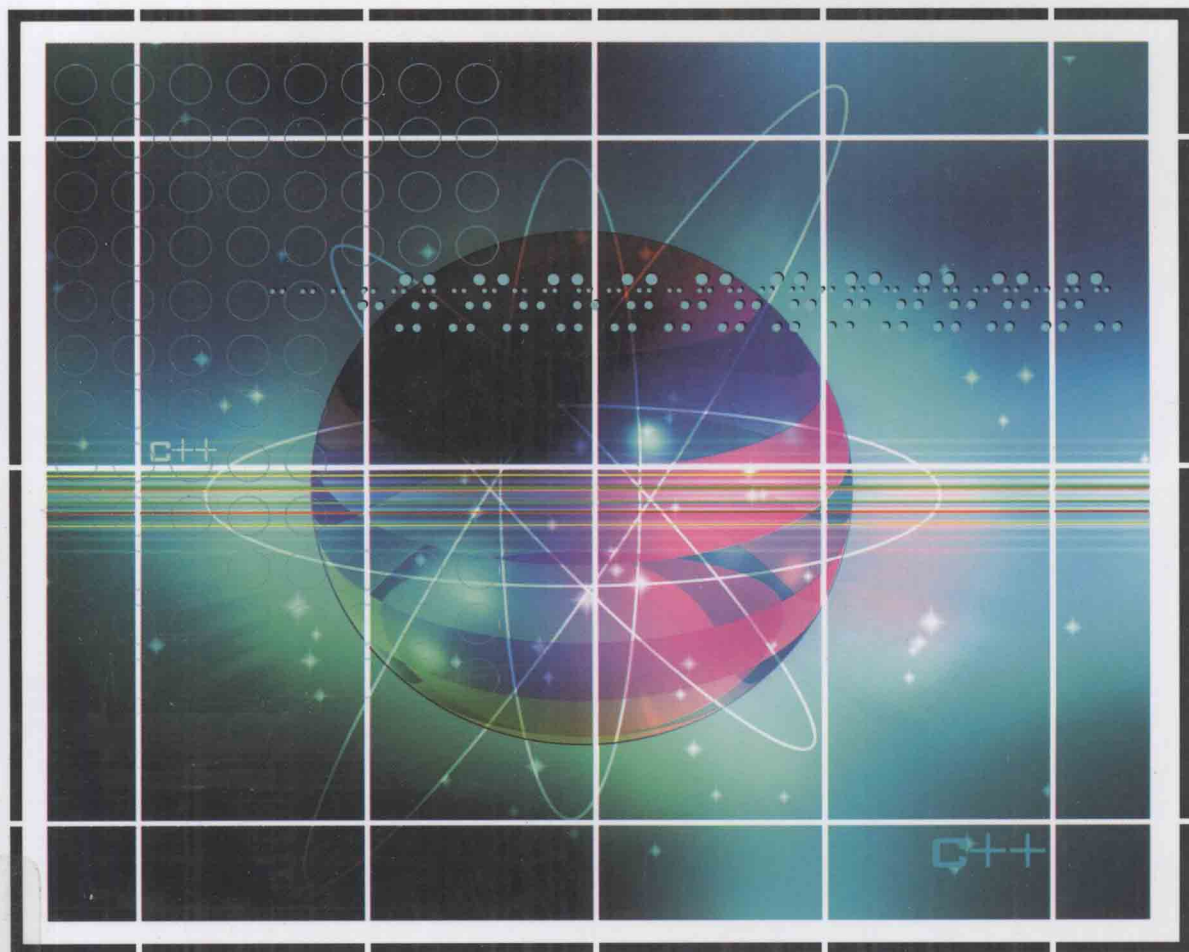




普通高等教育“十一五”国家级规划教材

C++程序设计语言

李雁妮 陈 平 王献青 编著



西安电子科技大学出版社
<http://www.xduph.com>

普通高等教育“十一五”国家级规划教材

C++程序设计语言

李雁妮 陈 平 王献青 编著

西安电子科技大学出版社

2009

内 容 简 介

本书分三部分,共16章。第一部分对C++语言的基本机制,即对C++语言中用于支持面向过程与面向模块化程序设计的语言机制进行了较为准确与全面的介绍;第二部分重点介绍了C++支持面向对象与类属程序设计的各种语言机制,同时,在该部分对C++的异常处理机制进行了较为详尽的介绍;第三部分对C++标准模板库STL进行了简要阐述,由于程序一般都要进行字符串与输入/输出处理,因此,在该部分重点对标准类库中的string类和C++的I/O类进行了较具体的介绍。

本书针对计算机专业的本科生编写。书中注有星号的章节为C++中较深入的一些问题,在教学中可视教学时数与教学对象进行适当取舍。本书除作为本科生C++程序设计的教材之外,还可供计算机或电子类相关专业的研究生或工程技术人员参考学习。

★本书配有电子教案,需要者可登录出版社网站,免费下载。

图书在版编目(CIP)数据

C++程序设计语言 / 李雁妮, 陈平, 王献青编著.

