普通高等院校规划教材

数据结构与算法

C语言版

程玉胜　主编

中国科学技术大学出版社

普通高等院校规划教材

数据结构与算法(C语言版)

主　编　程玉胜

中国科学技术大学出版社

内 容 简 介

“数据结构”是计算机科学与技术和信息工程专业重要的公共基础课程，更是提高编程能力以及学习后续课程的基础。

本书主要针对一般本科高校层次的计算机及相关专业编写，主要内容包括线性表、栈和队列、数组和字符串、矩阵、二叉树、图、排序和查找等基本数据结构和算法。

考虑到知识点较多，算法学习相对较难，实验内容较多等问题，我们同时出版了《数据结构与算法(C语言版)习题精编与实验指导》，作为主教材的配套教学参考书，旨在帮助大家及时通过练习掌握重要知识点的解题方法，通过相应的实验内容分解，旨在提高初学者的综合编程能力。

本书可供从事“数据结构”及其相关课程教学的教师作为参考书，也可供相关专业学生作为学习、实验和考研的参考书。

图书在版编目(CIP)数据

数据结构与算法：C语言版/程玉胜主编. —合肥：中国科学技术大学出版社，2015.1
ISBN 978-7-312-03603-3

Ⅰ.数… Ⅱ.程… Ⅲ.①数据结构—高等学校—教材 ②算法分析—高等学校—教材 ③C语言—程序设计—高等学校—教材 Ⅳ.①TP311.12 ②TP312

中国版本图书馆CIP数据核字（2014）第247755号

出版 中国科学技术大学出版社
安徽省合肥市金寨路96号，邮编：230026
http://press.ustc.edu.cn
印刷 合肥现代印务有限公司
发行 中国科学技术大学出版社
经销 全国新华书店
开本 787 mm×1092 mm 1/16
印张 16.75
字数 430千
版次 2015年1月第1版
印次 2015年1月第1次印刷
定价 30.00元

本书编委会

前　言

“数据结构”是计算机科学与技术和信息工程专业重要的专业基础课程，是多数高校计算机专业及相关专业研究生入学考试的必考科目之一，也是提高综合编程能力，学习后继课程的基础。该书系统地介绍了软件设计中常见的线性表、串、栈和队列、数组、矩阵、树和二叉树、图、查找和排序等数据结构及其算法在计算机中的存储和各种操作的实现方法。

书中涉及大量的算法和存储结构，课程内容丰富，学习量大，其算法又相对抽象，学生学习困难，压力较大，甚至影响到学生的进一步深造。为此，我们还组织了“数据结构”教学一线教师，根据一般本科高校学生学习的特点，撰写了《数据结构与算法(C 语言版)习题精编与实验指导》，作为主教材的配套学习用书，其内容涵盖了选择、填空、判断和解答四种常见题型，收集了多年来考研、教学辅导等 4000 多套试题，并且将经典题型或者重点知识点题型纳入“例题精解和习题实训”中，对其进行解题分析并提供参考答案。同时，为方便学生自主式学习，本书还提供了安庆师范学院数据结构课程组开发的教学资源，供大家学习参考，希望能够激发学生的学习兴趣，巩固学习的知识点。

全书内容共 9 章，第 1 章介绍了数据结构课程的研究内容、特点以及数据结构相关定义、算法描述和复杂度度量等基础性问题。第 2 章介绍了线性表的逻辑结构特点及其相关运算，重点讨论了顺序表和链表两种存储结构及其基本运算的实现，其重要知识点在配套参考书中进行了大量分析和精解。第 3 章介绍了两种最常用的数据结构——栈和队列，它们属于线性结构，是一种特殊的线性表。紧紧围绕它们的特点，介绍了这两种数据结构的存储和实现方法，以及这两种结构的典型应用实例，如进制转换、打印杨辉三角等。第 4 章介绍了串，它属于线性结构在存储数据类型上的拓展，主要介绍了串的常用操作。第 5 章介绍了数组和广义表，主要包括寻址、压缩存储、矩阵转置等；另外还介绍了广义表的概念和相关操作。第 6 章介绍了树和二叉树的基本概念和主要操作，重要知识点是二叉树的遍历算法及其应用，树和森林到二叉树的相互转换，哈夫曼树的构造等；介绍了线索化二叉树及其相应的算法。第 7 章介绍了图的相关概念和两种最常用的物理存储结构，重点介绍了图的遍历、最小生成树、最短路径、拓扑排序等主要算法。第 8 章介绍了查找操作，主要包括软件设计中最常用的基本查找方法，重点讨论了基于线性表的、基于树的、基于函数的三种查找方法。第 9 章介绍了插入类、交换类、选择类、归并类四大种常用的排序算法及其性能分析，介绍了基数排序的方法。

