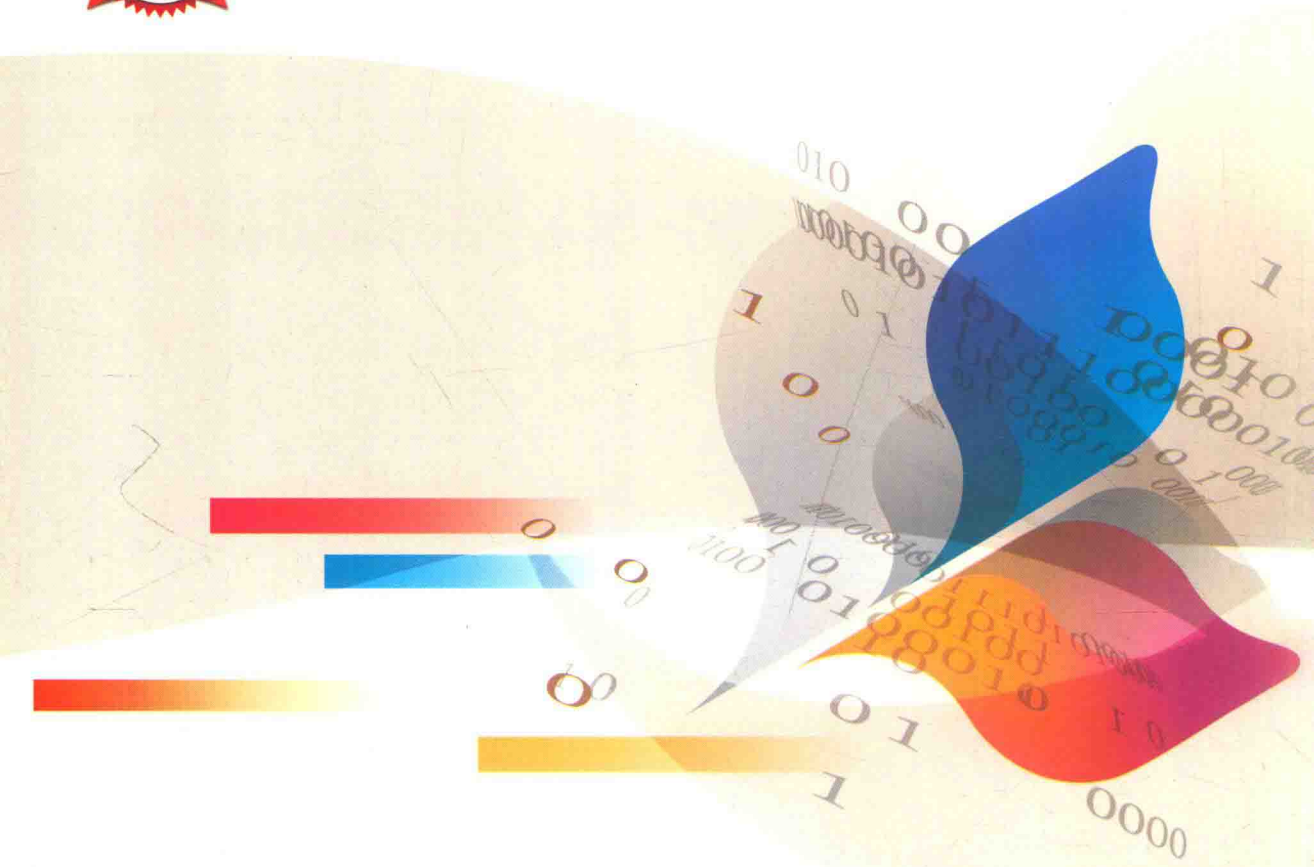
普通高等教育“十三五”规划教材



标准C++程序设计（第2版）

◎ 牛连强　马广焜　任 义　张 刚　编著

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

普通高等教育“十三五”规划教材

标准 C++程序设计

（第 2 版）

牛连强　马广焜　任　义　张　刚　编著

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内 容 简 介

本书系统介绍了C++语言的语法规则和面向对象程序设计技术。一方面，本书对众多知识点利用合理的线索进行组织、分散，力求从基本思想和存在的问题入手，逐步引入解决问题的技术和方法，再用通俗的方式阐明道理，而不是简单地用代码代替。同时，对解决问题的核心技术给予总结、概括和突出，并选取有代表性的工程应用实践问题作为设计案例，使学习更靠近应用，激发课程学习和解决实际问题的兴趣。

