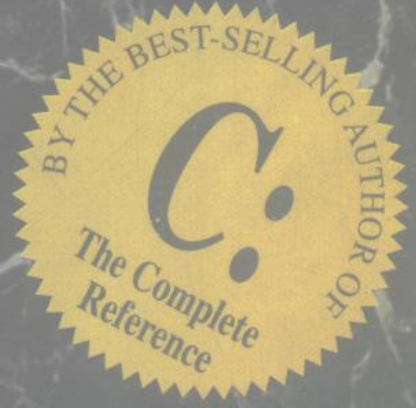
Osborne McGraw-Hill

C++ 语言大全

C++: the Complete Reference

[美] Herbert Schildt 著
杨长虹 徐 磊 等 译



McGraw-Hill
電子工業出版社

C++: The Complete Reference

C++语言大全

赫伯特 刘德
〔美〕Herbert Schildt 著
杨长虹 徐 碧 等译
刘德贵 校

电子工业出版社

(京)新登字 055 号

内 容 简 介

《C++语言大会》是根据国际著名的 C 语言专家 Herbert Schildt 的有关专著翻译的。这是一本 C++语言的百科全书，包括 C 和 C++的命令、功能、编程和应用等方面的内容，是集专家及 C 语言编程人员的多方面专门经验之作。全书分三大部分，共 23 章和一个附录。第一部分为 C++基础，讨论 C 和 C++语言的共性，内容包括表达式、C 语句、数组和串、指针、函数、数据类型、输出和输入与处理等；第二部分详细介绍了 C++本身的专有特性，讨论了类和对象、虚函数和继承性、构造函数和析构函数、面向目标的输入与输出系统以及函数和运算符重载特性等；第三部分用实例讨论了 C++软件的开发技术，包括字符串、弹出窗口和链表等基本应用软件结构技术。

本书内容全面，叙述清晰，为 PC 机及其广大用户和程序开发人员提供了一部很有用的工具书，是计算机应用人员、有关大专院校师生及 PC 机软件开发人员的必备参考书。

参加本书翻译的还有曾凡奎和李刚。

JS11 6/10

C++, The Complete Reference
〔美〕Herbert Schildt 著
Osborne McGraw-Hill 1991 年出版

C++语言大全
杨长虹 徐 碚 等译
刘德贵 校
责任编辑 路石
特邀编辑 杨宝琪
*
电子工业出版社出版（北京市万寿路）
电子工业出版社发行 各地新华书店经销
电子工业出版社出版计算机排版室排版
北京市顺义县天竺颖华印刷厂印刷
*
开本：787×1092 毫米 1/16 印张：28.75 字数：700 千字
1994 年 8 月第一版 1994 年 8 月第一次印刷
印数：10100 册 定价：49.95 元
ISBN 7-5053-2739-9/TP·859

著 者 序 言

我从事了多年的C语言程序设计工作，在过去的七年中出版了不少关于C语言的书籍，似乎对它有特别的偏好。提起程序设计，我就非常激动，以致于不知道下面该说些什么。我相信C++是自C语言诞生以来，在程序设计艺术和学术上的最重大的飞跃。写这本书时，我坚信它一定能改进读者进行程序设计的方法。

C++代表着在程序设计方面的一个重要进步。在C语言的基础之上，C++作了一些功能扩充以支持面向对象的程序设计。同时，这些扩充极大地增强了这种语言的能力。程序设计的语言和方法，自二十世纪五十年代诞生以来，一直在不断地发展。C++代表了C语言的发展趋势。本书后半部分将阐述面向对象的程序设计，其目的就在于使程序设计人员可管理愈来愈大、愈来愈复杂的程序。因此，C++表现得特别成功。

然而，值得注意的是，C++不仅仅有益于面向对象的程序设计，而且由于其技术先进，以致在编写非面向对象的程序时也变得特别的容易。

在写这本书的时候，C语言早已成为专业编程人员的基本设计工具。由于C++是C语言的增强型版本，在不久的将来它有望得到极大的普及应用。现在，许多计算机产业分析家预言：九十年代将是C++的年代。坦率地讲，C++的能力及其通用性，以及C语言普及所构成的技术基础，实际上已经注定了它的成功。

