21世纪高等教育计算机规划教材

COMPUTER

数据结构与算法

Data Structure and Algorithm

王曙燕 主编
王春梅 副主编

- 总结比较各类算法的特点和适用范围
- 案例选取强调与后续课程的衔接
- 注重“计算思维”和创新实践能力的培养

人民邮电出版社
POSTS & TELECOM PRESS

21世纪高等教育计算机规划教材

COMPUTER

数据结构与算法

Data Structure and Algorithm

王曙燕　主编
王春梅　副主编

人民邮电出版社
北京

图书在版编目（CIP）数据

数据结构与算法 / 王曙燕主编. -- 北京 : 人民邮电出版社, 2013.9
21世纪高等教育计算机规划教材
ISBN 978-7-115-32420-7

Ⅰ. ①数… Ⅱ. ①王… Ⅲ. ①数据结构－高等学校－教材②算法分析－高等学校－教材 Ⅳ. ①TP311.12

中国版本图书馆CIP数据核字(2013)第167933号

内 容 提 要

数据结构与算法设计是高等院校理工科各专业计算机应用能力提高的重要技术基础。本书将数据结构和算法分析与设计的基础知识相结合，以实际应用为驱动，将数据结构的各种知识融入到实际问题的解决中，对相关算法的核心思想进行深入剖析，并总结比较各类算法的特点和适用范围，重点培养学生使用数据结构知识分析问题和解决问题的能力，为后继课程的学习以及从事软件开发工作打下良好的基础。

本书系统地讲解了数据结构与算法设计的相关知识，全书共有 9 章，论述了数据结构的基本概念、线性表、栈与队列、串、数组和广义表、树、图、查找以及排序等内容。为了让读者能够及时地检查自己的学习效果，把握自己的学习进度，每章后面都附有丰富的习题。

本书既可以作为高等院校各专业“数据结构”课程的教材，也可供准备考研的读者阅读参考，同时也可作为工程技术人员和计算机爱好者的参考资料。

◆ 主　　编　王曙燕
副 主 编　王春梅
责任编辑　李海涛
责任印制　彭志环　焦志炜

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
邮编　100061　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
三河市海波印务有限公司印刷

◆ 开本：787×1092　　1/16
印张：18.5　　2013 年 9 月第 1 版
字数：487 千字　　2013 年 9 月河北第 1 次印刷

定价：42.00 元

读者服务热线：(010)67170985　印装质量热线：(010)67129223
反盗版热线：(010)67171154

前　言

计算机科学技术飞速发展，其应用已经深入到社会各个领域，不仅有效地解决了各种数值计算问题，而且更广泛地解决了大量非数值的数据处理问题，包括文本处理、信息检索、图像处理以及人工智能等诸多领域的问题。

数据结构是一门面向设计，且处于计算机学科核心地位的技术基础和主干必修课，也是算法分析与设计、操作系统、编译技术、计算机图形与图像处理等专业课程的先修课程。国内外许多科技人员对学习、研究数据结构都非常重视。美国 IEEE 和 ACM 的教学计划 CC2005 把“数据结构与算法”列入计算机以及信息技术相关学科专业的本科必修基础课程。

“数据结构与算法”是一门理论和实际紧密结合的重要的计算机类专业基础课程。本书根据学科的最新发展，对所教授课程的教学内容进行必要的筛选、补充、更新和重组，使其既能反映该学科领域最基本最核心的知识，又能反映该学科最新的进展和动态，注重学生“计算思维”能力和创新实践能力的培养，并补充了后续课程和相关领域应用的实例。

计算机科学的重要基石是算法，数据结构又是算法研究的基础。本书将数据结构的知识和算法分析与设计的基础知识相结合，以实际的应用案例为驱动，将各种数据结构与算法的知识融入到实际问题的解决中，对相关算法的核心思想进行深入剖析，并总结比较各类算法的特点和适用范围，重点培养学生利用数据结构知识分析和解决实际问题的能力，为后继课程的学习以及从事计算机软、硬件开发工作打下良好的基础。

本书的参考学时为 64 ~ 80 学时，建议采用理论实践一体化的教学模式，各章的参考学时见下面的学时分配表。

学时分配表

章　节	课 程 内 容	学　时
第 1 章	引论	4
第 2 章	线性表	8
第 3 章	栈和队列	6
第 4 章	串	4～6
第 5 章	多维数组和广义表	6～8
第 6 章	树	10～12
第 7 章	图	10～14
第 8 章	查找	8～12
第 9 章	排序	8～10
课时总计		64～80

