TURING 高等院校计算机教材系列

C++语言程序设计

蒋爱军　刘红梅　王泳　梁小萍　编著

人民邮电出版社
POSTS & TELECOM PRESS

TURING 高等院校计算机教材系列

C++语言程序设计

蒋爱军　刘红梅　王泳　梁小萍　编著

人民邮电出版社
北　京

图书在版编目（CIP）数据

C++语言程序设计 / 蒋爱军等编著. —北京：人民邮电出版社，2008.7
（高等院校计算机教材系列）
ISBN 978-7-115-17638-7

Ⅰ. C… Ⅱ. 蒋… Ⅲ. C语言—程序设计—高等学校—教材 Ⅳ. TN312

中国版本图书馆CIP数据核字（2008）第019142号

内 容 提 要

本书紧密结合C++语言的新标准，以C++语言为工具讲述面向对象程序设计方法。全书分为两部分：第一部分介绍C++语言基础内容及结构化程序设计方法，包括基本类型、表达式、语句、函数、数组、指针等；第二部分介绍面向对象程序设计方法及C++语言中支持面向对象程序设计的主要机制，包括类、继承、多态、模板、命名空间、异常处理、标准库及泛型算法等。书中每章都包含丰富的代码及习题，供读者分析和练习。

本书既可作为计算机专业本科生程序设计课程的入门教材，也可作为相关专业高年级学生面向对象程序设计课程的教材，还可供软件开发人员参考。

高等院校计算机教材系列

C++语言程序设计

◆ 编　　著　蒋爱军　刘红梅　王　泳　梁小萍
责任编辑　杨海玲　刘　静

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街14号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京顺义振华印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：787×1092　1/16
印张：27.25
字数：818千字　　2008年7月第1版
印数：1－4000册　　2008年7月北京第1次印刷

ISBN 978-7-115-17638-7/TP

定价：39.00元

读者服务热线：(010)88593802　印装质量热线：(010)67129223
反盗版热线：(010)67171154

读者意见交流卡

亲爱的读者:

感谢您对我们的支持与爱护。为了今后为您提供更优秀的图书，请您抽出宝贵时间填写本表(或通过我们的网站 www.turingbook.com 填写本表)，将您的意见及时告知我们。您将有机会免费获赠我们出版的图书，并能获得最新的出版信息和更多服务，谢谢!

系列书名：高等院校计算机教材系列
本 书 名：C++语言程序设计

读者资料：
姓　　名：_____ 性　　别：☐男　☐女　年　　龄：_____
职　　业：_____ 文化程度：_____通信地址：_____
电　　话：_____ 传　　真：_____电子信箱（E-mail）：_____

1. **您是如何得知本书的：**
   ☐别人推荐　☐书店　☐出版社图书目录
   ☐杂志、报纸、网络等的介绍（请指明）
   ☐其他（请指明）_____
2. **您从何处购得本书：**
   ☐新华书店　☐电脑专业书店　☐网上书店　☐其他_____
3. **影响您购买本书的因素：**
   ☐内容和质量　☐装帧设计　☐价格
   ☐内容提要、前言或目录　☐书评广告
   ☐出版社名气　☐作者名气　☐其他_____
4. **您对本书封面和封底设计的满意度：**
   ☐很满意　☐比较满意　☐一般　☐较不满意　☐不满意
   ☐建议_____
5. **您认为本书：**
   价格: ☐高　☐合适　☐低
   翻译质量: ☐高　☐一般　☐差
   图书印刷质量: ☐高　☐一般　☐差
6. **您希望本书哪些方面进行改进?**

_____

_____

7. **您感兴趣或希望出版的图书有：**

_____

_____

请寄: 北京市西四环北路 140 号京鼎原商务楼 405 房间　人民邮电出版社图灵公司　市场部收
邮编: 100097　电话: 010-88593802　传真: 010-88593803
电子信箱（E-mail）: contact@turingbook.com　网址: www.turingbook.com

前　言

本书以C++语言为工具，循序渐进地介绍程序设计的基本方法与理念，重点介绍目前的主流程序设计方法——面向对象程序设计（Object-Oriented Programming）。

本书在体系结构上，将程序设计的基本理念和C++语言的基础知识有机地结合在一起；在选材上，充分考虑了读者知识结构和能力结构的形成规律，合理布局知识点，恰当安排内容难度、深度和广度。

全书包括有机联系的14章和6个附录。

第1章介绍程序设计语言和程序设计方法的发展，讨论使用C++语言构造程序的基本方法和步骤，介绍C++程序的基本结构。

第2章介绍C++语言中提供的内置数据类型以及与类型有关的基本概念，同时也简单介绍C++语言程序设计环境中标准输入/输出库的基本概念与用法。

第3章介绍语句与程序控制结构的基本概念以及C++语言中提供的基本语句。其中，重点介绍结构化程序设计采用的3类基本控制结构（特别是选择结构与循环结构）。

第4章介绍C++程序中函数的声明和使用方法，以及有关子程序的基本概念（如参数传递方式、变量作用域与生命期等）。其中主要介绍两点：一是函数的声明与调用，为第5章引入类与对象的概念打下基础；二是递归函数。

第5章介绍枚举类型、结构类型、数据封装与信息隐藏、C++中类的定义、构造函数与析构函数、对象创建等内容。重点介绍类与对象的定义和使用方法，并阐述面向对象程序设计中以类作为程序基本构造单位的思想。

