普通高等教育 电气信息类 应用型规划教材

数据结构——C++实现

（第二版）

缪淮扣　沈　俊　顾训穰　编著

科学出版社

免费提供电子教案

普通高等教育电气信息类应用型规划教材

数据结构——C++实现

（第二版）

缪淮扣　沈　俊　顾训穰　编著

科学出版社

北京

内 容 简 介

数据结构是计算机专业教学计划中的一门核心课程，也是信息管理、通信电子、自动控制等与计算机技术关系密切的专业的一门基础课程。从事与计算机科学与技术相关的工作，尤其是计算机应用领域的开发和研制工作，必须具备坚实的数据结构的基础。本书对 C++语言作了简单介绍，介绍了线性表、栈、队列、数组、广义表、树、图等数据结构，并介绍了查找和排序的方法。全书用 C++语言描述并实现了所有数据结构的类和程序，并附有习题，便于教学。

本书是为高等院校开设数据结构课程编著的教材，可作为计算机专业本科生教材使用，也可供从事计算机开发和应用的工程技术人员阅读、参考。

图书在版编目（CIP）数据

数据结构：C++实现/缪淮扣，沈俊，顾训穰编著. —2 版. —北京：科学出版社，2014

（普通高等教育电气信息类应用型规划教材）

ISBN 978-7-03-040739-9

I. ①数… II. ①缪…②沈…③顾… 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 109194 号

责任编辑：陈晓萍 / 责任校对：马英菊

责任印制：吕春珉 / 封面设计：耕者设计工作室

科学出版社 出版

北京东黄城根北街 16 号

邮政编码：100717

http://www.sciencep.com

铭浩彩色印装有限公司印刷

科学出版社发行 各地新华书店经销

*

2002 年 8 月第 一 版 开本：787×1092 1/16

2014 年 6 月第 二 版 印张：22

2014 年 6 月第九次印刷 字数：550 000

定价：44.00 元

（如有印装质量问题，我社负责调换〈骏杰〉）

销售部电话 010-62142126 编辑部电话 010-62138978-2009

前　言

作为计算机程序组成部分的数据结构和算法的研究，一直受到计算机领域工作者的高度重视。数据结构是计算机专业教学计划中的一门核心课程，也是信息管理、通信电子、自动控制等与计算机技术关系密切的专业的一门基础课程。

要从事与计算机科学与技术相关的工作，尤其是计算机应用领域的开发和研制工作，必须具备坚实的数据结构的基础。

数据结构课程的教学目的是使学生学会分析研究计算机所要加工处理的数据的特征，掌握组织数据、存储数据和处理数据的基本方法，并加强在实际应用中选择合适的数据结构和相应算法的训练。

面向对象技术是软件工程领域中的重要技术，它不仅是一种程序设计方法，更重要的是一种对真实世界的抽象思维方式。目前，面向对象的软件分析和设计技术已发展成为软件开发的主流方法。为了适应软件开发方法与技术的发展以及应用领域的要求，就有必要改进和充实数据结构的教学内容。因此，用面向对象的观点来描述数据结构就成为一种顺理成章的选择。

用面向对象的观点来描述数据结构，要涉及面向对象程序设计语言的选用问题。C 语言是一种广泛用于程序设计语言教学的语言，C++是以 C 语言为基础的、使用比较普遍的面向对象程序设计语言。因此本书采用了 C++作为数据结构的描述语言。

数据结构课程内容丰富，学习量大；隐藏在各部分内容中的方法和技术多，贯穿于全书的动态链表存储结构和递归技术令不少初学者望而生畏。本书的编写者长期从事数据结构课程的教学，对该课程的教学特点和难点有比较深切的体会。作者在认真总结近 30 年讲授数据结构课程的基础上参考了美国 ACM/IEEE CS 所颁布的《计算 2001 教程》，吸收了国内外相关数据结构教材的优点，对多年来形成的数据结构课程的教学内容进行了合理的剪裁，既强调了数据结构的原理和方法，又注重了其实践性，使之适应于现代大学生的学习特点和要求。

本书自 2002 年第一版出版以来，八次印刷，达 17000 余册，被国内近 20 所高校作为教材使用。本书在第一版的基础上对每一章的内容作了调整和修改，删减了对本科生来说不易理解的抽象数据类型的概念以及各种数据结构的抽象数据类型的描述，增加了部分实例。对各章的习题作了扩充和修改，习题的形式也更加丰富。

