普通高等教育“十一五”国家级规划教材 计算机系列教材

C语言
程序设计基础

陈东方 黄远林 李顺新
李文杰 王晓峰 编著

清华大学出版社

普通高等教育“十一五”国家级规划教材　计算机系列教材

陈东方　黄远林　李顺新　李文杰　王晓峰　编著

C语言程序设计基础

清华大学出版社
北京

内 容 简 介

本书以标准 C 为框架，以 Visual C++ 6.0 为编程环境，按照紧扣基础和面向应用的原则，介绍了 C 语言程序设计的基本规范、思路和方法。本书从培养学生的实际编程能力出发，注重实例教学和实践练习，突出重点讲解和难点分析，图文并重，文字流畅。

本书概念清楚、内容全面、题例和习题丰富，书中所有示例程序均给出了算法思路的分析和算法步骤，并上机调试运行后给出了结果，每个程序都遵循标准化的编程风格，便于学生理解和自学。

本书适合作为高等院校各类专业“C 语言程序设计”课程的教材，亦适合初学者自学或供广大程序设计及开发人员参考。

图书在版编目（CIP）数据

C 语言程序设计基础/陈东方等编著．—北京：清华大学出版社，2010.3
ISBN 978-7-302-21642-1

Ⅰ．①C…　Ⅱ．①陈…　Ⅲ．①C 语言－程序设计　Ⅳ．①TP312

中国版本图书馆 CIP 数据核字(2010)第 016290 号

责任编辑：魏江江　薛　阳
责任校对：李建庄
责任印制：李红英

出版发行：清华大学出版社
http://www.tup.com.cn
社　　总　　机：010-62770175
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 刷 者：清华大学印刷厂
装 订 者：三河市新茂装订有限公司
经　　销：全国新华书店
开　　本：185×260　　**印　张**：20　　**字　数**：496 千字
版　　次：2010 年 3 月第 1 版　　**印　次**：2010 年 3 月第 1 次印刷
印　　数：1～4000
定　　价：29.50 元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题，请与清华大学出版社出版部联系调换。
联系电话：010-62770177 转 3103　　产品编号：034938-01

在众多的程序设计语言中，C 语言一直是广大程序设计人员和编程爱好者学习软件编程的首选语言。C 语言简洁紧凑，语言表达能力强，其结构化的流程控制有助于编制结构良好的程序。C 语言程序经编译后生成的目标程序代码效率高，几乎可以与汇编语言媲美。C 语言既具备高级语言使用方便、接近自然语言和数学语言的特性，同时也具备对计算机硬件系统的良好操纵和控制能力。C 语言可移植性好，一个 C 语言源程序可以不做改动，或者稍加改动，就可以从一种型号的计算机移转到另外一种型号的计算机上编译运行。因此，C 语言被广泛应用于各类系统软件和应用软件的开发。

但 C 语言语法规则较多，要灵活使用，对初学者来说有一定难度。本书编著者根据多年从事 C 语言教学的经验，按照紧扣基础和面向应用的撰写原则，以标准 C 为框架，以 Visual C++ 6.0 为编程环境，介绍了 C 语言程序设计的基本规范、思路和方法，力图做到概念解释通俗化、文字介绍简洁化、算法描述多样化、例程解释详尽化和复杂问题图表化，使之适合初学者学习和掌握。

本书的特点是：循序渐进，由浅入深，通俗易懂，实践性强。本书概念清楚、内容全面、题例和习题丰富，书中所有示例程序均给出了算法思路的分析和算法步骤，并上机调试运行后给出了结果，每个程序都遵循标准化的编程风格，便于学生理解和自学，同时兼顾等级考试。

全书共分为 11 章，其内容包括：第 1 章介绍程序设计和算法的基本概念，同时对 C 语言及其编程环境简单介绍；第 2 章介绍数据类型及其运算；第 3～5 章介绍了顺序、选择和循环结构程序设计；第 6～8 章分别介绍了数组、函数和预处理命令；第 9～11 章分别介绍了指针、结构体与共用体、文件。

本书是编著者在总结多年从事 C 语言教学和编程实践的基础上，参考了国内外相关资料和书籍编著而成，适用于计算机专业的本科生、研究生、大专生、专升本的学生使用，也可以作为各类非计算机专业计算机公共课教材和计算机等级考试参考书，也可供编程人员、自学人员自学参考。

