清华大学计算机系列教材

数据结构习题解析

（用面向对象方法与C++语言描述）

殷人昆
徐孝凯 编著

清华大学出版社
http://www.tup.tsinghua.edu.cn

清华大学计算机系列教材

清华大学计算机系列教材

数据结构习题解析

（用面向对象方法与C++语言描述）

殷人昆　徐孝凯　编著

清华大学出版社

(京)新登字 158 号

内 容 简 介

本书是清华大学出版社出版的《数据结构　用面向对象方法与 C++语言描述》主教材的配套教学参考书。本书首先介绍了"数据结构"课程的学习指导和考试指导；然后，从每章的"复习提要"、"难点和重点"开始，给出主教材中绝大多数习题的解析和参考答案；最后，针对每章涉及的概念，又补充了一部分练习。

本书可作为大学计算机专业"数据结构"课程的教学参考书，也可供参加硕士、博士、工程硕士研究生入学考试的考生以及计算机应用技术人员参考。

版权所有，翻印必究。
本书封面贴有清华大学出版社激光防伪标签，无标签者不得销售。

图书在版编目(CIP)数据

数据结构习题解析：用面向对象方法与 C++语言描述 殷人昆，徐孝凯编著. —北京：清华大学出版社，2002.3
清华大学计算机系列教材
ISBN 7-302-05288-3

Ⅰ. 数… Ⅱ. ①殷…②徐… Ⅲ. 数据结构－解题－高等学校　学习参考资料 Ⅳ. TP311.12-44

中国版本图书馆 CIP 数据核字(2002)第 013197 号

出版者：清华大学出版社(北京清华大学学研大厦，邮编 100084)
http://www.tup.tsinghua.edu.cn
印刷者：北京昌平环球印刷厂
发行者：新华书店总店北京发行所
开　本：787×1092 1/16　印张：16　字数：387 千字
版　次：2002 年 4 月第 1 版　2002 年 7 月第 3 次印刷
书　号：ISBN 7-302-05288-3/TP・3108
印　数：8001～15000
定　价：26.00 元

前　言

“数据结构”是有关计算技术及信息管理技术专业的一门必修的核心课程。“数据结构”课程的任务是讨论在应用问题求解时数据的逻辑组织、在计算机中的存储实现以及相关操作的算法。“数据结构”课程的目的是使学生掌握在实际问题解决过程中如何组织数据、如何存储数据和如何处理数据的基本方法，为进一步学习后续课程，以及为以后从事软件开发和应用打下坚实的基础。

本书是清华大学出版社出版的“清华大学计算机系列教材”《数据结构(用面向对象方法与C++语言描述)》和“教育部人才培养模式和开放教育试点教材”《数据结构》的配套教材，提供了学习“数据结构”的一些指导性建议和考试的样例。它给出主教材中绝大多数习题的参考答案和分析，还针对基本概念补充了一些知识性的练习，对于复习和准备考试的学生有一定参考价值。但对于正在学习“数据结构”课程的学生，应以掌握知识和培养能力为主，不应过多地依赖现成的习题解答。本书只应作为一个参考，不应当作拐杖。

当前，数据结构的教授方法有两种。一种是注重数据结构本身的知识体系，认为“数据结构”课程应以学生是否能掌握在应用开发中如何组织、存储、变换数据为主，使用什么语言描述数据结构和算法都可以。持这种观点的教材多以类Pascal、类C或其他伪码作为数据结构和算法的描述工具。另一种是注重数据结构的工程应用，直接以某种可以在计算机上实现的语言来描述数据结构和算法。随着软件开发技术的发展，后一种数据结构的教学方法在国际上普遍得到认可。因为面向对象的软件分析和设计方法已经成为软件开发的主流方法，如果学生在学习数据结构和程序设计的课程时养成一种动辄用面向过程的传统方法解决问题的习惯，将来学习面向对象方法时将难以接受新的思维，或者不知如何着手，因此，必须及早改变这种学习方式，从学习面向对象程序设计和用面向对象方法描述的数据结构入手，培养学生用先进的方法来描述问题和解决问题的能力。

