清华大学计算机系列教材

数据结构

（用面向对象方法与C++描述）

殷人昆 陶永雷 谢若阳 盛绚华 编著

清华大学出版社

清华大学计算机系列教材

数 据 结 构

(用面向对象方法与C++描述)

殷人昆 陶永雷 谢若阳 盛绚华 编著

清华大学出版社

(京)新登字 158 号

内 容 简 介

数据结构是计算机专业的核心课程,是从事计算机软件开发、应用人员应当必备的专业基础。随着计算机的日益普及,简单的数据结构知识已经下放到中学的计算机课程中,并已成为计算机软件考试的必考课程之一。本书是根据作者在北京清华大学及美国密西根州 Grand Valley 州立大学多年教学的经验,并参考了近年出版的多种国外大学数据结构和面向对象软件工程教科书编写的。内容包括:数组、链接表、栈和队列、递归、树与森林、图、堆与优先级队列、集合与搜索结构、排序、索引结构与散列等。书中采用面向对象的观点讨论数据结构技术,并以兼有面向过程和面向对象双重特色的 C++ 语言作为算法的描述工具,强化基本知识和基本能力的双基训练。全书条理清晰,通俗易懂,图文并茂,适于自学。

本书适合作大专院校中计算机或软件专业的教材,也可供计算机软件人员和计算机用户阅读。

图书在版编目(CIP)数据

数据结构:用面向对象方法与 C++ 描述/殷人昆等编著.—北京:清华大学出版社,1999
(清华大学计算机系列教材)
ISBN 7-302-03405-2

Ⅰ.数… Ⅱ.殷… Ⅲ.数据结构 Ⅳ.TP311.12

中国版本图书馆 CIP 数据核字(1999)第 07546 号

出版者:清华大学出版社(北京清华大学学研楼,邮编 100084)
http://www.tup.tsinghua.edu.cn
印刷者:北京市清华园胶印厂
发行者:新华书店总店北京发行所
开 本:787×1092 1/16 **印张**:26 **字数**:646 千字
版 次:1999 年 7 月第 1 版 2000 年 3 月第 4 次印刷
书 号:ISBN 7-302-03405-2/TP·1845
印 数:20001～28000
定 价:26.00 元

前　言

由于计算机的普及，极大地改变了人们的生活。目前各个行业、各个领域都与计算机建立了紧密的联系。这也随之带来了开发各种软件的需求。为了能够以最少的成本，最快的速度，最好的质量开发出合乎需要的软件，不能再像以前那样，把软件看作是个人雕琢的精品，必须遵循软件工程的原则，把软件的开发、维护活动标准化、工程化。就软件产品而言，最重要的就是建立合理的软件体系结构和程序结构，设计有效的数据结构。因此，为能做好软件开发工作，必须了解应如何组织各种数据在计算机中的存储、传递和转换。这样，《数据结构》这门课程显得格外重要。自 1978 年美籍华裔学者冀中田在国内首开这门课程以来（当时作者也在场），经过 20 余年的发展，这门课程已成为各大学计算机专业本科的主干课程，也成为非计算机类学生和研究生学习计算机的必修课程。

《数据结构》课程脱胎于《离散数学结构》，它涉及各种离散结构（如向量、集合、树、图、代数方程、多项式等）在计算机上如何存储和处理。其内容丰富，涉及面广泛，而且还在随各种基于计算机的应用技术的发展，不断增加新的内容。特别是面向对象技术出现以后，人们认识到，用它开发出来的软件体系结构更加符合人们的习惯，质量更容易得到保证，尤其是对使用者和用户不断提出的新的需求，更容易适应。因此，在国际上，面向对象技术得到迅速普及，出现了大批面向对象的软件开发工具。为了适合形势的要求，有必要开设结合面向对象技术的数据结构课程。

