HZ BOOKS
华章教育

“十二五”普通高等教育本科国家级规划教材

高等学校计算机专业规划教材

C语言程序设计与实践

第2版

凌云 谢满德 陈志贤 吴海燕 编著

The C Language Programming and Practice Second Edition

机械工业出版社
China Machine Press

“十二五”普通高等教育本科国家级规划教材

高等学校计算机专业规划教材

C语言程序设计与实践

第2版

凌云 谢满德 陈志贤 吴海燕 编著

The C Language Programming and Practice Second Edition

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

C 语言程序设计与实践 / 凌云等编著 . —2 版 . —北京：机械工业出版社，2017.1
（高等学校计算机专业规划教材）

ISBN 978-7-111-55849-1

I. C… II. 凌… III. C 语言－程序设计－高等学校－教学参考资料 IV. TP312.8

中国版本图书馆 CIP 数据核字（2017）第 013378 号

本书从结构上分成两大部分：第一部分为 C 语言的基础语法介绍，包括第 1 ～ 11 章；第二部分为项目实训和常用算法指导，包括第 12 章和第 13 章，以项目实训的形式引导和帮助学生解决实际问题，并对程序设计竞赛中的常见算法及其算法应用进行了介绍。

本书结合案例介绍 C 语言的语法知识，内容全面，注重实训，可作为计算机类专业本科或专科教材，也可作为信息类或其他相关专业的选修教材或辅助读物。

出版发行：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：吴晋瑜　　责任校对：董纪丽
印　　刷：北京市荣盛彩色印刷有限公司　　版　　次：2017 年 2 月第 2 版第 1 次印刷
开　　本：185mm × 260mm　1/16　　印　　张：18.5
书　　号：ISBN 978-7-111-55849-1　　定　　价：39.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991　88361066　　投稿热线：（010）88379604
购书热线：（010）68326294　88379649　68995259　　读者信箱：hzjsj@hzbook.com

前　言

C 语言程序设计是一门理论与工程实践密切相关的专业基础课程，在计算机学科教学中具有十分重要的地位。大力加强该课程的建设，提高该课程的教学质量，有利于教学改革和教育创新，有利于创新人才的培养。通过本课程的学习，学生应培养良好的编程风格，掌握常见的算法思路，真正提高运用 C 语言编写程序解决实际问题的综合能力，为后续课程的实践环节打好基础。

目前国内关于 C 语言的教材较多，有些教材语法知识介绍细致，较适合作为非专业的等级考试类教学用书；有些教材起点较高，内容深奥，不适于初学者。为了帮助广大学生更好地掌握 C 语言编程技术，我们组织 C 语言程序设计课程组的教师进行了深入的讨论和研究，并针对学生学科竞赛和课时压缩的背景，将该课程的建设与其他信息类专业的课程体系改革相结合，发挥我们在计算机和电子商务、信息管理等专业上的办学优势，编写了《C 语言程序设计与实践》一书。本书以程序设计为主线，采用了渐进式的体系结构，在详细阐述程序设计基本概念、原理和方法的基础上，结合实践教学和学科竞赛的实际情况，通过大量经典实例讲解和实训，帮助学生掌握利用 C 语言进行结构化程序设计的技术和方法，提高他们的实践动手能力和培养创新协作精神。

相对第 1 版而言，第 2 版主要做了以下修改：

1）根据这几年用书单位的反馈，对一些章节的安排和组织进行了调整。

2）根据课程组近几年实施开放视频课程的经验，引入了以一个实例贯穿整个课程的授课策略。实例由简单到复杂到优化，循序渐进地演化，通过实际应用场景的不断变化和实例功能的不断扩展，依次引入 C 语言的各个语法元素，从工程的角度阐述各个 C 语言概念。每个语法的引入，都通过实例的实际环境无缝连接，并采用对比等教学手段，加强学生对知识点的理解和运用，特别是加深学生对各个知识点使用场合的理解。课程学完后，一个完整的程序也完成了。这种有一定代码量的实例，能规避通常教学中由小例子导致的“只见树木不见森林”“一叶障目不见泰山”的缺陷，有利于培养学生的工程实践能力。

