普通高等教育“十一五”国家级规划教材

软件技术基础

RUANJIAN JISHU JICHU [第三版]

黄迪明 主编

电子科技大学出版社

普通高等教育“十一五”国家级规划教材

软件技术基础

[第三版]

黄迪明　主编

李玉柏　许家珆　黄迪明　廖建明　蔡洪斌　编著

电子科技大学出版社

图书在版编目（CIP）数据

软件技术基础 / 黄迪明主编. —3 版. —成都：电子科技大学出版社，2009.7
普通高等教育“十一五”国家级规划教材
ISBN 978-7-5647-0172-7

Ⅰ.软... Ⅱ.黄... Ⅲ.软件—高等学校—教材 Ⅳ.TP31

中国版本图书馆 CIP 数据核字（2009）第 077637 号

普通高等教育“十一五”国家级规划教材

软件技术基础

[第三版]

黄迪明 主编

李玉柏 许家 黄迪明 廖建明 蔡洪斌 编著

出　　版：电子科技大学出版社（成都市一环路东一段 159 号电子信息产业大厦 邮编：610051）
策划编辑：吴艳玲
责任编辑：吴艳玲
主　　页：www.uestcp.com.cn
电子邮箱：uestcp@uestcp.com.cn
发　　行：新华书店经销
印　　刷：成都中铁二局永经堂印务有限责任公司
成品尺寸：185mm×260mm　　印张 23.5　　字数 571 千字
版　　次：2009 年 7 月第一版
印　　次：2009 年 7 月第一次印刷
书　　号：ISBN 978-7-5647-0172-7
定　　价：32.90 元

内 容 提 要

本书为高校计算机基础教育第二层次的教材，是第一层次“计算机应用基础”和“程序设计语言”的后继课程用书。本书的第一版是国家电子信息类规划教材，本书是普通高等教育“十一五”国家级规划教材。全书共五章，主要内容包括：数据结构、操作系统、软件工程方法、数据库技术、网络技术基础。每章之后有小结和习题。全书内容紧凑、翔实，简明扼要，深入浅出，注重实用。

本书内容是按国家教育部高教司颁发的“工科非计算机专业计算机基础教学指南”中有关软件技术基础课程的教学要求编写的。本书可作为工科非计算机专业大学本科生、研究生教材，也可作为应用软件人员的培训教材或工程技术人员的参考教材。

前　言

本书为高校计算机基础教育第二层次的教材，是第一层次的后继课程用书。本书的第一版是国家电子信息类规划教材，第三版是普通高等教育“十一五”国家级规划教材。本教材由电子科技大学黄迪明主编。

《软件技术基础》是为非计算机专业的大学生和研究生学习计算机软件基础知识而编写的一本综合性教材。本教材内容符合国家教育部高教司颁发的“工科非计算机专业计算机基础教学指南”中有关软件技术基础课程的教学要求。

作为“计算机应用基础”和“程序设计语言”的后继课程，本教材以应用为目的，从计算机软件课程中选取了数据结构、操作系统、软件工程方法、数据库技术、网络技术基础五部分内容，简明扼要地介绍了计算机软件中的一些重要概念，以及软件技术的基础知识和方法，以培养学生利用计算机解决问题的意识和能力，为计算机在各专业中的应用奠定基础。

本教材的参考学时数为68学时(含上机)。各部分内容相对独立，自成体系，讲授时可根据专业需要及教学对象，酌情取舍(有的内容可供自学)。各部分内容之后附有小结和习题。

本教材由李玉柏编写第一章，许家珀编写第二章，黄迪明编写第三章，蔡洪斌编写第四章，廖建明编写第五章。全书由黄迪明统筹并修改定稿。电子科技大学王正智教授、兰家隆教授、俞永康教授对本书的编写提出了各种有益的建议，电子科技大学教务处及出版社对本书的出版给予了大力的支持，在此表示诚挚的感谢。由于编者水平有限，书中难免还存在一些缺点和错误，殷切希望广大读者批评指正。

