21世纪高等学校计算机规划教材

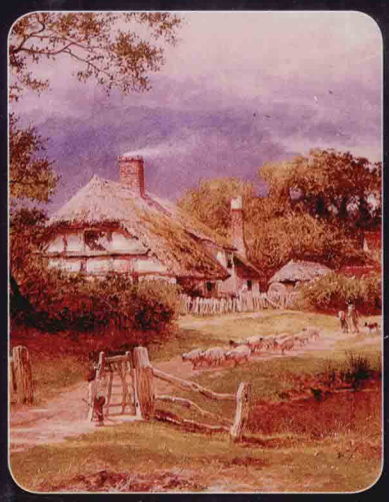
21st Century University Planned Textbooks of Computer Science

数据结构
（C语言版）

Data Structure

严蔚敏 李冬梅 吴伟民 编著

- 精选内容，贴近普通高校课程现状
- 顺应发展，符合最新考研大纲要求
- 类C描述，培养算法设计应用能力



名家系列

人民邮电出版社
POSTS & TELECOM PRESS

21世纪高等学校计算机规划教材

21st Century University Planned Textbooks of Computer Science

数据结构
（C语言版）

Data Structure

严蔚敏 李冬梅 吴伟民 编著

名家系列

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 / 严蔚敏, 李冬梅, 吴伟民编著. -- 北京 : 人民邮电出版社, 2011.2
21世纪高等学校计算机规划教材
ISBN 978-7-115-23490-2

Ⅰ. ①数… Ⅱ. ①严… ②李… ③吴… 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(2011)第006222号

内 容 提 要

本书在选材与编排上，贴近当前普通高等院校“数据结构”课程的现状和发展趋势，符合最新研究生考试大纲，内容难度适度，突出实用性和应用性。全书共8章，内容包括绪论，线性表，栈和队列，串、数组和广义表，树和二叉树，图，查找和排序。全书采用类C语言作为数据结构和算法的描述语言。

本书可作为普通高等院校计算机和信息技术相关专业“数据结构”课程的教材使用，也可供从事计算机工程与应用工作的科技工作者参考。

21世纪高等学校计算机规划教材

数据结构（C语言版）

◆ 编　　著　严蔚敏　李冬梅　吴伟民
责任编辑　蒋　亮

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街14号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京鑫正大印刷有限公司印刷

◆ 开本：787×1092　1/16
印张：15.25　　2011年2月第1版
字数：398千字　　2011年2月北京第1次印刷

ISBN 978-7-115-23490-2

定价：28.00元

读者服务热线：(010)67170985　印装质量热线：(010)67129223
反盗版热线：(010)67171154

前　言

“数据结构”是普通高等院校计算机和信息技术等相关专业的核心课程之一。随着当前高等教育的发展和企业对于人才需求的不断变化，对于“数据结构”课程的内容也提出了新的要求。一方面，“数据结构”课程是学生从传统的形象思维转向科学的抽象思维的第一门课程，需要学生在思维认识上有一个转变，这也使得“数据结构”成为一门公认的比较难学的课程，这对于教材的内容也提出了更高的要求。另一方面，随着高等教育的普及化，许多老师和同学都反映，原有的“数据结构”相关教材的难度太大，不利于学生学习兴趣的培养和学习能力的提升。因此，我们计划根据当前普通高等院校“数据结构”课程的现状和发展趋势，编写一本结构合理、难度适中、实用性强的“数据结构”教材。

本书在前期策划过程中，通过会议、问卷、走访等多种形式，广泛征求了“数据结构”课程教学一线教师的意见和建议。对于教材内容，我们根据计算机专业最新研究生考试大纲要求和目前教学及人才能力需求实际，在保留“数据结构”课程核心知识内容的基础上，删除了部分落后当前技术发展实际和与其他课程重复的内容；对于教材难度，我们对一些难度比较大的算法进行了修改，同时增加了与实际工作联系密切的实例和应用；对于教材配套，我们针对各章内容，参考硕士研究生考试题型，增加了大量的习题，便于学生进行课后练习。

