

普通高等教育“十五”国家级规划教材

高职高专计算机系列教材

C 语言程序设计

徐建民 张 萍 王 苗 杨学全 编著

電子工業出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

程序设计是计算机及相关专业学生必备的基本技能之一。因此,“程序设计”课程是计算机及相关专业的一门重要的必修课。根据中国计算机学会高职高专教育学组审定的教学大纲和技能培养的基本方法,本书在介绍 C 语言基本概念和知识的基础上,重点强调了基本技能的训练。

全书共分 13 章,分别讲述了程序设计的基础知识、C 语言的基本概念、顺序结构程序设计、分支结构程序设计、循环结构程序设计、函数、指针、数组、结构体和共用体、编译预处理、位运算、文件和图形处理等内容。根据技能培养的基本要求,本书给出了比较多的例题、习题,以供学习者模仿和练习。

本书讲述力求准确、简练,强调知识的层次性,例题和习题选用讲究、丰富,强调编程技能的培养。在内容安排上,本书遵循了“难点分解”的原则,即将复杂的内容分解到相关的、不同的章节中讲述,从而减少学生学习的难度。本书既可作为计算机及相关专业本、专科学生的教材,也可以作为编程工作者,尤其是 C 语言初学者的参考书。

本书被教育部列入普通高等教育“十五”国家级规划教材。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有,侵权必究。

图书在版编目(CIP)数据

C 语言程序设计/徐建民等编著. —北京:电子工业出版社,2002.8

高职高专计算机系列教材

ISBN 7-5053-7913-5

I. C… II. 徐… III. C 语言—程序设计—高等学校:技术学校—教材 IV. TP312

中国版本图书馆 CIP 数据核字(2002)第 059884 号

责任编辑:张孟玮 特约编辑:韩玉彬

印 刷:北京李史山胶印厂

出版发行:电子工业出版社 <http://www.phei.com.cn>

北京市海淀区万寿路 173 信箱 邮编 100036

经 销:各地新华书店

开 本:787×1092 1/16 印张:17.25 字数:442 千字

版 次:2002 年 8 月第 1 版 2002 年 8 月第 1 次印刷

印 数:10 100 册 定价:20.00 元

凡购买电子工业出版社的图书,如有缺损问题,请向购买书店调换。若书店售缺,请与本社发行部联系。
联系电话:(010)68279077

出版说明

高职高专的计算机专业面临着两方面的巨大变化,一是计算机技术的飞速发展,另一方面是高职高专教育本身的改革和重组。

当前,计算机技术正经历着高速度、多媒体网络化的发展,计算机教育特别是计算机专业的教材建设必须适应这种日新月异的形势,才能培养出不同层次的合格的计算机技术专业人才。为了适应这种变化,国内外都在对计算机教育进行深入的研究和改革。美国 IEEE 和 ACM 在推出了《Computing Curricula 2000》之后,立即又推出了《Computing Curricula 2001》。全国高校计算机专业教学指导委员会和中国计算机学会教育委员会在 1999 年 9 月也提出了高等院校《计算机学科教学计划 2000》(征求意见稿)。目前,国内许多院校老师、专家正在研究《Computing Curricula 2001》,着手 21 世纪的中国计算机教育的改革。

高专层次和本科层次的计算机教育既有联系又有区别,高专层次的计算机教育旨在培养应用型人才。自 20 世纪 70 年代末高等专科学校计算机专业相继成立以来,高等专科学校积极探索具有自己特色的教学计划和配套教材。1985 年,在原电子工业部的支持下,由全国数十所高等专科学校参加成立了中国计算机学会教育委员会大专教育学组,之后又成立了大专计算机教材编委会。从 1986 年到 1999 年,在各校老师的共同努力下,已相继完成了三轮高等专科计算机教材的规划与出版工作,共出版了 78 种必修课、选修课、实验课教材,较好地解决了高专层次计算机专业的教材需求。

