GENERAL EDUCATION

高等学校通识教育系列教材

数据结构与算法简明教程（Java语言版）

叶小平 陈瑛 编著

清华大学出版社

GENERAL EDUCATION

高等学校通识教育系列教材

数据结构与算法简明教程（Java语言版）

叶小平　陈瑛　编著

清华大学出版社
北京

内 容 简 介

本书是“数据结构与算法”课程(Java 语言描述)的基本教材。全书突出数据逻辑结构主线,在编写思路和材料组织上具有体现整体架构、注重本质关联、彰显关键细节和强化实例讲解等特点。书中基本算法和实例实现程序都经过 Java 8 标准版(JDK 1.8 版本)平台调试运行,能够实现课程的教材学习到实验操作的有效对接。

本书可分为三部分(共 10 章):第一部分是课程概述(第 1 章);第二部分是基于内存的数据结构(第 2～7 章),包括线性结构(第 2～4 章)、树结构(第 5～6 章)、图结构(第 7 章);第三部分是高级部分(第 8～10 章),包括查找(第 8 章)、排序(第 9 章)和文件(第 10 章)。

本书可作为高等院校计算机信息科学与技术及其相关专业本科生教材,也可作为非计算机专业开设相应计算机专业基础课的教材,还可作为自学教材。

图书在版编目(CIP)数据

数据结构与算法简明教程:Java 语言版/叶小平,陈瑛编著.--北京:清华大学出版社,2016
高等学校通识教育系列教材
ISBN 978-7-302-43982-0

Ⅰ.①数… Ⅱ.①叶… ②陈… Ⅲ.①数据结构－高等学校－教材 ②算法分析－高等学校－教材 ③JAVA 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(2016)第 120551 号

责任编辑:刘向威　王冰飞
封面设计:文　静
责任校对:焦丽丽
责任印制:何　芊

出版发行:清华大学出版社
网　　址:http://www.tup.com.cn, http://www.wqbook.com
地　　址:北京清华大学学研大厦 A 座　　邮　　编:100084
社 总 机:010-62770175　　邮　　购:010-62786544
投稿与读者服务:010-62776969, c-service@tup.tsinghua.edu.cn
质 量 反 馈:010-62772015, zhiliang@tup.tsinghua.edu.cn
课 件 下 载:http://www.tup.com.cn,010-62795954

印 装 者:北京国马印刷厂
经　　销:全国新华书店
开　　本:185mm×260mm　　**印　　张**:21.5　　**字　　数**:519 千字
版　　次:2016 年 9 月第 1 版　　**印　　次**:2016 年 9 月第1 次印刷
印　　数:1～2000
定　　价:44.00 元

产品编号:069652-01

前言

“数据结构”是计算机及相关领域重要专业基础课之一。近年来，随着计算机网络技术的快速发展，Java 语言使用日趋普遍；同时，计算机相关领域对计算机软件技术使用的需求与层次也在不断地深化，非计算机专业开设诸如“数据结构与算法”等计算机课程也已逐步成为一种常态。根据不同层次学科专业编写基于 Java 语言相关教材业已形成一种日益明显的现实需求。本书就是在这种实际背景之下根据近年来教学实践编写完成的。

本书可分为三部分（共 10 章）：第一部分是课程概述（第 1 章）；第二部分是基于内存的数据结构（第 2～7 章），包括线性结构（第 2～4 章）、树结构（第 5～6 章）、图结构（第 7 章）；第三部分是高级部分（第 8～10 章），包括查找（第 8 章）、排序（第 9 章）和文件（第 10 章）。“数据结构与算法”早已成为经典课程，本书组织体系建立在常规框架之内；但由于“经典”通常都具涉及面广泛以及“重点”与“难点”同步情形，因此“数据结构”常常被认为是需要下工夫才能“教好”和花气力方可“学通”的专业基础课程之一，对于非计算机专业的相应教学来说更是如此。本书编著者和其他计算机科研与教学领域的同仁们一样，期望在本课程教学与改革方面尽自己的一份努力。通过基于 C 和 C++语言“数据结构与算法”教学积累与基于 Java 语言数据结构方面教学实践，希望在教材中体现出下述特点，以适应现实教学方面的需求。

（1）突出主线，从整体上理解相关内容。数据逻辑结构是本课程主线，课程所有内容都围绕其展开。既然是主线，就可以在各个层面上都有所体现。在教材开首“绪论”中提出“集合—线性表—树形—图”的基本逻辑结构线索，同时在讲授其后各章具体内容中也不断展开螺旋式上推强化。

