# 数据结构

## ——使用C语言

陈一华　刘学民　潘道才　编

电子科技大学出版社

## 内 容 提 要

本教材是在1994年由全国大专计算机专业教材编审委员会推荐出版的《数据结构》（潘道才、陈一华编）的基础上，本着增强科学性、适应性、实践性和应用性的原则，对教材内容进行了削枝强干的增删和调整；同时，为了适应许多高等工程专科学校把C语言作为主要程序设计语言进行教学的新情况，对有关的算法都使用C语言加以描述。其主要内容为数据的逻辑结构、物理结构以及对各种结构所定义的运算和应用。其中包括线性表、串、数组、广义表、树、图、文件等数据结构。对于同一种逻辑结构的数据，讨论其不同的物理结构和相应的有关算法。为了增强教材的应用性，除了在各章节中通过一些实例介绍数据结构的应用外，还专门讨论了查找和排序的各种方法，在第九章中还讨论了综合应用数据结构的例子。

本书可作为大专类计算机软、硬件专业的教材，也可作为使用计算机的广大科技人员的自学参考书。

## 声　　明



# 数 据 结 构

## ——使用C语言

陈一华　刘学民　潘道才　编

出　　版：电子科技大学出版社　（成都建设北路二段四号，邮编：610054）
责任编辑：张　俊
发　　行：电子科技大学出版社
印　　刷：四川建筑印刷厂
开　　本：787×1092　　印张　17.125　　字数　414千字
版　　次：1998年11月第一版
印　　次：1998年11月第一次印刷
书　　号：ISBN7—81043—993—6/TP·455
印　　数：1—5000册
定　　价：22.00元

# 出 版 说 明

为做好全国电子信息类专业“九五”教材的规划和出版工作，根据国家教委《关于“九五”期间普通高等教育教材建设与改革的意见》和《普通高等教育“九五”国家级重点教材立项、管理办法》，我们组织各有关高等学校、中等专业、学校、出版社，各专业教学指导委员会，在总结前四轮规划教材编审、出版工作的基础上，根据当代电子信息科学技术的发展和面向21世纪教学内容和课程体系改革的要求，编制了《1996～2000年全国电子信息类专业教材编审出版规划》。

本轮规划教材是由个人申报，经各学校、出版社推荐，由各专业教学指导委员会评选，并由我部教材办协商各专指委、出版社后，审核确定的。本轮规划教材的编制，注意了将教学改革力度较大、有创新精神、特色风格的教材和质量较高、教学适用性较好、需要修订的教材以及教学急需，尚无正式教材的选题优先列入规划。在重点规划本科、专科和中专教材的同时，选择了一批对学科发展具有重要意义，反映学科前沿的选修课、研究生课教材列入规划，以适应高层次专门人才培养的需要。

限于我们的水平和经验，这批教材的编审、出版工作还可能存在不少缺点和不足，希望使用教材的学校、教师、同学和广大读者积极提出批评和建议，以不断提高教材的编写、出版质量，共同为电子信息类专业教材建设服务。

电子工业部教材办公室

# 前　言

本教材系全国电子信息类专业“九五”(1996～2000年)规划教材，由电子工业部教材办公室和全国大专计算机专业教学指导委员会推荐出版。本教材由上海大学陈一华、刘学民、潘道才编写，主审朱儒荣，责任编委李逊林。

本教材是在1994年由全国大专计算机专业教材编审委员会推荐出版的《数据结构》(潘道才、陈一华编)的基础上，本着增强科学性、适应性、实践性和应用性的原则，对教材内容进行了削枝强干的增删和调整；同时，为了适应许多高等工程专科学校把C语言作为主要程序设计语言进行教学的新情况，对有关的算法都使用C语言加以描述。

