国家级特色专业“通信工程”系列教材

数据结构与STL

SHUJU JIEGOU YU STL

肖 波 徐雅静 编著

北京邮电大学出版社
www.buptpress.com

内容简介

数据结构是计算机及其相关专业的重要课程，是计算机软件开发及应用人员必备的专业基础。本书首先介绍数据结构与算法的基础知识，然后系统地论述线性表、栈、队列、串、数组和广义表、树和二叉树、图等基本数据结构，并讨论了常用的查找和排序技术。在用例选择方面充分考虑了电子信息类专业特点，尤其突出信息与通信工程相关专业的特色。在各章最后描述了相应的标准模板库(STL)，旨在使读者了解 STL 与数据结构的关系，并且能够掌握各类 STL 的应用，提高实际应用能力和程序设计的效率。

本书内容丰富、层次清晰、讲解深入浅出，可作为计算机及相关专业，尤其是电子信息类专业本专科数据结构课程的教材，也可供从事计算机软件开发和应用的工程技术人员阅读和参考。

北京邮电大学通信工程专业是教育部批准的第一批高等学校特色专业建设项目(TS2055)，本系列教材的编写获得了该项目的资助，其目标是围绕该项目的建设，打造通信工程专业的精品教材。

图书在版编目(CIP)数据

数据结构与 STL/肖波，徐雅静编著．—北京：北京邮电大学出版社，2010.9
ISBN 978-7-5635-2352-8

Ⅰ.①数… Ⅱ.①肖… ②徐… Ⅲ.①数据结构②C 语言—程序设计 Ⅳ.①TP311.12②TP312

中国版本图书馆 CIP 数据核字(2010)第 157485 号

书　　名：数据结构与 STL
编 著 者：肖　波　徐雅静
责任编辑：陈岚岚
出版发行：北京邮电大学出版社
社　　址：北京市海淀区西土城路 10 号(邮编：100876)
发 行 部：电话：010-62282185　传真：010-62283578
E-mail：publish@bupt.edu.cn
经　　销：各地新华书店
印　　刷：北京忠信诚胶印厂
开　　本：787 mm×960 mm　1/16
印　　张：18.75
字　　数：408 千字
印　　数：1—3 000 册
版　　次：2010 年 9 月第 1 版　2010 年 9 月第 1 次印刷

ISBN 978-7-5635-2352-8　　定　价：34.00 元

编者的话

数据结构是计算机及相关专业的重要专业基础课。它不仅是计算机专业学生的必修课程，也是许多非计算机专业的重要课程。数据结构的知识内容及其涉及的技术方法是计算机、电子、信息与通信等领域中诸多课程的基础，同时也是软件工程研究、开发和应用中必备的基础。

数据结构所包含的内容丰富，知识抽象，许多算法技巧性强，学生学习难度大。本书在内容选择上不仅涵盖了数据结构的基础知识及重要数据结构和算法，还结合了电子信息类专业特点，从内容到用例都经过精心选择。撰写本书旨在打造成一本具有一定针对性的电子信息类专业的数据结构教材，同时又不失一般性，亦可作为普通的数据结构教材使用。本书作者长期从事数据结构课程的教学工作，在本书的写作过程中注重知识点的难易把握，突出数据结构在实际问题中的应用，同时对内容进行合理的剪裁和扩充，梳理出清晰的数据结构学习主线。

本书的特点主要表现在以下几个方面。

1. 内容全面

本书第 1 章系统介绍数据结构基本概念及算法分析方法。第 2～6 章系统论述线性表、栈、队列、串、数组和广义表、树和二叉树、图等基本数据结构，对其逻辑结构、存储表示及运算操作进行阐述。最后两章讨论了常用的查找和排序技术。

2. 图表丰富，通俗易懂

全书内容既注重原理又注重实践，配有大量的图表和图示。对 C＋＋语言描述的算法作了详细的注解和简要的性能分析，引入实例说明各种数据结构的具体应用。为使读者理解所设计的类的作用，书中加入了实际用例和测试代码。

3. 突出电子信息类专业特色

全书内容充分考虑了电子信息类专业的特点，很多实例直接选择相关专业较为基础的实际问题，尤其是信息与通信工程相关专业。因此本书既可作为电子信息类专业有针对性的数据结构教材，又可作为普通的数据结构教材使用。

