KH 北京科海
25年·科技图书出版专家

21世纪高职高专计算机技能与应用系列规划教材

数据结构

时 磊 梁晓丽 主 编
原锦明 李 佳 副主编

DATA STRUCTURE

中国人民大学出版社
北京科海电子出版社
www.khp.com.cn

21世纪高职高专计算机技能与应用系列规划教材

数据结构

时　磊　梁晓丽　主　编
原锦明　李　佳　副主编

中国人民大学出版社
·北京·

北京科海电子出版社
www.khp.com.cn

图书在版编目(CIP)数据

数据结构/时磊，梁晓丽主编.
北京：中国人民大学出版社，2009
（21 世纪高职高专计算机技能与应用系列规划教材）
ISBN 978-7-300-11305-0

Ⅰ. 数…
Ⅱ. ①时… ②梁…
Ⅲ. 数据结构—高等学校：技术学校—教材
Ⅳ. TP311.12

中国版本图书馆 CIP 数据核字（2009）第 179938 号

21 世纪高职高专计算机技能与应用系列规划教材
数据结构
时　磊　梁晓丽　主编

出版发行	中国人民大学出版社　北京科海电子出版社		
社　　址	北京中关村大街 31 号	**邮政编码**	100080
	北京市海淀区上地七街国际创业园 2 号楼 14 层	**邮政编码**	100085
电　　话	（010）82896594　62630320		
网　　址	http: //www.crup.com.cn		
	http: //www.khp.com.cn（科海图书服务网站）		
经　　销	新华书店		
印　　刷	北京市科普瑞印刷有限责任公司		
规　　格	185 mm×260 mm　16 开本	**版　　次**	2010 年 1 月第 1 版
印　　张	18.5	**印　　次**	2010 年 1 月第 1 次印刷
字　　数	450 000	**定　　价**	29.00 元

内容提要

本书根据高职高专院校相关专业的数据结构课程的特点和教学大纲的规定，详细讲解了数据结构的基本概念、基本结构和算法等重要内容。全书共分为9章，主要内容包括数据结构概述，线性表，栈和队列，串、数组和广义表，树，图，查找，内部排序，以及综合实例等。每章后配有丰富的练习题和上机实验，以利于读者理解知识内容和适应考试。

本书通俗易懂、重点突出、实例丰富，具有概念表达严谨、知识结构逻辑性强等特点，既便于老师教学又便于学生自学。

本书可作为高职高专院校学生学习数据结构的教材，也可作为计算机培训班的教材及自学者的参考书。

前言

《数据结构》是计算机和信息类相关专业的重点课程。从理论上讲，通过本课程的学习可以使学生掌握对不同数据结构的组织方法和对具体数据结构所实施的若干算法，并能分析算法的优劣，进一步进行高效率的计算机程序开发。根据笔者多年的教学经验，学生们对《数据结构》课程感觉最深刻的是，通过对本课程系统地学习能够提高程序设计的能力。笔者认为，只学会了某种高级程序设计语言（如 C/C++语言），而未学会（不是学过）数据结构，必定对学生将来在计算机技术等相关领域的发展带来极大的阻碍。

对于高职高专人才的培养更应该注重能力的培养，而不是简单地掌握理论。因此，本书在编写的过程中给出了大量实例算法，并在上机实验中给出了完整的参考程序代码，以及完整的注释，所有代码均在 Visual C++ 6.0 环境中运行通过，具有很好的实用价值，也给学习者带来了方便。希望学生能够理解这些代码，开阔自己的思路，提高自己的程序设计水平。

在《数据结构》课程的学习中，教师需要在课堂上对大量的算法进行讲解，而学生应该在此基础上大量阅读并理解有关数据结构的经典算法。因此，本书中对所有的算法都做了详细的注释。而对于一些难度比较大的算法，只要求学生理解其思想，对程序的实现不做具体要求。

本书共 9 章：第 1 章是数据结构概述；第 2 章～第 4 章介绍了线性结构中的线性表、栈、队列、串、数组及广义表的基本定义、基本算法、基本应用等；第 5 章和第 6 章介绍了非线性结构中的树、二叉树和图；第 7 章和第 8 章介绍了应用较为广泛的查找和排序的基本算法；第 9 章详尽地讲解了学生成绩管理系统的开发过程。本书在各章内容的安排上不求大而全，力求少而精，讲解透彻、重点突出。本书在编写时，尽量做到内容通俗易懂、由浅入深、图文并茂、便于理解。