① 线性结构的基础性。“线性表”是所有逻辑结构的基础，这可从下述两个方面进行描述，首先，体现在线性表内容的组织编排上：线性表的一般概念与技术，基于更新限制的线性表——“栈”和“队列”，基于元素数据类型扩充与限制的线性表——“多维数组”和“字符串”；另外，其他逻辑结构在最终技术处理的末端都能归结为线性表：通过“遍历”技术，“树”和“图”实际上转化为线性表进行处理，“查找”和“排序”初始结构以线性表为基础。

② 集合结构的灵活性。“集合”作为一种逻辑数据结构，其结构约束最为简洁，由于“查找”和“排序”内容众多、技术复杂，使用集合作为初始数据结构就为各种具体算法实现提供了灵活展开的平台，如将所需排序的数据集合视为按照“工作顺序”组成的查找表时，就可根据实际需要采用基于“顺序表”查找、基于“树表”查找和基于“散列表”查找等。

③ 逻辑结构的层次性。图型结构是最为一般的数据结构，树形可视为“无环有根”的图结构，线性表可视为至多只有唯一父结点和子结点的树结构，集合可视为按照“工作顺序”得到的线性表，由此“高层复杂”结构可以通过适当方法转化“底层基本”结构进行操作。

(2) 强调实例,从具体中把握抽象原理。“数据结构”课程中不少概念和算法都比较抽象,如树和二叉树的递归定义、模式匹配的 KMP 算法、平衡树算法和希尔算法等。其抽象性表现在两个方面,首先是一般描述抽象,其次是编程实现抽象。大凡抽象之物都是人们出于各种需要对“具体实际”进行了不同层面和角度的“加工”,越是抽象,就越需要弄清其“直观”的本源,越需要通过“具体”的形象作为理解与把握方面的支撑。实际上,只有建立在“直观实例”之上的理解才是真正哲学和心理学意义上的理解。因此,选择和编排好相应的实例图示是讲清讲透抽象概念与算法的关键点。本书努力做到在保持科学性和逻辑性前提之下,凡是重要的概念、原理和算法都配有相应的直观图解,而难度较高的部分其相应图示都尽量详尽细微。实践证明,这样做教师可能会感觉“好讲”,同学们也可能会感到“易学”。

(3) 注重关联,从逻辑网络中建构知识。“数据结构”课程涉及概念和算法众多,初看起来似乎内容庞杂。实际上,作为一门经典课程,经过无数专家们的千锤百炼,早已成为一个逻辑脉络清晰与彼此精密套接的科学体系,问题是在教和学的过程当中实时进行把握和梳理,并不断地加以强调和体现。各个知识点只有放在整体体系合适的部位才能彰显其意义和作用,才能具有“鲜活”机体部件的价值。“数据结构”就是数据元素之间的“组织关系”,这种关系从计算机逻辑处理角度上考虑,是按照集合的“具有相同类型”的非结构特征语义关系、线性表的“前驱/后继”顺序关系、树形结构的“双亲/子女”层次关系和图结构的“邻接/路径”的到达关系等由简单到复杂的“正向”逐次推进;而如果从技术处理角度上来看,却又是“图”通过“生成树”与树关联、“树”通过“遍历”与线性表关联这样由复杂到简单的“反向”递解推回。此外,“图”的各种概念繁多,但其中逻辑框架却可以理清:图的基础元语是“顶点”和“边(弧)”,表示数据元素的顶点之间的关系元语是“邻接”。具邻接关系的顶点可以看作具有“强”关系,由非邻接关系分出一类“弱”关系——路径相连关系,这与树结构中从“父子关系”到“祖先子孙关系”演进过程类似。图的其他众多概念与算法都以此为初始,例如所有顶点都具“邻接关系”的图是“完全图”、所有顶点都具“路径连接”的图是“连通图”,再继续一般化,在非连通图中分解出连通分支等,如此这般,各基本概念都在图的元语整体框架中找到自己的位置,由此又有对这些概念进行处理验证和应用的各类算法,分散的个别对象形成同一的严密整体。

