SHUJU JIEGOU SHIJIAN JIAOCHENG

数据结构实践教程

主　编　周　莹　郑　茵
副主编　刘志宝　王　珂　王婷婷
参　编　郭桂杰　朱伟华　孙文武

東北大學出版社
Northeastern University Press

数据结构实践教程

主　编　周　莹　郑　茵
副主编　刘志宝　王　珂　王婷婷
参　编　郭桂杰　朱伟华　孙文武

东北大学出版社
·沈　阳·

图书在版编目（CIP）数据

数据结构实践教程／周莹，郑茵主编. —沈阳：东北大学出版社，2014.7
ISBN 978-7-5517-0631-5

Ⅰ. ①数…　Ⅱ. ①周…　②郑…　Ⅲ. ①数据结构—教材　Ⅳ. ①TP311.12

中国版本图书馆 CIP 数据核字（2014）第 141083 号

出 版 者：东北大学出版社
地址：沈阳市和平区文化路 3 号巷 11 号
邮编：110819
电话：024-83687331（市场部）　83680267（社务室）
传真：024-83680180（市场部）　83680265（社务室）
E-mail：neuph@neupress.com
http：//www.neupress.com
印 刷 者：沈阳中科印刷有限责任公司
发 行 者：东北大学出版社
幅面尺寸：185mm×260mm
印　　张：10.5
字　　数：256 千字
出版时间：2014 年 8 月第 1 版
印刷时间：2014 年 8 月第 1 次印刷
策划编辑：郭爱民
责任编辑：潘佳宁　　责任校对：孙　爽
封面设计：刘江旸　　责任出版：唐敏志

ISBN 978-7-5517-0631-5　　定　　价：29.00 元

前　言

“数据结构实践教程”是计算机专业中一门重要的专业基础课程，主要面向计算机软件设计、项目管理、数据库管理等行业，培养从事计算机软件的开发、管理、维护等岗位工作的应用型人才。分析当前高职学校计算机类专业群的工作岗位、人才的职业能力、工作任务和工作内容，确定典型工作任务，建立基于工作过程的课程体系。在课程体系中，确定“数据结构”课程在各专业培养计划中的支撑作用，据此进行教学内容、教学模式和教学方法的设计，以追求更好的教学效果。

本书以项目为导向，以任务驱动模式组织教学，工学结合，其宗旨是将数据结构与算法设计有机地结合起来，系统介绍了数据结构的基本概念及主要的算法设计方法。将数据结构算法更立体的应用到实践中。提高学生的动手能力、创新能力以及就业能力。通过项目的设计，让学生在模拟工作岗位中担任各自的角色，为就业打下良好的基础。

本教材具有以下特点：

（1）新教材体现了以学生发展为本的教育理念；

（2）新教材以“任务驱动法”为体例，有利于学生进行探究性学习；

（3）为提高教材的适用性，所选项目内容丰富，能使学生产生浓厚兴趣，从而增强学习积极性。

本书共分6个学习情境，分别介绍了线性表、栈和队列、树和二叉树、图以及常用的排序和查找方法。通过学习，学生应能够理解数据结构的概念，掌握线性结构、树形结构和图形结构的基本存储方式、基本算法和简单应用，熟练掌握常用的排序和查找的算法，并能进行简单的算法分析。本书编写的原则是着眼于实用、注重发展。

参加本书编写的有吉林电子信息职业技术学院周莹、郑茵、刘志宝、王珂、王婷婷、郭桂杰、朱伟华、孙文武。周莹、郑茵参与编写全部学习情境，刘志宝、王珂、王婷婷、郭桂杰、朱伟华、孙文武依次编写学习情境一、二、三、四、五、六。周莹、郑茵担任主编，刘志宝、王珂、王婷婷担任副主编，郭桂杰、朱伟华、孙文武参与编写。在编写的过程中，得到许多同行的支持和帮助，在此表示衷心的感谢！