由于编者水平有限，疏漏之处在所难免，欢迎广大读者发送电子邮件到 shilei791207@163.com 进行批评指正或提出宝贵意见。

编者

2009 年 9 月

目　录

第1章

数据结构概述

本章学习重点：

- 熟悉各名词、术语的含义，掌握基本概念，特别是数据的逻辑结构和存储结构之间的关系。分清哪些是逻辑结构的性质，哪些是存储结构的性质
- 理解算法五个要素的确切含义
- 掌握计算语句频度和估算算法时间复杂度的方法

在20世纪40年代，世界上诞生了第一台计算机。初期，人们把计算机主要应用在科学和工程计算等领域中的纯数值计算，比如给定x的值，求解函数 $f(x)=ax^2+b$ 的值。而如今，人们对计算机的应用远远超出了数值计算范围，计算机技术已经渗透人类生活的一切领域。随着计算机应用领域的扩大和软、硬件的发展，计算机处理的对象也从纯数值发展到了非数值性信息。那么什么是非数值性信息呢，下面来看一个例子。

【例】 在图1.1中共有6个建筑物，建筑物之间连线上的数字表示建筑物之间的距离。请问，邮递员送信，从医院出发，到达工厂的最短距离是多少？

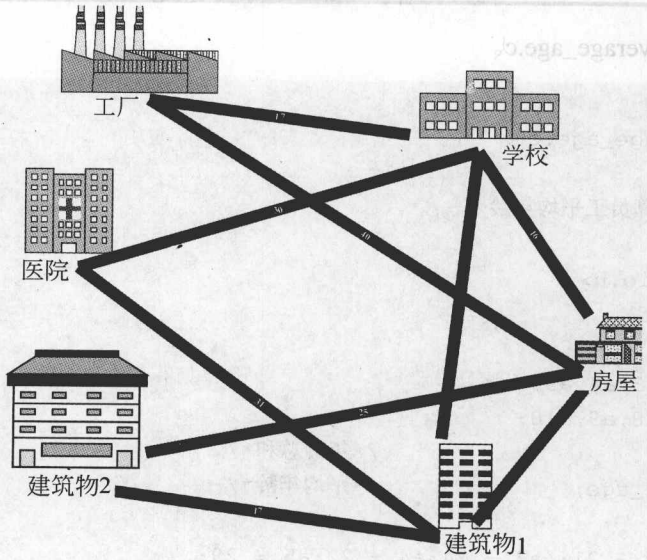


图 1.1

很显然，这个问题无法用某个具体公式计算出，必须根据实际的数据通过某种算法才能得到所要的结果。

《数据结构》是计算机专业的专业基础课程之一。用计算机解决任何实际问题都离不开数据表示和数据处理，而数据表示和处理的核心问题之一是数据结构及其实现——这正是数据结构课程的基本内容。因此，要想更好地运用计算机来解决实际问题，仅掌握几种计算机程序设计语言是难以应付众多复杂的课题的。学好《数据结构》这门课程，对于学习计算机专业的其他课程，如操作系统、编译原理、数据库管理系统、软件工程等都是十分有益的。

1.1 数据结构基本概念

初学者刚开始可能会觉得数据结构是一门比较深奥的学科，实际上，从大家接触程序设计开始，就已经在使用数据结构了。下面通过一个非常简单的C语言实例来说明数据结构和程序设计之间的关系。

【例】 表1.1是一个公司10名员工的年龄列表，请设计一个小程序，计算这10名员工的平均年龄。

表1.1

姓　名	年　龄	姓　名	年　龄
刘　磊	28	王　华	23
赵　刚	21	王　敏	23
王　利	24	张　才	24
刘　莉	22	马　立	27
刘　年	29	王　刚	30

【程序实例】 average_age.c。

```
/*
 * 文件名:average_age.c
 * 作　者:李明
 * 程序功能：计算员工平均年龄
 */
#include <stdio.h>
void main(void)
{
   int a1,a2,a3,a4,a5;
   int a6,a7,a8,a9,a10;
   int sum;                          /*年龄总和*/
   int average_age;                  /*平均年龄*/

   a1 = 28; a2 = 21; a3 = 24;a4 = 22;a5 = 29;
   a6 = 23; a7 = 23;a8 = 24;a9 = 27;a10 = 30;
```

```
    sum = a1 + a2 + a3 + a4 + a5 + a6 + a7 + a8 + a9 + a10;  /*计算年龄总和*/
    average_age = sum / 10;                                  /*计算平均年龄*/
    printf("员工平均年龄是: %d\n",average_age);              /*输出平均年龄*/
}
```

