天津市高等学校计算机基础课程规划教材

软件技术基础

方大寿 主编　边奠英 主审

天津大学出版社
TIANJIN UNIVERSITY PRESS

天津市高等学校计算机基础课程规划教材

软件技术基础

方大寿　主编　　边奠英　主审
熊聪聪　韩其睿　宋志卿　编著

天津大学出版社
Tianjin University Press

内容提要

本书系统地介绍了软件技术的基础知识。内容包括数据结构、操作系统、数据库技术基础和软件开发技术等。每篇有练习题。

本书注重概念，从实用角度出发，突出重点，深入浅出，通俗易懂。各个部分相对独立，自成体系，教师可根据专业需要和学时数调整讲课次序或筛选教学内容。

本书可作为高等学校非计算机专业的理、工科本科和研究生的教材，也可作为科技工作者的参考用书。

图书在版编目(CIP)数据

软件技术基础/方大寿主编.—天津：天津大学出版社，2005.2

ISBN 7-5618-2092-5

Ⅰ.软… Ⅱ.方… Ⅲ.软件-基本知识 Ⅳ.TP31

中国版本图书馆 CIP 数据核字(2005)第 006390 号

出版发行 天津大学出版社
出 版 人 杨风和
地 址 天津市卫津路 92 号天津大学内(邮编：300072)
网 址 www.tjup.com
电 话 发行部：022-27403647 邮购部：022-27402742
印 刷 天津新华印刷三厂
经 销 全国各地新华书店
开 本 185mm × 260mm
印 张 18.5
字 数 462 千
版 次 2005 年 2 月第 1 版
印 次 2005 年 2 月第 1 次
印 数 1 - 3 000
定 价 24.00 元

天津市高等学校计算机基础课程规划教材
编写委员会

序　言

中国要振兴，归根到底要靠我们中国人自己努力奋斗，要靠我们的全体劳动者创造出数十倍于今天的劳动生产率。这是一个全体国民素质提高的过程，人们必然要寄希望于教育。教育，特别是学校教育，是为几年乃至几十年之后的社会需求培养人才，所以教育必须面向未来。

要搞好教育，有许多事情要做，其中一条就是教材建设。面对已经到来的信息社会，学校课程到底应该让学生具备哪些基本素质，实现哪些发展，这是当前课程改革的一个重要问题。20世纪末国家提出的课程计划明确了“促进学生个性健康发展”的目标，重视认知与情感的统一、知识与能力的统一、主体精神与社会责任的统一，强调学生的素质发展，强调学生的探索创新能力、实践能力、学习能力和信息素养。

为适应课程目标的变化，需要重新审视课程内容，要删除陈旧过时的内容，吸收科学和文明发展的新成果，反映本学科最新发展动态。

要编写出课程内容具有科学性、系统性和先进性，符合本课程内在逻辑体系和学生认知规律，表达形式符合国家有关规范标准的教材，不是一件易事。为此，我们邀请了本市各高校长期从事计算机基础课教学的教师组成写作班子。这些老师们认真总结了“九五”规划教材的编写经验，反复讨论新制定的教学大纲，把课程内容有机地组合起来，把基本概念、基本原理和基本技能提炼出来，形成一个具有逻辑性、系统性的知识系统，使之有利于学生对知识的理解与迁移。

这套教材的出版，旨在推动我市高校计算机基础教育活动，提高大学生计算机基础知识水平和应用能力。我们殷切地希望广大学生、教师和专家提出宝贵意见，以便再版时修改补充。

这套教材在编写出版过程中，得到了各方人士的大力支持和帮助，特别是天津大学出版社始终给予积极配合。在此，我们一并表示衷心的感谢。

天津市普通高等学校计算机基础课程教学指导委员会

2004年5月

前　言

《软件技术基础》是为非计算机专业学生学习软件基础知识编写的一本综合性教材，也可作为计算机技术人员和科技工作者的培训教材或参考书。学习《软件技术基础》需要具有一定的计算机基础知识，也应初步具备高级语言编程的能力。

软件方面课程内容繁多，根据近年来教学需要，本书从实用角度出发选编了数据结构、操作系统、数据库技术基础和软件开发技术四部分。考虑到网络技术发展较快，不少学校独立开设计算机网络等课程，因此未将其编入本书。本书在编写上力求扼要、结合实际、通俗易懂。书中各部分相对独立、自成体系，可根据专业需要和学时的多少任意调整讲课次序，筛选教学内容。