本书由程玉胜教授(安庆师范学院)任主编，金萍(皖西学院)、王秀友(阜阳师范学院)、王刚(铜陵学院)、方祥圣(安徽经济管理学院)、江克勤、石冰(安庆师范学院)任副主编，黄忠、吴超云、刘娟、宗瑜、南淑萍、杨慧等老师参编，同时，安庆师范学院 2012 级统计学研究生梁辉、胡飞、黎康、任勇、何姗姗同学参与了本书的校对和整理工作。

在本书编写过程中，编者参考了大量有关数据结构的书籍和资料，在此对这些参考文献

的作者表示感谢。

由于编者水平及时间有限,书中错误和不足之处在所难免,欢迎读者批评指正。有任何问题,请与作者联系。联系方式:chengysh@aqtc. edu. cn。

编　者

2014 年 6 月

目　　录

第1章 绪 论

【学习概要】

“数据结构”是计算机专业的基础和核心课程，是学习计算机软件和计算机硬件的重要基础。学好该课程，不仅对操作系统、数据库原理、编译原理等后续课程的学习有很大帮助，而且能在实际应用中发挥其重要的作用。

计算机是进行数据处理的基本工具，而数据结构则是主要研究数据的各种组织形式以及建立在这些组织形式之上的各种运算算法的实现，它把数据划分为集合、线性表、树和图等4种类型，后3种是数据结构研究的重点。本章将学习数据结构的基本概念、抽象数据类型及其描述方法、算法的特性及算法分析的方法和技巧。

1.1 数据结构的基本概念

本节将要学习数据结构的一些基本概念和术语，主要包括数据、数据元素、数据对象、数据结构的逻辑结构与存储结构、数据类型等。

1. 数据(Data)

数据是信息的载体，是对客观事物的符号表示，是所有能输入到计算机中并被计算机程序处理的符号集合。数据不仅包括整型、实型等数值类型，还包括字符及声音、图像、视频等非数值数据。例如，一张照片是图像数据，一部电影是视频数据，常见的网页通常包含文字、图片、声音、视频等多种数据。各种数据在计算机内部都是用二进制形式来表示。

2. 数据元素(Data Element)

数据元素是组成数据的基本单位，在计算机中通常作为一个整体进行考虑和处理。

3. 数据对象(Data Object)

数据对象是性质相同的数据元素的集合，是数据的一个子集。例如，正整数数据对象是集合 $N=\{1,2,3,\cdots\}$，大写字母字符数据对象是集合 $C=\{'A','B','C',\cdots'Z'\}$。

4. 数据结构(Data Structure)

结构，简单地理解就是关系，比如分子结构就是组成分子的原子之间的排列方式。在任何问题中，数据元素都不是孤立存在的，而是在它们之间存在着某种关系，这种数据元素相互之间的关系称为结构。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，它包括数据元素的集合及元素间关系的集合，即数据的组织形式。因此，计算机所处理的数据并不是数据的简单堆积，而是具有内在联系的数据集合。

数据结构主要研究数据的逻辑结构(数据元素之间的逻辑关系，它与所使用的计算机无关)和存储结构(数据结构在计算机中的存储表示，既包括数据元素在计算机中的存储方式，也包括数据元素之间的逻辑关系在计算机中的表示，因此它依赖于具体的计算机)以及施加于该数据结构上的操作及其相应算法。

(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的,与数据元素本身的内容和形式无关,与所使用的计算机无关。根据数据元素之间关系的不同特性,通常将关系分为:集合、线性结构、树形结构、图状结构或网状结构等四类,其中后两类又被称为非线性结构。图 1.1 为这四类基本数据结构的示意图。