(4) 突出细节,由精微处体现关键。从某种意义上来看,“数据结构与算法”课程中许多概念,特别是算法都堪称“艺术精品”,而艺术品成功的关键在于其有“闪光细节”的展现。其实,课程中许多“细节”还是理解和掌握相应问题的关键所在,正如常说的“细节决定品质”和“一滴水中见太阳”。在理清整体脉络框架的前提之下,讲清有关“精微细节”,不仅可以有效深刻地掌握相应概念算法,同时也可以使得略显枯燥的课程能不时闪现出自己特有的魅力,激发学习者的学习热情。如在相关概念当中,单链表头结点的意义、链栈和链队不设头结点的缘由、循环队列辅助存储单元 rear 的设置、广义表嵌套性与数据树形结构的关联以及二分查找的中点 mid 设计等。再如相关算法当中,KMP 算法“已有匹配信息”有效重用,顺序查找算法中“监视哨”的设立,希尔排序中的“跳跃式”分组,快速排序中的“大跨步”移动数据元素和堆排序中的“输出和调整”策略等等。这些也许并不适合仅做“描述性”的一般讲授,可能需要参透讲清,这些“细节”实际上正是课程极为精彩的组成部分,既是理解和把握相关原理技术的“牛鼻子”,也是值得课程学习者欣赏和玩味的闪光点。

当然,上述只是我们编写教材的一些基本设想并依此进行了初步实践,相关考虑还多有

商榷之处，即使有了一些想法也未必在书中就得到比较满意的实现。实际上，由于编著者水平所限，教材中疏漏和不足之处在所难免，恳请专家和同仁们不吝指教。

本书编写过程中参考借鉴了国内外相关教材，其中主要部分参见教材的参考书目，特在此向各位专家作者表示衷心感谢！

本书可作为计算机专业和非计算机相关专业“数据结构与算法”教材，也适合于具有Java语言基础的读者自学。

教材由叶小平组织统筹，其中第1、2、3、8和9章主要由叶小平编写，第4、5、6、7和10章主要由陈瑛编写，何文海参与第7、10章工作并编写测试教材中主要程序。

编　者

2016年5月

目　录

第 1 章　绪论 …… 1

1.1　数据与数据类型 …… 1
1.1.1　数据的基本概念 …… 1
1.1.2　数据项与数据元素 …… 2
1.1.3　数据类型与抽象数据类型 …… 3
1.2　数据逻辑与存储结构 …… 7
1.2.1　数据逻辑结构 …… 7
1.2.2　数据存储结构 …… 8
1.3　数据运算与算法 …… 10
1.3.1　数据运算 …… 10
1.3.2　算法及其基本要求 …… 11
1.3.3　算法设计与分析 …… 12
1.4　“数据结构”课程的地位与教材内容 …… 15
1.4.1　“数据结构”课程的地位 …… 15
1.4.2　本书内容组织 …… 16
本章小结 …… 17

第 2 章　线性表 …… 21

2.1　线性表概念 …… 21
2.1.1　线性表逻辑结构 …… 21
2.1.2　线性表 ADT 描述 …… 22
2.2　线性表的顺序存储 …… 24
2.2.1　顺序存储结构 …… 24
2.2.2　顺序表的基本操作 …… 26
2.3　线性表的链式存储 …… 31
2.3.1　单链表概念 …… 31
2.3.2　单链表的基本操作 …… 34
2.3.3　线性表存储结构比较 …… 40
2.4　链式存储其他实现方式 …… 41
2.4.1　循环链表 …… 41

2.4.2 双向链表 …… 43
2.4.3 静态链表 …… 46
2.5 单链表应用及迭代器 …… 47
2.5.1 单链表倒置 …… 47
2.5.2 两个有序链表合并 …… 48
2.5.3 一元多项式计算 …… 49
2.5.4 迭代器 …… 52
本章小结 …… 54

第 3 章 栈和队列 …… 57

3.1 栈 …… 57
3.1.1 栈基本概念 …… 57
3.1.2 栈的顺序存储 …… 59
3.1.3 栈的链式存储 …… 62
3.2 栈的应用 …… 65
3.2.1 数制转换 …… 65
3.2.2 栈在递归中的应用 …… 66
3.2.3 栈在括号匹配中的应用 …… 74
3.2.4 表达式求值 …… 76
3.2.5 迷宫求解 …… 80
3.3 队列 …… 84
3.3.1 队列基本概念 …… 84
3.3.2 队列的顺序存储 …… 86
3.3.3 队列的链式存储 …… 91
3.4 队列的应用 …… 94
本章小结 …… 98

第 4 章 数组和串 …… 101

4.1 数组 …… 101
4.1.1 二维数组 …… 101
4.1.2 矩阵的顺序表示与实现 …… 102
4.1.3 特殊矩阵的压缩存储 …… 103
4.1.4 稀疏矩阵的压缩存储 …… 107
4.2 串 …… 114
4.2.1 串及相关概念 …… 115
4.2.2 串的基本操作 …… 115
4.2.3 串的顺序存储 …… 117
4.2.4 串的链式存储 …… 121
4.2.5 串的模式匹配 …… 122
本章小结 …… 129