计算机科学是一门实践性、操作性很强的学科，加强上机实践是学好本课程的重要环节。本书每章附有适量习题，部分章节附有上机程序，可供教学时参考使用。

本书第一篇由熊聪聪编写；第二篇、第四篇由韩其睿编写；第三篇由宋志卿编写。全书由方大寿、熊聪聪统稿，由边奠英教授主审。

在本书的编写和出版过程中，得到了天津市教委的大力支持，天津大学出版社的领导和编辑也付出了大量的辛勤劳动，在此一并表示由衷的感谢。

由于时间仓促，经验不足，书中错误不当之处，请批评指正。

编著者　2004.5

目　录

第一篇　数据结构

第二篇 操作系统

第三篇 数据库技术基础

第四篇 软件开发技术

第一篇　数据结构

第1章　基本概念和相关术语

1.1　什么是数据结构

“数据结构”是计算机学科中一门重要的基础课程。在计算机科学领域中，尤其是涉及系统软件和应用软件的设计和实现中，都肯定要涉及各种数据结构。学习“数据结构”既为进一步学习其他计算机软件课程提供了必要的准备知识，同时又有助于提高软件设计和程序编制水平。

计算机程序主要是对信息进行加工处理。在很多情况下，这些信息并不是没有组织的，信息（数据）之间往往具有这样那样的关系，而且主要是结构关系，这就是数据结构的内容。那么，什么是数据结构呢？

先看一个书店图书查询系统的例子。

有一个书店要完成一个图书的库存查询系统，要建立并且维护一个图书目录。此目录记录了书店中库存图书的名称和数量，假定按如下形式安排：

$$(a_1,b_1)(a_2,b_2)\cdots(a_n,b_n)$$

其中，a_i、$b_i(i=1,2,\cdots,n)$分别表示图书的名字和对应的库存数量。要设计一个算法，当给定任何一本书的名字时，该算法能够打印出此图书的库存数量。如果书店中根本就没有这本书，则该算法也能够报告没有这本书的信息。

那么，这个图书目录应该如何组织，如何用计算机程序实现，如何设计查找算法，这些都是数据结构要讨论的课题。

1.1.1　数据结构的定义

数据结构是在整个计算机科学与技术领域上广泛被使用的术语。它用来反映一个数据的内部构成，即一个数据由哪些成分构成，以什么方式构成，呈现什么样的结构。

数据结构分为逻辑上的数据结构和物理上的数据结构。逻辑上的数据结构反映成分数据之间的逻辑关系，而物理上的数据结构反映成分数据在计算机内部的存储安排。所以，简单地说，数据结构是数据存在的形式。

或者可以这样说：数据结构研究的是数据逻辑结构和物理结构以及它们之间的相互关系，并对这种结构定义相应的运算，而且确保经过这些运算后所得到的新结构仍然是原来的结构类型。

因此,数据结构是信息的一种组织方式,目的是为了提高算法的效率。它通常与一组算法的集合相对应,通过这组算法集合可以对数据结构中的数据进行某种操作。

1.1.2 数据结构主要研究内容

作为一门学科,数据结构主要研究数据的各种逻辑结构和存储结构以及对数据的各种操作。因此,数据结构主要研究三方面的内容:

①数据的逻辑结构;

②数据的物理结构;

③对数据允许进行的操作(或算法),也称数据处理或数据运算。

通常,算法的设计取决于数据的逻辑结构,算法的具体实现取决于数据的物理存储结构。

1.2 基本名词术语

1.2.1 数据

数据(Data)是人们利用文字符号、数字符号以及其他一些规定的符号对现实世界的事物及其活动所做的描述。因此,可以把一本小说、一篇论文、一张图表甚至一个句子、一个单词、一个算式以至一个数值、一个字符等都可以当成是数据。在计算机领域,人们把能够被计算机加工的对象,或者说能够被计算机输入、存储、处理和输出的一切信息都叫做数据。