3）更新了许多教学示例，重写了第 12 章和第 13 章。在第 13 章中，引入了一些有趣的游戏实例和加解密、权限管理等工程概念，以培养学生的工程实践能力。

本书分为两部分。第一部分（第 1 ～ 11 章）主要介绍 C 语言的基础语法知识，这部分内容按 C 语言的知识点循序渐进地介绍，同时针对 C 语言中的重点和难点，例如指针部分，精心设计了丰富的实例，用了大量的篇幅从不同方面对其进行讲解，旨在帮助读者理解并掌握这些重点和难点。第二部分（第 12 ～ 13 章）为项目实训和常用算法指导，通过项目开发全过程的全方位指导，从需求分析、算法设计到程序编写和过程调试，以项目实训的形式引导和帮助学生解决实际问题，提高学生解决具体问题的能力，并对程序设计竞赛中常见的一些算法及其应用进行了介绍。在教学过程中，教师应注重融入良好编程风格和程序调试相关知识的介绍，本书网站及华章网站上将提供相应的教学素材，供教师参考。

C 语言程序设计是一门强调实践练习的课程，因此教师对本书的教学组织可依据两条主

脉络进行：一条是从字、词、数据、表达式、语句到函数、数组、指针，这也是语法范畴构成的基本脉络；另一条则以程序功能（即以组织数据和组织程序）为基本脉络。安排课程内容时应注意以下几点：①介绍程序设计语言语法时要突出重点。C 语言语法比较庞杂，有些语句可以相互替代，有些语法不常使用。课程中要重点介绍基本的、常用的语法，不要面面俱到。②注重程序设计语言的共性。计算机的发展日新月异，大学期间不可能介绍所有的计算机语言，所以在本课程的学习过程中，教师应该介绍计算机程序设计语言共性的东西，使学生具有自学其他程序设计语言的能力。③由于课时的限制，课程不能安排太多的时间专门讲授程序设计理论。在教学过程中，教师应以程序设计为主线，结合教材中的实例分析，将程序设计的一般方法和技术传授给学生。

本书由浅入深地介绍了程序设计的技术与技巧，内容全面、自成一体，对启发、提高读者的程序设计能力很有裨益，适合不同层次的读者学习。本书可作为计算机类专业的本科或专科教材，也可以作为信息类或其他相关专业的选修教材，还可以作为其他一些课程的辅助读物，如数据结构、编译器设计、操作系统、计算机图形学、嵌入式系统及其他要用 C 语言进行项目设计的课程。

本书的作者均为浙江工商大学承担程序设计、数据结构等课程的骨干教师。凌云负责全书的策划、组织和指导，谢满德负责编写第 1、2、12、13 章，并负责对全书进行统稿和校对，陈志贤负责编写第 6、7、8、9、10、11 章，吴海燕负责编写第 3、4、5 章。

本书及其配套实验用书《C 语言程序设计与实践实验指导》已经入选“十二五”普通高等教育本科国家级规划教材，也是浙江省精品课程“高级语言程序设计”的教学用书。除本书外，我们还提供了多媒体电子教案、习题与实验指导，以及教学网站和教学资源库等开放资源。读者可以上网共享我们的网络资源，网址为：e-lesson.zjgsu.edu.cn。

在本书的编写过程中，我们参考了部分图书资料和网站资料，在此向文献的原作者表示衷心的感谢。由于作者水平有限，书中恐有不足之处，恳请业界同仁及读者朋友提出宝贵意见和真诚的批评。

