

湖南省高等院校统编教材 湖南省九五重点建设教材

C 语言程序设计与分析教程

主编：郭浩志 副主编：林亚平

湖南科学技术出版社



Yuyan Chengxu Sheji Yu Fenxi Jiaocheng

湖南省高等院校统编教材
湖南省九五重点建设教材

C 语言程序设计与分析教程

TP 312/57

顾问：李洪基
主编：郭浩志
副主编：林亚平
编委：(以姓氏笔画为序)
陈洛资 周大庆 林亚平 郝三如
郭浩志 谢深泉 蒋大文 曾三槐
组织策划：蒋易春 易建华 王文斌 刘少波

园林系园林专业三班

李玉良

湖南科学技术出版社

1998.9.12.

湖南省高等院校统编教材 湖南省九五重点建设教材

C 语言程序设计与分析教程

主 编：郭浩志

副 主 编：林亚平

责任编辑：胡海清

出版发行：湖南科学技术出版社

社 址：长沙市展览馆路 66 号

印 刷：湖南省新华印刷一厂

厂 址：长沙市芙蓉北路 564 号

邮 编：410008

(印装质量问题请直接与本厂联系)

出版日期：1998 年 7 月第 1 版第 1 次

开 本：787mm×1092mm 1/16

印 张：26.25

字 数：592000

印 数：1—10100

书 号：ISBN 7-5357-2467-1/TP·112

定 价：20.00 元

(版权所有·翻印必究)

内 容 简 介

本书为湖南省普通高校非计算机专业C语言程序设计课程的全省统编教材，湖南省“九五”重点建设教材。它是在湖南省教委高教处的领导与组织下，由国防科技大学等八所大学的教授编写而成的。全书包括三篇。第1篇为基础篇，重点介绍与语言、程序有关的基础概念，以及用C语言编写程序的预备知识；第2篇为程序设计与分析篇，详细阐述了十种程序设计和分析的基本方法与技巧；第3篇是程序开发与调试篇，讨论了开发与调试Turbo C程序的方法、工具与环境。各章均有丰富的练习与实习。全书遵循全国、全省有关大纲，精选了典型数据结构和算法，强化编写程序、阅读程序和调试程序，并安排了课程设计。

本书是大学公共基础课，不仅适用于众多非计算机专业，也适用于计算机专业；不仅可供教师参考，也可供企事业单位管理人员和科技人员使用。

基 础 章 : 何 福

志 得 源 : 姜 主

平 亚 林 : 姜 主 编

(有 关 画 像 刀 裁 剪) : 姜 姜

成 三 强 平 亚 林 刘 大 刚 黄 杏 洲

刘 三 曾 文 大 群 泉 菊 衡 志 得 源

黄 少 欣 郑 文 王 李 慧 晨 姜 得 源 : 版 策 编 排

因 林 系 因 林 系 三 册

并 编 出 本 对 学 将 南 湖

李 亚 平

51.8.8/14

本书为湖南省普通高校非计算机专业C语言程序设计的全省统编教材,“九五”重点建设教材。它是在湖南省教委高教处的领导与组织下,由国防科技大学等八所高等院校的教授依照国家教委面向21世纪教学内容和课程体系改革的精神及1998年《湖南省普通高校非计算机专业学生计算机应用水平等级考试大纲》的要求编写而成的。主编是郭浩志教授,副主编是林亚平教授。

编写本书的指导思想主要有三点:

1. 加强大学生计算机素质教育。本课程属于计算机基础教学第二层次的范畴,主要训练学生利用计算机处理问题的思维方式与工作方式,这是21世纪人材知识、能力和素质结构中不可缺少的重要组成部分,是当今高等教育的公共基础课。

2. 培养各专业复合型人材的计算机软件技术基础。围绕C语言这一当今使用最为广泛的人机交互工具,使学生掌握计算机软件技术中有关程序的基础知识、程序设计与分析方法,以及用计算机解决主专业问题的能力。