第 5 章　树 …… 131

5.1　树结构及相关概念 …… 131
5.1.1　树的基本概念 …… 132
5.1.2　树的相关概念 …… 135
5.2　树的存储 …… 136
5.2.1　父结点表示法存储 …… 136
5.2.2　子结点表示法存储 …… 137
5.2.3　左子/右兄弟结点表示法存储 …… 140
5.3　树的遍历 …… 141
5.3.1　广度优先遍历 …… 141
5.3.2　深度优先遍历 …… 143
本章小结 …… 145

第 6 章　二叉树及应用 …… 147

6.1　二叉树的概念及性质 …… 147
6.1.1　二叉树及其相关概念 …… 147
6.1.2　二叉树的基本性质 …… 151
6.2　二叉树的存储 …… 152
6.2.1　二叉树的顺序存储 …… 152
6.2.2　二叉树的链式存储 …… 153
6.3　二叉树的遍历 …… 157
6.3.1　先序遍历、中序遍历与后序遍历 …… 157
6.3.2　基于递归遍历算法 …… 158
6.3.3　基于非递归遍历算法 …… 161
6.4　线索二叉树 …… 167
6.4.1　线索与线索二叉树 …… 167
6.4.2　线索二叉树创建 …… 169
6.4.3　线索二叉树操作 …… 170
6.5　Huffman 树及其应用 …… 171
6.5.1　编码及分类 …… 172
6.5.2　Huffman 树 …… 172
6.5.3　基于顺序存储 Huffman 树 …… 174
6.5.4　Huffman 编码 …… 177
6.6　树与二叉树的转换 …… 180
6.6.1　树转换为二叉树 …… 180
6.6.2　二叉树还原为树 …… 181
6.6.3　森林与二叉树的转换 …… 182
本章小结 …… 183

第 7 章　图 …… 185

7.1　图的数据结构 …… 185
7.1.1　图的基本概念 …… 185
7.1.2　路径与连通 …… 188
7.2　图的存储 …… 190
7.2.1　基于邻接矩阵存储 …… 190
7.2.2　基于邻接表存储 …… 193
7.3　图的遍历 …… 197
7.3.1　深度优先遍历 …… 197
7.3.2　广度优先遍历 …… 199
7.4　生成树与最小生成树 …… 202
7.4.1　图的生成树 …… 202
7.4.2　无向连通图最小生成树 …… 204
7.5　有向网图的应用 …… 214
7.6　有向无环图的应用 …… 220
7.6.1　AOV 网与拓扑排序 …… 221
7.6.2　AOE 网与关键路径 …… 224
本章小结 …… 231

第 8 章　查找 …… 234

8.1　数据查找 …… 234
8.2　基于线性表的查找 …… 236
8.2.1　顺序查找 …… 237
8.2.2　分块查找 …… 238
8.2.3　二分查找 …… 240
8.3　基于二叉树的查找 …… 244
8.3.1　二叉查找树概念 …… 244
8.3.2　基于二叉查找树的查找 …… 245
8.3.3　二叉查找树插入与生成算法 …… 247
8.3.4　二叉查找树删除 …… 249
8.3.5　平衡二叉树 …… 253
8.4　基于散列表的查找 …… 257
8.4.1　常用散列函数构建 …… 258
8.4.2　散列冲突处理 …… 260
本章小结 …… 265

第 9 章　排序 …… 267

9.1　数据排序 …… 267

9.1.1 排序的基本概念 …… 267
9.1.2 排序算法性能分析 …… 270
9.2 插入排序 …… 271
9.2.1 直接插入排序 …… 271
9.2.2 二分插入排序 …… 275
9.2.3 Shell 排序 …… 277
9.3 交换排序 …… 279
9.3.1 冒泡排序 …… 279
9.3.2 快速排序 …… 281
9.4 选择排序 …… 285
9.4.1 直接选择排序 …… 285
9.4.2 堆排序 …… 287
9.5 归并排序 …… 293
9.6 外排序 …… 295
9.6.1 外排序的基本步骤 …… 296
9.6.2 败者树 k-路归并算法 …… 297
9.6.3 k-路归并算法实现 …… 298
本章小结 …… 299

