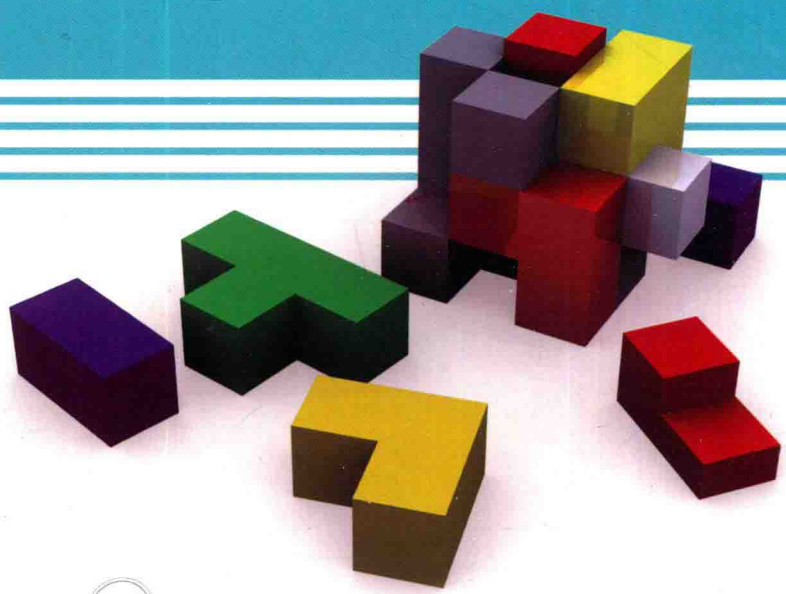
高等学校数据结构课程系列教材

数据结构简明教程

（第2版）学习与上机实验指导

◎ 李春葆 主编



单项选择题170道　填空题112道

简答题62道　算法设计题57道　上机实验题118题

清华大学出版社

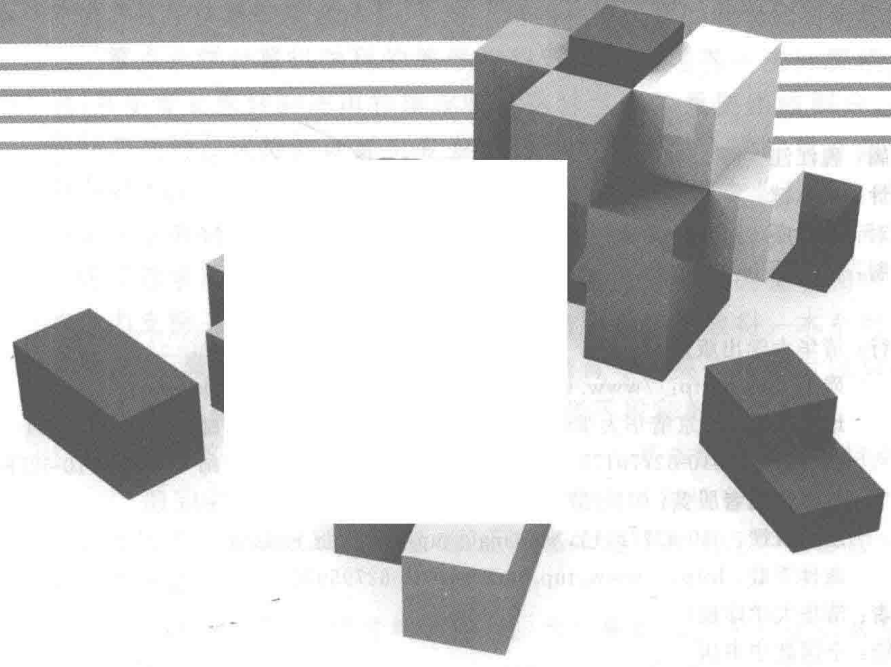
高等学校数据结构课程系列教材

数据结构简明教程

（第2版）学习与上机实验指导

◎ 李春葆　主编

蒋林　方颖　喻丹丹　曾平　编著



清华大学出版社

北京

内 容 简 介

本书是《数据结构简明教程》(第2版,李春葆等编著,清华大学出版社,2018)的配套学习和上机实验指导书。书中练习题和实验题不仅涵盖数据结构课程的基本知识点,还融合了各个知识点的运用和扩展。学习、理解和借鉴这些参考答案是掌握和提高数据结构知识的最佳捷径。本书自成一体,可以脱离主教材单独使用,适合高等院校计算机及相关专业本、专科生使用。

