

世界著名计算机教材精选

标准

C 程序设计

(第5版)

E Balagurusamy 著
金名李丹程 等译
刘莹那俊

PROGRAMMING IN ANSI C

Fifth Edition

清华大学出版社



世界著名计算机教材精选

标准 C 程序设计

(第 5 版)

E Balagurusamy 著

金 名 李丹程
刘 莹 那 俊 等译

清华大学出版社

北 京

E Balagurusamy
Programming in ANSI C, 5e
EISBN:0-07-068182-1

Copyright © 2011 by The McGraw-Hill Companies, Inc.

Original language published by The McGraw-Hill Companies, Inc. All Rights reserved. No part of this publication may be reproduced or distributed by any means, or stored in a database or retrieval system, without the prior written permission of the publisher.

Simplified Chinese translation edition is published and distributed exclusively by Tsinghua University Press under the authorization by McGraw-Hill Education(Asia)Co., within the territory of the People's Republic of China only, excluding Hong Kong, Macao SAR and Taiwan. Unauthorized export of this edition is a violation of the Copyright Act. Violation of this Law is subject to Civil and Criminal Penalties.

本书中文简体字翻译版由美国麦格劳-希尔教育出版(亚洲)公司授权清华大学出版社在中华人民共和国境内(不包括中国香港、澳门特别行政区和中国台湾)独家出版发行。未经许可之出口,视为违反著作权法,将受法律之制裁。未经出版者预先书面许可,不得以任何方式复制或抄袭本书的任何部分。

北京市版权局著作权合同登记号 图字:01-2011-4035号

本书封面贴有 McGraw-Hill 公司防伪标签,无标签者不得销售。
版权所有,翻印必究。举报电话:010-62782989 13701121933

图书在版编目(CIP)数据

标准C程序设计(第5版)/(印)巴拉古路萨米(Balagurusamy, E.)著;金名,李丹程,刘莹,那俊等译. —北京:清华大学出版社,2011.9
(世界著名计算机教材精选)
书名原文:PROGRAMMING IN ANSI C, 5E
ISBN 978-7-302-26184-1

I. ①标… II. ①巴… ②金… ③李… III. ①C语言—程序设计—教材 IV. ①TP312

中国版本图书馆CIP数据核字(2011)第136974号

责任编辑:龙啟铭
责任校对:白蕾
责任印制:李红英

出版发行:清华大学出版社 地 址:北京清华大学学研大厦A座
http://www.tup.com.cn 邮 编:100084
社 总 机:010-62770175 邮 购:010-62786544
投稿与读者服务:010-62776969, c-service@tup.tsinghua.edu.cn
质 量 反 馈:010-62772015, zhiliang@tup.tsinghua.edu.cn

印 装 者:清华大学印刷厂
经 销:全国新华书店
开 本:185×260 印 张:32.5 字 数:785千字
版 次:2011年9月第5版 印 次:2011年9月第1次印刷
印 数:1~3000
定 价:59.00元

产品编号:042471-01

译 者 序

C 语言是所有本科生课程的一门核心课，也是现今使用最广泛的计算机语言。国内国外的 C 语言图书已经非常多了，但通过本书的翻译，我们觉得这本书还是很有引进价值的，具体表现在它的以下几个特点上：

- 本书基于最新的 C 语言标准。
- 附录 IV 给出了 4 个完整的应用程序开发示例。
- 扩展讨论了 C 的指针。
- 每章后面的“谨记”一节给出了很有用的编程提示以及容易出错的问题。
- 每章后面的案例学习给出了 20 多个真实的开发，展示了 C 程序的设计过程。
- 80 多个程序设计范例向读者阐述了良好程序设计的基本原则。
- 还有近 200 个复习题和 130 多个项目设计题。

总之，本书的语言简洁易懂，示例非常丰富且具有很强的实际指导意义，是一本很好的 C 程序设计的教材。

本书的前面版本被印度的很多学院和大学用作教材。在过去的几年中，本书还被一些重要的软件培训与开发公司用作培训教材。

由于本书的独特性、与众不同的学习方法以及简明的写作风格，本书第 3 版和第 4 版在国内出版已受到国内学生和老师的欢迎。

本书由金名、李丹程、刘莹、那俊、石凯等主译，张长富、李晓春、王春桥、龚亚萍、韦笑、何雄、周云、袁科萍、王雷、贺军、贺民、陈安南、霍丽娜、史广飞、侯鹏、张红军、董武、陈河南、王峰、沈宏、郑晓蕊、李伟、白晓平、李月、汤效平、李东锋、邵世磊、张新苗、刘大为、薛飞、邹晓东、陈占军、夏绪虎、刘占坤、冯苗、裘蕾、任世华、金颖、吴霞、韩毅、马以辉、樊庆红等人也参与了部分翻译工作。欢迎广大读者指正。