编　者

2009年7月于电子科技大学

目　录

第1章 数 据 结 构

随着计算机应用领域的不断扩大，大数据量的数值运算和非数值数据的处理变得尤为重要，这些数据的元素之间大多是相互有关的，它们的关系决定了对它们进行处理的方法，以及最后程序实现的方式。因此，讨论数据元素之间的逻辑关系、数据元素在计算机中的存储方式及在数据元素集合上设立的运算如何实现，这是研究数据处理的前提，是算法设计的基础，是程序编写的组成部分。本章首先介绍数据结构的相关概念，着重讨论线性结构、树型结构和图形结构三类数据结构，最后介绍数据结构中常用的查找和排序等基本运算。

1.1 数据结构的基本概念

1.1.1 什么是数据结构

早期的计算机多用于进行简单的数值运算，输入和输出的数据量不大，数据元素间的关系较为简单。但随着计算机应用的扩展，目前计算机被更多地用于大数据量的数值运算和非数值处理，如矩阵运算、管理与控制操作，合理安排数据元素之间的关系直接影响计算机运算的效率和使用的存储空间大小。正是如此，才产生了一门重要的计算机课程——数据结构。数据结构已成为学习和设计计算机系统软件、应用程序必不可少的基础知识。

顾名思义，数据结构是讨论计算机系统中数据的组织形式及其相互关系。在计算机系统中，数据不仅包含了通常数值的概念，还有更广泛的含义。它把客观事物采用计算机进行识别、存储和加工所进行的描述，统称为数据。例如，十进制、二进制数；字母、字符；程序段、图形图像、语言等数据信息。数据的基本单位为数据元素（有些书上直接称为数据节点）。

结构是指事物间的相互关系和约束。以一定存储方式存储在计算机系统中的数据元素，其排列并非杂乱无章，而是具有某种组织形式，包括数据元素内在的逻辑关系和存储形式上外在关系。数据元素的相互关系最终反映在对数据元素的一些运算和操作上。因此，研究数据结构，就是要研究以下三方面的内容：

（1）数据元素之间的逻辑关系是什么？

（2）适宜选用什么样的存储结构？

（3）采用什么样的操作实现算法效率更高？

为了增加对数据结构的感性认识，下面举例来具体说明上述概念。

例1 学生成绩表为：

学号	姓名	数学	物理	电路基础	C语言
200701101	丁一	87	90	86	84
200701102	马二	80	81	86	90
200701103	张三	91	88	90	87
200701105	李四	93	88	92	85
200701106	王五	67	66	70	65
⋮	⋮	⋮	⋮	⋮	⋮

这张学生成绩表就可称为一个数据结构，表中的每一行为一个数据元素。它是由学号、姓名、各科成绩等数据项组成。该表以学生学号顺序说明数据元素之间的相互关系，是一个线性表结构：对表中任一元素，与它相邻且在它前面的数据元素（亦称为直接前趋）最多只有一个；与表中任一数据元素相邻且在它后面的数据元素（亦称直接后继）也最多只有一个。表中只有第一个元素没有直接前趋，故称为开始数据元素。同时，也只有最后一个元素没有直接后继，故称之为终点数据元素。例如，表中“马二”所在元素的直接前趋和直接后继元素分别是“丁一”和“张三”所在的数据元素。上述元素间的学号顺序关系构成了这张学生成绩表的逻辑关系。

这张学生成绩表的存储方式则是指在计算机存储器中如何表示数据元素之间的这种关系，即表中的数据元素是顺序地邻接存储在一片连续的单元之中，还是用指针将这些元素链接在一起？在这张表中，可能要经常查看某一学生的成绩，当学生退学时要删除相应元素，当招收新学生时要添加新的数据元素，等等。那么，究竟要怎样进行查找、删除、插入，这就是数据高效操作所涉及的问题。搞清楚了上述三个问题也就弄清了学生成绩线性表这种数据结构。

