高职高专电子信息类精品课程规划教材

数据结构

■主编　刘　肖

西安电子科技大学出版社
http://www.xduph.com

精品课程

高职高专电子信息类精品课程规划教材

数据结构

主　编　刘　肖
参　编　石　锋　刘文哲
　　任　静　王　建

西安电子科技大学出版社
2010

内 容 简 介

本书从实际应用的角度出发，介绍了数据结构的基本知识和各种数据结构的实际应用。全书共分 8 章，主要内容包括线性表、栈与队列、串与数组、树、图、查找及排序等。各部分内容均从实际应用问题引入基本知识的讲解和描述，使读者更容易理解所学知识的应用目标，并在讲解中使用大量的实例来说明基本知识的应用。除第 1 章外，每章还包括了两个实训项目，配置了多种类型的习题，以突出实际应用能力的培养。

本书可作为高职高专学校计算机类专业学生学习“数据结构”的教材，也可作为软件技术人员的参考用书。为方便读者学习，本书的算法部分均采用 C 语言描述，实训项目也是完整的 C 语言程序，读者可以很方便地对书中的算法进行上机测试。

图书在版编目(CIP)数据

数据结构 / 刘肖主编. —西安：西安电子科技大学出版社，2010.2

高职高专电子信息类精品课程规划教材

ISBN 978-7-5606-2376-4

Ⅰ. 数… Ⅱ. 刘… 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 240101 号

策　　划　杨　璠

责任编辑　杨　璠

出版发行　西安电子科技大学出版社(西安市太白南路 2 号)

电　　话　(029)88242885　88201467　　邮　　编　710071

网　　址　www.xduph.com　　电子邮箱　xdupfxb001@163.com

经　　销　新华书店

印刷单位　陕西天意印务有限责任公司

版　　次　2010 年 2 月第 1 版　　2010 年 2 月第 1 次印刷

开　　本　787 毫米×1092 毫米　1/16　印　张　13.5

字　　数　319 千字

印　　数　1～3000 册

定　　价　20.00 元

ISBN 978-7-5606-2376-4/TP・1196

XDUP 2668001-1

前　言

“数据结构”是计算机程序设计的重要理论基础，是计算机相关专业必修的一门重要基础课程和核心课程。“数据结构”主要介绍各种类型的数据结构及在其上进行的各种运算，包含了较多的理论知识，也有较高的实践要求。为了达到理论与实践的结合统一，本书秉承以应用为主体，注重培养实践能力的指导思想，在强调理论知识的理解和运用的同时，更加注重结合高职高专教学的要求，体现对学生应用能力的培养。

本书主要面向高职高专院校计算机相关专业的学生。

全书共分 8 章。第 1 章为概述，主要阐述数据结构的相关概念、算法的描述和算法分析方法；第 2 章为线性表，主要讨论线性结构的顺序和链式表示方法、存储方法和各种操作方法以及应用；第 3 章为栈和队列，主要讨论栈和队列这两种限定性的线性表的表示、存储方法和操作方法以及应用；第 4 章为串与数组，主要讨论串与数组的存储方法和各种应用；第 5 章为树与二叉树，主要讨论树与二叉树的定义、存储方法、运算方法及应用；第 6 章为图，主要讨论图的各种概念、存储方法、遍历方法以及图的多种实际应用；第 7 章为查找，主要讨论查找的概念、各种查找方法及应用；第 8 章为排序，主要讨论排序的概念、各种排序方法及应用。

本书的编写力求概念简洁、叙述明了，以应用为主线编排和组织教学内容，突出实用性。总体来看，本书具有以下特点：

(1) 针对学生的学习特点，在每章开始均以“问题引入”、“知识要点”和“能力要求”开篇，以学习、工作及生活中的实际应用问题引入该章内容，以便学生能了解所学知识的应用目标，从而提高学生的学习兴趣，同时，给出该章的知识要点和能力要求，使学生明确学习目的。