第5版前言

C 是功能强大、灵活性好、可移植且结构良好的程序设计语言。由于 C 集成了高级语言和汇编语言的优点，因此它可适用于系统和应用程序的开发。今天，C 语言无疑是使用最广的通用语言。

自从 1989 年标准化以来，C 经历了一系列的修订和改进，以提高该语言的可用性。现在，融合了这些新特性的版本称为 C99。

本书结构

本书首先在第 1 章概要地介绍了 C 语言，包括 C 的历史、C 程序的基本结构和运行。第 2 章讨论了如何声明常量、变量和数据类型。第 3 章讨论了内置操作符以及如何使用它们来构建表达式。第 4 章介绍了输入输出操作。第 5 章介绍了决策判断与分支，包括 if-else、switch 和 goto 语句。第 6 章讨论了决策与循环，包括 while、do 和 for 循环。第 7 章和第 8 章介绍了数组和数据元素的有序排列，这些对所有程序设计语言都很重要。第 8 章还介绍了字符串。第 9 章和第 10 章介绍了函数、结构和联合。指针是 C 语言中最难理解和掌握的部分了，第 11 章以对用户非常友好的方式介绍了指针。第 12 章和第 13 章分别介绍了文件管理和动态内存分配。第 14 章介绍了预处理器。最后，第 15 章开发了一个 C 程序，让读者了解如何进行程序的开发。如果读者按照本书结构认真学习，将可以很好地理解 C 语言。

本书内容进行了修订，以紧跟 C 程序设计领域的发展和当今的需要。与以往一样，全书仍然强调“用示例学习”的概念。在深入讨论了 C 语言的每个主要特性后，给出了一个完整的程序示例来演示其使用。示例程序既简单，又很有启发性。

每章的开头含有一小节，介绍本章的主要内容，并快速浏览了本章将要介绍的特色内容。只要需要，就使用图像来描述有关的概念，以提高描述的清晰性，方便读者理解。基本的语言技巧和其他需要特别考虑的内容以“注意”的形式凸显出来。为使本书更加实用，本书融合了以下主要特性。

- 带注释的代码全书随处可见，这些注释说明了 C 语言的各种特性是如何集成在一起以实现特定任务的。
- 补充信息和“注意”对正文进行了必要的补充，但独立于正文之外。
- 最后一章给出了开发高效的 C 程序的一些指导原则，以及经验不足的 C 程序员容易犯的常见错误列表。



- 每章末尾的**案例研究**演示了把 C 特性集成在一起的常用方式, 并显示了一些实际的应用。
- 每章末尾的“**谨记**”一节列举了一些有用的提示和可能出问题的地方。
- 每章末尾的“**问题**”和“**练习**”为读者复习所学概念和实际的应用开发提供机会。
- 附录给出的**编程项目**介绍了开发大型程序时如何集成 C 的各种特性。