本书由武汉科技大学陈东方副教授、黄远林副教授主编，具体撰写人员是陈东方、李顺新、李文杰、黄远林和王晓峰，陈东方负责全书策划、总纂与定稿工作。本书在编著过程中得到了武汉科技大学计算机学院众多教师的大力支持和帮助，在此一并致谢。

由于编者水平有限，书中不妥之处在所难免，恳请各位同行和读者赐教。

编 者

2010 年 1 月于武汉

第 1 章　概述　/1
1.1　程序设计及程序设计语言　/1
1.1.1　低级语言　/1
1.1.2　高级语言　/1
1.2　算法及其表示方法　/2
1.2.1　算法的基本概念及特性　/2
1.2.2　算法的表示方法　/3
1.3　C 语言简介　/5
1.3.1　C 语言的产生与发展　/5
1.3.2　C 语言的特点　/6
1.3.3　C 语言程序的基本结构　/7
1.3.4　C 语言的基本语法单位　/10
1.3.5　运行 C 语言程序的一般步骤　/12
1.4　Visual C++ 6.0 集成开发环境简介　/13
习题 1　/16

第 2 章　数据类型、运算符与表达式　/18
2.1　C 语言的数据类型　/18
2.2　常量　/20
2.2.1　直接常量　/20
2.2.2　符号常量　/25
2.3　变量　/26
2.3.1　变量名与变量值　/26
2.3.2　变量的定义　/26
2.3.3　变量初始化　/27
2.4　运算符与表达式　/27
2.4.1　C 语言的运算符简介　/27
2.4.2　算术运算　/28
2.4.3　关系运算　/31
2.4.4　逻辑运算　/32
2.4.5　赋值运算　/36
2.4.6　逗号运算　/38
2.4.7　位运算　/39

2.4.8 数据之间的混合运算 /41
习题 2 /44

第 3 章 输入输出与简单程序设计 /47
3.1 概述 /47
3.2 流程控制结构与语句 /47
3.3 基本的标准输入输出函数 /49
3.4 单个字符的输入和输出 /50
3.4.1 字符输入 /50
3.4.2 字符输出 /51
3.5 格式化输出 /53
3.5.1 整数的输出 /55
3.5.2 实数的输出 /56
3.5.3 单个字符的输出 /57
3.5.4 字符串的输出 /57
3.5.5 混合数据的输出 /58
3.5.6 使用 printf 函数时的注意事项 /58
3.6 格式化输入 /59
3.6.1 整数的输入 /60
3.6.2 实数的输入 /61
3.6.3 字符串的输入 /62
3.6.4 混合数据类型的输入 /63
3.6.5 使用 scanf 函数时的注意事项 /63
3.7 简单程序设计 /65
习题 3 /68

第 4 章 选择结构程序设计 /69
4.1 if 语句 /69
4.1.1 if 语句的 3 种形式 /69
4.1.2 if 语句的嵌套 /72
4.1.3 条件表达式 /74
4.2 switch 语句 /74
习题 4 /76