本书介绍了C++的类C特性及其专有特性。但书中重点还是放在C++的专有特性方面。鉴于许多读者已熟悉并精通了C语言，我们将C++语言的类C特性放在C++特性之前单独讨论。这种安排可以避免有C语言编程丰富经验的读者重复涉足大量他们早已熟悉的知识。这样，有C语言编程经验的人员可直接开始阅读C++技术内容。

本书的第一部分介绍了C++语言与C语言的共性内容，第二部分详细讨论了C++对C语言扩充和增强的内容，第三部分列举了应用C++和面向对象程序设计的若干实例供读者参考。

目　录

第一部分　C++ 基础:类 C 特性

第一章　C 语言概述 …… (2)
1.1　C 语言的起源 …… (2)
1.2　C 语言是中级语言 …… (2)
1.3　C 语言是结构化语言 …… (3)
1.4　C 语言是程序员的语言 …… (4)
1.5　C 语言的程序结构 …… (5)
1.6　库和连接 …… (6)
1.7　分离编译 …… (7)
第二章　表达式 …… (8)
2.1　五种基本数据类型 …… (8)
2.2　修饰基本类型 …… (9)
2.3　标识符命名 …… (10)
2.4　变量 …… (10)
2.4.1　变量在哪里说明 …… (10)
2.4.2　局部变量 …… (11)
2.4.3　形式参数 …… (13)
2.4.4　全局变量 …… (13)
2.5　存取修饰符 …… (14)
2.5.1　const …… (15)
2.5.2　volatile …… (16)
2.6　存储分类符 …… (16)
2.6.1　外部的 …… (16)
2.6.2　静态变量 …… (17)
2.6.3　寄存器变量 …… (19)
2.7　变量初始化 …… (20)
2.8　常量 …… (21)
2.8.1　十六进制和八进制常量 …… (21)
2.8.2　串常量 …… (22)
2.8.3　反斜线字符常量 …… (22)
2.9　运算符 …… (23)
2.9.1　赋值运算符 …… (23)
2.9.2　赋值中的类型转换 …… (23)
2.9.3　多重赋值 …… (24)
2.9.4　算术运算符 …… (24)

2.9.5 增量和减量运算符 …… (25)
2.9.6 关系和逻辑运算符 …… (26)
2.9.7 位操作符 …… (28)
2.9.8 ? 操作符 …… (31)
2.9.9 指针操作符 & 和 * …… (31)
2.9.10 编译时操作符 sizeof …… (33)
2.9.11 逗号操作符 …… (33)
2.9.12 点(.)和箭头(->)操作符 …… (34)
2.9.13 方括号[ ]和括号( )操作符 …… (34)
2.9.14 关于优先级的小结 …… (34)
2.10 表达式 …… (35)
2.10.1 求值顺序 …… (35)
2.10.2 表达式中的类型转换 …… (35)
2.10.3 构成符 …… (36)
2.10.4 空格与括号 …… (37)
2.10.5 C 语言中的简写形式 …… (37)
第三章 C 语言的语句 …… (38)
3.1 C 语言的真值和假值 …… (38)
3.2 选择语句 …… (38)
3.2.1 if 语句 …… (39)
3.2.2 嵌套的 if 语句 …… (40)
3.2.3 多重嵌套的 if 语句:阶梯型 if-else-if 语句 …… (41)
3.2.4 操作符? 的替代 …… (42)
3.2.5 条件表达式 …… (45)
3.2.6 switch 开关语句 …… (45)
3.2.7 嵌套的 switch 语句 …… (48)
3.3 迭代语句 …… (48)
3.3.1 for 循环 …… (48)
3.3.2 for 循环的变化形式 …… (49)
3.3.3 无限循环 …… (52)
3.3.4 没有循环体的 for 循环 …… (53)
3.3.5 while 循环 …… (53)
3.3.6 do-while 循环 …… (55)
3.4 转移语句 …… (56)
3.4.1 return 语句 …… (56)
3.4.2 goto 语句 …… (56)
3.4.3 break 语句 …… (57)
3.4.4 exit 语句 …… (58)
3.4.5 continue 语句 …… (59)
3.5 表达式语句 …… (60)
3.6 块语句 …… (61)