学习本书之前，最好对C语言程序设计知识有初步了解。全书共分10章，包括C++语言与面向对象程序设计概述，C++语言基础，类、对象与封装，类的静态成员、友元与指针访问，继承与重用，虚函数与多态性，运算符重载，流与文件操作，类模板、容器与泛型算法，异常处理。每章最后配备了若干思考题和相当数量的习题，且部分重点章节安排了若干有实际背景的设计案例。

本书可作为大专院校程序设计课程的教材，也可作为软件开发技术人员的参考书。

图书在版编目 (CIP) 数据

标准C++程序设计 / 牛连强等编著. —2版. —北京：电子工业出版社，2017.2

ISBN 978-7-121-30663-1

I. ①标… II. ①牛… III. ①C语言－程序设计－教材 IV. ①TP312

中国版本图书馆CIP数据核字(2016)第308421号

策划编辑：王羽佳
责任编辑：周宏敏
印　　刷：三河市鑫金马印装有限公司
装　　订：三河市鑫金马印装有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路173信箱　　邮编：100036
开　　本：787×1092　1/16　印张：16　　字数：473千字
版　　次：2017年2月第1版
印　　次：2017年2月第1次印刷
定　　价：39.90元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：(010)88254888，88258888。

质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。

本书咨询联系方式：(010)88254535，wyj@phei.com.cn。

前　言

C++是计算机软件领域中覆盖面最广的编程语言，得到了极为广泛的关注并有大量使用C++开发的应用系统。以介绍C++编程为目的的图书数量众多，包括国内外学者撰写的各种设计类、教材类书籍，其中不乏学习和研究C++语言程序设计的经典，如《C++ Primer中文版》、《C++语言程序设计（特别版）》和《Effective C++》等。不过，因为C++语言本身的内容多，也有很多技术和概念较为复杂，还有部分技术甚至脱离了开发者的范畴，所以，如何对内容进行取舍，以什么样的方式讨论所涉及的技术，怎样有效地结合工程实践，都是值得思考的问题。事实上，各类学校开设C++课程的时间、先期的课程安排、教学大纲的要求等都不尽相同，而多数学习者也不是以成为C++语言的专家为目标，注重的是掌握必备的知识，能更好地从事软件开发工作。基于这些观点，我们编写了《标准C++程序设计》的第一版，力图以有限的篇幅透彻讲解标准C++语言的核心内容。

本书在第一版的基础上，结合教学要求的变化和教学过程反映出来的问题，对原书做了较大幅度的调整，对结构化的程序设计内容进行压缩，扩展对面向对象分析与设计方法的讨论，并增加了项目案例以强化对工程实践的要求。总体上，本书继承了第一版的特色，同时又彰显了一些新的特点：

- **简洁通俗、平实透彻讲解**　从过程化程序设计过渡到面向对象意味着对问题的看待和处理方法的转变，甚至要花相当的精力来转变观念。因此，将重点问题进行概括、展开，用通俗的语言讲清道理，而不是用代码代替。
- **概括、突出问题核心**　对解决一类问题的核心内容给予总结、概括和突出，说明此类问题的实质和解决方法的关键而不是一个具体题目的解法。
- **由浅入深、循序渐进展开问题**　在对相关知识点进行铺垫的基础上，从基本思想和存在的问题入手，边引导边展开，逐步说明解决问题的技术和方法，避免突兀和跳跃，增强沉浸感。
- **适当引入工程问题**　选取领域中有代表性的工程应用实践问题作为设计案例，使学习更靠近应用，激发课程学习和解决实际问题的兴趣。
- **突出重要理念和思想**　作为教材，对于重要的核心问题、关键技术和应该遵循的理念给出特殊提示，这些提示贯穿于各主要知识点，对应重点掌握的核心知识进行强化。同时，这些提示可作为注意事项，对程序设计者积累经验、养成良好习惯具有很好的警示作用。