1.2.2 ·数据元素

数据元素(Data Element)是一个数据整体中相对独立的单位。如对于一个文件来说,每个记录就是它的数据元素;对于一个字符串来说,每个字符就是它的数据元素;对于一个数组来说,每一个数组元素就是它的数据元素。

数据和数据元素是相对而言的。如对于一个数据库中的记录来说,它相对于所在的数据表文件被认为是数据元素,而它相对于其中的每个属性又被认为是数据。

1.2.3 数据记录

数据记录(Data Record)简称记录,是数据处理领域组织数据的基本单位。它又是由更小的单位——数据项(Item)所组成。一个数据记录一般包括一个或若干个固定的数据项(当然每一个数据项还可以是记录的形式)。就拿对图书目录管理来说,每个记录表示一本图书的目录信息,如表 1.1.1 所示。

表 1.1.1 图书的目录信息表

书号	书名	价格	库存数量
001	操作系统	20.00	35
002	汇编语言	25.00	46
003	数据结构	30.00	55
004	接口技术	18.00	27

1.2.4 数据对象

数据对象(Data Object)是指性质相同的数据元素的集合,是数据的一个子集。数据对象

可以是有限的，也可以是无限的。

1.2.5 数据结构

前面已经介绍过数据结构(Data Structure)的概念，这里，分别给出它的简单定义和二元组定义。

1.简单定义

数据结构是相互之间存在一种或多种特定关系的数据元素的集合。

2.二元组定义

为了更确切地描述数据结构，通常采用二元组表示：

$G=(D,R)$

G 是一种数据的逻辑结构。它由数据元素的集合 D 和 D 上二元关系的集合 R 所组成。其中

$$D=\{d_i \mid 1\leqslant i\leqslant n, n\geqslant 0\}$$

$$R=\{r_j \mid 1\leqslant j\leqslant m, m\geqslant 1\}$$

d_i 表示第 i 个数据元素。n 为 G 中数据元素的个数。若 $n=0$，则 D 是一个空集，因而也就无结构可言，是数据结构中的一种特殊情况。r_j 表示第 j 个关系，m 为 R 上关系的个数。

在本篇所讨论的数据结构中，一般只讨论 $m=1$ 的情况，即 R 中只包含一个关系($R=\{r\}$)的情况。

1.2.6 逻辑结构和物理结构

被计算机加工的数据元素不是互相孤立的，它们彼此间一般存在着某些逻辑上的联系，这些联系在对数据进行存储和加工时反映出来。数据结构主要指逻辑结构和物理结构。

1.数据的逻辑结构

数据的逻辑结构只抽象地反映数据元素间的逻辑关系，而不管其在计算机中的存储表示方式。

数据的逻辑结构分为线性结构和非线性结构。若各数据元素之间的逻辑关系可以用一个线性序列简单地表示出来，则称为线性结构，否则称为非线性结构。线性表是典型的线性结构，而树、图等都是非线性结构。

如果分得更细的话，数据的逻辑结构通常分为以下四类基本结构。

1)集合　　结构中的数据元素除了同属于一种类型外，别无其他关系。

2)线性结构　　结构中的数据元素之间存在一对一的关系。

3)树形结构　　结构中的数据元素之间存在一对多的关系。

4)图状结构或网状结构　　结构中的数据元素之间存在多对多的关系。

2.数据的物理结构

数据的物理结构是逻辑结构在计算机存储器里的实现。为全面地表示一个逻辑结构，它在存储器中的存储应包括数据元素自身值的表示和数据元素之间关系的表示两个方面。

1.2.7 数据处理

数据处理(Data Processing)也称数据运算，是指对数据进行查找、插入、删除、合并、排序、统计、简单计算等操作过程。

早期，计算机主要用于科学和工程计算；现阶段，计算机更多地被用于数据处理方面。随着时间的推移和计算机应用的进一步普及，计算机用于数据处理方面的比例必将进一步

增大。

数据运算是定义在数据的逻辑结构上的,但运算的具体实现要在存储结构上进行。数据的各种逻辑结构有相对应的各种运算,每种逻辑结构都有一个运算的集合。常用的运算有检索、插入、删除、更新、排序等。

