21世纪高等教育计算机规划教材

COMPUTER

数据结构（C++语言版）

Data Structure (C++ Language Version)

秦锋 汤亚玲 主编

- 定位明确，面向全国应用型本科院校，难度适中，基础和能力并重
- 内容深入浅出，算法设计简明可读性强、文字叙述精练
- 案例丰富，注重C++语言在算法中的实现，实践性强

人民邮电出版社
POSTS & TELECOM PRESS

21世纪高等教育计算机规划教材

COMPUTER

数据结构（C++语言版）

Data Structure (C++ Language Version)

秦锋 汤亚玲 主编

陈桂芬 章曙光 汪军 林芳 司秀丽 陈学进 秦飞 副主编

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 / 秦锋, 汤亚玲主编. -- 北京 : 人民邮电出版社, 2014.9
21世纪高等教育计算机规划教材
ISBN 978-7-115-35861-5

Ⅰ. ①数… Ⅱ. ①秦… ②汤… 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178478号

内 容 提 要

本书在简要回顾基本 C++程序设计概念的基础上，全面系统地介绍了队列、堆栈、树、图等基本数据结构。本书将 C++语言作为数据结构的算法描述语言。一方面对传统的数据结构内容进行了 C++语言实现，另一方面将数据结构与面向对象技术结合起来，围绕抽象数据类型的概念来讨论每一种数据结构及算法。书中大量 C++语言的程序实例既是数据结构的具体实现，又是面向对象技术的算法基础。本书理论与实践并重，每章都有大量的习题，强调数据结构的应用价值。

本书可作为计算机类及信息类相关专业的核心教材，也可供广大研究开发人员自学参考使用。

◆ 主　　编　秦　锋　汤亚玲
副 主 编　陈桂芬　章曙光　汪　军　林　芳　司秀丽
　　陈学进　秦　飞
责任编辑　邹文波
执行编辑　吴　婷
责任印制　彭志环　杨林杰

◆ 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：16.75　　2014 年 9 月第 1 版
字数：438 千字　　2014 年 9 月河北第 1 次印刷

定价：39.00 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

写出高质量的程序是每个软件开发者追求的目标。要达到这个目标，仅靠学习几门高级语言是远远不够的，正如一个人仅靠认识几个汉字绝不会写出好文章一样。数据结构这门课程正是开启程序设计知识宝库的金钥匙，其主要目的是培养学生将现实世界抽象为数据和数据模型的能力以及利用计算机进行数据存储和数据加工的能力。

数据结构在计算机科学的各领域中应用十分广泛，如编译系统中要使用栈、语法树等，操作系统中要使用队列、存储管理表、目录树等，数据库系统中要使用线性表、链表、索引树等，人工智能中要使用广义表、检索树、图等。同样，在面向对象的程序设计、计算机图形学、多媒体技术、软件工程等领域，都会用到各种不同的数据结构。因此，学好“数据结构”这门课，能掌握更多的程序设计技巧，并能评价出算法的优劣，为以后学习计算机专业课程及走上工作岗位从事计算机大型软件开发打下良好的基础。

本书内容共 10 章。第 1 章介绍了数据结构与算法等一些基本概念，并对算法描述及算法分析做了简单说明，介绍了衡量算法优劣的主要因素：时间复杂度和空间复杂度的求法。第 2 章简单介绍了 C++基本知识，让熟悉 C 语言但对 C++比较陌生的读者能迅速掌握 C++的基本要点，为后续章节的学习打下基础。第 3 章～第 5 章介绍了线性表、栈、队列、串等的线性结构的逻辑特性、存储结构，以及常用的操作算法的实现和基本应用。第 6 章～第 8 章介绍了多维数据、广义表、树、二叉树、图等非线性结构的逻辑特征，在计算机中的存储表示，一些常用算法实现及基本应用。第 9 章～第 10 章介绍了在计算机中使用非常广泛的排序和查找两种运算，对一些常用的查找、排序算法进行了详细描述，并给出了实现的算法及效率分析。

