HZ BOOKS
华章教育

高等院校计算机教材系列

面向对象程序设计
C++语言编程

张 冰 编著

本书配有电子教案

机械工业出版社
China Machine Press

高等院校计算机教材系列

面向对象程序设计
C++语言编程

张 冰 编著

机械工业出版社
China Machine Press

本书采用举例、比拟等多种手法，用通俗易懂的语言及生动活泼的例子讲解了面向对象程序设计的基本概念和基本方法，并运用了软件工程的思想和方法，为学生从事具体软件项目开发奠定了基础。本书共分9章：第1、2章介绍C++程序设计语言基础；第3章介绍面向对象程序设计的基本方法和思想，详细说明抽象、数据封装和信息隐藏、概括等面向对象特性；第4～8章围绕面向对象程序设计的数据封装、继承性、多态性三个基本特性，讲述类与对象、构造函数与析构函数、继承与派生、虚函数与多太性、友元函数与友元类、静态成员、模板、异常以及输入输出流等内容；第9章简要介绍了利用MFC类库设计Windows应用程序的基本方法和思想。

本书循序渐进，书中语言基础、程序设计和编程应用三部分内容相互衔接，前后呼应，每章还提供大量富有启发性的习题和配套的实验以方便读者复习、巩固。本书可作为高等院校计算机及相关专业本科面向对象程序设计课程的教材，也可作为广大工程技术人员和计算机爱好者的自学教材。

图书在版编目（CIP）数据

面向对象程序设计：C++语言编程/张冰编著. —北京：机械工业出版社，2008.1
（高等院校计算机教材系列）

ISBN 978-7-111-22664-2

Ⅰ.面…　Ⅱ.张…　Ⅲ.①面向对象语言-程序设计　②C语言-程序设计　Ⅳ.TP312

中国版本图书馆CIP数据核字（2007）第166252号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）
责任编辑：刘立卿
三河市明辉印装有限公司印刷·新华书店北京发行所发行
2008年1月第1版第1次印刷
184mm×260mm·20.25印张
标准书号：ISBN 978-7-111-22664-2
定价：32.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换
本社购书热线：（010）68326294

前　言

程序设计语言是人与计算机进行交流的一种形式语言，是人们运用计算机分析问题并解决问题的一种基本工具。熟练掌握一门程序设计语言是每个工科学生必须掌握的基本功。C++ 语言作为一种高级程序设计语言，是学习面向对象程序设计的首选语言之一。

学生在学习 C++ 语言面向对象程序设计过程中往往面临以下问题：概念比较抽象难于理解，方法比较独特难于接受，内容比较新颖难于适应。针对上述问题，本书作者结合自己的教学和编程实践经验，并参考借鉴国外教材的先进方法和思想编写了本书，力图用通俗易懂的语言并结合实际应用例子来讲解各个知识点。重要和抽象的概念采用比拟的手法，借用学生熟悉的一些实际生活中的例子，以生动活泼的形式加以描述。全书的编写贯穿软件工程的基本观点和实践方法，注重用先进的开发工具和开发方法讲述 C++ 语言面向对象程序设计的基本概念和基本方法，说明为什么要采用面向对象程序设计，以及怎样采用 C++ 语言实现面向对象的程序设计。通过大量的例题和练习，介绍了数组、链表、队列、堆栈等基本数据结构，讲解了多种求 π、求素数、搜索、排序和递归等常用算法的设计和实现。通过对本书的学习，学生能够理解和掌握面向对象程序设计的基本概念和基本方法，具备一定的运用基本数据结构和算法进行程序设计的能力。

本书共分为9 章。第 1 章和第 2 章讲述面向过程和面向对象程序设计的基本特性，C/C++ 语言的特点，C/C++ 程序的结构，基于 Visual C++ 6.0 集成开发环境的 C++ 程序的编辑、编译和运行步骤，基本的 C++ 语言语法。包括数据类型，运算符，表达式，顺序、选择和循环 3 种基本程序结构，数组，函数，指针，引用等，这一部分构成了 C++ 程序设计语言的基础。第 3 章进一步介绍面向对象程序设计的基本方法和思想，说明“抽象”、“数据封装和信息隐藏”、“概括”作为面向对象程序设计的 3 个基本方法较比以高内聚、低耦合为特点的结构化程序设计方法的优越性。第 4 章至第 8 章围绕面向对象程序设计的数据封装、继承性和多态性 3 个基本特性，讲述类与对象、构造函数与析构函数、继承与派生、虚函数与多态性、友元函数与友元类、静态成员、模板、异常以及输入输出流等内容，这部分主要讲解 C++ 实现数据封装和信息隐藏以及软件重用和接口重用等面向对象程序设计的基本特征和基本方法。第 9 章作为面向对象程序设计思想和方法的一个具体应用，以 Visual C++ 6.0 作为开发环境，简介在 Windows 环境下利用 MFC 类库设计 Windows 应用程序的基本方法和思想，讲述了 MFC 应用程序框架、MFC 类库的层次结构、Windows 消息处理机制和基于 MFC 的消息映射方式，介绍了鼠标消息、键盘消息和用户自定义消息的处理方法等。