(2) 运用高职高专教育先进的教学思想方法，做到理论知识的阐述由浅入深、通俗易懂，内容的组织和编排以应用为目的，略去了一些理论推导和数学证明的过程，淡化算法的设计分析和复杂的时空定量分析，采用算法说明和定性、定量分析相结合的方法设计和分析算法。

(3) 各章(除第 1 章外)都配有“实训指导”。每个“实训指导”均包含两个实训项目，第一个项目的重点在于对所学知识的理解和应用，第二个项目的重点在于对实际问题的解决。通过这两个实训项目，不仅帮助学生加深对基础理论知识的理解，而且培养学生解决实际应用问题的能力。

(4) 各章配有本章小结和课后习题，方便学生课后复习、巩固。

(5) 为方便学生学习，本书的算法部分均采用 C 语言描述，实训项目也是完整的 C 语言程序，读者可以很方便地对书中的算法进行上机测试。

本教材的教学时数以 60～80 为宜，其中上机安排 20 学时左右。教师可以根据实际的

教学时数和学生情况等自行调整教学进度和教学内容。

本书由西安航空技术高等专科学校计算机工程系刘肖老师主编，石锋老师、刘文哲老师、任静老师、王建老师等参与了编写。其中，刘肖老师编写了第 1、3、4 章，刘文哲老师编写了第 2 章，任静老师编写了第 5、6 章，石锋老师编写了第 7 章，王建老师编写了第 8 章。西安航空技术高等专科学校计算机工程系的领导和其他老师在本书编写过程中给予了大力的协助和支持，在此表示衷心的感谢。

由于时间紧迫，加之编者水平有限，书中难免存在疏漏，恳请读者批评指正。

编　者

2009 年 7 月

目　录

第1章 概 述

问题引入

我们知道，现在计算机的应用已涉及到生活和工作的方方面面，计算机所能处理的信息几乎无所不包。例如学校的教学管理系统就要把学生的信息、专业的信息、教师的信息、课程的信息等存储在计算机中并加以处理，以便能高效率地对教学过程做出安排和管理，方便各种信息的查询。那么，在计算机中如何设计、组织、存储和处理这些信息并最终使其帮助我们解决问题是值得讨论和研究的。这正是数据结构需要学习的内容。

知识要点

(1) 数据与数据结构的概念，数据的逻辑结构和存储结构；
(2) 数据结构研究的主要内容；
(3) 算法、算法描述与算法分析。

能力要求

明确数据结构需要学习和研究的内容以及数据结构在计算机学科中的作用。

1.1 引 言

早期的计算机主要用来处理数值数据的计算问题，如大数据量的方程计算等。如今，计算机已不再局限于数值型的科学计算方面的应用，而更多地用于数据处理、信息管理等非数值计算的各个方面。为了能使计算机有效地接收和处理这些数据，就必须分析这些数据的特性和数据之间的关系，合理地组织、存储这些数据和关系，提出处理这些数据的有效方法，这就是“数据结构”这门学科形成和发展的背景。

我们可通过对日常生活中的多种问题进行分析、归纳，提出应用计算机解决问题的方案，提高工作效率。

实例 1-1 学校学生信息结构。

表 1-1 所示的学生信息表就是一个简单的数据结构。表中每一行是一个学生的全部信息，每一行又有若干个数据项。假设一个班有 50 名学生，每个学生有 8 项数据，那么，一个班共有 400 个数据项。我们不能杂乱无章地存放这 400 个数据，通常是将每一个学生的信息放在一行。可将每一行看做是一个记录，多个学生的信息即多个记录排成若干行，呈

现一种线性关系。这样，才能方便地进行查找、浏览、插入、修改、删除、统计等工作。我们在日常生活中能见到大量类似的信息表，均可组织为线性结构，它们所需进行的运算也基本相同，我们就可把具有此种结构的问题归纳抽象为一类问题进行分析和研究，给出相应的解决方法，再将这种方法应用在该类的其它问题中。