本书可作为第二门程序设计课程的教材，最好应在学过C语言之后使用。本书内容由过程化设计和面向对象两部分构成，但对过程化设计只以很少篇幅做提要式介绍。全书共分10章。第1章介绍C++语言的预备知识，并用简单示例比较了过程化程序设计与面向对象程序设计在思考问题上的差异，介绍面向对象程序设计的主要特点，基本的面向对象问题分析和程序设计方法。第2章介绍C++语言的过程化语法，并对C++语言中的基本对象做了引导性的介绍。第3、4章介绍C++语言的封装特性，第5、6章分别介绍继承和多态性。第7、8章分别讨论运算符重载和流技术。第9章简要说明了建立在类模板基础上的泛型编程技术。第10章介绍了C++的异常处理机制。

本书每章开始以精炼的语言扼要说明其主要内容，难点被适当地分解在各章里。部分重点章节安排了若干有实际背景的设计案例。每章最后配备了若干思考题和相当数量的习题。思考题有助于理解语言的语法现象，值得认真对照教材去分析，或者构造适当的例子去实验，而通过完成这些习题，有助于对知识点的透彻掌握。

书中所有的完整程序代码都由作者在C++ Builder 6.0环境下调试通过，它们都可以在其他环境下

正常运行。为了帮助学习者顺利进行编程实践，书后以附录形式对 C++ Builder 6.0、DEV C++和 Visual C++ 6.0 这几种环境的编程甚至程序调试方法都给出了适度介绍。

对于学习者，掌握程序设计技术最好的途径就是上机实验。为此，本书每章末给出了大量的实践性题目，它们都有不同程度的应用背景。我们认为，它们是使学习者经过自身动手实践并最终掌握利用 C++设计程序的不可跳跃的阶梯。

另外，本书注重程序设计的理念和实际编程技术，目标是努力提高学生的编程能力，摆脱计算机专业毕业生不会编程的现象。

本书由牛连强组织编写工作，并进行统稿；马广焜、任义和张刚分别主要参与了第 5、6 章、第 3、4 章和第 8、9 章的编写工作。

还应该说明，这是一本可以用作 48～64 学时教学的教材。我们努力从实用的角度来介绍标准 C++语言的基本内容和技术精华，但限于篇幅，若进行更深、更详细的研究时可参考书后列出的参考文献，其中不乏一些经典的著作。

本书为授课教师提供电子课件等较为全面的学习辅助资源，有需要者请登录华信教育资源网（www.hxedu.com.cn）免费下载。

我们希望本书能够高质量地满足工科学校计算机相关专业的教学需要，也特别希望读者能够不吝指出书中的缺点和错误，以便再版时能够得到改进。

作者的电子邮箱：niulq@sut.edu.cn。

目　录

第 1 章　C++语言与面向对象程序设计概述

C++语言是以 C 语言为基础发展起来的面向对象程序设计语言，但二者是相互独立的。本章简要说明 C++语言的发展历程，以及 C++程序的一般结构和主要成分。同时，概括说明面向对象程序设计的思想、问题分析和基本设计方法，以期使读者建立起面向对象程序设计的初步印象。

1.1　C++语言概述

C++是一种以 C 语言为基础开发的高级语言，保留了 C 作为它的一个子集。与 C 语言的面向系统和底层程序开发的目的不同，C++的设计目标是面向大型应用程序的开发与设计。虽然 C++基本包含了 C 语言的所有主要特性，但更趋向于对数据结构的表述，这使得 C++的核心由支持过程化程序设计转向以类为基础的面向对象程序设计，这种转变导致了程序分析、设计方法的重大改变。

考虑到目前多数读者都有学习 C 语言的经历，而且 C++本身所包含的内容、技术较多，本书的写作是以假设读者具有一些 C 语言编程知识为前提的，同时也有选择地忽略或淡化了某些不太常用或复杂的内容，以使其易于作为教材使用。

1.1.1　标准 C++语言的产生与发展

C++的发明者是 AT&T 贝尔实验室的 Bjarne Stroustrup 博士，他给出了 C++的第一个定义。首先，C++保留了 C 作为一个子集，使其具有 C 语言的处理复杂底层系统程序设计工作的能力。其次，为了增加面向对象特性，C++从 Simula 语言引入了类的概念，包括派生类和虚函数。此外，C++还借鉴了 Algol 语言的运算符重载等特性，形成了 C++的早期版本，被称为“带类的 C”，这种转变是因为引入事件驱动机制所导致的，对面向对象的支持还不够完善。早期的 C++编译系统只是一个将 C++代码翻译成 C 代码的预编译系统。