本书的特点是采用面向对象程序设计语言，即 C++语言作为算法的描述语言，所有算法都已经在 VC++6.0 环境下上机调试通过。由于篇幅所限，大部分算法都以单独的函数形式给出；若读者要运行这些算法，还必须给出一些变量的说明及主函数来调用所给的函数。为方便读者对算法的理解和验证，书中尽量避免采用复杂的 C++机制，如函数模板、类模板、虚基类、多重继承等，只采用通俗易懂的类定义、数据封装和对象指针及简单的继承等基本机制，让读者更多地关注算法本身的设计思想。

“数据结构”是一门实践性很强的课程。读者在进行理论学习的同时，需要动手编写大量的程序并上机调试，以加深对所学知识的理解，也只有这样才能提高自己的编程能力。书中配有丰富的各种类型的习题。对于每章的算法设计题，希望读者思考并有选择地上机验证。

为方便教学，本书免费为授课教师提供电子教案和习题解答，可在人民邮电出版社教学服务与资源网（www.ptpedu.com.cn）上进行下载。

目前，“数据结构”是我国高校计算机专业的核心课程之一，也是其他信息类专业如信息管理、通信工程、信息与计算科学等的必修课程之一。正因为它在计算机培养计划中的重要地位，大多数高校计算机专业研究生入学考试都将“数据结构”作为必考课程之一。

本书可作为高等院校计算机类或信息类相关专业“数据结构”课程教材，建议理论课时为 50～70 学时，上机及课程设计等实践课时为 20～30 学时。各院校可根据本校的专业特点和学生的实际情况，适当增删。

本书由秦锋、汤亚玲任主编，负责全书的编撰和整理，以确保各章节内容的完整和风格的统一；由陈桂芬、章曙光、汪军、林芳、司秀丽、陈学进、秦飞任副主编。

其中，第 1 章、第 7 章由秦锋编写，第 8 章由汤亚玲编写，第 6 章由陈桂芬编写，第 9 章由章曙光编写，第 4 章由汪军编写，第 10 章由林芳编写，第 3 章由司秀丽编写，第 5 章由陈学进编写，第 2 章由秦飞编写。全书由秦锋、汤亚玲负责修改并统稿。

因编者水平有限，书中难免有不足甚至错误之处，敬请广大读者批评指正！

建议或者意见请联系 fqin@ahut.edu.cn、tangyl@ahut.edu.cn。

编　者

2014 年 6 月

目 录

第1章 绪论

教学提示：数据结构主要研究4方面的问题：（1）数据的逻辑结构，即数据之间的逻辑关系；（2）数据的物理结构，即数据在计算机内的存储方式；（3）对数据的加工，即基于某种存储方式的运行算法；（4）算法的分析，即评价算法的优劣。本章重点介绍数据结构研究问题所涉及的基本知识和概念。

教学目标：了解研究数据结构的目的及相关基本概念和术语，掌握算法基本概念和算法评价依据——时间复杂度和空间复杂度。

1.1 数据结构的概念

软件设计是计算机学科各领域的核心。在计算机发展的早期，软件设计所处理的数据都是整型、实型等简单数据，绝大多数的应用软件都是用于数值计算。随着信息技术的发展，计算机逐渐进入金融、商业、管理、通信及制造业等行业，广泛地应用于数据处理和过程控制。计算机加工处理的对象也由纯粹的数值型数据发展到字符、表格和图像等各种具有一定结构的数据，这就给程序设计带来一些新的问题。为了设计出一个结构好而且效率高的程序，必须研究数据的特性、相互关系及对应的存储表示，并利用这些特性和关系设计出相应的算法和程序，这正是数据结构课程研究的内容。

1.1.1 什么是数据结构

用计算机解决一个具体问题时，一般需要经过下列几个步骤：首先要从该具体问题抽象出一个适当的数学模型，然后设计或选择一个解此数学模型的算法，最后编写出程序进行调试、测试，直至得到最终的解答。

由于早期计算机所涉及的运算对象是简单的整型、实型或布尔类型数据，程序设计者的精力主要集中于程序设计的技巧上，无须重视数据结构。随着计算机应用领域的扩大和软、硬件技术的发展，非数值计算问题显得越来越重要。据统计，当今用计算机处理的非数值计算性问题占90%以上的机器时间，如图书资料的检索、职工档案管理、博弈游戏等，这类问题涉及的数据结构相当复杂，数据元素之间的相互关系无法用数学方程或数学公式来描述。这类问题的处理对象中的各分量不再是单纯的数值型数据，更多的是字符、字符串或其他编码表示的信息。因此，首要的问题是把处理对象中的各种信息按其逻辑特性组织起来，再研究如何把它们存储到计算机中。只有做完了这些工作，才能设计解决具体问题的算法，并编写出相应的程序。下面列举的具体问题就属于这一类。

