计算机课程设计与综合实践规划教材

数据结构课程设计

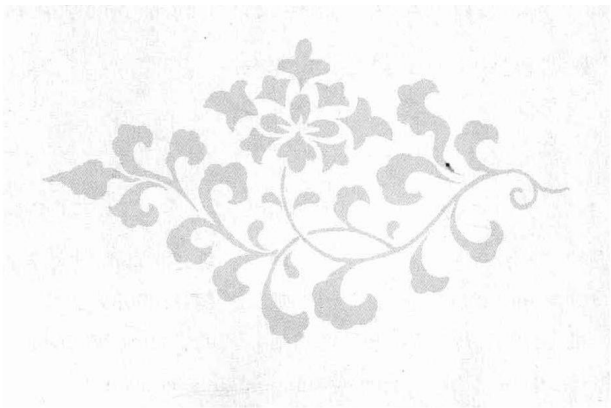
滕国文　编著

清华大学出版社

计 算 机 课 程 设 计 与 综 合 实 践 规 划 教 材

数据结构课程设计

滕国文　编著



清华大学出版社
北京

内 容 简 介

本书列举了数据结构课程设计实例，通过综合训练，能够培养学生实际分析问题、解决问题、编程和动手操作等多方面的能力，最终目的是帮助学生系统地掌握该门课程的基本内容，并运用所学的数据结构知识去解决实际问题。

全书共8章，内容包括数据库课程设计概述、线性表、栈、队列、串、多维数组和广义表、树状结构、图状结构等问题的应用。

本书是一本独立于具体的数据结构教材的课程设计辅导书，通过针对每种数据结构的具体实例，循序渐进地启发学生完成设计。书中给出的实例都是完整可运行的，同时给出了测试样例、总结与思考等，是一本很好的教学辅导参考书。

本书可作为高等院校计算机专业及相关专业教材或参考书，也可供从事软件开发工作和计算机编程爱好者参考。

图书在版编目（CIP）数据

数据结构课程设计/滕国文编著．—北京：清华大学出版社，2010.9
（计算机课程设计与综合实践规划教材）
ISBN 978-7-302-23241-4

Ⅰ. ①数…　Ⅱ. ①滕…　Ⅲ. ①数据结构－课程设计－高等学校－教材　Ⅳ. ①TP311.12

中国版本图书馆CIP数据核字(2010)第144354号

责任编辑：袁勤勇　赵晓宁
责任校对：焦丽丽
责任印制：李红英

出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦A座
http://www.tup.com.cn　　**邮　　编**：100084
社　总　机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62795954，jsjjc@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn

印 装 者：北京国马印刷厂
经　　销：全国新华书店
开　　本：185×260　　**印　张**：14.75　　**字　数**：335千字
版　　次：2010年9月第1版　　**印　次**：2010年9月第1次印刷
印　　数：1～3000
定　　价：25.00元

产品编号：034291-01

FOREWORD

前言

“数据结构”课程的教学目标是要求学生学会分析数据对象特征，掌握数据组织方法和计算机的表示方法，以便为应用所涉及的数据选择适当的逻辑结构、存储结构以及相应算法，初步掌握算法时间空间分析的技巧，培养良好的程序设计技能。

数据结构的学习过程是进行复杂程序设计的训练过程。技能培养的重要程度不亚于知识传授，学生不仅要理解授课内容，还应培养应用知识解答复杂问题的能力，形成良好的算法设计思想、方法技巧与风格，进行构造性思维，强化程序抽象能力和数据抽象能力。因此，学习数据结构，仅从书本上学习是不够的，必须经过大量的实践，在实践中体会构造性思维的方法，掌握数据组织与程序设计的技术。

在该课程的学习过程中，初学者会感到困惑，其主要原因：一是数据结构内容抽象；二是动态存储结构难以理解；三是使用多种技术，如递归技术等掌握较为困难；四是算法描述、设计无从下手等。

为了使学生更好地学习本课程，理解和掌握算法设计所需的技术，为整个专业的学习打好基础，本人根据学生的学习特点及自己二十多年的教学经验和总结，编写了本书，希望能给学生带来一些启发。

编写本书的出发点不是要给学生几个课程设计实例，而是希望通过一些典型的课程设计实例训练，使学生掌握如何利用数据结构知识去解决实际问题。

全书共分为 8 章。第 1 章是关于数据结构课程设计的概述；第 2～8 章按照一般教学顺序，分别给出线性表、栈、队列、串、多维数组和广义表、树状结构和图状结构的课程设计实例。

