21世纪高等学校规划教材

SHUJU JIEGOU

数据结构

(C语言版)

王聪华 主编
杨晓波 胡永 王弘 副主编

中国电力出版社
http://jc.cepp.com.cn

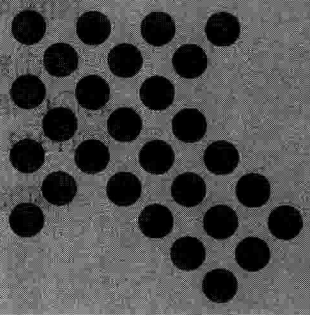
21世纪高等学校规划教材

SHUJU JIEGOU

数据结构

(C语言版)

主　编　王聪华
副主编　杨晓波　胡　永　王　弘
编　写　雷　萌　张晓煜
主　审　杨兴运



中国电力出版社
http://jc.cepp.com.cn

内 容 提 要

本书为21世纪高等学校规划教材。本书主要内容包括数据结构的概念和术语、线性表、栈与队列、串与数组、广义表、树与二叉树、图及应用、查找和排序。每章之后配有多种类型的习题，方便读者加深对所学基本概念的理解，巩固所学知识。与本教材配套的还有实验指导书，方便读者上机训练和实践。本教材力求语言通俗，讲解细腻，概念表述简明、严谨；力求突出重点，分解难点，算法表达精练、易读易懂；注重应用实践，注意训练学生的应用和实践能力。

本书可作为应用型本科计算机及相关专业的教材，也可供自学人员及工程技术人员学习参考。

图书在版编目（CIP）数据

数据结构：C语言版 / 王聪华主编. —北京：中国电力出版社，2010

21世纪高等学校规划教材

ISBN 978-7-5083-9853-2

Ⅰ. ①数… Ⅱ. ①王… Ⅲ. ①数据结构－高等学校－教材②C语言－程序设计－高等学校－教材 Ⅳ. ①TP311.12 ②TP312

中国版本图书馆CIP数据核字（2010）第000395号

中国电力出版社出版、发行

（北京三里河路6号 100044 http://jc.cepp.com.cn）

汇鑫印务有限公司印刷

各地新华书店经售

*

2010年2月第一版 2010年2月北京第一次印刷

787毫米×1092毫米 16开本 13.5印张 324千字

印数 0001—3000册 定价 **21.60** 元

前　言

计算机从诞生的那一天起就与数据处理紧密相关，最初的想法是用于解决科学上复杂、繁重的数值计算问题，以减轻脑力劳动、提高运算效率。随着计算机科学技术的发展，计算机解决问题的范围也更加广泛，目前已涉及字符、文字、图形、图像、声音等非数值计算问题。数据结构主要以非数值计算问题为研究对象，研究数据的表示、数据之间的关系和数据处理的算法。它是计算机专业及其他信息类专业的重要基础课和核心课，也是软件设计的重要支撑基础。

本教材编写的主导思想是解决欠发达地区高校学生层次较低、基础较差、学习数据结构课程有一定困难的问题，定位于应用型本科层次。教材不拘泥于“五脏齐全”，主要着眼于数据结构的基本概念、知识点和基本的算法，以及数据结构解决问题的思想和方法，重点训练和培养学生程序设计的基本能力和思维，力求为操作系统原理、编译原理等后续课程的学习奠定基础。

依据主导思想，教材在写作上语言通俗，讲解细腻，表达精练，尽量避免出现深奥的专业术语。在内容安排上结构清晰、层次分明，力求突出重点，分解难点，算法表达精练、易读易懂；注重应用，讲究实用。

教材共分为9章。第1章介绍数据结构的概念和基本术语，算法和算法分析；第2章介绍线性表的顺序存储和链式结构，以及基本运算的实现；第3章介绍栈和队列概念、存储、基本运算和应用；第4章介绍串的概念、存储和基本运算，模式匹配算法，数组的概念和存储，特殊矩阵和稀疏矩阵的存储；第5章介绍树和二叉树的概念、存储和基本运算，线索二叉树的概念、创建和遍历算法；第6章介绍图的概念、各种存储结构和图的应用，图的遍历和图的应用；第7章介绍各种查找算法的实现过程；第8章介绍各种内排序算法的实现；第9章介绍各种文件组织方式。

教材每章之后都配有一定数量的习题，习题种类丰富，既考虑到数据结构各章节基本概念和知识点的掌握，也考虑到算法设计能力的训练和提高，读者可以通过做练习和上机调试程序，加深对概念和算法的理解，巩固和提高所学知识，提高分析问题和解决问题的能力。与本教材配套出版的还有实验指导书，该书中有大量的习题参考答案、实验实例和模拟试卷，以供读者配合数据结构课程的学习。

