



普通高等教育“十二五”规划教材

C语言 程序设计教程

C YUYAN CHENGXU SHEJI JIAOCHENG

主 编 ● 王洪平 陈滢生 周金容
副主编 ● 郑 杰 邓永生 安 宓



西南交通大学出版社
Http://press.swjtu.edu.cn

普通高等教育“十二五”规划教材

C 语言程序设计教程

主 编 王洪平 陈滢生 周金容

副主编 郑杰 邓永生 安 宓

西南交通大学出版社

· 成 都 ·

内容简介

全书共分 11 章,包括程序设计的基础知识、算法与程序流程控制,指针、结构体、文件访问与编译预处理等各方面的知识。具体有 C 语言概述, Turbo C 和 Visual C++ 6.0 开发环境的使用, C 语言的数据类型、程序控制结构、数组应用、函数调用、文件操作以及结构体、指针、编译预处理等。通过对大量案例、实例的分析讲解,再利用适量习题进行练习与巩固,可以使学生快速掌握 C 语言程序设计的基本方法和编程技巧。

本书结构清晰、图文并茂、实用性强。程序实例有详细的讲解,各章节配有多媒体课件,易于教学与个人自学。本书既可作为高等院校和高职高专计算机专业的教材,又可作为初学者的自学用书和 C 语言计算机等级考试的参考书。对信息类专业可以讲授本教材的全部内容,对非信息类专业可以讲授本教材的部分内容。

图书在版编目(CIP)数据

C 语言程序设计教程 / 王洪平, 陈滢生, 周金容主编.
—成都: 西南交通大学出版社, 2013.8
普通高等教育“十二五”规划教材
ISBN 978-7-5643-2541-1

I. ①C… II. ①王… ②陈… ③周… III. ①
C 语言—程序设计—高等学校—教材 IV. ①TP312

中国版本图书馆 CIP 数据核字(2013)第 188324 号

普通高等教育“十二五”规划教材

C 语言程序设计教程

主编 王洪平 陈滢生 周金容

*

责任编辑 李芳芳

特邀编辑 李 丹

封面设计 何东琳设计工作室

西南交通大学出版社出版发行

四川省成都市金牛区交大路 146 号 邮政编码: 610031 发行部电话: 028-87600564

<http://press.swjtu.edu.cn>

四川锦祝印务有限公司印刷

*

成品尺寸: 185 mm × 260 mm 印张: 15.25

字数: 379 千字

2013 年 8 月第 1 版 2013 年 8 月第 1 次印刷

ISBN 978-7-5643-2541-1

定价: 32.00 元

图书如有印装质量问题 本社负责退换
版权所有 盗版必究 举报电话: 028-87600562

前 言

C 语言是一门面向过程的计算机语言, 蕴含了程序设计的基本思想, 囊括了程序设计的基本概念, 是理工科高等院校、高等职业院校计算机及相关专业的一门重要基础课程。学习 C 语言, 目的是培养学生的程序设计理念, 掌握程序设计的基本方法, 为后续课程的学习打下坚实的基础。

虽然面向对象的程序设计在今天已经很流行, 但到目前为止, C 语言仍然是国内外广泛使用的程序设计语言之一, 同时也是国内外大学开设的程序设计基础课程。C 语言自身具有功能丰富、表达力强、使用方便灵活、程序代码效率高、可移植性好等特点。此外, C 语言不但具有高级语言的特点, 又具有汇编语言的特点, 且具有较强的系统处理能力。

本书的目标是作为程序设计课程的入门教材或自学读物, 讲述 C 语言程序设计中涉及的最基本的语法规则和程序设计的基本方法。全书内容尽可能简练、通俗易懂, 注重强化编程基础知识与程序设计能力的训练, 每章都配备了适量的例题。

基本理论是学习 C 语言的基础和钥匙, 由于 C 语言对一般初学者来说, 规则较多, 不易掌握, 所以学习上会有一定的困难, 而 C 语言的应用范围越来越广, 所涉及的知识越来越多, 这也增加了学习的难度。全书对 C 语言的精华部分做了较为细致的介绍, 对必要的原理进行了适当的阐释, 模块设计努力做到“必需、够用、实效”。