用面向对象的观点讨论数据结构，与传统的面向过程的讲法相比，变化较大。各种数据结构的讨论都是基于抽象数据类型和软件复用的。有新意，也有继承。我们力图与过去的讲授体系保持一致，但又必须引入一些新的概念。为了能够让读者容易学习，我们对内容进行了精选。许多概念，如双端堆、二项堆、最小－最大堆、斐波那契堆、左斜树、扁树、红－黑树、B＊树这些从基本数据结构派生出来的都舍去了。同时，把动态存储管理部分归到《操作系统》课程，把文件组织部分归到《数据库原理》，只保留重要的应用最广泛的一些结构。对这些结构做全面深入的讲解。阐明数据结构内在的逻辑关系，讨论它们在计算机中的存储表示，并结合各种典型事例说明它们在解决应用问题时的动态行为和各种必要的操作。并以 C++ 语言为表述手段，介绍在面向对象程序设计过程中各种数据结构的表达和实现。只要是学过 C 或 PASCAL 语言，就能够很容易地阅读和理解。并因此学习 C++ 语言，提高读者的软件设计和编程能力。

本书是作为清华大学计算机系的《数据结构》教科书编写的。它的作者是在清华大学和美国密西根州 Grand Valley 州立大学从事《数据结构》第一线教学的教师。他们有着丰富的数据结构和软件工程教学的经验，教学效果良好。陶永雷教授 1997 年春专程回国参加教材的编写工作。陶教授根据他在美国讲授软件工程和数据结构课程的经验，就内容选择、讲授体系、习题选编提出了很多建议，并帮助我们搜集了多种在美国受欢迎的数据结构方面的参考书和实验指导书，亲自审查和修改了全书的类声明和程序。北方交通大学盛绚华副教授

亲自带领该校通控系一些同学对这些程序进行了验证。全书的执笔由清华大学计算机系《数据结构》课程讲员殷人昆和谢若阳担任。

全书共分 10 章，第 1 章是预备知识，主要介绍什么是数据，数据与信息的关系；什么是数据结构，数据结构的分类。通过学习，读者能够了解抽象数据类型和面向对象的概念，并对对象、类、继承、消息以及其它关系的定义、使用有基本认识。由于我们选择了具有面向过程和面向对象双重特点的 C++ 语言，可以帮助读者自然而轻松地从传统程序设计观念向面向对象方法转变。在这一章的最后还讨论了算法设计和简单的算法分析方法。

第 2 章和第 3 章是全书的基础，讨论了数组和链表结构，以及利用它们定义出来的各种结构，如顺序表、代数多项式、稀疏矩阵、字符串等。通过学习，读者可以了解对象和类的基本实现，并通过模板、继承、多态性等的使用，以及游标类的引入，对数据抽象概念有进一步的理解。

第 4 章引入三种存取受限的表，即栈、队列和优先级队列。通过对它们的定义、实现和应用的深入介绍，使读者能够了解在什么场合使用它们，为以后更复杂数据结构和算法的实现，提供了多种辅助手段。

第 5 章介绍在许多领域中经常遇到的递归程序设计概念，以及递归的非递归实现问题。在本章的后半部分重点介绍了广义表，这是一种应用广泛又十分灵活的结构。它的递归结构导致了它的许多操作的递归实现，从而加深对递归的认识。

第 6 章和第 8 章介绍在实际应用中最重要的非线性结构——树与图。在管理、电子设计、机械设计、日常生活中许多方面都会用到它们。

第 7 章和第 10 章介绍集合、查找、索引、散列。在实际与信息处理相关的应用中，这些结构十分重要。许多非数值处理都涉及到这些结构，它们与内存、外存上的数据组织关系密切，如在外存组织文件时全面应用了这些结构。它们又是许多新结构的生长点。因此，读者学习这些内容将获益匪浅。

第 9 章介绍排序。这也是应用十分广泛的技术。只要是数据处理，就少不了排序。如何才能高效地完成排序，本章分别就内、外存使用的多种排序方法进行介绍和讨论，读者可以深入了解排序的机制，并也能从中学到许多程序设计的技巧。

本书的篇幅虽然较大，但给读者一个选择，可以根据时间、能力，适当对全书需要学习的内容加以剪裁。本着少讲多练的原则，可以对每种结构只介绍类定义和关键操作的实现，其它内容可以让同学自学。通过上机练习，加深理解。在本书目录中加 * 的章节可以酌情不讲。

在本书的成书过程中得到计算机系领导的关心，清华大学出版社的支持。此外，清华大学计算机系 1994 级全体 150 多位同学和中文系 1994 级 6 位同学参加了资料的翻译、整理工作。

