21世纪高等院校计算机系列教材

Computer

□ 李克清 主编

数据结构导学
——C语言描述

华中科技大学出版社

21 世纪高等院校计算机系列教材

数据结构导学——C 语言描述

主　编：李克清

副主编：张方山　崔洪芳

操保华　章　英　夏祥胜

华中科技大学出版社

图书在版编目(CIP)数据

数据结构导学——C 语言描述/李克清　主编
武汉:华中科技大学出版社,2005 年 2 月
ISBN 7-5609-3308-4

Ⅰ. 数…
Ⅱ. 李…
Ⅲ. 数据结构
Ⅳ. TP311

数据结构导学——C 语言描述　　李克清　主编

责任编辑:彭保林　曾光　　封面设计:刘　卉
责任校对:陈　骏　　责任监印:熊庆玉

出版发行:华中科技大学出版社
武昌喻家山　　邮编:430074　　电话:(027)87557473

录　　排:北京搜获科技有限公司
印　　刷:华中科技大学印刷厂

开本:787×1092　1/16　　印张:10.5　　字数:230 000
版次:2005 年 2 月第 1 版　　印次:2005 年 2 月第 1 次印刷　　定价:16.00 元
ISBN 7-5609-3308-4/TP・544

内 容 简 介

本书为配合《数据结构》课程的教学和实习而编写，与同期出版的《数据结构——C语言描述》教材相配套。本书内容分为两部分，第一部分为习题解析，给出了教材各章后面习题的分析与解答，并且针对各章内容提供了一定数量的练习题，同时附上了参考答案或提示。第二部分为课内实习，对一些典型的数据结构，提供了密切相关的实习项目，包括实习题目、实习目的、分析以及大部分可操作的核心代码。

全书内容丰富，涉及面广，实用性强，与《数据结构》课程内容紧密结合。可供各类学生课程学习实践与考前复习使用，也可供《数据结构》教学的教师和相关专业技术人员参考。

“21世纪高等院校计算机系列教材”丛书编委会

主任　何炎祥

委员（按姓氏拼音排序）：

戴光明	都志辉	桂　超	金先级
柯敏毅	李康顺	李克清	李禹生
刘腾红	卢强华	陆　迟	吕顺营
沈海波	石　清	王江晴	王伟军
王　忠	叶骏民	余敦辉	湛为芳

序　言

21世纪是信息时代，以计算机为核心的信息技术是21世纪科技发展的大趋势。作为计算机专业人才培养基地的大学计算机专业和相关专业，如何适应这种发展，培养出符合时代要求和社会欢迎的人才，是近年来计算机教育界讨论的热门话题，也是我们长期思考并努力探索的课题。

教材是人才培养的基础。在华中科技大学出版社的委托下，我们组织了有关高等院校的部分专家、教授共同编写了这套“面向21世纪计算机系列教材”，以期在适应21世纪的教材建设方面做出自己的努力。由于计算机行业发展日新月异，“21世纪计算机系列教材”编委会将负责系列教材的选题、每本教材大纲的编写和审定，以及教材、教学辅导书和课件的修订、更新等工作，以确保教材的正确性和先进性，使这套教材努力走在同类教材的前列。

这套系列教材包括计算机专业课和部分专业基础课教材，以及与之配套的实践课教材和教学辅导书等等。

我们希望这套教材具有以下特点：

1. 注重基础性和先进性的结合。计算机学科的一个显著特点就是知识和技术更新快，这对教学内容、课程知识结构的选取和组织提出了新的要求。我们把编写的重点放在基础知识、基本技能和基本方法上，希望在提高学生的理论素养和分析问题、解决问题的能力的同时，注重介绍新的技术和方法，以拓展学生的知识面，激发他们学习的积极性和创新意识。

2. 注重理论性与应用性的结合。良好的理论素养是应用的前提，而掌握理论的目的就是为了更好的应用。在教材的编写过程中，我们注意理论的系统性，在讲深讲透主要知识的基础上，融理论性和应用性于一体，注意基本方法的讲授，以培养学生应用理论和技术的能力。