第6章介绍C++中提供的两种数据类型：数组和指针。讨论用这两种数据类型构造复杂数据结构的方法。

第7章介绍字符串的表示与使用。

第8章介绍继承和组合的概念、C++中对继承机制的支持。重点介绍继承机制的用法以及使用继承机制组织类层次。

第9章介绍重载的基本概念，以及本质上一致的两类重载：函数重载和操作符重载。同时也介绍了构造函数重载的一种特殊形式：复制构造函数。

第10章介绍C++中的输入/输出机制，重点介绍C++中的标准输入/输出流以及文件流。

第11章描述多态性的概念、C++语言对多态性的支持，以及多态性的引入对类的一般特性的影响，并讨论使用多态性的注意事项。

第12章介绍C++语言的异常处理机制，以及如何通过异常处理使程序更健壮。

第13章介绍泛型编程的基本概念以及C++语言对泛型编程的支持，总结C++对代码级软件重用的支持。其中，重点介绍了函数模板和类模板。

第14章介绍标准模板库（STL）的3个主要组件：容器、迭代器和算法。重点介绍STL的使用及其对软件重用的支持。STL可说是C++标准库的精华所在，掌握STL的使用可以大大提高C++程序开发的效率和质量。因此本书除了在第14章对STL进行概要介绍之外，还在附录E中给出了关于STL中泛型算法的简介，供读者参考。这也是本书相比于传统C++教材的一大特色。

附录A给出了C++语言的保留字。

附录B为标准ASCII代码表，供读者在使用字符类型时参考。

附录C给出了C++标准库中提供的常用数学函数。

附录D给出了C++标准库头文件的列表，为读者使用相关头文件提供线索。

附录E为标准库泛型算法简介，可作为读者使用相关算法的快速索引。

附录F为本书所涉及主要术语的英汉对照表，可供读者在阅读相关英文书籍时作为参考。

本书以程序设计的思想方法和程序设计语言的知识要点为线索，以C++标准（International Standard ISO/IEC 14882）为依托，既注重内容的完整性，又尽量精简对C++语言的介绍，对实际应用中很少使用的内容尽量不涉及，从而避免让读者过多地陷入语法细节；既注重理论知识的介绍，又强调实际的应用，力求提高读者利用面向对象程序设计方法和C++语言解决实际问题的能力。

学习程序设计一要自己动手多编程序并上机调试，二要多阅读并评价别人编写的程序，因此，本书每章都包含丰富的代码实例（书中给出的程序代码均在Microsoft Visual C++ .NET 2003中编译通过），并且每章都给出了一个具有应用背景的综合性编程实例，通过该实例深化应用该章的主要内容，讲解如何使用C++语言解决具体问题，从而提高读者的编程与动手能力，为进行软件开发及学习其他相关课程打下良好基础。同时，每章均提供具有针对性的典型习题，以帮助读者掌握该章内容。

本书注重培养读者对面向对象程序设计方法和C++语言的实际运用能力，书中给出了大量的“提示”和“注意”，旨在强调重要的知识点，提醒常犯的错误，引导读者深入思考。书中经常对不同程序设计方法进行比较探讨，对C++语言特征上的优缺点进行描述，以期拓宽读者的专业视野。

本书作者从事程序设计课程教学多年，积累了一些经验，出版了多本译著和教材，其中，《C++ Primer（第4版）》（翻译）及《C++ Primer（第4版）习题解答》（编著）在读者中反响较好[①]，《C++程序设计实验教程》被教育部评为2007年度普通高等教育精品教材。本书正是根据作者多年的教学实践经验，在对国内外同类教材进行了深入的比较研究后编写而成的。

本书既可作为计算机专业本科生程序设计课程的入门教材，也可以作为相关专业高年级学生面向对象程序设计课程的教材，还可供软件开发人员参考。

程序设计的世界非常广阔，没有一本教材能够囊括程序设计的所有相关知识，更多时候是要靠学习者的探索和发挥，但是，入门和兴趣是最重要的，这本教材正是这样一本可以将读者引入C++程序设计大门的书。

本书在编写过程中得到了中山大学信息科技学院相关老师的帮助，在此表示衷心的感谢。本书的出版也离不开人民邮电出版社各位编辑的辛勤劳动，在此表示衷心的感谢。

由于作者水平所限，书中不当之处在所难免，恳请读者批评指正。

作 者

2008年3月

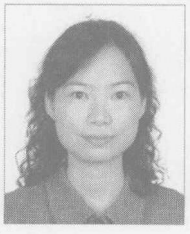
① 这两本书均由人民邮电出版社出版。——编者注

作 者 简 介

蒋爱军 中山大学信息科学与技术学院讲师，在读博士。1998 年于中山大学计算机科学系计算机软件专业获硕士学位。主要研究方向为软件工程、面向对象技术、面向服务计算等。主讲程序设计、数据结构、数据库系统、操作系统、计算机文化等课程。作为主要作者出版了多本译著和教材，其中，《C++ Primer 中文版（第 4 版）》（翻译）及《C++ Primer（第 4 版）习题解答》（编著）在读者中反响较好，《C++ 程序设计实验教程》被教育部评为 2007 年度普通高等教育精品教材。