作　者

1998 年 12 月于北京

目　录

第1章　绪论 …… 1
1.1　什么是数据结构 …… 2
1.2　抽象数据类型及面向对象概念 …… 4
1.2.1　数据类型 …… 4
1.2.2　数据抽象与抽象数据类型(ADTs：Abstract Data Types) …… 4
1.2.3　面向对象的概念 …… 5
1.2.4　用于描述数据结构的语言 …… 7
1.3　数据结构的抽象层次 …… 7
*1.4　用C++描述面向对象程序 …… 9
1.4.1　C++的函数特征 …… 9
1.4.2　C++的数据声明 …… 10
1.4.3　C++的作用域 …… 11
1.4.4　C++的类 …… 11
1.4.5　C++中的对象 …… 13
1.4.6　C++的输入输出 …… 13
1.4.7　C++中的函数 …… 15
1.4.8　C++中的参数传递 …… 15
1.4.9　C++中的函数名重载和操作符重载 …… 16
1.4.10　C++中的动态存储分配 …… 17
1.4.11　友元(friend)函数 …… 18
1.4.12　内联(inline)函数 …… 18
1.4.13　结构(struct)与类 …… 18
1.4.14　联合(Union)与类 …… 19
1.5　算法定义 …… 19
1.6　模板(template) …… 20
1.7　性能分析与度量 …… 23
1.7.1　算法的性能标准 …… 23
*1.7.2　算法的后期测试 …… 24
1.7.3　算法的事前估计 …… 26
1.7.4　空间复杂度度量 …… 26
1.7.5　时间复杂度度量 …… 27
1.7.6　时间复杂度的渐进表示法 …… 31
1.7.7　渐进的空间复杂度 …… 34

习题 …… 34

第2章 数组 …… 37
2.1 作为抽象数据类型的数组 …… 37
2.1.1 在C++中数组的定义和初始化 …… 37
2.1.2 作为抽象数据类型的数组 …… 38
2.1.3 数组的顺序存储方式 …… 40
2.2 顺序表 …… 42
2.2.1 顺序表的定义和特点 …… 43
2.2.2 顺序表的类定义 …… 43
2.2.3 顺序表的查找、插入和删除 …… 45
2.2.4 作为抽象数据类型,使用顺序表的事例 …… 47
*2.3 多项式抽象数据类型 …… 47
2.3.1 多项式抽象数据类型 …… 48
2.3.2 多项式的表示 …… 49
2.3.3 多项式的相加 …… 51
2.4 稀疏矩阵 …… 53
2.4.1 稀疏矩阵的抽象数据类型 …… 53
2.4.2 稀疏矩阵的压缩表示 …… 53
* 2.4.3 稀疏矩阵的转置 …… 54
* 2.4.4 稀疏矩阵相乘 …… 57
2.5 字符串 …… 59
2.5.1 字符串抽象数据类型和类定义 …… 59
2.5.2 字符串操作的实现 …… 61
2.5.3 字符串的模式匹配 …… 63
* 2.5.4 模式匹配的改进算法——KMP(D.E.Knuth—J.H.Morris—V.R.Pratt)算法 …… 64
习题 …… 68

第3章 链表 …… 71
3.1 单链表(Singly Linked List) …… 71
3.1.1 单链表的结构 …… 71
3.1.2 单链表的类定义 …… 72
3.1.3 单链表中的插入与删除 …… 73
3.1.4 带表头结点的单链表 …… 75
3.1.5 用模板定义的单链表类 …… 76
3.1.6 单链表的游标(Iterator)类 …… 79
3.1.7 静态链表 …… 81
3.2 循环链表 …… 82

3.2.1 循环链表的类定义 …… 82
3.2.2 用循环链表求解约瑟夫问题 …… 83
*3.3 多项式及其相加 …… 84
3.3.1 多项式的类定义 …… 84
3.3.2 多项式的加法 …… 85
3.4 双向链表 …… 87
3.5 稀疏矩阵 …… 92
3.5.1 稀疏矩阵的类定义 …… 92
3.5.2 稀疏矩阵的建立 …… 94
3.5.3 删除稀疏矩阵 …… 95
3.6 C++中的虚函数和动态联编 …… 96
3.6.1 C++中的继承(inheritance) …… 96
3.6.2 基类与派生类对象指针的转换 …… 97
3.6.3 虚函数(virtual function) …… 99
3.6.4 纯虚函数和抽象基类 …… 99
3.6.5 多态性(polymorphism)和动态联编 …… 99
习题 …… 101