为了适应计算机技术的飞速发展以及高职高专计算机教育形势发展的需要,中国计算机学会教育委员会高职高专教育学组和高职高专计算机教材编委会于 2000 年 7 月开始,又组织了一批本科高校、高等专科学校、高等职业技术学院和成人教育高等院校的有教学经验的老师,学习研究参考了高等院校《计算机学科教学计划 2000》(征求意见稿),提出了按照新的计算机教育计划和教学改革的要求,编写高专、高职、成人高等教育三教统筹的第四轮教材。

第四轮教材的编写工作采取了以招标的方式征求每门课程的编写大纲和主编,要求投标老师详细说明课程改革的思路、本课程和相关课程的联系、重点和难点的处理等。在第四轮教材的编写过程中,编委会强调加强实践环节、强调三教统筹、强调理论够用为度的原则,要求教学计划、教学内容适应高等教育发展的新形势。本套教材的编者均为各院校具有丰富教学实践经验的教师。因此,第四轮教材的特点是体系结构比较合理、内容新颖、概念清晰、通俗易懂、理论联系实际、实用性强。

竭诚希望广大师生对本套教材提出批评建议。

中国计算机学会教育委员会高职高专教育学组
2001 年 1 月

先后参加中国计算机学会教育委员会高职高专教育学组和高职高专计算机教材编委会学术活动的部分学校名单

山西师范大学	天津轻工业学院
河北师范大学	浙江大学
承德石油高等专科学校	宁波高等专科学校
河北大学	福州大学
保定职业技术学院	重庆电子职业技术学院
北京科技大学	湖南大学
北京市机械工业管理局职工大学	湖南计算机高等专科学校
北方工业大学	中国保险管理干部学院
北京船舶工业管理干部学院	湖南税务高等专科学校
海淀走读大学	长沙大学
北京信息工程学院	湖南财经高等专科学校
中国人民大学	邵阳高等专科学校
北京师范大学	江汉大学
沈阳电力高等专科学校	中国地质大学
辽宁交通高等专科学校	武汉职业技术学院
吉林大学	河南职业技术学院
吉林职业师范学院	平原大学
黑龙江大学	安阳大学
哈尔滨工业大学	开封大学
哈尔滨师范大学	洛阳大学
上海理工大学	河南大学
上海第二工业大学	广州市财贸管理干部学院
上海交通大学	广东轻工职业技术学院
上海商业职业技术学院	广州航海高等专科学校
上海电机技术高等专科学校	韶关大学
上海旅游高等专科学校	佛山科学技术学院
金陵职业大学	南宁职业技术学院
南京建筑工程学院	广西水利电力职业技术学院
南京工程学院	桂林电子工业学院
南京师范大学	柳州职业技术学院
常州工学院	成都电子机械高等专科学校
无锡职业技术学院	电子科技大学
苏州市职工大学	成都师范高等专科学校
空军后勤学院	四川师范学院
连云港化工高等专科学校	云南财贸学院
泰州职业技术学院	西安电子科技大学
潍坊高等专科学校	兰州石化职业技术学院
青岛化工学院	兰州师范高等专科学校

前 言

程序设计是计算机及其相关专业学生必备的技能之一,是一门重要的必修课。“程序设计”课程不仅仅应当注重知识的讲授,还应该强调基本技能的训练。为了实现技能训练的基本目标,本书的编写主要遵循了以下原则。

1. 作者认为技能训练的过程是一个循序渐进的过程,学生程序设计技能的培养应该分三步走:掌握基本知识,然后从读实例开始,逐步达到能够编写程序的目的。相应地,本教材内容的安排与讲述就遵循这一原则。

2. 人们认识复杂事物的一个基本方法是分解,软件开发的过程就遵循了这一原则。对学生来说,一门新课相当于是一个需要接受的新事物,采用“分解与综合”的方法可以使之感觉到学习更容易。因此本教材内容的安排基本遵循“模块大小适中”的原则,即每一章、每一节乃至一个知识点都尽可能保持内容难易程度适中,将难点适当分解,便于学生掌握。