—西安: 西安电子科技大学出版社, 2009.1

普通高等教育“十一五”国家级规划教材

ISBN 978-7-5606-2151-7

I. C… II. ①李… ②陈… ③王… III. C语言—程序设计—高等学校—教材

IV. TP312

中国版本图书馆 CIP 数据核字(2008)第 175588 号

策 划 臧延新

责任编辑 张晓燕

出版发行 西安电子科技大学出版社(西安市太白南路2号)

电 话 (029)88242885 88201467 邮 编 710071

网 址 www.xduph.com 电子邮箱 xdupfxb001@163.com

经 销 新华书店

印刷单位 陕西华沐印刷科技有限责任公司

版 次 2009年1月第1版 2009年1月第1次印刷

开 本 787毫米×1092毫米 1/16 印 张 26

字 数 618千字

印 数 1~4000册

定 价 37.00元

ISBN 978-7-5606-2151-7/TP·1097

XDUP 2443001-1

*** 如有印装问题可调换 ***

本社图书封面为激光防伪覆膜, 谨防盗版。

前 言

本书作者均长期致力于 C/C++ 语言的教学与科研工作, 其中陈平教授为 ISO C++ 语言标准化技术专家、教育部计算机专业教学指导委员会委员, 本书是他及所带领的 C++ 课程组成员在长期的教学、科研实践中所形成的智慧与成果的结晶。

C++ 是一种支持多种程序设计范型、优秀的通用程序设计语言。目前, 国内多家出版社已陆续出版了多种 C++ 程序设计语言教材, 这些教材都各具特色, 但总的来说:

(1) 已有教材内容没有完全涵盖最新的 2004 年 CCSE(Computing Curricula-Software Engineering, 计算机软件工程学科)和 CSEC(China Software Engineering Curricula, 中国软件工程学科)在相应课程中的知识点, 部分内容缺失或深度不够。

(2) 已有教材内容大多仅立足于语言语法本身, 没有将其内容提升到引导与培养学生正确、适当地使用各种语言机制的高度上, 即在内容的组织与选取上对如何培养、训练学生成为一个优秀的 C++ 程序员考虑不够。

(3) C++ 是支持多种程序设计范型且广泛使用的高级程序设计语言之一。大多数教材没有清楚地阐述 C++ 支持多种程序设计范型这一特征, 亦没有完整地阐述 C++ 支持多种程序设计范型的各种语言机制, 这易使读者不知或误用各种语言机制。

(4) 已有教材的内容大多缺少实际软件项目的研发成果, 内容及示例显得空泛。

目前, 国外有多种 C++ 经典教材, 其典型代表为 B.Strostrup 所编写的《C++ Programming Language》和 Bruce Eckel 所编写的《Thinking in C++》等。基于我国国情, 上述国外原版教材并不完全适宜用作我国大多数院校本科低年级学生的 C++ 课程教材, 原因如下:

(1) 教材为英文, 并具有相当的深度、难度, 对国内大多数院校本科低年级学生来讲, 难度太大; 其翻译版亦普遍存在译文质量不高、信息损失等问题。

(2) 教材并非按 CCSE 和 CSEC 编写, 且上述两本原版著作是按照全面、深入阐述 C++ 语言特性, 培养高级 C++ 程序员、系统分析员这一宗旨而撰写的, 故部分章节内容不适合作为本科生低年级学生的 C++ 课程教材。

基于上述原因, 作者在对国内外 C++ 教材进行了大量阅读、研究、实践的基础上, 根据 CCSE 和 CSEC 编写了这本适合我国国情并与软件人才培养目标接轨, 能准确阐述 C++ 各种语言特性、用法, 且包含作者长期教学与科研成果的《C++ 程序设计语言》。

全书分三部分, 共 16 章。第一部分对 C++ 语言的基本机制, 即对 C++ 语言中用于支持面向过程与面向模块化程序设计的语言机制进行了较为准确与全面的介绍; 第二部分重点介绍了 C++ 支持面向对象与类属程序设计的各种语言机制, 同时, 在该部分对 C++ 的异常处理机制进行了较为详尽的介绍; 第三部分对 C++ 标准模板库 STL 进行了简要阐述, 由

于程序一般都要进行字符串与输入/输出处理，因此，在该部分重点对标准类库中的 `string` 类和 C++ 的 I/O 类进行了较具体的介绍。

本书有如下特色：

(1) 内容涵盖最新的 2004 年 CCSE 和 CSEC 在相应课程中的知识点，并从深度和广度等各方面与之要求吻合。

(2) 注重 C++ 语言支持多种程序设计范型这一特征，完整、准确地阐述了其语言的各种机制，并通过示例说明各种机制适用及不适用的场合，着重培养与训练学生设计、编写各种程序的能力。

(3) 准确地阐述各种语言机制的语法，其立足点与软件人才培养目标相吻合。

(4) 包含课程组长期以来的教学与科研成果，教材示例中加大经裁剪后的实际项目程序的比例，通过系统软件领域中的实例解释说明一些关键性的程序设计概念与技术。通过示例，向学生展示实际 C++ 应用程序的构建、编程方法。