第 5 章　循环结构程序设计　/78

5.1　while 语句　/78

5.2　do…while 语句　/80

5.3　for 循环　/82

5.4　用 goto 语句和 if 语句构成循环　/85

5.4.1　goto 语句　/85

5.4.2　带标号语句　/85

5.5　循环的嵌套　/86

5.6　循环语句小结　/87

5.7　break 语句和 continue 语句　/88

5.7.1　break 语句　/88

5.7.2　continue 语句　/88

习题 5　/90

第 6 章　数组　/91

6.1　一维数组的定义和引用　/91

6.1.1　一维数组的定义　/91

6.1.2　一维数组元素的引用　/92

6.1.3　一维数组的初始化　/93

6.1.4　一维数组程序举例　/93

6.2　二维数组的定义和引用　/95

6.2.1　二维数组的定义　/95

6.2.2　二维数组元素的引用　/96

6.2.3　二维数组的初始化　/96

6.2.4　二维数组程序举例　/97

6.3　字符数组　/99

6.3.1　字符串常量　/99

6.3.2　字符数组的定义　/99

6.3.3　字符数组的引用　/100

6.3.4　字符数组的初始化　/100

6.3.5　字符串处理函数　/101

6.3.6　字符数组程序举例　/103

习题 6　/105

第 7 章 函数 /107
7.1 模块化程序设计与函数 /107
7.1.1 模块化程序设计 /107
7.1.2 函数的概述 /108
7.2 函数的定义 /109
7.2.1 返回确定值的函数定义 /109
7.2.2 不返回结果的函数定义 /111
7.3 函数的参数和函数的值 /112
7.3.1 形式参数和实际参数 /112
7.3.2 函数的返回值 /115
7.4 函数的调用 /119
7.4.1 函数调用的一般形式 /119
7.4.2 函数调用的方式 /120
7.4.3 被调用函数的声明和函数原型 /122
7.5 函数的嵌套调用 /125
7.6 函数的递归调用 /126
7.7 数组作为函数参数 /132
7.7.1 数组元素作函数实参 /132
7.7.2 数组名作为函数参数 /134
7.8 局部变量和全局变量 /143
7.8.1 局部变量 /143
7.8.2 全局变量 /145
7.9 变量的存储类别 /147
7.9.1 动态存储方式与静态存储方式 /147
7.9.2 auto 变量 /148
7.9.3 用 static 声明局部变量 /148
7.9.4 register 变量 /150
7.9.5 用 extern 声明外部变量 /150
7.9.6 内部函数和外部函数 /152
习题 7 /153

第 8 章　编译预处理 /158
8.1　宏定义 /158
8.1.1　无参宏定义 /158
8.1.2　带参宏定义 /161
8.2　条件编译 /165
8.3　文件包含 /167
习题 8 /169

第 9 章　指针 /172
9.1　地址和指针的基本概念 /172
9.2　变量的指针和指向变量的指针变量 /174
9.2.1　指针变量的定义 /174
9.2.2　指针变量的类型 /175
9.2.3　指针变量的初始化 /176
9.2.4　指针变量的引用 /177
9.2.5　指针变量的运算 /180
9.2.6　指针变量作为函数参数 /182
9.3　数组指针和指向数组的指针变量 /185
9.3.1　一维数组的指针 /185
9.3.2　通过指针访问一维数组 /187
9.3.3　通过指针在函数间传递一维数组 /190
9.3.4　指向二维数组的指针变量 /198
9.4　指针与字符串 /203
9.4.1　字符串与指向字符串的指针 /203
9.4.2　使用字符串指针变量与
字符数组的区别 /204
9.5　函数指针变量 /208
9.6　指针型函数 /212
9.7　指针数组和指向指针的指针 /214
9.7.1　指针数组的概念 /214
9.7.2　指向指针的指针 /219

9.7.3 main 函数的参数 /221
9.8 有关指针的数据类型和指针运算的小结 /222
习题 9 /224

第 10 章 结构与联合 /226
10.1 概述 /226
10.2 结构类型的说明与引用 /227
10.2.1 结构类型的声明 /227
10.2.2 声明结构类型变量的方法 /229
10.2.3 结构变量的初始化 /230
10.2.4 访问结构的成员 /231
10.2.5 结构的嵌套 /235
10.2.6 结构中的数组 /236
10.3 结构数组 /238
10.3.1 结构数组的声明 /239
10.3.2 结构数组的初始化 /240
10.3.3 结构数组元素的引用 /241
10.4 指向结构类型数据的指针 /243
10.4.1 指向结构类型变量的指针 /243
10.4.2 指向结构数组的指针 /245
10.5 结构与函数 /248
10.5.1 结构成员作为函数的参数 /248
10.5.2 结构作为函数的参数 /249
10.5.3 将指向结构的指针作为函数的参数 /250
10.5.4 结构和结构指针作为函数的返回值 /251
10.6 单链表 /253
10.6.1 静态数据结构和动态数据结构 /253
10.6.2 C 语言的动态存储分配函数 /255
10.6.3 单链表 /257

10.7 联合 /265
10.7.1 联合的声明 /265
10.7.2 联合变量的说明 /266
10.7.3 联合变量的引用 /267
10.7.4 联合与结构的区别与联系 /269
习题 10 /271

第 11 章 文件 /273
11.1 文件概述 /273
11.2 文件的分类 /274
11.3 文件类型指针 /275
11.4 文件的打开与关闭 /276
11.4.1 标准文件 /276
11.4.2 文件的打开与关闭函数 /276
11.5 文本文件的顺序读写 /280
11.5.1 字符读取函数 /281
11.5.2 写字符函数 /282
11.5.3 字符串读取函数 /283
11.5.4 写字符串函数 /283
11.5.5 格式化读写函数 /284
11.6 数据块读写函数 /287
11.6.1 读数据块函数 /287
11.6.2 写数据块函数 /288
11.6.3 使用数据块读写函数的注意事项 /288
11.7 文件的随机读写 /290
11.7.1 文件头定位函数 /290
11.7.2 文件随机定位函数 /291
11.7.3 文件当前位置函数 /293
11.8 其他函数 /294
习题 11 /294

