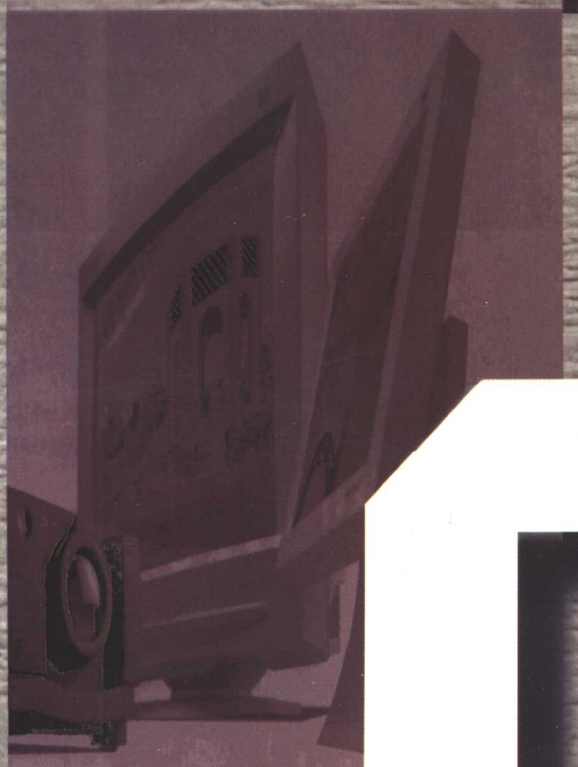

吕凤翥 著

C++语言基础教程

清华大学出版社
http://www.tup.tsinghua.edu.cn

(京)新登字 158 号

内 容 简 介

该书全面和系统地讲述了C++语言的基本概念、基本语法和编程方法，较详尽地讲述了C++语言面向对象的重要特征：类和对象、继承和派生类、多态性和虚函数等内容。本书具有丰富的例题，每章后面备有相当数量的练习题和作业题。

该书通俗易懂，由浅入深，突出重点，偏重应用。本书不仅可用作大专院校理科学生C++语言课程的教材，还适用于广大电脑爱好者选作学习C++语言的教材或参考书。

图书在版编目(CIP)数据

C++语言基础教程/吕凤翥著. —北京：清华大学出版社，1999.4
ISBN 7-302-03321-8

Ⅰ.C… Ⅱ.吕… Ⅲ.C语言-程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(1999)第 00832 号

出 版 者：清华大学出版社(北京清华大学学研大厦，邮编 100084)
http:// www.tup.tsinghua.edu.cn
印 刷 者：北京市清华园胶印厂
发 行 者：新华书店总店北京发行所
开　　本：787×1092 1/16　印张：21.75　字数：510 千字
版　　次：1999 年 3 月第 1 版　2001 年 6 月第 9 次印刷
书　　号：ISBN 7-302-03321-8/TP・1787
印　　数：80001～100000
定　　价：26.00 元

前　言

本书是在多年来从事C++语言教学的基础上编写的。书中总结了教学中的经验和教训，并针对学生在学习过程中遇到的困难和提出的问题。该书的特点是：通俗易懂，适于自学；由浅入深，便于理解；概念明确，语言简洁；例题丰富，内容全面；重点突出，难点详解。

本书较全面较系统地讲述了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，并且可以达到使用C++语言编写简单程序的目的。

本书共分9章。前4章讲述了C++语言的基本词法和语法规则。包括字符集、词法规则、程序结构、运算符和表达式、各类语句、函数格式和调用方式、作用域及存储类等内容。这些内容中，C++语言只是对C语言中的相应内容进行一些改进和补充，与C语言的内容很相近。第5章至第8章这4章中讲述了C++语言所支持的面向对象的程序设计方法的内容。这里主要包含有类和对象的概念和定义格式、对象的赋值和运算、继承性和派生类、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。这部分内容是C语言所没有的。第9章讲述了标准文件的读写函数和对一般文件的操作。本书每章都有较多的例题，全书共有164个例题。例题的针对性较强。每章后面都备有相当数目的练习题和作业题。读者通过练习题可以检查自己对本章所学内容掌握的情况，练习题的覆盖面很大。作业题可以练习所讲过的主要内容，包含概念性的训练和方法技巧训练。初学C++语言的人，一是要弄清基本概念，二是要多看程序，从中学习方法和技巧，从而积累编程经验。本书提供了这两方面的训练。

