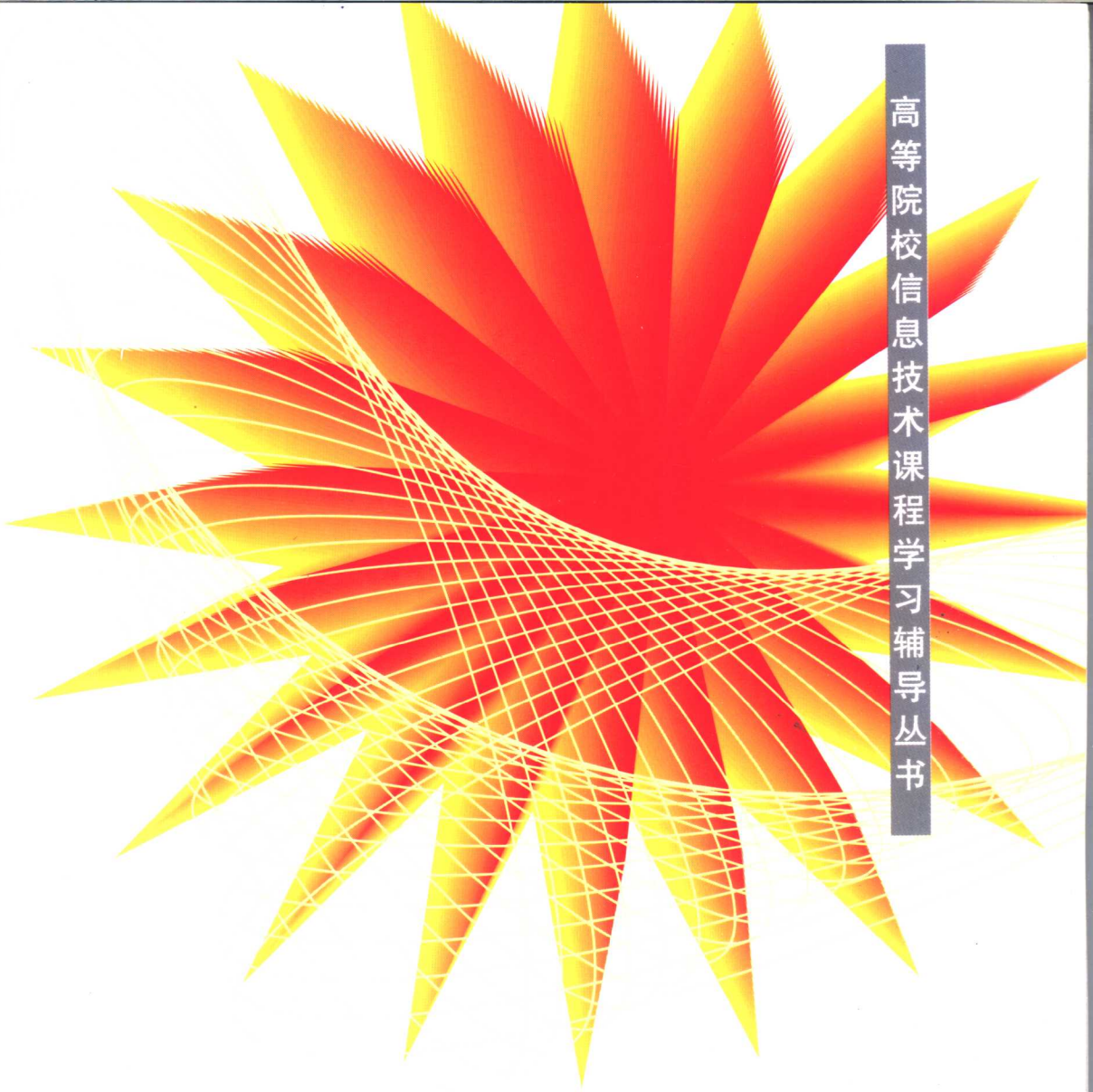


高等院校信息技术课程学习辅导丛书

C++程序设计学习辅导

孙一平 王庆宝 编著

清华大学出版社

高等院校信息技术课程学习辅导丛书

C++程序设计学习辅导

孙一平 王庆宝 编著

清华大学出版社
北京

内容简介

C++语言是计算机科学及其相关专业的重要基础课程。本书通过实例和测试题介绍C++语言的基本内容和特点，并涵盖了C++的面向对象程序设计思想。

本书共10章，叙述了数据类型、控制语句、函数、算法分析、类与对象、继承与派生、重载、虚函数、文件操作等重要概念及应用实例和编程技巧。书中每章辅以大量实例，并且进行解析，说明算法、流程和知识点。最后一章是综合测试，给出多份测试试卷，对较难的题或概念性强的题设有难点提示。每章后面还有自测题，让读者可以进行多种技能的训练。本书所举实例和试题均在VC++ 6.0下调试通过，具有可实践性。

本书采用统一的结构，以章为单位，每章讨论一个专题，均由知识要点、实例解析、测试题组成。所有测试题都附有参考答案。

本书面向计算机及其相关专业本、专科学生，是学习C++语言课程的参考书。对于计算机等级水平考试者，本书也具有一定的参考价值。

图书在版编目(CIP)数据

C++程序设计学习辅导/孙一平，王庆宝编著. —北京：清华大学出版社，2006.3
(高等院校信息技术课程学习辅导丛书)
ISBN 7-302-12050-1

Ⅰ. C…　Ⅱ. ①孙…②王…　Ⅲ. C语言－程序设计－高等学校－教学参考资料　Ⅳ. TP312

中国版本图书馆CIP数据核字(2005)第126051号

出 版 者：清华大学出版社
http://www.tup.com.cn
社 总 机：010-62770175
地　　址：北京清华大学学研大厦
邮　　编：100084
客户服务：010-62776969
组稿编辑：袁勤勇
文稿编辑：霍志国
印 刷 者：北京季蜂印刷有限公司