本书针对计算机专业的本科生编写。书中注有星号的章节为 C++ 中较深入的一些问题，在教学中可视教学时数与教学对象进行适当取舍。本书除作为本科生 C++ 程序设计的教材之外，还可供计算机或电子类相关专业的研究生或工程技术人员参考学习。

感谢同事王黎明、邓岳，感谢西安电子科技大学软件学院 99 级学生刘志鹏和 04 级学生王晓丽、吴涛，他们对本书提出了许多宝贵的建设性意见，作者据此对全书进行了全面修正。

在本书的编著过程中，我们虽力图详尽、准确，但限于作者水平，疏漏之处在所难免，恳请广大读者批评指正。

编 者

2008 年 11 月

目 录

第一部分 C++语言的基本机制

第1章 绪论	2	3.1 类 型	22
1.1 C++语言的发展历史及特点	2	3.2 C++中的基本数据类型	24
1.1.1 C++语言的发展历史	2	3.2.1 布尔类型	24
1.1.2 C++语言的特点	3	3.2.2 字符类型	25
1.2 学习 C++语言的注意事项	3	3.2.3 整数类型	26
1.2.1 如何学习 C++	3	3.2.4 浮点类型	27
1.2.2 如何使用本教材	4	3.2.5 C++数据类型存储量的大小	29
1.3 C++语言中一些重要的程序设计理念	5	3.3 void 类型	30
小结	7	3.4 枚举类型	30
练习题	7	3.5 类型的声明与定义	32
3.5.1 声明的语法规则			33
第2章 C++语言概述	8	3.5.2 C++中的标识符	34
2.1 C++语言及程序设计范型	8	3.5.3 标识符的作用域	35
2.1.1 C++语言的概念	8	3.5.4 typedef	35
2.1.2 程序设计范型	9	3.6 类型转换	36
2.1.3 第一个 C++程序及 C++程序结构	9	小结	37
2.2 过程程序设计范型	10	练习题	38
2.2.1 过程程序设计范型介绍	10	第4章 运算符与语句	39
2.2.2 变量和算术运算符	11	4.1 C++运算符概述	39
2.2.3 条件判断与循环	12	4.1.1 算术运算符和自增、自减运算符	40
2.2.4 指针与数组	12	4.1.2 关系和逻辑运算符	42
2.3 模块化程序设计范型	13	4.1.3 位运算符	43
2.4 数据抽象	14	4.1.4 内存申请与释放运算符 new 和 delete	46
2.5 面向对象程序设计范型	17	4.1.5 赋值运算符	47
2.6 类属/通用程序设计范型	19	4.1.6 类型转换运算符	47
小结	21	4.1.7 C++运算符概览及其优先级次序	49
练习题	21	4.2 C++语句	50
第3章 类型与声明	22		

4.2.1 表达式语句和空语句	51	6.4.2 重载函数的匹配规则	102
4.2.2 注释语句及意义	51	6.4.3 重载函数与函数的返回类型	105
4.2.3 复合语句	52	6.4.4 重载与作用域	105
4.2.4 选择判断语句	52	6.5 缺省的函数参数值	106
4.2.5 循环语句	57	6.6 递归	107
4.2.6 跳转语句	62	6.6.1 递归的基本概念	107
小结	65	6.6.2 递归的定义及递归函数的 编写模式	108
练习题	66	6.7 参数数目可变的函数	110
第 5 章 指针、数组和结构	67	6.8 函数指针	114
5.1 指针	67	6.9 综合示例	115
5.1.1 指针与指针变量	67	小结	128
5.1.2 为什么要使用指针变量	68	练习题	128
5.1.3 指针变量的声明与定义	69	第 7 章 名字空间与异常处理	129
5.1.4 指针变量的操作	69	7.1 模块与接口的基本概念	129
5.1.5 常量零(0)	71	7.2 名字空间	131
5.2 数组	71	7.2.1 名字空间的基本概念	131
5.2.1 数组的定义与初始化	71	7.2.2 名字空间中的名字解析	132
5.2.2 字符串字面值	73	7.2.3 模块的多重接口	137
5.3 指向数组的指针	74	7.3 异常处理	138
5.3.1 指向一维数组的指针	74	7.4 综合示例	141
5.3.2 指向多维数组的指针	75	小结	146
5.3.3 取数组元素及数组的遍历	76	练习题	147
5.4 指向函数的指针	80	第 8 章 源文件和程序	148
5.5 指向 void* 的指针	81	8.1 分别编译	149
5.6 常量	82	8.2 链接	149
5.7 引用	85	8.2.1 链接与一致性的基本概念	149
5.8 结构	88	8.2.2 头文件	151
小结	94	8.2.3 #include 指令	152
练习题	95	8.2.4 用户头文件内容的设计	152
第 6 章 函数	96	8.3 头文件的有效使用	154
6.1 函数的声明	98	8.4 命令行参数	156
6.1.1 函数接口/原型声明	98	8.5 程序	158
6.1.2 函数的定义	98	8.5.1 程序的执行	158
6.2 函数的参数传递	99	8.5.2 程序的终止	159
6.3 函数的返回值	101	小结	159
6.4 函数名的过载/重载	102	练习题	160
6.4.1 函数名过载/重载的基本概念	102		