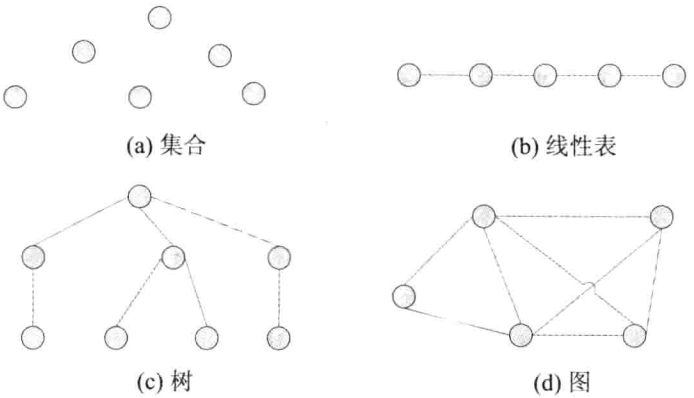

图 1.1 四类基本数据结构示意图

① 集合结构:结构中的数据元素之间除了"同属于一个集合"的关系外,无任何其他关系,类似于数学中的集合。

② 线性结构:结构中的数据元素之间存在着一对一的线性关系。

③ 树形结构:结构中的数据元素之间存在着一对多的层次关系。

④ 图状结构:结构中的数据元素之间存在着多对多的任意关系。

数据的逻辑结构在形式上用一个二元组(D,S)来表示,D 是数据元素的有限集,S 是 D 上关系的有限集,而每个关系都是从 D 到 D 的关系。设 r 是一个从 D 到 D 的关系,$r\in S$,若 $d_1,d_2\in D$,且$\langle d_1,d_2\rangle\in r$,则称 d_2 为 d_1 的直接后继,d_1 是 d_2 的直接前驱,此时 d_1 和 d_2 是相邻(相对于关系 r)的结点;如果不存在一个 d_2,使得$\langle d_1,d_2\rangle\in r$,则 d_1 为 r 的终端结点;如果不存在一个 d_1,使得$\langle d_1,d_2\rangle\in r$,则 d_2 为 r 的开始结点;如果既不是终端结点,也不是开始结点,则称其为内部结点。

【例 1-1】 在计算机科学中,定义复数为如下的一种数据结构 Complex$=(C,R)$,其中 C 是包含两个实数的集合$\{c_1,c_2\}$;$R=\{P\}$,P 是定义在集合 C 上的一种关系$\{\langle c_1,c_2\rangle\}$,有序对$\langle c_1,c_2\rangle$表示 c_1 是复数的实部,c_2 是复数的虚部。

(2) 数据的存储结构

数据的存储结构(又称物理结构)是逻辑结构在计算机中的存储映像,应能正确反映数据元素之间的逻辑关系,因此它是逻辑结构在计算机中的实现,包括数据元素的表示和关系的表示。

数据元素的存储结构形式有两种:顺序存储结构和链式存储结构。顺序存储是借助数据元素在存储时的相对位置来表示数据元素之间的关系,即把数据元素存放在一块地址连续的存储单元里,其数据间的逻辑关系和物理关系是一致的。顺序存储结构如图 1.2 所示。

链式存储是把数据元素存放在任意的存储单元里,这组存储单元的地址可以是连续的,也可以是不连续的,数据元素的存储关系并不能反映其逻辑关系,因此需要用一个指针存放

数据元素的地址，通过地址可以找到相关联数据元素的位置。链式存储结构如图1.3所示。

图1.2 顺序存储结构

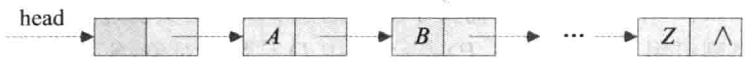

图1.3 链式存储结构

(3) 数据的逻辑结构与存储结构的关系

数据的同一逻辑结构可以对应不同的存储结构，任何一个算法的设计依赖于数据的逻辑结构，而算法的实现则取决于采用的存储结构。

研究数据结构的目的是为了解决实际的应用问题，所以讨论数据结构必须同时讨论在数据结构上执行的相关运算及其算法。通过对运算及其算法的性能分析和讨论，使得在求解实际应用问题时，能恰当选择和设计相应的数据结构，并编写出高效的程序。

5. 数据类型(Data Type)