附录 /297

附录 A ASCII 字符编码一览表 /297

附录 B C 语言运算符 /298

附录 C C 语言中的关键字 /299

附录 D 常用标准库函数 /300

参考文献 /306

第1章 概 述

本章主要介绍程序设计的基本内容，包括程序设计语言、算法的基本概念及其表示方法、C语言程序的基本结构以及C语言程序的开发环境等内容。

本章重点：

理解算法和程序的基本概念，掌握C程序的基本结构。

本章难点：

算法流程图的绘制和C程序的上机操作。

1.1 程序设计及程序设计语言

程序是用程序设计语言描述的某一问题的解决步骤。程序设计是将解题任务转变成程序的过程，一般包括：分析问题、确定算法（对复杂算法需要画出程序流程图）、用选定的程序设计语言编写源程序、上机调试和运行程序等基本步骤。

程序设计语言是计算机能够理解的、用于人和计算机交流的语言，程序设计语言可分为低级语言和高级语言。

1.1.1 低级语言

低级语言分为机器语言和汇编语言。

机器语言用二进制代码表示机器指令和数据。机器语言程序能够直接被机器理解和执行。虽然这种语言程序效率高，但编程繁琐，且不便于记忆和阅读，因而程序维护困难。

汇编语言采用符号来表示机器指令和数据的内存地址，是一种符号化的低级语言。用汇编语言编写的程序称为汇编语言源程序。汇编语言程序必须被转换为机器语言程序才能被计算机理解和执行，完成这种转换任务的系统软件称为汇编程序，该转换过程称为汇编。

低级语言是面向机器的。用低级语言编写的程序效率高，但没有可移植性，即不能从一个机器系统移到另一个机器系统上运行。此外，用低级语言编写源程序要求程序员必须懂得具体机器系统的硬件结构（指令系统）。

1.1.2 高级语言

高级语言起始于20世纪50年代中期，是一种能够被计算机接收同时更接近自然语言的程序设计语言。和汇编语言相比，高级语言去掉了与机器有关但与完成工作无关的细节，大大简化了程序中的指令。同时，由于省略了很多细节，编程者不需要有太多的专业知识。因此，使用高级语言编写的程序可读性强，编程方便。

高级语言所编写的程序叫做源程序。源程序不能直接被计算机识别和理解，必须经过

转换才能被计算机执行。其转换方式可分为解释方式和编译方式两种。

1. 解释方式

解释方式类似于日常生活中的“同声翻译”，即一边将源代码由相应语言的解释器“翻译”成目标代码，一边执行，因此效率比较低，而且不能生成可执行文件。但这种翻译方式比较灵活，可以及时发现错误，动态地调整、修改应用程序。BASIC、Prolog 等语言就是采用解释方式。

2. 编译方式

编译方式是指在源程序被执行之前，就将源程序源代码“翻译”成目标代码(机器语言)，因此其目标程序可以脱离其语言环境独立执行，使用比较方便，效率较高。但源程序一旦需要修改，必须先修改源代码，再重新编译生成新的目标文件才能执行。目前大多数高级语言均采用编译方式，如 Pascal、FORTRAN、C 语言等。

目前高级语言的种类有近千种之多。常用的有 BASIC、FORTRAN、Pascal、C/C++ 和 Java 等。

1.2 算法及其表示方法

1.2.1 算法的基本概念及特性

计算机解决问题的方法和步骤，就是计算机的算法。每一个算法都是由一系列的操作指令组成的，研究算法的目的就是研究怎样把各种类型的问题的求解过程分解成一些基本的操作。

为便于计算机执行算法而用程序设计语言或半形式语言所表示的算法被称为程序。所有算法都能被编制成程序，但由程序设计语言书写的程序并不一定都满足算法的特性。一个算法应具有以下重要特性。

