计算机基础教育丛书

程序设计与开发技术

谭浩强　张基温　徐士良

清华大学出版社

程序设计与开发技术

谭浩强　张基温　徐士良　编著

清华大学出版社

内 容 简 介

"程序设计与开发技术"是继程序设计语言之后进一步提高程序设计能力的一门课程。

本书从实际应用出发，通过大量实例，全面地介绍程序设计的基本技术与开发工具。全书共分六章，内容包括：程序设计的概念，程序设计的基本准则，算法设计的基本方法，应用数据结构，程序的测试与调试，软件开发。在附录中简单地介绍了五种常用的操作系统。

本书可作为大专院校非计算机专业学生和研究生的《程序设计与开发技术》或《计算机软件技术基础》课程的教材，也可作为学过一、二门程序设计语言的读者的自学参考书。

(京) 新登字158号

计算机基础教育丛书

程 序 设 计 与 开 发 技 术

谭浩强　张基温　徐士良　编著

责任编辑　范素珍

☆

清华大学出版社出版

(北京　清华园)

中国科学院印刷厂印刷

新华书店总店科技发行所发行

☆

开本：787×1092 1/16　印张：16.5　字数：411千字

1991年12月第1版　1991年12月第1次印刷

印数：0001-8000

ISBN 7-302-00921-X/TP·338

定价：7.50元

《计算机基础教育丛书》出版说明

近年来，我国的计算机应用事业迅速发展，大批科技人员、大中学生、管理人员，以及各行各业的在职人员都迫切要求学习计算机知识，他们已经认识到，计算机知识是当代知识分子的知识结构中不可缺少的重要部分。

计算机应用人才的队伍由两部分人组成：一部分是从计算机专业毕业的计算机专门人才，他们是计算机应用人才队伍中的骨干力量；另一部分是各行各业中从事计算机应用的人才，他们既熟悉本专业的业务，又掌握计算机应用的技术，人数众多，是计算机应用人才队伍的基本力量。他们掌握计算机知识的情况和应用计算机的能力在相当大程度上决定了我国计算机应用的水平。因此，在搞好计算机专业教育的同时，在广大非计算机专业中开展计算机基础教育是十分必要的。

非计算机专业中的计算机教学，无论就目的、内容、教学体系、教材、教学方法等各方面都与计算机专业有很大的不同，它以应用为目的，以应用为出发点。如果不注意这个特点，将会事倍功半。广大非计算机专业的师生、在职干部迫切希望有一套适合他们的教材，以便循序渐进地迈入计算机应用领域，并且不断地提高自己的水平。我们在前几年陆续编写了一些适合初学者使用的教材，受到广大群众的欢迎。许多读者勉励我们在此基础上进一步摸索和总结规律，为我国的广大非计算机专业人员编写一整套合适的教材。

近年来，全国许多专家、学者在这个领域作了有益的探索，写出了一批受到群众欢迎的计算机基础教育的教材。特别是全国高等学校计算机基础教育研究会作了大量的工作，在集思广益的基础上，提出了在高等学校的非计算机专业中进行计算机教育的四个层次的设想，受到广泛的注意和支持。我们认为：计算机的应用是分层次的，同样，计算机人才的培养也是分层次的；非计算机专业中各个领域的情况不同，也不能一律要求，在进行计算机教育时也应当有不同的层次。对于每一个学习计算机知识的人，还有一个由浅入深、逐步提高的过程。

我们认为，编辑出版一套全面而有层次的计算机基础教育的教材，目前不仅是十分必要的，而且是完全有条件的。在全国高等学校计算机基础教育研究会和许多同志的积极推动和清华大学出版社的大力支持下，我们决定编辑《计算机基础教育丛书》。它的对象是：高等学校非计算机专业的学生、计算机继续教育或培训班的学员、广大在职自学人员。

本丛书包括计算机科学技术的一些最基本的内容，例如计算机各种常用的高级语言、计算机软件技术基础、计算机硬件技术基础、微型计算机的原理与应用、算法与数据结构、数据库基础、计算机辅助设计基础、微机网络与应用、系统分析与设计等，形成多层次的结构，读者可以根据需要与可能选学。