第四章　数组和字符串 …… (62)
4.1　一维数组 …… (62)
4.2　产生指向数组的指针 …… (63)
4.3　向函数传递一维数组 …… (64)
4.4　字符串 …… (65)
4.5　二维数组 …… (66)
4.5.1　字符串数组 …… (69)
4.6　多维数组 …… (71)
4.7　带下标的指针 …… (71)
4.8　数组初始化 …… (73)
4.8.1　变长数组的初始化 …… (74)
4.9　棋盘游戏实例 …… (75)
第五章　指针 …… (78)
5.1　指针是什么 …… (78)
5.2　指针变量 …… (78)
5.3　指针运算符 …… (79)
5.4　指针表达式 …… (80)
5.4.1　指针赋值 …… (80)
5.4.2　指针运算 …… (80)
5.4.3　指针比较 …… (81)
5.5　指针和数组是紧密相联的 …… (83)
5.5.1　指针数组 …… (84)
5.6　多级间址 …… (85)
5.7　指针的初始化 …… (86)
5.8　指向函数的指针 …… (87)
5.9　C 语言的动态分配函数 …… (89)
5.10　指针应用中的某些问题 …… (90)
第六章　函数 …… (94)
6.1　函数的一般形式 …… (94)
6.2　函数作用域的规则 …… (95)
6.3　函数变元 …… (95)
6.3.1　传值调用与引用调用 …… (96)
6.3.2　引用调用的建立 …… (96)
6.3.3　用数组调用函数 …… (97)
6.4　传给 main()的变元 argc 与 argv …… (99)
6.5　返回语句 …… (102)
6.5.1　从函数返回 …… (102)
6.5.2　返回值 …… (103)
6.6　返回非整型值的函数 …… (104)
6.7　函数原型 …… (106)
6.8　返回指针 …… (107)

6.9 void 类型的函数 …… (108)
6.10 main()返回什么? …… (109)
6.11 递归 …… (109)
6.12 说明变长参数表 …… (110)
6.13 传统的与现代的函数说明 …… (110)
6.14 实现问题 …… (111)
6.14.1 参数和通用函数 …… (111)
6.14.2 效率 …… (112)
第七章 结构、联合、枚举和用户定义的类型 …… (113)
7.1 结构 …… (113)
7.1.1 引用结构元素 …… (115)
7.1.2 结构赋值 …… (115)
7.2 结构数组 …… (116)
7.3 向函数传递结构 …… (116)
7.3.1 向函数传递结构元素 …… (116)
7.3.2 向函数传递完整结构 …… (117)
7.4 结构指针 …… (118)
7.4.1 结构指针的应用 …… (118)
7.5 结构中的数组和结构 …… (120)
7.6 位域 …… (121)
7.7 联合 …… (122)
7.8 枚举 …… (124)
7.9 用 sizeof 增强移植性 …… (127)
7.10 类型定义 typedef …… (127)
第八章 控制台 I/O …… (129)
8.1 一个重要的应用说明 …… (129)
8.2 读写字符 …… (130)
8.2.1 getchar()的有关问题 …… (130)
8.2.2 getchar()的替代 …… (131)
8.3 读写字符串 …… (131)
8.4 格式化的控制台 I/O …… (134)
8.5 printf() …… (134)
8.5.1 打印字符 …… (135)
8.5.2 打印数字 …… (135)
8.5.3 显示一个地址 …… (136)
8.5.4 %n 描述符 …… (136)
8.5.5 格式化代码修饰符 …… (136)
8.5.6 最小域宽描述符 …… (137)
8.5.7 精度描述符 …… (138)
8.5.8 调整输出 …… (139)
8.5.9 处理其它数据类型 …… (139)

