高等职业技术教育计算机系列教材

数据结构

（C语言版）

杨振生　编著

SHUJU JIEGOU
(C YUYAN BAN)

SHUJU JIEGOU
(C YUYAN BAN)

中国科学技术大学出版社

高等职业技术教育计算机系列教材

数据结构

（C语言版）

杨振生　编著

中国科学技术大学出版社
2004·合肥

内 容 提 要

本书系统地介绍了“数据结构”的基本内容，阐述概念准确、通俗易懂、思路清晰、结构严谨。根据高等职业技术教育的特点和培养目标的要求，本书注重基础知识，突出应用性、实用性和可读性，强调理论联系实际，培养学生分析问题和解决问题的技能。

全书共分十章，主要内容包括：数据结构的基本概念、线性表、栈与队列、串和数组、树、图、查找、排序和文件等。书中例题丰富，侧重基础理论、算法的思路与算法的实现。每章最后都附有习题和实习参考题，供教师选择或参考。在教学实施过程中，根据实际情况，可对部分内容做适当删减。

本书可作为大专、高职类院校的计算机专业、信息工程专业的教材，也可供相关专业科技人员参考。

图书在版编目(CIP)数据

数据结构：C语言版/杨振生编著. —合肥：中国科学技术大学出版社，2004.4
ISBN 7-312-01651-0

Ⅰ.数… Ⅱ.杨… Ⅲ.①数据结构-高等学校：技术学校—教材 ②C语言—程序设计—高等学校：技术学校—教材 Ⅳ.①TP311.12 ②TP312

中国版本图书馆CIP数据核字(2003)第112066号

中国科学技术大学出版社出版发行
(安徽省合肥市金寨路96号，邮编：230026，发行电话：0551-3602905，3602906)
合肥学苑印务有限公司印刷
全国新华书店经销

开本：787mm×1092mm 1/16 印张：12.75 字数：320千
2004年3月第1版 2004年3月第1次印刷
印数：1—4000册
ISBN 7-312-01651-0/TP·336 定价：15.00元

前　言

近些年来，我国高等教育事业获得了蓬勃发展，特别是高等职业教育异军突起，其发展速度之迅猛、势头之强劲，超出了人们的预料。据统计高等职业教育的学生规模已占据了全国整个高等教育的半壁江山。这说明高等职业教育和普通高等教育可相提并论，在我国高等教育体系中占有相当重要的地位。

高等职业教育作为一种新的办学模式，其培养目标是具备必要的理论基础和较强的实践能力，直接面向工作在生产、管理和服务于第一线的应用型、技能型的高级实用人才。根据这一培养目标，传统的高等院校教材已不适应高职教育的需要，高职教材必须符合高职教育培养目标的要求，逐渐形成自己的教材特色。因此，当前高职教材建设是一个刻不容缓的迫切任务。

目前真正适合于高职教育的各类教材极少，甚至没有。现在使用的基本上是沿用传统的高等院校的教材或在其基础上进行了压缩，没有形成具有高职教育的特色，尤其在以“应用型”、“技能型”为主旨和特征的教材内容体系上，存在着明显不足。为适应这一新形势，仓促编写出的一些教材，一般质量不高，错误漏洞较多，难以适应高等职业技术教育培养目标的要求。本书的编写，吸取了同类教材的长处和优点，尽力体现“应用型”和“技能型”的特点，实际上是对高职教材建设方面的一种探索。

全书共分 10 章。第 1 章介绍了数据结构的基本概念；第 2 章至第 5 章分别介绍了线性表、栈和队列、串、数组的数据结构和基本算法及其应用；第 6 章和第 7 章介绍了树和图的各种存储结构以及应用；第 8 章和第 9 章分别介绍了查找和排序的基本方法；第 10 章介绍了文件的基本概念和几种重要的文件组织方式。每章最后都配有习题和实习参考题。