本书共 8 章内容，其中第 1 章为绪论，综述数据、数据结构和抽象数据类型等基本概念；第 2 章至第 6 章从抽象数据类型的角度，分别讨论线性表、栈、队列、串、数组、广义表、树和二叉树以及图等基本类型的数据结构及其应用；第 7 章和第 8 章分别讨论查找和排序，除了介绍各种实现方法之外，还着重从时间上进行定性或定量的分析和比较。本书突出了抽象数据类型的概念。对每一种数据结构，都分别给出相应的抽象数据类型规范说明和实现方法。

全书中采用类 C 语言作为数据结构和算法的描述语言，在对数据的存储结构和算法进行描述时，尽量考虑 C 语言的特色，同时兼顾数据结构和算法的可读性。学生在实际上机操作时，可以很容易地将本书中的数据结构和算法转换成 C 或 C++程序。

本书可作为普通高等院校计算机和信息技术相关专业“数据结构”课程的教材使用，也可供从事计算机工程与应用工作的科技工作者参考。

因编者水平有限，书中错误在所难免，恳请批评指正。

编者

2011 年 1 月

目 录

第 1 章 绪论

早期的计算机主要用于数值计算，现在，计算机主要用于非数值计算，包括处理字符、表格和图像等具有一定结构的数据。这些数据内容存在着某种关系，只有明确数据的内在关系，合理地组织数据，才能对它们进行有效的处理，设计出高效的算法。如何合理地组织数据、高效地处理数据，这就是“数据结构”主要研究的问题。本章简要介绍有关数据结构的基本概念和算法分析方法。

1.1 数据结构的研究内容

计算机主要用于数值计算时，一般要经过如下几个步骤：首先从具体问题抽象出数学模型，然后设计一个解此数学模型的算法，最后编写程序，进行测试、调试，直到解决问题。在此过程中寻求数学模型的实质是分析问题，从中提取操作的对象，并找出这些操作对象之间的关系，然后用数学语言加以描述，即建立相应的数学方程。例如，用计算机进行全球天气预报时，就需要求解一组球面坐标系下的二阶椭圆偏微分方程；预测人口增长情况的数学模型为常微分方程。求解这些数学方程的算法是计算数学研究的范畴，如高斯消元法、差分法、有限元法等算法。数据结构主要研究非数值计算问题，非数值计算问题无法用数学方程建立数学模型，下面通过三个实例加以说明。

【例 1.1】 学生学籍管理系统。

高等院校教务处使用计算机对全校的学生情况作统一管理。要了解学生的基本信息，包括学生的学号、姓名、性别、籍贯、专业等，如表 1.1 所示。每个学生的基本情况按照不同的顺序号，依次存放在“学生基本信息表”中，根据需要对这张表进行查找。每个学生的基本信息记录按顺序号排列，形成了学生基本信息记录的线性序列，呈一种线性关系。

表 1.1　　学生基本信息表

学　号	姓　名	性　别	籍　贯	专　业
060214201	杨阳	男	安徽	计算机科学与技术
060214202	薛林	男	福建	计算机科学与技术
060214215	王诗萌	女	吉林	计算机科学与技术
060214216	冯子晗	女	山东	计算机科学与技术

诸如此类的线性表结构还有图书馆的书目管理系统、库存管理系统等。在这类问题中，计算机处理的对象是各种表，元素之间存在简单一对一的线性关系，因此这类问题的数学模型就是各种线性表，施加于对象上的操作有查找、插入和删除等。这类数学模型称为“线性”的数据结构。

【例 1.2】 人机对弈问题。

计算机之所以能和人对弈是因为已经将对弈的策略在计算机中存储好。由于对弈的过程是在一定规则下随机进行的，所以，为使计算机能灵活对弈，就必须对把对弈过程中所有可能发生的情况及相应的对策都加以考虑。以最简单的井字棋为例，初始状态是一个空的棋盘格局。对弈开始后，每下一步棋，则构成一个新的棋盘格局，且相对于上一个棋盘格局的可能选择可以有多种形式，因而整个对弈过程就如同图 1.1 所示的“一棵倒长的树”。在这棵“树”中，从初始状态（根）到某一最终格局（叶子）的一条路径，就是一次具体的对弈过程。

