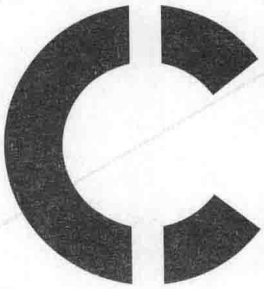
C语言程序设计

本书编写组 编

化学工业出版社



语言程序设计

本书编写组　编

化学工业出版社

·北京·

本书从C语言程序设计的基本原理及程序设计的基本思想出发，将应用的概念和实际操作贯穿于全书的始终，秉承帮助读者不但掌握知识，而且具备应用知识能力的编写理念。

书中的主要内容包括数据类型、运算符、表达式、分支、循环、函数、数组、指针、结构体、文件的概念和应用以及指针和各种构造类型的混合应用等，除了具体教学内容外还引入了一些逻辑推理题作为实际案例，供读者分析讨论使用，大大提高了阅读的兴趣性。

本书既可作为各类高等院校、计算机水平考试培训、成人教育学校作为开设程序设计课程的教材，也可供计算机爱好者自学使用。

图书在版编目（CIP）数据

C语言程序设计 / 本书编写组编. —北京：化学工业出版社，2013.7

ISBN 978-7-122-17390-4

Ⅰ. ①C… Ⅱ. ①本… Ⅲ. ①C语言-程序设计 Ⅳ. ①TP312

中国版本图书馆CIP数据核字（2013）第101205号

责任编辑：宋 薇　　装帧设计：张 辉

责任校对：战河红

出版发行：化学工业出版社（北京市东城区青年湖南街13号 邮政编码100011）

印　　装：化学工业出版社印刷厂

787mm×1092mm 1/16 印张17¾ 字数545千字 2014年3月北京第1版第1次印刷

购书咨询：010-64518888（传真：010-64519686） 售后服务：010-64518899

网　　址：http://www.cip.com.cn

凡购买本书，如有缺损质量问题，本社销售中心负责调换。

定　　价：45.00元

版权所有 违者必究

前言

C 语言是目前国内外广泛使用的程序设计语言之一，也是国内外大学都在开设的重要的基础课之一。C 语言功能丰富、表达力强、使用方便灵活、程序执行效率高、代码可移植性好，既有高级语言的特点，又有汇编语言的特点，有很强的系统处理能力。它支持自顶向下、逐步细化的程序设计技术，函数式结构为实现程序模块化设计提供了强有力的保障，因此被广泛应用于系统软件和应用软件的开发。

学习程序设计语言是为了获得一种编程工具，并用这个工具解决其他课程和工作中遇到的问题，因此本书将“应用”作为贯穿全书的主线，不但给读者介绍 C 语言程序设计的相关“知识”，而且帮助读者将这些知识转换为编程的“能力”。在章节和内容的编排上，本书有如下的特点：

（1）通俗易懂。全书没有拘泥于烦琐的 C 语言的语法规则，不对语法规则做过多的叙述，而是将规则应用于示例中，读者可以通过读例程了解语法规则的规定。

（2）分散难点。指针是 C 语言的最大特色之一，也是读者学习的难点之一。为了让读者能更好地掌握指针的内容，本书将指针部分的内容尽可能提前，这样读者可以早一些接触到指针的概念，延长读者学习指针的时间，可以掌握得更加牢固。

（3）突出重点。函数是学习的重点，本书将函数的相关内容分散成 2 个章节进行讲解，并在介绍完最基本的一些概念后就引入了函数的基本概念，并在后续章节中反复强化这些概念，在读者能充分掌握这些概念的基础上，再用另外一章来讲述函数与程序结构的问题，使读者能了解程序设计总体设计的方法。

（4）紧扣应用。软件开发的工程化方法非常重要，本书不是讲述软件工程的教材，但一方面考虑到部分读者不再继续学习软件工程的课程，另一方面考虑到让读者从最初接触软件开发就有工程化的意识，所以本书虽然没有讲述软件工程的章节，但从开始就贯彻了软件工程的思想。本书在标识符命名中使用了“匈牙利命名法”，虽然这种方法也有其不完美之处，但对培养读者进行规范化的程序设计方面还是利大于弊的。