作者

2016 年 11 月

教学建议

教学内容	学习要点及教学要求	课时安排 / 学时	
		计算机专业	非计算机专业
第 1 章 C 语言与程序设计概述	了解指令与程序的概念，了解程序设计的过程，了解 C 语言的历史、特点及其程序结构	2	2
第 2 章 示例驱动的 C 语言语法元素	了解 C 语言的基本语法元素，包括变量与常量、算术运算、控制流、函数、数组、基本输入 / 输出等，让学生对 C 语言有一个整体的感性认识，能模仿编写简单的小程序	2	2
第 3 章 基本数据类型和表达式	了解 C 语言的各种数据类型，掌握整型常量、浮点常量、字符常量的表示法，掌握各种运算符和表达式	4	4
第 4 章 输入 / 输出语句	掌握数据输出（printf、putchar）函数和数据输入（scanf、getchar）函数，熟练使用输入 / 输出语句中常用的格式说明、控制字符串	2	2
第 5 章 C 语言程序结构	了解语句的分类、结构化程序设计的基本概念，掌握循环、分支等控制语句的语法，并能熟练使用这些流程控制语句编写小的程序	6	6
第 6 章 数组	了解数组在内存中的表示方法，掌握数组（一维、二维、字符数组）的定义、引用和应用，掌握数组的典型应用示例，能利用数组编程解决实际问题	6	6
第 7 章 函数	了解基于函数的 C 语言程序组织方式，掌握函数的定义、函数的调用、函数参数的传递规则、内部函数和外部函数、变量的 4 种存储类别声明以及变量的作用域和生存期。本章的重点与难点在于基于函数参数的传递、嵌套函数和递归函数及其应用以及变量的作用域和生存期及其应用	6 ～ 8	6（选讲）
第 8 章 编译预处理	了解编译预处理的 3 种方式，掌握文件包含和宏定义的使用方法	2	2
第 9 章 指针	了解地址的基本概念及地址在 C 语言中的表示方法，掌握变量和函数的地址在 C 语言中的表示方法、指针变量的定义和引用、指针作为函数参数、指针与数组的关系、指针的运算、字符指针、字符串处理函数、指针数组、指向指针的指针、指向函数的指针以及命令行参数的传递	6 ～ 10	6（选讲）
第 10 章 结构与联合	掌握结构类型的定义、结构变量的定义和引用、结构数组的定义和引用、结构变量的参数传递规则、指向结构变量的指针、结构指针，以及链表的建立和链表元素的插入、删除、查找等内容。掌握联合和枚举类型的定义及变量的定义和引用。本章的难点在于链表的基本操作	6 ～ 10	6 ～ 8（选讲）

（续）

教学内容	学习要点及教学要求	课时安排 / 学时	
		计算机专业	非计算机专业
第 11 章 文件操作	了解文件的概念、文本文件和二进制文件的概念以及非缓冲文件的概念。掌握缓冲文件指针的定义、缓冲文件的打开和关闭以及缓冲文件读和写（文本文件方式、二进制文件方式）	2	2
第 12 章 综合实训	通过项目开发过程的全方位指导，将所学的知识点串起来。本章详细分析了几个实际项目的开发全过程，从需求分析、算法设计到程序编写、过程调试，通过实例指导，引导和帮助学生解决实际问题，提高学生解决具体问题的能力	2 ～ 4	2 ～ 6（选讲）
第 13 章 初涉 ACM/ICPC	本章结合程序设计大赛将常见算法分为 9 类加以介绍，包括这些算法的应用实例	2 ～ 6（选讲）	2（选讲）
	教学总学时建议	50 ～ 64	50 ～ 54

说明：

1）本书作为计算机专业本科学生的 C 语言程序设计教学用书时，建议课堂授课学时为 50 ～ 64（包含习题课、课堂讨论等必要的课堂教学环节，实验另行安排学时）。不同学校可以根据各自的教学要求和计划学时数酌情对教学内容进行取舍。其中，第 12 章实训部分可以选取其中一个例子详细讲解，其他例子让学生自学完成。第 13 章的内容可在开放实验教学中体现，并在整个课程教学过程中贯穿编程风格与程序调试的介绍。

2）非计算机专业的师生在使用本书时应适当降低教学要求。第 12、13 章可以不介绍。若授课学时数少于 50，则建议适当简化第 7 章中的递归和第 10 章中的链表部分的内容。

课堂教学建议：

1）本书的基础部分是第 5 ～ 11 章，这一部分从字、词、数据、表达式、语句到函数、数组、指针等，是语法范畴构成的基本脉络；建议教师在程序示例中融入语法元素，而不要单纯讲语法，以避免学生产生厌倦情绪。

2）C 语言程序设计是一门强调实践练习的课程，对本书的教学组织应以程序设计为主线，介绍程序设计语言语法时重点要突出，不要面面俱到。可以尝试对一个例子不断进行扩充，来逐渐引入新的语法元素，产生新的程序设计效果。