【例 1.1】 学生成绩查询。

假定要编写一个计算机程序以查询某大学或某地区学生英语四级考试成绩。解决此问题，首先要构造一张成绩登记表，表中每个登记项至少有 3 个信息：准考证号、姓名与考试成绩。写出查找算法的好坏，取决于这个登记表的结构及存储方式。最简单的方式是把表中的信息按照某种次序（如登记的次序）依次存储在计算机内一组连续的存储单元中。用高级语言表述，就是把整个表作为一个数组，表的每项（一个人的准考证号、姓名与考试成绩）是数组的一个元素。按准考证号查找时，从表的第一项开始，依次查对准考证号，直到找出指定的准考证号或确定表中没有要找的准考证号为止。这种查找算法对于一个规模不大的学校或许是可行的，但对于一个有几十万甚至几百万考生的省份或地区就不适用了。因此，一种常见的做法是把成绩登记表按照准考证号从小到大排序（见表 1-1），并存储在计算机内一组连续的存储单元中，在查找时可以采用折半查找算法（第 9 章有详细介绍），查找速度可大大提高。

表 1-1　　英语四级成绩表

准 考 证 号	姓　　名	成绩（按百分制算）
100405003	张　山	75
100405004	陈　明	59
100405123	王芳芳	63
…	…	…
100507082	丁凯乐	56
100507083	李　明	66

从这个例子可以看出，成绩登记表如何构造、如何存储在计算机内存中将直接影响查找算法的设计以及算法的执行效率。表 1-1 是为解决英语四级成绩查询问题而建立的数学模型。这类模型的主要操作是按照某个特定要求（如给定姓名或准考证号等）对登记表进行查询。诸如此类的还有人事档案管理、图书资料管理等。这类文档管理的数学模型中，计算机处理的对象之间通常存在一种简单的线性关系，故这类数学模型可称为线性的数据结构。

【例 1.2】 判定树问题。

有 8 枚硬币，分别用 a、b、c、d、e、f、g、h 表示，其中有且仅有一枚硬币是伪造的，假硬币的质量和真硬币的质量不同，可能轻，也可能重。现要求设计算法，用最少的比较次数挑选出假硬币，并同时确定这枚假硬币的质量比其他硬币的质量是重还是轻。

该问题的解决似乎比较难，借助一种叫作判定树的数据结构，将此问题的求解过程描述成图 1.1。很容易看出，只要比较 3 次就能找出答案。图中字母 H 和 L 分别表示假硬币较其他真硬币重、轻。

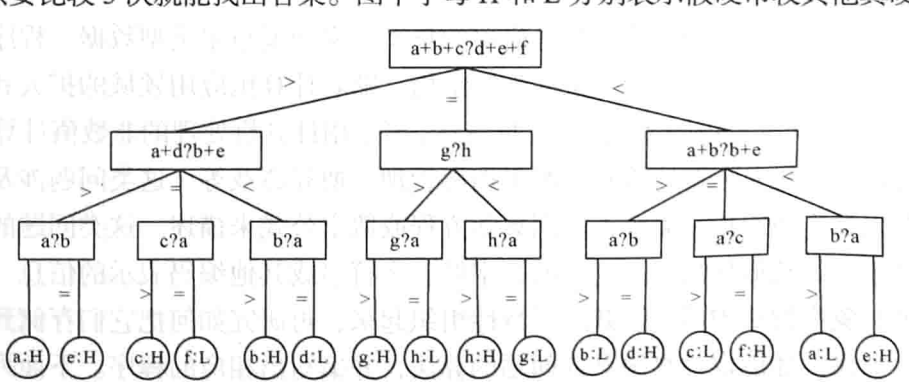

图 1.1　判定树

显然，这种树结构不是线性结构，它比线性结构复杂。

【例 1.3】 最小代价问题。

假设几个村庄之间要架设输电线路，根据电能的可传递性，并不需要在每对村庄之间架设线路。如何用最小代价架设线路让每个村庄都能用上电。

处理此类问题需要用到图这种数据结构。用顶点代表村庄，每对顶点之间的边代表村庄之间的线路，边的权值代表线路的建设费用，如图 1.2 所示。只要把这个图的有关信息存储在计算机中，利用图论中最小生成树算法，在此图中找出不能形成回路的 $n-1$ 条边，并使得权值最小。显然，这种图结构比树结构更加复杂。

