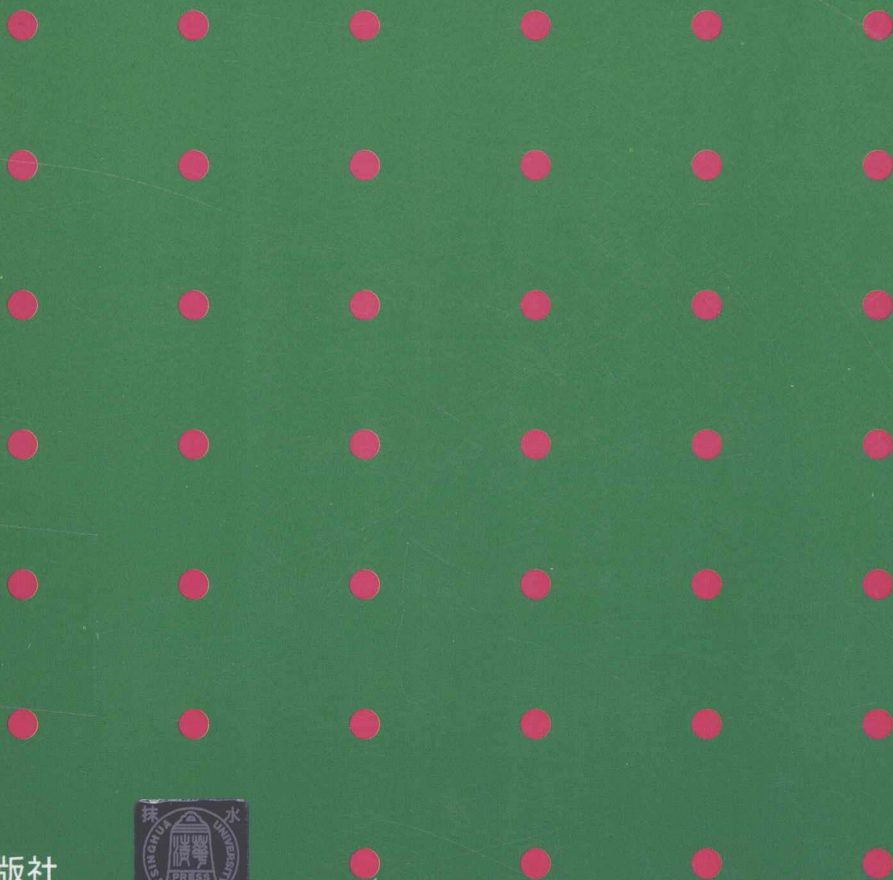


普通高校本科计算机专业特色教材精选·算法与程序设计

# 数据结构——从概念到C实现

王红梅 皮德常 编著



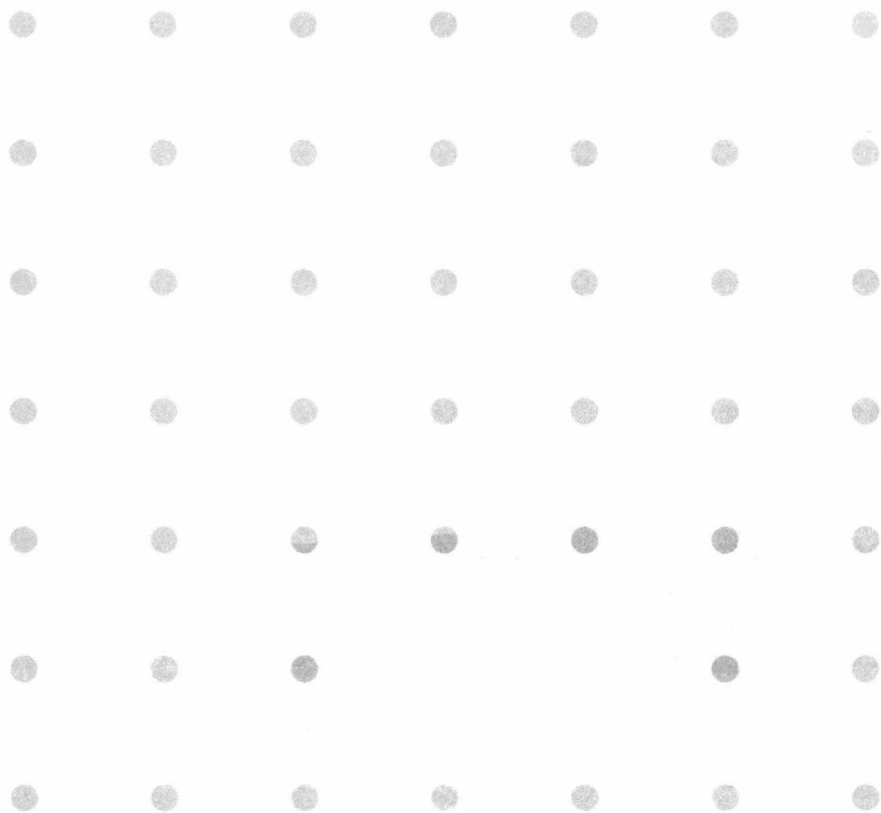
清华大学出版社



普通高校本科计算机专业特色教材精选·算法与程序设计

# 数据结构——从概念到C实现

王红梅 皮德常 编著



清华大学出版社  
北京

## 内 容 简 介

数据结构是计算机及相关专业的核心课程,也是计算机及相关专业硕士研究生入学考试的必考科目,而且是理工专业的热门公选课程。本书介绍了数据结构、算法以及抽象数据类型的概念;介绍了线性表、栈和队列、字符串和 multidimensional array、树和二叉树、图等常用数据结构;讨论了基本的查找和排序技术。

本书合理规划教学内容,梳理知识单元及其拓扑结构,兼顾概念层和实现层,既强调了数据结构的基本概念和原理方法,又注重了数据结构的程序实现和实际运用,在提炼基础知识的同时,进行了适当的扩展和提高。

本书内容丰富,层次清晰,深入浅出,结合实例,可作为计算机及相关专业数据结构课程的教材,也可供从事计算机软件开发和应用的工程技术人员参考和阅读。

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

版权所有,侵权必究。侵权举报电话:010-62782989 13701121933

### 图书在版编目(CIP)数据

数据结构:从概念到C实现/王红梅,皮德常编著.—北京:清华大学出版社,2017

(普通高校本科计算机专业特色教材精选·算法与程序设计)

ISBN 978-7-302-45149-5

I. ①数… II. ①王… ②皮… III. ①数据结构—高等学校—教材 IV. ①TP311.12

中国版本图书馆CIP数据核字(2016)第232939号

责任编辑:袁勤勇

封面设计:常雪影

责任校对:时翠兰

责任印制:刘海龙