3）注重程序设计语言的共性。计算机的发展日新月异，大学期间不可能介绍所有的计算机语言。所以在本课程的学习过程中，教师应该介绍计算机程序设计语言共性的东西，使学生具有自学其他程序设计语言的能力。

4）如果受课时的限制，第 13 章可以略去不讲。应该在平时授课过程中，自然地将编程风格和程序调试的内容融入进去。

实验教学建议：

本书有配套实验用书《C 语言程序统计与实践实验指导》，可以参考其中的内容来布置实验内容。

目　录

第 1 章　C 语言与程序设计概述

1.1　初见 C 语言程序

我国古代数学家张邱建在其编写的《算经》里提出了历史上著名的“百钱买百鸡”问题：今有鸡翁一，值钱五；鸡母一，值钱三；鸡雏三，值钱一。凡百钱买鸡百只，问鸡翁、母、雏各几何？对于这个问题，很多读者在小学或初中的竞赛中可能都见到过，而且通常都采用不定方程求解。现在我们用 C 语言解决该问题。通过例 1-1 所示的程序，初学者一方面可以对 C 语言有一个感性的认识，另一方面可以初步领略计算机高效和强大的解决问题的能力。

例 1-1　用 C 语言程序解决“百钱买百鸡”问题。

```
#include <stdio.h>                          /* 包含标准库的信息 */
int main()                                  /* 定义名为 main 的函数，它不接受参数值 */
{
    int x, y, z, money;                     /* 声明 x, y, z, money 为整型变量 */
    printf("cocks hens chicks\n");          /* 输出表头信息 */
    for (x = 0; x <= 20; x++)               /* 控制循环次数，x 由 0 变到 20，共循环 21 次 */
        for (y = 0; y <= 33; y++)           /* 控制循环次数，y 由 0 变到 33，共循环 34 次 */
            for (z = 0; z <= 100; z++)
            {
                money = 5 * x + 3 * y + z / 3;
                if (x + y + z == 100 && money == 100 && z % 3 == 0)
                    printf("%5d%5d%7d\n", x, y, z);     /* 输出可行解 */
            }
    return 0;
}
```

运行程序，得到图 1-1 所示的结果。

```
cocks hens chicks
    0   25     75
    4   18     78
    8   11     81
   12    4     84
```

图 1-1　例 1-1 的运行结果

例 1-1 显示了一个完整的 C 语言程序，虽然规模很小，功能很简单，但能解决一个实际的问题。从程序中可以看出，在该问题的求解过程中，我们采用穷举法对所有可能的组合逐一进行检测，将符合要求的筛选出来。假设购买的鸡翁数量为 x，购买的鸡母数量为 y，购买的鸡雏数量为 z，共买鸡 100 只，则 x、y、z 均应小于等于 100。进一步分析，如果 100 元钱全部用来买鸡翁，则最多可买鸡翁 20 只，因此 x 的取值范围为 0 ～ 20，同理，y 的取值范围为 0 ～ 33，z 的取值范围为 0 ～ 100。对以上范围内所有 x、y、z 的组合，如果 x + y + z 的总和为 100，并且购买 x、y、z 花费的总钱数为 100，则 x、y、z 就是满足条件的解。事实上，穷举法是计算机求解问题时常用的一种方法。