本书的主教材在清华大学计算机系已经讲授了3轮，在中央电大计算机应用专业也应用了两年。从学生反映来看，是成功的。我们将根据在教学实践中取得的经验和暴露的问题，不断改进教学体系和教学内容，增强实践环节，使“数据结构”课程能够紧跟国际发展，为培养高水平计算机软件人员打下良好的基础。

本教材是基于多年来的教学实践整理成的，希望能对广大读者的学习起到促进作用。但由于时间仓促和我们的水平所限，错误和疏漏在所难免，敬请读者提出宝贵意见。

作　者

2002年1月

目　录

试读结束：需要全本请在线购买：　www.ertongbook.com

第0章　数据结构导论

0.1　数据结构学习指导

0.1.1　课程地位

“数据结构”是计算机科学与技术专业本科生的专业基础课程之一。用计算机解决任何实际问题都离不开数据表示和数据处理，而数据表示和处理的核心问题之一是数据结构及其实现——这正是数据结构课程的基本内容。从这个意义上来说，数据结构课程在知识学习和技能培养两个方面都是要求达到的目标，是理论和实践要求都相当高的课程。本课程不仅为操作系统、数据库系统、编译方法、计算机网络等后续课程提供了必要的知识基础，而且也为计算机及其专业人员提供了必要的技能训练。

对清华大学计算机系历届毕业生和部分研究生追踪调查显示：几乎所有的学生都认为“数据结构”是他们在学校里学过的最有用的课程之一，它也是国内外许多软件开发机构要求考核的基本课程之一。由此可见“数据结构”这门课程的重要性。

0.1.2　课程要求

根据课程的教学大纲要求，“数据结构”主要讨论在软件开发中如何进行数据结构和算法的设计。因此，用抽象数据类型以及面向对象的方法组织、存储各种类型的数据是本课程的重点，也是学生需要掌握的重点。面向对象方法以及结构化技术都是建立高质量软件的技术，通过“数据结构”课程的学习和实践，可以加深对这些先进软件开发方法的理解和体会。因此，“数据结构”课程的任务是按照软件工程思想，介绍用面向过程和面向对象方法进行数据设计和程序设计的基本思想，在必要的课程实践中逐步熟练掌握。

通过本课程的学习，应达到知识和技能两方面的目标：

(1) 知识方面：从数据结构的类定义和对象的使用、存储表示和操作的实现这两个层次，系统地学习和掌握常用的基本数据结构(包括数组、顺序表、多项式、字符串、链表、栈与队列、优先级队列、广义表、树与森林、二叉树、堆、集合、图、搜索结构、索引结构、散列结构等)及其不同的实现，了解并掌握分析、比较和选择不同数据结构、不同存储结构、不同算法的原则和方法，为后续课程的学习打好基础。

(2) 技能方面：系统地学习和掌握对象类的设计方法和面向对象的程序设计风格，在不同的存储结构上实现的算法的设计思想，从中体会和掌握选择结构的方法和算法设计的思考方式及技巧，提高分析问题和解决问题的能力。

0.1.3 课程学习指导

学生在初学时往往感到“数据结构”课程内容多、面授容易接受，但自学难度大，特别是在编写小程序时常常无从着手。为改善自学效果以有限的时间和精力学好这门课程，应当注意以下几点：

(1) 注意先修课程的知识准备

在学习本课程之初，必须注意复习在用 C/C++程序设计语言编写小程序时的语法规则和方法，为本课程的学习打下基础。

C 语言与 Pascal 语言一样，是一种面向过程的语言。C 程序结构的特点是遵循“输入-处理-输出”的模式来解决问题。C++保留了 C 的面向过程编程的成分，但由于引入了面向对象的成分，在数据组织方面很自然地实现了抽象数据类型的思想，并利用模板机制实现了软件复用。在复习 C/C++语言时应注意：

① 函数的概念和相关问题。包括函数类型，函数特征，函数名重载，函数参数的传递。特别注意传值参数和引用参数在使用上的区别。还有函数返回值的 4 种类型之间的区别。

② 函数中局部变量的作用域，它的创建和回收的有效范围。特别注意在函数中对局部变量的任何改变，因为在退出函数过程时局部变量被释放而不能当作函数返回值返回。

③ 类和对象的定义方式。特别注意为保证信息隐蔽，对类中的所有数据成员和成员函数将规定其存取级别：public，private 和 protected。放在 public 域中表示对于其他程序都是可用的；放在 private 域中表示对于其他程序是不可用的，对于其派生类也是不可用的；放在 protected 域中表示对于其他程序是不可用的，对于其派生类是可用的。