这是一个非常简单的程序，但是程序所使用的实现方法的扩展性很差，如果员工人数变成了几百人，甚至上千人，那么整个程序的变动就非常大。其实，大家很容易想到，如果改用数组来保存员工年龄，会是一个很好的方法。因此，程序改成如下所示。

【程序实例】 average_age_modify.c。

```
/*
* 文件名:average_age_modify.c
* 作  者:李明
* 程序功能: 计算员工平均年龄
*/
#include <stdio.h>
void main()
{
int a[10] = {28, 21, 24, 22, 29, 23, 23, 24, 27, 30};
int sum = 0;                    /*年龄总和*/
int average_age;                /*平均年龄*/
int i;
for(i = 0; i < 10; i++)                              /*计算年龄总和*/
{
sum + = a[i];
}
average_age = sum / 10;                              /*计算平均年龄*/
printf("员工平均年龄是: %d\n",average_age);          /*输出平均年龄*/
}
```

相信大家很容易看出，average_age_modify.c 中所使用的方法好。那么为什么好呢？是直觉，还是经验？实际上，正是因为 average_age_modify.c 中采用了一种数据结构——数组，使得程序设计既简便、又高效。因此，我们学习数据结构的目的就是在进行程序设计时，能够找到一种好的数据结构，并采用某种方法或策略（通常称算法）来有效地解决存在的问题。归纳起来，主要实现以下的目标。

- 程序运行速度快。
- 数据占用内存空间少。
- 能够较快地访问数据。

在系统地学习数据结构知识之前，先对一些基本概念和术语做介绍。

1. 数据（Data）

数据是信息的载体，它能够被计算机识别、存储和加工处理。数据可以是数值数据，也可以是非数值数据。数值数据是一些整数、实数或复数，主要用于工程计算、科学计算等；非数值数据包括字符、文字、图形、图像、语音等。

2. 数据元素（Data Element）

数据元素是数据的基本单位。在不同的条件下，数据元素又可称为元素、结点、顶点、记录等。例如，学生信息检索系统中学生信息表中的一个记录等。数据元素可以是简单数据类型，如 int、char、float 等；也可以是复杂数据类型，如 struct 等。如果数据元素属于复杂类型，数据元素可由若干个数据项（Data Item）组成。例如，学籍管理系统中学生信息表的每一个数据元素就是一个学生记录，它包括学生的学号、姓名、性别、籍贯、出生年月、成绩等数据项。这些数据项可以分为两种：一种叫做初等项，如学生的性别、籍贯等，这些数据项是在数据处理时不能再分割的最小单位；另一种叫做组合项，如学生的成绩，它可以再划分为数学、物理、化学等更小的项。通常，在解决实际应用问题时，把每个学生记录当做一个数据元素进行访问和处理。

3. 数据对象（Data Object）

数据对象是具有相同性质的数据元素的集合，是数据的一个子集。在某个具体问题中，数据元素都具有相同的性质（元素值不一定相等），属于同一数据对象，数据元素是数据对象的一个实例。比如，一组课程信息就是一个数据对象。

4. 数据结构（Data Structure）

数据结构是指互相之间存在着一种或多种关系的数据元素的集合。在任何问题中，数据元素之间都不是孤立的，在它们之间都存在着这样或那样的关系，这种数据元素之间的关系称为结构。数据结构分为逻辑结构和存储结构。