诸如此类的树结构还有计算机的文件系统、一个单位的组织机构等。在这类问题中，计算机处理的对象是树结构，元素之间是一种一对多的层次关系，施加于对象上的操作有查找、插入和删除等。人机对弈问题的数学模型就是如何用树结构表示棋盘和棋子等，算法是博弈的规则和策略。这类数学模型称为“树”的数据结构。

【例 1.3】 最短路径问题。

从城市 A 到城市 B 有多条线路，但每条线路的交通费不同，那么，如何选择一条线路，使城市 A 到城市 B 的交通费用最少。解决的方法是，可以把这类问题抽象为图的最短路径问题。如图 1.2 所示，图中的顶点代表城市，有向边代表两个城市之间的通路，边上的权值代

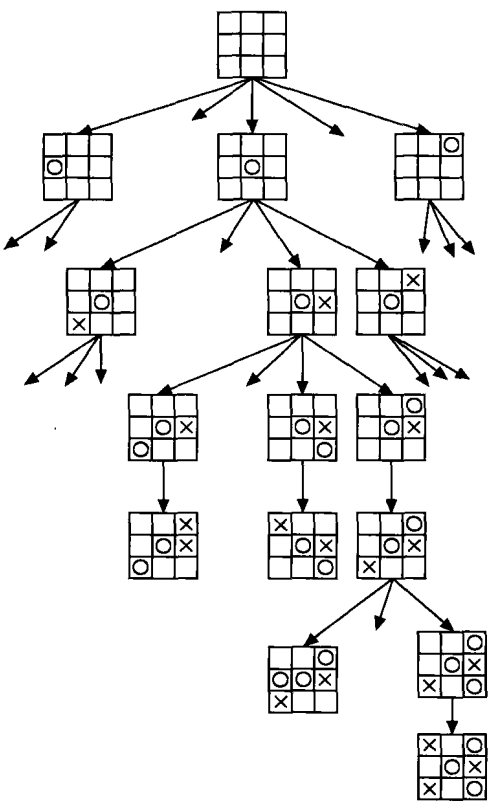


图 1.1 井字棋的对弈树

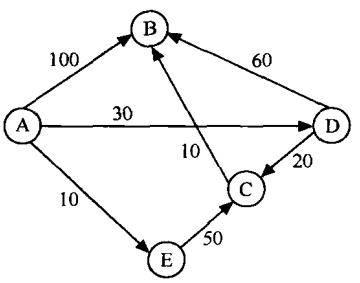


图 1.2 最短路径问题

表两个城市之间的交通费。求解 A 到 B 的最少交通费用，就是要在有向图中 A 点（源点）到达 B 点（终点）的多条路径中，寻找一条各边权值之和最小的路径，即最短路径。

诸如此类的图结构还有网络工程图和网络通信图等，这类数学模型称为“图”的数据结构。在这类问题中，元素之间是多对多的网状关系，施加于对象上的操作依然有查找、插入和删除等。最短路径问题的数学模型就是图结构，算法是求解两点之间的最短路径。

从上面三个实例可以看出，非数值计算问题的数学模型不再是数学方程，而是诸如线性表、树和图的数据结构。因此，简单地说，数据结构是一门研究非数值计算程序设计中的操作对象，以及这些对象之间的关系和操作的学科。

“数据结构”作为一门独立的课程在国外是从 1968 年才开始设立的，同年，著名计算机科学家 D.E.Knuth 教授发表了《计算机程序设计艺术》第一卷《基本算法》，这是第一本较系统地阐述“数据结构”基本内容的著作。之后，随着大型程序和大规模文件系统的出现，结构化程序设计成为程序设计方法学的主要研究方向，人们普遍认为程序设计的实质就是对所处理的问题选择一种好的数据结构，并在此结构基础上施加一种好的算法，著名科学家 N.Wirth 教授的《算法+数据结构=程序》正是这种观点的集中体现。

