国家级优秀教学团队教学成果

数据结构实用教程

（C++版）

万 健 主 编

王立波 赵葆华 吴志芳 副主编

高等学校工程创新型"十二五"规划计算机教材

工程创新

Engineering Innovation

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

国家级优秀教学团队教学成果
高等学校工程创新型“十二五”规划计算机教材

数据结构实用教程
（C++版）

万 健 主编
王立波 赵葆华 吴志芳 副主编

電子工業出版社
Publishing House of Electronics Industry
北京 • BEIJING

内 容 简 介

本书为国家级优秀教学团队教学成果。

本书根据教育部高等学校计算机科学与技术教学指导委员会制定的《高等学校计算机科学与技术专业发展战略研究报告暨专业规范》编写，首先介绍了数据结构的核心基础知识——数据、数据类型、数据结构等基本概念和算法、算法的性能度量等知识，然后集中讨论了四种基本的数据结构——集合、线性表、树和图，同时介绍了栈、队列、串、数组以及广义表等数据结构，最后介绍了排序和查找的几种基础算法及实现（用C++语言）。

本书强调数据结构的工程应用，以模板的形式给出各种不同数据对象应用数据结构的多个实例，从而实现数据结构与工程应用的有机结合。

本书可以作为高等院校计算机及相关专业学生的教材，也可供培训机构及自学者参考。

图书在版编目(CIP)数据

数据结构实用教程：C++版 / 万健主编 —北京：电子工业出版社，2011.1
高等学校工程创新型“十二五”规划计算机教材
ISBN 978-7-121-11076-4

Ⅰ. ①数… Ⅱ. ①万… 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（2010）第 108448 号

策划编辑：章海涛
责任编辑：章海涛　　特约编辑：曹剑锋
印　　刷：北京市顺义兴华印刷厂
装　　订：三河市双峰印刷装订有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编　100036
开　　本：787×1092　1/16　印张：17.5　字数：446 千字
印　　次：2011 年 1 月第 1 次印刷
印　　数：4 000 册　　定价：32.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。
服务热线：（010）88258888。

前　言

数据结构的概念最早由 C.A.R.Hoare 于 1966 年提出。在他的经典论文《数据结构笔记》中，他首次系统地论述了一组数据结构的构造、表示和操作等问题。1973 年，D. E. Knuth 在《计算机程序设计技巧》第一卷中给出了关于“信息结构”的系统论述。1976 年，N.Wirth 用“算法+数据结构=程序”这个公式表达了算法与数据结构的联系和它们在程序设计中的地位。从此确立了数据结构在计算机相关专业中的核心基础课程地位。

“数据结构”是一门关于非数值数据在计算机中表示、变换及处理的课程。这里的“数据”实质是指计算机所能表示的各种不同数据对象的集合。对于每一具体的数据对象，其数据元素之间的关系都不是孤立的。数据元素之间的内在联系被称为结构。从数据元素之间的关系特征分析，各种数据对象的数据元素之间的关系仅呈以下四种结构之一：集合结构、线性结构、树型结构、图型结构。

“数据结构”课程的主要内容是针对以上四种结构，先从逻辑层面讨论结构的关系特征及抽象操作，再讨论结构在计算机中的存储表示（映像），并在存储表示的基础上给出相应结构的基本操作及实现，在此基础上讨论各种结构的应用。

目前的《数据结构》教材大体上按上述结构，在描述算法时大多采用类 C 或 Pascal 的算法描述语言。算法描述语言具有简洁和更关注于算法本身的优点，但对许多刚学完计算机程序设计语言、编程能力尚且不足的学生，将这些算法转变为可编译运行的程序时，会碰到一些困难。

在长期的教学过程中，我们认为，“数据结构”是一门兼具理论性与实践性的课程，在掌握程序设计语言后，本课程还是一门加强与提高学生程序设计能力的重要课程。因此，本书以传统的数据结构的主要内容为主线，在充分讨论结构的逻辑特征与存储表示的基础上，用 C++语言完成数据结构的描述和实现。同时，我们更加强调数据结构的应用，对不同的结构类型设计多个应用实例，每一算法或程序的编写力求高效、易读，并遵循程序设计的规范，从而帮助读者将数据结构与工程应用有机结合起来。