数据结构的基本单位是数据元素，数据元素的类型可以是基本类型，如 int 整型、float 实数型，也可以是构造类型。在数据结构的讨论中，往往并不注重数据元素的类型，而是把它看成一个节点，着重讨论节点之间的关系。在上述学生成绩表中，数据元素的类型可用一个结构类型表示：

```
typedef struct student
    { int id;
      char name [20];
      float score [4];
    } elemtype;
```

这样就可以用 elemtype 来表示抽象了的数据元素类型。

1.1.2 数据结构中的基本概念

通常把运用数据结构来描述的数据元素之间的逻辑关系、数据在计算机系统中的存储方式和数据的运算抽象成数据结构的三个层次：数据的逻辑结构、数据的存储结构和数据操作集合。

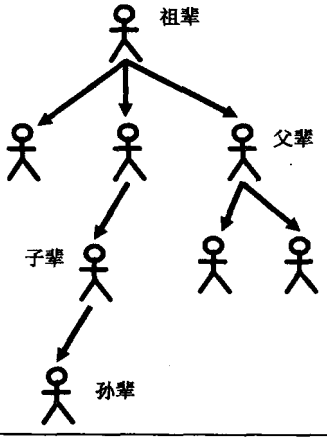
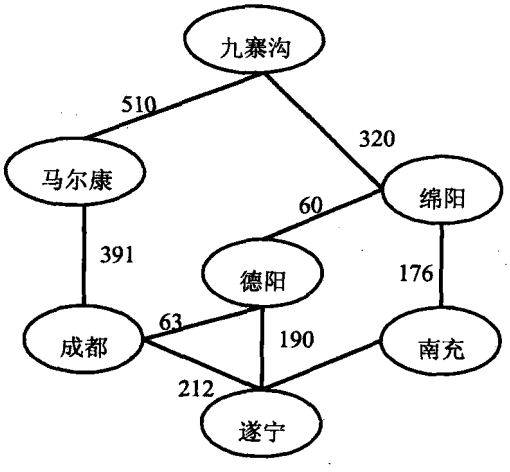
反映数据元素之间关系的数据逻辑结构可分为两大类：

（1）线性结构：线性结构的逻辑特征是，有且仅有一个开始数据元素和一个终点数据

元素，并且所有数据元素都最多只有一个直接前趋和一个直接后继。线性表就是一个典型的线性结构。

（2）非线性结构：非线性结构的逻辑特征是，该结构中一个数据元素可能有多个直接前趋和直接后继。非线性结构中最一般的结构是图结构，在图结构中，对任何数据元素的直接前趋和直接后继的个数都不作限制。在非线性结构中有一类较特殊的结构，我们称为树结构，它的逻辑特征是，有且仅有一个称为根的元素无直接前趋，其他元素有且仅有一个直接前趋，所有数据元素（除根元素外）都存在一条从根元素到该元素的路径。

根据逻辑结构对数据结构的分类可以总结如下：

<table>
<tr><th colspan="2">分类</th><th>逻辑关系特征</th><th>实例</th></tr>
<tr><td>线性结构</td><td>线性表</td><td>● 元素间为严格的一对一关系
● 元素最多只有一个直接前趋和直接后继</td><td>例 1 中的学生成绩表；</td></tr>
<tr><td rowspan="2">非线性结构</td><td>树结构</td><td>● 元素间为严格的一对多关系
● 元素最多只有一个直接前趋，但部分元素有多于一个的直接后继</td><td>比如组织关系和家族关系：
 
</td></tr>
<tr><td>图结构</td><td>● 元素间为多对多关系
● 元素有多于一个的直接前趋和直接后继</td><td>比如电信网络和公路交通：
 
</td></tr>
</table>

而反映数据元素在计算机中的存储方法就是数据的存储结构。数据的存储结构，有时也称为数据的物理结构，它是数据的逻辑结构在存储器里的实现。

数据的存储方法可分为如下四类：

（1）顺序存储方法

该方法是把逻辑上相邻的数据元素存储在物理位置上相邻的存储单元里，元素间的逻辑关系由存储单元的邻接关系体现。由此得到的存储表示称为顺序存储结构。顺序存储方法主要应用于线性的数据结构，如线性表、数组等。非线性的数据结构也可以通过某种线性化的方法来实现顺序存储。