第4章 栈和队列 …… 103
4.1 栈 …… 103
4.1.1 栈的抽象数据类型 …… 103
4.1.2 栈抽象数据类型的顺序存储表示与实现——顺序栈 …… 104
4.1.3 栈抽象数据类型的链接存储表示——链式栈 …… 106
*4.2 表达式的计算(Expression Evaluation) …… 107
4.2.1 表达式 …… 107
4.2.2 应用后缀表示计算表达式的值 …… 108
4.2.3 中缀表达式转换为后缀表达式 …… 111
4.3 队列 …… 113
4.3.1 队列的抽象数据类型 …… 113
4.3.2 队列的顺序存储表示——循环队列 …… 114
4.3.3 队列的链接存储表示——链式队列 …… 116
4.3.4 队列的应用举例——打印二项展开式$(a+b)^i$的系数 …… 118
4.4 优先级队列(Priority Queue) …… 119
4.4.1 优先级队列的定义 …… 119
4.4.2 优先级队列的存储表示和实现 …… 120
*4.5 事件驱动模拟(Event-Driven Simulation) …… 121
习题 …… 130

第5章 递归(RECURVE) …… 132

5.1 递归的概念 …… 132
5.2 迷宫问题 …… 135
5.3 递归过程与递归工作栈 …… 138
*5.4 利用栈实现的迷宫问题非递归解法 …… 142
5.5 广义表(General Lists) …… 145
5.5.1 广义表的概念 …… 146
5.5.2 广义表的表示及操作 …… 147
5.5.3 广义表存储结构的实现 …… 149
5.5.4 广义表的访问算法 …… 151
5.5.5 广义表的递归算法 …… 153
* 5.5.6 三元多项式的表示 …… 159
习题 …… 161

第6章 树与森林 …… 163
6.1 树和森林的概念 …… 163
6.1.1 树的定义 …… 163
6.1.2 树的术语 …… 163
6.1.3 树的抽象数据类型 …… 164
6.2 二叉树(Binary Tree) …… 165
6.2.1 二叉树的定义 …… 165
6.2.2 二叉树的性质 …… 166
6.2.3 二叉树的抽象数据类型 …… 167
6.3 二叉树的表示 …… 168
6.3.1 数组表示 …… 168
6.3.2 链表存储表示 …… 169
6.4 二叉树遍历(Binary Tree Traversal) …… 172
6.4.1 中序遍历(Inorder Traversal) …… 173
6.4.2 前序遍历(preorder Traversal) …… 173
6.4.3 后序遍历(Postorder Traversal) …… 174
6.4.4 应用二叉树遍历的事例 …… 175
* 6.4.5 二叉树遍历的游标类(Tree Iterator) …… 177
* 6.4.6 不用栈的二叉树中序遍历算法 …… 182
*6.5 线索化二叉树(Threaded Binary Tree) …… 182
6.5.1 线索(Thread) …… 182
6.5.2 中序线索化二叉树 …… 183
6.5.3 前序与后序的线索化二叉树 …… 188
6.6 堆(Heap) …… 189
6.6.1 堆的定义 …… 190
6.6.2 堆的建立 …… 191

6.6.3 堆的插入与删除 …… 192
6.7 树与森林 …… 194
6.7.1 树的存储表示 …… 194
6.7.2 森林与二叉树的转换 …… 199
6.7.3 树的遍历 …… 200
6.7.4 森林的遍历 …… 202
*6.8 二叉树的计数 …… 203
6.9 霍夫曼树(Huffman Tree) …… 206
6.9.1 路径长度(Path Length) …… 206
6.9.2 霍夫曼树 …… 207
6.9.3 霍夫曼编码 …… 209
习题 …… 210