表 1-1　学生信息表

学号	姓名	性别	班级	出生日期	年级	党/团员	生源地
07311201	刘畅	女	信息 121	1987-5-30	2007	团员	江苏
07311202	李宁	女	信息 121	1987-7-16	2007	团员	浙江
07311203	张军	男	信息 121	1987-1-20	2007	团员	陕西
…	…	…	…	…	…	…	…

实例 1-2　学校学生的组织结构。

实例 1-1 中虽然列出了学生的名单，但每个学生是有所属系和班级的。一个学校的组织机构可用图 1-1 所示的结构来描述，它是一种层次结构，上、下层数据之间的连线表明了各个机构之间存在着隶属关系。如果要查询某班的学生，可沿着隶属关系从上层到下层进行查询。这种结构很像一颗倒挂的树，树根在最上层。类似于这类结构，如家族的族谱、操作系统中文件的组织结构等，都可用这种树形结构进行描述。

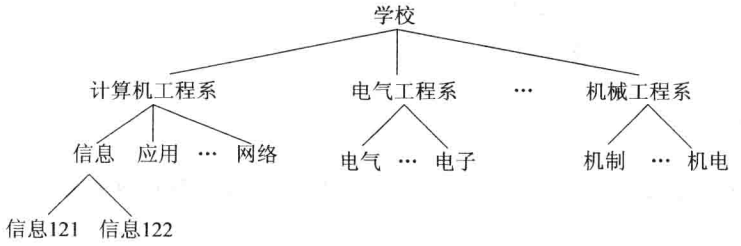


图 1-1　学校学生的组织结构

实例 1-3　城市交通图。

如果我们在某城市旅游，旅游景点有多个，它们之间都有直接或间接的交通路线。从经济或时间的角度考虑，我们都希望寻找一条最佳的路线，这就需要我们在如图 1-2 所示的图形结构中找出满足我们需求的线路。这一问题如果交给计算机来解决，就需要计算机能掌握所有景点的信息以及景点和景点之间的有关联系(例如是否可直达，所花费的时间是多少等)，然后通过计算，从中找出所需线路。

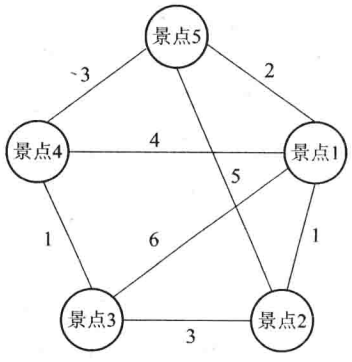


图 1-2　旅游景点交通图

通过上面的几个实例我们可以看出，计算机要处理的信息是复杂的，但不是杂乱无章的，每个问题中不仅包含大量的数据信息，而且这些数据之间还存在着各种各样的关联关系，只有了解和掌握了这些数据和它们之间的关联关系，才能进一步找寻解决问题的方法。我们可将这些关联关系进行归纳、分类，从而可以针对一类问题进行分析、研究，设计出该类问题在计算机中的解决方案。

可以简单地说，数据结构就是研究非数值类型数据以及它们之间的关系和操作运算的一门学科。

1.2　基本术语及概念

1.2.1　基本术语

1．数据(data)

数据是指所有能够输入到计算机中，并被计算机识别、存储和处理的符号的集合。数据可分为数值数据和非数值数据。数值数据包括整数、实数等；非数值数据包括字符、文字、图形、声音和图像等，这些数据通常通过数字编码方式实现。

例如，一个整数，一篇 Word 文档，一幅 jpg 格式的照片，一首 mp3 歌曲，一段数字视频，一张包含学号、姓名及各科成绩的成绩单等，都可称为数据。

2．数据元素(data element)

数据元素是组成数据的基本单位，在计算机程序设计中通常作为一个整体进行考虑和处理。一个数据元素可以包含一个或若干个数据项。在不同的结构中有时又把数据元素称为元素、结点、顶点或记录。