刘红梅 中山大学信息科学与技术学院副教授。1992 年本科、1996 年硕士毕业于清华大学计算机系，2001 年博士毕业于中山大学电子系。主要研究方向为数字水印与信息隐藏、多媒体信号处理与通信、信息安全技术，曾发表论文多篇并获多项发明专利。主讲程序设计、数据库系统、操作系统等课程。

王泳 中山大学信息科学与技术学院讲师，在读博士。主要研究方向为数字水印与信息隐藏、多媒体信号处理与通信、信息安全技术，特别是数字音频领域的信息隐藏与数字水印。 主讲程序设计、操作系统等课程。



梁小萍 毕业于西安电子科技大学，获得通信工程学学士学位、信息与通信工程学硕士学位，曾于中山大学担任助教、讲师。

目　录

第 1 章　程序设计与 C++语言入门 …… 1
1.1　程序及相关概念 …… 1
1.1.1　计算机与用户（人） …… 1
1.1.2　算法 …… 2
1.1.3　程序 …… 2
1.2　程序设计 …… 3
1.2.1　程序设计的基本概念 …… 3
1.2.2　程序设计过程 …… 3
1.2.3　程序设计方法 …… 4
1.3　程序设计语言 …… 7
1.3.1　机器语言 …… 7
1.3.2　汇编语言 …… 7
1.3.3　高级语言 …… 7
1.3.4　编译型语言与解释型语言 …… 8
1.3.5　C++语言 …… 8
1.4　C++程序的结构 …… 8
1.4.1　C++程序的基本成分 …… 8
1.4.2　以函数为单位的程序结构 …… 9
1.4.3　以类为单位的程序结构 …… 10
1.5　C++程序的实现过程 …… 13
小结 …… 14
习题 …… 14
第 2 章　内置数据类型与基本输入输出 …… 15
2.1　数据类型概述 …… 15
2.1.1　数据类型的基本概念 …… 15
2.1.2　C++语言类型系统的基本特点 …… 15
2.2　标识符 …… 15
2.2.1　C++语言中的基本记号 …… 16
2.2.2　标识符 …… 17
2.3　常量和变量 …… 17
2.3.1　变量和变量的声明 …… 18
2.3.2　常量和常量的声明 …… 19
2.4　内置数据类型 …… 20
2.4.1　内置数据类型概述 …… 20
2.4.2　字符类型常量和变量 …… 21
2.4.3　整数类型常量和变量 …… 22
2.4.4　浮点类型常量和变量 …… 23
2.4.5　布尔类型常量和变量 …… 23
2.4.6　字符串类型常量和变量 …… 24
2.5　操作符和表达式 …… 24
2.5.1　操作符与表达式的基本概念 …… 24
2.5.2　各种操作符和表达式详解 …… 26
2.6　类型之间的关系 …… 30
2.6.1　隐式类型转换 …… 30
2.6.2　显式（强制）类型转换 …… 31
2.7　标准库的使用和简单的输入输出 …… 31
2.7.1　输出 …… 31
2.7.2　输入 …… 32
2.8　应用举例 …… 32
小结 …… 33
习题 …… 33
第 3 章　语句与基本控制结构 …… 35
3.1　C++语句概述 …… 35
3.1.1　带标号语句 …… 35
3.1.2　表达式语句 …… 36
3.1.3　复合语句 …… 36
3.1.4　声明语句 …… 37
3.1.5　try块 …… 38
3.1.6　转移语句 …… 38
3.2　程序的基本控制结构 …… 39
3.3　选择语句 …… 40
3.3.1　if语句 …… 40
3.3.2　switch语句 …… 43
3.4　循环语句 …… 44
3.4.1　while语句 …… 45
3.4.2　do-while语句 …… 46
3.4.3　for语句 …… 46
3.4.4　break语句及其在循环语句中的使用 …… 48
3.4.5　continue语句及其在循环语句中的使用 …… 49
3.5　应用举例 …… 50
小结 …… 53
习题 …… 54
第 4 章　函数 …… 57