3. 注重时代性和实用性的结合。力求精简旧的知识点，增加新的知识点，体现教材的时代特征。而且充分考虑一般高校目前所拥有的师资条件和教学设备，注重教材的实用性。

4. 注重科学性与通俗性的结合。概念、原理、新技术的阐述力求准确、精练；写作上尽量通俗易懂、深入浅出、图文并茂，增强可读性，便于学生自学。

5. 网络技术辅助教学。针对本系列教材我们开发有专门的网站（http://www.hzpress.org）、课件发布演示系统和考试系统等，以便为任课老师的教学提供更便捷、更全面的服务，并将通过网站开展各种形式的教材网上专家答疑、内容修订发布、课件定期升级等活动，以与读者随时互动，为读者提供立体化的服务。

教学改革是需要不断探索的课题。要达到以上目标，还需要不断地努力实践和完善。欢迎使用这套教材的教师、学生和其他读者提出宝贵意见。

最后，对参加这套教材编写的所有作者，对为这套教材的编写提供支持的有关学校、院系的领导和老师表示诚挚的谢意！感谢华中科技大学出版社为本系列教材的出版所付出的辛勤劳动！

教材编委会主任　何炎祥

（教授、博导、武汉大学计算机学院院长）

前　言

数据结构是计算机科学与技术专业教学计划中的一门核心课程，同时也是信息计算、电子信息技术等其他非计算机专业的一门重要专业基础课程。计算机科学与技术及其他相关学科都会用到各种数据结构，数据结构已经成为计算机科学与技术工作者，尤其是计算机应用领域的开发人员的必备知识。

数据结构的任务是根据对象的数据特征和对象之间的关系，选择合适的数据组织方法、存储方法和相应的算法。这些方法有助于设计出周密、有效和风格良好的程序。

编者在总结多年来数据结构课程的教学经验的基础上，结合新时期大学生的学习特点编写了《数据结构导学——C 语言描述》，旨在通过对一些典型习题的剖析和具体实现环节来给学生一些解题示范和启发，帮助学生更好地学习和掌握课程内容，理解和掌握算法设计的方法和技巧，为整个专业学习打下一个良好的基础。

本书是《数据结构——C 语言描述》的配套教材，全书共分为两部分。第一部分习题解析，主要针对教科书中的习题，给出分析和解答。每一章提供的习题分为简述题和编程题，简述题复习巩固本章基本内容，编程题侧重基本方法和技术的能力训练。所给习题都在原教科书的基础上有一定的扩充，以期扩大学生的知识掌握面。第二部分上机实习指导，提供了 6 个方面的上机实习内容，每个实习都给出了 1～2 道上机实习题目，每个题目都有实习目的、主要采用的设计方法以及 C 语言实现的大部分算法，使学生了解并掌握如何运用数据结构知识去解决实际问题，培养他们设计较为复杂程序的能力。

在使用本书的过程中需要注意以下几点：第一，要与课堂教学同步，便于巩固消化所学的知识；第二，由于算法设计不是惟一的，本书一般只给出了 1～2 种算法实现，读者可在学习、理解、领会已给算法的基础上，自己动手设计出新的算法，尽量做到举一反三，触类旁通，只有这样，才可能运用课程内所学的典型知识去解决实际问题；第三，要遵循循序渐进的原则，本书的内容由浅入深、由特殊到一般，建议读者按次序阅读。

本书第一部分中的第 1、3、7、9 章和第二部分的实习 2、6 由李克清执笔，第一部分中的第 2 章和第二部分的实习 1 由张万山执笔，第一部分中的第 4 章由章英执笔，第一部分中的第 5 章和第二部分的实习 3 由崔洪芳执笔，第一部分中的第 6 章和第二部分的实习 4 由操保华执笔，第一部分中的第 8 章和第二部分的实习 5 由夏祥胜执笔。全书由李克清主编，负责统稿、审阅和定稿。