本书采用的编写顺序是先面向过程后面向对象，先语言基础后程序设计再到编程应用，进而以应用促进基本概念和基本程序设计方法的理解和掌握。书中的语言基础、程序设计和编程应用三部分内容相互衔接，前后呼应，便于读者循序渐进地学习。本书每一章都附有大量有启发性的习题，还提供了配套的上机实验便于读者加深理解和巩固提高。本书可作为高

等院校计算机及相关专业本科面向对象程序设计课程的教材，也可作为工程技术人员和广大计算机爱好者自学的参考书。

本书在编写过程中得到了深圳大学信息工程学院相关课程组老师的大力支持，部分师生对本书的修改提出了宝贵的意见，本人在此表示感谢。

本书的所有程序都经过了调试，并运行无错。由于作者水平有限，书中难免有错误和不妥之处，欢迎广大读者批评指正。

张　冰

2007 年 10 月

目　录

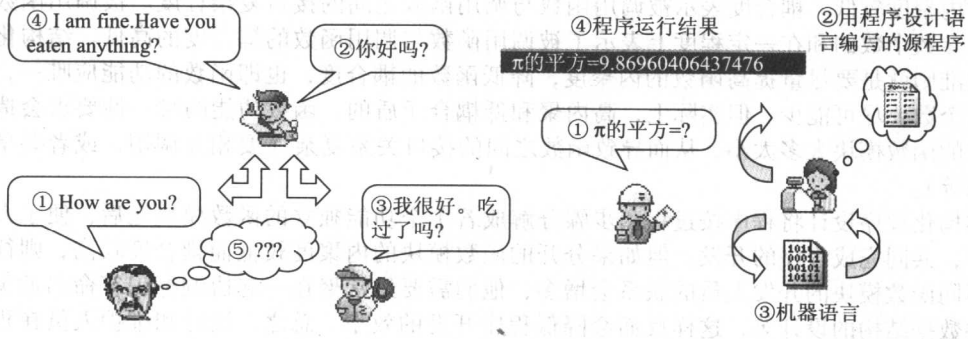
第1章　程序设计与C++语言

1.1　程序设计与程序设计语言

程序设计如同电子、机械和建筑设计，也是一种工程设计。程序设计的本质就是用程序设计语言编写计算机为完成某一特定任务而必须执行的一系列指令。按照面向对象程序设计的观点，程序设计可以看成是从问题空间到程序空间的一个映射，程序空间中的程序就是一个现实世界问题的软件模型。这里，问题的每一个实体用一个软件组件来实现。每个软件组件模拟它所表示的现实实体的状态和动作，编程就是建立对象的模型。程序设计的难点在于怎样将现实世界待求解的问题用软件模型来描述。

程序设计语言是人与计算机进行交流沟通的一种形式语言，是人们运用计算机分析和解决问题的一个基本工具。程序设计语言由一组特定的文字集和一定的语法规则组成。最早的程序设计语言是原始的计算机指令，即一串可被计算机直接识别的二进制数序列，称之为机器语言。随后，在机器语言的基础上增加了便于人们记忆的助记符，即所谓的汇编语言。再之后，Fortran、Basic、C、C++、Java等上百种高级程序设计语言应运而生。

图1-1形象地表示了人与人之间采用不同自然语言交流和人运用程序设计语言与计算机交流的相似与不同之处。



a) 人们采用不同自然语言(英文和中文)进行交流　　b) 程序员运用程序设计语言与计算机进行交流

图1-1　自然语言与程序设计语言

图1-1a描绘了一个不懂中文的美国人用英语和一个不懂英语的中国人用中文交流的情景，两者之间的交流经过一个既懂英语又懂中文的翻译将双方的对话翻译成各自的母语（图中的数字序号表示对话的顺序）。由于文化背景的不同，双方在交流中可能会发生相互误解的情形：当中国人出于客套，问美国人是否吃过饭了，美国人可能认为这个中国人要请他吃饭呢！

图1-1b则形象地表示了程序员将某个计算问题和求解该问题的算法用程序设计语言加以描述，然后经编译程序（其功能相当于图1-1a中的翻译）编译，转换成计算机能够直接识别和执行的机器语言的过程。计算机通过执行用机器语言表示的指令，将计算结果通过其输出装置反馈给人们。