装 订 者：三河市春园印刷有限公司
发 行 者：新华书店总店北京发行所
开　　本：185×260　**印张**：17.5　**字数**：409千字
版　　次：2006年3月第1版　2006年3月第1次印刷
书　　号：ISBN 7-302-12050-1/TP·7799
印　　数：1～4000
定　　价：23.00元

FOREWORD

前 言

程序设计是大学计算机教育的重要组成部分和培养创新能力的工具。在众多的高级语言中，C++语言在面向对象编程、运行效率、理解计算机的基本概念、用户的使用比例等方面都占有独特的地位，已经成为许多院校计算机专业和非计算机专业第一门算法语言。

C++语言是C语言的一个超集，它是一门混合型的语言，既支持传统的结构化程序设计，又支持面向对象的程序设计。结构化程序在程序代码的空间顺序和程序执行的时间顺序基本一致，程序结构清晰。但是在结构化程序设计中，数据和对数据的操作(即函数)分离，当程序达到一定规模后，为了修改一个小的错误，常可能引出多个大的错误。其原因是在传统的程序设计方式上，数据的表示发生变化，则与之相关的所有函数均要修改，使得原程序难于维护。另外，结构化的程序代码的复用性较差。

面向对象的程序设计方法的思考方式是面向问题的结构，它认为现实世界是由对象组成的，首先确定需要解决的问题是由哪些对象组成，即从问题中抽出合适的对象。下一步是设计封装在对象内部的属性和操作方法，而对于操作方法的设计，核心仍然是算法的设计，这又需要充分利用结构化程序设计的思想。可见，学习面向对象的程序设计离不开结构化程序设计的基本知识。

面向对象程序设计具有对象的封装性、继承性和派生以及实体的多态性等特点，便于解决大型程序设计以及程序的维护，所以得到广泛应用。

VC++语言系统是个很完善，且很复杂的系统。除了一般的程序设计外，还包括API函数应用、MFC程序设计、MDI等多种高级应用的设计。本书的编写目的是辅助C++语言的初学者能够尽快地进入角色，掌握C++程序设计的精髓，学会编程及应用。