3. 软件工程的基本思想之一是规范化文档。一开始就注意培养良好的程序设计风格,对一个学生养成好的习惯十分重要,所以本书遵循的第三个原则是每一道例题程序都尽可能有良好的程序风格。

4. 学生是学习的主体,学生主动参与教学过程对学生的学习十分重要。本教材内容的安排强调学生参与教学过程,希望使用该教材的教师能注意这一点,适当安排学生一些自己思考和分析的时间。

编写一本优秀的教材是一件非常不容易的工作,涉及到诸多因素,比如内容安排的科学性、编写者的文字表达能力、作者之间的协作精神等。尽管本书三易其稿,每写一稿编者都考虑本书和其他教材相比有什么特色、有什么优点;尽管本书的定稿经过了不止一人的审阅,但编者仍感觉不尽人意,惟恐对不起孜孜求学的学子们,对不起那些关心这本教材编写工作的朋友们。由于作者水平、精力所限,再加上时间紧迫,也只好暂且这样了。

衷心感谢南京师范大学的俞光昀老师,洛阳大学的朱乃立老师,电子工业出版社的张孟玮先生。他们的勤奋与敬业使我们受益匪浅。

衷心感谢参加这本书审阅的王长仑、张晓丽等老师,他们的许多中肯意见对本书的编写质量起到了很大作用。

衷心感谢所有关心本书的师长和朋友。

编著者
2002 年 4 月

目 录

第 1 章 程序设计基础	(1)
1.1 程序设计的基本概念	(1)
1.1.1 程序	(1)
1.1.2 算法与数据结构	(1)
1.1.3 程序设计语言	(4)
1.2 程序设计方法	(5)
1.2.1 程序设计的一般步骤	(5)
1.2.2 结构化程序设计方法	(6)
1.3 程序设计风格	(6)
习题	(7)
第 2 章 C 语言程序设计初步	(9)
2.1 C 语言简介	(9)
2.1.1 C 语言的特点	(9)
2.1.2 C 程序的结构	(9)
2.1.3 C 语句的概述	(11)
2.2 常量和变量	(13)
2.2.1 常量	(13)
2.2.2 变量	(14)
2.3 简单的数据类型	(15)
2.3.1 整数类型	(15)
2.3.2 实数类型	(16)
2.3.3 字符类型	(17)
2.4 运算符及表达式	(18)
2.4.1 算术运算符与算术表达式	(18)
2.4.2 赋值运算符与赋值表达式	(20)
2.4.3 逗号运算符与逗号表达式	(22)
2.5 类型转换	(22)
2.5.1 自动类型转换	(23)
2.5.2 强制类型转换	(23)
习题	(24)
第 3 章 顺序结构程序设计	(26)
3.1 顺序结构程序设计的思想	(26)
3.2 实现顺序结构程序设计的基本语句	(26)
3.2.1 赋值语句	(26)
3.2.2 标准输入/输出	(27)

3.3 顺序结构程序设计举例	(34)
习题	(35)
第4章 分支结构程序设计	(37)
4.1 分支结构的设计思想	(37)
4.2 实现分支结构判断条件的构成	(37)
4.2.1 关系运算符与关系表达式	(37)
4.2.2 逻辑运算符与逻辑表达式	(38)
4.3 实现分支结构程序设计的语句	(40)
4.3.1 if 语句	(40)
4.3.2 switch 语句	(44)
4.4 分支结构程序设计举例	(46)
习题	(51)
第5章 循环结构程序设计	(54)
5.1 循环结构的设计思想	(54)
5.2 实现循环结构的语句	(54)
5.2.1 while 语句	(54)
5.2.2 for 语句	(56)
5.2.3 do~while 语句	(58)
5.3 循环嵌套的概念及实现	(59)
5.4 循环结构程序设计举例	(63)
5.5 其他语句	(67)
5.5.1 break (间断语句)	(67)
5.5.2 continue (接续语句)	(69)
5.5.3 goto (转向语句)	(70)
习题	(71)
第6章 函数	(76)
6.1 函数的定义	(76)
6.1.1 概述	(76)
6.1.2 函数定义的一般形式	(77)
6.1.3 函数参数和返回值	(78)
6.2 函数调用	(79)
6.2.1 函数的声明	(79)
6.2.2 函数的调用	(80)
6.2.3 函数调用的数据传递方式	(82)
6.3 函数的嵌套调用和递归调用	(83)
6.3.1 函数的嵌套调用	(83)
6.3.2 函数的递归调用	(85)
6.4 变量的作用域和存储类别	(87)
6.4.1 变量的作用域、内部变量和外部变量	(87)
6.4.2 变量的存储类别	(89)