由于编者水平有限，书中难免有不妥之处，敬请广大读者批评指正。

编　者

目 录

学习情境1 学生成绩管理系统的构建

学习情境2 航空客运订票系统的构建

学习情境 1

学生成绩管理系统的构建

“学生成绩管理系统”工作任务单

项目名称	项目 1：学生成绩管理系统 （课外：考试报名管理系统）		
学习领域	数据结构的两种存储结构		
班级		小组成员 第（ ）组	

一、目的

通过完成本工作任务，使学生能够根据用户需求合理选择数据结构，并能制定出合理的算法设计方案。本项目的实质是完成对学生成绩信息的建立、查找、插入、修改、删除等功能，可以首先定义项目的数据结构，然后将每个功能写成一个函数来完成对数据的操作，最后完成主函数以验证各个函数功能并得出运行结果。

二、要求

1. 根据需求给出该项目的数据流分析；
2. 根据需求制定数据存储的表示；
3. 完成该项目的算法设计；
4. 实现该项目的具体功能及测试；
5. 提交项目成果。

三、环境要求

软件：Visual C++6.0

四、工作内容

1. 每个学生的信息包括：学号、姓名、性别、出生年月、成绩、科目、电话等。系统能够完成对学生成绩信息的查询、更新、插入、删除、排序等功能。

2. 作为模拟系统，全部数据可以只存放在内存中。

3. 通过此系统可以实现如下功能。

① 排序。按不同关键字，对所有学生的成绩信息进行排序。

② 查询。按特定条件查找学生的相关成绩。

③ 更新。按编号对某个学生的某项信息进行修改。

④ 插入。加入新学生的信息。

⑤ 删除。按编号删除相关的学生成绩信息。

1.1 任务1：

数据结构的基本概念

1.1.1 子任务1：什么是数据结构

目前，计算机已广泛地应用于情报检索、事务管理、系统工程等领域。与此相应的，计算机加工处理的对象也从简单的纯数值型信息发展到数、字符、字符串、表、文件、图像、声音等各种复杂的、具有一定结构的数据。人们称前者为数值问题，而后者为非数值问题。

非数值问题要求用复杂的数据结构来描述系统的状态，它们的运算是实现对数据结构的访问或修改。当前要设计出效率高、可靠性强的非数值程序，要求程序设计人员不但要掌握一般的程序设计技巧，而且还必须研究计算机程序加工的对象，即研究各种数据的特性以及数据之间的关系，这就促进了数据结构这一学科的发展。然而，数据必须在计算机中进行处理，因此，不仅要考虑数据本身的数学性质，还必须考虑数据在计算机内的存储方式和相应的运算，从而扩大了数据结构研究的范围。随着数据库系统、情报检索系统的不断发展，在数据结构领域中又增加了文件结构，特别是增加了大型文件的组织和B树、B+树的知识，使得数据结构逐步成为了一门比较完整的学科。

什么是数据结构呢？先举一个简单的例子予以说明，然后再给出明确的定义。

假定有一个学生成绩管理系统，记录了某班级学生的姓名和相应的成绩等信息，现要求设计一个算法：当给定某一位学生的姓名时，计算机能够查出该学生的成绩及相关信息，如果根本就没有这个人，计算机也能报告“查无此人”。

这种任务称为“查找”。我们可以发现，这个算法的设计完全依赖于学生成绩管理系统中姓名和相应信息在计算机内的存储方式。

一种方法是学生成绩管理系统中学生的姓名是随意排列的，其顺序没有任何规律。那么，当给定一个姓名时，就只能从学生成绩管理系统的第一个姓名开始，逐个与给定的姓名进行比较，直至找到指定的姓名，接着打印出他的相关成绩信息，或者查完整个学生成绩管理系统还没找到此人，这时应给出相应的标志。这种方法很简单，当学生人数较少时是可行的，但对于一个学生人数较多的情况来说，就相当费时间且效率太低。