由于时间仓促以及作者水平有限，本书难免有不足之处，敬请广大读者批评指正。

编　者

2005 年 1 月

目　录

第1章 引 言

学习要求

- 理解数据结构的基本概念；
- 了解数据、数据元素和数据结构之间的联系；
- 理清数据的逻辑结构和物理(存储)结构之间的关系；
- 学会用抽象数据类型(ADT)来描述数据结构；
- 进一步认识逐步求精和分而治之等方法在计算机问题求解中的作用，并学会使用这些典型的结构化程序设计方法；
- 掌握算法的定义及其特点；
- 学会分析算法的时间复杂度和空间复杂度。

1.1 内容提要

一、知识点

1. 数据结构。数据结构是相互之间存在一种或多种特定关系(函数)的数据元素的集合。数据结构主要关心的是数据的逻辑表示与物理表示之间的联系，对计算机事务处理中经常遇到的数学模型(如线性表、栈、队列、树、图、排序和查找等)进行了比较深入的探讨，并给出了相应处理算法。

2. 数据结构的逻辑表示与物理表示。数据的逻辑表示是数据元素之间的逻辑关系，是脱离于实际存储结构的数据表示方法，而数据的物理表示，则是数据与数据之间的关系在物理存储器上的表示，与计算机体系结构直接相关。也可以这样认为，数据结构的逻辑表示是数据的外部表现形式，而物理表示则是其内部存储形式。良好的物理结构对算法的执行效率和实现起着决定性的作用。数据的逻辑结构可分为线性结构和非线性结构；数据的物理结构可分为顺序存储结构、链式存储结构、索引存储结构和散列存储结构。

3. ADT(Abstract Data Type)。ADT 即抽象数据类型，是指一个数学模型及定义在该模型上的一组操作(运算)。ADT 只考虑数据的逻辑结构，并不涉及其物理存储，因此在研究数据结构时，借助于 ADT 能尽快抓住问题的本质，便于分析问题和解决问题。而在具体

实现算法时，又要和数据的存储结构联系起来。所以说ADT是数据结构中的一种分析问题的工具，并不涉及算法的实现部分。

4. 结构化程序设计方法。结构化程序设计方法是在合理的时间内，用所引出的方法，组织人们的思想，书写一个正确的可读的程序，达到计算任务具有可理解的表达格式。结构化程序设计的三种基本结构是顺序、选择和循环，采用这三种基本结构可以构造出任何复杂的程序。逐步求精和分而治之是两种典型的结构化程序设计方法。另外，ADT描述的算法可以很容易地用结构化程序设计方法来实现。

5. 算法。算法是一个有穷规则的集合，其中的规则确定了一个解决某一特定类型问题的运算序列。算法具有五个特点：有穷性、确定性、输入、输出和能行性。算法的正确性判定是一个比较复杂的问题，对于简单的算法，可以采用穷举法或程序正确性证明等方法来确定算法的正确性；而对于复杂的算法，只能进行所谓的“白盒测试”或“黑盒测试”等方法来说明算法在给定的测试下没有错误发生，但不能说明算法是正确的。

6. 算法分析。评价一个算法的好坏主要是从算法的时间复杂度和空间复杂度两个方面来考察。算法的时间复杂度和空间复杂度是两个相互矛盾的性能指标，一个算法要么所占存储空间较小，要么运行时间较短，两者性能均好的算法往往难以找到。

二、重点

1. 数据结构及其表示。数据结构包括数据的逻辑结构和物理结构，数据呈现给用户是其逻辑结构，其物理存储对用户是透明的。为了有效地处理数据，数据结构必须给出数据集上操作的统一接口，而忽略其存储结构。

2. ADT。ADT是一种描述数据结构的常用工具，能很好地描述数据的逻辑结构和操作。在用ADT描述数据结构时，可以不考虑其存储结构。

3. 结构化程序设计方法。结构化程序设计方法是数据结构中一种典型的程序设计方法。结构化程序设计方法将问题划分成一个个的相互独立又相互联系的子问题，解决子问题相对容易，而且子问题的正确性也易于保证。