（1）逻辑结构。

数据元素之间的逻辑关系被称为逻辑结构。数据的逻辑结构可以看做是从具体问题抽象出来的数学模型，它与数据的存储无关。逻辑上，数据结构通常可以采用一个二元组来表示：

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。

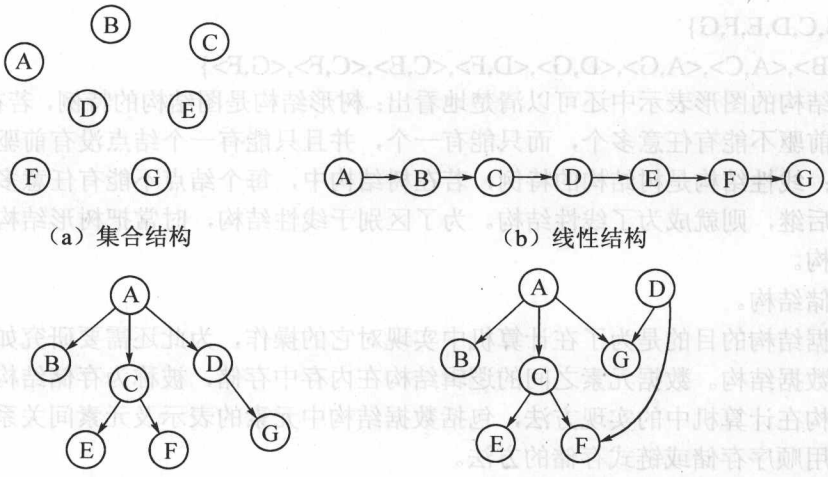
逻辑结构可以区分为下列 4 类基本的结构，如图 1.2 所示。

- 集合结构。在集合结构中，数据元素间的关系是“属于同一个集合”。集合是元素关系极为松散的一种结构，如图 1.2（a）所示。由于计算机无法直接处理集合。通常情况下，需要将集合转化成其他类型的结构后进行处理。
- 线性结构。该结构的数据元素之间存在着一对一的关系，这种关系被称为“前驱-后继”关系。除了起始点数据没有前驱，终点元素没有后继外，其他任何数据元素都有唯一的前驱和唯一的后继。

如图 1.2（b）所示，它有一个头元素 A 和一个尾元素 G，其余为中间元素；每个中间元素既有前驱元素，又有后继元素。如 B 的前驱元素为 A，后继元素为 C；C 的前驱元素为 B，后继元素为 D；头元素 A 没有前驱元素，只有后继元素 B；尾元素 G 只有前驱元素 F，没有后继元素。线性结构中的元素集合 K 和二元关系 R 分别为：

K={A,B,C,D,E,F,G}

R={<A,B>,<B,C>,<C,D>,<D,E>,<E,F>,<F,G>}



（a）集合结构　（b）线性结构

（c）树形结构　（d）图结构

图 1.2

因为 A 和 B 构成一对前驱和后继关系，对应图 1.2（b）中的一条有向边，它的起点为 A，终点为 B，它就构成了 R 中的一个有序对。同理，B 和 C、C 和 D 等都是 R 中的有序对。由于该线性结构包含有 7 个元素，所以二元关系 *R* 中共含有 6 个有序对。

● 树形结构。该结构的数据元素之间存在着一对多的关系。

树形结构的图形表示像倒着画的一棵树，如图 1.2（c）所示，树中有一个树根元素 A，它有 3 个后继元素，又称为 A 的孩子结点 B、C 和 D，C 结点有两个孩子结点 E 和 F，D 结点有一个孩子结点 G，由于 B、E、F、G 没有孩子结点，所以称它们为叶子结点，而 A、C、D 被称为树枝结点或分支结点，同时 A 又是唯一的树根结点。

在树形结构中，树根结点只有后继结点，而没有前驱结点，如 A 为树根结点，它没有前驱结点，或者说其前驱结点为空，它的后继结点为 B、C 和 D。除树根结点外，每个结点都有唯一的前驱结点，又称为是父结点或双亲结点，如 C 的前驱结点为 A，G 的前驱结点为 D，每个结点的前驱结点即双亲结点，从图形中都能够很容易得到。

树形结构中的元素集合 *K* 和二元关系 *R* 分别为：

K={A,B,C,D,E,F,G}

R={<A,B>,<A,C>,<A,D>,<C,E>,<C,F>,<D,G>}

因为 A 和 B 构成一对前驱和后继关系，所以它是 *R* 中的一个有序对，同理，A 和 C、A 和 D 等都是 *R* 中的有序对。

● 图结构。该结构的数据元素之间存在着多对多的关系，任何两个元素都可以有关系。

在图结构中，每个结点或称顶点都可以有任意多个前驱结点和任意多个后继结点。

如图 1.2（d）所示，顶点 A 没有前驱结点，或者说它有 0 个前驱结点，A 有 3 个后继结点 B、C 和 G；G 有 2 个前驱结点 A 和 D，有一个后继结点 F；E 有一个前驱结点 C 和 0 个后继结点，或者说，E 没有后继。