本书共10章，叙述了数据类型、控制语句、函数、算法分析、类与对象、继承与派生、重载、虚函数、文件操作等重要概念及应用实例和编程技巧等问题。每章最后有自测题，最后一章是综合测试卷，并给出答案。本书所举实例和试题均在VC++ 6.0下调试通过，具有可实践性。希望读者阅读本书的同时，能把握好实践环节，上机操作验证，举一反三，做到融会贯通，归纳总结，进一步提高程序设计方法和应用软件开发的能力。

本书由孙一平教授、王庆宝副教授等编写。限于时间和编者水平，书中不妥之处难免，敬请批评指正。

编 者

2005年12月

CONTENTS

目 录

CHAPTER 1

第1章 C++概述

C++语言是C语言的一个超集，它是一门混合型的语言，既支持传统的结构化程序设计，又支持面向对象的程序设计，这是C++语言成功流行的一个重要原因。C++作为一门混合型语言，在增加对面向对象方法支持的同时，还继承了结构化程序设计语言C的优点，克服了其不足之处，使得自身既适用于结构化程序设计，又能满足面向对象程序设计的要求，这就符合广大程序员逐步更新其程序设计观念和方法的要求。对于传统的资源不是完全抛弃，而是继承并发展之，是C++语言成功的重要原因。

C语言是美国贝尔实验室在1969—1973年开发的。同时，还用它开发了UNIX操作系统。C语言又是由B语言衍生而来的，B语言是贝尔实验室的Ken Thompson在BCPL语言的基础上开发的，并用它编写了第1个UNIX操作系统。BCPL语言是英国剑桥大学的Martin Richards在20世纪60年代在美国MIT时设计的。

1971年，贝尔实验室的Dennis Ritchie扩展了B语言（通过增加类型），称为NB，即New B。在更改了B语言的结构，并重写了B语言的编译器后，Ritchie称他的新语言为C。1983年，出现了许多C语言的版本，美国国家标准化协会（ANSI）在1989年出版了标准的C语言，称之为ANSI C。

1983年，贝尔实验室的Bjarne Stroustrup在C语言的基础上，创建了C++语言，它是为UNIX系统环境设计的。C++语言增强了C语言的能力，使得程序员能够改进编写程序的质量，并易于程序代码的复用。C++语言的ISO标准已在1997年11月一致通过，1998年8月被正式批准。

1.1 知识要点

1.1.1 程序和程序设计

程序规定了计算机执行的动作和动作的顺序。如同开会的议程、每周的课程安排表。一个程序应包括以下两方面的内容。

（1）对数据的描述：在程序中要指定数据的类型和数据的组织形式，即数据结构。

（2）对操作的描述：即操作步骤，也就是算法。

数据是操作的对象，操作的目的是对数据进行加工处理，以得到期望的结果。作为程序设计人员，必须认真考虑和设计数据结构和操作步骤。著名的计算机科学家Nikiklaus Wirth提出了一个公式：

程序 = 数据结构 + 算法

用计算机语言为计算机编写程序，解决某种问题，称为程序设计。一些程序员，尤其是初学程序设计者，常常认为程序设计就是用某种程序设计语言编写代码，这其实是错误的认识。上述工作应该被看成为编码(coding)，它是在程序设计完成之后才开始的。拿房屋设计的例子来讲，房屋设计这个过程不涉及砌砖垒瓦的具体工作，这些工作是房屋施工阶段进行的。在完成了房屋设计，有了设计图纸之后，施工阶段才能开始。如果不做设计，直接施工，很难想像房屋能不能建造完成，或者建造的房屋合不合要求。同样，程序设计一定要在具体的程序编码之前完成。程序设计完成的好坏直接影响后面的编码质量。与房屋设计中的图纸一样，程序设计也有自己的表达方式。一种常用的程序设计表达方式是程序流程图。流程图也有不同的表示形式，这里只介绍传统的流程图。

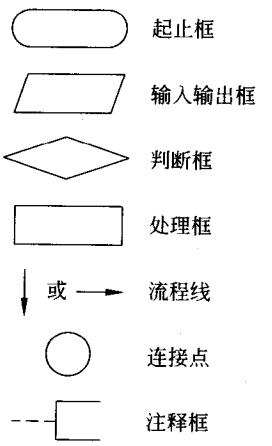


图 1-1 流程图符号

程序流程图是用一些图框表示各种操作，形象直观，易于理解。美国国家标准化协会(American National Standard Institute，ANSI)规定了一些常用的流程图符号，已为世界各国程序工作者普遍采用。如图 1-1 所示。