4. 算法。算法是实现结构化程序设计方法的基础，一个子问题可以用一个算法来实现，将各个子问题的算法有机地结合起来就构成了整个问题的实现算法。

三、难点

1. 数据结构的逻辑结构和物理结构之间的关系。
2. ADT与结构化程序设计方法。
3. 算法及其分析。

1.2　典型例题分析

一、简述题

例 1　设有一数据的逻辑结构为：B=(D, S)，其中：

D={d1, d2, …, d9}

S={<d1,d3>, <d1, d8>, <d2, d3>, <d2, d4>, <d2, d5>, <d3, d9>, <d4, d7>, <d4, d6>, <d5, d6>, <d8, d9>, <d9, d7> }

画出这个逻辑结构示意图。并确定相对于关系 S，哪些结点是开始结点，哪些结点是终端结点？

分析：所谓逻辑结构示意图就是要求把上述数据结构的逻辑结构用图示的形式表述出来。在表示时，每一个数据元素用一个结点表示，它们之间的关系用一条带有方向的连线关联起来。画出关系图后，再来回答开始结点和终端结点就一目了然了。

解答：图 1.1 表示了数据结构 B 的逻辑关系。从图中不难看出，相对于关系 S 来说，结点 d1 和结点 d2 是开始结点，而结点 d6 和结点 d7 是终端结点。

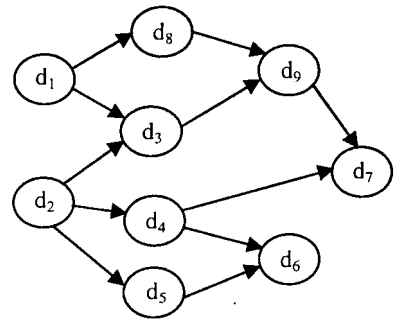

图1.1　数据结构B的逻辑关系图

例 2　何谓算法是正确的？算法和程序的异同点是什么？

分析：算法和程序是两个不同的概念，本题主要是要求读者能够对两者进行区分。

解答：算法是一个有穷规则的集合，其中的规则确定了一个解决某一特定类型问题的运算序列。算法具有有穷性、确定性、输入、输出和能行性等五个基本特点。算法的正确性是指对于一切合法的输入数据都能产生满足要求的结果。由于算法正确性证明较为困难，一般情况下，如果一个算法对于所有精心挑选出来的典型、苛刻甚至是十分特殊的测试输入数据均能够得出满足要求的结果的话，那么就认为该算法是正确的。

算法和程序都具有确定性、输入、输出和能行性等特征，它们是一对相互联系又相互区别的概念，其主要区别在于：

① 算法要求是有穷的，而程序没有这种要求。一个程序可以一直处于运行状态，如操作系统程序。

② 算法的描述语言可以是任意的，只要阅读者(这里指人)能够理解就可以了，没有特别严格的语法方面的限制；而程序必须按照严格的语法规则书写，才能为其阅读者(这里指计算机)所理解并执行。程序的描述语言也可以是任意一种计算机语言，一旦选择了某种语言就要遵循其语法规则，不能随意改动。

③ 算法一般可以转化为程序中的一个或几个函数，也就是说，算法处理的对象是最基

本问题的计算机求解方法，而程序则是这些求解方法的有机结合体。

例 3　计算下列算法 Bubble 和 BubbleSort 的时间复杂度。

```
void Bubble (int a[ ], int n)
{
    for( int i=0; i<n-1; i++)
     if(a[i] > a[i +1]) Swap(a[i], a[i+1] );
}
void BubbleSort (int a[ ], int n)
{
    for( int i=n; i>1; i- -)
    Bubble(a ,i) ;
}
```

