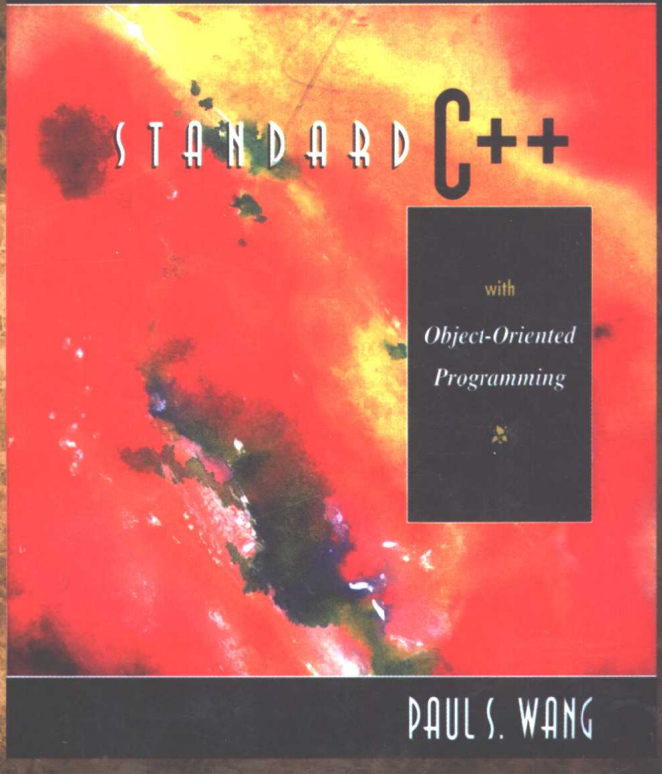
计 算 机 科 学 丛 书

原书第2版

标准C++与面向对象程序设计

Paul S. Wang 著 李健 等译



Standard C++
with Object-Oriented Programming
Second Edition

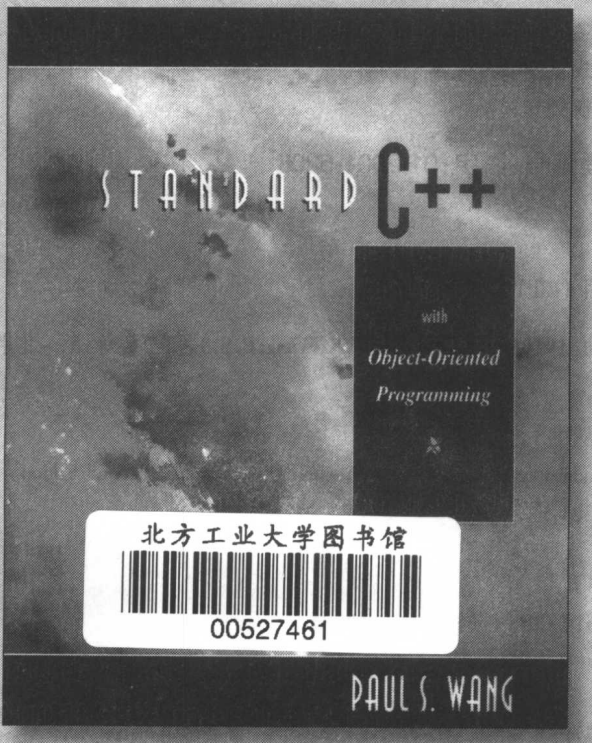
机械工业出版社
China Machine Press

计 算 机 科 学 丛 书

原书第2版

标准C++与面向对象程序设计

Paul S. Wang 著　　李健 等译



Standard C++
with Object-Oriented Programming
Second Edition

机械工业出版社
China Machine Press

本书包含了 C++ 语言面向对象程序设计的所有内容，包括类、操作符与功能重载、内联函数、单个和多重继承、虚函数和模板，以及名称空间、模板库、运行时类型标志和空闲存储管理操作符等。本书内容的组织和介绍均以简单、明确、易学为出发点。书中还有一些有趣的例子和具有挑战性的习题，这些都将帮助你一步一步地学习编写程序。