由于作者水平有限，书中不妥和疏漏之处在所难免，敬请专家学者和广大读者予以批评指正。

杨振生

2003 年 11 月于合肥

目 录

第1章 绪 论

本章要点:(1)数据、数据表示和数据处理;(2)数据结构的含义;(3)数据的逻辑结构、存储结构与运算;(4)算法的描述与分析。

本章要求:(1)理解数据结构的有关概念;(2)搞清数据的逻辑结构、存储结构及其相互关系;(3)了解数据结构的基本类型与特点;(4)掌握算法描述和分析的基本方法。

当今的世界是一个充满信息的世界,社会中的诸多领域都形成了各自的信息海洋。在解决实际问题时,人们只关心对自己有用的信息,并利用计算机对这些信息进行高效处理,从而实现处理要求,最终获得所需的结果。因为信息是通过数据表达的,因此对信息的处理实质上就是对数据的处理。那么利用计算机怎样才能实现对数据的高效处理呢?这就必须研究和解决如何科学地组织数据,选择或建立合理的数据结构,设计或挑选一个好的算法,提高程序执行的效率。"数据结构"这门学科就是在这样的历史背景下逐步形成和发展起来的。

"数据结构"是计算机专业和信息管理专业的一门十分重要的专业基础课程。计算机的所有系统软件和应用软件设计都要涉及到各种类型的数据结构。若想编写出好的、高效的程序,仅仅熟悉计算机语言是不够的,必须牢固地掌握数据结构这门课程,才能大大增强解决实际问题的能力和提高程序设计的水平。

1.1 数据、数据表示和数据处理

在介绍"数据结构"这一重要概念之前,必须首先介绍与之有关的一些最基本的概念和术语。

1.1.1 数据

所谓数据是指人们为了描述客观事物的特征及其活动所采用的符号表示。例如,字母、数字、汉字以及其他规定的符号。实际上,随着计算机科学的发展和应用普及,数据已具有广泛的含义,且扩展到字符串、表格、图像甚至语音等。总之,凡能被计算机接收、存储和加工的对象统称为数据。

数据元素是数据的基本单位,在计算机中通常作为一个整体进行考虑和处理。在本课程中,数据元素又被称为元素、结点、顶点或记录。

通常数据元素又由若干数据项组成,而数据项是具有独立含义的、不可再分割的数据的最小单位。数据项有时又称为字段或域。例如,在表1.1中是人事工资数据,每个职工的有关信

息在表中占一行，这就作为一个数据元素或记录。而每个数据元素由编号、姓名、性别、工作日期、职称、婚否、基本工资、奖金、实发金额等 9 个数据项组成。

表 1.1 人事工资表

编 号	姓 名	性别	工作日期	职 称	婚否	基本工资	奖金	实发金额
58001	杨 洋	男	58－09－01	高 工	.T.	1650.00	180.00	1830.00
85001	李庆乾	男	85－07－15	助 工	.F.	500.00	65.00	565.00
70005	林 楠	女	70－09－01	工程师	.T.	800.00	105.00	905.00
65012	方 圆	男	65－09－01	高 工	.T.	1140.50	150.00	1290.50
71024	王丽萍	女	71－09－01	工程师	.T.	750.00	100.00	850.00

由此可见，数据、数据元素和数据项实际上反映了数据组织的三个层次，即数据可由若干数据元素构成，而数据元素又可由若干个数据项构成。

1.1.2 数据表示

数据在计算机存储器之外的存在形式，即在现实生活和实际问题中的呈现形式，则称为数据的机外表示。将数据输入到计算机存储器中的存在形式，则称为数据的机内表示。因此，为了让计算机对数据进行加工处理，必须首先将数据从机外表示转化为机内表示，这项工作称为数据表示。

1.1.3 数据处理

仅仅将数据转化为机内表示并不能解决问题，还必须根据实际问题的处理要求和数据的组织形式定义一组操作或运算，并编制程序，让计算机执行所编的程序去完成这组操作从而实现处理要求，并得到所需的处理结果。这项工作称为数据处理。

总之，无论我们解决任何实际问题，必须完成两项基本任务：数据表示和数据处理。应当指出，数据表示与数据处理是密切相关的，数据处理方式总是与数据的表示形式相联系，反之亦然。

1.2 什么是数据结构

数据结构是指数据的组织形式，即数据元素之间的相互关系。数据结构的内容包括三个方面：数据的逻辑结构、数据的存储结构和数据的运算。

1.2.1 数据的逻辑结构

通常所说的数据结构是指数据的逻辑结构。

所谓数据的逻辑结构是指具有逻辑关系的数据元素的集合。所谓逻辑关系是指数据元素之间的关联方式，它反映了数据的组织结构。

为了更确切地描述一种数据结构，通常采用二元组表示：

$$DS=(K,R)$$

其中 DS(Data Structure 的缩写)是一种数据结构，K 为数据元素的集合，R 是 K 上二元关系

的集合。且 $K=\{k_1,k_2,\cdots,k_n\}$，n 为 K 中数据元素的个数，当 K 是一个空集时，则 DS 也就无结构而言。而 $R=\{r_1,r_2,\cdots,r_m\}$，m 为 R 中二元关系的个数，若 R 为空集，表明 K 中的数据元素之间不存在任何关系，彼此是独立的。

