高等学校规划教材

省精品课程教材

数据结构教程（C++版）

吉根林　陈　波　主编
王　琼　周俊生　于　泠　编著

计算机学科教学计划

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

高等学校规划教材
省精品课程教材

数据结构教程
（C++版）

吉根林　陈　波　主编
王　琼　周俊生　于　泠　编著

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

内 容 简 介

本书是省精品课程的教学成果，全书共分 10 章，介绍各种常用的数据结构，包括线性表、栈、队列、串、数组、特殊矩阵、广义表、树、二叉树、图等；阐述各种数据结构的基本概念、逻辑关系、存储结构、操作运算及其实现算法；介绍各种常用的查找算法和排序算法，并对各种算法的性能进行分析。书中使用 C++类定义各种数据结构，利用 C++伪代码描述算法；给出了许多经典算法和典型题例；每章均附有小结、习题和上机实验题；附录给出了 5 套课程考试样卷和 5 道课程设计题。

本书既注重基本原理，又重视算法实现；既体现先进性，又强调实用性；内容丰富，重点突出，条理清晰，由浅入深。本书的 PPT 课件和相关教学资源可从江苏省精品课程和南京师范大学精品课程“数据结构”网站 http://mcs.njnu.edu.cn/datastructure/index.asp 下载。

本书可作为高等学校计算机、软件工程、信息与计算科学、信息管理与信息系统等专业教材，也可供计算机软件开发人员参考。

图书在版编目（CIP）数据

数据结构教程（C++版）/ 吉根林，陈波主编. —北京：电子工业出版社，2009.2
高等学校规划教材
ISBN 978-7-121-08061-6

Ⅰ. 数… Ⅱ. ①吉… ②陈… 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. TP311.12 TP312

中国版本图书馆 CIP 数据核字（2008）第 211139 号

策划编辑：童占梅
责任编辑：童占梅
印　　刷：北京市天竺颖华印刷厂
装　　订：三河市鑫金马印装有限公司
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：787×1 092　1/16　印张：15.25　字数：380 千字
印　　次：2009 年 2 月第 1 次印刷
印　　数：4 000 册　　定价：26.00 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”是计算机科学与技术、软件工程、信息与计算科学、信息管理与信息系统等相关专业最重要的专业基础课之一，主要研究分析计算机存储、组织数据的方式和相关操作运算算法。通过本课程学习，要求学生掌握数据结构和算法的基本概念和技术，掌握数组、线性表、栈、队列、串、广义表、树、二叉树、图等常用数据结构及相关算法，以及排序、查找等重要技术，能针对应用问题选择合适的数据结构，并设计相应的操作运算算法。

南京师范大学“数据结构”课程经过多年建设与探索，进行了教学内容与教学方法的改革与实践，提高了教学质量，被评为**江苏省精品课程**。我们在教学中贯彻了下列指导思想：

（1）基础性。数据结构、算法和程序设计是计算机科学的核心，本课程应为学生的软件开发能力的培养打下扎实的基础。

（2）系统性。本课程以系统的观点研究数据组织和操作算法，必须在抽象思维、算法设计等方面加强学生的能力培养。

（3）先进性。本课程的新思想和新方法不断产生，必须不断更新教学内容以拓宽学生的知识面，适应计算机应用和发展的需要。

（4）实践性。本课程是一门实践性很强的课程，在“数据结构”的课程实验中不仅要训练学生的计算机实验技能和操作能力，更应包括设计算法的创造性实验能力。

本教材在编写过程中集作者多年“数据结构”精品课程的教学经验，体现科学性、先进性和实用性原则，既注重基本原理，又重视算法实现；力求内容丰富，重点突出，条理清晰，由浅入深，语言流畅，具有特色。全书使用**C++类定义各种数据结构**，利用**C++伪代码描述算法**；**给出许多经典算法和典型题例**；**每章均附有小结、习题和上机实验题**；**附录给出了5套课程考试样卷和5道课程设计题**，以供教学参考。

