国家级特色专业“通信工程”系列教材

信息通信专业教材系列

数据结构与STL习题解析与实验指导

SHUJU JIEGOU YU STL
XITI JIEXI YU SHIYAN ZHIDAO

徐雅静　肖　波　编著

北京邮电大学出版社
www.buptpress.com

国家级特色专业“通信工程”系列教材
信息通信专业教材系列

数据结构与STL习题解析与实验指导

徐雅静　肖　波　编著

北京邮电大学出版社
www. buptpress. com

内 容 简 介

本书是《数据结构与 STL》的学习辅导教材，数据结构作为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，而本书的目的就是帮助读者更好地理解和掌握程序设计的思想和方法，提高应用数据结构的相关知识解决实际问题的能力。

本书分为 3 篇，第 1 篇是习题和讲解，按照《数据结构与 STL》教材的 1～8 章，分别撰写了每一章节的课程 MAP、扩展学习、课后习题讲解和练习题，并附综合试卷 5 套，方便读者对学习的程度进行自我考查；第 2 篇是实验，根据课程内容设计了 7 章对应的实验，并针对典型实验进行了讲解和实现；第 3 篇是课程设计，也是本书的特色，讲解了数据结构知识在实际问题上的应用方法和范例。

本书为读者学习数据结构及其相关知识、提高程序设计的能力提供了充足的内容，适合作为大学各专业数据结构课程的辅导书和实验教材，也可供程序爱好者自学使用。

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，打造信息与通信工程学科各专业的精品教材。

图书在版编目（CIP）数据

数据结构与 STL 习题解析与实验指导 / 徐雅静，肖波编著. -- 北京：北京邮电大学出版社，2015.3
ISBN 978-7-5635-4270-3

Ⅰ.①数… Ⅱ.①徐… ②肖… Ⅲ.①数据结构－高等学校－教学参考资料②C 语言－程序设计－高等学校－教学参考资料 Ⅳ.①TP311.12②TP312

中国版本图书馆 CIP 数据核字（2014）第 303813 号

书　　名：数据结构与 STL 习题解析与实验指导
著作责任者：徐雅静　肖　波　编著
责任编辑：刘　颖
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：北京鑫丰华彩印有限公司
开　　本：787 mm×960 mm　1/16
印　　张：18.5
字　　数：400 千字
版　　次：2015 年 3 月第 1 版　2015 年 3 月第 1 次印刷

ISBN 978-7-5635-4270-3　　定　价：38.00 元

前　言

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。

本书是《数据结构和 STL》的配套辅导教材，由于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，因此，本书在内容选择上不仅扩展了相关基础知识，详细分解了重要实例的数据结构和算法，还结合了电子信息类专业特点，对实验内容、课程设计内容都做了精心选择。撰写本书旨在对《数据结构和 STL》这本书进行有益的补充，有一定针对性地作为电子信息类专业的数据结构辅导教材。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. 扩展学习

本书第 1 篇重点是对《数据结构和 STL》书上的方法举一反三，从内容的深度和广度两个层面上进行扩展，使学习能力较强的学生能够不局限于课本内容，使之提高自己运用计算机算法解决问题的能力。

2. 分级实验设计

本书第 2 篇将实验按照难度分成基础实验、应用实验和扩展实验。可以根据自己的能力来选择适合的题目进行实验，并增加了典型实验的详细讲解和实现，有助于读者理解实验的内涵，独立完成实验的内容。

3. 实验设计与课程设计的衔接

本书第 2 篇实验中的扩展实验设计用来解决实际的问题，即实验也可以作为课程设计的一部分，方便读者对数据结构重点章节内容的扩展和理解；此外和第 3 篇的课程设计在内容设计上既有关联又有深度的扩展，使读者可以在深度和广度两个方面扩展数据结构知识的应用。