第二部分 C++的抽象机制

第9章 类与对象	162	10.7.2 函数调用操作符的重载	212
9.1 类的基本概念	162	10.7.3 指针/指向操作符的重载	214
9.2 类中成员	163	10.7.4 自增、自减操作符的重载	216
9.2.1 类中的成员	163	10.7.5 流输入与流输出操作符的重载	217
9.2.2 类的访问控制	165	10.8 综合示例	219
9.2.3 类的构造函数	168	小结	237
9.2.4 类的静态成员	170	练习题	238
9.2.5 对象的拷贝	173		
9.2.6 常量(Const 或称只读)成员函数	175	第11章 继承与多态	239
9.2.7 对象的自身引用——this	176	11.1 概述	239
9.3 定义有效、高质量的类	178	11.2 子类/派生类	240
9.4 对象	180	11.2.1 子类/派生类与继承的基本概念	240
9.4.1 对象是什么	180	11.2.2 子类对象的存储结构	242
9.4.2 C++中对象的类别	182	11.2.3 子类中的成员	242
9.4.3 对象的析构——析构函数	183	11.2.4 子类的构造与析构函数	243
9.4.4 默认构造函数	184	11.2.5 子类对象拷贝	246
9.4.5 几种主要类别对象的构造与析构	184	11.2.6 public、protected 和 private 继承	247
9.4.6 对象的构造与析构次序	187	11.3 虚函数与多态性	253
9.5 综合示例	189	11.3.1 类型域	253
小结	195	11.3.2 虚拟函数	255
练习题	196	11.3.3 抽象基类与实例类	257
		11.3.4 多态	259
第10章 操作符重载	198	11.3.5 虚拟的析构函数	261
10.1 概述	198	*11.4 运行时的类型识别	263
10.2 操作符重载	199	11.4.1 dynamic_cast 运算符	263
10.2.1 二元操作符的重载	200	11.4.2 typeid 运算符	266
10.2.2 一元操作符的重载	201	*11.5 指向类成员的指针	266
10.3 类型转换操作符	202	11.5.1 指向类成员的指针	266
10.3.1 类型转换函数	202	11.5.2 指向类的成员函数指针的 应用场合	268
10.3.2 歧义性(二义性)问题	204	11.6 多重继承	270
10.4 友员	204	11.7 综合示例	273
10.5 大型对象	206	小结	279
10.6 类中应具有的基本操作	207	练习题	280
10.7 几种特殊操作符的重载	211		
10.7.1 下标运算符的重载	211		

