“十二五”普通高等教育本科国家级规划教材
普通高等学校计算机教育“十二五”规划教材

数据结构（C 语言版）

（第 3 版）

DATA STRUCTURE
(C PROGRAMMING LANGUAGE VERSION)
(3rd edition)

李云清 杨庆红 揭安全 ◆ 编著

人民邮电出版社
POSTS & TELECOM PRESS

“十二五”普通高等教育本科国家级规划教材
普通高等学校计算机教育“十二五”规划教材

数据结构（C 语言版）

（第 3 版）

DATA STRUCTURE
(C PROGRAMMING LANGUAGE VERSION)
(3rd edition)

李云清 杨庆红 揭安全 ◆ 编著

人 民 邮 电 出 版 社
北 京

图书在版编目（CIP）数据

数据结构 : C语言版 / 李云清, 杨庆红, 揭安全编著. -- 3版. -- 北京 : 人民邮电出版社, 2014.9
普通高等学校计算机教育“十二五”规划教材
ISBN 978-7-115-36463-0

Ⅰ. ①数… Ⅱ. ①李… ②杨… ③揭… 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. ①TP311.12②TP312

中国版本图书馆CIP数据核字(2014)第179899号

内 容 提 要

本书介绍了数据结构的基本概念和基本算法。全书共分为 10 章，包括概论、线性表及其顺序存储、线性表的链式存储、字符串、数组和特殊矩阵、递归、树型结构、二叉树、图、检索、内排序等内容，附录给出了较为详细的基础实验和几类综合课程设计题。

本书内容丰富，逻辑性强，文字清晰流畅，既注重理论知识，又强调工程实用。书中既体现了抽象数据类型的概念，又对每个算法的具体实现给出了完整的C语言源代码描述。

与本书配套的电子教案、书中所有算法的源代码和习题参考答案均可从人民邮电出版社教学服务与资源网（www.ptpedu.com.cn）上免费下载。

本书可作为高等院校计算机专业及相关专业本科生“数据结构”课程的教材，也可作为从事计算机工程与应用的广大读者的参考书。

◆ 编　　著　李云清　杨庆红　揭安全
责任编辑　邹文波
责任印制　彭志环　焦志炜
◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路 11 号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
三河市中晟雅豪印务有限公司印刷
◆ 开本：787×1092　1/16
印张：19　　2014 年 9 月第 3 版
字数：499 千字　　2014 年 9 月河北第 1 次印刷

定价：42.00 元

读者服务热线：(010)81055256　印装质量热线：(010)81055316
反盗版热线：(010)81055315

前　言

数据结构是计算机专业重要的专业基础课程与核心课程之一。全国硕士研究生入学统一考试计算机科学与技术学科全国联考的考试科目定名为计算机学科专业基础综合，该考试科目涵盖 4 门课程，数据结构为其中重要的 1 门，其所占比例和分值最高。

数据结构课程主要是学习基本数据结构及其应用、检索和排序算法的各种实现方法与分析比较，对于选用何种描述工具对数据结构和算法进行描述，我们认为各有千秋。通过教学实践和教学改革研究表明，在数据结构教学中，对数据结构本质和各种算法思想的掌握与理解并不依赖于描述工具。实践表明，对于熟练掌握 C 语言的读者，使用 C 语言描述算法，读者可以将学习精力集中于算法思想的理解，有利于数据结构的教学。

为适应我国计算机科学技术的应用和发展，进一步提高计算机专业和相关专业“数据结构”课程的教学质量，2004 年，编者根据多年教学的体会，结合高等教育大众化的趋势，在分析国内、外多种同类教材的基础上，编写出版了《数据结构（C 语言版）》，2009 年修订出版了《数据结构（C 语言版）（第 2 版）》。该书先后印刷多次，受到读者的关注和欢迎。

本书修订了第 2 版中的一些表述，尽量结合实际应用描述相关知识，希望提升学生的学习兴趣。另外，根据一些使用本教材教师的建议和教学实际，增加了 2 个附录，分别是基础实验和综合实验。