本教材共分10章。

第1章简要介绍数据结构和算法的基本概念；

第2～5章介绍线性结构及其算法，包括线性表、栈、队列、串、数组和特殊矩阵；

第6～8章介绍非线性结构及其算法，包括广义表、树、二叉树、图；

第9章介绍各种常用的查找算法；

第10章介绍各种常用的排序算法。

本书的**PPT课件和相关教学资源可从江苏省精品课程和南京师范大学精品课程“数据结构”网站http://mcs.njnu.edu.cn/datastructure/index.asp下载**。

本教材由南京师范大学“数据结构”课程组共同编写。其中，第 1 章和第 10 章由吉根林教授编写；第 2 章和第 3 章由陈波副教授编写；第 6 章和第 7 章由王琼副教授编写；第 5 章和第 8 章由周俊生副教授编写；第 4 章和第 9 章由于泠副教授编写。全书由吉根林和陈波担任主编，并最后统稿、修改和定稿。本书出版过程中得到了电子工业出版社的大力支持，在此表示衷心的感谢！

由于作者水平有限，书中难免存在不妥之处，敬请读者批评指正。

编者 E-mail：glji@njnu.edu.cn。

编　者

目　录

第 1 章　绪论 …… 1
1.1　数据结构课程的研究内容 …… 1
1.2　基本概念及术语 …… 2
1.3　算法与算法分析 …… 5
1.3.1　算法 …… 5
1.3.2　算法分析 …… 7
本章小结 …… 9
习题 1 …… 10
上机实验题 1 …… 11
第 2 章　线性表 …… 12
2.1　线性表的基本概念 …… 12
2.2　线性表的存储结构 …… 12
2.2.1　顺序存储结构 …… 13
2.2.2　链式存储结构 …… 14
2.3　线性表的操作算法 …… 16
2.3.1　顺序表的操作算法 …… 16
2.3.2　链表的操作算法 …… 21
2.4　线性表的应用 …… 29
2.5　顺序表和链表的综合比较 …… 33
本章小结 …… 33
习题 2 …… 34
上机实验题 2 …… 34
第 3 章　栈和队列 …… 36
3.1　栈 …… 36
3.1.1　栈的基本概念 …… 36
3.1.2　栈的存储结构 …… 36
3.1.3　栈的操作算法 …… 38
3.1.4　栈的应用 …… 41
3.2　队列 …… 46
3.2.1　队列的基本概念 …… 46
3.2.2　队列的存储结构 …… 47
3.2.3　队列的操作算法 …… 49
3.2.4　队列的应用 …… 51
本章小结 …… 52
习题 3 …… 52
上机实验题 3 …… 53

第 4 章　串 …… 54
4.1　串的基本概念 …… 54
4.2　串的存储结构 …… 54
4.2.1　串的顺序存储结构 …… 54
4.2.2　串的链式存储结构 …… 55
4.3　串的操作算法 …… 55
4.3.1　串的基本操作算法 …… 56
4.3.2　串的模式匹配 …… 57
4.3.3　串的应用——文本编辑软件 …… 62
本章小结 …… 63
习题 4 …… 63
上机实验题 4 …… 64
第 5 章　数组和特殊矩阵 …… 65
5.1　数组 …… 65
5.1.1　数组的基本概念 …… 65
5.1.2　数组的存储结构 …… 65
5.2　特殊矩阵的压缩存储 …… 66
5.2.1　对称矩阵的压缩存储 …… 67
5.2.2　三角矩阵的压缩存储 …… 67
5.2.3　对角矩阵的压缩存储 …… 68
5.2.4　稀疏矩阵的压缩存储 …… 69
本章小结 …… 74
习题 5 …… 74
上机实验题 5 …… 74
第 6 章　广义表 …… 76
6.1　广义表的基本概念 …… 76
6.2　广义表的存储结构 …… 77
6.2.1　广义表中结点的结构 …… 77
6.2.2　广义表的存储结构举例 …… 77
6.3　广义表的操作算法 …… 79
6.3.1　构造算法 …… 79
6.3.2　遍历广义表 …… 80
6.3.3　广义表算法举例 …… 81
本章小结 …… 83
习题 6 …… 83
上机实验题 6 …… 84
第 7 章　树和二叉树 …… 85
7.1　树的概念和性质 …… 85
7.1.1　树的定义 …… 85
7.1.2　树的基本术语 …… 86