随后，C++的语法经过了若干次审查和修订，主要包括对重载的解析、连接以及存储管理，并在 static 成员函数、const 成员函数、protected 成员、多重继承、模板、异常处理、运行时类型识别和名字空间等方面进行了扩充，还增加了一些重要的数据结构和算法，目的是使 C++程序更容易编写。1988 年诞生了第一个真正的 C++编译系统。对于容器和泛型算法的支持，最初以标准模板库 STL 形式提供，目前也已正式纳入 C++标准，通常称为 C++标准库。同时，C++标准化的工作也在推进。

1991 年，C++标准化 ANSI 委员会的 C++标准化工作正式成为 ISO 的 C++标准化工作的一部分。1998 年，ISO/ANSI C++标准正式通过并发布。鉴于 C++产生和应用的实际状况，C++标准保留了对 C 及早期 C++版本的兼容，利用文件名的不同体现它们之间的差异，并容忍一些早期的语言现象如函数声明等。

总体上，C++是一种混合语言，或者说 C++是一种集过程化设计、面向对象、基于对象和泛型算法等多种技术于一体的编程语言。

本书以标准 C++为基准，重点介绍语言对面向对象程序设计的支撑技术，以及面向对象程序的分析和设计方法，体现在对数据结构（类型）以及建立在此基础上的程序处理、组织技术。在学习 C++语言时，最重要的是集中关注概念和思想，不要迷失在语言的技术细节中，因为对于程序设计内涵和设计技术的理解远比对细节的理解更重要。

1.1.2 编写简单的C++语言程序

最简单的C++程序由如下代码给出，它不做任何工作，仅是一个程序框架：

```
int  main( )                                    //主函数定义
{
  return  0;
}
```

与C程序类似，一个完整的C++程序必须且只能有一个名字为main的函数，称作“主函数”，而这个面向过程的主函数是使C++被称为混合语言的象征。但C++的程序只依靠main函数作为程序入口，供操作系统调用，其他成分以类而不是函数形式构成。main函数负责组织、协调类的对象共同协作而不是函数调用来完成既定任务。

在细节上，C++的main函数的声明类型为int而非C语言常见的void，此时，需要在函数末尾加上“return 0;”语句。

这里给出了一个简单的C++程序示例，功能是计算两个字符串的“和”并将结果输出到屏幕上。

```
/* 示例程序 Example1_1.cpp */
#include <iostream>                             //头文件包含
#include <string>
using namespace std;                            //名字空间声明
int  main( )
{
  string  x("Hello"), y("world.");              //对象生成（变量定义）
  string  z;
  z = x + " " + y;                              //消息驱动（运算）
  cout << z << endl;                            //消息驱动（输出）
  return  0;
}
```

程序运行时的输出结果为：

```
Hello world.
```

1. 程序结构

在整体框架上，C++程序的外观结构与C较为类似，主要包括预处理指令部分、声明部分和定义部分，包括main函数定义。

预处理指令用于指定编译器的某些操作，包括文件包含、宏定义和条件编译等。声明部分包括函数声明、类型声明和变量声明等，它们存在的主要目的是供编译器进行语法检查，以发现设计中存在的错误。函数定义是指规定函数的格式和要执行的代码。这些代码放在一对花括号“{}”内，称为“函数体”。函数体一般包括变量和对象定义、输入、运算和输出等内容。

2. 头文件包含与名字空间

程序中的“#include <iostream>”和“#include <string>”是用于包含头文件的预处理指令。程序包含<iostream>和<string>分别是因为使用的对象cout、常量endl和string类型定义在这两个头文件中。这里的string是一个C++的字符串类（类型），用于替代以'\0'结尾的C字符串。在没有头文件被包含时，编译器不能识别cout、endl和string的含义。应注意在指定头文件时没有使用文件扩展名.h或.hpp。