（5）趣味性强。“兴趣”是最好的老师，本书引入了一些逻辑分析等方面的例程，特别增加了“图形用户界面设计与应用”的相关内容，并分析了“俄罗斯方块”游戏的代码，希望这些内容可以让读者有兴趣去调试更多的代码，能更好地掌握相关知识。

本书第 1 章介绍了程序设计语言的知识及 C 语言的基本情况；第 2 章、第 3 章介绍了 C 语言中数据类型、运算符和表达式等程序设计的基础知识；第 4 章和第 5 章分别讲解了分支结构和循环结构，帮助读者建立了结构化程序设计的概念；第 6 章讲述了函数的基本概念，读者可以了解到模块化程序设计的思路和方法；第 7 章介绍了图形化界面设计的最基础知识，并引入了俄罗斯方块游戏，希望读者能有兴趣调试更多的程序代码；第 8 章介绍了指针的基本概念；第 9 章引入了数组，并将数组与前面讲述过的函数和指针结合起来，强化读者对这些重点和难点内容的理解与掌握；第 10 章介绍了结构体等内容；第 11 章介绍了指针中有一定难度的内容，并引入了链表；第 12 章介绍文件的使用；第 13 章中讨论了函数与程序结构

方面有深度的内容；第 14 章分析了俄罗斯方块游戏的代码。

为了弥补课本内容的局限性，本书还配有计算机教学辅助平台（www.5ic.net.cn），该平台为教师教学和学生学习提供了练习系统和考试系统，同时还发布有教学相关电子资源，以形成对图书的有益补充。需要使用该计算机教学辅助平台的老师和同学可以通过 email 与出版社联系：swx123@cip.com.cn。

本书由郭祥丰、邓玉洁、朱红伟任主编，张倩、曹琨、李芳、吴海燕任副主编，参加编写的还有程杰、杨丽华、李德亮、张勇、李良、刘建军、徐翔、卢丹丹、薛文朝、彭巍、何志标、安静、张政、刘静静、李怡等。

由于作者水平所限，书中若有不妥之处，敬请读者批评指正。

编　者

2014 年 1 月

此为试读，需要完整PDF请访问：www.ertongbook.com

目录

1 引言 1

1.1 欢迎 / 1
1.2 程序设计概述 / 3
1.2.1 指令与程序 / 3
1.2.2 程序设计语言 / 4
1.2.3 程序开发的步骤 / 6
1.3 算法 / 7
1.3.1 算法的概念 / 7
1.3.2 算法的复杂性 / 8
1.3.3 算法的表示方法 / 8
1.4 结构化程序设计 / 10
1.5 C 语言的编译与集成环境 / 11
1.5.1 C 语言程序开发步骤 / 11
1.5.2 集成环境 / 12

2 数据类型 13

2.1 信息存储 / 13
2.1.1 信息编码 / 13
2.1.2 定点数与浮点数 / 14
2.1.3 信息存储 / 14
2.2 标识符 / 15
2.3 基本数据类型 / 17
2.4 常量 / 18
2.4.1 整型常量（整常数） / 18
2.4.2 实型常量 / 18
2.4.3 字符型常量 / 19

2.4.4 字符串常量 / 20
2.4.5 符号常量 / 21
2.5 变量 / 22
2.5.1 变量的定义 / 22
2.5.2 数据的存储 / 23
2.6 数据的输入与输出 / 25
2.6.1 输出字符 putchar() / 25
2.6.2 输入字符 getchar() / 26
2.6.3 格式化输出函数 printf / 26
2.6.4 格式化输入函数 scanf / 30

3
运算符与表达式 33

3.1 数据类型转换 / 33
3.2 运算符 / 34
3.3 表达式 / 34
3.4 算术表达式 / 35
3.4.1 算术运算符 / 35
3.4.2 自增运算符和自减运算符 / 35
3.4.3 算术运算符的优先级与结合性 / 36
3.4.4 算术表达式 / 37
3.5 位运算 / 37
3.6 赋值运算符 / 38
3.6.1 赋值运算符 / 38
3.6.2 赋值表达式 / 38
3.6.3 复合的赋值运算符 / 40
3.7 逗号运算符与逗号表达式 / 41
3.8 深入讨论表达式 / 42

4
分支结构 45

4.1 逻辑运算符和逻辑表达式 / 45
4.1.1 逻辑真与逻辑假 / 46
4.1.2 逻辑运算符 / 46
4.1.3 逻辑表达式 / 48
4.2 关系运算符和关系表达式 / 50
4.3 分支语句 / 52
4.3.1 if 分支结构 / 52