3. 与等级考试接轨。本书内容力求覆盖等级考试中关于程序语言以及程序设计、分析和调试能力的要求。

全书共分三篇。第1篇为基础篇,重点介绍与语言、程序有关的基础概念,C语言的主要特点、简史、用处以及用C语言编写程序的预备知识;第2篇为程序设计与分析篇,详细阐述了十种程序设计和分析的基本方法和技巧;第3篇是程序开发与调试篇,主要以Turbo C为背景,讨论了C语言程序开发、调试的方法、工具与环境。

本书主要特点如下:

1. 精选了常见的数据结构与典型算法;
2. 为强化编写程序和阅读程序的训练,第2篇各章均有代表性示例分析编写程序的思路和方法,解剖程序结构和功能;
3. 为加强动手能力,多数篇章给出了内容和形式都比较丰富的、有针对性的练习题及实习示例题;
4. 突出程序调试训练。第3篇专门讨论了有关的程序排错方法、调试工具和

常见错误分析；

5. 安排了小型课题（约几百行）的课程设计，为进一步学习软件开发打下初步的基础；

6. 与等级考试接轨。凡参加一、二级考试者均可适用。

建议本科学时数为 60~80 学时，大专学时数为 50~70 学时。凡标有 ‘*’ 的章节专供本科和参加二级考试使用。

参加本书编写的人员有：国防科技大学郭浩志（前言，第 2 篇第六、七、八章），湖南大学林亚平（第 1 篇第一、二章）和李军义（第 2 篇第三、四、五章），湖南财经学院周大庆（第 2 篇第九、十、十一、十二章）以及长沙铁道学院陈洛资（第 3 篇第十三、十四、十五章 15.1 节）、黄汉永（15.2 节）和杨彦（15.3 节）。全书由郭浩志统编。蒋易春、易建华、王文斌、刘少波和胡春晖等做了大量组织工作。郭旭东为本书很多示例做了调试工作。本书撰写过程中，得到了中南工业大学、长沙铁道学院、湖南大学、湖南师范大学、湖南农业大学、湖南财经学院、湘潭大学、国防科技大学以及其他很多高等院校的关心和支持。湖南科学技术出版社，尤其是胡海清编审为本书的出版付出了辛勤的劳动。在此一一表示衷心的感谢！

虽然本书大多数内容已在若干所大学试用，并反复修改过，但是由于编写时间仓促，有些院校近年教学改革的好经验，可能未来得及融入本书之中。加上编者水平和经验有限，书中错误、缺点在所难免，热忱欢迎读者提出宝贵的意见和建议。

编者

1998 年 5 月

目录

CONTENTS

第1篇 基础	1
第一章 高级语言程序设计基础	1
1.1 高级语言程序设计概述	1
1.2 算法与数据结构	2
1.2.1 算法	2
1.2.2 数据结构	3
1.3 程序设计基础知识	3
1.3.1 算法描述工具	4
1.3.2 结构程序设计	6
1.3.3 程序设计风格	7
1.4 练习与实习	9
1.4.1 练习	9
1.4.2 实习	9
第二章 C语言概述	10
2.1 C语言历史及发展概况	10
2.2 C语言的特点	10
2.3 C语言开发环境和程序结构	11
2.3.1 C语言开发环境	11
2.3.2 C语言程序的基本构造	12
2.4 练习与实习	14
2.4.1 练习	14
2.4.2 实习	14
第2篇 程序设计与分析	16
第三章 简单数据类型与顺序程序设计	16
3.1 常量与变量	16
3.1.1 常量	16
3.1.2 变量	17
3.2 字符型数据	17
3.2.1 字符常量	17
3.2.2 字符变量	18
3.2.3 字符串常量	19
3.3 整型数据	19

