高等工科院校电子、信息类教材

现代程序设计

C++与数据结构
面向对象的方法与实现

Advanced Programming Based on C++ and Data Structure

沈晴霓　聂　青　苏京霞　编著

北京理工大学出版社
BEIJING INSTITUTE OF TECHNOLOGY PRESS

现代程序设计

——C++与数据结构面向对象的方法与实现

沈晴霓　聂　青　苏京霞　编著

北京理工大学出版社
·北 京·

图书在版编目(CIP)数据

现代程序设计—C++与数据结构面向对象的方法与实现/沈晴霓,聂青,苏京霞编著.—北京:北京理工大学出版社,2002.8
ISBN 7-81045-962-7

Ⅰ.现… Ⅱ.①沈… ②聂… ③苏… Ⅲ.C语言-程序设计 Ⅳ.TP312

中国版本图书馆CIP数据核字(2002)第038053号

出版发行 /北京理工大学出版社
社　　址 /北京市海淀区中关村南大街5号
邮　　编 /100081
电　　话 /(010)68914775(办公室)　68912824(发行部)
网　　址 /http://www.bitpress.com.cn
电子邮箱 /chiefedit@bitpress.com.cn
经　　销 /全国各地新华书店
印　　刷 /北京房山先锋印刷厂
装　　订 /天津高村装订厂
开　　本 /787毫米×1092毫米　1/16
印　　张 /22.75
字　　数 /545千字
版　　次 /2002年8月第1版　2002年8月第1次印刷
印　　数 /1-4000册　　责任校对 /郑兴玉
定　　价 /34.00元　　责任印制 /李绍英

前　言

计算机与网络技术的高速发展，特别是面向对象技术的出现，使得C++的软件开发得到了迅速普及。这种面向对象的技术，不仅符合人们思维习惯，而且用它开发出来的软件可靠性很强。数据结构又是计算机学科中的一门核心课程，就软件产品而言，关键是建立合理的软件体系结构和程序结构。如果脱离数据结构，只是进行纯粹的C++语法功能介绍，就不能形成体系的结构和思路，不符合现代程序设计的要求。作为非计算机专业的大学生，要提高程序设计和软件开发能力，就必须在学习了《C 程序设计》的基础上，进一步学习、掌握面向对象的 C++程序设计和有效组织各种数据在计算机中的存储、传递和转换的方法。因此，用不多的学时开设一门既介绍面向对象的程序设计方法，又将系统的数据结构思路融合到面向对象的方法上来的课程，对于非计算机专业的大学生来说，是很有必要的，符合形势发展的需要。

本教材《现代程序设计》就是通过系统地介绍面向对象的 C++程序设计方法和数据结构的 C++描述，并通过具体的实例分析，使学生真正掌握数据结构应用于面向对象程序设计的方法，从而更好地从事软件开发和工程应用。本教程在北京理工大学电子工程系作为教学改革项目开设已经 3 年，深受学生欢迎；书中所有例题，也是教师们在多年的教学过程中，精心挑选、编写和调试过的，具有一定的代表性。

本书可以作为电子专业、通信专业和其他非计算机专业的高校本科生、成人教育学院的本科生以及计算机专业的大专生选用的教材，也可供程序设计工程人员和爱好者自学或参考使用。

全书内容分三部分，共十八章。

第一部分是面向对象的程序设计基础，包括第一章到第六章。

第一章引入面向对象的思想、概念和基本特征，并讨论了一些从 C 过渡到 C++的语法变化。第二章主要介绍 C++类的定义及其封装性的体现，并重点介绍了构造函数和析构函数的作用和定义方式。第三章主要介绍 C++中的几种常用语法和一些特殊用途，包括函数和运算符的重载、友员和引用的定义和使用。第四章主要介绍继承及派生类的定义和使用，包括单继承和多继承、继承方式及基类数据在派生类中访问权限的变化情况的介绍和应用。引入了虚基类的概念并介绍了它的用途。第五章主要介绍多态性在 C++中的体现，并重点介绍了运行时的多态性——虚函数的引入、定义和使用，以及纯虚函数和抽象类的概念。第六章主要介绍了引入模板的意义、定义和使用模板函数和模板类的方法及模板函数重载和模板类派生的方式。