7.1.3 树的基本性质 …… 87
7.2 二叉树的概念和性质 …… 88
7.2.1 二叉树的定义 …… 88
7.2.2 二叉树的基本性质 …… 89
7.3 二叉树的存储结构 …… 91
7.3.1 二叉树的顺序存储结构 …… 91
7.3.2 二叉树的链式存储结构 …… 92
7.4 二叉树的遍历 …… 95
7.4.1 二叉树遍历的概念 …… 95
7.4.2 二叉树遍历算法 …… 98
7.4.3 二叉树的构造和析构算法 …… 100
7.5 二叉树的其他操作算法 …… 104
7.6 线索二叉树 …… 107
7.6.1 线索二叉树的概念 …… 107
7.6.2 线索二叉树的存储结构 …… 109
7.6.3 线索二叉树的操作算法 …… 109
7.7 树的存储结构与算法 …… 113
7.7.1 树的存储结构 …… 113
7.7.2 树的操作算法 …… 118
7.8 Huffman 树与 Huffman 编码 …… 122
7.8.1 Huffman 树的定义 …… 122
7.8.2 Huffman 树的构造 …… 123
7.8.3 Huffman 编码与译码 …… 126
7.8.4 Huffman 树的其他应用——程序设计流程优化 …… 127
7.9 树与等价类 …… 129
7.9.1 等价类问题 …… 129
7.9.2 等价类的实现 …… 129
7.9.3 性能分析与改进 …… 130
本章小结 …… 132
习题 7 …… 132
上机实验题 7 …… 133
第 8 章 图 …… 135
8.1 图的基本概念 …… 135
8.1.1 图的定义 …… 135
8.1.2 图的基本术语 …… 136
8.2 图的存储结构 …… 139
8.2.1 邻接矩阵表示法 …… 139
8.2.2 邻接表表示法 …… 142
8.3 图的遍历 …… 144
8.3.1 图的遍历的概念 …… 144

8.3.2 深度优先搜索 …… 145
8.3.3 广度优先搜索 …… 146
8.3.4 图的遍历算法的应用 …… 148
8.4 最小生成树 …… 150
8.4.1 最小生成树的概念及其性质 …… 150
8.4.2 Prim 算法 …… 151
8.4.3 Kruskal 算法 …… 154
8.5 最短路径 …… 156
8.5.1 最短路径的概念 …… 156
8.5.2 单源最短路径 …… 156
8.5.3 每对顶点之间的最短路径 …… 160
8.6 AOV 网与拓扑排序 …… 163
8.6.1 有向无环图与 AOV 网的概念 …… 163
8.6.2 拓扑排序 …… 164
*8.7 AOE 网与关键路径 …… 166
8.7.1 AOE 网的概念 …… 166
8.7.2 关键路径 …… 167
本章小结 …… 168
习题 8 …… 169
上机实验题 8 …… 170
第 9 章 查找 …… 171
9.1 查找的基本概念 …… 171
9.2 顺序表的查找 …… 172
9.2.1 顺序查找 …… 172
9.2.2 折半查找 …… 173
9.2.3 分块查找 …… 176
9.3 树表的查找 …… 177
9.3.1 二叉排序树 …… 177
9.3.2 平衡二叉树 …… 183
9.3.3 B 树 …… 185
9.3.4 B+树 …… 190
9.4 Hash 查找 …… 191
9.4.1 Hash 查找的基本概念 …… 191
9.4.2 Hash 表的构造 …… 192
9.4.3 Hash 查找算法及分析 …… 196
本章小结 …… 197
习题 9 …… 197
上机实验题 9 …… 199
第 10 章 排序 …… 200
10.1 排序的基本概念 …… 200