数据类型是一个值的集合和定义在这个值集上的一组操作的总称，它最早出现在高级程序语言中，用于刻画操作对象的特性。在用高级程序语言编写的程序中，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。数据类型明显或隐含地规定了在程序执行期间变量或表达式所有可能取值的范围，以及在这些值上允许进行的操作。例如，C语言中的字符类型所占空间是8位，这就决定了它的取值范围，在其范围内可以进行赋值运算、比较运算等。

按“值”的不同特性，高级程序语言中的数据类型可分为两类：一类是非结构的原子类型，它的值是不可分解的，例如C语言中的基本类型(整型、实型、字符型和枚举型)、指针类型和空类型。另一类是结构类型，它的值是由若干个类型组合而成，可以再分解，例如，整型数组是由若干整型数据组成的，结构体类型的值也是由若干个类型范围的数据构成，它们的类型都是相同的。

1.2 抽象数据类型

1.2.1 抽象数据类型的定义

抽象数据类型(Abstract Data Type，简称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，与其在计算机内部的表示和实现无关，即不论其内部结构如何变化，只要它的数学特性不变，都不影响其外部的使用。例如，计算机中的整数数据类型是一个抽象数据类型，尽管在不同的处理器上实现的方法可以不同，但其逻辑特性相同，即加、减、乘、除等运算是一致的。

一个抽象数据类型定义了一个数据对象、数据元素之间的关系及对数据元素的操作。抽象数据类型通常是指用户定义的解决应用问题的数据类型，包括数据的定义和操作。例如，C++的类就是一个抽象数据类型，它包括用户类型的定义和在用户类型上的一组操作。本课程中将要学习的线性表、栈、队列、串、树、图等结构就是一个个不同的抽象数据类型。

抽象数据类型体现了程序设计中的问题分解、抽象和信息隐藏等特性。抽象数据类型把实际应用问题分解为多个规模小且容易处理的问题,然后建立起一个计算机能处理的数据模型,并把每个功能模块的实现细节作为一个独立的单元,在模块内部给出相应的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,从而将具体的实现过程隐藏起来,每一个基本操作不需要了解其他基本操作的实现过程。

抽象数据类型可用三元组(D,S,P)表示,其中D是数据对象,S是D上的关系集,P是对D的基本操作集。常见的描述方式如下:

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采用数学符号和自然语言描述,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,并返回相应的出错信息。"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。

【例 1-2】 给出"简化线性表"的抽象数据类型的定义。

ADT Linear_list{

数据对象:所有数据元素a_i属于同一数据对象,$i=1,2,\cdots,n,n\geqslant 0$;

数据关系:所有数据元素$a_i(i=1,2,\cdots,n-1)$存在次序关系$\langle a_i,a_{i+1}\rangle$,$a_1$无直接前驱,$a_n$无直接后继;

基本操作:

① InitList(&L)

操作结果:构造一个空的线性表L。

② ListLength(L)

初始条件:线性表L已存在。

操作结果:返回L中数据元素个数。

③ GetElem(L,i,&e)

初始条件:线性表L已存在,$1\leqslant i\leqslant$ListLength(L)。

操作结果:用e返回L中第i个数据元素的值。

④ ListInsert(&L,i,e)

初始条件:线性表L已存在,$1\leqslant i\leqslant$ListLength(L)+1。

操作结果:在L中第i个位置之前插入新的数据元素e,L的长度加 1。

⑤ ListDelete(&L,i,&e)

初始条件:线性表L已存在且非空,$1\leqslant i\leqslant$ListLength(L)。

操作结果：删除 L 的第 i 个数据元素，并用 e 返回其值，L 的长度减 1

}ADT Linear_list

在上述定义中，数据元素所属的数据对象没有局限于一个具体的整型、实型或其他类型，所具有的操作也是抽象的数学特性，并没有具体到某一种计算机语言指令与程序编码。

1.2.2 抽象数据类型的表示与实现

抽象数据类型可通过固有数据类型来表示和实现，即利用处理器中已存在的数据类型来说明新的结构，用已经实现的操作来组合新的操作。本书将采用 C 语言作为描述工具来实现抽象数据类型，主要包括以下两个方面：