本书是在作者的“数据结构”讲义和指导学生的“课程设计大作业”基础上编写而成的。第 2～8 章的课程设计分别由宫耀勤、李闯、张伟、丛飚、逯洋、李淑梅和英昌盛完成修改或设计，2007 级学生王旭峰、杨名、张洋铭、袁洋、杨静、王珊珊和张群等参加了部分代码编写和程序调试，夏凤琴、刘艳玲、姚建盛、李颖、张桂杰、梁微、代胜男、罗琳、郝万萍和王金平等人进行了文稿的校对，最后由英昌盛对源程序统一整理，作者谨此一并致以诚挚的谢意。全书由滕国文教授统稿、审阅和整理后定稿。

在本书的编写过程中，作者参阅并借鉴了国内外诸多同行的文章和著作，这里不再一

一列举、标明，在此向他们致以谢意。

由于作者知识水平有限，时间仓促，本书难免有不足之处，恳请专家和读者批评指正。

滕国文

2010年4月

CONTENTS

目 录

CHAPTER 1

第1章 数据结构课程设计概述

1.1 数据结构简介

1. 数据结构课程的重要地位

数据结构是计算机理论与技术的重要基石，是计算机科学的核心课程之一。用计算机求解任何问题都离不开程序设计，而程序设计的实质是数据表示和数据处理。著名的瑞士计算机科学家沃思(N. Wirth)教授曾指出：算法＋数据结构＝程序。这里的数据结构是指数据的逻辑结构和存储结构，而算法则是对数据运算的描述。由此可见，程序设计的实质是对实际问题选择一种好的数据结构，再设计一个好的算法，而好的算法在很大程度上取决于描述实际问题的数据结构。数据结构不仅是一般程序设计的基础，而且是设计和实现操作系统、数据库及其他系统程序和大型应用程序的重要基础。

1968年，著名的美国算法大师克努特(D. E. Knuth)教授开创了“数据结构”的最初体系，他所著的《计算机程序设计艺术》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物理结构及其操作的著作。20世纪70年代初，“数据结构”作为一门独立的课程开始进入大学课堂。

数据结构是计算机科学与技术各专业的核心课程，它既是理论性较强的基础课，又是实践性很强的专业技术课，在计算机科学领域的主干课程中具有承上启下的作用。它的先行课程有计算机基础、程序设计语言、离散数学和数学等；后继课程有操作系统、数据库原理、编译原理和软件开发技术等。

“数据结构”的发展趋势包括两个方面：一方面是面向专门领域中特殊问题的数据结构的研究和发展，如图形数据结构、知识数据结构和空间数据结构；另一方面从抽象数据类型的角度出发，用面向对象的观点来讨论数据结构，已成为新的发展趋势。

2. 数据结构课程的教学目标

数据结构课程的教学目标是要求学生学会分析数据对象特征，掌握数据在计算机中的组织方法和表示方法，以便为应用所涉及的数据选择适当的逻辑结构、存储结构及相应算法，初步掌握算法分析的技巧，培养良好的程序设计技能。

人类解决问题的思维方式可分为两大类：一类是推理方式，凭借公理系统思维方法，从抽象公理体系出发，通过演绎、归纳、推理来求证结果，解决特定问题；另一类是算法方

式，凭借算法构造思维方式，从具体操作规范入手，通过操作过程的构造和实施解决特定问题。在开发一个优秀软件系统的全过程中，所凭借的思维方式本质上不同于常规数学训练的公理系统思维方式，而是一种算法构造性思维方式。系统开发是创造性思维过程的实现，因而对于一个开发人员，只知道开发工具的语言规则和简单使用过程是不够的，还需要有科学方法指导开发过程，以及在编程技术和应用技能上的不断积累和提高。让学生理解、熟悉、习惯这一套算法构造思维方式，是计算机软件课程教学的重要内容和主要难点。

学习数据结构对于培养人的抽象思维能力、数据建模能力、算法创新能力、程序设计能力、语言描述能力和综合应用能力等具有特定的作用。在信息社会高速发展的时代，信息素质是一个人适应信息社会生存和发展的最基本、最重要的素质之一。

3. 数据结构课程的学习特点

数据结构的学习特点主要表现在以下三个方面：

(1) 内容的广泛性。数据结构研究的问题非常广泛，内容极为丰富。1974 年，获图灵奖的克努特教授编写了一套巨著《计算机程序设计艺术》，目前已发行 4 卷，每卷 500～600 页，可见，数据结构研究的内容之多令人惊叹。

