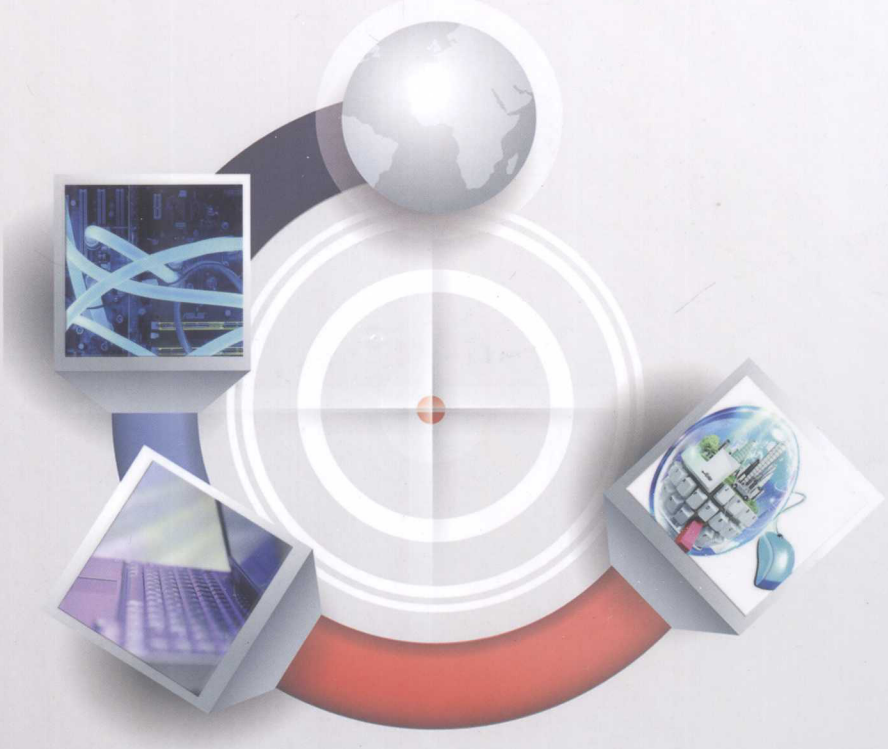
普通高等教育“十三五”规划教材

C语言程序设计（第二版）

C YUYAN CHENGXU SHEJI

◎孙　辉　吴润秀　编著



中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

普通高等教育“十三五”规划教材

C 语言程序设计

（第二版）

编 著 孙 辉 吴润秀

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

内 容 简 介

结合 Visual C++ 6.0 集成开发环境，本书详细介绍了 C 程序设计的基本概念、程序设计方法、集成开发环境中的调试工具和常用的调试技巧。同时，增加了 Windows 窗口程序（图形程序）设计的内容，并在附录中介绍了 Visual C++ 6.0 编译器的使用。书中全部程序均可在 lcc 3.3 编译器或者 Microsoft Visual Studio 2010 编译下通过，前 10 章中的程序，除个别程序外，也能在 gcc 3.0 及以上版本的 C 编译器下编译通过。

全书的写作参照了建构主义的教学理论，简化了语法说明，突出了编程实践，同时注重调试技术，强调动手能力的培养。本书配有《C 语言程序设计实验指导与习题集》一书。

本书适合作为大学本科计算机相关专业“C 语言程序设计”课程教材或教学参考书。如果去除 Windows 程序设计一章及部分较难的程序，也可供高职相关专业作为教材使用。

图书在版编目（CIP）数据

C 语言程序设计 / 孙辉，吴润秀编著. -- 2 版. -- 北京 : 中国铁道出版社，2016.2 (2016.7重印)
普通高等教育“十三五”规划教材
ISBN 978-7-113-21308-4

Ⅰ. ①C… Ⅱ. ①孙… ②吴… Ⅲ. ①C 语言－程序设计－高等学校－教材 Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字（2016）第 025930 号