本丛书的宗旨是针对广大非计算机专业的需要和特点来组织教材。敢于破除框框，从实际出发，用读者容易理解的体系和叙述方法，深入浅出、循序渐进地帮助读者更好

地掌握课程的基本内容。希望我们的丛书能在这方面闯出自己的风格，在实践中接受检验。

本丛书的作者大多数是高等学校中有较丰富教学经验的教师。但是，由于计算机科学技术的飞速发展以及我们的水平有限，丛书肯定会存在许多不足，丛书的书目和内容也应当不断发展和更新。我们热情地希望得到社会各界和广大读者的批评指正。

主编　谭浩强　林定基　刘瑞挺

1988.10.

前　言

程序设计是计算机基础教育的重点和基础，这已经被越来越多的人所认识。作为高等学校理工科学生和工程技术人员不能仅满足于能简单地使用别人已编好的程序，还应当有独立设计程序的能力。

目前，全国绝大多数的大、中专院校都已开设了计算机语言程序设计课程。但是，计算机程序设计语言的学习仅仅是计算机教育的入门。只靠几十小时的学习还难以深入掌握计算机应用的知识。入门以后还要继续提高，在计算机科学技术高速发展的今天，这种要求显得日益迫切。在学习了一、二门计算机语言以后应该如何继续提高呢？全国高校计算机基础教育研究会提出了"四个层次"的方案。在软件知识方面，一个途径是学习"软件技术基础"，即侧重于扩展软件的基本知识。在1990年举行的全国高校计算机基础教育研究会年会上，又提出了另一个途径，即沿着提高程序设计能力这个线索进行学习和提高，使学生能比较深入而扎实地掌握程序设计所需的知识，并且具备程序设计与开发的能力。这一途径更侧重于应用方面，因为多数人将来工作的性质可能是使用应用软件或者自己开发软件。因此程序设计的知识和能力是至关重要的。本书就是为此目的而进行的一个尝试。希望能抛砖引玉，促进计算机教育的深入。

计算机科学技术飞速发展，已经出现了非过程化的语言。以后人们可以不必指定计算机"怎么做"，而只要指出让计算机"做什么"就可以了。这是计算机应用的一个新阶段，将使更多的人更容易地进入计算机应用领域。但是，目前甚至以后一个相当长时期内，非过程化程序设计还不能完全代替过程化程序设计。程序设计依然是计算机应用人员的一项基本功。程序设计教育的目的，一方面是为了培养学生使用计算机解题的能力；另一方面是对学生的一种"计算机思维"的训练，使人们了解计算机是怎样进行工作的，使自己的思维方式适应计算机化社会的要求。要发展自己的计算机应用的能力，这一点是十分重要的。不能满足于"知其然而不知其所以然"的低水平的应用。

因此，我们认为，当前的程序设计课程的内容仍应以过程化程序设计为主。程序设计可以用下面的公式来概括：

程序设计＝算法＋数据结构＋方法＋工具

本书的第一章介绍了这个公式的含义和有关概念，全书围绕这个公式展开，使读者全面掌握有关程序设计与开发的知识。

本书的对象是已学习过一门或一门以上计算机语言并已有初步程序设计知识的读者。我们力图回避高深的数学和计算机专业中的高深的专门知识与术语。相信多数读者是能够掌握本书的内容的。由于篇幅关系，不可能在本书中对所涉及的各个领域的内容作详细而深入的叙述，只是从应用的角度作必要的说明。读者如果需要对有关内容进一步深入了解，可以参考其它有关书籍。

