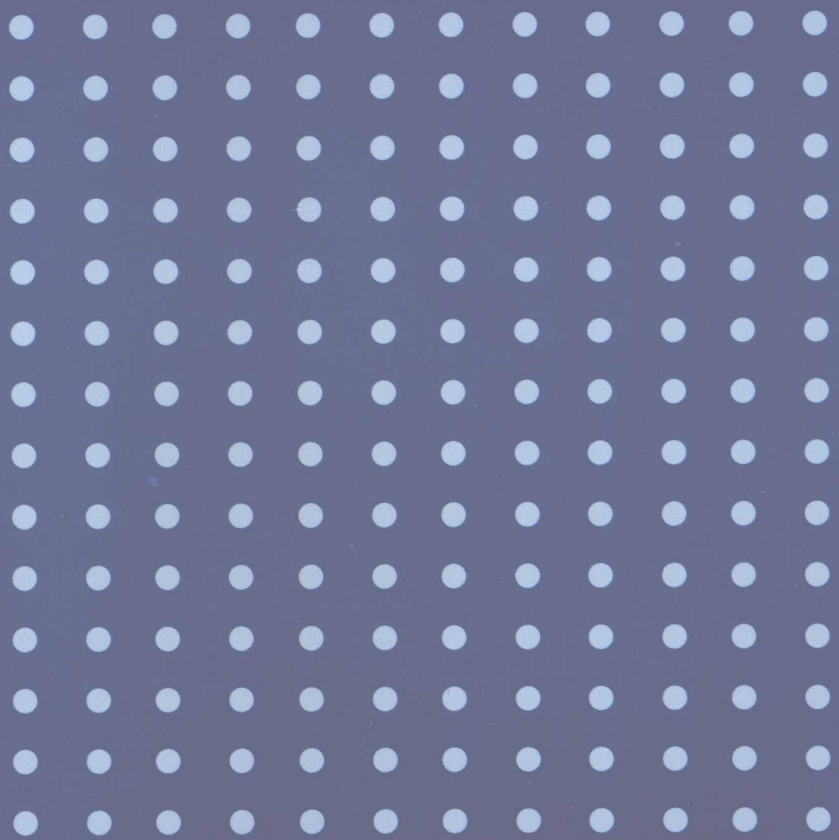
重点大学计算机专业系列教材

数据结构教程(第3版)上机实验指导

李春葆 尹为民 李蓉蓉 蒋晶珏 喻丹丹 安杨 编著

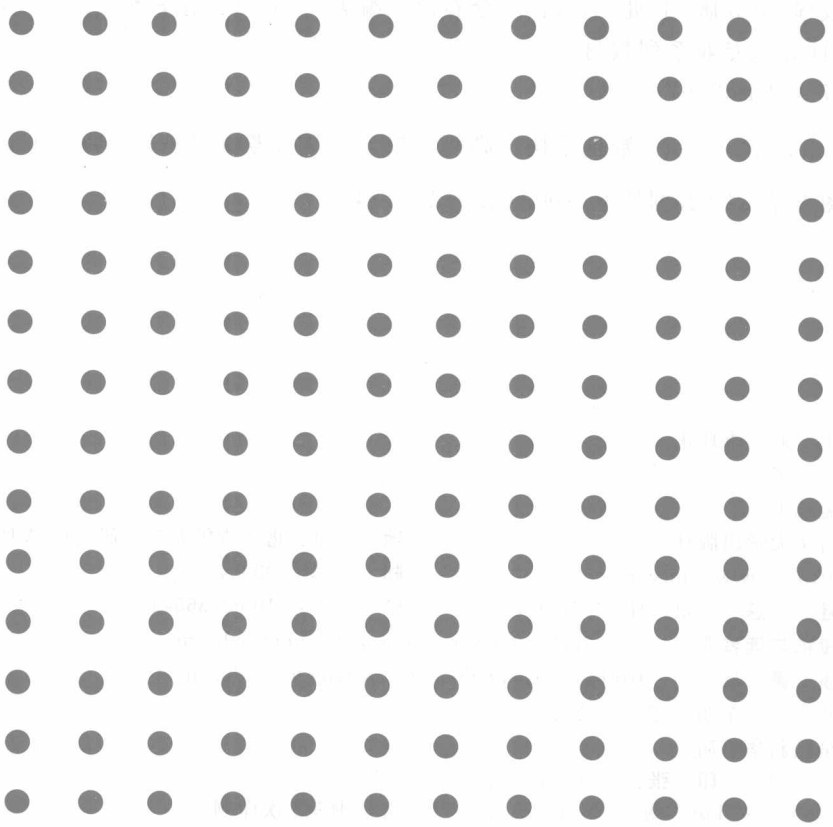


清华大学出版社

重点大学计算机专业系列教材

数据结构教程(第3版)上机实验指导

李春葆 尹为民 李蓉蓉 蒋晶珏 喻丹丹 安杨 编著



清华大学出版社
北京

内 容 简 介

本书是《数据结构教程(第3版)》(李春葆编著,清华大学出版社出版)的配套上机实验指导书。两书章次一一对应,内容包括绪论、线性表、栈和队列、串、数组和稀疏矩阵、递归、树形结构、图、查找、内排序、外排序、文件和综合实验题解析。书后附录中给出了VC++6.0环境下编写C程序所需要的基本知识及学生提交的实验报告格式。书中所有程序都在VC++6.0环境下调试通过,读者可以从http://www.tup.com.cn网站免费下载。书中列出了全部的上机实验题目,因此自成一体,可以脱离主教材单独使用。

本书适合高等院校计算机及相关专业本科生及研究生使用。

图书在版编目(CIP)数据

数据结构教程(第3版)上机实验指导/李春葆等编著. —北京:清华大学出版社,2009.3
(重点大学计算机专业系列教材)
ISBN 978-7-302-19380-7

Ⅰ.数… Ⅱ.李… Ⅲ.数据结构—高等学校—教学参考资料 Ⅳ.TP311.12

中国版本图书馆CIP数据核字(2009)第012400号

责任编辑:丁 岭 李玮琪
责任校对:梁 毅
责任印制:孟凡玉
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:010-62776969,c-service@tup.tsinghua.edu.cn
