数据结构及其C语言实现

李少辉 郑志华 刘 丽 王 皓◎主编

- **化繁为简** 把抽象的问题具体化
- **图文并茂** 以图形、表格为全新的视角描述问题
- **主线分明** 以认知顺序为主线，循序渐进
- **结构严谨** 提出问题，分析问题，解决问题

北京邮电大学出版社
www.buptpress.com

数据结构及其C语言实现

李少辉 郑志华 刘 丽 王 皓◎主编

- 化繁为简 把抽象的问题具体化
- 图文并茂 以图形、表格为全新的视角描述问题
- 主线分明 以认知顺序为主线，循序渐进
- 结构严谨 提出问题，分析问题，解决问题

内 容 简 介

本书共分 9 章，包括数据结构基础、线性表、栈和队列、串、数组与广义表、树、图、查找、内部排序。本书以每种数据元素的数据描述、数据元素之间的关系、对该数据元素的主要操作、C 语言实现为主线进行编写，每一章都设置了大量的习题，方便读者对所学内容的掌握。该书结构清晰、易教易学、实例丰富、可操作性强、注重能力，对在学习过程中常见的重点和难点进行立体、详细的讲解，以帮助读者更好地掌握数据结构的基本知识。

本书适合作为高等院校计算机及相关专业本、专科学生教材，也适合数据结构的初学者研读和考研复习之用，还可作为从事计算机软件开发和应用研究人员的参考书。

图书在版编目(CIP)数据

数据结构及其 C 语言实现 / 李少辉等主编 . -- 北京 : 北京邮电大学出版社，2015.1

ISBN 978-7-5635-4272-7

Ⅰ. ①数… Ⅱ. ①李… 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. ①TP311.12②TP312

中国版本图书馆 CIP 数据核字（2014）第 304426 号

书　　名：数据结构及其 C 语言实现

主　　编：李少辉　郑志华　刘　丽　王　皓

责任编辑：刘春棠

出版发行：北京邮电大学出版社

社　　址：北京市海淀区西土城路 10 号（邮编：100876）

发 行 部：电话：010-62282185　传真：010-62283578

E-mail：publish@bupt.edu.cn

经　　销：各地新华书店

印　　刷：北京鑫丰华彩印有限公司

开　　本：787 mm×1 092 mm　1/16

印　　张：17

字　　数：443 千字

印　　数：1—3 000 册

版　　次：2015 年 1 月第 1 版　2015 年 1 月第 1 次印刷

ISBN 978-7-5635-4272-7　　定　价：35.00 元

· 如有印装质量问题，请与北京邮电大学出版社发行部联系 ·

前　言

“数据结构”课程是计算机科学与技术、计算机通信及相关专业的专业基础课，是我国计算机教学中较早形成和完善的一门专业基础课程，也是计算机课程体系中的核心课程之一。通过这门课程的学习，学生能够掌握执行速度快、占用空间少、可靠性高、可读性好的程序的编写方法与技巧，在面对一个具体应用问题时，能选择最佳的逻辑结构、存储结构及实现算法，并能使用时间、空间复杂度来正确地评价算法。

多年来，在数据结构方面出版了许多很好的教材，这些教材一般都各有特色并在各高校的计算机教学中发挥了重要的作用。随着计算机技术的飞速发展，程序设计方法及软件开发技术也出现了重大的发展及变革，这对各专业课程的教材更新提出了新的要求。为了适应程序设计方法的变革，我们编写了本书。本书讲解时始终贯穿由浅入深、易于理解的宗旨，对于重要和复杂的概念，讲述时配有例题进行示范，并使用 C 语言实现了关键的算法；在文字表达上力求简练清楚、概念清晰、通俗易懂；实现的算法具体，易于调试；描述的数据结构和算法结构清晰、可读性高、符合软件工程的规范，学生的学习过程也就是复杂程序的设计过程。本书的算法使用类 C 语言进行描述，配有相关的解释，以帮助读者理解。

