

21 世纪高校计算机课程教材

新编 C 语言程序设计教程

朱贵良 等编著

孙玉强 主 审

人 民 邮 电 出 版 社

图书在版编目(CIP)数据

新编 C 语言程序设计教程 / 朱贵良等编著. —北京: 人民邮电出版社, 2002.8

21 世纪高校计算机课程教材

ISBN 7-115-10539-1

. 新... . 朱... . C 语言—程序设计—高等学校—教材 . TP312

中国版本图书馆 CIP 数据核字(2002)第 059564 号

内 容 提 要

本书编写的目的旨在提高 C 语言的教学质量,改善客观上存在的 C 语言难讲、难学、难以掌握的现状,着眼于培养学生独立编程能力和对程序设计语言的悟性,按照难点分解、平滑过渡、循序渐进的思路组织结构框架和编配内容。各章均配有大量附有答案的习题,努力为课堂教学过渡到独立编程提供模拟环境和更大的知识惯性。本书内容分两部分:第 1~3 章介绍了 C 语言程序设计的基础知识,这也是高级语言的共性部分,第 4~10 章是 C 语言程序设计的提高部分,对 C 语言公认的难点(例如指针、结构体、函数、数组等)进行了详细分解和重点阐述。

本书适合高校本专科生的第一语言或第二语言教学用书。亦可供各类 C 语言学习对象阅读参考。

出版说明

C 语言是现代程序设计中最活跃的高级语言之一。在可视化语言出现之前,目前最流行的 Windows、UNIX、Linux 三大操作系统都是用 C 语言写的。实际上,可视化语言能够实现的东西,用 C 语言几乎都可以实现。在高校教学过程中,C 语言的预期教学目标与实际教学效果存在较大距离,C 语言难教、难学、难以掌握的现象普遍存在。因此,本书从 C 语言体系的框架结构及内容出发,对提高 C 语言课程教学质量的途径与方法,减轻来自教与学两个方面的“三难”压力,进行了认真分析和探讨。当然能否实现初衷,还需要通过实践检验。

- 本书的特点之一是对 C 语言公认的难点进行了分解,例如指针、结构体、函数、数组等,本着先易后难、平滑过渡、循序渐进、易读易懂的思路进行了整体构造设计。在结构体系方面充分注意到章节知识点的渐进性、前后表述的一致性和疑难点的可释性。

- 本书的特点之二是针对解题难、实验难的问题,结合全国计算机等级考试和高校计算机等级考试编附了配有答案的大量习题,这对于深化许多重要概念,掌握各章知识要点、解读知识难点会有很大帮助,也会使读者节约许多时间。C 语言程序设计教学的最终目的是培养学生又快又好的独立编程能力。缩短课堂教学到编程能力养成的时间差,最有效的方法是阅读大量配有答案的习题,然后进行编程模拟训练,当编程模拟量积累到一定程度时,破题编程就会变的轻松自如。

- 本书的特点之三是强调引导性教学,一方面强调教师精讲,另一方面强调学生完成自学内容,各章内容都提出了课堂讲授与指导性自学建设性计划,每章最后都对内容的重点要点进行了提炼性小结。学生在学习各章自学内容之前,教师应花 10~20 分钟时间讲解有关要点,引导学生应掌握哪些知识点,注意哪些问题。自学能力是大学生的重要培养目标之一。对自学部分的重要章节,建议至少要安排一次学生交流讨论机会。

本书共分 10 章,1~3 章是 C 语言程序设计的基础部分,也是高级语言的共性部分;4~10 章是 C 程序设计的提高部分。对于课时比较充分的院校(60~70 课时),课时分配计划建议为:课堂教学与上机实习为 40~50 学时,其中包括若干次的课堂讨论。大型作业安排 2~4 个为宜(15~20 个学时)。安排大型系统作业的目的是用于系统巩固全书各章知识的要点和重点。

朱贵良对全书进行了统编修订并撰写了第 1~4 章;第 5~7 章由胡子义和朱贵良编写;第 8~9 章由胡子义和连卫民编写;第 10 章由朱贵良和王巍编写。全书习题由连卫民编写调试,最后进行了集体修订。河南师范大学的孙玉强教授仔细审阅了全书,并提出宝贵的建议。本书在编写过程中始终得到河南省教育厅、河南省高校许多同仁的指导和关心,在此一并表示衷心感谢。