质 量 反 馈:010-62772015,zhiliang@tup.tsinghua.edu.cn
印 装 者:北京鑫海金澳胶印有限公司
经 销:全国新华书店
开 本:185×260 印 张:17.5 字 数:429千字
版 次:2009年3月第1版 印 次:2009年3月第1次印刷
印 数:1~4000
定 价:26.00元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题,请与清华大学出版社出版部联系调换。联系电话:(010)62770177转3103 产品编号:031989-01

INTRODUCTION

出版说明

随着国家信息化步伐的加快和高等教育规模的扩大，社会对计算机专业人才的需求不仅体现在数量的增加上，而且体现在质量要求的提高上，培养具有研究和实践能力的高层次的计算机专业人才已成为许多重点大学计算机专业教育的主要目标。目前，我国共有16个国家重点学科、20个博士点一级学科、28个博士点二级学科集中在教育部部属重点大学，这些高校在计算机教学和科研方面具有一定优势，并且大多以国际著名大学计算机教育为参照系，具有系统完善的教学课程体系、教学实验体系、教学质量保证体系和人才培养评估体系等综合体系，形成了培养一流人才的教学和科研环境。

重点大学计算机学科的教学与科研氛围是培养一流计算机人才的基础，其中专业教材的使用和建设则是这种氛围的重要组成部分，一批具有学科方向特色优势的计算机专业教材作为各重点大学的重点建设工程成果得到肯定。为了展示和发扬各重点大学在计算机专业教育上的优势，特别是专业教材建设上的优势，同时配合各重点大学的计算机学科建设和专业课程教学需要，在教育部相关教学指导委员会专家的建议和各重点大学的大力支持下，清华大学出版社规划并出版本系列教材。本系列教材的建设旨在“汇聚学科精英、引领学科建设、培育专业英才”，同时以教材示范各重点大学的优秀教学理念、教学方法、教学手段和教学内容等。