3.3.1 整型变量	19
3.3.2 整型常量	20
3.4 实型数据	21
3.4.1 实型变量	21
3.4.2 实型常量	21
3.5 数据类型的转换	21
3.5.1 数据类型的自动转换	21
3.5.2 强制转换数据类型	22
3.6 运算符和表达式	23
3.6.1 运算符简介	23
3.6.2 算术运算符及算术赋值运算符	23
3.6.3 赋值运算符和赋值表达式	24
3.6.4 逗号运算符与逗号表达式	26
3.6.5 运算符的优先级和结合规则	26
3.7 简单输入输出函数	27
3.7.1 输入函数	27
3.7.2 输出函数	29
3.8 顺序程序设计	32
3.8.1 C 语句分类	32
3.8.2 顺序程序设计	33
3.9 程序设计与分析示例	34
3.9.1 程序设计示例	34
3.9.2 程序分析示例	35
3.10 练习与实习	36
3.10.1 练习	36
3.10.2 实习	37
第四章 分支程序设计	38
4.1 关系及逻辑运算	38
4.1.1 关系运算符及关系表达式	38
4.1.2 逻辑运算符及逻辑表达式	38
4.2 if 语句	39
4.2.1 双分支选择 if-else 语句	39
4.2.2 单分支选择 if 语句	41
4.2.3 if 语句的嵌套	41
4.2.4 条件运算表达式	43
4.3 switch 语句	43
4.4 程序设计与分析示例	47
4.4.1 程序设计示例	47
4.4.2 程序分析示例	50
4.5 练习与实习	51
4.5.1 练习	51
4.5.2 实习	53
第五章 重复执行程序设计	54
5.1 概述	54

5.2	while 语句	54
5.3	do~while 语句	56
5.4	for 语句	58
5.5	循环的嵌套	60
5.6	循环的中途退出	62
5.6.1	break 语句	62
5.6.2	continue 语句	63
5.7	goto 语句及语句标号	64
5.8	程序设计与分析示例	65
5.8.1	程序设计示例	65
5.8.2	程序分析示例	66
5.9	练习与实习	68
5.9.1	练习	68
5.9.2	实习	69
第六章	数组类型与进一步的循环程序设计	72
6.1	一维数组	72
6.1.1	一维数组的定义与引用	72
6.1.2	初始化	73
6.2	一维字符数组	74
6.3	多维数组	76
6.4	程序设计与分析示例	79
6.4.1	程序设计示例	79
6.4.2	程序分析示例	84
6.5	练习与实习	89
6.5.1	练习	89
6.5.2	实习	94
第七章	函数与模块程序设计	99
7.1	函数定义	99
7.1.1	用户自定义函数	99
7.1.2	main () 函数	101
7.1.3	函数说明	101
7.1.4	void 类型	102
7.2	函数调用	102
7.2.1	函数的一般调用	102
7.2.2	参数的传递机制	103
7.2.3	函数值的返回	104
7.2.4	函数的嵌套调用	105
7.2.5	将数组用作函数的参数	106
7.3	局部变量与全局变量	109
7.3.1	局部变量	109
7.3.2	全局变量	110
* 7.4	递归函数	111
7.4.1	直接递归与间接递归	111
7.4.2	递归与递推	111

100	7.4.3 递归函数的执行特征	113
002	7.4.4 递归调用与嵌套调用	117
002	7.5 程序结构	117
000	7.5.1 C 语言源程序结构	117
000	7.5.2 变量的属性	117
000	7.5.3 自动型变量	118
000	7.5.4 静态型变量	119
100	7.5.5 寄存器型变量	120
000	7.5.6 外部型变量	121
000	7.5.7 分别编译	122
000	7.5.8 变量定义与变量说明	123
000	7.5.9 变量的存储类型	123
000	7.5.10 函数的存储类型	124
000	7.6 标准函数初步	124
000	7.7 适于 C 语言的程序设计方法	125
000	7.7.1 程序设计方法概述	125
000	7.7.2 结构程序设计方法 (SP 方法)	125
000	7.8 程序设计与分析示例	128
000	7.8.1 程序设计示例	128
000	7.8.2 程序分析示例	134
000	7.9 练习与实习	138
000	7.9.1 练习	138
000	7.9.2 实习	144
000	第八章 编译预处理的程序设计	151
000	8.1 预处理	151
000	8.2 宏定义	151
000	8.2.1 简单宏定义	151
000	8.2.2 带参数的宏定义	154
000	8.3 文件包含	156
100	8.4 条件编译	158
100	8.5 程序设计与分析示例	160
001	8.5.1 程序设计示例	160
001	8.5.2 程序分析示例	164
001	8.6 练习与实习	167
001	8.6.1 练习	167
001	8.6.2 实习	172
000	第九章 指针类型与指针程序设计	175
001	9.1 指针的概念	175
001	9.1.1 指针变量	175
001	9.1.2 指针的说明与初始化	176
001	9.1.3 与指针有关的运算符 & 和 *	176
101	9.2 指针的运算	177
101	9.2.1 赋值运算	177
101	9.2.2 算术运算	177