图书在版编目(CIP)数据

数据结构简明教程(第2版)学习与上机实验指导/李春葆主编. —北京:清华大学出版社,2019
(高等学校数据结构课程系列教材)
ISBN 978-7-302-51629-3

Ⅰ. ①数… Ⅱ. ①李… Ⅲ. ①数据结构—高等学校—教学参考资料 Ⅳ. ①TP311.12

中国版本图书馆CIP数据核字(2018)第257384号

责任编辑:魏江江　薛　阳
封面设计:刘　键
责任校对:李建庄
责任印制:杨　艳

出版发行:清华大学出版社
网　　址:http://www.tup.com.cn, http://www.wqbook.com
地　　址:北京清华大学学研大厦A座　　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
课件下载:http://www.tup.com.cn,010-62795954

印 装 者:清华大学印刷厂
经　　销:全国新华书店
开　　本:185mm×260mm　　**印　　张**:17.5　　**字　　数**:423千字
版　　次:2019年2月第1版　　**印　　次**:2019年2月第1次印刷
印　　数:1～1500
定　　价:49.00元

产品编号:080398-01

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项目成果得到肯定。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《数据结构简明教程》(第2版,李春葆等编著,清华大学出版社,以下简称为《教程》)的配套学习与上机实验指导书。全书共分为9章,章次安排与《教程》相对应。

书中练习题共401道(含单项选择题170道,填空题112道,简答题62道,算法设计题57道),上机实验题共118道(含基础实验题37道,应用实验题81道)。

所有练习题均给出了参考答案,算法设计题包含算法设计思路和完整的C/C++语言描述代码。所有上机实验题均上机调试通过。考虑向下的兼容性,所有程序调试执行采用低版本的Visual C++ 6.0作为编程环境,稍加修改,可以在Dev C++或者其他更高版本的编程环境中执行。全部上机实验题的源程序代码可以从清华大学出版社网站http://www.tup.edu.cn免费下载。

书中同时列出了全部练习题和上机实验题,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,书中仍可能存在疏漏和不足之处,敬请教师和同学们批评指正。

编　者

2018年12月

CONTENTS

目录

CHAPTER 1

概　论　第1章

1.1 练习题1及参考答案

1.1.1 练习题1

1. 单项选择题

(1) 线性结构中数据元素之间是(　　)关系。

A. 一对多　　B. 多对多　　C. 多对一　　D. 一对一

(2) 数据结构中与计算机无关的是数据的(　　)结构。

A. 存储　　B. 物理　　C. 逻辑　　D. 物理和存储

(3) 在计算机中存储数据时,通常不仅要存储各数据元素的值,而且要存储(　　)。

A. 数据的处理方法　　B. 数据元素的类型

C. 数据元素之间的关系　　D. 数据的存储方法

(4) 数据采用链式存储结构时,要求(　　)。

A. 每个结点占用一片连续的存储区域

B. 所有结点占用一片连续的存储区域

C. 结点的最后一个域必须是指针域

D. 每个结点有多少后继结点,就必须设多少个指针域

(5) 计算机算法是指(　　)。

A. 计算方法　　B. 排序方法

C. 求解问题的有限运算序列　　D. 调度方法

(6) 计算机算法必须具备输入、输出和(　　)等5个特性。

A. 可行性、可移植性和可扩充性

B. 可行性、确定性和有穷性

C. 确定性、有穷性和稳定性

D. 易读性、稳定性和安全性

(7) 算法分析的目的是(　　)。

A. 找出数据结构的合理性　　B. 研究算法中的输入和输出的关系

C. 分析算法的效率以求改进　　D. 分析算法的易懂性和文档性

(8) 算法分析的两个主要方面是(　　)。

A. 空间复杂性和时间复杂性　　B. 正确性和简明性

C. 可读性和文档性　　D. 数据复杂性和程序复杂性

(9) 某算法的时间复杂度为$O(n^2)$,表明该算法的(　　)。

A. 问题规模是n^2　　B. 执行时间等于n^2

C. 执行时间与n^2成正比　　D. 问题规模与n^2成正比

(10) 某算法的空间复杂度为$O(1)$,表明执行该算法时(　　)。

A. 不需要存储空间　　B. 需要的临时存储空间为常量

C. 需要的存储空间恰好为1　　D. 需要的临时存储空间为1

2. 填空题

(1) 数据结构包括数据的(　①　)、数据的(　②　)和数据的(　③　)三个方面的内容。

(2) 数据结构按逻辑结构可分为两大类,它们分别是(　①　)和(　②　)。

(3) 数据结构被形式地定义为(D,R),其中,D是(　①　)的有限集合,R是D上的(　②　)的有限集合。

(4) 在线性结构中,开始元素(　①　)前驱元素,其余每个元素有且只有一个前驱元素;最后一个元素(　②　)后继元素,其余每个元素有且只有一个后继元素。

(5) 在树状结构中,树根结点没有(　①　)结点,其余每个结点有且只有(　②　)个前驱结点;叶子结点没有(　③　)结点,其余每个结点的后继结点数可以是(　④　)。