本书共分 9 章，包括数据结构基础、线性表、栈和队列、串、数组与广义表、树、图、查找、内部排序。本书以每种数据元素的数据描述、数据元素之间的关系、对该数据元素的主要操作、C 语言实现为主线进行编写，每一章都设置了大量的习题，方便读者对所学内容的掌握。该书结构清晰、易教易学、实例丰富、可操作性强、注重能力，对在学习过程中常见的重点和难点进行立体、详细的讲解，以帮助读者更好地掌握数据结构的基本知识。

本书以培养、提高学生的基本专业素质及综合应用能力为目标，注重体现以下特色。

1. 简易性。采用深入浅出的讲解方法，对于重要的、比较难的知识点，采用不同的角度进行讲解，力求简单易懂。采用的问题解决方案也具有较好的可靠性、可维护性和可复用性。

2. 实用性。在本教材中设置了大量的习题，每一节中的习题可以帮助学生更好地理解该节中的知识点；每一章中的习题有助于学生课后进行练习、巩固。

3. 适应性。在本教材中对每一种数据类型都有比较规范的表述过程，对每一种算法都有比较规范统一的说明步骤，对算法的含义、参数与功能、工作变量说明、处理过程都进行明确的说明，并通过图示、文字注释、实例的执行过程等多种方式来帮助学生理解算法，提高学生的计算机思维能力。

在本课程学习结束时，希望学生能够彻底地理解数据类型及其不同数据类型的有关算法，并通过算法解决实际遇到的问题，为将来考研提供帮助，或为成为一名真正的程序员打下良好的基础。

本书第 1 章、第 2 章由李少辉编写，第 3～5 章由刘丽编写，第 6 章、第 7 章由郑志华编写，第 8 章、第 9 章由王皓编写，最后由李少辉进行统稿。

由于本书在总体策划及实现方法都做了一些新的尝试，加之作者水平有限，书中难免存在缺点与疏漏，敬请读者及同行批评指正。

目　录

此为试读，需要完整PDF请访问：www.ertongbook.com

第1章　数据结构基础

本章内容提要：

数据结构的基本概念、常用术语，数据结构发展的历史以及数据结构在计算机科学中的地位，数据结构描述语言，抽象数据类型（ADT），数据的存储结构，算法描述、分析及其复杂度问题。

当今社会被称为信息社会，在信息社会中，信息、知识成为重要的生产力要素，和物质、能量一起构成社会赖以生存的三大资源。信息的符号化被称为数据，是进行各种统计、计算、科学研究或技术设计等所依据的数值。有用的信息是对数据进行加工处理后得到的结果。现在社会已经进入大数据（big data）时代，指的是所涉及的信息量规模巨大到无法通过目前主流软件工具，在合理时间内达到撷取、管理、处理并整理成为帮助经营决策更积极目的的资讯。哈佛大学的社会学教授加里·金称："这是一场革命，庞大的数据资源使得各个领域开始了量化进程，无论学术界、商界还是政府，所有领域都将开始这种进程。"随着大数据应用的爆发性增长，它已经衍生出了自己独特的架构，而且也直接推动了存储、网络以及计算技术的发展。毕竟处理大数据这种特殊的需求是一个新的挑战。硬件的发展最终还是由软件需求推动的，我们很明显地看到大数据分析应用需求正在影响着数据存储基础设施的发展。随着结构化数据和非结构化数据量的持续增长以及分析数据来源的多样化，此前存储系统的设计已经无法满足大数据应用的需要，因此应该探寻新的数据结构以适应这些新的要求。

计算机是处理信息的机器。数据结构不仅研究这些信息的数学性质，也关心如何在计算机中有效地存储和处理这些信息。随着计算机信息量的增加、信息范围的拓宽和信息结构复杂化的加深，为了编写出高质量的程序，还必须分析这些信息的特征以及它们之间存在的关系。程序设计的实质就是为确定的问题选择一种适当的数据结构并设计一个好的算法。