例 1-1 所示的程序称为 C 语言的源程序。在 C 语言源程序的描述中，要注意以下几点：

1）C 语言源程序的扩展名必须为 .c 或 .cpp。

2）C 语言是大小写敏感的，也就是说，在 C 语言的源程序中，大小写是有区别的。

3）如果源程序中出现的逗号、分号、单引号和双引号等符号不是出现在双引号的内部，则均应该在英文半角状态下输入，比如分号不能写成中文分号，而应写成英文半角分号。

4）花括号、小括号、用作界定符的单引号和双引号等都必须成对出现。

例 1-1 是一个用 C 语言编写的解决实际问题的程序示例。读者可以思考一下，我们生活中碰到的哪些问题可以用类似的方法让计算机帮助我们解决。

1.2 计算机与程序设计

计算机的功能非常强大，能做非常复杂、人脑难以胜任的许多工作。然而，从电子市场买回 CPU、主板、内存、硬盘等硬件并组装好一台计算机后，你却发现这台计算机什么也做不了。究其原因，就是因为该计算机上还没有安装任何计算机程序，即软件。硬件是计算机拥有强大功能的前提条件，但是如果没有“大脑”（也就是计算机程序）去指挥它，它将什么也做不了，所以计算机程序的存在是计算机能够工作、能够按指定要求工作的必要条件。因此，计算机程序（Program，通常简称程序）可以简单理解为人们为解决某种问题而用计算机可以识别的代码所编排的一系列加工步骤。计算机能严格按照这些步骤去执行任务。计算机只是一个机器，只能按照既定的规则工作，这个规则是为了实现某个目标而人为制订的，因此我们制订的规则必须能够让计算机“理解”，才能使其按要求去工作，人们按照计算机能够理解的“语言”来制订这些规则的过程，就是程序设计的过程。

1.2.1 指令与程序

计算机的功能强大，但是没有智能，而且每次只能完成非常简单的任务。计算机必须通过一系列的简单任务有序组合才能完成复杂任务。因此，人只能以一个简单任务接一个简单任务的方式来对计算机发出指令。这个简单任务称为计算机的**指令**。一条指令本身只能完成一个最基本的功能，如实现一次加法运算或一次大小的判别。不同的指令能完成不同的简单任务。但是通过对多条指令的有序组织，就能完成非常复杂的工作，这一系列计算机指令（也可理解为人的命令）的有序组合就构成了**程序**，对这些指令的组织过程就是编程的过程，组织规则就是编程的语法规则。

例 1-2 假设计算机能识别的指令有以下四条。

Input X：输入数据到存储单元 X 中。

Add X Y Z：将 X、Y 相加并将结果存到 Z 中。

Inv X：将 X 求反后存回 X。

Output X：输出 X 的内容。

请编写一段由上述指令组成的虚拟程序，实现以下功能：输入 3 个数 A、B 和 C，求 A+B−C 的结果。

程序如下：

```
Input A;          输入第 1 个数据到存储单元 A 中
Input B;          输入第 2 个数据到存储单元 B 中
Input C;          输入第 3 个数据到存储单元 C 中
Add A B D;        将 A、B 相加并将结果存在 D 中
Inv C;
Add C D D;        将 C、D 相加并将结果存在 D 中
Output D;         输出 D 的内容
```

由例 1-2 可以看出，通过指令的有序组合，能完成单条指令无法完成的工作。上述程序中的指令是假设的，事实上，不同 CPU 支持的指令集也不同（由 CPU 硬件生产商决定提供哪些指令）。有点硬件常识的读者都知道，计算机的 CPU 和内存等都是集成电路，其能存储

和处理的对象只能是 0、1 组成的数字序列。因此这些指令也必须以 0、1 序列表示，最终程序在计算机中也是以 0、1 组成的指令码（用 0、1 序列编码表示的计算机指令）来表示的，这个序列能够被计算机 CPU 所识别。程序与数据均存储在存储器中。运行程序时，将准备运行的指令从内存调入 CPU 中，由 CPU 处理这条指令。CPU 依次处理内存中的所有指令，这就是程序的运行过程。

1.2.2 程序与程序设计

计算机程序是人们为解决某种问题用计算机可以识别的代码编排的一系列数据处理步骤，是计算机能识别的一系列指令的集合。计算机能严格按照这些步骤和指令去操作。**程序设计**就是针对实际问题，根据计算机的特点，编排能解决这些问题的步骤。程序是结果和目标，程序设计是过程。

1.2.3 程序设计和程序设计语言

程序设计是按指定要求编排计算机能识别的特定指令组合的过程，而**程序设计语言**是为方便人进行程序设计而提供的一种手段，是人与计算机交流的语言。程序设计语言随着计算机技术的发展而不断发展。