程序设计语言可分为低级语言和高级语言两大类。人们用诸如机器语言的低级程序设计语言编写的程序能直接被计算机的控制器所识别，从而控制计算机的运算器进行运算，其工作方

式就像两个人用同一种语言交谈一样。而采用高级程序设计语言（如 C++）编写的程序，必须经过编译器编译，转换成机器语言之后计算机才能识别与理解，就像只会中文的人和只会英语的人对话一样，必须经过翻译。计算机将高级程序设计语言编写的程序翻译成计算机能识别的机器指令的过程叫做编译。

程序设计语言的发展是一个从低级到高级、从具体到抽象的过程，语言越高级意味着它越接近自然语言。目前，程序设计语言呈线性等差级数的发展态势，还远远跟不上计算机硬件呈指数等比级数发展的趋势。

1.2 面向过程和面向对象程序设计方法简介

1.2.1 面向过程的程序设计

结构化程序设计（面向过程的程序设计）的基本思想是：自顶向下、逐步求精、模块化设计和结构化编码。首先是从问题出发，找出解决问题的基本思路和过程步骤，将抽象复杂的功能不断细分为更小更简单的独立的子功能，直到人们对每个子功能的实现过程比较清晰。然后用函数作为模块化的单位，将每个子功能用一个独立的函数，采用顺序、选择和循环三种结构化控制语句的组合来实现。结构化编码限制程序指令的任意跳转，便于人们抽象地理解程序的高层功能，提高了程序的可读性和可理解性。

衡量结构化程序设计的好坏有两个指标：内聚度（cohesion）和耦合度（coupling）。内聚度是指同一个函数模块中各语句之间的关联程度。若一个函数内的多条语句实现了一个共同的功能，则该函数是高内聚的。如果一个函数内的语句代码完成的是若干互不相关的功能，则该函数的内聚度较低。耦合度表示被调用函数与调用函数之间的接口复杂程度。被调用函数的输入输出参数个数的和在一定程度上表示了被调用函数与调用函数的耦合度的高低。结构化程序设计的准则就是要尽量提高函数的内聚度，降低函数的耦合度，也即函数的功能应唯一，函数的参数个数要尽可能少。但实际上，高内聚和低耦合矛盾的，函数功能的唯一性要求会造成划分出来的函数模块太多太小，从而导致函数之间的接口关系复杂（要相互调用，或者共享同样的数据等）。

结构化程序设计将程序按过程和步骤分解成若干个功能独立的函数模块之后，便于人们分工合作，共同完成程序的开发。但如果分开的函数模块的内聚度较低而耦合度较高，则往往会导致不同函数模块的开发人员的联系会增多，他们需要经常聚在一起协调标识符命名冲突、进行公共数据结构的设计等，这样反而会降低程序开发的效率。总之，设计和维护人员在程序的开发和维护过程中需要考虑的因素太多，容易出现难以发现和改正的错误。

结构化程序设计不能保证将程序中应该放在一起的函数模块放在同一个单元。例如，访问和修改某个特定数据的一组函数（比如 getData()、editData()和 printData()等）应该放在程序源代码的同一个地方，这样有助于维护人员和新的程序设计人员理解程序员原来编写程序时的意图。但结构化程序设计语言（如 C 语言）并不能提供这样一种机制保证这些函数能聚集在一起，这些函数有可能是分散在同一个源程序文件的不同地方，也有可能是分散在同一个程序的多个源程序文件之中。这样，函数模块之间的联系会增多，导致多个函数模块访问同一数据，这些都会增加理解和维护程序代码的难度。

导致上述问题的根本原因在于结构化程序设计方法本身的缺陷，亦即程序设计是以功能为中心、按步骤来进行的。数据分散在通过对数据进行处理而实现程序某个子功能的函数模块之间。同一类数据可能会由多个函数模块来处理，增加这些函数模块的耦合程度。另一方面，程序设计不是以数据为中心，造成的结果往往是应该分开的，关系不紧密、低耦合的多个函数模块却聚集在了一个大的函数模块中，降低了该函数模块的内聚度，增加了软件开发和维护的费用。