语句“using namespace std;”的作用是说明使用名字空间std。这是一种C++为了减少程序中的名

字冲突而引入的技术。名字空间用于规划程序中的空间范围，不同的定义归属不同的名字空间，程序通过指定名字空间来表明所使用的名字的来源。标准 C++的所有定义都属于名字空间 std。该语句的作用是向编译器表明，以下程序中出现的定义如果不是局部的，应属于名字空间 std。只要使用 C++标准库，就应该说明使用 std 名字空间。

说明一个名字的所属名字空间有几种不同的方式，这里采用了一种统一说明的方法，即将使用名字空间语句“using namespace std;”加在程序开头。当然，也可以不统一说明，在使用每个名字时将 std 直接放在它的前面，如：

```
std::cout << z << std::endl                    //输出
```

这种方式更明确地表明了一个名字的所属关系。

3．注释

程序注释是对程序中的代码、变量等所做的说明，在程序编译时对注释不做任何处理。程序中添加注释有助于提高程序的可读性，是非常必要的部分。在一些特殊程序中，注释可达整个篇幅的 1/3 以上。

C++支持两种注释，其一是继承自 C 的“/* 注释 */”，一般称为“注释对”形式的注释；另一种是由//引导的“//注释”形式。前者是段落形式的注释，可以包含连续的任意多行，可以在任何允许插入空格的地方插入。后者称为“行式注释”，标志着从“//”开始到本行结束的内容均为注释。

示例程序 Example1_1.cpp 中包含了上述两种注释。

通常，可以这样对程序进行注释：

（1）在程序头，用于说明程序名、功能、作者、目的、用途、修改史等；

（2）在函数或方法声明前，用于说明它们的功能、参数、返回值和副作用等；

（3）在变量定义前（或后），说明变量的含义和作用；

（4）在函数体开头，用于解释算法思路；

（5）在类定义前或类定义中添加注释，用于说明类和成员的功能。

限于篇幅，本书通常只在行末添加少量、必要的注释，这些注释以中文形式给出，多用于解释所在行代码的功能、含义和应注意的事项。

注释是程序的重要组成部分，必须养成及时添加注释的习惯，且尽量使用 C++的行式注释，因为//可以嵌套在/**/中，但/**/不可以嵌套。不过，应注意位于宏末尾的行式注释有可能被看作宏的一部分。

4．输入/输出对象

程序中通常要从键盘接收用户的输入，并将运算结果输出到显示器上。与 C 语言的函数式输入/输出技术不同，C++借助标准库中定义的对象来实现。输入数据需要使用的对象是 cin，语法形式示例如下：

```
int  x;
double  y;
cin >> x >> y;                                 //也可写成：cin>>x; cin>>y;
```

上述语句可以将用户输入的数据保存到变量 x 和 y 中。

输出数据使用对象 cout 实现，语法形式示例如下：

```
int x = 10;
cout << "x is " << x << '.' << endl; //输出 x is 0.
```

上述语句连续输出 4 部分的值。语句末端的 endl 是一个 std 空间中定义的常量，含义是刷新输出缓冲区，其作用与字符‘\n’基本相同，可以将光标转移到下一行的开头。

5. 编码习惯

C和C++的程序都以书写格式自由著称，这是指程序的书写几乎不受什么限制，可以在一行上书写几个语句，或者把一个语句写在多个行上，但正确的做法一般是一个语句占用一行。应该说，尽量保持好的书写风格是必须养成的习惯。因此，应尽量注意程序的书写“格式”，如缩进格式和成对符号的对齐排列等。本书所提供的示例程序将尽量给读者提供一种可借鉴的规范，偶尔（如习题中）可能将相近的代码写在同一行内，这纯粹是出于篇幅的考虑。

程序的良好格式（如一致的缩进编排风格）是使程序具有可读性、使设计者能与他人合作的基础，有时比解决问题本身还重要。

1.2 由过程化到面向对象程序设计

作为一种设计技术，面向对象对软件研发、应用甚至日常生活的影响都是巨大的，且这种作用仍在扩大并发展。互联网上大量的应用和功能也都以对象的方式呈现，我们每天都面对着各种各样的对象，了解面向对象程序设计技术已成为学习软件开发者的必然，这意味着设计思想由面向过程向面向对象的转变。本节仅通过一个简单的应用来比较一下两种技术和思想的差别，具体技术细节将在后文中展开。