8.5.10 * 和 # 修饰符 …… (139)
8.6 scanf() …… (140)
8.6.1 格式说明符 …… (140)
8.6.2 输入数字 …… (141)
8.6.3 输入无符号整数 …… (141)
8.6.4 用 scanf()读单个字符 …… (141)
8.6.5 读字符串 …… (141)
8.6.6 读入地址 …… (142)
8.6.7 %n 描述符 …… (142)
8.6.8 使用一个扫描集 …… (142)
8.6.9 丢弃不期望的空白符 …… (143)
8.6.10 控制串中的非空白符 …… (143)
8.6.11 必须向 scanf()传递地址 …… (143)
8.6.12 格式化修饰符 …… (144)
8.6.13 压缩输入 …… (144)
第九章 ANSI C 的标准文件 I/O …… (145)
9.1 历史回顾 …… (145)
9.2 流和文件 …… (145)
9.2.1 流 …… (145)
9.2.2 文件 …… (146)
9.3 文件系统基础 …… (146)
9.3.1 文件指针 …… (147)
9.3.2 打开一个文件 …… (147)
9.3.3 关闭一个文件 …… (149)
9.3.4 写一个字符 …… (149)
9.3.5 读一个字符 …… (149)
9.3.6 使用 fopen(),getc(),putc()和 fclose() …… (150)
9.3.7 使用 feof() …… (151)
9.3.8 用 fputs()和 fgets()操作字符串 …… (152)
9.3.9 rewind() …… (153)
9.3.10 ferror() …… (154)
9.3.11 删除文件 …… (156)
9.3.12 清除一个流 …… (157)
9.4 fread()和 fwrite() …… (157)
9.4.1 使用 fread()和 fwrite() …… (157)
9.5 freek()和随机存取 I/O …… (158)
9.6 fprintf()和 fscanf() …… (160)
9.7 标准流 …… (161)
9.7.1 控制台 I/O 连接 …… (161)
9.7.2 使用 freopen()重定向标准流 …… (162)
第十章 C 语言的预处理程序和注释 …… (164)

10.1 C语言的预处理程序 …… (164)
10.2 #define …… (165)
10.3 #error …… (166)
10.4 #include …… (167)
10.5 条件编译指令 …… (167)
10.5.1 #if,#else,#elif,和#endif …… (167)
10.5.2 #ifdef 和#ifndef …… (170)
10.6 #undef …… (171)
10.7 #line …… (171)
10.8 #pragam …… (172)
10.9 #和##预处理操作符 …… (172)
10.10 预定义的宏名 …… (173)
10.11 注释 …… (173)

第二部分 C++ 的专有特征

第十一章 C++概述 …… (176)
11.1 C++的起源 …… (176)
11.2 面向对象的程序设计是什么 …… (177)
11.2.1 对象 …… (177)
11.2.2 多态性 …… (178)
11.2.3 继承性 …… (178)
11.3 C++的程序设计风格 …… (178)
11.4 C++的类 …… (182)
11.5 函数重载 …… (185)
11.6 运算符重载 …… (187)
11.7 继承性 …… (187)
11.8 构造函数和析构函数 …… (192)
11.9 C++的关键字 …… (195)
11.10 C++程序的一般结构 …… (195)
第十二章 类和对象 …… (197)
12.1 类 …… (197)
12.2 结构和类 …… (200)
12.3 联合和类 …… (201)
12.4 友元函数 …… (203)
12.5 内联函数 …… (207)
12.6 在类中定义内联函数 …… (209)
12.7 参数化的构造函数 …… (210)
12.8 静态类成员 …… (212)
12.8.1 静态数据成员 …… (212)
12.8.2 静态成员函数 …… (214)
12.9 何时执行构造函数和析构函数 …… (216)