本书是以张基温老师编写的讲义为基础，修改、提高而成的。在1989年夏，全国高等学校计算机基础教育研究会举办的"程序设计技术师资研讨班"上曾以本书的初稿作

为教材进行讲授，受到参加者的欢迎和鼓励，要求正式出版以供广泛使用。经征求多方面意见后，作了必要的修改补充，由清华大学出版社出版。请国内专家和读者指正。

谭浩强

1991.2

目　录

第一章　程序设计的概念 …… 1
§1.1　算法 …… 1
1.1.1　算法的要素 …… 1
1.1.2　算法的特征 …… 5
§1.2　数据结构 …… 5
1.2.1　数据类型 …… 5
1.2.2　数据结构 …… 7
1.2.3　数据结构与算法 …… 9
§1.3　程序设计方法和风格 …… 9
1.3.1　结构化程序设计 …… 11
1.3.2　模块化程序设计 …… 12
1.3.3　逐步细化的设计过程 …… 13
1.3.4　程序的风格 …… 14
§1.4　工具 …… 18
1.4.1　算法描述工具 …… 18
1.4.2　程序设计语言 …… 21
习题 …… 22
第二章　程序设计的基本准则 …… 25
§2.1　抽象准则 …… 25
2.1.1　模型 …… 25
2.1.2　概念与子概念 …… 27
2.1.3　自顶向下逐步细化 …… 28
2.1.4　数据抽象 …… 29
§2.2　结构化准则 …… 30
2.2.1　选择型程序设计 …… 30
2.2.2　循环型程序设计 …… 35
2.2.3　带有GOTO语句的结构 …… 39
§2.3　局部化与信息隐蔽准则 …… 40
2.3.1　基本概念 …… 40
2.3.2　PARNAS方法 …… 40
2.3.3　模块间的通讯方式 …… 41
2.3.4　模块的评价 …… 44
习题 …… 48
第三章　算法设计的基本方法 …… 51
§3.1　归纳法 …… 51
3.1.1　穷举归纳法 …… 52
3.1.2　枚举归纳法 …… 55

3.1.3 递推 ……58
3.1.4 递归 ……64
§3.2 分治法 ……66
3.2.1 二分检索 ……67
3.2.2 快速排序 ……68
3.2.3 归并排序 ……72
3.2.4 快速算法 ……75
§3.3 数值法 ……77
3.3.1 逼近 ……78
3.3.2 离散化方法 ……84
3.3.3 数值方法的一些特点 ……87
§3.4 数字仿真法 ……92
3.4.1 确定性数字仿真 ……93
3.4.2 随机性数字仿真 ……95
3.4.3 数字仿真语言 …… 112
§3.5 回溯法 …… 114
习题 …… 117
第四章 应用数据结构 …… 125
§4.1 顺序表 …… 127
4.1.1 基本概念 …… 127
4.1.2 顺序表的运算及数组 …… 128
4.1.3 堆栈 …… 132
4.1.4 队列 …… 143
§4.2 链表 …… 147
4.2.1 向前链表 …… 147
4.2.2 循环链表 …… 155
4.2.3 双向链表 …… 157
§4.3 索引存储 …… 159
4.3.1 索引存储的基本概念 …… 159
4.3.2 “顺序——索引——顺序”存储方法 …… 161
4.3.3 “顺序——索引——链接”存储方法 …… 162
4.3.4 多重索引存储 …… 163
§4.4 树与二叉树 …… 165
4.4.1 树结构及其存储 …… 165
4.4.2 二叉树 …… 172
习题 …… 181
第五章 程序的测试与调试 …… 184
§5.1 测试的概念 …… 184
5.1.1 程序的可靠性 …… 184
5.1.2 测试及其特征 …… 184
5.1.3 程序错误分类 …… 187
§5.2 测试用例设计 …… 190
5.2.1 白箱测试技术 …… 191