本章介绍数据结构的基本概念与基本术语。要求熟练掌握这些概念和术语，为后续章节的学习与理解打下基础。

1.1　数据结构的基本概念

1.1.1　数据结构的产生和发展

数据结构是随着电子计算机的产生和发展而发展起来的一门计算机学科。最近20年来，电子计算机技术飞速发展，这不仅体现在计算机本身运算速度的不断提高、信息存储量日益扩大，而且更重要的是其应用范围的扩展。早期的电子计算机主要用于科学计算，所处理的对象

是纯数值性的信息。这类问题解题的算法比较复杂，但数据量较少，结构也简单，因此早期是以研究程序及所描述的算法为中心。目前，计算机已广泛地应用于情报检索、事务管理、系统工程等领域。与此相应，计算机加工处理的对象也从简单的纯数值性信息发展到文字、数字、图形、图像、音频、视频等各种复杂的、具有一定结构的数据。人们称前者为数值问题，而后者为非数值问题。非数值问题要求用复杂的数据结构来描述系统的状态，它们的运算是实现对数据结构的访问或修改。当前要设计出效率高、可靠性强的非数值程序，程序设计人员不但要掌握一般的程序设计技巧，而且还必须研究计算机程序加工的对象，即研究各种数据的特性以及数据之间的关系，这就促进了数据结构这一学科的发展。然而，数据必须在计算机中进行处理，因此不仅要考虑数据本身的数学性质，还必须考虑数据在计算机内的存储方式和相应的运算，从而扩大了数据结构研究的范围。随着数据库系统、情报检索系统的不断发展，在数据结构的技术中又增加了文件结构，特别是增加了大型文件的组织和树、图的知识，使得数据结构逐步成为一门比较完整的学科。“数据结构”是计算机各专业及其相关专业本、专科生必修的核心课程，是研究计算机程序设计的重要理论和技术的专业基础课程。

1.1.2 数据结构的基本概念

计算机的算法与数据结构密切相关，即每一个算法无不依赖于具体的数据结构，而数据结构直接影响着算法的选择和算法的效率。数据结构还必须给出适用于每种结构类型所定义的各种运算。所以，数据结构是一门研究程序设计中计算机操作的对象以及它们之间的关系和运算的一门学科。

数据结构是指数据以及数据相互之间的关系，可以看作是相互之间存在的某种特定关系的数据元素的集合，因此可以把数据结构看成是带结构的数据元素的集合。

1.1.3 常用术语

1. 数据(data)

计算机中的数据是广义的，包括日常生活中使用的数字、字符、字符串、表、文件、声音、图形、图像等。数据是人们利用文字符号、数字符号以及其他规定的符号对现实世界的事物及其活动所作的抽象描述，是对客观事物的符号表示。在计算机科学中其含义是指所有能够输入到计算机中并被计算机程序处理的符号集合，信息的符号化被称为数据。它是计算机处理信息的某种特定的符号表示形式。计算机解决问题的实质是对数据进行加工处理。

2. 数据元素(data element)

数据元素是数据集合中的一个实体，是计算机程序中加工处理的基本单位。在计算机程序中通常作为一个整体来考虑和处理。每个数据元素可以由若干个数据项组成，数据项的取值范围通常称为域(field)。例如，学生信息管理系统中，每个学生的情况(一条记录)为一个数据元素，而其中的学号、姓名、性别、年龄等信息分别称为数据项，年龄的取值范围只能是0～120岁，超出范围就会报错，这被称为数据的有效性，也被称为域。**数据项就是数据中不可再分割的最小单位**。数据元素在不同的数据结构里也被称为结点(node)、顶点(vertex)和记录(record)等。另外，**数据元素**是数据(集合)中的一个“个体”，**是数据的基本单位**。

3. 数据对象(data object)

数据对象是性质相同的数据元素的集合。它是数据的一个子集。数据对象可以是有限

的，也可以是无限的。

4. 数据结构(data structure)