目前，数据结构在计算机科学中是一门综合性的专业基础课。数据结构的研究不仅涉及计算机硬件（特别是编码理论、存储装置和存取方法等）的研究范围，而且和计算机软件的研究有着密切的关系，无论是编译程序还是操作系统都涉及数据元素在存储器中的分配问题。在研究信息检索时也必须考虑如何组织数据，以使查找和存取数据元素更为方便。因此，可以认为数据结构是介于数学、计算机硬件和软件三者之间的一门核心课程。

有关“数据结构”的研究仍不断发展，一方面，面向各专门领域中特殊问题的数据结构正在研究和发展；另一方面，从抽象数据类型的观点来讨论数据结构，已成为一种新的趋势，越来越被人们所重视。

1.2　基本概念和术语

下列概念和术语将在以后各章节中多次出现，本节先对这些概念和术语赋予确定的含义。

1.2.1　数据、数据元素、数据项和数据对象

数据（Data）是客观事物的符号表示，是所有能输入到计算机中并被计算机程序处理的符号的总称。如数学计算中用到的整数和实数，文本编辑中用到的字符串，多媒体程序处理的图形、图像、声音及动画等通过特殊编码定义后的数据。

数据元素（Data Element）是数据的基本单位，在计算机中通常作为一个整体进行考虑和处理。在有些情况下，数据元素也称为元素、结点、记录等。数据元素用于完整地描述一个对象，如前一节示例中的一名学生记录，树中棋盘的一个格局（状态），以及图中的一个顶点等。

数据项（Data Item）是组成数据元素的、有独立含义的、不可分割的最小单位。例如，学生基本信息表中的学号、姓名、性别等都是数据项。

数据对象（Data Object）是性质相同的数据元素的集合，是数据的一个子集。例如：整

数数据对象是集合 $N=\{0, \pm1, \pm2, \cdots\}$，字母字符数据对象是集合 $C=\{$'A', 'B', …, 'Z', 'a', 'b', …, 'z'$\}$，学生基本信息表也可是一个数据对象。由此可以看出，不论数据元素集合是无限集（如整数集），或是有限集（如字母字符集），还是由多个数据项组成的复合数据元素（如学生表），只要性质相同，都是同一个数据对象。

1.2.2 数据结构

数据结构（Data Structure）是相互之间存在一种或多种特定关系的数据元素的集合。换句话说，数据结构是带“结构”的数据元素的集合，“结构”就是指数据元素之间存在的关系。

数据结构包括逻辑结构和物理结构两个层次。

1. 逻辑结构

数据的**逻辑结构**是从逻辑关系上描述数据，它与数据的存储无关，是独立于计算机的。因此，数据的逻辑结构可以看作是从具体问题抽象出来的数学模型。

数据的逻辑结构有两个要素：一是数据元素；二是关系。数据元素的含义如前所述，关系是指数据元素间的逻辑关系。根据数据元素之间关系的不同特性，通常有四类基本结构，如图 1.3 所示。它们的复杂程度依次递进。