本教材可作为大专类计算机软、硬件专业的教材，讲授学时为60左右(若学时较少，建议带“*”号的章节可不作为讲授内容)，上机实习为20～30学时。其主要内容为数据的逻辑结构、物理结构以及对各种结构所定义的运算和应用。其中包括线性表、串、数组、广义表、树、图、文件等数据结构。对于同一种逻辑结构的数据，讨论其不同的物理结构和相应的有关算法。为了增强教材的应用性，除了在各章节中通过一些实例介绍数据结构的应用外，还专门讨论了查找和排序的各种方法，在第九章中还讨论了综合应用数据结构的例子。数据结构是一门实践性较强的计算机软件基础课程，该课程对提高学生逻辑分析、抽象思维和程序设计的能力，培养优良的程序设计风格是极为重要的。做习题和上机实习则是学好数据结构的重要环节，为此，本教材在各章后配有一定份量的习题，并在附录一中给出了上机实习的目的要求、安排和上机题。

本次修编，我们在原有基础上努力使本教材具有以下特色：

1. 既考虑学科的发展，又兼顾大专的水平，对教材内容进行合理编排，补充实例，强调应用；

2. 结合高工专试点专业教学改革实践进行编写，力求深入浅出、分散难点，详略得当、举一反三，设疑伏笔、增加兴趣，以启迪学生思维，提高学习自觉性和积极性；

3. 以文字描述和框图描述相结合的方法叙述算法，有利于专科层次学生的理解和掌握；

4. 书中给出的C语言程序，全部在TURBO C 2.0环境中调试通过。

本教材共分九章和两个附录。由上海大学刘学民修编第二、三、五、七章，潘道才修编第八、九章，陈一华修编第一、四、六章和两个附录并统编全稿。上海大学顾训穰副教授、上海理工大学唐俊杰副教授和上海第二工业大学沈韵霞副教授参加了审阅工作，他们提出了许多宝贵的意见，在此表示诚挚的感谢。由于编者水平有限，书中难免还会有不少的缺点和错误，恳请广大读者批评指正。

编　者

1998年5月

# 目　录

# 第一章

# 绪　论

## 1.1　数据结构课程的形成和发展

**数据结构**(Data Structure)是一门随着计算机科学的发展而逐渐形成的新兴学科。

早期的电子计算机主要用于数值计算。后来，随着计算机软、硬件的发展，计算机的应用扩大到实时控制、数据处理等方面，并渗透到工业、农业、商业、军事、文教卫生、家庭等社会的各个领域。计算机处理的对象也从简单的数字、字符发展到文字、图像、声音等各种复杂的具有一定结构关系的数据，而且，要处理的信息量也越来越大。让我们看以下两个例子。

**例1**　教材的计算机管理问题

高校的教材科要负责全校教材的预订、采购、发放和管理等工作。所以，经常要查询教材的库存量等信息和作各种统计，有关部门和教师有时也需了解某专业所用教材的名称、作者、单价和出版日期等信息。如何利用计算机来解决这些问题呢？首先要考虑的就是对教材的各种信息如何加以组织和存储？具体的方法多种多样，一种常用的方法是建立一个表(文件)，每本教材的各种信息占表的一行，为一个数据元素，如图1.1所示。这样，上面的问题就归结为应用计算机对一个表的运算，用户就可以根据需要，按各种不同的项目进行查询或统计，从而快速地得到准确结果。

| 编号 | 教材名 | 作者 | 出版日期 | 出版社 | 单价 | 适用专业 | 库存量 |
|---|---|---|---|---|---|---|---|

图1.1　教材数据元素

**例2**　考虑一个古典问题：设有n个正常人和n个精神病患者(以下简称患者)要过一条河，现只有一条能容纳c(c<2n)个人的小船，为防止患者伤害正常人，要求无论在河的哪一边，正常人的个数不得少于患者的个数(除非正常人个数为0)。又设每个人都会划船，试设计一个过河的最佳(小船来回次数最少)方案。

这样的问题是比较复杂的，我们先构造出相应的数学模型：用一个三元组(x,y,t)表示渡河过程中的某个状态。其中，x表示起始岸上正常人的个数，y表示起始岸上患者的个数，t表示小船的位置：

$$t=\begin{cases}1 & \text{表示小船在起始岸边}\\ 0 & \text{表示小船在目的岸边}\end{cases}$$

我们构造一个图，图中每一个顶点代表一个合法状态(不难推得，合法状态所对应的三元组(x,y,t)必须满足：x=0或x=n或x=y)，图中的边则表示该边所依附的两个顶点(即