考虑到不同学校、不同专业对 C 语言程序设计课程的教学要求不同, 各位教师可以根据课程的课时设置, 选择授课内容。全书推荐授课课时为 60~70 学时。全书共由 11 章组成, 主要内容有: C 语言概述, Turbo C 和 Visual C++ 6.0 开发环境的使用, C 语言的数据类型、程序控制结构、数组应用、函数调用、文件操作以及结构体、指针、编译预处理等。建议授课教师将教学重点放在表达式、控制结构与语句、数组、函数、指针上, 并反复上机训练。

全书由南充职业技术学院王洪平负责制定、编写大纲, 确定样章, 全书统稿的工作。重庆电子工程职业学院郑杰老师编写了第 1 章, 并协助全书的统稿工作; 重庆机电职业技术学院邓永生老师编写了第 2、3 章; 西南交通大学希望学院周金容老师编写了第 4、5 章; 重庆人文科技学院陈滢生老师编写了第 6、7 章; 南充职业技术学院王洪平老师编写了第 8、9 章; 南充职业技术学院安宓老师编写了第 10、11 章。

由于作者水平有限, 再加上时间仓促, 书中疏漏之处在所难免, 恳请各位同行和读者不吝赐教。

编 者

2013 年 6 月

目 录

第 1 章 程序设计及 C 语言概述	1
1.1 程序与算法	1
1.2 C 语言概述	10
1.3 C 语言的程序结构	11
1.4 Turbo C 开发环境	14
1.5 Visual C++ 6.0 开发环境	18
本章小结	27
习 题	28
第 2 章 数据类型、运算符与表达式	30
2.1 基本字符、标识符和关键字	30
2.2 C 语言的的数据类型	32
2.3 运算符和表达式	41
2.4 数据类型转换	45
本章小结	46
习 题	47
第 3 章 顺序程序设计	52
3.1 程序的三种基本结构	52
3.2 顺序结构程序的设计思想	53
3.3 C 语句概述	53
3.4 赋值语句	54
3.5 字符数据的输入与输出	55
3.6 格式化输入	57
3.7 程序举例	62
本章小结	64
习 题	64
第 4 章 选择结构程序设计	68
4.1 关系运算符与关系表达式	68
4.2 逻辑运算符与逻辑表达式	69
4.3 if 语句	71
4.4 switch 语句	78

4.5 程序举例	81
本章小结	83
习 题	83
第 5 章 循环控制	89
5.1 while 语句	89
5.2 do-while 语句	91
5.3 while 和 do-while 循环比较	93
5.4 for 语句	94
5.5 goto 语句以及用 goto 语句构成循环	97
5.6 循环的嵌套	98
5.7 break 和 continue 语句	98
5.8 程序举例	100
本章小结	102
习 题	103
第 6 章 数 组	109
6.1 一维数组	109
6.2 二维数组	114
6.3 字符数组与字符串	118
本章小结	126
习 题	126
第 7 章 函 数	130
7.1 函数的概念	130
7.2 函数的定义和调用	131
7.3 函数的参数传递方式	136
7.4 函数的嵌套调用与递归调用	138
7.5 变量的作用域和生命期	141
本章小结	147
习 题	148
第 8 章 指 针	151
8.1 指针概述	151
8.2 访问变量的地址	152
8.3 指针变量	153
8.4 指针与数组	162
8.5 指针与字符串	167
8.6 指针与函数	169
本章小结	172
习 题	173

第 9 章 编译预处理与位运算	177
9.1 宏定义	177
9.2 文件包含	180
9.3 条件编译	181
9.4 常用库函数的应用	184
9.5 位运算	187
本章小结	195
习 题	195
第 10 章 结构体、共用体与枚举类型	198
10.1 结构体类型与结构体变量	198
10.2 结构体变量的引用和初始化	201
10.3 结构体数组	203
10.4 共用体	206
10.5 枚举类型	208
10.6 类型定义	209
本章小结	211
习 题	211
第 11 章 文件管理	215
11.1 文件概述	215
11.2 文件的打开与关闭	217
11.3 文件的读写	219
11.4 其他函数	227
本章小结	228
习 题	229
附录一 ASCII 码表	232
附录二 TC 常见错误编译信息	233
参考文献	236

第1章

程序设计及C语言概述

【学习目标】

1. 了解程序、程序设计、高级语言的概念；
2. 了解C程序的基本结构；
3. 了解算法的概念；
4. 理解并掌握一些常用算法；
5. 掌握用流程图的方法来描述算法。