1.2.2 面向对象的程序设计

面向对象程序设计的基本思想是将数据和对数据进行操作（输入、访问、修改和输出等）的函数绑定封装在一个称为类的数据结构中。通过类将程序中相关的内容合并在一起，解决了结构化程序设计存在的问题。这样在修改某个数据时，只要熟悉描述这个数据的类就行了，而不必担心程序中还有其他函数会访问或修改该数据，这就是面向对象程序设计的封装性（encapsulation）；并且还可通过访问控制机制将数据私有化，以保证只有与该数据封装在同一个类的函数才能访问这些数据，即面向对象程序设计的隐藏性。封装性和隐藏性构成了面向对象程序设计的基础，数据与处理数据的函数封装构成了对象，数据受到了保护，要访问对象的数据只能通过调用对象的成员函数来进行。图1-2中以一个公司销售、人事和财务三个部门的组织和运作方式为例说明了这一概念。每个对象（部门）有自己的数据和处理数据的方法，每个对象的数据是私有的，在对象外是不可见的。例如，人事部的数据（人事档案）对财务和销售部门而言是不能随便访问的。如果财务部需要销售部的销售数据，财务部经理不能够直接访问销售部的销售数据，而必须首先给销售部经理发一条消息（电话、书面请求、E-mail）请求销售部经理对销售数据进行处理，并将结果返回给财务部经理。

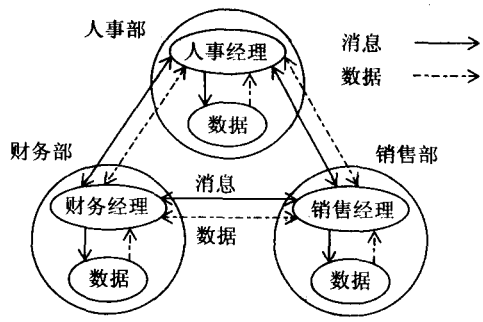


图1-2 一个公司的组织机构及其运作方式

封装性和隐藏性是衡量一个采用面向对象程序设计方法设计的程序的基准。采用面向对象的程序设计方法，是将程序编写为一组相互协作的对象，而不是一组相互协作的函数。类作为程序的基本单位可减少各个部分之间的相互联系，降低程序代码的耦合性。类的成员函数对同样的数据进行操作，有助于实现模块化，提高程序代码的内聚度。通常一个类可由同一个程序员来开发实现，降低了类之间的相互依赖性，进而减少各个类的开发人员之间的联系，避免由于经常需要相互协调而容易出现遗漏、误解而导致错误的情况发生。

面向对象程序设计的封装性和隐藏性带来的最大好处就是提高了软件的可维护性。由于数据和对数据的所有操作被封装在类中，数据被设置成私有的只能通过调用类的对数据进行处理的函数来操作。这样，软件维护人员可以通过类的成员函数名清晰地掌握软件设计人员的意图，判断代码的含义。同时，当数据发生改变时，只需要修改相应类中对这些数据进行处理的成员函数就可以了。例如，程序员开始时只使用最后两位数来表示年份，到了2000年后，维护人员需要将用2位数改成用4位数来表示年份。采用传统的方法就需要检查代码中所有涉及年份的地方，一旦有一处遗漏就会产生“千年虫”问题。而采用面向对象程序设计的方法，只需找到包含有表示年份数据的所有类的描述，然后对类中处理年份数据的函数加以修改即可，而不需要对调用这些类的函数的代码做任何修改。

除了封装和数据隐藏，面向对象程序设计的另一个特性就是继承（inheritance）。继承性提供类复合的实现机制，有助于实现代码的重用。Microsoft公司就是运用面向对象程序设计方法，将Windows编程用到的几百个应用程序接口函数（API）封装成类，并通过继承机制将其组织成MFC(Microsoft Foundation Class）类库。程序员在编写基于Windows的应用程序时，可通过从MFC类库中派生出基本的对Windows底层操作的类，然后对派生类进行修改和扩充来满足应用程序的要求。继承性使得软件的可重用性大大提高，它是面向对象程序设计的关键。

多态性（polymorphism）是指对象能用不同的接口访问同名的函数，除此还表现为运算符重载（operator overloading）和虚函数（virtual function）。具有不同接口的多个同名函数称为函数的重载，它解决了多个程序员分工合作，共同完成程序开发时经常遇到的变量标识符和函数名命名冲突的问题。程序员只要保证函数的命名和接口为函数的调用者所知即可，而不需要与

开发同一程序的其他程序员经常聚在一起协调函数的命名，这样他们就可以集中精力完成各自所承担的开发工作。运算符重载是对运算符的功能进行重新定义，给运算符赋予新的功能并应用于对象的运算，使程序设计变得简单直观，易于理解阅读，提高程序的可读性和可理解性。虚函数是采用动态联编（dynamic binding）技术，根据对象在程序运行时的动态类型将函数调用绑定到具体的函数实现上，体现了接口与实现分离，以及接口重用的面向对象程序设计思想。