第二部分是数据结构的抽象类及其实现，包括第七章到第十五章。

第七章主要引入了数据结构的定义及抽象数据类型的概念，而且讨论了数据结构所研究的三个方面（逻辑结构、存储结构和算法）的内容，还简要介绍了数据结构的分类及其抽象层次。第八章到第十二章依次介绍了线性表、数组、串、堆栈和队列的概念、抽象类描述及不同存储表示的各种类的具体定义、实现和应用。第十三章和第十四章依次介绍了树和图的

基本概念和多种存储表示，并给出了典型表示的抽象类定义、实现和应用。第十五章主要介绍了索引与散列表的概念、散列函数的设计方法及解决冲突的多种策略。

第三部分是面向对象程序设计实例，包括第十六章到第十八章。

第十六章主要概述 C++面向对象程序设计的一般规范，并介绍了一些常用算法的设计思想及面向对象程序设计的一般过程。第十七章和第十八章是两个综合实例，分别介绍图的着色问题和货郎担问题的描述、模型的建立、算法分析及其程序实现。

本书的第二章到第六章由苏京霞老师编写，第七章~第十二章由聂青老师编写，第一章和第十三章~第十八章由沈晴霓老师编写。特别感谢罗森林副教授的帮助和指导，他为我们的初稿提出了许多宝贵的意见。在整个教材的编写过程中还得到了苏广川教授的大力支持和细心指导，在此也表示我们的衷心谢意。

由于时间仓促和水平有限，难免有一些错误和不足之处，恳请读者批评指正。

作　者

2002 年 4 月 23 日

目　录

第一部分　面向对象的C++程序设计基础

第一章　面向对象的概述 …… (1)
1.1　面向对象的思想 …… (1)
1.1.1　面向对象程序设计 …… (1)
1.1.2　面向对象的语言 …… (1)
1.2　面向对象的基本概念 …… (2)
1.2.1　对象 …… (2)
1.2.2　消息 …… (3)
1.2.3　类 …… (3)
1.3　面向对象的基本特性 …… (3)
1.3.1　封装性 …… (3)
1.3.2　继承性 …… (4)
1.3.3　多态性 …… (4)
1.4　从C过渡到C++ …… (4)
1.4.1　函数原型说明 …… (4)
1.4.2　变量的说明 …… (5)
1.4.3　输入和输出 …… (5)
1.4.4　const说明符 …… (6)
1.4.5　void类型 …… (6)
习题 …… (6)
第二章　类及其对象的封装性 …… (7)
2.1　类的定义 …… (7)
2.2　类的成员函数 …… (8)
2.2.1　成员函数的定义 …… (8)
2.2.2　内置成员函数 …… (8)
2.3　对象的定义 …… (9)
2.3.1　类与对象 …… (9)
2.3.2　定义对象 …… (9)
2.4　构造函数和析构函数 …… (11)
2.4.1　构造函数 …… (11)
2.4.2　析构函数 …… (18)
2.4.3　动态存储 …… (21)
2.5　C++中的封装性 …… (24)
习题 …… (27)

第三章　友员、重载和引用 …… (28)
3.1　友　员 …… (28)
3.1.1　友员的定义 …… (29)
3.1.2　友员函数 …… (29)
3.1.3　友员成员 …… (30)
3.1.4　友员类 …… (31)
3.2　重载 …… (33)
3.2.1　函数重载 …… (33)
3.2.2　运算符重载 …… (36)
3.3　引用 …… (46)
3.3.1　引用的概念 …… (46)
3.3.2　引用的应用 …… (47)
3.3.3　引用传递参数 …… (48)
3.3.4　引用返回值 …… (49)
3.3.5　常引用 …… (52)
习题 …… (53)
第四章　继承与派生 …… (54)
4.1　继承与派生 …… (54)
4.2　派生类 …… (54)
4.2.1　派生类生成过程 …… (55)
4.2.2　派生类定义 …… (55)
4.2.3　访问权限 …… (56)
4.2.4　派生类的构造函数和析构函数 …… (61)
4.3　多继承 …… (63)
4.3.1　多继承的概念 …… (63)
4.3.2　多继承的定义 …… (64)
4.3.3　多继承的构造函数与析构函数 …… (65)
4.4　虚基类 …… (71)
4.4.1　问题的提出 …… (71)
4.4.2　虚基类的概念 …… (72)
4.4.3　虚基类的初始化 …… (74)
习题 …… (77)
第五章　虚函数与多态性 …… (78)
5.1　多态性 …… (78)
5.1.1　多态性概述 …… (78)
5.1.2　编译时的多态性 …… (79)
5.1.3　运行时的多态性 …… (80)
5.2　虚函数 …… (80)
5.2.1　对象指针 …… (80)

