高等专业学校
电子信息类　规划教材

软件技术基础

黄迪明　主　编
黄迪明　李玉柏　许家珆　张徐亮　丁香荣　编著

电子科技大学出版社

高等专业学校
电子信息类　规划教材

软件技术基础

黄迪明　主编

黄迪明　李玉柏　许家珆　张徐亮　丁香荣　编著

出　　版：电子科技大学出版社　（成都建设北路二段四号）　邮编（610054）

责任编辑：吴艳玲

发　　行：新华书店发行

印　　刷：四川建筑印刷厂

开　　本：787×1092　1/16　印张 20.5　字数 495 千字

版　　次：1998 年 10 月第一版

印　　次：1998 年 10 月第一次印刷

书　　号：ISBN7—81043—928—6/TP・407

印　　数：1—4000 册

定　　价：23.00 元

内容简介

本书是国家电子类“九五”规划教材，重点介绍了计算机软件技术基础的几个主要部分。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、简明扼要、深入浅出、注重实用。

本书内容是按国家教委高教司颁发的“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。

出 版 说 明

为做好全国电子信息类专业“九五”教材的规划和出版工作，根据国家教委《关于“九五”期间普通高等教育教材建设与改革的意见》和《普通高等教育“九五”国家级重点教材立项、管理办法》，我们组织各有关高等学校、中等专业学校、出版社，各专业教学指导委员会，在总结前四轮规划教材编审、出版工作的基础上，根据当代电子信息科学技术的发展和面向21 世纪教学内容和课程体系改革的要求，编制了《1996—2000 年全国电子信息类专业教材编审出版规划》。

本轮规划教材是由个人申报，经各学校、出版社推荐，由各专业教学指导委员会评选，并由我部教材办协商各专指委、出版社后，审核确定的。本轮规划教材的编制，注意了将教学改革力度较大、有创新精神、特色风格的教材和质量较高、教学适用性较好、需要修订的教材以及教学急需，尚无正式教材的选题优先列入规划。在重点规划本科、专科和中专教材的同时，选择了一批对学科发展具有重要意义，反映学科前沿的选修课、研究生课教材列入规划，以适应高层次专门人才培养的需要。

限于我们的水平和经验，这批教材的编审、出版工作还可能存在不少缺点和不足，希望使用教材的学校、教师、同学和广大读者积极提出批评和建议，以不断提高教材的编写、出版质量，共同为电子信息类专业教材建设服务。

电子工业部教材办公室

前　言

本教材系按电子工业部的《1996－2000 年全国电子信息类专业教材编审出版规划》，由全国自动控制专业教学指导委员会编审、推荐出版。本教材由电子科技大学黄迪明主编，王正智主审，西安交通大学郑南宁任责任编委。

《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委高教司颁发的“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《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，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，为计算机在专业应用中奠定基础。

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，张徐亮编写第四章，丁香荣编写第五章。全书由黄迪明统筹并修改定稿。电子科技大学兰家隆教授、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于了大力的支持，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

1998 年6 月于电子科技大学

目　录

第一章　数据结构

§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、二进制常数；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（有些书上称为数据结点）。

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，即，数据元素之间有一定相互关系，这种相互关系则是以对数据元素的一些运算来表示。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，下面举一例来具体说明上述概念。

例1　学生成绩表为：

学号	姓名	班级	数学	物理	电路基础	C 语言
1	丁一	960131	87	90	86	84
2	马二	960131	80	81	86	90
3	张三	960131	91	88	90	87
4	李四	960131	93	88	92	85
5	王五	960131	67	66	70	65
⋮	⋮	⋮	⋮	⋮	⋮	⋮

这张学生成绩表就可称为一个数据结构，它是一个线性表结构，表中的每一行为一个数据元素。它是由学号、姓名、各科成绩、班级等数据项组成。该表中数据元素之间的逻辑关系是：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。上述元素间的关系构成了这张学生成绩表的逻辑关系。该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，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，而是把它看成一个结点，着重讨论结点之间的关系。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];
      int classes;
      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，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）都存在一条从根元素到该元素的路径。