本系列教材在规划过程中体现了如下一些基本组织原则和特点。

1. 面向学科发展的前沿，适应当前社会对计算机专业高级人才的培养需求。教材内容以基本理论为基础，反映基本理论和原理的综合应用，重视实践和应用环节。

2. 反映教学需要，促进教学发展。教材要能适应多样化的教学需要，正确把握教学内容和课程体系的改革方向。在选择教材内容和编写体系时注意体现素质教育、创新能力与实践能力的培养，为学生知识、能力、素质协调发展创造条件。

3. 实施精品战略，突出重点，保证质量。规划教材建设的重点依然是专业基础课和专业主干课；特别注意选择并安排了一部分原来基础比较好的优秀教材或讲义修订再版，逐步形成精品教材；提倡并鼓励编写体现重点大学

计算机专业教学内容和课程体系改革成果的教材。

4. 主张一纲多本,合理配套。专业基础课和专业主干课教材要配套,同一门课程可以有多本具有不同内容特点的教材。处理好教材统一性与多样化的关系;基本教材与辅助教材以及教学参考书的关系;文字教材与软件教材的关系,实现教材系列资源配套。

5. 依靠专家,择优落实。在制订教材规划时要依靠各课程专家在调查研究本课程教材建设现状的基础上提出规划选题。在落实主编人选时,要引入竞争机制,通过申报、评审确定主编。书稿完成后要认真实行审稿程序,确保出书质量。

繁荣教材出版事业,提高教材质量的关键是教师。建立一支高水平的以老带新的教材编写队伍才能保证教材的编写质量,希望有志于教材建设的教师能够加入到我们的编写队伍中来。

教材编委会

FOREWORD

前言

本书是《数据结构教程(第3版)》(李春葆等编著,清华大学出版社出版,以下简称为《教程》)的配套上机实验指导书。

全书分为13章,第1章为绪论——上机实验题1解析;第2章为线性表——上机实验题2解析;第3章为栈和队列——上机实验题3解析;第4章为串——上机实验题4解析;第5章为数组和稀疏矩阵——上机实验题5解析;第6章为递归——上机实验题6解析;第7章为树形结构——上机实验题7解析;第8章为图——上机实验题8解析;第9章为查找——上机实验题9解析;第10章为内排序——上机实验题10解析;第11章为外排序——上机实验题11解析;第12章为文件——上机实验题12解析;第13章为综合实验题解析。各章次与《教程》的章次相对应。

另外,书后给出了两个附录,附录A为使用VC++6.0系统,较系统地给出在VC++6.0环境下编写C程序所需要的基本知识,附录B为学生应提交的实验报告格式。

书中所有程序都在VC++6.0环境下调试通过,读者可以从http://www.tup.com.cn网站免费下载。

书中列出了全部的上机实验题目,因此自成一体,可以脱离《教程》单独使用。

由于水平所限,尽管编者不遗余力,仍可能存在错误和不足之处,敬请教师和同学们批评指正。

编　者

2008.10.30

CONTENTS

目录

CHAPTER 1

绪论——上机实验题 1 解析 第 1 章

实验题 1.1 求素数

设计一个程序，输出所有小于等于 n（n 为一个大于 2 的正整数）的素数。要求：(1)每行输出 10 个素数；(2)尽可能采用较优的算法。

解：本工程 proj1_1 的组成结构如图 1.1 所示。本程序的模块结构图如图 1.2 所示，图中方框表示函数，方框中指出函数名，箭头方向表示函数间的调用关系，虚线方框表示文件的组成，即指出该虚线方框中的函数存放在哪个文件中。