计算机能直接识别的是由“0”和“1”组成的二进制数——二进制是计算机语言的基础。一开始，人们只能用计算机能直接理解的语言去命令计算机工作，即写出一串串由“0”和“1”组成的指令序列交给计算机执行，这种语言称为**机器语言**。使用机器语言编写程序是一项十分痛苦的工作，特别是在程序有错需要查找、修改时更是如此。而且，由于每台计算机的指令系统往往各不相同，因此在一台计算机上执行的程序，要想在另一台计算机上执行，必须重新修改程序，这就造成了重复工作。所以，现在已经很少有人用机器语言直接写程序。

为了减轻使用机器语言编程的痛苦，人们进行了一种有益的改进：用一些简洁的英文字母、有一定含义的符号串来替代一个特定指令的二进制串，比如，用“ADD”代表加法，用“SUB”代表减法，用“MOV”代表数据传递等，这样一来，人们很容易读懂并理解程序在干什么，从而使得纠错及维护都变得方便了，这种程序设计语言称为**汇编语言**，即第二代计算机语言。然而对于计算机而言，它只认识“0”和“1”组成的指令，并不认识这些符号，这就需要一个专门的程序来将这些符号翻译成计算机能直接识别和理解的二进制数的机器语言，完成这种工作的程序被称为**汇编程序**，它充当的就是一个翻译者的角色。汇编语言同样十分依赖于机器硬件，移植性不好，但效率很高。现代的桌面计算机，功能已经非常强大，效率已经不是首要关注目标。所以，通常只有在资源受限的嵌入式环境或与硬件相关的程序设计（如驱动程序）过程中，汇编语言才会作为一种首选的软件开发语言。

虽然机器语言发展到汇编语言已经有了很大的进步，但是由于每条指令完成的工作非常有限，因此编程过程仍然很烦琐，语义表达仍然比较费力。于是，人们期望有更加方便、功能更加强大的高级编程语言。这种高级语言应该接近于数学语言或人的自然语言，同时又不依赖于计算机硬件，编出的程序能在所有机器上通用。C 语言就是一种能满足这种要求的语言，它既有高级语言的通用性又有底层语言的高效性，展示出了强大的生命力，几十年来一直被广泛应用。许多高校也将 C 语言作为计算机专业和相关专业的重要必修课，作为高校在校学生接触的第一门编程语言。同样，计算机本身并不“认识”C 语言程序，因此需要将 C 语言程序先翻译成汇编程序，再将汇编程序翻译成机器语言，这个过程往往由编译程序来

完成。

为了使程序设计更加接近自然语言的表达，方便用户实现功能，包括C语言在内的所有程序设计语言必须具有数据表达和数据处理（称为控制）这两方面的能力。

1. 数据表达

为了充分有效地表达各种各样的数据，人们通常会对常见数据进行归纳总结，确定其共性，最终尽可能地将所有数据抽象为若干种类型。数据类型（data type）就是对某些具有共同特点的数据集合的总称。如常说的整数、实数就是数据类型的例子。

在程序设计语言中，一般都事先定义几种基本的数据类型供程序员直接使用，如C语言中的整型、浮点型、字符型等。这些基本数据类型在程序中的具体对象主要有两种形式：常量（constant）和变量（variable）。常量在程序中是不变的，例如，987是一个整型常量。对于变量，则可对其做一些相关的操作，例如，改变它的值。

同时，为了使程序员能更充分地表达各种复杂的数据，C等程序设计语言还提供了丰富的构造新数据类型的手段，如数组（array）、结构（struct）、联合（union）、文件（file）和指针（pointer）等。

2. 数据处理的流程控制

高级程序设计语言除了能有效地表达各种各样的数据外，还必须能对数据进行有效的处理，提供一种手段来表达数据处理的过程，即程序的控制过程。

一种比较典型的程序设计方法是：将复杂程序划分为若干个相互独立的模块，使每个模块的工作变得单一而明确，在设计一个模块时不受其他模块的影响。同时，通过现有模块积木式地扩展又可以形成复杂的、更大的程序模块或程序。这种程序设计方法就是结构化程序设计方法，C语言就是典型的采用这种设计方法的语言。按照结构化程序设计的观点，任何程序都可以将模块通过三种基本的控制结构（顺序、选择和循环）的组合来实现。