(2) 学科的交叉性。数据结构研究的内容包括计算机硬件范围的存储装置和存取方法；软件范围的文件系统、数据的动态管理、信息检索；数学范围的集合、逻辑学等方面的知识。此外，还有一些综合性知识，如数据类型、数据表示、数据运算、数据存取和程序设计方法等。因此，数据结构是由数学、计算机硬件和软件知识交叉形成的一门综合性学科。

(3) 知识的抽象性。由学科的交叉性可知，数据结构涉及诸多知识领域，这些知识本身就具有一定的抽象性，难度更大的是利用计算机解决实际问题时，必须将实际问题抽象成计算机能够接受并处理的数据模型才能实现，而数学建模不仅需要具备不同领域和学科的专业知识，更需要敏锐的洞察力、高度的抽象思维能力、独特的创新能力以及精湛的表达实现能力等。

数据结构的学习过程是进行复杂程序设计的训练过程。技能培养的重要程度不亚于知识传授，学生不仅要理解授课内容，还应培养应用知识解答复杂问题的能力，形成良好的算法设计思想、方法技巧与风格，进行构造性思维，强化程序抽象能力和数据抽象能力。因此，学习数据结构，仅从书本上学习是不够的，必须经过大量的实践，在实践中体会构造性思维方法，掌握数据组织与程序设计的技术。

在学习中注重广泛阅读，加深理解，把书本越读越厚；再通过归纳总结，提纲挈领，把书本越读越薄。

1.2 课程设计目标和特点

1. 课程设计目标

在数据结构课程的学习过程中，初学者会感到困惑，其主要原因：一是数据结构内容抽象；二是动态存储结构难以理解；三是使用多种技术，如递归技术等掌握较为困难；四是

算法描述、设计无从下手等。

为了使学生更好地学习本课程，理解和掌握算法设计所需的技术，为整个专业的学习打好基础，作者根据学生的学习特点及自己二十多年的教学经验和总结，编写了本书。该书针对每种数据结构都给出了由简单到复杂的课程设计实例，目的是通过课程设计的综合训练，培养学生实际分析问题、解决问题、编写程序和动手操作的能力，最终通过课程设计的形式，帮助学生系统掌握本课程的主要内容，具有较强的程序设计能力。

编写本书的出发点不是要给学生几个具体的课程设计实例，而是希望通过一些典型的课程设计实例训练，给学生一些示范和启发，调动学生学习的积极性，扩展其思维和想象力，最终使学生掌握利用数据结构知识去解决实际问题的能力。

2. 课程设计特点

课程设计是学习数据结构与算法的一个重要环节，学生通过课程设计的综合训练，在学习理论知识的同时进一步提高解决实际问题的能力，强化综合应用能力，扩充知识，开阔视野，同时熟练掌握利用计算机解决问题的一般步骤。

本书课程设计的主要特点如下：

(1) 这是一本独立于具体的数据结构教材的课程设计辅导书，是用于指导学生完成“数据结构课程设计”大作业的理想教材。

(2) 通过针对每种数据结构的具体实例，循序渐进地启发学生完成设计。每个课程设计实例都从提出问题、设计要求，到选择使用的数据结构、问题的分析与实现，最后给出完整可运行的源程序，同时给出了测试样例。

(3) 每个课程设计实例的总结与思考是该课程设计的拓展部分。学生可以在具体实例的基础上，根据指导自己去开发设计，举一反三，真正培养学生的实践能力。

(4) 对于较大的课程设计实例，可以将其划分为几个子项目，多个学生分工合作共同完成，以培养学生的团队合作精神。

1.3 编写说明

1. 各章的基本结构

第 1 章是关于数据结构课程设计的概述。从第 2～8 章分别对各种数据结构进行课程设计，其基本结构分为两部分：

第一部分：给出每种数据结构的存储结构及 C 语言描述，并存放于 *.h 文件中；并在相应的存储结构基础上，给出常用基本运算的算法，并存放于 *.c 文件中。

第二部分：对应每种数据结构给出具有代表性的综合应用的课程设计实例 3～5 个，统一按照 1.4 节所讲的格式书写。每个课程设计实例源文件的命名规则为 zj*.c，其中数字 z 代表第几章，数字 j 代表第几节，* 代表实例的名称。例如，24differ.c 表示第 2 章第 4 节课程设计实例源文件，其名称是 differ，代表多项式求导。并将所有课程设计实例源文件都附在随书赠送的光盘里，方便读者学习使用。