Paul S. Wang: Standard C++ with Object-Oriented Programming, Second Edition (ISBN 0-534-37131-0). Brooks/Cole

图书在版编目（CIP）数据

标准 C++ 与面向对象程序设计 /王鲍尔(Wang,P.S.)著；李健等译. -北京：机械工业出版社，2003.1

（计算机科学丛书）

书名原文：Standard C++ with Object-Oriented Programming, Second Edtion

ISBN 7-111-11384-5

Ⅰ.标… Ⅱ.①王…②李… Ⅲ.C 语言-程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2002)第 102868 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：朱 劼

北京第二外国语学院印刷厂印刷 · 新华书店北京发行所发行

2003 年 2 月第 1 版第 1 次印刷

787mm × 1092mm 1/16 · 27 印张

印数：0 001 - 5 000 册

定价：39.00 元

中文版序

程序设计是一种既抽象又具体的工作。说它“抽象”是因为要操纵那看不见又摸不着的位、字节和其他数据结构;说它“具体”是因为一切细节都要符合严格的规则。同一动作,同一反应,无一例外。软件设计人员必须像数学家那样具备高度的抽象思维能力。然而,单靠思维能力仍是不够的。一个好的软件设计人员还要像音乐家那样有熟练的技巧、丰富的经验及细心和耐心。

大型软件是人类有史以来所创造的最复杂的结构。软件的设计、制作、维护和改进需要庞大、昂贵的人力。面向对象程序设计(Object Oriented Programming,OOP)是新近发展的软件技术,它的应用能使软件的难度和费用大幅减低,因此已为世界软件产业带来革命性的突破。

OOP 提倡以软件对象来模拟实际。一个对象就像一个独立的小计算机,有明确的外部功能和内部运作。这种内外分隔可大幅减低软件的复杂性。以对象的相互联系合作而构成的程序更能降低软件的抽象度以及“命题领域”和“解答领域”的理解间隔。

标准 C ++ 是被最为广泛接受的一种 OOP 语言。本书对 C ++ 做了详尽的介绍,而且每个 C ++ 的特点和运用都举例进行了说明。一个缺乏 OOP 设计的程序最多是个传统程序,更坏的是被扭曲变形来产生类。本书把 OOP 的方法和设计融合在 C ++ 中一并说明。更重要的是解释如何将 C ++ 各项功能综合并运用来达到 OOP 的目的。

希望本书能满足国内电脑从业者及学生们的需要。同时读者可在作者网站上找到其他有关 OOP 的书籍。作者衷心盼望读者能给予批评和指教,来函请寄作者网站:

http://sofpower.com

http://monkey.cs.kent.edu/ ~ pwang

王士弘(Paul S. Wang)

译 者 序

C++ 是一种非常流行的面向对象程序设计(OOP)语言。它提供了编写程序过程中所有常用的工具。本书包含了 C++ 的 OOP 方面的所有内容,包括类、操作符重载和函数重载、引用、内联函数、单一和多重继承、虚函数以及模板。此外还有名称空间、模板库、运行时间类型标志和自由存储管理操作。

对于应用程序开发人员来说,本书提供的类库有助于完成许多任务,以前需要的大量工作,现在简化成了很少的代码。

经过三个月的艰苦努力,我们终于放下手中的笔,松了一口气,此时此刻,我们感到极度疲惫,但也难抑心中的激动,但愿给广大读者带来的益处能使这份良苦用心和心血得到补偿。

在本书的翻译和校对工作中得到了很多同仁的帮助和指导,在本书出版之际,对此表示衷心的感谢。

由于时间仓促,译文中难免出现错误和不当之处,敬请广大读者指正。[1]

李 健

〔1〕 参与本书翻译的人员有:李健、曹元大、付英华、于严兵、卢迪、陈天鹏、陈建强、王佳楠、李翊、王艳梅、王栋、苏蕾、祝烈煌、关芳、王大震、陈刚、李春玲、刘美华、田小佶、高颖、石日宝、王正红等。本书最后由李健(博士),付英华(硕士)统稿。——译者注