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](mailto:c-service@tup.tsinghua.edu.cn)

质量反馈:010-62772015, [zhiliang@tup.tsinghua.edu.cn](mailto:zhiliang@tup.tsinghua.edu.cn)

课件下载: <http://www.tup.com.cn>, 010-62795954

印 装 者:北京泽宇印刷有限公司

经 销:全国新华书店

开 本:185mm×260mm

印 张:18.75

字 数:426千字

版 次:2017年2月第1版

印 次:2017年2月第1次印刷

印 数:1~2000

定 价:39.00元

产品编号:069975-01

# 前言

## PREFACE

数据结构是计算机及相关专业的核心课程，也是计算机及相关专业硕士研究生入学考试的必考科目，而且是理工专业的热门公选课程。作为程序设计的重要补充和延伸，数据结构所讨论的知识内容、蕴含的技术方法、体现的思维方式，无论进一步学习计算机专业的其他课程，还是从事计算机领域的各项工作，都有着不可替代的作用。

数据结构课程的知识丰富，内容抽象，隐藏在各知识单元的概念和方法较多，贯穿于各知识单元的链表和递归更是加重了学习难度。本书的编写者长期从事数据结构的研究和教学，深切理解学生在学习数据结构过程中遇到的问题和困惑，深入探究掌握数据结构的有效途径和方法，深刻思考数据结构对培养程序设计和计算思维的地位和作用，深度把握课程的教学目标和重点难点，本书在教学内容和教学设计等方面进行了如下处理。

1. **合理规划教学内容。** 紧扣《高等学校计算机专业核心课程教学实施方案》和《计算机学科硕士研究生入学考试大纲》，涵盖教学方案及考研大纲要求的全部知识点。

2. **遵循认知规律，理清教学主线。** 根据学生的认知规律和课程的知识结构，按照从已知到未知的思维进程逐步推进教学内容，梳理和规划了各知识单元及其拓扑结构，设计了清晰的教学主线。知识单元及其拓扑结构如图 1 所示。