④ 在 C++中的动态存储分配和动态存储回收方式。

⑤ 类的实例对象的建立和回收方式，构造函数和析构函数的使用，对象数据成员初始化的方式等。特别注意成员函数的作用域。

⑥ 在 C/C++中的输入输出流文件的定义和使用。特别注意文件的连接、ifstream 类和 ofstream 类，文件的打开与关闭、文件模式的作用。

⑦ 友元类与友元函数的定义和使用。

⑧ 模板类的定义和模板类的使用。

(2) 在学习“数据结构”时，要注意知识体系

数据结构课程中的知识本身具有良好的结构性。从总体上来说，课程的主要内容是围绕着数组、线性表、顺序表、链表、栈、队列、优先级队列、广义表、树与二叉树、图、集合与搜索结构、索引与散列结构等常用的数据结构和递归、排序等常用算法来组织的。其中有些结构是面向应用的，有些结构是面向实现的。在课文中要注意这两个层次以及它们之间的联系。

为了完全了解数据结构的机制以及为便于数据结构的复用，课文中对每一个结构都做了较为完整的介绍，学习者最好将课文中介绍的每一结构都能以“模板类”和“模板类的成员函数的实现”的形式存于“.h”文件中，并在后续的算法或类的实现中直接复用它们。

(3) 注意比较

在学习中应当注意从“横向”和“纵向”进行对比以加深理解。纵向对比将一种结构与它的各种不同的实现加以比较，从继承的角度或从聚合的角度建立类的实例之间的联系；横向

对比包括具有相同逻辑结构的不同数据结构(如线性表、栈、队列、优先级队列和串)的比较，具有相同功能的不同算法的比较等，了解类的层次结构和利用包的概念将不同语义的类关联起来，从而加深对抽象类和虚函数的使用。

(4) 注意复习和反复阅读

有些内容在初读时难以透彻理解或熟练掌握，或者看起来似乎明白了，但用的时候容易犯晕。在继续学习的过程中如遇到或用到有关内容时，应当及时复习或重读，这往往能够化难为易，温故知新。

(5) 充分利用考核说明及有关学习的难点和重点的说明

在进入每一章学习之前或在每一章学习之后，仔细阅读考核说明和各章学习的难点和重点，有助于集中思路和自我检查。如自我检查的效果不理想，千万不要将就，更不要急于向后读，应回过头来把握住要求，有计划地重读课本，重看学习导引，重做习题，直到本章内容掌握为止。

(6) 注意循序渐进

在进入具体内容(如类的定义和类的成员函数的实现)之前，首先领会基本概念、基本思想，这一点极为重要。特别是在阅读算法之前，一定要先弄清其基本思想、基本步骤，这将大大降低理解算法的难度。如果读"懂"了算法而不知其基本思想，仍不算是真懂。可以通过事例学习以加深理解。

(7) 注意练习

习题练习是课程的基本要求之一。只看书不做题，不可能真正学会有关知识，更不能达到技能培养的目的。另一方面，做题也是自我检查的重要手段。此外，在做算法设计型的习题时，应优先考虑数据结构的定义，可直接使用以前定义的类和相应的操作(成员函数)，综合已学过的知识，以提高编程的能力。为此，可以逐渐积累学过的数据结构的实现，以模板类的方式标准化，以方便在后续学习中复用它们。这样做所获得的回报是相当丰厚的，但不要指望立即获得回报。

(8) 算法思路的理解

算法及其思想、评价是本课程的重点之一，在课文中占了相当大的比例。在学习每一个算法时，建议首先理解问题的要求，在此基础上利用一个事例来模拟问题的求解，自己试着构思一下解题的思路，能够写出来更好。然后再阅读算法，在读懂之后，不要急于往下进行，而应停下来思考下列问题：该算法从逻辑上可以划分为哪几个部分(步骤)？每一部分的功能是什么？这一功能又是如何实现的？能否用别的方法实现？各部分之间的关系如何？如果问题的要求有所不同，应当如何修改算法？