前　言

在面向对象程序设计(OOP)语言中,C++ 语言是最流行的语言之一,对象技术(OT)的核心方法使软件产业发生了革命性的变化。OOP 所建立的程序具有许多优点,例如,组织好、容易理解和修改灵活,并且可以在许多不同的场合重复使用等等。它减少了复杂性并且使软件的生产和维护更加经济。C++ 语言由国际标准化组织(ISO)和美国国家标准化组织(ANSI)共同进行了标准化。ISO/ANSI C++ 语言标准(ISO/IEC FDIS 14882)已经在 1997 年 11 月获得批准。

在标准 C++ 语言中引入了许多新的特性,它们包括布尔类型、异常、名称空间、运行时类型标识符、类型转换符表示法和带有通用算法的模板库。标准 C++ 语言也修改和扩展了许多已有特性,包括宽字符类型、模板和函数调用解析。一个新的 string 类使得字符串更加容易使用。

本书是《C++ with Object-Oriented Programming》的一个修订本,包括了标准 C++ 的所有内容并且保留了以前的内容。C++ 的构造与清晰、简明的 OOP 技术相匹配。语言机制独立地进行解释并再组合起来以达到 OOP 的目的。基于对象的、面向对象的和通用的程序设计技术在本书的实例程序中都有示范,以显示它们在实际中的应用。第 12 章还讲述了如何将 C++ 语言应用到 Web CGI 程序设计中。

面向对象

因为 C++ 语言效率非常高并且支持 OOP,所以它非常重要。因此 OOP 的概念和技术就作为 C++ 程序设计的集成部分来讲述。这样就将 OOP 概念带回了现实中,使得它们变得容易理解。面向对象很早就进行过介绍并且用许多完整的例子进行了说明。在这里将明确地讲述怎样用这些技术来解决问题,以及这些技术怎样使程序变得更加灵活,并可复用。

关键的面向对象的概念,如数据抽象、封装、信息隐藏、用对象处理的问题、通用化、继承和多态性,将在本书中进行全面的介绍。面向对象设计的基础知识也包含在其中。它们通过许多 C++ 的实例进行说明。例如本书包含了一个银行账户的例子,这个例子贯穿了许多章节,并且不断地有新的内容加入。

动手实践的方法

学习编程的最好方法就是去写程序。介绍了清晰的概念和很好的例子之后,本书就鼓励你尽快编写有趣的程序。第 1 章和第 2 章介绍 C++ 语言的重要组成部分和面向对象程序设计,从而使你快速入门。关于对象内容的介绍(如 C++ 编程技巧)和风格向导将帮助你开始学习编程。

在第 5 章介绍一个袖珍计算器模拟程序。这个程序随着新章节和例子的引入而不断更新。最后,在第 13 章就会完成一个逼真的小计算器程序了,它包含了许多重要的 OOP 概念和C++ 结构。

内容的全面概括

本书全面而深入地介绍了标准 C++ 语言。根据学生的需要，将标准 C++ 描述为一个完整的自我解释的语言。本书还包含了许多不断更新的例子，提供了许多机会重新访问熟悉的代码，并且将注意力集中在引入的新的概念和特性上。

可以将本书作为一门课程的教材。除了基本的主题外，本书还包括标准库、I/O 流类、模板、标准模板库(STL)、通用算法、程序组织、头文件的使用、错误和异常处理、预处理和编译(在UNIX上和在 PC 机上)。

面向对象的重点主题包括以下内容：建立软件对象，比较内部工作与外部行为，用封装减少复杂性，在已有的类上派生出新类，编写通用的、可以在许多情况下重用的代码，创建使用了插入兼容的软件“黑匣子”，并且建立与多种类型对象一起工作的多态性过程和对象。同时也包含面向对象的设计技术、方法和应用。

OOP 使得编程变得简单

标准 C++ 是一个巨大并且复杂的语言。它在新的 OOP 的概念和支持 C++ 结构方面很容易使人迷惑。只有通过努力才能使这个复杂的课程变得容易掌握和易于理解。这个过程是以简单的话题和关键的概念作为基础的。然后，以逻辑的顺序添加更高级的主题。书中还描述可能遇到的问题并且提供了答案。清晰有趣并且实用的例子说明了如何编写面向对象的程序和怎样应用所介绍的概念和技术。