4. 难易适中，启发性强

书中内容难易适中，在内容选择上不仅注重基础知识的阐述，而且兼顾重要算法的分

析。对于已超出本书讲述范围的非常复杂的问题只给出算法思想，同时引导读者阅读更深层次的资料。本书注重培养学生的思考能力和创新能力，对应各个知识点配以若干思考问题，启发读者进一步地分析和思考。

5. 联系实际，实用性强

书中的很多例题都结合实际问题，使读者容易理解。同时，书中还在各章最后加入了标准模板库(STL)的内容，使读者在理解数据结构的同时与 STL 联系，进而掌握 STL 的实现机制和应用，能够快速地利用 STL 编写程序。

本书共有 8 章和一个实验指导附录。前 4 章和附录由肖波编写，后 4 章由徐雅静编写。全书由肖波统稿。书中的所有 C++代码、PPT 等都可通过出版社网站下载。

本书的写作过程中，作者得到了同事及研究生的广泛支持和帮助。特别感谢蔺志青教授和赵衍运副教授对本书内容的总体把握和指导，并对各章内容提出很多重要的修改意见。感谢研究生周兵、赵娜同学帮助收集相关资料，并参与本书的校对工作。本书的写作还得到郭军教授的大力支持，并提出很多有益的建议，在此一并表示感谢。

由于作者水平有限，书中难免有错误和缺点，在此欢迎广大读者和同行专家多提宝贵意见和建议，对书中错误疏漏之处批评指正，可直接将意见发送至 xiaobo@bupt.edu.cn，作者将非常感谢。

作者

2010 年 6 月于北京邮电大学

目　录

第1章 绪 论

人们在进行程序设计时通常关注两个重要问题，一是如何将待处理的数据存储到计算机内存中，即数据表示；二是如何设计相应的算法操作这些数据，即数据处理。数据表示的本质是数据结构设计，数据处理的本质是算法设计，这两个问题往往是相互作用的，数据结构设计得好，处理算法效率就高。数据结构课程主要讨论数据的逻辑表示、存储方法和数据处理的基本问题，这些内容对于进行高效率的计算机程序开发非常重要。本章将系统地论述数据结构的基本概念、算法分析的基本方法，并对C++的标准模板类(STL)进行简要介绍。

1.1 数据结构的起源

数据结构是计算机及其相关专业的重要课程。随着计算机科学和技术的不断发展，众多的计算机系统软件和应用软件都要用到数据结构，如通信协议软件的开发、信息处理技术、数据挖掘等。数据结构起源于程序设计，并随着程序设计的发展而发展。

在计算机发展初期，人们使用计算机主要是处理数值计算问题。由于当时计算机的计算能力低下，所涉及的运算对象是简单的整数、实数及布尔型数据，所以程序设计者将主要精力集中于程序设计的技巧上，而不需要设计复杂的存储方法，因此无须重视数据结构。

在20世纪60～80年代，随着计算机应用领域的扩大和软硬件的发展，“非数值处理问题”显得越来越重要。如何表示数据成为程序设计的重要问题，数据结构及抽象数据类型被人们提出，从而使程序设计越来越规范。PASCAL语言之父，著名的瑞士计算机科学家沃思(Niklaus Wirth)教授曾提出：算法＋数据结构＝程序。从这里可以看出数据结构和算法是程序的两个重要组成部分，数据结构是指数据的逻辑结构和存储方法，而算法是指对数据的操作方法的描述。因此，通常将程序设计的本质看成是为实际问题选择一种合理的数据结构，加之设计一个好的算法。算法的优劣往往又取决于数据结构选择得

是否合适。

到了20世纪80年代初，面向对象(object oriented)技术出现，并且已成为目前最流行的程序设计技术。在面向对象技术中，问题世界中的相关实体被看成是一个对象，对象由属性和方法构成，属性描述了对象的特征或状态，方法用以改变对象的状态、操作对象的特征或描述对象的行为。一组具有相同属性和方法的对象的集合抽象为类，每个具体的对象都是类的一个实例。例如，"学生"是一个类，"张三"、"李四"等对象都是"学生"类的实例。

数据结构与面向对象具有天然的对应关系，如图1-1所示。数据结构主要研究两个方面的问题，即数据的表示及针对数据的基本操作。数据实质上是对对象实体状态和特性的描述；针对数据的操作构成对对象实体行为的描述。