例如，学生信息表中的每一行、旅游景点交通图中的一个景点就是一个数据元素。

3．数据项(data item)

数据项是数据不可分割的、具有独立意义的最小单位，有时也称为字段或域。

例如，学生信息表中每一行的姓名就是一个数据项。

4．数据对象(data object)

数据对象是一组具有相同性质的数据集合。在程序设计语言中通常都是用标准的数据类型和构造类型来描述一个数据对象的。

例如，学生信息表就是一个数据对象，在 C 语言中可描述如下：

```
typedef struct
{ char no[8];                  //学号
  char name[20];               //姓名
  char sex[2];                 //性别
  …                            //其它数据项
}student;                      //学生成绩单类型
student  stu[100];             //可存放由 100 个学生信息组成的数据对象
```

数据、数据元素、数据项是数据组织的三个层次，数据可由若干个数据元素组成，而数据元素又可由一个或若干个数据项组成。数据对象是数据的一个子集，具体到一个实际问题的时候，我们都是用数据对象来描述这个问题所涉及到的各种信息的。所以，解决问题实际上就是在数据对象上完成必要的操作运算以求得我们所需的结果。

1.2.2　数据结构

简单地说，数据结构就是数据的组织形式，它研究的内容包括数据与数据之间的逻辑关系、数据在计算机中的存储方式和在其上定义的一组运算。

1．数据的逻辑结构

数据元素之间存在的一种或多种逻辑上的关系称为数据的逻辑结构。数据的逻辑结构从逻辑关系上描述数据，与数据的存储无关，是独立于计算机的。数据的逻辑结构可以看做是从很多具体问题抽象出来的一种模型。