3. **提炼基础知识，适当扩展提高。** 考虑到不同学校教学要求的差异以及不同学生学习需求的差别，一方面本着“够用、实用”的原则，抓牢核心概念，提炼基础性知识，贯彻数据结构课程的基本教学要求；另一方面对某些知识点进行了适当的扩充和提高（图 1 中打星号部分），这部分内容可用于选讲，也可用于学生自学或课外阅读（教学建议：目录中打一个星号可用于选讲，打两个星号可用于课外阅读，其他是必讲的基础知识）。

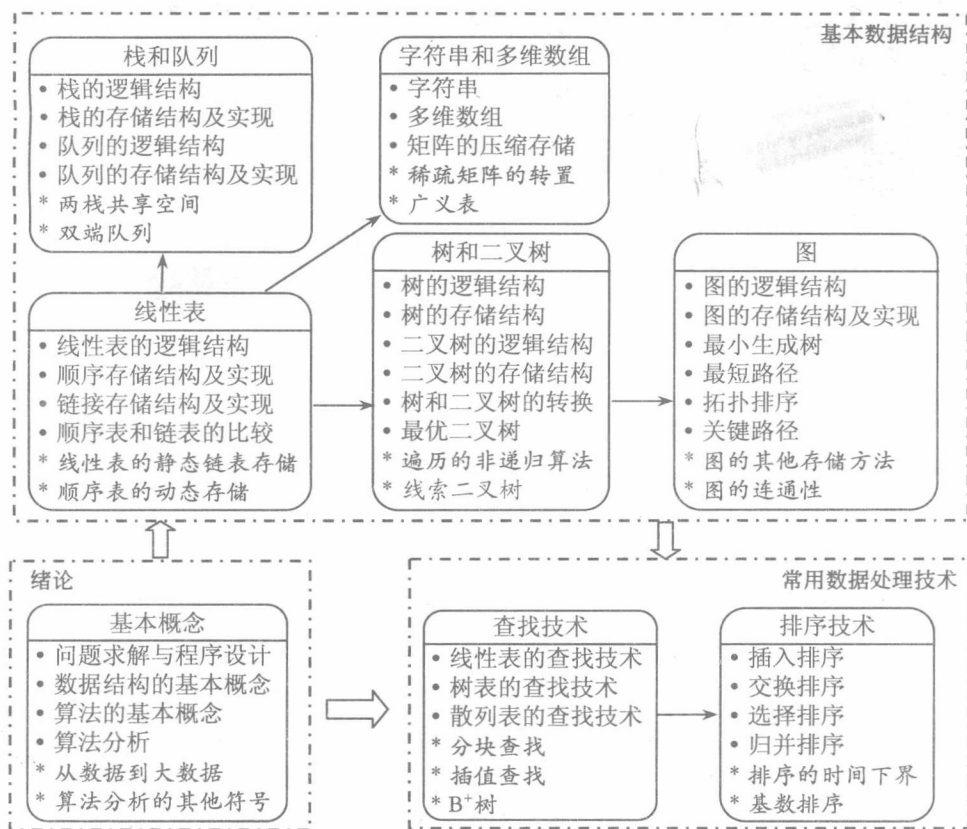


图 1 数据结构课程的知识单元及拓扑结构

4. 兼顾概念层和实现层。从抽象数据类型的角度将数据结构的实现过程分为抽象层、设计层和实现层，其中抽象层定义数据结构和基本操作集合，设计层是数据结构的存储表示和算法设计，实现层用 C 语言实现数据结构，既强调了数据结构的基本概念和原理方法，又注重了数据结构的程序实现和实际运用。

5. 展现求解过程，培养计算思维。通过讲思路讲过程讲方法，按照“问题→想法→算法→程序”的模式进行问题求解，采用“阐述基本思想→伪代码描述算法→C 语言实现算法”的模式进行算法设计，这个过程正是计算思维的运用过程。每章通过两个应用实例展示问题求解过程以及算法设计过程。

6. 明确重点，化解难点。每一章开篇即给出该章的重点难点，以及各知识点的教学要求，在具体阐述时还有针对性的处理方法。针对数据结构内容抽象的特点，全书设计了大量插图，将抽象内容进行了具体化处理，降低了理解问题的复杂性。

总之，本书在概念的描述、实例的选择、知识的前后衔接、内容的组织结构，以及教学内容的理解、教学目标的实现、教学意图的融入、教学方法的运用等方面进行了系统思考和统筹设计，力图通过本书为读者构建多层次的知识体系。在问题求解层面，以数据表示和数据处理为主线，给出“问题→想法→算法→程序”的思维模式；在算法设计层面，通过伪代码描述算法，强调计算思维的培养；在算法分析层面，理解什么是

