13

高等学校计算机应用基础通用教材

C 语言程序设计

赵东明　杨　蓓　编著

郑州大学出版社

图书在版编目(CIP)数据

C 语言程序设计/赵东明,杨蓓编著. —郑州:郑州大学出版社,2002.2
ISBN 7-81048-576-8

Ⅰ.C… Ⅱ.①赵… ②杨… Ⅲ.C 语言-程序设计-高等学校-教材 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2002)第 008072 号

出版社:郑州大学出版社
(地址:郑州市大学路 40 号 邮政编码:450052)
发行单位:郑州大学出版社
承印单位:郑州市文华印刷厂
开本:787 mm×1 092 mm 1/16
印张:16.375
字数:388 千字 印数:1~4 500 册
版次:2002 年 2 月第 1 版 印次:2002 年 2 月第 1 次印刷

书号:ISBN 7-81048-576-8/O·5 定价:21.80 元

内 容 提 要

本书是参照国家教育部计算机基础课程教育指导委员会制定的有关大纲和作者长期从事C语言程序设计教学实践中的总结来编写的。全书共分为9章，主要包括程序设计基础、C语言的运算符和表达式、语句和流程控制、数据的构造类型、函数、C语言的预处理、变量的存储类别、指针和文件。

本书力求内容精炼、结构清晰。作者编写了许多典型的例子，目的是突出C语言的特点并培养学生程序设计的能力。

本书可作为高等院校计算机程序设计语言课程的教材，也可以作为各类计算机应用人员自学C语言的参考书。

前　言

在各类程序设计语言中，C 语言是一种最通用的高级语言。由于它具有丰富的数据类型和运算符、目标代码效率高、可移植性强等独特之处，不仅适合编写系统软件，而且也非常适合编写应用软件。

本书是参照国家教育部计算机基础课程教育指导委员会制定的有关大纲和作者长期从事 C 语言程序设计教学实践中的总结来编写的。全书共分为 9 章，主要包括程序设计基础、C 语言的运算符和表达式、语句和流程控制、数据的构造类型、函数、C 语言的预处理、变量的存储类别、指针和文件。

作者认为，要学好一门计算机语言，不仅要掌握语言本身的特点，而且要学会用这种语言来进行程序设计，只有这样才能达到学习计算机语言的目的。

本书力求语言精炼、概念准确、结构清晰，避免空泛的论述，以使读者能在有效的时间内，通过阅读较少的篇幅来掌握大纲中所规定的内容。作者编写了许多典型的例子，目的是突出 C 语言的特点并培养学生程序设计的能力。

本书是由赵东明、杨蓓编著，柳宏川和刘钺老师参加了编写工作。其中，赵东明编写了第一章、第五章、第六章、第八章；杨蓓编写了第二章、第三章、第四章；柳宏川编写了第七章；刘钺编写了第九章。在编写过程中，郑州大学教务处和计算机科学系的各位领导给予了大力的支持，范明教授、周清雷教授提出了许多指导性的建议，谨此一并表示感谢。

限于编者的水平，本书难免有不妥之处，敬请各位同行和读者给予指教。

编者

2001 年 9 月

目 录

第1章　程序设计基础

随着计算机深入到人类社会的各个领域,20世纪末发生了一场席卷全球的信息技术(IT)革命,人们将这场革命视为21世纪——知识经济时代的前奏曲。在这场革命中,软件扮演了极其重要的角色,软件表达了由计算机硬件体现的潜在能力。

软件是由程序、数据和文档组成的,它是能够完成预定功能和任务的可执行的计算机程序。因此,无论是系统软件,还是应用软件,无疑程序设计在软件产品的开发中起着最根本、最关键的作用。

1　程序设计基本概念

一般来讲,用计算机解决一个实际问题时,大致需要下列几个步骤:①需求分析:针对具体任务,建立适当的数学模型;②画流程图:描述数学模型的解题步骤,即算法;③编写程序:用计算机能理解的语言对算法进行设计;④调试运行:检查程序是否符合题意要求。

1.1　程序

所谓程序,就是为了完成某个任务而编写的计算机能够接受并运行的一条条指令。简言之,程序就是指令的序列。程序是对所要解决问题的各个对象和处理规则进行的描述。程序具有下列性质。

(1)目的性

程序必须有明确的目标,即完成什么样的任务。

(2)有序性

程序不是指令的随意堆积,而是解题步骤的有机顺序。

(3)有限性

程序必须经过有限步骤结束,不能让计算机无终止地执行下去。

(4)操作性

程序中的指令操作,计算机都能够实现。

著名的计算机科学家沃斯(N. Wirth)给程序下了一个简单又确切的定义:

算法 + 数据结构 = 程序

1.2　算法

算法是对特定问题求解步骤的一种描述,它集中反映了程序的执行过程。数据结构是对计算机要处理的数据以及它们之间的关系进行的描述。一个算法应具有下列五个特

性。

(1)有穷性

一个算法必须总是在执行有限步骤之后结束,并且每一步都必须在有限时间内完成。

(2)确定性

算法中的每一步必须有确切的含义,人们对此并不会产生二义性的理解。并且,在任何情况下,算法只有惟一的一条执行路径。即相同的输入只能得到相同的输出。

(3)可行性

算法中描述的操作,都可以通过已实现的基本运算执行若干次组合来实现。

(4)输入

一个算法可以有零个或多个输入数据。

(5)输出

一个算法可以有零个或多个输出。

一个"好"的算法不仅要满足具体问题的需求,而且还要有较高的时间效率和空间效率。即尽可能运行时间短,占用存储空间少。

【例 1-1】 判断一个正整数 N 是否为素数。

算法一:

在 2 ~ N-1 范围内查找有没有 N 的因数,若有,则 N 不是素数;否则 N 是素数。

算法二:

在 2 ~$\sqrt{N}$范围内查找有没有 N 的因数;若有,则 N 不是素数,否则 N 是素数。

显然,算法二判断查找的范围少,因此时间效率高。

计算机处理的对象是数据,数据是描述客观事物且计算机能够接受和处理的符号的集合。由于数据的多样性,因此在数据结构中要描述数据的类型。数据类型体现了数据可能取的值以及数据所能参加的运算操作。此外,复杂的数据结构还描述了一批相关数据之间的构造方式。

数据结构与算法紧密相关,只有明确了问题的算法,才能更好地构造数据结构;但要选择好的算法,又常常依赖于好的数据结构。因此,算法和数据结构是程序的两个重要方面。

1.3 程序设计语言

程序设计语言是计算机能够理解的,人们用来编写程序的计算机语言。它是人与计算机进行信息交流的重要工具。

程序设计语言一般分为三大类:

(1)面向机器的语言

面向机器的语言主要是针对特定的机器而设计的计算机语言。它依赖于具体的计算机。如机器语言、汇编语言。

(2)面向过程的语言

面向过程的语言又称为结构化语言。这类语言独立于计算机。用户用这类语言编写程序需要描述问题的解题步骤。即告诉计算机“如何做”。如 PASCAL 语言、C 语言和 Ada 语言。

(3)面向问题的语言

面向问题的语言又称为面向对象的语言。这类语言也独立于计算机。但用户用这类语言编写程序不需要描述详细的解题步骤。只需告诉计算机“做什么”。如 SQL、VC 等。

1.4　程序设计方法

程序设计就是针对具体任务进行设计、编写和调试计算机程序的过程。

编写的程序要具有正确性、可靠性、可读性和可维护性。因此,进行程序设计不仅要掌握计算机语言本身的基本结构和语句,还要学习程序设计的方法和技巧。

结构化程序设计,就是按照一组能提高程序可读性和可维护性的规则,采用自顶向下、逐步求精的思想,对程序进行的模块化设计。

所谓模块化,就是将一个复杂的大问题分解成若干个功能较独立的小问题来进行设计。每个小问题就是一个模块。模块一定要简单、功能独立,这样才能使程序具有一定的灵活性和可靠性。

模块的划分采用自顶向下的思想,这样符合人们考虑问题的思维方式。模块的编写也要遵循由粗到细、由概括到具体的过程,这就是逐步求精。

从结构化程序设计角度来看,程序有四个基本结构:①顺序结构:按顺序执行每个处理过程。②选择结构:根据条件,在几个处理中选择一个处理过程执行。③循环结构:若干个处理过程反复执行,当满足某个条件时结束循环。④子程序结构:用一条语句替代一组语句的处理过程。子程序结构又称为模块结构。

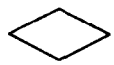
2　流程图

用流程图来描述问题的解题步骤,可使算法十分明确、具体直观,同时也便于程序的检查和修改。因此特别适合初学者。

2.1　传统的流程图

传统的流程图常用的符号如表 1 - 1。

表 1 - 1　传统流程图典型符号表

	端　点	表示开始、结束、停止或中断
	处　理	对框内符号进行处理，一个入口，一个出口
	判　断	对框内的条件进行判断，一个入口，两个出口
	输入、输出	对框内信息进行输入或输出，一个入口一个出口
	流程线	表示信息流向的控制