6.4.3	内部变量的存储类别	(89)
6.4.4	外部变量的存储类别	(92)
6.5	内部函数和外部函数	(94)
6.5.1	外部函数	(94)
6.5.2	内部函数	(95)
6.5.3	举例	(95)
	习题	(96)
第 7 章	指针	(101)
7.1	概述	(101)
7.1.1	地址	(101)
7.1.2	指针	(101)
7.2	指针变量	(102)
7.2.1	指针变量的定义	(102)
7.2.2	指针变量的使用	(103)
7.3	指针和函数	(106)
7.3.1	指针变量做函数参数	(106)
7.3.2	函数返回地址值	(108)
7.4	指向函数的指针变量	(109)
7.4.1	指向函数的指针变量的定义	(109)
7.4.2	用指向函数的指针变量调用函数	(110)
7.5	应用举例	(113)
	习题	(114)
第 8 章	数组	(121)
8.1	一维数组	(121)
8.1.1	一维数组的定义	(121)
8.1.2	一维数组元素的引用	(121)
8.1.3	一维数组的初始化	(122)
8.1.4	一维数组应用举例	(122)
8.2	二维数组	(126)
8.2.1	二维数组的定义	(126)
8.2.2	二维数组元素的引用	(126)
8.2.3	二维数组的初始化	(127)
8.2.4	二维数组应用举例	(127)
8.3	字符数组	(130)
8.3.1	字符数组的定义和使用	(130)
8.3.2	字符串和字符数组	(131)
8.3.3	常用字符串处理函数	(132)
8.3.4	举例	(134)
8.4	一维数组和指针	(135)
8.4.1	一维数组名及数组元素的地址	(135)

8.4.2	指向一维数组的指针变量	(136)
8.4.3	举例	(137)
8.5	二维数组和指针	(138)
8.5.1	二维数组名及数组元素的地址	(138)
8.5.2	指向二维数组元素的指针变量	(139)
8.5.3	指向二维数组的行指针变量	(141)
8.6	字符串和指针	(143)
8.6.1	指向字符数组的指针变量	(143)
8.6.2	指向字符串常量的指针变量	(144)
8.7	向函数传递数组	(145)
8.7.1	值传递方式与地址传递方式	(145)
8.7.2	数组元素做实参	(145)
8.7.3	数组名做实参	(147)
8.8	指针数组及带参 main 函数	(149)
8.8.1	指针数组的定义和使用	(149)
8.8.2	main 函数的参数	(151)
	习题	(153)
第 9 章	结构体、共用体和枚举类型	(158)
9.1	结构体类型与结构体变量	(158)
9.1.1	结构体类型的定义	(158)
9.1.2	结构体变量的定义和初始化	(159)
9.1.3	结构体变量的引用	(160)
9.1.4	举例	(161)
9.2	结构体数组	(162)
9.2.1	结构体数组的定义、初始化	(162)
9.2.2	结构体数组的引用	(163)
9.2.3	举例	(164)
9.3	向函数传递结构体型数据	(165)
9.3.1	向函数传递结构体变量的成员	(165)
9.3.2	向函数传递结构体变量	(165)
9.3.3	向函数传递结构体变量的地址	(165)
9.3.4	向函数传递结构体数组	(167)
9.4	链表	(168)
9.4.1	链表的特点	(168)
9.4.2	创建链表	(170)
9.4.3	在链表中插入节点	(173)
9.4.4	在链表中删除节点	(174)
9.4.5	举例	(175)
9.5	共用体	(176)
9.5.1	共用体类型的定义	(176)