总之，面向对象程序设计提供了一种新的解决程序设计问题的方法。对象是面向对象程序设计的核心。类就是对一组相似对象特征的抽象描述，这些对象具有共同的属性（数据成员）和处理问题的办法（成员函数）。对象是类的一个具体实例，同一个类的不同对象具有其自身的数据，处于不同的状态中。

下面通过一个假设的简化功能的 Siemens-01 型手机为例来说明面向对象的方法和概念。

所有的 Siemens-01 型手机模型（见图 1-3）可用一个 Siemens-01 类来描述，它有如下的属性和操作：

属性：

- 手机框架尺寸
- 电路芯片
- 电源
- 按钮
- 键盘
- 黑白显示幕

操作：

- 开机
- 关机
- 打电话
- 接电话
- 调节音量

图 1-3 Siemens-01 型手机模型

一部 Siemens-01 型手机就是 Siemens-01 类的一个对象，你和你朋友的两部 Siemens-01 型手机是 Siemens-01 类的两个不同对象。你在打电话的时候，你朋友可能没有打，或者你在接电话，而你朋友在打电话，这时两个手机对象处于不同的状态中。打电话就是通过“按钮”调用手机对象的方法，发送一条消息给另一个手机对象，收到这条消息的手机对象立即操作电路芯片响应收到的消息，建立起两部手机对象之间的通信信道。

继承性体现在产品的更新换代上，假设 Siemens-02 型手机在 Siemens-01 型手机的基础上增加了编辑短信功能，Siemens-02 型手机不需要从头开发。描述 Siemens-02 型手机模型的 Siemens-02 类可从 Siemens-01 类中派生而来。Siemens-01 型手机所有的功能都被 Siemens-02 型手机继承下来，并且 Siemens-02 型手机对象增添了新的短信编辑功能且可对从 Siemens-01 手机那里继承的功能进行完善和补充。

多态性体现在不同类型的手机采用不同的界面来实现同一个功能。比如 Siemens-01 型手机和 Siemens-02 型手机分别使用了不同的电路和芯片来控制音量设置的功能，虽然用户在设置音量时的操作界面或功能按键可能不一样，但产生的效果应该是一样的。这样两种不同类的对象产生同样结果的行为就是多态行为。

相对于结构化程序设计语言，支持面向对象程序设计的语言通常比较复杂，学习难度相对比较大，对程序开发人员和管理人员进行培训的费用会比较高，所以在开发小型程序时，面向对象的方法可能不占优势。另外，面向对象的分析和设计方法要比面向过程的方法相对更新颖，更难掌握。因此，在项目的分析和设计阶段通常要花费更长的时间，尤其对于已经习惯于采用面向过程程序设计方法的有经验的分析员和程序员而言，要改变他们的固定思维模式往往是比

较困难的。其结果可能是开发出来的软件不但体现不了面向对象方法的优点，反而使得程序可能比传统的程序更长、更复杂、更慢甚至更难以维护。编写本书的目的也就是要在介绍 C++ 面向对象程序设计语言的同时，介绍怎样避免和解决上述问题的方法，从而实现用面向对象程序设计方法全面提高软件开发性能的目标。

1.3 C 语言和 C++ 语言的特点

1.3.1 C 语言的特点

C 语言作为一种结构化程序设计语言，具有如下一些特点。

1）语言简洁紧凑，使用方便灵活。C 语言只有 32 个关键字，程序书写形式较为自由。

2）具有丰富的运算符和数据类型，支持结构化的块结构和流控制。适于实现复杂的算法和复杂的数据结构。

3）提供了直接访问内存地址的机制，能进行位操作，允许程序员控制计算机中寄存器、端口、标志位屏蔽码等硬件资源，生成的目标代码效率高。C 语言成为一种面向性能的系统程序设计语言，在操作系统等系统软件以及面向硬件资源的嵌入式系统开发中得到广泛的应用。

4）可移植性好。C 语言在源代码级上支持程序的可移植性，用 C 语言写的源代码可以不加修改地在不同的编译器上进行编译或者在不同的操作系统或不同的硬件平台上进行编译，经编译后的可执行程序就可以在不同的操作系统或不同的硬件平台上运行。

C 语言是一个结构化程序设计语言，不支持面向对象的程序设计方法。所以，当用 C 语言编写的程序的规模达到一定程度时，C 语言的局限性就体现出来了。首先，程序员很难控制程序的复杂性。C 语言没有一种能保证将数据和对数据进行处理的所有函数聚合在一起的机制，数据分散在对数据进行处理的函数之间，而且函数与函数之间的逻辑关系和紧密程度不能够清晰地表示出来，导致程序不便于维护。其次，C 语言没有支持代码重用的语言结构，设计好的程序很难被重用。另外，C 语言的不足之处还表现在以下两个方面。