第 10 章 文件 …… 304

10.1 文件及其分类 …… 304
10.1.1 文件概述 …… 304
10.1.2 文件结构与操作 …… 305
10.2 顺序文件 …… 308
10.2.1 顺序文件存储结构 …… 308
10.2.2 顺序存储的实现 …… 308
10.3 索引文件 …… 309
10.3.1 索引表与索引文件 …… 309
10.3.2 ISAM 文件 …… 311
10.3.3 VSAM 文件 …… 313
10.4 动态索引 B-树 …… 314
10.4.1 B-树 …… 315
10.4.2 B^+ 树 …… 321
10.5 散列文件 …… 322
10.6 多关键码文件 …… 324
10.6.1 多重表文件 …… 325
10.6.2 倒排文件 …… 326
本章小结 …… 326

参考文献 …… 329

第1章 绪 论

作为计算机学科重要的专业基础课，“数据结构与算法”早在1968年就开始作为一门独立课程出现在高校计算机专业课程设置当中。计算机需要处理的对象是“数据”，而数据经历了一个由主要用于“科学计算”的纯粹“数值型数据”到主要用于字符、表格以及图形图像等具有一定结构的“非数值型数据”的发展过程。事实上，现今计算机都在大量和经常性地存储、加工和处理各类非数值型数据。无论是数值型还是非数值型数据，其进入计算机的前提都是需要建立相应数据模型或数据结构。数值型数据的模型通常是数学公式或数学方程，其着重点在于数据“值”的精确或近似计算；非数值型数据的模型主要是线性表、树和图等基本结构，其着重点在于数据元素之间的“关联”描述和“结构”处理。本教材内容就是学习非数值型数据模型，也就是数据结构以及建立在其上相应数据操作，它是计算机学科的核心课程之一，也是各类后续课程和各种应用设计的必备基础。本章是全书内容的概述，目的在于对课程学习内容有一个整体性的概括了解，同时提出了在以后各章学习中需要深入理解和熟练使用的各种基本概念、原理与方法。在本章内容学习过程中，需要注意以下问题：

- 数据、数据元素和数据对象等基本概念；
- 数据类型和抽象数据类型的联系区别及其意义；
- 数据逻辑结构和存储结构，存储结构需要同时存储数据本身和数据元素间关系；
- 算法概念与特性，算法的效率评估。

1.1 数据与数据类型

理解数据结构意义和作用的前提是掌握数据基本概念与相关特征。

1.1.1 数据的基本概念

作为信息的载体，从某种意义上考虑，数据可以看作是能够被人们识别、存储和加工处理的各种形式的总和，其表现形式或为实体印记（如古时的结绳刻痕等），或为抽象符号（如现代的数目字符等）。但“数据”是一个“元概念”，难以给出严格意义上的定义，通常是基于其基本特征进行“描述性”的界定。在计算机应用环境下来讲，可以将数据看作是能够进入计算机并能由计算机处理的抽象符号集合。

数据是计算机程序加工的原始材料，其要点一是“符号”集合，二是表达一定的含义。但数据的含义或语义依赖于人们使用时的语境，常常需要进行解释才能予以确定，即从数据中获取信息。例如28在直观意义之下应该可以看作数据，但在没有特定的背景之下，通常难以确定28是“年份”还是“年龄”，是天气的摄氏温度还是两个地点间的距离。

从计算机处理数据角度来看,有“数值型”和“非数值型”数据之分。

(1) **数值型数据**的特征是通过简单的数制转换进入计算机并由计算机“直接计算”结果,这类数据主要包括整数、实数和复数等。计算机对数值型数据进行的操作主要是加减乘除等四则运算,这类数据主要应用于计算方法、工程计算和商务处理等领域中的数值计算问题,例如,利用数值分析方法解代数方程的程序的处理对象都是整数和实数。数值型数据的“直接计算”往往需要应用深奥的数学原理和建立复杂的数学公式或方程。数值型数据是“计算数学”或“数值分析”等学科的基本研究对象。

(2) **非数值型数据**的特征是需要经过各种比较复杂的编码方式才能进入计算机,计算机系统对这些数据主要不是进行“直接计算”,而是着眼于数据之间各种关系的存储和处理,其主要研究在文字处理、多媒体管理以及更为复杂的逻辑推理等领域当中的数据组织、管理和查找等与“直接计算”无关的课题。这类数据主要包括字符、图形、图像、音频和视频等,非数值型数据的处理方法相对简单,而数据间关系的计算机存储与管理却相当复杂,涉及的数据量也极为庞大。整个计算机学科特别是本课程主要讨论非数值型数据。

