



21世纪高职高专规划教材

计算机系列

C++语言程序设计

刘晶 裘旭光 编著



清华大学出版社

<http://www.tup.tsinghua.edu.cn>



北方交通大学出版社

<http://press.njtu.edu.cn>



清华大学出版社

清华大学出版社

C++ 语言程序设计

清华大学出版社



清华大学出版社

21 世纪高职高专规划教材·计算机系列

C++语言程序设计

刘晶 裘旭光 编著

清华大学出版社
北方交通大学出版社
·北京·

内 容 简 介

本书全面和系统地介绍 C++ 语言的基本概念, 基本语法和编程方法。以面向对象的程序设计方法引出必要的语法知识: 类和对象, 继承和派生类, 多态性和虚函数等。使读者在掌握 C++ 语言的同时, 对较简单的现实世界中的问题及解决方法能用它来描述。本书力求将复杂的概念用简洁的语言描述, 做到深入浅出, 并且提供实例和完整的程序解析便于读者学习。

本书可以作为高职高专院校 C++ 语言程序设计开发的教材, 也可供自学者参考。

版权所有, 翻印必究。

本书封面贴有清华大学出版社激光防伪标签, 无标签者不得销售。

图书在版编目 (CIP) 数据

C++ 语言程序设计 / 刘晶, 裘旭光编著. —北京: 北方交通大学出版社, 2003.10
(21 世纪高职高专规划教材. 计算机系列)
ISBN 7-81082-184-9

I. C… II. ①刘… ②裘… III. C 语言—程序设计—高等学校: 技术学校—教材 IV. TP312

中国版本图书馆 CIP 数据核字 (2003) 第 074782 号

责任编辑: 谭文芳 特邀编辑: 牛喘月

印刷者: 北京市黄坎印刷厂

出版发行: 北方交通大学出版社 邮编: 100044 电话: 010-51686045, 62237564

清华大学出版社 邮编: 100084

经 销: 各地新华书店

开 本: 787×1092 1/16 印张: 16.75 字数: 425 千字

版 次: 2003 年 10 月第 1 版 2003 年 10 月第 1 次印刷

印 数: 5 000 册 定价: 22.00 元

21 世纪高职高专规划教材·计算机系列 编审委员会成员名单

主任委员 李兰友 边奠英

副主任委员 周学毛 崔世钢 王学斌 丁桂芝 陈跃安
赵伟 韩瑞功 汪志达

委员 (按姓名笔画排序)

马辉	万志平	万振凯	王永平	王建明
丰继林	左文忠	叶华	叶伟	付慧生
江中	刘炜	刘建民	刘晶	曲建民
孙培民	邢素萍	华铨平	吕新平	陈小东
陈月波	李长明	李可	李志奎	李琳
李源生	李群明	李静东	邱希春	沈才梁
宋维堂	汪玉华	汪繁	张文明	张权范
张宝忠	张爱娟	张琦	金忠伟	林长春
林文信	苗长云	竺士蒙	周智仁	孟德欣
柏万里	宫国顺	柳炜	胡敬佩	姚策
赵英杰	高娟	高福成	贾建军	徐建俊
殷兆麟	唐健	黄斌	章春军	曹豫莪
程琪	韩其睿	韩劼	裘旭光	童爱红
谢婷	曾瑶辉	管致锦	熊锡义	潘玫玫
薛永三	操静涛	鞠洪尧		

出版说明

高职高专教育是我国高等教育的重要组成部分，它的根本任务是培养生产、建设、管理和服务第一线需要的德、智、体、美全面发展的高等技术应用型专门人才，所培养的学生在掌握必要的基础理论和专业知识的基础上，应重点掌握从事本专业领域实际工作的基本知识和职业技能，因而与其对应的教材也必须有自己的体系和特色。

为了适应我国高职高专教育发展及其对教学改革和教材建设的需要，在教育部的指导下，我们在全国范围内组织并成立了“21世纪高职高专教育教材研究与编审委员会”（以下简称“教材研究与编审委员会”）。“教材研究与编审委员会”的成员单位皆为教学改革成效较大、办学特色鲜明、办学实力强的高等专科学校、高等职业学校、成人高等学校及高等院校主办的二级职业技术学院，其中一些学校是国家重点建设的示范性职业技术学院。