1）数据类型检查机制相对较弱，在表达式和函数调用中可以自动地进行数值类型之间的转换，这使得程序中的一些错误不能在编译阶段被发现。例如，表达式 1/2 的值为 0 而不是 0.5。

2）数组下标越界既不能在编译阶段被发现，也不能在运行时被发现，从而造成因非法使用下标导致内存被破坏的错误。此类诸如指针操纵、参数传递、变量初始化等都是导致程序出错的根源。

C 语言好比是一把砍树的电锯，用它来砍树比起用斧头效率要高得多，但如果不熟练，可能锯断了自己的腿哦！

1.3.2 C++ 语言的特点

C++ 语言是从 C 语言进化而来的，它是 C 语言的超集。C++ 语言在继承了 C 语言的全部特征和优点的同时，对 C 语言进行了扩充，主要是引进了“类”这一复合数据类型以支持面向对象的程序设计。C++ 语言对 C 语言完全向后兼容，符合程序设计人员逐步更新其程序设计观念和方法的要求，成为最流行的程序设计语言之一。C++ 同时也是学习面向对象程序设计的经典语言。

C++ 语言的特点具体体现在以下几个方面。

1）在 C 语言的结构数据类型的基础上增加了“类”这一复合数据类型，C++ 的类将关系密切的一组函数聚集，并与其共同处理的数据结合在一起。数据被限定在类的范围内，数据的访问只能通过调用类中的函数来实现，在类的外部不能直接修改数据。

2）C++ 程序由对象组成。任何事物都是对象，复杂的对象可以由比较简单的对象以某种方式组合而成。客观世界由各种对象组成，整个世界就是一个最复杂的对象。因此采用 C++ 语言进行程序设计与现实世界的客观规律和人类习惯的思维方式相符合。

3）C++提供了类继承和派生的实现机制，高层次的类可以重用低层次的类。类复合和类继承便于程序员在计算机中建立现实世界复杂的软件模型，是支持代码重用实现面向对象程序设计继承性的根本保证。

4）C++通过构造函数与析构函数自动地管理对象所占用的资源。在对象定义时，不需要显式地调用初始化函数，就可自动地调用构造函数为对象分配资源，实现对象的初始化。当对象生存期结束对象撤销前会自动调用析构函数，收回分配给对象的资源。

5）C++支持函数的重载，具有不同接口的函数可以使用相同的函数名。在不同的类的内部也可以使用相同的变量名和函数名，这些机制消除了命名冲突的隐患。运算符重载使运算符的操作数从基本的数据类型扩充到类对象，提高了软件的可理解性和可读性。

6）C++提供了虚函数来实现程序设计的多态性，使得基类的函数接口可以在其派生类中继承重用。

7）C++采用模板，使得函数和类的定义可适用于多种数据类型，使程序设计标准化和通用化，提高了软件开发的效率。

C++语言的缺点表现在以下两个方面。

1）C++语言比C语言要复杂得多，学习C++语言要比学习C语言难得多。在编写小型程序时，C++可能还不如C语言简捷。

2）使用C++语言本身并不会给程序的设计与开发带来任何优点，C++语言必须与面向对象程序设计方法相结合，在程序中大量使用C++语言提供的面向对象的特征和功能，才能体现C++的特点。否则编写出来的程序的执行效率可能会与使用结构化程序设计语言一样，甚至更糟糕。

1.4 C++语言初步

1.4.1 C++语言的词法

C++语言的词法包括标识符、分隔符、关键字、注释符和运算符等。本节先介绍C++的标识符、分隔符和关键字。注释符和运算符的表示及使用方法将在后续章节中讲解。

C++的标识符是由程序员自定义的，用于表示变量名、类型名和字符常量等程序实体的一个单词。标识符只能以字母或者下划线“_”开头，而不能以数字或其他符号开头。标识符的其余字符可以是英文大小写26个字母、数字0～9和下划线。分隔符有空格符、逗号（,）、分号（;）、冒号（:）、单引号（'）、双引号（"）、中括号（[]）和大括号（{}）等。关键字即系统保留字，程序中的其他标识符不能与关键字同名。下面列出的是C语言和C++语言共有的关键字：

```
auto      break     case      char      const     continue  default   do        double
else      enum      extern    float     for       goto      if        int       long
register  return    short     signed    sizeof    static    struct    switch    typedef
union     unsigned  void      volatile  while
```

以下是C++增加的不属于C语言的31个关键字：

```
asm       bool      catch     const_cast  class     delete      dynamic_cast  explicit
export    false     friend    inline      mutable   namespace   new           operator
private   protected public    reinterpret_cast      static_cast template    this
throw     true      try       typeid      typename  using       virtual       wchar_t
```