从逻辑结构角度来说，数据结构可分为四种类型，其形态如图 1-3 所示。

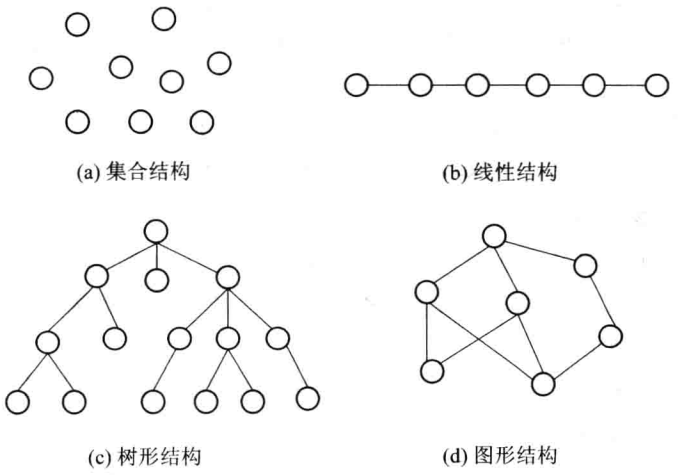


图 1-3　四种类型的数据结构示意图

(1) 集合结构：数据元素之间除了同属一个集合外没有其它关系；

(2) 线性结构：数据元素之间存在着一对一的线性关系；

(3) 树形结构：数据元素之间存在着一对多的层次关系；

(4) 图形结构：数据元素之间存在着多对多的图形关系。

其中，线性结构、树形结构和图形结构是我们学习的重点。树形结构和图形结构又统称为非线性结构。

2．数据的存储结构

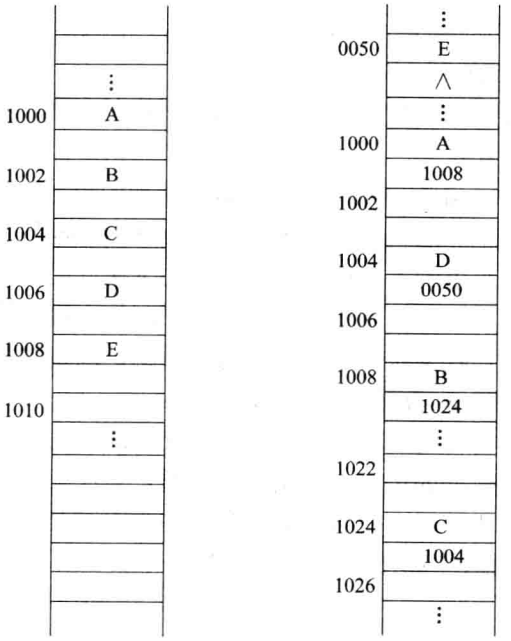
数据元素及其关系在计算机存储器内的表示称为数据的存储结构。数据的存储结构是逻辑结构在计算机存储器中的实现，一般用计算机高级语言实现。也就是说，存储结构依赖于计算机高级语言。本教材选择类 C 语言来描述和讨论数据的存储结构。

数据的存储结构主要有两种，即顺序存储结构和链式存储结构。

顺序存储的特点是使用内存中一组地址连续的存储单元来存放数据。数据元素之间的逻辑关系通过元素在内存中存放的相对位置来确定，也称为向量存储，在高级语言中通常用数组实现。例如，将线性表(A，B，C，D，E)以顺序存储结构的方式存放在以 1000 为起始地址的内存向量中，每个字符占两个存储单元，其存储形式如图 1-4(a)所示。在此结构中，

若要读取第三个字符，可通过该线性表的起始地址和要读取的数据元素的次序计算得到。

链式存储的特点是使用内存中一组地址不连续的存储单元来存放数据，数据元素之间的逻辑关系通过指针来反映，也称为动态存储。例如，将线性表(A，B，C，D，E)以链式存储的方式存放，那么，第一个数据元素可存放在地址为1000的内存单元中，第二个数据元素的存放地址可以是其它任何位置，但为了保持与第一个数据元素的线性关系，必须在第一个数据元素的有关位置再存储一个指向第二个数据元素的指针(即存储地址)，其存储形式如图1-4(b)所示。



(a) 顺序存储结构　　(b) 链式存储结构

图1-4 顺序存储结构和链式存储结构示意图

3. 数据的运算

对数据元素施加的各种操作称为数据的运算。数据的运算定义在数据的逻辑结构上，而运算的实现依赖于数据的存储结构。

在高级语言中，对已有的标准数据类型，其运算已提供，例如C语言中的整数类型就可进行加、减、乘、除以及求模等算数运算。对用户自定义的数据对象，就需设计人员提供相应的运算，例如在C语言中要实现学生信息的管理，就需自行定义查询、插入、删除等操作的运算方法。

1.3 算法描述与算法分析

解决任何一个实际问题都需要有一种适合的、有效的方法。数据结构研究的一项重要内容是数据的运算，也就是解决问题的步骤和方法，这就是算法的概念。

1.3.1　算法与算法描述

1．算法

算法(algorithm)是指解决特定问题的一种方法或有穷步骤的集合。

例如，我们要求解 100 以内的所有素数，可以采用如下的步骤进行：

(1) 给出数据序列 2，3，4，…，100；

(2) 以序列中最小的数 2 作为因子，将其余数中所有能被 2 整除的数从序列中删除；

(3) 取剩余序列中最小的数 i 作为因子，将其余数中所有能被 i 整除的数从序列中删除；

(4) 重复步骤(3)，直到取得序列中最后一个数。

这就是用筛选法求素数的一个算法，对 2～100 之间的整数经过有限、可行的运算步骤后得到了 100 以内的所有素数。

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》是第一部较系统地阐述数据的逻辑结构、存储结构及算法的著作。随着人们工作需求的不断增加和软件业的飞速发展，各种应用程序变得越来越庞大和复杂，所以程序的结构设计就变得越来越重要，作为程序设计方法学基础的数据结构也就得到了更多的重视，普遍认为程序设计的实质就是确定数据结构，并加上一个好的算法，即人们所说的“程序 = 数据结构 + 算法”。