本书的一个重要特点就是将程序设计的基础与数据结构的方法尽可能地结合起来。第二章介绍 C++语言时尽可能给出比较完整的程序，使学生能对 C++语言有比较全面和深入的了解，也便于上机实习，从而为数据结构课程的实验建立良好的基础。

全书共九章，第一、二章介绍了数据结构、算法及其复杂度的基本概念，对 C++作了简单介绍。第三章至第五章介绍了线性结构——线性表、栈、队列、数组、广义表；第六章和第七章介绍了非线性结构——树和图；第八章和第九章分别介绍了查找和排序

的方法。

本书第一至第五章由缪淮扣编写，第六至第九章由沈俊编写。全书由缪淮扣统稿。顾训穰参与了本书第一版的编写工作，是本书第一版的作者之一。

本书可与作者编写的辅助教材《数据结构——C++实现 习题解析与实验指导》配套使用。

本书编写过程得到了国家自然科学基金项目（批准号：60970007、61170044）和上海市教育委员会第五期重点学科建设项目（编号：J50103）的支持；也得到了科学出版社、上海大学教务处和计算机学院的支持。在此致以诚挚感谢。

由于作者水平有限，书中难免有不妥之处，敬请广大读者批评指正。

缪淮扣

2014 年 3 月

目　录

第一章 绪 论

1.1 （算法+数据结构）= 程序

计算机能快速计算，能证明“四色定理”，会帮助工程师设计图纸，能帮助医生为病人诊断配药。计算机之所以如此神通广大，聪明能干，是人教给它本领，人是用“程序”来“教”计算机的。让计算机解题实际上就是为计算机编程序。随着计算机科学与技术的不断发展，计算机的应用领域也在不断扩大。“程序”越来越庞大，越来越复杂，因而解题的过程就不仅仅是编程序了，而是一个包括编程序在内的软件开发过程。该过程包括对用户的需求进行分析，对要开发的系统进行软件设计，然后编程序，再进行测试和改错，最后再运行等一系列的阶段。软件不仅仅指程序，而是包括程序以及开发程序的过程中所产生的各种文档。软件开发的目标是产生能让计算机有效工作的程序，因此程序是软件的核心。

程序到底是什么呢？计算机科学家，图灵奖获得者 N.Wirth 给出过一个著名的公式：算法+数据结构=程序。从这个公式可以看到，数据结构和算法是构成程序的两个同样重要的组成部分。这个公式在软件开发的进程中产生了深远的影响。但是它并没有强调数据结构与解题的算法是一个整体。

我们知道，非面向对象的过程语言，如 FORTRAN、C 和 PASCAL，其数据结构是问题解的中心。一个软件系统的结构是围绕一个或几个关键数据结构为核心而组成的。“算法+数据结构=程序”完全体现了这种思想。但随着软件系统的规模越来越大、复杂性不断增长，人们不得不对“关键数据结构”重新评价。在这种系统中，许多过程和函数（子程序）的实现严格依赖于关键数据结构，如果这些关键数据结构的一个或几个数据有所改变，则涉及整个软件系统，许多过程和函数必须重写，甚至因为几个关键数据结构的改变，导致软件系统整个结构彻底崩溃。

20 世纪 90 年代，面向对象的方法受到了很大的重视，并得到比较广泛的推广和应用。在面向对象程序设计中，密切相关的数据与过程被定义为一个整体（即对象），而且一旦作为一个整体定义了之后，就可以使用它，而无需了解其内部的实现细节，从而提高软件开发的效率。

封装和数据隐藏是面向对象问题解和面向对象程序设计的基本要素。它可以使软件

的许多维护问题简单化。一旦数据（类型）结构修改了，只要对实现模型的局部代码进行适当修改，软件系统的其余部分原封不动。

有人主张将 Wirth 的公式“算法+数据结构=程序”修改为“（算法+数据结构）=程序”，以体现面向对象的方法。

本书以面向对象的观点来介绍各种数据结构以及与这些数据结构有关的算法的知识。

第一章将介绍数据结构以及算法的基本概念，并介绍用来描述数据结构和算法的语言 C++。

1.2 数据结构的基本概念

计算机科学是一门研究信息表示和处理的科学，人们是用程序来处理信息的。信息的表示和组织直接关系到处理信息的程序的效率。由于许多系统软件和应用软件的程序规模很大，结构又相当复杂，因而必须对程序设计方法进行系统的研究。这不仅涉及研究程序结构和算法，同时也涉及研究程序加工的对象。