4.3.2 空语句 / 55
4.3.3 复合语句 / 55
4.4 条件运算符 / 59
4.5 多分支 if-else-if / 60
4.6 嵌套的 if 语句 / 63
4.7 switch 和 break 语句 / 69

5
循环 77

5.1 while 语句 / 78
5.2 do-while 语句 / 81
5.3 for 循环语句 / 83
5.4 三种循环语句的比较 / 86
5.5 循环嵌套 / 87
5.6 break 和 continue 语句 / 88
5.7 goto 语句 / 92
5.8 程序实例和分析 / 93

6
函数的使用 97

6.1 编写一个简单的函数 / 97
6.2 调用库函数 / 99
6.2.1 C 语言的标准库函数 / 100
6.2.2 库函数的头文件 / 100
6.2.3 函数类型、函数名与形参 / 100
6.2.4 调用库函数 / 101
6.3 定义用户自定义函数 / 101
6.4 调用自定义函数 / 102
6.5 函数声明 / 104
6.5.1 函数声明 / 104
6.5.2 在函数外部进行函数声明 / 105
6.5.3 省略函数说明 / 106
6.6 函数的参数传递 / 107
6.6.1 参数传递 / 107
6.6.2 实参和形参数据类型不同 / 109
6.7 函数的返回值 / 110
6.7.1 函数返回 / 110
6.7.2 返回值 / 111

6.7.3 void 类型的函数 / 112
6.7.4 函数返回值的数据类型 / 112
6.8 C 语言程序的执行过程 / 113
6.9 函数的嵌套调用 / 114
6.10 局部变量和全局变量 / 115
6.10.1 局部变量 / 115
6.10.2 全局变量 / 115
6.10.3 内部变量和外部变量 / 116
6.11 变量的存储类别 / 117
6.11.1 存储期属性 / 117
6.11.2 存储类别属性 / 118

7
图形化界面 124

7.1 Turbo C 绘图基础 / 126
7.2 设置图形工作环境 / 127
7.3 图形绘制函数 / 128
7.4 图形模式的文本显示 / 129
7.5 俄罗斯方块初步 / 129
7.6 综合应用 / 130

8
指针 132

8.1 变量的指针与指针变量 / 132
8.1.1 变量的指针与取地址运算符 / 132
8.1.2 指针变量 / 133
8.2 用指针访问数据 / 135
8.3 指针变量的运算 / 139
8.3.1 指针的赋值运算 / 139
8.3.2 指针的关系运算 / 141
8.3.3 指针的算术运算 / 141
8.4 指针作为函数的参数 / 143

9
数组 150

9.1 一维数组的定义和引用 / 150

9.1.1 一维数组的定义和引用 / 150
9.1.2 一维数组的存储 / 152
9.1.3 一维数组的初始化 / 153
9.1.4 一维数组程序举例 / 153
9.2 指针与一维数组 / 154
9.2.1 用数组名指针法访问数组元素 / 154
9.2.2 用指针访问数组元素 / 155
9.2.3 数组元素的指针访问法 / 155
9.2.4 数组元素的指针下标访问法 / 158
9.3 数组作为函数的参数 / 159
9.4 二维数组的定义和引用 / 162
9.4.1 二维数组的定义和引用 / 162
9.4.2 二维数组的存储 / 163
9.4.3 二维数组的初始化 / 164
9.4.4 二维数组程序举例 / 164
9.5 字符数组 / 168
9.5.1 字符数组的定义 / 168
9.5.2 字符数组的初始化 / 169
9.5.3 字符数组的输入输出 / 171
9.5.4 字符数组与字符串 / 173
9.6 指针与字符串 / 176
9.6.1 指向字符数组的指针 / 176
9.6.2 指向字符串常量的指针 / 177
9.6.3 字符串作为函数参数 / 179
9.6.4 字符串处理库函数 / 180
9.7 动态内存分配 / 186
9.7.1 void 类型的指针 / 186
9.7.2 指针的强制类型转换 / 186
9.7.3 动态内存分配 / 187
9.8 综合实例 / 189
9.9 掷骰子游戏 / 191
9.10 显示游戏文本信息 / 194