教材中的全部算法用 C 语言描述，配套实验指导书中所涉及的算法实现全部以 Visual C++的环境为基础，读者的学习可以完全依托于 Windows 操作系统。使用本教材授课时，实验题目可采用配套实验指导书的内容。与本书配套的电子教案可去 http://jc.cepp.com.cn 下载。

本书由杨兴运主审。本教材的作者都有着多年的数据结构课程教学经验，并对教材的编写和算法的设计倾注了大量的精力，但课程所涉及的内容较多，算法数量大、细节多，难免存在错误和不足，敬请同行专家和读者批评指正。电子邮箱为 wangcshui@126.com。

作　者

2009 年 10 月

目　录

第1章　绪　　论

现实世界中包罗万象的客观事物要被计算机处理就都必须转化为计算机所能识别的数据，而计算机对数据的加工和处理都与数据的表示、数据之间的关系及数据的存储表示密切相关，这些表示、相互关系和存储的解决好坏直接影响到数据能否被计算机处理以及处理的效率。为此，有必要研究数据的表示、数据之间的关系以及在计算机中的存储表示，以便设计出可靠性强、运行正确且高效率的算法以及程序。本章以数据结构主要研究的问题入手，介绍数据结构相关的基本概念、数据结构的概念、算法概念、算法描述方法及算法分析。

1.1　数据结构研究的问题

随着计算机科学技术及其他相关学科的不断发展，计算机的应用已经从最初的数值计算问题转向了非数值计算的处理，其数据处理的对象已经涉及字符串、图形、图像、语音等复杂的数据，计算机应用也已广泛应用于办公自动化、金融银行、建筑设计、工业制造和企事业管理等各个领域，在未来的计算机应用领域中应以非数值计算问题为重点。数据结构正是研究非数值问题的非常重要的学科，它在整个计算机学科理论和技术中占有极其重要的地位。数据结构课程不但是计算机专业重要的基础课和专业课，也是计算机应用和软件设计方面的重要基础，在计算机应用的各行各业中凡是属于软件设计和开发的问题都离不开数据结构的知识。要学好数据结构，首先必须明确数据结构所研究的问题，以下通过几个典型的例子来说明数据结构研究的问题。

【例 1-1】　高考成绩查询系统。

我国普通高等学校统一考试考生成绩单见表 1-1，考生需要输入账号、密码和考生号登录，才可以查询。据统计，我国每年报名参加高考的考生人数约为 1 千多万，为了快速、高效地对表 1-1 进行查询，必须考虑两个方面的内容：①数据的组织；②查询算法的设计。

表 1-1　　高考考生成绩单

姓名	账　　号	考生号	准考证号	语文	数学	外语	综合	总分	排名
齐天	565072119903253328	19283567890567	678234561	126	115	140	286	667	16
……	…	…	…	…	…	…	…	…	…
……	…	…	…	…	…	…	…	…	…

如果表 1-1 中每一行的数据都按输入时的自然顺序排列，那只能按输入的账号逐一查找，可想而知在有上千万行数据的一张数据表中逐一查找的查询效率是相当低的，特别是当查找的数据不存在时，要将上千万行的数据全部搜索一遍，这在网络的查询中是绝对不允许的。如果将输入的数据按照一定的要求有组织的排列，那么查找的效率就大不一样，例如，按账

号从小到大排列，则查找的算法不再是逐一比较查找，可以采用第 7 章中介绍的折半查找，这样即便查找的数据不存在，也不必将上千万行的数据全部搜索一遍。在实际生活中像表 1-1 这样每行数据之间存在先后顺序关系的例子还有很多，例如，学生学籍管理、人事档案管理和图书馆书籍管理等都有类似的数据表，这种只存在前后顺序关系的数据结构称为线性数据结构。

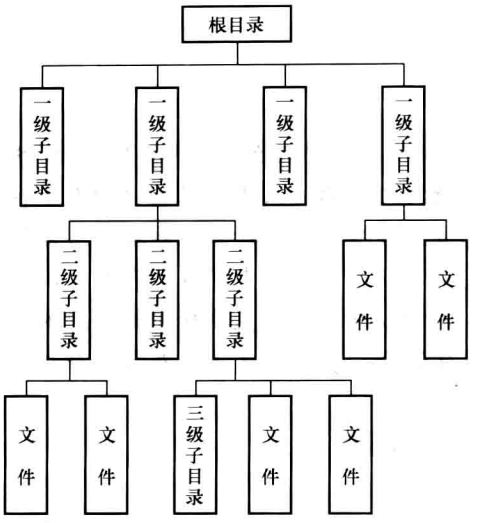


图 1-1　文件组织结构图

【例 1-2】　计算机系统的文件管理问题。

操作系统的主要功能之一是文件管理，为了组织和管理好文件，将文件管理结构组织成为一个多级层次目录结构，如图 1-1 所示。

这种多级目录层次结构形成一棵倒立的树。树中的目录或文件抽象成数据元素，称为结点，结点之间的关系是一对多关系。此类具有一对多关系的数据结构称为树型结构，简称树结构或树。