本书总结了编者主持“数据结构”省级精品课程和省级资源共享课程建设的最新成果，传承了第 2 版教材的特色。修订后，本书具有以下特色。

1. 对数据结构的基本概念、基本理论的阐述注重科学性和严谨性。全书贯穿抽象数据类型的思想，在讨论一种数据结构时，都采用抽象数据类型的观点进行描述，以便读者更好地掌握理论知识，提高抽象思维的能力。

2. 对各种基本算法描述尽量详细，叙述清楚。对许多算法给出了详尽的图示，帮助读者理解算法的思路。

3. 注重工程实用。对于数据结构和算法的描述，本书并没有使用伪代码，而是采用了大多数读者熟悉的 C 语言进行描述，每个算法都用 C 语言的函数形式给出。书中所有的算法实现都对应一个 C 语言的函数，一个数据结构的基本函数实现置于一个 C 文件中，如同使用 C 语言库函数一般，使用十分方便。限于篇幅，书中没有给出使用实例。在我们提供的源代码中，有许多测试程序，读者可以通过参阅这些测试程序掌握书中算法的使用方法。

4. 由于递归在算法设计中经常使用，我们单独用一章的篇幅介绍了递归算法的特点、递归的执行过程以及递归和非递归的转换等。

5. 附录 1 给出了较为详细的基础实验指导，附录 2 给出了几类综合课程设计题供教学参考，方便教师开展课程实验教学，帮助学生掌握数据结构知识，提高

算法设计和实践能力。完整的基础实验案例可从人民邮电出版社网站下载。

为了方便教师使用，编者制作了与本书配套的电子教案和习题参考答案等教学资源，教师在使用时可以根据需要对电子教案进行修改。该电子教案和书中所有算法的源代码等均可从人民邮电出版社教学与服务资源站（www.ptpedu.com.cn）上免费下载。

本书包含正文和附录。正文共 10 章，第 1 章、第 2 章、第 3 章和第 10 章由李云清撰写，第 4 章、第 5 章、第 6 章和第 7 章由杨庆红撰写，第 8 章、第 9 章由揭安全撰写。附录 1、附录 2 由揭安全执笔。全书由李云清统稿。

在本书写作过程中，得到了许多老师的指导和帮助。第 2 版出版后，一些老师和同学也提出了许多富有建设性的意见和建议，在此，笔者向他（她）们表示衷心的感谢。

本书可以作为普通高等院校计算机及相关专业本科、专升本教材，也可作为研究生入学考试的复习参考书。

由于编者水平有限，加上时间仓促，书中难免有错误之处，恳请同行专家及广大读者批评指正。编者的电子邮箱是 dscourse2009@126.com。

编　者

2014.7

目 录

第 1 章 概论

数据结构讨论的是数据的逻辑结构、存储方式以及相关操作。本章讲述数据结构的基本概念及相关术语，介绍数据结构、数据类型和抽象数据类型之间的联系，介绍算法的特点及算法的时间复杂度与空间复杂度。

1.1 数据结构的基本概念与术语

1.1.1 数据结构的基本概念

人们常把计算机称为数据处理机，在计算机问世的初期，计算机所处理的数据基本上都是数值型数据，也就是说，计算机发展的初期主要是用于数值计算，那时的软件设计者将主要精力用于程序设计的技巧上，而对如何在计算机中组织数据并不需要花费太多的时间和精力。然而，随着计算机软、硬件的发展，计算机的应用范围在不断扩大，计算机处理数据的数量也在不断扩大，计算机处理的数据已不再是单纯的数值数据，而更多的是非数值数据。此时，如果仅在程序设计技巧上花功夫，而不去考虑数据的组织，那么，对大量数据的处理将会是十分低效的，有时甚至是无法进行的。

需要处理的数据并不是杂乱无章的，它们一定有内在的联系，只有弄清楚它们之间本质的联系，才能使用计算机对大量的数据进行有效的处理。

例如，某电信公司的市话用户信息表如表 1.1 所示。

表 1.1　　用户信息表

序　号	用 户 名	电 话 号 码	用 户 住 址	
			街 道 名	门 牌 号
00001	万方林	33800***	北京西路	1659*
00002	吴金平	33800***	北京西路	2099*
00003	王　冬	55700***	瑶湖大道	1987*
00004	王　三	55700***	瑶湖大道	2008*
00005	江　凡	68800***	学府大道	5035*