数据运算是数据结构中的一个重要方面,讨论任何一种数据结构时都离不开对该结构上的数据运算及实现算法的讨论。

1.2.8 数据存储方式

有很多种不同的方式可以实现数据的逻辑结构在计算机存储器中的存储。下面介绍两种最主要的存储结构——顺序存储和链式存储。大多数数据结构的存储表示都会采用其中的一种结构,或两种结构的结合。

1.顺序存储结构

顺序存储结构主要应用于线性的数据结构,它把逻辑上相邻的数据元素存储在物理上相邻的存储单元里。节点之间的关系由存储单元的邻接关系实现。

顺序存储结构的主要特点是:

①数据节点中只有自身信息域(即只存储数据的值),没有连接信息域,因此存储密度大,存储空间利用率高;

②可以通过计算直接确定数据结构中第 i 个数据节点的存储地址 L_i,计算公式为 $L_i = L_0 + (i-1) \times m$,其中 L_0 为第一个节点的存储地址,m 为每个数据节点所占用的存储单元个数;

③插入、删除运算不便,会引起大量数据节点的移动。

2.链式存储结构

链式存储结构就是在每个节点中至少包括一个指针域,用指针来表现数据元素之间逻辑上的联系。这种存储结构把人们从计算机存储单元的连续性限制中解放出来。它可以把逻辑上相邻的两个元素存放在物理上不相邻的存储单元中;还可以在线性编址的计算机存储器中表示数据节点之间的非线性联系。

链式存储结构的主要特点是:

①数据节点中除自身信息外,还有表示连接信息的指针域(即不仅存储数据的值还同时存储数据的联系),因此比顺序存储结构的存储密度小,存储空间利用率低;

②逻辑上相邻的节点物理上不必邻接,可用于线性表、树、图等多种逻辑结构的存储表示;

③插入、删除操作灵活方便,不必移动数据节点,只要改变数据节点中的指针值即可。

除上述两种主要存储方式外,索引存储、散列存储也是在线性表和集合的存储表示中常用的两种重要存储方式,将在后面介绍。

1.2.9 数据类型

数据类型(Data Type)是指在一种程序设计语言中变量所具有的数据种类。数据类型是某一种语言中具体实现数据结构必须掌握的知识。

习　题

一、名词解释

1.数据

2.数据结构

3.逻辑结构

4.存储结构

5.线性结构

6.非线性结构

二、填空题

1.数据逻辑结构包括_____、_____、_____和_____。

2.数据的存储结构包括_____、_____、_____和_____。

3.数据结构主要研究数据的_____、_____、和_____。

4.线性结构中的元素之间存在_____的关系，树形结构中的元素之间存在____的关系，图形结构的元素之间存在_____的关系。

三、单项选择

1.数据结构通常是研究数据的(　　)及它们之间的相互关系。

A)存储结构和逻辑结构　　B)存储和映像

C)联系和抽象　　D)联系与逻辑

2.数据结构中，在逻辑上可把数据结构分成(　　)。

A)动态结构和静态结构　　B)紧凑结构和非紧凑结构

C)线性结构和非线性结构　　D)内部结构和外部结构

3.数据在计算机存储器内表示时，物理地址和逻辑地址相同并且是连续的，称之为(　　)。

A)存储结构　　B)逻辑结构多对多关系

C)顺序存储结构　　D)链式存储结构

4.非线性结构的数据元素之间存在(　　)。

A)一对一关系　　B)多对多关系

C)一对多关系　　D)B或C

第2章　算法及相关知识

2.1　算法

2.1.1　算法的定义

算法是在有限步骤内求解某一问题使用的一组定义明确的规则。通俗地说，就是计算机解题的过程。在这个过程中，无论是形成解题思路还是编写程序，都是在实施某种算法。只不过前者是推理实现的算法，后者是操作实现的算法。简单地说，算法就是解决特定问题的方法。