算法和数据结构有着密切的联系。数据结构不仅要确定对应问题的逻辑结构和存储结构，还要在此基础上研究数据的运算。数据运算要通过算法实现。实际上，在数据结构的学习中，算法占了很大的比例，也是数据结构学习的重点和难点，它要求学习者要具有一定程序设计的基础，掌握描述算法的高级语言的基本方法和技巧，并达到灵活应用的程度。

2. 算法的特性

算法作为解决问题的步骤和程序设计的依据，需具有下列特性：

(1) 输入：一个算法具有零个或多个输入数据。

(2) 输出：一个算法具有一个或多个输出数据。

(3) 确定性：算法执行的每一步都必须有确切的唯一定义，不能有二义性。

(4) 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。

(5) 有限性：一个算法必须在经过有限个步骤之后正常结束，不能形成死循环。

从算法的特性上看，算法在某种意义上和程序非常相似，但又不同。对比这两者可以看出算法侧重于对解决问题的方法描述，即要做些什么；而程序是算法在计算机程序语言中的实现，即具体要去怎样做。所以从严格意义上讲，算法和程序是两个不同的概念。但有时，我们直接把计算机程序看做是算法的一种描述，那么，算法和程序就是一致的了。

3．算法描述

算法的描述有很多种。前面求素数的实例就是用自然语言描述算法，我们学习高级语言的时候也曾编写过求素数的 C 语言程序，那也是一种算法描述。通常，可采用下列四种方法描述一个算法：

(1) 流程图算法描述：这种描述方法直观、易懂，但用来描述比较复杂的算法时显得不

够方便，也不够清晰简洁。

(2) 非形式算法描述：即用自然语言，同时还使用一些程序设计语言中的语句来描述算法。这种方法比较自然，方便表达，但经常会不够准确，容易产生二义性。

(3) 类语言算法描述：类语言算法又称为伪语言算法。这种算法不能直接在计算机上运行，但专业设计人员经常使用类 C 语言、类 Java 语言等来描述算法，它容易编写、阅读和统一格式，也比较容易转换为高级语言程序，便于算法实现。

(4) 高级语言编写的程序或函数：这是可在计算机上运行并获得结果的算法，使给定问题能在有限时间内被求解，但要求符合高级语言的规则。这种算法实际上就是程序。

下面我们通过实例来看一下不同算法描述的具体方法。

实例 1-4　求两个整数 m，n(m≥n)的最大公因子。

(1) 流程图描述：在学习程序设计语言的时候，我们都曾学习过流程图的设计方法，针对此问题，可以根据题意给出如图 1-5 所示的流程图。它通过比较直观的方法给出了解决问题的流程。

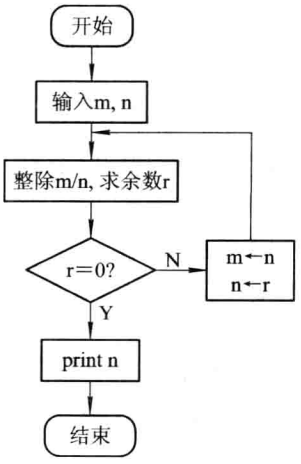


图 1-5　流程图描述法

(2) 非形式语言描述：

① 求余数：以 n 除 m，并令 r 为余数(0≤r<n)；

② 判断余数是否为零：若 r = 0，则结束算法，n 就是最大公因子；

③ 替换并返回①：若 r ≠ 0，则 m←n，n←r，返回①。

(3) 类语言描述：

```
int   max_common_factor (int m, int n)
{ int   r ;
  r = m % n;
  while (r != 0)
     {m = n ;   n = r ;   r = m % n;}
  return   n ;
 }
```

(4) 程序描述：

```
void   main()
{ int   m,n,r;
  scanf("please input m,n:\n",&m,&n);
  r = m % n;
  while (r != 0)
    {m = n ;   n = r ;   r = m % n;}
  printf("max_common_factor is: ",r);
 }
```