对于上面的数据，每一行是一个用户的有关信息，它由序号、用户名、电话号码和用户住址等项组成。序号、用户名和电话号码等项称为基本项，是有独立意义的最小标识单位，而用户住

址称为组合项，组合项由一个或多个基本项或组合项组成，是有独立意义的标识单位。这里的每一行称为一个结点，每一个组合项称为一个字段。结点是由若干个字段构成的。对于能唯一地标识一个结点的字段或几个字段的组合，如这里的序号字段，称为关键码。当要使用计算机处理用户信息表中的数据时，必须弄清楚下面 3 个问题。

1. 数据的逻辑结构

这些数据之间存在什么样的内在联系？在这些数据中，有且只有一个结点是表首结点，它前面没有其他结点，后面有一个和它相邻的结点；有且只有一个结点是表尾结点，它后面没有其他结点，前面有一个和它相邻的结点；除这两个结点之外，表中所有其他的结点都有且仅有一个和它相邻的位于它之前的结点，也有且仅有一个和它相邻的位于它之后的结点。上述这些就是用户信息表的逻辑结构。

2. 数据的存储结构

数据在计算机中的存储方式称为存储结构。将用户信息表中的所有结点存入计算机时，就必须考虑存储结构。使用 C 语言进行设计时，常见的方式是用一个结构数组来存储整个用户信息表，每一个结构数组元素是一个结构，它对应于用户信息表中的一个结点。用户信息表中相邻的结点，对应的结构数组元素也是相邻的，或者说在这种存储方式下，逻辑相邻的结点就必须物理相邻。这是一种被称为顺序存储的方式，当然，还有其他的存储方式。

3. 数据的运算集合

对数据的处理必定涉及到相关的运算。在上述用户信息表中，可以进行删除一个用户、增加一个用户和查找某个用户等操作。应该明确指出这些操作的含义。比如删除操作，是删除序号为 5 的用户还是删除用户名为王三的用户是应该明确定义的，如果需要可以定义两个不同的删除操作。为一批数据定义的所有运算（或称操作）构成一个运算（操作）集合。

对于一批待处理的数据，只有分析清楚上面 3 个方面的问题，才能进行有效的处理。

数据结构就是指按一定的逻辑结构组成的一批数据，使用某种存储结构将这批数据存储于计算机中，并在这些数据上定义了一个运算集合。

在讨论一个数据结构时，数据结构所含的 3 个方面缺一不可，也就是说，只有给定一批数据的逻辑结构和它们在计算机中的存储结构，并且定义了数据运算集合，才能确定一个数据结构。例如，在后面的章节中将要介绍的栈和队列，它们的逻辑结构是一样的，它们都可以用同样的存储结构，但是由于所定义的运算性质不同，它们成为两种不同的数据结构。

1.1.2 数据的逻辑结构

数据的逻辑结构是数据和数据之间所存在的逻辑关系，它可以用一个二元组

$$B = (K, R)$$

来表示，其中 K 是数据，即结点的有限集合；R 是集合 K 上关系的有限集合，这里的关系是从集合 K 到集合 K 的关系。在本书的讨论中，一般只涉及一个关系的逻辑结构。

例如，有 5 个人，分别记为 a，b，c，d，e，其中 a 是 b 的父亲，b 是 c 的父亲，c 是 d 的父亲，d 是 e 的父亲，如果只讨论他们之间存在的父子关系，则可以用下面的二元组形式化地予以表达：

$$B = (K, R)$$

其中，$K = \{a, b, c, d, e\}$

$R = \{r\}$

$r=\{<a, b>, <b, c>, <c, d>, <d, e>\}$

也可以用图形的方式表示数据的逻辑结构，K 中的每个结点 k_i 用一个方框表示，而结点之间的关系用带箭头的线段表示。这 5 人之间的逻辑结构用图形的方式表达，如图 1.1 所示。