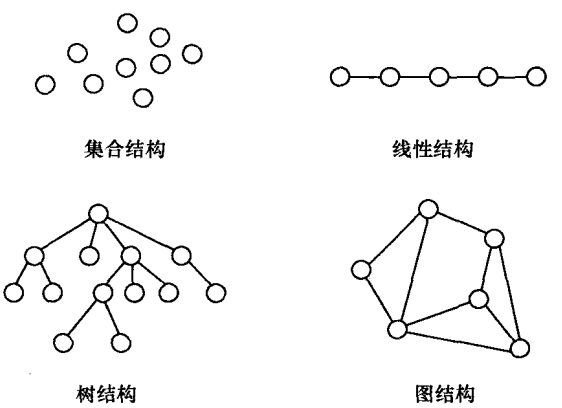


图 1.3　四类基本逻辑结构关系图

下面四种结构中所举的示例是以某班级学生作为数据对象（数据元素是学生的学籍档案记录），来分别考察数据元素之间的关系。

（1）集合结构

数据元素之间除了“属于同一集合”的关系外，别无其他关系。例如，确定一名学生是否为班级成员，只需将班级看做一个集合结构。

（2）线性结构

数据元素之间存在一对一的关系。例如，将学生信息数据按照其入学报到的时间先后顺序进行排列，将组成一个线性结构。

（3）树结构

数据元素之间存在一对多的关系。例如，在班级的管理体系中，班长管理多个组长，每位组长管理多名组员，从而构成树结构。

（4）图结构或网状结构

数据元素之间存在多对多的关系。例如，多位同学之间的朋友关系，任何两位同学都可以是朋友，从而构成图结构或网状结构。

其中集合结构、树结构和图结构都属于非线性结构。

线性结构包括线性表（典型的线性结构，如学生基本信息表的例子）、栈和队列（具有特殊限制的线性表，数据操作只能在表的一端或两端进行）、字符串（也是特殊的线性表，其特殊性表现在它的数据元素仅由一个字符组成）、数组（是线性表的推广，它的数据元素是一个线性表）、广义表（也是线性表的推广，它的数据元素是一个线性表，但不同构，即或者是单元素，或者是线性表）。非线性结构包括树（具有多个分支的层次结构）和二叉树（具有两个分支的层次结构）、有向图（一种图结构，边是顶点的有序对）和无向图（另一种图结构，边是顶点的无序对）。这几种逻辑结构可以用一个层次图描述，如图 1.4 所示。

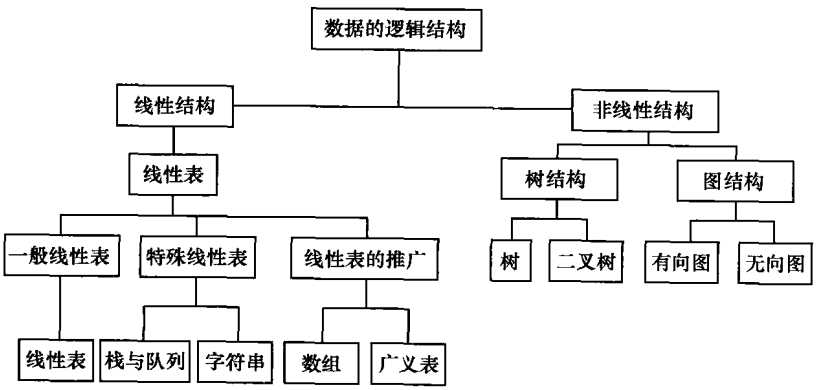


图 1.4　几种逻辑结构层次图

2. 存储结构

数据对象在计算机中的存储表示称为数据的**存储结构**，也称为**物理结构**。把数据对象存储到计算机时，通常要求既要存储各数据元素的数据，又要存储数据元素之间的逻辑关系，数据元素在计算机内用一个结点来表示。数据元素在计算机中有两种基本的存储结构，分别是顺序存储结构和链式存储结构。

（1）顺序存储结构

顺序存储结构是借助元素在存储器中的相对位置来表示数据元素之间的逻辑关系，通常借助程序设计语言的数组类型来描述。

对于前面的“学生基本信息表”，假定每个结点（学生记录）占用 50 个存储单元，数据从 0 号单元开始由低地址向高地址方向存储，对应的顺序存储结构如表 1.2 所示。

表 1.2　　顺序存储结构

地　址	学　号	姓　名	性　别	籍　贯	专　业
0	060214201	杨阳	男	安徽	计算机科学与技术
50	060214202	薛林	男	福建	计算机科学与技术
100	060214215	王诗萌	女	吉林	计算机科学与技术
150	060214216	冯子晗	女	山东	计算机科学与技术