图结构中的元素集合 *K* 和二元关系 *R* 分别为：

K={A,B,C,D,E,F,G}

R={<A,B>,<A,C>,<A,G>,<D,G>,<D,F>,<C,E>,<C,F>,<G,F>}

从数据结构的图形表示中还可以清楚地看出：树形结构是图结构的特例，若在图结构中，每个结点的前驱不能有任意多个，而只能有一个，并且只能有一个结点没有前驱，则就成为了树形结构；线性结构是树结构的特例，若在树结构中，每个结点不能有任意多个后继，而只能有一个后继，则就成为了线性结构。为了区别于线性结构，时常把树形结构和图结构称为非线性结构。

（2）存储结构。

研究数据结构的目的是为了在计算机中实现对它的操作，为此还需要研究如何在计算机中表示一个数据结构。数据元素之间的逻辑结构在内存中存储，被称为存储结构。它所研究的是数据结构在计算机中的实现方法，包括数据结构中元素的表示及元素间关系的表示。存储结构可采用顺序存储或链式存储的方法。

- 顺序存储方法是把逻辑上相邻的元素存储在物理位置相邻的存储单元中，由此得到的存储表示称为顺序存储结构。顺序存储结构是一种最基本的存储表示方法，通常借助于程序设计语言中的数组来实现。
- 链式存储方法对逻辑上相邻的元素不要求其物理位置相邻，元素间的逻辑关系通过附设的指针字段来表示，由此得到的存储表示称为链式存储结构，链式存储结构通常借助于程序设计语言中的指针类型来实现。

【综合实例】

表 1.2 为某个公司的简表，该表中包含有 10 条职工记录，每条记录都由 6 个数据项组成，由于每条记录的职工号各不相同，所以可把每条记录的职工号作为该记录的关键码，并在下面的讨论中，用记录的关键码来代表整个记录。

表 1.2

职工号	姓名	性别	出生日期	职务	部门
01	万明华	男	1962-03-20	经理	
02	赵　宁	男	1968-06-14	主管	销售部
03	张　利	女	1964-12-07	主管	财务部
04	赵书芳	女	1972-08-05	主任	办公室
05	刘永年	男	1959-08-15	科员	销售部
06	王明理	女	1975-04-01	科员	销售部
07	王　敏	女	1972-06-28	科员	财务部
08	张　才	男	1967-03-17	科员	财务部
09	马立仁	男	1975-10-12	科员	财务部
10	邢怀常	男	1976-07-05	科员	办公室

对于上面所述的一张表，假定不考虑该表中记录之间的任何次序，则该表中的数据就是一个集合结构，对应的记录集合以及二元关系为：

K={01,02,03,04,05,06,07,08,09,10}

R={ }

若考虑到该表中的职工记录是按照职工号从小到大有序排列的这个特点，则这个职工表

又是一个线性结构。其中，表头元素为 01 号职工，接着为 02 号职工，依次类推，表尾元素为 10 号职工。该线性结构包含的记录集合和二元关系为：

K={01,02,03,04,05,06,07,08,09,10}

R={<01,02>,<02,03>,<03,04>,<04,05>,<05,06>,<06,07>,<07,08>,<08,09>,<09,10>}

有时需要按职工的出生日期进行数据结构的定义和处理，则可以把表中的所有职工看做是按照出生日期从小到大有序的，由此对应的数据结构也是一种线性结构。其中，05 号职工的出生日期最早，即年龄最大，为表头元素，01 号职工年龄次之，为这种线性结构中的第 2 个元素，依次类推，10 号职工的出生日期最晚，对应的图形表示如图 1.3 所示。

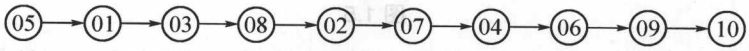


图 1.3

从图形表示中能够清楚地看出每个职工出生日期的先后排列次序。

在上面表 1.2 所示的职工表中，还存在职工人员之间领导与被领导的关系，其中 01 号职工为经理，直接领导 02、03 和 04 号职工，他们分别是相应部门的主管，02 号职工直接领导 05 和 06 号职工，03 号职工直接领导 07、08 和 09 号职工，04 号职工直接领导 10 号职工。由此可知，该职工表是一种树形结构，包含的职工集合和二元关系分别为：