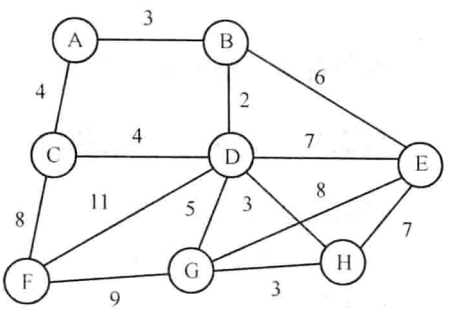

图 1.2　输电线路示意图

从上述 3 个例子可见，描述非数值计算问题的数学模型不再是数学方程或数学公式，而是诸如表、树、图之类的数据结构。从这些例子也可以看出，非数值计算问题的求解往往比较复杂，涉及数据的逻辑结构、数据存储、数据加工等。这也正是“数据结构”这门课程所要研究的内容。

1.1.2　学习数据结构的意义

数据结构是计算机科学与技术专业的核心基础课程。所有的计算机系统软件和应用软件都要用到各种类型的数据结构。因此，要想更好地运用计算机来解决实际问题，仅掌握几种计算机程序设计语言是难以应付众多复杂课题的。要想有效地使用计算机、充分发挥计算机的性能，还必须学习和掌握好数据结构的有关知识。学好“数据结构”这门课程，对于学习计算机专业的其他课程，如操作系统、编译原理、数据库管理系统、软件工程、人工智能等都是十分有益的。

数据结构作为一门独立的课程，最早是由美国开设的。1968 年，美国唐• 欧• 克努特教授开创了数据结构的最初体系。他所著的《计算机程序设计技巧》第一卷《基本算法》，是第一本较系统地阐述数据的逻辑结构和存储结构及其操作的著作。20 世纪 60 年代末至 70 年代初，出现了大型程序，软件也相对独立，结构程序设计成为程序设计方法学的主要内容，数据结构的地位显得更为重要。人们认为，程序设计的实质是对确定的问题选一个好的结构，加上一种好的算法。当时，数据结构几乎和图论，特别是表、树的理论为同义语。随后，数据结构这个概念被扩充到包括图、集合、格、关系等方面，从而变成现在称为“离散结构”的内容。然而，由于数据必须在计算机中进行处理，因此，不仅要考虑数据本身的数学特性，还要考虑数据的存储结构，这就进一步扩大了数据结构的内容。

瑞士计算机科学家沃斯（N.Wirth）教授曾以“算法+数据结构=程序”作为他的一本著作的名称。可见，程序设计的实质是对实际问题选择一种好的数据结构，并设计一个好的算法。

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，而且与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，都涉及如何组织数据，使检索和存取数据更为方便。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。

目前，数据结构是我国高校计算机专业的核心课程之一，也是其他信息类专业如信息管理、通信工程、信息与计算科学等的必修课程之一。

1.2 基本概念和术语

本节将对一些基本概念和术语加以定义和解释，这些概念和术语将在以后的章节中多次出现。

1.2.1 数据与数据元素

数据（Data）是对客观事物的符号表示，它能被计算机识别、存储和加工处理。它是计算机程序加工的“原料”。例如，一个代数方程求解程序所用到的数据是整数和实数，一个编译程序处理的对象是字符串（源程序）。在计算机科学中，数据的含义相当广泛。如客观世界中的声音、图像等，经过某些处理也可被计算机识别、存储和处理，因而它们也属于数据的范畴。

数据元素（Data Element）是数据的基本单位，有时也称为元素、结点、顶点、记录。一个数据元素可能由若干数据项（Data Item）组成。例如，例 1.1 中的英语四级成绩表中，每个人的准考证号、姓名与成绩组成了一个数据元素，即一个数据元素包含 3 个数据项。整个英语四级成绩表就是计算机程序要处理的数据。数据项是最小标识单位，有时也称为字段、域或属性。数据元素也可以仅有一个数据项。