本书由王曙燕任主编，王春梅任副主编。各章节编写分工如下：王曙燕编写了第1、3章，王春梅编写了第5、7、8、9章，初建玮编写了第6章，王燕编写了第4章，王晓梅编写了第2章，王曙燕和王春梅对全书进行了细致的修改和统稿。

尽管本书经过了作者的反复修改和推敲，但由于编者水平和经验有限，书中难免有欠妥和错误之处，恳请读者批评指正。

编　者

2013年5月

目　录

第1章 引　论

1946年，世界上第一台电子数字式计算机在美国宾夕法尼亚大学诞生了，这就是电子数值积分计算机（Electronic Numerical Integrator and Calculator，ENIAC）。ENIAC 奠定了电子计算机的发展基础，开辟了一个计算机科学技术的新纪元。在此后短短的几十年间，计算机的发展突飞猛进。伴随着硬件的发展，计算机软件、计算机网络技术也得到了迅速的发展，计算机的应用领域也由最初的数值计算扩展到人类生活的各个领域，现在计算机已成为最基本的信息处理工具。

早期的计算机主要用于数值计算，随着计算机科学的发展，计算机的应用已经远远超出了单纯进行数值计算的范围，数据处理已在计算机应用中占有越来越重要的地位。据统计，目前非数值型信息（文字、字符、图、树、表、视频、音频等）的处理占据了计算机90%以上的机时。这样，如何有效地组织数据，以便设计出更高效、高质量的程序来解决现实中的问题，已成为计算机科学工作者十分关心的事情。

“数据结构”作为一门独立的课程最早是美国的一些大学开设的，1968 年美国唐·欧·克努特教授开创了数据结构的最初体系，他所著的《计算机程序设计技巧》第一卷《基本算法》是第一本较系统地阐述数据的逻辑结构和存储结构及其操作的著作。从20世纪60年代末到70年代初，出现了大型程序，软件也相对独立，结构程序设计成为程序设计方法学的主要内容，人们就越来越重视数据结构，认为程序设计的实质是对确定的问题选择一种好的结构，加上设计一种好的算法。从20世纪70年代中期到80年代初，各种版本的数据结构著作就相继出现。目前在我国，“数据结构”不仅仅是计算机专业的教学计划中的核心课程之一，而且是其他非计算机专业的主要选修课程之一。

著名的计算机科学家沃思（Niklaus Wirth）曾经提出一个著名的公式：

数据结构+算法=程序

该公式阐明了数据结构和算法对于程序设计的重要性。计算机科学的重要基石是算法，数据结构又是算法研究的基础。

数据结构与算法的研究涉及构筑计算机求解问题过程的两个重要方面：刻画实际问题中信息及其关系的数据结构和描述问题解决方案的逻辑抽象算法。

1.1　数据结构的概念

数据结构主要是研究数据（特别是非数值型数据）的组织、存储及运算方法的课程。它是计算机科学中的一门综合性的专业基础课。

首先介绍数据结构的相关术语。

1. 数据（Data）

数据是描述客观事物的数值、字符以及能输入到计算机中且能被处理的各种符号集合。换句话说，数据是对客观事物采用计算机能够识别、存储和处理的形式所进行的描述。数据不仅包括整型、实型、布尔型等数值型数据，还包括字符、声音、图像等一切可以输入到计算机中的符号集合。例如：一个编译程序或文字处理程序的处理对象是字符串。

2. 数据元素（Data Element）

数据元素是**组成数据的基本单位**，是数据集合的个体，在计算机中通常作为一个整体进行考虑和处理。一个数据元素可由一个或多个数据项组成，数据项（Data Item）是数据的不可分割的最小单位。例如：通讯录是数据，每一个人的信息就是一个数据元素，此时的数据元素通常称为记录（Record），而通讯录中的每一项（如姓名、电话号码）为一个数据项。如表 1.1 所示。

表 1.1　　通讯录

序　号	姓　名	性　别	电话号码	单　位	地　址
1	王平	女	13008892088	西安交通大学	西安市友谊东路
2	李明	男	13992900001	西安邮电大学	西安市长安区
3	张鹏飞	男	13200022223	西安邮电大学	西安市长安区
……	……	……	……	……	……