同一数据对象中的数据不是孤立的，而是彼此相关的，是相互之间存在一种或多种特定关系的数据元素的集合。数据结构研究的是数据元素之间抽象化的相互关系和这种关系在计算机中的存储表示。可以根据每种结构定义各自的运算，设计出相应的算法。数据结构与数据对象不同，在描述一种数据结构时，不但要描述数据对象，还要描述数据元素之间的相互关系。

5. 数据类型(data type)

数据类型是程序设计语言中各变量可取的种类。各种程序设计语言不仅规定了每种类型变量的取值范围，而且还规定了该类型变量能执行的运算。有些语言允许由内部类型构造出新的数据类型。可以把数据类型看作是程序设计语言中已经实现的数据结构。数据结构是在程序设计语言的基础上由用户建立起来的，它依靠语言提供的数据类型来描述数据的逻辑结构，也依靠语言提供的各种设施来定义、描述运算及其算法。这些运算按实际问题的需要由用户自己定义，而不是由语言系统事先规定。因此，数据类型和数据结构的主要区别是：前者面向系统，后者面向对象，是高一层的数据抽象。如果程序设计语言提供有抽象数据类型设施，则二者可以统一起来。

6. 抽象数据类型(abstract data type)

抽象数据类型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仅取决于它的一组逻辑特性，而与它在计算机内的表示和实现无关，即不论其内部结构如何变化，只要它的数学特性不变，就不会影响其外部的使用。在面向对象的程序设计(object oriented programming)技术中，数据和对数据的操作是融合在一起的。一般借助“对象”来描述抽象数据类型，一旦定义了一个对象，就可使用对象的名字来说明变量，调用其中的操作，从而实现信息的隐蔽和封装。抽象数据类型可以通过高级语言中已有的数据类型来表示和实现。数据结构的选择首先会从抽象数据类型的选择开始。

1.2 数据结构的研究对象

数据结构研究的内容可以包含以下三个方面。

1. 数据的逻辑结构(logical structure)

数据的逻辑结构主要用于描述数据元素之间的逻辑关系，是用户按使用需要建立起来，并呈现在用户面前的数据元素的结构形式。数据结构中所说的“关系”实际上是指数据元素之间的逻辑关系。根据**数据元素**之间的**关系**，有以下四类基本的结构。

(1) 集合(元素之间无密切关系，如稀疏矩阵)。

(2) 线性结构(元素之间有一对一的关系(1∶1)，如线性表)。

(3) 树形结构(元素之间有一对多(1∶n)的关系，如二叉树)。

(4) 图状结构或网状结构(元素之间有多对多(n∶m)的关系)。

其中数据元素之间的关系如图 1-1 所示。

线性表、堆栈、队列、串都可认为是线性结构，树和二叉树都是树形结构，而图则属于图状结构。也可以说，**数据元素之间的关系分为线性和非线性两类**。

2. 数据的存储结构

数据元素及其关系在计算机中的存储方式，即数据的存储结构，也称为数据的物理结构(physical structure)，是指数据在计算机内实际的存储形式。常见的数据存储结构包括顺序存储结构、链式存储结构、索引存储结构、哈希存储结构(也叫散列存储结构)。每种数据结构都可通过映像的方式得到相应的存储结构。常用的映像方式有两种：顺序映像和非顺序映像。由此，可得出两种不同的最常用的存储结构：顺序存储结构和链式存储结构。

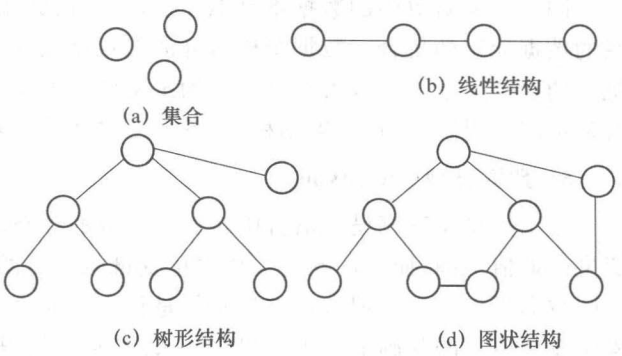

图 1-1　数据元素之间的关系

(1) 顺序存储结构(相当于 C 语言中的数组)：借助于数据元素的相对存储位置来表示数据元素之间的逻辑结构，把数据元素存储在一段连续的存储单元里，结点之间的关系由存储单元的关系来直接或间接反映。其主要特点如下。