将思考的结果记载下来，在复习时会有很大帮助。另外，这样的分析也是灵活运用所学知识的关键。

(9) 努力培养算法设计的能力

在课程学习中最令学习者感到吃力的是设计和编写算法。算法的设计水平是软件设计的基础。要提高算法的设计水平，首先需要掌握问题要求的基本内容，通过反复体会和练习来实现。通常需要根据具体问题的要求，选择合适的数据结构，设计有效的算法。有条件的学生应当在机器上多做练习。

(10) 及时总结

在学习完每一节、章后，要及时地进行总结，用简短的文字记录这部分的内容，尽可能用自己的话复述一遍。在本课程全部学完后还应对本课程进行全面系统的复习和总结，认真做模拟试卷。在进行总复习时，可通过对比各种数据结构的异同和它们之间的相互关系，加深对各种数据结构的理解。

在复习时，可以按所记录的要点反向地进行，即依据要点找出其具体内容。按这样的方法进行，也许还能够节省时间，提高学习效果。

最后应当强调的是，学习方法主要靠自己摸索，多总结，多思考，勤上机，勤交流，这是把“数据结构”这门课程真正学好的关键。

0.2 考试指导

数据结构考试可能的题型有以下 5 种：

单选题：给出一些有关数据结构性质、特点及一些简单算法性能的不完全叙述，要求学生从题后给出的供选择的答案中选择合适的答案，补足这些叙述；

填空题：给出程序说明及一段部分语句缺失的程序，让学生补充成为完整的程序；

简答题：应用作图方法或简单计算，使用给定数据建立或操作一些数据结构；

判断题：给出一些有关数据结构或算法的叙述，要求学生回答这些叙述正确与否；

综合算法题：给出算法设计要求，编制出部分算法程序，用来考察几个知识点的综合应用。

下面根据各种题型举例分析，说明解题方法及考试时的注意事项。

0.2.1 单选题

【例 1.1】从供选择的答案中选出正确的答案，将其编号填入下列叙述中的(　　)内。

(1) 向一个有 127 个元素的顺序表中插入一个新元素并保持原来顺序不变，平均要移动(　A　)个元素。

(2) 设有一个二维数组 $A[m][n]$，假设 $A[0][0]$ 存放位置在 $644_{(10)}$，$A[2][2]$ 存放位置在 $676_{(10)}$，每个元素占一个空间，则 $A[4][5]$ 在(　B　)位置，$_{(10)}$ 表明用 10 进数表示。

(3) 一个有序顺序表有 255 个对象，采用顺序搜索法查表，平均搜索长度为(　C　)。

(4) 含 5 个结点(元素值均不相同)的二叉搜索树有(　D　)种。

(5) 在分析折半搜索的性能时常加入失败结点，即外结点，从而形成扩充的二叉树。若设失败结点 i 所在层次为 l_i，那么搜索失败到达失败结点时所做的数据比较次数是(　E　)。

(6) 设有一个含 200 个表项的散列表，用线性探查法解决冲突，按关键码查询时找到一个表项的平均探查次数不超过 1.5，则散列表项应能够至少容纳(　F　)个表项。(设搜索成功的平均搜索长度为 $S_{nl}=\{1+1/(1-\alpha)\}/2$，其中 α 为装填因子)

(7) n 个顶点的连通图至少有(　G　)条边。

(8) 一个二叉树按顺序方式存储在一个一维数组中，如图

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	B	C	D		E	F		G			H		I	J

则结点 E 在二叉树的第(H)层。

供选择的答案

A. ① 8　② 63.5　③ 63　④ 7

B. ① $692_{(10)}$　② $626_{(10)}$　③ $709_{(10)}$　④ $724_{(10)}$

C. ① 128　② 127　③ 126　④ 255

D. ① 54　② 42　③ 36　④ 65

E. ① l_i+1　② l_i+2　③ l_i-1　④ l_i

F. ① 400　② 526　③ 624　④ 676

G. ① $n-1$　② n　③ $n+1$　④ 0

H. ① 1　② 2　③ 3　④ 4

【解答】A. ②　B. ③　C. ①　D. ②　E. ④　F. ①　G. ①　H. ②

【分析】此类题主要是考核学生对基本知识的掌握程度，涉及范围较广。所有各章内容都可能牵涉到。主要考察对各种数据结构的定义、特点、性质(包括公式计算)、主要操作的性能分析(包括算法中多趟执行时每一趟的通项公式)，因此要求复习时必须仔细看书。不提倡死记硬背，但要学会推导。