我们的愿望是好的,工作是努力的,但限于时间、水平和能力,书中难免存在不妥之处,恳请同仁读者不吝赐教,以期再版或再次印刷时,得到修订,使本书更加完善。

编 者
2002 年 5 月

目 录

第 1 章 初识 C 语言	1
1.1 学习 C 语言的必要性	1
1.2 C 语言发展简史	1
1.3 几个重要术语概念	2
1.4 算法与描述工具	3
1.4.1 算法	3
1.4.2 伪代码	3
1.4.3 传统流程图	3
1.4.4 N-S 流程图	4
1.5 设计一个好程序	4
1.5.1 好程序的标准	4
1.5.2 结构化程序设计方法简介	5
1.6 C 语言特点	6
1.7 关键字与标识符	7
1.7.1 关键字	7
1.7.2 标识符	8
1.8 C 程序的基本结构与语句组成	8
1.8.1 C 程序的基本结构	8
1.8.2 C 程序的语句组成和特点	9
1.9 Turbo C 2.0 简介与上机操作	10
1.9.1 程序运行步骤	10
1.9.2 Turbo C 2.0 编译系统简介	12
1.9.3 上机操作	12
本章小结	17
习题	18
第 2 章 C 语言的数据类型	21
2.1 C 语言的数据类型	21
2.2 常量类型	22
2.2.1 整型常量	22
2.2.2 实型常量	23
2.2.3 字符常量	23
2.2.4 转义字符常量	24
2.2.5 字符串常量	25

2.2.6 符号常量	25
2.3 变量	26
2.3.1 变量概述	26
2.3.2 变量的存储类型	27
2.3.3 变量数据类型	28
2.3.4 变量的作用范围	29
2.3.5 变量说明举例	31
2.4 C 语言的运算符	33
2.5 C 语言的表达式	34
2.5.1 基本算术表达式	35
2.5.2 数据类型转换表达式	36
2.5.3 赋值表达式	37
2.5.4 复合运算符的赋值表达式	37
2.5.5 自加 1、自减 1 算术表达式	38
2.5.6 逗号运算符和逗号表达式	40
本章小结	40
习题	41
第 3 章 程序设计初步	45
3.1 文件包含预处理命令	45
3.2 输出函数 printf()	46
3.2.1 printf()函数应用格式说明	46
3.2.2 printf()函数使用说明	50
3.3 输入函数 scanf()	51
3.3.1 scanf()函数应用格式说明	51
3.3.2 scanf()函数使用说明	52
3.4 选择结构程序设计	53
3.4.1 关系运算和逻辑运算	54
3.4.2 if 语句	56
3.4.3 条件表达式构成的选择结构	61
3.4.4 switch 语句	61
3.5 循环结构程序设计	65
3.5.1 while 语句	66
3.5.2 do-while 语句	68
3.5.3 for 语句	70
3.5.4 break 语句与 continue 语句	73
3.5.5 循环的嵌套	75
3.6 位运算	78
3.6.1 位逻辑运算符	78

3.6.2 位移运算符	79
3.6.3 位自反赋值运算符	81
本章小结	82
习题	83
第 4 章 数组基础	92
4.1 一维数组	92
4.1.1 一维数组的定义	93
4.1.2 一维数组的初始化与赋初值	93
4.1.3 一维数组的引用	95
4.1.4 一维数组应用举例	96
4.2 字符数组与字符串	97
4.2.1 字符数组的定义	98
4.2.2 字符数组初始化	98
4.2.3 字符数组的引用	99
4.2.4 字符数组与字符串的关系	100
4.2.5 字符串的输入与输出	101
4.3 字符串处理函数	102
4.4 二维数组	106
4.4.1 二维数组的定义	106
4.4.2 二维数组的初始化	107
4.4.3 二维数组的引用	108
4.4.4 二维数组应用举例	108
本章小结	110
习题	111
第 5 章 用户自定义数据类型	118
5.1 结构体类型	118
5.1.1 结构体类型的概念	118
5.1.2 定义结构体变量的方法与初始化	120
5.1.3 结构体变量的引用	122
5.2 结构体数组	124
5.2.1 定义结构体数组的方法	124
5.2.2 结构体数组的初始化	125
5.2.3 结构体数组的引用	125
5.3 共用体类型	127
5.3.1 共用体类型的概念	127
5.3.2 共用体变量成员的引用	128
5.4 枚举类型	129