【例 1-3】　教学计划课程设置问题。设某高校计算机科学与技术专业教学计划中的部分课程设置见表 1-2。

表 1-2　　**计算机专业课程设置**

课程代号	课程名称	先修课程	课程代号	课程名称	先修课程
C_1	计算机应用基础	无	C_7	编译原理	C_2、C_4
C_2	C 语言程序设计	C_1	C_8	操作系统原理	C_4、C_6
C_3	离散数学	C_2	C_9	高等数学	无
C_4	数据结构	C_1、C_2、C_3	C_{10}	线性代数	C_9
C_5	汇编语言	C_2	C_{11}	数据库系统原理	C_4
C_6	计算机组成原理	C_5	C_{12}	数值分析	C_2、C_9、C_{10}

表 1-2 课程设置之间有的存在先后关系，必须按顺序开设，有的没有先后关系，可以并行开课。用一个直观的图形表示课程之间的先后顺序关系如图 1-2 所示。

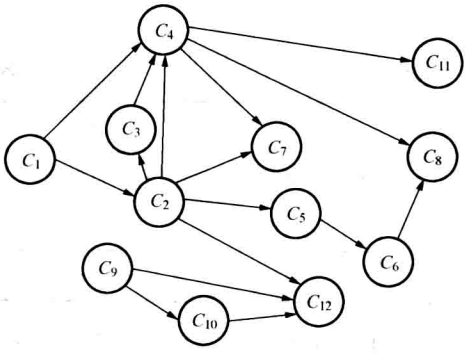


图 1-2　课程设置关系图

从图 1-2 中可以看出，一门课程可以有多门先行课，也可以有多门后继课。这种元素之间存在多对多关系的数据结构称为图结构或网状结构。图结构是一种最复杂的数据结构。

由以上三个例子可以看出，现实生活中有许多问题是不能通过抽象地列出数学方程的数值计算来求解的，大量的非数值计算问题的解决首先要考虑建立数据之间的关系，进而研究解决在这些关系上的运算。因此，可以说数据结构是研究非数值计算的程序设计问题中计算机的操作对象以及它们之间的关系和操作的学科。

对数据结构学科概念的理解应抓住以下三个要点。

（1）数据结构研究的对象是程序设计中非数值计算的操作对象。非数值计算的操作对象不能使用现有的数学模型计算，需要专门的研究方法，而在计算机应用中，大约90%的计算和数据处理问题都是非数值型数据。

（2）数据结构研究的是非数值操作对象之间的关系。有逻辑关系和存储关系，或称逻辑结构和存储结构。

（3）数据结构研究的主要问题是非数值对象的算法设计。

在数据结构的课程中，始终贯穿着以上三个方面。

1.2 基本概念与术语

1.2.1 数据的有关概念

1. 数据

数据（Data）是对客观事物的符号表示，是所有能够被计算机识别、存储和加工处理的符号的总称。数据是一个抽象的概念，不仅仅指数值数据，其广泛的含义是非数值数据，如文字、图形、图像、音频、视频等各种媒质载体。数据是信息的载体，信息是数据的内涵。

2. 数据项

具有独立逻辑含义且不能再分解的数据称为数据项（Data Item）。要认识一个客观对象，必须抓住该对象的一个个特征，而要表示一个对象，需要使用一些抽象的数据来表示。这些表示客观对象某个特征的数据就称为数据项。数据项也称为原子数据项，是指它不能再分解成更小的、具有独立逻辑意义的数据单位。数据项是客观对象某一特征的数据表示，是数据处理的最小数据单位。

3. 数据元素

数据元素（Data Element）是相关数据项的集合。数据元素是同一对象、多个特征的抽象数据表示。从逻辑角度看，一个数据项只能反映客观对象的一个侧面的特性，多个数据项可以反映对象的多个特征，它们作为一个整体，才能全面刻画该对象的性质和状态。从存储角度看，数据元素是作为一个完整的数据存储的，在数据处理过程中，数据的读写和存储都是以数据元素为基本单位的。所以，把数据元素看成是数据处理的基本数据单位。数据元素在不同的学科和不同场合下还有其他名称。例如，在数据库中称为记录，C 语言中称为结构，在数据结构的树和图中分别称为结点和顶点等。

4. 数据对象

具有相同性质的数据元素构成的集合称为数据对象（Data Object）。一个数据元素表示一个客观对象，描述一类在某些侧面具有共同特征的对象所得到的一组数据元素构成的一个集合就是一个数据对象。如果说数据元素是个体性概念，那么数据对象就是集合性概念。这是二者的本质区别。在计算机的数据存储管理中，一个数据对象被组织成一个文件（或表）的形式进行整体存储、维护并永久保存。因此，数据对象是数据存储管理单位。