1.4.2 C++程序的框架结构

1. 结构化程序设计的C++程序结构

C++既支持面向对象的程序设计，也支持面向过程按功能划分模块的结构化程序设计。按

照结构化程序设计的方法，一个 C++ 的程序由函数来组成，其基本的框架结构和组织形式为：

- 预处理程序命令
- 全局变量的定义和说明
- 用户自定义函数
- 主函数 main

程序 1-1 是一个简单的 C++ 程序。程序首先输出“This is my first C++ program”，然后提示用户输入一个角度值，分别计算该角度的正弦和余弦值并输出结果。

【程序 1-1】

```
/* 预处理程序命令*/
#include <iostream>
#include <math.h>
using namespace std;

#define  PI  3.1415926

/* 用户自定义函数*/
void showInfo()
{
    cout << "This is my first C++ program" << endl;
}

/* 主函数*/
void main()
{
double  sinval,cosval,ang_deg,ang_rad;

    showInfo();

    cout << "Please enter an angle in degree:";
    cin >> ang_deg;

    ang_rad = PI* ang_deg /180.0;
    sinval = sin(ang_rad);
    cosval = cos(ang_rad);

    cout << "The sine value of" << ang_deg << "is" << sinval << endl;
    cout << "The cosine value of" << ang_deg << "is" << cosval << endl;
}
```

程序运行结果为：

```
This is my first C++ program
Please enter an angle in degree:60
The sine value of 60 is 0.866025
The cosine value of 60 is 0.5
```

以下将结合程序 1-1 对构成 C++ 程序的一些基本概念加以简单的说明，目的是使读者对 C++ 语言和程序有一个大概的认识，这些概念的详细的论述将在后续章节中逐步介绍。

（1）预处理程序命令

预处理程序命令是以“#”开头并占用一整行的命令，预处理程序命令位于程序的开始供编译程序的预处理器处理。源程序在编译过程中，首先由编译程序的预处理器对源程序中的预处理程序命令进行处理，处理结果再传送给编译程序进行编译。使用预处理程序命令的主要目的是减少源程序代码。C++ 提供了三类预处理命令：宏定义命令#define、文件包含命令#include 和条件编译命令#ifdef…#endif。下面主要介绍文件包含命令。

```
#include <iostream>
#include <math.h>
```

```
using namespace std;
```

上述代码的第一行是一个文件包含命令，指示编译程序的预处理器在编译程序所在目录将iostream头文件插进来。iostream文件主要包含了完成屏幕输出和键盘输入的标准输入cin对象和标准输出cout对象的说明。第二行将math.h头文件包含进来，该文件说明了计算正弦和余弦等三角函数值的C++定义的标准库函数。第三行指示编译程序去识别由头文件引入的C++标准库，这是一个C++为了避免标识符命名冲突而引入的称之为名空间的特征。两个不同名空间的标识符不会发生冲突，要完整地访问一个标识符必须采用“名空间名::标识符名”的形式，例如cout对象名应用std::cout来访问。C++提供的所有标准库函数都是在std名空间定义的，using namespace std语句的功能是告诉编译程序当前的全局名空间为std名空间。这样在程序的后续代码中若用到C++标准头文件说明的对象或函数，就可以不必给出std名空间名，例如直接写成sin(ang_rad)而不必用std::sin(ang_rad)就可调用C++计算正弦的标准库函数。名空间只在较高版本的C++编译器中支持，在早期的C++中，头文件的扩展名必须为.h，故

```
#include < iostream >
using namespace std;
```

要写成：

```
#include < iostream. h >
```

程序员也可以定义自己的名空间，例如：

```
#include < iostream >
namespace yournamespace
{
    double angle;
}
namespace mynamespace
{
    using yournamespace::angle;          //将 yournamespace 中的 angle 输出到 mynamespace 名空间
}
using namespace mynamespace;             //将 mynamespace 输出到全局名空间
void main()
{
    std::cin >>mynamespace::angle;       //cin 和 cout 都是 std 名空间的对象
    std::cout <<angle;                   //angle 是 yournamespace 的对象,已被输出到全局名空间
}
```

上述代码中建立了yournamespace和mynamespace两个名空间，加上缺省的std名空间共有3个名空间。yournamespace名空间中的angle被using语句输出到mynamespace名空间，而mynamespace名空间又被using语句输出到全局名空间，故angle和mynamespace::angle都是指yournamespace名空间中定义的angle。由于std名空间未被输出到全局名空间，要访问定义在std名空间的cin和cout对象，则必须采用std::cin和std::cout的形式。