书　　名：C 语言程序设计（第二版）
作　　者：孙　辉　吴润秀　编著

策　　划：刘丽丽　曹莉群
责任编辑：周　欣　曹莉群
封面设计：刘　莎
责任校对：汤淑梅
责任印制：郭向伟

出版发行：中国铁道出版社（100054，北京市西城区右安门西街 8 号）
网　　址：http://www.51eds.com
印　　刷：北京明恒达印务有限公司
版　　次：2007 年 2 月第 1 版　2016 年 2 月第 2 版　2016 年 7 月第 2 次印刷
开　　本：787mm×1092mm　1/16　**印张：**20　**字数：**471 千
书　　号：ISBN 978-7-113-21308-4
定　　价：42.00 元

第二版前言

FOREWORD

本次改版，基本上保留了原有教材的特点，对已过时的内容进行了更新，补充了新的案例，尤其是 C 语言调试方法的案例。附录 A 中新增的调试案例，如果教师能仔细阅读，并帮助学生掌握相关内容，则不但对于学生掌握集成开发环境中的工具，而且对于掌握调试技巧乃至于更好地学习 C 语言，均有帮助。为了保证教材内容的简洁与完整，与全国计算机等级考试相关的内容，放在配套的《C 语言程序设计实验指导与习题集》一书中。由于 Microsoft Visual Studio 2010 一类新的集成开发环境过于庞大，使用复杂，不适用初学者，故第二版的 C 程序，仍基于 Visual C++ 6.0 集成开发工具中的 C 编译器。

全书由孙辉、吴润秀编著。吴润秀撰写了第 5 章、第 8 章、第 9 章，孙辉撰写了其余章节，并在出版前对全书进行了统一的修改及最后的统稿。

由于编者水平所限，书中不足之处在所难免，恳请读者批评指正。作者的电子信箱：Sun_hui2006@yahoo.com.cn。

编　者

2015 年 12 月

第一版前言

FOREWORD

计算机高级语言程序设计是计算机专业和大多数非计算机专业学生必须掌握的内容之一。该课程的主要目的，一是使学生具有初步的程序设计能力，以便为后续相关课程打下基础；二是加深学生对计算机的理解，提高计算机的应用能力。在目前常用的几种高级语言中，C 语言由于其良好的结构化特性、简练的语法、强大的功能被大多数高校选作高级语言课程的教学语言。

学过 C 语言的人，一般都感到 C 语言难学。在编写本书时，针对 C 语言在学习过程中的问题，我们在内容的编排、例题的选择、编写的方式等方面作了一些新的尝试。

1．简化语法说明，突出编程实践

学习程序语言，并不仅是学习语言的语法，而是要通过语言的学习，掌握必要的程序设计技术，提高计算机应用能力，因此，本书十分重视初学者能力的训练。

根据建构主义的教学理论，学生是学习的主体，他们通过与客观环境的交互来建构自己的知识结构。因此，在教材中，对语法不作冗长的讲解，而是通过大量短小的演示程序，让读者通过程序阅读及上机实践来掌握相应的语法。事实上，过多的语法讲解反而会使问题复杂化。本书的演示程序较小，不含复杂的算法，读者可以方便地将其进行改造，以便理解相应的语法知识。教师可通过引导，让学生自己作为学习的主体，在程序的阅读和上机实践中学会分析语法。

除演示程序外，全书安排了 70 多道实例程序。实例程序注重算法的分析和编程的实践。此外，本书还特别注意各实例之间的联系，很多实例以两种方式贯穿全书。一是通过用新学过的知识重新编写原来的程序，二是通过将原来较小的程序进行扩充、组合，使之成为较大、较复杂的程序。对学过的程序用新的结构重新编写，可以有效地复习原有知识，又可加深对新知识的理解。将原有的程序进行扩充，一方面能使读者学会编写较大的程序，另一方面也是一种温故知新的方式。

建构主义的教学理论认为，人们学习知识的过程，是一个不断在已有知识背景下对新知识进行建构的过程。因此，用新学过的知识对已有问题重新编程，或者将已有的程序进行组合、扩充，其实是在新的知识背景下对原有知识重新建构的过程。

2．注重调试技术，强调动手能力

C 语言难学的一个重要原因在于细节难于掌握。许多 C 语言的初学者，面对因细节造成的程序错误，往往束手无策。在实际教学中发现，如果能较好地掌握程序的调试技术，就较容易排除这类错误，增强学习的信心。