上述学生成绩管理系统的组织方式问题就是一个数据结构问题，不同的数据结构，得出不同的算法。由此可见，计算机的算法与数据结构密切相关，即每一个算法无不依赖于具体的数据结构，而数据结构直接影响着算法的选择和算法的效率。

通过以上讨论可以直观地认为，数据结构是一门研究程序设计中计算机操作的对象以及它们之间关系和运算的一门学科。

几种基本的数据结构是集合、线性结构、树形结构和图形结构。

1.1.2 子任务2：数据结构相关术语

(1) 数据 (Data)

计算机中的数据是广义的，包括了数、字符、字符串、表、文件等，声音、图形、图像都属于数据的范畴。

(2) 数据元素 (Data element)

数据元素是数据的基本单位，即数据集合中的个体，在计算机程序中通常作为一个整体来考虑和处理。每个数据元素可以只有一个数据项，通常称为域 (field)，也可以由若干个数据项组成。例如，学生成绩管理系统中，每个学生的情况为一个数据元素，而其中的学号、姓名、成绩，年龄等信息分别称为数据项。数据项是数据的最小单位。数据元素的同义语有结点 (Node)，顶点 (Vertex) 和记录 (Record) 等。

(3) 数据对象 (Data object)

数据对象是性质相同的数据元素的集合。它是数据的一个子集，例如，整型数据对象是集合{0，±1，±2，…}。数据对象可以是有限的，也可以是无限的。

(4) 数据结构 (Data structure)

同一数据对象中的数据不是孤立的，而是彼此相关的。数据结构研究的是数据元素之间抽象化的相互关系和这种关系在计算机中的存储表示。对每种结构定义各自的运算，设计出相应的算法。

数据结构与数据对象不同，在描述一种数据结构时，不但要描述数据对象，还要描述数据元素之间的相互关系。

1.2 任务2：

数据结构的两种存储结构

1.2.1 子任务1：数据结构的分类

严格地说，数据结构包括两方面的内容：数据的逻辑结构 (Logical structure) 和数据的物理结构 (Physcical structure)。

(1) 数据的逻辑结构

数据的逻辑结构指各数据元素之间的逻辑关系，是用户按使用需要建立起来，并呈现在用户面前的数据元素的结构形式，如图1.1所示。

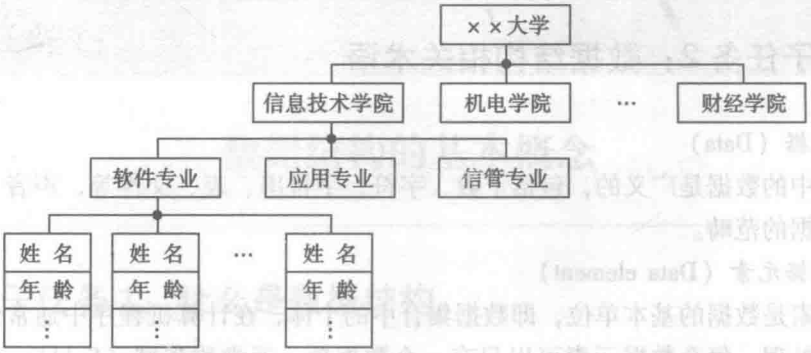


图 1.1 数据的逻辑结构

(2) 数据的物理结构

数据的物理结构又称为数据的存储结构，是指数据在计算机内实际的存储形式。举例来说，如果要建立某学院教师的人事档案，其逻辑结构如图 1.1 所示，它实际上是一种树形结构。要把这些数据存储到计算机中，不仅要考虑数据存放的先后次序，还要考虑那些反映教师各种情况的数据项，如姓名、年龄、工资……如何在计算机中存放，即要考虑数据的物理结构。

数据的逻辑结构和物理结构是密切相关的。我们将会看到，在有的结构类型中这二者是一致的，有的则是不一致的。为了叙述上的方便和避免产生混淆，常把数据的逻辑结构简称为数据结构，将数据的物理结构称为存储结构。