(6) 在图形结构中,每个结点的前驱结点个数和后继结点个数可以是(　　)。

(7) 数据的存储结构主要有4种,它们分别是(　①　)、(　②　)、(　③　)和(　④　)存储结构。

(8) 一个算法的效率可分为(　①　)效率和(　②　)效率。

(9) 在分析算法时,其时间复杂度是(　　)的函数。

(10) 在分析算法时,其空间复杂度是指执行该算法时所需(　　)的大小。

3. 简答题

(1) 简述数据结构中运算描述和运算实现的异同。

(2) 以下各函数是算法中语句的执行频度,n为问题规模,给出对应的时间复杂度。

$$T_1(n)=n\log_2 n-1000\log_2 n$$

$$T_2(n)=n^{\log_2 3}-1000\log_2 n$$

$$T_3(n)=n^2-1000\log_2 n$$

$$T_4(n)=2n\log_2 n-1000\log_2 n$$

(3) 分析下面程序段中循环语句的执行次数。

```
int j=0,s=0,n=100;
do
```

```
{   j=j+1;
    s=s+10*j;
} while (j<n && s<n);
```

(4) 执行下面的语句时，语句 $s++$ 的执行次数为多少？

```
int s=0;
for (i=1;i<n-1;i++)
    for (j=n;j>=i;j--)
            s++;
```

(5) 设 n 为问题规模，求以下算法的时间复杂度。

```
void fun1(int n)
{   int x=0,i;
    for (i=1;i<=n;i++)
        for (j=i+1;j<=n;j++)
            x++;
}
```

(6) 设 n 为问题规模，是一个正偶数，试计算以下算法结束时 m 的值，并给出该算法的时间复杂度。

```
void fun2(int n)
{   int m=0;
    for (i=1;i<=n;i++)
        for (j=2*i;j<=n;j++)
            m++;
}
```

1.1.2 练习题 1 参考答案

1. 单项选择题

(1) D　(2) C　(3) C　(4) A　(5) C

(6) B　(7) C　(8) A　(9) C　(10) B

2. 填空题

(1) ① 逻辑结构　② 存储结构　③ 运算(不限制顺序)

(2) ① 线性结构　② 非线性结构(不限制顺序)

(3) ① 数据元素　② 关系

(4) ① 没有　② 没有

(5) ① 前驱　② 一　③ 后继　④ 任意多个

(6) 任意多个

(7) ① 顺序　② 链式　③ 索引　④ 哈希(不限制顺序)

(8) ① 时间　② 空间(不限制顺序)

(9) 问题规模(通常用 n 表示)

(10) 辅助或临时空间

3. 简答题

(1) **答**：运算描述是指逻辑结构施加的操作，而运算实现是指一个完成该运算功能的算法。它们的相同点是，运算描述和运算实现都能完成对数据的“处理”或某种特定的操作。不同点是，运算描述只是描述处理功能，不包括处理步骤和方法，而运算实现的核心则是处理步骤。

(2) **答**：$T_1(n)=O(n\log_2 n)$，$T_2(n)=O(n^{\log_2 3})$，$T_3(n)=O(n^2)$，$T_4(n)=O(n\log_2 n)$。

(3) **答**：$j=0$，第1次循环：$j=1$，$s=10$。第2次循环：$j=2$，$s=30$。第3次循环：$j=3$，$s=60$。第4次循环：$j=4$，$s=100$。while条件不再满足。所以，其中循环语句的执行次数为4。

(4) **答**：语句$s++$的执行次数：$\sum_{i=1}^{n-2}\sum_{j=n}^{i}1=\sum_{i=1}^{n-2}(n-i+1)=n+(n-1)+\cdots+3=\frac{(n+3)(n-2)}{2}$。