程序的调试技术同样是初学者难于掌握的，也是 C 语言学习过程中的一个难点问题。程序的调试，与其说是技术，不如说是经验，更是一种动手能力。为此，本书除了在附录 A 中介绍了 C 语言程序的调试技巧外，更将程序的调试技术贯穿全书。根据教学实践，读者如果能仔细阅读相关内容，并对其中介绍的调试技术在计算机上进行实践，就不难掌握 Visual C++ 6.0 环境下的程序调试技巧。

3. 引入窗口程序，展示最新趋势

标准 C 语言只能进行字符程序（或称控制台程序）设计，但是，目前程序设计的主流是窗口程序。为此，本书中增加了窗口程序（也就是图形程序）设计的简单介绍。通过窗口程序的学习，不但可以了解窗口程序的消息驱动机制，还能了解 Windows API 的使用。这些知识对读者更好地理解 Windows 操作系统的工作原理也会有所帮助。

本书介绍的 C 语言开发环境是 Windows 下高效的 Visual C++ 6.0，尽管其后续版本已经推出，但它们与 Visual C++ 6.0 相差不大。

事物总是一分为二的，在本书编写过程中的一些新的尝试，也会带来相应的问题，如会增加教材的篇幅，语法的系统性不够，对教师提出了更高的要求等。还有更重要的一点，由于编者的水平所限，有些想法不一定能达到所期望的目标。

从教材编写的角度来说，我们有两点考虑。第一点考虑是本书是作为教材而不是使用手册来编写的，因此，全书的内容是按照初学者的学习过程来安排的；第二点考虑是计算机已经非常普及，学习者在学习的过程中能随时上机实践。学习计算机有一句至理名言“向计算机学习计算机”。这告诉我们，学习计算机不能崇尚教条或书本，主要是靠多实践。在课堂上是学不会计算机的。初学者在学习的过程中，应注重实践，多上机调试程序。

教材中程序的源代码绝大多数是编者为了教学需要而编写的，而且在教学中已多次使用。全书的所有程序源代码都已经过严格测试，以确保代码的正确性。读者在使用本书的程序代码时，如果没有输入错误（要特别注意符号的输入，如果将英文字母或标点符号输入成中文符号，则肯定会出错），代码都能编译通过并正确运行。

本书是针对 C 语言的初学者编写的。为了适应某些中级读者，也是为了初学者能尽快提高水平，自第 6 章开始，内容的难度有了较大幅度的增加。有些程序较长，而且有一定的难度。但只要读者能将程序输入计算机并调试通过，然后通过跟踪程序的运行，这些程序也是不难理解的。

为了方便教师的使用和学习者的学习，教材配有用两种工具（PowerPoint 和 Flash）开发的电子演示教案。

全书由孙辉、吴润秀编著。吴润秀撰写了第 5 章、第 8 章、第 9 章，孙辉撰写了其余章节，并在出版前对全书进行了统一的修改及最后的统稿。

由于编者水平所限，书中不足之处在所难免，恳请读者批评指正。作者的电子信箱：Sun_hui2006@yahoo.com.cn。

编　者

2006 年 12 月

目 录

CONTENTS

第 1 章 绪论 1

1.1 C 语言简介 1
1.2 程序设计的基本概念 1
1.2.1 程序 2
1.2.2 程序设计 2
1.2.3 算法 2
1.2.4 数据结构 2
1.3 常用计算机高级语言简介 2
1.4 C 语言程序的开发过程 4
1.5 对于 C 语言学习的认识 6
1.6 本教材对 C 语言的处理 6

第 2 章 基本数据类型及顺序结构程序设计 8

2.1 几个简单的 C 语言程序 8
2.2 C 语言的字符集、关键字和标识符 10
2.2.1 字符集 10
2.2.2 关键字 10
2.2.3 标识符 10
2.3 C 语言的数据类型 11
2.3.1 数据及数据类型的概念 11
2.3.2 基本类型 12
2.3.3 构造类型 12
2.4 常量 12
2.4.1 整型常量 13
2.4.2 实型常量 13
2.4.3 字符常量 13
2.4.4 转义字符 13
2.4.5 字符串常量 14
2.4.6 符号常量 14
2.5 变量 15
2.5.1 变量的概念 15
2.5.2 变量的说明 16
2.5.3 变量的类型 17
2.5.4 变量的初始化 17