1.1.2 数据项与数据元素

“数据”自身不能严格定义,但对“数据”进行描述性界定之后,却可以在其基础之上进一步地引入数据项、数据元素和数据对象等相关概念,以便有效地描述和使用数据概念。

1. 数据项

研究任何问题都需要设定一个所处理项目的最小“粒度(单位)”,计算机管理数据的“语义粒度”就是“数据项”。

数据项(Data Item):计算机进行数据语义(意义)管理中不可再分割的最小单位,具有自身独立含义。

表 1-1 是一个教学班级同学的学籍表格,用于存储学生相关数据。其中每一行是一个学生的数据记录,这些记录中每一个字段就是该记录的一个数据项,如姓名、住址等。数据项也称为数据属性,它在数据记录中具有“原子性”,如“赵向伟”这个姓名数据项不能再进行分解,否则难以得到学生姓名特征的实际辨识。

表 1-1 学生学籍表(Students)

学号 Sno	姓名 Sname	性别 Ssex	籍贯 Snplace	年龄 Sage	住址 Sadd
2015101	赵向伟	男	湖南	20	长沙
2015102	李爱萍	女	广东	19	广州
⋮	⋮	⋮	⋮	⋮	⋮

2. 数据元素

实际问题中不同数据项一般具有相关性,例如,一个同学的“学号”、“姓名”、“性别”和“籍贯”等都是个别意义下的数据项,但彼此相关,都从某个侧面表达了该同学的相关信息,在实际中通常会“一并”使用。基于此考量,计算机技术处理的不是单个数据项,而是相关数据项的“整体”,这就是“数据元素”的概念。

数据元素(Data Elements):计算机对数据进行技术处理过程中一个不再进行分割的最小单位,它在计算机程序中作为一个整体考虑。

表 1-1 中第 2 行起以后每一行都是“学生学籍表”中的一个数据元素。数据元素也称为(数据)记录,在关系数据库中称为元组。数据元素可分为“单数据项”和“多数据项”数据元素两类情形。

(1) **单数据项元素**由单个数据项组成,也称为或原子型数据元素,如整数 8、字符 d 等。

(2) **多数据项元素**由多个数据项组成,如程序语言中常用的“结构”和“数组”等。表 1-1 中从第 2 行开始以后的数据元素就是由学号、姓名、性别、籍贯、出生年月和住址等 6 个数据项组成,而这些数据项具有不尽相同的类型特征,例如“学号”是整型、“姓名”是字符型和“出生年月”是日期型等。这种由不同类型的多个数据项组成的就是“结构”型数据元素。如果数据元素是由相同类型的多数据项构成,这样的数据元素就是“数组”,例如全班同学“数据结构”课程期末考试成绩数据项组成的数据元素就是一个(一维)数组。

在 Java 中,数据元素表示为类的成员变量,而相应数据项的访问权限通常都为 private。上述“学生学籍表”就是一个类,其中类成员部分用 Java 表示如下。

```
private class students{
    private long Sno;
    private char Sname;
    private char Ssex;
    private char Snplace;
    private int Sage;
    private char Sadd;
}
```

3. 数据对象

计算机处理数据是一个对多数据元素“同时”操作的过程,因此,一个计算过程中的材料不会是单个数据元素而是“同类”数据元素的整体,这个“整体”就是“数据对象”概念。

数据对象(Data Object):“同类”数据元素的集合,它是相应研究问题的一个子集。

例如,整数数据对象是由单项数据元素整数构成的集合 N={0,±1,±2,…},字母数据对象是由单项数据对象字符构成的集合 C={'A','B',…,'Z'};而表 1-1 就是同型的多项数据项构成的数据元素——记录或元组的集合,数据对象在关系数据库中也称为关系表。

本课程中使用的“数据”概念在没有特殊说明情况下都是指“数据元素”。

1.1.3 数据类型与抽象数据类型

使用计算机处理数据,首先需要解决下述问题。

(1) 计算机存储。为进入计算机数据分配适当大小的存储空间,然后再根据分配的存储空间进行具体实际的存储。

(2) 计算机操作。对于存储的数据能够进行相关的数据运算,运算需要具有封闭性,运算结果仍然能够按照参与运算对象的存储格式进行存储并能“再”参与运算。

数据通常需要解释其语义,描述“语义”的基础是“数据值”,因此数据总是与“值”密切相关,解决上述问题也需要根据“数据值”进行探讨。通常是对将要使用的数据规定其“值”的种类,这样在使用某种高级程序设计语言进行数据处理过程中,就需要根据数据“值”的“种类”来刻画程序操作对象的特征并完成相应存储与操作。程序设计语言当中“数据类型”就是一个用于刻画程序操作对象即数据“值”种类形态的基本概念。