第 12 章 模板283

12.1 概述283

12.2 类模板285

12.2.1 类模板的定义285

12.2.2 类模板参数及其限制288

12.3 函数模板289

12.3.1 函数模板的定义289

12.3.2 函数模板的重载290

12.3.3 函数调用的匹配原则292

12.3.4 编写函数模板时的注意事项292

12.4 模板与继承297

12.5 综合示例299

小结313

练习题313

第 13 章 异常处理314

13.1 概述314

13.2 C++异常处理结构 try、throw
和 catch315

13.2.1 抛出异常315

13.2.2 重新抛出异常316

13.2.3 捕获所有的异常317

13.3 异常类层次317

*13.4 捕获 new 操作所产生的异常318

*13.5 C++ 标准库异常层次321

小结321

练习题322

第三部分 C++标准模板库 STL 简介

第 14 章 string 类324

14.1 string 概述324

14.2 string 类的构造函数与析构函数324

14.3 string 类重载的操作符325

14.4 string 类的成员函数328

14.5 string 的基本操作329

14.5.1 元素访问329

14.5.2 赋值330

14.5.3 从 string 转换到 C 风格的字符串 ...331

14.5.4 字符串的比较331

14.5.5 附加与插入333

14.5.6 查找子串334

14.5.7 替换335

14.5.8 求子串336

14.5.9 string 对象的大小和容量336

14.5.10 输入输出336

14.6 C 风格的字符串337

14.6.1 C 字符串操作函数337

14.6.2 将数值字符串转换到数值的函数 ...338

14.6.3 字符分类338

14.7 迭代器339

14.8 字符串流处理340

小结340

练习题340

第 15 章 C++ 输入/输出系统基础342

15.1 C++ 中的流概述342

15.1.1 C++的输入/输出流类库中的
头文件343

15.1.2 输入/输出流类和对象343

15.2 输出流344

15.3 输入流345

15.3.1 流读取运算符345

15.3.2 用于输入的一些成员函数346

15.4 成员函数 read 和 write 的无格式
输入/输出347

15.5 流操纵算子347

15.5.1 设置整数流的基数348

15.5.2 设置浮点数精度348

15.5.3 设置输出域宽350

小结351

练习题352

第 16 章 标准模板库 STL 简介	353
16.1 STL 概述	353
16.1.1 容器	353
16.1.2 算法	353
16.1.3 迭代器	354
16.1.4 其它 STL 元素	354
16.2 容器类	355
16.3 STL 类的一般操作原理	356
16.4 vector 容器	357
16.4.1 通过迭代器访问 vector 矢量中的 元素	360
16.4.2 vector 的其它成员函数	361
16.4.3 在 vector 中存储自定义类型的 对象	363
16.5 list 容器	364
16.6 deque 双向队列	369
16.7 关联容器	369
16.7.1 map 关联容器类	369
16.7.2 set 和 multiset 关联容器类	372
16.8 容器适配器	373
16.8.1 stack 适配器	373
16.8.2 queue 适配器	375
16.8.3 priority_queue 适配器	376
16.9 算法	376
16.9.1 fill、fill_n、generate 与 generate_n 算法	378
16.9.2 equal、mismatch 和 lexicographical_ compare 算法	380
16.9.3 remove、remove_if、remove_copy 和 remove_copy_if 算法	382
16.9.4 replace、replace_if、replace_copy 和 replace_copy_if 算法	384
16.9.5 一些常用的数学算法	386
16.9.6 基本查找与排序算法	388
16.9.7 swap、iter_swap 和 swap_ranges 算法	390
16.9.8 copy_backward、merge、unique 和 reverse 算法	392
16.9.9 inplace_merge、unique_copy 和 reverse_copy 算法	393
16.9.10 集合操作	394
16.9.11 lower_bound、upper_bound 和 equal_range 算法	396
16.9.12 堆排序	398
16.9.13 min 和 max 算法	400
16.10 函数对象	401
16.10.1 一元函数对象与二元函数对象	401
16.10.2 STL 内置的函数对象	401
16.10.3 绑定器	404
参考文献	406

第一部分 C++ 语言的基本机制

这部分主要对 C++ 语言的子集 C 语言、C++ 语言支持面向过程及面向模块程序设计范型所提供的语言机制进行了全面、系统的介绍。

该部分所涵盖的内容如下：

第 1 章 绪论

第 2 章 C++ 语言概述

第 3 章 类型与声明

第 4 章 运算符与语句

第 5 章 指针、数组和结构

第 6 章 函数

第 7 章 名字空间与异常处理

第 8 章 源文件和程序

第 1 章 绪 论

本章要点:

- ▣ C++ 语言的发展历史及特点;
- ▣ 学习 C++ 语言的注意事项;
- ▣ C++ 语言中一些重要的程序设计理念。

1.1 C++语言的发展历史及特点

1.1.1 C++ 语言的发展历史

C++ 语言是当今流行的、能支持多种程序设计范型(Programming Paradigm)的一种优秀的程序设计语言。

20 世纪 70 年代末,随着计算机应用的普及与深入,软件的规模及复杂性以前所未有的趋势大幅度地增长,但当时缺乏一种能准确地描述问题域与解、支持数据抽象(Abstract Data Type, ADT)、能快速开发并有效地组织与维护大型程序、支持面向对象程序设计(Object-Oriented Programming, OOP)范型的通用程序设计语言。正是在这种应用需求的强烈驱动下,1979 年 10 月,C++ 语言的第一个版本——带类的 C 应运而生。

C++ 语言自诞生之日起到发展成熟并最终走向标准化经历了约 20 年的时间。C++ 起源于 1979 年 4 月 Bjarne Stroustrup 博士在美国新泽西州 Murray Hill 贝尔实验室计算科学研究中心开始的如何将所分析的 UNIX 内核分布到局域网上这一研究工作。为了解决描述复杂系统的模块结构及模块间的通信模式问题,并试图书写事件驱动的模拟仿真程序,Bjarne Stroustrup 开始设计并实现一个带类的 C(C with class)的工作。1979 年 10 月,第一个带类的 C(称为 Cpre)的实现在 Murray Hill 贝尔实验室投入使用。1983 年 8 月第一个 C++实现走出实验室并正式投入使用,同年 12 月带类的 C 正式改名为 C++(发音为 C plus plus)。1985 年 10 月,C++ 1.0 版本(称为 Cfront Release 1.0)开始正式商业发布。同时,Bjarne Stroustrup 出版了其经典著作《The C++ Programming Language》(第 1 版)。之后,1987 年 2 月、1989 年 6 月及 1991 年 10 月,Cfront Release 1.2、Cfront Release 2.0 和 Cfront Release 3.0 相继问世。Bjarne Stroustrup 博士的《The C++ Programming Language》的第 2 版和第 3 版亦于 1991 年 6 月和 1997 年 7 月分别出版。

C++自诞生之日起便以其表达能力强、高效、支持多种程序设计范型等特点受到业界的广泛认可和欢迎。随着 C++ 语言的普及及其应用领域爆炸性地扩张,C++ 标准化问题提