(1) 有穷性：一个算法应该包含有限个的操作步骤，而不是无限的，也就是说一个算法的实现应该在有限的时间之内完成。

(2) 确定性：算法中的每个步骤都是确定的、含义清楚的，而不是模棱两可的，算法每个步骤的动作是唯一的。

(3) 有效性：或者说可行性，算法中的每一个步骤都应当能够有效地被执行，并得到确定的结果。即在计算机能力范围内，并且在有限时间内能正确地执行每一个步骤。

(4) 有 0 个或者多个输入：所谓输入是指在执行算法时需要从外界取得的必要的信息。

(5) 有一个或者多个输出：算法的目的是为了解决问题，“解决了的问题”就是输出。输出是算法执行的结果。

【例 1-1】 统计某班单科成绩不及格的人数。

设用 n 表示某班的人数，x 表示每个人的单科成绩，k 表示成绩不及格的人数，求解该问题的算法描述为：

(1) 输入某班人数 n 的值,并置 k 为 0。

(2) 按顺序输入每个学生的单科成绩并对每个学生的单科成绩 x 进行检查,如果 x 是小于 60 就按成绩不及格进行计数(即使 k 的值加 1)。

(3) 当输入单科成绩的学生人数为 n 时,成绩输入完毕,输出统计成绩不及格的人数 k 的值。

(4) 停止。

1.2.2 算法的表示方法

把算法用一种适当的方式描述出来称为算法的表示。表示算法的方法有多种,在此仅介绍用自然语言、流程图和类 C 语言方法表示算法。下面通过一个简单的例子来说明。

【例 1-2】 在 a,b,c 中存放 3 个不同的数,要求对其排序,最后的结果使 a,b,c 中的数按升序排列并输出 a,b,c。对完成此排序任务的算法描述如下。

1. 用自然语言表示算法

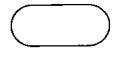
(1) 按顺序输入 a, b, c 3 个数的值。
(2) 若 a>b, 则转(3),否则转(4)。
(3) a, b 两数交换。
(4) 若 b>c,则转(5),否则转(6)。
(5) b, c 两数交换。
(6) 若 a>b,则转(7),否则转(8)。
(7) a,b 两数交换。
(8) 输出 a, b, c。
(9) 结束。

自然语言就是人们日常用的语言,可以是汉语、英语或其他语言。自然语言描述算法通俗易懂,但是叙述比较繁琐,文字冗长,又容易出现多义性,不易精确描述算法。自然语言用来描述顺序执行步骤还比较方便,若有判断和转移(即含有分支和循环的算法)的情况时,则其表示就不够直观,即不能清晰地表示算法中各步骤间的逻辑顺序。

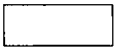
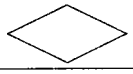
2. 用流程图表示算法

流程图是用图形表示算法,用一些几何图形的框来代表各种不同性质的操作。表 1.1 给出了国际信息处理标准 ISO 8631—1986E 中规定使用的部分图形符号。

表 1.1 常用的流程图符号

图形符号	符号名称	说 明	流 线
	起始、终止框	表示算法的开始或结束	开始框:一流入线结束框;一流出线
	输入、输出框	框中标明输入、输出的内容	只有一流入线和一流出线

续表

图形符号	符号名称	说　明	流　线
	处理框	框中标明进行什么处理	只有一流入线和一流出线
	判断框	框中标明判定条件并在框外标明判定后的两种结果的流向	一流入线两流出线(T 和 F)但同时只能一流出线起作用
	流线	表示从某一框到另一框的流向	
	连接圈	表示算法流向出口或入口连接点	一条流线

下面介绍例 1-2 中算法的流程图描述，如图 1.1 所示。

用流程图表示算法，使算法的逻辑结构比较明显，容易形成算法的模块结构，直观易学。但是流程图是通过流程线指出各框的执行顺序，对流程线的使用没有严格限制，使用者可随意地使流程转来转去。当用该流程图表示算法时，若不注意流程线的使用，就容易出现混乱，难以阅读，难以理解算法的逻辑。

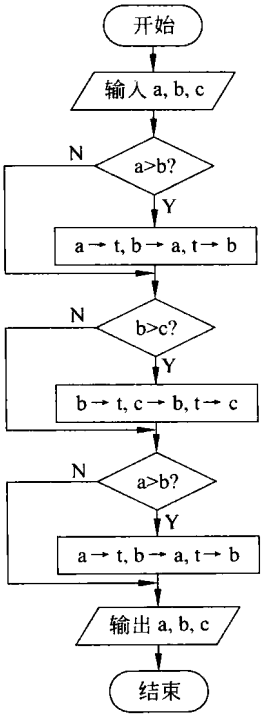


图 1.1　排序算法流程图

3. 用类 C 语言表示算法

用流程图表示算法，直观易懂，但画起来比较费时，不便于在计算机上直接实现。为此，可用计算机易接受的程序设计语言来描述算法。同时，为了突出算法描述，便于阅读理解，可适当省略程序设计语言某些语法上的严格要求，采用类似某种程序设计语言来描述，例如类 C 语言、类 Pascal 语言等。