继承是 OOP 的一个关键特性，它对于初学编程的程序员来说比较难于理解。书中绘制了一个派生类清晰的结构图以及它与一个基类的关系，派生类是建立在这个基类之上的。而且明确地提供了派生的原理和它的恰当用法。多重继承部分也有清晰的好例子。

多态性和插入兼容是 OOP 的核心技术，应该非常熟练地使用它们。在本书中用了整整一章来描述这两个主题，提供的材料足以向最聪明的学生提出挑战。

Web 应用

用 C++ 编写的 CGI 程序可以快速、高效地执行。本书用一章简练地描述了 HTML 格式和 CGI 程序设计的过程，这一章是 OOP 概念和 C++ 技术的一个很好的应用。该章还给出和解释了一个用于 CGI 编程的 C++ 类库，以及怎样使用这个库进行 CGI 程序设计。

灵活的用法

本书可以作为一门程序设计课程的教材，课程对象可以是大学三年级、四年级的本科生或刚入学的研究生。本书假设读者没有 C 语言基础，但是最好有一定的程序设计经验，这样可以更好地理解软件的复杂性。如果你已经具有 C 语言或者 ANSI C 语言方面的知识，那么它将减少你的学习量。对于一个初级的读者来说，前面章节中的内容应该仔细研究。高级的主题，如用户定义的自由存储管理、CGI 编程(第 12 章)和面向对象设计(第 13 章)可以忽略不读。对于一个高层次

读者来说,第 1 章和第 14 章可以作为背景材料进行阅读,从而为其他的主题留出更多的时间。

就像在每章结尾的习题中建议的那样,对于那些有实际程序设计经验的人,应强调通用程序设计、模板编写和面向对象程序设计项目。在这种情况下,应该早一点学习第 13 章中的程序设计内容。

无论何时,只要觉得合适就可以介绍(或者让学生自己阅读)第 14 章中的预处理和编译内容。第 2 章到第 11 章是本书的核心内容,它们具有挑战性并且适合于任何读者阅读。

本书也可以作为 OOP、数据结构或用 C++ 实现面向对象设计课程的一个很好的补充。

对读者系统的要求

C++ 语言是一个独立的系统。示例程序可以运行在任何具有 C++ 实现的地方,如工作站、PC 机和多用户服务器。第 14 章给出了关于预处理、编译和执行程序的一些一般信息。自由软件基金会为我们提供了 **g ++** ,它是一个很好的 C++ 实现,你可以在下面的站点免费下载:

```
http://www.gnu.ai.mit.edu/software/gcc    (用于 UNIX 用户)
http://www.delorie.com/djgpp/             (用于 PC 用户)
```

容易查找的参考

本书采用了循序渐进的论述方法,在旧的概念上建立新的概念使得本书变得更加容易理解。然而,本书也是一个很好的参考工具,书中组织了许多信息使你可以很容易进行参考,如表、图、语法注释、例子和总结。所有 C++ 语言的重要结构都收集在附录 A 中以便于快速查找。附录 B 总结了一些特殊成员函数的用法。其他的一些附录包括调试、库函数和 C++ 语言与 C 语言的混合用法。

示例程序包

在本书中,概念和编程结构都有足够的实例来说明。示例程序包按照章节来组织,大约有 400 个文件包含了完整的用来编译的源代码。用于 UNIX 和 PC 的完整的示例程序包都可以在下面的站点中得到:

```
www.brookscole.com
```

引　言

面向对象程序设计(OOP)无论在产业界还是在学术界都已经成为软件发展史上的典范。它相对于传统的程序设计方法有许多优点,如容易构造软件,容易维护、修改和重用等。面向对象(OO)的技术知识将使你成为一个更加出色的程序员。