1.1 程序与算法

有人以为计算机是“万能”的，会自动进行所有的工作，甚至觉得计算机神秘莫测。这是很多初学者的误解，其实，计算机的每一个操作都是根据人们事先指定的指令进行的。例如用一条指令要求计算机进行一次加法运算，用另一条指令要求计算机将某一运算结果输出到显示屏。为了使计算机执行一系列的操作，必须事先编好一系列的指令，输入到计算机。

1.1.1 程序及程序设计语言

1. 程 序

所谓程序，就是计算机能识别和执行的一组指令。每一条指令使计算机执行特定的操作。只要让计算机执行这个程序，计算机就会“自动地”执行各条指令，有条不紊地进行工作。一个特定的指令序列，用来完成一定的功能。为了使计算机系统实现各种功能，需要成千上万个程序。这些程序大多数是由计算机软件设计人员根据需要设计好的，作为计算机软件系统的一部分提供给用户使用。此外，用户还可以根据自己的实际需要设计一些应用程序，例如学生成绩统计程序、财务管理程序、工程计算程序等。

总之，程序就是一系列的操作步骤，计算机程序就是由人事先规定的计算机完成某项工作的操作步骤。每一步骤的具体内容由计算机能够理解的指令来描述，这些指令告诉计算机“做什么”和“怎样做”。只有懂得程序设计，才能真正了解计算机是怎样工作的，才能更深入地使用计算机。

2. 程序设计语言

人与人之间的交流可依靠语言完成。人和计算机的交流，也需要语言。创造一种计算机和人都能识别的语言，这就是计算机语言。计算机语言主要有以下几种：

1) 机器语言

机器语言就是机器指令的集合，机器指令是能被计算机直接理解和执行的指令，机器指令在形式上是由“0”和“1”构成的一串二进制代码，每种计算机都有自己特有的一套机器指令。机器语言与人所习惯的自然语言差别很大，相比较而言机器语言较为难学、难记、难读，因此很难用机器语言来开发实用的计算机程序。

2) 汇编语言

汇编语言是采用助记符来代替机器码，如用 ADD 表示加法（Addition），用 SUB 表示减法（Subtraction）等，同时又用变量取代各类地址，如用 A 取代地址码等。汇编语言源程序就是用汇编语言编写的程序。这种程序必须经过翻译（也称为汇编），变成机器语言程序才能被计算机识别和执行。汇编语言在一定程度上克服了机器语言难于辨认和记忆的缺点，但对大多数用户来说，仍然是不便理解和使用的。

3) 高级语言

高级程序设计语言克服了低级语言的缺点，它是类似于“数学表达式”，接近自然语言，又能为机器所接受的程序设计语言。高级语言具有容易学习、方便使用、通用性强、移植性好的特点，便于各类人员学习和应用。

4) 过程式语言和非过程式语言

面向解题过程，告诉计算机“怎么做”，这种语言称为过程式语言。面向处理对象，告诉计算机“做什么”而不必指出“怎么做”，计算机就能完成所要求的任务。这种语言称为非过程式语言。

5) 解释型与编译型语言

计算机不能直接识别高级语言，要让计算机能执行高级语言，需要将高级语言翻译成等价的机器语言，这种翻译有两种形式：

- ① 边翻译边执行；
- ② 全部翻译完成后才执行。

前一种称为解释方式，后一种称为编译方式。C, FORTRAN, PASCAL 等为编译型语言。BASIC, Java 等为解释型语言。用高级语言写成的程序称为源程序（源代码）。翻译成的机器语言称为目标程序（目标代码）。

1.1.2 算法及其特性

1. 算法

一个程序主要包括以下两方面的信息：

1) 对数据的描述

在程序中要指定用到哪些数据以及这些数据的类型和数据的组织形式。这就是数据结构 (Data Structure)。

2) 对操作的描述

即要求计算机进行操作的步骤,也就是算法 (Algorithm)。

数据是操作的对象,操作的目的是对数据进行加工处理,以得到期望的结果。下面以调酒师调酒为例进行说明,现需要调酒师调制一些鸡尾酒,那么,调酒师需求如下:①所用原料,指出为了做出鸡尾酒,应该使用哪些材料;②操作步骤,指出有了这些原料,应按什么样的步骤进行加工,才能做出所需的口味的鸡尾酒。没有原料无法加工出所需的鸡尾酒,而对同一些原料可以加工出不同口味的鸡尾酒。作为程序设计人员,必须认真考虑和设计数据结构和操作步骤(即算法)。著名计算机科学家沃思(Nikiklaus Wirth)提出一个公式:

$$\text{算法} + \text{数据结构} = \text{程序}$$

直到今天,这个公式对于过程化程序来说依然是适用的。

实际上,一个过程化的程序除了以上两个主要要素之外,还应当采用结构化程序设计方法进行程序设计,并且用某一种计算机语言表示。因此,算法、数据结构、程序设计方法和语言工具4方面是一个程序设计人员应具备的知识,在设计一个程序时要综合运用这几方面的知识。在这4个方面中,算法是灵魂,数据结构是加工对象,语言是工具,编程需要采用合适的方法。

什么是算法?算法是解决“做什么”和“怎么做”的问题。程序中的操作语句,实际上就是算法的体现。显然,不了解算法就谈不上程序设计。

生活中的小事也处处都是算法的体现,例如:吃饭的过程,就是吃饭的算法。一首歌曲的乐谱也可以称为该歌曲的算法。这些事情,我们都要按它们的规定去做。只是人们习以为常,从没去细究其思维支配的过程。

要让计算机去做一件事,就必须把如何做这件事情分解成许多小步骤,每一步由计算机语言的命令来完成,最后执行完一系列命令,把所要做的事情做完。描述这些步骤的方法有自然语言、流程图和N-S图等。把计算机每一条命令对应的步骤组成一个规则的整体,就是计算机程序。计算机程序设计“算法”就是利用计算机语言,让计算机完成解决问题的步骤。

从广义地讲,算法是为完成一项任务所应当遵循的一步一步的、规则的、精确的、无歧义的描述,它的总步数是有限的。从狭义上讲,算法是解决一个问题采取的方法和步骤的描述。

例如,想要计算机得到三个数中的最大数,那么首先要知道其算法,算法如下:

- (1) 输入 A, B, C 三个数。
- (2) A 与 B 中大的一个放入 $\max$ 中。
- (3) 把 C 与 $\max$ 中大的一个放入 $\max$ 中。
- (4) 输出 $\max$, $\max$ 即为最大数。

若想让计算机实现“输入10个数,打印输出其中最大的数”这一功能时,同样需要知道其算法,具体算法如下:

- (1) 输入1个数,存入变量 A 中,将记录数据个数的变量 N 赋值为1,即 $N=1$ 。

- (2) 将 A 存入表示最大值的变量 $\max$ 中, 即 $\max=A$ 。
- (3) 再输入一个值给 A , 如果 $A > \max$, 则 $\max=A$, 否则 $\max$ 不变。
- (4) 让记录数据个数的变量增加 1, 即 $N=N+1$
- (5) 判断 N 是否小于 10, 若成立则转到第 (3) 步执行, 否则转到第 (6) 步。
- (6) 打印输出 $\max$ 。

计算机算法可分为两大类: 数值运算算法和非数值运算算法。数值运算的目的是求数值解, 例如求方程的根、求一个函数的定积分等都属于数值运算范围。非数值运算包括的范围十分广, 最常见的是用于事务管理领域, 例如图书检索、人事管理和行车调度管理等。目前, 计算机在非数值运算方面的应用远远超过了在数值运算方面的应用。

2. 算法的特性

要编写程序, 必须学会设计算法。不要以为任意写出的一些执行步骤就构成一个算法。一个有效的算法应该具有以下特点。

1) 有穷性

一个算法应包含有限的操作步骤, 而不能是无限的。

事实上, “有穷性”往往指“在合理的范围之内”。如果让计算机执行一个历时 1 000 年才结束的算法, 这虽然是有穷的, 但超过了合理的限度, 人们也不能把它视作有效算法。什么叫“合理限度”, 并无严格标准, 由人们的常识和需要而定。

2) 确定性

算法中的每一个步骤都应当是确定的, 而不应当是含糊的、模棱两可的。例如, 有一个健身操的动作要领, 其中有一个动作是“手举过头顶”, 这个步骤就是不确定的、含糊的。是双手都举过头, 还是左手(或右手)举过头顶? 而且具体要举过头顶多少厘米呢? 不同的人可以有不同的理解。算法中的每一个步骤应当不致被解释成不同的含义, 而应是十分明确无误的。也就是说, 算法的含义应当是唯一的, 而不应当产生“歧义性”。所谓“歧义性”是指可以被理解为两种或多种的可能含义。