2.6 运算符和表达式 18
2.6.1 算术运算符和算术表达式 19
2.6.2 赋值运算符和赋值表达式 20
2.6.3 自增（++）、自减（--）运算符 21
2.6.4 逻辑量的概念 22
2.6.5 关系运算符和关系表达式 22
2.6.6 逻辑运算符和逻辑表达式 23
2.6.7 条件运算符 24
2.6.8 位运算 25
2.6.9 逗号运算符和逗号表达式 29
2.6.10 sizeof运算符 29
2.7 运算符的优先级 29
2.8 混合运算中的类型转换问题 30
2.8.1 自动类型转换 30
2.8.2 强制类型转换 31
2.9 数据输出和输入 32
2.9.1 数据输出 32
2.9.2 数据输入 36
2.10 顺序结构程序设计举例 41
小结 43
习题 44

第 3 章 分支结构 47

3.1 if语句 47
3.1.1 if...else语句 47
3.1.2 if语句的两种变形 49
3.1.3 if语句的嵌套 53
3.2 if语句应用举例 56
3.3 switch语句 58
3.4 无条件转移语句 goto 63
小结 64
习题 65

第 4 章 循环结构 67

4.1 while循环结构 67
4.1.1 while循环的结构 67
4.1.2 while循环的使用 68
4.2 do...while 循环结构 72

4.2.1 do…while 循环的结构 72
4.2.2 do…while 循环的使用 72
4.3 for 循环结构 74
4.3.1 for 循环的结构 74
4.3.2 for 循环的使用 75
4.4 循环结构的嵌套 78
4.5 循环中 break 和 continue 语句的使用 80
4.6 循环语句的使用举例 81
小结 84
习题 85

第 5 章 函数 87

5.1 函数的定义与调用 87
5.1.1 函数定义的一般形式 89
5.1.2 函数过程的调用 90
5.1.3 函数的返回值 92
5.1.4 函数的声明 92
5.2 变量的存储类别 94
5.2.1 自动变量 95
5.2.2 外部变量 95
5.2.3 有多个源程序文件的程序中外部变量的引用 96
5.2.4 静态变量 99
5.2.5 register 变量 101
5.3 变量的作用域 101
5.3.1 局部变量 101
5.3.2 全局变量 102
5.4 内部函数和外部函数 104
5.4.1 内部函数 104
5.4.2 外部函数 104
5.5 函数的递归调用 105
5.5.1 递归算法的概念 105
5.5.2 C 函数的递归调用 105
小结 110
习题 111

第 6 章 数组 113

6.1 一维数组 114
6.1.1 一维数组的定义 114

6.1.2 一维数组的引用 114
6.2 一维字符数组与字符串 118
6.3 字符串常用库函数 122
6.4 二维数组及多维数组 126
6.4.1 二维数组的定义 126
6.4.2 二维数组的初始化 127
6.4.3 三维数组及讨论 128
6.4.4 二维数组应用举例 128
6.5 数组综合应用举例 134
小结 138
习题 138

第 7 章 指针 141

7.1 指针的概念 141
7.1.1 指针变量的说明 143
7.1.2 指针运算符 144
7.1.3 指针的赋值 145
7.1.4 指针的算术运算 146
7.1.5 指针运算符与单目运算符的优先级 147
7.2 指针应用程序举例 148
7.3 动态内存分配 149
7.3.1 动态内存分配函数 149
7.3.2 动态内存分配程序设计 151
7.4 参数指针的使用 158
7.5 多级指针 158
7.6 指针与数组 159
7.6.1 利用指针访问数组元素 160
7.6.2 数组指针 161
7.6.3 指针数组 165
7.7 函数指针 166
7.7.1 函数指针的定义 166
7.7.2 函数指针的引用 166
7.8 命令行参数的使用 168
小结 170
习题 171