（2）链式存储结构

顺序存储结构要求所有的元素依次存放在一片连续的存储空间中，而链式存储结构无需占用一整块存储空间。但为了表示结点之间的关系，需要给每个结点附加指针字段，用于存放后继元素的存储地址。所以链式存储结构通常借助于程序设计语言的指针类型来描述。

假定给前面的“学生基本信息表”中的每个结点附加一个“下一个结点地址”，即后继指针字段，用于存放后继结点的首地址，则可得到如表 1.3 所示的链式存储结构。从表中可以看出，每个结点占用两个连续的存储单元，一个存放结点的信息，另一个存放后继结点的首地址。

表 1.3　　链式存储结构

地址	学号	姓名	性别	籍贯	专业	后继结点的首地址
0	060214201	杨阳	男	安徽	计算机科学与技术	100
50	060214216	冯子晗	女	山东	计算机科学与技术	^
100	060214202	薛林	男	福建	计算机科学与技术	150
150	060214215	王诗萌	女	吉林	计算机科学与技术	50

为了更清楚地反映链式存储结构，可采用更直观的图示来表示，如“学生基本信息表”的链式存储结构可用如图 1.5 所示的方式表示。

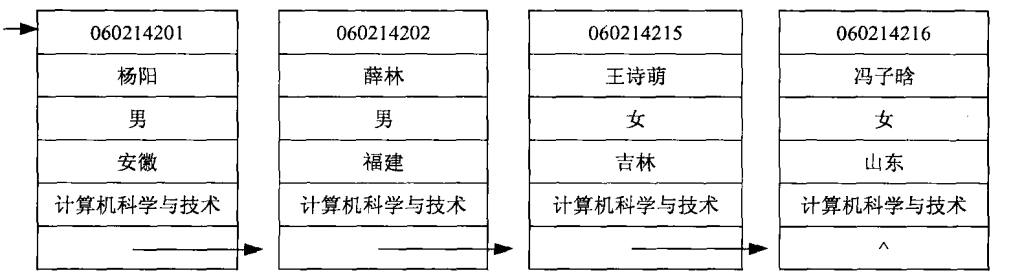


图 1.5　链式存储结构示意图

1.2.3　数据类型和抽象数据类型

1. 数据类型

数据类型（Data Type）是高级程序设计语言中的一个基本概念，前面提到过顺序存储结构可以借助程序设计语言的数组类型描述，链式存储结构可以借助指针类型描述，所以数据类型和数据结构的概念密切相关。

一方面，在程序设计语言中，每一个数据都属于某种数据类型。类型明显或隐含地规定了数据的取值范围、存储方式以及允许进行的运算，**数据类型**是一个值的集合和定义在这个值集上的一组操作的总称。例如，C 语言中的整型变量，其值集为某个区间上的整数（区间大小依赖于不同的机器），定义在其上的操作为加、减、乘、除和取模等算术运算；而实型变量也有自己的取值范围和相应运算，比如取模运算是不能用于实型变量的。程序设计语言允许用户直接使用的数据类型由具体语言决定，数据类型反映了程序设计语言的数据描述和处理能力。C 语言除了提供整型、实型、字符型等基本类型数据外，还允许用户自定义各种

类型数据，例如数组、结构体和指针等。

2. 抽象数据类型

抽象就是抽取出实际问题的本质。在计算机中使用二进制数来表示数据，在汇编语言中则可给出各种数据的十进制表示，它们是二进制数据的抽象，使用者在编程时可以直接使用，不必考虑实现细节。在高级语言中，则给出更高一级的数据抽象，出现了数据类型，如整型、实型、字符型等，可以进一步利用这些类型构造出线性表、栈、队列、树、图等复杂的抽象数据类型。

抽象数据类型（Abstract Data Type，ADT）一般指由用户定义的、表示应用问题的数学模型，以及定义在这个模型上的一组操作的总称，具体包括三部分：数据对象，数据对象上关系的集合，以及对数据对象的基本操作的集合。