目 录

第 1 章 C 语言概述	1	2.9 存储类的声明.....	34
1.1 C 语言的历史.....	1	2.10 变量的赋值.....	35
1.2 C 语言的主要特性.....	2	2.10.1 赋值语句.....	36
1.3 示例程序 1: 显示一条消息.....	3	2.10.2 从键盘读取数据.....	38
1.4 示例程序 2: 两个数相加.....	6	2.11 符号常量的定义.....	40
1.5 示例程序 3: 利息计算.....	7	2.11.1 可修改性.....	40
1.6 示例程序 4: 子例程的使用.....	9	2.11.2 可理解性.....	41
1.7 示例程序 5: 数学函数的使用.....	10	2.12 将变量声明为常量.....	42
1.8 C 程序的基本结构.....	12	2.13 将变量声明为可变的.....	42
1.9 编程风格.....	13	2.14 数据的溢出.....	42
1.10 运行一个程序.....	13	2.15 案例学习.....	43
1.11 UNIX 系统环境下.....	14	2.15.1 平均数计算.....	43
1.11.1 创建程序.....	14	2.15.2 温度转换问题.....	44
1.11.2 编译与链接.....	15		
1.11.3 运行程序.....	15	第 3 章 运算符与表达式	49
1.11.4 创建自己的可运行文件.....	16	3.1 概述.....	49
1.11.5 多个源文件问题.....	16	3.2 算术运算符.....	49
1.12 MS-DOS 系统环境下.....	17	3.2.1 整数算术运算.....	50
		3.2.2 实数算术运算.....	50
		3.2.3 混合算术运算.....	51
第 2 章 常量、变量及数据类型	22	3.3 关系运算符.....	51
2.1 概述.....	22	3.4 逻辑运算符.....	53
2.2 字符集.....	22	3.5 赋值运算符.....	54
三元字符.....	23	3.6 递增和递减运算符.....	55
2.3 C 标记符.....	23	3.7 条件运算符.....	56
2.4 关键字与标识符.....	24	3.8 逐位运算符.....	57
2.5 常量.....	24	3.9 特殊运算符.....	57
2.5.1 整型常量.....	24	3.9.1 逗号运算符.....	57
2.5.2 实数常量.....	25	3.9.2 sizeof 运算符.....	58
2.5.3 单字符常量.....	27	3.10 算术表达式.....	59
2.5.4 字符串常量.....	27	3.11 表达式的计算.....	60
2.5.5 反斜杠字符常量.....	27	3.12 算术表达式的优先级.....	61
2.6 变量.....	28	3.13 一些可计算性问题.....	62
2.7 数据类型.....	29	3.14 表达式中的类型转换.....	63
2.7.1 整数型.....	30	3.14.1 隐式类型转换.....	63
2.7.2 浮点数类型.....	30	3.14.2 显式类型转换.....	65
2.7.3 void 类型.....	31	3.15 运算符的优先级及其关联.....	67
2.7.4 字符类型.....	31	3.16 数学函数.....	68
2.8 变量的声明.....	31	3.17 案例学习.....	69
2.8.1 基本类型的声明.....	31	3.17.1 销售人员的工资.....	69
2.8.2 自定义类型的声明.....	33		



3.17.2 二次方程的求解	70	6.4.2 for 循环的其他特性	149
第 4 章 输入输出操作管理	78	6.4.3 for 循环的嵌套	150
4.1 概述	78	6.5 循环中的跳转	153
4.2 读取一个字符	78	6.5.1 跳出循环	153
4.3 写字符	81	6.5.2 跳过循环的一部分	157
4.4 格式化输入	82	6.5.3 避免使用 goto 语句	159
4.4.1 整数输入	83	6.5.4 跳出程序	159
4.4.2 实数输入	85	6.6 简洁的测试表达式	160
4.4.3 字符串输入	86	6.7 案例学习	161
4.4.4 混合数据类型的读取	88	6.7.1 二项式系数表	161
4.4.5 错误输入的检测	88	6.7.2 柱状图	162
4.4.6 使用 scanf 函数时应记住的 几个要点	90	6.7.3 最小成本	164
4.5 格式化输出	91	6.7.4 描绘两函数的曲线图	165
4.5.1 整数的输出	91	第 7 章 数组	175
4.5.2 实数的输出	92	7.1 概述	175
4.5.3 单个字符的显示	94	7.2 一维数组	176
4.5.4 字符串的显示	95	7.3 一维数组的声明	177
4.5.5 混合数据的输出	96	7.4 一维数组的初始化	179
4.5.6 提高输出的可读性	97	7.4.1 编译时初始化	180
4.6 案例学习	98	7.4.2 运行时初始化	181
4.6.1 库存报告	98	7.5 二维数组	183
4.6.2 可靠性图形	100	7.6 二维数组的初始化	187
第 5 章 判断与分支	106	7.7 多维数组	191
5.1 概述	106	7.8 动态数组	192
5.2 if 判断语句	106	7.9 与数组相关的其他内容	192
5.3 简单 if 语句	107	7.10 案例学习	193
5.4 if...else 语句	110	7.10.1 数列的中值问题	193
5.5 嵌套 if...else 语句	113	7.10.2 标准偏差的计算	196
5.6 阶梯式 else if 语句	116	7.10.3 测试评分	197
5.7 switch 语句	119	7.10.4 产品与销售分析	200
5.8 ?: 运算符	122	第 8 章 字符数组与字符串	212
5.9 goto 语句	125	8.1 概述	212
5.10 案例学习	128	8.2 字符串变量的声明与初始化	212
5.10.1 数值的分布范围	128	8.3 使用 scanf 函数从终端读取 字符串	214
5.10.2 账单计算	129	8.3.1 读取一行文本	216
第 6 章 判断与循环	140	8.3.2 使用 getchar 和 gets 函数	216
6.1 概述	140	8.4 在屏幕上显示字符串	218
6.2 while 语句	142	8.4.1 使用 printf 函数	218
6.3 do 语句	144	8.4.2 使用 putchar 和 puts 函数	222
6.4 for 语句	146	8.5 字符的算术运算	222
6.4.1 简单的 for 循环语句	146	8.6 将字符串组合在一起	224
		8.7 两个字符串的比较	225