本书的另一个特点是将面向对象的方法引入数据结构领域。面向对象技术不仅是一种程序设计方法学，而且是一种认识方法学，“数据结构”讨论的正是数据的描述和处理，与面向对象的认知方法具有天然的联系。面向对象程序设计语言提供的封装、继承、多态和泛型程序设计等机制，为数据结构抽象数据类型的程序实现提供了很好的描述工具。本书以 C++为

程序设计语言，因此课程讲授时，可针对学生的具体情况，重点复习一下类与对象、运算符重载、模板、STL 等相关知识。

历经 30 多年的发展，“数据结构”课程的主要讨论范畴基本已取得共识。尽管计算机应用领域仍在不断地扩大并产生了许多新的数据结构和算法，但数据结构最基本和最核心的内容还是各种经典教材中反复强调的最具有代表性的那些知识。2006 年，教育部高等学校计算机科学与技术教学指导委员会编制了《高等学校计算机科学与技术专业发展战略研究报告暨专业规范》。其中，算法与数据结构涉及 AL1、AL2、AL3、AL4、AL5、PF2、PF3、PF4 等多个知识单元，知识点包括：递归、面向对象程序设计的基本理论、基本数据结构（包括堆栈、队列、链表、哈希表串、数组和广义表、树型结构及应用、图型结构及应用）、常用排序算法、常用查找技术、算法分析基础等。2009 年，教育部考试中心制定了全国硕士研究生入学统一考试关于“数据结构”科目的考试大纲。以上内容构成了我们编写本书的大纲依据。

本书的编写团队是国家级优秀教学团队的重要成员，在科学研究、工程软件开发方面具有长期的工作经验，特别是在长期的计算机专业教学中，针对教与学环节上的问题，形成了编写《数据结构》教材的共识。历经一年多的时间，编写团队反复讨论研究，精心选取案例，相信对读者会有所帮助。

本书的编写将做到“三个结合”的原则：与现代软件开发技术相结合、与工程应用背景相结合、与提高学生的程序设计能力相结合，以 C++为程序设计工具，避免传统的采用伪算法语言造成的与实际程序设计相脱节的现象，突出数据组织与算法的具体实现以及工程问题的数据结构表达，从而实现将“数据结构”回归为程序设计与软件开发基础课程的目标。

本书第 1 章由万健编写，第 2、4 章由王立波、吴志芳编写，第 3 章由赵葆华编写，第 5 章由周丽编写，第 6 章由徐翀编写，第 7 章由沈静编写，李卫明审阅并修改了全书的案例程序，任雪萍审阅了全书内容并编写了教学课件。全书的内容由上述编写者共同多次讨论定稿。