3) 有零个或多个输入

所谓输入是指在执行算法时需要从外界取得必要的信息, 一个算法可以没有输入, 也可以有两个或多个输入。例如, 求两个整数 m 和 n 的最大公约数, 则需要输入 m 和 n 的值。

4) 有一个或多个输出

算法的目的是为了求解, “解”就是输出。算法的输出与计算机的打印输出不同, 一个算法得到的结果就是算法的输出, 没有输出的算法是没有意义的。

5) 有效性

算法中的每一个步骤都应当能被有效地执行, 并得到确定的结果。例如, $b=0$, 则 a/b 是不能有效执行的。

可以使用别人已设计好的现成算法,只需根据算法的要求给以必要的输入,就能得到输出的结果。这时算法如同一个“黑箱子”一样,可以不了解“黑箱子”中的结构,只是从外部特性上了解算法的作用,即可方便地使用算法。例如,对一个“输入3个数,求其中最大值”的算法,只要输入 a, b, c 这3个数,执行算法后就能得到其中最大的数。

1.1.3 算法的描述

为了表示一个算法,可以用不同的方法。常用的方法有:自然语言、程序流程图、N-S盒图和伪代码语言等。

1. 用自然语言表示算法

自然语言就是人们日常使用的语言,可以是汉语、英语或其他语言。用自然语言表示通俗易懂,但文字冗长,容易出现歧义。自然语言表示的含义往往不太严格,要根据上下文才能判断其正确含义。例如有这样一句话:王先生对赵先生说他的孩子考上公务员了,请问是王先生的孩子考上公务员还是赵先生的孩子考上公务员了呢?仅从这句话本身难以判断。此外,用自然语言来描述包含分支和循环的算法不太方便。因此,除了那些简单的问题以外,一般不用自然语言表示算法。

2. 用流程图表示算法

流程图是用一些图框来表示各种操作。用图形表示算法,直观形象且易于理解。美国国家标准协会 ANSI 规定了一些常用的流程符号,如图 1.1 所示,已为世界各国程序工作者普遍采用。

图 1.1 中椭圆表示开始点和结束点;平行四边形表示输入输出;矩形表示处理过程;箭头表示数据流。泡茶事件的过程的流程图如图 1.2 所示。

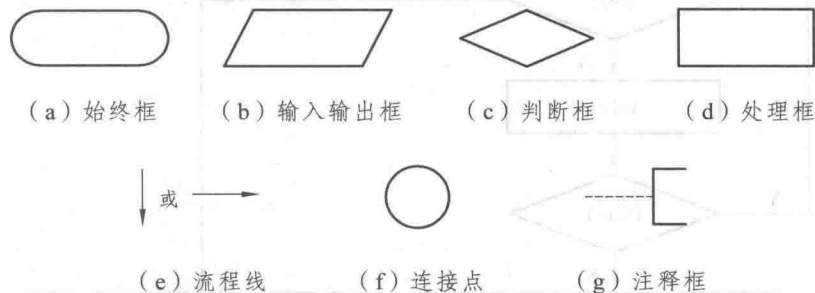


图 1.1 流程图符号

连接点(小圆圈)是用于将画在不同地方的流程线连接起来。如图 1.3 中有两个以①为标志的连接点,它表示这两个点是互相连接在一起的,实际上它们是同一个点,只是画不下才分开来画。用连接点可以避免流程线交叉或过长,使流程图清晰。