10.2 冒泡排序 …… 201
10.3 选择排序 …… 202
10.4 插入排序 …… 203
10.4.1 直接插入排序 …… 203
10.4.2 折半插入排序 …… 205
10.5 希尔排序 …… 205
10.6 快速排序 …… 207
10.7 堆排序 …… 209
10.8 归并排序 …… 214
10.8.1 二路归并排序的非递归实现 …… 214
10.8.2 二路归并排序的递归实现 …… 216
10.9 基数排序 …… 217
10.9.1 多关键字排序 …… 217
10.9.2 链式基数排序 …… 218
本章小结 …… 219
习题 10 …… 221
上机实验题 10 …… 222
附录 A 数据结构试题 …… 223
附录 B 数据结构课程设计题 …… 230
参考文献 …… 231

第1章 绪 论

“数据结构”是计算机科学与技术、软件工程、信息安全、信息管理与信息系统、信息与计算科学等专业的一门十分重要的专业核心基础课程，主要学习计算机中数据的组织方式、存储结构和处理方法。数据结构课程的学习将为计算机及相关专业的后续课程（如操作系统、编译原理、数据库原理、软件工程等）的学习打下基础。实际上，要编写一个“好”的程序，无非是要选择一个合理的数据结构和好的算法，而“好”算法的选择很大程度上取决于描述实际问题所采用的数据结构，因此要编写出“好”的程序，仅仅学习程序设计语言是不够的，必须很好地掌握数据结构的基本知识和基本技能。本章将概要地介绍数据结构课程的研究内容、基本概念和基本思想。

1.1 数据结构课程的研究内容

数据结构起源于程序设计，随着计算机科学技术的发展，计算机应用领域不再局限于科学计算，而更多地应用于信息处理、智能控制、办公自动化等领域。计算机处理的对象由数值发展到字符串、表格、图形、图像、声音等数据，而且处理的数据量也越来越大。在程序设计中面对这样的数据，我们应如何来组织和处理它们呢？这就是“数据结构”课程需要研究的问题。

计算机解决问题时，一般要经过几个步骤：首先，要将实际问题抽象出数学模型，然后针对数学模型设计出求解算法，最后编写程序上机调试，直到求出最终结果。数值计算问题的数学模型一般可由数学方程或数学公式来描述。然而，对于非数值计算问题，例如图书资料的检索、人—机博弈、课程表编排、最短路径求解等问题，它们的数学模型无法用数学方程或数学公式来描述，而是要用线性表、树、图等数据结构来描述，并且要对这些模型设计相应算法来求解。数据结构就是研究计算机非数值计算问题中的数据对象以及它们之间的关系和操作算法的学科，具体主要包含三个方面的内容：①数据的逻辑结构；②数据的存储结构；③数据的操作算法。

数据结构作为一门独立的课程是 1968 年率先在美国开设，在这之前，它的某些内容在其他课程中涉及。1968 年，斯坦福大学的 E.Knuth 教授开创了数据结构的最初体系，他所著的《计算机程序设计艺术》第一卷《基本算法》是一本较系统地阐述数据的逻辑结构、存储结构及其操作算法的著作。后来各种版本的数据结构著作相继出现，例如有 Pascal 版、C 版、C++版、Java 版。我国于 20 世纪 80 年代初开设“数据结构”课程，该课程不仅仅是计算机专业教学计划中的核心课程之一，而且也是其他信息类专业的必修课。

数据结构的内容随着程序设计技术的发展而发展，它经历了结构化阶段和面向对象阶段。20 世纪 60～80 年代，计算机开始广泛应用于非数值计算领域，数据组织成为程序设计的重要问题。人们认识到程序设计规范的重要性，提出了结构化程序设计的思想。数据结构概念的引入，对程序设计的规范化起到了重要作用。图灵奖获得者、瑞士计算机科学家 N.Wirth 教授曾提出“程序=数据结构+算法”，由此可以看出，数据结构和算法是构成程序的两个重