当要处理的问题比较复杂时，为了增强程序的可读性和可维护性，我们常常将程序分为若干个相对独立的子模块，在C语言中，子模块的实现通过函数完成。

1.2.4 程序设计过程

采用高级程序设计语言，指挥计算机完成特定功能，解决实际问题的程序设计过程通常包括以下几个步骤：

1）明确功能需求。程序员通过交流和资料归纳，总结和明确系统的具体功能要求，并用自然语言描述出来。

2）系统分析。根据功能要求，分析解决问题的基本思路和方法，也就是常说的算法设计。

3）编写程序。程序员根据系统分析和程序结构编写程序。这一过程我们称为编程，最后将所编写的程序存入一个或多个文件，这些文件称为源文件。一般把用C按照其语法规则编写的未经编译的字符序列称为源程序（source code，又称源代码）。

4）编译程序。通过编译工具，将编写好的源文件编译成计算机可以识别的指令集合，最后形成可执行的程序。这一过程包括编译和链接。计算机硬件能理解的只有计算机的指令，也就是0、1组成的指令码，用程序设计语言编写的程序不能被计算机直接接受，这就需要一个软件将相应的程序“翻译”成计算机能直接理解的指令序列。对C语言等许多高级程序设计语言来说，这种软件就是编译器（compiler），因此编译器充当着类似于“翻译”的

角色，其精通两种语言：机器语言和高级程序设计语言。编译器首先要对源程序进行词法分析，然后进行语法与语义分析，最后生成可执行的代码。

5）程序调试。运行程序，检查其有没有按要求完成指定的工作，如果没有，则回到第3步和第4步，修改源程序，形成可执行程序，再检查，直到获得正确结果。

为了使程序编辑（Edit）、编译（Compile）、调试（Debug）等过程简单，方便操作，许多程序设计语言都有相应的编程环境（称为集成开发环境 IDE）。程序员可以直接在该环境中完成程序编辑、代码编译，如果程序出错还可以提供错误提示、可视化的快捷有效的调试工具等。所以，在 IDE 环境下，程序员可以专注于程序设计本身，而不用关心编辑、编译的操作方法。

在 Windows 操作系统下，C 语言的集成开发环境主要有：

1）Borland 公司的 Turbo C 环境。

2）Microsoft 公司的 Visual C++ 环境。

在 Linux 操作系统下，C 语言的集成开发环境主要有：

1）Eclipse。

2）GCC、g++ 等开源工具。

本书所有示例和实验均在 Visual C++ 6.0 环境下进行。

1.3 C 语言学习与自然语言学习的关系

C 语言相对来说是一门比较难的语言，很多初学者学了很久还一头雾水，不知道到底要学些什么、怎么学。本书拟通过对 C 语言学习过程与自然语言学习过程进行对照，使初学者能从熟悉的自然语言学习中理解 C 语言学习的方法和内容。

学习任何一门新的自然语言，都是先学一个个的字或单词，掌握它们的含义和用法；然后学习词语或短语，理解其构词方法和含义；再学习句法，包括句子结构、句型、造句语法、使用场合；最后学习文章写法，包括根据题目进行分析、段落组织、逻辑语义划分、句型组织等。这些都是学习自然语言的基本内容。但是如果只学好这些，只能说会一种语言，离灵活运用、精通一门语言还有很大的差距。运用一门语言最重要、最直接的途径就是写文章，一篇合格的文章必须没有语法错误，且必须紧扣题意。没有语法错误是写文章的基本要求，但是没有语法错误并不能说明该文章就是一篇合格的文章。如果下笔千言，但是离题万里，这样的文章还是不合格。所以在保证无语法错误的前提下，文章必须紧扣题意，满足题目要求。要写出一篇优秀的文章，则还要求论述充分、观点独到、行文流畅等。