5.2.2 黑箱测试技术 …… 193
5.2.3 综合策略 …… 197
§5.3 程序测试的层次 …… 197
5.3.1 模块测试 …… 198
5.3.2 整体测试 …… 200
5.3.3 高级测试 …… 200
§5.4 程序测试的方式和过程 …… 201
5.4.1 程序的静态分析 …… 201
5.4.2 程序的动态测试 …… 202
5.4.3 自动测试工具 …… 202
§5.5 程序调试 …… 203
5.5.1 调试与测试的联系及区别 …… 203
5.5.2 诊断错误的实验方法 …… 204
5.5.3 错误诊断的推理技术 …… 206
5.5.4 错误修改的原则 …… 209
习题 …… 209
第六章 软件开发 …… 211
§6.1 软件开发环境 …… 211
6.1.1 软件与软件工程 …… 211
6.1.2 软件生产环境的变迁 …… 212
6.1.3 软件工程支撑环境的基本组成 …… 213
§6.2 操作系统 …… 214
6.2.1 操作系统的功能及任务 …… 214
6.2.2 操作系统的发展过程 …… 216
6.2.3 操作系统的分类 …… 220
6.2.4 优良的操作环境——多窗口系统 …… 221
§6.3 软件设计的基本技术 …… 222
6.3.1 概述 …… 222
6.3.2 面向过程的设计技术 …… 225
6.3.3 面向对象的设计方法 …… 230
习题 …… 232
附录
附录一 CP/M 操作系统 …… 233
附录二 UNIX 操作系统 …… 235
附录三 MS-DOS 操作系统 …… 244
附录四 OS/2 操作系统 …… 251
附录五 VAX/VMS 操作系统 …… 252
参考文献 …… 254

第一章　程序设计的概念

什么是程序设计呢?不同的人从不同的角度可以作出不同的解释。通常的解释是:“将事先确定的解题步骤,用机器指令或机器所能理解的语言描述出来的过程”。这一解释是比较容易理解的。但是,这一定义并没有真正深刻揭示出程序设计这一概念的内涵。程序设计牵涉到几个方面的内容,可用下面的公式来概括:

程序设计＝算法＋数据结构＋方法＋工具

这四个方面互相依存,不可分割。

§1.1　算　　法

1.1.1　算法的要素

任何解题过程都是由一定的步骤组成的。通常,把解题过程的准确而完整的描述称作解该问题的算法。下面先看几个例子。

例 1.1.1　换录音乐。

A、B两人各有一盘音乐磁带,现两人想换听音乐,又不想交换磁带,就不得不借助于另一盘磁带M,并按如下算法换录:

S1:将A带中的音乐录入M中;

S2:将B带中的音乐录入A中;

S3:将M带中的音乐录入B中;

S4:结束。

例 1.1.2　计算 $a+|b|$。

根据公式

$$a+|b|=\begin{cases}a+b & b\geqslant 0\\ a-b & b<0\end{cases}$$

可以得到如下算法:

S1:输入a、b;

S2:判断　b≥0?　若b≥0,则执行S2.1,否则执行S2.2;

　　S2.1:a＋b⇒c;

　　S2.2:a－b⇒c;

S3:输出c;

S4:结束。

例 1.1.3　读书。

读书可按下面的算法进行:

S1:打开书,翻到要看的页码;

S2:重复地执行S2.1、S2.2,直到不愿再看为止;

S2.1：看一页；

S2.2：翻到下一页；

S3：合上书；

S4：结束。

由上面三个简单的算法可以看出，一个算法由一些操作组成，而这些操作又是按一定的控制结构所规定的次序执行的。如例 1.1.1 中的操作 S1、S2、S3、S4 是顺序地执行的；例 1.1.2 中的 S2 中又包含了 S2.1 与S2.2 两种操作，而 S2.1 与S2.2 不能都执行，要根据 b 的值决定执行哪个操作；例 1.1.3 中的 S2 中的 S2.1 与 S2.2 是重复执行的，而重复要一直延续到满足条件"不愿再看为止"。简而言之，算法由操作与控制结构两要素组成。

1.1.1.1 操作

在解决任何一个问题时，所使用的操作是与问题的要求以及所使用的工具的功能有关。如一辆汽车上装了一车沙子，现要将它们卸下来，若汽车没有自卸功能，就得靠人一铣一铣地操作；若汽车有自卸功能，则应操纵卸车系统。因此，每个算法实际上是按解题要求从所用解题工具能进行的操作中选择合适的操作所组成的序列。能由计算机执行的算法称为计算机算法。它们是由计算机能执行的操作所组成的序列。