5.2.2 虚函数的定义及使用 …… (82)
5.3 抽象类 …… (87)
5.3.1 纯虚函数与抽象类 …… (87)
5.3.2 纯虚函数多态性的体现 …… (89)
习题 …… (92)
第六章 模板 …… (94)
6.1 模板的概念 …… (94)
6.2 函数模板 …… (95)
6.2.1 函数模板和模板函数 …… (95)
6.2.2 重载模板函数 …… (97)
6.3 类模板 …… (98)
6.3.1 类模板与模板类的概念 …… (98)
6.3.2 类模板的派生 …… (101)
习题 …… (104)

第二部分 数据结构的C++抽象类及其实现

第七章 绪论 …… (105)
7.1 数据结构的基本概念 …… (105)
7.2 抽象数据类型的面向对象概念 …… (107)
7.2.1 数据类型 …… (107)
7.2.2 数据抽象与抽象数据类型 …… (107)
7.3 算法和算法分析 …… (108)
7.3.1 算法 …… (108)
7.3.2 算法设计的要求 …… (108)
7.3.3 算法效率的度量 …… (109)
7.4 数据结构的抽象层次 …… (111)
习题 …… (112)
第八章 线性表 …… (113)
8.1 线性表的定义 …… (113)
8.1.1 线性表的逻辑结构 …… (113)
8.1.2 线性表的顺序存储表示 …… (113)
8.1.3 线性表的链式存储表示 …… (114)
8.2 抽象链表类 …… (115)
8.2.1 抽象链表类的定义 …… (115)
8.2.2 抽象链表中典型成员函数的实现 …… (116)
8.3 单链表 …… (118)
8.3.1 单链表的定义 …… (118)
8.3.2 单链表的常用操作 …… (120)
8.3.3 单链表类的定义 …… (120)

8.3.4 单链表的常用成员函数的实现 …… (122)
8.3.5 一元多项式加法 …… (127)
8.4 循环链表 …… (131)
8.4.1 循环链表的定义 …… (131)
8.4.2 循环链表类的定义 …… (131)
8.4.3 循环链表常用函数的实现 …… (132)
8.4.4 约瑟夫(Josephu)问题 …… (139)
8.5 双向链表 …… (139)
8.5.1 双向链表的定义 …… (139)
8.5.2 双向链表类的定义 …… (140)
8.5.3 双向链表的常用成员函数的实现 …… (141)
习题 …… (147)
第九章 数组 …… (149)
9.1 数组的定义 …… (149)
9.1.1 数组的逻辑结构 …… (149)
9.1.2 数组的存储结构 …… (149)
9.1.3 数组的常用操作 …… (150)
9.2 抽象数组类 …… (150)
9.2.1 抽象数组类的定义 …… (150)
9.2.2 抽象数组类常用函数的实现 …… (151)
9.3 数组类 …… (154)
9.3.1 数组类的定义 …… (154)
9.3.2 数组类常用函数的实现 …… (155)
9.4 一元多项式加法 …… (160)
习题 …… (164)
第十章 串 …… (165)
10.1 串的概念 …… (165)
10.1.1 串的定义 …… (165)
10.1.2 串的基本术语 …… (165)
10.1.3 串的存储表示和实现 …… (166)
10.1.4 串的基本运算 …… (166)
10.2 字符串类 …… (167)
10.2.1 字符串类定义 …… (167)
10.2.2 字符串类中常用成员函数的实现 …… (168)
习题 …… (179)
第十一章 堆栈 …… (181)
11.1 堆栈的概念及其运算 …… (181)
11.1.1 堆栈的定义 …… (181)
11.1.2 堆栈的有关运算 …… (181)