9.5.2	共用体变量的定义	(177)
9.5.3	共用体变量的引用	(178)
9.5.4	举例	(178)
9.6	枚举类型	(179)
9.6.1	枚举类型的定义	(179)
9.6.2	枚举类型变量的定义和使用	(180)
9.6.3	举例	(181)
9.7	用户自定义类型	(181)
9.7.1	用户自定义类型的含义和格式	(181)
9.7.2	用 typedef 声明基本类型	(181)
9.7.3	用 typedef 声明构造类型	(182)
9.7.4	用户自定义类型的应用	(183)
习题		(184)
第 10 章	编译预处理	(189)
10.1	宏定义	(189)
10.1.1	不带参数的宏定义	(189)
10.1.2	带参数的宏定义	(190)
10.1.3	终止宏定义	(191)
10.2	文件包含	(192)
10.3	条件编译	(194)
习题		(195)
第 11 章	位运算	(198)
11.1	位运算和位运算符	(198)
11.1.1	“按位取反”运算符~	(198)
11.1.2	“按位与”运算符&	(199)
11.1.3	“按位或”运算符 	(200)
11.1.4	“按位异或”运算符^	(201)
11.1.5	左移运算符<<	(202)
11.1.6	右移运算符>>	(203)
11.1.7	位复合赋值运算符	(203)
11.2	位段	(203)
11.3	应用举例	(207)
习题		(210)
第 12 章	文件	(213)
12.1	C 文件的基础知识	(213)
12.1.1	C 文件的基本格式	(213)
12.1.2	缓冲文件和非缓冲文件系统	(213)
12.1.3	C 文件操作的一般方法	(214)
12.2	文件类型指针	(214)
12.2.1	文件类型	(214)

12.2.2 文件类型指针	(215)
12.3 文件的打开与关闭	(215)
12.3.1 文件的打开 (fopen) 函数	(215)
12.3.2 文件的关闭 (fclose) 函数	(216)
12.4 文件的读写	(217)
12.4.1 字符读写函数	(217)
12.4.2 数据块读写函数	(218)
12.4.3 格式化读写函数	(221)
12.4.4 字读写函数	(221)
12.4.5 字符串读写函数	(222)
12.4.6 读写其他类型数据	(223)
12.5 文件的定位	(223)
12.5.1 fseek 函数	(223)
12.5.2 rewind 函数	(224)
12.5.3 ftell 函数	(224)
12.6 检测函数	(224)
12.6.1 ferror 函数	(224)
12.6.2 clearerr 函数	(225)
12.6.3 feof 函数	(225)
12.7 应用举例	(225)
习题	(228)
第 13 章 图形处理	(234)
13.1 基本概念	(234)
13.1.1 坐标系	(234)
13.1.2 图形模式	(235)
13.2 图形函数	(236)
13.2.1 图形系统控制函数	(236)
13.2.2 画图和填充函数	(237)
13.2.3 颜色控制函数	(239)
13.2.4 屏幕和图形窗口管理函数	(240)
13.2.5 图形方式下的文本输出函数	(240)
13.2.6 图形存取函数	(241)
13.2.7 错误处理	(242)
13.3 应用举例	(242)
附录 A ASCII 码表	(246)
附录 B C 语言中的关键字	(247)
附录 C 运算符和结合性	(248)
附录 D C 库函数	(250)
附录 E 简单的上机操作和程序的调试	(256)
参考文献	(263)

第 1 章 程序设计基础

1.1 程序设计的基本概念

1.1.1 程序

我们做任何事情都遵循着一定的程序，如开会的议程、做饭的食谱、演奏的乐谱等都是程序，在程序的指导下，人们可以有秩序地、有效地完成一项工作。随着计算机的问世和普及，“程序”逐渐被专业化，它通常特指：为让计算机完成特定任务（如解决某一算题或控制某一过程）而设计的指令序列。