5.5 类型标识符的重命名	131
本章小结	132
习题	133
第 6 章 指针	142
6.1 指针与指针变量的概念	142
6.1.1 变量地址、指针与指针变量	142
6.1.2 指针变量的定义与指针运算符	143
6.1.3 指针的初始化	144
6.2 指针与数组	145
6.2.1 指针与一维数组	145
6.2.2 指针与二维数组	148
6.3 指针与字符串	152
6.3.1 字符串指针	152
6.3.2 指向字符串的指针变量的定义	154
6.3.3 指向字符串的指针变量的引用	154
6.4 指针数组	156
6.4.1 指针数组的概念	156
6.4.2 指针数组的使用	156
6.5 多级指针	159
6.5.1 多级指针的概念	159
6.5.2 多级指针的引用	160
6.6 指针数组与 main() 形参	161
本章小结	162
习题	163
第 7 章 自定义函数	173
7.1 函数的概念	173
7.2 函数间的数据传递	176
7.2.1 实参与形参	176
7.2.2 变量作为函数参数	177
7.2.3 用数组作为函数参数	180
7.2.4 结构体变量作为函数的参数	185
7.3 函数的嵌套调用与递归调用	185
7.3.1 函数的嵌套调用	185
7.3.2 函数的递归调用	187
7.4 内部函数与外部函数	190
7.4.1 内部函数	191
7.4.2 外部函数	192

本章小结	194
习题	195
第 8 章 指针与函数	206
8.1 指针函数	206
8.2 指针作为函数参数	209
8.3 指向函数的指针	211
8.4 指向函数的指针数组	214
8.5 指向 void 类型的指针及对指针的强制类型转换	215
本章小结	217
习题	218
第 9 章 指针与结构体	223
9.1 指针与结构体变量	223
9.1.1 指向结构体变量的指针	223
9.1.2 指向结构体数组的指针	225
9.1.3 用指向结构体变量的指针变量作函数的参数	226
9.2 单链表	229
9.2.1 单链表概念	229
9.2.2 链表的建立	231
9.2.3 单链表的插入与删除	233
9.2.4 输出链表	236
9.3 其它链式结构	237
本章小结	238
习题	238
第 10 章 文件	246
10.1 C 语言文件概述	246
10.1.1 文件的概念	246
10.1.2 文件的分类	247
10.1.3 文件的处理过程	249
10.1.4 文件类型指针	250
10.2 数据文件的 I/O	251
10.2.1 文件的打开和关闭	251
10.2.2 文件的字符 I/O 函数	252
10.2.3 文件的字符串 I/O 函数	254
10.2.4 文件的格式化 I/O 函数	257
10.2.5 文件的数据块 I/O 函数	259
10.2.6 整数 (字) I/O 函数	262

10.3 文件的随机 I/O	263
10.3.1 取得文件指针的当前位置：ftell()函数	263
10.3.2 改变文件指针的当前位置：fseek()函数	263
10.3.3 置文件指针于文件开头位置：rewind()函数	265
10.4 标准设备文件的 I/O	266
10.4.1 标准设备文件的换向	266
10.4.2 标准设备文件的字符 I/O 函数	267
10.4.3 标准设备文件的字符串 I/O 函数	267
10.5 文件的错误检测	268
10.6 程序设计举例	269
本章小结	274
习题	275
习题参考答案	282
附录 1 常用字符的 ASCII 码表	287
附录 2 C 语言的库函数	289
参考书目	293

第 1 章 初识 C 语言

● 本章教学目标与学习要求

通过本章教学，使读者了解 C 语言和 C 程序的基本结构和组成。熟悉两种流程图的作用与基本画法，认识并理解结构化程序设计要点，掌握上机操作的基本过程。

● 授课内容：1.4 节、1.8 节。引导学生自学其余各节。

1.1 学习 C 语言的必要性

当你开始学习本课程的时候，你可能没用过，但肯定听说过 Windows、Linux 和 UNIX 三个术语。这是目前最著名、最有影响、应用最广泛的三个操作系统（Operating System，简称 OS）。OS 是计算机系统（由软硬件两个子系统构成）的核心和灵魂。它是软件中最庞大最复杂的系统软件。如 Linux 2.3.16 版的源代码有 200 多万行，其中核心代码就有 20 多万行。Windows、Linux 和 UNIX 三个不同的 OS 都是用 C 语言编写的。既然如此庞大复杂的 OS 都可以用 C 语言编写，那么，就狭义而言，还有什么系统软件和应用软件不能用 C 语言编写呢！国内外的一些学者评价 C 语言时说：“它是世界上最好的语言”；“它是进入计算机世界不可逾越的一道门槛”；“不学 C 语言就等于没学计算机”等等。严肃地说，这些近乎结论性的评价是不科学的，他们不适当地、过分地夸大了 C 语言。但有一点是肯定的，C 语言是一门十分优秀而又重要的语言。如果你要成为一名优秀的软件工程师，就必须认真地学好 C 语言。