11.2 抽象栈类 …… (182)
11.3 顺序栈 …… (183)
11.3.1 顺序栈的定义 …… (183)
11.3.2 顺序栈类的定义 …… (183)
11.3.3 顺序栈类中典型成员函数的实现 …… (184)
11.3.4 多栈共享空间问题 …… (187)
11.4 链式栈 …… (189)
11.4.1 链式栈的定义 …… (189)
11.4.2 链式栈类的定义 …… (190)
11.4.3 链式栈中典型成员函数的实现 …… (191)
11.5 堆栈的应用举例 …… (195)
11.5.1 数制转换 …… (195)
11.5.2 一个趣味游戏——迷宫问题 …… (196)
习题 …… (201)
第十二章 队列 …… (202)
12.1 队列的定义及其运算 …… (202)
12.1.1 队列的定义 …… (202)
12.1.2 队列的有关运算 …… (202)
12.2 抽象队列类 …… (202)
12.3 顺序队列 …… (203)
12.3.1 队列的顺序存储结构——循环队列 …… (203)
12.3.2 顺序循环队列类的定义 …… (206)
12.3.3 循环队列中常用成员函数的实现 …… (207)
12.4 链式队列 …… (209)
12.4.1 链式队列的定义 …… (209)
12.4.2 链式队列类的定义 …… (209)
12.4.3 链式队列中常用成员函数的实现 …… (210)
12.4.4 链式队列的应用举例 …… (213)
12.5 优先级队列 …… (215)
12.5.1 优先级队列的定义 …… (215)
12.5.2 优先队列类的定义 …… (215)
12.5.3 优先队列中常用成员函数的实现 …… (217)
习题 …… (220)
第十三章 树 …… (222)
13.1 基本概念 …… (222)
13.1.1 树 …… (222)
13.1.2 二叉树 …… (224)
13.1.3 树与森林 …… (228)
13.2 二叉树的抽象类和树的抽象类 …… (232)

13.2.1 二叉树的抽象类 …… (232)
13.2.2 树的抽象类 …… (238)
13.3 二叉树的遍历和树的遍历 …… (246)
13.3.1 二叉树的遍历 …… (246)
13.3.2 树的遍历 …… (252)
13.4 二叉排序树 …… (255)
13.5 二叉树的计数 …… (260)
13.6 Huffman 树 …… (262)
13.6.1 Huffman 树 …… (262)
13.6.2 Huffman 编码及其用法 …… (264)
习题 …… (266)
第十四章 图 …… (268)
14.1 图的基本概念 …… (268)
14.1.1 图的定义 …… (268)
14.1.2 图的术语 …… (269)
14.1.3 图的基本操作 …… (271)
14.1.4 图的存储表示 …… (271)
14.2 图的抽象类 …… (276)
14.2.1 图的邻接矩阵类 …… (276)
14.2.2 图的邻接表类 …… (282)
14.3 图的遍历 …… (289)
14.3.1 深度优先搜索 DFS …… (289)
14.3.2 广度(或宽度)优先搜索 BFS …… (290)
14.4 图的连通性与最小生成树 …… (291)
14.4.1 无向图的连通分量和生成树 …… (291)
14.4.2 最小生成树 …… (291)
14.4.3 关节点和重连通分量 …… (297)
14.5 最短路径 …… (299)
14.5.1 图结点的可达性 …… (300)
14.5.2 从某个源点到其余各顶点的最短路径 …… (301)
14.5.3 每一对顶点之间的最短路径 …… (303)
14.6 活动网络 …… (305)
14.6.1 用顶点表示活动的网络 …… (305)
14.6.2 用边表示活动的网络 …… (306)
习题 …… (308)
第十五章 索引与散列结构 …… (310)
15.1 散列表与散列方法 …… (310)
15.2 散列函数的构造方法 …… (311)
15.2.1 直接定址法 …… (312)

15.2.2 数字分析法 …… (312)
15.2.3 平均取中法 …… (312)
15.2.4 折叠法 …… (313)
15.2.5 除留余数法 …… (314)
15.2.6 随机数法 …… (314)
15.3 冲突解决策略 …… (315)
15.3.1 开散列方法 …… (315)
15.3.2 闭散列方法 …… (315)
15.4 散列表的查找 …… (318)
习题 …… (319)