本书共分为3篇，第1篇前4章相关内容由肖波编写，后4章相关内容和试题习题由徐雅静编写；第2篇实验题目前4个实验相关内容由肖波编写，其余由徐雅静编写；第3篇课程设计1和2由肖波编写，课程设计3由徐雅静编写。全书由徐雅静统稿。书中的所有C++代码、试题答案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、别红霞教授、赵衍运副教授、胡佳妮副教授、马占宇副教授、黄平牧老师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。感谢研究生薛超、黎功福、王珊、袁彬、桂柯易同学帮助收集相关资料和习题答案验证，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 xyj@bupt.edu.cn，作者将非常感谢。

作　者

目　录

第 1 篇　习题与讲解

第 2 篇 实验题目与指导

第 3 篇　课程设计

第1篇

习题与讲解

第1章 绪　论

1.1 本章导学

1.1.1 知识点 MAP 图

本章的知识点 MAP 图如图 1.1 所示，其中第 3 层代表了本章学习的主线。

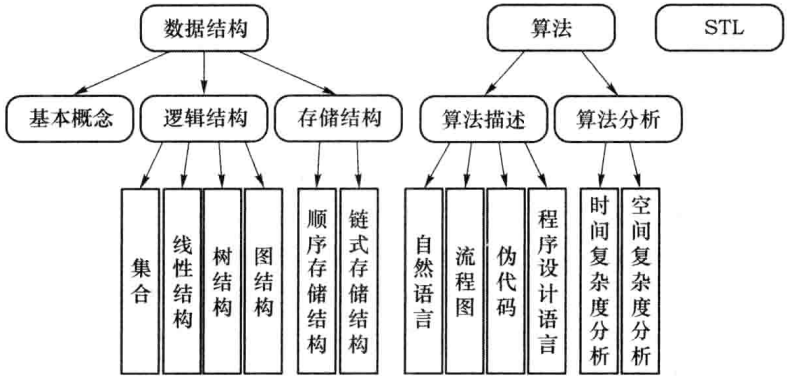

图 1.1　知识点 MAP 图

1.1.2 学习重点

本章介绍了数据结构的基本概念，研究内容，常见的逻辑结构和存储结构，算法的常见描述方法，算法的时间复杂度分析和空间复杂度分析，STL 基本概念。

本章的学习重点包括以下内容：

(1) 数据结构的研究内容；

(2) 常见逻辑结构的特点；

(3) 常见存储结构的特点；

(4) 算法描述方法；

(5) 算法时间复杂度分析方法。

本章的学习难点包括以下内容：

(1) 数据结构的研究内容；

(2) 算法的时间复杂度分析。

1.2 扩展学习

1.2.1 深入理解数据结构课程的学习内容

我们知道，现实生活中的数据是多种多样的，通过计算机对不同种类的数据进行操作前，首先考虑这些数据如何在计算机中进行存储。因为计算机中的存储空间是一维的，而现实中的数据是千变万化的。本质上说，数据结构的核心是研究各类数据如何在计算机的一维存储空间中进行存储。

数据在计算机中进行存储时，一方面要考虑数据自身的特点，另一方面还要考虑存储后进行操作的便利性。因此，首先需要根据其特点对现实生活中的数据进行分类，每一类具有一定的特点，符合这一类特点的数据便可采用相同的方法进行存储和处理。数据的特点主要表现在数据元素之间的关系上，也被称为数据的逻辑结构。不同特点的数据的逻辑结构是不同的，常见的逻辑结构有集合、线性结构、树结构、图结构等，它们的逻辑特点有明显的区别：

(1) 集合是指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，不考虑元素之间的先后次序。

(2) 线性结构中的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(称为直接前驱)，每个结点的后面有且只有一个结点(称为直接后继)。显然线性结构的数据元素之间在一维线性空间中有确定的先后次序。例如，我们常用的一维数组，就是一种线性结构。对于 $n(n\geqslant 2)$ 维数组，可认为是一种广义上的线性结构，每个结点是一个 $n-1$ 维的数组。