数据结构（Data Structure）是指数据元素之间的相互关系，即数据的组织形式。它一般包括以下 3 方面的内容：

（1）数据元素之间的逻辑关系，也称为数据的逻辑结构（Logical Structure）。它独立于计算机，是数据本身所固有的。

（2）数据元素及逻辑关系在计算机存储器内的表示方式，称为数据的存储结构（Storage Structure）。它是逻辑结构在计算机存储器中的映射，必须依赖于计算机。

（3）数据运算，即对数据施加的操作。运算的定义直接依赖于逻辑结构，但运算的实现必须依赖于存储结构。

1.2.2 数据的逻辑结构

数据的逻辑结构是从逻辑关系上描述数据，不涉及数据在计算机中的存储，是独立于计算机的。可以说，数据的逻辑结构是程序员根据具体问题抽象出来的数学模型。数据中元素通常有下列四种形式的逻辑关系。

（1）集合：任何两个元素之间都没有逻辑关系，每个元素都是孤立的。

（2）线性结构：结构中的元素之间存在一对一的关系，即所谓的线性关系。例 1.1 中的四级考试成绩表就是一个线性结构。在这个结构中，元素（由一个人的准考证号、姓名和考试成绩组成）的排列十分有序，第一个元素之后紧跟着第二个元素，第二个元素之后紧跟着第三个元素，以此类推，整个结构就像一条“链”，故有“线性结构”之称。

（3）树形结构：结构中的数据元素之间存在一对多的关系，如例 1.2 中的判定树。这种结构像自然界中倒长的“树”一样，呈分支、层次状态。在这种结构中，元素之间的逻辑关系通常称作双亲与子女关系。例如，家谱、行政组织结构等都可用树形结构来表示。

（4）图状结构：结构中的元素之间存在多对多的关系。也就是说，元素间的逻辑关系可以是任意的，如例 1.3 的数学模型。在这种结构中，元素间的逻辑关系也称作邻接关系。通常将集合、树形结构、图状结构归纳为非线性结构。因此，数据的逻辑结构可分为两大类，即线性

结构和非线性结构。

1.2.3　数据的存储结构

数据的存储结构是指数据在计算机内的表示方法，是逻辑结构的具体实现。因此，存储结构应包含两方面的内容，即数据元素本身的表示与数据元素间逻辑关系的表示。数据的存储结构有下列 4 种基本方式。

（1）顺序存储：将数据元素依次存储于一组地址连续的存储单元中，元素间的逻辑关系由存储单元的位置直接体现，由此得到的存储表示称为顺序存储结构（Sequential Storage Structure）。高级语言中，常用一维数组来实现顺序存储结构。该方法主要用于线性结构。非线性结构也可通过某种线性化的处理，实现顺序存储。

（2）链接存储：将数据元素存储在一组任意的存储单元中，用附加的指针域表示元素之间的逻辑关系，由此得到的存储表示称为链接存储（Linked Storage Structure）。使用这种存储结构时，往往把一个数据元素及附加的指针一起称作一个结点。高级语言中，常用指针变量实现链接存储。

（3）索引存储：该方法的特点是在存储数据元素的同时，还可以建立附加的索引表。索引表中每一项称为索引项。索引项的一般形式是：（关键字，地址）。关键字是指能唯一标识数据元素的数据项。若每个数据元素在索引表中均有一个索引项，则该索引表称为稠密索引（Dense Index）。若一个索引项对应一组数据元素，则该索引表称为稀疏索引（Sparse Index）。

（4）散列存储：该方法是依据数据元素的关键字，用一个事先设计好的函数计算出该数据元素的存储地址，然后把它存入该地址中。这种函数称为散列函数，由散列函数计算出的地址称为散列地址。

上述 4 种存储方式既可单独使用，也可组合使用。逻辑结构确定后，采取何种存储结构，要根据具体问题而定，主要的考虑因素是运算方便、算法效率与空间的要求。例如为了提高查找例 1.1 中英语四级成绩表的效率，也可以采用索引存储方式，将姓张的、姓李的、姓王的……考生成绩分别按照姓氏存储在一起，同时建一个姓氏索引表。如果要查找李某的成绩，先到索引表中找到姓李的考生存储首地址，再去查找李某的成绩。