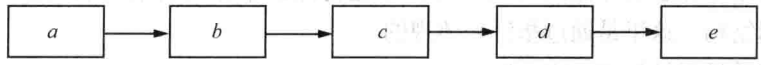


图 1.1　数据的逻辑结构图

若 $k_i \in K$，$k_j \in K$，$<k_i, k_j> \in r$，则称 k_i 是 k_j 的相对于关系 r 的前驱结点，k_j 是 k_i 的相对于关系 r 的后继结点。因为一般只讨论具有一种关系的逻辑结构，即 $R=\{r\}$，所以简称 k_i 是 k_j 前驱，k_j 是 k_i 的后继。如果某个结点没有前驱结点，称为开始结点；如果某个结点没有后继结点，称为终端结点；既不是开始结点也不是终端结点的结点称为内部结点。

对于一个逻辑结构 $B=(K, R)$，如果它只有一个开始结点和一个终端结点，而其他的每一个结点有且仅有一个前驱和一个后继，称为线性结构。如果它有一个开始结点，有多个终端结点，除开始结点外，每一个结点有且仅有一个前驱，称为树型结构。如果每个结点都可以有多个前驱和后继，称为图形结构。树型结构和图形结构都是非线性结构。

图 1.2 所示的是一个有 7 个结点的数据结构的逻辑关系的图形表示。

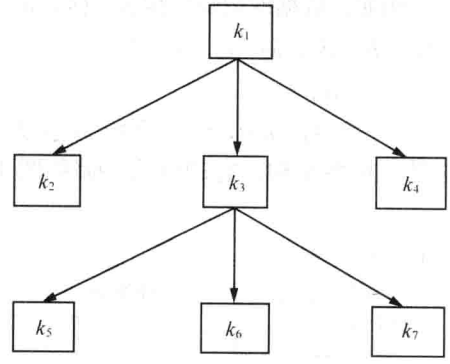


图 1.2　一个逻辑关系的图形表示

这里 k_1 是开始结点，k_2，k_4，k_5，k_6，k_7 是终端结点，k_3 是内部结点。

以后，在不引起误解的情况下，我们把数据的逻辑结构简称为数据结构。

1.1.3　数据的存储结构

数据的逻辑结构是独立于计算机的，它与数据在计算机中的存储无关。要对数据进行处理，就必须将数据存储在计算机中，如果将数据在计算机中无规律地存储，那么在处理时会非常糟的，是没有用的。试想一下，如果一本英汉字典中的单词是随意编排的，那这本字典谁也不会用。对于一个数据结构 $B=(K, R)$，必须建立从结点集合到计算机某个存储区域 M 的一个映像，这个映像要直接或间接地表达结点之间的关系 R。如前所述，数据在计算机中的存储方式称为数据的存储结构。数据的存储结构主要有以下 4 种。

1. 顺序存储

顺序存储通常用于存储具有线性结构的数据。将逻辑上相邻的结点存储在连续存储区域 M 的相邻存储单元中，使得逻辑相邻的结点一定是物理位置相邻。这种映像是通过物理上存储单元的相邻关系来体现结点间相邻的逻辑关系。

例如，对于一个数据结构 $B=(K, R)$

其中，$K=\{k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9\}$

$R=\{r\}$

$r=\{<k_1, k_2>, <k_2, k_3>, <k_3, k_4>, <k_4, k_5>, <k_5, k_6>, <k_6, k_7>, <k_7, k_8>, <k_8, k_9>\}$

它的顺序存储方式如图 1.3 所示。

2. 链式存储

链式存储方式是给每个结点附加一个指针段，一个结点的指针所指的是该结点的后继存储地址，因为一个结点可能有多个后继，所以指针段可以是一个指针，也可以是多个指针。在链式存储中逻辑相邻的结点在连续存储区域 M 中可以不是物理相邻的。前面讲到，数据的存储结构一定要体现它的逻辑结构，这里是通过指针来体现的。