1.2.1 过程化程序设计

用计算机解决问题时总要设计程序，即以某种语言为工具编写控制计算机执行的动作序列，每个动作规定了一定的基本操作。自计算机问世以来，人们不断地实践并总结着有效的程序设计方法。当然，这种变革是以简化编程和提高软件生产率为目的的。

由于编程问题起源于最初的科学计算，因此，在相当长的时间里，程序都是围绕着数据的组织与算法（处理、操作）的切分展开的。面对一个具体问题，首先要“建立需求分析和系统规格说明”，即“建立一系列规则，根据它判断任务什么时候完成，以及客户怎样才能满意”。如同现实社会中的一个大型项目或产品开发一样，先将一个很大的“产品”分成适当的部件，再由这些部件组合成整个产品，而程序员的职责就是考虑如何更好更快地实现这些部件，一般称为“过程”或“模块”。这样的处理方式被认为是基于或面向过程的。在这里，数据和算法是问题的中心，程序设计的中心工作是研究数据的描述、存储和数据间的关系，以及作用在数据结构上的算法，每个算法通过一个或几个过程来体现，完整的程序由算法（模块）之间互相调用组成，如图1-1所示。此时，人们认为“算法+数据结构＝程序”（N. Wirth的观点）。

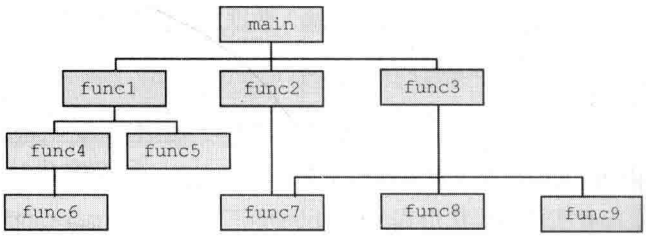


图1-1 由函数组成的程序

过程化的问题处理思路形成了一套有效的程序设计方法，称为结构化方法，它体现在以下三个方面。

（1）程序设计采用自顶向下、逐步细分的方法展开。

（2）模块化。这是指程序中的过程体和组成部分应以模块表示，模块还可以称为过程、子程序或函数。每个模块应具有较高的独立性，即强内聚性和弱外联性。这里的内聚性是指模块内部功能的单一性，而外联性是指同其他模块的联系。这样做的目的是使对一个模块的修改不致于对其他模块造成太大的影响。

（3）使用三种基本控制结构。这是指描述任何实体的操作序列只需要“顺序、选择、循环”这三种基本流程控制结构，参见图 1-2。从整体上看，模块中的指令（语句）总是由上到下按顺序逐个执行的，但局部可能有选择或循环。三种基本结构的共同特点是每种结构只有一个入口和一个出口，这使程序更容易理解和维护。

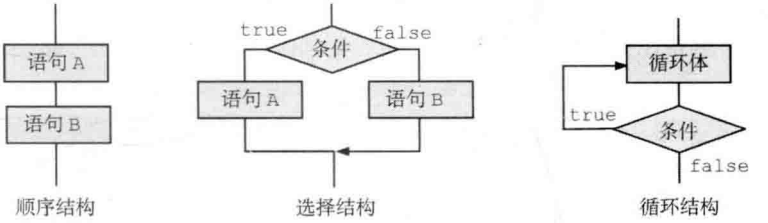


图 1-2　三种基本结构

这里考虑一个五子棋游戏的设计示例。采用过程化方法可以将整个问题按处理过程分解如下：

开始函数 start( )；

绘制棋局画面函数 drawChess（棋局数据）；

走棋函数 play（选手）；

判断输赢函数（棋局数据）；

输出结果函数 judgeWinner（棋局数据）；

结束函数 stop( )。

实现游戏的程序可以按下述简化流程实现：

```
int  main( )
{
   drawChess(初始棋局);              //绘制棋局
   start( );                         //初始化工作，如启动计时器，假定黑棋先走等
   do                                //重复执行下述操作
   {
     play(执黑选手);                 //执黑选手走棋
     drawChess(棋局);                //刷新棋局
     if(judgeWinner(棋局))           //判断，已定出输赢
       break;                        //停止重复
     play(执白选手);                 //执白选手走棋
     drawChess(棋局);                //刷新棋局
     if(judgeWinner(棋局))           //判断，已定出输赢
       break;                        //停止重复
   }while(未超时);                   //达到预定时间时停止重复
   showResult(棋局);                 //输出胜负结果等
   stop( );                          //终止
}
```