存储结构的描述与程序设计语言有关。用机器语言描述，则存储结构是数据元素在存储器中的物理位置；用高级语言描述，则不必涉及计算机的内存地址，可用类型说明来描述存储结构。

1.2.4　数据运算

数据运算是对数据施加的操作。每种逻辑结构都有一个基本运算的集合。例如，最常用的基本运算有检索（查找）、插入、删除、更新、排序等。因为这些运算是在逻辑结构上施加的操作，因此它们同逻辑结构一样也是抽象的，只规定“做什么”，无须考虑“如何做”。只有确定存储结构后，才能考虑“如何做”。简言之，运算在逻辑结构上定义，在存储结构上实现。

必须注意，数据结构包含逻辑结构、存储结构和运算三方面的内容。同一逻辑结构采用不同存储结构，得到的是不同的数据结构，常用不同的数据结构名来标识它们。例如，线性结构采用顺序存储时称为顺序表，采用链接存储时则称为链表。同样，同一逻辑结构定义不同的运算也会导致不同的数据结构。例如，若限制线性结构的插入、删除在一端进行，则该结构称为栈；若限制插入在一端进行，而删除在另一端进行，则称为队列。更进一步，若栈采用顺序存储结构，则称为顺序栈；若栈采用链式存储结构，则称为链栈。顺序栈与链栈也是两种不同的数据结构。

1.2.5 数据类型

数据类型（Data Type）是和数据结构密切相关的一个概念，几乎所有高级语言都提供这一概念。数据类型是一个值的集合和在这个集合上定义的一组操作的总称。例如，C++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、除和取模等运算。

按“值”可否分解，可把数据类型分为两类。

（1）原子类型：其值不可分解，如 C++的基本类型（整型、字符型、实型、枚举型）、指针类型和空类型。

（2）结构类型：其值可分解成若干成分（或称分量），如 C++的数组类型、结构类型等。结构类型的成分可以是原子类型，也可以是某种结构类型。可以把数据类型看作程序设计语言已实现的数据结构。

引入数据类型的目的，从硬件角度考虑，是作为解释计算机内存中信息含义的一种手段；对用户来说，实现了信息的隐蔽，即将一切用户不必了解的细节都封装在类型中。例如，用户在使用整数类型时，既不需要了解整数在计算机内如何表示，也不必了解其操作（如两个整数相加）硬件是如何进行的。

1.2.6 抽象数据类型

抽象数据类型（Abstract Data Type，ADT）是指一个数学模型，以及定义在该模型上的一组操作。抽象数据类型的定义取决于它的一组逻辑特性，而与其在计算机内部如何表示和实现无关。也就是说，不论其内部结构如何变化，只要它的数学特性不变，都不影响其外部的使用。

抽象数据类型和数据类型实质上是一个概念。例如，整数类型是一个 ADT，其数据对象是指能容纳的整数，基本操作有加、减、乘、除和取模等。尽管它们在不同处理器上的实现方法可以不同，但由于其定义的数学特性相同，在用户看来都是相同的。因此，“抽象”的意义在于数据类型的数学抽象特性。

但在另一方面，抽象数据类型的范畴更广。它不再局限于前述各处理器中已定义并实现的数据类型，还包括用户在设计软件系统时自己定义的数据类型。为了提高软件的重用性，在近代程序设计方法学中，要求在构成软件系统的每个相对独立的模块上，定义一组数据和施于这些数据上的一组操作，并在模块的内部给出这些数据的表示及其操作的细节，而在模块的外部使用的只是抽象的数据及抽象的操作。这也就是面向对象的程序设计方法。

抽象数据类型的定义可以由一种数据结构和定义在其上的一组操作组成，而数据结构又包括数据元素间的关系，因此抽象数据类型一般可以由元素、关系及操作三种要素来定义。

抽象数据结构的特征是使用与实现相分离，实行封装的信息隐蔽。也就是说，在抽象数据类型设计时，把类型的定义与其实现分离开来。

1.3 算法和算法分析

1.3.1 算法定义及描述

数据运算是通过算法来描述的，因此讨论算法是数据结构课程的重要内容之一。

算法（Algorithm）是对特定问题求解步骤的描述，是指令的有限序列，其中每条指令表示一个或多个操作。对于实际问题，不仅要选择合适的数据结构，还要有好的算法，才能更好地求解问题。