“好”算法，给出算法分析的基本方法；在存储结构层面，通过存储示意图理解数据表示，再给出存储结构定义；在程序实现层面，给出所有数据结构的 C 程序实现以及使用范例；在数据结构和算法的运用层面，通过应用实例理解如何为求解问题设计适当的数据结构，如何基于数据结构设计算法，从而将数据结构、算法和程序设计有机地融合在一起。

参加本书编写的还有董亚则、王涛、党源源、肖巍、刘冰、王有敬等老师，2015 级硕士研究生唐东凯和李芬田同学参与了本书的代码调试工作，本书所有代码均在 Dev C++ 5.11、Visual Studio 2013、Microsoft Visual C++ 6.0 编程环境下调试通过，读者可向出版社或作者索要程序源码。

由于作者的知识 and 写作水平有限，书稿虽再三斟酌几经修改，仍难免有缺点和错误，欢迎专家和读者批评指正。作者的电子邮箱是：wanghm@ccut.edu.cn。

作者

2016 年 2 月

# 目 录

## CONTENTS

<b>第 1 章 绪论</b> .....	1
1.1 问题求解与程序设计 .....	2
1.1.1 程序设计的一般过程 .....	2
1.1.2 数据结构在程序设计中的作用 .....	4
1.1.3 算法在程序设计中的作用 .....	6
1.1.4 本书讨论的主要内容 .....	7
1.2 数据结构的基本概念 .....	8
1.2.1 数据结构 .....	8
1.2.2 抽象数据类型 .....	11
1.3 算法的基本概念 .....	12
1.3.1 算法及算法的特性 .....	12
1.3.2 算法的描述方法 .....	14
1.4 算法分析 .....	15
1.4.1 算法的时间复杂度 .....	16
1.4.2 算法的空间复杂度 .....	17
1.4.3 算法分析举例 .....	18
1.5 扩展与提高 .....	20
**1.5.1 从数据到大数据 .....	20
**1.5.2 算法分析的其他渐进符号 .....	22
习题 1 .....	23
<b>第 2 章 线性表</b> .....	25
2.1 引言 .....	26
2.2 线性表的逻辑结构 .....	27
2.2.1 线性表的定义 .....	27
2.2.2 线性表的抽象数据类型定义 .....	27

2.3	线性表的顺序存储结构及实现	29
2.3.1	顺序表的存储结构定义	29
2.3.2	顺序表的实现	30
2.3.3	顺序表的使用	34
2.4	线性表的链接存储结构及实现	35
2.4.1	单链表的存储结构定义	35
2.4.2	单链表的实现	37
2.4.3	单链表的使用	44
2.4.4	双链表	45
2.4.5	循环链表	47
2.5	顺序表和链表的比较	48
2.6	扩展与提高	48
*2.6.1	线性表的静态链表存储	48
*2.6.2	顺序表的动态分配方式	51
2.7	应用实例	52
2.7.1	约瑟夫环问题	52
2.7.2	一元多项式求和	55
习题 2		59
<b>第 3 章</b>	<b>栈和队列</b>	<b>63</b>
3.1	引言	64
3.2	栈	65
3.2.1	栈的逻辑结构	65
3.2.2	栈的顺序存储结构及实现	66
3.2.3	栈的链接存储结构及实现	68
3.2.4	顺序栈和链栈的比较	71
3.3	队列	72
3.3.1	队列的逻辑结构	72
3.3.2	队列的顺序存储结构及实现	73
3.3.3	队列的链接存储结构及实现	77
3.3.4	循环队列和链队列的比较	80
3.4	扩展与提高	81
*3.4.1	两栈共享空间	81
*3.4.2	双端队列	82
3.5	应用举例	83
3.5.1	括号匹配问题	83
3.5.2	表达式求值	85
习题 3		88



