普通高等教育“十三五”规划教材（软件工程专业）

数据结构（C语言版）习题解答及实训指导

SHUJU JIEGOU (C YUYAN BAN)
XITI JIEDA JI SHIXUN ZHIDAO

Data Structure

主　编　李根强　谢月娥

- 内容丰富，题型多样。
- 涉及面广，适用性强。
- 习题精解，综合训练。

中国水利水电出版社
www.waterpub.com.cn

普通高等教育“十三五”规划教材（软件工程专业）

数据结构（C 语言版）习题解答及实训指导

主　编　李根强　谢月娥

中国水利水电出版社
www.waterpub.com.cn
·北京·

内 容 提 要

本书是与《数据结构（C 语言版）》（李根强、刘浩、谢月娥主编）一书配套的辅导教材。全书包含两部分内容：教材的习题及解答、上机实训指导。书中除给出配套教材中习题的解答外，还给出了典型例题的算法分析、算法实现；上机实训指导给出了十套实训题，每套实训题包含多个上机题目，给出了实训目的、算法提示、算法实现，各院校相关人员可根据实际情况选取。

本书内容丰富、题型多样、涉及面广、适用性强，与《数据结构（C 语言版）》一书的内容紧密结合，既可以作为高等院校本科及专科教材，也可以作为硕士研究生入学考试的参考书，还可以供自学人员参考使用。

本书提供源代码，读者可以从中国水利水电出版社网站或万水书苑上免费下载，网址为：http://www.waterpub.com.cn/softdown/和 http://www.wsbookshow.com。

图书在版编目（CIP）数据

数据结构（C语言版）习题解答及实训指导 / 李根强，谢月娥主编. -- 北京 : 中国水利水电出版社，2017.5
普通高等教育“十三五”规划教材. 软件工程专业
ISBN 978-7-5170-5336-1

Ⅰ. ①数… Ⅱ. ①李… ②谢… 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(2017)第083692号

策划编辑：周益丹　　责任编辑：封　裕　　封面设计：李　佳

书　名	普通高等教育“十三五”规划教材（软件工程专业） 数据结构（C 语言版）习题解答及实训指导 SHUJU JIEGOU (C YUYAN BAN) XITI JIEDA JI SHIXUN ZHIDAO
作　者	主　编　李根强　谢月娥
出版发行	中国水利水电出版社 （北京市海淀区玉渊潭南路 1 号 D 座　100038） 网址：www.waterpub.com.cn E-mail：mchannel@263.net（万水） sales@waterpub.com.cn 电话：（010）68367658（营销中心）、82562819（万水）
经　售	全国各地新华书店和相关出版物销售网点
排　版	北京万水电子信息有限公司
印　刷	三河市铭浩彩色印装有限公司
规　格	184mm×260mm　16 开本　15.75 印张　390 千字
版　次	2017 年 5 月第 1 版　2017 年 5 月第 1 次印刷
印　数	0001—3000 册
定　价	32.00 元

前　言

“数据结构”课程是计算机专业及相关专业的一门重要专业基础课，也是一门必修的核心课程。在计算机科学的各领域中，都会用到各种不同的数据结构：编译系统中要使用栈、散列表、语法树等；操作系统中要使用队列、存储管理表、目录树等；数据库系统中要使用线性表、链表、索引树等；人工智能中要使用广义表、检索树、有向图等；同样在面向对象程序设计、计算机图形学、软件工程、多媒体技术、计算机辅助设计等领域，也会用到各种不同的数据结构。因此，学好“数据结构”这门课程，对于从事计算机技术及相关领域的工作人员来说是非常重要的，它可以使我们掌握各种不同的数据结构及其使用场合、算法的具体实现，以及每一种算法的时间复杂度分析和空间复杂度分析，知道在哪种情况下，使用哪种数据结构最方便，为以后开发大型程序而使用各种不同的数据结构奠定基础。