第三部分 面向对象的C++程序设计实例

第十六章 面向对象的程序设计 …… (321)
16.1 多范型的C++程序设计 …… (321)
16.1.1 程序设计范型 …… (322)
16.1.2 支持数据抽象的C++ …… (323)
16.1.3 支持面向对象的C++ …… (326)
16.2 算法设计思想 …… (326)
16.3 面向对象的程序设计过程 …… (327)
16.3.1 分析阶段 …… (327)
16.3.2 设计阶段 …… (327)
16.3.3 编码阶段 …… (327)
16.3.4 测试和维护阶段 …… (328)
习题 …… (328)
第十七章 图的着色问题 …… (329)
17.1 图的m着色最优化问题 …… (329)
17.2 问题的模型建立 …… (330)
17.3 图的着色方案及算法实现 …… (331)
17.3.1 贪心法 …… (331)
17.3.2 递归回溯法 …… (335)
习题 …… (337)
第十八章 货郎担问题 …… (338)
18.1 货郎担问题 …… (338)
18.2 问题的模型建立 …… (338)
18.3 货郎担问题的求解方案与算法实现 …… (339)
18.3.1 最小成本LC-分支限界检索方案 …… (339)
18.3.2 算法实现 …… (342)
习题 …… (349)
参考文献 …… (350)

第一部分　面向对象的 C++程序设计基础

第一章　面向对象的概述

本章主要介绍面向对象的思想及面向对象的几个基本概念：对象、消息和类，还介绍了面向对象程序设计的三个基本特征：封装性、继承性和多态性。为了更好地介绍 C++，引入了一些有关 C++对 C 的改进和增强的知识，包括函数原型、变量定义、const 说明符、void 类型和流输入输出等概念。

1.1　面向对象的思想

早在 1980 年，C++由贝尔实验室的 Bjarne Stroustrup 创建。C++是在 C 语言的基础上，增加了面向对象程序设计的功能，它适合编制复杂的大型软件系统。

1.1.1　面向对象程序设计

在面向程序设计之前，占主流的是结构程序化设计，也就是面向过程的程序设计。例如：Fortran、Pascal 及最初的 C 语言等都是面向过程的程序设计。结构化程序设计的主要思想是：自顶向下，逐步求精。它的程序结构是按功能划分为若干个基本模块，各模块这间的关系尽可能简单，各模块的功能相对独立，这些模块联合形成一个树状结构。虽然结构化程序设计方法有很多优点，但它只是一种面向过程的程序设计方法。它把数据和处理数据的操作分离为相互独立的实体，当数据结构发生变化时，所有相关的处理数据的操作都要进行相应的修改。因此不适合编制复杂的大型软件系统。

面向对象的程序设计方法，是将数据及对数据的操作集合在一起，作为一个相互依存、不可分离的整体即对象。对同类型对象抽象出其共性，形成类。类中数据，通过本类的方法进行处理。类通过外部接口与外界联系，对象与对象之间通过消息进行通讯。面向对象程序设计方法的优点是代码可重用性好，可维护性好。

1.1.2　面向对象的语言

面向对象程序设计语言必须支持抽象数据类型和继承性。面向对象的程序设计语言经历了一个漫长的发展过程。例如，LISP 语言、Simula67 语言及 Smalltalk 语言等等，它们都或多或少地引入了面向对象的一些概念，如数据抽象的概念、类机构和继承性机制。如今，应用最广泛的面向对象程序设计语言是在 C 语言基础上扩充出来的 C++语言。因为 C++对 C 的向后兼容，所以很多已有的程序稍加修改就可以重用，许多有效的算法也可以重新利用。C++

是一种混合型的面向对象程序设计语言，由于它的出现，才使得面向对象的各种语言越来越多地得到重视及广泛应用。

1.2 面向对象的基本概念

在深入了解和学习面向对象程序设计之前，有必要先简单介绍一下面向对象方法中的几个基本概念。

1.2.1 对象

对象是C++面向对象程序设计的核心，正确地认识和定义对象，对今后掌握面向对象的理论有很大帮助。

一、对象的定义

在面向对象程序设计中的对象包含两个含义，即对象的属性和它的行为。属性一般为对象的自然属性即自身状态，如人的性别、年龄、身高及体重等。行为则指对象的功能，如人的一些特长、技能等。对象是其自身所具有的特征及可以对这些状态施加的操作结合在一起构成的独立实体，它具有以下特性：