为了保证规划教材的出版质量，“教材研究与编审委员会”在全国范围内选聘“21世纪高职高专规划教材编审委员会”（以下简称“教材编审委员会”）成员和征集教材，并要求“教材编审委员会”成员和规划教材的编著者必须是从事高职高专教学第一线的优秀教师或生产第一线的专家。“教材编审委员会”组织各专业的专家、教授对所征集的教材进行评选，对列选教材进行审定。

目前，“教材研究与编审委员会”计划用2~3年的时间出版各类高职高专教材200种，范围覆盖计算机应用、电子电气、财会与管理、商务英语等专业的主要课程。此次规划教材全部按教育部制定的“高职高专教育基础课程教学基本要求”编写，其中部分教材是教育部《新世纪高职高专教育人才培养模式和教学内容体系改革与建设项目计划》的研究成果。此次规划教材编写按照突出应用性、实践性和针对性的原则编写并重组系列课程教材结构，力求反映高职高专课程和教学内容体系改革方向；反映当前教学的新内容，突出基础理论知识的应用和实践技能的培养；适应“实践的要求和岗位的需要”，不依照“学科”体系，即贴近岗位群，淡化学科；在兼顾理论和实践内容的同时，避免“全”而“深”的面面俱到，基础理论以应用为目的，以必要、够用为度；尽量体现新知识、新技术、新工艺、新方法，以利于学生综合素质的形成和科学思维方式与创新能力的培养。

此外，为了使规划教材更具广泛性、科学性、先进性和代表性，我们希望全国从事高职高专教育的院校能够积极加入到“教材研究与编审委员会”中来，推荐“教材编审委员会”成员和有特色、有创新的教材。同时，希望将教学实践中的意见与建议及时反馈给我们，以便对已出版的教材不断修订、完善，不断提高教材质量，完善教材体系，为社会奉献更多更新的与高职高专教育配套的高质量教材。

此次所有规划教材由全国重点大学出版社——清华大学出版社与北方交通大学出版社联合出版。适合于各类高等专科学校、高等职业学校、成人高等学校及高等院校主办的二级职业技术学院使用。

21世纪高职高专教育教材研究与编审委员会
2003年9月

前 言

本书是在作者多年从事 C++ 语言教学的基础上编写的。书中总结了教学中的经验，结合现代高职高专和成人教育的特点修改补充而成。本书的特点是：由浅入深、便于理解、语言简洁、例题丰富、内容全面、重点突出。

在本书的编写中，针对初学者和自学者的特点，力求做到深入浅出，将复杂的概念用简洁浅显的语言讲述，以结合实例讲解基本概念和方法为主。

本书较全面、系统地讲述了 C++ 语言的基本概念和编程方法。通过对本书的学习，能够正确地理解 C++ 语言中的面向对象的方法，基本掌握 C++ 语言中的词法、语法，并且可以运用所学的知识解决一些实际问题。

本书共分 9 章。前 4 章讲述 C++ 语言的基本词法和语法规则，包括字符集、语法规则、程序结构、运算符和表达式、各类语句、函数格式和调用方法等内容。在这些内容中，C++ 内容和 C 语言内容很相近。第 5 章至第 8 章讲述 C++ 语言面向对象程序设计的基本概念，涉及 C++ 的特性，包括数据抽象、数据封装、继承与派生、动态联编、多态性和虚函数等内容。这些内容使 C++ 语言成为一种面向对象的程序设计语言，这是学习 C++ 语言的重点和难点。这部分是 C 语言没有的。第 9 章讲述了读写函数和对一般文件的操作。每章后面都提供相当数目的习题，读者可以通过习题辅助自己学习，循序渐进地提高。