每种数据结构都可通过映象的方式得到相应的存储结构。常用的映象方式有两种：顺序映象和非顺序映象。由此，可得出两种不同的存储结构：顺序存储结构和链式存储结构。

顺序存储结构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，链式存储结构则是借助指示元素地址的指针表示数据元素之间的逻辑关系。

人们也常常用 Hash 和索引方式映象，从而可得到 Hash 存储结构和索引存储结构。

(3) 数据类型 (Data type)

数据类型是程序设计语言中各变量可取的种类。各种程序设计语言不仅规定了每种类型变量的取值范围，而且还规定了该类型变量能执行的运算。有些语言允许由内部类型构造出新的数据类型。可以把数据类型看作是程序设计语言中已经实现的数据结构。

数据结构是在程序设计语言的基础上由用户建立起来的，它依靠程序设计语言提供的数据类型来描述数据的逻辑结构，也依靠程序设计语言提供的各种设施来定义，描述运算及其算法。这些运算按实际问题的需要由用户自己定义，而不是由语言系统事先规定。因此，数据类型和数据结构的主要区别是：前者面向系统，后者面向对象，是高一层的数据抽象。如果程序设计语言提供有抽象数据类型设施，则二者可以统一起来。

(4) 抽象数据类型 (Abstract data type)

抽象数据类型是指一个数学模型以及定义在该模型上的一组操作。抽象数据类型的定义仅仅取决于它的一组逻辑特性，而与它在计算机内的表示和实现无关，即不论其内部结构如何变化，只要它的数学特性不变，就不会影响其外部的使用。

在面向对象的程序设计 (Object oriented programming) 技术中，数据和对数据的操作

是融合在一起的。一般借助“对象”来描述抽象数据类型，一旦定义了一个对象，就可使用对象的名字来说明变量，调用其中的操作，从而实现信息的隐蔽和封装。

抽象数据类型可以通过高级语言中已有的数据类型来表示和实现。

(5) 数据结构的运算

数据结构的主要运算包括以下几种：

① 建立（Create）一个数据结构的运算；

② 消除（Destroy）一个数据结构的运算；

③ 从数据结构中删除（Delete）一个数据元素的运算；

④ 把一个数据元素插入（Insert）到一个数据结构中的运算；

⑤ 对一个数据结构进行访问（Access）的运算；

⑥ 对一个数据结构进行修改（Modify）的运算；

⑦ 对一个数据结构进行排序（Sort）的运算；

⑧ 对一个数据结构进行查找（Search）的运算。

以上是主要运算，还有一些其他运算。

1.2.2 子任务2：数据结构的案例

【例1.1】电话号码查询系统。

信息技术学院	12020501
财经学院	12010601
国际交流学院	12030701
机电学院	12040801
材料学院	12050901
M	M

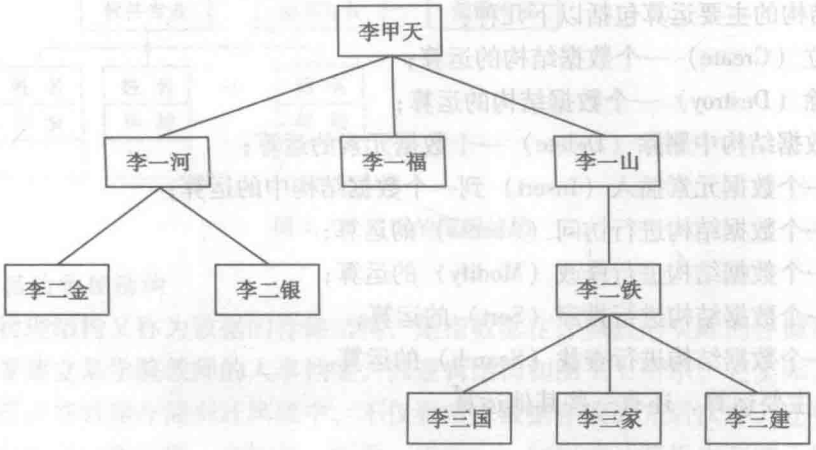
12020501	信息技术学院
12010601	财经学院
12030701	国际交流学院
12040801	机电学院
12050901	材料学院
M	M