1.2 C 语言发展简史

任何事物都有发生、发展、成熟、消亡四个阶段，C 语言也是如此。

20 世纪 60 年代曾经流行 ALGOL 60 语言，随着时间推移，ALGOL 语言不能满足计算机发展需要时，便由英国学者马丁·理查德（Mating Richards）在 ALGOL 语言的基础上发明了 BCPL 语言，简称 B 语言。1972 年美国贝尔实验室的布朗·W·卡尼汉（Biran W Kernighan）和丹尼斯·M·利奇（Dennis M Ritchie）两人对 BCPL 语言进行了成功的改造，改造后的 BCPL 语言命名为 C 语言，并用 C 成功地改写了 UNIX 操作系统。1983 年美国标准化协会（ANSI）制定了 C 语言标准，1987 年进行了修订完善。目前 PC 机上广泛使用

的 C 语言有 Turbo C、Microsoft C、Quick C 等版本都是以 ANSI C 为标准的。由于 Turbo C 工具库、函数库丰富，在我国又广为流行，所以本书涉及到的例题与习题均是在 Turbo C 下实现的。

1.3 几个重要术语概念

C 语言是 C 高级算法语言的简称。用 C 语言编写的程序叫 C 程序。C 程序的操作对象是数据。那么什么是数据、高级算法语言、指令、命令、程序等十分重要而又必须掌握的概念，有些学生到毕业时可能也是模糊不清的。

1. 数据

凡是能够被计算机接收、识别、处理的一切符号称之为数据。“1234”是数据，一幅图片，一首歌曲，一串有确切含意的字符，如“北京大学”、“雅鲁藏布江”也是数据。根据数据的不同属性，数据又可以划分为不同的类型，如能够进行算术运算的叫数值型数据；能够说明事务某个特征的一串字符叫字符型数据，如“好吃的羊肉串”、“ABCDE”等。

2. 指令

在计算机中，指令和命令是同义语（尽管有时候两者还有一点差异）。它指示计算机进行什么操作和操作什么。进行什么操作的符号叫操作码，操作的对象叫操作数。所以一条指令包含操作码和操作数两个部分。例如：`sum=2+3`；是一条 C 程序指令，该指令省略了操作码，但包括了加法运算和赋值运算两步操作，`sum`、`2`、`3`是指令的操作对象。

3. 程序

程序是什么？程序是符号+规则，用规定的符号和符合规则的一组有意义的有序指令集合称之为程序。为了解决同一个问题而编写的若干个程序的集合叫做程序集。在程序集中，首先被执行的程序叫主程序，在主程序执行期间被调用的程序称之为子程序。在程序定义中的“有意义”强调了程序要解决问题的目的性。

4. 机器指令和机器语言（低级语言）

计算机只能识别 0 和 1 两个符号，用 0 和 1 两个符号按一定规则组成一串具有确切含义的代码叫一条机器指令，机器指令的集合叫机器语言（低级语言）。机器语言是计算机唯一可以直接识别的语言。人们能够识别的文字、图片、音乐与常用的数学符号等，只有被转换为机器指令时才能够被计算机进行处理。

5. 高级算法语言

毋庸置疑，用机器语言编写的程序是难以看懂的，编写程序是一件非常困难的事。于是人们在不断认识的过程中，又发明并不断完善了多种高级算法语言。这种语言的高级性是指语言的书写表达更合乎、更接近人们的一般思维方式，更接近人们考虑问题的顺序性和复杂问题分解方式。用高级算法语言编写的程序计算机不能直接识别，它必须被中间件（编译程序）翻译（或叫转换）为机器语言格式才能被计算机执行。

目前常用的高级算法语言主要有：Fortran（用于工程数值计算）、Qbasic（用于小型数值计算）、COBOL（用于数据处理）、FoxPro（用于数据处理）和 C 语言（用于编写系统软件和应用软件）。汇编语言用于系统软件和工业过程控制，属于低级语言。