（2）链接存储方法

该方法不要求逻辑上相邻的元素其物理位置上亦相邻，元素间的逻辑关系是由附加的指针字段表示的。由此得到的存储表示称为链式存储结构。链式存储结构要借助于程序语言的指针类型来描述元素的存储地址，即在此存储方法中，每个数据元素所占存储单元分成两部分：一部分为元素本身数据项；而另一部分为指针项，指出其后继或前趋元素的存储地址，从而形成一个链。

（3）索引存储方法

该方法通常是在存储元素信息的同时，还建立附加的索引表，索引表中的每一项称为索引项，索引项的一般形式是：（关键字、地址）。关键字是能唯一标识一个元素的数据项。若每个元素在索引表中都有一个索引项，则该索引表称为稠密索引（Dense Index）；若一组元素在索引表中只对应一个索引项，则该索引表称之为稀疏索引（Sparse Index）。稠密索引中索引项的地址指示元素所在的存储位置，而稀疏索引中索引项的地址则指示一组元素的起始存储位置。

（4）散列存储方法

该方法的基本思想是根据元素的关键字直接计算出该元素的存储地址。即在数据元素的字段中有一个或几个字段的值，通过某一散列函数唯一地确定该元素的存储地址。有时又称散列存储方法为“关键字一地址”转移法。

把数据以一定的逻辑结构组织起来，并以适当的方法存储在计算机系统的存储器里，其最终目的是为了有效处理数据，提高数据处理的运算速度。

在数据结构中，要讨论的常用数据处理与运算有下列几种：

（1）遍历：在数据结构的各个元素中移动，或查看所有数据元素。

（2）插入：往数据结构中添加新的元素。

（3）更新：修改或替代数据结构中指定元素的一个或多个数据项（字段值）。

（4）删除：把指定的数据元素从数据结构中去掉。

（5）查找：在数据结构中查找满足一定条件的数据元素。

（6）排序：在保持数据结构中数据元素个数不变的前提下，把元素按指定的顺序重新排列。排序一般是建立在线性逻辑结构的基础上。

值得指出的是，很多教科书上是将数据的逻辑结构和数据的存储结构定义为数据结构，而将数据的运算定义为数据结构上的操作。但是，无论怎样定义数据结构，都应该将数据的逻辑结构、数据的存储结构及数据的运算这三方面看成一个整体，希望读者学习时，不

要孤立地去理解一个方面，而要注意它们之间的联系。

上述四种基本的存储方法也可以组合起来对数据结构进行存储映像。同一种逻辑结构采用不同的存储方法，可以得到不同的存储结构。选择何种存储结构来表示相应的逻辑结构，主要是使其运算方便及根据算法的时空要求来具体确定。

正是因为存储结构是数据结构不可缺少的一个方面，所以我们常常将同一逻辑结构的不同存储结构冠以不同的数据结构名称来标识它们。例如，线性表是一种逻辑结构，若采用顺序方法的存储表示，则称该结构为顺序表；若采用链接方法的存储表示，则称该结构为链表；若采用散列方法的存储表示，则可称其为散列表。

同理，由于数据的运算也是数据结构不可分割的一个方面，所以给定了数据的逻辑结构和存储结构，若定义的运算集合及其运算的性质不同，也可能导致完全不同的数据结构。例如，若对线性表上的插入、删除运算限制仅在表的一端进行，则该线性表称之为栈；而插入限制在表的一端进行，删除限制在表的另一端进行的线性表称为队列。更进一步，若线性表采用顺序表和链表作为存储结构，则对插入和删除运算做了上述限制之后可分别得到顺序栈和链栈、顺序队列和链队列。