要组成部分。

20 世纪 90 年代以来，面向对象技术成为最流行的程序设计技术。在面向对象技术中，现实世界的实体被看做是一个对象，对象是由属性和方法构成的，属性描述实体的状态和特征，方法用以改变实体的状态或行为。一组具有相同属性和方法的对象集合称为类，每个具体的对象称为类的一个实例。例如，“学生”是一个类，“张丽”、“李明”等对象都是“学生”类的实例。

数据结构主要强调两个方面的内容：①数据之间的关系，即数据之间的逻辑结构和存储结构；②针对这些关系的基本操作。这两个方面实际上蕴涵着面向对象的思想：类描述实体的属性和行为，而数据结构描述数据之间的关系及其基本操作。类与数据结构之间的对应关系，如图 1-1 所示。

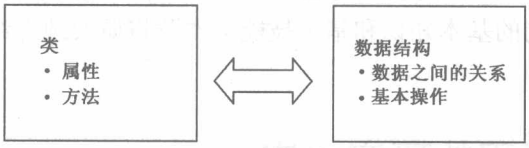


图 1-1　类与数据结构之间的对应关系

值得一提的是，数据结构的发展并未终结。一方面，数据结构将继续随着程序设计技术的发展而发展；另一方面，面向专门领域的数据结构得到研究和重视。例如，研究人员提出了一些空间数据结构。

1.2　基本概念及术语

本节，将介绍与数据结构相关的基本概念及名词术语。

1. 数据

数据（Data）是信息的载体，是指所有能输入到计算机中并能被计算机程序识别和处理的符号集合。数据可以分为两大类：一类是整数、实数等数值数据；另一类是图形、图像、声音、文字等非数值数据。

2. 数据元素

数据元素（Data Element）是组成数据的基本单位，在计算机程序中通常作为一个整体进行考虑和处理。构成数据元素的不可分割的最小单位称为数据项。例如，对于学生档案登记表，每个学生的档案就是一个数据元素，而档案中的学号、姓名、出生日期等是数据项。数据元素是讨论数据结构时涉及的最小数据单位，其中的数据项一般不予考虑。

数据元素具有广泛的含义，一般来说，能独立、完整地描述问题世界的一切实体都是数据元素。例如，对弈中的棋盘格局、教学计划中的某个课程、一年中的四个季节，甚至一次学术报告、一场足球比赛都是数据元素。数据元素又称为元素、结点、顶点或记录。

3. 数据对象

数据对象（Data Object）是具有相同性质的数据元素的集合，是数据的子集。在实际应用中处理的数据元素通常具有相同性质。例如，学生档案登记表中每个数据元素具有相同数目和类型的数据项，所有数据元素（学生的档案）的集合就构成了一个数据对象。又如，字

母数据集合 M={'A', 'B',…, 'Z'}也是一个数据对象。

4. 数据结构

数据结构（Data Structure）是指数据元素及其相互关系的集合，这种相互关系，也就是数据的组织形式，它分为数据的逻辑结构和数据的存储结构。

数据的逻辑结构（Logical Structure）是指数据元素之间的逻辑关系。所谓逻辑关系是指数据元素之间的关联方式或邻接关系。数据的逻辑结构分为 4 类：

（1）集合。数据元素之间的关系是“属于同一个集合”，除此之外，没有任何关系。

（2）线性结构。数据元素之间存在着一对一的线性关系。

（3）树结构。数据元素之间存在着一对多的层次关系。

（4）图结构。数据元素之间存在着多对多的任意关系。

其中，树结构和图结构也称为非线性结构。

数据的逻辑结构常用逻辑结构图来描述，其描述方法是：将每一个数据元素看做一个结点，用圆圈表示，元素之间的逻辑关系用结点之间的连线表示，如果强调关系的方向性，则用带箭头的连线表示关系。图 1-2 描述了 4 种基本的数据逻辑结构图。