0.2.2 判断题

【例 2.1】判断下列各个叙述的正误。对，在题号前的括号内填入“√”；错，在题号前的括号内填入“×”

(　)(1) 有 n 个结点的不同的二叉树有 $n!$ 棵。

(　)(2) 直接选择排序是一种不稳定的排序方法。

(　)(3) 在 2048 个互不相同的关键码中选择最小的 5 个关键码，用堆排序比用锦标赛排序更快。

(　)(4) 当 3 阶 $B_$树中有 255 个关键码时，其最大高度(包括失败结点层)不超过 8。

(　)(5) 一棵 3 阶 $B_$树是平衡的 3 路搜索树，反之，一棵平衡的 3 路搜索树是 3 阶 $B_$树。

(　)(6) 在用散列表存储关键码集合时，可以用双散列法寻找下一个空桶。在设计再散列函数时，要求计算出的值与表的大小 m 互质。

(　)(7) 在只有度为 0 和度为 k 的结点的 k 叉树中，设度为 0 的结点有 n_0 个，度为 k 的结点有 n_k 个，则有 $n_0=n_k+1$。

(　)(8) 折半搜索只适用于有序表，包括有序的顺序表和有序的链表。

【解答】(1) ×　(2) √　(3) ×　(4) √　(5) ×　(6) √　(7) ×　(8) ×

【分析】此类题要求对课文各章中许多细节比较清楚。事实上，数据结构课程中讲到的许多数据结构的细节是最基础的知识，将这些内容理解透彻了，以后学生在自主开发软件时将会得到回报的。在解答此类题时，不要立即提笔就答，应当把题看完，分析前后叙述是否矛盾，

试读结束：需要全本请在线购买：　www.ertongbook.com

是否叙述中有漏洞，必要时，还需要做一下简单的推导。

0.2.3 阅读理解题

【例 3.1】说明下列递归过程的功能。

```
int unknown(BinTreeNode * t) {
//指针 t 是二叉树的根指针
    if (t==NULL) return 0;
    else
      if (t->leftChild==NULL && t->rightChild==NULL) return 1;
      else return unknown(t->leftChild)+unknown(t->rightChild);
}
```

【解答】计算二叉树的叶结点的个数。

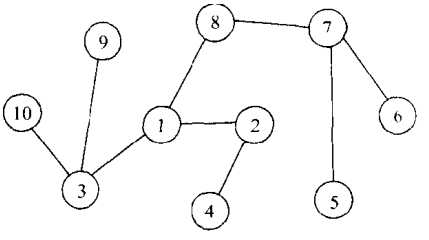
【分析】此类题是递归算法问题，一般要求读懂它即可。但要理解它首先需要掌握递归算法的结构，以及相应问题的背景。有时需要用一个简单的例子来帮助理解。

0.2.4 简答题

【例 4.1】如下所示的连通图，请画出：

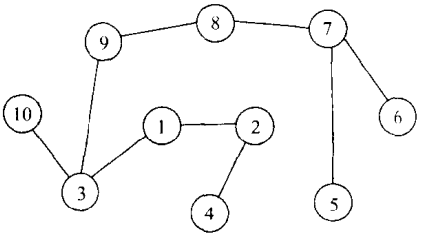
(1) 以顶点①为根的深度优先生成树；

(2) 如果有关节点，请找出所有的关节点。



【解答】

(1) 该连通图从①出发做深度优先搜索，得到的深度优先生成树为：



结果不唯一

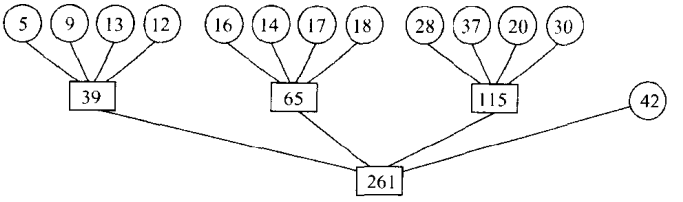
(2) 关节点为 ①、②、③、⑦、⑧

【分析】这是有关重连通图的问题。深度优先搜索是一个需要重点掌握的方法，包括实例分

析和算法。它是一个带回溯的探索问题，需要用递归算法求解。另外，关节点的判断准则有两个：其一为根至少有两个或两个以上的子女；其二为除根和叶结点外，其他结点的子女或子孙有指向该结点祖先的“回边”。掌握了这些概念，此问题就迎刃而解了。还需要了解：重连通图中任意两个顶点之间至少有两条路径相通，这有助于将非重连通图改为重连通图。