通常，计算机可以执行的基本操作是以"指令"的形式向人们提供的。一台计算机能执行的指令的集合，称为该机器的指令系统。计算机的解题程序，就是按解题要求从计算机的指令系统中选择合适的指令所组成的指令序列。

计算机科学的发展，为我们提供了丰富多采的计算机机型。这些计算机尽管具有不同的特色，但它们所以能被称之为"计算机"，是因为它们具备了作为计算机的最基本的功能操作。这些功能操作包括以下几个方面：

（1）逻辑运算："与"、"或"、"非"；

（2）算术运算：加、减、乘、除；

（3）数据比较：大于、小于、等于、不等于；

（4）数据传送：输入、输出、赋值。

虽然每种运算在不同的计算机上的表现形式不同，但我们在解题的开始阶段应当把精力集中在对问题需求的分析上，而不应当强调如何发挥所使用的某一计算机本身的特色。即计算机算法的设计应从上述计算机的四种基本功能操作考虑，按解题要求从这些基本操作中选择合适的操作组成解题的操作序列，而不考虑这些算法在某一机器上所具有的具体形式。下面是几个计算机算法的例子。

例 1.1.4 交换两个变量的值。

变量是存储单元在程序中的标识。与例 1.1.1 相似，要交换两个变量的值，必须引入一个中间变量。

S1：输入 a、b；

S2：a⇒m；

S3：b⇒a；

S4：m⇒b；

S5：输出 a、b；

S6：结束。

这里的a⇒m表示将存储单元a中的值传送到存储单元m。应注意算法执行过程中a、b、m的值的变化，计算机存储器存储数据的特点是“取之不尽，新来旧去”。即一个变量的值赋给另一变量后，它原来的值并未改变，只有向它赋以新值时，原值才被取代。

例 1.1.5 求 $\sum_{i=1}^{100} i$。

算法如下：

S1：输入n（即100⇒n）；

S2：0⇒sum；

S3：1⇒i；

S4：重复执行下面的操作，直到i>n；

S4.1：i+sum⇒sum；

S4.2：i+1⇒i；

S5：输出sum；

S6：结束。

只有理解了算法是由一步一步的操作构成的，才算拿到了开启程序设计大门的一把钥匙。这对已经接受了良好的数学训练，习惯于按演绎方式求解问题的读者来说更是十分必要。演绎数学以公理系统为基础，问题的求解过程是通过有限次推演完成的，每次推演都将对问题作更进一步的描述。如此不断推演，直到能直接将解描述出来为止。而算法则是使用一些最基本的操作（加、减、乘、除、“与”、“或”、“非”、传送、比较），通过对已知条件进行一步步的加工、变换，从而实现解题目标。或者说，演绎方法着重于静态地描述，而算法着重于动态地执行。这两种方法的解题思路是不同的。

有些人认为算法太原始、太呆板、太简单，不如公理系统那样正规、那样丰富而有技巧。其实，算法的历史要比公理系统更为长久。在古代，当数学体系还未建立时，人们已经可以用算法解很多难度很大的问题了。这种使用算法解题的思想集中反映在算术学中。后来，公理系统不断成熟并发展了，人们的注意力和兴趣逐渐转向公理系统，但是许多环节还要用算法作为补充，甚至许多（如智力测验）问题的求解还必须使用算法去解决。电子计算机的出现，使算法的研究出现了新的转机，要求人们努力去研究各种解题算法。有人预料：我们已经把数学描述成了一个公理系统，也必将能把数学描述成为一个算法系统，使算法与公理系统成为数学的两大平行的思想体系。因此，初学程序设计的人一定要把自己的解题思路从演绎转到算法的思路上来。

1.1.1.2 *算法的控制结构*

一个算法的功能不仅取决于所选用的操作，而且还决定于各操作之间的执行顺序，即控制结构。算法的控制结构给出了算法的框架，决定了各操作的执行次序。用流程图可以形象地表示出算法的控制结构。图1.1、图1.2、图1.3分别是例1.1.4、例1.1.2、例1.1.5中算法的流程图。