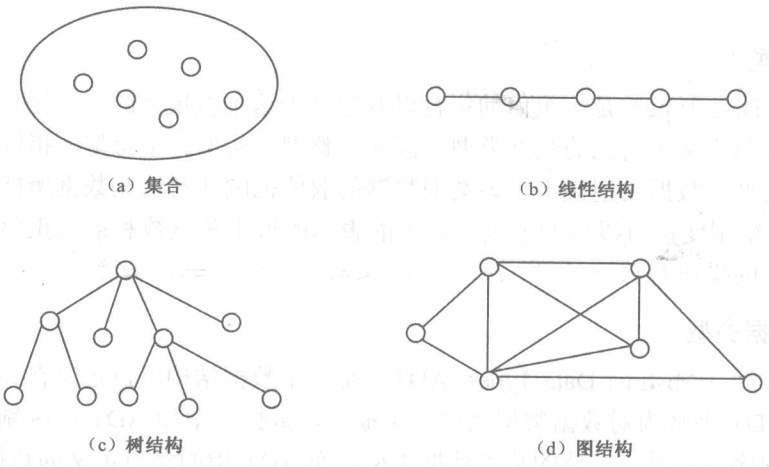


图 1-2　4 种基本的数据逻辑结构图

为了更确切地描述一种数据结构，通常采用如下二元组形式化定义数据结构：

$$\text{Data\_Structure} = (D, R)$$

其中，D 是数据元素的有限集合，S 是 D 上关系的有限集合。

例如，有一种数据结构 $T=(D, R)$，其中：

$D=\{a, b, c, d, e, f, g, h, i, j\}$

$R=\{(a, b), (a, c), (a, d), (b, e), (c, f), (c, g), (d, h), (d, i), (d, j)\}$

显然，数据结构 T 是一个树形结构。

数据的存储结构（Storage Structure）又称为物理结构，是数据及其逻辑结构在计算机中的表示。换言之，存储结构除了存储数据元素之外，必须隐式或显式地存储数据元素之间的逻辑关系。通常有两种存储结构：顺序存储结构和链式存储结构。

顺序存储结构的基本思想是，用一组连续的存储单元存储数据元素，数据元素之间的逻辑关系是由元素的存储位置来表示的。例如，线性表(a, b, c)的顺序存储示意图如图 1-3 所示。

链式存储结构的基本思想是，用一组任意的存储单元存储数据元素，数据元素之间的逻辑关系是用指针来表示的。例如，线性表(a, b, c)的链式存储示意图如图 1-4 所示。

数据的逻辑结构和存储结构是密切相关的两个方面，一般来说，一种数据的逻辑结构可以用多种存储结构来存储，而采用不同的存储结构，其数据处理的效率往往是不同的。

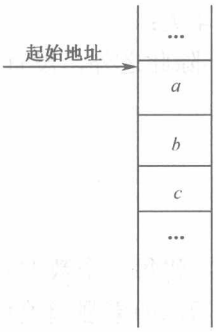


图 1-3　线性表的顺序存储示意图

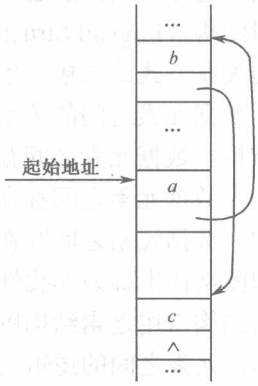


图 1-4　线性表的链式存储示意图

5．数据类型

数据类型（Data Type）是一组值的集合以及定义于这个值集上的一组操作的总称。每一种程序设计语言都定义了自己的数据类型。例如，整型、实型、字符型、指针、数组、结构体，类等数据类型。数据类型规定了该类型数据的取值范围和对这些数据所能采取的操作。例如，C++中的整型变量可以取的值是机器所能表示的最小负整数和最大正整数之间的任何一个整数，允许的操作有+、－、*、/、%、<、<=、>、>=、==、!=等。