10
结构体和共用体 **195**

10.1 结构体 / 195
10.2 结构体数组 / 199
10.3 结构体指针 / 201
10.4 结构体与函数 / 205

10.5 共用体 / 207
10.6 枚举类型 / 208
10.7 用 typedef 定义类型 / 209

11
深入讨论指针 210

11.1 指针数组 / 210
11.2 指向指针的指针 / 214
11.3 返回指针的函数 / 215
11.4 指向函数的指针 / 218
11.4.1 指向函数的指针 / 218
11.4.2 指向函数的指针作为函数参数 / 220
11.5 链表的概念 / 221
11.6 单向链表的基本操作 / 223

12
文件 231

12.1 文件类型 / 231
12.2 磁盘文件系统 / 232
12.3 文件类型指针 / 232
12.4 文件打开与关闭 / 233
12.5 文件读写 / 235

13
深入讨论函数与程序结构 240

13.1 函数的递归调用 / 240
13.1.1 编写递归函数求 n! / 240
13.1.2 迭代和递归 / 244
13.2 带参的 main 函数 / 245
13.3 内部函数和外部函数 / 246
13.3.1 内部函数 / 246
13.3.2 外部函数 / 246
13.4 编译预处理 / 247
13.4.1 宏定义 / 247
13.4.2 文件包含 / 249

此为试读，需要完整PDF请访问：www.ertongbook.com

13.4.3 条件编译 / 251

14
综合应用设计 254

14.1 实例说明 / 254
14.2 俄罗斯方块程序源代码 / 255
14.3 新出现的函数 / 267
14.3.1 捕捉按键 / 267
14.3.2 memset 函数 / 267
14.3.3 memcpy 函数 / 267
14.4 基本位置参数 / 268
14.4.1 面板区 / 268
14.4.2 下一个提示区 / 268
14.4.3 信息显示区域 / 269
14.5 画面内容的刷新 / 269
14.5.1 将方块嵌入面板 / 269
14.5.2 删除满行 / 269
14.6 方块的运动 / 270
14.6.1 定时下落 / 270
14.6.2 判断翻转 / 270
14.6.3 翻转 / 271
14.6.4 判断移动 / 271
14.6.5 移动 / 271
14.6.6 下落 / 271
14.6.7 直落到底 / 271

参考文献 272

1 引　言

本章要点

- 什么是程序？程序设计语言包括哪些功能？
- 结构化程序设计有什么要求？
- 什么是算法，如何表达算法？
- C 语言有哪些特点？
- C 语言程序的编译步骤包括哪些阶段？

本章主要介绍 C 语言编程的基础知识，简单易懂，但在编程使用的过程中常会出错，因此，牢记并理解基础概念对于编程是大有裨益的。

1.1 欢　迎

【例 1-1】 在屏幕上输出“Hello，world!”。

```
main()
{
    printf("Hello,world!");
}
运行后程序输出：
Hello,world!
```

运行【例 1-1】在计算机屏幕上输出了“Hello，world!”，同时也欢迎读者来到 C 语言的世界。

对于将要开始学习 C 语言程序设计的读者来说，希望能快速掌握 C 语言程序设计的方法，并能尽早开始学会用 C 语言设计程序。计算机的编程设计，是人类利用和开发计算机各种功能最深入、最直接的方法。学会计算机编程，意味着真正走进了计算机的世界，而 C 语言本身就是与计算机进行交互的有利工具。计算机编程即程序设计，是伴随着计算机应用和程序设计语言的发展而发展起来的一门学科，是使用和开发计算机的重要工具。在程序设计中，程序员需要了解各种开发语言和开发平台的优缺点，并且懂得如何根据问题的大小和难易程度选择最合适的开发工具。让我们再看一个稍微复杂些的例子吧。

【例 1-2】 编写函数求两个整数中大的数。

```
#include <stdio.h>

/*自定义函数*/
int Max(int nNum1, int nNum2)
{
     if (nNum1 > nNum2) {                    /*判断两个数的大小*/
          return nNum1;                      /*第一个数大*/
     }
     return nNum2;                           /*第二个数大*/
}
main()
{
     int nNum1, nNum2, nResult=0;            /*声明变量*/
     printf ("Please Input First Num:");     /*显示提示信息*/
     scanf("%d", &nNum1);                    /*读取一个整数值*/
     printf ("Please Input Second Num:");    /*显示提示信息*/
     scanf("%d", &nNum2);                    /*读取一个整数值*/
     nResult = Max(nNum1,nNum2);             /*比较大小*/
     printf("Max Is: %d \n", nResult);       /*打印结果*/
}
运行后程序输出:
Please Input First Num:3
Please Input Second Num:5
Max Is:5
```