图1.1中各操作是依次执行的，这种控制结构称为顺序（sequence）结构。图1.2中，S2包含S2.1与S2.2两种操作，且不全要执行，而要由条件b≥0是否成立来决定

选择执行 S2.1 还是 S2.2，这种控制结构称为选择（selection）结构。图 1.3 中，S4 包含的 S4.1 与 S4.2 两个操作要重复执行，直到 i>n（即i>100）为止（注意，当 i=n 时，还要执行一次），这种控制结构称为重复（repetition或iteration）结构，或循环（loop）结构。

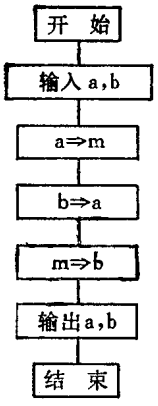

图 1.1

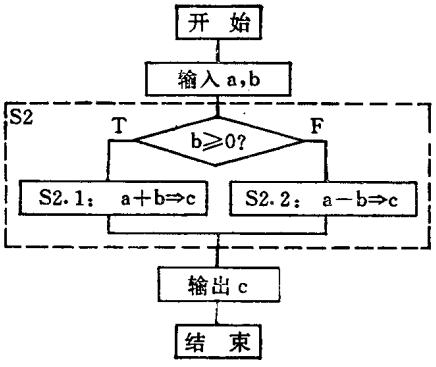

图 1.2

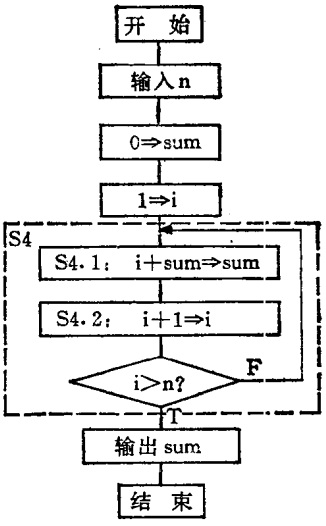

图 1.3

1966 年，Bohm 和 Jacopini 证明了任何复杂的算法都可以用顺序、选择、循环三种控制结构组合而成。所以这三种控制结构称为算法的三种基本控制结构。图 1.4 是这三种基本控制结构的一般形式。其中（c）与（d）是循环型结构的两种形式，（c）称为直到型循环，(d)称为当型循环。图 1.3 是一种直到型循环，它首先进入循环体(S4.1，S4.2)，然后判断条件“直到×××××”是否成立，若不成立则再返回去执行循环体。当型循环则是先判断条件“当×××××”是否成立，若成立则执行循环体，执行一次后