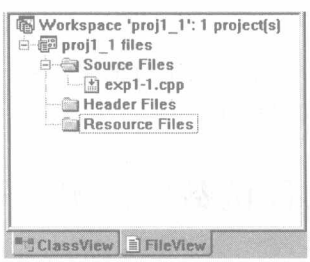


图 1.1 proj1_1 工程组成

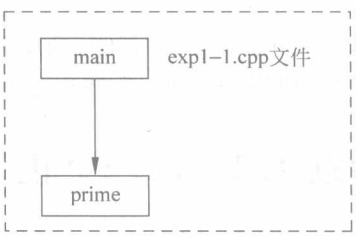


图 1.2 proj1_1 工程的程序结构图

exp1-1.cpp 文件包含如下函数。

- Prime(x)：其功能是判断正整数 x 是否为素数。采用的方法是，若 x 是素数，则 x 不能被 $2\sim\sqrt{x}$ 的任何整数整除。

对应的程序如下(设计思路详见代码中的注释)：

```
//文件名：exp1 - 1.cpp
#include <stdio.h>
#include <math.h>
int prime(int x)                                    //判断正整数 x 是否为素数
{
    int i;
    for (i = 2;i<(int)sqrt(x);i++)
```

```
        if (x % i == 0) return 0;                    //若 x 不是素数,则退出并返回 0
    return 1;
}
void main()
{
    int n,i,j = 0;                                   //j 用于累计素数个数
    printf("n:");
    scanf(" % d",&n);
    printf("小于等于 % d 的素数:\n",n);
    if (n>2)
    {   printf(" % 4d",2);
        j++;
    }
    for (i = 3;i< = n;i + = 2)
        if (prime(i) == 1)
        {   printf(" % 4d",i);
            if (j! = 0 && ++j % 10 == 0)             //每行最多显示 10 个素数
                printf("\n");
        }
    printf("\n");
}
```

连编本工程生成可执行文件 proj1_1.exe。程序的一次执行结果如下:

```
n:100
小于等于100的素数:
   2   3   5   7   9  11  13  15  17  19
  23  25  29  31  35  37  41  43  47  49
  53  59  61  67  71  73  79  83  89  97
```

对于 prime(x),其时间复杂度为 $O(\sqrt{x})$,由于偶数不可能是素数,所以程序中只对奇数进行素数的判断。因此,上述程序的时间复杂度较低。

实验题 1.2　求一个正整数的各位数字之和

编写一个程序,计算任一输入的正整数的各位数字之和,并分析算法的时间复杂度。

解:本工程 proj1_2 的组成结构如图 1.3 所示。本程序的模块结构图如图 1.4 所示,图中方框表示函数,方框中指出函数名,箭头方向表示函数间的调用关系,虚线方框表示文件的组成,即指出该虚线方框中的函数存放在哪个文件中。

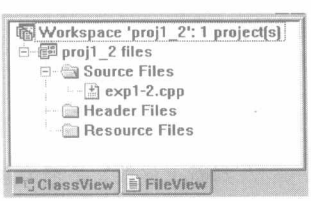


图 1.3　proj1_2 工程组成

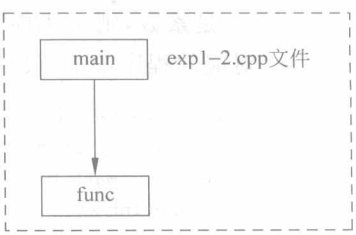


图 1.4　proj1_2 工程的程序结构图

exp1-2.cpp 文件包含如下函数。

- int func(int num)：分解 num 的各位数字，返回这些数字之和。采用的方法是对 num 边分解边进行累加，直到分解完毕。

对应的程序如下(设计思路详见代码中的注释)：

```
//文件名：exp1 - 2.cpp
#include <stdio.h>
int func(int num)          //分解 num 的各位数字,返回其和
{
    int s = 0;
    do
    {   s + = num % 10;    //累计各位数字之和
        num/ = 10;         //求下一数值位
    } while(num);
    return(s);
}
void main()
{
    int n;
    printf("\n");
    printf("输入一个整数:");
    scanf("%d",&n);
    printf("各位数字之和:%d\n",func(n));
    printf("\n");
}
```

连编本工程生成可执行文件 proj1_2.exe。程序的一次执行结果如下：

```
输入一个整数:12345678
各位数字之和:36
```

func(n)的时间复杂度为 $O(\text{len}(n))$，其中 len(n)为正整数 n 的位数。程序的时间复杂度亦为 $O(\text{len}(n))$。

实验题 1.3　求一个字符串是否为回文

编写一个程序，判断一个字符串是否为“回文”(顺读和倒读都一样的字符串称为“回文”)。并分析算法的时间复杂度。

解：本工程 proj1_3 的组成结构如图 1.5 所示。本程序的模块结构图如图 1.6 所示，图中方框表示函数，方框中指出函数名，箭头方向表示函数间的调用关系，虚线方框表示文件的组成，即指出该虚线方框中的函数存放在哪个文件中。