2.1.2　算法的特性

一个算法应该具有以下五个重要的特征。

1)有穷性　一个算法必须保证执行有限步之后结束。

2)确切性　算法的每一步骤必须有确切的定义。

3)输入　一个算法有零个或多个输入，以刻画运算对象的初始情况。所谓零个输入是指没有输入，也就是算法本身定制了初始条件。

4)输出　一个算法必须有一个或多个输出，以反映输入数据加工后的结果。没有输出的算法是毫无意义的。

5)可行性　算法原则上能够精确运行，而且人们用笔和纸做有限次运算后即可完成。

2.1.3　算法的评价

对于解决同一个问题，往往能够编写出许多不同的算法。例如，对于排序问题，在后面第6章中就将介绍多种算法。

进行算法评价的目的，就是要从解决同一问题的不同算法中选择出较为合适的一种，同时也能够知道如何对现有的算法进行改进，从而设计出更好的算法。

评价一个好的算法有以下几个标准。

1)正确性　算法应满足具体问题的需求。正确性是设计和评价一个算法的首要条件，如果一个算法不正确，其他方面就无从谈起。一个正确的算法是指在合理的数据输入下，能在有限的运行时间内得出正确的结果。通过对数据输入所有可能情况的分析和上机调试，可以证明算法是否正确。

2)可读性　算法应该易懂、好读，便于其他人对实现算法的程序的理解。

3)简单性　最简单和最直接的算法往往不是最有效的，但算法的简单性使得证明其正确性比较容易，同时便于编写、修改、阅读和调试。所以算法的简单性还是应当强调和不容忽视的。不过对于那些需要经常使用的算法来说，高效率(即尽量减少运行时间和压缩存储空间)比简单性更为重要。

4)健壮性　算法应该具有容错处理。当输入非法数据时，算法应对其作出反应，而不是产生莫名其妙的输出结果。

5)算法的效率　效率指的是算法执行的时间,同时也包括存储量需求(指算法执行过程中所需要的最大存储空间),或叫时间/空间复杂度。一般,这两者与问题的规模有关。

上面简单讨论了如何从五个方面来评价一个算法的问题。这里还需要指出,除了算法的正确性之外,其余四个方面往往是相互矛盾的。例如,当追求较短的运行时间时,可能带来占用较多的存储空间和较繁琐的算法;当追求占用较少的存储空间时,可能带来较长的运行时间和较繁琐的算法;当追求算法的简单性时,可能需要较长的运行时间和占用较多的存储空间;追求可读性好时,往往不能保证算法的简单性。所以在设计一个算法时,要从这五个方面综合考虑,此外还要考虑到算法的使用频率、算法的结构化以及所使用机器的硬软件环境等因素,才能设计出比较好的算法。

2.2　算法的效率度量

算法的复杂度是算法效率度量的标准,是评价算法优劣的重要依据。一个算法复杂度的高低,体现在运行该算法所需要计算机资源的多少上面。所需的资源越多,该算法的复杂度越高;反之,所需的资源越少,则该算法的复杂度越低。

所谓计算机的资源,最重要的是时间(CPU 的运行时间)和空间(存储器的占用)资源。因而,算法的复杂度就要从算法的运行时间(时间复杂度)和占用的存储空间(空间复杂度)来描述。

2.2.1　运行时间

运行时间是指一个算法在计算机上运算所花费的时间。它大致等于计算机执行一种简单操作(如赋值操作、转向操作、比较操作等)所需的时间与算法中进行简单操作次数的乘积。因为执行一种简单操作所需的时间随机器而异,它是由机器本身硬软件环境决定的,与算法无关。所以这里只讨论影响运行时间的另一个因素——算法中进行简单操作的次数。

显然,在一个算法中,进行简单操作的次数越少,运行时间也就相对越少;次数越多,运行时间也就相对越多。因此,通常把算法中包含简单操作次数的多少叫做算法的时间复杂度,它是一个算法运行时间的相对度量单位。

若解决一个问题的规模为 n,例如在排序问题中,n 表示待排元素的个数;在矩阵求逆中,n 表示矩阵的阶数;在图的遍历中,n 表示图的顶点数或边数等,那么,算法的时间复杂度就是 n 的一个函数,通常记为 $T(n)$。

下面通过例子来分析算法的时间复杂度。

【例 2.1】　累加求和。

```
float sum(A,n)                    //A 表示一维实型数组,n 表示数组的大小
{
  (1)s=0;                         //给累加变量 s 赋初值
  (2)for(i=0;i<n;i++)             //进行累加求和
    s=s+A[i];
  (3)return s;                    //把 s 值(即累加和)作为函数的值传回去
}
```

计算机执行这个算法时,第(1)步和第(3)步都只需一次赋值操作,为了分析第(2)步包