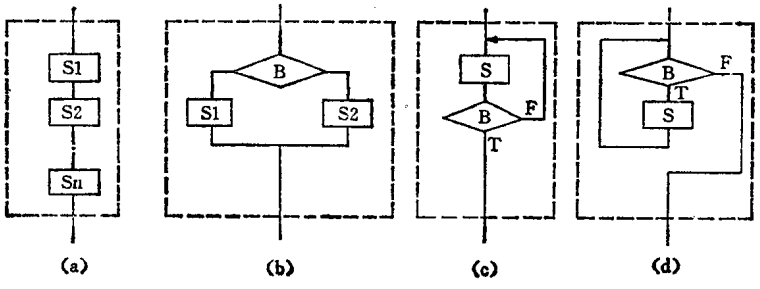

图 1.4

再判断条件还是否成立，若不成立便退出循环。图 1.5 是例 1.1.5 使用当型循环的一种算法。当型循环与直到型循环可以互相转换。

如果把每种基本控制结构看成一个算法单位，则整个算法便可以看作是由各算法单位顺序串接而成，好象“妇女脖子上的一串珍珠项链”一样，结构清晰，来龙去脉一目了

然，容易阅读、也容易理解。这样的算法称为是结构化的。

1.1.2 算法的特征

算法的概念实际上是很普遍的。但是，越是普遍的概念越难以给出严格的定义。通常认为算法具有以下几个特征：

（1）算法是由一套规则组成的一个过程。所谓过程就是一些步骤，这些步骤连在一起能给出一类问题的解答。因此，算法实际上是一种抽象的解题方法，它具有动态性。

（2）组成算法的规则是确定的、可执行的。例如，算法中不允许出现诸如“计算x/0”或“将一些数进行运算”等不可进行的操作或含混不清、具有两义性的描述。同时每个算法所规定的操作必须是基本的。

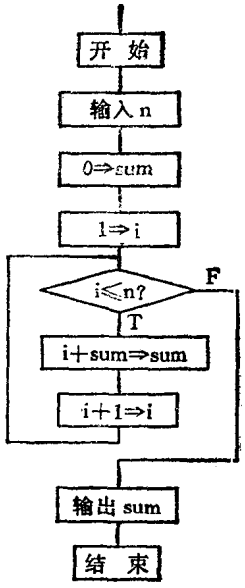

图 1.5

（3）每种算法必须有确定的控制结构。即按这些规则组成的操作，必须按一定的顺序执行。

（4）每个过程必须是有效的。即它最后必须能够得到确定的结果，产生一个或多个输出。

（5）解答必须在有限步内得到。也就是说，算法的执行步数是有限的，因而执行是可终止的。任何不会终止的算法是没有意义的。有限性还包含了实际上可以容忍的合理限度，如果一个算法要执行千万年，实际上是不可容忍的。

综上所述，我们可以得出如下的定义：

算法是一个过程，这个过程由一套明确的规则组成，这些规则指定了一个操作的顺序，以便用有限的步骤提供特定类型问题的解答。

§ 1.2 数据结构

数据就是计算机化的信息。广义地讲，计算机可以称作信息处理机，它所处理的对象就是数据。任何一个算法总是与某些特定的数据相联系的。这些数据或作为输入，或作为中间结果，或作为输出。

信息不仅各有属性，而且在多数情形下互有联系，尤其是在非数值处理领域中更是如此。在程序中，信息的特性往往被抽象为数据类型和值，而信息及其联系又被抽象为数据结构（也称信息结构）。

1.2.1 数据类型

数据的类型可以分为原始的与组合的两大类型。

原始类型有以下几种：

• 数值类型（又可分为整型与实型）
• 逻辑类型
• 字符类型
• 指针类型

组合类型是由原始类型按某种方式（可以很简单，也可以很复杂）组合而成的。如数组类型、记录类型、枚举类型、集合类型、子界类型等。

具体地说，类型的概念可以从以下三个方面来理解：

（1）一个类型规定了一个以值为元素的集合，即规定了该类型中的数据的定义域。如整数类型取值为计算机表数范围内的整数；实数类型取值于计算机表数范围内的实数；逻辑类型取值为"真"（TRUE）或"假"（FALSE）；字符类型取值于计算机可以表示的字符集中的元素；指针类型的值一般对应于存储器里的一个绝对地址或相对地址，其值为整数。

（2）类型定义了一个运算的集合。也就是说，能够施加到不同类型数据上的运算是不同的。如对数值类型的数据可以施加算术运算，对逻辑类型数据可以施加逻辑运算，对字符类型数据可以施加连接运算与求子字符串运算。由于指针类型数据对应于存储器的某个地址，因此不能进行乘除运算。

（3）类型又规定了数据在计算机中的存储与表示的方式。例如，在 IBM PC BASIC 中，每个整型数据用两个字节存放，每个实型数据要用四个（单精度）或八个（双精度）字节存放，每个字符用一个字节存放。在表示一个字符串常数时，要用一对引号将它们括起来，而数值型数据则不需要引号括起来，等等。

在高级语言程序中，每一个数据都是属于一定的类型，不属于某一类型的数据是不存在的。类型这个信息既可用于防止或检查程序中一些无意义的结构，又可用于确定计算机中数据表示和处理的方式。因此，在程序中使用每个数据之前，都应当定义（或指定）其类型。