操作对象：{单位名，号码}

关　　系：线性关系

算　　法：查询、插入、修改、删除……

【例 1.2】家谱树。



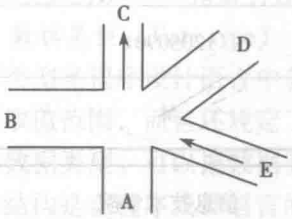
操作对象：家谱排序

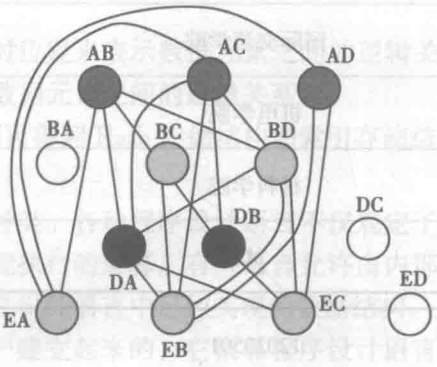
关　　系：非线性关系（树形结构）

算　　法：查询、插入、修改、删除……

【例 1.3】多叉路口交通灯管理问题。

在多叉路口需设几种颜色的交通灯才能使车辆相互之间不碰撞，又能达到最大流通量。





操作对象：通路

关系：非线性关系（图形结构）

算法：查询、插入、修改、删除……

1.3 任务3：

算 法

1.3.1 子任务1：算法的基本概念

算法（解决问题的方法）处理的对象就是数据。

算法与数据的结构密切相关，算法无不依附于具体的数据结构，数据结构直接关系到算法的选择和效率。程序设计的实质是对实际问题选择一种好的数据结构，加之设计一个好的算法。严格地讲，算法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。

1.3.2 子任务2：算法的特性

算法是执行特定计算的有穷过程，该过程有下述几个特性。

① 有穷性。当执行一个算法时，不论是何种情况，在经过了有限步骤后，该算法一定会终止。

② 确定性。算法中的每一条指令都必须是清楚的，执行算法时不会产生二义性。并且在任何条件下，算法只有唯一的一条执行路径，即对于相同的输入有相同的输出。

③ 输入。一个算法有零个或多个由外界提供的量。

④ 输出。一个算法产生一个或多个输出。

⑤ 可行性。算法是可行的，即算法中描述的操作都可以通过已实现的基本运算的执行来完成。

算法的设计要求如下。

(1) 正确性

算法应满足具体问题的需求。“正确”的含义在通常的用法中有很大的差别，大体可分为以下四个层次：

① 程序不含语法错误；

② 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；

③ 程序对于精心选择的典型、苛刻而带有刁难性的几组数据能够得出满足规格说明要求的结果；

④ 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。

通常以第③层意义的正确性作为衡量一个程序是否合格的标准。

(2) 可读性

算法在正确的前提下，可读性是摆在第一位的，这在当今大型软件需要多人合作完成的环境下是至关重要的。另一方面，晦涩难读的程序易于隐藏错误而难以调试。

(3) 健壮性

算法应具有容错处理能力。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，而不是产生莫名其妙的输出结果。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) 效率与低存储量需求

效率，指的是算法执行的时间（时间复杂性）；存储量需求，指算法执行过程中所需要的最大存储空间（空间复杂性）。一般这两者与问题的规模有关。

1.3.3 子任务3：算法分析及举例

一个算法的执行时间等于其所有语句执行时间的总和。而任一语句的执行时间为该语句执行一次所需时间与执行次数的乘积。要精确地计算各语句执行一次所需要的时间是十分困难的，因为它取决于执行算法的具体的计算机及其所配备的编译系统的质量以及运行的环境等众多因素。因此，在我们的分析工作中只是粗略地将算法中语句执行的最大次数（称为语句的执行频度）作为算法时间的量度，而不需要计算出它的具体执行时间。