12.10 嵌套类 …… (217)
12.11 作用域分辨符 …… (217)
12.12 局部类 …… (218)
12.13 向函数传递对象 …… (219)
12.14 返回对象 …… (220)
12.15 对象赋值 …… (221)
第十三章 数组、指针和引用 …… (223)
13.1 对象数组 …… (223)
13.2 指向对象的指针 …… (225)
13.3 this 指针 …… (226)
13.4 指向派生类型的指针 …… (228)
13.5 指向类成员的指针 …… (230)
13.6 引用 …… (232)
13.6.1 引用参数 …… (232)
13.6.2 向对象传递引用 …… (234)
13.6.3 返回引用 …… (235)
13.6.4 独立引用 …… (236)
13.6.5 对引用的限制 …… (237)
13.7 格式问题 …… (237)
13.8 C++的动态分配符 …… (238)
13.8.1 分配对象 …… (240)
第十四章 函数和运算符重载 …… (245)
14.1 函数重载 …… (245)
14.1.1 函数重载和二义性 …… (246)
14.2 重载的过去和现在 …… (249)
14.3 重载构造函数 …… (249)
14.4 求重载函数的地址 …… (251)
14.5 运算符重载 …… (252)
14.5.1 创建成员 operator 函数 …… (252)
14.6 使用 friend 重载运算符 …… (257)
14.6.1 使用 friend 重载++和—— …… (258)
14.6.2 friend operator 函数增加了灵活性 …… (260)
14.7 重载的 new 和 delete …… (262)
14.8 重载某些特殊运算符 …… (265)
14.8.1 重载 [] …… (266)
14.8.2 重载 () …… (268)
14.8.3 重载 -> …… (270)
14.9 重载逗号运算符 …… (270)
第十五章 继承性 …… (273)
15.1 基类存取控制 …… (273)
15.1.1 继承和受保护成员 …… (275)

15.2 继承多个基类 …… (278)
15.3 构造函数、析构函数和继承 …… (279)
15.3.1 何时执行构造函数和析构函数 …… (279)
15.3.2 向基类构造函数传递参数 …… (282)
15.4 授权存取 …… (285)
15.5 虚基类 …… (287)
第十六章 虚函数和多态性 …… (291)
16.1 虚函数 …… (291)
16.1.1 继承虚属性 …… (293)
16.1.2 虚函数的层次性 …… (294)
16.2 纯虚函数 …… (297)
16.2.1 抽象类 …… (298)
16.3 使用虚函数 …… (299)
16.4 早期和后期联编 …… (301)
第十七章 C++的 I/O 系统基础 …… (302)
17.1 C++的流 …… (302)
17.2 基本的流类 …… (302)
17.2.1 C++的预定义流 …… (303)
17.3 格式化的 I/O …… (303)
17.3.1 用 ios 成员格式化 …… (303)
17.3.2 设置格式标志 …… (304)
17.3.3 清除格式标志 …… (305)
17.3.4 setf()的重载形式 …… (305)
17.3.5 检查格式标志 …… (307)
17.3.6 设置所有标志 …… (309)
17.3.7 使用 width()、precision()和 fill() …… (310)
17.3.8 用操纵符格式化 I/O …… (311)
17.4 重载<< 和 >> …… (313)
17.4.1 创建自己的插入符 …… (313)
17.4.2 创建自己的提取符 …… (318)
17.5 创建自己的操纵符函数 …… (320)
17.5.1 创建无参数的操纵符 …… (320)
17.5.2 创建带参数的操纵符 …… (323)
17.6 关于老式流类库的简短说明 …… (326)
第十八章 C++文件 I/O …… (327)
18.1 fstream.h 和文件类 …… (327)
18.2 打开和关闭文件 …… (327)
18.3 读和写文本文件 …… (329)
18.4 二进制 I/O …… (331)
18.4.1 put()和 get() …… (331)
18.4.2 read()和 write() …… (333)

18.5 另外的 get()函数 …… (335)
18.6 getline() …… (335)
18.7 跟踪 EOF …… (336)
18.8 ignore()函数 …… (338)
18.9 peek()和 putback()…… (339)
18.10 flush() …… (339)
18.11 随机存取 …… (339)
18.12 I/O 状态 …… (343)
18.13 定制的 I/O 和文件 …… (344)
第十九章 基于数组的 I/O …… (348)
19.1 基于数组的类…… (348)
19.2 创建基于数组的输出流…… (348)
19.3 使用数组作输入…… (350)
19.4 使用二进制 I/O …… (352)
19.5 基于数组的输人/输出流 …… (352)
19.6 数组内随机存取…… (353)
19.7 使用动态数组…… (353)
19.8 操纵符和基于数组的 I/O …… (354)
19.9 定制的提取符和插人符…… (355)
19.10 基于数组格式的用途 …… (357)
第二十章 问题和高级论题…… (359)
20.1 缺省函数变元…… (359)
20.1.1 正确使用缺省变元 …… (262)
20.2 创建转换函数…… (363)
20.3 拷贝构造函数…… (366)
20.4 动态初始化…… (368)
20.5 常量和可变成员函数…… (368)
20.6 使用关键字 asm …… (369)
20.7 连接说明…… (369)
20.8 以前的重载…… (370)
20.9 C 和 C++之间的区别 …… (370)
20.10 C++的发展方向 …… (371)