许多程序设计语言支持 OOP。有一些语言(如 SMALLTALK)是彻底的面向对象语言。其他一些语言,如 C++ 、Java 和 CLOS(公用的 Lisp 语言对象系统),既支持面向对象的程序设计,也支持传统的面向过程的程序设计。在它们当中,C++ 是一个流行的面向对象程序设计语言。人们广泛接受 C++ 主要有两个原因。第一,C++ 语言提供了一组设计良好的机制来全力支持 OOP。第二,C++ 是一个国际标准语言,并且是 ANSI C 语言的一个扩展。因此,无论对于新的程序员还是有经验的程序员来说,想要全面了解 OOP 的概念和技术,从 C++ 语言入手无疑是一种比较好的选择。

什么是 OOP

OOP 的核心思想是使用软件对象(object)来编写程序。一个对象可以看做是一个带有数据和程序的自我包含的计算实体。在现代的工作站中,如窗口、菜单和文件夹等经常都是以软件对象的形式表现出来。但是对象可以应用到许多种程序中。一个对象可以是一个航班预定记录、一个银行账户、甚至是一个汽车引擎。在一个汽车内,引擎对象可以包括一些数据和程序,其中的数据是用来描述引擎物理属性的,而程序则是用来管理引擎的内部工作并且处理它与其他部件(也是软件对象)之间的交互的。

一个个人管理系统可以将工程师、秘书和经理看成是软件对象。一个航空事件控制系统可以将飞机跑道、航线和乘客出入的门看成是软件对象。这样,在面向对象的程序设计中,软件对象就和应用领域中的真实对象密切相关。这种对应关系就使得计算机程序变得容易理解和操纵。相反,传统程序设计处理字节、变量、数组、索引和其他程序设计中用到的工具,它们很难与手头的实际问题联系起来。传统的程序设计还主要集中在称为算法的一步一步的过程上来完成任务。出于这些原因,传统程序设计可以称为面向过程的程序设计。

OOP 的优点

编写一个大的计算机程序需要用到许多复杂的机制,设计、实现、测试、维护和修改一个大型系统的费用非常高。因此,找出一种方法使这些任务变得简单,并且减小错误发生频率就显得非常重要了。从这一角度考虑,OOP 具有相当大的潜力。

由于这些明显的原因,OOP 具有下面这些主要优点:

- 简单:因为软件对象为应用领域中的实际对象建立了模型,所以减少了程序的复杂性并且

使程序结构变得清晰、简单。

- 模块化：每个对象组成了一个包含内部工作的独立实体，这些工作是从系统中其他部分分离出来的。
- 可修改性：在一个面向对象的编程过程中很容易对数据和过程进行小的修改。一个对象的修改对于程序中其他部分没有任何影响，只要对象的外部行为不变就可以了。
- 可扩展性：添加新的特性或对改变操作环境做出响应可以通过定义一个新的对象来实现。这个新的对象是对已经存在的对象的一个扩展。
- 灵活性：面向对象的程序在适应不同情况时显得非常灵活，因为不用修改对象就可以改变对象之间的交互模式。
- 可维护性：对象可以分开维护，可以很容易地确定错误发生位置并且修改它，在必要的时候甚至还可以在程序中添加“铃声和口哨”以作为标志。
- 可重用性：对象可以在不同的程序中重用。例如，一个表对象可以在任何需要进行表查找的程序中使用。这样，如果需要在短时间内从设计草图中编出程序，就可以利用预先做好的和预先测试好的部分程序。

OOP 的概念

面向对象的关键概念是过程与数据的连接。这个概念改变了传统程序中数据和程序分隔的状态。将程序和数据包装在一起叫做封装，并且封装的结果就是一个软件对象。例如，在一个图形用户窗口系统中，一个窗口对象(图 1)包括窗口的物理大小，在屏幕中的位置、前景和背景颜色、边界风格，以及其他相关数据。封装这些数据的程序可以用来移动窗口，重新设置窗口的大小，改变窗口颜色，显示文本，将窗口缩小为一个图标等等。用户界面程序的其他部分可以简单地通过发送一个定义好的消息给一个窗口对象从而调用一个窗口对象以便完成这些任务。剩下的就由这个窗口对象来执行适当的动作并且更新它的内部数据。完成这些任务的确切方法和内部数据的结构与对象之外的程序没有关系。一个对象可以理解由消息集合组成的公共接口，并且这个公共接口定义了使用那个对象的方法。内部细节的隐藏使得一个对象变得抽象起来，有时这些技术被称为数据抽象。