抽象数据类型的定义格式如下：

```
ADT 抽象数据类型名 {
  数据对象：〈数据对象的定义〉
  数据关系：〈数据关系的定义〉
  基本操作：〈基本操作的定义〉
} ADT 抽象数据类型名
```

其中，数据对象和数据关系的定义采用数学符号和自然语言描述，基本操作的定义格式为

```
基本操作名（参数表）
  初始条件：〈初始条件描述〉
  操作结果：〈操作结果描述〉
```

基本操作有两种参数：赋值参数只为操作提供输入值；引用参数以“&”打头，除可提供输入值外，还将返回操作结果。“初始条件”描述了操作执行之前数据结构和参数应满足的条件，若初始条件为空，则省略。“操作结果”说明了操作正常完成之后，数据结构的变化状况和应返回的结果。

1.3　抽象数据类型的表示与实现

运用抽象数据类型描述数据结构，有助于在设计一个软件系统时，不必首先考虑其中包含的数据对象，以及操作在不同处理器中的表示和实现细节，而是在构成软件系统的每个相对独立的模块上定义一组数据和相应的操作，把这些数据的表示和操作细节留在模块内部解决，在更高的层次上进行软件的分析和设计，从而提高软件的整体性能和利用率。

抽象数据类型的概念与面向对象方法的思想是一致的。抽象数据类型独立于具体实现，将数据和操作封装在一起，使得用户程序只能通过抽象数据类型定义的某些操作来访问其中的数据，从而实现了信息隐藏。在 C++中，我们可以用类的声明表示抽象数据类型，用类的实现来实现抽象数据类型。因此，C++中实现的类相当于数据的存储结构及其在存储结构上实现的对数据的操作。

抽象数据类型和类的概念实际上反映了程序或软件设计的两层抽象：抽象数据类型相当于在概念层（或称为抽象层）上描述问题，而类相当于在实现层上描述问题。此外，C++中的类只是一个由用户定义的普通类型，可用它来定义变量（称为对象或类的实例）。因此，在C++中，最终是通过操作对象来解决实际问题的，所以我们可将该层次看做是应用层。例如，main程序就可看做是用户的应用程序。

由此可以看出，最终表示和实现抽象数据类型，最好用面向对象的方法，比如用C++语言的类描述比较方便、有效，但本课程大都在大学低年级开设，用C语言的描述方法更符合学生的实际情况。另外，由于实际问题千变万化，数据模型和算法也形形色色，因此抽象数据类型的设计和实现，就不可能像基本数据类型那样规范和一劳永逸，本书所讨论的数据结构及其算法主要是面向读者的，故采用介于伪码和C语言之间的类C语言作为描述工具。这使得数据结构与算法的描述与讨论简明清晰，不拘泥于C语言的细节，又容易转换成C或C++程序。

本书采用的类C语言精选了C语言的一个核心子集，同时做了若干扩充修改，增强了语言的描述功能，以下对其做简要说明。

（1）预定义常量及类型：

```
//函数结果状态代码
#define OK 1
#define ERROR 0
#define OVERFLOW -2
// Status是函数返回值类型，其值是函数结果状态代码。
typedef int Status;
```

（2）数据结构的表示（存储结构）用类型定义（typedef）描述；数据元素类型约定为ElemType，由用户在使用该数据类型时自行定义。

（3）基本操作的算法都用如下格式的函数来描述：

```
函数类型 函数名（函数参数表）{
    //算法说明
    语句序列
}//函数名
```

当函数返回值为函数结果状态代码时，函数定义为Status类型。为了便于描述算法，除了值调用方式外，增加了C++语言引用调用的参数传递方式。在形参表中，以“&”打头的参数即为引用参数。传递引用给函数与传递指针的效果是一样的，形参变化实参也发生变化，但引用使用起来比指针更加方便、高效。

（4）内存的动态分配与释放。

使用new和delete动态分配和释放内存空间：

分配空间　指针变量=new 数据类型;