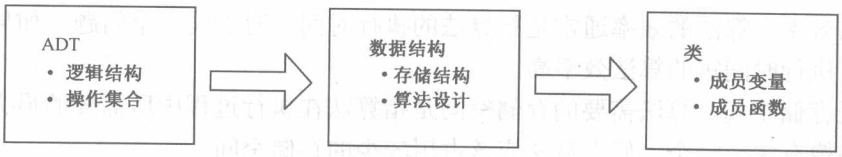
6．抽象数据类型

抽象数据类型（Abstract Data Type，ADT）是一个数据结构以及定义在该结构上的一组操作的总称。ADT 理解为对数据类型的进一步抽象，数据类型和 ADT 的区别在于：数据类型指的是高级程序设计语言支持的基本数据类型，而 ADT 指的是自定义的数据类型。例如，本课程将要学习的表、栈、队列、树、图等结构就是一些不同的 ADT。

ADT 包括定义和实现两个方面，其中定义是独立于实现的，定义仅给出一个 ADT 的逻辑特征，不必考虑如何在计算机中实现。ADT 的特征是使用与实现分离，实现封装和信息隐藏。例如，整数的数学概念和施加到整数的运算构成一个 ADT，C++的变量类型 int 是对这个 ADT 的物理实现。各种程序设计语言都拥有整数类型，尽管它们在不同处理器上实现的方法不同，但由于其 ADT 相同，在用户看来都是相同的。

在设计 ADT 时，把 ADT 的定义和实现分开来。定义部分只包含数据逻辑结构的定义和所允许的操作集合，一方面，使用者依据这些定义来使用 ADT，即通过操作集合对该 ADT 进行操作；另一方面，ADT 的实现者依据这些定义来完成该 ADT 的各种操作的具体实现。图 1-5 给出了 ADT 的不同视图。

C++中的类体现了抽象数据类型的思想。在 ADT 中定义的每个操作由类中的成员函数来实现，数据以及数据之间的逻辑关系由类中的成员变量来实现。



（a）使用视图—ADT的定义　　（b）设计视图—ADT的设计　　（c）实现视图—ADT的实现

图 1-5　ADT 的不同视图

1.3　算法与算法分析

1.3.1　算法

1. 什么是算法

算法（Algorithm）是计算机求解特定问题的方法和步骤，是指令的有限序列。通常一个问题可以有多种算法，一个给定算法解决一个特定的问题。

算法具有下列 5 个重要特性：

（1）输入。一个算法有零个或多个输入（即算法可以无输入），这些输入通常取自于某个特定的对象集合。

（2）输出。一个算法有一个或多个输出（即算法必须要有输出），通常输出与输入之间有着某种特定的关系。

（3）有穷性。一个算法必须（对任何合法的输入）在执行有穷步之后结束，且每一步都在有穷时间内完成。

（4）确定性。算法中的每一条指令必须有确切的含义，不存在二义性，并且在任何条件下，对于相同的输入只能得到相同的输出。

（5）可行性。算法描述的操作可以通过已经实现的基本操作执行有限次来实现。

算法与程序不同。程序（Program）是对一个算法使用某种程序设计语言的具体实现，原则上，任一算法可以用任何一种程序设计语言实现。算法的有穷性意味着不是所有的计算机程序都是算法。例如，操作系统是一个在无限循环中执行的程序，而不是一个算法，然而我们可以把操作系统的各个任务看成是一个单独的问题，每一个问题由操作系统中的一个子程序通过特定的算法来实现，得到输出结果后便终止。

2. 算法的评价

数据结构与算法之间存在着本质联系，本课程学习的目的就是要在某一种数据结构基础上学习算法的设计方法，不但要设计正确的算法，而且要设计“好”的算法。那么什么样的算法是“好”的算法呢？通常一个“好”算法具有下列 5 个基本特性：