菱形框的作用是对一个给定的条件进行判断,根据给定的条件是否成立决定如何执行其后的操作。它有一个入口、两个出口。例如判断素数算法流程图,如图 1.4 所示。

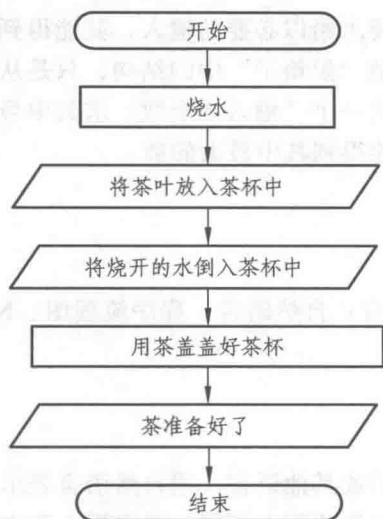


图 1.2 泡茶流程图

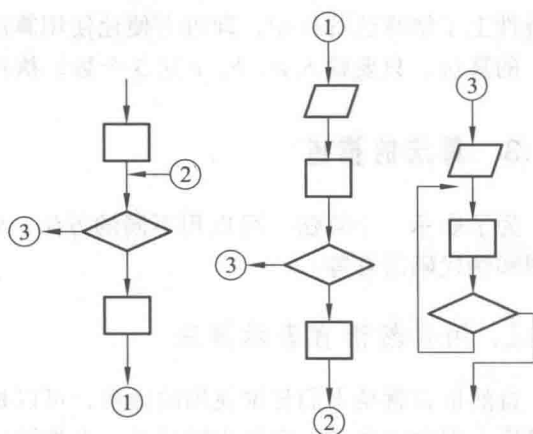


图 1.3 连接流程图

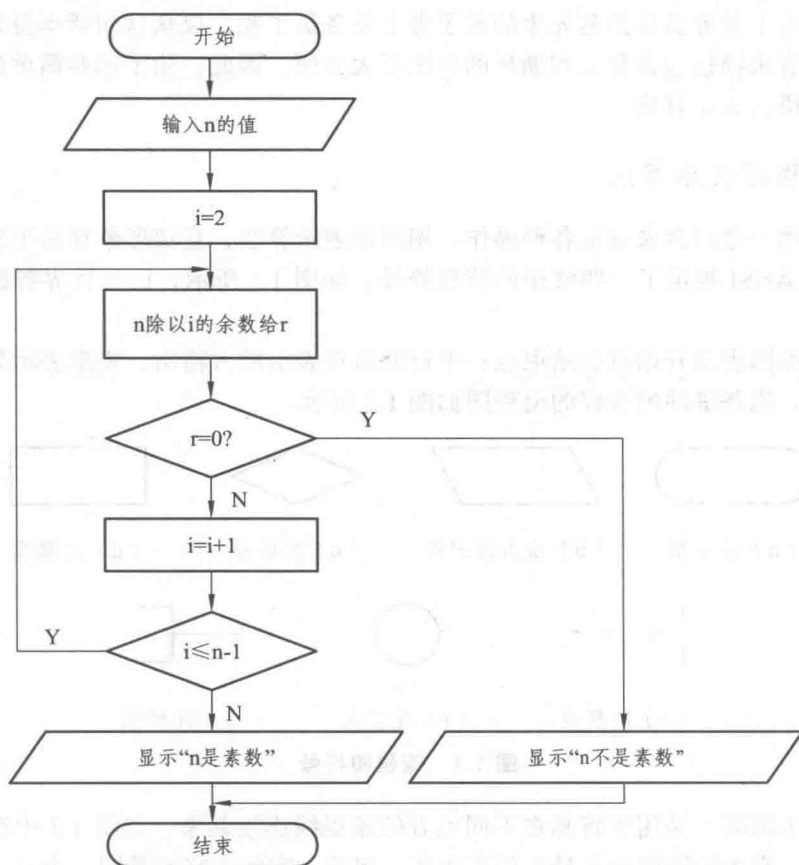


图 1.4 判断素数算法流程图

注释框不是流程图中必要的部分，不反映流程和操作，只是为了对流程图中某些框的操作做必要的补充说明，以帮助阅读流程图的人更好地理解流程图的作用。

由图 1.4 可以看出,流程图可以清晰地描述计算机程序的流程,但对于大而复杂的程序用流程图来表示时,会显得凌乱繁杂,1966 年有人提出了程序设计的 3 种基本结构,即顺序结构、条件判断分支结构和循环控制结构。

- ① 顺序结构,如图 1.5 所示;
- ② 条件判断分支结构,如图 1.6 所示;
- ③ 循环控制结构,如图 1.7 所示。



图 1.5 顺序结构图

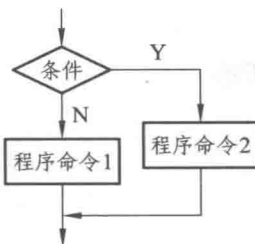


图 1.6 条件分支结构图

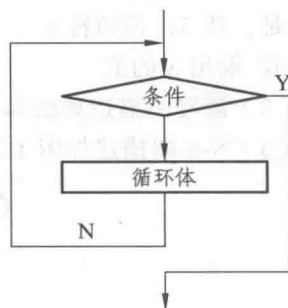


图 1.7 循环控制结构图

3. 用 N-S 盒图表示算法

结构化程序设计可以用程序设计基本结构组成的构件,顺序组合成各种复杂的算法结构,这样程序流程图的程序流程线就显得多余。1973 年美国学者 Nassi 和 Shneiderman 提出了用矩形框表示的 3 种基本结构,而使用 3 种基本结构的矩形框嵌套,就可以组成各种复杂的程序算法了。