用传统流程图的符号来表示算法非常简单，方框表示处理，菱形表示判断条件，箭头表示控制流。图 1 - 1 表示了三种基本程序结构的构成元素。

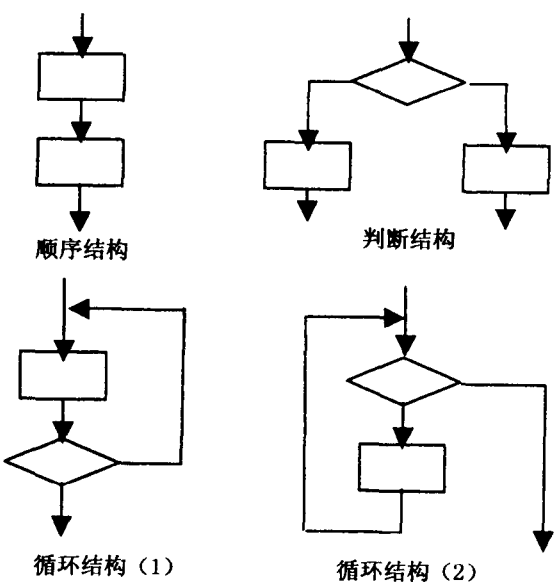


图 1 - 1 三种基本构成元素

用上述基本构成元素很容易做到在一种结构中嵌套其他结构，从而可实现程序的任何逻辑要求。

【例 1 - 2】　输入一正整数，判断是否为完全平方数（用流程图描述算法）。

如果一个自然数可表示为另一个自然数的平方，则称这个数为完全平方数。如：25 =

5^2。

此问题的解题方法如图1 - 2。

这种流程图曾经是最广泛使用的图形表示方法，但它也是最广泛地被滥用的方法。它无法限制GOTO 语句的使用，循环结构仍采用判断结构符号，因此在程序流程图中不易区分哪种结构。

一般来说，如果需要从一组嵌套的循环或条件中退出，完全依赖结构化的构成元素将会导致效率降低。更重要的是，退出路径上的复杂逻辑测试将会影响程序的控制流，增加出错的可能，也降低了程序的可读性和可维护性。

那么，如何解决这一问题呢？

一种方法是重新设计过程表示的图符，保证内层嵌套的控制流不需要退出分支；另一种方法是允许在一定的范围内破坏结构化的构成元素，设计一条特殊的退出路径。显然，第一种方法是理想的，不过第二种方法也不破坏结构化编程的意图。

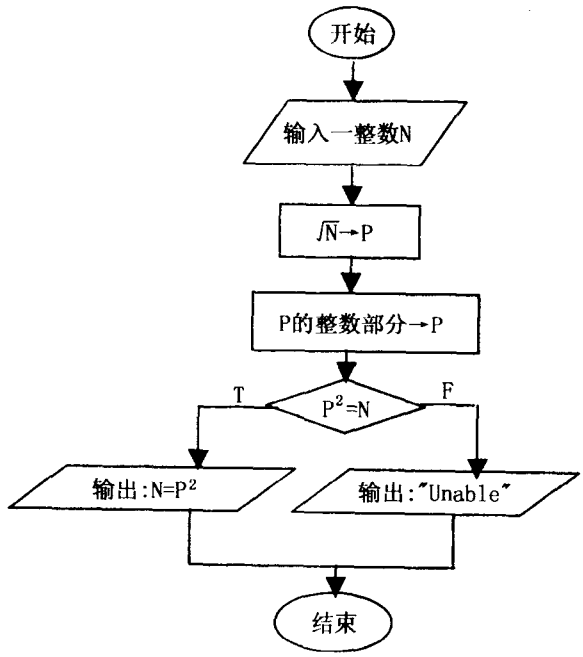


图 1 - 2　例 1 - 2 的流程图

2.2　N - S 流程图

N - S 图是美国 Nassi 和 Shneiderman 于 1973 年提出的结构化流程图。Chapin 在 1974 年又对其进一步扩展。因此，N - S 图又称为 Chapin 图或盒状图。

N - S 图的目标是开发一种不破坏结构化基本构成元素的过程设计表示，它的特征是：①功能定义明确，表示清晰；②不允许随意的控制流；③局部和全局数据的作用域容易确定；④表示递归方便。