1. 数据类型

数据元素中每个数据项主要是通过将其作为程序设计语言中的变量、常量或表达式而进入计算机,而数据项取值需要有一个表明其"值"特性的且确定的数据"类型"。

数据类型(Data Type):一组性质相同的值的集合以及定义在该集合上一组操作。数据类型需要明显或隐含地规定计算机数据处理期间变量或表达式所有可能的取值范围和在相应数据值上的各种操作。因此,"数据类型"概念实际上包含如下两方面信息。

- 该类型数据可能的取值范围,即数据值的特定集合;
- 该类型数据可执行的一组运算,即数据值的特定操作。

数据类型概念来源于高级程序设计语言,其本质在于给定了数据的类型就确定了数据的存储方式和相应操作。例如,C 语言里的整型、字符型等数据类型就是数据类型实例,整型数据取值范围是-32 768—32 767,可执行加、减、乘、除和取模等一组运算;字符型数据取值范围为 ASCII 码范围所确定,可以执行字符串的串接与匹配等基本操作。

数据类型是数据取"值"特性的刻画和相关数据操作的设定,按照数据"值"的不同组织形式和特点,可将数据类型分为"内置数据类型"和"用户定义类型"。

(1) **内置数据类型**。如 C 语言中的整型、实型、字符型、指针型和布尔型等基本类型等,其基本特征是数据值不可进行再分解,通常由语言系统预先内置,用户只需直接调用。内置数据类型也称为原子类型,具有原子类型的数据元素通常由单个数据项组成。

(2) **用户定义类型**。如 C 语言中的"数组"和"结构"等都是属于此种类型,其基本特征是数据值可以进行分解,相应数据元素由多数据项组成。其中,"数组"类型中所有数据项都属于同一数据域即具有相同数据类型,既可以是同一原子数据类型,也可以是同一的已由用户定义数据类型;"结构"数据类型中数据项通常属于不同数据域即具有不同数据类型。如表 1-1 中的学籍记录就是一个用户定义数据类型实例,它由 6 个数据域组成,每个数据域都定义了各自数据类型。用户定义类型也称为复杂类型,在高级程序设计语言中,复杂类型由用户根据需要定义,具有很大的灵活性和方便性。

在高级程序设计语言中引入数据类型概念的意义在于:

(1) 向计算机解释进入内存中数据的值含义和操作特征。

(2) 将数据的计算机操作与实现细节封装,对用户不必了解的底层信息进行屏蔽。

Java 中使用的数据类型主要有内置的基本数据类型和引用数据类型,如图 1-1 所示。

各种基本数据类型存储空间与取值范围如表 1-2 所示。

表 1-2 Java 数据类型存储与取值

类型	存储	取 值 范 围
byte	8b	-128～127
short	16b	-32 768～32 767
int	32b	-2 147 483 648～2 147 483 647
long	64b	-9 223 372 036 854 775 808～9 223 372 036 854 775 807
float	32b	-3.4E38～-3.4E38
double	64b	-1.7E318～-1.7E318
char	16b	Unicode
boolean		true,false

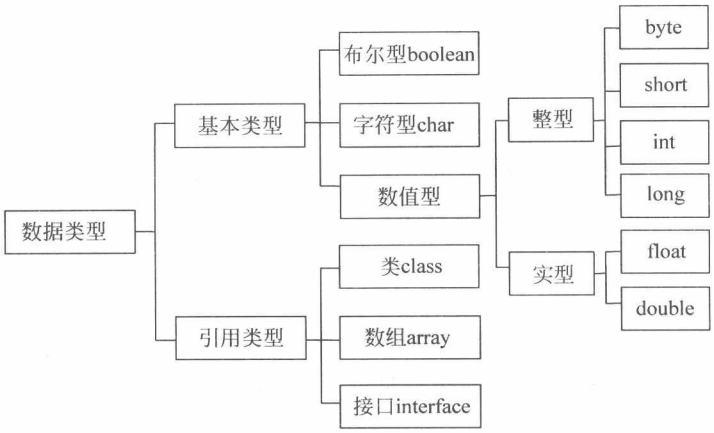

图 1-1　Java 数据类型

在 Java 中，基本数据类型按其取值范围可表示为(从低级到高级)(byte，short，char)→int→long→float→double，低级类型变量可直接自动转换为高级类型变量，高级类型变量转换为低级类型变量则需强制转换，并可能导致溢出或精度下降。byte、short、char 这 3 种类型是平级的，因此不能自动转换，但可以强制转换。Java 中数据类型合法转换如图 1-2 所示，其中实线、箭头表示自动转换，虚线箭头表示可强制转换。