本书的所有例题和习题中的程序都曾在 Visual Studio.net 系统中运行过，在其他版本的编译系统中一般也都可以运行。

由于时间仓促、水平有限，书中难免有欠妥之处，欢迎阅读与使用本书的所有教师、同学和读者提出宝贵意见。

编者

2003 年 10 月

目 录

第 1 章 C++语言概述	(1)
1.1 C++的产生	(1)
1.2 C++与 C 语言的关系	(1)
1.3 C++支持面向对象编程	(2)
1.4 C++词法及词法规则	(2)
1.4.1 C++的字符集	(2)
1.4.2 词法记号和词法规则	(3)
1.5 最简单的 C++程序	(4)
1.6 数据的输入与输出	(6)
1.6.1 I/O 流	(6)
1.6.2 标准输入和标准输出	(6)
1.7 C++程序的编译与运行	(7)
1.7.1 C++程序的实现	(7)
1.7.2 处理语法错误	(8)
1.8 Visual Studio.net 程序编辑环境	(8)
1.8.1 编译器窗口	(9)
1.8.2 编译器	(9)
1.8.3 编写第一个程序	(10)
1.9 小结	(12)
习题	(13)
第2章 数据类型和表达式	(15)
2.1 基本数据类型	(15)
2.2 常量	(16)
2.2.1 整型常量	(16)
2.2.2 浮点型常量	(16)
2.2.3 字符常量	(17)
2.2.4 字符串常量	(17)
2.2.5 布尔常量	(17)
2.2.6 符号常量	(18)
2.3 变量	(18)
2.3.1 变量名	(18)
2.3.2 变量的类型和初值	(19)
2.4 数组类型	(19)

2.4.1	一维数组的定义和引用	(19)
2.4.2	二维数组的定义和引用	(20)
2.4.3	数组元素的赋初值	(20)
2.5	指针和引用	(22)
2.5.1	指针	(22)
2.5.2	指针和数组	(25)
2.5.3	引用	(27)
2.6	自定义数据类型	(28)
2.6.1	类型定义语句	(28)
2.6.2	枚举类型	(29)
2.6.3	结构体	(30)
2.6.4	联合体	(31)
2.7	运算符和表达式	(33)
2.7.1	算术运算符	(33)
2.7.2	算术表达式	(35)
2.7.3	赋值运算符	(35)
2.7.4	赋值表达式	(36)
2.7.5	关系运算符与逻辑运算符	(36)
2.7.6	关系表达式与逻辑表达式	(38)
2.7.7	位操作运算符	(39)
2.7.8	条件运算符	(41)
2.7.9	条件表达式	(41)
2.7.10	逗号运算符	(42)
2.7.11	逗号表达式	(42)
2.7.12	求字节运算符	(43)
2.7.13	类型转换	(43)
2.7.14	运算符小结	(44)
2.8	小结	(45)
	习题	(45)
第 3 章	C++语句	(50)
3.1	语句	(50)
3.1.1	表达式语句	(50)
3.1.2	复合语句	(50)
3.2	选择语句	(51)
3.2.1	if 语句	(51)
3.2.2	switch 开关语句和 break 语句	(56)
3.3	循环语句	(60)
3.3.1	while 循环语句	(60)

3.3.2	do-while 语句	(62)
3.3.3	for 语句	(63)
3.4	多重循环	(66)
3.5	跳转语句	(69)
3.5.1	break 语句	(69)
3.5.2	continue 语句	(69)
3.5.3	goto 语句	(70)
3.6	小结	(71)
	习题	(71)

第 4 章 C++ 函数 (75)

4.1	函数的定义和调用	(75)
4.1.1	函数的定义	(75)
4.1.2	形参与实参	(76)
4.1.3	函数的调用	(77)
4.2	函数的嵌套调用和递归调用	(79)
4.2.1	函数的嵌套调用	(79)
4.2.2	函数的递归调用	(80)
4.3	函数原型	(81)
4.4	参数的传递机制	(83)
4.4.1	值调用	(83)
4.4.2	引用调用	(85)
4.5	内联函数	(86)
4.6	函数参数的省略	(87)
4.7	变量的存储类别	(89)
4.7.1	自动类别变量	(89)
4.7.2	静态类别变量	(90)
4.7.3	外部类别变量	(91)
4.7.4	寄存器类别变量	(92)
4.7.5	作用域分辨符 ::	(93)
4.8	数组作为函数的参数	(94)
4.9	函数与指针	(96)
4.9.1	指针作为函数的参数	(96)
4.9.2	指针函数	(97)
4.9.3	函数指针	(98)
4.10	函数的重载	(99)
4.10.1	参数类型不同的重载函数	(99)
4.10.2	参数个数不同的重载函数	(100)
4.11	C++ 的系统函数	(100)