本书在编写过程中，得到了胡维华教授的关心和指导，并认真审阅了书稿，提出了很好的建议，在此表示衷心感谢！

本书的所有案例程序均在 Dev C++和 Visual C++下编译通过，程序说明见附录 A，并可在 http://www.hxedu.com.cn 上下载。

限于水平和时间，书稿虽经多次修改，但仍难免存在缺点与错误，恳请广大读者予以批评与指正。

作　者

目　录

第 1 章　绪论……1

1.1　数据与数据类型……1

1.1.1　数据……1

1.1.2　数据的计算机表示与数据类型……2

1.1.3　抽象数据类型……5

1.2　数据结构……7

1.3　算法与算法分析……10

1.3.1　算法……10

1.3.2　算法的性能分析与度量……11

1.3.3　算法的时间复杂度……11

1.3.4　算法的空间复杂度……13

习题 1……14

第 2 章　线性表……15

2.1　线性表的类型定义及结构特征……15

2.2　线性表类型的实现——顺序映像……17

2.3　线性表类型的实现——链式存储映像……26

2.3.1　单链表……26

2.3.2　其他形式的链表……35

2.4　线性表的应用……37

2.4.1　两个有序表的合并……38

2.4.2　集合运算……40

2.4.3　一元多项式的表示和相加……42

习题 2……47

第 3 章　其他线性结构……48

3.1　栈……48

3.1.1　栈的定义和基本操作……48

3.1.2　栈的存储结构及操作实现……49

3.1.3　栈的应用举例……55

3.2　队列……68

3.2.1　队列的定义和基本操作……68

3.2.2　队列的存储结构及操作实现……70

3.2.3　队列应用举例……77

3.3 串 ……83
3.3.1 串的基本概念和基本操作 ……83
3.3.2 串的存储结构 ……85
3.4 数组 ……87
3.4.1 数组的定义和基本操作 ……87
3.4.2 数组的存储表示 ……89
3.4.3 特殊矩阵的压缩存储 ……90
3.4.4 稀疏矩阵的压缩存储 ……92
3.5 广义表 ……96
3.5.1 广义表的基本概念和基本操作 ……97
3.5.2 广义表的存储结构 ……99
习题3 ……101
第4章 树 ……104
4.1 树、森林的定义及基本术语 ……104
4.2 二叉树 ……105
4.2.1 二叉树的结构定义 ……105
4.2.2 几种特殊形态的二叉树 ……106
4.2.3 二叉树的性质 ……107
4.2.4 二叉树的存储结构 ……108
4.2.5 二叉链表类的定义 ……110
4.2.6 二叉树的递归遍历 ……114
4.2.7 几个二叉树基本操作的例子 ……116
4.2.8 二叉树的非递归遍历 ……118
4.2.9 其他部分成员函数的实现 ……121
4.2.10 主函数（演示二叉链表类部分基本操作的执行结果） ……123
4.2.11 线索二叉树 ……126
4.3 树与森林的再讨论 ……129
4.3.1 树的存储结构 ……129
4.3.2 树和森林的遍历 ……132
4.4 树型结构的应用 ……134
4.4.1 算术表达式求值 ……134
4.4.2 树与等价问题 ……139
4.4.3 赫夫曼树及赫夫曼编码 ……145
习题4 ……154
第5章 图 ……156
5.1 图的定义和术语 ……156
5.2 图的存储结构 ……160
5.2.1 邻接矩阵表示法 ……160

5.2.2 邻接表表示法……162
5.2.3 十字链表表示法……164
5.2.4 邻接多重表表示法……165
5.3 图的基本操作……167
5.3.1 类的定义与实现……167
5.3.2 图的遍历……175
5.3.3 图的连通性……181
5.4 最小生成树……182
5.4.1 Prim 算法……183
5.4.2 Kruskal 算法……186
5.5 拓扑排序和关键路径……187
5.5.1 有向无环图……187
5.5.2 拓扑排序……188
5.5.3 关键路径……190
5.6 最短路径……194
5.6.1 单源最短路径……194
5.6.2 每对顶点间的最短路径……197
习题 5……199
第 6 章 查找……201
6.1 查找表的定义……201
6.2 静态查找表……202
6.2.1 顺序查找……202
6.2.2 折半查找……203
6.2.3 分块查找……206
6.3 动态查找表……207
6.3.1 二叉排序树……207
6.3.2 平衡二叉排序树……213
6.3.3 B-树和 B+树……221
6.4 哈希查找……226
6.4.1 哈希表的定义……227
6.4.2 哈希函数的构造方法……227
6.4.3 处理冲突的办法……229
6.4.4 哈希表的查找及分析……231
6.4.5 哈希表的构建与查找算法……233
习题 6……236
第 7 章 排序……238
7.1 插入类排序……238
7.1.1 直接插入排序……239

7.1.2 折半插入排序……240
7.1.3 2-路插入排序……241
7.1.4 希尔排序……241
7.2 分划类排序……243
7.2.1 冒泡排序……243
7.2.2 快速排序……244
7.3 选择类排序……248
7.3.1 简单选择排序……248
7.3.2 树形选择排序……249
7.3.3 堆排序……250
7.4 归并类排序……253
7.5 基数排序……254
7.5.1 多关键字的排序……254
7.5.2 基数排序……255
7.6 内部排序的比较……260
7.7 外部排序……262
7.7.1 外部存储设备……262
7.7.2 外部排序的方法……263
7.7.3 败者树……264
习题7……265
附录A……267
参考文献……270