3. 数据对象（Data Object）

数据对象是**性质相同的数据元素的集合**，是数据的一个子集。例如：正整数数据对象是集合 $N=\{0, 1, 2, \cdots\}$，字母字符数据对象是集合 $C=\{'A', 'B', \cdots, 'Z'\}$，本节中的表 1.1 中的通讯录也可看作一个数据对象。

4. 数据结构（Data Structure）

数据结构是指相互之间存在一种或多种特定关系的数据元素集合。通常数据对象中的数据元素不是孤立的，而是彼此之间存在着关系，如表结构（表 1.1 通讯录，元素之间存在线性关系）、树型结构（图 1.1 学校组织结构图，元素之间存在一对多的层次关系）、图结构（图 1.2 施工进度图，元素之间存在多对多的任意关系），我们把数据元素相互之间的关系称为“结构”，即数据的组织形式，所以也可以说数据结构是带有结构的数据元素的集合。

数据结构是一个二元组：Data_Structure=（D,R）

其中 D 是数据元素的有限集，R 是 D 上关系的有限集。

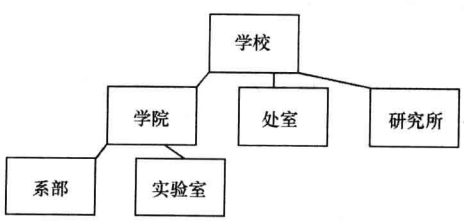

图 1.1　学校组织结构图

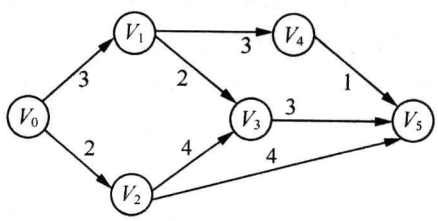

图 1.2　施工进度图

5. 数据类型（Data Type）

数据类型是一组性质相同的值集合以及定义在这个值集合上的一组操作的总称。数据类型中

定义了两个集合，即该类型的取值范围和该类型中可允许使用的一组运算。例如，高级语言中的数据类型就是已经实现的数据结构的实例。数据类型是高级语言中允许的变量种类，如在高级语言中的整型类型，它可能的取值范围是某个区间（区间大小和不同的机器相关），可进行的运算为加、减、乘、除、乘方、取模等（如 C 语言中的+、–、*、/、%）。

从硬件的角度看，它们的实现涉及“字”、“字节”、“位”、“位运算”等等；从用户观点看，并不需要了解整数在计算机内是如何表示、运算细节是如何实现的，用户只需要了解整数运算的外部运算特性，而不必了解机器内部位运算的细节，就可运用高级语言进行程序设计。引入数据类型的目的是实现信息隐蔽，将一切用户不必关心的细节封装在类型中。如两整数求和问题，程序用户只需关注其数学求和的抽象特性，而不关心加法运算涉及的内部位运算实现。

按“值”的不同特性，高级程序语言中的数据类型可分为两大类：一类是非结构的原子类型，原子类型的值是不可分解的，如 C 语言中的基本类型（整型、实型、字符型等）及指针；另一类是结构类型，结构类型的值是由若干成份按某种结构组成的，因此是可以分解的，并且它的成份可以是非结构的，也可以是结构的，例如数组、结构体等。

6. 数据抽象与抽象数据类型

抽象是对一种事物或一个系统的简化描述，它集中注意力于事物或系统的本质方面，而忽略非本质的细节。

计算机界名人 E.Dijkstra 关于如何对付日益增长的软件复杂度问题说：“设计并实现一个大规模的软件的中心问题是怎样减小复杂度，一个途径就是通过抽象。”软件技术发展 50 多年的历史证明了这一点。程序设计语言的发展从机器语言→汇编语言→高级语言→非过程化语言（面向对象语言）就是不断抽象的过程。因为用户关心的只是软件能做什么，而不是为什么能这样做及实现的细节。

前面讨论的数据类型，抽象程度不够高，且对问题来说，不能比较自然地加以表达，因为它只描述了数据的形式，而没有描述他们的功能。

（1）抽象数据类型（Abstract Data Type）

抽象数据类型（简称 ADT）是指一个数学模型以及定义在该模型上的一组操作。或者说是基于一类逻辑关系的数据类型以及定义在这个类型之上的一组操作。“抽象”的意义在于数据类型的数学抽象特性。抽象数据类型的定义取决于客观存在的一组逻辑特性，而与其在计算机内如何表示和实现无关。即不论其内部结构如何变化，只要它的数学特性不变，都不影响其外部使用。在某种意义上，抽象数据类型和数据类型实质上是一个概念。整数类型就是一个简单的抽象数据类型实例，尽管在不同处理器上实现的方法可以不同，但其定义的数学特性是相同的，都可以实现+、–、*、/、%等运算，在用户看来都是相同的。