【例 4.2】 设有 13 个初始归并段，其长度分别为 28，16，37，42，5，9，13，14，20，17，30，12，18。试画出 4 路归并时的最佳归并树，并计算它的带权路径长度 WPL。

【解答】 因为(13－1)%(4－1)＝12%3＝0，所以不需要添加空段。最佳归并树为



WPL＝(5＋9＋13＋12＋16＋14＋17＋18＋28＋37＋20＋30) * 2＋42＝4

【分析】 这是外排序中求最佳归并树问题，它是只有度为 0 和度为 k 的正则 k 叉树，满足条件 $n_0=(k-1)n_k+1$，其中，n_0 是度为 0 的结点个数，n_k 是度为 k 的结点个数，k 是归并路数。最佳归并树的构造仿照霍夫曼树，长度小的初始归并段离根远，长度大的初始归并段离根近。需要注意的是：当 $(n_0-1)\%(k-1)=m\neq 0$ 时，需要补充 $k-m-1$ 个空的初始归并段，才能构成正则 k 叉树。

【例 4.3】 设散列表为 HT[0..12]，即表的大小为 $m=13$。采用双散列法解决冲突。散列函数和再散列函数分别为：

H_0(key)＝key%13；　(注：%是求余数运算)

$H_i=(H_{i-1}+\mathrm{REV}(\mathrm{key}+1)\%11+1)\%13$；　$i=1, 2, 3, \cdots, m-1$

其中，函数 REV(x)表示颠倒 10 进制数 x 的各位，如 REV(37)＝73，REV(7)＝7 等。若插入的关键码序列为 2，8，31，20，19，18，53，27。试画出插入这 8 个关键码后的散列表。

【解答】 散列表 HT[0..12]，散列函数与再散列函数为

H_0(key)＝key %13；

$H_i=(H_{i-1}+\mathrm{REV}(\mathrm{key}+1)\%11+1)\%13$；

插入关键码序列为{2，8，31，20，19，18，53，27}

$H_0(2)=2$，比较次数为 1

$H_0(8)=8$，比较次数为 1

$H_0(31)=5$，比较次数为 1

$H_0(20)=7$，比较次数为 1

$H_0(19)=6$，比较次数为 1

$H_0(18)=5$，冲突，$H_1=9$，比较次数为 2

$H_0(53)=1$，比较次数为 1

$H_0(27)=1$，冲突，$H_1=7$，冲突，$H_2=0$，比较次数为 3

插入 8 个关键码后的散列表

0	1	2	3	4	5	6	7	8	9	10	11	12
27	53	2			31	19	20	8	18			

【分析】散列表长度和再散列函数的选取是关键。由于要求再散列函数计算出来的结果，即发生冲突时向后跨多大距离找下一个“空位”，必须与表的长度互质。因此为简化问题求解的难度，一般设表的长度为质数，本题设为 13。其次，再散列函数是待散列对象的关键码的函数，计算结果的取值范围在 1～$n-1$ 之间，其中 n 是表的长度。因此选取 REV(key+1)%11+1，其取值范围在 1～11 之间，除 1 退化为线性探查外，其他都与 13 互质。需要注意的是：双散列法对表长度的要求不如二次探查法严格，那里要求表的长度一定是满足 $4j+3$ 的质数且表的装载因子不超过 0.5。此题容易出错的地方是应用再散列函数处理冲突时发生简单计算错误，一处出错，会影响后面一连串出错，一定要仔细。

0.2.5 综合算法题

【例 5.1】有一种简单的排序算法，叫做计数排序(count sorting)。这种排序算法对一个待排序的表(用数组表示)进行排序，并将排序结果存放到另一个新的表中。必须注意的是，表中所有待排序的关键码互不相同。计数排序算法针对表中的每个记录，扫描待排序的表一趟，统计表中有多少个记录的关键码比该记录的关键码小。假设针对某一个记录，统计出的计数值为 c，那么，这个记录在新的有序表中的合适的存放位置即为 c。