8.8 字符串处理函数	226	9.20 多文件程序	280
8.8.1 strcat()函数	226	9.21 案例学习	283
8.8.2 strcmp()函数	227	曲线下的面积计算	283
8.8.3 strcpy()函数	228		
8.8.4 strlen()函数	228	第 10 章 结构体与共用体	292
8.8.5 其他字符串函数	229	10.1 概述	292
8.9 字符串表	231	10.2 结构体的定义	292
8.10 字符串的其他特性	233	10.3 声明结构体变量	293
8.11 案例学习	234	10.4 访问结构体成员	295
8.11.1 计算文本中的字数	234	10.5 结构体的初始化	296
8.11.2 客户列表处理程序	236	10.6 结构体变量的复制与比较	298
		10.7 单个成员的运算	299
第 9 章 用户自定义函数	243	10.8 结构数组	300
9.1 概述	243	10.9 结构体中的数组	302
9.2 为什么需要自定义函数	243	10.10 结构体中的结构体	303
9.3 多函数程序	244	10.11 结构体与函数	305
9.4 自定义函数的元素	246	10.12 共用体	308
9.5 函数的定义	246	10.13 结构体的大小	309
9.5.1 函数头	247	10.14 位域	309
9.5.2 函数名与类型	247	10.15 案例学习	312
9.5.3 形参列表	247	书店库存	312
9.5.4 函数体	248		
9.6 返回值及其类型	249	第 11 章 指针	322
9.7 函数调用	250	11.1 概述	322
9.8 函数声明	252	11.2 理解指针	322
9.9 函数的类型	254	11.3 访问变量的地址	324
9.10 无参数无返回值的函数	254	11.4 指针变量的声明	325
9.11 有参数无返回值的函数	256	11.5 指针变量的初始化	326
9.12 有参数有返回值的函数	259	11.6 通过指针访问变量	328
返回浮点值	261	11.7 指针链	330
9.13 无参数但有一个返回值的函数	262	11.8 指针表达式	330
9.14 返回多个值的函数	263	11.9 指针的递增与比例因子	332
9.15 函数的嵌套	264	11.10 指针与数组	333
9.16 函数的迭代	265	11.11 指针与字符串	336
9.17 将数组传递给函数	266	11.12 指针数组	338
9.17.1 一维数组	266	11.13 指针作为函数的参数	339
9.17.2 二维数组	270	11.14 函数返回指针	342
9.18 将字符串传递给函数	270	11.15 指向函数的指针	342
9.19 变量的作用域、可见性和生存期	271	11.16 指针与结构体	345
9.19.1 自动变量	272	11.17 案例学习	348
9.19.2 外部变量	273	11.17.1 考试成绩处理程序	348
9.19.3 外部声明	276	11.17.2 库存更新程序	351
9.19.4 静态变量	277		
9.19.5 寄存器变量	278	第 12 章 文件管理	357
9.19.6 嵌套代码块	279	12.1 概述	357