数据项、数据元素、数据对象既是数据逻辑组织的层次单位，又是计算机存储管理数据的层次单位。三者之间的关系是：相关数据项的集合构成一个数据元素，相关数据元素的集

合构成一个数据对象。再高一层的数据单位是数据库，它是相关数据对象的集合。数据库层的存储处理由“数据库系统原理”学科专门讨论。

1.2.2 数据结构的相关术语

1. 数据结构

数据结构（Data Structure）是指相互之间存在着一种或多种关系的数据元素的集合。这里强调数据元素集合和一组关系的集合两点。简单地说即：数据结构=数据元素集合+一组关系集合。形式地表示为

$$Data\_Structure=(D, R)$$

其中：D 表示数据元素集合，R 表示数据元素之间特定关系的集合。

例如：$D=\{a, b, c, d, e, f, g, h\}$，$R=\{(a, b), (b, c), (c, d), (d, e), (e, f), (g, h)\}$，$DS=(D, R)$ 是一个数据结构。

又如一种数据结构 tree=（D，R），其中，$D=\{01, 02, 03, 04, 05, 06, 07, 08, 09, 10\}$，$R=\{<01, 02>, <01, 03>, <01, 04>, <02, 05>, <02, 06>, <03, 07>, <03, 08>, <03, 09>, <04, 10>\}$，可以表示成如图 1-3 所示的树结构图。

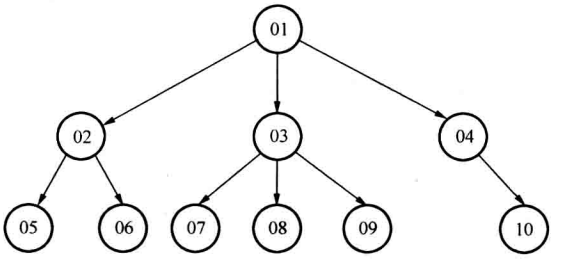


图 1-3 一种树结构

这里的数据结构概念与 1.1 节中数据结构作为一门学科所描述的概念是两个不同的术语，此处数据结构的概念，着重强调数据及其关系，是数据组织格式的一种界定，它包含逻辑结构和物理结构两个层面的含义。数据的逻辑结构是指在完全不考虑机器实现的情况下，单纯从数据本身及其逻辑关系出发，讨论其组织描述问题。其实质是对客观对象及其关系进行抽象化、数据化的表示。数据的物理结构（也称存储结构）是指数据的逻辑结构在计算机内部的存储表示。存储结构包括两个方面的内容，即数据元素本身的存储方式和关系的表示。

2. 逻辑结构

根据数据元素之间特定的逻辑关系不同，数据的逻辑结构（Logical Structure）分为四种。

（1）集合结构（Set Structure）。该结构的数据元素之间没有任何联系，只是“属于同一个集合”，是一种完全松散的结构。

（2）线性结构（Liner Structure）。该结构的数据元素之间存在一对一的序关系，这种一对一的序关系称为完全序。

（3）树结构（Tree Structure）。该结构的数据元素之间存在一对多关系，这种结构表示具有层次结构的数据是最佳形式。

（4）图结构（Graph Structure）。也称网状结构，该结构的数据元素之间存在多对多关系，这种结构用于表示数据之间存在的复杂网络关系。

以上的逻辑结构中，树结构和图结构是非线性结构。四种逻辑结构的图形表示如图 1-4 所示。

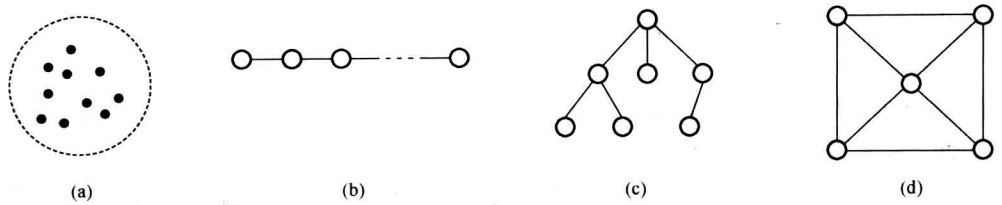


图 1-4　4 种逻辑结构的示意图

（a）集合结构；（b）线性结构；（c）树结构；（d）图结构

3. 物理结构

按照数据元素的存储方式和关系表示方法的不同，数据的物理结构（Physical Structure）可分为以下四种。

（1）顺序存储结构（Sequential Storage Structure）。简称顺序结构，这种存储方式是用一块较大的连续存储区域，按照预先约定的某种次序，将数据元素逐个连续地存放在一起，用物理位置的相邻关系表示数据元素之间的逻辑关系。这是一种既简单又经济的存储方式。

（2）链式存储结构（Link Storage Structure）。简称链式结构，这种存储方式是对每一个数据元素用一块较小的连续区域存放，称为一个结点（Node），不同的数据元素存储区可以连续，也可以不连续（离散存储），然后用指针（Pointer）表示逻辑关系。在结点中设置一个或多个指针，指向它的前驱或后继元素的地址（Address）。