第7章 集合与搜索 …… 212
7.1 集合及其表示 …… 212
7.1.1 集合基本概念 …… 212
7.1.2 以集合为基础的抽象数据类型 …… 213
7.1.3 用位向量实现集合抽象数据类型 …… 213
7.1.4 用有序链表实现集合的抽象数据类型 …… 215
7.2 等价类和并查集 …… 219
7.2.1 等价关系与等价类 …… 219
7.2.2 确定等价类的链表方法 …… 220
7.2.3 并查集 …… 222
7.3 静态搜索结构 …… 227
7.3.1 搜索的概念 …… 227
7.3.2 静态搜索结构 …… 228
7.3.3 顺序搜索 …… 229
7.3.4 基于有序顺序表的折半搜索 …… 231
* 7.3.5 基于有序顺序表的斐波那契搜索和插值搜索 …… 234
7.4 二叉搜索树 …… 235
7.4.1 定义 …… 235
7.4.2 二叉搜索树上的搜索 …… 237
7.4.3 二叉搜索树的插入 …… 238
7.4.4 二叉搜索树的删除 …… 239
* 7.4.5 与二叉搜索树相关的中序游标类 …… 240
*7.5 最优二叉搜索树 …… 242
7.5.1 扩充二叉搜索树 …… 242
7.5.2 最优二叉搜索树 …… 243
7.6 *AVL* 树 …… 247

7.6.1 *AVL* 树的定义 …… 247
7.6.2 平衡化旋转 …… 248
7.6.3 *AVL* 树的插入和删除 …… 252
7.6.4 *AVL* 树的高度 …… 255
习题 …… 256

第 8 章 图 …… 259
8.1 图的基本概念 …… 259
8.1.1 图的基本概念 …… 259
8.1.2 图的抽象数据类型 …… 261
8.2 图的存储表示 …… 262
8.2.1 邻接矩阵 …… 262
8.2.2 邻接表 …… 265
8.2.3 邻接多重表 …… 269
8.3 图的遍历与连通性 …… 270
8.3.1 深度优先搜索 …… 271
8.3.2 广度优先搜索 …… 272
8.3.3 连通分量 …… 273
8.3.4 重连通分量 …… 274
8.4 最小生成树 …… 277
8.4.1 克鲁斯卡尔(Kruskal)算法 …… 278
8.4.2 普里姆(Prim)算法 …… 280
8.5 最短路径 …… 283
8.5.1 边上权值非负情形的单源最短路径问题 …… 283
* 8.5.2 边上权值为任意值的单源最短路径问题 …… 286
8.5.3 所有顶点之间的最短路径 …… 288
8.6 活动网络 …… 290
8.6.1 用顶点表示活动的网络 …… 290
8.6.2 用边表示活动的网络 …… 294
习题 …… 298

第 9 章 排序 …… 301
9.1 概述 …… 301
9.2 插入排序 …… 303
9.2.1 直接插入排序 …… 303
9.2.2 折半插入排序 …… 304
9.2.3 链表插入排序 …… 305
9.2.4 希尔排序 …… 307
9.3 交换排序 …… 309

9.3.1 起泡排序 …… 309
9.3.2 快速排序 …… 310
9.4 选择排序 …… 312
9.4.1 直接选择排序 …… 313
9.4.2 锦标赛排序 …… 313
9.4.3 堆排序 …… 317
9.5 归并排序 …… 318
9.5.1 归并 …… 318
9.5.2 迭代的归并排序算法 …… 319
9.5.3 递归的表归并排序 …… 321
*9.6 基数排序 …… 323
9.6.1 多关键码排序 …… 323
9.6.2 链式基数排序 …… 324
9.7 外排序 …… 326
9.7.1 外排序的基本过程 …… 326
9.7.2 *k* 路平衡归并 …… 329
9.7.3 初始归并段的生成 …… 333
* 9.7.4 并行操作的缓冲区处理 …… 337
9.7.5 最佳归并树 …… 339
习题 …… 341

第 10 章 索引结构与散列 …… 344
10.1 静态索引结构 …… 344
10.1.1 线性索引 …… 344
10.1.2 倒排表 …… 346
10.1.3 *m* 路静态搜索树 …… 347
10.2 动态索引结构 …… 348
10.2.1 动态的 *m* 路搜索树 …… 348
10.2.2 B_树 …… 350
10.2.3 B_树的插入 …… 352
10.2.4 B_树的删除 …… 354
10.2.5 B+树 …… 358
* 10.3 *Trie* 树 …… 361
10.3.1 *Trie* 树的定义 …… 361
10.3.2 *Trie* 树的搜索 …… 362
10.3.3 在 *Trie* 树上的插入和删除 …… 363
10.4 散列(Hashing) …… 364
10.4.1 词典(Dictionary)的抽象数据类型 …… 364
10.4.2 散列表与散列方法 …… 365