4.11.1	C++系统函数概述	(100)
4.11.2	字符串处理函数	(101)
4.12	编译预处理	(104)
4.12.1	文件包含命令	(105)
4.12.2	条件编译命令	(105)
4.12.3	宏定义命令	(106)
4.13	小结	(109)
	习题	(109)
第 5 章	类与数据封装	(112)
5.1	对象、类与实体	(112)
5.2	如何设计一个类	(113)
5.2.1	一个与类很类似的概念	(113)
5.2.2	类	(114)
5.2.3	数据封装	(115)
5.3	构造函数和析构函数	(124)
5.3.1	构造函数	(124)
5.3.2	析构函数	(126)
5.3.3	复制构造函数	(128)
5.4	作用域和可见性	(131)
5.4.1	作用域	(131)
5.4.2	可见性	(133)
5.5	生存期	(134)
5.6	静态成员	(134)
5.6.1	静态数据成员	(135)
5.6.2	静态成员函数	(137)
5.7	友元	(138)
5.7.1	友元函数	(138)
5.7.2	友元类	(139)
5.8	小结	(140)
	习题	(140)
第 6 章	类的应用	(144)
6.1	共享数据的保护	(144)
6.1.1	常引用	(144)
6.1.2	常对象	(145)
6.1.3	用 const 修饰的对象成员	(145)
6.2	对象指针和对象引用	(147)
6.2.1	指向类的成员的指针	(147)

6.2.2	对象指针和对象引用做函数参数	(149)
6.2.3	this 指针	(152)
6.3	对象数组	(153)
6.3.1	对象数组的定义和赋值	(153)
6.3.2	指向数组的指针和指针数组	(154)
6.3.3	带参数的 main()函数	(159)
6.4	常数组和常指针	(160)
6.4.1	常数组	(160)
6.4.2	常指针	(160)
6.5	子对象和堆对象	(161)
6.5.1	子对象	(161)
6.5.2	堆对象	(163)
6.6	类的聚集	(165)
6.6.1	类聚集构造函数	(165)
6.6.2	前向引用声明	(167)
6.7	小结	(167)
	习题	(168)
第 7 章	继承	(171)
7.1	基类和派生类	(171)
7.2	基础类下的数据封装	(172)
7.3	派生类的定义	(179)
7.3.1	公有继承	(180)
7.3.2	私有继承	(182)
7.3.3	保护继承	(183)
7.4	派生类的构造函数与析构函数	(186)
7.4.1	构造函数	(186)
7.4.2	析构函数	(189)
7.5	赋值兼容规则	(190)
7.6	日期和时间实例	(191)
7.7	小结	(194)
	习题	(195)
第 8 章	多态性和虚函数	(200)
8.1	多态性	(200)
8.1.1	多态性概述	(200)
8.1.2	多态的类型	(200)
8.1.3	多态性的实现	(201)
8.2	运算符重载	(202)

8.2.1	运算符重载的问题	(203)
8.2.2	运算符重载为成员函数	(204)
8.3	虚函数	(205)
8.3.1	一般虚函数成员	(205)
8.3.2	虚析构函数	(207)
8.4	抽象类	(207)
8.4.1	纯虚函数	(207)
8.4.2	抽象类的作用	(208)
8.5	程序实例	(209)
8.6	小结	(214)
	习题	(214)
第 9 章	C++的 I/O 流库	(219)
9.1	流类的概念	(219)
9.2	输出流	(220)
9.2.1	使用预定义的插入符	(220)
9.2.2	使用成员函数	(221)
9.3	数据输出格式	(222)
9.3.1	输出宽度设置	(222)
9.3.2	输出精度设置	(223)
9.3.3	输出数据对齐方式与数制设置	(224)
9.3.4	文件输出形式	(225)
9.4	输入流	(228)
9.4.1	键盘输入	(228)
9.4.2	数据输入函数	(229)
9.5	流错误的处理	(231)
9.5.1	状态字和状态函数	(231)
9.5.2	清除/设置流的状态位	(232)
9.6	小结	(233)
	习题	(233)
	参考答案	(235)
	参考文献	(253)