两个合法状态)间可由一次划船而相互变换。

例如,当n=2,c=2时,各合法状态及其变换如图1.2所示。

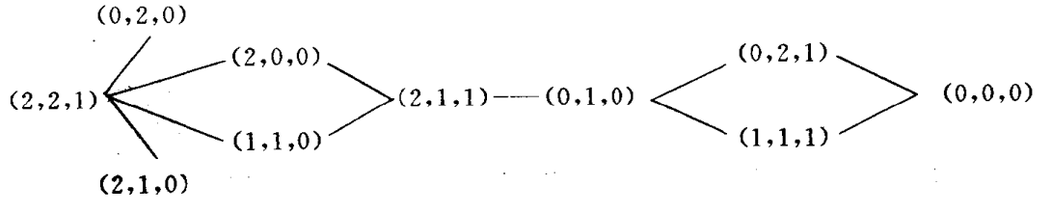


图1.2 各合法状态及其变换图

于是,过河方案的求解就转换成一个图的搜索问题,即搜索一个图,找出从起始顶点(n,n,1)到目的顶点(0,0,0)的一条包含边数最少的通路(若该通路存在,否则给出无解的信息)。

对于图1.2,下面的通路是最佳方案之一:

(2,2,1)⟶(1,1,0)⟶(2,1,1)⟶(0,1,0)⟶(0,2,1)⟶(0,0,0)

即从初始状态(2,2,1)开始,先由一个正常人和一个患者一起从起始岸划船到目的岸,此时的状态为(1,1,0);再由正常人单人划船回起始岸,使状态为(2,1,1);然后两个正常人一起从起始岸划船到目的岸,这时的状态为(0,1,0);接着由一个患者单人划船从目的岸到起始岸,使状态变为(0,2,1);最后由两个患者一起从起始岸划船到目的岸,此时状态为(0,0,0);整个过程中的各个状态都是合法的,这样,就完成了过河的任务。

易知,最佳方案不唯一,下面的方案也是最佳的:

(2,2,1)⟶(2,0,0)⟶(2,1,1)⟶(0,1,0)⟶(1,1,1)⟶(0,0,0)

从上述的两个例子可见,许多实际问题是不能用数值计算来解决的,而且,它们的数据往往是大量的。所以,要使计算机高效正确地解决问题,对数据如何表示、组织、存储以及如何操作等问题的研究就显得越来越迫切,越来越重要了。

60年代初期,还没有独立的"数据结构"课程,但有关的内容已散见于操作系统、编译原理和表处理语言等课程之中。1968年,美国一些大学的计算机科学系的教学计划中,确定"数据结构"为一门课程,但对课程的内容范围仍没有作明确规定。当时,数据结构几乎和图论,特别是表和树的理论是同义语。以后,数据结构的概念被不断扩充,包括网络、集合代数论、关系等现在称之谓"离散数学结构"的内容,它与现在数据结构的某些内容合在一起,被称为"数据结构"。由于数据的加工处理是在计算机中进行的,因此,除了研究数据本身的数学性质外,还必须考虑数据的存储结构,这就进一步扩大了数据结构的内容。自从1968年,美国计算机科学的著名教授唐·欧·克努特(D. E. Knuth)所著的"计算机程序设计技巧"(The Art of Computer Programming)问世以后,逐渐将数据的数学概念及性质等内容独立出来,形成了现在的"离散数学结构",而把数据的逻辑结构、存储结构以及对每种结构所定义的运算组成了"数据结构"的主要内容。此后,各大学纷纷开设"数据结构"课程,我国高校在70年代后期开始陆续开设该课程。

数据结构与数学、计算机软、硬件,特别是计算机软件有着密切的关系,它是计算机专业的一门核心课程;是编译原理、操作系统、数据库等课程的基础;也是设计和实现系统程序及大型应用程序的重要基础。