4.1 概述……57
4.2 函数定义与函数原型……59
4.2.1 函数定义……59
4.2.2 函数原型……60
4.3 函数调用与参数传递……61
4.3.1 函数调用……61
4.3.2 参数传递……63
4.4 标识符的作用域……68
4.4.1 作用域的基本概念……68
4.4.2 作用域的具体规则……69
4.4.3 变量的声明与定义……70
4.4.4 名字空间……72
4.5 变量的生命期……73
4.6 预处理指令……76
4.6.1 文件包含……76
4.6.2 宏定义……76
4.6.3 条件编译……76
4.7 标准库函数……77
4.8 函数的接口设计和注释……77
4.8.1 前置条件和后置条件……77
4.8.2 函数的注释……78
4.8.3 函数的接口与实现……78
4.8.4 函数接口的设计……79
4.9 递归……80
4.9.1 什么是递归……80
4.9.2 递归的实现……81
4.9.3 汉诺塔问题……82
4.10 应用举例……83
小结……85
习题……85
第5章 枚举、结构与类……87
5.1 简单数据类型与构造式数据类型……87
5.2 枚举类型……87
5.3 结构类型……89
5.3.1 结构类型的定义和结构类型变量的声明和使用……89
5.3.2 结构变量的整体操作……91
5.3.3 层次结构……93
5.3.4 匿名结构类型……94
5.4 抽象、封装与信息隐藏……94
5.4.1 抽象……94
5.4.2 数据封装与隐藏……95
5.5 类与对象……98
5.5.1 类……98
5.5.2 对象的创建……103
5.5.3 对象的初始化……104
5.6 关于面向对象程序设计的若干基本问题……108
5.6.1 面向过程与面向对象……108
5.6.2 术语……112
5.7 应用举例……112
小结……115
习题……115
第6章 数组与指针……117
6.1 数组类型……117
6.1.1 一维数组……117
6.1.2 二维数组……124
6.2 指针类型……132
6.2.1 基本概念……132
6.2.2 指针常量与指针变量……133
6.2.3 指针的运用……136
6.3 指针类型与数组……141
6.3.1 通过指针引用数组元素……141
6.3.2 数组作函数参数的进一步讨论……145
6.3.3 动态分配内存……147
6.3.4 二维数组与指针……151
6.4 main函数的形参……153
6.5 指向结构变量的指针……155
6.6 对象指针……156
6.6.1 基本概念……156
6.6.2 对象的动态创建和撤销……157
6.6.3 对象的复制……158
6.7 函数指针……159
6.8 应用举例……160
小结……166
习题……166
第7章 字符串……169
7.1 C风格字符串……169
7.1.1 字符串字面值……169
7.1.2 字符数组……169
7.2 标准库字符串操作函数……171
7.2.1 C风格字符串的连接操作……171
7.2.2 C风格字符串的比较……172
7.2.3 C风格字符串的复制……173
7.2.4 求C风格字符串的长度……173
7.2.5 C风格字符串转换成其他类型的数据……174
7.2.6 cctype库中处理字符的函数……174
7.3 标准库类string……175

7.3.1 string对象的声明、初始化与赋值 …… 176
7.3.2 string对象的长度 …… 176
7.3.3 string对象的连接操作 …… 176
7.3.4 string对象的比较 …… 177
7.3.5 string对象的子串 …… 177
7.3.6 string对象转换成C风格字符串 …… 178
7.3.7 string对象的输入和输出 …… 178
7.4 应用举例 …… 178
小结 …… 182
习题 …… 183
第8章 继承与组合 …… 186
8.1 继承的概念 …… 186
8.2 C++中的继承 …… 187
8.2.1 基本概念 …… 187
8.2.2 继承实例 …… 190
8.2.3 派生类中继承成员函数的重定义 …… 195
8.2.4 继承层次中的构造函数和析构函数 …… 195
8.3 组合 …… 199
8.3.1 组合的语法和图形表示 …… 199
8.3.2 组合与构造函数和析构函数 …… 200
8.3.3 组合的实例 …… 202
8.4 继承与组合的比较 …… 206
8.5 多重继承与重复继承 …… 206
8.5.1 多重继承 …… 206
8.5.2 多重继承的构造函数 …… 209
8.5.3 多重继承中存在的问题：名字冲突 …… 210
8.5.4 重复继承 …… 211
8.6 应用举例 …… 214
小结 …… 227
习题 …… 228
第9章 重载 …… 232
9.1 函数重载 …… 232
9.1.1 什么是函数重载 …… 232
9.1.2 为什么要使用函数重载 …… 237
9.1.3 使用函数重载时需要注意的问题 …… 237
9.2 复制构造函数 …… 241
9.2.1 复制构造函数的语法形式 …… 241
9.2.2 复制构造函数的使用场合 …… 242
9.3 操作符重载 …… 254
9.3.1 C++操作符的函数特性 …… 254
9.3.2 操作符重载的规则 …… 254
9.3.3 类成员操作符重载 …… 254
9.3.4 友元操作符重载 …… 260
9.4 应用举例 …… 263
小结 …… 270
习题 …… 270
第10章 I/O流与文件 …… 273
10.1 概述 …… 273
10.1.1 何为I/O …… 273
10.1.2 应用程序、操作系统与I/O …… 273
10.1.3 标准I/O流cin和cout …… 274
10.1.4 文件I/O流 …… 275
10.2 二进制文件I/O …… 278
10.2.1 文本文件I/O与二进制文件I/O …… 278
10.2.1 二进制文件I/O …… 279
10.3 应用举例 …… 282
小结 …… 285
习题 …… 285
第11章 多态性与虚函数 …… 286
11.1 绑定方式与多态性 …… 286
11.1.1 基本概念 …… 286
11.1.2 多态性的作用 …… 286
11.2 虚函数 …… 287
11.2.1 虚函数举例 …… 288
11.2.2 使用虚函数的特定版本 …… 290
11.2.3 虚析构函数 …… 291
11.3 纯虚函数和抽象类 …… 292
11.3.1 纯虚函数 …… 292
11.3.2 抽象类 …… 293
11.4 应用举例 …… 294
小结 …… 304
习题 …… 305
第12章 异常处理 …… 306
12.1 异常处理概述 …… 306
12.2 C++语言中的异常处理 …… 306
12.2.1 throw语句 …… 307
12.2.2 try块与异常的捕获及处理 …… 308
12.2.3 标准库异常类 …… 321
12.2.4 异常说明 …… 322
12.3 应用举例 …… 323
小结 …… 336