第1章 绪 论

学习要点

理解数据结构知识领域中的基本概念：数据、数据项、数据元素、数据对象、数据类型、抽象数据类型、数据结构等。

数据结构是数据的逻辑结构和物理结构（存储结构）的总称。掌握数据的四种基本逻辑结构：集合、线性、树状和图状；掌握两种基本的存储结构：顺序存储结构和链式存储结构。

掌握算法的时间复杂度和空间复杂度的概念，掌握利用 O 算子对时间复杂度和空间复杂度进行定性度量的方法。

“数据结构”是计算机科学中的重要组成部分，是计算机软件开发技术的核心基础知识。本章从计算机数据处理的过程和原理出发，介绍数据、数据类型、数据结构等基本概念，在此基础上讨论算法、算法的性能度量等知识。

1.1 数据与数据类型

1.1.1 数据

从1946年诞生第一台电子数字计算机ENIAC开始，现代计算机的体系结构都是遵循冯·诺依曼模型来设计的。按照冯·诺依曼模型的要求，一个计算机系统可以分为四部分：存储器、运算器（即算术逻辑单元）、控制器和输入/输出单元。

- 存储器：用于存储数据及程序代码。
- 运算器：对数据进行算术运算或逻辑运算。
- 控制器：依据程序代码控制计算机各部件的运行。
- 输入/输出单元：执行数据的输入/输出操作。

依照这种模型设计的计算机，从本质上说，就是一个可编程的数据处理器（如图1.1所示）。

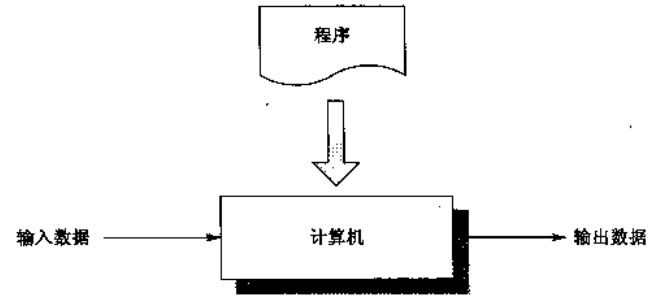


图1.1 可编程的数据处理器模型

在图1.1的模型中，计算机运行的结果（即输出数据）依赖于两个因素的共同作用：输入数据和程序。因此，在计算机中，所有子系统都围绕数据和程序这两个核心因素来设计。

随着计算机技术的发展，计算机应用的领域越来越广泛。最初的计算机系统主要用于科学计算，但现今的计算机应用领域已拓展到事务管理、计算机辅助设计、自动控制、文化传媒与娱乐等，纷繁复杂的应用领域导致数据处理复杂性的不断增加，但从总体上看，不管面对什么样的特殊应用领域，计算机的工作过程归结起来分为以下两个步骤：数据表示——对待处理事物的数据组织与表达；数据处理——依据特定程序对数据进行处理。

因此，在开发一个应用软件系统时，首先遇到的问题就是如何利用程序设计语言或数据描述语言（如在数据库系统中）来组织和表达所要处理的数据。

什么是数据？数据是通过物理观察得来的事实和概念，是关于现实世界中的地点、事件、人物或是某种概念和知识等客观事物的描述。这种描述是符号化的，可以有多种表示形式，如数字、字符、图形等，也可以是多种表示形式的组合。

例如，某个学生的姓名为“李小明”，年龄为21岁，并有一张个人照片，那么“李小明”、21岁、照片图像就是三个不同的数据，分别用字符、数字和图像这三种类型的数据来表示。同时，可以将这些数据结合起来，构成描述李小明这个学生的一个相对完整的数据。在这种情况下，还可以将姓名、年龄和照片称为学生这一类数据的三个数据项。

【定义 1.1】 数据（data）是对客观事物的符号化表示，是信息、概念与知识的载体。

【定义 1.2】 数据项（data item）是构成数据的相对独立的分项，反映客观事物的某种特性。

除了可以用不同类型的数据项来构成一个数据外，还有一种数据的组成形式。如果一个数据是一组相同性质的数据单元的集合，那么这些数据单元就称为数据元素。例如，一个班级由多个学生组成，描述每个学生的形式都是一致的，但他们又是独立的实体，在这种情况下，称每个学生是构成“班级”这个数据的数据元素。