程序是让计算机运行的。程序就是对解决问题所需要的数据的描述和对数据处理的描述。对数据的描述称之为“数据结构”，对数据处理的描述称之为“算法”，而编写程序所用的计算机语言，称之为“程序设计语言”。

1.1.2 算法与数据结构

著名计算机科学家沃斯（N.Wirth）提出过一个经典公式：

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

算法反映了计算机的执行过程，是对解决特定问题的操作步骤的一种描述。数据结构是对参与运算的数据及它们之间的关系进行的描述，算法和数据结构是程序的两个重要方面。

1. 算法

【例 1.1】 输入 3 个数，求其最大值。

问题分析：设 num1, num2, num3 存放 3 个数，max 存放其最大值。

为求其最大值，就必须对 3 个数进行比较，可按如下步骤去做：

- ① 输入 3 个数 num1, num2, num3;
- ② 先把第 1 个数 num1 的值赋给 max;
- ③ 将第 2 个数 num2 与 max 比较，如果 num2 > max，则把第 2 个数 num2 的值赋给 max（否则不做任何工作）；
- ④ 将第 3 个数 num3 与 max 比较，如果 num3 > max，则把第 3 个数 num3 的值赋给 max（否则不做任何工作）；
- ⑤ 输出 max 的值，即最大值。

从例 1.1 可以看出，首先分析题目，然后寻找一种实现这个问题所要完成功能的方法，这种方法的具体化就称为算法。因此可以说，算法是由一套明确的规则组成的一些步骤，它指定了操作顺序并通过有限个步骤将问题解决、得出结果。

一个算法应具有以下 5 个特性：

(1) 有穷性

一个算法必须总是在执行有限个操作步骤和可以接受的时间内完成其执行过程。也即

是说，对于一个算法，要求其在时间和空间上均是有穷的。例如：一个采集气象数据并加以计算进行天气预报的应用程序，如果不能及时得到结果，超出了可以接受的时间，就起不到天气预报的作用。

(2) 确定性

算法中的每一步都必须有明确的含义，不允许存在二义性。例如：“将成绩优秀的同学名单打印输出”，在这一描述中“成绩优秀”就很不明确，是每门功课均为 95 分以上？还是指总成绩在多少分以上？

(3) 有效性

算法中描述的每一步操作都应该能有效地执行，并最终得到确定的结果。例如：当 $Y=0$ 时， X/Y 是不能有效执行的。

(4) 输入

一个算法有零个或多个输入数据。例如：计算 1~10 的累计和的算法，是无须输入数据，而对 10 个数据进行排序的算法，却需要从键盘上输入这 10 个数据。

(5) 输出

一个算法应该有 1 个或多个输出数据。执行算法的目的是为了求解，而“解”就是输出，因此没有输出的算法是毫无意义的。

2. 算法的表示方法

算法的表示方法很多，常见的有：自然语言、传统流程图、N-S 结构图、伪代码、PAD 图等。

(1) 用自然语言表示

自然语言就是人们日常使用的语言，可以是中文、英文等。用自然语言表示的算法通俗易懂，但一般篇幅冗长，表达上往往不易准确，容易引起理解上的“歧义性”。所以，一般用于算法较简单的情况，如例 1.1 所示。

(2) 用传统流程图表示

传统流程图是用规定的一组图形符号、流程线和文字说明来表示各种操作算法的表示方法。传统流程图常用的符号如表 1.1 所示。

表 1.1 传统流程图常用符号

符 号	符 号 名 称	含 义
	起止框	表示算法的开始和结束
	输入/输出框	表示输入/输出操作
	处理框	表示对框内的内容进行处理
	判断框	表示对框内的条件进行判断
	流程线	表示流程的方向
	连接点	表示两个具有同一标记的“连接点”应连接成一个点

【例 1.2】 用传统流程图描述算法如图 1.1 所示。

用传统流程图表示算法直观形象，算法的逻辑流程一目了然，便于理解。但占用篇幅较大，画起来比较麻烦，而且又由于允许使用流程线，使用者可以随心所欲，使流程可以任意转移，从而造成阅读和修改上的困难。

