普通高等教育“十二五”规划教材
高等院校计算机系列教材

数据结构（C语言版）

陈倩诒　邓红卫 ◉ 主编

华中科技大学出版社
http://www.hustp.com

普通高等教育“十二五”规划教材
高等院校计算机系列教材

数据结构(C语言版)

主　编　陈倩诒　邓红卫
副主编　肖增良　许建国　王自全
　　乐晓波　黄　薇

华中科技大学出版社
中国·武汉

内 容 提 要

“数据结构”是计算机及相关专业的专业基础核心课程。本书所有算法都采用C语言描述，书中不仅讲解了数据结构的基本理论知识，还提供了大量实例来帮助读者理解和掌握知识点。全书共分9章，内容包括绪论、线性表、栈和队列、串、数组与广义表、树与二叉树、图、查找、内部排序等，每章都对相关数据结构的逻辑结构、存储结构、基本操作、综合算法等做了全面、深入的阐述。

本书各章内容翔实，算法和例题典型，实践性强，可作为本、专科院校的计算机及相关专业“数据结构”课程的教材，也可作为计算机软件开发人员、参加硕士研究生入学考试和软件资格(水平)考试人员的参考书。

图书在版编目(CIP)数据

数据结构(C语言版)/陈倩诒　邓红卫　主编. —武汉：华中科技大学出版社，2013.1
ISBN 978-7-5609-8338-7

Ⅰ.数…　Ⅱ.①陈…　②邓…　Ⅲ.①数据结构-高等学校-教材　②C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(2012)第205115号

数据结构(C语言版)　　陈倩诒　邓红卫　主编

责任编辑：朱建丽
封面设计：范翠璇
责任校对：张　琳
责任监印：周治超
出版发行：华中科技大学出版社(中国·武汉)
　　武昌喻家山　　邮编：430074　　电话：(027)81321915
录　　排：华中科技大学惠友文印中心
印　　刷：华中科技大学印刷厂
开　　本：787mm×1092mm　1/16
印　　张：16
字　　数：392千字
版　　次：2013年1月第1版第1次印刷
定　　价：29.80元

高等院校计算机系列教材

编　委　会

前　言

“数据结构”是计算机及相关专业的专业基础核心课程。

在计算机科学的各领域中，经常要使用到各种不同的数据结构，例如，操作系统中要使用队列、目录树等；数据库系统中要使用线性表、索引树等；编译系统中要使用栈、哈希表、语法树等；人工智能中要使用广义表、检索树、有向图等。因此，在掌握了各种常用数据结构，能对算法进行时间复杂度和空间复杂度分析后，便可知道在何种情况下使用何种数据结构最方便有效，从而为以后研究和开发大型程序打下坚实基础。由此可见，学好数据结构，对从事计算机技术及相关领域工作的人员来说，是多么重要。

数据结构主要是研究现实世界中的各种数据（数字、字符、声音、图形、图像等）的逻辑结构，在计算机中的存储结构及相关算法；分析针对同一问题的各种不同算法的优劣。学生在学习该课程后，将具备用各种数据结构来解决实际问题及评价算法优劣的能力，为后续计算机专业课程的学习打下坚实基础，可以说这门课程起到了一个承上启下的关键作用。

本书涵盖了硕士研究生入学考试中数据结构考试大纲所规定的考试内容。全书共分为9章，第1章，介绍数据结构与算法等一些基本术语，并对算法描述及算法分析做了简单说明，介绍衡量算法优劣的时间复杂度和空间复杂度；第2章至第5章，介绍线性结构（线性表、栈、队列、串、数组）的逻辑结构、物理存储结构、常用算法的实现及基本应用；第6章至第7章，介绍非线性结构（树、二叉树、图）的逻辑结构、物理存储结构、常用算法的实现及基本应用；第8章至第9章，介绍在计算机中使用非常广泛的两种运算，即查找和排序，对多种常用的查找、排序方法进行详细说明，并给出各种方法的实现算法及时间复杂度和空间复杂度的分析。各章内容相对独立，便于不同院校不同专业按需要组织教学。全书力求将数据结构理论知识与具体应用实例相结合，有助加深学生的理解和掌握。

本书采用常用的C语言作为算法的描述语言，形式上学生更容易理解和接受。编写本书的教师均为有讲授本课程十几年教学经验的教师，对数据结构中的各种算法都进行了认真的研究和分析，因此，本书中所选的例题和习题都具有一定的针对性和代表性。