习题 ……336
第 13 章 模板……338
13.1 泛型编程概述……338
13.2 函数模板……338
13.2.1 函数模板的定义……339
13.2.2 函数模板的实例化……339
13.2.3 函数模板与重载……341
13.3 类模板……344
13.3.1 类模板的定义……344
13.3.2 类模板的实例化……348
13.3.3 模板编译与类模板的实现……349
13.4 非类型模板形参……352
13.4.1 函数模板的非类型形参……352
13.4.2 类模板的非类型形参……352
13.5 应用举例……353
小结……367
习题……367
第 14 章 标准模板库……370
14.1 概述……370
14.2 迭代器……370
14.3 容器……372
14.3.1 顺序容器……372
14.3.2 关联容器……383
14.3.3 容器适配器……390
14.4 泛型算法……393
14.4.1 算法简介……393
14.4.2 算法举例……396
14.5 应用举例……397
小结……407
习题……408
附录 A C++保留字表……409
附录 B 标准 ASCII 代码表……410
附录 C 常用数学函数……411
附录 D C++标准库头文件……412
附录 E 标准库泛型算法简介……413
附录 F 主要术语英汉对照表……420
参考文献……424

第1章 程序设计与C++语言入门

C++语言是目前应用广泛的一门程序设计语言。本章将介绍与程序设计相关的基本概念和C++程序的基本结构，讨论使用C++语言构造程序的基本方法和步骤。

1.1 程序及相关概念

1.1.1 计算机与用户（人）

电子计算机，简称**计算机**，是一种电子设备，也有人称之为“智力工具”，是一种能够接受输入数据，存储和处理数据，并产生输出数据的设备。

遵循冯·诺依曼[①]体系结构的现代计算机由以下5个部件构成。

- **运算器**。又称**算术逻辑单元**，简称ALU（arithmetic and logic unit），主要完成各种算术运算和逻辑运算。
- **控制器**。对各部件加以控制，使得各部件能够协调工作。
- **存储器**[②]。存储数据和程序。
- **输入设备**。接受数据和程序。
- **输出设备**。将处理结果呈现给用户。

各部分的关系如图1-1所示。

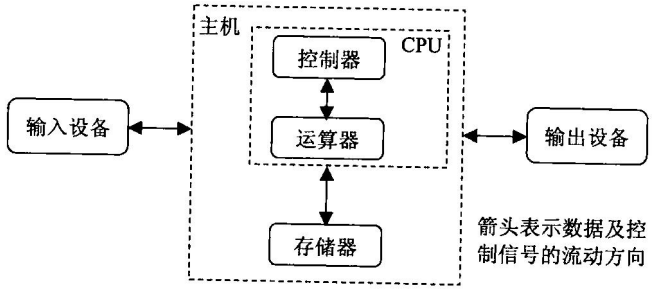


图1-1 计算机的基本组成部件

其中，运算器和控制器统称为CPU（central processing unit，中央处理器），而CPU和存储器通常又可以统称为主机。

冯·诺依曼体系结构的主要思想：

- 计算机由5大基本部件构成（参见图1-1）；
- 计算机内部采用二进制数表示指令和数据；

① 冯·诺依曼（Von Neumann，1903—1957），美籍匈牙利数学家，公认的现代计算机之父。

② 严格地说，这里的存储器指的是内存。一般而言，存储器包括内存（主存储器）和外存（辅助存储器）。

- 将程序（由一系列指令组成）和数据存放在计算机内部（内存），并让计算机自动执行。

计算机要能够工作，必须具备硬件和软件两方面的条件。硬件就是计算机本身的构成部分，也就是上面提及的5大部件；软件就是计算机系统中的程序及相关支持文件。

计算机的使用者称为“用户”，也就是人。计算机是电子设备，是机器，无法与人进行直接交流。例如，在没有软件的支持下，如果我们需要用计算机计算“5+8”，直接对它说：“嗨，告诉我5+8等于多少！”是没有用的，必须有一个**程序**（例如，计算器程序），人通过这个程序来操纵计算机硬件，进行相关计算并获得计算结果。计算机是人们用来解决问题的通用工具，而程序则是解决某一特定问题的专用工具。因此，计算机用户实际上就是程序的用户，通过各种各样的程序使计算机完成工作。

1.1.2 算法

算法（algorithm）一词源于算术（algorism）[①]，原意是一个由已知推求未知的计算过程。后来，人们把它加以推广，将解决问题的过程（方法和步骤）的描述统称为算法。

例如，判断任意给定自然数n是否为素数的算法如下。

步骤1：令$d=2$

步骤2：令$r=$（n除以d的余数）

步骤3：如果r等于0且d不等于n，则断定n不是素数并终止；
　　否则，转步骤4

步骤4：令d值加1并转步骤5

步骤5：如果d大于n的平方根，则断定n是素数并终止；
　　否则，转步骤2

其中，d表示divisor，即除数；r表示remainder，即余数。

根据上述步骤，对任意给定的自然数，我们都可以断定它是不是一个素数。

1. 算法的特征

任意一个算法都应该具有如下基本特征。

- 以数据为主要操作对象。上述算法中的数据包括n、d、r等。
- 确定性 。算法中的每一个步骤必须有确切的定义，不能有二义性。
- 有穷性 。一个算法必须能够在执行有限步骤之后终止。
- 0个或多个输入。输入是指执行算法时从外界取得的数据。例如，上述算法的输入为n。具有0个输入的算法在执行时不需要从外界获取数据。
- 一个或多个输出。算法的输出就是算法所求问题的“解”，用以反映对输入数据进行处理之后的结果。上述算法的输出为对n是否为素数的判定。没有输出的算法是无意义的。
- 有效性。算法中的每一个步骤都必须能够有效地执行且得到确定的结果。对上述算法而言，若d为0，则步骤2不能有效执行（当然，事实上，该算法中的d不可能为0）。