- (1) 处理,如图 1.8 所示;
- (2) 判断,如图 1.9 所示;
- (3) 循环,如图 1.10 所示。



图 1.8 N-S 处理图

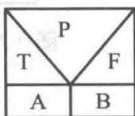


图 1.9 N-S 判断图



图 1.10 N-S 循环图

4. 伪代码语言

伪代码语言是一种解释性语言,其执行方式介于编译型语言和解释型语言之间。伪代码编程语言无需编译,在执行时自动转换成二进制形式。这种可执行的二进制文件是以伪代码的形式而不是机器语言的形式存储的,不必在每次运行该程序的时候,都必须将其转换成二进制,在第一次转换成伪代码后,以后每次执行都可使用该伪代码版本,其速度比编译型语言及程序慢,但比解释型语言快。

5. 具体算法的多种描述

【例 1.1】 用不同方法描述计算 1~100 的自然数相加的和的算法。

(1) 自然语言描述算法。

① $0 \Rightarrow S$ 单元。

② $1 \Rightarrow n$ 单元。

③ $S+n=S$ 。

④ $n+1 \Rightarrow n$ 。

⑤ 判断 $n \leq 100$?

是, 转 3; 否则转 6。

⑥ 输出 S 的值。

(2) 流程图描述算法如图 1.11 所示。

(3) N-S 图描述如图 1.12 所示。

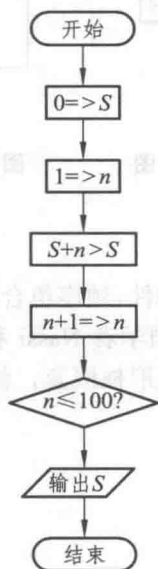


图 1.11 流程图实例

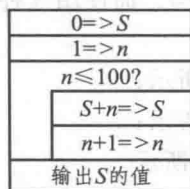


图 1.12 N-S 图实例

(4) 伪代码描述算法:

$0 \Rightarrow S$

$1 \Rightarrow n$

if $n \Rightarrow 100$

$S+n \Rightarrow S$

$n+1 \Rightarrow n$

print S

在实际应用中, 会根据实际情况的不同, 采取不同的描述方式。

1.1.4 结构化程序设计方法

结构化程序设计 (Structured Programming) 就是以模块功能和处理过程设计为主的详细设计。它最早由 E. W. Dijkstra 在 1965 年提出, 是软件发展的一个重要的里程碑。它的主要

观点是采用自顶向下、逐步求精及模块化的程序设计方法；使用三种基本控制结构构造程序，任何程序都可由顺序、选择、循环三种基本控制结构构造。这种程序便于编写、阅读、修改和维护，这就减少了程序出错的机会，提高了程序的可靠性，保证了程序的质量。

结构化程序设计的详细描述过程常用三种工具：图形、表格和语言。

图形：程序流程图、N-S图、PAD图。

表格：判定表。

语言：过程设计语言（PDL）。

结构化程序设计强调程序设计风格和程序结构的规范化，提倡清晰的结构。怎样才能得到一个结构化的程序呢？如果面临一个复杂的问题，是难以一下子写出一个层次分明、结构清晰、算法正确的程序的。结构化程序设计方法的基本思路是：把一个复杂问题的求解过程分阶段进行，每个阶段处理的问题都控制在人们容易理解和处理的范围内。具体来说，采取以下方法来保证得到结构化的程序：

1. 自顶向下，逐步细化

“自顶向下，逐步细化”的设计思想，其出发点是从问题的总体目标开始，抽象低层的细节，先专心构造高层的结构，然后一层一层地分解和细化。这使设计者能把握主题，高屋建瓴，避免一开始就陷入复杂的细节中，使复杂的设计过程变得简单明了，过程的结果也容易做到正确可靠。