12.2	定义并打开文件	358	14.4.4	情形 4	414
12.3	关闭文件	359	14.5	ANSI C 的其他预处理器指令	415
12.4	文件的输入输出操作	360	14.5.1	#elif 指令	416
12.4.1	getc 与 putc 函数	360	14.5.2	#pragma 指令	416
12.4.2	getw 和 putw 函数	361	14.5.3	#error 指令	416
12.4.3	fprintf 与 fscanf 函数	363	14.5.4	字符串化运算符#	417
12.5	I/O 操作的错误处理	365	14.5.5	标记符粘贴运算符##	417
12.6	随机访问文件	367			
12.7	命令行参数	372			
第 13 章	动态内存分配与链表	377	第 15 章	C 程序开发指导原则	420
13.1	概述	377	15.1	概述	420
13.2	动态内存分配	377	15.2	程序设计	420
	内存分配过程	377	15.2.1	问题分析	420
13.3	用 malloc 函数分配一块内存	378	15.2.2	勾勒程序结构	420
13.4	用 calloc 函数分配多个内存块	380	15.2.3	算法开发	421
13.5	用 free 函数释放已用的空间	380	15.2.4	控制结构	421
13.6	用 realloc 函数改变内存块的 大小	381	15.3	程序编码	422
13.7	链表的概念	382	15.3.1	自身文档化	422
13.8	链表的优点	385	15.3.2	语句构造	423
13.9	链表的种类	386	15.3.3	输入/输出格式	423
13.10	再论指针	386	15.3.4	程序的通用性	423
13.11	创建链表	388	15.4	常见的程序错误	424
13.12	插入一个数据项	391	15.4.1	丢失分号	424
13.13	删除一个数据项	394	15.4.2	误用分号	424
13.14	链表的应用	396	15.4.3	丢失括号	425
13.15	案例学习	397	15.4.4	丢失引号	426
13.15.1	在已排序链表中插入 数据	397	15.4.5	误用引号	426
13.15.2	构建一个已排序的 链表	401	15.4.6	使用不正确的注释字符	426
			15.4.7	未定义变量	427
第 14 章	预处理器	407	15.4.8	忽视了运算符的优先级	427
14.1	概述	407	15.4.9	忽视了递增递减运算符的 计算顺序	428
14.2	宏替换指令	407	15.4.10	忽视了函数参数的说明	428
14.2.1	简单宏替换	408	15.4.11	在函数调用中实参和 形参类型不匹配	428
14.2.2	含参数的宏	410	15.4.12	函数未声明	428
14.2.3	宏嵌套	411	15.4.13	在 scanf 的参数中丢失了 &运算符	429
14.2.4	取消宏定义	411	15.4.14	超出了数组的边界	429
14.3	文件包含	412	15.4.15	忘记了给字符串的空字符 留出空间	430
14.4	编译器控制指令	412	15.4.16	使用未初始化的指针	430
14.4.1	情形 1	413	15.4.17	丢失了间接运算符和 地址运算符	430
14.4.2	情形 2	414	15.4.18	在指针表达式中 丢失括号	431
14.4.3	情形 3	414			

15.4.19 在宏定义语句中参数 遗漏了括号	431	IV.4 矩阵相乘	495
15.5 程序测试与调试	431	附录 V C99 的特性	498
15.5.1 错误的类型	432	V.1 概述	498
15.5.2 程序测试	432	V.2 新关键字	498
15.5.3 程序调试	433	V.3 新注释	498
15.6 程序的效率	434	V.4 新数据类型	499
15.6.1 运行时间	434	V.4.1 <code>_Bool</code> 类型	499
15.6.2 内存需求	434	V.4.2 <code>_Complex</code> 和 <code>_Imaginary</code> 类型	499
附录 I 位级程序设计	436	V.4.3 <code>long long</code> 类型	499
I.1 概述	436	V.5 变量声明	500
I.2 逐位逻辑运算符	436	V.6 I/O 格式化的变化	500
I.2.1 逐位与操作	436	V.7 数组处理	500
I.2.2 逐位或操作	438	V.7.1 可变长度数组	501
I.2.3 逐位非或操作	438	V.7.2 数组声明中的类型说明	501
I.3 逐位移位运算符	438	V.7.3 结构的灵活数组成员	501
I.4 逐位求反运算符	439	V.8 函数实现	501
I.5 掩码计算	440	V.8.1 默认为 <code>inline</code> 类型的 规则	502
附录 II 字符的 ASCII 值	441	V.8.2 显式函数定义	502
附录 III 标准 C 的库函数	442	V.8.3 <code>return</code> 语句的限制	502
附录 IV 项目设计	445	V.8.4 声明函数为 <code>inline</code>	503
IV.1 库存管理系统	445	V.9 受限指针	503
IV.2 登录记录系统	467	V.10 编译器限制的变化	504
IV.3 链表	487	V.11 其他改进	504
		参考文献	505

第 1 章 C语言概述

1.1 C语言的历史

对于一种程序设计语言，字母“C”看上去是一个奇怪的名字。但是这个奇怪而好听的语言却是现今最为流行的计算机语言之一，因为它是一种结构化的、高级的、与机器无关的语言。它使得软件开发人员在开发程序时，无须担心实现这些程序的硬件平台。

所有现代语言的起源都是 ALGOL 语言，该语言是在 20 世纪 60 年代早期引入的。ALGOL 是最先使用块结构的计算机语言。尽管它从来没有在美国流行开来，但在欧洲却被广泛使用。ALGOL 语言给计算机科学界带来了结构化程序设计的概念。20 世纪 60 年代，计算机科学家，如 Corrado Bohm、Guiseppe Jacopini 和 Edsger Dijkstra 使这一概念大众化了。随后，又宣布开发了好几种计算机语言。

1967 年，Martin Richards 开发了一种称为 BCPL（基本组合程序设计语言）的计算机语言。该语言主要用于系统软件的开发。1970 年，Ken Thompson 创建了一种计算机语言，该语言继承了 BCPL 的很多特性，就简单地称为 B 语言。在贝尔实验室，B 语言用来开发 UNIX 操作系统的早期版本。BCPL 和 B 语言都是“无类型”的系统程序设计语言。