第 1 章 C++语言概述

本章要点:

- 了解 C++发展史
 - 了解 C++与 C 语言的关系
 - 了解 C++支持面向对象编程的特性
 - 掌握 C++词法及语法规则
 - 了解 C++程序的组成
 - 掌握数据的输入输出
 - 了解 C++程序的实现
-

1.1 C++的产生

C++的发展历史起源于 C 语言。因此，C++是 C 的一个超集。C 语言最初是由贝尔实验室的 Dennis Ritchie 在 B 语言的基础上开发出来的，经过多次改进，现在流行的 C 语言版本基本上都是以 ANSI C 为基础的。

C 语言有很多特征，如功能强大、使用灵活，运算符和数据结构丰富，使用结构化编程语言等。这些优点使 C 语言拥有大批支持者，并成为很多程序员的首选编程语言。既然 C 是一个成功的编程语言，为什么还要发明 C++。答案在于程序的复杂性。一旦项目达到一定的规模，程序管理的复杂性就会超出程序员的管理能力。为了满足管理程序的复杂性需要，1980 年，贝尔实验室的 Bjarne Stroustrup 开始对 C 进行改进和扩充。最初的成果称为“带类的 C”，1983 年正式取名为 C++。在经历 3 次修订后，于 1994 年制定出 ANSI C++标准的草案。以后经过不断的完善，成为目前的标准 C++。

C++是以 C 为基础的，C++包含了整个 C。C++包括 C 的全部特征、属性和优点，同时添加了对面向对象编程(OOP)的完全支持。

1.2 C++与C语言的关系

1. C++保持对 C 语言的兼容

首先，这种兼容性表现在许多 C 代码不经修改就可以为 C++所用。用 C 语言编写的许多库函数和应用软件也都可以用于 C++。使用 C 语言的程序员学习 C++更容易、更方便，只要掌握 C++语言的新功能就可以了。

但是，C++的这种兼容性使它不是一个纯正的面向对象程序设计语言。因为 C 语言是面向过程的语言，C++要与 C 语言兼容，所以 C++也要支持面向过程的程序设计。但是在学习 C++时，要转变以往传统的面向过程的思路去学习面向对象的技术。

2. C++对 C 语言的改进

C++保持了 C 的简洁、高效和接近汇编语言等特点，同时对 C 进行了扩充，增加了面向对象的新技术。面向对象程序设计的实现需要封装和数据隐藏技术，需要继承和多态性技术。C++能够很好地支持面向对象程序设计。

1.3 C++支持面向对象编程

1. C++支持数据封装和数据隐藏

在 C++中，类是支持数据封装的工具，对象则是数据封装的实现。C++通过建立用户定义类型(类)支持数据封装和数据隐藏。

在面向对象的程序设计中，将数据和对该数据进行合法操作的函数封装在一起作为一个类的定义。对象被说明为具有一个给定类的变量。每个给定类的对象包含这个类所规定的若干私有成员、公有成员及保护成员。完好定义的类一旦建立，就可看成完全封装的实体，可以作为一个整体单元使用。类的实际内部工作隐藏起来，使用完好定义的类的用户不需要知道类是如何工作的，只要知道如何使用它即可。

2. C++支持继承和重用

在 C++现有类的基础上可以声明新类型，这就是继承和重用的思想。通过继承和重用可以更有效地组织程序结构，明确类间关系，并且充分利用已有的类来完成更复杂、深入的开发。新定义类为子类，成为派生类。它可以从父类那里继承所有非私有的属性和方法，作为自己的成员。

3. C++支持多态性

采用多态性为每个类指定表现行为。多态性形成由父类和它们的子类组成的一个树型结构。在这个树中的每个子类可以接收一个或多个具有相同名字的消息。当一个消息被这个树中一个类的一个对象接收时，这个对象动态地决定给予子类对象的消息的某种用法。多态性的这一特性允许使用高级抽象。

继承性和多态性的组合，可以轻易地生成一系列虽然类似但独一无二的对象。由于继承性，这些对象共享许多相似的特征。由于多态性，一个对象可有独特的表现方式，而另一个对象有另一种表现方式。