设计房屋就是用自顶向下，逐步细化的方法。先进行整体规划，然后确定建筑物方案，再进行各部分的设计，最后进行细节的设计（如门窗、楼道等），而绝不会在没有整体方案之前先设计楼道和厕所。而在完成设计，有了图纸之后，在施工阶段则是自下而上实施的，用一砖一瓦先实现一个局部，然后由各部分组成一个建筑物。

应当掌握自顶向下，逐步细化的设计方法。这种设计方法的过程是将问题求解由抽象逐步具体化的过程。

2. 模块化设计

一个复杂问题，肯定是由若干稍简单的问题构成。模块化是把程序要解决的总目标分解为子目标，再进一步分解为具体的小目标，把每一个小目标称为一个模块。在拿到一个程序模块（实际上是程序模块的任务书）以后，根据程序模块的功能将它划分为若干个子模块，如果这些子模块的规模还嫌大，可以将其再划分为更小的模块。这个过程采用自顶向下的方法来实现。

程序中的子模块在C语言中通常用函数来实现。程序中的子模块一般不超过50行，即把它打印输出时不超过一页，这样的规模便于组织，也便于阅读。划分子模块时应注意模块的独立性，即用一个模块完成一项功能，耦合性愈少愈好。模块化设计的思想实际上是一种“分而治之”的思想，把一个任务分为若干个子任务，每一个子任务就相对简单了。

3. 结构化编码

在设计好一个结构化的算法之后，还要善于进行结构化编码（Coding）。所谓编码就是将

已设计好的算法用计算机语言来表示,即根据已经细化的算法正确地写出计算机程序。结构化的语言(如 Pascal, C, Visual Basic 等)都有与 3 种基本结构对应的语句,进行结构化编码是不困难的。

总之,结构化程序设计方法用来解决人脑思维能力的局限性和被处理问题的复杂性之间的矛盾。

1.2 C 语言概述

1.2.1 C 语言的发展

1960 年出现的 ALGOL 60 是一种面向问题的高级语言,离硬件比较远,不宜编写系统程序。1963 年英国剑桥大学推出 CPL (Combined Programming Language) 语言,规模较大,较难实现。1967 年英国剑桥大学 Martin Richards 对 CPL 语言做了简化,推出了 BCPL (Basic Combined Programming Language) 语言。1970 年贝尔实验室的 Ken Thompson 对 BCPL 语言进行简化,设计出很简单的而且接近硬件的 B 语言,并用 B 语言编写了第一个 UNIX 操作系统。但 B 语言过于简单,功能有限。1972 年,贝尔实验室的 D. M. Ritchie 在 B 语言的基础上设计出 C 语言。C 语言只是为描述和实现 UNIX 操作系统提供一种工作语言而设计。1973 年 K. Thompson 和 D. M. Ritchie 两人合作用 C 语言改写了 UNIX。

1977 年可移植 C 语言使 C 移植到其他机器时的工作大大简化,推动了 UNIX 操作系统在各种机器上的应用。随着 UNIX 的日益广泛使用,C 语言也得到迅速推广,现在 C 语言已成为世界上应用最广泛的几种计算机语言之一。

1983 年,美国国家标准化协会(ANSI)根据 C 语言问世以来的各种版本对 C 的发展和扩充,制定了新的标准,称为 ANSI C,比原来的标准 C 有了很大的发展。1987 年 ANSI 又公布了新标准——87 ANSI C。1990 年,国际标准化组织 ISO (International Standard Organization) 接受 87 ANSI C 为 ISO C 的标准,目前流行的 C 编译系统都是以它为基础的,本书叙述基本上以 ANSI C 为基础进行讲解。

1.2.2 C 语言的特点

(1) 语言简洁、紧凑,使用方便、灵活。C 语言共有 32 个关键字,9 种控制语句,程序书写格式自由。

(2) 丰富的数据类型和运算符。C 的数据类型有整型、实型、字符型、数组类型、指针类型、结构体类型和共用体类型等,可以实现链表、树、栈等复杂的数据类型。

(3) 兼具高级语言和低级语言的优点,既可用于编写系统软件,也可用来编写应用软件。C 语言能进行位操作,能实现汇编语言的大部分功能,可以直接对硬件操作。具有双重性,既是成功的系统描述语言,又是通用的程序设计语言。

(4) C 语言是结构化程序设计语言。C 语言具有结构化程序的基本控制语句,如用于循环的 for, while, do-while 语句,用于判定的 if-else, switch 语句等,以满足结构化程序设计