10.4.3 散列函数 …… 365
10.4.4 处理溢出的闭散列方法 …… 369
10.4.5 处理溢出的开散列方法——链地址法 …… 376
10.4.6 散列表分析 …… 378
*10.5 可扩充散列 …… 379
10.5.1 二叉 *Trie* 树 …… 379
10.5.2 将二叉 *Trie* 树转换为目录 …… 380
10.5.3 插入与目录扩充 …… 383
10.5.4 删除与目录收缩 …… 385
10.5.5 性能分析 …… 386
习题 …… 388

附录 实习要求与实习报告 …… 391
实习 1 栈和队列 …… 396
实习 2 串(内容：全屏幕文本编辑器) …… 396
实习 3 树(内容：作业调度) …… 398
实习 4 图(内容：某公园导游图) …… 399
实习 5 查找、排序(内容：简单的职工管理系统) …… 399
参考文献 …… 401

第1章 绪 论

众所周知，无论是使用计算机进行工程和科学计算，还是使用计算机做工业控制或信息管理，都属于数据处理的范畴。那么，如何在计算机中组织、存储、传递数据，就成为一个必须解决的问题。下面让我们看几个例子。

【例1】 在一个学生选课系统中，有两个数据实体（现实世界中的事物），即“学生”和“课程”，形成了两个数据表格，其中的“学生”实体包括了许多学生记录，它们按照每个学生学号递增的次序，顺序存放在“学生”表格中；而“课程”实体包括了各个课程记录，每个课程也有个课程编号，在“课程”表格中各个课程按照其课程编号递增的次序依次排列。如图1.1所示。

“学生”表格

	学 号	姓 名	性别	籍 贯	出生年月
1	98131	刘激扬	男	北 京	1979.12
2	98164	衣春生	男	青 岛	1979.07
3	98165	卢声凯	男	天 津	1981.02
4	98182	袁秋慧	女	广 州	1980.10
5	98203	林德康	男	上 海	1980.05
6	98224	洪 伟	男	太 原	1981.01
7	98236	熊南燕	女	苏 州	1980.03
8	98297	宫 力	男	北 京	1981.01
9	98310	蔡晓莉	女	昆 明	1981.02
10	98318	陈 健	男	杭 州	1979.12

“课程”表格

	课程编号	课 程 名	学时
1	024002	程序设计基础	64
2	024010	汇编语言	48
3	024016	计算机原理	64
4	024020	数据结构	64
5	024021	微机技术	64
6	024024	操作系统	48
7	024026	数据库原理	48

图 1.1 学生选课系统中的两个数据实体

在“学生”表格中，各个学生记录顺序排列，形成一个学生记录的线性序列，每个记录在序列中的位置有先后次序，它们之间形成一种线性关系。“课程”表格中的情况完全相同。

【例2】 在学生选课系统中，一个学生可以选修多门课程，一门课程可以被多个学生选修，这在“学生”和“课程”实体之间形成多对多的关系。为了便于处理，引入一个新的实体“选课”，它的每一个记录是一张选课单，包含有如下的信息：

学 号	课程编号	成 绩	时 间

此时，在“学生”实体、“课程”实体和“选课”实体间形成如图 1.2 所示的关系，这是一种

网状关系。学生对选课,课程对选课都是一对多的关系。

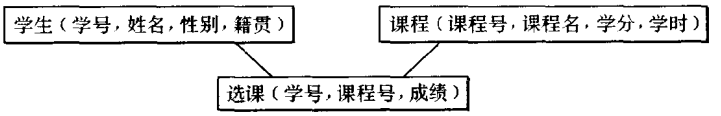


图 1.2 学生选课系统中实体构成的网状关系

【例 3】 一个典型的 UNIX 文件系统的系统结构如图 1.3 所示。这是一个层次结构:在此系统结构图中,顶层结点代表整个系统,用根目录"/"表示;它的下一层结点代表系统的各个子系统,即根的子目录,如"/bin"、"/lib"、"/user"等;再下一层结点代表更小的子目录,如"/user/yin"、"/bin/ds",如此类推,直到底层,即为文件,如"/user/yin/queue.cpp"。