9.2.3	关系运算	178
9.3	指针与数组	178
9.3.1	指针与一维数组	178
9.3.2	指针与二维数组	180
9.4	字符指针与字符串	182
9.5	指针与函数	184
9.5.1	指针作为函数参数	184
9.5.2	返回指针值的函数	186
9.5.3	指向函数的指针	187
9.6	指针数组	190
9.6.1	指针数组的说明	190
9.6.2	用指针数组处理多维数组	190
9.6.3	用字符指针数组处理多个字符串	191
9.6.4	函数指针数组	193
* 9.7	指针的指针	194
* 9.8	动态存储分配与动态数组	195
9.8.1	动态存储分配	195
9.8.2	动态数组	196
9.9	命令行参数	198
9.10	程序设计与分析示例	199
9.11	练习与实习	205
9.11.1	练习	205
9.11.2	实习	208
第十章	结构、联合、枚举及其程序设计	211
10.1	结构的概念	211
10.1.1	结构类型的定义	211
10.1.2	结构变量的说明	212
10.1.3	结构变量的初始化	213
10.1.4	结构变量的引用	213
10.2	结构数组	215
* 10.3	指向结构的指针	216
10.3.1	结构指针	216
10.3.2	在函数之间传递结构	217
* 10.4	嵌套结构	219
* 10.5	链表	220
* 10.6	联合	223
* 10.7	枚举类型	224
* 10.8	类型定义	225
* 10.9	程序设计与分析示例	227
10.10	练习与实习	237
10.10.1	练习	237
10.10.2	实习	240
* 第十一章	位运算与面向机器的程序设计	245
11.1	位运算	245

851	11.1.1 按位与运算 (&)	245
851	11.1.2 按位或运算 ()	246
851	11.1.3 按位异或运算 (^)	246
081	11.1.4 按位取反运算 (~)	249
381	11.1.5 右移运算 (>>)	249
181	11.1.6 左移运算 (<<)	250
481	11.1.7 位操作赋值运算	252
381	11.2 位段	252
781	11.2.1 位段的说明	252
091	11.2.2 位段的引用	252
001	11.2.3 位段应用举例	254
001	11.3 程序设计与分析示例	255
101	11.4 练习与实习	258
001	11.4.1 练习	261
401	11.4.2 实习	261
20	第十二章 文件与输入输出程序设计	263
001	12.1 文件的概念	263
001	12.1.1 文本文件与二进制文件	263
801	12.1.2 缓冲型与非缓冲型文件系统	263
001	12.1.3 顺序存取与随机存取	263
203	12.2 缓冲型文件系统	264
203	12.2.1 打开与关闭文件	264
803	12.2.2 字符输入/输出函数	265
113	12.2.3 字符串输入/输出函数	269
113	12.2.4 格式化输入/输出函数	270
113	12.2.5 数据块输入/输出函数	271
313	12.3 文件定位与随机存取	273
813*	12.4 非缓冲型文件系统	274
313	12.5 标准设备文件	275
313	12.6 程序设计与分析示例	276
313	12.7 练习与实习	278
313	12.7.1 练习	278
713	12.7.2 实习	279
	第3篇 程序开发与调试	283
03	第十三章 程序开发	283
333	13.1 程序开发环境	284
133	13.1.1 程序开发的基本要素	284
333	13.1.2 编译模式	285
733	13.1.3 Turbo C 集成开发环境简介	291
733	13.2 Turbo C2.0 的集成开发环境 TC	291
733	13.2.1 Turbo C2.0 的安装	291
043	13.2.2 引导 Turbo C2.0 集成开发环境	293
343	13.2.3 Turbo C2.0 的热键	295
343	13.2.4 Turbo C2.0 子菜单功能	296