程序设计需要有一定的方法来指导，例如，一元二次方程 $ax^2+bx+c=0$ 的求解：从数学的角度来求解这个问题，首先需要判断它的 3 个系数 a、b、c 的关系：

① 若 $a=0$，不是二次方程；

② 若 $b^2-4ac=0$，则有两个相等实根 $x=-b/2a$；

③ 若 $b^2-4ac>0$，则有两个不相等实根 $x_{1,2}=\dfrac{-b\pm\sqrt{b^2-4ac}}{2a}$；

④ 若 $b^2-4ac<0$，则有两个共轭复根 $x_1=-\dfrac{b}{2a}+\dfrac{\sqrt{-(b^2-4ac)}}{2a}\mathrm{i}$ 和 $x_2=-\dfrac{b}{2a}-\dfrac{\sqrt{-(b^2-4ac)}}{2a}\mathrm{i}$。

那么在计算机语言中的算法也是如此，首先判断它的 a、b、c 的关系，在不同的分支下进行不同的处理。如果用流程图表示，如图 1-2 所示。

通过上面的例子，可以看出流程图是表示算法的较好工具。一个流程图包括以下几部分：

(1) 表示相应操作的框；

(2) 带箭头的流程线；

(3) 框内外必要的说明文字。

需要注意的是：流程线必须加箭头，因为它是反映程序的执行先后次序的，如果不画箭头就难以判定各框的执行顺序了。

对于求素数、求阶乘、判断闰年等问题的求解也都是与数学的算法有关，而对于字符串的处理还涉及字符串的合并、复制、比较等，不是一个简单算法能够表达的。也就是说，

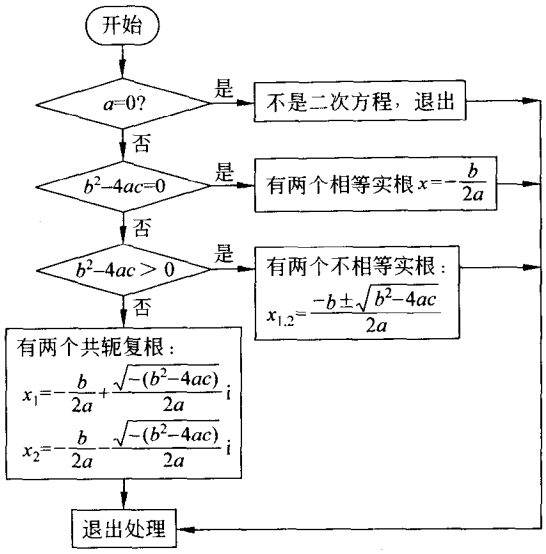


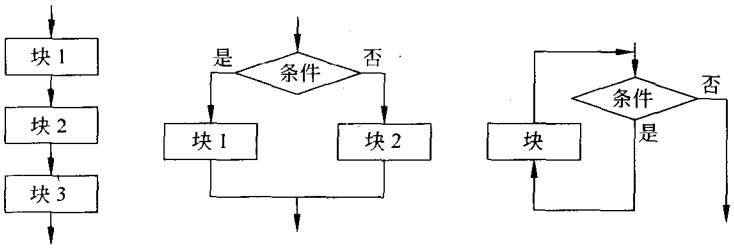
图 1-2　解一元二次方程流程图

这需要对问题进行分解。对问题如何进行抽象和分解，对程序如何进行组织，使得程序的可维护性、可读性、稳定性、效率等更好，这是程序设计方法研究的问题。

目前，有两种重要的程序设计方法：结构化的程序设计和面向对象的程序设计。下面分别进行简单的介绍。

1.1.2　结构化程序设计

结构化程序设计(structured programming，SP)方法是由 E. Dijkstra 等人于 1972 年提出来的，它建立在 Bohm、Jacopini 证明的结构定理的基础上。结构定理指出：任何程序逻辑都可以用顺序、选择和循环 3 种基本结构来表示，如图 1-3 所示。