不同的高级语言所提供的数据类型的种类以及类型定义的方式又是不相同的。

PASCAL 语言可以提供丰富的数据类型，它包括了前面所列举的各种类型。其中整型、实型、字符型、逻辑（布尔）型四种标准类型及数组类型是系统直接确定的，在定义变量时只需在关键字 VAR 后面用 INTEGER、REAL、CHAR、BOOLEAN 及 ARRAY 加以说明即可。其它的类型可以由用户用关键字 TYPE 定义。

FORTRAN77 提供的数据类型有整型、实型、复型、逻辑型、数组类型、记录类型等几种。其中原始类型的变量允许用隐含规则（I-N 规则）、类型说明语句（INTEGER、REAL、DOUBLE PRECISION、COMPLEX、LOGICAL）或隐含声明语句（IMPLICIT）来定义。数组类型除进行上述定义外，还要用 DIMENSION 语句指定数组的名称、维数及每维的上下界。

True BASIC 提供数值与字符两大类数据类型，用它们还可以定义数组类型。数值类型是隐含的，字符类型由变量名后面加说明符"$"来定义。

语言提供的数据类型是多少最合适，目前的看法是不一致的。提供的数据类型多，用户能方便地处理各种问题，但增加了初学者在理解上的困难。一般认为数值型、字符型及数组类型是最基本的。

1.2.2 数据结构

数据结构研究的内容是如何合理地组织被处理的数据，通常要涉及三方面的内容：

- 数据的逻辑结构
- 数据的物理结构
- 数据结构上所施加的运算

1.2.2.1 数据的逻辑结构

在研究数据结构时，通常把组成该数据结构的数据元素（可以是原始类型及其简单或复杂的组合）称作一个结点或称原子。各结点之间的逻辑结构是数据元素所代表的客体之间联系的抽象。

数据结构分为线性结构与非线性结构。

线性结构有两个特点：

（1）仅有一个终端结点和一个开始结点；

（2）所有的结点（除终端结点和开始结点外）有且只有一个前驱结点和一个后继结点，而终端结点只有一个前驱结点，开始结点只有一个后继结点。

属于线性结构的有串、栈、队、表等。外存数据结构文件也是一种线性结构。

非线性结构则不符合上述两点，即有多于一个的终端结点或开始结点，至少有一个结点有多于一个的前驱结点或后继结点。属于非线性结构的有树、图等。

1.2.2.2 数据的物理结构

数据的物理结构也称存储结构，是数据逻辑结构的物理实现方式，是依赖于计算机的。

数据的物理结构的实现方式大致有四种：

（1）顺序方式

这种方式主要用于线性结构，它把逻辑上相邻的结点存储在物理上相邻的存储单元中，结点之间的关系由存储单元的邻接关系来体现。图 1.6 是将线性结构 K1——K2——K3——K4——K5——K6——K7 顺序地存放在存储单元中的示意图。

（2）链接方式

使用链接方式时，每个结点要由两部分组成：一部分存放数据信息，另一部分存放指针。其中指针用于指向该结点的前驱或后继。图 1.7 为用链接方式存储线性结构的示意图。它在每个结点中设一个指针，用以指出其后继结点的地址。最后一个结点的指针为空，用“∧”表示。开始结点要专门用一个指针去指向它，在此把这个指针记为“H”。当然，也可以用指针指出各结点的前驱结点来确定线性结构中的顺序关系。

链接方式也能存储非线性结构。图 1.8（a）及（b）为一棵二叉树的逻辑结构及物理结构的示意图。在物理结构中，每个结点设两个指针指向它的子女结点，另外设一个专门的指针R指向树根。

由此我们看到，用顺序方式实现线性结构比较自然，而实现非线性结构就比较困难；链接方式既可用于实现线性结构，又可用于实现非线性结构。

（3）索引方式

在这种存储方式中，是通过结点在线性序列中的位置来确定结点的实际存储地址。