一个 ADT 定义了一个数据对象、数据对象中各元素间的结构关系，以及一组处理数据的操作。ADT 通常是指由用户定义、用以表示应用问题的数据模型，通常由基本的数据类型组成，并包括一组相关服务操作。

ADT 包括定义和实现两方面，其中定义是独立于实现的。定义仅给出一个 ADT 的逻辑特性，不必考虑如何在计算机中实现。抽象数据类型的特征是使用与实现分离，实现封装和信息隐蔽。就是说，在抽象数据类型设计时，类型的定义与其实现分离。

抽象数据类型的范畴包括各种不同的计算机处理器中已定义并实现的固有数据类型，还包括用户在设计软件系统时自己定义的数据类型。所定义的数据类型的抽象层次越高，含有该抽象数据类型的软件复用程度就越高。

ADT 物理实现作为私有部分封装在其实现模块内，使用者不能看到，也不能直接操作该类型所存储的数据，只有通过界面中的服务来访问这些数据。从实现者的角度来看，把抽象数据类型的物理实现封装起来，有利于编码、测试，也有利于修改。需要改进数据结构，只要界面服务的使用方式不变，只需要改变抽象数据类型的物理实现，所有使用该抽象数据类型的程序不需要改变，提高系统的稳定性。

抽象数据类型是近年来计算机科学中提出的最重要的概念之一，它集中体现了**程序设计中一**些最基本的原则：分解、抽象和信息隐藏。

一个抽象数据类型确定了一个模型，但将模型的实现细节隐藏起来；它定义了一组运算，但将运算的实现过程隐藏起来。我们可以用抽象数据类型来指导问题求解的过程。

（2）ADT 的定义格式

ADT <ADT 名>

{ 数据对象:<数据对象的定义>

结构关系:<结构关系的定义>

基本操作:<基本操作的定义>

}ADT <ADT 名>

其中数据对象和结构关系的定义采用数学符号和自然语言描述，而基本操作的定义格式为：

<操作名称> (参数表)

操作前提: <操作前提描述>

操作结果: <操作结果描述>

例如：一个线性表的抽象数据类型的描述如下。

```
ADT Linear_list
{
```

数据对象：所有 a_i 属于同一数据对象，i=1，2，…，n　　$n \geqslant 0$；

结构关系：所有数据元素 a_i（i=1，2，…，n-1）存在次序关系<a_i，a_{i+1}>，a_1 无前趋，a_n 无后继；

基本操作：设 L 为 Linear_list；

Initial(L)：初始化空线性表；

Length(L)：求线性表表长；

Get(L,i)：取线性表的第 i 个元素；

Insert(L,i,b)：在线性表的第 i 个位置插入元素 b；

Delete(L,i)：删除线性表的第 i 个元素；

} ADT Linear_list

上述 ADT 很明显是抽象的。数据元素所属的数据对象没有局限于一个具体的整型、实型或其他类型。所具有的操作也是抽象的数学特性，并没有具体到何种计算机语言指令与程序编码。

（3）抽象数据类型的实现

实现抽象数据类型需要借助于高级语言，对于 ADT 的具体实现依赖于所选择的高级语言的功能。从程序设计的历史发展来看有传统的面向过程的程序设计、“包”、“模型”的设计方法、面向对象的程序设计等几种不同的实现方法。

下面分三种情况予以介绍。

第一种情况：传统的面向过程的程序设计。也就是我们现在常用的方法，根据逻辑结构选定

合适的存储结构，根据所要求操作设计出相应的子程序或子函数。

在标准 PASCAL、C 等面向过程的语言中，用户可以自己定义数据类型。由此可以借助过程和函数、利用固有的数据类型来表示和实现抽象数据类型。由于标准 PASCAL 语言的程序结构框架是由严格规定次序的“段”（包括程序首部、标号说明、常量定义、类型定义、变量说明、过程或函数说明和语句部分）组成，因此，所有使用已定义的抽象数据类型的外部用户必须将已定义的抽象数据类型说明和过程说明嵌入到自己程序的适当位置。