用户自己定义的非标准库中的函数可在用户自己编写的头文件中加以描述，并存放在用户自己指定的目录里，此时用#include指令将其包含在源程序中时，头文件的文件名必须用双引号括起来。例如：

```
#include"c: \mydir \mywork.h"
```

这条命令指示编译程序的预处理器将位于c:\mydir目录下的mywork.h文件的内容插入到源程序文件中。

宏定义命令#define用来定义宏，它的第一个参数指出被替换的文本，第二个参数指出用作替换的文本。当编译程序的预处理器在后续源代码中遇到了与第一个命令参数相对应的符号时，预处理器就用第二个命令参数替换该符号。

例如，程序1-1中的

```
#define  PI  3.1415926
```

当编译程序的预处理器对代码

```
ang_rad = PI* ang_deg/180.0;
```

在编译的初始阶段进行预处理时，将下列一行作为结果传给编译程序：

```
ang_rad = 3.1415926* ang_deg/180.0;
```

宏替换由于不进行参数类型校验和栈空间分配，所以其运行速度通常比函数快，但是带来的问题是代码不直观，不好理解。C++ 语言用内联函数取代了宏定义命令，在保证程序运行速度的前提下，提高了程序的可读性和可理解性。在定义常量标识符时，C++ 也用能说明常量类型的 const 语句代替了纯文本替换的#define 命令。

（2）全局变量的定义和说明

在计算机中，数据和程序都是存放在计算机的内存里的。在编写程序对数据进行处理时，通常还要在存储器里保存计算的中间结果以便必要时再读出来作为下一步计算的数据。程序员不关心输入输出数据和中间结果存放在内存的哪个地址单元里，也不希望在访问数据时要给出存放这些数据的内存物理地址。为此，C++ 语言用一个变量来表示一个存放数据，具有一定物理地址的内存单元，变量的值即该内存单元存放的数值。用什么样的标识符来表示变量，也即怎样表示存放数据的内存单元，以及这些内存单元存放什么类型的数值，是程序员要考虑的问题。变量标识符怎样与物理内存地址联系起来，这是编译程序要解决的问题，程序员不必理会。

C++ 是一个类型检查的语言，即标识符用作某个变量或函数的名称在程序中被引用之前必须被定义或说明。变量的定义就是指明标识符所表示的变量的数据类型和存储类型，并为该变量在内存的相应区域分配适当的存储空间。不同类型的变量所占用的内存空间的大小是不同的。变量的说明是为了建立变量名与变量类型之间的对应关系，变量说明通常表示该变量的定义将随后在程序中出现。变量的说明（declaration）和变量的定义（definition）是两个不同的概念，变量在说明时并不分配内存，变量的定义是唯一的，而变量的说明可以重复出现多次。

例如，在语句

```
sinval = sin(ang_rad);
```

中，出现了三个标识符：sinval、sin 和 ang_rad。标识符 sinval 和 ang_rad 通过变量的定义语句

```
double  sinval,cosval,ang_deg,ang_rad;
```

在函数内被定义为两个双精度型（double）的变量，它们在内存中各占 8 个字节。sinval 用来存放正弦函数最终的结果，ang_rad 存放输入角度的弧度值，是一个中间结果。

标识符 sin 在程序 1-1 中未被显式定义，但在由#include 预处理程序命令所包含的 math 文件中，可以找到对 sin 标识符的说明。从其说明的格式可以判断 sin 表示的是一个函数名。sin 函数的定义在哪里呢？凡是在 C++ 编译程序提供的头文件中说明的函数，其定义以目标代码的形式存放在由 C++ 提供的标准库函数中！

（3）用户自定义函数

按照结构化程序设计的观点，程序按功能划分成模块。函数是实现模块的基本代码单元，每个函数实现一个基本功能。复杂的功能可以通过嵌套调用实现简单功能的函数来实现。所以按照结构化程序设计的思想，程序是由函数组成的，程序的主要内容就是函数。函数分为用户自定义的函数和 C++ 编译程序提供的标准库函数两大类。库函数是一些常用的可供程序员直接调用的函数，库函数的定义（实现）事先已经做好并经编译以目标代码的形式组成标准函数库。库函数在 C++ 编译程序提供的标准头文件中有加以说明。程序员要调用库函数，只需用#include 预处理程序命令将包含相应库函数说明的标准头文件加入到源程序即可。用户自定义函数是用户自己编写的能实现特定应用程序功能的函数。如果用户自定义函数的调用出现在其定义之前，则需在调用之前对该函数进行说明。用户自定义函数可以有多个，一个程序可以由多个源程序文件组成，一个源程序文件里可出现多个函数。在程序 1-1 中，showInfo 为用户自定义函数，同时还出现了对标准库函数 sin 和 cos 的调用。