（3）索引存储结构（Index Storage Structure）。简称索引结构，这是一种顺序加链式的存储方式，数据元素按顺序结构存放，然后将每个数据元素的关键字和存储地址构造一个索引表，单独存储（按关键字值有序存放）。这种存储结构不反映关系，用于存储集合结构。

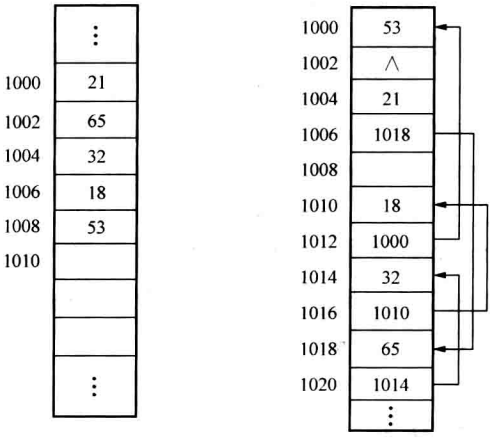


图 1-5　存储结构示意图

（4）哈希存储结构（Hash Storage Structure）。简称哈希结构，这种存储方式是数据元素按顺序或链式存储，并在数据元素的关键字与存储地址之间建立一种映射，这种存储结构也不表示数据元素之间的关系，适合于集合结构的存储。

例如：将数列 21，65，32，18，53 分别按顺序和链式存储，在机器内的存储如图 1-5 所示。

顺序存储结构和链式存储结构是最基本、最常用的存储结构，也是索引存储结构和哈希存储结构的基础，这两种结构在具体实现时要用到前两种结构。

1.2.3　数据类型的概念

数据类型是数据结构中最重要的概念之一，为什么要区分数据类型，有下面几个原因：一是数据类型决定运算，不同类型的数据可以施加的运算是不同的；二是数据类型决定存储分配，不同类型的数据在存储时分配的存储空间的大小也不同；三是数据类型决定数据的访问，读写数据时是按存储地址进行访问的，而数据的存储地址计算依赖于数据的类型。所以，

构造合适的数据类型，对于程序设计是非常重要的。

1. 数据类型

一组值的集合与定义在该集合上的一组运算（或操作）称为一个数据类型（Data Type）。简单地说即：数据类型=数据值的集合+运算的集合。

这里定义的运算多数是非数值计算性的，与逻辑结构有关的，机器能够支持实现的操作。

数据类型分为原子类型和结构类型。原子类型是机器本身能够实现的具体数据类型，如各种程序设计语言中提供的标准字符型。结构类型是通过原子类型构造出的更为复杂的数据类型，如 C/C++语言中的数组、结构体和联合等，这些类型机器本身并不支持，但可以通过程序和原子类型实现。

数据类型强调运算，它把数据和运算融合在一起作为一个整体对待，这是它与数据结构概念的主要区别。

例如：C 语言中的整型数据是一个数据类型，它由计算机所能表示的整数集合与定义在该集合上的+、–、*、/、%五种运算构成。布尔类型是由 0 和非 0（表示假和真）以及三种逻辑运算&&、||、！构成的一个数据类型。

2. 抽象数据类型

一个数学模型和定义在该模型上的一组操作（运算）称为一个抽象数据类型（Abstract Data Type，ADT）。简单地说，抽象数据类型就是数据结构与数据运算的统一体。即：抽象数据类型=数据结构+数据运算。

定义中使用了数学模型一词，应理解为数据是抽象的和有结构的，这些抽象数据是用变量和它们的定义域表示的，而不是具体值的集合。结构是对数据之间的关系给予确切的定义。因此，抽象数据类型应包括三个方面的含义：数据的结构、数据的取值范围和数据的运算。抽象数据类型概念与数据类型概念没有本质上的差别，只是在不同层次上的抽象描述而已。数据类型是对计算机硬件系统已经实现或支持的一些具体数据类型的描述，它是对机器硬件系统能够存储表示和操作运算的一种抽象。而抽象数据类型是在更高一层的抽象层上的数据类型，不考虑机器如何实现，因而其适应范围更加广泛。两者的关系就像面向对象程序语言中的类与对象的关系一样，应注意它们之间的细微差别。抽象数据类型的定义可以包括数据对象、数据关系和数据基本操作三个方面的内容，其详细要点读者可以参阅参考文献［1］。