(a) 顺序结构　　(b) 选择结构　　(c) 循环结构

图 1-3　3 种基本控制结构的流程图

在结构定理的基础上，Dijkstra 主张避免使用 goto 语句(goto 语句表示无条件转移，它会破坏这 3 种结构形式)，而仅仅用上述 3 种基本结构反复嵌套来构造程序。在这一思想指导下，进行程序设计时，可以用所谓“自顶向下逐步求精”的方式，对问题进行分解。对于复杂的问题可以分成如下几步：

(1) 用顺序方式进行分解，确定各个部分的时间顺序；

(2) 用选择方式进行分解，确定某个部分的条件；

(3) 用循环方式进行分解，确定某个部分重复开始和结束的条件。

结构化程序设计方法的两个要点：一是采用“自顶向下逐步求精”的设计思想；二是程序中只采用顺序、循环和选择 3 种基本控制结构。

现在以求 $n!$ 为例说明流程图如何表示程序流程：根据计算阶乘的算法 $n!=1\times2\times3\times\cdots\times(n-1)\times n$，设 P 和 I 的初值均为 1，然后进行连乘，乘积存入 P，再将 I+1 存入 I，重复回去进行连乘。如果需要求 5!，则要判断 I 的值是否到达 5。未达到，则重复前面的步骤，即循环，I 到达 5 后则退出循环，进行输出操作，见图 1-4。该流程中就是采用了顺序、判断和循环 3 种结构。

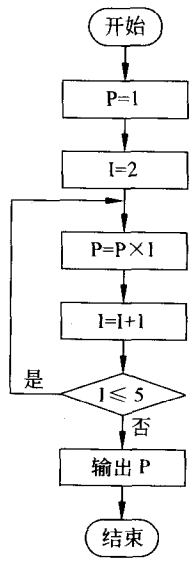


图 1-4　计算阶乘流程

用 SP 方法设计的程序只存在 3 种基本结构，程序代码的空间顺序和程序执行的时间顺序基本一致，程序结构清晰。用 C++ 编写的结构化程序是由许多函数组成的，每个函数只有一个入口和一个出口，没有 goto 语句，这种程序称为结构化程序。

结构化的程序设计仍然是广泛使用的一种程序设计方法，但是它有如下缺点：

(1) 结构化程序设计首先要对功能进行恰当正确的分解。然而对于用户需求来讲，变化最大的部分往往就是功能的改进、添加和删除。结构化程序要实现这种功能变化并不容易，有时甚至要重新设计整个程序的结构。

(2) 在结构化程序设计中，数据和对数据的操作(即函数)分离，函数依赖数据类型的表示。许多重要的函数或过程的实现主要取决于关键的数据结构。如果一个或多个这样的数据结构发生了变化，这种变化将涉及许多方面，许多函数和过程必须重写。有时几个关键的数据结构发生变化，将导致整个软件系统的结构崩溃。随着软件规模和复杂性的增长，这种缺陷日益明显。当程序达到一定规模后，为了修改一个小的错误，常可能引出多个大的错误，究其原因，问题就出在传统的程序设计方式上。数据的表示发生变化，则与之相关的所有函数均要修改，使得程序难于维护。

(3) 结构化的程序代码的复用性较差。例如，调用一个函数或使用一个公共的用户定义的数据类型时，由于数据结构和函数密切相关，使得函数并不具有一般特性，如一个求方程实根的函数不能应用于复数的情形。

当前的软件应用领域已从传统的科学计算和事务处理扩展到了其他的很多方面，如人工智能、计算机辅助设计和辅助制造等，所需处理的数据也已从简单的数字和字符串发展为记录在各种介质上并且有多种格式的多媒体数据，如数字、正文、图形、声音和影像等。数据量和数据类型的空前激增导致了许多程序的规模和复杂性均接近或达到了用结