N - S 图结构化框架的图形表示如图 1 - 3。

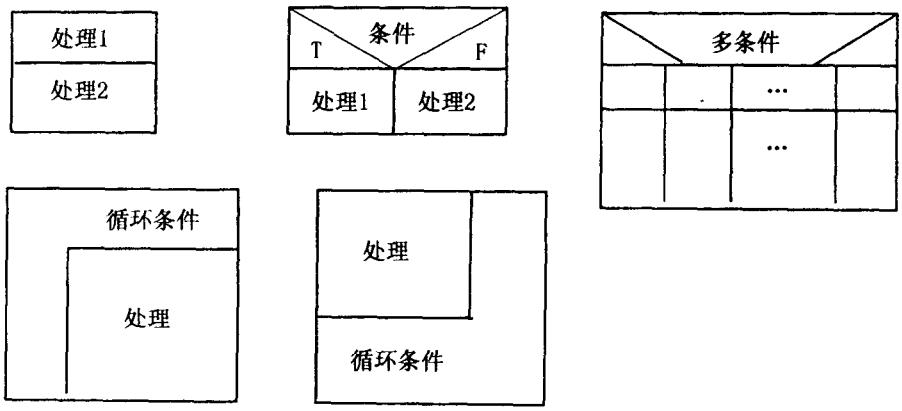


图 1-3　N-S 图构成元素

N-S 图的最基本的成分是长方形框，它只能有一个入口和一个出口，盒内用不同形状的线分割来表示不同的结构。

【例 1-3】　输入若干个整数(0 表示输入结束)，统计输入的个数并求出其中的最大数和最小数。

图 1-4 用 N-S 流程图描述了问题的解决方法。

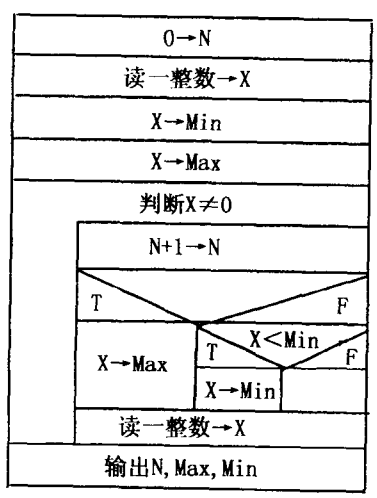


图 1-4　例 1-3 的 N-S 图

3　C 语言简介

C 语言是一种通用的计算机程序设计语言。C 语言的产生是与 UNIX 操作系统紧密相关的。事实上，当初贝尔实验室的 D. M. Ritchie 开发 C 语言就是为了更好地描述 UNIX 操作系统。操作系统是计算机必不缺少的系统软件。过去大多数系统软件都是用汇编语言编写的，但由于汇编语言过多地依赖于计算机的硬件环境，它开发出的软件可读性和可移植性就较差。而一般的高级语言又难以实现一些与计算机硬件有关的操作。于是，C 语言的出现协调了两者的关系。

自从有了 C 语言之后，UNIX 系统的核心程序、所有 UNIX 的命令解释程序、各种实用程序（包括 C 本身的编译程序）都是用 C 开发的。一些应用系统的程序大多也是用 C 编写的。

3.1　C 语言的由来

C 语言是 UNIX 操作系统设计者之一 D. M. Ritchie 在 1972 年为重写 UNIX 系统而开发的。追溯起来，其发展过程为：

ALGOL60 ⟶ CPL ⟶ BCPL ⟶ B ⟶ C
（1960 年）（1963 年）（1967 年）（1970 年）（1972 年）

ALGOL60 是一种面向过程的高级语言，比 FORTRAN 语言晚几年，但由于它的结构非常严密，其设计者十分注重语法规则和分程序结构，因此 ALGOL60 对后来的程序设计语言，特别是 PASCAL 语言、PL/1 语言有着极其重要的影响。不过，由于它与计算机硬件相距太远，所以不宜写系统软件。

CPL（Combined Programming Language）是剑桥大学在 1963 年为了使 ALGOL60 得到实用，仿照 ALGOL60 而设计的计算机语言。1967 年剑桥大学的 Martin. Richards 对 CPL 进行了精简，并保持了它的基本特点，又设计出了 BCPL（Basic Combined Programming Language）语言。由于 BCPL 语言规模小，移植性强，目前有一些欧洲国家仍在使用 BCPL。

B 语言是贝尔实验室的 K. Thompson 为了摆脱汇编语言的困扰，在 1970 年以 BCPL 为基础而开发的一种更有效描述 UNIX 操作系统的高级语言。B 语言之所以没有流行起来，主要是因为它缺乏丰富的数据类型。

C 语言是 D. N. Ritchie 在 1972 年对 B 语言的改进。C 语言增加了丰富的数据类型和强有力的控制结构。1973 年 K. Thompson 和 D. N. Ritchie 用 C 语言重写了 UNIX，并进一步扩充了 UNIX 的功能，从而开创了 UNIX 发展的新局面。

UNIX 操作系统的许多特点，如高效、功能强、易于移植和可靠性强，都得益于 C 语言。UNIX 系统的 95% 代码是用 C 语言写的。另一方面，C 语言之所以有今天这种局面，又因为它根植于 UNIX 系统上。目前，许多著名软件都是 C 语言开发的，如 ORACLE、FOXBASE、NETWARE、WINDOWS 等。