【定义 1.3】 数据元素（data element）是构成数据的具有相同性质的基本单元。

数据、数据元素、数据项这三个概念表示了对一个客观事物的三种不同观察角度：数据是对事物整体的表达和描述；从反映事物的不同特性的角度来观察，数据是由数据项构成的；从构成事物的组成元素的角度来观察，数据是由数据元素构成的，在此情况下，称这些数据元素的集合为数据对象。

【定义 1.4】 数据对象（data object）是性质相同的数据元素的集合。

在“数据结构”这门课程的知识领域里，更多考虑的是构成一个数据对象中数据元素之间的关系。数据元素之间关系不同，一方面反映出数据的不同组织形式，另一方面也带来了不同的处理和操作的方法。

1.1.2 数据的计算机表示与数据类型

“数据”这一概念本身是与存储介质无关的，原始人结绳计数，后人在纸张上书写，都是一种数据存储方式。计算机诞生以后，就出现了如何在计算机内部存储或表示数据的问题。冯·诺依曼模型的一个重要特征就是使用“二进制编码”来表达、存储和使用数据。采用二进制可以使计算机硬件设计简单化，因为在计算机内部是以电子信号的出现和消失这两种状态来存储数据的，这两种状态分别用“0”和“1”来表示。但是，在计算机外部，用于表达客观事物的数据通常表现为文本、数字、图形、音频、视频等直观的形式，因此，为了对数

据进行计算机处理，必须将这些复杂的数据形式转变为以“0”和“1”构成的二进制数值序列，这个过程称为“编码”。计算机对数据进行处理后，再将二进制序列还原成数据的直观表示形式，以便被使用者接受，这个过程称为“解码”。图 1.2 表示的就是数据的编码与解码过程。

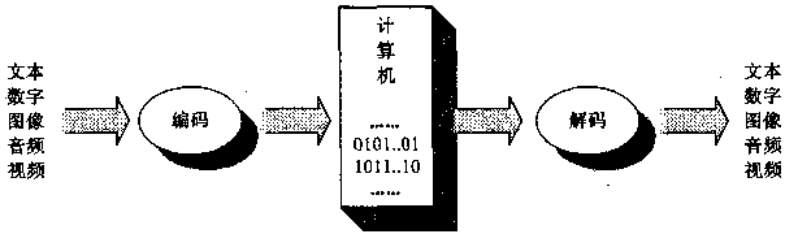


图 1.2 计算机中数据的编码与解码过程

不同类型的数据在计算机内部均编码成“0”和“1”的序列，反过来，同样的“0”、“1”序列通过不同的编码规则，可以理解成不同类型的数据。常用的编码方式如下：

- 文字——美国信息交换标准码 ASCII、中国国家标准汉字编码 GB 2312—1980、计算机厂商联合制订的多国文字编码 Unicode。
- 整数——无符号整数用原码表示，有符号整数用补码表示。
- 浮点数——IEEE 协会的浮点数编码标准 IEEE 754。
- 图像——BMP、JPEG 等。
- 音频——PCM、WAVE、MP3 等。

编码方式决定了“0”、“1”序列所表示数据的不同类型，因此，在软件系统中广泛使用了“数据类型”这一概念。例如，在程序设计语言中，对变量、函数返回值等均需进行数据类型声明，在关系型数据库系统中，也必须将每个字段指定为某种数据类型。无论是程序设计语言中的变量、函数返回值，还是数据库系统中的字段，一经声明为某种数据类型，实际上也就指明了它们存储在存储器（内存或外存）中的数据编码方式，也决定了这种类型数据的取值范围，即值域。例如，16 位无符号整数的取值范围为 0～65 535，而 16 位有符号整数的取值范围为-32 768～32 767。

数据类型的第二个含义是决定了对该类型数据允许进行的操作，如对整数型数据可以进行的操作有：加、减、乘、除、取模、取余、位操作（按位与、或、非）等。值域和操作是构成数据类型的两个基本因素。