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——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 型也有三种型式：float，double 和longdouble，对应浮点型、双精度浮点型和扩长的双精度浮点型，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于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，如有变量说明：

```
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 抽象数据类型

数据类型这个概念最早出现在高级程序设计语言的实现中。众所周知，计算机硬件系统并不能直接处理整数、浮点数、双精度数等，需按一定规则转换成二进制码来表示和处理。从计算机硬件系统的角度看，计算机能处理的数据类型只包括位、字节和字。高级程序设计语言设计者们引入整数、浮点数、双精度数等，对计算机硬件系统来说就是一种新的数据结构。对这种数据结构的操作需通过编译器转化成机器语言的位、字节和字这样的计算机硬件能接受的数据类型的操作来实现。从高级程序设计语言用户的角度看，数据结构实现了信息的隐蔽，即将一切用户不必了解的细节都封装在数据结构之中，从而实现了数据的抽象。例如，硬件系统直接支持的机器语言程序需考虑整数的具体机器表示和整数运算的机器实现算法，而高级程序设计语言用户在使用整数时，不需要了解整数在计算机内部是如何表示和各种运算是如何实现的。对如两整数求和这样的问题，高级程序设计语言的用户只需了解其数学上求和的抽象操作含义。这样，在硬件系统机器语言的数据类型

支持下，高级程序设计语言实现了整数这样的数据结构。进一步，对高级程序设计语言来说，整数就是它的数据类型，或称固有数据类型，高级程序设计语言用户可直接使用。在用高级程序设计语言编写的程序中，每个变量、常量或表达式都有一个它所属的确定的数据类型。数据类型显式或隐式地规定了在程序执行期间变量、常量或表达式所有可能的取值范围，以及允许进行的操作。例如，C 语言中的整数类型，其取值范围为[－maxint，maxint）上的整数（maxint 为特定计算机所允许的最大整数），定义在该取值区间上的一组操作为：加（＋）、减（－）、乘（＊）、整除（/）和取余（%）。

因此数据类型的定义为：数据类型是一个值的集合和定义在这个值的集合上的一组操作的总称。或者说，数据类型是某种程序设计语言中已实现的数据结构。

抽象数据类型（ADT）：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取决于它的一组逻辑特性，而与其在计算机内部如何表示和实现无关，即不论其内部结构如何变化，只要它的数学特性不变，都不影响其外部的使用。

抽象数据类型和数据类型实质上是一个概念。例如，各个计算机都拥有的“整数”类型是一个抽象数据类型，尽管它们在不同处理器上实现的方法可以不同，但由于其定义的数学特性相同，在用户看来都是相同的。因此，“抽象”的意义在于数据类型的数学抽象特性。

另一方面，抽象数据类型的范畴更广，它不再局限于前述各处理器中已定义并实现的数据类型（也可称这类数据类型为固有数据类型），还包括用户在设计软件系统时自己定义的数据类型。为了提高软件的复用率，在近代程序设计方法学中指出，一个软件系统的框架应建立在数据之上，而不是建立在操作之上（后者是传统的软件设计方法所为）。即在构成软件系统的每个相对独立的模块上，定义一组数据和施于这些数据上的一组操作并在模块内部给出这些数据的表示及其操作的细节，而在模块外部使用的只是抽象的数据和抽象的操作。显然，所定义在数据类型的抽象层次越高，含有该抽象数据类型的模块的复用程度也就越高。

如前所述，抽象数据类型的定义由一个值域和定义在该值域上的一组操作组成。若按其值的不同特性，可细分为下列三种类型：

（1）原子类型（Atomic Data Type）：属原子类型的变量的值是不可分解的。这类抽象数据类型较少，因为一般情况下，已有的固有数据类型足以满足需求。但有时也有必要定义新的原子数据类型，例如：定义值为0～100 的整数。

（2）固定聚合类型（Fixed-Aggregate Data Type）：属该类型的变量，其值由确定数目的成分按某种结构组成。例如，复数是由两个实数依确定的次序关系构成。

（3）可变聚合类型（Variable-Aggregate Data Type）：和固定聚合类型相比较，构成可变聚合类型“值”的成分的数目不确定。例如，可定义一个“有序整数序列”的抽象数据类型，其中序列的长度是可变的。

显然，后两种类型可统称为结构类型。

一个含抽象数据类型的模块通常应包含定义、表示和实现三个部分。和数据结构的形式定义相对应，抽象数据类型采用以下格式定义：

ADT 抽象数据类型名｛

　　数据对象：〈数据对象的定义〉

数据关系:〈数据关系的定义〉

基本操作:〈基本操作的定义〉

} ADT 抽象数据类型名

其中,数据对象和数据关系的定义用文字或伪码描述,基本操作的定义格式为

基本操作名(参数表)

初始条件:〈初始条件描述〉

操作结果:〈操作结果描述〉

基本操作有两种参数:赋值参数只为操作提供输入值;引用参数以& 打头,除可提供输入值外,还将返回操作结果。“初始条件”描述了操作执行之前数据结构和参数应满足的条件,若不满足,则操作失败,并返回相应出错信息。“操作结果”说明了操作正常完成之后,数据结构的变化状况和应返回的结果。若初始条件为空,则省略之。

抽象数据类型的定义与使用实现了数据的封装和隐藏,为对象程序设计方法提供了基础,也使过程语言的可维护性大大提高。近年来,抽象数据类型的定义迅速发展和流行起来。为了便于读者今后进一步学习新的软件知识,在后面两节的小结中,分别给出了常见的线性数据结构和非线性数据结构的抽象数据类型的定义,有兴趣的读者可仔细阅读一下,并比较一下它们与常规的数据结构的定义有什么不同。

小　　结

本节对数据结构的基本概念和它主要研究的内容进行简要的介绍,尤其是数据和数据结构中的一些基本名词和术语进行了解释;并回顾了C 语言中数据类型说明的方法,这将有助于后续内容和程序范例的阅读和理解。

§1.2　线 性 结 构

线性结构是数据结构中最简单且最常用的一种数据结构。线性结构的基本特点是数据元素有序并有限。正如上一节所述:线性结构的逻辑特征是在其结构中,有且仅有一个无直接前趋而仅有一个直接后继的数据元素为起始元素;有且仅有一个无直接后继而仅有一个直接前趋的数据元素为终点元素;其余均为内部元素,它们各有一个直接前趋和直接后继。因此,线性结构的数据元素可排成一个线性的序列:

$a_1,a_2,a_3,\cdots,a_n$

其中,a_1 为起始元素,a_n 为终点元素,a_i 为索引号为i 的数据元素。

线性结构有各种类型,如线性表、堆栈、队列、数组、串等。

1.2.1　线性表

线性表是n(n≥0)个相同类型的元素$a_1,a_2,\cdots,a_n$ 所构成的有限线性序列,通常表示为$(a_1,a_2,\cdots,a_n)$,其中n 为线性表的长度。$a_i(1\leqslant i\leqslant n)$是线性表中第i 个序号的数据元素。$a_i$ 是抽象表示符号,在不同的情况下含义不同。例如:一个整数序列:(1,12,123,1234,321,21,22)是一个线性表,表中元素a_i 是一个整数,表长为7。而一周的七天:(SUN,