(3) 树结构中的数据元素之间存在着一对多的层次关系。类似于家族谱，每个结点只有一个父结点，但可以有多个孩子结点。显然，这种结构区别于集合和线性结构。

(4) 图结构中的数据元素之间存在着多对多的关系。每个数据元素与多个元素有连接关系。

我们理解了常见的四种逻辑结构的特点后，对于碰到的实际数据，就可以根据它们的特点判断属于或接近哪一种逻辑结构。

由于每一种逻辑结构具有自身明显的特点，因此在设计其存储方式时就要结合其自

身特点，并考虑到对其操作的便利性。其实每种逻辑结构都可以有多种存储方式，很难说哪种是最好的。常见的存储方式有顺序存储结构和链接存储结构，以及二者的结合。数据结构课程重点研究之一的就是常见的各种逻辑结构的不同存储方式。

数据的存储结构确定后，就可以使用编程语言进行描述，如C语言的结构类型、C++的类。同时对数据进行各种操作的算法就可以设计和实现。算法的设计也是数据结构课程的重要研究内容，算法设计的输出是使用某种描述语言，如自然语言、流程图、伪代码等，将算法的处理过程准确地描述出来即可。而算法的实现不是数据结构的主要内容，这是学习各类编程语言的目标，即把描述好的算法翻译成某种编程语言。教材中采用C++程序设计语言对算法进行实现。

通过以上的分析，用一句话概括，数据结构课程内容是学习各种逻辑结构的数据的常见存储结构以及相关操作算法。

1.2.2 算法的时间复杂度分析

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取决于多个因素，但评价算法的时间复杂度主要考虑问题的规模。问题规模通常指算法处理的数据量的大小。将问题规模设为 n，运行算法所需要的时间 T 可看作问题规模 n 的函数，记作 $T(n)$。

一个算法的执行时间，应该是该算法每条语句执行的时间之和。假定每条语句执行一次所需的时间是单位时间，则每条语句执行的时间正比于该语句执行的次数。通常将语句执行的次数称为该语句的频度(Frequency Count)。算法运行所需要的时间可认为是算法中所有语句的频度之和。若算法为多项式时间算法，则时间复杂度可表示为

$$T(n) = \sum_{k=0}^{p} c_k n^k = O(c_p n^p) = O(n^p)$$

也就是说，随着 n 的增大，$T(n)$的值主要取决于 n 的最高次幂，即 $T(n)$与 n 的最高次幂是同阶的。由于算法中语句的数量是确定的，因此若能选择一条关键语句，其执行次数与算法中所有语句的频度之和是同阶的，则可将关键句的执行次数作为整个算法的时间复杂度，从而简化了时间复杂度计算。

例 1.1 分析下面算法的时间复杂度。

```
i = 1;
while(i <= n)
   i *= 2;
```

解：

从关键语句 `i *= 2;`不难发现，每循环一次，i 增大为原来的 2 倍，设关键语句执行频度为 T，则有 $2^T = n$，因此，算法时间复杂度 $T(n) = \log_2 n$。

若算法调用递归函数，其时间复杂度分析较为复杂，通常将其时间复杂度的分析转化

为一个递归方程的求解渐进阶的问题。递归方程的形式有多种多样，因此求解方法也各有不同，比较常见的方法如：

(1) 代入法(Substitution Method)：先推测递归方程的显式解，然后用数学归纳法来验证该解是否合理。

(2) 迭代法(Iteration Method)：迭代地展开递归方程的右端，使之成为一个非递归的和式，然后通过对和式的估计来达到对方程左端即方程的解的估计。

(3) 套用公式法(Master Method)：该方法针对形如"$T(n)=a*T(n/b)+f(n)$"的递归方程。这种递归方程是分治法的时间复杂性所满足的递归关系，即一个规模为 n 的问题被分成规模均为 n/b 的 a 个子问题，递归地求解这 a 个子问题，然后通过对这 a 个子问题的解的综合，得到原问题的解。