将公共接口与内部操作分开并不难理解。实际上，在我们日常生活中它是非常普通的现象。例如，考虑一个银行出纳员。客户去任何一个银行时，他对出纳员讲的都是同样一组“消息”：账号、存款、提款、余额等等。每个银行或出纳员实际保存记录或在内部执行任务的方法与客户无关。这些经过检验而且可靠的原理在各个层次上简化了交易过程，并且给编写程序带来了同样的好处。当一个面向对象(OO)的程序运行时，建立了许多对象，这些对象发送完消息后就被销毁。这些仅仅是对象上允许的操作。一个对象中的内部(私有)数据或过程与公有数据或过程是无关的。将对象的私有机制与对象

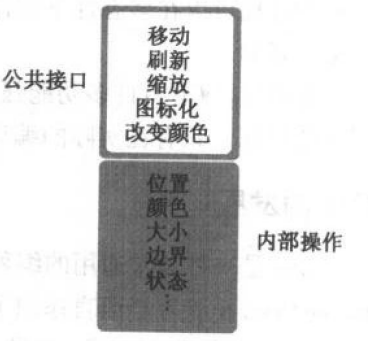


图 1　一个窗口对象

外部的例程分离开来大大地减小了一个程序的复杂性。

常常有这样的场合,程序中需要同种类型的多个对象。例如,许多窗口经常需要显示在同一个工作站的屏幕上。通常,一个给定类型的对象是一个*类*的*实例*,在这个类的定义中详细说明了这些对象的私有(内部)操作,以及它们的公共接口。这样在 OOP 中,可以为每种编程过程需要的不同对象定义类。类就成为制作一个特殊对象的蓝图。一个类的定义和*初始值*可以用来创建类的实例(对象)。这个操作就是我们所知道的*实例化*。术语*对象*有时可以指一个类,有时可以指一个类的实例。

OOP 提供了一些简单的方法来在其他对象的顶层构造类。有两个主要方法:*合成*和*继承*。合成允许现有的对象作为建造其他对象的一个组件。例如,一个计算器对象可以由一个算术单元对象和用户接口对象组成。继承是 OOP 的一个主要特性,它允许一个相似的或相关的对象(*派生对象*)来源于另外一个对象(*基对象*)。在 C++ 中,它是通过类派生来实现。一个派生类可以继承它的基类的属性,也可以添加它自己的数据和例程。例如,一个图形窗口、一个文本窗口以及一个终端仿真器窗口都可以从一个基窗口类中派生。除此之外,一张支票、一张发票以及一个申请表都可以从一个基商业表单类派生。继承允许在相似或相关的类中提取它们的共性。它也允许面向对象的软件库中的类应用于不同的或无法预见的目的。在这些例子中,派生类都是从一个基类中派生出来的,因此它叫做*单一继承*。一个类也可以从多个基类中派生出来(就像继承了父母的特征一样),从而获得*多重继承*。

OOP 的另外一个特点是*重载*,也就是给操作符和函数分配多重含义。这种操作符和函数由于具有多种不同任务而重载。例如,操作符 +,它的主要任务是对数字进行加法运算,但是当重载时,它也可以用于将两个字符串连接到一起。重载可以简化程序,并且可以将同样的操作应用到许多不同类型的操作对象中。

除了编写基于对象的程序(object-based programming)之外,OOP 更进一步地鼓励用户从已有的类中派生出新类,鼓励用户建立对象的层次结构,并且还鼓励用户在*多态的*(polymorphic)程序中使对象可交换。多态性(polymorphism)是指同一个过程与不同对象共同工作的能力,这些对象就像可交换的黑匣子(black box)。它允许不同(但相关)对象的创建过程在一组操作下相容。如果一个解决问题的程序仅仅使用这些操作的话,那么它将为所有这类对象工作。例如,一个驾驶汽车的过程可以在多态性下制作,这样它就可以应用到所有类型的驾驶的汽车中。这样的过程就是多态的。