以上概括了 C++对面向对象程序设计的一些主要特征的支持，在后面的相关章节中将对这些支持做详细介绍。

1.4 C++词法及词法规则

1.4.1 C++的字符集

字符是一些可以区分的最小符号。字符集是构成 C++语言的基本元素。C++语言的字符集

由下列字符组成。

(1) 大小写英文字符: A~Z, a~z

(2) 数字字符: 0~9

(3) 特殊字符:

```

空格  !  #  %  ^  &  *  _(下划线)
—    +  =  ~  <  >  /  \  |
.    ,  :  ;  ?  ‘  “  0  []
{}

```

1.4.2 词法记号和词法规则

词法记号又称单词,它是由若干个字符组成的具有一定意义的最小词法单元。下面介绍几种 C++ 语言的词法记号及语法规则。

1. 标识符

标识符是由程序员定义的单词,用它来命名程序中的一些实体,如常量名、变量名、函数名、对象名、标号名、类型名等。C++ 标识符的构成规则包括:标识符以大小写字母、数字字符(0~9)和下划线组成,以字母或下划线开始,其后跟零或多个字母、数字字符或下划线组成。

- ◆ 标识符中区分大小字母,如 ABC, AbC, aBc, abc 等是不同的标识符。
- ◆ 标识符的长度(组成标识符的字符个数)虽然是任意的,但编译系统能够识别的标识符的长度是受到限制的。有的仅识别前 32 个字符。
- ◆ 建议尽量使用有意义的单词作为标识符。
- ◆ 标识符不能与系统的保留字相同,保留字是系统已经预定义的标识符,包括关键字和设备字等。

2. 关键字

关键字是 C++ 预定义的单词,在程序中有特殊的用途。

下面列举一些常用的关键字:

```

auto    break    case    char    class    const    continue  default
delete  do        double  else    enum    explicit  extern    float
for     friend    goto    if      inline   int      long     mutable
new     operator  private protected public    register  return    short
signed  sizeof    static  static_cast  struct  switch   this     throw
true   typedef   typeid  union   unsigned virtual   void     while

```

用户对这些关键字不可重新定义,其用法将在以后章节中逐步介绍。

3. 运算符

运算符也称操作符,是实现运算的各种符号,如+、-、*、/、…。有关各种运算符的功能在后面的章节中将详细介绍。

4. 分隔符

分隔符也称为标点符号。它用来分隔各个词法记号或程序正文。C++中，常用的分隔符有下列5种。

(1) 空格符：用来作为词法记号之间的分隔符。

(2) 逗号：用来作为说明多个变量或对象类型时变量之间的分隔符；也可作为函数的多个参数之间的分隔符。

(3) 分号：仅用在 for 循环语句中 for 关键字后面括号中作为三个表达式的分隔符。

(4) 冒号：作为语句标号与语句之间的分隔符。

5. 常量

常量就是直接使用符号表示的值。常量有数字常量、字符常量、字符串常量等。

6. 注释符

注释在程序中的作用是对程序进行注解和说明，以便于阅读。编译系统在对源程序进行编译时不编译注释部分，因此注释对程序的功能实现不起任何作用。由于编译是忽略注释部分的，所以注释内容不会增加最终产生的可执行程序的大小。注释的目的是为了便于阅读程序，提高程序的可读性。

C++中采用如下两种方法进行注释。

第一种是使用“/*”和“*/”括起来进行注释。例如：

```
/* This is a      */  
/*   C++ program. */
```

“/*”和“*/”之间的所有字符都被作为注释符处理。

第二种方法是使用“//”，从“//”开始，到它所在行的行尾，所有字符都被作为注释符处理。例如：

```
// This is a C++ program.
```

7. 空白符

空白符是空格符、换行符和水平制表符等的统称。需要注意的是，空白符不等于空格符，只是空白符包含空格符。另外，要把空白符与空字符区分开。空字符是指 ASCII 码值为 0 的字符。在 C++中，存放在内存中的字符串常量都在最后有一个结束符，空字符就被用来作为字符串的结束符，它用转移序列表示为 ‘\0’。

1.5 最简单的C++程序

了解 C++程序结构的特点，先看一个最简单的 C++程序。

[例 1-1] 输出一句话“Hello World!”。

```
// This is a C++ program.  
#include<iostream.h>  
void main()
```