(1) 给出适用于计数排序的数据表定义；

(2) 使用 C++语言编写实现计数排序的算法；

(3) 对于有 n 个记录的表，关键码比较次数是多少？

【解答】

(1) 数据表定义

```
const int  DefaultSize=100;
template <class Type> class datalist                    //数据表的前视声明
template <class Type> class Element {                   //数据表元素类的定义
private:
  Type key;                                             //关键码
  field otherdata;                                      //其他数据成员
public:
  Type getKey() { return key; }                         //取当前结点的关键码
  void setKey( const Type x) { key=x; }                 //将结点的关键码修改为 x
}

template <class Type> class datalist {
//用顺序表来存储待排序的元素，这些元素的类型是 Type
public:
  datalist(int MaxSz=DefaultSize) : MaxSize(Maxsz), CurrentSize(0)
    { Vector=new Element <Type> [MaxSz]; }
private:
  Element <Type> * Vector;                              //存储待排序元素的向量
  int MaxSize, CurrentSize;                             //最大元素个数与当前元素个数
```

```
}
```

(2) 计数排序的算法

【方案 1】

```
template <class Type>
void countsort(datalist<Type>& initList, datalist<Type>& resultList) {
  int i, j, c; Type refer;
  for(i=0; i < initList.CurrentSize; i++) {
    c=0;
    refer=initList.Vector[i].getKey();
    for ( j=0; j < initList.CurrentSize; j++)
      if (initList.Vector[j].getKey() < refer) c++;
    resultList.Vector[c]=initList.Vector[i]
  }
  resultList.CurrentSize=initList.CurrentSize;
}
```

【方案 2】

```
template <class Type>
void countsort(datalist<Type> & initList, datalist<Type> & resultList) {
   int * c=new int[initList.CurrentSize];
   for(int i=0; i < initList.CurrentSize; i++) c[i]=0;
   for(i=0; i < initList.CurrentSize-1; i++)
     for ( int j=i+1; j < initList.CurrentSize; j++)
       if (initList.Vector[j].getKey() < initList.Vector[i].getKey()) c[i]++;
       else c[j]++;
   for(i=0; i < initList.CurrentSize; i++)
     resultList.Vector[ c[i] ]=initList.Vector[i];
   resultList.CurrentSize=initList.CurrentSize;
}
```

(3) 对于 n 个记录的表,【方案 1】中关键码比较次数为 n^2,【方案 2】中关键码比较次数为 $n(n-1)/2$。

【分析】编写算法的题可能是学生比较棘手的问题,特别是在考试这样一个氛围,时间又短促,想编出一个好算法不太容易。一个建议是首先仔细阅读试题,了解它到底要你干什么。然后用一个简单的例子走一下,总结每一步向下走用什么语句,再做归纳。也可以按照结构化程序设计的方法,先搭框架,再根据例子填入细节。总之,要想又快又好地编写出算法,还要靠平时的基本训练,多做题,自主做题,各种题型都见过了,考试时自然文思泉涌,笔下就流畅了。

0.2.6 填空题

【例 6.1】下面给出的是一个在二叉树中查找值为 x 的结点,并打印该结点所有祖先结点的

算法。在此算法中，假设值为 x 的结点不多于一个。此算法采用后序的非递归遍历形式。因退栈时需要区分其左、右子树是否已经遍历，故在结点进栈时附带有一个标志，=0，进入左子树，=1，进入右子树。用栈 ST 保存结点指针 ptr 及其标志 tag。top 是栈顶指针。

```
void print(BinTreeNode * t;  Type &x) {
    stack ST; int i, top;
    top=0;                                          //置空栈
    while(t != NULL && t->data != x || top != 0) {
      while(t != NULL && t->data != x) {  //寻找值为 x 的结点
          [          ①          ];
          ST [top].ptr=t;                           //进栈
          ST [top].tag=0;
          [          ②          ];
      }
      if(t != NULL && t->data==x)          //找到值为 x 的结点
          for(i=1;      [    ③    ];  i++)
          cout << ST[i].ptr->data << endl;
          else {
      while(top > 0 && ST [top].tag==1)    //未找到值为 x 的结点
          top--;
      if(top > 0) {
      ST [top].tag=1;                           //转向右子树
[                    ④                    ];
   }
}
}
} /* print */
```