类 C 语言(即类似程序设计语言 C 语言的一种语言)是以 C 语言为基础的一种简化的高级语言，它既不像 C 语言那样形式化，又不像自然语言那样非形式化，称它为半形式程序设计语言或伪形式程序设计语言。用它描述算法时，重点突出算法的实质，暂时避开繁琐的语法细节，集中研究算法的基本思想和主要结构，这种用于描述算法的半形式语言称为算法描述语言。算法描述语言比较灵活，虽然不能直接将算法在计算机上执行，却能很方便地将其转换成计算机上能实现的高级语言程序。

下面给出例 1-2 中算法的类 C 语言表示。

```
void  sort()
{   scanf("%d%d%d",&a,&b,&c);
    if (a>b) {t=a;a=b;b=t;}
    if (b>c)
        {t=b;b=c;c=t; }
    if (a>b)
            {t=a;a=b;b=t;}
        printf("%d%d%d\n",a,b,c);
}
```

至此，以3个不同的数进行排序为例，介绍了3种表示算法的方法：自然语言、流程图和半形式语言(类C语言)。它们各有其特点：自然语言比较容易理解，但表述起来文字冗长，而且容易出现“歧义性”，表述的含义不唯一；流程图很直观，但是当算法较复杂时画起来很麻烦，尤其是不易修改；半形式算法语言书写方便，将它所描述的算法转换成计算机能接受的高级语言程序比较容易，但是它不如流程图直观。所以，常常将流程图和半形式算法语言二者结合起来使用，即先用流程图粗略地描述算法，在流程图的基础上论证算法的正确性，然后进一步细化为半形式算法语言描述。这样可以提高算法设计的效率，保证设计的质量。在实际使用时，可以根据个人的情况和特点来选择适合自己的方法来表示算法。

1.3 C语言简介

C语言是目前世界上使用最为广泛的一种程序设计语言。C语言的产生有其深刻的技术背景与应用需求背景。

1.3.1 C语言的产生与发展

C语言的最早起源可以追溯到1957年产生的FORTRAN语言。FORTRAN语言成功的两个重要特征是：程序的编写接近人类使用的自然语言和数学公式，同时编译后产生的目标代码的执行速度与汇编语言编写的程序的执行速度相仿。

借助FORTRAN语言的成功，人们1960年又设计了一种通用语言ALGOL 60，实现了程序共享，使得一般人员能够用该语言编程进行数值处理。

由于ALGOL 60缺乏对计算机硬件的操作能力，不宜用来编写系统程序，英国剑桥大学于1963年在ALGOL 60的基础上设计了CPL(Combined Programming Language)语言。该语言能够对机器硬件进行操作，但CPL语言过于复杂，规模过大，学习和使用比较困难。

针对CPL的弱点，英国剑桥大学的Martin Richards于1967年对CPL语言进行了简化，提出了简化的CPL语言——BCPL。

1970年，美国贝尔实验室的K. Thompson对BCPL进一步简化，取名为B语言，并用B语言编写了第一个UNIX操作系统，在DEC PDP-7上实现。B语言突出了硬件处理能力，不过它过于简单，功能有限。

1972—1973年，美国贝尔实验室的D. M. Ritchie对B语言进行了完善和扩充，即在保留B语言强大的硬件处理能力基础上，扩充了数据类型，恢复了通用性，并取名为C语言。1973年，D. M. Ritchie和K. Thompson用C语言重写了UNIX操作系统。

1977—1978年，C完全独立于UNIX和PDP，成为计算机上通用的计算机语言，使得C语言不依赖于具体机器，C语言能够移植到其他机器上。

1978年以后，C的不断发展导致了各种C语言版本的出现。不同的C语言版本对传统C都有所扩充和发展。1988年，美国高价标准协会(ANSI)综合了各版本对C的扩充和发展，制定了新的C语言文本标准，称为ANSI C。ANSI C实现了C语言的规范化。