设有以下三个程序段：

```
A    {x++;s+=x;}                          (最大执行次数为1)
B    for(i=1;i<=n;i++)                    (最大执行次数为n)
     {x++;s+=x;}
C    ①   for(i=1;i<=n;i++)                (最大执行次数为n²)
     ②       for(j=1;j<=n;j++)
     ③           {x++;s+=x;}
```

对于情况C，虽然只是一个语句，但是可以把它分为若干个子句。这样，既可使问题得到简化，又不影响分析的结论。其各步执行的次数如下：

行号	次数
①	$n+1$
②	$n\cdot(n+1)$
③	n^2

该语句总执行次数f(n)为：

$$f(n)=2n^2+2n+1 \tag{1.1}$$

显然，执行次数f(n)为n的函数，其中，n为问题的规模，如线性表的长度、多项式的项数、矩阵的阶、图中的顶点数等。算法分析就是要确定f(n)是n的什么函数，进而分析f(n)随n变化的情况和确定f(n)的数量级（Order of magnitude）。

一般认为，随着n的增大，函数f(n)增长速度较慢的算法为优。同时用符号“o”来表示数量级的概念，则式（1.1）可表示为：

$$f(n)=o(n^2)$$

该式的含义是：在n较大时，该算法的执行时间和n^2成正比，或者说f(n)数量级和n^2的数量级相同。f(n)称为算法的时间复杂度（Time complexity）。

根据以上分析，当给定一个算法时，总可以对其中每个语句的执行次数进行分析，并

求出它的总和。这样便可得到一个多项式：

$$f(n)=c_k n^k+c_{k-1}n^{k-1}+\cdots+c_1 n+c_0$$

其中，c_i 为常数，$c_k\neq 0$，n 为一个参数，各项代表着不同的数量级。若取其最高数量级作为算法的数量级，即把算法中具有最大执行次数的语句的数量级作为算法的数量级，则有：

$$f(n)=o(n^k)$$

程序段 A 和 B 的时间复杂度分别为 $o(1)$ 和 $o(n)$。

数量级 $o(1)$、$o(n)$、$o(n^2)$ 和 $o(n^3)$ 分别称为常量阶、线性阶、平方阶和立方阶，算法出现的数量级还可能有对数阶 $o(\log_2 n)$ 和指数阶 $o(2^n)$ 等。若一个算法执行的时间为 $o(\log_2 n)$，则当 n 充分大时，它比执行时间为 $o(n)$ 的算法速度要快得多。同样 $o(n\log_2 n)$ 比 $o(n^2)$ 要佳，但不如 $o(n)$。这 7 个数量级的增长率由小到大的排列顺序为：

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

算法 1.1　两个 n 阶矩阵的乘法 C = A · B 的算法如下：

```
void maxtrixmult(int n,float a[ ],float b[ ],float c[ ])
{
    int i,j,k;
    float x;
    for(i=1;i<=n;i++)                          ①
    {
        for(j=1;j<=n;j++)                      ②
        {
            x=0;                               ③
            for(k=1;k<=n;k++)                  ④
                x+=a[i][k]*b[k][j];            ⑤
            c[i][j]=x;                         ⑥
        }
    }
}//maxtrixmult
```

算法中各语句的执行频度分别是：

① $n+1$

② $n(n+1)$

③ n^2

④ $n^2(n+1)$

⑤ n^3

⑥ n^2

则算法的时间复杂度为所有语句执行频度之和 $t(n)=2n^3+3n^2+2n+1$，故时间复杂度为 $o(n^3)$。

算法 1.2　判断任意整数 n 是否为素数的算法如下：

```
void prime(int n)
```