由于过程化设计中的数据与过程是分离或者说相互独立的，且数据（如棋局）有时是全局的。一

个过程完全可以作用到并不相关的数据上，也不能保证对数据操作的合理性，数据对于算法（操作）完全是被动的。这种操作是一种“谓语＋宾语”结构。

1.2.2 面向对象的程序设计

面向对象设计方法将客观世界看成是由对象组成的。对象是一种实体，具有自己的属性和行为。一个程序由对象及其相互间的协作来实现。

例如，对于五子棋游戏，完成一次比赛应由两个选手、一个裁判和一个组织者组成。选手只负责走棋，裁判负责判定输赢和确定比赛是否结束，组织者提供场所和设施。这里的每个人是一个对象，对象之间通过消息实现协作。不同对象具有不同的行为。例如，比赛选手负责走棋，裁判负责确定比赛开始、判断输赢、确定比赛结束，而组织者负责绘制画面和输出结果等。为此，首先要定义如下数据结构（类）：

```
class  Player                          //选手描述
{
    string  piece;                     //执黑或执白
  public:
    Player(string  p);                 //构造器
    play( );                           //走棋
}
class  Referee                         //裁判描述
{
  public:
    start();
    judgeWinner(棋局);
    judgeTime();                       //判断超时
    stop();
}
class  Playmaker                       //组织者描述
{
    ChessInfo  chessInfo;              //棋局信息
  public:
    drawChess();
    showResult();
    announceInfo();                    //公布棋局
}
```

于是，可以按如下方式组成程序：

```
int  main( )
{
   Player  player1("执黑"), player2("执白");    //初始化，生成对象
   Referee  referee;
   Playmaker  playmaker;
   playmaker.drawChess();
   referee.start( );                   //初始化工作，如启动计时器，假定黑棋先走等
   do
   {
```

```
        player1.play();
        if(referee.judgeWinner(playmaker.announceInfo()))
          break;
        playmaker.drawChess( );
        player2.play();
        if(referee.judgeWinner(playmaker.announceInfo()))
          break;
        playmaker.drawChess();
      }while(referee.judgeTime());
      playmaker.showResult();                    //输出胜负结果等
      referee.stop();

      return 0;
    }
```

在程序中，Player、Referee、Playmaker 和 ChessInfo 分别是对参赛选手、裁判、组织者和棋局这些概念的简化描述（ChessInfo 的定义未给出），可以理解为是一些数据类型。每种概念所生成的实例（变量）称为“对象”。棋局信息由组织者维护，并可以向外界公布。其中，棋手对象 player1 和 player2 负责走棋，并告知组织者对象 playmaker 有关棋子布局的变化。组织者对象接收到这些信息后负责在屏幕上显示这种变化，裁判对象 referee 负责对棋局以及比赛时间进行判定。程序实现时，组织者先显示棋局，裁判宣布比赛开始，执黑和执白选手交替落子。每次落子后，裁判确定是否该选手已获胜，一旦获胜或比赛时间到则终止比赛。否则，组织者刷新棋局。

尽管上述程序还不能全面解释面向对象程序设计的所有特性，但可以明显看出，面向对象是以功能而不是步骤来划分问题的，这种方式与人的日常思维方式吻合。不同对象维护着自身的相关信息（数据、属性），并具有一定的功能（函数、方法）。所有对象各司其职。对象自身属性和行为方式的改变不会影响到其他对象，因为对象间仅通过互通消息实现合作，如图 1-3 所示。

图 1.3 中的接口就是指一个对象能够对外提供的服务（方法）。从实现上看，对象的每次操作都是在该对象接收到一定消息（用“对象.函数名”形式表示）后的自主行为，具有“主语＋谓语”的形式。