(3) 用 N-S 结构图表示

针对传统流程图存在的问题，美国学者 I.Nassi 和 B.Shneiderman 于 1973 年提出一种新的结构化流程图形式，即简称为 N-S 结构图。Chapin 在 1974 年对其进行了进一步的扩展，因此，N-S 结构图又称为 Chapin 图或盒状图。

N-S 结构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。其主要特点是完全取消了流程线，不允许有随意的控制流，全部算法写在一个矩形框内，该矩形框以 3 种基本结构（顺序、分支、循环）描述符号为基础复合而成。

N-S 结构图表示的 3 种基本结构如下：

① 顺序结构。顺序结构是最简单的基本结构。在顺序结构中，要求顺序地执行且必须执行由先后顺序排列的每一个最基本的处理单位。图 1.2 (a) 所示是用传统流程图表示的顺序结构，

图 1.2 (b) 所示是用 N-S 结构图表示的顺序结构，先执行处理 A，然后再顺序执行处理 B。

② 分支结构。分支结构又称做选择结构。在分支结构中，要根据逻辑条件的成立与否，分别选择执行不同的处理。如图 1.3 所示，当逻辑条件成立时，执行处理 A，否则执行处理 B。

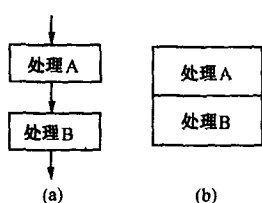


图 1.2 顺序结构

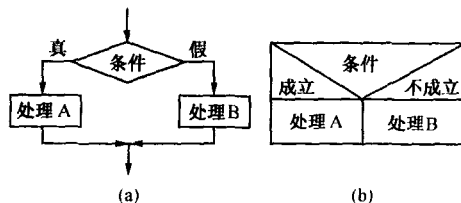


图 1.3 分支结构

③ 循环结构。循环结构一般分为当型循环和直到型循环。

(a) 当型循环。在当型循环结构中，当逻辑条件成立时，就反复执行处理 A（称为循环体），直到逻辑条件不成立时结束，如图 1.4 所示。

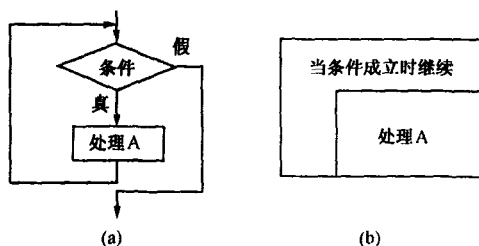


图 1.4 当型循环结构

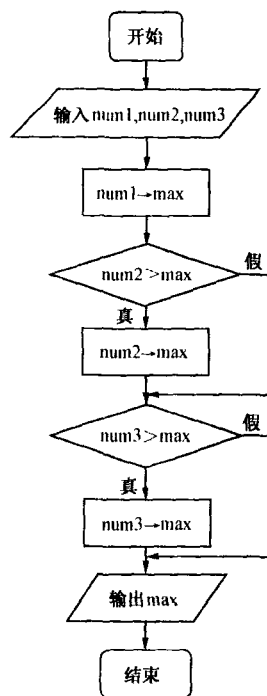


图 1.1

(b) 直到型循环。在直到型循环结构中，反复执行处理 A，直到逻辑条件成立时结束（即逻辑条件不成立时继续执行），如图 1.5 所示。

【例 1.3】 用 N-S 结构图描述的例 1.1，求 3 个数中的最大值的算法，如图 1.6 所示。

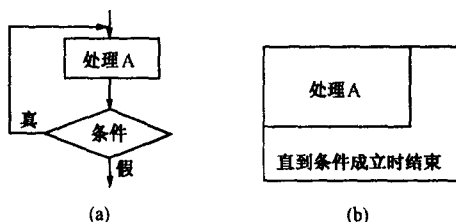


图 1.5 直到型循环结构

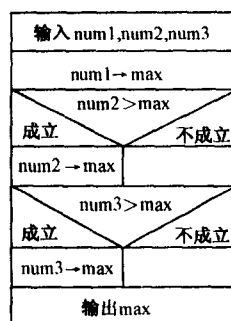


图 1.6

(4) 用伪代码表示

伪代码是用一种介于自然语言和计算机语言之间的文字和符号来描述算法。伪代码的表现形式比较灵活自由，没有严谨的语法格式。

【例 1.4】 用伪代码描述例 1.1，求 3 个数中的最大值的算法。

```
input num1,num2,num3
num1 → max
if num2 > max then num2 → max
if num3 > max then num3 → max
print max
```

理论已经证明，任何复杂的算法均可以用顺序、分支、循环这 3 种基本结构组合、嵌套进行描述。由于 N-S 结构图无箭头指向，而局限在一个个嵌套的框中，最后描述的结果必然是结构化的，因此 N-S 结构图描述表示的算法，适用于结构化程序设计（有关结构化程序设计的概念将在后面介绍），本书将主要采用 N-S 结构图表示算法。