K 上的一个关系 R 是序偶的集合。假设 K 中的两个数据元素 k_i 和 k_j 存在逻辑关系，则用序偶$\langle k_i,k_j\rangle$表示，我们把 k_i 叫做序偶的第一元素，把 k_j 叫做序偶的第二元素，又称 k_i 是 k_j 的直接前驱(简称前驱)，称 k_j 为 k_i 的直接后继(简称后继)。

在数据结构中，数据元素又称为元素或结点。若某个结点没有前驱，则称该结点为开始结点；若某个结点没有后继，则称该结点为终端结点。

对于对称序偶，$\langle k_i,k_j\rangle\in R$，$\langle k_j,k_i\rangle\in R$，则用圆括号代替尖括号，即$(k_i,k_j)\in R$。

由此可知，R 是 K 中所有具有逻辑关系的数据元素所构成的序偶集合。

例 1.1 给出表 1.1 人事工资的数据结构。

在表 1.1 中共有 5 个数据元素，依次用 $k_1\sim k_5$ 表示，其对应的逻辑关系是相邻关系。所对应的数据结构表示为 DS=(K,R)，其中

$K=\{k_1,k_2,k_3,k_4,k_5\}$

$R=\{<k_1,k_2>,<k_2,k_3>,<k_3,k_4>,<k_4,k_5>\}$

可用图 1.1 表示。

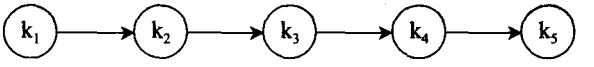

图 1.1 表 1.1 数据结构的示意图

例 1.2 给出以下逻辑结构图，如图 1.2 所示。

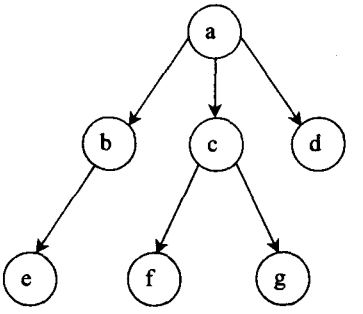

图 1.2 一个逻辑结构图

试写出它的数据结构。

这种逻辑结构我们将要介绍是一种树形结构，其对应的数据结构可描述为：

DS=(K,R)

其中， $K=\{a,b,c,d,e,f,g\}$

$R=\{<a,b>,<a,c>,<a,d>,<b,e>,<c,f>,<c,g>\}$

数据的逻辑结构就是数据的组织形式，可通过图示描述。图中的小圆圈表示结点，一个结点代表一个数据元素，结点之间的连线表示逻辑关系，即数据元素之间的邻接关系。

数据的逻辑结构可分为以下四种基本类型：

(1) 集合

集合中任何两个结点之间都没有逻辑关系，彼此是独立的，其组织形式是散列的。如图1.3所示。在数据结构DS=(K,R)中，R为空集，即空关系。

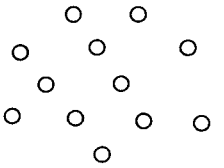

图1.3 集合结构

(2) 线性结构

线性结构中的结点按逻辑关系依次相邻而排列成一条“锁链”，结点间存在一个对一个的关系，如图1.4所示。

图1.4 线性结构

(3) 树形结构

树形结构中结点的组织形式具有分支、层次特点，结点之间存在一个对多个的关系，其形态如同自然界中的树。树中每个结点最多只有一个前驱，但可以有多个后继，且可有多个终端结点。如图1.5所示。