例如，数据的逻辑结构 $B=(K, R)$

其中，$K=\{k_1, k_2, k_3, k_4, k_5\}$

$R=\{r\}$

$r=\{<k_1, k_2>, <k_2, k_3>, <k_3, k_4>, <k_4, k_5>\}$

这是一个线性结构。它的链式存储如图 1.4 所示。

很明显，在这种存储方式下，必须要知道开始结点的存储地址。

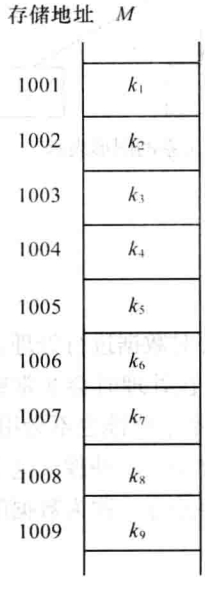
例如，数据的逻辑结构 $B=(K, R)$

其中，$K=\{k_1, k_2, k_3, k_4, k_5\}$

$R=\{r\}$

$r=\{<k_1, k_2>, <k_1, k_3>, <k_2, k_4>, <k_4, k_5>\}$

这是一树型结构，它的链式存储如图 1.5 所示。



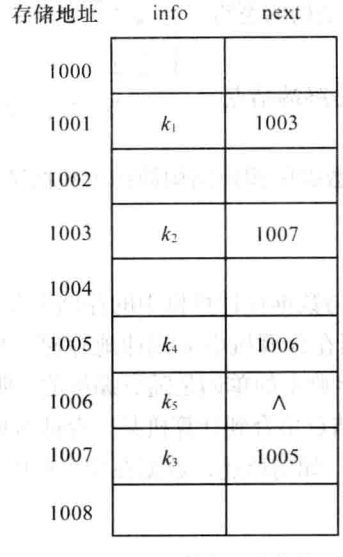


图 1.3 顺序存储的图示　图 1.4 一个线性结构的链式存储图示

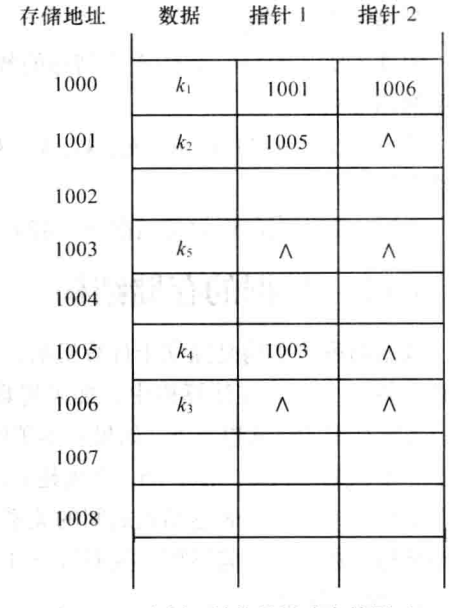


图 1.5 一个树型结构的链式存储图示

3. 索引存储

在线性结构中，设开始结点的索引号为 1，其他结点的索引号等于其前继结点的索引号加 1，则每一个结点都有唯一的索引号。索引存储就是根据结点的索引号确定该结点的存储地址。例如，一本书的目录就是各章节的索引，目录中每个章节后面标识的页码就是该章节在书中的位置。如果某本书的每个章节所占页码总数相同，那么可以由一个线性函数来确定每个章节在书中的位置。

4. 散列存储

散列存储的思想是构造一个从集合 K 到存储区域 M 的函数 h，该函数的定义域为 K，值域

为 M，K 中的每个结点 k_i 在计算机中的存储地址由 $h(k_i)$ 确定。

一个数据结构存储在计算机中，整个数据结构所占的存储空间一定不小于数据本身所占的存储空间，通常把数据本身所占存储空间的大小与整个数据结构所占存储空间的大小之比称为数据结构的存储密度。显然，数据结构的存储密度不大于 1。顺序存储的存储密度为 1，链式存储的存储密度小于 1。

1.1.4　数据的运算集合

对于一批数据，数据的运算是定义在数据的逻辑结构之上的，而运算的具体实现依赖于数据的存储结构。

例如，在一个线性结构中查找一个值为 x 的结点，可以这样定义查找运算："从该结构的第 1 个结点开始，将结点值与 x 比较，如果相等则查找成功结束；如果不相等，则沿着该结点的后继继续比较，直到找到一个满足条件的结点成功结束，或找遍所有结点都没有找到而以查找失败结束。"