第三部分 C++的一些应用

第二十一章 字符串类…… (373)
21.1 定义字符串类型…… (373)
21.2 字符串类…… (374)
21.3 构造函数和析构函数…… (376)
21.4 字符串 I/O …… (377)
21.5 赋值函数…… (378)

21.6 连接……(380)
21.7 子字符串减法……(381)
21.8 关系运算符……(383)
21.9 各种字符串函数……(384)
21.10 完整的字符串类……(385)
21.11 字符串的使用……(393)
第二十二章 弹出式窗口类……(396)
22.1 弹出式窗口……(396)
22.2 创建一些视频支持函数……(397)
22.2.1 PC 视频系统……(397)
22.2.2 存取 BIOS……(398)
22.2.3 确定视频 RAM 的地址……(399)
22.2.4 写视频 RAM……(399)
22.2.5 置光标……(400)
22.3 窗口类……(401)
22.4 显示和移去窗口……(403)
22.5 窗口 I/O……(406)
22.6 一个完整的窗口系统……(410)
22.7 修改……(421)
第二十三章 链表类……(423)
23.1 双向链表类……(423)
23.1.1 store()函数……(425)
23.1.2 remove()函数……(426)
23.1.3 显示表……(427)
23.1.4 改变和查找表中的对象……(428)
23.1.5 完整的 dblink 类和样板程序……(428)
23.2 定义一个样板双向链表基类……(433)
23.2.1 双向链表基类……(434)
23.2.2 定义说明双向链表类……(434)
23.3 其它实现方法……(439)
附录 A 一些公共类……(441)
A.1 复数类……(441)
A.2 BCD 类……(443)

第一部分　C++ 基础:类 C 特性

- C 语言概述
- 表达式
- C 语句
- 数组和串
- 指针
- 函数
- 结构、联合、枚举及用户定义的类型
- 控制台 I/O
- ANSI C 语言标准文件 I/O
- C 语言的预处理程序和注释

本书的第一部分讨论 C++语言(以下简称为 C++)的类 C 特性。读者也许知道,C++是 ANSI 标准 C 语言的增强版本。正是基于这一原因,任何 C++编译器都被定义为一个 C 编译器。由于 C++是建立在 C 语言之上的,所以掌握了 C 语言的编程,便能够用 C++编程。C 语言的许多基本概念也是 C++的语言基础,它们将在这部分中一一介绍。由于 C++是 C 语言的扩展,因而本部分的一切描述也都适于 C++。C++的语言特性将在本书的第二部分详述。C++的类 C 特性专门占用一部分内容,这是为了使具有丰富 C 语言编程经验的读者能够方便、迅速地找到 C++的特征信息,而不必重复涉足他们早已掌握的信息。

注意,本书第一部分摘引了《C 语言大全》的部分内容,如果读者对 C 语言特别感兴趣,那么阅读这本书将会很有帮助。它全面介绍了 ANSI C 标准和标准 C 库函数,此外,还有许多 C 语言的应用实例。

第一章　C语言概述

- C语言的起源
- C语言是中级语言
- C语言是结构化语言
- C语言是程序员的语言
- C语言的程序结构
- 库和连接
- 分离编译

本章的目的是介绍C语言的概貌、起源、应用情况及构成原理。由于C++是建立在C语言之上的，因此本章也介绍了C++产生的历史背景。

1.1　C语言的起源

C语言是由Dennis Ritchie发明并首先在配备UNIX操作系统的DEC PDP-11计算机上实现的。C语言是一种比较古老的语言BCPL发展过程中的产物。BCPL是由Martin Richards开发的，它影响了由Ken Thompson发明的B语言，而B语言又导致了C语言在70年代的发展。