数据结构在计算机科学中的地位是十分重要的。随着计算机科学的飞速发展,特别是人

工智能、图形、图像、声音和自然语言的计算机处理等领域研究的不断深入，数据结构这门新兴的学科将更显出勃勃生机，得到进一步的发展。

## 1.2 数据结构与算法

著名的计算机科学家，PASCAL 语言的发明者 N. 沃思(Niklaus Wirth)教授曾提出一个有名的公式：

算法＋数据结构＝程序

它清楚地揭示了算法和数据结构这两个计算机科学的重要支柱的重要性和统一性。我们不能离开数据结构去抽象地分析求解问题的算法，也不能脱离算法去孤立地研究程序的数据结构。N. 沃思教授还说，不了解施加于数据上的算法就无法决定如何构造和组织数据，反之，算法的选择常常在很大程度上要依赖于作为基础的数据结构。

### 1.2.1 什么是数据结构

在计算机科学中，**数据**(data)是计算机程序加工处理的对象。抽象地说，数据是对客观事物所进行的描述，而这种描述是采用了计算机系统能够识别、存储和处理的形式来进行的。例如，计算一个几何图形面积的程序，其加工处理的数据是实数和整数；编译一个程序所处理的数据是字符串。因此，对计算机而言，数据的含义极为广泛，如数字、字符、图形、色彩、声音等都是数据。

我们把组成数据的基本单位称为**数据元素**(data element)，数据结构要研究的数据不是一二个孤立的数据，而是大量的相互关联的数据。数据元素之间存在的相互关系称为结构。数据元素之间抽象化的相互关系称为数据的**逻辑结构**(logical structure)，这种相互关系可用一组运算及相应的运算规则来描述。通常，把这种数据逻辑结构简称为数据结构。例如，有一本电话号码本，我们相应地构造一个一维数组，它的每个数据元素是一个数对$(a_i,b_i)$。其中，$a_i$ 和 $b_i$ 分别表示第 i 个用户的名称和对应的电话号码。对于这个一维数组：$(a_1,b_1)$，$(a_2,b_2),\cdots,(a_i,b_i),\cdots,(a_n,b_n)$(其中，n 为用户个数)，我们可以定义若干运算及相应的运算规则，如进行查找，插入或删除某个数据元素等。这样，就构成了上述电话号码本的数据结构。

在设计算法和程序时，不仅要考虑数据的逻辑结构及其运算，而且要考虑这些数据在计算机存储器中的存储方式。这就是数据的**存储结构**(storage structure) 或称**物理结构**(physical structure)。一种逻辑结构可以通过映像得到与它相应的存储结构。本书中将要讨论的逻辑结构有线性的，包括栈、队列和串等；也有非线性的，包括树和图。它们分别通过顺序和非顺序的映像，可以得到不同的存储结构。

我们从集合论的观点出发，可以对数据结构作出形式化的描述。数据结构是一个二元组，即 D. S＝(D,R)

其中，D 是数据元素的有限集合，R 是 D 上关系的有限集合。两者的有机结合，就是数据结构。数据结构要研究的不仅是数据的逻辑结构和物理结构，还要研究相应结构上数据的有关运算。

在数据结构中，往往涉及**数据类型**(data type)和**数据对象**(data object)的概念。数据类

型是指某种程序设计语言所允许使用的变量种类。如在C语言中，有整型、实型等基本类型，还有数组、结构体等构造类型和指针类型。一个数据类型不仅定义了相应变量可以设定的值的集合和存储方法，而且还规定了对变量允许进行的一组运算及其规则。所以，我们可以把数据类型看作是程序设计语言中已经实现了的数据结构。数据对象是指某种数据类型的数据元素的集合，是数据的一个子集。如整数的数据对象是集合I={0，±1，±2，…}，英文字母(大写)的数据对象是集合C={A，B，…，Z}。

### 1.2.2 算法的概念和特性

**算法**(algorithm)的意义非常类似于处方、过程、方法和规程。通俗地说，所谓算法是指为解决给定问题而需施行的有穷操作过程的描述。在计算机科学中，算法则是描述计算机解决给定问题的操作过程。

通常，一个算法必须具备以下五个重要特性：

1. 有穷性

有始有终是算法最基本的特征。一个算法必须在它所涉及的每一种情形下，都能在执行有穷步操作之后结束。有的运算过程，操作步骤看似有限，但不能保证它在动态环境下实际执行次数的有穷性。因此，判断一个算法的有穷性应对算法所涉及的各种情形作动态分析，从而作出判断。看下面两个例子。