如果一个线性结构采用顺序存储，比如用一个一维数组存放，则查找运算中一个结点的后继是通过数组下标的递增实现的，如 $i=i+1$。如果采用链式存储，一个结点后继是通过形如 p = p->link 的方式来实现的。

数据的运算集合要视情况而定，一般而言，数据的运算包括插入、删除、检索、输出和排序等。

插入是指在一个结构中增加一个新的结点。

删除是指在一个结构中删除一个结点。

检索是指在一个结构中查找满足条件的结点。

输出是指将一个结构中所有结点的值打印、输出。

排序是指将一个结构中所有结点按某种顺序重新排列。

1.2　数据类型和抽象数据类型

在程序设计中，数据和运算是两个不可缺少的因素。所有的程序设计活动都是围绕着数据和其相关运算进行的。从机器指令、汇编语言中的数据没有类型的概念，到现在的面向对象程序设计语言中抽象数据类型概念的出现，程序设计中的数据经历了一次次抽象。数据的抽象经历了三个发展阶段。

第一个发展阶段是从无类型的二进制数到基本数据类型的产生。在机器语言中，程序设计中所涉及的一切数据，包括字符、数、指针、结构数据和程序等都是由程序设计人员用二进制数字表达的，没有类型的概念。人们难以理解、辨别这些由 0 和 1 所组成的二进制数表达的含义。这种程序的易读性、可维护性和可靠性极差。随着计算机技术的发展，出现了 Fortran、Algol 高级程序设计语言，在这些高级程序设计语言中引入了整型、实型和布尔类型等基本数据类型，程序员可以将其他的数据对象建立在基本数据类型之上，避免了复杂的机器表示。这样，程序员不必和繁杂的二进制数字直接打交道就能完成相应的程序设计任务。数据类型就像一层外衣，它反映了一个抽象层次，使得程序设计人员只需知道如何使用整数、实数和布尔数，而不需要了解机器的内部细节。此外，高级程序设计语言的编译程序可以利用类型信息进行类型一致性检查，及早发现程序中的语法错误。

第二个发展阶段是从基本数据类型到用户自定义类型的产生。Fortran 等语言只是引入了

整型、实型和布尔型等基本的类型，而在程序的开发过程中，许多复杂的数据对象难以用这些基本的类型表示，这就给程序的设计带来了很大的困难。例如，程序要实现一个数组栈（即栈中的元素为数组）的操作，不能将数组作为基本对象来处理，必须通过其下标变量对数组的分量逐个处理。如果数组的分量又是一个数组，则还必须对这个用作分量的数组的分量逐一处理，直到最底层为整数、实数或布尔数等基本类型的数据为止。也就是说，虽然经过第一个阶段的发展，用户可以不必涉及机器内部细节，但是，仅仅引入几种基本类型，用户在处理复杂的数据时，仍要涉及其数据结构的细节。如果可以把所要处理的数据对象作为某种类型的对象来直接处理，而不必涉及其数据表示细节，将会给控制程序的复杂性带来很大的益处，并且编译程序的类型检查机制可以及早发现错误。为此，PL/I 曾试图引入更多的基本类型（如数组、树和栈等），以便将数组、树和栈作为直接处理的数据对象。但这不是解决问题的好方法。因为，一个大型系统所涉及的数据对象极其复杂，任何一种程序语言都没有办法使所有的类型作为其基本的类型。解决问题的根本方法是，程序设计语言必须提供这样一种机制，程序员可以依据具体问题，灵活方便地定义新的数据类型，即用户定义类型的机制。在大家熟知的 Pascal 和 C 语言中就引入了用户定义类型的机制，程序中允许有用户自己定义的新类型。