【例 1-2】不但程序的代码长度远远超过了【例 1-1】，而且【例 1-2】中涉及了 C 语言的很多基本概念和规则。

（1）C 语言的基本组成。

① 标识符。在 C 语言中，最主要的标识符是保留字和用户自定义标识符。保留字时 C 语言有专门用途的标识符，例如：int。用户自定义标识符包括程序中定义的变量名、函数名等。例如：nNum1, nNum2, nResult 和 Max。

② 常量。C 语言里的常量就是在程序运行过程中不会改变的数据。例如【例 1-2】中出现的数值 0 和【例 1-1】中出现的“Hello，world!”。

③ 运算符。程序的重要功能是对数据的加工和处理，运算符表示对数据进行的运算。例如：关系运算符“>”。

④ 分隔符。在程序中分割不同部分的符号。例如变量定义语句中不同变量之间的逗号。

（2）函数。在 C 语言程序中基本的结构是函数，函数组合在一起构成 C 语言源程序。在【例 1-2】中有 4 个函数，分别是 main()、Max()、printf()和 scanf()。其中 printf()和 scanf()是 C 语言的开发环境中已经设计好的“库函数”，分别用于数据输入和输出，我们直接使用就可以了。Max ()函数是我们自己编写的函数，完成计算两个数的和的功能。

在所有的 C 语言程序中必须且只能有一个名为 main()的函数。main 是每一个程序都必须具有的，它是由系统定义的。无论 main()函数在整个程序中位置如何，每一个 C 语言程序都是从函数 main 开始的，也结束于 main 函数最后一个花括号。

（3）语句。函数内部的每一行都是一条语句，语句由单词按照一定的语法规则构成。C 语

言中有多种类型的语句，由这些语句构成函数，再由函数构成程序。C 语言的所有语句都是以分号“;”结尾的。通常见到的语句包括：

① 变量定义。定义程序中需要用到的变量。例如：int nNum1, nNum2, nResult=0。

② 表达式。运算符与运算对象有意义组合起来就是表达式。如：nNum1 > nNum2 和 nResult = Max(nNum1,nNum2)。

③ 结构。C 语言支持的结构包括：顺序结构、分支结构和循环结构。在【例 1-2】中有一个 if 分支结构。

④ 函数调用。无论是调用库函数，还是调用用户自定义函数，都是以一条语句的方式。

⑤ 复合语句。用一对花括号将若干条语句组合在一起，就是一条复合语句。

（4）书写格式。C 语言程序书写格式自由，但为了增加程序的可读性，在编写 C 语言源程序的时候通常要注意如下几个问题：

① 虽然 C 语言的语法规则上允许一行写多条语句，但通常要求一行只写一条语句，以方便程序调试。

② 花括号要注意对齐，花括号内的语句要注意缩进；

③ C 语言习惯上使用小写字母，一个完全用大写字母书写的标识符通常有特定的含义；

当然，我们在这里只是写出了很少的几条规则，真正的软件开发项目会对此有更详细的规定。

（5）注释。注释是程序中的可执行语句，对于程序的运行没有任何意义。将在程序的编译阶段被编译器剔除，但注释是程序不可缺少的组成部分。一个具有良好书写风格的程序，其注释应该是充分的。注释有助于阅读和理解程序，这对于复杂程序尤为重要，即使对编程者而言，注释也是十分重要的。不同语言的注释符不同。C 语言中提供多行注释（或块注释），即程序中位于符号/*开始和符号*/之间的字符为注释内容。编程时在程序中添加适当的注释是一个良好的编程习惯：在每个规模较大或所处地位较重要的函数外头最好添加关于该函数的注释，包括函数所采用的主要算法、参数含义、所用到的全局变量、编写的时间等。这些注释有助于阅读和理解该函数，对程序维护和重用非常必要。