**例3** 一个非算法的计数过程

(1) 开始

(2) n<=0

(3) n<=n+1

(4) 重复(3)

(5) 结束

粗一看，这个过程仅有五个步骤，是有穷的，但事实上，该过程一开始，就永无休止。因而在动态执行中它并不具备有穷性，它不是一个算法。当然，只要对上述过程稍加修改，限定其计数之上限，即可使之成为一个算法。这就是下面的例4。

**例4** 不超过一万次的计数算法

(1) 开始

(2) n<=0

(3) n<=n+1

(4) 若n=10000，则顺次执行(5)，否则重复(3)

(5) 输出n

(6) 结束

2. 确定性

算法的每一步操作，其顺序和内容都必须确切定义，而不得有任何歧义性。

3. 数据输入

一个算法有n(n≥0)个初始数据的输入。

4. 信息输出

算法是用来解决给定问题的，所以，一个算法必须将人们所关心的信息输出。也就是说，

一个算法必须有一个或多个有效信息的输出，它是同输入数据有某种特定关系的信息。

5. 能行性

一个算法的任何一步操作都必须是可行的。即必须是可以付诸实施并能具体实现的基本操作。也就是说，每步操作均能由计算机（或人们用纸和笔）操作有限次即可完成。

### 1.2.3 数据结构与算法的关系

数据结构和算法是计算机科学的两个重要支柱。它们是一个不可分割的整体。

计算机求解问题的过程，从某个角度来说，无非是：

初始数据输入——→计算机处理——→结果输出

对于一个给定问题的初始数据，如何组织，在计算机内如何存储，以节约存储空间和便于计算机处理；同时，如何选择合适的算法以提高求解问题的可靠性和效率，这都是至关重要的。我们不能强调了一面而轻视另一面，也不能把两者分开来作孤立的讨论。

比如前面所举的教材的计算机管理问题，我们用一个表来存储初始数据，现要查询《数据结构》教材的库存量以决定是否要征订，如何进行呢？如果初始数据在表内的存放是随机的，即没有什么规律的，那么，我们的查询算法只能是在表中顺序地逐行逐行地查找，显然其效率是不高的。如果想提高效率，则必须对数据的组织和存储作相应的调整和改进。比如，可以让教材按教材名的字典顺序排列或按专业归类存储，并建立索引表，同时选择更有效的算法。这样，就可大大提高查询的速度。

由此可知，不同的算法必须选用相适应的数据结构才能发挥作用，也就是说，数据结构的不同，直接影响算法的选择和效率。

所以，对于一个特定问题，究竟采用何种数据结构和选用什么算法，不能一概而论，需要具体问题具体分析。要看问题的具体要求和现有的各种条件，把数据结构和算法这两者有机地结合起来考虑，这样才能设计出较好的计算机程序。

## 1.3 抽象数据类型

### 1.3.1 抽象——程序设计最基本的思想方法

对于一个复杂问题，往往会涉及到许许多多的因素。为了使问题得到简化，以便建立相应的数学模型进行深入研究，并进而加以解决，通常采用“抽象”这一思想方法，抽取反映问题本质的东西，舍去其非本质的细节。

在计算机软件的发展过程中，抽象这一思想方法得到了充分的应用。请看以下程序设计的几个阶段：

计算机机器指令；

符号化的汇编语句；

高级语言的执行语句；

高级语言的过程（或函数）模块；

面向对象的软件开发系统。

在这些阶段中，后一阶段都是在前一阶段的基础上经过进一步的抽象后得以建立起来

的。

通过一步步的抽象，不断地突出“做什么”，而将“怎么做”隐蔽起来，即把一切用户不必了解的细节封装起来，从而简化了问题。所以，抽象是程序设计最基本的思想方法。

### 1.3.2 抽象数据类型

**抽象数据类型**(Abstract Data Type，简记为 ADT) 是定义了一组运算的数学模型，它是程序设计语言中数据类型概念的推广，是初等数据类型基础上的进一步抽象。

抽象数据类型和数据类型在本质上是一致的。例如，“整数”数据类型，各种程序设计语言中都定义为初等的数据类型，但我们也可把它看作是一种抽象数据类型，因为整型数据在不同的程序设计语言或不同的计算机上的具体表示及操作是不相同的，而其“整型”数学特性则是完全一致的。而且，程序设计语言中对整数一般都定义了四则运算、比较、输入和输出等操作，用户在使用时只需知道其功能及使用规则就可以了，而不必知道数据在内存中的具体表示和操作的具体实现过程。当然，抽象数据类型主要是指用户在程序设计时自己定义的数据类型。