<b>第4章 字符串和多维数组</b> .....	91
4.1 引言 .....	92
4.2 字符串 .....	92
4.2.1 字符串的逻辑结构 .....	92
4.2.2 字符串的存储结构 .....	94
4.2.3 模式匹配 .....	94
4.3 多维数组 .....	98
4.3.1 数组的逻辑结构 .....	98
4.3.2 数组的存储结构与寻址 .....	99
4.4 矩阵的压缩存储 .....	100
4.4.1 特殊矩阵的压缩存储 .....	100
4.4.2 稀疏矩阵的压缩存储 .....	102
4.5 扩展与提高 .....	105
**4.5.1 稀疏矩阵的转置运算 .....	105
**4.5.2 广义表 .....	107
4.6 应用实例 .....	111
4.6.1 发纸牌 .....	111
4.6.2 八皇后问题 .....	112
习题4 .....	115
<b>第5章 树和二叉树</b> .....	119
5.1 引言 .....	120
5.2 树的逻辑结构 .....	121
5.2.1 树的定义和基本术语 .....	121
5.2.2 树的抽象数据类型定义 .....	123
5.2.3 树的遍历操作 .....	123
5.3 树的存储结构 .....	124
5.3.1 双亲表示法 .....	124
5.3.2 孩子表示法 .....	125
5.3.3 孩子兄弟表示法 .....	126
5.4 二叉树的逻辑结构 .....	127
5.4.1 二叉树的定义 .....	127
5.4.2 二叉树的基本性质 .....	129
5.4.3 二叉树的抽象数据类型定义 .....	130
5.4.4 二叉树的遍历操作 .....	131
5.5 二叉树的存储结构 .....	133
5.5.1 顺序存储结构 .....	133
5.5.2 二叉链表 .....	134

5.5.3 三叉链表	138
5.6 森林	138
5.6.1 森林的逻辑结构	138
5.6.2 树、森林与二叉树的转换	139
5.7 最优二叉树	141
5.7.1 哈夫曼算法	141
5.7.2 哈夫曼编码	143
5.8 扩展与提高	145
*5.8.1 二叉树遍历的非递归算法	145
*5.8.2 线索二叉树	148
5.9 应用实例	151
5.9.1 堆与优先队列	151
5.9.2 并查集	154
习题 5	155
<b>第 6 章 图</b>	<b>159</b>
6.1 引言	160
6.2 图的逻辑结构	161
6.2.1 图的定义和基本术语	161
6.2.2 图的抽象数据类型定义	163
6.2.3 图的遍历操作	164
6.3 图的存储结构及实现	167
6.3.1 邻接矩阵	167
6.3.2 邻接表	170
6.3.3 邻接矩阵和邻接表的比较	174
6.4 最小生成树	175
6.4.1 Prim 算法	176
6.4.2 Kruskal 算法	178
6.5 最短路径	182
6.5.1 Dijkstra 算法	183
6.5.2 Floyd 算法	185
6.6 有向无环图及其应用	187
6.6.1 AOV 网与拓扑排序	187
6.6.2 AOE 网与关键路径	190
6.7 扩展与提高	193
*6.7.1 图的其他存储方法	193
*6.7.2 图的连通性	194
6.8 应用实例	196

6.8.1 七巧板涂色问题	196
6.8.2 医院选址问题	198
习题 6	200
<b>第 7 章 查找技术</b>	<b>205</b>
7.1 概述	206
7.1.1 查找的基本概念	206
7.1.2 查找算法的性能	207
7.2 线性表的查找技术	207
7.2.1 顺序查找	207
7.2.2 折半查找	208
7.3 树表的查找技术	211
7.3.1 二叉排序树	211
7.3.2 平衡二叉树	217
7.3.3 B 树	220
7.4 散列表的查找技术	225
7.4.1 散列查找的基本思想	225
7.4.2 散列函数的设计	226
7.4.3 处理冲突的方法	227
7.4.4 散列查找的性能分析	231
7.4.5 开散列表与闭散列表的比较	232
7.5 各种查找方法的比较	232
7.6 扩展与提高	233
**7.6.1 顺序查找的改进——分块查找	233
**7.6.2 折半查找的改进——插值查找	234
**7.6.3 B 树的改进——B <sup>+</sup> 树	235
习题 7	236
<b>第 8 章 排序技术</b>	<b>239</b>
8.1 概述	240
8.1.1 排序的基本概念	240
8.1.2 排序算法的性能	241
8.2 插入排序	241
8.2.1 直接插入排序	241
8.2.2 希尔排序	243
8.3 交换排序	245
8.3.1 起泡排序	245
8.3.2 快速排序	247

8.4 选择排序 .....	250
8.4.1 简单选择排序 .....	250
8.4.2 堆排序 .....	252
8.5 归并排序 .....	256
8.5.1 二路归并排序的递归实现 .....	256
8.5.2 二路归并排序的非递归实现 .....	257
8.6 各种排序方法的比较 .....	259
8.7 扩展与提高 .....	261
**8.7.1 排序问题的时间下界 .....	261
*8.7.2 基数排序 .....	262
习题 8 .....	264
<b>附录 A 预备知识</b> .....	269
A.1 数学术语 .....	269
A.2 级数求和 .....	269
A.3 集合 .....	270
A.4 关系 .....	271
<b>附录 B C 语言基本语法</b> .....	273
B.1 程序结构 .....	273
B.2 数据的基本表现形式——常量和变量 .....	274
B.3 数据类型 .....	275
B.4 控制语句 .....	277
B.5 函数 .....	278
B.6 动态存储分配 .....	281
<b>附录 C 词汇索引</b> .....	283
<b>参考文献</b> .....	287