分析：计算算法的时间复杂度，可以转化为先计算算法中每一个语句在最坏情况下的执行次数，然后再将这些次数相加，最后对语句执行总次数取极限即可。

解答：先给算法中的每个语句加上语句序号，特别要标明循环语句中的执行语句。

```
(01)  void Bubble (int a[ ], int n)
(02)  {
(03)        for( int i=0; i<n-1; i++)
(04)          if(a[i] > a[i +1]) Swap(a[i], a[i+1] );
(05)  }
(06)  void BubbleSort (int a[ ], int n)
(07)  {
(08)        for( int i=n; i>1; i- -)
(09)          Bubble(a ,i) ;
(10)  }
```

可执行语句为语句标号 03，04，08，09 所标定的语句，它们执行的次数分别为：n，n-1，n，n-1，因此算法 Bubble 的执行次数为 $T_1(n)=n+n-1=2n-1=O(n)$，由于算法 BubbleSort 中的第 09 句每次执行的次数为算法 Bubble 的执行时间次数，因此算法 BubbleSort 的执行次数为 $T_2(n)=n+(n-1)(2n-1)=2n^2-2n+1=O(n^2)$。

所以算法 Bubble 和算法 BubbleSort 的时间复杂度分别为 $O(n)$，$O(n^2)$。

例 4　一个算法的执行时间约为 1000n，另一个算法的执行时间约为 2^n。这两个算法的时间复杂度分别是多少？哪个更高？当问题的规模 n<13 时，选用哪个算法合适？

分析：对于算法的时间复杂度不能一概而论，有的算法在小规模的问题上具有很高的执行效率，而当问题规模急剧增大时就不再适应了，而有的算法正好相反，只适合于大规模的问题求解。因此在解决实际问题时，应该针对不同的问题规模选用不同时间复杂度的算法。

解答：由于 $\lim\limits_{n\to\infty}\frac{1000n}{n}=1000$(常数)，所以第一个算法的时间复杂度为 $O(n)$，而第二个算法的时间复杂度为 $O(2^n)$，因此第二个算法的时间复杂度要高于第一个算法的时间复杂度。

对于给定的两个函数 1000n 和 2^n 来说，它们在区间[0，15]中的函数图像如图 1.2 所示。

当 n=13.74681 时，两个函数相交，从图中不难看出，当 n<13.74681 时，1000n 比 2^n 都要大，因此在 n<13 时，应该选择第二个算法。

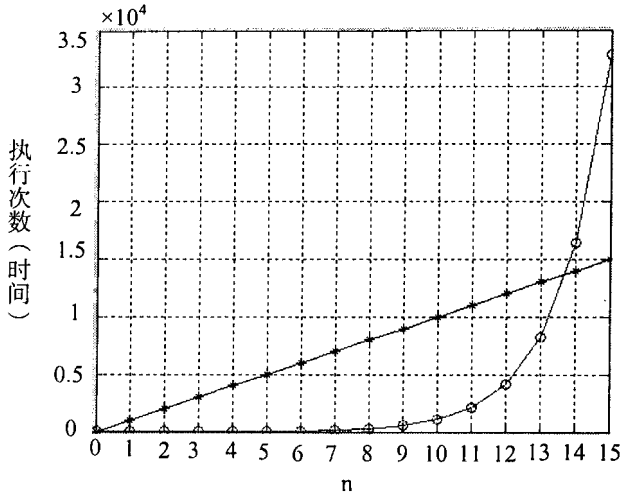

图1.2　两个函数执行时间的对比图

二、综合题

例 1　有一个含有 100 个元素的数组，每个元素的值都是实数，请用逐步求精方法写出求最大元素的值及下标的算法。

分析：按照逐步求精方法中的步骤，一步步地实现问题的求解。先是写出“做什么”，在 100 个数中，挑出最大元素；然后是“如何做”，对前面讨论的问题进一步细化，直到能写出实现的语句为止。

解答：假定讨论的实数数组具有最简单的数据结构，如

```
float a[100];
```

第一步，根据题设提出的问题，算法的大致框架结构描述如下。

```
int loc = 0;      /* 最大元素下标 */
float x = 0.0f;   /* 最大元素值 */
x = “数组 a 中的最大元素”;
loc = “数组 a 中的最大元素的下标”;
```