(1) 有一个名字以区别于其他对象；

(2) 有一个状态用来描述它的一些特性；

(3) 有一组操作，每一个操作决定对象的一种功能或行为；

(4) 对象的操作可分两类：一类是自身所承受的操作，另一类是施加于其他对象的操作。

例如有一个人叫李玉，身高：1.65m，体重：50 kg，可以教高等数学课，会程序设计。下面是这个对象的特征描述：

对象名：李玉

对象的状态：

身高：1.65m

体重：50 kg

对象的功能：

回答身高
回答体重 } 自身所承受的操作

教高等数学课
程序设计 } 施加于其他对象的操作

二、对象的特性

对象具有以下三个特性：

(1) 模块独立性：利用封装技术将对象的数据隐藏在模块内部，当你使用时只需了解它具有的功能就行了，不必知道功能是如何实现的。因此，外界的变化不会影响模块内部状态。各模块可独立为系统所组合选用，也可被程序员重用。

(2) 动态连接：可以通过消息激活机制，把对象之间动态的联系连接在一起，使整个机体运转起来。

(3) 易维护性：由于对象的数据和操作都封装在模块内部，所以对它们的修改及完善都

限制在模块内部，不会涉及到对象外部。这就使得对象和整个系统变得非常容易维护。

1.2.2 消息

在C++中，消息的具体表现为函数。它是对象之间相互请求或相互协作的途径，是要求某个对象执行其中某个功能操作的规格说明。对象之间的联系只能通过相互传递消息来进行。对象只有在接收到消息时，才会激活有关的对象代码，“知道”如何去操作它的私有数据和完成所要求的功能。

消息具有三个性质：

(1) 同一对象可接收不同形式的多个消息，并产生不同的响应；

(2) 相同形式的消息可以传送给不同类型的对象，所产生的响应可能截然不同；

(3) 消息的发送可以不考虑具体的接收者，对象可以响应消息，也可以不响应消息。

在面向对象系统中，为了完成某个事件，有时需要向同一个对象发送多个消息或者向不同的对象发送多个消息，我们称这多个消息为消息序列。对于一个对象来说，其中由外界对象直接向它发送的消息，称为公有消息；由它自己向本身发送的消息，称为私有消息。

针对某个特定对象，根据消息的功能来分，大体有以下三种类型：

(1) 可以返回对象的内部状态的消息；

(2) 可以改变对象的内部状态的消息；

(3) 可以完成一些特定操作，并改变系统状态的消息。

1.2.3 类

类是面向对象系统中最重要的概念，面向对象的程序设计主要归结为类的建立和使用。在实际工作中，单纯描述一个对象是没有意义的，往往是描述一个类的工作。俗话说“物以类聚”，我们把具有相同特性的物质归为一个类。类的确定采用归纳法来完成：首先对所遇到的若干同类对象进行分析，归纳出它们的共同特性，包括数据特性和行为特性，建立一个类。

C++中的类是对所要处理问题中若干对象的抽象描述，它对逻辑上相关的函数和数据进行封装。类是对多个对象的抽象，而对象是类的实例。如：王洋是一个教师。教师是一个类，属于抽象的概念；而王洋是教师类的一个具体对象，即教师类的一个实例。

1.3 面向对象的基本特性

在面向对象程序设计中，最重要的特性是封装性、继承性和多态性。

1.3.1 封装性

类是对具有相同特征的客观对象的抽象描述，它将抽象出来的数据和操作行为封装在类中。在类中将一部分行为作为对外部的接口，将数据和其他行为进行有效的隐藏。外界只能通过外部接口才能访问封装在类中的数据，这就可以达到对数据访问权限的合理控制。

封装具有以下几个特性：

(1) 对象所有的私有数据和内部程序对外界是不可直接访问的；

(2) 具有外部接口，这个接口描述了对象之间相互请求和响应的消息格式和功能；

(3) 对象内部的代码实现是隐藏的，即外部对象不能直接修改有关的代码细节。

1.3.2 继承性

继承是面向对象系统的另一个重要的特性。类和类之间有时是相互独立的，有时会出现一些相似的特征。继承所表达的正是这种类与类之间的相交关系。它使得某类的对象可以继承另一个类的特征和能力。

C++为我们提供了类的继承机制，允许在保持原有类的特性的基础上，为其继承类进行更具体、更详细的类的定义。就是说，可以定义一个包含公共成员的基类，通过继承，从基类中派生出其他类，为基类增加新的操作和成员，还可以改写基类的部分内容。