构化程序设计方法无法管理的程度。

因此尽管结构化程序设计技术具有这样那样的优点，它的这些局限性注定了这种程序设计方法对于规模较小的软件，结构化程序设计是适用的；但当软件规模大到一定程度，这种程序设计方法就显现出稳定性低、可修改性和可复用性差的弊端。

1.1.3 面向对象的程序设计

面向对象的程序设计是另一种重要的程序设计方法，它能够有效地改进结构化程序设计中存在的问题。面向对象的程序与结构化的程序不同，由 C++ 编写的结构化的程序是由一个个函数组成，而由 C++ 编写的面向对象的程序是由一个个对象组成，对象之间通过消息相互作用。

在结构化的程序设计中，要解决某一个问题，就是要确定这个问题能够分解为哪些函数，数据能够分解为哪些基本的类型，如 int、double 等。也就是说，思考方式是面向机器结构的，不是面向问题的结构，需要在问题结构和机器结构之间建立联系。面向对象的程序设计方法的思考方式是面向问题的结构，它认为现实世界是由对象组成的。面向对象的程序设计方法解决某个问题，要确定这个问题是由哪些对象组成的。

1. 对象

客观世界中任何一个事物都可以看作一个对象。或者说，客观世界是由千千万万个对象组成的，它们之间通过一定的渠道相互联系。例如，一所学校是一个对象，一个班级也是一个对象。实际生活中，人们往往在一个对象中进行活动，或者说对象是进行活动的基本单位。例如，在一个班级中，学生进行上课、休息、开会和文娱活动等。作为对象，它应该至少具有两个因素：一是从事活动的主体，如班级中的若干名学生；二是活动的内容，如上课、开会等。

从计算机的观点看，一个对象应该包括两个因素：一是数据，相当于班级中的学生；二是需要进行的操作，相当于学生进行的活动。对象就是一个包含数据以及与这些数据有关的操作的集合。图 1-5 表示一个对象是由数据和操作代码组成的。

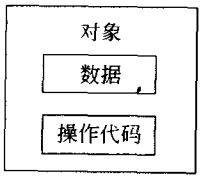


图 1-5 对象的组成

每一个实体都是对象。有一些对象是具有相同结构和特性的。例如，三年级一班、二班、三班是 3 个不同的对象，但是它们属于同一类型，它们具有完全相同的结构和特性。而四年级一班、二班、三班的类型也是相同的，但是它们同三年级班级的类型并不相同。每个对象都属于一个类型。包括 C++ 在内的许多面向对象程序设计语言当中，对象的类型叫作类(class)。类代表了某一批对象的共性和特征。可以说，类是对象的抽象，而对象是类的具体实例。例如，可以先声明“城市”这样一个类，那么北京、上海、南京都是属于这个类的对象。类是用来定义对象的一种抽象数据类型，或者说它是产生对象的模板。用这种数据类型来描述该类所具有的属性和行为，其中属性用数据进行描述，而行为用一系列函

数描述，也称操作。例如，定义一个矩形类，矩形的顶点坐标就是它的数据，而将移动矩形位置、扩大或缩小矩形大小编成几个函数供给调用，这作为行为方法可以操作。这个矩形类就可以用来定义若干个矩形对象，这些矩形对象就是同类对象，都有各自的顶点坐标，都可以通过方法来进行移动、扩大或缩小。

2. 消息

对象之间的产生相互作用所传递的信息称作消息。例如，汽车和人两个对象，人开动汽车，就是向汽车发送消息，汽车接到消息及其参数执行相应的操作。在面向对象设计的程序中，对象之间的相互作用也是通过消息机制实现的。

3. 面向对象的程序设计的特点

面向对象程序设计的基本点在于对象的封装性和继承性，以及由此带来的实体的多态性。

(1) 封装性：对象是一个封装体，在其中封装了该对象所具有的属性和操作，对象作为操作的基本单元，实现了数据和数据处理相结合。C ++ 通过建立类来支持封装和数据隐藏，一个定义完好的类一旦建立就形成了完全的封装体，作为一个整体单元使用。用户不需要知道这个类是如何工作的，只需要知道如何使用就行。另一方面，封装增加了数据的可靠性，保护类中的数据不被类以外的程序随意使用。