K={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>}

对应的图形表示如图 1.4 所示。

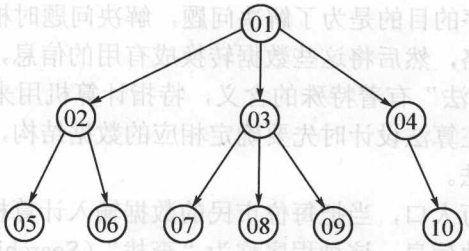


图 1.4

若要反映职工之间的好友关系，假定 01 和 02 号职工是好友，01 和 04 号是好友，02 和 03、02 和 06、02 和 07、03 和 07、04 和 06、05 和 07 之间是好友，则反映这种好友关系的数据结构是图结构，二元组中的元素集合和有序对集合分别为：

K={01,02,03,04,05,06,07,08,09,10}

R={<01,02>,<02,01>,<01,04>,<04,01>,<02,03>,<03,02>,<02,06>,<06,02>,
<02,07>,<07,02>,<03,07>,<07,03>,<04,06>,<06,04>,<05,07>,<07,05>}

对应的图形表示如图 1.5（a）所示，其中 08、09 和 10 号职工无好友，在图形中为孤立顶点，省略未画。图形中每对顶点之间的两条相反的有向边，表示这两个顶点职工是一对好朋友，为了简化起见，两条相反的有向边可以用一条无向边来代替，隐含着该无向边是双向的，即连接的两个顶点互为前驱和后继，则对应的无向图表示如图 1.5（b）所示。

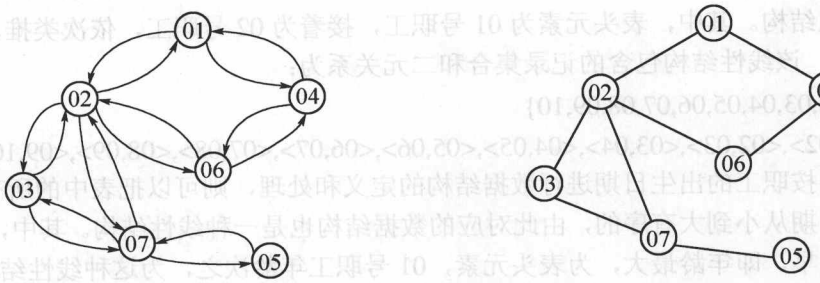


图 1.5

从上面图 1.5（a）中可以看出，R 是 K 上的对称关系，即若存在<x,y>这个序偶，则必然存在<y,x>这个序偶与之对应。为了简化起见，在二元组表示中把<x,y>和<y,x>这两个对称序偶用一个无序对(x,y)或(y,x)来代替，这样 R 关系可改写为：

R={<01,02>,<01,04>,<02,03>,<02,06>,<02,07>,<03,07>,<04,06>,<05,07>}

可见 R 成为了一个无序对的集合，其中的每个元素为用圆括号括起来的一个无序对，对应图形中的一条无向边，由无向边构成的图形称为无向图，反之为有向图。

1.2 算　　法

1.2.1 算法定义

人们用计算机编写程序的目的是为了解决问题，解决问题时根据要处理的数据的类别，选择恰当的数据结构和策略，然后将这些数据转换成有用的信息，以便解决问题。

在计算机科学中，“算法”有着特殊的含义，特指计算机用来解决某一问题的方法。算法与数据结构紧密相关，在算法设计时先要确定相应的数据结构，而在讨论某一种数据结构时也必然会涉及相应的算法。

【例】 某城市有九百万人口，当把每位市民的数据输入计算机后，接着要设计程序，使用姓名查找身份证号或其他信息，这种程序称为“查找”（Searching）。一个好的查找算法也许只花费几秒或几分钟，但差的查找算法可能会花几十分钟、几小时，甚至数天。

算法（Algorithm）是对特定问题求解步骤的一种描述，它是指令的有限序列。其中每一条指令表示一个或多个操作。所有算法应该具有下面 5 个特性。

- 输入。一个算法具有零个或多个输入，这些输入取自特定的数据对象集合。
- 输出。一个算法具有一个或多个输出，这些输出同输入之间存在某种特定的关系。
- 有穷性。一个算法必须在有穷步之后结束，即必须在有限时间内完成。
- 确定性。算法的每一步必须有确切的定义，无二义性。算法的执行对应的相同的输入仅有唯一的一条路径。
- 可行性。算法中的每一步都可以通过已经实现的基本运算的有限次执行得以实现。