2. 课程设计实例的程序编排模板

本书给出的课程设计实例程序均在 Turbo C2.0、WIN-TC1.9、C Free4.1 和 Visual C++ 6.0等软件开发环境下调试运行通过。

本书中算法描述采用的是标准的 Turbo C 函数，当需要在计算机上完整地实现算法时，必须设计构造一个完整的可以执行的源程序，为此给出 C 语言实现课程设计的程序编排模板。

模板的基本结构如下：

(1) 包含必要的标准头文件和通用的常量定义。如标准输入输出头文件 stdio.h 等；OK、ERROR 等通用的常量定义，本书将之统一定义在 consts.h 头文件中，读者使用时只需包含该文件。

consts.h 头文件的内容如下：

```
#include<string.h>
#include<malloc.h>                    /*malloc()等*/
#include<limits.h>                    /*INT_MAX等*/
#include<stdio.h>                     /*EOF(=^Z或F6),NULL*/
#include<stdlib.h>                    /*atoi()*/
#include<io.h>                        /*eof()*/
#include<math.h>                      /*floor(),ceil(),abs()*/
#include<process.h>                   /*exit()*/
/*函数结果状态代码*/
#define TRUE 1
#define FALSE 0
#define OK 1
#define ERROR -1
#define INFEASIBLE -1
```

(2) 将某一数据结构所对应的存储结构的描述存放在一个头文件 *.h 中，将某一数据结构所对应的基本操作算法存放在一个 C 文件 *.c 中。例如，将单链表存储结构的 C 语言描述存放在 linklist.h 中，将单链表的基本操作算法存放在 linklist.c 中，需要时通过文件包含 #include "linklist.h" 和 #include "linklist.c"，可以实现对其中数据类型的引用及有关操作函数的调用。

(3) 编写基于某种数据结构的具体问题的算法。

(4) 编写主函数，其中进行合理的函数调用，形成一个可执行程序。

1.4 课程设计实例的标准格式

本书所给出的每个课程设计实例都包括以下 6 个部分：

(1) 问题描述。给出课程设计的具体内容，要求表述简单、准确。

(2) 设计要求。指出该课程设计所要达到的基本要求和所需满足的约束条件。实现

时，在完成基本要求的情况下可以适当扩展课程设计的功能。

(3) 数据结构。该课程设计需要使用的数据结构有哪些(指出使用的数据结构和存储结构)。

(4) 分析与实现。分析该课程设计的实现方法，数据结构的使用和算法的设计。给出算法实现的详细代码，要求有必要的注释，提高程序的可读性。

(5) 运行与测试。给出有代表性的测试用例，并加以简单的文字说明，注意程序运行要覆盖算法的各种情况。

(6) 总结与思考。主要指出算法的特点，在实现该课程设计基本要求的前提下，还可以进行哪些方面的功能扩展，特别是重点说明独创的部分，相关课程设计项目最有价值的内容，在哪些方面需要进一步了解或得到帮助，以及编程实现课程设计的感悟等内容。

CHAPTER 2

第2章 线性表的应用

2.1 存储结构与基本运算的算法

1. 顺序表

采用顺序存储结构的线性表简称为顺序表。

(1) 顺序表的C语言描述。

顺序表可借助于高级程序设计语言中的一维数组来表示，一维数组的下标与数据元素在线性表中的序号相对应，但不是直接的一一对应关系，因为C语言中数组的下标是从0开始。

顺序表的C语言描述如下(存放于seqlist.h文件中)：

```
#include "consts.h"
#define MAXNUM 100                  /*表示线性表可能达到的最大长度*/
typedef int DataType;               /*数据元素类型定义*/
typedef struct
{
    DataType data[MAXNUM];          /*存放线性表的数据元素*/
    int last;                       /*用来存放线性表最后一个数据元素在数组中的下标*/
}SeqList;
```

(2) 基本运算的算法如下(存放于seqlist.c文件中)：

① 置空表。

```
void SeqLSetNull(SeqList *l)
{
    l->last=-1;
}
```

② 求表的长度。

```
int SeqLLength(SeqList *l)
{
    return l->last+1;
}
```

③ 取结点。

```
DataType SeqLGet(SeqList * l,int i)
{
    if(i<1||i>l->last+1 )
    {
        printf("\t i 的位置不正确\n");
        return ERROR;
    }
    return l->data[i-1];
}
```

④ 定位运算。

```
int SeqLLocate(SeqList * l,DataType x)
{
    int i;
    for(i=0;i<=l->last;i++)
        if(l->data[i]==x) return(i+1);
    return 0;
}
```