百密一疏，在所难免，恳请读者提出好的意见和建议。

编　者

2012年7月

目　录

第1章 绪 论

本章主要知识点

- ❖ 数据结构的常用术语及其基本概念
- ❖ 集合、线性结构、树形结构、图形结构的逻辑特点
- ❖ 抽象数据类型
- ❖ 算法、算法描述及算法分析

1.1 引言

自1946年计算机诞生以来，计算机以迅猛的速度发展起来。迄今为止，计算机的应用已经不再局限于数值计算，而是全面地深入到人类社会的各个领域。除了数值计算外，人们更多地利用计算机进行控制、管理及数据处理等非数值计算的数据处理。要利用计算机对诸如字符、表格、图像等具有一定结构的数据进行处理，就要研究分析数据之间存在的关系，并考虑如何在计算机中有效地组织这些数据，从而有效地对数据进行各种处理。这就是“数据结构”这门课程要讨论的核心内容。

本章主要介绍数据结构中一些常用的术语，并介绍数据结构中常见的几种结构，即集合、线性结构、树形结构和网状结构(或称为图形结构)。同时，对一些基本概念做概括性的叙述，这些基本概念将贯穿课程的整个过程。

1.2 常用术语和基本概念

数据(Data)是人们利用文字符号、数字符号及其他规定的符号对客观现实世界的事物及其活动所做的抽象描述的信息，是计算机程序加工的“原料”。例如，某班甲同学的姓名为“张三”，乙同学的姓名为“李四”，就是关于班上同学姓名的描述。

表示一个事物的一组数据称为一个数据元素(Data Element)，它是数据的基本单位，在计算机中通常作为一个整体来进行考虑和处理。一般情况下，一个数据元素由若干个数据项(Data Item)构成。例如，描述一个学生信息的一个数据元素包含该学生的学号、姓名、性别、年龄、籍贯、通信地址等数据项，如表1.1所示。

表1.1 学生信息表

学 号	姓 名	性 别	年 龄	籍 贯	通 信 地 址
01	张三	男	21	长沙	长沙市韶山路129号
02	李四	男	23	北京	北京市西单235号

续表

学　号	姓　名	性　别	年　龄	籍　贯	通信地址
03	王五	女	22	上海	上海市南京路19号
04	赵六	男	24	武汉	武汉市解放路276号

数据对象(Data Object)是性质相同的数据元素的集合,是数据的一个子集。例如,描述N个学生的有关信息的N个数据元素构成了一个数据对象。

数据类型(Data Type)是一组性质相同的值的集合,以及定义在这个集合上的一组操作的总称。例如,高级语言中用到的整数数据类型,是指某个区间(如[-32768,32767])上的整数构成的集合及定义在这个集合上的一组操作(加、减、乘、除、乘方等)的总称。

抽象数据类型(Abstract Data Type)通常是指由用户定义、用于表示实际应用问题的数据模型,一般由基本数据类型或其他已定义的抽象数据类型及定义在该模型上的一组操作组成。在C或C++语言中,一般可用struct或直接用"类"来定义抽象数据类型。抽象数据类型的定义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。

在本书中,在具体使用C语言描述某种抽象数据类型之前,将采用如下ADT格式来定义该抽象数据类型。

```
ADT  抽象数据类型名{
数据对象:<数据对象D的定义>
数据关系:<数据关系的集合R的定义>
基本操作:<基本操作的集合P的定义>
} ADT抽象数据类型名
```

例如,下面是一个关于抽象数据类型"三元组"的ADT格式。

```
ADT triplet {
数据对象:
    D={e1,e2,e3 | e1,e2,e3∈ElemSet(ElemSet为某数据元素的集合)}
数据关系:
    R={R1}
    R1={<e1,e2>,<e2,e3>}
基本操作:
(1) InitTriplet(&T,v1,v2,v3)
        操作结果:构造三元组T,分别把参数v1、v2、v3的值赋予元素e1、e2、e3
(2) DestroyTriplet(&T)
        操作结果:销毁三元组T
(3) Get(T,i,&e)
        初始条件:三元组T存在,1≤i≤3
        操作结果:用e返回T的第i元的值
(4) Put(&T,i,e)
        初始条件:三元组T存在,1≤i≤3
        操作结果:改变T的第i元的值为e
```

(5) IsAsending(T)

初始条件:三元组 T 存在

操作结果:如果 T 的三个元素按升序排列,则返回 1,否则返回 0

(6) IsDsending(T)