(1) 通过结构体将 int、float、char 等基本数据类型组合到一起，构成一个结构体类型，再用 C 语言中 typedef 自定义类型为该类型或该类型指针重新起一个名字，以增强程序的抽象性、简洁性和可读性。

(2) 用 C 语言的子函数实现各个基本操作。

若用 C 语言实现【例 1-2】给出的抽象数据类型“简化线性表 Linear_list”，可以用“一维数组”类型来描述顺序存储结构，或者用 C 语言提供的“指针”来描述链式存储结构。

①线性表的顺序存储结构的类型定义如下：

```
#define MAXSIZE 100          /*线性表存储空间大小*/
typedef struct{
    int elem[MAXSIZE];       /*存储线性表中元素的数组*/
    int length;              /*线性表当前的长度*/
}SqList;
```

②线性表的链式存储结构的类型定义如下：

```
typedef struct LNode{
    int data;                /*存储线性表中元素信息的数据域*/
    struct LNode *next;      /*存储直接后继存储位置的指针域*/
}LNode, *LinkList;
```

在定义好线性表的类型(顺序存储或链式存储)后，用 C 语言的子函数可以实现线性表的各个基本操作，详细的算法参见第 2 章。

1.3 算法和算法分析

著名的计算机科学家、图灵奖获得者 N. Wirth 教授曾专门出版了《数据结构＋算法＝程序》一书指出，程序是由数据结构和算法组成的，程序设计的本质是对要处理的问题选择好的数据结构，同时在此结构上施加一种好的算法。

对于一个程序来说，数据是“原料”。一个程序所要进行的计算或处理总是以某些数据为对象的。将松散、无组织的数据按某种要求组成一种数据结构，对于设计一个简明、高效、可靠的程序是大有益处的。

对求解一个问题而言，算法是解题的方法。没有算法，程序就成了无本之末，无源之水。算法在程序设计、软件开发甚至在整个计算机科学中的地位都是极其重要的。

1.3.1 算法的定义及其特性

算法是对特定问题求解步骤的一种描述,是指令的有限序列,其中每一条指令表示一个或多个操作。它有五大特性:有穷性、确定性、可行性、有输入、有输出。

① 有穷性。一个算法必须总是(对任何合法的输入值)在执行有穷步之后结束,且每一步都可在有穷时间(合理、可接受的)内完成。

② 确定性。算法中每一条指令必须有确切的含义,不会产生二义性。且在任何条件下,算法只有唯一的一条执行路径,即对于相同的输入只能得到相同的输出。

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

④ 有输入。一个算法有零个或多个的输入。

⑤ 有输出。一个算法有一个或多个的输出。

1.3.2 算法设计的要求

当用算法来求解一个问题时,算法设计的目标是正确、可读、健壮、高效、低耗。通常作为一个"好"的算法,一般应具有以下几个基本特征:

(1) 正确性

算法的正确性是指算法应满足具体问题的求解需求。其中"正确"的含义可以分为以下四个层次:

① 算法所设计的程序没有语法错误。

② 算法所设计的程序对于几组输入数据能够得到满足要求的结果。

③ 算法所设计的程序对于精心选择的典型、苛刻且带有刁难性的几组输入数据能够得到满足要求的结果。

④ 算法所设计的程序对于一切合法的输入数据都能产生满足要求对结果。

显然,达到第四层意义下的正确是极为困难的,所有不同输入数据的数量大得惊人,逐一验证的方法是不现实的。一般情况下,通常以第三层意义的正确性作为衡量一个算法是否正确的标准。

(2) 可读性

一个好的算法首先应便于人们理解和相互交流,其次才是机器可执行。可读性好的算法有助于人对算法的理解;晦涩难懂的算法易于隐藏较多的错误,难以调试和修改。

(3) 健壮性(鲁棒性)

即对非法输入的抵抗能力。当输入数据非法时,算法也能适当地做出反应或进行处理,而不会产生莫名其妙的输出结果或陷入瘫痪。

(4)高效率和低存储量

算法的效率通常是指算法执行的时间。对于同一个问题如果有多个算法可以解决,执行时间短的算法效率高。存储量的需求是指算法执行过程中所需要的最大存储空间。效率与存储量需求这两者都与问题的规模有关。

1.3.3 算法的分析

通常对于一个实际问题的解决,可以提出若干个算法,如何从这些可行的算法中找出最