13.2.5 Turbo C2.0 的 TC 编辑器 Edit	298
13.3 在集成开发环境 TC 下开发 C 语言源程序的主要过程	301
13.3.1 加工、运行一个单个的源程序文件	301
13.3.2 编辑一个 C 语言源程序	303
* 13.3.3 使用多个源程序文件	304
* 13.4 Turbo C++ IDE 简介	305
13.5 程序开发方法与实用工具软件	309
13.5.1 程序开发方法	309
* 13.5.2 命令行编译器 TCC	310
13.5.3 Turbo C 的实用工具软件	315
* 13.5.4 Turbo C 的连接程序 TLINK	316
* 13.5.5 MAKE	319
* 13.5.6 Turbo 库管理程序 TLIB	331
第十四章 程序调试	332
14.1 程序开发与程序调试	333
14.2 集成调试工具	339
14.3 跟踪语法错误	340
* 14.4 跟踪运行错误	346
14.5 常见错误分析	352
第十五章 课程设计	352
15.1 课程设计的目的与要求	352
15.2 课程设计之一——分栏打印程序	352
15.2.1 问题描述	352
15.2.2 设计要求	356
* 15.3 课程设计之二——上机考试模拟系统	356
15.3.1 问题描述	356
15.3.2 设计要求	369
15.3.3 课程设计报告格式	370
附录 A ASCII 字符及其编码	371
附录 B 关键字与预编译符	373
附录 C C 语言常用语法表	373
附录 D 常见库函数	378
附录 E 常用编辑命令表	385
附录 F 常见编译错误信息	387
附录 G 本书各章主要示例索引	399
主要参考资料	405

第 1 篇 基 础

第一章 高级语言程序设计基础

程序设计语言主要经历了机器语言、汇编语言和高级语言这三代语言的演变。机器语言和汇编语言是面向机器的低级语言，编制程序十分不便。传统的高级语言是面向过程的语言，其优越性是提高了编程人员的工作效率，比较低级语言，用高级语言编制程序更容易编写、调试和阅读理解。由于不依赖机器，在某种机器或某种操作系统下用高级语言编制的程序，容易移植到其他机器或操作系统下执行，也就是说高级语言编制的程序具有很好的可移植性。现代计算机技术的发展，又导致了使用更为方便的第四代语言的出现，但从严格意义上说，第四代语言与前述的三代语言有着明显的差别，它属于一类自动化程度比较高，且通常是面向某些专门应用的开发工具或开发环境，如数据库查询语言，用于 Web 网页制作的 HTML 语言（超媒体语言），以及诸如报表、屏幕格式等应用生成程序。前三代语言在程序设计上更讲究方法和技巧，强调的是如何做程序；而第四代语言强调的是自动化地做什么应用。

综上所述，为了理解程序设计的概念，设计方法和技巧，乃至理解各种形式计算机软件（包含第四代语言）的使用，学习高级语言程序设计是十分必要的。因此，高级语言程序设计也属于计算机基础教育中的一种基本训练。本章主要讲述高级语言程序设计的基本知识。

1.1 高级语言程序设计概述

程序设计语言是人们为了描述计算过程而定义的一组有一定语法和语义规则的记号。语法表示程序的结构或形式，它描述了实现语言基本记号的组合规则，用户如果违背了这些规则，编译程序时便会出错。语义表示的是程序的含义，它描述了语言记号组合所表示的特定含义，语义的解释与语言本身及编译程序相关，而与用户使用一般无关。

在高级语言发展初期，流行的语言主要有 Fortran、Algo60 和 Cobol 语言。Fortran 语言是第一个问世的高级语言，现在 Fortran 语言仍然流行，目前常用的版本是 Fortran77 和 Fortran90；Algo60 语言现在已不流行，但它对后续的一些语言如 Pascal 等有着重要影响；Cobol 语言在数据库技术出现以前，广泛用于商业事务处理，它对数据库技术的发展有着重要作用。继 Fortran 之后高级语言发展迅速，出现了数百种语言，其中 Basic 语言以其简单易学而拥有最为广泛的用户群。