第 8 章 结构 174

8.1 结构的定义和变量说明 174

8.1.1　结构的定义 174
8.1.2　结构类型变量的定义 175
8.2　结构变量的初始化和引用 177
8.2.1　结构变量的初始化 177
8.2.2　结构变量的引用 178
8.3　结构数组 179
8.4　结构指针 181
8.5　结构变量作为函数的参数 182
8.6　链表的概念及简单应用 186
8.6.1　链表的概念 186
8.6.2　链表中结点的数据定义方式 187
8.6.3　链表的简单应用 187
8.6.4　链表内结点的删除 189
8.6.5　链表内结点的插入 193
8.7　联合的概念及简单应用 195
8.8　枚举类型 197
8.9　用 typedef 定义类型 198
8.10　位域的概念及简单应用 200
小结 203
习题 203

第 9 章　文件 206

9.1　文件的概念 206
9.1.1　C 语言文件概述 206
9.1.2　标准级（流式）输入输出 206
9.1.3　文件指针 206
9.2　文件的打开与关闭 207
9.2.1　文件的打开 207
9.2.2　文件的关闭 207
9.3　常用文件读写函数 209
9.3.1　字节级 209
9.3.2　字符串级 211
9.3.3　格式化读写函数 212
9.3.4　块读写函数 215
9.4　文件操作错误检测 218
9.5　文件定位与随机读写 219
小结 221
习题 221

第 10 章 预处理命令 223

10.1 宏定义 223
10.1.1 无参数的宏定义 223
10.1.2 带参数的宏定义 226
10.2 文件包含 #include 228
10.3 条件编译 230
10.3.1 第 1 种条件编译 230
10.3.2 第 2 种条件编译 231
10.3.3 第 3 种条件编译 231
小结 232
习题 232

第 11 章 Windows 窗口程序设计 233

11.1 Windows 窗口程序设计的概念 233
11.1.1 Windows 用户界面介绍 233
11.1.2 初识 Windows 窗口程序 234
11.1.3 Windows 窗口程序最基本的结构 235
11.1.4 Windows 程序中的消息机制 238
11.1.5 Windows 程序基本结构的详细说明 239
11.2 Windows 窗口程序中的输出 241
11.3 Windows 窗口程序中的常用数据类型 244
11.4 Windows 窗口程序中的资源文件 245
11.4.1 菜单资源的使用 246
11.4.2 对话框资源的使用 250
11.5 Windows 窗口程序中的输入 256
11.6 Windows 图形程序设计 263
小结 266
习题 267

附录 A Visual C++6.0 的使用 268

A.1 Visual C++6.0 概述 268
A.2 Visual C++ 6.0 的工作环境 269
A.2.1 Visual C++ 6.0 开发环境总览 269
A.2.2 File 菜单 271
A.2.3 Edit 菜单 272
A.2.4 View 菜单 273
A.2.5 Insert 菜单 273
A.2.6 Project 菜单 274

A.2.7　Build 菜单 274
A.2.8　Tools 菜单 275
A.2.9　Windows 菜单 276
A.2.10　Help 菜单 276
A.3　Visual C++ 6.0 编译器简介 276
A.3.1　编译参数的设置 277
A.3.2　连接参数的设置 280
A.3.3　其他编译参数的设置 281
A.3.4　注意事项 281
A.4　调试器的使用 281
A.4.1　常用调试工具 281
A.4.2　设置断点 283
A.4.3　单步执行 283
A.5　集成开发环境中程序调试实例 284
A.5.1　使用调试（Debug）模式和发布（Release）模式 284
A.5.2　多文件程序的编译 285
A.5.3　程序调试实例 287
A.6　Edit and Continue 简介 293
A.6.1　Edit and Continue 的使用 293
A.6.2　Set Next Statement 命令的使用 295
A.6.3　在调试中增加函数或变量 296
附录 B　常用 C 库函数 297
B.1　数学函数 297
B.2　字符函数和字符串函数 298
B.3　输入输出函数 299
B.4　动态存储分配函数 301
附录 C　常用字符与 ASCII 代码对照表 302
参考文献 303

第1章 绪 论

C 语言是目前比较流行的高级语言。它的规模虽然较小，但功能强大。与其他很多高级语言不同，在 C 语言的发展过程中，开始时并无一个明确的目标——设计一个全新的高级语言。因此，C 语言的历史，有许多故事可说。