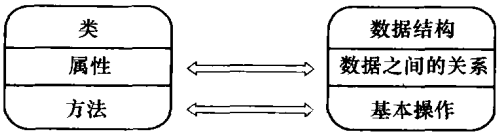

图1-1 数据结构与面向对象之间的关系

需要说明的是，数据结构的发展并未终结。一方面是因为程序设计仍然在发展，另一方面是因为不同的研究或应用领域都需要适合自身特点的数据结构，从而使数据结构得到长足的研究和发展，各种实用有效的高级数据结构被设计出来，各种特定的数据结构也在探索中。

1.2 数据结构的基本概念

为了便于理解数据结构的概念，先给出几个相关概念。

① 数据(data)：是信息的载体，能够被计算机识别、存储和加工处理。在计算机领域中，人们通常将数据分为两大类：一类是数值型数据，如代数方程求解程序中所使用的整数或实数数据；另一类是非数值型数据，如音视频播放器程序播放的声音或视频、互联网络中的Web数据等。

② 数据元素(data element)，是数据的基本单位，在计算机程序中通常作为一个整体进行处理。例如，学生成绩单中每个学生的信息就是一个数据元素。有些情况下，数据元素也称为元素、结点、顶点或记录。

③ 数据项(data item)，是构成数据元素的不可分割的最小单位。每个数据元素可以包含多个不同的数据项，每个数据项具有独立的含义。例如，学生成绩单中每个学生的信息可以包含学生的班级、学号、姓名、成绩等，这些都是数据项。有时也将数据项称为字段或域。

④ 数据类型(data type),是具有相同性质的计算机数据的集合以及在这个数据集合上的一组操作。数据类型可以分为简单类型(或称为原子类型)和构造类型(或称为结构类型)。例如,C++语言中,整数、实数、字符等都是简单的数据类型,而数组、结构类型、类等都是构造类型。每种类型的数据都有各自的特点及相关运算。

⑤ 数据结构(data structure),是指按照某种逻辑关系组织起来的一组数据,按一定的存储方式存储在计算机的存储器中,并在这些数据上定义了一组运算的集合。通常人们认为数据结构包含 3 个方面的内容:

- 数据元素之间的逻辑关系,也称为数据的逻辑结构(logical structure);
- 数据元素及其关系在计算机存储器内的存储形式,称为数据的存储结构(storage structure)或物理结构;
- 对数据的操作或运算。

数据的逻辑结构描述了数据相互间的关联形式或邻接形式,反映了数据内部的构成方式,定义了数据的本质特点,因此人们常常将数据的逻辑结构直接称为数据结构。数据的逻辑结构独立于计算机,与存储方式无关,可认为是从具体问题抽象出来的数学模型。数据元素之间不同的逻辑特点代表不同的逻辑结构,常见的逻辑结构如下。

① 集合,其数据元素之间的逻辑特点是满足"共同属于一个集合"的关系。通常要求集合中的元素不可重复。

② 线性结构,其数据元素之间的逻辑特点是有且只有一个起始结点和一个终端结点,并且其他结点的前面有且只有一个结点(称为直接前趋),每个结点的后面有且只有一个结点(称为直接后继)。本书第 2、3 章介绍的线性表、栈、队列及串都是线性结构,第 4 章介绍的多维数据和广义表可认为是广义的线性结构。

③ 树结构,其数据元素之间存在着一对多的层次关系。本书第 5 章介绍的树、二叉树等结构都是树结构。

④ 图结构,数据元素之间存在着多对多的关系。图结构将在本书第 6 章介绍。

为了与线性结构对应,通常将树结构和图结构称为非线性结构。图 1-2 给出了以上 4 种数据结构的示意图。

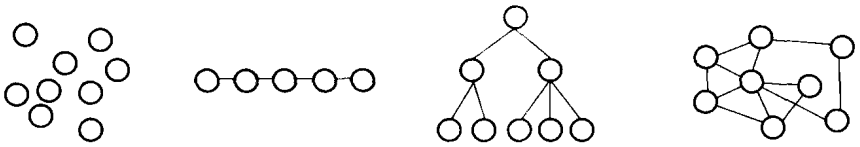

图 1-2　4 种数据结构示意图

数据的存储结构考虑的是如何在计算机的存储器中存储各个数据元素,并且同时反映数据元素间的逻辑关系。对于每种逻辑结构,都可以设计多种存储方法,基本的存储结

构通常有以下两大类。

① 顺序存储结构，即用一组连续的存储单元依次存储各个数据元素。数据元素间的逻辑关系由存储单元的邻接关系来体现。通常顺序存储结构借助于程序设计语言的数组来描述。例如，字母表(A,B,…,Z)的顺序存储示意图如图 1-3 所示。