1.4 算法与描述工具

1.4.1 算法

算法是为了解决某个特定问题而采取的确定且有限的步骤。一个算法有以下三个特性。

1. 有穷性

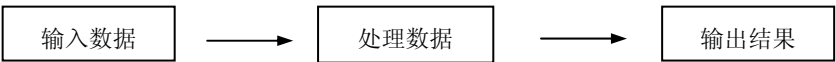
一个算法通过若干步骤即可实现预定目标，算法的每一步都可以在合理的时间内完成。

2. 确定性

算法中的每一条指令都必须有确切的含义，同样的步骤、同样的条件，结果也是相同的。

3. 输入和输出

计算机工作三步曲。



计算机工作三步曲具有不可逆的方向性。再简单再复杂的计算机概莫能外。也就是说，一个算法再简单，也要有输入、处理、输出三个步骤。

算法是可以描述的，描述算法的常用工具是伪代码和流程图。

1.4.2 伪代码

所谓伪代码是指用自然语言和指令语句（不要求绝对正确的语句）结合起来描述算法的一种方式。这种方式与画流程图相比省时省力，转换为程序容易，但是清晰度与层次性不如流程图。伪代码描述算法没有统一规定，写出来只要自己或别人能看懂即可。

1.4.3 传统流程图

传统流程图是描述算法的一种常用工具，常由图 1-1 中所示的几种基本图素组成。

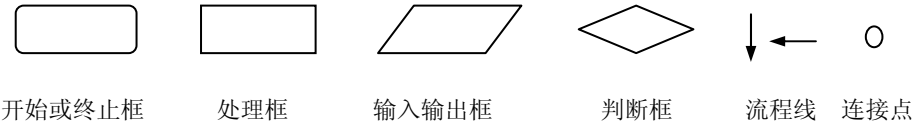


图 1-1 传统流程图的基本图素

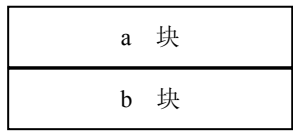
由这些框和流程线组成的流程图来表示算法，允许任意转向，形象直观，简单方便。但是，这种流程图描述复杂的算法时，所占篇幅较多，勾画费时费力，且不易阅读。

1.4.4 N-S 流程图

N-S 流程图是由美国学者 I.Nassi 和 B.Shneiderman 于 1973 年提出来用于描述算法的一种形式。算法的每一步都用一个矩形来描述。把一个个的矩形框按执行的先后次序连接起来就构成了一个完整的算法描述。N-S 流程图的特点是取消了流程线，禁止任意转向，描述复杂算法时和传统流程图相比所占篇幅要少得多，勾画时也省时省力，层次性、易读性要好于传统流程图。

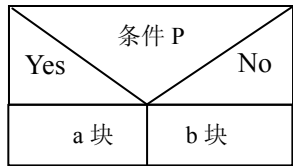
N-S 流程图常用的矩形框如图 1-2、图 1-3、图 1-4 和图 1-5 所示。

两种流程图描述算法举例参见以后各章的有关例题，要求读者熟悉三种算法的描述形式，并掌握 N-S 流程图的使用方法。



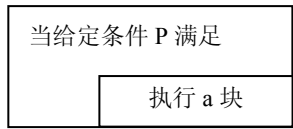
先执行 a 块语句，再执行 b 块语句。

图 1-2 顺序结构框



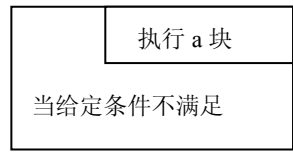
如果条件 P 成立（满足）则执行 a 块，如果条件 P 不能成立（不满足），则执行 b 块。

图 1-3 条件结构框



当给定条件满足执行 a 块一次，再次判断给定条件 P 是否满足，如果满足则再次执行 a 块一次。如此往复直到给定条件不满足为止。

图 1-4 循环结构框 1



先执行 a 块一次，判断给定条件是否满足，如果满足则再次执行 a 块一次。如此往复直到给定条件不满足为止。

图 1-5 循环结构框 2

1.5 设计一个好程序

1.5.1 好程序的标准

学习一门计算机语言的最终目的，是用该门语言又快又好的编出程序。一个好程序的

标准有以下四条。

1. 正确性

正确性是指给出正确的数据，得到正确的结果。毋庸置疑，程序运行结果不正确，前功尽弃，再好的程序也是空谈。保证结果的正确性由许多环节构成，主要有：所选题目存在正确结果；题目是可解的；解题算法是正确的；解题程序设计和调试是正确的、合理的。