1.1.3 C 语言的数据类型

基于本书是使用 C 语言来进行算法的描述，为了便于读者阅读程序范例，了解各种常见的数据元素类型，在这里我们简要回顾一下 C 语言一些重要的数据类型。

（1）C 语言的基本数据类型

C 语言支持三种基本数据类型：int 型、float 型和 char 型。int 型可以有三个限定词：short、long 和 unsigned，对应短整型、长整型和无符号整型。float 型也有三种类型：float，double 和 longdouble，对应浮点型、双精度浮点型和增长的双精度浮点型。char 型为字符型，只占一个字节的存储空间。

（2）C 语言的指针类型

C 语言允许直接对存放变量的地址进行操作。如定义 int a，b，则&a 表示变量 a 的地址，也称为指向变量 a 的指针。存放地址的变量称为指针变量，指针变量指向的对象称为目标变量。如再定义 int *pa，则语句 pa = &a，表示 pa 为指向目标变量 a 的指针变量，目标变量 a 的值可用*pa 表示，则语句 b = *pa 等效于 b = a。另外，这里 pa 的含义不只是指针，而且是确定的指向整型类型的指针。

C 语言中指针的一个重要作用是在函数间传递数值。对一个形参为简单变量的函数，参数传递的是值传递，即把实参的值拷贝给形参，这样函数内形参的值被修改时实参的值不会跟着变化。若函数采用指针作形参，则参数传递采用地址传递，即把实参的地址传送给形参，或者说实参、形参共用一个存储单元。这时若函数内形参的值改变，则实参中的值将跟着变化。利用指针形参可设计函数的输出参数和变参。

在以后的程序举例中，我们会多次使用指针变量作为函数参数来实现在函数调用时，数据的双向传输。有时，还会使用指向指针的指针变量来作为函数的参数，以实现一个地址（指针）的双向传输。读者应特别留心。

（3）数组类型与字符串

在 C 语言中，数组是同一类型的一组有序数据的集合。数组名标识了这一组数，同时

代表了这一组数的起始地址，也就是说，数组名是一个指针变量。数组元素下标指示了数组元素在该数组中的顺序位置。可以有一维数组和多维数组，例如 int a [100]；定义了一个一维整数数组，下标为 0～99，一共 100 个元素；而 float b [10] [10]；定义一个二维浮点型数组，两个下标分别从 0～9 取值，一共也是 100 个元素。二维数组的两个下标分别称为行下标（第一维）和列下标（第二维）。

两维数组和多维数组的多下标结构，只是一种逻辑上的关系表示，因为，计算机的内存是一维的，只能一个一个元素顺序存放。C 语言中其物理存储结构是行主序存放方法实现的，即第一行的所有元素存完后，再顺序存放第二行的元素，以此类推。

C 语言中没有单独的字符串类型。字符串定义成字符数组。每个字符串由转义符′ \0′ 指示其结束。一个字符串常量由一对双引号指示。一个字符串常量被存入内存，系统自动在其末尾添加字符串结束标志 '\0' 。如 char name[20]；说明了一个 char 型的一维数组 name，其数据范围是 20 个字符。语句 name="Li Si"；则是对 name 赋一个字符串常量，字符串长为 5，在其末尾还有一个串结束标志 '\0'。

C 语言子程序函数提供有关字符串操作的各种函数，如字符串长度函数 strlen（name），字符串的拷贝函数 strcpy（str1，str2），字符串的连接函数 strcat（str1，str2），等等，读者可参考有关 C 语言的资料。

（4）C 语言的结构类型

结构类型由一组称为结构成员的项组成，每个结构成员都有自己的标识符。在前面定义 elemtype 类型时，就使用了结构类型的定义。

在 C 语言中定义一个结构变量由两步组成：一是定义结构类型；二是利用该类型来说明结构类型变量，如：

```
elemtype list [100];
```

说明了每个元素为类型 elemtype 的一个数组 list。

除了以上所列举的数据类型，C 语言还支持诸如枚举类型、联合类型等其他类型，这些类型的应用相对较少，这里就不一一介绍，读者可参考有关 C 语言的资料。

1.1.4 算法的基本概念与算法效率

最初算法这个术语是对一种人类思维方式的定义，即从具体的操作规范入手，通过操作过程的构造与实施来解决给定问题的思维方法。这表明，算法既可以用来描述人解决给定问题的操作过程，也可以用来描述计算机解决给定问题的操作过程，后者正是我们编写程序的基础，是计算科学意义下的算法。一般地讲，计算机中科学的算法，是指为解决给定问题的有穷操作规则的有序集合。