C 语言是 Dennis Ritchie 于 1972 年在贝尔实验室从 ALGOL、BCPL 和 B 语言的基础上发展而来的。C 语言利用了这些语言的很多概念，并添加了数据类型的概念以及其他功能强大的特性。由于它是与 UNIX 操作系统一起被开发出来的，因此它与 UNIX 有着很强的关联。UNIX 操作系统（也是在贝尔实验室开发出来的）几乎完全是用 C 语言编码的。UNIX 是现今使用最为流行的网络操作系统，也是因特网数据超高速路的心脏。

多年以来，C 语言主要用于科研环境下，但后来，随着多种商用 C 编译器的发布，以及 UNIX 操作系统的不断流行，在计算机专业中也开始获得广泛支持。今天，C 语言可以运行在多种操作系统和硬件平台下。

20 世纪 70 年代，C 语言发展为现在所说的“传统 C 语言”。自 1978 年由 Brian Kerningham 和 Dennis Ritchie 著作的“The C Programming Language”一书的出版，C 语言成了最为流行的语言。该书很受欢迎，以至于在程序设计界，C 语言就认为是“K&R C”。C 语言的快速发展导致了不同版本的语言出现，这些语言相互类似但往往不兼容。这给系统开发人员带来了一个严重的问题。

为了确保 C 语言的标准，1983 年，美国国家标准局（American National Standards Institute, ANSI）任命了一个技术委员会来定义 C 语言的标准。该委员会于 1989 年批准了一个 C 语言版本，这就是现在的 ANSI C。该版本又于 1990 年被国际标准组织（International Standards Organization, ISO）批准。C 语言的这个版本又称为 C89。

20 世纪 90 年代，一种完全基于 C 的语言——C++语言，经历了大量的改进和变化，于 1989 年 11 月成为了一种获得 ANSI/ISO 批准的语言。C++在 C 的基础上添加了一些新特

性,使之不仅成为一种真正的面向对象的语言,而且是一种更通用的语言。与此同时,美国的 Sun 公司创造了一种新的语言——Java 语言,它是以 C 和 C++为模型的。

所有流行的计算机语言其本质都是动态的。它们通过加入新特性来不断地提高其功能和使用范围,C 语言也不例外。尽管 C++和 Java 语言都是从 C 发展而来的,C 语言标准委员会认为,如果 C++/Java 语言的一些特性加入到 C 中,会提高 C 语言的性能。于是就产生了 C 语言的 1999 标准。这个版本的 C 语言通常称为 C99。C 语言的历史和发展如图 1.1 所示。

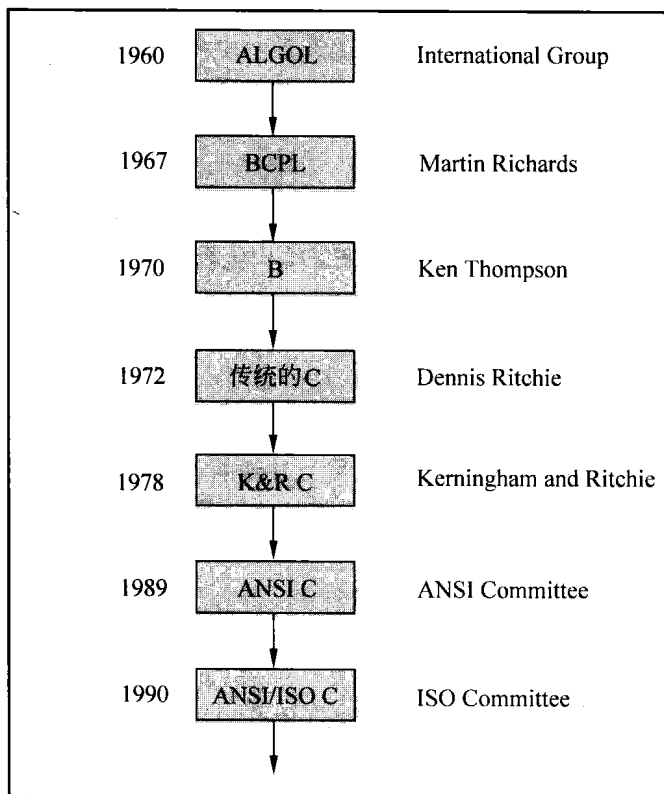


图 1.1 C 语言的历史

尽管 C99 是一个改进版本,但很多常用的编译器仍不支持 C99 的所有新特性。因此,我们将在附录中单独介绍加入到 C99 的新特性,这样,感兴趣的读者就可以快速地参考这些新特性,并在可能的情况下使用它们。