结构程序在程序设计语言发展过程中具有重要意义，1968 年 E. W. Dijkstra 指出高级语言中使用转向语句带来的问题，导致结构程序设计这一新的程序设计方法问世。

E. W. Dijkstra 证明, 任何程序都可以只用顺序、选择和循环这三种结构的语句构造, 因此可以避免使用转向语句带来的程序阅读性差和其他问题。在这一时期出现了以 Pascal 为代表的结构程序设计语言。尽管目前大多数高级语言都保留了转向语句, 但其功能和形式都变得简单, 一般只用于辅助其他的控制结构。不少语言在控制结构上提供了丰富的顺序, 选择和循环语句, 如 Pascal, C 等语言。新出现的 Basic 语言版本 (如 True Basic 等) 也在控制结构上进行了扩充, 提供了多种循环控制语句。语言的控制结构加上自顶向下, 逐步求精的模块化程序设计方法构成结构程序设计的框架。结构程序设计的思想也扩充到现代软件工程。

目前比较流行, 有代表性且广泛用于教学的传统高级语言主要有 Basic, C, Fortran 和 Pascal 等。比较而言, Basic 语言简单易学, 适合初学者; Fortran 语言注重效率, 适合科学计算; Pascal 语言精致简明, 可阅读性好, 适合用于教学和描述算法; 本书介绍的 C 语言则十分灵活, 是目前最为流行的系统软件与应用软件开发程序语言之一。

如前所述, 传统高级语言是一种面向过程的程序语言, 或称强制性语言。在高级语言发展过程中, 人们也开始注意发展其他风格, 其他范型的语言。这些语言主要可以分为函数型, 逻辑型和面向对象型语言。函数型语言因为其形式很类似数学上的函数而得名。典型的例子有 Lisp 语言, 它主要用于人工智能处理等方面; 逻辑型语言以逻辑程序设计思想为理论基础, 利用逻辑规则和推理机制产生结果, 最有代表性的逻辑语言是 Prolog 语言, 它主要用于人工智能领域如专家系统方面的程序设计; 面向对象高级程序语言的出现是程序语言的一次重大变革。它与传统过程性语言的主要区别在于: 在传统过程性语言中把数据以及处理它们的子程序当作互不相关的成分分别加以处理, 而在面向对象语言中则把这两者统一作为对象封装在一起进行处理。面向对象的思想充分体现了软件工程要求的模块性, 信息隐蔽性和抽象性。最早的面向对象程序设计语言是 Smalltalk, 该名字的本义就是“少说话”, 意指只要很少的语句就可以做许多工作。继 Smalltalk 之后, 许多面向对象的程序语言或系统先后出现, 如 C++, VB, Java 等有代表性的语言。面向对象技术已成为目前计算机软件的主要开发技术。

用编辑程序编制的程序称之为源程序, 源程序必须翻译成机器指令才能在机器上执行。根据翻译的方式不同, 可将高级语言分成编译型和解释型。大多数高级程序属于编译型, 即源程序通过编译生成目标程序, 然后经连接程序 (Linker) 生成可执行程序; 解释型高级语言是对程序语句一边翻译一边执行, Basic 语言支持解释程序。近年来出现的 Java 语言具有另一特点, Java 源程序经过编译后生成的代码是一种与机器无关的虚目标代码, 这种代码在不同的机器上再通过与机器相关的解释程序执行。Java 这种编译和解释相结合的翻译方式, 不存在程序移植问题, 因此特别适合采用浏览器/Web 服务器这种跨平台方式的 Web 应用。

1.2 算法与数据结构

程序设计中一个有名的公式是: 程序 = 算法 + 数据结构, 其基本意思是指编一个好的程序取决于选用好的算法和设计有效的数据结构。算法决定了程序解题时的运行时间复杂性和存储空间复杂性, 而数据结构的设计则是为实现算法的程序存取数据更为方便和有效。因此, 了解算法和数据结构的基本概念对程序设计十分有益。