初始条件:三元组 T 存在

操作结果:如果 T 的三个元素按降序排列,则返回 1,否则返回 0

(7) Max(T,&e)

初始条件:三元组 T 存在

操作结果:用 e 返回 T 的三个元素中的最大值

(8) Min(T,&e)

初始条件:三元组 T 存在

操作结果:用 e 返回 T 的三个元素中的最小值

}

数据结构(Data Structure)是指相互之间存在一种或多种特定关系的数据元素的集合。具体来说,数据结构包含三个方面的内容,即数据的逻辑结构、数据的存储结构(也称为物理结构)和对数据所施加的一组操作。

数据的逻辑结构是数据元素之间本身所固有的独立于计算机的一种结构,这种结构可以用数据元素之间固有的关系的集合来描述。

数据的存储结构是逻辑结构在计算机存储器中的具体存放方式的体现,是逻辑结构在计算机存储器中的映像。

操作是指对数据对象进行处理的一组运算或处理的总称。操作的定义直接依赖于逻辑结构,但操作的实现必须依赖于存储结构。

根据数据元素之间存在关系的不同特性,数据结构通常可以分为如下 4 类基本结构。

(1) 线性结构。其数据元素之间存在一对一的线性关系,即除了第一个元素和最后一个元素外,每个元素都有一个直接前驱和一个直接后继,第一个元素没有前驱,最后一个元素没有后继,其示意图如图 1.1 所示。

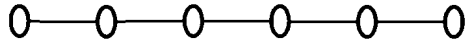

图 1.1 线性结构示意图

(2) 树形结构。其数据元素之间存在着一对多的关系。例如,老师 T 指导 3 个硕士研究生 G_1、G_2、G_3,每个研究生 G_i 又分别指导 3 个本科生 S_{i1}、S_{i2}、S_{i3},则数据元素之间呈现树形结构,如图 1.2 所示。

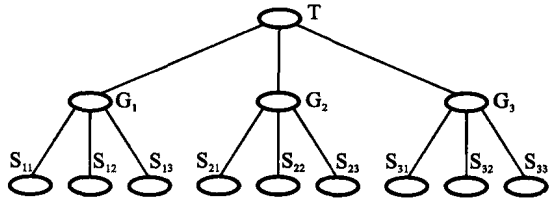

图 1.2 树形结构示意图

(3) 图形结构(或称为网状结构)。其数据元素之间存在多对多的关系,其示意图如图1.3所示。

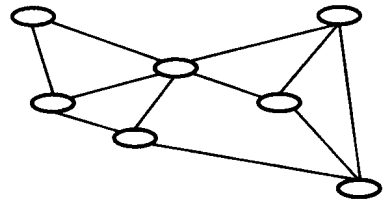

图1.3 图形结构示意图

(4) 集合结构。集合结构的数据元素之间无任何关系,其示意图如图1.4所示。

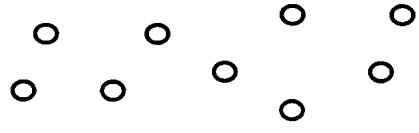

图1.4 集合结构示意图

1.3 算法与算法分析

算法(Algorithm)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,每条指令表示一个或多个操作。