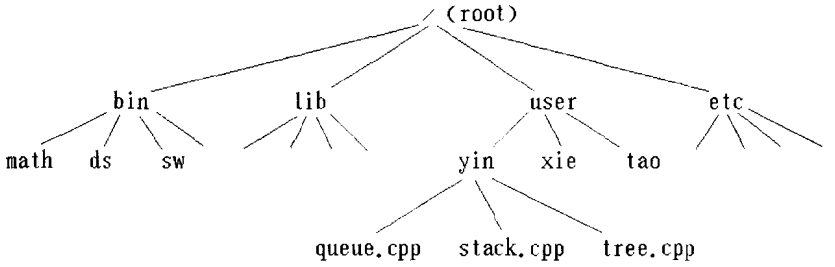


图 1.3 UNIX 文件系统的系统结构图

综上所述,在应用程序中涉及到各种各样的数据,为了存储它们,组织它们,需要讨论它们的归类及它们之间的关系,从而建立相应的数据结构,并依此实现要求的软件功能。本课程的目的就是为系统开发者提供解决此问题的基本知识,使得读者在学习完本课程之后,能够在系统开发时运用学过的知识,在设计系统体系结构的同时设计出有效的数据结构,正确、高效地实现整个系统。

1.1 什么是数据结构

古往今来,大千世界,人们会遇到各种信息,如古汉边塞用来报警的烽火、人们用语言交流的思想、银行与商店的商业交易等等。这些信息必须转换成数据才能在计算机中进行处理,因此,我们可以给数据下一个定义:

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

事实上,我们平时所用到的数据主要有两类,一类是数值性数据,包括整数、实数、复数、双精度数,主要在工程和科学计算,以及商业事务处理中使用;另一类是非数值数据,主要包括字符和字符串,以及文字、图形、语音等的数据。

从传统的观点来看,在解决日常所遇到的应用问题时,总把数据按其性质归类到一些我们称之为数据对象(data object)的集合中。在数据对象中所有数据成员(即数据元素),都具有相同的性质,它们是数据的子集。例如,整数数据对象可以是集合 N = {0, ±1, ±2,

±3,…}。英文字母数据对象可以是集合 LETTER = {'A', 'B', …, 'Z'}。

学校中的学生是更复杂一点的数据对象,它的每一个数据元素就是一个学生记录,它包括学生的学号、姓名、性别、籍贯、出生年月、成绩等数据成员或数据项,以表明学生在某一方面的属性。这些数据项可以分为两种,一种叫做初等项,如学生的性别、籍贯等,这些数据项是在数据处理时不能再分割的最小单位;另一种叫做组合项,如学生的成绩,它可以再划分为物理、化学等更小的项。通常,在解决实际应用问题时是把每个学生记录当作一个基本单位进行访问和处理的。

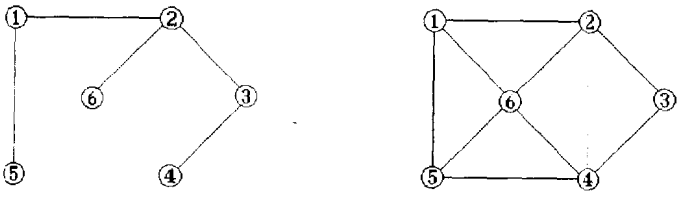
在学生选课系统中,可能以一个班级的学生记录作为学生数据对象,也可能以一个年级的学生记录作为学生数据对象。所以,如何选择数据对象将依问题而定。

一个数据对象中所有数据成员之间一定存在某种关系。例如,招生考试时把所有考生按考试成绩从高到低排队,这样,所有考生记录都处在一种有序的序列中。又例如,在 n 个网站之间建立通信网络,要求以最小的代价将 n 个网站连通,如图 1.4(a)所示,这样,在所有网站之间形成一种树形关系;反之,要求在网络中任一网站出现故障时,整个网络仍然保持畅通,这样,在所有网站之间形成一种网状关系,如图 1.4(b)所示。若综合考虑数据对象及其所有数据成员之间的关系,就可得到数据结构的定义:

数据结构由某一数据对象及该对象中所有数据成员之间的关系组成,记为:

Data_Structure = {D, R}

其中,D 是某一数据对象,R 是该对象中所有数据成员之间的关系的有限集合。



(a) 网站之间的树形关系　　(b) 网站之间的网状关系

图 1.4　n 个网站之间的连通关系

依据所有数据成员之间的关系的不同,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成员依次排列在一个线性序列中;非线性结构中各个数据成员不再保持在一个线性序列中,每个数据成员可能与零个或多个其它数据成员发生联系。此外,数据结构依据视点的不同,分为数据的逻辑结构和物理结构。