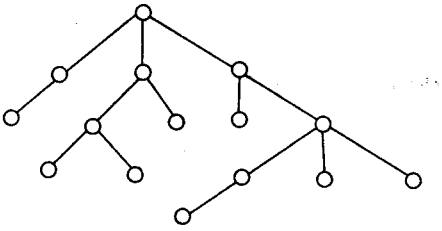

图1.5 树形结构

(4) 图状结构

图状结构是最复杂的，在这种结构中，结点间既存在纵向联系，又存在横向联系，即结点间存在多个对多个的关系。每个结点的前驱和后继的个数都是任意的。如图1.6所示。

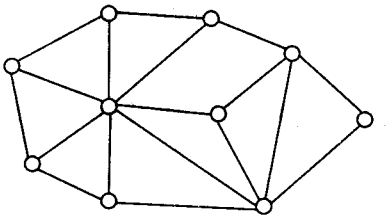

图1.6 图状结构

概括地说，数据的逻辑结构可分为两大类：线性结构和非线性结构。而非线性结构又分为树形结构和图状结构。根据上面的定义可知：线性结构是树形结构的特例，而树形结构是

图状结构的特例。

1.2.2 数据的存储结构

数据的存储结构是数据的逻辑结构在机内的表示。或者说,数据的存储结构是数据的逻辑结构在计算机存储器中的存储实现。一般地说,一个存储结构必须包括以下两个主要部分:

1. 存储结点,每个存储结点存放一个数据元素;

2. 数据元素之间关联方式的表示,也就是逻辑关系的机内表示。

通常,存储结点之间的关联方式有四种,故分为以下四种基本的存储方式:

(1) 顺序存储方式

每个存储结点只含一个数据元素,把逻辑上相邻的结点相继存放在一个连续的存储区里。用存储结点间的位置关系表示数据元素之间的逻辑关系。按这种方式表示逻辑关系的存储结构称为顺序存储结构。

(2) 链式存储方式

每个存储结点不仅含有一个数据元素,还包含一个(或一组)指针。每个指针指向一个与本结点有逻辑关系的结点,即用附加的指针表示逻辑关系。按这种方式组织的存储结构称为链式存储结构。

(3) 索引存储方式

每个存储结点只含一个数据元素,所有存储结点连续存放,另外增设一个索引表,用结点的索引号来确定结点的存储位置。按这种方式组织的存储结构称为索引存储结构。

(4) 散列存储方式

每个结点含有一个数据元素,根据结点的值用散列函数来确定各结点的存储位置。按这种方式组织的存储结构称为散列存储结构。

1.2.3 数据的运算

数据的运算是指在数据的逻辑结构上定义一组操作(又称运算),以实现数据处理要求。

设S为某种逻辑结构,在S上到底定义哪些操作,可以根据需要而定义。因此,操作的种类是没有限制的。而常用的基本操作主要有以下几种:

(1) 读取。在S中某指定位置上读出结点的内容;

(2) 插入。在S中的指定位置增添一个新的结点;

(3) 删除。删去S中某个指定位置上的结点;

(4) 查找。在S中找出满足某种条件的结点;

(5) 更新。修改S中某指定结点的内容。

综上所述,关于数据结构的含义可以概括为:一个数据结构是由一个逻辑结构S和定义在S上的一个基本运算集Δ所构成的整体(S,Δ)。于是,数据结构的主要内容侧重于两方面:

1) 数据结构(包括逻辑结构和基本运算集)的定义;