多年来，UNIX操作系统上配备的C语言一直被作为C语言的公认标准（见Brian Kernighan和Dennis Ritchie的《程序设计语言C》，Englewood Cliffs，N. J.：Prentice-Hall，1978）。随着微型机的发展，出现了一大批C语言系统。由于当时不存在统一的标准，因而不同的实现系统存在着差异和不相容。为了改变这种局面，ANSI在1983年初夏成立了一个委员会以制定一劳永逸的C语言标准。目前，ANSI C标准已被采纳，并且大多数C++和C语言编译器都实现了这个标准。

1.2　C语言是中级语言

C语言通常被称为中级计算机语言。这并不意味着它的功能差，难以使用，或者比BASIC、Pascal那样的高级语言原始，也不意味着它象汇编语言那样，会给使用者带来类似的麻烦。C语言之所以被称为中级语言，是因为它把高级语言的成分同汇编语言的功能结合起来了。下表示出了C语言在计算机中所处的地位。

作为中级语言，C支持对位、字节和地址这些有关计算机功能的基本成分进行操作。C语言非常容易移植。可移植性表现为可将某种计算机写的软件改编到另一种机器上实现。举例来说，如果为苹果机写的一个程序能够方便地修改并在IBM PC机上运行，则这个程序称为可移植的。

所有的高级语言都支持数据类型的概念。一个数据类型定义了一个变量的取值范围和可在其上操作的一组运算。常见的数据类型是整型、字符型和实型。虽然 C 语言有五种基本数据类型，但与 Pascal 和 Ada 相比，它算不上强类型语言。C 语言支持几乎所有的类型转换。例如，字符型和整型数据能够自由地混合在大多数表达式中。C 语言不支持诸如数组越界等运行错误检查，检查运行错误是程序员的责任。

高级	Ada
	Modula-2
	Pascal
	COBOL
	FORTRAN
	BASIC
中级	C
	FORTH
	Macro-assembler
低级	Assembler

C 语言的特色是它支持对位、字(字节)和指针的直接操作。这使得它非常适于经常进行上述操作的系统程序设计。C 语言的另一个重要特点是它仅有 32 个关键字(其中 27 个来源于 Kernighan 和 Ritchie 的公认标准，另 5 个是 ANSI 标准化委员会增加的)。这些关键字构成了 C 语言的命令集。和 IBM PC 用的 BASIC 相比，后者包含的关键字多达 159 个。

1.3 C 语言是结构化语言

尽管把 C 语言叫做块结构语言是不严格的，但它还是常被归为结构化语言。因为它与其它结构化语言，如 ALGOL、Pascal 和 Modula-2 有许多相似之处。(从技术上讲，块结构语言允许在过程和函数中定义过程和函数。用这种方法，全局和局部的概念可以通过“作用域”规则加以扩展，“作用域”管理变量的“可见性”。因为 C 语言不允许在函数中定义函数，所以不能称之为通常意义上的块结构语言。)

结构化语言的一个显著特征是程序和数据的分离。这种语言能够把执行某个特殊任务的指令和所有数据信息与程序的其余部分分离开并隐藏起来。获得隔离的一个方法是调用使用局部(临时)变量的子程序。通过使用局部变量，我们能够写出对程序其它部分没有副作用的子程序。这使得编写共享代码段的 C 程序变得十分简单。如果开发了一些分离得很好的函数，在引用时仅需知道函数做什么，而不必知道它如何去做。切记，过度使用全局变量(可以被全部程序访问的变量)会由于意外的副作用而在程序中引入错误。设计过 BASIC 程序的人对这个问题都深有体会。

结构化语言为设计者提供了大量的程序设计功能。它直接支持某些循环结构，如 WHILE、DO-WHILE 和 FOR。在结构化语言中禁止或不提倡使用 GOTO 语句，不能象 BASIC 和 FORTRAN 中那样把它作为常用的程序控制语句。结构化语言允许将语句缩行，但又不必拘于严格的程序格式(象 FORTRAN 语言那种格式)。