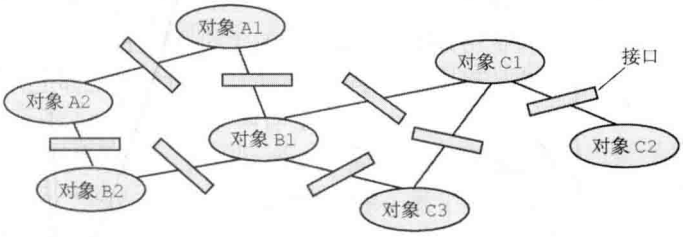


图 1-3　面向对象运行模式

在采用面向对象技术解决问题时，首先要考虑整个项目的求解是由哪些对象组成的，再分析对象应具有什么样的行为，最后考虑对象之间应怎样协作。

总之，过程化与面向对象是两种分析和解决问题的不同方法，对于一些简单的问题，基于过程的解决方法是十分有效的，而对于大型、复杂的系统，采用面向对象方法更能显示出优势，有利于采用对象构成软件“积木插件”，进而在一定程度上解决软件重用的难题。

1.3　面向对象程序的主要特征

客观世界是由对象组成的，这是面向对象程序设计（Object Oriented Programming，OOP）的基础和出发点。例如，考虑“我看电视”这样一个场景。“人”、“遥控器”、“电视”是问题中所涉及的概念，

每种概念在C++中用一个数据类型来描述，称为“类”（class），而客观存在的实体对象都是某种概念的一个具体实例，有形体特征和具体行为。这里有一个“我”，是“人”类的一个对象。一个具体的电视机，是“电视”类的一个对象，还包括一个“遥控器”类的对象。在实际工作时，对象“我”操纵遥控器对象，遥控器对象向电视机对象发送开机、关机和调换频道等消息（指令），电视机对象响应这些消息，执行相应的操作。

1.3.1 抽象与封装

尽管一些简单的数据如整数、字符等可以由系统提供的基本数据类型来刻画，但更一般的概念必须自己来描述，其结果就是定义一个类。下述代码给出了对电视的一种概略描述：

```
class  TV
{
  public:                                          //实际设计时应采用 private 关键字
     int color;                                    //颜色属性
     int size;                                     //尺寸属性
     //...                                         //其他属性
   public:
     TV(参数列表);
     void open();                                  //打开
     void close();                                 //关闭
     void changeChannel(int channel);              //调换频道
     virtual void display(int channel);            //显示频道
     //...                                         //其他方法
};
```

通常，任何概念都会包含两方面的内涵，其一体现事物的状态，其二是其行为。例如，“人”是一种概念，通过身高、体重、性别、年龄和肤色等体现了人的自然状态，而生长、学习、吃饭和劳动等说明了人可具有的行为。这种状态特征用数据来刻画，称为类的“属性”，而行为特征用函数（算法）来体现，称为类的“方法”，属性和方法都是类的成员。

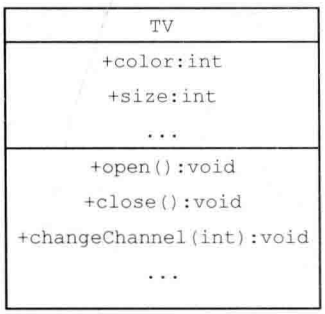


图1-4 类TV的图形描述

在定义中，类TV有两类成员，color、size等为属性，open、close和changeChannel等为类的方法。通常，类可由图1-4所示的图描述，它来自于统一建模语言UML的表示方法。

类的定义是一种数据类型定义，因此，可以像C++的内置类型那样生成变量，这种变量就称为一个类的实例或对象。例如，下述代码生成了类TV的一个对象：

```
TV  tv(参数列表);                                  //对象定义
```

生成对象时，参数列表被传递给类的一个特殊方法TV，它与类的名字相同，称为“构造器”。因为构造器的存在，使我们能够生成具有各种各样状态而非千篇一律的对象。可见，类是一种用来生产对象的“模板”。

一个类定义所达到的最重要的效果是实现了对数据和函数的封装（encapsulation），而封装的主要目的是数据隐藏。这不仅是对客观实体的一种合理完整的刻画，也使对象本身所包含的内部数据不致于被无意中破坏。对电视机的使用者来说，如果需要调换频道，可以通过遥控器向电视机tv发送调换频道信号（消息）。电视机可以接收和响应此消息，但如何调换频道是电视机本身的能力，由电视机而