2) 数据结构的实现(包括存储的实现和运算的实现)。

1.3 算法及其描述

数据结构的各种操作都是以算法形式描述的，而算法的实现都是以程序形式表现的。故数据结构、算法和程序是密不可分的，它们之间的关系可表示为：

数据结构＋算法＝程序

本节先介绍算法的定义和特性，然后介绍算法的描述方法。

1.3.1 什么是算法

在解决实际问题时，当确定了数据的逻辑结构和实现了数据的存储结构之后，必须进一步研究与之相关的一组操作(或运算)的实现。为了实现某种操作通常需要一种算法。

确切地说，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。实质上，算法是在存储结构上的操作实现方法。

一个算法应具有以下五个重要特性：

(1) 有穷性。一个算法必须在执行有穷步之后结束，且每一步都可在有穷时间内完成。

(2) 确定性。一个算法中每一条指令必须有确切的含义，不会产生二义性。

(3) 可行性。一个算法是可行的，即算法中描述的操作都可通过已经实现的基本运算执行有限次来实现。

(4) 输入。一个算法有零个或多个输入，这些输入取自于某个选定的对象的集合。

(5) 输出。一个算法有一个或多个输出，这些输出是同输入有某些特定关系的量。

例 1.3 设计一个算法：求一元二次方程

$$ax^2+bx+c=0$$

的实根。

该算法由下列步骤构成(用 C 语言表达)：

第 1 步：计算 d=b * b−4 * a * c；

第 2 步：如果 d>0，则转第 5 步；

第 3 步：如果 d=0，则转第 9 步；

第 4 步：如果 d<0，则转第 12 步；

第 5 步：计算 x1=(−b+sqrt(d))/(2 * a)；

第 6 步：计算 x2=(−b−sqrt(d))/(2 * a)；

第 7 步：显示两个不同的实根 x1 和 x2 的值；

第 8 步：转第 13 步；

第 9 步：计算 x=(−b)/(2 * a)；

第 10 步：显示两个相同的实根 x 的值；

第 11 步：转第 13 步；

第 12 步：显示没有实根的信息；

第 13 步：结束。

1.3.2 算法的描述

算法的描述是指用某种方式将算法的实现具体的描写出来。

算法的描述可以有多种方式，如语言方式、图形方式和表格方式等。在本书中将采用C语言描述，C语言是一种高效、灵活和精炼的高级程序设计语言，其优点是数据类型丰富、语句简洁，编写的程序结构化程度高、可读性强。例如，例1.3的算法用C语言描述如下：

```
void solution(float a, float b, float c)
{ float d, x, x1, x2;
  d=b*b-4*a*c;
  if(d>0)
  { x1=(-b+sqrt(d))/(2*a);
    x2=(-b-sqrt(d))/(2*a);
    printf("x1=%f, x2=%f\n", x1,x2);
  }
  else if(d==0)
      { x=(-b)/(2*a);
        printf("x=%f\n", x);
      }
      else printf("不存在实根\n");
  }
```

用于描述算法的常用C语言语句：

(1) 输入语句

Scanf(<格式控制串>,<输入项表>);

(2) 输出语句

printf(<格式控制串>,<输出项表>);

(3) 赋值语句

变量名=表达式;

(4) 条件语句

if(<条件>)<语句>;

或者 if(<条件>)<语句1> else <语句2>;

(5) 循环语句

While(<表达式>)<循环体语句>;

do

<循环体语句>;

While(<表达式>);

for(<赋初值表达式1>;<条件表达式2>;<步长表达式3>)

<循环体语句>;

(6) 返回语句

return(<返回表达式>);

(7) 函数定义语句

<函数返回值类型><函数名>(<类型名><形参1>,<类型名><形参2>,…)

{ <说明部分>;
<执行语句部分>;
}

(8) 调用函数语句

<函数名>(<实参 1>,<实参 2>,…);

1.4 算法分析

一个算法设计完后,需要对其进行分析,以确定该算法的优劣。评价一个算法的优劣主要考查这个算法的性能和效率。

1.4.1 算法的性能标准

算法与数据结构的好坏直接相关。判断一个算法的优劣主要有以下几个标准:

(1) 正确性。算法应当满足实际问题的需求,能够正确地实现预先规定的功能。这是最重要也是最基本的准则。

(2) 可读性。算法应当可读性好,这样才有助于对算法的阅读和理解。为达此要求,算法的思路应当是清晰的,层次逻辑应当是结构化的。

(3) 健壮性。要求算法具有很好的容错性,即提供例外处理,能够对非法数据进行检查并作出相应反应,而不会产生莫明其妙的输出结果。处理出错的方法应是返回一个错误信息,且中止程序的执行。

(4) 效率。算法的效率主要是指算法执行时所消耗的计算机资源,包括存储空间和运行时间的开销。前者叫做算法的空间代价,后者叫做算法的时间代价。

1.4.2 算法的效率分析

算法的效率主要是指算法的空间效率和时间效率。空间效率又称空间复杂度,时间效率又称时间复杂度。前者是算法所需要的存储量,后者是算法所包含的计算量。

1. 空间复杂度

我们首先应当了解什么是问题的规模。问题的规模或一个算法的输入规模是指该算法输入的数据所含数据元素的数目,或与此数目有关的其他参数。例如,对于表 1.1 的查找算法中,其输入规模是表中所含职工的数目。

算法的空间复杂度是指当问题的规模以某种单位由 1 增至 n 时,解决该问题的算法实现也以某种单位由 1 增至 f(n),且记作

$$S(n)=O(f(n))$$

它表示随问题规模 n 的增大,算法所需存储空间的增长率和 f(n)的增长率相同。

2. 时间复杂度

一个算法的运行时间是指计算机从开始运行到结束所花费的时间,它大致等于计算机执行某种基本操作(如赋值、比较等)所需时间与其执行次数的乘积。

在一个算法中某语句重复执行的次数称为该语句的频度。

把算法中所含基本操作重复执行的次数称为算法的时间复杂度。一般情况下,算法中基