1.1 C 语言简介

1970 年，美国电话电报公司（AT&T）贝尔实验室的 Ken Thompson 以 BCPL（Basic Combind Programming Language）语言为基础，设计出简单且接近硬件的 B 语言（取 BCPL 的第一个字母），并用 B 语言编写了第一个 UNIX 操作系统，在 PDP-7 上实现。1971 年在 PDP-11/20 上实现了 B 语言，并写成了 UNIX 操作系统。由于 B 语言过于简单，所以其功能有限。1972 年，美国电话电报公司（AT&T）贝尔实验室的 Dennis Ritchie 在 B 语言的基础上，设计出 C 语言（取 BCPL 的第二个字母）。C 语言既保持了 BCPL 和 B 语言的优点，又克服了它们的缺点。1973 年，Ken Thompson 和 Dennis Ritchie 合作，将 UNIX 的 90%以上的代码用 C 改写，即 UNIX 第 5 版。

后来，C 语言做了多次的改进。直到 1975 年 UNIX 第 6 版公布后，C 语言的突出优点才引起人们的注意。1977 年出现了不依赖于具体机器的 C 语言编译文本《可移植的 C 语言编译程序》，使 C 语言移植到其他的机器时所做的工作大大减少。随着 UNIX 的广泛使用，C 语言也得到迅速的推广。

1978 年，B.W.Kernighan 和 D.M.Ritchit 以发表的第 7 版 UNIX 中的 C 编译程序为基础，合著了著名的 *The C Programming Language* 一书。通常简称为 *K&R*，也有人称之为 *K&R* 标准。其实，在 *K&R* 中并没有定义一个完整的标准 C 语言。1983 年，美国国家标准化协会（ANSI）根据 C 语言问世以来各种版本对 C 的发展和扩充，制定了新的标准，称为 ANSI C。

早期的 C 语言主要用于 UNIX 系统。由于 C 语言的强大功能和各方面的优点逐渐为人们认识，到了 20 世纪 80 年代，C 语言开始进入其他操作系统，并很快在各类大、中、小和微型计算机上得到了广泛的使用，并成为当代优秀的程序设计语言之一。

目前，C 语言新的标准 C99 已经推出，其实，它更多的是将目前各种已存在的对 C 语言的扩充部分标准化。在新的 C 标准中，没有引入面向对象的特性。

1.2 程序设计的基本概念

在学习程序设计语言时，有必要了解几个基本概念。

1.2.1 程序

所谓程序，简单地说，就是控制计算机完成特定功能的一组有序指令的集合。通常，一个计算机程序主要描述两部分的内容，问题的每个对象及它们之间的关系；对这些对象进行处理的规则。对象及它们之间的关系涉及数据结构，处理规则亦即是求解问题的算法。因此，程序被描述为：数据结构+算法。

1.2.2 程序设计

所谓程序设计，就是根据计算机要完成的任务，提出相应的需求，在此基础上设计数据结构和算法，然后编写相应的程序代码，测试并修改代码使其能得到正确的运行结果。程序设计的方法很重要，一个好的设计方法能够大大提高程序的高效性、合理性。程序设计有一套完整的算法，也称程序设计方法学。为此，有人提出：

程序设计=数据结构+算法+程序设计方法学

程序设计方法学在程序设计中占有重要地位，尤其对于大型软件更是如此，它是软件工程的组成部分。

1.2.3 算法

所谓算法，就是问题的求解方法。通常，一个算法由一系列求解步骤来完成。正确的算法要求组成算法的规则和步骤的意义是唯一确定的，不能存在二义性，而且这些规则指定的操作是有序的，必须按算法指定的操作顺序执行，并能够在有限的执行步骤后给出正确的结果。

算法的建立过程通常是逐步求精。一般情况下，先给出粗略的计算框架，然后对框架中的具体内容进行逐步细化，添加必要的细节，使之成为较为详细的描述。当然，细化有时不能一步到位，因此还要进行更进一步的细化，直到能够把需求通过编程语言完全描述为止。