下面我们举例说明抽象数据类型的使用。

设某软件中需涉及矩阵的建立、转置、相加、相乘等运算，则可把矩阵定义成一个抽象数据模型，不妨记为 MATRIX，关于矩阵元素的具体类型是什么，是整型还是实型？则是非本质的细节，在此阶段不去管它，即实现了数据抽象。

我们把有关的操作用函数名表示之：

| | |
|---|---|
| CREATE(m,n) | 建立一个 m 行 n 列的矩阵 |
| TRANS(a,b) | 矩阵 a 是已知的，函数求得矩阵 a 的转置矩阵 b |
| MATADD(a,b) | 实现矩阵相加：a<=a+b |
| MATMULTIPLY(c,a,b) | 实现矩阵相乘：c<=a * b |

各个函数的具体算法如何？技巧怎样，甚至采用何种计算机语言，在此阶段也不必去理会，即实现了过程(函数)抽象，于是，软件中遇到矩阵上述操作的地方则分别用相应的函数调用代替。

上面是对矩阵这一抽象数据类型的定义或称为规范说明，如何实现则要根据具体的软、硬件环境来确定具体的算法和数据结构。

一个软件系统可看作是由数据、操作过程和接口控制所组成的，当使用自顶向下，逐步求精的方法设计软件时，为便于从宏观上把握全局，要对它们进行抽象，即数据抽象、过程抽象和控制抽象。抽象数据类型不仅包含数学模型，还包含了模型上的运算，所以它将数据抽象和过程抽象结合为一体。抽象数据类型确定了一个数学模型，但将构成模型的具体细节加以隐蔽；它定义了一组运算，但又将运算的实现过程隐蔽了起来。

运用抽象数据类型的概念来设计软件的过程可用图 1.3 来表示。

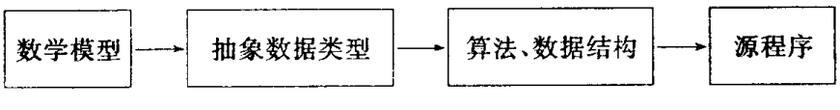


图 1.3 软件设计过程

一个抽象数据类型的具体实现是可以多种多样的，而在软件中使用该抽象数据类型的地方则可以把它看作一般的初等类型，而不必管它的具体实现方法是什么，对抽象数据类型的定义及有关操作的设计、修改、完善等工作仅局限于相应的模块中。所以，抽象数据类型概念的引入，降低了大型软件设计的复杂性，使软件设计中普遍遵循的模块化、信息隐蔽、代码共享等思想得到更充分的体现。

## 1.4 算法的描述和分析

数据结构是一门具有较强实践性的学科。通过该课程的学习，应使学生能运用数据结构的知识和技巧更好地进行算法和程序的设计。所以，我们在讨论各种数据结构的基本运算时，都给出了相应的算法。对于算法的描述，我们力求做到通俗易懂和便于自学，所以采用文字框图进行图示。读者在掌握和理解了框图所示的设计思想后，可以较方便地使用自己熟悉的算法语言来编制源程序。另外，考虑到C语言是当前国际上广泛流行和很有发展前途的计算机高级语言，它能较好地体现结构化程序设计的原则，并且简洁明了，便于教学，所以本书中大部分算法在给出文字框图的同时还给出了用C语言编写的源程序片断。对一些较长的程序则收在附录中。书中虽然仅给出程序片断，但程序本身已在计算机上调试通过。