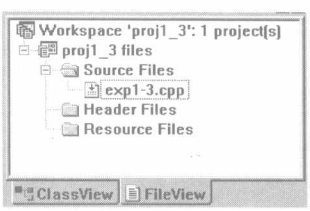


图 1.5　proj1_3 工程组成

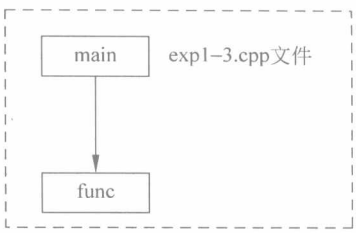


图 1.6　proj1_3 工程的程序结构图

exp1-3.cpp 文件包含如下函数。

- func(s)：判断串 s 是否为回文。采用的方法是：用 flag 表示是否为回文，其初值为 1。用 i 从左向右扫描字符串 s，用 j 从右向左扫描字符串 s，若 s[i]与 s[j]不相等，则 flag=0(表示不是回文)并退出循环，否则，继续比较直到 $i<j$ 不成立。

对应的程序如下(设计思路详见代码中的注释)：

```
//文件名：exp1-3.cpp
#include <stdio.h>
#include <string.h>
#define MAX 100                                  //字符串的最大长度
int func(char s[])                               //判断串 s 是否为回文
{
    int flag=1;
    int i,j,slen=strlen(s);                      //slen 为字符串 s 的长度
    for (i=0,j=slen-1;i<j;i++,j--)
        if (s[i]!=s[j])
        {   flag=0;
            break;
        }
    return(flag);
}
void main()
{
    char s[MAX];
    printf("输入一字符串:");
    scanf("%s",s);
    if (func(s)==1)
        printf("%s字符串是回文\n",s);
    else
        printf("%s字符串不是回文\n",s);
}
```

连编本工程生成可执行文件 proj1_3.exe。程序的一次执行结果如下：

```
输入一字符串:abcdcba
abcdcba字符串是回文
```

在 func(s)算法中，for 循环语句的执行次数为 $\frac{n}{2}$(n 为字符串 s 的长度)，则它的时间复杂度为 $O(n)$。程序的时间复杂度亦为 $O(n)$。

CHAPTER 2

线性表——上机实验题 2 解析 第 2 章

实验题 2.1 实现顺序表各种基本运算的算法

编写一个程序 algo2-1.cpp，实现顺序表的各种基本运算，并在此基础上设计一个主程序完成如下功能：

(1) 初始化顺序表 L。

(2) 依次采用尾插法插入 a,b,c,d,e 元素。

(3) 输出顺序表 L。

(4) 输出顺序表 L 长度。

(5) 判断顺序表 L 是否为空。

(6) 输出顺序表 L 的第 3 个元素。

(7) 输出元素 a 的位置。

(8) 在第 4 个元素位置上插入 f 元素。

(9) 输出顺序表 L。

(10) 删除 L 的第 3 个元素。

(11) 输出顺序表 L。

(12) 释放顺序表 L。

解：本工程 proj2_1 的组成结构如图 2.1 所示。本程序的模块结构图如图 2.2 所示，图中方框表示函数，方框中指出函数名，箭头方向表示函数间的调用关系，虚线方框表示文件的组成，即指出该虚线方框中的函数存放在哪个文件中。

根据《教程》中 2.2 节的算法得到 algo2-1.cpp 文件，其包含如下函数。

- InitList(SqList *&L)：初始化顺序表 L。
- DestroyList(SqList *L)：释放顺序表 L。
- ListEmpty(SqList *L)：判断顺序表 L 是否为空表。