1.2.1 算法

为求解一个问题而采用的方法和解题步骤称为算法。求解同一问题可能存在不同的算法, 因此存在算法的选择问题。例如, 判定某不小于 3 的自然数 n 是不是素数, 可以用 $[2, n-1]$

1] 间的自然数依次去除 n 而得到答案; 但也可以只用 2 到 $\sqrt{n}$ 之间的自然数依此去除 n , 就可得到答案。显然, 第二种算法比第一种算法解题更快。在第一种算法中, 为确定 n 是素数, 必须做 $n-2$ 次除法, 其时间复杂性与 n 属于同一级别, 我们用 $O(n)$ 表示; 而用第二种算法确定 n 是素数, 只须做 $\sqrt{n}$ 次除法, 当 n 较大时, $\sqrt{n}$ 比 n 小得多, 两者不是一个数量级别, 我们记第二种算法的时间复杂性为 $O(\sqrt{n})$ 。由此可知, 选择好的算法可以事半功倍。

算法可以分为数值计算算法和非数值计算算法, 数值计算指的是通常意义下的计算, 如求线性方程组的解, 求方程的根等计算方法; 非数值计算算法包含对表、树、图等非数值性的结构的处理方法, 以及对各种数据进行排序, 查找等方面的算法。

1.2.2 数据结构

在确定了解题的算法, 准备编程实现算法时, 首先要考虑的是如何将算法中涉及的数据用合理、有效的结构予以组织, 以便存取数据。这便关系到数据结构问题。

数据是描述客观事物的数、字符、符号、表格以及图形、图像、声音等可由计算机程序处理的对象集合。简单而言, 数据结构是指带有结构的数据元素的集合。数据结构包含两个方面的意义, 一方面指的是数据元素之间的逻辑关系, 即说明元素之间的某种顺序关系, 称之为逻辑结构; 另一方面指的是数据的物理结构, 它描述的是数据逻辑结构在计算机存储器中的映像, 也称为数据的存储结构。数据的存储结构可以分为顺序存储结构和链式存储结构两大类, 所谓顺序结构指的是数据元素按照它的逻辑结构顺序地存放在一组连续的存储单元中; 而链式结构中则不要求数据元素连续存储, 但需要采用一种指针变量来指示数据元素之间的顺序关系。相同逻辑结构的数据元素采用相同的存储结构, 因此, 一般统一用数据结构这一名词陈述。

程序语言中提供的数据类型是语言已经实现的一些基本数据结构。高级语言中提供的数据类型主要可分为三大类:

1. 简单类型: 包括整型、实型、字符型等数据。
2. 构造类型: 构造类型是由已知类型如简单类型按一定规则构造而成的类型, 例如数组就是最简单但又最常用的构造类型, 数组可以有整型数组、字符型数组等。C 和 Pascal 语言中还提供更为复杂的构造类型, 如结构 (Pascal 语言中称记录) 和联合。
3. 指针类型: 它主要用于构造链表、树、图等各种复杂的数据结构。通过指针这种特殊的变量来指示数据元素之间的顺序关系。在 C 语言中, 指针实际上就是存储单元的地址, 其应用范围尤其广泛, 但要小心使用。

程序语言提供的数据类型丰富与否, 在很大程度上决定了该语言的强弱, C 和 Pascal 语言的数据类型就比 Basic 和 Fortran 丰富得多, 因此用 C 和 Pascal 实现一些复杂的算法 (特别是非数值算法) 比 Basic 和 Fortran 方便得多。

利用高级语言提供的基本数据类型, 可以构造其他有用的数据结构。例如, 高级语言都提供有数组, 在数组的基础上, 我们可以构造队列, 堆栈等线性表结构。许多算法的实现都依赖一定的数据结构, 我们将结合 C 语言, 在以后的章节中适当予以介绍。

1.3 程序设计基础知识

早期的程序设计, 在确定了算法和数据结构之后, 便交给程序员去编写程序, 编写的程序只要在机器上通过, 便算完成任务。但如果编写的程序较大或从软件工程的角度去衡量, 情