一个算法必须具备有穷性、确定性、数据输入、信息输出以及可行性五项基本特征。

（1）有穷性：有始有终是算法最基本的特征。有始无终的过程决不是算法。换言之，一个算法必须在它所涉及的每一种情形下，都能在执行有穷步操作之后而告结束。

（2）确定性：算法的每一步操作，其顺序和内容都必须唯一地确定，不得有任何模糊。如果一个过程的某操作步骤有“既可这样，又可那样”的非确定性，那么它不是算法。

（3）数据输入：一个算法有零个或多个从约定对象集合中取出的原始数据输入。

（4）信息输出：既然算法是用来解决给定问题的，所以一个算法必须将人们关心的信息输出来。也就是说，一个算法至少有一个已获得的有效信息输出。

（5）可行性：一个算法的任何一步操作都必须是可以付诸实施并能完成的基本操作。换言之，各步骤不仅在理论上而且在实践上均能由人们用笔和纸做有穷次运算来完成。

在设计和构造算法时，要确保所得算法真正同时具备这五项基本特征，缺一不可。既然算法必须解决给定问题，算法的处理对象就必然是所论问题的有关数据。算法与数据结构之间存在密切关系，算法是建立在数据结构基础之上的。没有确定对数据进行何种操作，就无法决定如何构造数据结构。同样，确定算法也依赖于作为基础的数据结构。只有明确了问题求解的算法，才能够较好地设计数据结构，但是要设计好的算法，又常常依赖于合理的数据结构，数据结构是设计算法的基础。

对于一些复杂的问题，常常因为数据结构的差异，问题的求解算法也会完全不同，设计合理的数据结构常常可以有效地化简算法。例如，要在具有 100 万个用户的电话簿中查找某人的电话号码，根据数据结构的不同有如下两种算法来实现：一是电话簿中所有用户电话记录都是随机存放的，则只能按顺序依次查找，找到一个用户电话号码平均需要查看 50 万个用户记录；二是电话簿中所有用户电话记录按某个规则排序（序号顺序、字母排序等）存放，则可以用二分查找，平均只需要 28 次查找。两种查找算法的速度和效率相去甚远，主要是数据构造有明显差别。

在本章后面数据结构分析中，常常需要分析在特定数据结构上的算法的效率（由于本教材中算法举例都是很简单的，一个算法的效率往往就是一个基本运算的执行情况，所以会直接分析运算的效率来表征整个算法的效率。在更大型的程序和较复杂的处理中算法的效率和基本运算的效率是不同的两个概念），这里就给出算法效率的定义和计算方法。

算法的效率可分为时间效率和空间效率。算法执行时间需通过依据该算法编制的程序在计算机上运行时所消耗的时间来度量。空间效率主要考虑除存储数据结构本身外，实现算法所需要的辅助空间有多少。

算法执行时间需通过依据该算法编制的程序在计算机上执行时所消耗的时间来度量。而算法所需的运行时间与书写算法的程序设计语言、算法选用何种策略、编译后所产生的机器代码的质量、机器执行指令的速度以及待解决的问题规模大小有关。在这些因素中，前 4 个是与具体的机器或语言有关，在本章不予考虑，仅考虑算法本身效率高低。可以认为，一个特定算法的效率是问题规模的函数。假设整个算法的执行时间为 T，问题的规模为 n，一般情况下，算法中基本操作的重复执行次数是问题规模 n 的某个函数 f(n)，则算法的时间量度记为

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

式中，O[f(n)] 称为语句频度，它表示随问题规模 n 的增大，算法执行时间的增长率和 f(n) 的增长率相同，称为算法的渐近时间复杂度，简称时间复杂度。语句的频度是指语句重复执行的次数，例如：