描述算法的常用工具是流程图，也称程序框图。流程图是算法的图形描述，它往往比程序更直观，易于阅读和理解。当然，它只是一种表现工具，计算机不能直接识别和运行流程图。

1.2.4 数据结构

数据结构是指数据对象及其相互关系和构造方法。程序中的数据结构描述了程序中数据间的组织形式和结构关系。

数据结构与算法密不可分。一个良好的数据结构，会使算法简单化；只有明确了问题的算法，才能较好地设计数据结构，因此，两者是相辅相成的。

对于计算机程序，其构成与数据结构关系密切。程序在实现算法的同时，还必须完整地体现作为算法操作对象的数据结构。对于复杂问题的求解，由于对数据的表示方式和结构的差异，对问题的抽象求解算法也会完全不同。对同一个问题的求解会有不同的算法，当然也会有不同的数据结构。

1.3 常用计算机高级语言简介

编写计算机程序所使用的语言称为程序设计语言。高级语言是相对低级语言而言的。低

级语言是计算机能直接识别的语言（即机器语言）或符号化的机器语言（即汇编语言）。高级语言是一种通用的、面向用户的各类需要、与特定的机器相分离、并遵循一定的严格规定与形式的语言，其语言格式接近于自然语言，或接近于数学函数形式。如 Quick Basic、Fortran、Cobol、Pascal、C、Lisp、Prolog、Ada、Java、PL/1 等均为高级语言。

在高级语言中，面向解题过程，告诉计算机“怎么做”，这种语言称为过程式语言。面向处理对象，告诉计算机“做什么”而不必指出“怎么做”，计算机就能完成所要求的任务，这种语言称为非过程式语言。

Quick Basic、Fortran、Pascal、C 等为过程式语言。SQL 查询语言、Smalltalk 等为非过程式语言。

在高级语言教学中，常用的教学语言有 Pascal、Fortran、C、C++、Java、Quick Basic、Visual Basic 等。

Pascal 是优秀的结构化程序设计语言，用它来描述数据结构很方便。以前，很多计算机专业都选用 Pascal 作为教学语言。但现在，由于用它开发的实际应用不多（主要集中在 Borland 公司的 Delphi），且计算机专业的后继课程（如“数据结构”等），也普遍采用 C 语言描述算法，所以，用它来作为教学语言越来越少。

Quick Basic 是一种结构化的程序设计语言，兼容旧版本的 Basic，又有着良好的现代化的开发环境，学生很容易学会使用它。所以，有很多的非计算机专业选择它作为高级语言程序设计课程的教学语言，也有一些计算机专业选它作为 C 语言的先行课程。但是，Quick Basic 是 DOS 下的程序设计语言，随着全国计算机等级考试中停考 Quick Basic，它在教学中逐步减少。

Visual Basic 是 Windows 下的窗口程序设计语言，用它来开发 Windows 下的应用程序很方便，学习它也不是很困难，所以很多的非计算机专业用它作为高级语言课程的教学语言。不过，Visual Basic 主要是提供窗口程序设计的一些控件、与数据库的接口等，对于其中使用的程序设计语言，还是要另外学习，否则，很难用 Visual Basic 解决实际问题。

Fortran 语言是结构化的程序设计语言，Fortran 语言主要用于大规模科学计算上，对于一般的应用，Fortran 语言没有优势。微型计算机上经常使用的 DOS 下的 MS Fortran 语言只支持 Fortran 77 的子集，且其开发环境陈旧，使用不便，不适于教学。在 Windows 95 及 Windows NT 下，微软公司的 Fortran 开发环境 Fortran PowerStation 及 HP-COMPACK 公司的 Visual Fortran（其实是微软公司将它的 Fortran PowerStation 4.0 卖给了原来的 DEC 公司，DEC 公司利用它在小型机上的 Fortran 语言的编译器优势，结合微软 Fortran 的优秀界面而推出的）是极为优秀的 Fortran 语言开发环境。该编译环境功能强大，不但支持 Fortran 90 或 Fortran 95 的全集，而且还有许多扩充。但这类开发环境过于庞大，初学者很难掌握。其他一些 Fortran 编译器（如 NDP Fortran 等）尽管功能强大，但开发环境使用不便，而且在国内也不多见。因此，用 Fortran 语言作为教学语言，对工科一些需要使用大规模数值计算软件，如有限元等的专业比较适用，但对大多数非计算机专业的学生并不适用，对计算机专业也不适用，因为它不适合用来进行系统程序设计，用它来描述算法也不如 C 语言方便。