从而可看出，具有算法特性的程序一定会结束，但并不是所有的程序都拥有此项特性。例如，一个死循环程序，或一个循环等待程序等。

1.2.2 算法描述

算法可以使用多种不同的方法来描述，如下所述。

- 自然语言——方便人们阅读，但不严谨。
- 伪码语言——介于高级语言和自然语言之间。
- 流程图——结构化的图标，简洁、明了。

本书为了使算法的描述和讨论简明清晰、容易被人理解，采用C语言描述，读者可以直接输入代码，上机调试书中的程序。同时，书中会给出完整实例的程序代码，以及相应的运行结果，以帮助大家拓展思路，上机练习。书中涉及到的基本操作的算法都用以下形式的函数描述：

```
函数类型 函数名(函数参数表)
{
//算法代码语句
}
```

1.2.3 算法性能分析

求解同一问题可能有许多不同的算法，究竟如何来评价这些算法的好坏，以便从中选出较好的算法呢？选用的算法首先应该是“正确”的。此外，主要考虑如下3点。

- 执行算法所耗费的时间。
- 执行算法所耗费的存储空间，其中主要考虑辅助存储空间。
- 算法应易于理解、易于编码、易于调试等。

一个占用存储空间小、运行时间短、其他性能也好的算法是很难做到的。原因是上述要求有时相互抵触：要节约算法的执行时间往往要以牺牲更多的空间为代价；而为了节省空间可能要耗费更多的计算时间。因此只能根据具体情况有所侧重。在目前计算机硬件价格快速下降的趋势下，算法的时间效率应首先予以考虑。

由于语句的执行要由源程序经编译程序翻译成目标代码、目标代码经装配再执行，语句执行一次实际所需的具体时间是与计算机的软、硬件环境（机器速度、编译程序质量、输入数据量等）密切相关，与算法设计的好坏无关。所以，若要独立于计算机的软、硬件系统来分析算法的时间耗费，则设每条语句执行一次所需的时间均是单位时间，一个算法的时间耗费就是该算法中所有语句的执行次数之和。

一个算法所耗费的时间=算法中每条语句的执行时间之和

每条语句的执行时间=语句的执行次数(即频度(Frequency Count))×语句执行一次所需时间

【例】 在一个一维数组中查找元素，其算法如下。

```
# define n 100          //n可根据需要定义,这里假定为100
int SequenceSearch(int a[], int n, int item)    //若查找成功则返回元素的下标,否则返回-1
{
  for(int i=0; i<n; i++)          //n+1     (1)
```

```
        if(a[i]==item) return i;          //n        (2)
    return -1;                        //1        (3)
}
```

该算法中所有语句的频度之和（即算法的时间耗费）为：

$$T(n)=n+1+n+1$$

分析：语句（1）的循环控制变量 i 要增加到 n，测试到 $i=n$ 成立才会终止，故它的频度是 $n+1$。但是它的循环体（即语句（2））却只能执行 n 次，语句（2）的频度是 n。同理可得语句（3）的频度是 1。

许多时候要精确地计算 $T(n)$是困难的，所以引入渐进时间复杂度在数量上估计一个算法的执行时间，也能够达到分析算法的目的。

定义（大 O 记号）：如果存在两个正常数 c 和 n_0，使得对所有的 n，$n \geq n_0$，有

$$f(n) \leqslant cg(n)$$

则有

$$f(n) = O(g(n))\ (O(g(n))\text{读做大 } O \text{ 的 } g(n))$$

$O(g(n))$给出了函数 $f(n)$的上界。

【例】 求如下两个 n 阶矩阵相乘运算程序段的时间复杂度。

```
for(i=1;i<=n;i++)
  for(j=1;j<=n;j++)
    {c[i][j]=0; /*基本语句 1*/

      for(k=1;k<=n;k++)
        c[i][j]=c[i][j]+a[i][k]*b[k][j]; /*基本语句 2*/
    }
```

设基本语句的执行次数为 $f(n)$，有

$$f(n)=n^2+n^3$$

因程序段的时间复杂度 $T(n)=f(n)=n^2+n^3 \leqslant c \times n^3=O(n^3)$，其中取 c 为 2，所以该程序段的时间复杂度为 $O(n^3)$。

使用大 O 记号表示的算法的时间复杂度，称为算法的渐进时间复杂度。

1.3 C 语言基础知识

1.3.1 C 语言基本知识点

在 C 语言中，数据类型可分为基本数据类型、构造数据类型、指针类型、空类型 4 大类。图 1.6 所示为 C 数据类型结构图。

1. 常量

常量按其表示方法可以分为直接常量和符号常量。