本书的所有例题和作业中分析输出结果程序都曾在Visual C++ 5.0版本的编译系统下运行过，在其他版本的编译系统一般也都可以运行。

本书可选作大专学院理科学生的教科书，也可用作教员和学生的参考书。本书还适用于广大电脑爱好者自学C++语言的教材。

本书在编写过程中，查阅了许多有关外文资料和说明书，并阅读过一些翻译的书籍，现谨对这些书的作者和译者提供的帮助表示最衷心的感谢。本书全部内容由本作者编写，

由于时间仓促，水平有限，书中难免会有缺点和错误之处，请广大读者提出宝贵的意见。

谢谢喜欢阅读本书的读者！

作　者

1998年8月底于

北京大学燕北园

目　录

第1章　C++语言概述

C++语言是一种应用较广的面向对象的程序设计语言，使用它可以实现面向对象的程序设计。本书主要讲述C++语言的特点和语法，介绍使用C++编程的方法。因此，为了对C++语言的特性的了解，在这章中首先介绍一些有关面向对象程序设计的基本概念。因为面向对象的设计与面向过程的设计是有很大区别的，面向对象的程序设计是在面向过程的程序设计的基础上一个质的飞跃。学习C++语言首先要认识它面向对象的特性和实现面向对象的方法。

1.1　面向对象程序设计的基本思想和有关概念

1.1.1　面向对象的由来和发展

下面回顾一下计算机语言的发展过程，看面向对象的方法是如何产生的。

早期的汇编语言比机器语言(二进制码)方便得多。

20世纪50年代中期，出现了高级的程序设计语言FORTRAN，它在计算机语言发展史上具有划时代的意义。该语言引进了许多现在仍然使用的程序设计概念，如变量、数组、循环、分支等。但是，该语言在使用中也发现了它的一些不足。例如，不同部分的相同变量名容易发生混淆等。

20世纪50年代后期，高级语言Algol在程序段内部对变量实施隔离。Algol 60提出了块结构的思想，由"Begin ... End"来实现块结构，这样就不会造成不同块内同名变量相互混淆，对数据实行了保护。这实际上也是一种初级的封装。

20世纪60年代开发的Simula 67，它是面向对象语言的鼻祖。它将Algol 60中的块结构概念向前推进了一大步，提出了对象的概念。对象是代表着待处理问题中的一个实体，在处理问题过程中，一个对象可以某种形式与其他对象通信。从概念上讲，一个对象是既包含有数据又包含有处理这些数据操作的一个程序单元。Simula语言中也使用了类的概念，类是用来描述特性相同或相近的一组对象的结构和行为。该语言还支持类的继承。继承可将多个类组成为层次结构，进而允许共享结构和行为。

20世纪70年代出现的Ada语言是支持数据抽象类型的最重要的语言之一。数据抽象是一种数据结构及作用在数据结构上的操作组成的一个实体。把数据结构隐藏在操作接口的后面，通过操作接口实现外部的交流。对外部来讲，只知道做什么，而不知道如何

做。再将类型扩展到数据抽象上，即将某种类型的操作汇集起来作为一个整体看待，并与该类型一起看作一个独立的单元，构成了抽象数据类型。因此，可以说，数据抽象类型是数据抽象封装后的类型。它包含了该类型下的操作集和由操作集间接定义的数据类型的值集。Ada 语言中面向对象的抽象结构是包。它支持数据抽象类型、函数和运算符重载以及多态性等面向对象的机制。但是，Ada 语言不是全面地支持继承，因此人们常称它为一种基于对象的语言。

后来出现的 Smalltalk 语言是最有影响的面向对象的语言之一。它丰富了面向对象的概念。该语言并入了 Simula 语言的许多面向对象的特征，包括类和继承等。在该语言中，信息的隐藏更加严格，每种实体都是对象。在 Smalltalk 环境下，程序设计就是向对象发送信息，这个信息将表示为一种操作，如两个数相乘，创建一个新类的对象等。Smalltalk 语言是一种弱类型化的语言，一个程序中的同一个对象可以在不同时间内表现为不同的类型。