很明显,OOP 有许多功能强大的概念。C++ 语言具备一组设计非常好的语言机制来帮助达到这些目标。只有在实际的编程过程中,才能充分体验到这些概念的深层含义。

C++ 的发展

C++ 是一种非常通用的编程语言。它可以与 ANSI C 语言相兼容。在 1980 年,AT&T 公司的 Bjarne Stroustrup 为 C 语言添加了类和一些其他的特性,从而出现了我们所知道的带有类的 C 语言。在 1983 年和 1984 年,又做了一些扩展,加入了操作符重载和虚函数,从而出现了一种叫做 C++ 的语言。通过进一步的改良之后,在 1985 年 C++ 语言变得非常有用。在 C++ 语言应用日益

广泛的同时,新的特性和扩展也不断地添加到其中。这些新的特性和扩展包括多重继承、模板、异常处理、名称空间和运行时类型标志。C++ 语言的标准化过程是从 1987 年开始的,并且在 1995 年发布了一个标准草案。最终,在 1997 年 11 月通过了 ISO/ANSI C++ 语言标准(ISO/IEC FDIS 14882)。由 Bjarne Stroustrup 编写的《The C ++ Programming Language(3rd ed)》是一本很好的参考手册,在该书中你可以了解到标准 C++ 语言的一些详细内容。

当面向对象技术变为主流的时候,C++ 的使用也日益流行。C++ 既支持传统的面向过程的程序设计,也支持面向对象的程序设计(OOP)。这一点非常重要,因为大多数现实中的程序需要的是面向过程和面向对象的混合体。今天,C++ 成为一种非常流行的 OOP 语言。所有主要的计算机销售商都在他们的机器中提供 C++ 。C++ 的实现包括 Hewlett-Packard aC++ 、Sun Microsystems Visual Workshop C++ 、GUN g++ (来自自由软件基金会的C++)和 Microsoft Visual C++ 。

C++ 语言的特性

C++ 语言提供了编写程序过程中所有常用的工具。C++ 的主要 OOP 特性包括类、操作符与函数重载、引用、内联函数、单个和多重继承、虚函数和模板。此外还有名称空间、模板库、运行时类型标志和空闲存储管理操作符。

C++ 类通过封装数据成员及封装函数成员来定义软件对象。成员可以是私有类型、保护类型或公有类型。它为定义公共接口和一个对象的私有域提供了一个非常便利的方法。

类派生是 C++ 语言用于继承的一种机制,它支持类型之间的相互关系和代码的重用。单一继承和多重继承都是可能的。虚函数机制支持多态性并且允许在一个统一的外部接口下定义可互换的对象。C++ 语言抽象基类机制允许用户在相关的类中提取具有共性的部分,就像规划它们的接口一样。

模板工具支持类型参数化,它允许一个函数或一个类使用一个还没有确定的类型来定义,只有当一个模板放到实际程序中应用时才能确定并修改这个类型。模板支持为任何数据/对象类型编写通用程序。

作为 C++ 标准库的一部分,标准模板库(STL)支持用于容器类和通用算法的模板。C++ 还提供了一个基于对象的 I/O 库。I/O 操作符(> > 和 < <)使得接收和显示数据变得简单和直接。这些操作可以重载到用户定义的输入和输出对象中。

在本书中,你将学到 C++ 中所有的特性,并且学会怎样有效地使用它们来编写面向对象的程序。

本书的组织

本书介绍了标准 C++ 和它在 OOP 方面的有效使用方法。C++ 是一种从基础到高级的完整语言。在编程过程中也需要另外的编程语言的知识和足够的编程经验。如果已经具有 C 语言方面的知识,则会减少编程时的工作量。在编写程序的过程中,访问一个 C++ 编译器是必须的。本书适合作为程序设计课程教材,也适合作为大学本科生或刚入学的研究生的一门关于 C++ /OOP 课程的教材。