3. 数据结构

计算机处理的对象是数据，数据是描述客观事物的数、字符以及所有计算机能够接受和处理的信息符号的总称。数据结构是指数据的类型和数据的组织形式。数据类型体现了数据的取值范围和合法的运算，数据的组织形式体现了相关数据之间的关系。

数据结构与算法有着密切的关系，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。事实上，程序就是在数据的某些特定的表示方式和结构的基础上对抽象算法的具体表述。因此，编写一个程序的关键就是合理地组织数据和设计好的算法。

1.1.3 程序设计语言

程序设计语言是人们用来编写程序，为计算机能够接受并理解的计算机语言。它通常是一个能够完整准确地表达人的意图，并赖以控制计算机实现给定运算的符号系统。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 3 大类：面向机器的语言、面向过程的语言和面向对象的语言。

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，用这类语言进行程序设计时，必须用语言的语句、函数、命令描述一步一步地解决问题的步骤，即告诉计算机“如何做”。这类语言独立于计算机，如：Pascal 语言、C 语言等。

3. 面向对象的语言

面向对象的语言适于解决大型的任务。用这类语言设计程序时，只需告诉计算机“做什么”，而不用说明“怎么做”。这类语言也独立于计算机，如 C++ 等。

程序是用程序设计语言编写的，不同的程序设计语言有其不同的特点。因此，在开始编程之前要根据实际情况选择适合本系统的程序设计语言。通常可以按照如下原则选择编程语言：

① 程序设计语言是否能充分描述软件的处理功能，其中包括算法的复杂度、数据结构的要求、运行效率和性能要求等；

② 程序设计语言的适用性，其中包括语言的复杂程度及程序员和用户对这种语言的熟悉情况等。

1.2 程序设计方法

1.2.1 程序设计的一般步骤

程序设计就是针对给定问题进行设计、编写和调试计算机程序的过程。

程序设计是一种构造性技术。作为一名程序设计者，要想设计好一个程序，不仅要掌握程序设计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，并通过程序设计的实践，不断地发现总结其规律性东西，从而进一步提高程序设计的能力。

一般程序设计方法步骤如下：

① 分析问题，确定解题方案。首先根据用户的具体要求进行：用户需求分析（详细而具体地理解用户要解决的问题）；数据及处理分析（已知或需要的输入数据、需要输出的数据、需要进行的处理）；可行性分析（用户提出的问题是是否可解？可解的价值如何？）；运行环境分析（硬件环境和软件环境分析）。

然后在以上分析的基础上，将实际问题抽象化建立相应的数学模型，并确定解题方案。

② 确定算法。根据选取的数学模型和确定的解题方案，设计出具体的操作步骤，并可以通过流程图将确定的算法清晰、直观地表示出来。

③ 编写程序。选用较为适宜的程序设计语言，以书面形式将算法描述出来，即形成用程序设计语言编制的源程序。