① 结点中只有自身信息域，没有连接信息域，因此存储密度大(存储密度＝信息域字节数/结点总字节数)，存储空间利用率高。

② 存储空间是连续的，可以通过计算直接确定数据结构中任意一个结点的存储地址。

(2) 链式存储结构(相当于 C 语言中的指针)：借助指示元素地址的指针来指示逻辑上相邻的数据元素在存储器中的物理位置，因此可以把逻辑上相邻的两个元素存放在物理上不相邻的存储单元中。其主要特点如下。

① 结构中除自身信息外，还有表示连接信息的指针域，因此比顺序存储结构的存储密度小，存储空间利用率低。

② 存储空间可以是不连续的，因而更适合动态数据的管理。

通常把数据的逻辑结构简称为数据结构，将数据的物理结构称为存储结构。人们也常常用哈希和索引方式映像，从而可得到哈希存储结构和索引存储结构。与孤立的数据元素表示形式不同，数据结构中的数据元素不但要表示其本身的实际内容，还要表示清楚数据元素之间的逻辑结构。

3. 数据运算

施加在该数据结构上的操作，即数据运算。操作算法主要包括查找、插入、删除、修改(更新)、排序等。

在计算机科学或信息科学中，数据结构是计算机中存储、组织数据的方式。通常情况下，精心选择的数据结构可以带来最优效率的算法。

1.3　抽象数据类型

1.3.1　抽象数据类型的基本概念

抽象数据类型就是一个数学模型以及定义在该模型上的一组操作。作用是可以使我们更

此为试读，需要完整PDF请访问：www.ertongbook.com

容易描述现实世界。例如，用线性表描述学生成绩表，用树或图描述遗传关系。使用它的人可以只关心它的逻辑特征，不需要了解它的存储方式。定义它的人同样不必要关心它如何存储。

抽象数据类型主要采用三元组表示：(D,R,P)，其中 D 是数据对象，R 是 D 上的关系集，P 是对 D 的基本操作集。

定义格式为：

```
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
```

1.3.2 抽象数据类型的 C 语言实现

基本要求：利用抽象数据类型设计并实现一个可进行复数运算的演示程序。

要求：由输入的实部和虚部生成一个复数；从已知复数中分离出实部和虚部；对复数进行四则混合运算。

【解析】

抽象数据类型在 C 语言中可以使用结构体实现，具体的定义描述如下。

数据对象：由结构体类型定义一个复数 cpxNum。

数据关系：(1) 两个操作数 real 和 imag 具有序偶关系，real 表示实部，imag 表示虚部。

(2) 定义 c1、c2 两个复数，分别用 c1. real 和 c1. imag 表示 c1 的实部和虚部。

基本操作：(1) CreatComplexNumber (cpxNum * c,double a,double b)

操作结果：构造两个复数。

(2) cplus(cpxNum * c,cpxNum c1,cpxNum c2)

操作结果：实现两个复数加法，并输出结果。

(3) cminus(cpxNum * c,cpxNum c1,cpxNum c2)

操作结果：实现两个复数减法，并输出结果。

(4) cmultiply(cpxNum * c,cpxNum c1,cpxNum c2)

操作结果：实现两个复数乘法，并输出结果。

(5) cdivide(cpxNum * c,cpxNum c1,cpxNum c2)