数据的逻辑结构是指从解决问题的需要出发,为实现必要的功能所建立的数据结构,它属于用户的视图,是面向问题的,如在招生系统中建立的按考分排列的考生记录的有序表格。数据的物理结构是指数据应该如何在计算机中存放,是数据逻辑结构的物理存储方式,是属于具体实现的视图,是面向计算机的。数据的逻辑结构根据问题所要实现的功能建立,数据的物理结构根据问题所要求的响应速度、处理时间、修改时间、存储空间和单位时间的处理量等建立,是逻辑数据的存储映象。

通常我们讨论数据结构,不但要讨论各种在解决问题时可能遇到的典型的逻辑结构(在本书下文中简称为数据结构),还要讨论这些逻辑结构的存储映象(在本书下文中简称为存储实现),此外还要讨论这种数据结构的相关操作及其实现。

1.2 抽象数据类型及面向对象概念

1.2.1 数据类型

首先我们回顾一下在程序设计语言中出现的各种数据类型。以C语言为例,有5种基本的数据类型,即字符型、整型、浮点型、双精度型和无值,分别用保留字 **char**, **int**, **float**, **double**和 **void** 标识。这些数据类型不但规定了使用该类型时的取值范围,而且还规定了该类型可以用的一组操作。例如,与整型相关的操作有 +、-、*、/,与浮点型相关的操作也有 +、-、*、/。然而整型的"/"操作与浮点型的"/"操作虽然它们的形式一样,却是两个不同的操作,例如整型运算 6/4 的运算结果为 1,浮点型运算 6/4 的运算结果为 1.2。因为操作"/"用于几个数据类型,故称它是一种重载的操作。

事实上,数据类型是一组性质相同的值的集合以及定义于这个值集合上的一组操作的总称。例如在高级程序设计语言中已实现了的,或非高级语言直接支持的数据结构即为数据类型。在程序设计语言中,一个变量的数据类型不仅规定了这个变量的取值范围,而且定义了这个变量可用的操作,例如,一个变量 K 定义为整型,则它可能的取值范围是 -32 767 ~ 32 767,可用的操作有双目运算符 +、-、*、/、%,单目运算符 +、-,关系运算符 <、>、<=、>=、==、!=,赋值运算符 = 等。在如C或Pascal语言这样的程序设计语言中,不但规定了一些基本的数据类型,还提供了一些构造组合类型(如数组型、构造型、文件型等)的规则,程序员可以利用这些规则,自行定义为解决应用问题所必需的数据类型,它是确切地描述数据对象、正确地进行相关计算的有效工具。

1.2.2 数据抽象与抽象数据类型 (ADTs: Abstract Data Types)

在软件设计时,常常提到"抽象"和"信息隐蔽"。那么,什么是抽象呢?

抽象的本质就是抽取反映问题本质的东西,忽略非本质的细节。对于数据的抽象,可以用一个例子说明。在计算机中使用二进制定点数和浮点数实现数据的存储和运算,而在汇编语言中则给出了各种数据的自然表示,如 15.5,1.3E10,10 等,它们是二进制数据的抽象,使用者在编程时可以直接使用它们,不必考虑实现的细节。到了高级语言,给出了更高一级的数据抽象,出现了整型、实型、字符型、双精度型等。待到抽象数据类型出现时,可以进一步定义出更高级的数据抽象,如各种表、队列、图,甚至窗口、管理器等。这种数据抽象的层次为设计者提供了有力的手段,使得设计者可以从抽象的概念出发,从整体上进行考虑,然后自顶向下,逐步展开,最后得到所需的结果。

抽象数据类型通常是指由用户定义,用以表示应用问题的数据模型,抽象数据类型由基本的数据类型组成,并包括一组相关的服务(或称操作)。抽象数据类型有些类似于 Pascal 语言中的记录(**record**)类型和C语言中的构造(**struct**)类型,但它增加了相关的服务。下面我们给出自然数(*Natural Number*)的抽象数据类型定义。

ADT *NaturalNumber* **is**

objects:

一个整数的有序子集合,它开始于0,结束于机器能表示的最大整数(*MAXINT*)。