1.2 C 语言的主要特性

C 语言之所以越来越受欢迎,是因为它具有很多可取的特性。它是一种健壮的语言,其丰富的内置函数集和运算符可用来编写任意复杂的程序。C 语言编译器融合了汇编语言的性能和高级语言的特性,因此很适合编写系统软件和商业软件。事实上,市场上很多的 C 编译器都是用 C 语言编写的。

用C语言编写的程序高效且运行速度快。这归功于它的各种数据类型和功能强大的运算符。其运行速度是BASIC语言的好几倍。例如，一个计算0~15000个变量的程序，用C语言编写，其运行时间为1秒，而用BASIC语言编写的则要运行50秒。

C语言有32个关键字，它们的长度取决于其内置函数。C语言有很多标准函数可用于开发程序。

C语言是高度可移植的。这意味着，为某一台计算机编写的C程序，只要稍作修改甚至不用修改就可以在另一台计算机上运行。如果我们计划使用不同操作系统的新计算机，可移植性很重要。

C语言很适于结构化程序设计，因而要求用户以功能模块的方式来思考问题。这些模块的恰当结合，就可以形成一个完整的程序。这种模块化的结构使得程序的调试、测试和维护更加容易。

C语言另一个重要特性是它的自我扩展能力。一个C程序基本上是各种函数的集合，这些函数由C函数库支持。我们可以不断地将自己的函数添加到C函数库中去。由于有了这么大量的函数，程序设计就变得简单了。

在讨论C语言的具体特征之前，我们先来看看一个C范例程序，分析并了解程序是如何工作的。

1.3 示例程序 1：显示一条消息

请看图1.2所示的一个很简单的程序。

```
main()
{
    /*.....显示开始.....*/

    printf("I see, I remember");

    /*.....显示结束.....*/
}
```

图 1.2 显示一行文字的程序

该程序将产生如下的输出：

```
I see, I remember.
```

让我们来详细看看该程序。第一行告诉操作系统程序名是main，从这一行开始执行。main()函数是C系统使用的一个特殊函数，用来告诉计算机程序的运行起点。每个程序必须正好有一个main函数。如果有多个main函数，编译器就无法知道哪一个是程序的起始点。紧跟在main后面的空括号对表明main函数不带参数。关于参数的概念我们将在讨论函数（第9章）时再详细介绍。

第二行的开始是“{”，表明main函数的开始，而最后一行的闭括号则表示该函数的结

束。在上面范例程序中，闭括号还标志着整个程序的结束。这两个括号之中的所有语句就形成了函数体。函数体包含有一个指令集，从而完成指定的任务。

在上面的示例程序中，函数体包含有 3 个语句，其中只有 `printf` 一行是可执行的语句。以 `/*` 开始，以 `*/` 结尾的行，称为注释行。程序中恰当地使用注释行，可以提高程序的可读性。更容易让人理解。由于注释行不是可执行语句，因此 `/*` 与 `*/` 之间的内容全部被编译器忽略掉。通常，一个注释可以插入到程序的任何空白处——行的开始、中间或结尾处——但不能插入到一个词的中间。

尽管注释行可以出现在程序的任意地方，但在 C 语言中它们不能嵌套。这就意味着，不能在注释行中再插入注释行。一旦编译器发现了注释的开始标志，它就将忽略掉后面的所有内容，直到再发现一个结束标志为止。下面注释行：

```
/* = = /* = = */. = = */
```

是不合法的，因此将产生一个错误。

由于注释行不会影响程序的运行速度以及编译后程序的大小，因此我们应大方地在程序中使用注释。注释有助于开发人员和其他用户理解程序的不同函数和运算，对程序的调试和测试也有帮助。我们将在后面的范例程序中体会到注释行的作用。

现在让我们来看 `printf()` 函数，这是该范例程序中唯一可执行的语句：

```
printf("I see, I remember! ");
```

`printf` 是预定义的标准的 C 函数，用于显示输出。预定义的含义就是，该函数已编写好并已编译。在链接时，与我们的程序链接在一起。编译和链接的概念将在本章的后面介绍。`printf` 函数将两个引号之间的内容显示出来。在本示例程序中，其输出为：

```
I see, I remember!
```

注意，打印行以分号结尾。C 语言的每条语句都必须以分号结尾。

假如我们要如下地将输出显示成两行：

```
I see,
I remember!
```

这可以通过添加两个 `printf` 函数来实现，具体如下：

```
printf("I see, \n ");
printf("I remember! ");
```