由于数据结构的原理和算法比较抽象，而该课程一般在刚学完程序设计后开设，学生在程序设计方面的知识较少，因此，对于仅具有一些计算机程序设计基础的初学者来说，理解和掌握数据结构中的原理和算法就比较困难，在解答数据结构习题时，往往感到无从下手，更不知道算法如何描述、如何编写，作者在多年的教学中也深有感触。作者通过多年的教学实践，收集、整理进而编写了这本《数据结构（C 语言版）习题解答及实训指导》，目的是：通过对习题的解答，使学生充分掌握数据结构的原理、求解数据结构问题的思路和方法，并编写出正确的算法，进一步加深学生对基本概念的理解，提高学生分析问题和解决问题的能力，为其整个专业的学习打下坚实基础，并为今后大型软件开发积累较丰富的经验。

本书是《数据结构（C 语言版）》的配套教材，共两大部分内容。第一部分为习题及解答，包含配套教材中的习题和一些综合题，主要是对教材中的习题给出解答思路提示、算法分析，最后再给出完整的答案、算法或程序。每章都包含基本概念和算法设计方面的题型，以帮助学生掌握数据结构的基本概念，提高其使用各种不同的数据结构分析问题、解决问题的能力。第二部分为上机实训指导，包含 C 语言的上机环境介绍，并给出了十套实训题，每套实训题包括多道上机题目，都给出了实训目的、采用的主要方法和算法设计技巧，最后给出了完整的 C 语言源程序来实现算法，本书所有 C 语言源程序都在 VC++ 6.0 环境下运行通过。这些实训题可以使学生了解并学会如何运用数据结构知识去解决现实世界中的某些实际问题，且具备设计较复杂算法的基本能力。

本书的目的是帮助学生学好“数据结构”这门课程，所以在使用本书的过程中，请注意以下几点：第一，该书要与《数据结构（C 语言版）》教材一起使用，以便与课程内容同步，有利于学生进一步掌握各种基础知识，加深对课堂所讲授内容的理解；第二，算法设计是不唯一的，除了书中给出的算法外，学生可能还会有其他的算法，这说明学生对这方面的知识掌握得很好；第三，本书旨在帮助学生学习和理解数据结构知识，请学生在做作业前，先自己动脑思考、动手写，不要盲目照抄答案，否则只会收到事倍功半的效果。

本书的最大特点是采用 C 语言作为算法的描述语言，所有算法都已经上机调试通过。但是，由于篇幅所限，大部分算法都是以单独的函数形式给出，若读者要运行这些算法，还必

须给出一些变量的说明及主函数来调用所给的函数。因此，书中的算法描述比用类 C 语言描述的算法更直观，更易于读者理解和接受。作者在十几年的“数据结构”课程教学中，对数据结构中的各种算法进行了认真的研究和分析，在这方面积累了丰富的经验，因此，书中所选的综合题和习题都具有一定的针对性，都是针对特定的数据结构来描述的，能为复杂的数据结构算法描述架桥铺路。

本书可以作为高等院校本科及专科学生学习数据结构的参考资料，也可作为数据结构自学者的参考书和硕士研究生入学考试的参考教材，同时还可供从事计算机应用工作的工程与技术人员参考。选用本书的学校，可以根据学校自身的条件，在十套实训题中有针对性地选择。