(5) **答**：其中$x++$语句为基本运算语句，$T(n)=\sum_{i=1}^{n}\sum_{j=i+1}^{n}1=\sum_{i=1}^{n}(n-i)=\frac{n(n-1)}{2}=O(n^2)$。

(6) **答**：由于内循环j的取值范围，所以$i\leqslant n/2$，则$m=\sum_{i=1}^{n/2}\sum_{j=2i}^{n}=\sum_{i=1}^{n/2}(n-(2i-1))=n^2/4$，该程序段的时间复杂度为$O(n^2)$。

1.2 上机实验题1及参考答案

1.2.1 上机实验题1

1. 基础实验题

(1) 有以下两个求$1+2+\cdots+n$的算法：

```
long Sum1(long n)                    //算法1
{   long s=0;
    for (long i=1;i<=n;i++)
        s+=i;
    return s;
}
long Sum2(long n)                    //算法2
{   long s=n*(n+1)/2;
    return s;
}
```

现在需要计算$1+(1+2)+(1+2+3)+\cdots+(1+2+3+\cdots+n)$。编写一个程序分别调用上述两个算法求解，给出$n=100000$时两个算法的执行时间。

(2) 编写一个程序求$1!+2!+3!+\cdots+n!$，其中，n为正整数。请给出直接累计$i!(1\leqslant i\leqslant n)$的算法和改进算法。并采用相关数据测试(在上机实验时n比较大时结果会溢出，不必考虑结果溢出情况)。分析两个算法的时间复杂度。

2. 应用实验题

(1) 有一个整型数组 a,其中含有 n 个元素,设计尽可能好的算法求其中的最大元素和次大元素,并采用相关数据测试。

(2) 定义单个复数的抽象数据类型为 AComplex,其中,复数的实部和虚部均为整数,包含创建一个复数和输出一个复数的基本运算。在此基础上再定义两个复数运算的抽象数据类型 BComplex,包含两个复数的加法、减法和乘法运算。编写程序实现这两个抽象数据类型,并采用相关数据测试。

1.2.2 上机实验题1参考答案

1. 基础实验题

(1) **解**:对应的实验程序如下。

```
#include <stdio.h>
#include <time.h>                          //含 clock_t, clock, CLOCKS_PER_SEC
long Sum1(long n)                          //算法 1
{   long s=0;
    for (long i=1;i<=n;i++)
        s+=i;
    return s;
}
long Sum2(long n)                          //算法 2
{   long s=n*(n+1)/2;
    return s;
}
void display(long n)                       //输出测试结果
{   long sum=0,i;
    clock_t t;
    printf("算法 1: ");
    t=clock();                             //求调用前的时刻
    for (i=1;i<=n;i++)
        sum+=Sum1(i);
    t=clock()-t;                           //求所有调用 Sum1 花费的时间
    printf("结果=%ld ",sum);
    printf("时间=%lf 秒\n",((float)t)/CLOCKS_PER_SEC);
    sum=0;
    printf("算法 2: ");
    t=clock();                             //求调用前的时刻
    for (i=1;i<=n;i++)
        sum+=Sum2(i);
    t=clock()-t;                           //求所有调用 Sum2 花费的时间
    printf("结果=%ld ",sum);
    printf("时间=%lf 秒\n",((float)t)/CLOCKS_PER_SEC);
}
void main()
{
    display(100000);
}
```

上述程序的一次执行结果如图1.1所示。

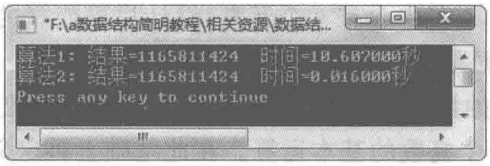


图1.1 实验程序执行结果

(2) **解**：改进算法的思路是：当$(i-1)!$求出后，其结果为$p1$，在求$i!$时不必从1开始累乘，而是为$p1*i$即可。对应的实验程序如下。

```
#include <stdio.h>
#include <time.h>                          //含 clock_t, clock, CLOCKS_PER_SEC
#define MAX 999
long Product1(long n)                      //算法 1: 求所有 i!并累加
{   long p=0,p1;
    for (long i=1;i<=n;i++)
    {       p1=1;
        for (long j=1;j<=i;j++)            //求 j!
            p1 *=j;
        p+=p1;
    }
    return p;
}
long Product2(long n)                      //算法 2: 改进算法
{   long p=0,p1=1;
    for (long i=1;i<=n;i++)
    {   p1 *=i;                            //i!=(i-1)! *i
        p+=p1;
    }
    return p;
}
void display(long n)                       //输出测试结果
{   long p;
    clock_t t;
    printf("算法 1: ");
    t=clock();
    p=Product1(n);
    t=clock()-t;
    printf("结果=%ld ",p);
    printf("时间=%lf 秒\n" ,((float)t)/CLOCKS_PER_SEC);
    printf("算法 2: ");
    t=clock();
    p=Product2(n);
    t=clock()-t;
    printf("结果=%ld ",p);
    printf("时间=%lf 秒\n" ,((float)t)/CLOCKS_PER_SEC);
}
void main()
{
```

```
    display(100000);
}
```