用计算机解题首先应从具体问题抽象出一个适当的数学模型，然后才能设计算法和编制程序。而构建数学模型的过程就是分析和概要设计的过程，要从对问题的分析中提取操作的对象，并找出这些操作对象之间的关系，然后用数学的语言加以描述。例如，许多工程中的数值计算问题采用的数学模型是线性方程组或微分方程。但更多的非数值计算问题是难以用数学方程来描述的。

1.2.1 两个简单的数据结构实例

为了获得对数据结构的感性认识，我们先来看两个简单的例子。

例 1-1 人事登记表。

在任何一个单位，人事登记表是人事部门关于职工信息的必不可少的表格。例如，当单位职工有几千名时，如果要查找当年退休的人员，或要查找基本工资在 800 元以下的人员，那么由人工来查找显然是很费时的。但如果在计算机中，有一个关于职工信息的登记表，由有关的检索系统来查找，那是很方便的。下面就是一张简单的人事登记表。

表 1-1 中，每个职工的信息放在一行中，称为一个数据元素（又称记录），所有职工的信息按照某一种顺序依次存放在表中。一般称由数据元素组成的这种线性表为文件。

类似这样表格在不同的计算机软件系统中是很多的，如学校的学生信息管理系统中的学生学籍登记表，图书馆管理系统中的图书目录表等。这类表格有一个共同的特性，就是各数据元素之间的关系是线性的关系，即一个元素的前面只有一个元素，后面也只有一个元素，第一个数据元素前面和最后一个数据元素后面没有元素。这样的一类表格就是一种数据结构，称为线性数据结构。

表 1-1 人事登记表

编号	姓名	性别	出生日期	婚否	基本工资
0001	王 军	男	1960/5/30	已	650
0002	李 平	女	1953/6/2	已	710
0003	周丽娟	女	1948/7/8	已	980
0004	赵忠良	男	1950/12/2	已	950
0005	张国庆	男	1978/10/1	未	500
⋮	⋮	⋮	⋮	⋮	⋮

例 1-2 一个典型的学校行政机构如图 1-1 所示。顶层结点“学校”代表整个系统，它的下一层结点代表这个系统的各个子系统，即部处与学院。再下一层结点代表更小的机构，如教务科、计算机系等，直到最底层一个小组或一个教研室等。在该图中，每一个结点代表一个数据元素，每一个结点的下面可能有多个结点。这样的一种结构称为层次型数据结构。

在各种应用程序中会涉及各种各样的数据，为了组织它们，存储它们，并且能对它们进行操作，就要考虑它们的归类以及它们之间的关系，从而建立相应的数据结构，并依此实现所要求的功能。

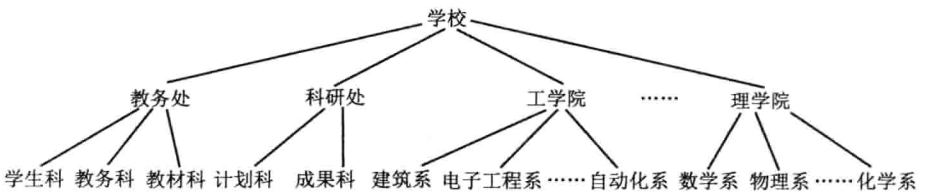

图 1-1 学校机构的“树结构”

1.2.2 数据结构的定义

一个水平很高的厨师，即使他可以把烹调某个菜肴的过程掌握得很好，但如果不给他原料，他也做不出色、香、味俱全的菜。这也就是人们常说的“巧妇难为无米之炊”。对一个程序来讲，数据就是“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。

大千世界中有各种各样的信息，如马路上的交通灯，进出地铁站的交通卡，股市投资者与证券商之间的交易，人们用语言交流的思想等。这些信息必须转换成数据才能在计算机中进行处理。让我们先来定义什么是数据以及与之相关的概念，然后再来回答“什么是数据结构”这个问题。