20 世纪 80 年代中期以后，面向对象的程序设计语言广泛地应用于程序设计，并且有许多新的发展，出现了更多的面向对象的语言。归纳起来，大致可分为如下两类：

1. 开发全新的面向对象的语言

其中具有代表性的全新的面向对象的语言有 Object-C，它是在 C 语言上扩展而成的，它是 Smalltalk 语言的变种。Eiffel 语言除了有封装和继承外，还集成了几种强而有力的面向对象的特征，它是一种很好的面向对象的语言。Smalltalk 80 语言经历了多次修改和更新，新版本有很大改进。这类全新的面向对象的语言学习起来要从头开始。

2. 对传统语言进行面向对象的扩展

这类语言又称混合型语言。它的代表有 C++语言。C++是于 20 世纪 80 年代早期由贝尔实验室设计的一种面向对象的语言。它是在 C 语言的基础上增加了对面向对象程序设计的支持。这类语言的特点是既支持传统的面向过程的程序设计，又支持新型的面向对象的程序设计。对于一些已经较好地掌握了 C 语言的人来讲，学习 C++语言相对容易一些。另外，C++语言具有 C 语言的丰富的应用基础和开发环境的支持，普及起来也相对快些。这些就是 C++语言当前得以广泛应用的主要原因。

1.1.2 抽象在面向对象中的作用

学习面向对象的语言首先要搞清楚抽象在面向对象程序设计中的重要作用。

1. 抽象的概念

从前面介绍的计算机语言发展的历史来看，语言所提供的抽象支持程序在不断地提高。面向对象的程序设计比面向过程的程序设计要更加强调抽象的重要性。讨论面向对象的程序设计问题不可避免地要对抽象进行研究。

什么是抽象？一般地讲，抽象是通过从特定的实例中抽取共同的性质以形成一般化的概念的过程。抽象是对某个系统的简化的描述，即强调了该系统中的某些特性，而忽略了一部分细节。对系统进行抽象的描述称为对它的规范说明，对抽象的解释称为它的实现。抽象是具有层次的，可分高层次抽象和低层次抽象两大类。高层次抽象将其低层次抽象作为它的一种实现。

抽象是人们在理解复杂现象和求解复杂问题中处理复杂性的主要工具。

2. 面向对象抽象的原理

面向对象抽象的原理有4个，它们分别是：数据抽象，行为共享，进化和确定性。这4个原理概括了面向对象计算的本质。

(1) 数据抽象：它为程序员提供了一种对数据和为操作这些数据所需要的算法的抽象。数据抽象包含了两个概念：模块化和信息隐藏。这两个概念是相互独立的又是密切相关的。

模块化是将一个复杂的系统分解为若干个模块，每个模块与系统中某个特定模块有关的信息保持在该模块内。一个模块是对整个系统结构的某一部分的一个自包含的和完整的描述。模块化的优点是便于修改或维护，系统发现问题后，可以确定问题出在哪个模块上。这种模块化的设计方法构成了面向对象计算的本质。

信息隐藏是指将一个模块的细节部分对用户隐藏起来，用户只能通过一个受保护的接口来访问某个模块，而不能直接地访问一个模块内部的细节。这个接口一般由一些操作组成，这些操作定义了一个模块(或称实体)的行为。这是复杂问题处理中的一种主要工具。另外，在支持信息隐藏的系统中，错误的影响也通常被限制在一个模块内，增强了系统的可靠性。

数据抽象包含了模块化和信息隐藏这两种抽象，这是面向对象方法的核心。

(2) 行为共享：支持行为共享是面向对象程序设计的第二个原理。行为是数据抽象引进的概念，行为是由实体的外部接口进行定义的。行为共享是指许多实体具有相同的接口，这将增加系统的灵活性。例如，同样的一个操作(如显示，即行为)，被系统中的几个实体共享，各个实体对该操作的实现可能不同。这就是行为共享的含义。这种行为共享实际上是增强了在一个系统中的抽象。

分类和层次分类是支持行为共享的最为明显的方式。行为共享是面向对象计算的另一个重要的概念，实现面向对象方法的一个重要的任务是对进行分类的研究。

一个分类是由一组实体共同的行为而构成，因此一个特定分类中的所有实体而将共享共同的行为。层次分类更是一种普遍的行为共享形式。层次分类允许一个分类包含另一个分类，层次分类是分类的求精。例如，A分类被包含在B分类之中，A分类和B分类各自有特定的行为，而B分类将共享这些行为。