2. 算法的表示

算法有各种各样的表示方法，上文中给出的素数判定算法是用自然语言描述的，但同一算法也可以用其他方式描述，常用的描述方法还有：流程图、N-S图、PAD图、伪代码和UML活动图等。这里所说的算法表示，指的是人与人之间为交流算法思想而采用的表示方式，实质上，程序是对算法最精确的表示，通常称为算法的实现。

1.1.3 程序

如果要判断给定自然数是否为素数，可以用纸笔作为工具执行1.1.2节中的算法得到结果，也可以

① 最早源自波斯数学家Al-Khwarizmi的名字。——编者注

用计算机作为工具完成上述计算。二者在本质上没有什么区别，但执行的效率是有区别的。一般而言，对于较为复杂的计算问题，计算机的执行效率远比人要高。这也是人们经常选用计算机来完成某些工作的原因。

如果以计算机作为工具解决某个问题（例如，判断某自然数是否为素数），必须将解决问题的步骤（即算法）告诉计算机，因为人无法与计算机直接交流，所以必须使用程序将算法表示成计算机能够理解的形式（这一过程通常称为算法的实现），然后让计算机执行程序来完成指定的任务。

程序（program）是由计算机执行的指令序列，用来表示程序员要求计算机执行的操作。也就是说，程序是算法在计算机系统中的表示[①]。

1.2 程序设计

1.2.1 程序设计的基本概念

程序设计（programming）是指编制程序的活动，也就是用计算机能够理解的形式表达算法的过程。一般而言，进行程序设计必须具备以下4个方面的知识。

- 应用领域的知识。这是构造问题解决方案的基础。例如，要解决素数判断问题，必须了解素数的概念（即除了1和它本身之外没有其他约数的自然数）以及素数判断的相关知识（即，不能被2至$\sqrt{n}$之间的任意整数所整除的自然数n就是素数），这些就是应用领域的知识。如果程序设计者不具备应用领域的知识，当然不可能开发出解决应用问题的程序。
- 程序设计方法。在具有应用领域知识的基础上，还必须掌握某种程序设计方法，才能运用适当的思维方式，构造出问题的解决方案。
- 程序设计语言。要使用计算机解决问题，必须将解决方案转换为能被计算机理解和执行的程序，因此程序员需要掌握程序设计语言（如C++语言）这一工具。
- 程序设计环境与工具。程序设计环境与工具可以提供许多可重用的基本程序，让程序员在设计程序时使用，因此，开发程序（尤其是大型程序）时，通常需要利用程序设计环境和工具，以便提高程序开发的效率以及程序的质量。例如，当用C++语言开发程序时，可以选用微软公司的Visual C++，也可以选用开源软件GCC。

1.2.2 程序设计过程

程序设计过程可以分为4个阶段：分析阶段、设计阶段、实现阶段和测试阶段。

分析阶段的主要任务是理解问题，弄清楚要解决的问题是什么，也就是所开发的程序要做什么。例如，要开发一个素数判断程序，分析阶段的任务就是明确该程序需要做什么，即可能要“判断用户从键盘输入的一个自然数是否为素数”；也有另一种可能，即“判断某个文件中存放的所有自然数是否为素数”。这是两个有区别的需求，与此相对应的解决方案也有所不同。分析阶段就是要明确程序到底要“做什么”。

分析阶段对问题加以明确定义之后，就进入设计阶段。设计阶段的目标是针对问题的要求开发出相应的解决方案，也就是要开发出解决问题的逻辑步骤序列，即算法。一般而言，得出问题的解决方案之后，还需要对方案进行验证和确认，确保该方案的确能解决对应的问题。

确立了问题的解决方案之后，就需要用某种程序设计语言对方案进行严格地描述，这一过程称为实现。实现阶段的主要任务就是将算法转换为程序。

获得程序之后，需要对程序进行测试[②]，通过测试的程序才能发布给用户使用。

---

① 后文会谈到，程序也是实体在计算机系统中的表示。实体用于对算法进行有效组织。

② 严格说来，对于大型程序的开发，在每个阶段都需要进行相关测试，以保证程序的质量。

任何程序都是人的智力产品，而任何人都有可能犯错误，因此，很难保证一个实用程序是完全正确的。在程序发布之后，用户在使用过程中也有可能会发现程序中的错误，这时就需要对程序进行修改。此外，随着时间的推移，用户对程序的功能可能会产生新的要求，为了满足用户的新要求，往往也需要对程序进行修改。这一类修改工作通常称为对程序的“维护”，因此，程序在发布之后就进入了维护阶段。

程序设计初学者常犯的一个错误是：一拿到问题就开始编写代码，也就是直接进入实现阶段，忽略了程序设计过程中的分析和设计阶段，这会严重影响程序设计的质量，甚至导致开发项目的失败，尤其对大型程序的开发更是如此。分析、设计阶段要与实现阶段分开，在开始学习某门具体程序设计语言之前，我们就可以针对一些简单问题设计算法，这也可以作为学习程序设计的入门方法。