2. 易读性

易读性是指编写的程序不但自己很容易看懂（即使隔了很长时间），而且别人也很容易看懂。编写的程序在很大程度上是给别人用的，别人拿到你的程序如何能够很快看懂是至关重要的。除在程序中使用注释语句外，交付使用的程序应配备完整规范的说明书。

3. 健壮性

健壮性是指容易维护，其具体内容包括：

- (1) 功能性扩充易于实现；
- (2) 不因为输入数据或使用人员的变换导致输出结果错误；
- (3) 程序的自纠错能力；
- (4) 安全性；
- (5) 提示数据输入错误及程序运行错误信息。

4. 可移植性

可移植性是指编写的程序在不同机型的同一语言环境下仍然能够正确运行。

好的程序必须遵循结构化程序设计方法，结构化是一个非常重要的概念，以后读者还可能会碰到“结构化系统分析”、“结构化系统设计”、“结构化对象”、“结构化系统实现”等与“结构化”相关的术语。这里的“结构化”的含意基本上可以理解为“标准”两字，标准即规范，即按照一定的规范进行程序设计。这种方法称之为结构化程序设计方法。

1.5.2 结构化程序设计方法简介

“结构化程序设计方法”是荷兰学者迪克特拉（Dijkstra）等人提出来的，其要点包括：

- 编写的程序易读易理解；
- 参加设计人员必须按照同一规划、同一原则进行程序设计；
- 顺序结构、选择结构和循环结构是大而复杂程序的三大基本结构；
- 采取“自顶向下、逐步细化”和模块化的设计方法。

自顶向下是指从解题的题目开始向下进行分解，如确定题目的最终要求、确定题目运行的环境平台、选择一种合适的计算机语言，按照题目的内容结构划分为若干个子题目（模块），对子题目进行算法描述，逐步细化到基本结构为止，即把大问题分解为小问题，直到小问题很容易解决。

模块化从要解决的问题出发，将问题按照功能划分为若干个模块，采用自顶向下逐步细化的一种方法，即把大模块分解成若干个功能相对独立的小模块，直到该模块能够完成单一的功能为止。一个模块是相对独立的一个功能，这个功能可以是子系统或一个子程序，如学籍管理信息系统至少包括：输入、统计、查询、打印四个子系统，每个子系统都是一

个模块。查询模块又可以划分为按姓名查询、按学号查询、按身份证号查询等子模块。模块具有相对的独立性。模块化便于组织分工，尤其是便于消除“水波现象”（因程序的一个错误而影响程序的全局的现象），对提高编程和调试效率有很大帮助。

养成良好的结构化程序设计方法是对一个软件工程师的基本要求。

1.6 C 语言特点

C 语言是一种计算机高级算法语言，由于 C 提供了某些接近汇编语言的功能，人们有时又把 C 语言称为中级语言。C 程序的描述都是以函数形式出现的，所以有时人们也把它叫做函数语言。

C 语言主要的优点在于适合编写系统软件，也适合编写应用软件。如本书封底的条形码标志，就是用 C 程序来实现的。C 程序的代码可以比汇编程序的代码减少 20%，且易读性和结构性都比汇编程序要好。汇编程序虽然也可以实现 C 程序的功能，但是不同的硬件环境，同一个题目必须编写出不同的汇编程序，也就是说，汇编程序的可移植性能很差，对硬件依赖性很大。但是在单片机、单板机和嵌入式 PC 机中，汇编语言的功能和作用是其任何语言不可替代的。