操作结果：实现两个复数除法，并输出结果。

【C 语言实现】

```
#include <stdio.h>
#include <math.h>

//定义复数类型的存储结构
typedef struct {
double real; //复数的实部
double imag; //复数的虚部
}cpxNum; //定义结构体类型 cpxNum 表示"复数"

/* 用 double a, double b 初始化复数 c */
void CreateComplexNumber(cpxNum *c,double a,double b)
```

```
{
    c->real = a;
    c->imag = b;
}

/*实现两个复数 c1, c2 的加法，和作为结果输出*/
void cplus(cpxNum *c,cpxNum c1,cpxNum c2)
{
    c->real = c1.real + c2.real;
    c->imag = c1.imag + c2.imag;

printf("\n 数据和为：%3.2f%+3.2fi\n", c->real,c->imag);
}

/*实现两个复数 c1, c2 的减法，差作为结果输出*/
void cminus(cpxNum *c,cpxNum c1,cpxNum c2)
{
    c->real = c1.real - c2.real;
    c->imag = c1.imag - c2.imag;

printf("\n 数据差为：%3.2f%+3.2fi\n", c->real,c->imag);
}

/*实现两个复数 c1, c2 的乘法，积作为结果输出*/
void cmultiply(cpxNum *c,cpxNum c1,cpxNum c2)
{
    c->real = c1.real * c2.real - c1.imag * c2.imag;
    c->imag = c1.real * c2.imag + c1.imag * c2.real;

printf("\n 数据积为：%3.2f%+3.2fi\n", c->real,c->imag);
}

/*实现两个复数 c1, c2 的除法，输出结果*/
void cdivide(cpxNum *c,cpxNum c1,cpxNum c2)
{
    double result_real,result_imag;
    c->real = 1/(pow(c2.real,2) + pow(c2.imag,2)) * ( c1.real * c2.real + c1.imag * c2.imag);
    c->imag = 1/(pow(c2.real,2) + pow(c2.imag,2)) * (c1.imag * c2.real - c1.real * c2.imag);

printf("\n 结果为：%3.2f%+3.2fi\n",c->real,c->imag);
}

void main( )
{
    cpxNum c1,c2,c; //声明两个复数类型的变量 c1 和 c2,c 用来表示运算结果
    double a,b; //声明 2 个双精度数,用于接收复数的实部和虚部
    char e; //用于显示复数虚部的正负号
    int number;

    /*输入两个复数*/
    printf("输入第一个复数 \n ");
    printf("(如果你输入复数的虚部为负数,则在输入虚部之前输入空格): \n");
    scanf("%lf%c%lf",&a,&e,&b);
```

```
    CreateComplexNumber(&c1,a,b);
    printf("你输入的第一个数是：%2.1f%c%2.1fi\n",a,e,b);

    printf("输入第二个复数 \n");/
    printf("(如果你输入复数的虚部为负数,则在输入虚部之前输入空格)：\n");
    scanf("%lf%c%lf",&a,&e,&b);
    CreateComplexNumber(&c2,a,b);
    printf("你输入的第二个数是：%2.1f%c%2.1fi\n",a,e,b);

    /*调用复数加减乘除函数,并在每个函数中输出运算结果*/
    printf("选择一种运算,1为加法,2为减法,3为乘法,4为除法,5为输出所有结果:");
    scanf("%d",&number);

    switch(number)
    {
        case 1:
              cplus(&c,c1,c2);
              break;
        case 2:
              cminus(&c,c1,c2);
              break;
        case 3:
              cmultiply(&c,c1,c2);
              break;
        case 4:
              cdivide(&c,c1,c2);
              break;
        case 5:
              printf("\n-----复数运算结果-----\n");
              cplus(&c,c1,c2);
              cminus(&c,c1,c2);
              cmultiply(&c,c1,c2);
              cdivide(&c,c1,c2);
              break;
    }

}
```