1.3 算法和算法分析

1.3.1 算法及其特性

算法是数据结构中最为重要的概念，但又是难以严格定义的概念，在此仅给出算法概念的描述性的定义，并通过它的一些重要特征加以说明，从中领会和理解其内涵。

1. 算法的概念

对某个特定问题求解步骤的一种描述称为算法（Algorithm）。

算法概念分广义和狭义两个层次。广义地说，解决每个问题或做任何一件事情的过程，预先都必须有计划、步骤和次序，将整个解决问题过程的顺序和步骤完整地描述出来就是算

法。狭义的算法概念是指在数据结构中，利用计算机解决问题的步骤描述。

2. 算法的特性

算法除了求解步骤的顺序性之外还具有以下五个重要特征。

（1）有穷性。是指一个算法的操作步骤必须是有限的。换句话说，任何问题在经过有限个步骤之后，一定能够完成，因为用计算机求解问题不允许无限计算下去却永远得不到结果。

（2）确定性。它要求每一个步骤所要执行的操作或运算必须是完全确定的，不能似是而非。因为计算机不像人一样能灵活判断分析，只能按算法的步骤机械地执行规定的动作，当某个步骤的操作有歧义性时，机器将不知道该如何做。

（3）可行性。这是指算法的每一个步骤，计算机都能执行。计算机所能执行的动作是预先设计好的，一旦出厂就不会改变，所以，设计算法时应保证每个步骤都必须能用计算机所能执行的操作命令实现。

（4）输入性。一个算法应该有 0 个以上的输入数据。算法是数据加工处理过程的控制程序，而数据是被处理的原材料，执行一个算法首先应提供这些数据，当然，有些问题的算法可能不需要输入数据，这并不意味着没有数据处理，而是被处理的数据已经在机器内，或者是在加工处理中生成的。

（5）输出性。每个算法必须有 1 个以上的输出数据。无论如何，算法执行完后，一定要给出一个结果，或者是写出结果数据，或者是一个明确的是或不是、成功或不成功、对或错的回答信息，或者是算法本身出现的错误提示等。

1.3.2 算法的描述

（1）自然语言描述。用人类通常使用的语言表示算法。该方法的优点是自然、方便，易读、易理解；缺点是存在歧义性。

（2）图形描述。用某些图形表示算法步骤和控制流程。该方法的优点是简单、直观、形象、容易阅读理解；缺点是对于复杂的算法可能产生很大图形，画图和阅读不方便。

（3）某种类语言描述。类语言是介于自然语言和形式化的编程语言之间的一种专用语言。如常用的类 C、类 C++、类 Pascal、类 Java 等。它们形式上像程序设计语言，但不拘泥于严格的语法规则，具有自然语言那种较灵活自由的性质。

（4）某种编程语言描述。用某种编程语言描述就是程序。可以直接编译执行。由于编程语言的严格语法约束，一般不使用，只有当算法上机实现时才使用。

以下用实例分别说明自然语言和类 C 语言描述算法的基本方法。

【例 1-4】 假设有 n 个整数顺序存放在数组 $a[0..n-1]$中，设计算法从 n 个数中选出最大值和最小值。

（1）用自然语言描述。

①已知 $a[0]$，$a[1]$，…，$a[n-1]$和 n；

②假定最大值 Max=$a[0]$，最小值 Min=$a[0]$；

③i=1；

④如果 $a[i]$>Max，则 Max=$a[i]$，如果 $a[i]$<Min，则 Min=$a[i]$；

⑤i=i+1，如果 i<n，则转④，否则结束，得到最大值 Max 和最小值 Min。

（2）用类 C 语言描述。

```
void GetMax_Min(int a[], int n, int &Max, int &Min)
{                                          //求n个整数的最大值和最小值，并分别存入Max和Min
   Max=a[0]; Min=a[0];                     //最大值和最小值赋初值
   for(i=1; i<n; i++)                      //循环比较得到最大值和最小值
   { if(a[i]>Max)                          //如果第i个值大于最大值
          Max=a[i];                        //第i个值为最大值
     if(a[i]<Min)                          //如果第i个值小于最小值
        Min=a[i];                          //第i个值为最小值
   }
}
```

用类C语言描述算法时，要特别注意：为了增加算法的可读性，对关键语句要加注释，说明算法的功能。算法中要避免存在逻辑上的错误。

1.3.3 算法的设计要求

实际问题千奇百怪、变化多端，解决问题的算法多种多样，在设计算法的过程中，不同的设计者可以充分发挥自己的智慧、能力、经验和技巧，自由地展示各自的才能。但是也不能完全的自由化，需要遵循一些基本的规则和要求。在算法设计方面至少应符合以下几点要求。

（1）正确性。这是算法设计的最基本最重要的、第一位的要求。只有正确才能保证可信、可靠、可用。否则，一切无从谈起。正确性要求的含义分多个层次：首先是设计层面上的正确性，包括数据类型的建模正确，数据结构的选择正确，算法的策略正确。其次是算法表示层的正确性，包括数据格式定义的正确，算法程序的结构、语法和风格正确，符合规范化要求。最后是结构层次的正确性，结果必须符合实际问题，满足问题或客户的应用要求。