第三个发展阶段是从用户自定义类型到抽象数据类型的产生。抽象数据类型是用户自己定义类型的一个机制。数据和运算（即对数据的处理）是程序设计的核心，数据表示的复杂性决定了其上运算实现的复杂性，这也是整个系统复杂性的关键所在。20 世纪 60 年代末期出现的“软件危机”就是因为在软件开发中不能有效地控制数据表示，无法控制整个软件系统的复杂性，最终导致软件系统的失败。“软件危机”使人们认识到，在基于功能抽象的模块化设计方法中，模块间的连接是通过数据进行的，数据从一个模块传送到另一个模块，每一个模块在其上施加一定的操作，并完成一定的功能。尽管 Pascal 和 C 语言等的用户自定义类型机制使得用户可以将这些连接数据作为某种类型的对象直接处理，然而，由于这些用户自定义类型的表示细节是对外部公开的，没有任何保护措施，程序设计人员可以随意地修改这种类型对象的某些成分，添加一些不合法的操作，而处理这种数据对象的其他模块却一无所知，从而危害整个软件系统。这些不利因素将会对由多人合作进行的大型软件系统开发产生致命的危害。为了有效地控制大型程序系统的复杂性，必须从两个方面加以考虑。一方面是更新程序设计语言中的类型定义机制，使类型的内部表示细节对外界不可见，程序设计中不需要依赖于数据的某种具体表示；另一方面要寻求连接模块的新方法，尽可能缩小模块间的界面。面向对象程序设计语言 C++中的类就是实现抽象数据类型的机制。

1.2.1 数据类型

数据类型（或简称类型）反映了数据的取值范围以及对这类数据可以施加的运算。在程序设计语言中，一个变量的数据类型是指该变量所有可能的取值集合。例如，Pascal 语言中布尔类型的变量可以取值为 true 或 false，而不能取其他值。C 语言中的字符型的变量可以取值为 A 或 B 等单个字符，但不能取值 ABC。各种程序设计语言中所规定的基本数据类型不尽相同，利用基本数据类型构造组合数据类型的法则也不一样。因此，一个数据类型就是同一类数据的全体，数据类型用以说明一个数据在数据分类中的归属，是数据的一种属性，数据属性规定了该数据的变化范围。数据类型还包括了这一类数据可以参与的运算，例如，整数型的数据可以进行加、减、乘和除运算，而字符型的数据则不能进行这些运算。所以，数据类型包括两个方面，即数据属性和在这些数据上可以施加的运算集合。

数据结构是数据存在的形式，所有的数据都是按照数据结构进行分类的。简单数据类型对应于简单的数据结构；构造数据类型对应于复杂的数据结构。

1.2.2　抽象数据类型

抽象数据类型是数据类型的进一步抽象，是大家熟知的基本数据类型的延伸和发展。众所周知，整数类型是整数值数据模型（或简单理解为整数）和加、减、乘、除四则运算等的统一体，人们在程序设计中大量使用整数类型及其相关的四则运算，而没有人去追究这些运算是如何实现的。如果人们问到此问题，有人可能会简单回答为计算机自动进行处理。实际上，每一种基本数据类型都有一组与其相关的运算，而这组运算的具体实现都被封装起来，人们不知道也确实不必去关心这些细节。试想一下，如果程序设计人员在程序设计中要考虑基本数据类型的相关运算是如何具体实现的，那将是多么繁杂和令人头痛的事情，程序中的错误也会增加许多。将基本数据类型的概念进行延伸，程序设计人员可以在进行问题求解时，首先确定该问题所涉及的数据模型以及定义在该数据模型上的运算集合，然后求出从数据模型的初始状态到达目标状态所需的运算序列，就成为该问题的求解算法，这样做就是一种抽象。它使得用户不必去关心数据模型中的数据具体表示及相关运算的具体实现，大大提高软件开发中的开发效率和程序的正确性等。

抽象数据类型是与表示无关的数据类型，是一个数据模型及定义在该模型上的一组运算。定义一个抽象数据类型时，必须给出它的名字及各运算的运算符名，即函数名，并且规定这些函数的参数性质。一旦定义了一个抽象数据类型及具体实现，在程序设计中就可以像使用基本数据类型那样，十分方便地使用抽象数据类型。