# 第 1 章

## 绪 论

CHAPTER

教学重点	数据结构的基本概念;数据的逻辑结构、存储结构以及二者之间的关系;算法及算法的特性;大O记号				
教学难点	抽象数据类型;算法的时间复杂度分析				
教学内容和教学目标	知 识 点	教 学 要 求			
		了解	理解	掌握	熟练掌握
	程序设计的一般过程	√			
	数据结构在程序设计中的作用		√		
	算法在程序设计中的作用		√		
	数据结构的基本概念				√
	抽象数据类型		√		
	算法及算法的特性				√
	算法的描述方法			√	
	算法的时间复杂度			√	
算法的空间复杂度		√			
教学提示	<p>对本章的教学要抓住两条主线,一条主线是数据结构,包括数据结构的研究对象及相关概念,另一条主线是算法,包括算法的概念、描述方法以及时间复杂度的分析。从程序设计的一般过程入手,说明计算思维(模型化、形式化、抽象思维和逻辑思维)在程序设计过程中的运用,理解“数据结构+算法=程序”,注意强调数据结构和算法与程序之间的关系。</p> <p>对于数据结构部分,强调数据结构的核心理念是数据元素,注意通过具体实例引申数据元素之间的关系,并抓住两个方面:逻辑结构和存储结构,强调二者之间的关系。对于算法部分,说明算法是对问题求解方法的抽象描述,以算法的概念和特性为重点,并在以后的教学中注意应用算法的特性,不要孤立地讲授概念。对于算法的时间性能,强调三个要点:基本语句、执行次数、数量级。</p> <p>最后,强调数据结构和算法分析都是针对大数据量,即大数据量的组织、大数据量的处理效率。</p>				

## 1.1 问题求解与程序设计

计算机科学致力于研究如何用计算机求解人类生产生活中的各种实际问题,只有最终在计算机上能够运行良好的程序才能解决特定的实际问题,因此,程序设计的过程就是利用计算机求解问题的过程。

### 1.1.1 程序设计的一般过程

用计算机求解任何问题都离不开程序,但是计算机不能分析问题并产生问题的解决方案,必须由人(即程序设计者)分析问题,确定问题的解决方案,采用计算机能够理解的指令描述这个问题的求解步骤(即编写程序),然后让计算机执行程序最终获得问题的解。程序设计的一般过程如图 1-1 所示。

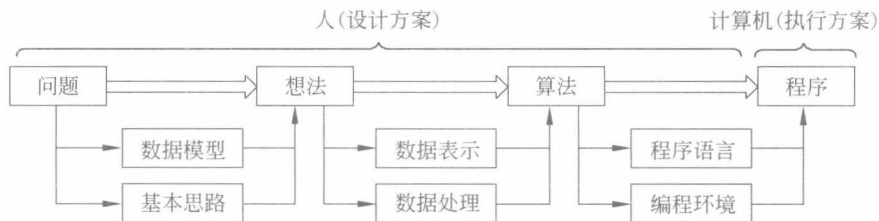


图 1-1 程序设计的一般过程

由问题到想法需要分析待处理的数据以及数据之间的关系,抽象出具体的数据模型,形成问题求解的基本思路。对于数值问题抽象出的数据模型通常是数学方程,对于非数值问题抽象出的数据模型通常是表、树、图等数据结构。

算法用来描述问题的解决方案,是具体的、机械的操作步骤。利用计算机解决问题的最重要一步是将人的想法描述成算法,即设计算法,也就是从计算机的角度设想计算机是如何一步一步完成这个任务的。由想法到算法需要完成数据表示和数据处理,即描述问题的数据模型,将数据模型从机外表示转换为机内表示;描述问题求解的基本思路,将问题的解决方案形成算法。

由算法到程序需要将算法的操作步骤转换为某种程序设计语言对应的语句,转换所依据的规则就是某种程序设计语言的语法,换言之,就是在某种编程环境下用程序设计语言描述要处理的数据以及数据处理的过程。