1.2.3 程序设计方法

程序设计方法是指组织程序内部数据和逻辑的方法。自从1946年诞生了世界上第一台通用电子计算机ENIAC开始，程序设计方法及程序设计语言就在不断地发展。随着计算机及通信、网络等相关技术的不断发展，计算机的应用越来越普及，程序的规模也越来越大，程序设计的目标不再是片面的高效率，而变为对程序的可理解性、可扩展性、可靠性和可重用性等因素的综合考虑，从而使程序设计方法和程序设计语言得到了极大的发展。

1. 早期程序设计

在计算机诞生之初，受硬件技术的限制，计算机的内存空间极为有限，运算速度也较慢，因此，在开发程序时，程序员更多地注重程序的执行效率，而程序的可理解性和可扩展性等质量因素往往只能作为次要因素考虑，甚至为了追求效率而被牺牲掉。在这个时期，基本上没有成型的程序设计方法，程序员主要依赖个人技巧和天分进行编程，编程成为一种“艺术”，编出来的程序往往在可理解性、可维护性和通用性等方面都比较差。

2. 结构化程序设计

随着计算机硬件技术及相关信息技术的发展，计算机的应用范围越来越广，要开发的程序也越来越大、越来越复杂，单纯依靠个人的编程技巧已难以编制出满足应用要求的程序。适应这一需要，20世纪60年代出现了**结构化程序设计**（Structured Programming，SP）方法，又称为**过程式程序设计**（procedural programming）方法。

SP方法的主要思想是：自顶向下、逐步求精、模块化编程，以及采用单入口/单出口的控制结构构造程序。所谓“自顶向下”是一种问题分解技术，指的是将复杂问题分解为一系列复杂性相对较低的子问题，逐个解决这些子问题，整个问题就得到了解决；“逐步求精”指的是对问题进行连续分解，直至最后的子问题小到易于解决，最终可用3种基本控制结构（顺序结构、选择结构、循环结构）来表示；“模块化编程”是指将较大的程序划分成若干子程序，每个子程序称为一个模块，较复杂的模块可以继续划分为更小的子模块，从而使得程序具有层次结构。

例如，要编一个程序解决如下问题：判断用户从键盘输入的一个自然数是否为素数。采用SP方法，设计过程如下。

首先将该问题分解为3个子问题，对应算法的如下3个步骤。

步骤1：输入自然数n

步骤2：判断n是否为素数

步骤3：输出判断结果并终止

其中步骤1和步骤3比较简单，基本上可直接解决；而步骤2则需要进一步细化，例如，可细化为如下子步骤。

步骤2.1：令$d=2$

步骤2.2：令r=（n除以d的余数）

步骤2.3：如果r等于0，则将结果置为“是”并转步骤3；否则，转步骤2.4

步骤2.4：令d值加1并转步骤2.5

步骤2.5：如果d大于n的平方根，则将结果置为“否”并转步骤3；否则，转步骤2.2

至此，就解决问题的算法而言，已经比较详细了，只要掌握了某门程序设计语言就可以编制出相应的程序了。

结构化程序设计方法将程序看作对数据的一系列处理过程，将数据和对数据的处理过程分开，以过程为中心，从程序的功能出发进行分解，其结果是一个程序由若干过程组成，每个过程完成一个确定的功能。C++语言中用**函数**（function）来表示过程。

3. 面向对象程序设计

结构化程序设计方法的思想符合人们处理问题的一般习惯，为处理复杂问题提供了有力的手段，而且，结构化程序相比于非结构化程序更容易理解和修改，因此结构化程序设计方法得到了广泛应用。

但是，结构化程序设计将数据和对数据的处理分开，程序中的某一数据可能会被许多过程处理，如果该数据发生变化将影响到所有相关过程（例如，如果对某数据的结构进行了修改，则所有使用到该数据的过程通常也必须修改），因此，当程序规模较大、数据量较大时，数据和处理的这种分离状态将使得程序难以被人理解，且难以维护。针对这一情况，出现了**面向对象程序设计**（Object-Oriented Programming，OOP）方法，并于20世纪80年代开始逐渐成为主流程序设计方法。

OOP的基本概念是**对象**（object）和**类**（class）。所谓“对象”，指的是客观存在的单个事物，又称为**实体**（entity）[①]，例如，一个人、一辆汽车、一架飞机和一个银行账户等都是对象。对象一般都具有属性和行为。属性用来描述对象的特征和状态，例如，汽车具有发动机、传动系统和燃料系统等属性，银行账户具有账号、户名和存款余额等属性。行为用于改变对象的状态，例如，汽车具有启动、加速和刹车等行为，这些行为可以改变汽车的状态，比如，启动行为可以让汽车的发动机从非工作状态转变为工作状态；银行账户具有存款和取款等行为，这些行为可以改变账户的存款余额。对象就是属性和行为的统一体。

对象可以归类，例如，张三是一个人，李四也是一个人，则这两个人是同类对象，可以归入“人”这个“类”。类描述了同类对象的共性。例如，“汽车”是一个描述某类交通工具的类，而你的汽车和我的汽车都是汽车的实际例子，因此都具有发动机等属性，也都具有启动和刹车等行为。因此，类是描述同类对象共同具有的属性和行为的模型，对象则可以看作根据这个模型制造出来的实际事物，对象又称为类的**实例**（instance）。