数据（data）是信息的载体，是描述客观事物的数、字符、图形、图像、声音以及所有能输入计算机中并被计算机程序识别和处理的符号的集合。

例如，一个解代数方程的程序处理的对象只是整数和实数，一个 C++语言的编译程

序处理的对象是由 ASCII 码字符组成的字符串等。

数据的基本单位是数据元素（data element）。一个数据元素可由若干个数据项（data item）组成，数据项是数据的最小单位。

为了解题的需要，具有相同特性的数据元素经常归类被称为数据对象的集合。数据对象是数据的子集。例如，所有的“数”构成了“数据”集合，而自然数集合={0，1，2，…}是“数”的数据对象；所有的字符是数据，字母集合 AS={A，B，…，Z，a，b，…，z}是该数据的数据对象。

在大多数情况下，一个数据元素不只是含有一个数据项，而是由多个数据项组成的。如对人事登记表的例子，一个职工的记录就是一个数据元素，它包括编号、姓名、性别、出生日期、婚否、基本工资六个数据项。

数据结构（data structure）的概念与数据的概念不同。若想描述一个数据结构，不仅要描述数据，而且还要描述诸数据元素之间的相互关系。

从以上两个例子可以看到，在实际应用中的各种数据元素都不是孤立存在的，它们之间存在着某种关系。这种数据元素之间的关系就是结构。根据数据元素之间关系的不同，数据结构分为两大类：线性结构和非线性结构。线性结构中各个数据元素依次排列成一个线性序列；而非线性结构中各个数据元素不再保持成为一个线性序列，每个数据元素可能与其他多个数据元素发生关系。这两类结构通常又可分为下列四类基本结构：集合，结构中的数据元素之间就是“同属于一个集合”，此外，没有其他关系；线性结构，结构中的数据元素之间存在的是一种线性关系，即一对一的关系，前面的例 1-1 给出的就是线性结构；树形结构，结构中的元素存在着一对多的关系，前面的例 1-2 给出的就是树形结构；图形结构或网状结构，结构中的元素之间存在着多对多的关系。图 1-2 描述了这四种不同结构的关系图：（b）描述的是线性结构，而（a）、（c）、（d）属于非线性结构。

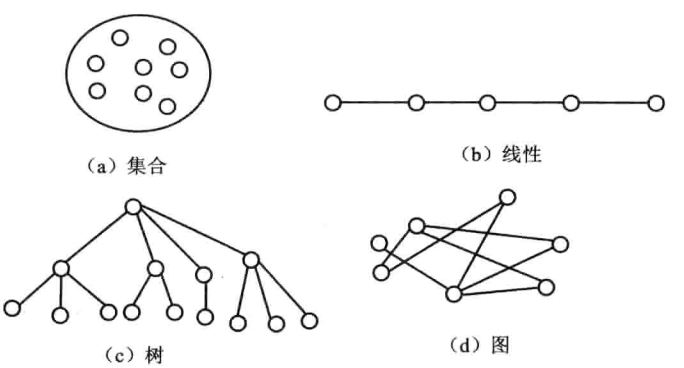

图 1-2　四种基本结构的关系图

对数据结构，又可按照其视点的不同分为数据的逻辑结构和物理结构。

数据的逻辑结构属于用户视图，是用户所看到的数据结构，是面向问题的。它描述的是数据元素之间的逻辑关系。数据的物理结构，又称存储结构，是数据的逻辑结构在计算机中的物理存储方式，它属于具体实现的视图，是面向计算机的。数据的逻辑结构

和物理结构是密切相关的两个方面。一般来说，算法设计是基于数据的逻辑结构，而算法实现则基于数据的物理结构。

本书在讨论各种数据结构时，既讨论其逻辑结构，又讨论其物理结构，还要定义其基本运算。

1.3 算法性能与复杂度

公元 825 年，一位名叫阿尔·花拉子米（Al-Khowarizmi）的波斯数学家写了一本教科书，书中概括了进行数字四则算术运算的法则，所有的数字都是用今天的印度十进制形式来表示的（按个、十、百位等排列，并有表示小数部位的小数点）。现代名词“算法”（algorithm）就来源于这位数学家的名字。美国 Webster's 字典中将算法定义为：解某种问题的任何专门的方法。在计算机科学里，算法这个词有一个专门的解释：算法——用计算机解题的精确描述。