继承分单继承和多继承，从一个基类派生出新类，称为单继承；从多个基类派生出新类，称为多继承。由某个类派生出所需的任意多的类，或者类之间通过单继承和多继承派生出多个类，这就形成类的层次关系。

在面向对象的系统中，引入继承机制后，主要有以下优点：

(1) 能清晰体现相关类之间的层次关系；

(2) 大大增加了程序的重用性，减少了重复编码；

(3) 通过增强一致性来减少模块之间的接口和界面，提高了程序的可维护性；

(4) 继承还可以实现对类的扩充，为程序的扩展和可用性提供了有效的方法。

1.3.3 多态性

C++中的多态性，是指同一个消息形式可以根据发送消息对象的不同，采用不同的行为方式。C++语言支持两种多态性：

- 编译时的多态性，通过重载来实现

C++允许为已有的函数和运算符重新赋予新的含义，就是说具有相同名字的函数或运算符在不同的场合可以表现出不同的行为，即函数和运算符重载。在定义函数重载时，函数名要相同，但函数所带的参数个数或参数类型必须有所区别，否则就会出现二义性。

- 运行的多态性，通过虚函数来实现

C++的虚函数使得用户可以在同个类族（基类及其各级派生类）中使用同名函数的多个版本，每个版本均属于同个类族中的不同的类。究竟使用哪个版本，需要在运行中决定。虚函数中的各个版本中，其返回值、函数参数的个数和类型必须一致。

1.4 从C过渡到C++

C++是目前和将来一段时间内使用最广泛的一种程序设计语言，比较全面地支持面向对象的设计思想，本书第一部分就是要根据面向对象的特征来介绍C++编程知识。在此之前，先介绍C++对C的一些修正和增强的知识，以便大家更好地理解和掌握C++。

1.4.1 函数原型说明

C++摒弃了C语言对函数原型随意简化的方式。实际上，这种随意简化是许多C程序错误的根源，这种对简化的放纵也助长了许多C编程人员不良编程习惯的形成。C++语言要求

编程人员为函数提供完全的原型，包括所有参数的类型和返回值类型，这样做的目的是：

(1) 编译系统可以对函数的定义和使用之间的一致性进行严格的检查，避免函数使用方法或类型的错误，如实际参数的个数或类型不对等。

(2) 根据函数定义的参数类型和返回值类型，以及函数调用的实际情况，确定和生成必要的类型转换，以保证函数使用的正确性。

例如：求两个整数的较大者，并返回其值的一个函数，用 C 语言一般声明成：

```
int max();
```

它只说明函数名和函数返回值类型，但这对 C++是不合适的！

C++中函数原型应该包括：函数的返回值类型、函数名和函数的各个参数的类型。对以上函数须声明为：

```
int max(int,int);
```

希望大家在 C++的程序设计中使用完全的函数原型，养成良好的习惯。

1.4.2 变量的说明

在 C 语言中，变量的定义必须出现在函数或复合语句的最前部，在正常执行语句后面不可能再定义变量。如：

```
int x,y;
x=20;  //执行语句
int z;   //出错!
```

而在 C++程序中，变量的定义可以出现在任何位置，例如，人们常用下面的方式写循环语句：

```
for(int i=0;i<n;i++)
    for(int j=0;j<i;j++) ...
```

1.4.3 输入和输出

C++语言除了保留了 C 语言标准库中的各种输入和输出函数外，还提供了一套新的输入和输出函数——流式输入输出。使用系统的流式输入输出函数应当引入头文件"iostream.h"。例如：

```
# include <io stream.h>
int x, y;
float z ;
cin>>x>>y>>z;  //从标准终端输入数据
char *str="Hello,How are you!";
cout<<"Please output the info of the string: "<<str<<"\n";  //向屏幕输出字符串
```

其中 “>>”（“<<”）是流式输入（输出）运算符，其左操作数为系统 isteam（ostream）类对象，右操作数为一个预定义类型的变量。cin（cout）是系统 istream（ostream）类的标准对象。使用“>>”（“<<”）进行输入（输出）操作时，不同类型的变量可以组合在一条语句中，编译程序在检查时，根据出现在“>>”（“<<”）操作符右边的变量或变量的类型来决定如何输入（输出）该数据。