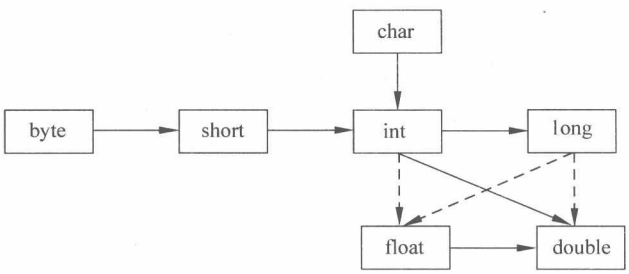

图 1-2　Java 基本数据类型合法转换

还有其他转换如利用包装类进行转换、字符串型与其他数据类型的转换等，可查阅类库中各个类提供的成员方法，这里不再详述。

2. 抽象数据类型

如前所述，引入数据类型目的是明确数据值的相关信息，以便机器分配相应存储空间和调用相关操作，以及屏蔽用户不必了解的计算机底层信息，但这样会涉及到不同机器的硬件属性。在计算机发展过程中，由于各阶段处理器不同，对于同一种数据类型数据的实现(存储)与处理(操作)也可能会有所不同。但从更高层面来看，计算机所需处理数据的相互间关系与相应数据操作在“逻辑”上却不会因为机器不同而改变。例如“整数”之间的“大小”关系和整数的四则运算操作在逻辑描述层面上对于任何机器都会相同。随着计算机技术的发展，特别是广泛地使用面向对象方法思想，人们发现需要将数据中这种不依赖于具体实现的一组逻辑特性和数据操作提取出来以适应更为一般的应用，这就引入了“抽象数据类型”概念。

抽象数据类型(Abstract Data Type，ADT)：基于某种应用的数据模型(结构)和定义于

该模型(结构)上的一组基本操作集合。ADT 的组成与描述如下。

ADT 抽象数据类型名称

{

数据对象——数据元素的组成定义;

数据关系——数据结构的基本描述;

数据操作——基本操作的函数声明;

}

ADT 抽象数据类型名称

数据对象、数据关系和数据操作是 ADT 的 3 个要素。

(1) 数据对象主要包括数据对象中数据元素所属的数据类型(例如,"年龄"为整型)和所在集合界定(例如,年龄为 0～150 之间的整数)。

(2) 数据关系即数据结构,它是基于给定数据逻辑结构之上的具体描述。

(3) 数据操作的相关函数特别是用户自定义函数特质的声明(不包括具体程序设计语言的实现),其基本声明格式如下。

- **基本函数名称**(参数列表):主要有赋值参数(为操作提供数据输入)和以 & 引导的引用参数(提供数据输入和结果输出);
- **初始条件**:这里的参数初始条件表示操作开始之前的数据结构与参数条件,当初始条件不满足时,停止操作执行并返回出错信息;
- **操作结果**:主要用于描述操作完成之后数据结构的改变情形及其需要返回的结果,其中当初始条件不满足时则无操作结果。

例如,某用户定义在线性表 list 下的一组基于 ADT 的基本操作可以表示为:

```
int InsertPre(List list,int i,int x)
```

在 list 中的第 i 个位置之前插入整型元素 x,并返回插入成功与否的标志。

```
int DeletePos(List list,int i)
```

在 list 中的删除第 i 个位置上的元素,并返回删除成功与否的标志。

⋮

3. ADT 作用与意义

ADT 本质在于确定所定义数据的一组逻辑属性和建立其上的数据操作,但不涉及这些属性在各类计算机内部的表示与实现。在 ADT 框架下,机器内部结构的任何变化都不会对用户的外部使用带来影响。在实际应用过程当中,ADT 通常是由用户定义的具有一组基本数据操作的数据类型。

ADT 与数据类型在严格意义上可看作是属于同一范畴,但两者具有不同特征。

(1) 数据类型来源于程序设计语言,或多或少带有"面向机器"的痕迹,可以将其看作是在"机器内"的数据界定。

(2) ADT 来源于更高抽象层次,具有不依赖于具体机器、更强调操作和更注重动态(数据操作可根据需要增删,操作具体含义也不是一成不变)的意味,具有"面向问题(对象)"的特征,可以将其看作是在"机器外"的数据抽象。