上述程序的一次执行结果如图 1.2 所示(结果发生了溢出)。其中,算法 1 的时间复杂度为 $O(n^2)$,算法 2 的时间复杂度为 $O(n)$。

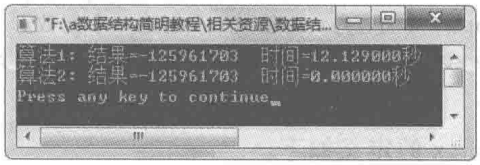


图 1.2 实验程序执行结果

2. 应用实验题

(1) **解**:maxs 算法用于返回数组 $a[0..n-1]$中的最大元素值 max1 和次大元素值 max2,max1 和 max2 设计为引用类型。对应的实验程序如下。

```
#include <stdio.h>
void maxs(int a[],int n,int &max1,int &max2)
{   int i;
    max1=max2=a[0];
    for (i=1;i<n;i++)                    //扫描 a[1..n-1]
        if (a[i]>max1)                   //当 a[i]>max1
        {   max2=max1;
            max1=a[i];
        }
        else if (a[i]>max2)              //当 a[i]≤max1 且 a[i]>max2
            max2=a[i];
}
void main()
{   int a[]={1,4,10,6,8,3,5,7,9,2};
    int n=sizeof(a)/sizeof(a[0]);
    int max1,max2;
    printf("a 中的元素:");
    for (int i=0;i<n;i++)
        printf("%d ",a[i]);
    printf("\n");
    printf("求最大元素 max1,次大元素 max2\n");
    maxs(a,n,max1,max2);
    printf("max1=%d, max2=%d\n",max1,max2);
}
```

上述程序的执行结果如图 1.3 所示。

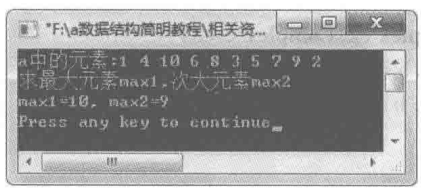


图 1.3 实验程序执行结果

(2) **解**：抽象数据类型 AComplex 的定义如下。

```
ADT AComplex
{
    数据对象：D={<a,b> | a,b 均为一个整数}
    运算的定义：
        void CreateComplex(&c,i,j)：由整数 i、j 创建一个复数 c
        void dispComplex(c)：输出复数 c
}
```

抽象数据类型 BComplex 的定义如下。

```
ADT BComplex
{   数据对象：D={ci∈AComplex | 0≤i≤n,n 为一个正整数}
    运算的定义：
        void add(c1,c2,c3)：c3=c1+c2        //复数加法运算
        void sub(c1,c2,c3)：c3=c1-c2        //复数减法运算
        void mult(c1,c2,c3)：c3=c1*c2       //复数乘法运算
}
```

对应的实验程序如下。

```
#include <stdio.h>
typedef struct
{   int a;                                  //实部
    int b;                                  //虚部
} Complex;                                  //声明复数类型
void CreateComplex(Complex &c,int i,int j)
{   c.a=i;
    c.b=j;
}
void dispComplex(Complex c)
{   printf("%d",c.a);
    if (c.b>=0)
        printf("+%di\n",c.b);
    else
        printf("-%di\n",-c.b);
}
void add(Complex c1,Complex c2,Complex &c3)
{   c3.a=c1.a+c2.a;
    c3.b=c1.b+c2.b;
}
void sub(Complex c1,Complex c2,Complex &c3)
{   c3.a=c1.a-c2.a;
    c3.b=c1.b-c2.b;
}
void mult(Complex c1,Complex c2,Complex &c3)
{   c3.a=c1.a*c2.a-c1.b*c2.b;
    c3.b=c1.a*c2.b-c1.b*c2.a;
}
void main()
{   Complex c1,c2,c3;
```