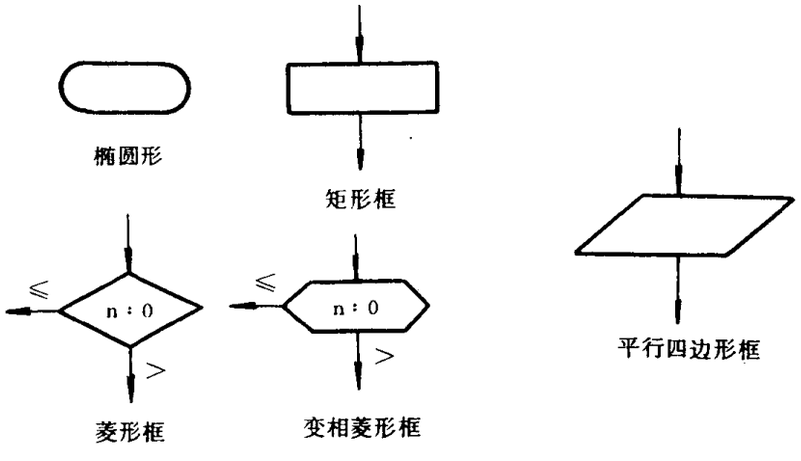


图 1.4 框图所用图形

关于C语言的内容，读者可以参阅有关书籍，这里不作介绍。本书框图中使用的图形符号见图1.4，它们各自的意义是：

1. 椭圆框：表示算法的“开始”或“结束”，框中用文字标明。

2. 矩形框：表示某些操作，如赋值、组织循环等。统称为操作框。

3. 平行四边形框：表示输入/输出操作，即提供运算所需的数据或记录运算结果的输出。

4. 菱形框(包括变相菱形框)：是判别框。框中符号“：”表示比较。例如n：0表示n与0相比较，比较的结果写于框外连接线条的旁边。带箭头的线条表示算法或程序走向，写于其旁的判断结果就是算法或程序的分支走向应满足的条件。

另外，对算法（或程序）的分析和评价通常较复杂。一般需考虑正确性、最优性、可读性、可维持性、运算量及占用存储量等诸多因素。为了简化讨论，本书主要考虑其中的两条标准：一是算法实现所需的存储量，其二是算法实现所需的运算量。我们用问题的某个参数的函数来加以估算，假设问题规模的参数为 n，此参数可以是矩阵的阶、线性表的长度、图的顶点数等显示该问题规模大小的参数。那末，所选数据结构上执行某种操作所需要的存储量及运算量是 n 的什么函数呢？我们引进记号“O”（读作“大 O”），对这些函数作数量级的估算。如，对于下述三个简单程序段：

```
(1) x=x+1;
(2) for(i=1;i<=n; i++)
        x=x+1;
(3) for(i=1;i<=n; i++)
      for(j=1;j<=n; j++)
        x=x+1;
```

在程序段 1 中，语句 x=x+1 不包含在任何循环体之中，则此语句只执行一次，运算量可记为 O(1)。在程序段 2 中，上述赋值语句在 for 循环之中，所以要执行 n 次，其执行时间和 n 成正比，运算量可记为 O(n)。在程序段 3 中，x=x+1 语句要执行 n * n 次，其执行时间和 $n^2$ 成正比，故运算量可记为 $O(n^2)$。然而，要对一个算法的运算量作仔细的分析是很复杂的，而且也不是本课程的主要内容，所以书中对一些算法的性能评价只根据算法中执行次数最多的语句来估算该算法的运算量的数量级。对于算法或程序所需的存储量，本书也作类似处理。

## 习 题 一

1. 举例说明什么叫数据、数据元素、数据类型和数据结构？
2. 数据结构研究的主要内容是什么？
3. 简述数据类型、数据结构和抽象数据类型之间的区别和联系。
4. 分析下列程序的运算量

```
main()
  { int i, j, p;
    for(i=1;i<=9;i++)
      printf("%d",i);
    for(i=1;i<=2;i++)
      printf("\n");
    for(i=1;i<=9;i++)
      { for(j=1;j<=i;j++)
        { p=i*j;
```

```
            printf("%5d",p);
        }
    for(j=1;j<=2;j++)
        printf("\n");
    }
}
```