(3) 进化：它是面向对象计算的第三个原理。进化是考虑到实际中的需求会很快地发生变化。面向对象的方法要支持进化过程就是要适应可能发生的不断变化，这是需求进化。进化的另一个方面是进化式的问题求解。这种观点是从开始到最终结果是以一种增量的方法逐步地对问题进行求解。特别是对于最终目标不能很好定义的问题，这种方法更具有吸引力。进化的问题将涉及到一个系统从开始直到后续的维护这一整个生命期。

(4) 确定性：这里确定性是指用于描述一个系统确定的行为。一个确定的系统应该确保其中每个行为项都有一个确切的解释，系统不会因不能响应某一行为而失败。这对一个大型系统或者复杂系统显得更为重要。确定性与类型的正确性有关，这实际上就是要求在一个系统中不会出现类型方面的错误。在面向对象的系统中，特别是行为共享和进化等机制增加了确保确定性的困难。在确定某个行为项是否有一个解释是可能的，但要确定这

个解释是不可能的，因为解释可能会随时间而变化。

1.1.3 面向对象计算的基本特征

前面介绍了抽象的概念和面向对象的抽象的4个原理。在此基础上进一步介绍面向对象计算的基本特性，实际上还是讲解一些面向对象的基本概念。

面向对象的系统包含了三个要素：对象、类和继承。这三个要素反映了面向对象的传统观念。

面向对象的语言应该支持这三种要素。首先，应该包括对象的概念。对象是状态和操作的封装体，状态是记忆操作结果的。满足这一点的语言被认为是基于对象的语言。其次，应该支持类的概念和特征，类是以接口和实现来定义对象行为的样板，对象是由类来创建的。支持对象和类的语言被认为是基于类的语言。最后，应该支持继承，已存在的类具有建立子类的能力，进而建立类的层次。支持上述三个方面的语言称为面向对象的语言。按这一标准来衡量，Ada语言是基于对象的语言，Clu是基于类的语言，而Simuls和Smalltalk是面向对象的语言。C++也是面向对象的语言。

下面将对象、类和继承这些面向对象方法中特别重要的概念解释一下。在本书后面讲解C++语言的过程中还会反复讲解这些概念，因为它们是理解和掌握面向对象程序设计语言的关键。

1. 对象

什么是对象？回答这个问题似乎很容易，但是实际上又是很难的。在不同领域中对于对象将有不同的解释。一般地认为，对象就是一种事物，一个实体。在面向对象的领域中，如何理解对象呢？最好从下面两个角度来理解它，一是从概念上讲什么是对象，二是在实际系统中如何实现一个对象。

从概念上讲，对象是代表着正在创建的系统中的一个实体。例如，一个商品销售系统，像顾客、商品、柜台、厂家等都是对象，这些对象对于实现系统的完整功能都是必要的。

从实现形式上讲，对象是一个状态和操作(或方法)的封装体。状态是由对象的数据结构的内容和值定义的，方法是一系列的实现步骤，它是由若干操作构成的。

对象实现了信息隐藏，对象与外部是通过操作接口联系的，方法的具体实现外部是不可见的。封装的目的就是阻止非法的访问，操作接口提供了这个对象的功能。

对象是通过消息与另一个对象传递信息的，每当一个操作被调用，就有一条消息被发送到这个对象上，消息带来了将被执行的这个操作的详细内容。一般地讲，消息传递的语法随系统不同而不同，其组成部分包括：目标对象，所请求的方法和参数。

2. 类

什么是类？类是创建对象的样板，它包含着所创建对象的状态描述和方法的定义。类的完整描述包含了外部接口和内部算法以及数据结构的形式。

由一个特定的类所创建的对象被称为这个类的实例，因此类是对象的抽象及描述，它是具有共同行为的若干对象的统一描述体。类中要包含生成对象的具体方法。

类是抽象数据类型的实现。一个类的所有对象都有相同的数据结构，并且共享相同的实现操作的代码，而各个对象有着各自不同的状态，即私有的存储。因此，类是所有对象的

共同的行为和不同状态的集合体。

3. 继承

类提供了说明一组对象结构的机制，再借助于继承这一重要机制扩充了类的定义，实现了面向对象计算的优越性。

继承提供了创建新类的一种方法，这种方法就是说，一个新类可以通过对已有类进行修改或扩充来满足新类的要求。新类共享已有类的行为，而自己还具有修改的或额外添加的行为。因此，可以说继承的本质特征是行为共享。