(2) 继承和派生：如果已经定义了汽车类，现在还需要定义小汽车。通常不需要重复描述属于汽车的那些共有特征，而只是继承汽车类特性的基础上，描述出属于小汽车的新特征。可以说，小汽车继承了汽车，或者说由汽车派生出了小汽车。面向对象的程序设计提供了这样的机制，当定义一个新类，这个新类与原来类相比较，只是增加或修改了部分属性和操作，这样在定义新类时，只需要说明新类继承原来的类，然后描述新类所特有的属性和操作即可。

(3) 多态性：多态性是指同样一个消息被不同对象接收时产生不同的结果。这种机制主要用于具有继承关系的类体系中。一个类体系中的不同对象可以用不同方式响应同一消息，并产生不同的结果，即实现“同一接口，多种方法”。

与结构化程序设计相比较，面向对象的程序设计具有更多的优点，适合开发大规模的软件工程项目。

1.1.4 C++语言程序设计的知识结构

面向对象的程序设计首先要确定这个问题是由哪些对象组成的。程序设计的核心是从问题中抽出合适的对象，也就是首先解决“做什么”的问题。至于“怎样实现”则是设计封装在对象内部的属性和操作方法，而对于操作方法的设计，核心仍然是算法的设计，这又需要充分利用结构化程序设计的思想。因此要掌握好 C ++ 程序设计的知识结构，如图 1-6 所示。

下面按这个知识结构图来介绍 C ++ 语言程序设计。

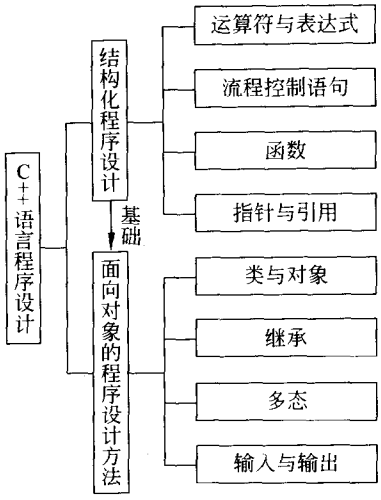


图 1-6　C++ 语言程序设计知识结构

1.2　C++ 基础知识

1.2.1　C++的字符集

C++ 中的字符包括大小写字母、阿拉伯数字及标点符号、汉字等，使用它们的约定如下：大小写英文字母、阿拉伯数字、运算符均采用 ASCII 编码，存储每个字符均占用一个字节单元。

汉字的处理采用汉字国标码，存储汉字采用机内码，它们的存储都占用两个字节。为了便于对各类字符的统一管理，采用了 Unicode 字符集，它共有 65 536 个字符，包括世界上多种语言的基本字符，并对它们进行了双字节编码。ASCII 字符集和国标字符集都是 Unicode 字符集的子集。C++ 中对这些字符的表达如下：

26 个小写字母　　abcdefghijklmnopqrstuvwxyz

26 个大写字母　　ABCDEFGHIJKLMNOPQRSTUVWXYZ

10 个阿拉伯数字　　0123456789

运算符及其他符号 + − * / = ,. _ : ; ? \ " ' ~ | ! # % & ( ) [ ] { } ^ < > [ ]以及空格符。

由这些字符构成 C++ 语言的词汇，并构成关键字、标识符、常量、变量、运算符、标点符号等 C++ 的基本元素。

1.2.2　关键字

关键字(keyword)又称保留字，是 C++ 系统定义的具有特定含义的英文单词，用来说明数据类型、存储类型、访问说明、语句、运算符及逻辑值等，如表 1-1 和表 1-2 所示。关键字不可由用户重新定义。在 C++ 中，区分字母的大小写，关键字全部由小写字母组