到了议事日程上。1987年 Bjarne Stroustrup 开始了 C++标准化的准备工作, 1989年12月, C++ 美国国家标准委员会 ANSI X3J16 组织成立, 1990年3月召开了第一次 ANSI X3J16 技术会议, 在此会议上确立了 C++ 美国国家标准, 并于同年5月发布了 ANSI C++ 标准化工作的基础文件《The Annotated C++ Reference Manual》。1991年6月召开了第一次 C++ 国际化标准 ISO WG21 会议, 1994年8月 ANSI/ISO 委员会 C++ 草案登记, 1995年4月 ISO C++标准草案提交公共审阅, 1997年10月 ISO C++ 标准通过表决并被接受, 至1998年11月, ISO C++ 标准(ISO/IEC 14882)被正式批准。目前采用的 C++ 标准即此标准。

1.1.2 C++ 语言的特点

C++ 语言是以 C 语言为基础, 并在此基础上扩充、发展而来的。C++ 语言是 C 语言的进化版本, 其名称正反映了这一点。Bjarne Stroustrup 设计并实现 C++ 的初衷是使 C++ 语言不仅具有像 Simula 语言管理与组织大型程序的机制, 同时又兼有 C 语言的高效性与灵活性; 更重要的是欲使当时大量的 C 程序和 C 库函数得以继承使用, 大批优秀的 C 程序员不丢弃长期积累的 C 编程经验, 只需要学习 C++ 加入的一些新特性就能快速、平滑地过渡到这种支持新的程序设计范型且表达力更强的语言。

C++ 语言不仅继承与发扬了 C 语言的优点, 而且吸纳了其它众多语言的优良特性。例如, C++ 语言中的一些新特性, 单行//注释来源于 BCPL's; 类的概念, 包括派生类及虚函数来源于 Simula 67; 操作符重载及自由的变量声明来源于 Algol 68 语言; 模板机制主要受 Ada 语言的启发; 错误处理主要来源于 Ada、Clu 和 ML 语言。C++ 语言的一些其它机制, 如多重继承、纯虚函数、名字空间等是在 C++ 语言的发展及应用过程中逐步产生的。

C++ 是支持多种程序设计范型的优秀程序设计语言之一, 其主要特点如下:

(1) 以 C 作为其子集, 兼取了 C 语言简洁、相对低级的特性, 但摒弃掉了 C 语言中若干不安全的特性, 其语言表现力远远强于 C 语言;

(2) 是一种强类型语言;

(3) 具有较高的可移植性和可维护性;

(4) 适合于大部分系统程序及应用程序的开发;

(5) 是一种不限定应用领域的通用程序设计语言;

(6) 是一种能支持面向过程、面向模块、面向对象和类属程序设计范型的混合型程序设计语言。

1.2 学习 C++ 语言的注意事项

1.2.1 如何学习 C++

C++ 语言的发明及实现者 Bjarne Stroustrup 博士告诫我们: “在学习 C++ 语言的时候, 最重要的是应把注意力集中在其概念方面, 而不是陷入只关注语言技术细节的误区。学习一种程序设计语言的目的是成为一个好的程序员, 在设计与实现新的系统、维护已有系统的过程中具有更高的效率。因此, 对于程序设计技术和软件设计技术的鉴赏, 要比理解语

言的细节重要得多。随着时间的延伸和实践的增加, 这些语言的细节将自然会被理解的。”

Bjarne Stroustrup 博士上述名言为我们如何学习及学好 C++ 语言指明了道路。在学习 C++ 语言的时候, 应注意如下几点:

(1) 学习 C++ 语言的目的是要将它作为一种工具很好地应用于软件系统的开发与维护, 而不能仅限于了解 C++ 语言的很多语法细节, 却不去关注如何正确地使用它。

(2) 程序设计风格或称程序设计范型(Programming Styles/Paradigm)通常由思维方式和语言的支持机制所决定, 而不同的应用领域要求的思维方式不同, 因而程序设计风格或范型不同, 对语言的支持要求也不同。因此应切记: C++ 语言是支持多种程序设计风格/范型的一种通用的、混合性程序设计语言。在学习及使用该语言时一定要注意其各种语言机制到底支持哪种程序设计风格/范型, 以避免对 C++ 语言机制的乱用与误用。

(3) C++ 同时支持多种程序设计风格/范型的能力, 使其应用领域很宽, 但其支持的语言机制绝对不是“放之四海而皆准”的。因此, 在学习 C++ 时, 一定要注意各种语言机制适用及不适用的场合。

(4) 学习 C++ 语言的途径不是唯一的, 学习方法及门坎的高低亦因人而异, 这些都与学习者已有的基础和预定的目标有关。我们期望学习者是为了更好地进行程序设计和软件设计而学习 C++ 的。

(5) C++ 是一个相对复杂的语言, 但不需要在掌握了这种语言的所有语言特性和技术内涵之后才开始真正使用它。C++ 可以在多个不同的专业层次上使用, 所以读者可以通过实践循序渐进地学习与掌握 C++。

(6) 跳过 C 语言的学习而直接学习 C++ 语言是值得提倡的一种学习方法。C++ 更安全, 表现力更强, 又减少了对低层技术的关注要求, 因此比 C 语言更容易使用与掌握。有了 C 语言的基础再学 C++ 语言, 虽然入门较快, 但实践证明最终很难摆脱 C 语言的思维方式, 也很难从用 C++ 写出 C 程序这一误区中走出来。