（2）可读性。可读性的含义是指算法思想表达的清晰性、易读性、易理解性、易交流性等多个方面，甚至还包括适应性、可扩充性和可移植性等。早期由于计算机硬件系统在速度、容量、外设等方面的限制，使得算法程序的设计者挖空心思、千方百计地寻找各种特别的算法技巧，以达到节省空间、提高速度的目的，许多漂亮的算法和程序往往令人望洋兴叹，难以读懂，无法交流借鉴，严重影响了其作用的发挥。随着计算机硬件方面的高速发展和软件方面的交流、共享、兼容、复用要求，速度、容量大幅度提高，外设品种和功能丰富多样，人们已经不再看中算法和程序的效率问题，而把可读性提高到了更加重要的地位。请记住，软件是为客户开发的，算法和程序是让他人阅读的。效率和可读性相比，后者更重要。

（3）健壮性。一个算法的健壮性是指它运行的稳定性、容错性、可靠性和环境适应性等。当出现输入数据错误，无意的操作不当或某种失误，软、硬件平台和环境变化等故障时，能否保证正常运行，不至于出现莫名其妙的现象、难以理解的结果，甚至经常瘫痪死机。要树立用户是上帝的理念，时刻想着用户对操作的简单性和方便性要求，保证程序能稳定地运行，适应情况的变化，避免异常现象的发生。

（4）经济性。是指算法的时空效率问题。虽然强调算法的可读性，但并非不考虑效率。应该在保证可读性的前提下，尽可能做到节省运行时间和存储空间。尽管计算机在硬件方面规模不断提高，但总有极限值，而所要解决问题的规模和复杂性也越来越大，软件的规模发展更快，时间和空间永远满足不了要求，因此，经济性仍然是不能忽略的问题之一。

1.3.4 算法分析

算法的性能分析是算法设计中非常重要的方面。所谓算法性能分析，实质上就是评价算法的时空效率问题。评价一个算法的优劣，主要从两个方面考虑，一是执行算法所耗费的时

间，二是算法中所使用的额外存储空间。前者称为时间复杂度，后者称为空间复杂度。

1. 算法的时间复杂度

一个算法执行所使用的时间称为该算法的时间复杂度。对一个算法时间复杂度大小的影响因素主要有四个方面：一是计算机硬件系统的运行速度。二是所使用的软件环境，包括所选择的程序设计语言和相应编译系统的质量问题。不同的编程语言和编译程序所产生的可执行代码的大小与执行速度有较大差别。三是算法本身的策略，采用不同的存储结构和不同的算法过程，是影响时间复杂度最本质的原因之一。四是所处理的数据量的多少，也就是指处理问题的规模。很显然，数据越多，所花费的时间就越多。

算法时间复杂度的大小与算法的策略有着密切关系，同一操作采用不同的算法策略其时间复杂度的差别较大。例如，第 7 章介绍查找算法，对 n 个数据的查找，顺序查找与二分法查找，其查找的效率有较大的差别。若采用顺序查找，需逐个比较进行查找，当找不到时，比较的次数是 n，即时间复杂度 $T(n)=n$。若采用折半查找，每次取中间元素比较，若相等则找到，返回下标，若小于则在左端继续折半查找，若大于则在右端继续折半查找，每次查找范围都缩小一半，比较的次数为 $\log_2 n$。假定 $n=2^k$，则顺序查找和折半查找的算法时间复杂度分别是 n 和 $k+1$。

虽然算法的时间复杂度与以上介绍的多个因素有关，但可以直接控制的因素是算法的策略（算法策略中包含了存储结构的选择）和处理问题的规模。一个算法是由一种编程语言的控制结构（顺序、分支和循环）和原操作（指固有数据类型的操作）构成的，算法时间取决于两者的综合效果。为了便于比较同一问题的不同算法，通常从算法中选择一种对于所研究问题来说是基本操作的原操作，以该基本操作重复执行的次数为算法的时间度量。

假定 t_1，t_2，…，t_m 是 m 个基本操作的时间，它们的执行次数分别为：n_1，n_2，…，n_m，则算法时间 $T(n)=\sum_{i=1}^{m} t_i n_i$，由于一个基本操作的时间 t_i 很难确定，仅以执行次数为准，即以 $\sum_{i=1}^{m} n_i$ 为时间总和，这说明了算法中基本操作重复执行的次数是问题规模 n 的某个函数 $f(n)$。一般情况下，算法的时间度量记为

$$T(n)=O(f(n))\quad(T(n)\text{ 为算法时间总和})$$

从极限的角度来解释 $T(n)=O(f(n))$ 的含义：说时间复杂度 $T(n)$ 是 $O(f(n))$，则存在常数 $c\neq 0$ 和 n_0，使得 $|T(n)|\leqslant c|f(n)|$，对一切 $n\geqslant n_0$ 成立。即 $\lim\limits_{x\to\infty}\dfrac{T(n)}{f(n)}=c\neq 0$ 成立。