1.2 程序设计概述

1.2.1 指令与程序

计算机的指令是指挥计算机进行基本操作的命令，是计算机能够识别的一组二进制编码，由操作码（指出应该进行什么样的操作）和操作数（参与操作的数本身或它在内存中的地址）组成。操作码指明该指令要完成的操作，如加、减、乘、除等。操作数是指参加操作的数或者数所在的单元地址。一台计算机所有指令的集合，称为该计算机的指令系统。指令系统反映了计算机的基本功能，不同的计算机其指令系统也不相同。

计算机执行指令一般分为两个过程，即取指令过程和执行指令过程。取指令过程是将要执行的指令从内存中取到 CPU 内。执行指令过程是 CPU 对取入的指令进行分析译码，判断该条指令要完成的操作，然后向其他部件发出完成该操作的控制信号，最后完成该条指令的功能。当一条指令执行完后，自动地进入下一条指令的取指操作。

程序（program）是为实现特定目标或解决特定问题而用计算机语言编写的命令序列的

集合。为实现预期目的而进行操作的一系列语句和指令，一般分为系统程序和应用程序两大类。程序就是为使计算机执行一个或多个操作，或执行某一任务，所设计的计算机指令的集合。

1.2.2 程序设计语言

程序设计（Programming）是给出解决特定问题程序的过程，是软件构造活动中的重要组成部分。程序设计往往以某种程序设计语言为工具，给出这种语言下的程序。程序设计过程应当包括分析、设计、编码、测试、排错等不同阶段。

（1）程序设计语言。程序设计语言（programming language）是用于书写计算机程序的语言。语言的基础是一组记号和一组规则。根据规则，由记号构成的记号串的总体就是语言。在程序设计语言中，这些记号串就是程序。程序设计语言有 3 个方面的因素，即语法、语义和语用。语法表示程序的结构或形式，也就是表示构成语言的各个记号之间的组合规律。语义表示程序的含义，就是表示按照各种方法所表示的各个记号的特定含义。语用表示程序与使用者的关系。程序设计语言的基本成分有：

① 数据成分，用于描述程序所涉及的数据；

② 运算成分，用以描述程序中所包含的运算；

③ 控制成分，用以描述程序中所包含的控制；

④ 传输成分，用以表达程序中数据的传输。

（2）程序设计语言的种类。程序设计语言按照语言级别可以分为低级语言和高级语言。低级语言有机器语言和汇编语言。低级语言与特定的机器有关，功效高，但使用复杂、繁琐、费时、易出差错。机器语言是表示成数码形式的机器基本指令集，或者是操作码经过符号化的基本指令集。汇编语言是机器语言中地址部分符号化的结果，或进一步包括宏构造。高级语言的表示方法要比低级语言更接近于待解问题的表示方法，其特点是在一定程度上与具体机器无关，易学、易用、易维护。很多时候 C 语言被认为是一种中级语言。它把高级语言的基本结构和语句与低级语言的实用性结合起来。用 C 语言编写的程序保持了高级语言的特征，但同时 C 语言可以像汇编语言一样对位、字节和地址进行操作，而这三者是计算机最基本的工作单元。程序设计语言按照用户的要求有过程式语言和非过程式语言之分。过程式语言的主要特征是，用户可以指明一列可顺序执行的运算，以表示相应的计算过程，如 FORTRAN、COBOL、PASCAL 等。

- 按照应用范围，有通用语言与专用语言之分。如 FORTRAN、COBOL、PASCAL、C 等都是通用语言。目标单一的语言称为专用语言，如 APT 等。
- 按照使用方式，有交互式语言和非交互式语言之分。具有反映人机交互作用的语言成分的语言称为交互式语言，如 BASIC 等。不反映人机交互作用的语言称为非交互式语言，如 FORTRAN、COBOL、ALGOL69、PASCAL、C 等。
- 按照成分性质，有顺序语言、并发语言和分布语言之分。只含顺序成分的语言称为顺序语言，如 FORTRAN、C 等。含有并发成分的语言称为并发语言，如 PASCAL、Modula 和 Ada 等。

（3）C 语言的发展历史。早期的操作系统等系统软件主要是用汇编语言编写的，如 UNIX 操作系统。由于汇编语言依赖于计算机硬件，程序的可读性和可移植性都比较差。为了提高可读性和可移植性，最好改用高级语言，但一般高级语言难以实现汇编语言的某些功能，而汇编语言可以直接对硬件进行操作，例如，对内存地址的操作、位（bit）操作等。人们设想能否找到一种既具有一般高级语言特性，又具有低级语言特性的语言，集它们的优点于一身。于是，C 语言就在这种情况下应运而生了，之后成为国际上广泛流