正如前面介绍的，C 语言是比较流行的实用程序设计语言，各种不同的开发环境都很优秀。C 语言的细节难于掌握，程序的调试也比较困难，初学者学习困难较多。但是，C 语言也有很多的特点，对计算机专业来说，它不但能培养学生的计算机程序设计能力，还是一些重要的后续课程的先行课。如计算机专业的重要专业基础课程“数据结构”，越来越多的教材

采用 C 语言来描述其算法。在计算机专业的重要专业课程“操作系统”中，由于涉及系统编程，故课程的实验一般都是用 C 语言完成（有些教材用 Java，其实只能是用来描述进程的算法，对其他的内容如内存的分配、信号量的处理等，基本上只能用 C，尤其是如果操作系统选用 UNIX 或 Linux 作为蓝本更是如此）。还有其他很多的课程要用到 C 语言，比如学习 Linux 操作系统，希望进行一些工作，如内核编译，或者想阅读 Linux 的系统源代码等，都需要用到 C 语言的知识，因此，对计算机专业来说，学好 C 语言是必须的。

对于非计算机专业的学习者，如果想对计算机有更多的理解，学习 C 语言也有很多的好处。它使学习者能更深入理解计算机工作原理，也能帮助学习者在学习面向对象编程时对对象有更多的理解。

C++是在 C 语言的基础上，增加了面向对象的成分。C++完全兼容 C 语言（C99 新标准中有一些新增加的内容如动态数组、复数运算等与 C++不兼容），用 C 语言编写的程序可以在 C++编译器中编译通过。但 C++编译器比 C 编译器的语法检查严格，所以，有些程序在 C 编译器下能够编译通过，或只给出警告，但在 C++编译器下不能编译通过。一般情况下，严格按 C 语法编写的程序在 C++编译器中都能编译通过。

C++除了本身一些内容如多重继承等难学之外，使其难学难用的另一个原因是 C++编程通常都要用到编译器提供的 C++类库，这些类库数量庞大，使用复杂。如果不用系统提供的类库而完全用 C++编程，那是不可想象的。

为了与 C 兼容，C++被迫作出了很多重大的设计妥协，结果导致语言过分华丽，过分复杂。尤其是 C++没有采用自动内存管理策略，从而丧失了修正 C 最严重问题的机会。

C++的应用主要集中在 GUI、游戏和多媒体工具包方面，在其他地方很少用到。事实上，面向对象也仅在这些领域被证明非常成功。

Java 语言是后起之秀，但大有后来居上之势。它的最大特点是其跨平台特性，所谓一次编译，到处运行。Java 语言的历史很短，1995 年才正式问世。Java 衍生于 C++，出于安全性及稳定性，去掉了 C++中不易理解和掌握的部分如指针等。Java 的基本语法和 C 语言几乎一模一样。与 C++不同，Java 是一个纯面向对象的高级语言。Java 语言的主要应用在网络编程上。由于受到运行速度的限制（Java 程序的运行速度与 C 相比慢很多），在一些高强度的计算方面，Java 语言并无优势。

总之，不同的高级语言各有优势，适应不同的应用，但没有一种语言能包打天下。

1.4 C 语言程序的开发过程

计算机不能直接识别高级语言，要让计算机能执行高级语言，需要将高级语言翻译成等价的机器语言，这种翻译有两种形式：

① 边翻译边执行。

② 全部翻译完成后再执行。

前一种称为解释方式，后一种称为编译方式。Fortran、Pascal、C 等为编译型语言，BASIC、Java 等为解释型语言。

用高级语言写成的程序称为源程序（或源代码），翻译成的机器语言称为目标程序（目标代码），将目标程序连接后生成的程序称可执行程序，即能在计算机上运行的程序。

C 语言为编译型语言，其开发过程如图 1-1 所示。