例如用 C 语言实现 ADT 时，数据类型可以用基本数据类型、结构体类型，也可用 typedef 自定义类型，以增强抽象性和可读性；基本操作可以用 C 语言的子函数实现。

第二种情况：“包”、“模型”的设计方法。Ada 语言提供了“包”（package），Module-2 语言提供了“模块”（module）结构，TURBO PASCAL 语言提供了“单元”（UNIT）结构，每个模块可含有一个或多个抽象数据类型，它不仅可以单独编译，而且为外部使用抽象数据类型提供了方便，用这类结构实现 ADT 比起第一种方法有一定的进步。

第三种情况：面向对象的程序设计（Object Oriented Programming，简称 OOP）。在面向对象的程序设计语言中，借助对象描述抽象数据类型，存储结构的说明和操作函数的说明被封装在一个整体结构中，这个整体结构称之为“类”（class），属于某个“类”的具体变量称之为“对象”（object）。OOP 与 ADT 的实现更加接近和一致。从前面对数据类型的讨论中看到，在面向对象的程序设计语言中，“类型”的概念与“操作”密切相关，同一种数据结构和不同的操作组将构成不同的数据类型，结构说明和过程说明被统一在一个整体对象之中。其中，数据结构的定义为对象的属性域，过程或函数定义在对象中称之为方法（method），是对象的性能描述。

面向对象的开发方法首先着眼于应用问题所涉及的对象，包括对象、对象属性和要求的操作，借助对象描述抽象数据类型，存储结构和操作函数的说明被封装在一个整体结构（类 class）中，软件易于修改。而传统的结构化的开发方法是面向过程的开发方法，首先着眼于系统要实现的功能。

1.2　数据结构的内容

“数据结构”在计算机科学中是一门综合性的专业基础课。数据结构的研究不仅涉及计算机硬件（特别是编码理论、存储装置和存取方法等）的研究范围，而且和计算机软件的研究有着更密切的关系，无论是编译程序还是操作系统，都涉及数据元素在存储器中的分配问题。在研究信息检索时也必须考虑如何组织数据，以便查找和存取数据元素更为方便。因此，可以认为数据结构是介于数学、计算机硬件和计算机软件三者之间的一门核心课程。在计算机科学中，数据结构不仅是一般程序设计（特别是非数值计算的程序设计）的基础，而且是设计和实现编译程序、操作系统、数据库系统及其他系统程序和大型应用程序的重要基础。它包括三个方面的内容：数据的逻辑结构、数据的存储结构和数据的运算。

1.2.1　数据的逻辑结构

数据的逻辑结构是指数据元素之间逻辑关系的描述。

可以用一个二元组，形式化的描述数据的逻辑结构：

Data_Structure=（D,R）其中 D 是数据元素的有限集，R 是 D 上关系的有限集。

这里的关系描述的是数据元素之间的逻辑关系，因此又称为数据的逻辑结构。一个数据元素通常称为一个结点，描绘时通常用一个圆圈表示。根据数据元素之间关系的不同特性，通常有四种基本结构：（1）集合结构；（2）线性结构；（3）树形结构；（4）图状结构。如图 1.3 所示。

（1）集合结构：结构中的数据元素之间除了同属于一个集合的关系外，无任何其他关系。

（2）线性结构：结构中的数据元素之间存在着一对一的线性关系。

（3）树形结构：结构中的数据元素之间存在着一对多的层次关系。

（4）图状结构：结构中的数据元素之间存在着多对多的任意关系。

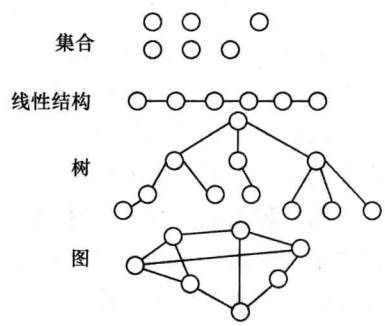

图 1.3　四类基本结构关系图

根据数据元素之间关系的不同特性，数据结构又可分为两大类：线性结构和非线性结构。按照这种划分原则，本书要介绍的数据结构可划分为线性结构：线性表、栈、队列、字符串、数组和广义表；非线性结构：树和图；另外，还将介绍基本的数据处理技术查找和排序方法。