释放空间　delete 指针变量;

（5）赋值语句：

简单赋值　变量名 = 表达式;

串联赋值　变量名1 = 变量名2 = ... = 变量名n = 表达式;

成组赋值 (变量名 1, ..., 变量名 *n*) = (表达式 1, ..., 表达式 *n*);

结构赋值 结构名 1 = 结构名 2;

结构名 = (值 1, 值 2, ..., 值 *n*);

条件赋值 变量名 = 条件表达式 ? 表达式 T: 表达式 F;

交换赋值 变量名 1 <-->变量名 2;

(6) 选择语句:

条件语句 1 if (表达式) 语句;

条件语句 2 if (表达式) 语句;

else 语句;

开关语句 switch (表达式) {

case 值 1: 语句序列 1 ;break;

case 值 2: 语句序列 2 ;break;

…

case 值 *n*: 语句序列 n;break;

default: 语句序列 *n*+1;

}

(7) 循环语句:

for 语句　　for (表达式 1; 条件; 表达式 2) 语句;

while 语句　　while (条件) 语句;

do-while 语句　do {

语句序列;

} while (条件);

(8) 结束语句:

函数结束语句　return 表达式;

return;

case 或循环结束语句　break;

异常结束语句　exit (异常代码);

(9) 输入输出语句使用 C++流式输入输出的形式:

输入语句　cin>>变量 1>>…>>变量 *n*;

输出语句　cout<<表达式 1<<…<<表达式 *n*;

(10) 基本函数:

求最大值　Max (表达式 1,...,表达式 *n*)

求最小值　Min (表达式 1,...,表达式 *n*)

下面以复数为例，给出一个完整的抽象数据类型的定义、表示和实现。

(1) 定义部分:

```
ADT Complex {
   数据对象: D={e1,e2|e1,e2∈R,R是实数集}
```

```
    数据关系：S={<e1,e2>|e1 是复数的实部，e2 是复数的虚部}
    基本操作：
      Creat( &C, x, y )
        操作结果：构造复数 C，其实部和虚部分别被赋以参数 x 和 y 的值。
      GetReal(C)
        初始条件：复数 C 已存在。
        操作结果：返回复数 C 的实部值。
      GetImag(C)
        初始条件：复数 C 已存在。
        操作结果：返回复数 C 的虚部值。
      Add( C1,C2)
        初始条件：C1，C2 是复数。
        操作结果：返回两个复数 C1 和 C2 的和。
      Sub( C1,C2)
        初始条件：C1，C2 是复数。
        操作结果：返回两个复数 C1 和 C2 的差。
} ADT Complex
```

在后面的章节中，每定义一个新的数据结构，都先用这种定义方式给出其抽象数据类型的定义，对于数据结构的表示方法，则根据不同的存储结构相应给出，同时用类 C 语言给出主要操作的实现方法。下面为了让读者对抽象数据类型有一个完整、正确的理解，给出复数的存储表示和相应操作的具体实现过程。

（2）表示部分：

```
typedef struct                                  //复数类型
   float Realpart;                              //实部
   float Imagepart;                             //虚部
}Complex;
```

（3）实现部分：

```
void Creat( &Complex C, float x, float y ){     //构造一个复数
   C.Realpart=x;
   C.Imagepart=y;
}
float GetReal(Complex C){                       //取复数 C=x+yi 的实部
   return C.Realpart;
}
float GetImag(Complex C){                       //取复数 C=x+yi 的虚部
   return C.Imagepart;
}
Complex Add(Complex C1, Complex C2){            //求两个复数 C1 和 C2 的和 sum
   Complex sum;
   sum.Realpart=C1.Realpart+C2.Realpart;
   sum.Imagepart=C1.Imagepart+C2.Imagepart;
   return sum;
}
Complex Sub(Complex C1, Complex C2){            //求两个复数 C1 和 C2 的差 difference
   Complex difference;
   difference.Realpart=C1.Realpart-C2.Realpart;
```