况就不一样了。从软件工程的角度出发,编写的程序不仅要保证正确性,而且要保证程序的可读性,可修改性,和可维护性。只有保证了这些因素,才能说保证了程序的质量。因此,程序员编写的程序不仅要考虑自己读懂,还要考虑别人看懂,而且还要提供可修改的余地。由此可见,编制程序必须注意程序设计基本方法和风格,以保证程序的可读性和可维护性。

1.3.1 算法描述工具

编写程序(特别是大型程序)之前,最好先用一种方法描述算法步骤,以方便编程。常用的方法主要有自然语言、程序流程图、问题分析图(PAD图)、N-S图(也称盒状图)等。下面主要介绍程序流程图和N-S图两种方法。

一. 程序流程图

程序流程图采用一些几何图形表示算法步骤中的各种操作,图1.3.1列出了常用的流程图形符号。

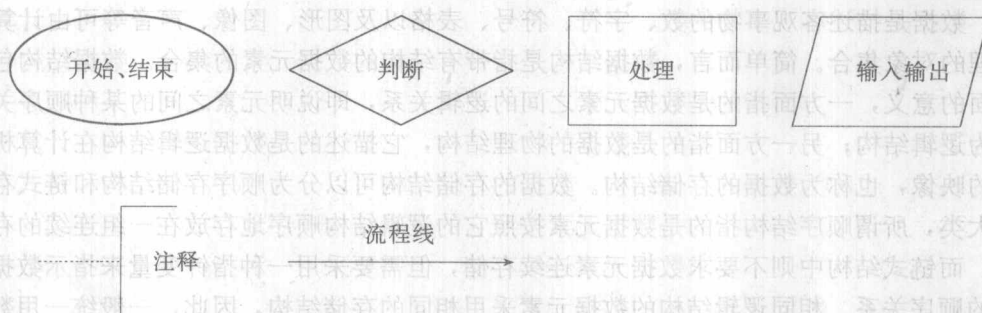


图 1.3.1 常用程序流程图符号

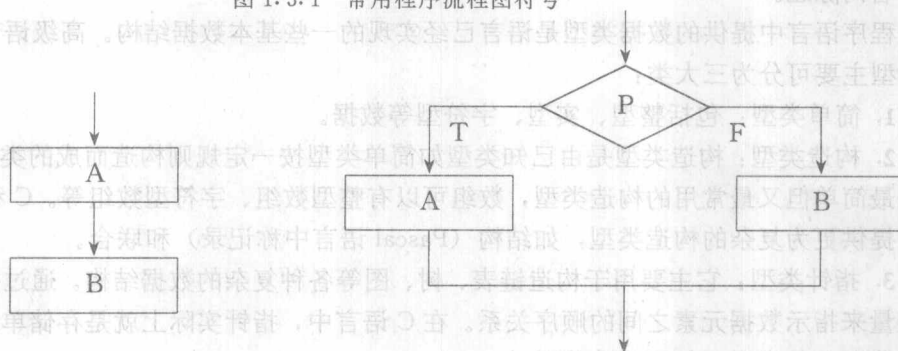


图 1.3.2 顺序结构

图 1.3.3 选择结构

利用流程图表示算法,直观形象化,容易理解和掌握,但由于它允许流程线指向图中任意一个程序框,导致程序走向任意,因而违反结构程序设计原则,在此基础上编制程序逻辑思路不清楚,可读性差。因此,在使用程序流程图时,应限制流程图只由几种基本控制结构嵌套而成。图1.3.2到图1.3.4分别给出了顺序、选择和循环三种基本控制结构所对应的程序流程。其中,图1.3.2和1.3.3分别表示顺序和选择控制,图1.3.4分别给出了几种常用的循环控制结构,在图1.3.2中,表示A操作完成后,执行B操作;图1.3.3中表示当条件P满足时,执行A操作,不成立时执行B操作;图1.3.4(a)中,表示当条件P成立时,反复执行A操作;循环结构的另一种形式见图1.3.4(b),它表示反复执行A操作,直到条件P被满足。注意,图1.3.4(a)中,表示先判定条件,再决定是否执行操作,因此A操作可能一次也不执行;而图1.3.4(b)中,表示先执行操作,再判定条件,决定是否继续执行操