1.2.3　抽象数据类型的描述和实现

抽象数据类型的描述包括给出抽象数据类型的名称、数据的集合以及数据之间的关系和操作的集合等方面的描述。抽象数据类型的设计者根据这些描述给出操作的具体实现，抽象数据类型的使用者依据这些描述使用抽象数据类型。

在描述抽象数据类型时，抽象数据类型的名称可以任意给定，但最好是用意义贴近的文字表示；对于数据的描述应该给出数据的集合以及这些数据之间关系的描述；对于操作方面的描述，要给出每个操作的名称、操作的前置条件和后置条件以及操作的功能等方面的描述。一个抽象数据类型的操作包括“构造操作集”和“非构造操作集”，读者可以简单地理解构造操作集是一个抽象数据类型中必须定义的操作集合，抽象数据类型中的数据元素均可以通过它们的组合操作获得，而非构造操作集是为了方便使用抽象数据类型而定义的一些操作的集合。例如，对于非负整数抽象数据类型，它的构造操作集有两个：一个是初始化操作 zero，一个是求后继的操作 succ。对任意一个非负整数都可以由这两个操作生成，如 2 = succ(succ(zero()))。当然，还要定义如相等、相加和相减等操作。一般而言，一个抽象数据类型中定义并实现的操作越多，使用就越方便，如同商业银行提供的服务项目越多就越受用户的欢迎一样。

抽象数据类型描述的一般形式如下：

```
ADT 抽象数据类型名称 {
    数据对象：
        ……
    数据关系：
```

```
        ……
    操作集合:
        操作名 1:
            ……
            ……
        操作名 n:
}ADT 抽象数据类型名称
```

抽象数据类型的具体实现依赖于程序设计语言。面向对象的程序设计语言中的“类”支持抽象数据类型，也支持信息隐藏。本书设定读者使用 C 语言进行程序设计，书中使用 C 语言进行算法的描述和实现，而 C 语言在对抽象数据类型的支持上是欠缺的。一个抽象数据类型 C 语言的不同实现，如一个使用顺序存储，一个使用链式存储，尽管在不同的实现中可以使同一操作对应的函数名相同，但是抽象数据类型的操作所对应的函数原型一般是不同的。因此，要用 C 语言完整地实现抽象数据类型是不现实的。尽管如此，本书在叙述时仍能努力贯彻抽象数据类型的思想，在给出一个抽象数据类型的描述时，对于不同的具体实现，为了和实现中的函数原型一致，在描述抽象数据类型中各操作的时候，有的地方采用了和实现有关的函数原型，并对函数的功能进行了描述。采用这种折衷的方法，是希望向只具有过程式程序设计经验的读者逐渐地灌输抽象数据类型的思想，同时也希望具有面向对象程序设计经验的读者也能从中获益。

1.3 算法和算法分析

1.3.1 算法的基本概念和基本特征

为了求解某问题，必须给出一系列的运算规则，这一系列的运算规则是有限的，表达了求解问题的方法和步骤，这就是一个算法。

一个算法可以用自然语言描述，也可以用高级程序设计语言或伪代码描述。本书采用 C 语言对算法进行描述。

算法是求解问题的一种方法或一个过程。更严格地讲，算法是由若干条指令组成的有穷序列，并满足以下 5 个特征。

① 有穷性　算法的执行必须在有限步内结束。

② 确定性　算法的每一个步骤必须是确定的、无二义性的。

③ 输入　算法可以有 0 个或多个输入。

④ 输出　算法一定有输出结果。

⑤ 可行性　算法中的运算都必须是可以实现的。

程序是用计算机语言表达的求解一个问题的一系列指令的序列，它和算法的主要区别是，算法具有有穷性，程序不需要具备有穷性。一般的程序都会在有限时间内终止，但有的程序却可以不在有限时间内终止，如一个操作系统在正常情况下是永远都不会终止的。

1.3.2 算法的时间复杂度和空间复杂度

求解一个问题可能有多个算法，如何评价算法的优劣呢？一个算法的优劣主要从算法执行时间和所需要占用的存储空间两个方面来衡量。算法执行时间的度量不是采用算法执行的绝对时