1.2.2　数据的存储结构

数据的逻辑结构是从逻辑上来描述数据元素之间的关系的，是独立于计算机的。然而讨论数据结构的目的是为了在计算机中实现对它的操作，因此还需要研究数据元素之间的关系如何在计算机中表示，这就是数据的存储结构。

大家知道，计算机的存储器是由很多存储单元组成的，每个存储单元有唯一的地址。数据存储结构要讨论的就是数据结构在计算机存储器上的存储映像方法。数据结构在计算机中的表示（又称映像）称为数据的物理结构或者存储结构。它包括数据元素的表示和关系的表示。

数据元素在计算机中用若干个二进制“位串”表示。

数据元素之间的关系在计算机中有两种表示方法：顺序映像和非顺序映像。并由此得到两种不同的存储结构：顺序存储和链式存储。顺序存储的特点是借助元素在存储器中的相对位置来表示数据元素之间的逻辑关系；链式存储的特点是借助指针表示数据元素之间的逻辑关系。

逻辑结构与存储结构的关系为：存储结构是逻辑关系的映象与元素本身的映象，是数据结构的实现；逻辑结构是数据结构的抽象。

任何一个算法的设计（决定有什么样的操作或运算）取决于选定的数据（逻辑）结构，而算法的实现依赖于采用的存储结构。

综上所述，数据结构的内容可归纳为三个部分，逻辑结构、存储结构和运算集合。按某种逻辑关系组织起来的一批数据，按一定的映象方式把它存放在计算机存储器中，并在这些数据上定义了一个运算的集合，就叫做数据结构。

1.3　算　　法

开发程序的目的，就是要解决实际问题。然而，面对各种复杂的实际问题，如何编制程序，往往令初学者感到茫然。程序设计语言只是一个工具，只懂得语言的规则并不能保证编制出高质量的程序。程序设计的关键是设计算法，算法与程序设计和数据结构密切相关。简单地讲，算法

是解决问题的策略、规则和方法。算法的具体描述形式很多，但计算机程序是对算法的一种精确描述，而且可在计算机上运行。

1.3.1　算法的概念

算法就是解决问题的一系列操作步骤的集合。比如，厨师做菜时，都经过一系列的步骤：洗菜、切菜、配菜、炒菜和装盘。用计算机解题的步骤就叫算法，编程人员必须告诉计算机先做什么，再做什么，这可以通过高级语言的语句来实现。通过这些语句，一方面体现了算法的思想，另一方面指示计算机按算法的思想去工作，从而解决实际问题。程序就是由一系列的语句组成的。

著名的计算机科学家沃思（Niklaus Wirth）曾经提出一个著名的公式：

数据结构+算法=程序

数据结构是指对数据（操作对象）的描述，即数据的类型和组织形式，算法则是对操作步骤的描述。也就是说，数据描述和操作描述是程序设计的两项主要内容。数据描述的主要内容是基本数据类型的组织和定义，数据操作则是由语句来实现的。

算法具有下列特性。

1. 有穷性

对于任意一组合法输入值，在执行有穷步骤之后一定能结束，即算法中的每个步骤都能在有限时间内完成。

2. 确定性

算法的每一步必须是确切定义的，使算法的执行者或阅读者都能明确其含义及如何执行，并且在任何条件下，算法都只有一条执行路径。

3. 可行性

算法应该是可行的，算法中的所有操作都必须足够基本，都可以通过已经实现的基本操作运算有限次实现。

4. 有输入

一个算法应有零个或多个输入，它们是算法所需的初始量或被加工的对象的表示。有些输入量需要在算法执行过程中输入，而有的算法表面上可以没有输入，实际上已被嵌入算法之中。

5. 有输出

一个算法应有一个或多个输出，它是一组与“输入”有确定关系的量值，是算法进行信息加工后得到的结果，这种确定关系即为算法的功能。

以上这些特性是一个正确的算法应具备的特性，在设计算法时应该注意。

程序是算法用某种程序设计语言的具体实现，程序可以不满足算法的性质 1。例如操作系统，它是一个在无限循环中执行的程序，因而不是一个算法。然而，可以把操作系统的各种任务看成是一些单独的问题，每一个问题由操作系统中的一个子程序通过特定的算法来实现，该子程序得到输出结果后便终止。