(4) 差分方程法(Difference Formula Method)：可以将某些递归方程看成差分方程，通过解差分方程的方法来解递归方程，然后对解做出渐近阶估计。

在此不再对每种求解方法进行详细阐述，只给出一个简单的实例。

例 1.2 分析下面产生斐波那契序列的递归算法的时间复杂度。

```
int Fib(int n)
{
    if (n<=1)  return 1;
    else return Fib(n-1)+Fib(n-2);
}
```

解：

通过分析递归过程，不难发现 $T(n)=T(n-1)+T(n-2)$，显然这是一个二阶常系数齐次线性差分方程，将其写成标准形式为

$$T(n+2)-T(n+1)-T(n)=0$$

设 $T(n)=X^n$ 为对应齐次方程的一个解，代入上式，得

$$X^{n+2}-X^{n+1}-X^n=0$$

即有

$$X^2-X-1=0$$

求解该方程得：$X_{1,2}=\frac{1\pm\sqrt{5}}{2}$，因此方程的通解为

$$T(n)=A_1\left(\frac{1+\sqrt{5}}{2}\right)^n+A_2\left(\frac{1-\sqrt{5}}{2}\right)^n$$

又 $T(0)=1$，$T(1)=1$，分别代入上式，得 $A_1=\frac{1+\sqrt{5}}{2\sqrt{5}}$，$A_2=\frac{\sqrt{5}-1}{2\sqrt{5}}$，因此，有

$$T(n)=\frac{1+\sqrt{5}}{2\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n+\frac{\sqrt{5}-1}{2\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n$$

$$= \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^{n+1} - \left(\frac{1-\sqrt{5}}{2}\right)^{n+1}\right)$$

$$= \left(\frac{1+\sqrt{5}}{2}\right)^{n} + \left(\frac{1+\sqrt{5}}{2}\right)^{n-1}\left(\frac{1-\sqrt{5}}{2}\right) + \cdots + \left(\frac{1-\sqrt{5}}{2}\right)^{n}$$

$$= O\left(\left(\frac{1+\sqrt{5}}{2}\right)^{n}\right)$$

显然，算法的时间复杂度为指数分布，近似值为 $O(1.6^n)$。

例 1.3 分析下面产生斐波那契序列的非递归算法的时间复杂度。

```
int Fib(int n)
{
    int a = 1,b = 1;
    for (int i = 1; i<n; i++)
    {
        int tp = a + b;
        b = a;
        a = tp;
    }
    return a;
}
```

解：

对于斐波那契序列的非递归算法，显然只有一个循环，因此其时间复杂度为 $O(n)$，其效率远远高于斐波那契序列的递归算法。

通过比较例 1.2 和例 1.3 不难发现，对于同样的功能，递归函数的执行效率往往要比非递归算法的效率低很多。

1.2.3 异常处理机制

程序设计的要求之一就是程序的健壮性。希望程序在运行时能够不出或者少出问题。但是，在程序的实际运行时，总会有一些因素导致程序不能正常运行。

我们在设计算法时，往往对算法的正常逻辑处理流程设计得比较准确，对异常情况的处理反而不容易设计全面，导致程序在出现异常情况时崩溃。如果软件出现这种情况会给用户带来极不友好的体验。

举一个简单的例子，试设计一个程序，运行后提示用户输入两个整数，两个整数用空格隔开，用户输入后，程序显示出两个数字的和。这个程序正常的逻辑处理非常简单，但若用户输入的两个字符串不是整数，程序应该给出提示，否则程序有可能会崩溃。因此针对异常情况的处理也是非常重要的，当然有时这种处理会比较复杂。

异常(Exception)是程序在运行时可能出现的会导致程序运行终止的错误。这种错