著名的计算机学者沃思<sup>①</sup>给出了一个公式:数据结构+算法=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重要的组成部分,一个“好”程序首先是将问题抽象出一个适当的数据结构,然后基于该数据结构设计一个“好”算法,或者说,学习数据结构的意义在于编写高质量、高效率的程序。下面以著名的哥尼斯堡七桥问题为例,说明

<sup>①</sup> 沃思(Niklaus·Wirth)1934年生于瑞士。1968年设计并实现了PASCAL语言,1971年提出了结构化程序设计,1976年设计并实现了Modula 2语言。除了程序设计语言之外,沃思在其他方面也有许多创造,如扩充了著名的巴科斯范式,发明了语法图等。1984年获图灵奖。

程序设计的一般过程。

**【问题】** 哥尼斯堡七桥问题(以下简称“七桥问题”)。17世纪的东普鲁士有一座哥尼斯堡城(现在叫加里宁格勒,在波罗的海南岸),城中有一座岛,普雷格尔河的两条支流环绕其旁,并将整个城市分成北区、东区、南区、岛区4个区域,全城共有七座桥将4个城区连接起来,如图1-2(a)所示。于是,产生了一个有趣的问题:一个人是否能在一次步行中经过全部的七座桥后回到出发点,且每座桥只经过一次。

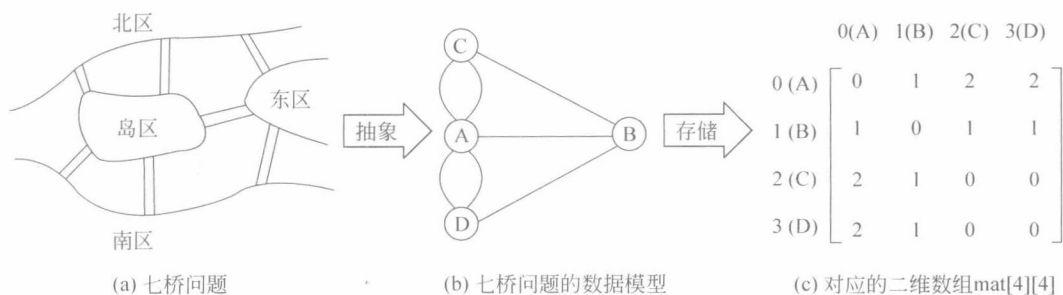


图 1-2 七桥问题的数据抽象过程

**【想法】** 将城区抽象为顶点,用 A、B、C、D 表示 4 个城区,将桥抽象为边,用 7 条边表示七座桥,抽象出七桥问题的数据模型如图 1-2(b)所示,从而将七桥问题抽象为一个数学问题:求经过图中每条边一次且仅一次的回路,后来人们称之为欧拉回路。欧拉回路的判定规则是:

- (1) 如果没有一个城区通奇数桥,则无论从哪里出发都能找到欧拉回路;
- (2) 如果通奇数桥的城区多于两个,则不存在欧拉回路;
- (3) 如果只有两个城区通奇数桥,则不存在欧拉回路,但可以从这两个城区之一出发找到欧拉路径(不要求回到出发点)。

由上述判定规则得到求解七桥问题的基本思路:依次计算与每个顶点相关联的边数,根据边数为奇数的顶点个数判定是否存在欧拉回路。

**【算法】** 将顶点 A、B、C、D 编号为 0、1、2、3,用二维数组  $\text{mat}[4][4]$  存储七桥问题的数据模型,如果顶点  $i(0 \leq i \leq 3)$  和顶点  $j(0 \leq j \leq 3)$  之间有  $k$  条边,则元素  $\text{mat}[i][j]$  的值为  $k$ ,如图 1-2(c)所示。求解七桥问题的关键是求与每个顶点相关联的边数,即是在二维数组  $\text{mat}[4][4]$  中求每一行元素之和,算法描述如下:

算法: EulerCircuit

输入: 二维数组  $\text{mat}[4][4]$

输出: 通奇数桥的顶点个数 count

1. count 初始化为 0;
2. 下标  $i$  从  $0 \sim n-1$  重复执行下述操作:
  - 2.1 计算第  $i$  行元素之和 degree;
  - 2.2 如果 degree 为奇数,则 count++;
3. 返回 count;