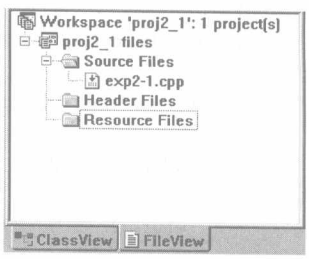


图 2.1 proj2_1 工程组成

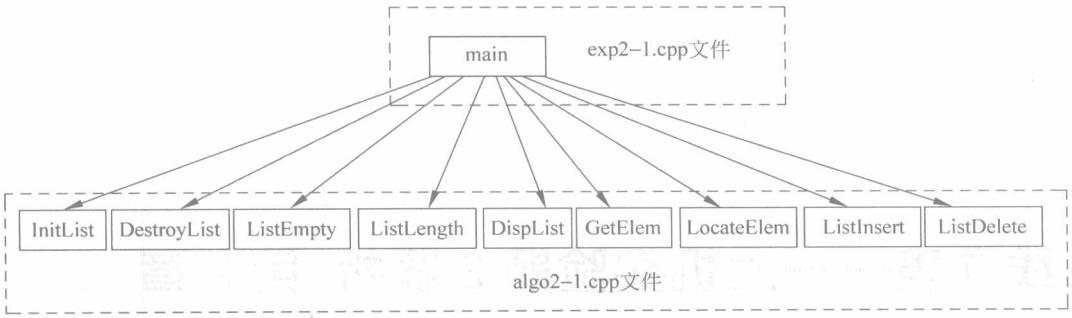


图 2.2　proj2_1 工程的程序结构图

- ListLength(SqList *L)：返回顺序表 L 的元素个数。
- DispList(SqList *L)：输出顺序表 L。
- GetElem(SqList *L,int i,ElemType &e)：获取顺序表 L 中第 i 个元素。
- LocateElem(SqList *L, ElemType e)：在顺序表 L 中查找元素 e。
- ListInsert(SqList *&L,int i,ElemType e)：在顺序表 L 中第 i 个位置上插入元素 e。
- ListDelete(SqList *&L,int i,ElemType &e)：在顺序表 L 中删除第 i 个元素。

对应的程序如下(设计思路详见代码中的注释)：

```
//文件名:algo2-1.cpp
#include <stdio.h>
#include <malloc.h>
#define MaxSize 50
typedef char ElemType;
typedef struct
{   ElemType data[MaxSize];
    int length;
} SqList;                          //顺序表类型定义
void InitList(SqList *&L)          //初始化顺序表 L
{
    L=(SqList *)malloc(sizeof(SqList));
    L->length=0;
}
void DestroyList(SqList *L)        //释放顺序表 L
{
    free(L);
}
int ListEmpty(SqList *L)           //判断顺序表 L 是否为空表
{
    return(L->length==0);
}
int ListLength(SqList *L)          //返回顺序表 L 的元素个数
{
    return(L->length);
}
void DispList(SqList *L)           //输出顺序表 L
```

```
{
    int i;
    if (ListEmpty(L)) return;
    for (i = 0;i<L->length;i++)
        printf("%c",L->data[i]);
    printf("\n");
}
int GetElem(SqList *L,int i,ElemType &e)        //获取顺序表 L 中第 i 个元素
{
    if (i<1 || i>L->length)
        return 0;
    e = L->data[i-1];
    return 1;
}
int LocateElem(SqList *L, ElemType e)           //在顺序表 L 中查找元素 e
{
    int i = 0;
    while (i<L->length && L->data[i]!=e) i++;
    if (i>=L->length)
        return 0;
    else
        return i+1;
}
int ListInsert(SqList *&L,int i,ElemType e)
//在顺序表 L 中第 i 个位置上插入元素 e
{
    int j;
    if (i<1 || i>L->length+1)
        return 0;
    i--;                                        //将顺序表位序转化为 data 下标
    for (j=L->length;j>i;j--)                   //将 data[i]及后面元素后移一个位置
        L->data[j]=L->data[j-1];
    L->data[i]=e;
    L->length++;                                //顺序表长度增 1
    return 1;
}
int ListDelete(SqList *&L,int i,ElemType &e)    //在顺序表 L 中删除第 i 个元素
{
    int j;
    if (i<1 || i>L->length)
        return 0;
    i--;                                        //将顺序表位序转化为 data 下标
    e=L->data[i];
    for (j=i;j<L->length-1;j++)
        L->data[j]=L->data[j+1];
    L->length--;
    return 1;
}
```