【定义 1.5】 *数据类型（data type）是对具有相同性质的数据的抽象，定义了一个值域及在这个值域上可以进行的一组操作。*

数据类型的概念并不局限于软件系统中，在 CPU 的指令系统里就有数据类型的概念，每一条指令操作的存储单元（寄存器、内存或外存中的存储单元）决定了所存储数据的值域，对这个数据的操作是通过硬件来实现的。在高层软件系统里，数据类型概念的引入，屏蔽了具体的硬件实现细节，在使用数据时，并不需要了解这些数据在物理器件上的存储与编码方式，对数据的操作也是由支撑的系统软件来实现的，如编译器或数据库管理系统。在系统软件支撑下，可以提供比硬件系统更加复杂、更加多样的数据类型。

一般情况，软件开发系统内建了一些基础的数据类型，如字符、整数、浮点数、指针、

引用等，这些类型通常称为“基本数据类型”，基本数据类型是原子的，即不可分割的。

软件开发系统中，仅有基本数据类型是远远不够的。计算机应用领域的拓展，使得程序开发人员必须面对一些新类型的数据，如图像、音频、视频等，软件开发系统也必须适应这个需求。但数据种类的复杂性和不可预知性又决定了作为支撑的系统软件不可能无限制地在基本数据类型之外提供新的数据类型。可行的做法就是提供一种机制，在这种机制下，软件开发人员可以利用基本数据类型来构造新的数据类型。

通过一定规则来组合一系列基础数据类型，可以构成一些派生的数据类型，如数组、结构、联合、枚举等。这些派生的数据类型是程序员自定义的，可以看做是在软件系统中引入了新的数据类型。新数据类型的引入，丰富了程序的语义表达功能。例如，下面是一个 C 程序片断，其中的 Student 就是利用 C 的结构体来实现的一个新数据类型，Student 结构中有三个数据项：数据项 name 的类型为字符数组，数据项 birth 的类型为 Date 结构，数据项 sex 的类型为 Sex 枚举。

```
struct Date              //  定义日期类型
{
    int year;
    int month;
    int day;
};

enum Sex                 //  定义性别类型
{
    Male, Female
};

struct Student           // 定义学生类型
{
    char name[9];
    Date birth;
    Sex sex;
};
```

上面的程序代码中，Student 结构用于表达和存储学生的一些基本信息。但仅用 Student 结构来描述学生这样一个客观事物是不够的，其中仅仅表达了学生的数值属性。要全面完整地表达“学生”这一概念，通常还需要描述其行为特性，如选课、参加社团等行为。这些行为特性实际上构成了“数据类型”定义中的“操作”概念。在 C 语言中，描述行为特性可以用函数来实现，如 RegisterCourse()、JoinClub()等。只有附加了这些操作，才真正实现了 Student 数据类型的完整定义。

作为一种传统的结构化程序设计方法，在 C 语言中，结构定义和函数定义是相对独立的。作为一种新型的软件开发方法，在 20 世纪 60 年代末出现了面向对象程序设计技术，面向对象技术更加强调了数值属性与行为特性的结合，提出了“对象”这一概念，对象是一组数据和对这组数据操作的封装。对一组相似对象的抽象叫“类”，类是数据的数值属性和操作声明的封装体。下面是用 C++来实现 Student 数据类型的程序代码。

```
class Student          // 定义学生类
{
public:
    // 对数值属性访问接口
    void SetName(const char *s);
    void GetName(char* &s);
    void SetBirth(int y, int m, int d);
    void GetBirth(int &y, int &m, int &d)
    void SetSex(Sex s);
    void GetSex(Sex &s);

    // 行为接口
    void RegisterCourse(int coursId);
    void JoinClub(const char *clubName);

private:
    char name[9];
    Date birth;
    Sex sex;
};
```

上面的 Student 类的声明是操作和数值属性的封装，public 节中的前 6 个成员函数用于对 private 节中定义的 3 个数据成员进行存取操作；public 节中的后 2 个成员函数 RegisterCourse 和 JoinClub 是与应用系统的业务相关的操作接口。

将数据成员声明为 private 是通行的做法，体现了面向对象程序设计技术中信息屏蔽（Information Hiding）的思想，即要把数据属性的实现细节屏蔽起来，达到外部接口与内部实现相分离的目的。例如，可以把“char name[9]”改为“char *name”，把“Date birth”改为“int year, month, day”，只要修改成员函数的实现，保持 public 节中函数的声明不变，外部程序依旧可以使用同样的类成员函数接口，而不需要做任何修改。遵循实现与接口相分离的原则，保持接口的相对稳定，可以降低程序模块之间的耦合度，有效地增强程序的清晰度和可维护性。