从一个类继承定义的新类，将继承了已有类的所有方法和属性，并且还可以添加所需要的新的方法和属性。新类被称为已有类的子类，而已有类称为父类，又叫基类，新类又叫派生类。

1.2 C++是一种面向对象的程序设计语言

在没有具体讲述C++语言的程序特点和语法规则及编程方法之前，先了解一下C++具有面向对象程序设计语言的哪些特点，这将对学习C++语言是有益的。

1.2.1 C++对面向对象程序设计方法的支持

1. C++支持数据封装

支持数据封装就是支持数据抽象。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。

面向过程的程序设计方法与面向对象的程序设计方法在对待数据和函数关系上是不同的。在面向过程的程序设计中，数据只被看成是一种静态的结构，它只有等待到调用函数来对它进行处理。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。另外，封装还提供一种对数据访问严格控制的机制。因此，数据将被隐藏在封装体中，该封装体通过操作接口与外界交换信息。

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有这个类所规定的若干个私有成员和公有成员以及保护成员。

在C语言中可以定义结构，但这种结构只包含数据，而不包含函数。C++中的类是数据和函数的封装体。在C++中，结构可作为一种特殊的类，它虽然可以包含函数，但是它没有私有或保护的成员。

2. C++类中包含私有、公有和保护成员

C++类中可定义三种不同访问控制权限的成员。一种是私有(private)成员，只有在类中说明的函数才能访问该类的私有成员，而在该类外的函数不可以访问私有成员；另一种是公有(public)成员，类外面也可访问公有成员，成为该类的接口；还有一种是保护(protected)成员，这种成员只有该类的派生类可以访问，其余的在这个类外不能访问。

3. C++中通过发送消息来处理对象

C++中是通过向对象发送消息来处理对象的，每个对象根据所接收到的消息的性质来决定需要采取的行动，以响应这个消息。因此，送到一个对象的所有可能的消息在对

象的类描述中都需要定义，即对每个可能的消息给出一个相应的方法。方法是在类定义中使用函数来定义的，使用一种类似于函数调用的机制把消息发送到一个对象上。

4. C++中允许友元破坏封装性

类中的私有成员一般是不允许该类外面的任何函数访问的，但是友元便可打破这条禁令，它可以访问该类的私有成员(包含数据成员和成员函数)。友元可以是在类外定义的函数，也可以是在类外定义的整个类，前者称友元函数，后者称为友元类。友元打破了类的封装性，它是C++另一个面向对象的重要特征。

5. C++允许函数名和运算符重载

函数名重载和运算符重载都属于多态性，多态性是指相同的语言结构可以代表不同类型的实体或者对不同类型的实体进行操作。C++支持多态性。C++允许一个相同的标识符或运算符代表多个不同实现的函数，这就称标识符或运算符的重载，用户可以根据需要定义标识符重载或运算符重载。

6. C++支持继承性

C++中可以允许单继承和多继承。一个类可以根据需要生成派生类。派生类继承了基类的所有方法，另外派生类自身还可以定义所需要的不包含在父类中的新方法。一个子类的每个对象包含有从父类那里继承来的数据成员以及自己所特有的数据成员。C++由于支持继承性。因此C++将具有继承所带来的好处。

7. C++支持动态联编

C++中可以定义虚函数，通过定义虚函数来支持动态联编。动态联编是多态性的一个重要特征。多态性形成由父类和它们的子类组成的一个树型结构。在这个树中的每一个子类可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，这个对象动态的决定给予子类对象的消息的某种用法。多态性中这一特性允许使用高级抽象。

以上概述了C++对面向对象程序设计中的一些主要特征的支持。有关这些支持的实现将是后面章节中的主要内容。

1.2.2 C++与C语言的关系

C语言是C++的一个子集，C++包含了C语言的全部内容。

1. C++保持与C语言的兼容

这种兼容性表现在许多C代码的程序员不经修改就可以为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++。C语言的程序员学习C++更容易更方便，只要掌握C++语言的新特征就可以了。

但是，这种兼容性使得C++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。于是C++是两种不同风格的程序设计技术溶于一体，使得初学者感到很头痛。另外，不能用面向过程的思路去学习面向对象的技术，因为二者是很不相同的。因此，在学习C++时，要转变以前那种传统的观念来从面向对象的角度学习新技术。

2. C++对C语言作了很多改进