本书内容的组织和介绍均以简单、简洁、易学为出发点。学习编程的最好方法就是写程序。这样你可以尽早地开始学习编程序了。本书中还有一些有趣的例子和一些具有挑战性的习题，这些内容都将帮助你熟悉一步一步地编写程序。为了使你很快地掌握 C++ 语言，本书前面的章节都是一些基础知识，并且给出了一些有关 C++ 语言的基本信息，其中包括对象。本书中还介绍了许多例子，基于对象的解决问题向导、编程技巧和推荐使用的格式。所有这些内容都将加速你学习本书的过程。当你学完本书之后，你将对 C++ 语言和它在 OOP 方面的应用有一个全面深刻的了解。关于 Web CGI 编程的一章使得 C++ 语言变得更加有趣和重要。本书中还包含了一些高级的内容，这些部分可以满足一些高级程序员的需要。

作为一种支持 OOP 的语言，C++ 比一个通常的面向过程的语言（如 C 语言或 Pascal 语言）具有更多的特点。它包含了各种结构，这一点非常重要。完成这一功能不仅需要解释每种结构的语法和语义，并且还需要举例说明怎样很好地利用这些结构。此外，还强调了一些合成不同结构的有效方法来解决实际问题。本书介绍了面向对象的程序设计，并且将它的内容综合在一起展现出来，因此学习面向对象的程序设计的过程自然就成为了一个学习编程的过程。在本书中，通过具体例子说明了怎样将一个问题划分成一些容易处理的逻辑块，还说明了程序怎样模块化、面向对象、分别测试和重用。许多类和程序都是以重用的形式给出的，因此它们可以在许多情况下应用。

目　录

第1章 C++入门之一:基础

第1章为刚开始使用C++语言的程序员介绍一些基本知识,第2章介绍基于对象的编程方法。这两部分初级内容都经过了很好的组织,以使读者有一个好的开端。

我们从整体程序结构和基础的知识开始介绍,包括程序结构、常量、变量、表达式、数组、函数和简单的输入输出。介绍控制流程的结构if、while、for和do-while。我们建议读者在自己的系统中测试一下书中给出的例子和它们的变量,这样会帮助你更快更好地理解本书中的内容。

本章全面而简洁地描述了基础部分的细节,为后面更有意义的内容和中心主题做准备。如果你有一些C语言的背景,就可以快速地通读本章。但由于C语言和C++语言的程序设计方法,甚至是基础语言都存在很多差异,建议你认真地看一下本章中的例子。如果以后需要某一细节,你可以在本章中很容易地找到它。

1.1 面向对象程序设计的结构

程序是一组用一种编程语言编写的指令,用来解决一个给定类型的问题,或者执行各种特殊任务。C++语言编程的源代码由函数(function)和类(class)组成。函数用于编写解决问题的程序,类用于描述表示实体的对象(object),这些实体在执行要求的任务时会相互作用。对象用于准确地模拟问题领域中的实际实体或逻辑实体。面向对象程序设计的一个很重要的方面就是在解决问题过程中确定这些实体和它们之间的相互作用。

一个函数接收参数(argument),按预定的算法执行有关参数的计算,并返回结果。一个函数调用(function call)激活(或者调用(invoke))一个函数,并向这个函数传递变量,产生它返回的值。

一个类是对象的蓝图,它描述了对象的数据结构和操作。类一经定义,就能够建立属于这个类的对象并在程序中使用。一个类提供一个名称,在该名称下面的数据和函数成员都集合在一个单元里,这个计算单元可用于独立操作程序中的其他部分。通过使用对象,一个很大的程序可以分解为许多小的、独立的、相互作用的单元。面向对象可以显著地降低程序的复杂性,提高灵活性,增强可重用性。

对于任何一个程序,必须有一个名为main的函数。主函数main是程序的开始点(entry point),是程序开始执行的地方。除这个基本结构外,C++非常灵活。函数和类都可以放置在一个或多个源代码文件中。

因此,在C++中,程序由函数进行编码,这些会在本章中介绍;对象由类进行定义,这些将在第2章中介绍。