1.1.3 抽象数据类型

在程序代码设计阶段，使用基本数据类型或是定义一个新的数据类型，都必须遵循程序设计语言规定的语法规则。例如，C/C++用 int 来声明整数型变量，而 Basic 用 integer 声明整数型变量；C/C++用 struct 来声明结构型变量，与此相类似，Pascal 则用 record 来声明记录型变量。

另外，使用同样的程序设计语言，也可以使用不同的方法实现相同的自定义数据类型。就如 1.1.2 节中叙述的那样，Student 类中姓名和出生日期可以采用不同的方法来定义，但即使实现方法不同，成员函数声明依旧不变，也就是说对外的接口形式没有改变。

从上面的两个角度来说，int 和 integer 表达的都是“整数”这一概念，不同实现方法的 Student 表达的是一样的“学生”概念。因此，如果在更高的层次上来考察 Student 这一数据类型，抛开具体的实现细节，“整数”和“学生”这两个概念都是抽象的，即它们都是一种

“抽象数据类型”。

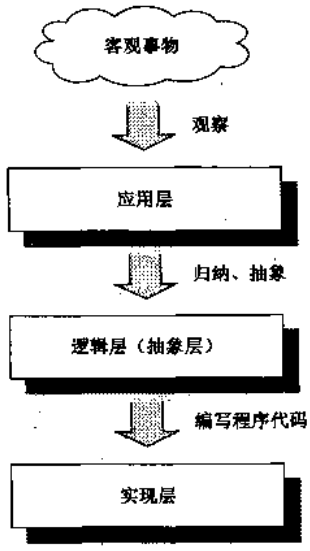


图 1.3　软件系统开发过程中的数据层次模型

【定义 1.6】抽象数据类型（Abstract Data Type，ADT）是与具体计算机内部表示与实现方式无关的数据类型，由一个逻辑上的数学模型和定义在该模型上的一组操作构成。

在软件系统开发的全过程中，对客观现实中存在的事物，存在三个观察层面，如图 1.3 所示。

在图 1.3 中，应用层是用户通过物理观察得到的客观事物的视图，是可以用自然语言描述的，或用图形、图像、音频、视频等物理量表达的在概念层次上的数据；逻辑层（抽象层）是从应用层次抽象出来的数据视图，利用抽象数据类型来对数据进行形式化描述；实现层明确表示出了数据的组织与存储结构，并用某种具体的程序设计语言进行代码实现。

从应用层到实现层，实际上就完成了用于描述客观事物的数据从主观感知到计算机表示的过程。抽象数据类型存在于逻辑层中，用来表示与具体实现技术无关的逻辑上的数据抽象。

抽象数据类型可以用三元组来表示：

$$ADT = (D, S, P)$$

其中，D 是数据对象，S 是 D 中数据的关系集合，P 是对 D 的操作集合。

本书中，采用以下格式来描述抽象数据类型：

```
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>
}
```

其中，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>

【例 1.1】　定义集合的抽象数据类型。

```
ADT Set
{
    数据对象：D={ai | ai∈ElemType, i= 1, 2,…, n, n≥0}
    数据关系：R={ai≠aj | ai,aj∈D}
    基本操作：
        Init()
```

```
        操作结果：构造一个空的集合
    Destroy()
        操作结果：销毁集合
    IsEmpty()
        操作结果：判断集合是否为空，如为空，则返回 TRUE；否则，返回 FALSE
    Insert(e)
        操作结果：在集合中加入一个元素。如元素已存在，返回 FALSE；否则，返回 TRUE
    Remove(e)
        操作结果：在集合中移除一个元素。如元素存在，则返回 TRUE；否则，返回 FALSE
    IsMember(e)
        操作结果：判断在集合中是否存在元素
    FindFirst(&e)
        操作结果：找到集合中的第一个元素。如成功，返回 TRUE；如果集合为空，返回 FALSE
    FindNext(&e)
        初始条件：已经执行过 FindFirst 或 FindNext 操作
        操作结果：在上一次查找的前提下，找到集合中的下一个元素；如成功，返回 TRUE；
                  如上次查找已到最后一个元素，返回 FALSE
    Union(s)
        操作结果：与另一个集合 s 做并运算，返回并集
    Intersection(s)
        操作结果：与另一个集合 s 做交运算，返回交集
    Difference(s)
        操作结果：与另一个集合 s 做差运算，返回差集
}
```