成。标准 C ++ 定义了 74 个关键字,但具体的 C ++ 编译器还会对关键字做一些增删。

表 1-1　常用关键字

用于数据类型说明符及修饰符的关键字		用于语句的关键字	
关键字	含义说明	关键字	含义说明
bool	逻辑型	break	退出循环
char	字符型	case	开关语句中的情况分支
const	常量说明	continue	退出本次循环
double	双精度	default	开关语句中的其他情况
enum	枚举型	do	与 while 一起构成循环
float	单精度	else	与 if 构成条件语句
int	整型	for	用于循环语句中
long	长整型	goto	无条件转向
short int	短整型	if	用于条件语句中
signed short int	有符号短整型	return	返回语句
struct	说明结构体	switch	多分支开关
union	共用体	while	用于循环语句
unsigned	无符号		
void	无返回值函数说明		
class	类		
volatile	说明 volatile 变量或函数		
typedef	自定义数据类型		
virtual	说明虚函数		
this	对象的指针		

表 1-2　续常用关键字

存储类型说明符		访问说明符		运算符及逻辑值	
关键字	含义说明	关键字	含义说明	关键字	含义说明
auto	自动	friend	友元成员说明	delete	动态回收内存
extern	外部	private	私有	false	假
inline	内联函数说明	protected	保护	true	真
register	寄存器	public	公共	new	动态分配内存
static	静态			sizeof	求数据类型的长度
				operator	重载运算符

1.2.3 标识符

标识符(identifier,ID)是程序员定义的英文单词,用来对程序中涉及的实体,如变量、函数、标号和其他各种用户自定义对象等命名。在C++中,标识符长度没有限制,但其有效长度为1～31个字符,长度超过31个字符者只识别前31个字符。C++中规定:标识符的第一个字符必须是字母或下划线,其后若有字符则必须为字母、数字或下划线。例如,count2,_x是正确的标识符形式,而hello!,3th则是错误的。在C++中标识符区分大小写,例如,ok、Ok、OK是3个不同的标识符。用户自定义的标识符不能和C++中的关键字相同,也不能和C++编译器提供的资源,如库函数名、类名、对象名等同名,否则那些资源将不能使用。

建议使用有意义的单词或拼音序列作为标识符,可大小写混用,以提高可读性;另外,C++系统本身自定义的内部符号一般以下划线开始,所以自定义标识符时不提倡以下划线开始。

1.2.4 编译和解释

每种计算机的CPU都有自己的指令集合,用这种指令码编写的程序称为机器语言程序,它完全可以控制计算机的行为,是计算机硬件能理解和执行的惟一语言。计算机的CPU种类不同,其指令码组合方式也不相同。同一个题目在不同CPU的计算机上计算时,必须编写不同机器语言的程序。因此机器语言是最低级的语言,是面向机器的,它涉及计算机硬件细节,所以不具有通用性。

CPU指令是用二进制数表示的,用机器语言编程也必须用二进制编程,必然很繁琐,非常消耗精力和时间、难记忆、易弄错,并且难以检查程序和调试程序,工作效率低。学习起来困难,编程效率也低,可读性、可维护性也差。为了方便这种低级语言的编程,产生了汇编语言。汇编语言是一种符号化的语言,将每条机器指令用助记忆的符号(英语缩略的指令集)来表达,编程时采用符号指令及一些伪指令进行,执行时需要将程序通过汇编程序翻译成机器码交付CPU执行。汇编语言仍属于低级语言,还是与机器的具体型号有关。

高级语言是一种接近自然(英语)的语言,编写的程序是由一系列语句(或函数)组成的,其中每一个语句都对应着几条、几十条甚至上百条机器指令的序列,这样的一条语句功能显然增强了,用它开发程序比用低级语言开发效率高得多。同时,由于高级语言的每个句子是一些类似英文的句子或数学表达式,易于理解。编写程序方式更接近人们的思维习惯,这样的程序易读、易懂、易于维护。

高级语言的另一个优点是,用它编写的程序具有一定的通用性。若要使用高级语言编写的程序在某一计算机上运行,只要该计算机提供该语言的翻译系统即可。翻译系统有编译系统和解释系统。编译系统是对源程序进行词法、语法分析,并进行优化处理,生成目标程序,再链接成可执行程序(即CPU能识别的二进制指令形式)送CPU执行。C++是一种高级语言,采用的是编译系统,其编译原理如图1-7和图1-8所示。

编译器将源程序分解为各种语言成分,并检查这些成分的使用是否正确。如果不正