（1）正确性。算法能满足具体问题的需求，即对任何合法的输入，算法都会得出正确的结果。

（2）健壮性（鲁棒性）。指算法对非法输入的抵抗能力，即对于错误的输入，算法应能识别并做出处理，而不是产生错误动作或陷入瘫痪。

（3）可读性。一个好的算法应该便于人们理解和相互交流。可读性好的算法有助于人们对算法的理解，反之，难懂的算法易于隐藏错误且难于调试和修改。

（4）高效率。算法的效率通常是指算法的执行时间。对于同一个问题，如果有多个算法可以解决，执行时间短的算法效率高。

（5）低存储空间。算法需要的存储空间是指算法在执行过程中所需要的最大存储空间，它与问题规模有关。一个“好”算法应该占用较少的存储空间。

3. 算法的描述方法

设计了一个算法之后，必须清楚准确地将所设计的求解步骤表达出来，即描述算法。算法描述方法通常有自然语言、流程图、伪代码和程序设计语言等方法。下面以欧几里得算法（用辗转相除法求两个自然数 m 和 n 的最大公约数，并假设 $m \geq n$）为例进行介绍。

（1）自然语言

用自然语言描述算法，最大的优点是容易理解，缺点是容易出现二义性，并且算法通常都很冗长。欧几里得算法用自然语言描述如下：

① 输入 m 和 n；

② 求 m 除以 n 的余数 r；

③ 若 r 等于 0，则 n 为最大公约数，算法结束；否则执行第④步；

④ 将 n 的值放在 m 中，将 r 的值放在 n 中；

⑤ 重新执行第②步。

（2）流程图

用流程图描述算法，优点是直观易懂，缺点是严密性不如程序设计语言，灵活性不如自然语言。欧几里得算法用流程图描述如图 1-6 所示。

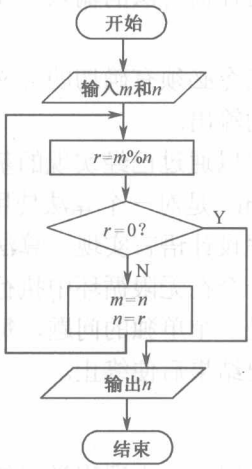


图 1-6　用流程图描述算法

在计算机应用早期，使用流程图描述算法占有统治地位，但实践证明，除了一些非常简单的算法以外，这种描述方法使用起来并不方便。

（3）伪代码

伪代码是介于自然语言和程序设计语言之间的方法，计算机科学家对伪代码的书写形式没有做出严格的规定，它采用某种程序设计语言的基本语法，操作指令可以结合自然语言来设计。至于算法中自然语言的成分有多少，取决于算法的抽象级别。抽象级别高的伪代码自然语言多一些，抽象级别低的伪代码程序设计语言的语句多一些。只要具有一些程序设计语

言基础的人都能阅读伪代码。欧几里得算法用C++伪代码描述如下：

```
int CommmonFactor(int m, int n)
{   r=m%n;
    while(r!=0)
    { m=n;
      n=r;
      r=m%n;
    }
    return   n;
}
```

伪代码不是一种实际的编程语言，但在表达能力上类似于编程语言，同时极小化了描述算法的不必要的技术细节，是比较合适的描述算法的方法，被称为“算法语言”。

本教材采用基于C++语言的伪代码来描述算法，使算法的描述简明清晰，既不拘泥于C++语言的实现细节，又容易转换为C++程序。

（4）程序设计语言

用程度设计语言描述的算法能由计算机直接执行，但缺点是抽象性差，使算法设计者拘泥于描述算法的具体细节，忽略了“好”算法和正确逻辑的重要性，此外，还要求算法设计者掌握程序设计语言及其编程技巧。欧几里得算法用C++语言书写的程序如下：

```
#include <iostream.h>
int CommonFactor(int m, int n)
{   int r = m % n;
    while (r! = 0)
    {   m=n;
        n=r;
        r=m % n;
    }
    return   n;
}
void main( )
{   cout<<CommonFactor(63, 54)<<endl;
}
```

1.3.2 算法分析

一种数据结构的优劣是由实现其各种操作的算法决定的，对数据结构的分析实质上就是对实现各种操作的算法进行分析。除了要验证算法是否正确解决问题外，还需要对算法的效率做出评价。对于一个实际问题的解决，可以提出若干算法，那么如何从这些可行的算法中找出最有效的算法呢？或者有了一个解决实际问题的算法，如何来分析它的性能呢？这些问题需要通过算法分析来确定。通常算法分析主要分析算法的时间代价和空间代价这两个主要