一个算法必须具备下列 5 个特性。

（1）有穷性：一个算法对于任何合法的输入必须在执行有穷步骤之后结束，且每步都可在有限时间内完成。

（2）确定性：算法的每条指令必须有确切含义，不能有二义性。在任何条件下，算法只有唯一的一条执行路径，即对相同的输入只能得出相同的结果。

（3）可行性：算法是可行的，即算法中描述的操作均可通过已经实现的基本运算的有限次执行来实现。

（4）输入：一个算法有零个或多个输入，这些输入取自算法加工对象的集合。

（5）输出：一个算法有一个或多个输出，这些输出应是算法对输入加工后合乎逻辑的结果。

程序与算法十分相似，但程序不一定要满足有穷性。例如，操作系统启动后，即使没有作业要处理，它仍然会处于等待循环中，以等待新的作业进入，所以不满足有穷性。另外，沃斯（N.Wirth）的“数据结构+算法=程序”公式意味着，可终止的程序的执行部分才是算法。

算法可以使用各种不同的方法来描述。

最简单的方法是使用自然语言。用自然语言来描述算法的优点是简单且便于人们对算法的阅读；缺点是不够严谨，容易产生二义性。

可以使用程序流程图、N-S 图等算法描述工具。其特点是描述过程简洁明了。

用以上两种方法描述的算法不能直接在计算机上执行。若要将它转换成可执行的程序，还有一个编程的问题。

可以直接使用某种程序设计语言来描述算法。不过直接使用程序设计语言并不容易，而且不太直观，常常需要借助注释才能使人看明白。

为了解决理解与执行之间的矛盾，人们常常使用一种称为伪码语言的描述方法来进行算法描述。伪码语言介于高级程序设计语言和自然语言之间，它忽略高级程序设计语言中一些严格的语法规则与描述细节，因此它比程序设计语言更容易描述和被人理解，而比自然语言更接近程序设计语言。它虽然不能直接执行，但很容易被转换成高级语言。本书采用 C++语言作为描述数据结构和算法的工具，但在具体描述时有所简化，如常常将类型定义和变量定义省略。考虑到部分读者有 C 语言的基础但对 C++不太熟悉，本书的第 2 章重点介绍 C++的基础知识。

1.3.2　算法评价

什么是好的算法？同一个问题可能有多种求解的算法，到底哪个更优？通常对算法的评价按照下面 4 个指标来衡量。

（1）正确性（Correctness）：算法的正确性主要有 4 个层次的要求，第一层次是指算法没有语法错误，第二层次是指算法对于几组输入数据能够得出满足规格说明要求的结果，第三层次是指算法对于精心选择的苛刻并带有刁难性的几组输入数据能够得出满足规格说明要求的结果，第四层次是指算法对于一切合法的输入数据都能得出满足规格说明要求的结果。显然，达到第四层次的正确性是非常困难的，因为所有不同输入数据的数量大得惊人，一一验证的方法既不可取也不现实。要证明一个算法完全正确还不是件容易的事情，目前也只处于理论研究阶段。当今对于大型软件进行专业测试，达到第三层次的正确性就认为该软件是合格产品。

（2）可读性（Readability）：指算法要便于人们阅读、交流与调试。可读性好有助于人们对算法的理解；晦涩难懂的算法易于隐藏错误且难以调试和修改。

在此提供几个在程序编写上提高可读性的方法，以帮助读者建立起良好的编写风格。

① 注释：一份良好的程序，除了程序本身外，最重要的是要有一份完整的程序说明文件。一份没有注释的程序，宛如一部天书，常常会让负责维护的程序员搞不懂设计者的设计思想。一份注释完整的程序，除了设计者自己阅读和排错上的方便外，更容易让人了解设计者的程序，并赞叹其在程序设计上的巧思与创意。通常，选择在重要的程序语句后面加上一个注释内容。