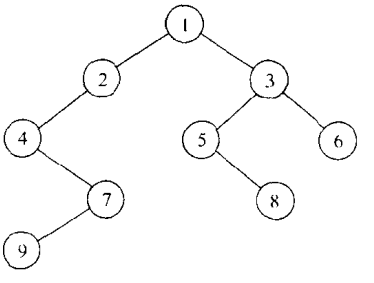
（1）请将缺失的语句补上 （8 分）

①

②

③

④



（2）请给出对于右图所示的二叉树，使用上述算法搜索值为 9 的结点和值为 10 的结点的结果，以及在栈 ST 中的变化。（top 是栈顶指针）

【解答】

（1）请将缺失的语句补上

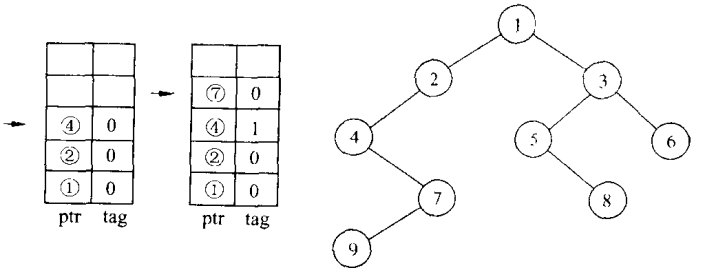
① top++

② t=t->leftChild

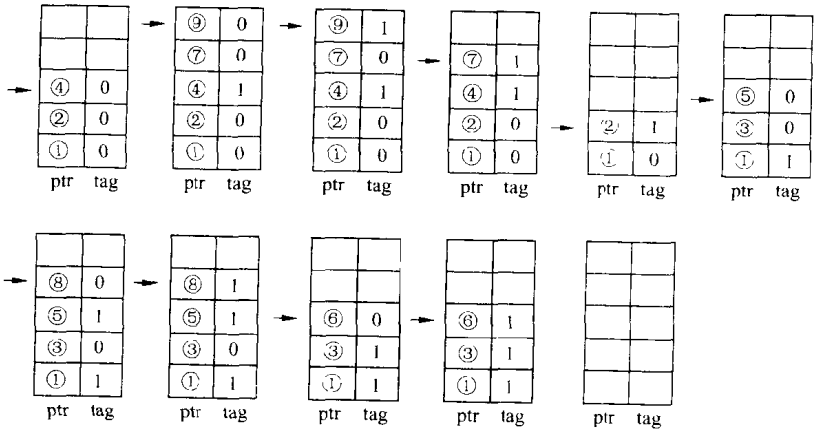
③ i <= top

④ t=ST[top].ptr->rightChild

(2) 搜索值为 9 的结点



搜索值为 10 的结点



【分析】人们常说，读别人的程序不如自己重编。此话有一定道理。各人的思路不同，编程语言的运用和编程技巧的水平不同，编程风格不同，即使是几个人编写同一个功能的程序，可能面貌相差极大，读别人的程序自然困难很大。但现实是将来要做技术支持必须读别人的程序。所以，必须有这方面的基本训练和技能。一个建议是：通过试题叙述阅读来了解思路，通过实例分析来了解程序结构，再通过实例跟踪来检验填空后的程序，从而得到最终的解答。

0.3 小　结

数据结构课程是计算机专业的基础课程，是一门知识性、实践性都很强的课程。数据结构及算法编写与分析都是不容易的。学好它需要付出极大的功夫，在平时勤学多练。如果有碰运气或取巧的想法，十有八九通不过考试。有一次在中央电大直播课堂进行期末答疑，一位学员不问问题，在网上传来一句话："救救我吧，我已经考了两次没通过了。"对此，只能为他遗憾。希望他能重新学习，把基础打扎实，考试才能顺利过关。就怕平时不看书，不做作业或抄别人的作业，到考试时照样不会，自然一次次通不过了。最后，祝愿同学们通过自己的勤奋努力，取得优良的成绩。

试读结束：需要全本请在线购买：　www.ertongbook.com