本教材采用类 C 语言来描述算法，主要目的是介绍算法的思路和实现过程，但也尽可能地将算法对应的 C 语言函数或程序提供给读者，以便读者更好地理解算法和实现算法。

1.3.2 算法分析

对于同一个问题可以设计出不同的算法，它们之间肯定有是否适合和“好”“差”之分。衡量一个适合的、“好”的算法，在其必须是正确的算法的基础上，还应考虑下列几个方面：

(1) 易读性：算法应易于阅读和理解，以便于调试、修改和扩充。

(2) 高效性：算法应具有较高的时间效率和空间效率，即占用较短的执行时间和较少的存储空间。当然，这两者都和问题的规模有关。

(3) 健壮性：正确的输入能得到正确的输出，这是算法必须具有的特性之一。但当遇到非法输入时，算法应能做出反应或处理(如提示信息等)，而不会产生不需要的或不正确的结果。

例如，我们在算法中总是采用较简单的方法以及模块化函数，以增强算法的易读性；采用步骤较少的求解方法，以提高算法的时间效率；对可能发生的各种现象及输入形式给出应对措施，以提高算法的健壮性。

要确定一个算法是适合的、“好”的算法，就需进行算法分析。算法分析的两个主要方面是分析算法的时间效率和空间效率，目的是以求改进算法或对不同的算法进行比较。在目前情况下，鉴于运算空间较为充足，我们把算法的时间效率分析作为主要内容。

算法运行的时间分析和程序运行的时间分析是有区别的。同一算法由不同的程序员来编写，所编写的程序会有优劣之分，程序运行的时间也就有所不同；程序在不同的机器上运行的速度又和机器本身的速度有关。而我们感兴趣的是对解决问题的算法作时间上的度量分析，或对解决同一问题的两种或两种以上算法的运行时间加以比较。

估算算法运行时间需要考虑的基本因素是问题的“规模”和算法执行“基本操作”的次数。一个算法的“规模”和“基本操作”要视具体问题而定。“规模”一般是指输入量的数目，比如在排序问题中，问题的规模可以是待排序的元素数目。数据量越大，算法执行花费的时间就越多，它是影响时间的最主要的因素。“基本操作”一般是指算法在解决某个问题时必须涉及到的主要操作，它与操作数的具体取值无关，比如在查找运算中，两个数据的比较就可视为基本操作。显然，在一个算法中，执行基本运算的次数越少，其运行时间也就相对地越短；执行次数越多，其运行时间也就相对地越长。通常采用算法中基本运

算的执行次数来描述算法的时间效率，称为算法的时间复杂度，它是与问题规模 n 有关的函数，记为 T(n)，主要用来估算当问题的规模变大时算法运行时间增长的速度，因此可将其表示为与问题规模函数 f(n)具有同样增长率的函数：

$$T(n)=O(f(n))$$

记号“O”表示随着问题规模 n 的增大，算法执行时间的增长率和 f(n)的增长率相同，即当 n 足够大时，T(n)与 f(n)成正比，因此，算法的时间复杂度主要是分析 T(n)的数量级。我们在实际计算时可忽略 f(n)的低阶项和常数项，这样既可以简化 T(n)的计算，又能比较客观地反映出当 n 很大时算法的时间性能。一般常见的时间复杂度形式有：$O(1)$、$O(\log_2 n)$、$O(n)$、$O(n\log_2 n)$、$O(n^2)$、$O(n^3)$、$O(2^n)$等。在 n 值较大的情况下，常见的时间复杂度之间存在下列关系：

$$O(1)<O(\log_2 n)<O(n)<O(n\log_2 n)<O(n^2)<O(n^3)<O(2^n)$$

实例 1-5 试分析在 n 个整数中查找最大数据的算法。

将 n 个整数存放在数组 array 中，用变量 large 保存当前最大的数据元素，依次和所有数据元素比较，具体算法如下：

```
int   largest (int *array,   int n)
{ int   large,   i ;
  large = array [0] ;
  for(i = 1; i < n ; i++)
    if (array [i]  > large)   large = array [i] ;
  return   large ;
}
```