第二步，对语句“数组 a 中的最大元素”和“数组 a 中的最大元素的下标”进一步细化，由于最大元素和最大元素的下标是同步的，因此可以将这两个问题合并。在数组 a 中寻找最大值，引入一个循环语句来实现。

```
int k;
for( k = 0; k < 100; k++ ) {
    x = “数组 a 中的当前最大元素”;
    loc = “数组 a 中的当前最大元素的下标”;
}
```

第三步，对语句“数组 a 中的当前最大元素”和语句“数组 a 中的当前最大元素的下标”进一步细化。可以把数组划分成两部分：(已排查过的区域)当前元素(未排查的区域)，已排查过的区域中的最大元素记载在变量 x 中，只要比较 x 和当前元素就可以得到当前的最大元素。因此可以细化为：

```
if( x < a[k] ) a = a[k];
if( x < a[k] ) loc = k;
```

为了便于阅读和优化程序，不妨将上述语句改写为：

```
if( a[loc] < a[k] ) loc = k;
```

整理上面的所有步骤，可以得到一个完整的求数组中最大元素及其下标的算法。

```
int k, loc = 0;   /* loc 为最大元素下标 */
float x = 0.0f;   /* 最大元素值 */
for( k = 0; k < 100; k++ )
    if( a[loc] < a[k] ) loc = k;
x = a[loc];
```

例 2 一个老板有一袋金块。每个月将有两名雇员会因其优异的表现分别被奖励一个金块。按规矩，排名第一的雇员将得到袋中最重的金块，排名第二的雇员将得到袋中最轻的金块。根据这种方式，除非有新的金块加入袋中，否则第一名雇员所得到的金块总是比第二名雇员所得到的金块重。如果有新的金块周期性的加入袋中，则每个月都必须找出最轻和最重的金块。假设有一架天平，请你设计一个算法用最少的比较次数找出最轻和最重的金块。

分析：在一袋金块中挑出最重和最轻的金块，实际上就是一个比较过程，通过不断地比较找出最重的和最轻的。可以采用排序的方法来实现，但由于排序过于耗时，在此不予考虑，故可以采取分而治之的方法来求得最重和最轻的金块。

解答：当n很小时，比如说，n≤2，识别出最重和最轻的金块，一次比较就足够了。当n较大时(n > 2)，第一步，把这袋金块平分成两个小袋A和B，不能平分的话，让A袋中比B袋多一个金块；第二步，分别找出在A和B中最重和最轻的金块。设A中最重和最轻的金块分别为HA与LA，以此类推，B中最重和最轻的金块分别为HB和LB；第三步，通过比较HA和HB，可以找到所有金块中最重的，通过比较LA和LB，可以找到所有金块中最轻的。在第二步中，若A和B中金块的个数超过2，则递归地应用分而治之方法。

假设n=10。这个袋子被平分为各有5个金块的两个袋子A1和A2。为了在A1中找出最重和最轻的金块，必须将A1中的5个金块分成两组A11和A12。A11中有3个金块，A12中有两个金块，根据约定，还要将A11再次划分为两组A111和A112，其中A111有2个金块，A112只有一个金块。可以用一次比较在A111中找出较重的金块HA111和较轻的金块LA111。由于A112中只有一个金块，所以HA112和LA112相同，即不用比较。回溯一步，比较HA111和HA112得到HA11。再次回溯一步，比较HA11和HA12得到HA1。最后回溯一步，比较HA1和HA2得到HA。按照相同的顺序，可以得到LA。整个比较的过程如图1.3所示。

从图1.3不难看出，求得HA需要9次比较，取得LA也需要9次比较。但如果将它们联合起来的话，由于叶子结点仅需比较一次即可确定轻重，所以比较的次数要少于两者之和。对于上面的例子，只要14次比较即可。