行的计算机高级语言。它适合于作为系统描述语言，既用来写系统软件，也可用来写应用软件。

C 语言是在 B 语言的基础上发展起来的，它的根源可以追溯到 ALGOL 60。1960 年出现的 ALGOL 60 是一种面向问题的高级语言，它离硬件比较远，不宜用来编写系统程序，1963 年英国的剑桥大学推出了 CPL（Combined Programming Language）语言。CPL 语言在 ALGOL 60 的基础上接近硬件一些，但规模比较大，难以实现。1967 年英国剑桥大学的 Matin Richards 对 CPL 语言作了简化，推出了 BCPL（Basic Combined Programming Language）语言。1970 年美国贝尔实验室的 Ken Thompson 以 BCPL 语言为基础，又作了进一步简化，它使得 BCPL 能挤压在 8K 内存中运行，这个很简单的而且很接近硬件的语言就是 B 语言（取 BCPL 的第一个字母），并用它写了第一个 UNIX 操作系统，在 DEC PDP-7 上实现。1971 年在 PDP-11/20 上实现了 B 语言，并写了 UNIX 操作系统。但 B 语言过于简单，功能有限，并且和 BCPL 都是“无类型”的语言。

1972 年至 1973 年间，贝尔实验室的 D.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，数据无类型等）。最初的 C 语言只是为描述和实现 UNIX 操作系统提供一种工具语言而设计的。1973 年，K.Thompson 和 D.M.Ritchie 两人合作把 UNIX 的 90%以上用 C 改写，即 UNIX 第 5 版。原来的 UNIX 操作系统是 1969 年由美国贝尔实验室的 K.Thompson 和 D.M.Ritchie 开发成功的，是用汇编语言写的，这样，UNIX 使分散的计算系统之间的大规模联网以及互联网成为可能。

后来，C 语言多次进行改进，但主要还是在贝尔实验室内部使用。直到 1975 年 UNIX 第 6 版公布后，C 语言的突出优点才引起人们普遍注意。1977 年出现了不依赖于具体机器的 C 语言编译文本《可移植 C 语言编译程序》，使 C 移植到其他机器时所需做的工作大大简化了，这也推动了 UNIX 操作系统迅速在各种机器上实现。例如，VAX，AT&T 等计算机系统都相继开发了 UNIX。随着 UNIX 的日益广泛使用，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、微型机上，如 IBM System/370、Honeywell 6000 和 Interdata 8/32，已独立于 UNIX 和 PDP 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。

以 1978 年由美国电话电报公司（AT&T）贝尔实验室正式发表的 UNIX 第 7 版中的 C 编译程序为基础，Brian W.Kernighan（柯尼汉）和 Dennis M.Ritchie（里奇）合著了影响深远的名著《The C Programming Language》，常常称它为“K&R”，也有人称之为“K&R 标准”或“白皮书”（white book），它成为后来广泛使用的 C 语言版本的基础，但在“K&R”中并没有定义一个完整的标准 C 语言。为此，1983 年，美国国家标准化协会（ANSI）X3J11 委员会根据 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 比原来的标准 C 有了很大的发展：K&R 在 1988 年修改了他们的经典著作《The C Programming Language》，按照 ANSI C 标准重新写了该书。1987 年，ANSI 又公布了新标准——87 ANSI C。目前流行的 C 编译系统都是以它为基础的。当时广泛流行的各种版本 C 语言编译系统虽然基本部分是相同的，但也有一些不同。在微型机上使用的有 Microsoft C(MS C)，Borland Turbo C，Quick C 和 AT&T C 等，它们的不同版本又略有差异。到后来的 Java、C++、C#都是以 C 语言为基础发展起来的。

（4）C 语言的特点。无论是哪个版本的 C 语言，通常都具有以下的共同特点。

① C 语言是一种结构化语言。结构式语言的显著特点是代码及数据的分隔化，即程序的各个部分除了必要的信息交流外彼此独立。这种结构化方式可使程序层次清晰，便于使用、

此为试读，需要完整PDF请访问：www.ertongbook.com