算法分析(Algorithm Analysis)是指从"时间"和"空间"两个方面来分析算法效率的过程。

1.3.1 算法的重要特性

(1) 输入性:具有零个或若干个输入量,这些输入量取自于某个特定的对象集合。

(2) 输出性:具有一个或多个输出量,且当有一个以上的输入量时,该输出依赖于一组确定的输入量,即某一组确定的输入量有唯一的输出量与之对应。

(3) 有限性:构成算法的指令序列一定是有限序列。

(4) 确定性:算法中的每条指令必须有确切的含义,无二义性。

(5) 可行性:算法中所描述的操作都可以用已有的有限条指令或已实现的有限个基本运算来实现。

1.3.2 算法设计的基本要求

(1) 正确性:算法应满足具体问题的需求,这是算法设计的基本目标。

(2) 可读性:算法的可读性好有助于人们对算法的理解,便于系统设计人员之间的合作与交流,也便于对算法相应的程序进行调试和查错。

(3) 健壮性:当输入非法数据时,算法要能做出适当处理,而不至于产生莫名其妙的输出结果。

(4) 高时间效率:算法的时间效率是指算法的执行时间效率。对同一问题而言,执行时间越短的算法,其时间效率越高。

(5) 高空间效率:算法在执行时一般需要额外的存储空间。对同一问题而言,如果有多个算法可供选择,则存储空间要求较低的算法具有较高的空间效率。

1.3.3 算法的描述方法

算法的描述方法有很多种,本书将采用C语言来描述算法。用C语言来描述算法要遵循如下规则。

1. 所有算法的描述都用C语言中的函数形式

函数的格式为

```
函数类型  函数名(形参与类型说明)
{  函数语句部分
   return(表达式值);
}
```

2. 函数中形参的两种传值方式

若形参为一般变量,则形参为单向传值参数;若形参前面增加"&"符号,则形参为双向传地址参数。

例如,函数 void swap(&i,&j,k),其中,i、j 均为双向传地址参数,k 为单向传值参数。

3. 函数的说明部分与函数的实现部分分离

当C语言用函数来描述算法时,为使之与面向对象程序设计相匹配,一般将函数的说明部分与函数的实现部分分离开来。

4. 输入函数

C语言中的输入函数的调用格式为

```
scanf("格式控制字符串",地址列表)
```

其中,格式控制字符串的作用是指定输入格式。地址列表用于列出各变量的地址,地址是由地址运算符"&"与变量名组成的。

5. 输出函数

C语言中的输出函数的调用格式为

```
printf("格式控制字符串",输出列表)
```

其中,格式控制字符串的作用与 scanf 函数中的类似,输出列表用于列出输出对象。

6. 变量的作用域

在C语言中,每个变量都有一个作用域。在函数内部声明的变量,仅在函数内部有效。在整个程序中都有效的变量称为全局变量,否则,称为局部变量。

1.3.4 算法分析

算法分析包含两部分工作,即对算法的时间效率的分析和对算法的空间效率的分析。时间效率用时间复杂度来度量,空间效率用空间复杂度来度量。

1. 语句频度

一个算法执行所耗费的时间,从理论上说应当与该算法中所有语句的执行总次数成正比。算法中的所有语句的执行总次数称为该算法的语句频度,记为 T(n)。

例 1.1 求下列算法段的语句频度。

```
for(i=1;i<=n;i++)          //(n+1)次
  for(j=1;j<=i;j++)        //(2+3+…+n+1)次
    x=x+1;                 //(1+2+3+…+n)次
```

解 该算法采用了二重循环,其外循环的 for 语句执行(n+1)次;外循环的 for 语句每执行一次将使内循环的 for 语句执行(i+1)(i=1,2,3,…,n)次,而 x=x+1 共要执行(1+2+3+…+n)次。所以语句频度为

$$T(n)=(n+1)+(2+3+\cdots+n+n+1)+(1+2+3+\cdots+n)=n+1+n(n+3)/2+n(n+1)/2$$

2. 时间复杂度

在语句频度 T(n)中,n 称为问题的规模,当 n 不断变化时,语句频度 T(n)也随之变化。在一般情况下,当 n 不断增大时,我们希望知道 T(n)的变化呈现的规律。为此,引入时间复杂度的概念。

设 T(n)的一个辅助函数为 g(n),若有

$$\lim_{n\to\infty}\frac{T(n)}{g(n)}=A$$

其中,A 为一常数,则称 g(n)是 T(n)的同数量级函数。把 T(n)表示成同数量级函数的形式,即 T(n)=O(g(n)),则 O(g(n))称为算法的时间复杂度。时间复杂度描述了 n 无穷大时算法语句频度的数量级。

例如,若有 $T(n)=n+n(n+3)/2+n(n+1)/2$,则有

$$\lim_{n\to\infty}\frac{T(n)}{n^2}=1$$

所以时间复杂度为 $O(n^2)$。

常见的时间复杂度有 O(1)、O(n)、$O(n^2)$、$O(n^3)$,分别称为常量阶、线性阶、平方阶和立方阶。另外,算法的时间复杂度有时还可能呈现为对数阶 O(lb n)或指数阶 $O(2^n)$的情况。

3. 空间复杂度

算法的空间复杂度作为算法所需存储空间的度量,是指在算法执行过程中需要的辅助空间数量。而辅助空间是指除算法本身和输入/输出所占的空间外,算法在运行过程中临时开辟的存储空间。空间复杂度与时间复杂度的描述方法类似,记为 S(n)=O(f(n))。

空间复杂度的讨论方法与时间复杂度的讨论方法基本类似,在此不再赘述。

小　结

本章以绪论的形式出现,重点介绍如下知识点。

1. 数据结构

相互间存在一种或多种特定关系的数据元素的集合称为数据结构。数据结构又可细分