在程序当中，对象的属性通常用数据来表示，而对象的行为通常用操作来表示。面向对象程序设计以结构化程序设计为基础，最大的改变是将数据和对数据的操作（即数据处理）当作一个整体——对象来对待，从而使得在改变数据的结构时，所涉及程序的修改仅限于有限的范围（即，对象的范围，具体而言，是该对象所属的类的范围）。

OOP方法以“对象”为中心进行程序设计，程序就是实体在计算机系统中的表示。采用OOP方法设计的程序中包含一系列对象，由这些对象的相互协作完成程序的功能。对象是类的实例，因此，面向对象程序的基本构造单位是类。OOP方法包括封装（及信息隐藏）、继承和多态性等基本特征。

(1) **封装**（encapsulation）

这是面向对象方法的重要原则，有两个重要作用：一是把对象的属性和行为结合在一起成为不可分割的独立单位；二是使得对象内部的实现细节可以尽可能地隐藏起来不被外界所见，外界只能通过

① 对象可以是现实世界的事物，如一辆汽车，也可以是思维世界的事物，如栈这种数据结构。

对象所提供的对外接口与对象发生联系。

封装的例子在现实生活中非常常见。例如，当我们购买一台冰箱时，获得的就是一个冰箱的封装体，生产厂家将冰箱内部的压缩机和控制电路等都隐藏起来，只提供一些控制按钮（接口）给我们，我们通过这些按钮来使用这台冰箱。

下面是封装的优点。

- **实现信息隐藏**。例如，冰箱的内部细节不会暴露给外界。
- **更容易使用**。用户只需了解怎样访问接口就能直接使用，无需考虑实现细节。例如，使用冰箱的人只要知道哪个按钮控制哪项功能，根本无需了解冰箱的内部工作原理，就能顺利使用。
- **更容易变更内部实现**。可以在不考虑用户的情况下变更（只需保持接口不变），因为内部的实现细节对用户的使用没有直接影响。例如，为了更省电可以改变冰箱的内部设计，但只要保持按钮的设置不变，就不会对用户的使用方式产生任何影响。

C++语言用类来支持封装，类封装了由同类对象所共享的属性和行为。C++中提供访问控制方式`private`和`protected`来隐藏内部信息，并提供`public`访问控制方式来定义公共接口。

(2) **继承**（inheritance）

客观事物中普遍存在着一种关系：一般和特殊的关系，也就是所谓的“是一种（is a）”关系。例如，梨是一种水果，莱阳梨是一种梨；汽车是一种交通工具，吉普车是一种汽车。针对这种一般和特殊的关系，如果有了关于一般概念的定义，在定义新的特殊概念时就不必一切从头做起。例如，有了水果的概念之后，在定义梨的概念时，就不必将梨所具有的水果的一般特征（如多汁、可以生食等）一一列举，只需简单地说明：梨是一种水果，然后再对梨不同于其他水果的特征加以描述即可。在这里，定义梨的概念时，使用了已经存在的水果这一概念，这就是**重用**（reuse）。重用了水果这一概念之后，定义梨这一概念的工作变得简单了，因为只需要描述梨这种水果不同于其他种类水果的特点；同时，对于已经知道了水果这一概念的人而言，理解梨这一新概念也变得简单了，因为只需要了解梨与其他种类水果的不同之处。

在现实生活中重用的例子极为常见。例如，以前的电视机功能比较单一，只要能收看电视节目就可以了，随着计算机的普及，许多家庭有了将电视机与计算机相连的需求，以便利用电视机的大屏幕播放视频文件。为了满足这种需求，生产电视机的厂家就推出了新的产品，这种新产品往往就是在原有产品的基础上增加新的部件，使得它可以与计算机连接。这里，生产厂家在重用旧产品设计的基础上，通过增加新的功能部件来设计新产品，这样就不必一切从头做起，可以大大加速新产品的推出速度。

面向对象程序设计中继承机制的主要作用就是支持软件重用。利用继承机制，程序员可以通过对现有的类进行扩充（增加新的成员）而定义新的类，用这种方式定义的新类称为**派生类**（derived class），被继承的类称为**基类**（base class）。派生类自动具有基类中定义的所有成员，在此基础上可以定义一些新的成员来满足新的需要。这样一来，定义基类成员的代码得到了重用，可大大简化定义派生类的工作量，提高程序开发效率。

C++语言直接支持继承机制。

(3) **多态性**（polymorphism）

在面向对象程序设计中，多态是指“同一名字，多种含义”，或者“同一接口，多种实现”，通常指函数和方法（即对象的行为）具有相同的名字，但有不同的行为特征。

日常生活中我们使用许多东西时，往往只关注它的功能和使用方法，至于产品的实现方法并不会太关注。例如，当使用全自动洗衣机时，我们只关心当按下开始按钮时，洗衣机是否能按我们事先设定的模式把衣服洗好，至于洗衣机内部采用哪种电机、哪种控制电路，我们并不关心。可能有采用不同电机的洗衣机，但它们的使用方法是相同的。同样的使用方法（接口），不同的内部实现，这就是多态性。