1.4 数据结构与算法的关系

1.4.1 算法的定义

算法是解决某个特定问题的一种方法或一个过程，是指在解决问题时按照某种步骤一定可以得到问题的结果(有解时给出解，无解时给出无解的结论)的处理过程。简言之，算法就是计算机解决问题的步骤。当面临某个问题时，需要找到用计算机解决这个问题的方法和步骤，算法就是解决这个问题的方法和步骤的描述。所谓机械步骤是指，算法中有待执行的运算和操作，必须是相当基本的。换言之，它们都是能够精确地被计算机运行的算法，计算机甚至不需要掌握算法的含义，即可根据该算法的每一步骤要求，进行操作并最终得出正确的结果。算

法由操作、控制结构、数据结构三要素构成。

计算机对数据的操作可以分为数值性和非数值性两种类型。在数值性操作中主要进行的是算术运算；而在非数值性操作中主要进行的是检索(查找)、排序、插入、删除、更新(修改)等。

设计算法的基本过程如下。

第一步，通过对问题进行详细的分析，抽象出相应的数学模型。

第二步，确定使用的数据结构，并在此基础上设计对此数据结构实施各种操作的算法。

第三步，选用某种语言将算法转换成程序。

第四步，调试并运行这些程序。

1.4.2 算法分析

算法分析的主要任务是对设计出的每一个具体的算法，利用数学工具，讨论其复杂度。对算法的分析一方面能深刻地理解问题的本质以及可能的求解技术，另一方面可以探讨某种具体算法适用于哪类问题，或某类问题宜采用哪种算法。算法分析就是研究算法从而达到优化计算机解决问题的效率的目的。

对算法的分析和评价一般应考虑正确性、可维护性、可读性、运算量、占用存储空间等诸多因素。其中评价算法的三条主要标准如下。

(1) 算法实现所耗费的时间。

(2) 算法实现所耗费的存储空间，其中主要考虑辅助存储空间。

(3) 算法应易于理解，易于编码，易于调试等。

1.4.3 数据结构与算法的联系

数据结构与算法关系密切，两者既有联系又有区别，下面就这两个方面分别进行讨论。

1. 数据结构与算法的联系

程序＝算法＋数据结构。数据结构是算法实现的基础，算法总是要依赖于某种数据结构来实现的。往往是在发展一种算法的时候，构建了适合于这种算法的数据结构。例如针对当前的大数据，就修改了基于块和文件的存储系统的架构设计以适应这些新的要求。

算法的操作对象是数据结构。算法的设计和选择要同时结合数据结构，简单地说数据结构的设计就是选择存储方式，如确定问题中的信息是用数组存储还是用普通的变量存储或其他更加复杂的数据结构。算法设计的实质就是对实际问题要处理的数据选择一种恰当的存储结构，并在选定的存储结构上设计一个好的算法。不同的数据结构的设计将导致差异很大的算法。数据结构是算法设计的基础。

用一个形象的比喻来解释：采金过程中，金子以各种形式深埋于地下。金矿石的结构就相当于计算机领域的数据结构，而金子就相当于一个个数据元素。开采金矿然后运输、加工这些“操作”技术就相当于算法。显然，如何开采、如何运输必须考虑到金矿的存储(物理)结构，只拥有开采技术而没有金矿是没有任何意义的。算法设计必须考虑到数据结构，算法设计是不可能独立于数据结构的。