(7) 在学习过程中, 读者还可通过多种途径学习与实践 C++, 如利用一些可用的工具、程序库和软件开发环境, 通过大量的教科书、手册、杂志、BBS、邮件组、会议和课程等学习 C++ 并得到其语言的最新发展信息。

最后, 至关重要的是在学习的过程中要大量地阅读优秀的 C++ 源代码, 从中吸取经验, 获取灵感。学习中一定要加大实践力度, 实践出真知! 只有通过阅读—模仿—实践—再实践的途径, 才能深入学习和掌握 C++。

1.2.2 如何使用本教材

本书的内容共分为三大部分。第一部分共计 8 章, 重点阐述 C++ 语言的基本机制。这一部分阐述了 C++ 语言的子集 C 语言的相关内容, 及 C++ 为支持过程程序设计、模块化程序设计范型而加入的一些语言新特性, 如函数重载(亦称函数过载)、异常处理、名字空间等。第二部分共计 5 章, 重点阐述了 C++ 支持面向对象、类属/通用程序设计范型的各种语言机制, 该部分为本书的重点。由于对一种语言的掌握与运用很大程度上取决于学习者对其类库的熟悉与掌握, 因而本书在第三部分对 C++ 标准输入/输出流类库、string 类进行了介绍, 最后, 对标准模板库(Standard Template Library, STL)进行了概述。

本书力图从基本概念出发,进而深入阐述 C++ 语言支持各种程序设计范型的各种语言机制。书中短小的程序用以阐明语言的语法,各章中的应用示例用于展现 C++ 语言各种机制的应用场合、应用技巧及 C++ 程序的组织架构方法。另外,在第一部分采用一个小型应用程序“小型桌面计算器的设计与实现”向读者分别展示如何利用 C++ 语言所提供的语言机制进行面向过程与面向模块的程序设计方法;在第二部分采用另外几个小型应用程序,如“单向链表的设计与实现”、“类模板 SortedSet 的设计与实现”及“函数模板 sort 的设计与实现”向读者展示 C++ 的面向对象及类属程序设计方法。希望读者在学习过程中认真研读书中的范例,从中启发思路,进而设计出更加优秀的实用程序。

1.3 C++ 语言中一些重要的程序设计理念

如前所述, C++ 语言是从 C 语言演化而来的,但其程序设计理念较 C 语言而言,很多方面已发生了质的飞跃。C++ 语言仍像其它程序设计语言一样内置了一些基本的数据类型,如 int、float、char 等,这些基本类型对应着现实世界中的整数、实数和字符等概念,但大多数应用中还普遍存在着一些概念,它们既不容易被表示成某个基本类型(即语言提供的内置类型(Build-in/Primitive Type)),也不容易被表示成某个没有相关数据的函数,如现实中人、汽车等概念就无法用 C++ 的基本类型准确地表示。因此, C++ 中提供了类(Class)来表示应用中这样的概念。类是 C++ 语言的第一概念,它实现了信息隐藏(Information Hiding)与封装(Encapsulation)。一个类是应用领域中一个概念的具体表示与抽象。

现实中,任何概念都不可能存在于真空之中,总是存在着一些由相关概念组成的簇(Cluster)。因此,在程序中组织(表示不同概念的那些)类之间的关系,经常比用单个的类来定义一个概念要困难得多。组织与管理这些复杂概念的最强有力的工具之一就是层次结构(Hierarchy),即将相关的一组概念组织成一棵树(Tree),使得最一般化的概念对应于树的根。在 C++ 中,派生类(Derived Class)就用于表示这样的结构。例如,应用中有一组相关的概念:人、雇员、学生、经理、本科生和研究生,这组相关的概念可用 C++ 提供的继承和派生类机制将其组织起来,其共性抽象到树的上层,下层对应着特性。这种组织方式使得对上层的修改自动波及到下层,而对下层的修改不会影响到上层。另外,这种程序概念的组织方式不仅大大提高了软件的重用(Reuse),而且极大地增强了软件的可扩充性与可维护性。上述相关类的组织层次图如图 1.1 所示。

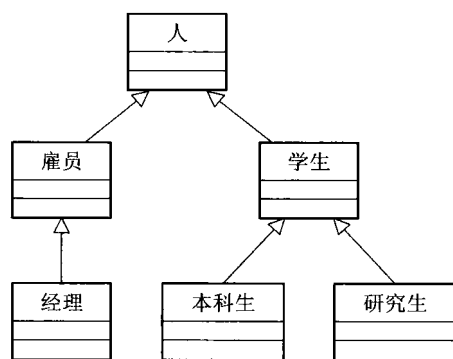


图 1.1 相关类层次图

有时,在程序组织上,即使是用有向无环图(Directed Acyclic Graph)也不足以组织一个程序中的概念,因为有些概念看上去是固有地相互依赖着。这种环形的依赖关系在现实生活中比比皆是,如母鸡与鸡蛋之间的关系即如此,如图 1.2 所示。针对这种环形依赖,程序设计时应当试图将这样的环形依赖限制在程序的局部,以免它们影响到整个程序的结构。