请读者将下面这一段程序与以前编写过的程序比较一下，看看这个程序是不是更容易读懂。

```
//= = = = = = = = = Program Description= = = = = = = = =
//程序名称：sum.c
//程序目的：设计一个计算二维数组每行元素之和的程序
//Written By  Hua  Li
//= = = = = = = = = = = = = = = = = = = = = = = = = = = = =

#include <iostream.h>
void RowSum(int A[][4], int nrow)
//计算二维数组 A 每行元素值的和，nrow 是行数
{
   for(int i = 0; i < nrow; i++)
   {
      for(int j = 1; j < 4; j++)
         A[i][0] += A[i][j];                    //每行和保存于数组 A 的第 0 列
   }
}
void main(void)                                 //主函数
{
   int Table[3][4] = {{1,2,3,4},{2,3,4,5},{3,4,5,6}};     //定义并初始化数组
   for(int i = 0; i < 3; i++)                              //输出数组元素
   {
      for(int j = 0; j < 4; j++)
         cout << Table[i][j] << "   ";
      cout << endl;
   }
   RowSum(Table,3);                                        //调用子函数，计算各行和
   for(i = 0; i < 3; i++)                                  //输出计算结果
   {
      cout << "Sum of row " << i << " is " <<Table[i][0]<< endl;
   }
}
```

② 变量命名：编程时常常需要定义变量，变量的取名不可随意。假设想要写一个计算学生成绩的程序，程序中需要用户输入学生学号、语文成绩、英语成绩、数学成绩，最后再计算出 3 科的平均成绩。此时如果程序中的变量声明为：

```
int   x;
int   A,B,C;
int   D;
```

没有人会看懂这几个变量代表什么。即使在程序之初就注明 X 代表学生学号、A 代表语文成绩、B 代表英语成绩、C 代表数学成绩、D 代表 3 科平均成绩，在程序编写或维护过程中，也会

很容易忘记它们的含义。如果把这 5 个变量声明为：

```
int    studentNum;                    //学生学号
int    chinese;                       //语文成绩
int    english;                       //英语成绩
int    math;                          //数学成绩
int    average;                       //3 科平均成绩
```

这样是不是比较清楚易懂呢？因为变量的声明通常会在某一个特定的区域（如程序开头），如果在变量之后再加上一些注释说明，这样在编写或修改程序之际，变量声明的区域就像一个小字典，提供给设计者所有在此程序中的输入和输出信息。

③ 程序缩排：在程序中经常会用到一些条件式语句或循环结构，这个语句包含了 C++语言中区块（Block），也就是一群单一语句的集合，通常以“{”及“}”来划分。“{”表示区块的开始，“}”表示区块的结束，在区块划分上有两种设计的风格。

一种是把开头的“{”与语句放在同一行，“}”要与第一行语句对齐，如：

```
for(i=0 ; i<10 ;  i++ )   {
    cout<<Data[i] ;                   //输出数据
    if(key == data[i])                //查找到数据时
            return i;
    counter++;                        //计数器递增
}
```

另一种是把开头的“{”不与语句放在同一行，“}”要与“{”对齐，如：

```
for(i=0 ; i<10 ; i++ )
{
    cout<<data[i];                    //输出数据
    if(key == data[i])                //查找到数据时
            return i;
    counter++;                        //计数器递增
}
```

这两种格式都可以，但第二种格式更规范，程序可读性更强。

另外程序中最好有缩排，可以帮助程序员减少排错的时间。缩排通常在一些区块、条件语句、循环结构上产生出层次的关系。缩排通常空 2 个空格、4 个空格或 8 个空格。其中以 4 个空格最佳。

④ 段落：程序中除了子程序以外，还有一些专为某一个目的所写的语句，这些语句的个数不等。在编写程序时，不同目的语句之间最好插入一个空白行，再加上注释，这可让程序有段落的感觉，在排错时也较容易找出错误。

（3）健壮性（Robustness）：当输入数据非法或运行环境改变时，算法能恰当地做出反应或进行处理，不会产生莫名其妙的输出结果。为此，算法中应对输入数据和参数进行合法性检查。例如，从键盘输入三角形的三条边的长度，求三角形的面积。当输入的 3 个值不能组成三角形时，不应继续计算，应该报告输入出错并进行处理。处理的方法应是返回一个表示错误或错误性质的值，并中止程序的执行，以便在更高的抽象层次上进行处理。

（4）时空效率（Efficiency）：要求算法的执行时间尽可能短，占用的存储空间尽可能少。但这两者往往相互矛盾，节省了时间可能牺牲空间，反之亦然。设计者应在时间与空间两方面