C 语言也是一门语言，是一门用于与计算机交流的语言，因此其学习方法和过程与学习通常的自然语言基本相似。也就是说，我们首先要学习 C 语言中的所有“单词”，即关键字的含义和用法，然后学习通过这些“单词”组成的词语与短语的含义，以及通过“单词”组成短语的方法；再学习 C 语言语句的基本句型、语法特点、使用场合和使用方法；最后学习写文章，即程序的写法，包括根据题目进行分析，段落组织（函数、模块划分），句型应用等。这些都是学习 C 语言的基本内容，但是只学好这些，离灵活运用、精通 C 语言还有很大的差距。运用 C 语言最重要、最直接的途径就是按照要求编写合格的 C 语言程序，一个合格的 C 语言程序必须能够在没有语法错误的情况下解决指定的问题。遵守 C 语言语法规则，没有语法错误是编写程序的基本要求，但是没有语法错误并不能说明该程序就是一个正确的程序。如果程序编写得很“唯美”，但是没有解决指定的问题，这样的程序还是不合

格的。所以在保证无语法错误的前提下，程序必须解决了指定的问题，获得了期望的结果。而一个优秀的程序，则还应具备书写风格良好、解决问题的方法独到、具有较高的效率等特征。

通过上述对比可以发现，学习C语言与学习任何一门自然语言具有相似的步骤，只是这个“文章”必须通过程序语言进行书写。

1.4 C语言的发展历史、现状与特点

1.4.1 C语言的发展历史和现状

C语言的发展历史可以追溯到1961年的ALGOL 60，它是C语言的祖先。ALGOL 60是一种面向问题的高级语言，与计算机硬件的距离比较远，不适合用来编写系统软件。1963年，英国剑桥大学推出了CPL（Combined Programming Language）。CPL对ALGOL 60进行了改造，在ALGOL 60基础上接近硬件一些，但是规模较大，难以实现。1967年，英国剑桥大学的Martin Richards对CPL进行了简化，在保持CPL的基本优点的基础上推出了BCPL（Basic Combined Programming Language）。1970～1971年，美国AT&T公司贝尔实验室的Ken Thompson对BCPL进行进一步简化，设计出了非常简单而且很接近硬件的B语言（取BCPL的第一个字母），并用B语言改写了UNIX操作系统。但B语言过于简单，且功能有限。1972～1973年，贝尔实验室的Dennis M. Ritchie在B语言的基础上设计出了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（精练、接近硬件），又克服了它们的缺点（过于简单、无数据类型等）。最初的C语言只是为了描述和实现UNIX操作系统提供一种工作语言而设计的。1973年，Ken Thompson和Dennis M. Ritchie两人合作把UNIX中90%以上的代码用C语言改写，即UNIX第5版（最初的UNIX操作系统全部采用PDP-7汇编语言编写）。

后来，C语言历经多次改进，但主要还是在贝尔实验室内部使用。直到1975年，UNIX第6版公布以后，C语言的突出优点才引起人们的普遍关注。1975年，不依赖于具体机器的C语言编译文本（可移植C语言编译程序）出现了，使C语言移植到其他机器时所需做的工作大大简化，这也推动了UNIX操作系统迅速在各种机器上实现。随着UNIX的广泛使用，C语言也迅速得到推广。C语言和UNIX可以说是一对孪生兄弟，在发展过程中相辅相成。1978年以后，C语言已先后移植到大、中、小和微型计算机上，已独立于UNIX和PDP计算机了。

现在，C语言已风靡全世界，成为世界上应用最广泛的几种计算机语言之一。许多系统软件和实用的软件包，如Microsoft Windows等，都是用C语言编写的。图1-2表示了C语言的“家谱”。

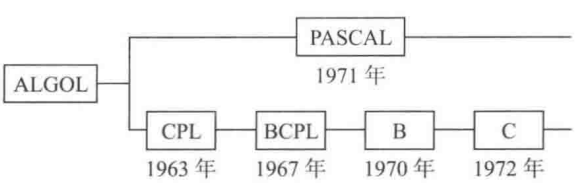

图1-2 C语言的“家谱”

以1978年发表的UNIX第7版中的C语言编译程序为基础，Brian W. Kernighan和Dennis M. Ritchie（合称K&R）合著了影响深远的经典著作《The C Programming Language》，这本书中介绍的C语言成为后来广泛使用的各种C语言版本的基础，被称为**旧标准C**。1983年，美国国家标准协会（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。ANSI C比旧标准C有了很大的发展。1987年，ANSI又公布了新标准——87 ANSI C，K&R于1988年修