化解依赖图最好的方法之一，就是明确地分离接口(Interface)与实现(Implementation)，C++ 的抽象类(Abstract Class)就是实现这一功能的工具。

在 C++ 中，表现概念之间共性的另一种方式是模板(Template)。C++ 中提供了类模板与函数模板机制。一个类/函数模板阐明了与一组类/函数有关的共同特征，模板允许将类型参数带入这组类/函数模板中，使所生成的每个模板类/模板函数具有各自独特的特性。例如，应用中有一组类：一队军人、一队大雁和一队学生，它们有其共性及个性，利用 C++ 的类模板机制我们可自动生成这些类。类模板及所生成的模板类如图 1.3~图 1.7 所示。

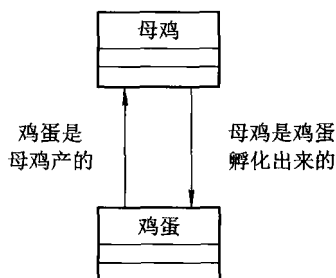


图 1.2 母鸡与鸡蛋之间的环形依赖关系图

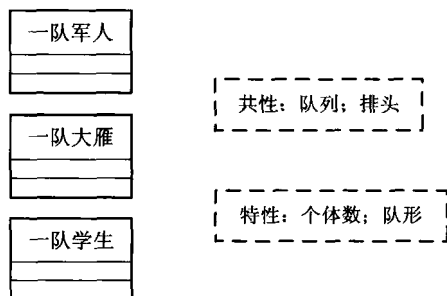


图 1.3 一队军人、一队大雁和一队学生类的共性与个性图

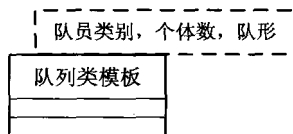


图 1.4 队列型类模板

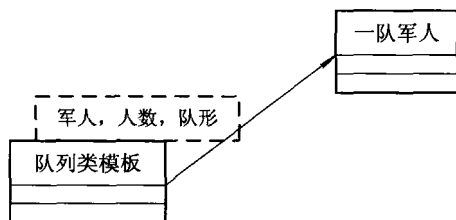


图 1.5 由队列类模板生成的一队军人模板类

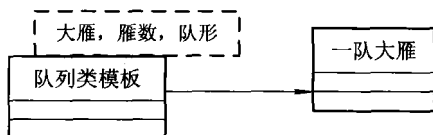


图 1.6 由队列类模板生成的一队大雁模板类

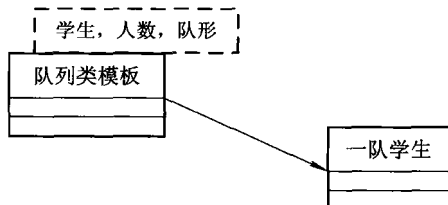


图 1.7 由队列类模板生成的一队学生模板类

程序设计中，我们抽取一队军人、一队大雁和一队学生类的共性，将其表达成队列型类模板，用其参数：队员类别、个体数和队形来表达其特性。当我们以相应的实参代入此类模板时，即可生成相应的具体模板类。

C++ 语言为支持面向过程、面向模块，特别是面向对象及类属程序设计提供了各种语言机制。当今，面向对象(Object-Oriented, OO)技术已是软件业公认的主流开发技术，上述 C++ 中这些重要的程序设计理念：类、派生类、继承、抽象类、接口与实现的分离及模板等重要的面向对象理念，正是 C++ 顺应应用的需求而产生的，又在应用中得到进一步的完善与发展。

学习 C++ 的根本目的是要成为一名优秀的 C++ 程序员,更高效地用 C++ 开发新系统和维护老系统。因此,上述 C++ 中这些重要的程序设计理念应成为我们学习 C++ 语言及进行 C++ 程序设计的灵魂与法宝。随着学习的深入,读者会逐渐地认识到: C++ 不仅是一种语言,更是一种编程思想,在我们学习与实践的历程中,它会伴随着我们一起成长!

小 结

C++ 语言是顺应应用的需求从 C 语言演化而来的,它是一种支持面向过程、面向模块、面向对象和类属程序设计范型的混合型语言。

学习与掌握 C++ 语言的根本目的是要成为一名优秀的 C++ 程序员,所以在学习 C++ 语言时,千万不要陷于语言语法细节的误区,应将注意力集中在 C++ 程序的组织与各语言机制的正确使用上。C++ 中为支持各种程序设计范型,特别是面向对象程序设计范型所提供的一些重要的语言机制与程序设计思想(如类、类的层次化组织、接口与实现的分离、模板等)应成为我们学习、使用 C++ 的重点,及进行 C++ 程序设计的灵魂。

练习题

1. 简述 C++ 语言的特点。
2. C++ 语言支持哪几种程序设计范型?
3. C++ 中有哪些重要的程序设计理念?