如一个算法时间总和 $T(n)$ 与 n^3 成正比，$f(n)=n^3$，则 $T(n)=O(n^3)$，估计算法的时间复杂度，需要与一个数量级函数 $f(n)$ 进行比较。

下面是常用的 O 数量级。

$O(1)$，称为常量级，即算法的时间复杂度是一个常数。

$O(n)$，称为线性级，时间复杂度是数据量 n 的线性函数。

$O(n^2)$，称为平方级，与数据量 n 的二次多项式函数属于同一数量级。

$O(n^3)$，称为立方级，是 n 的三次多项式函数。

$O(\log_2 n)$，称为对数级，是 n 的对数函数。

$O(n\log_2 n)$，是介于线性级和平方级之间的一种数量级。

$O(2^n)$，称为指数级，时间复杂度与数据量 n 的指数函数是一个数量级。

$O(n!)$，称为阶乘级，与数据量 n 的阶乘属于同一数量级。

其中前四种称为多项式数量级，中间两种称为对数数量级，后两种称为指数数量级。它们之间的关系是：$O(1)<O(\log_2 n)<O(n)<O(n\log_2 n)<O(n^2)<O(n^3)<O(2^n)<O(n!)$。

下面是算法性能分析举例。

【例 1-5】 计算执行以下程序段的时间复杂度（为叙述方便，每条语句都前加行号）。

```
①for(i=1; i<n; i++)
②{  y=y+1;
③   for(j=0; j<=2*n; j++)
④       x++;
  }
```

解 这是一个二重循环，各条语句的执行次数分别如下。

语句①n 次（$i=n$ 结束，做了 n 次判断和累加操作）；

语句②$n-1$ 次；

语句③$(n-1)(2n+1)$ 次（比循环体内语句多执行一次）；

语句④$2n(n-1)$ 次。

所以有 $T(n)=n+(n-1)+(n-1)(2n+1)+2n(n-1)=4n^2-2n-1$。

若用 O 数量级估计，则有 $T(n)=O(n^2)$。

【例 1-6】 分析以下程序段的时间复杂度。

```
int i=1;          ①
while(i<=n)
      i=i*2;      ②
```

算法的运行时间由程序中所有语句的频度（即该语句重复执行的次数）之和构成。

分析：显然，语句①的频度是 1，设语句②的频度是 $f(n)$，则有循环次数 $2^{f(n)}\leqslant n$，即 $f(n)\leqslant\log_2 n$，取最大值 $f(n)=\log_2 n$。

所以该程序段的时间复杂度 $T(n)=1+f(n)=1+\log_2 n=O(\log_2 n)$。

基于上述分析，在评价算法的优劣时，总是用某主要语句的执行频度代替时间复杂度。

对不同算法应适当选择其主要语句计算执行频度。如对于查找算法，通常只计算比较语句的执行次数，简称比较次数。对排序操作可计算比较次数，也可计算移动次数，但某些情况下移动数据较少，甚至不执行移动，还是计算比较次数。

2. 算法的空间复杂度

一个算法在计算机存储器上所占用的存储空间，包括存储算法程序代码本身所占用的存储空间、算法的输入输出数据所占用的存储空间和算法在运行过程中临时占用的存储空间这三个方面。在讨论算法的空间复杂度（Space Complexity）时一般不考虑代码部分和所处理的数据本身所占用的存储空间，只考虑一个算法在运行过程中临时占用的存储空间大小，这个存储空间的大小也与问题的处理规模 n 有关，可以认为是问题规模 n 的某个函数 $f(n)$。因此，算法的空间复杂度也可以记为 $S(n)=O(f(n))$，其中 n 为问题的规模。

1.4 算法描述工具简介

本教材使用类C语言或标准C语言定义数据类型、表示算法，以下对C语言的语句成分和语法规则作简单介绍。它们在以后各章中频繁使用，必须熟练记忆并会使用。

1. 符号常量定义

```
#define TRUE  1
#define FALSE  0
#define OK 1
#define ERROR  0
#define OVERFLOW  -1
```

2. 类型别名定义

用typedef为一个数据类型定义新的名字。如：

```
typedef int Bool;
typedef int Status;
```

3. 运算符

算术运算符：+、-、*、/、%（取余）

比较运算符：==、!=、<、>、<=、>=

逻辑运算符：||（或）、&&（与）、!（非）

4. 结构定义

（1）定义结构类型。

定义结构类型的格式为：

```
struct 结构类型名
{结构成员列表};
```

各结构成员变量间用分号分隔。

（2）加typedef定义结构类型。

加typedef定义结构类型的格式为：

```
typedef struct 结构类型名
{结构成员列表
}结构类型名表;
```

（3）定义结构类型的同时定义结构变量。

定义结构类型的同时定义结构变量的格式为：

```
struct 结构类型名
{结构成员列表
}结构变量名表;
```

各结构变量名间用分号分隔。

5. 函数

（1）自定义函数。

自定义函数的一般格式为：