另外，数据结构的设计和选择需要为算法服务。如果某种数据结构不利于算法实现它将没有太大的实际意义。知道某种数据结构的典型操作才能设计出好的算法。

总之，算法的设计同时伴有数据结构的设计，两者都是为最终解决问题服务的。一种数据结构如果脱离了算法，那还有什么用呢？实际上也不存在一本书单纯地讲数据结构，或者单纯

地讲算法。当然两者也是有一定区别的，算法更加抽象一些，侧重于对问题的建模，而数据结构则是具体实现方面的问题了，两者是相辅相成的。

2. 数据结构与算法的区别

数据结构关注的是数据的逻辑结构、存储结构以及基本操作，而算法更多的是关注如何在数据结构的基础上解决实际问题。算法是编程思想，数据结构则是这些思想的逻辑基础。

1.4.4 数据结构的主要运算

数据结构的主要运算包括以下几种。

(1) 建立(create)一个数据结构的运算。

(2) 消除(destroy)一个数据结构的运算。

(3) 从数据结构中删除(delete)一个数据元素的运算。

(4) 把一个数据元素插入(insert)到一个数据结构中的运算。

(5) 对一个数据结构进行访问(access)的运算。

(6) 对一个数据结构进行修改(modify)的运算。

(7) 对一个数据结构进行排序(sort)的运算。

(8) 对一个数据结构进行查找(search)的运算。

以上是主要的常见运算，还有一些其他运算，当用到时我们再逐一进行介绍。

1.4.5 算法的特性

算法是执行特定计算的有穷过程，该过程有下述几个特性。

(1) 有穷性：当执行一个算法时，不论是何种情况，在经过了有限步骤后，该算法一定会终止。

(2) 确定性：算法中的每一条指令都必须是清楚的，执行算法时不会产生二义性。并且任何条件下，算法只有唯一的一条执行路径，即对于相同的输入有相同的输出。

(3) 可行性：算法是可行的，即算法中描述的操作都是可以通过已实现的基本运算的执行来完成。

(4) 输入：一个算法有零个或多个由外界提供的量。

(5) 输出：一个算法产生一个或多个输出。

1.4.6 算法的描述方法

描述算法的方法很多，本书使用类 C 语言来描述。由于 C 语言不是抽象数据类型的理想描述工具，所以从 C 语言选出一个核心子集，并添加了可用抽象数据类型的 C++的引用调用参数传递方式等，构成了类 C 语言。C 语言是面向过程的，类 C 语言是面向对象的，是可以继承、重载、多态的。类 C 语言与 C 语言相似但又有一些不同，这种语言是专门为某种具体的应用而仿照 C 语言开发的，比如，在无线传感网络界比较有名的由加州大学伯克利分校(Berkeley)开发的 TinyOS 系统就是用类 C 语言(NesC)来写的。类 C 语言精选了 C 语言的一个子集，同时作了一定扩充。为便于算法的理解，我们有时也用自然语言描述。

类 C 语言采用了标准 C 语言的语法结构，同时对一些语法细节进行了简化，并添加了一些描述方法。用类 C 写的代码是伪代码，因为不完全符合 C 语言的规范，所以不能被 C 编译器编译。本文采用的类 C 语言基本语法如下。

1. 预定义常量和类型

```
#define TRUE 1
#define FALSE 0
#define OK 1
#define ERROR 0
#define INFEASIBLE -1
#define OVERFLOW -2
typedef int Status; //Status 是函数的类型,其值是函数结果状态代码
```

2. 存储结构用类型定义(typedef)描述

数据元素(结点)的类型名约定为 ElemType。**注意**:这不是一种具体的类型名,在具体使用时,必须用具体的数据类型类代替 ElemType。

3. 操作算法用以下形式的函数描述

```
函数返回值类型 函数名(参数表){
      //对算法的说明文字
          函数语句序列
  }
```

4. 选择语句

(1) 条件句 1

```
if(条件表达式)语句 T;
```

(2) 条件句 2

```
if(条件表达式)语句 T;
else  语句 F;
```

(3) 开关语句

```
格式 1:
switch (表达式){
case 值 1: 语句序列 1; break;
case 值 2: 语句序列 2; break;
...
case 值 n: 语句序列 n; break;
default : 语句序列 n+1;
}
```

```
格式 2:
switch {
case 条件 1:语句序列 1;break;
case 条件 2:语句序列 2;break;
...
case 条件 n:语句序列 n;break; default:
语句序列 n+1;
}
```

5. 循环语句

(1) for 语句

```
for(赋初值句;条件;修改句)语句;
```

(2) while 语句

```
while(条件)语句;
```

(3) do...while 语句

```
do{
  语句序列;
  }while(条件);
```

6. 结束语句

(1) 函数结束语句

```
return; 或 return(表达式);
```


此为试读,需要完整PDF请访问: www.ertongbook.com