② 链式存储结构，即用一组任意的存储单位存储各个数据元素，数据元素间的关系通常用指针来表示，例如，后一个元素的地址存储在前一个元素的某个特定数据项中。图 1-4 给出了链式存储结构的示意图。

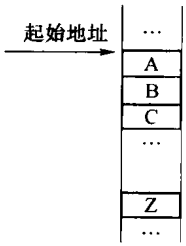

图 1-3　顺序存储示意图

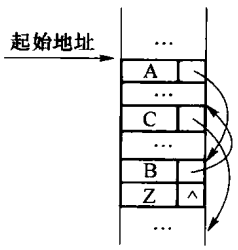

图 1-4　链式存储示意图

数据的逻辑结构反映了数据内部的逻辑关系，是面向实际问题的；而存储结构是面向计算机具体实现的，其目标是将数据及其逻辑关系存储到计算机中。仅有逻辑结构，只能确定对数据有哪些操作，而如何实现这些操作是不得而知的。因此，只有确定了数据的存储结构，才能设计对数据的具体操作算法。而且对于相同逻辑结构的同一种操作，如果采用不同的存储结构进行存储，对数据处理的效率往往也是不同的。因此，需要根据对数据的操作来设计合理的存储方式，以提高对其处理的效率。

在后续的章节中，每学习一种数据结构，将首先分析其逻辑结构及其相关操作。掌握了各种数据结构的逻辑特点，当分析待处理数据时可以根据数据特点来确定数据结构类型。分析了逻辑结构后，再学习其各种常见存储方式。对于每种存储结构，还要研究针对该存储结构的各种算法的实现，并对这些算法进行分析，使读者掌握不同存储方式下的各种操作或算法的优缺点，以便在实际问题中能够灵活应用。

1.3　算法和算法分析

数据的运算是通过算法(algorithm)描述的，讨论算法的效率和性能是数据结构课程的重要内容之一。

1.3.1　算法描述

算法就是解题的方法。从计算机处理的角度看，算法是由若干条指令组成的有穷序

列。通常一个问题可以有多种不同的算法，每个算法必须满足以下 5 个准则。

① 输入：具有 0 个或多个输入的参数。

② 输出：算法执行要有输出结果，不同的输入通常对应不同的输出。

③ 有穷性：算法中每条指令的执行次数必须是有限的，也就是说算法在执行了有穷步后能够结束。

④ 确定性：每条指令必须有确切的含义，无二义性。

⑤ 可行性：每条指令的执行时间都是有限的。

算法与程序的概念略有差别，程序可以不满足有穷性。例如，操作系统这种特殊的程序，或者其中的服务程序，只要系统不关闭或不遭破坏，它们就不会停止，而是无限循环地执行下去。一般地，将一个算法用计算机程序设计语言来编写后，便形成一个程序。

算法可以使用多种方法来描述，如自然语言、流程图、伪代码、程序设计语言或其他约定的语言。下面以欧几里得算法为例介绍算法的描述方法。欧几里得算法即用辗转相除法求两个自然数 m 和 n 的最大公约数（假定 $m \geqslant n$），算法的正确性不在这里讨论。

1. 自然语言

用自然语言描述算法最大的优点是容易理解，缺点是在描述时比较随意，容易产生二义性，并且描述往往比较冗长。欧几里得算法用自然语言描述如下：

① 输入 m 和 n；

② 取得 m 除以 n 的余数 r；

③ 若 $r=0$，则 n 为最大公约数，算法结束；否则执行第④步；

④ 将 n 放到 m 中，r 放到 n 中；

⑤ 重复执行第②步。

2. 流程图

用流程图描述算法直观易懂，是一种常用的方法。在计算机应用早期，使用流程图描述算法是主流的算法描述方法，然而实践证明，这种方法描述简单算法比较方便，但是对于复杂算法的描述则非常不方便。用流程图描述欧几里得算法如图 1-5 所示。