⑤ 插入运算。

```
int SeqLInsert(SeqList * l,int i,DataType x)
{
    int j;
    if(l->last>=MAXNUM-1)
    {
        printf("\t 溢出\n");
        return ERROR;
    }
    if(i<1||i>l->last+2)
    {
        printf("\t 插入位置不正确 \n");
        return ERROR;
    }
    else
    {
        for(j=l->last;j>=i-1;j--)
            l->data[j+1]=l->data[j];
            l->data[i-1]=x;
            l->last++;
    }
    return OK;
}
```

⑥ 删除运算。

```
int SeqLDelete(SeqList * l,int i)
{
    int j;
    if(i<1||i>l->last+1)
    {
        printf("\t 删除位置不正确 \n");
        return ERROR;
    }
    else
    {
        for(j=i;j<=l->last;j++)
            l->data[j-1]=l->data[j];
            l->last--;
    }
    return OK;
}
```

⑦ 建立顺序表。

```
void SeqLCreate(SeqList * l)
{
    int i,n;
    printf("\t 请输入表的长度:");
    scanf("%d",&n);
    l->last=n-1;
    printf("\t 依次输入表中的数据元素(整数):\n");
    for(i=0;i<n;i++)
    {
        printf("\t 第%d 个元素是:",i+1);
        scanf("%d",&l->data[i]);
    }
}
```

⑧ 输出顺序表。

```
void SeqLPrint(SeqList * l)
{
    int j;
    if(l->last<0)
    {
        printf("\t 表空!\n");
        exit(0);
    }
    else
```

```
    {
        printf("\n表的数据元素如下:\n( ");
        for(j=0;j<=l->last;j++)
        printf("%5d,",l->data[j]);
        printf("\b )\n");
    }
}
```

⑨ 顺序表运算的综合实例。

存放于21mainseqlist.c文件中。

```
#include "seqlist.h"
#include "seqlist.c"
int main(int argc,char * argv[])
{
    DataType y;
    SeqList * a,x;
    int m,t,read=0 ;
    a=&x;
    do
    {
        puts("          关于顺序表的操作\n");
        puts("          ========================\n");
        puts("          1 ------置空表");
        puts("          2 ------建表");
        puts("          3 ------求表长");
        puts("          4 ------取结点");
        puts("          5 ------定位");
        puts("          6 ------插入");
        puts("          7 ------删除");
        puts("          8 ------输出");
        puts("          0 ------退出");
        printf("          请选择代号(0—8):");
        scanf("%d",&read);
        printf("\n");
        switch(read)
        {
            case 1: SeqLSetNull(a);break;
            case 2: SeqLCreate(a);break;
            case 3: printf("\t表的长度是: %d\n",SeqLLength(a));break;
            case 4: printf("\t取结点的位置是: ");
                scanf("%d",&m);
                y=SeqLGet(a,m);
                if(y)
```

```
            printf("\t第%d个结点是%d\n",m ,y);break;
        case 5: printf("\t定位的数据元素是: ");
            scanf("%d",&y);
            t=SeqLLocate(a,y);
            if(t)
                printf("\t定位数据元素的位置是: %d\n",t);break;
        case 6: printf("\t插入数据元素是: ");
                scanf("%d",&y);
                printf("\t插入位置是: ");
                scanf("%d",&m);
                t=SeqLInsert(a,m,y);
                if(t)
                    printf("\t插入后表的数据元素是:\n");
                    SeqLPrint(a);break;
        case 7: printf("\t删除位置是: ");
                scanf("%d",&m);
                t=SeqLDelete(a,m);
                if(t)
                    printf("\t删除后表的数据元素是:\n");
                SeqLPrint(a);break;
        case 8: SeqLPrint(a);break;
        case 0: read=0 ;
      }
   }while(read!=0 );
   return 0;
}
```

2. 链表

采用链式存储结构的线性表简称链表。从链接方式的角度看，链表可分为单链表、单循环链表、双链表和双循环链表；从实现角度看，链表可分为动态链表和静态链表。下面以动态单链表(简称单链表)为例介绍。

(1) 单链表的C语言描述。

单链表的结点包括两个域。数据域(data)：用来存放结点的值，即存储数据元素；指针域(next)：用来存储该数据元素直接后继的地址(或位置)。

用C语言定义单链表如下(存放于linklist.h文件中)：

```
typedef struct node                    /*结点类型定义*/
{
    DataType data;
    struct node *next;
}LinkedList;
```