C 语言之所以比较优秀、应用广泛，是由其特点决定的。这些特点可大致归纳如下。

1. 语言简洁

C 语言只有 32 个关键字，除 BASIC 语言外，是所有计算机语言中关键字数量最少的。

2. 运算符丰富

C 语言有 44 种运算符，是目前计算机语言中具有运算符最多的一门高级语言。运算符多，可运算的对象种类也多。

3. 数据结构丰富

C 语言提供了多种特殊的、其它语言中所没有的数据类型，这些类型保证了 C 程序良好的结构化特性。

4. 具有良好的可移植性和丰富的库函数

C 程序本身不依赖机器硬件，所以具有良好的可移植性。例如 Turbo C 就提供了 300 多个库函数。

5. 生成高质量的目标代码

就效率而言，C 程序生成目标代码的效率仅比汇编程序低 15% 左右，这个数字与 C 语言的简洁、与 C 程序的易读和良好的可移植性相比几乎可以忽略不计。

任何事物的好与坏大都是相对的，C 语言也存在自己的不足。学习 C 语言与学习其它高级语言相比，有碎、繁、难的特点。由于 C 语言放宽了语法检查，增大了程序编写的灵活性，同时也增大了出错的概率，在一定程度上降低了系统的安全性（假如，系统是由数十个、数百个 C 程序构成）。学习过一门高级语言的读者还会感到，字符串处理、统计、

求和之类的功能，虽然通过C程序也能实现，但是显得笨手笨脚。尽管C语言有这些缺陷，但是瑕不掩瑜，与之优点相比也只不过是九牛一毛。

1.7 关键字与标识符

1.7.1 关键字

关键字又叫保留字，是C编译系统预先定义的一些具有特定含义的标识符。标准C（ANSI C）的关键字有存储类型符、数据类型符和语句定义符三类。它们只能按定义的方式加以使用，而不能作常规的标识符使用。因此，在程序中使用的变量名、语句标号、函数名等标识符时，不能与关键字相同。关键字共有三类。

1. ANSI C 关键字

ANSI C 规定了 32 个关键字，都是用小写字母组成的，如表 1-1 所示。

表 1-1 ANSI C 标准规定的 32 个关键字

1 auto	7 default	13 float	19 register	25 struct	31 while
2 break	8 do	14 for	20 return	26 switch	32 volatile
3 case	9 double	15 goto	21 short	27 typedef	
4 char	10 else	16 if	22 signed	28 union	
5 const	11 enum	17 int	23 sizeof	29 unsigned	
6 continue	12 extern	18 long	24 static	30 void	

2. PC 机扩充关键字

除 ANSI C 标准规定的 32 个关键字之外，为了充分利用 PC 机 80x86CPU 的存储结构，以及支持混合语言的编程设计、支持中断处理，Turbo C、Microsoft C 等还扩充了如下关键字，如表 1-2 所示。

表 1-2 PC 机上扩充的关键字

asm	_cs	_ds	_es	_ss	cdecl
far	huge	interrupt	neer	pascal	

3. 编译预处理中的专用命令

C 语言预处理中经常用到一些专用命令，如：

- (1) 宏定义预处理命令：#define。
- (2) 文件包含预处理命令：#include。
- (3) 条件编译预处理命令：#undef、#ifdef、#endif、#if、#else、#ifndef。

这些是 C 语言特有的编译预处理命令，其主要特征是它们均以“#”号开头，不同于前面的关键字。编译预处理命令虽然不属于关键字，但为了避免混乱，建议也把它们看成关键字，不要在程序中作为函数名或变量名使用。

1.7.2 标识符

在 C 程序中，能够标识函数、语句标号、变量的一串字符叫标识符。C 程序中使用标识符的具体规定如下：

- 标识符必须由英文字母、数字或下划线组成，且首字符只能是英文字母或下划线。正确的标识符，如：

b123 pust _abc TsX7 Lev_5

错误的标识符，如：

3H sUM\$ %asfg a*** @ffff

- C 程序中大、小写字母不等价。如：BIG、Big、big、bIG、BiG 等都是不同的标识符，如果在程序中用以表示变量，那么它们分别代表不同的变量。
- 禁止使用关键字作标识符。
- ANSI C 标准规定的标识符的长度为 1~32 个字符，但 PC 机规定的标识符的最大长度为 8 个字符，超过 8 个字符的部分不被识别（不同版本的 C 语言可能会略有差异）。

1.8 C 程序的基本结构与语句组成

1.8.1 C 程序的基本结构

每个 C 程序都是由一个或多个函数构成的。这些函数中有且仅有一个名为 main 的函数，称为主函数，也叫主程序。它是程序开始运行的第一个函数，而不管它在什么位置。其它函数相应的称为子程序或子函数。一个 C 程序中所用到的各个函数名必须是唯一的，C 程序的基本组成形式如下。

1. 主函数格式说明

预处理命令 { 宏定义 (经常使用)
 文件包含 (频繁使用)
 条件编译 (必要时使用)

main()

{

函数体 { 说明语句
 执行语句 { 单个语句
 复合语句

}