这个问题是在 n 个整数中查找最大的数据元素，所以问题的规模为 n，基本操作是“比较”，即将数组中的一个整数和现有的最大整数作比较。算法所花费的时间主要由基本操作——“比较”的执行次数决定，这是一个有关问题规模的函数 f(n)。通过分析可以得知，比较的次数也就是循环的执行次数，f(n)= n，因此，该算法的时间复杂度 T(n)= O(n)。

这表明了顺序查找数组中最大元素的算法的时间随着 n 的增长而线性增长。

实例 1-6 试分析求两个 n 阶矩阵相加(C = A + B)的算法。

```
void matriadd(int *A,int *B,int *C,int n)
{ int   i,   j ;
  for(i = 1; i < n ; i++)
    for(j = 1; j < n ; j++)
      C[i][j] = A[i][j] + B[i][j];
}
```

这个问题是将两个 n 阶矩阵相加，所以问题的规模为 n，基本操作是矩阵对应元素的“加”运算，这是由两重循环确定的运算，执行次数为 n^2，所以 $f(n)= n^2$，算法的时间复杂度 $T(n)= O(n^2)$。

一般情况下，计算一个算法的基本运算次数是有困难的，因为一个算法的不同输入往往产生不同的运算次数，而一个算法的所有不同输入的数目可能十分庞大。一种有效的方法是计算基本运算的平均运算次数或只考虑运算次数的最高阶次。

实例 1-7　试分析在 n 个整数中查找值为 k 的数据的算法。

```
int find(int a[ ],int n,int k)
{ int i;
  i=0;
  while(i<n && a[i]!=k)
     i++;
  return i
}
```

这个算法的执行时间不仅与问题规模 n 有关，而且还与实际输入的待查找数据 k 的值有关，即输入不同的 k 值，查找所需的比较次数不同。最好情况下，a[0]中存放的就是要查找的数据，循环一次也不执行；最坏情况下，a 中无要找的数据，循环的执行次数是 n 次。简单的方法，我们可以只考虑最坏的情况，即基本操作的执行次数是 f(n)= n，算法的时间复杂度是 T(n)=O(n)。另外，我们还可通过求其平均值的方法确定时间复杂度。在该问题中，k 值出现在任何位置的概率都为 1/n，循环内语句的平均执行次数为

$$\frac{0+1+2+\cdots+(n-1)}{n}=\frac{n-1}{2}$$

而时间复杂度只需考虑基本运算执行次数的最高阶次，忽略其系数、低阶项和常数项，所以算法的时间复杂度仍为 T(n)=O(n)。

小　结

本章主要介绍数据结构及相关的基本概念和数据结构研究的内容，应重点理解和掌握的知识和概念包括：

(1) 数据是指所有能够输入到计算机中，并被计算机识别、存储和处理的符号的集合。数据元素是组成数据的基本单位。数据结构就是数据的组织形式，它研究的内容包括数据与数据之间的逻辑关系、数据在计算机中的存储方式和在其上定义的一组运算。数据的逻辑结构可以看做是从很多具体问题抽象出来的一种模型。数据的存储结构是数据元素及其关系在计算机存储器内的表示。

(2) 在数据结构中，数据元素之间的逻辑关系可分为集合、线性、树形、图形四种类型。数据的存储结构则主要有两种形式，即顺序存储结构和链式存储结构。每一种数据的逻辑结构都可以选择任何一种存储结构来实现，针对不同的存储结构，同一种运算的实现方法不同。

(3) 算法是指解决特定问题的一种方法或有穷步骤的集合，是数据结构研究的重要内容之一，也是学习数据结构的难点。算法的实现有多种方法，但需满足算法的基本特性，即输入、输出、有穷性、确定性和可行性。一个“好”的算法应具有易读性、高效性和健壮性。可通过时间复杂度来描述算法的时间效率。算法的时间复杂度和算法中基本运算的执行次数具有相同的增长率。