两个括号之间的信息称为该函数的参数。第一个 `printf` 函数的参数是 `"I see,\n"`，第二个的是 `"I remember!"`。这些参数只是要显示出来的字符串。

注意，第一个 `printf` 函数的参数在字符串的末尾包含有字符 `\` 和 `n` 的组合。该组合称为新行字符。新行字符命令计算机换到下一行（即新行）。在概念上，它类似于打字机的回车键。在显示了逗号字符后，新行字符 `n` 的出现将使 `"I remember!"` 显示在下一行。字符 `\` 和 `n` 之间不允许有空格。

如果省略掉第一个 `printf` 语句的新行字符，那么输出仍为一行，即如下所示：

```
I see,I remember!
```

这类似于图 1.2 中的程序的输出。但是，注意，在“,”和“I”之间没有空格。

只使用一个 `printf` 语句，只要在适当的地方使用换行字符，也可以生成两行或多行输出。例如，语句：

```
printf("I see, \n I remember! ");
```

的输出为：

```
I see,
I remember!
```

而语句：

```
printf("I \n.. see, \n... .. I\n... .. remember! ");
```

的输出为：

```
I
.. see,
... .. I
... .. remember!
```

注意，有些作者推荐在所有程序的开头，加上如下的输入输出库函数：

```
# include <stdio.h>
```

但是，对 `printf` 和 `scanf` 并没有必要，因为这两个函数已经定义为 C 语言的一部分了。关于输入输出函数的更多内容参见第 4 章。

在我们继续讨论其他更多的示例之前，应必须注意很重要的一点：C 语言是区分大小写字母的。例如，`printf` 和 `PRINTF` 并不相同。在 C 语言中，通常都是写成小写字母。而大写字母用作表示常量的符号名。我们也可以在输出字符串中使用大写字母，例如，“I SEE”和“I REMEMBER”。

上面介绍的用于显示“I see, I remember!”的示例是最简单的程序之一。图 1.3 列举了这类简单程序的一般格式。所有 C 程序都需要一个 `main` 函数。

main 函数

`main` 函数是每个 C 程序的一部分。C 语言允许有如下多种形式的 `main` 函数声明：

```
main()
int main()
void main()
main(void)
void main(void)
int main(void)
```

空括号对表示该函数不带参数。这也可以在括号中使用关键字 `void` 来明确表示不带参数。我们还可以在 `main` 之前指定 `int` 或 `void`。关键字 `void` 表示该函数不给操作系统返回任何信息，而 `int` 表示函数将返回一个整数值给操作系统。如果指定为 `int`，那么程序的最后一行必须是“`return 0;`”。

为了简单起见，我们在本书的所有示例程序中使用第一种形式的 `main` 函数。

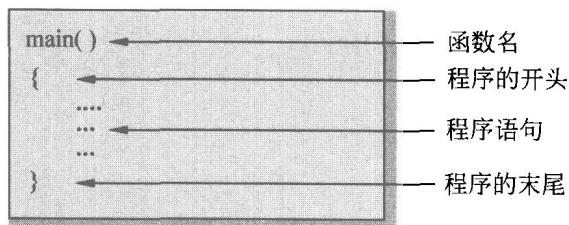


图 1.3 简单程序的结构

1.4 示例程序 2：两个数相加

下面来看另一个程序，它执行两个数的加法运算并显示其结果。整个程序如图 1.4 所示。

```

/* 程序名: ADDITION                                line-1 */
/* 作者: EBG                                       line-2 */
main() /*                                           line-3 */
{ /* line-4 */
    int number; /* line-5 */
    float amount; /* line-6 */
                                     /* line-7 */
    number = 100; /* line-8 */
                                     /* line-9 */
    amount = 30.75 + 75.35; /* line-10 */
    printf("%d\n", number); /* line-11 */
    printf("%5.2f", amount); /* line-12 */
} /* line-13 */

```

图 1.4 两个数相加的程序

当运行该程序时，将产生如下的输出：

```

100
106.10

```

该程序的头两行为注释行。在开始处添加注释行，用于给出诸如程序名、作者、编写日期等信息。这是一种很好的做法。其他行中也使用了注释符号，用于表示行号。

`number` 和 `amount` 字为变量名，用来存储数字数据。数字数据可以是整数型或实数型。在 C 语言中，所有变量都必须进行声明，从而告诉编译器所使用的变量名是什么、它们的数据类型是什么。变量必须在使用之前声明。在第 5 和第 6 行中，变量声明语句：

```

int number;
float amount;

```