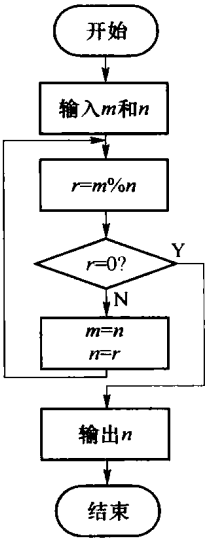

图 1-5　流程图描述算法

3. 伪代码

伪代码是介于自然语言和程序设计语言之间的方法。在描述算法时，它采用某种程序设计语言的基本语法、并且结合自然语言来设计。采用伪代码描述算法非常灵活，无须关注太多程序设计语言的语法限制，因此伪代码已成为算法描述的最常用方法之一，通常被称为算法语言。欧几里得算法采用符合 C/C++语言语法的伪代码描述如下：

```
[1]  input m,n
[2]  r=m%n;
[3]  while (r!=0)
  [3.1]  m=n;
  [3.2]  n=r;
  [3.3]  r=m%n;
[4]  output n;
```

4. 程序设计语言

用程序设计语言描述的算法可以直接通过计算机执行，但描述时通常需要考虑过多的算法具体细节以及程序设计语言的语法限制，从而使算法设计者不能将精力完全集中到算法的逻辑设计中。此外，还需要算法设计者掌握很多编程技巧。因此一般来说，不推荐采用程序设计语言直接描述算法。一般首先采用其他简单的方法将算法逻辑描述清楚，然后采用编程语言对算法进行设计。

欧几里得算法用C++语言书写的程序如下：

```
#include <iostream>
using namespace std;
int EUCLID (int m, int n){
    int r = m % n;
    while (r != 0){
        m = n;
        n = r;
        r = m % n;
    }
    return n;
}
void main(){
    cout << EUCLID (77, 55) << endl;
}
```

在采用面向对象的思想设计算法时，往往将算法看成是某个类的某种运算方法。例如，每个自然数都可以与其他自然数求解两者的最大公约数，因此这个求解运算可认为是自然数类的一种操作方法。如何采用面向对象的方法描述这种欧几里得算法呢？下面给出了简单的实例。

```
//定义自然数类
class NaturalNumber{
```

```
public:
    unsigned long int EUCLID(NaturalNumber & n);
                                                    //欧几里得算法求解最大公约数
    //...其他外部接口
private:
    unsigned long int num; //存储真正的自然数
};
//返回欧几里得算法求解最大公约数
unsigned long int NaturalNumber :: EUCLID(NaturalNumber & n)
{
    unsigned long int m = (num > n.num) ? num : n.num; //较大的自然数赋值给 m
    unsigned long int n = (num < n.num) ? num : n.num; //较小的自然数赋值给 n
    unsigned long int r = m % n;
    while (r != 0){
        m = n; n = r; r = m % n;
    }
    return n;
}
```

通过上面采用程序设计语言描述算法的两个例子不难看出，面向对象的程序设计与面向过程的设计方法是不同的。但在面向对象的设计中，具体到每个方法的设计往往离不开面向过程设计。

以上给出几种算法描述方法，通常认为伪代码最适合算法描述。它不是实际的编程语言，但在表达上又类似于编程语言，同时忽略了不必要的技术细节。本书中将采用基于C++语言的伪代码来描述算法。

1.3.2 算法分析

对于同一问题，往往有多种不同的求解算法。可以从多个角度评价这些算法的好坏，如算法的时间复杂度、空间复杂度、算法可读性等，其中最重要的就是算法的时间复杂度和空间复杂度。在本书中主要分析算法的时间特性，有时也会涉及空间特性的讨论。

1. 算法的时间复杂度

算法的时间复杂度是对算法执行时间的度量。一个算法的执行时间往往与算法本身、计算工具以及问题的规模等因素相关。在分析算法的时间复杂度时，可以忽略计算工具的因素，例如，同一个算法，在早期的286计算机和当前的酷睿双核计算机上运行的时间肯定是不同的。问题规模通常是指算法处理的数据量的大小，例如，同一个排序算法，对10个数排序和对10 000个数排序的用时是不同的。在分析算法时间复杂度时，要处