在上面的 ADT 描述中，ElemType 表示集合元素的数据类型。遵循 Set 这样的一个抽象数据类型的定义，下一步就可以用不同的程序设计语言来实现集合数据类型。

1.2 数据结构

在 1.1 节中，可以看到，一个抽象数据类型包含数据对象、数据关系和数据操作这三个因素，其中的数据关系定义了数据元素之间逻辑上的组织结构，用程序来实现抽象数据类型时，还必须考虑这些数据元素在计算机系统中的存储方式。按照图 1.3 所描述的软件系统开发过程中的数据层次模型，数据之间存在着两种结构关系：一是数据的逻辑结构，反映数据元素之间的逻辑关系，抽象数据类型实际上考虑的就是数据的逻辑结构，存在于层次模型的逻辑层中；二是数据的物理结构，也称为存储结构，考虑的是数据在计算机中是如何存储和组织的，这一概念属于层次模型中的实现层，实际上对应于逻辑层的抽象数据类型的程序化实现。

【定义 1.7】 *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。*

数据结构可以用二元组来表示：

$$\text{DataStructure} = (D, S)$$

其中，D 是数据元素的有限集，S 是 D 上关系的有限集。

【例 1.2】 *数组是一种线性的数据结构，是由 n 个元素有序排列而成的：*

$$Array = (D, S)$$

其中：

数据元素的集合 $D = \{a_1, a_2, \cdots, a_n\}$

关系的集合 $S = \{R\}$

关系 $R = \{< a_i, a_{i+1} > \mid 1 \leqslant i < n\}$（符号$< a_{i-1}, a_i >$表示序偶，即一对有序的元素）

例 1.2 表示了数据元素之间一种特殊的逻辑结构——线性结构，数据元素之间的关系除了线性结构外，还有集合结构、树型结构和图状结构，它们一起被称为数据元素之间的 4 种基本逻辑结构。

⊙ 集合结构——数据元素间的关系是“属于同一个集合”。

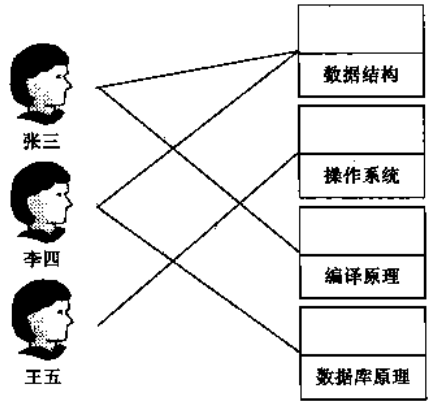
⊙ 线性结构——数据元素之间存在着一对一的关系。

⊙ 树型结构——数据元素之间存在着一对多的关系。

⊙ 图状结构——数据元素之间存在着多对多的关系，也称网状结构。

下面通过几个实际问题来说明这 4 种逻辑结构。

【例 1.3】 学生选课问题。要利用信息化技术管理学校里学生的选课情况，需要记录学生、课程和选课三类信息，如图 1.4 所示。



（a）学生选课关系图

学生表

学号	姓名
S0001	张三
S0002	李四
S0005	王五

课程表

课程号	课程名称
C0001	数据结构
C0002	操作系统
C0003	编译原理
C0004	数据库原理

选课表

学号	课程号	成绩
S0001	C0001	85
S0001	C0003	76
S0002	C0001	90
S0002	C0004	83
S0003	C0002	92

（b）学生、课程与选课的数据表结构

图 1.4　学生选课的数据结构例

这三类信息分别用三个数据表来记录，表中的每一行称为“记录”，表中的每一列称为“字段”。记录和记录之间在物理存储时是无序的，因此一个数据表中的记录就形成了集合结构。在实际应用中，通常会将数据表的一个或几个字段指定为“关键字”，在表中每条记录