本书在讨论各类数据结构的同时，介绍了对数据结构进行操作的算法，因此要对算法的概念作比较充分的讨论。

1.3.1 算法的定义

通常，人们将算法定义为一个用于实现某个特定任务的有穷指令集，这些指令规定了一个运算序列。一个算法应当具有以下特性。

1）输入性：一个算法必须具有零个或多个输入量。

2）输出性：一个算法应有一个或多个输出量，输出量是算法计算的结果。

3）确定性：算法中的每一条指令应含义明确，无歧义，即对每一种情况、需要执行的动作都应严格、清晰地规定。

4）有穷性：算法中的指令执行序列是有穷的，即算法无论在什么情况下都应在执行有穷步后终止。

5）有效性：每条指令必须是足够基本的。就是说，它们原则上都能精确到可用笔和纸来模拟实现其指令过程。对于每一步操作，仅有 3）中的确定性是不够的，它还必须是可行的。

需要指出的是，一个程序与一个算法对于上述 4）是有重大区别的。一个程序可以不满足特性 4）。例如，以用户名和口令来进行登录的系统就是一个处于“键入用户名、键入口令，检查是否已在系统中注册过并且匹配，如不满足，则重新输入”的循环之中，直至键入的用户名和口令匹配，这样的一个程序可能会不终止。但本书中所给出的程序均是可终止的，所以在本书中对“算法”和“程序”这两个术语不作严格的区分。

算法设计者在构思和设计了一个算法之后，必须准确清楚地将所设计的解题步骤记录下来，或提供交流，或编写成程序供计算机执行。

记录算法中的解题步骤又叫描述算法。常用的描述算法的方式有自然语言、流程图和程序设计语言等。

用汉语或英语这样的自然语言来描述算法，最大的优点是使用者不必对描述工具本身花精力去学习，对写出来的算法的理解是直接的。但其缺点是：容易出现二义性；语句一般太长，这就使得所建立的算法也显得冗长；算法中的分支及循环等结构表示不能清晰地显示出来。

使用规定式样的图形、指向线和文字说明组合起来的流程图方式来描述算法，其优点是直观、清晰、易懂，便于检查、修改和交流，其缺陷是严密性不如程序设计语言，灵活性不及自然语言；此外，对于大型的算法描述有困难。

用计算机程序设计语言描述算法显得清晰、明了，写出的算法一步到位，能由计算机处理。事实上，用程序设计语言来描述算法，就是对算法的实现。其缺点是抽象性差一些，可能会使写算法的人拘泥于计算步骤描述的细节，而忽略算法的实质。此外，必须熟练掌握程序设计语言及其编程技巧。

考虑到本书的使用者已经熟悉了像 C 这样的程序设计语言，并且掌握了程序设计的基本方法和技术，并因本书的内容是以面向对象的方法来讨论数据结构，因而采用 C++来描述算法。它的优点是类型丰富、语句精练，具有面向过程和面向对象的双重特点。此外，C++也支持抽象数据类型。为了使写出的算法可读性和可理解性更强，本书有时还采用了 C++语句与自然语言结合的方式来描述算法，有时对算法加以合适的注释。

1.3.2 算法的性能标准

在面向对象的程序中，通过类来实现数据结构，并且还要实现类中各个服务的算法。一旦确定了类的数据结构，就必须描述这些算法的设计和实现。

算法的设计主要有以下几个标准。

1）正确性：算法应确切地满足所要求解的问题的需求，这是最重要也是最基本的标准。

2）可用性：算法应能很方便地使用。为了便于用户使用，要求算法具有良好的界面，完备的用户文档。

3）可读性：算法应当是可读的，即易于理解的。一个算法不仅要被设计者自己读，而且也可能被他人读，不仅是编程、调试和测试时被读，而且在维护时也被读，因此，可读性是很重要的标准。

4）效率：算法的效率主要是指算法执行时存储单元的开销和运行时间的耗费，前者称为算法的空间代价，后者称为算法的时间代价。

5）健壮性：当输入非法数据时，算法应能作出适当的处理，而不应当产生不可预料的结果。这就要求算法中有对输入参数、打开文件、读文件记录以及子程序调用状态等操作的检错、报错和纠错等功能。

在设计一个算法时，上述的几条标准有时会有矛盾，如可用性强、可读性强会降低算法的效率。而对效率这个标准，算法的低时间代价和低空间代价也会产生矛盾。例如，有些问题若采用较多的内存空间可使时间代价降低，若采用较少的内存空间，则使时间代价提高。在计算机硬件价格快速下降的趋势下，算法的时间效率应首先予以考虑。