1.3.2　算法的评价标准

什么是“好”的算法，通常从下面几个方面衡量算法的优劣。

1. 正确性

正确性指算法能满足具体问题的要求，即对任何合法的输入，算法都会得出正确的结果。

算法的正确性一般可分为四个层次：程序不含语法错误；程序对于几组输入数据能够得出满足规格说明要求的结果；程序对于精心选择的典型、苛刻而带有刁难性的几组输入数据能够得出

满足规格说明要求的结果；程序对于一切合法的输入数据都能产生满足规格说明要求的结果。

2. 可读性

可读性指算法被理解的难易程度。算法主要是为了人的阅读与交流，其次才是为计算机执行，因此算法应该更易于人的理解。另一方面，晦涩难读的程序易于隐藏较多错误而难以调试。

3. 健壮性（鲁棒性）

健壮性又称鲁棒性，即对非法输入的抵抗能力。当输入的数据非法时，算法应当恰当地做出反应或进行相应处理，而不是产生奇怪的输出结果。并且，处理出错的方法不应是中断程序的执行，而应是返回一个表示错误或错误性质的值，以便在更高的抽象层次上进行处理。

4. 高效率与低存储量需求

通常，效率指的是算法执行时间；存储量指的是算法执行过程中所需的最大存储空间，两者都与问题的规模有关。尽管计算机的运行速度提高很快，但这种提高无法满足问题规模加大带来的速度要求。所以追求高速算法仍然是必要的。相比起来，人们会更多地关注算法的效率，但这并不因为计算机的存储空间是海量的，而是由人们面临的问题的本质决定的。二者往往是一对矛盾，常常可以用空间换时间，也可以用时间换空间。

1.3.3 算法的描述

描述算法的工具可用自然语言、框图或高级程序设计语言。若用高级程序设计语言描述有严格、准确的特点，但缺点是语言细节过多，所以我们一般采用**类语言**描述算法。

类语言是接近于高级语言而又不是严格的高级语言，具有高级语言的一般语句格式，撇掉语言中的细节，以便把注意力主要集中在算法处理步骤本身的描述上。伪代码用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。它采用某一程序设计语言的基本语法，如操作指令可以结合自然语言来设计，而且它不用符号，书写方便，没有固定的语法和格式，具有很大的随意性，便于向程序过渡。

本书采用类 C 语言描述算法，类 C 语言是介于伪代码和 C 语言的一种描述工具，其语法基本上全部取自标准 C 语言，因而易于转化成 C 或 C++程序，但它是简化的、不严格的，不可以真正在计算机上运行，主要反映在以下几点：

可以采用伪码语言取代某些不必确切描述的语句或语句串；

省略函数体内的简单变量的说明；

输入/输出函数只说明输出什么，不用考虑输入/输出的格式；

强化赋值语句的功能。

下面是类 C 语言的简要说明。

1. 预定义常量和类型

```
#define  TRUE  1
#define  FALSE  0
#define  MAXSIZE 100
#define  OK  1
#define  ERROR  0
```

2. 函数的表示形式

```
[数据类型]  函数名（[形式参数及说明]）
{   内部数据说明;
    执行语句组;
```

```
}  /*函数名*/
```

3. 赋值语句

（1）简单赋值

①〈变量名〉=〈表达式〉

②〈变量〉++

③〈变量〉--

（2）串联赋值

〈变量 1〉=〈变量 2〉=〈变量 3〉=…=〈变量 k〉=〈表达式〉

（3）成组赋值

(<变量>,<变量 2>,<变量 3>,…<变量 k>)=(<表达式 1>,<表达式 2>,<表达式 3>,…<表达式 k>)

〈数组名 1〉[下标 1][下标 2]=〈数组名 2〉[下标 1][下标 2]

（4）条件赋值

〈变量名〉=〈条件表达式〉？〈表达式 1〉:〈表达式 2〉

4. 条件选择语句

```
if  (<表达式>) 语句;
if  (<表达式>) 语句 1;
else  语句 2;
```

情况语句

```
switch  (<表达式>)
 { case   判断值 1:
          语句组 1;
          break;
   case   判断值 2:
          语句组 2;
          break;
 ……
   case   判断值 n:
        语句组 n;
        break;
  [default:
        语句组;
        break;]
  }
```

5. 循环语句

for 语句

```
for (<表达式 1>; <表达式 2>; <表达式 3>)
{
    循环体语句;
 }
```

while 语句

```
while (<条件表达式>)
 {
    循环体语句;
 }
```