**【程序】** 主函数首先初始化二维数组 `mat[4][4]`，即表示七桥问题对应的数据模型，然后调用函数 `EulerCircuit` 计算图模型中通奇数桥的顶点个数，再根据欧拉规则判定是否存在欧拉回路。程序如下：

```
#include <stdio.h>
int EulerCircuit(int mat[4][4],int n);    /* 函数声明,求通奇数桥的顶点个数 */

int main()
{
    int mat[4][4] = {{0,1,2,2},{1,0,1,1},{2,1,0,0},{2,1,0,0}};
    int num = EulerCircuit(mat,4);        /* 调用函数得到通奇数桥的顶点个数 */
    if (num >= 2)                          /* 两个以上的顶点通奇数桥 */
        printf("有%d个地方通奇数桥,不存在欧拉回路\n",num);
    else                                    /* 没有顶点通奇数桥 */
        printf("存在欧拉回路\n");
    return 0;
}

int EulerCircuit(int mat[4][4],int n)      /* 函数定义,二维数组作为形参 */
{
    int i,j,degree,count = 0;              /* count 累计通奇数桥的顶点个数 */
    for (i = 0; i < n; i++)                 /* 依次累加每一行的元素 */
    {
        degree = 0;                          /* degree 存储通过顶点 i 的桥数,初始化为 0 */
        for (j = 0; j < n; j++)
        {
            degree = degree+mat[i][j];        /* 将通过顶点 i 的桥数求和 */
        }
        if (degree % 2 != 0)                 /* 桥数为奇数 */
            count++;
    }
    return count;                            /* 结束函数,将 count 返回到调用处 */
}
```

### 1.1.2 数据结构在程序设计中的作用

有些问题难以求解的原因是无从下手,如果能将问题抽象出一个合适的数据模型,则问题可能会变得豁然开朗。下面这个著名的握手问题是爱丁堡大学的 Peter Ross 提出来的。

**【例 1-1】** 握手问题。Smith 先生和太太邀请 4 对夫妻来参加晚宴。每个人来的时候,房间里的一些人都要和其他人握手。当然,每个人都不会和自己的配偶握手,也不会跟同一个人握手两次。之后,Smith 先生问每个人和别人握了几次手,他们的答案都不一样。问题是,Smith 太太和别人握了几次手?



解：这个问题具有挑战性的原因是没有一个明显的思考点，但如果将此问题抽象出一个合适的数据模型，如图 1-3(a)所示，问题就变得简单了。

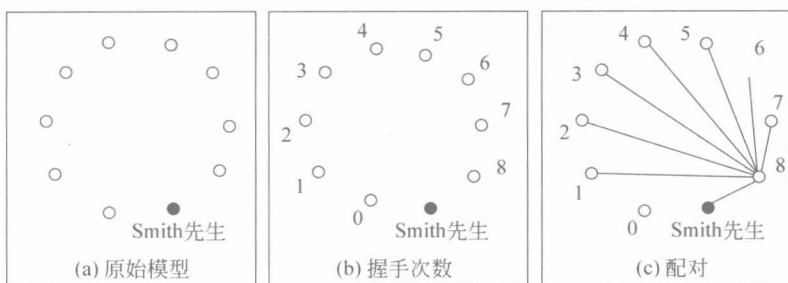


图 1-3 握手问题的数据模型

让我们来收集一下已知条件。Smith 先生问了房间的 9 个人，每个人的答案都不相同，因此，每个答案都在 0 和 8 之间，相应地修改模型如图 1-3(b)所示。

让我们来列出所有的握手信息。例如，8 号除了他(她)自己和配偶，与房间里的其他人总共握了 8 次手。基于这个观察，可以画出 8 号的握手信息并且知道 8 号的配偶是 0 号，如图 1-3(c)所示；7 号除了 0 号和 1 号，与房间里的其他人总共握了 7 次手，因此 7 号的配偶是 1 号……问题是不是变得豁然开朗了？

一个“好”程序首先是将问题抽象出一个适当的数据模型，基于不同数据模型的算法，其运行效率可能会有很大差别，例如下面的手机电话号码查询问题。

**【例 1-2】** 手机电话号码查询问题。假设某手机中存储了如表 1-1 所示的若干电话号码，如何在手机中查找某人的电话号码？

解：如果将电话号码集合线性排列，则查找某人的电话号码只能进行顺序查找。如果将电话号码集合进行分组，如图 1-4 所示，则查找某人的电话号码可以只在某个分组中进行。显然，后者的查找效率更高，当数据量较大时差别就更大。

表 1-1 某手机中的电话号码

姓名	王靓靓	赵刚	韩春颖	李琦勇	...	张强
电话	13833278900	13944178123	15594434552	13833212779	...	13331688900



图 1-4 分组——将电话号码集合组织为树结构