本书由李根强、谢月娥担任主编，由李根强老师统稿、修改、定稿。在本书的编写过程中，得到了湖南大学计算机与通信学院领导的大力支持，在此深表感谢。

由于时间仓促及作者水平有限，书中不妥和错误之处在所难免，敬请广大读者批评指正。

编 者

2017 年 2 月

目　　录

前言

第一部分　习题及解答

第 1 章　绪论 …… 1
1.1　基本概念 …… 1
1.1.1　数据结构 …… 1
1.1.2　存储方式 …… 1
1.1.3　算法及评价 …… 2
1.2　习题及解答 …… 2
1.2.1　配套教材中的习题 …… 2
1.2.2　综合题 …… 10
第 2 章　线性表 …… 14
2.1　基本概念及运算 …… 14
2.1.1　顺序表 …… 14
2.1.2　线性链表 …… 14
2.1.3　双向链表 …… 14
2.1.4　循环链表 …… 14
2.2　习题及解答 …… 14
2.2.1　配套教材中的习题 …… 14
2.2.2　综合题 …… 31
第 3 章　栈和队列 …… 37
3.1　基本概念及运算 …… 37
3.1.1　栈 …… 37
3.1.2　队列 …… 37
3.2　习题及解答 …… 37
3.2.1　配套教材中的习题 …… 37
3.2.2　综合题 …… 46
第 4 章　串 …… 53
4.1　基本概念及运算 …… 53
4.1.1　串的顺序存储及运算 …… 53
4.1.2　串的链式存储及运算 …… 53
4.2　习题及解答 …… 53
4.2.1　配套教材中的习题 …… 53
4.2.2　综合题 …… 62
第 5 章　多维数组和广义表 …… 67
5.1　基本概念及运算 …… 67
5.1.1　多维数组的概念及存储 …… 67
5.1.2　特殊矩阵及压缩存储 …… 67
5.1.3　稀疏矩阵及压缩存储 …… 67
5.1.4　广义表的存储及运算 …… 67
5.2　习题及解答 …… 68
5.2.1　配套教材中的习题 …… 68
5.2.2　综合题 …… 75
第 6 章　树和二叉树 …… 81
6.1　树的基本概念 …… 81
6.1.1　树的定义 …… 81
6.1.2　基本术语 …… 81
6.1.3　树的表示 …… 82
6.2　二叉树的基本概念和性质 …… 82
6.2.1　二叉树的定义 …… 82
6.2.2　二叉树的性质 …… 82
6.2.3　二叉树的存储结构 …… 83
6.2.4　二叉树的基本运算 …… 83
6.2.5　二叉树的应用 …… 83
6.2.6　树、森林和二叉树之间的相互关系 …… 83
6.3　习题及解答 …… 83
6.3.1　配套教材中的习题 …… 83
6.3.2　综合题 …… 93
第 7 章　图 …… 102
7.1　基本概念及运算 …… 102
7.1.1　图的基本术语 …… 102
7.1.2　图的存储形式 …… 103
7.1.3　图的基本运算 …… 103
7.2　习题及解答 …… 103
7.2.1　配套教材中的习题 …… 103
7.2.2　综合题 …… 123
第 8 章　查找 …… 131
8.1　基本概念 …… 131
8.1.1　顺序查找 …… 131
8.1 2　二分查找 …… 131
8.1.3　分块查找 …… 131

8.1.4 二叉排序树查找 …… 131
8.1.5 散列查找 …… 132
8.2 习题及解答 …… 132
8.2.1 配套教材中的习题 …… 132
8.2.2 综合题 …… 139
第 9 章 内排序 …… 144
9.1 基本概念 …… 144
9.1.1 插入排序 …… 144
9.1.2 交换排序 …… 144
9.1.3 选择排序 …… 144
9.1.4 归并排序 …… 144
9.1.5 分配排序 …… 144
9.2 习题及解答 …… 144
9.2.1 配套教材中的习题 …… 144
9.2.2 综合题 …… 155
第 10 章 外排序 …… 160
10.1 基本概念 …… 160
10.1.1 外排序的基本概念 …… 160
10.1.2 初始归并段的生成 …… 160
10.1.3 多路平衡归并 …… 160
10.2 习题及解答 …… 160
第 11 章 文件 …… 175
11.1 基本概念 …… 175
11.1.1 文件的基本概念 …… 175
11.1.2 文件的存储和组织 …… 175
11.2 习题及解答 …… 176

第二部分 上机实训指导

第 12 章 上机环境 …… 178
12.1 Turbo C 上机环境 …… 178
12.1.1 建立 C 语言源程序 …… 178
12.1.2 打开已存在的 C 语言源程序 …… 179
12.1.3 编译并运行 C 语言源程序 …… 179
12.2 Visual C++上机环境 …… 181
12.2.1 新建 C 语言源程序并编译和运行 · 181
12.2.2 打开已经存在的源程序并编译和运行 …… 187
12.2.3 源程序的保存 …… 189
第 13 章 实训内容 …… 190
实训题一 线性表的顺序存储 …… 190
一、实训目的 …… 190
二、实训内容 …… 190
三、算法描述 …… 190
实训题二 线性表的链式存储 …… 193
一、实训目的 …… 193
二、实训内容 …… 194
三、算法描述 …… 194
实训题三 栈和队列的应用 …… 203
一、实训目的 …… 203
二、实训内容 …… 203
三、算法描述 …… 203
实训题四 多维数组的应用 …… 212
一、实训目的 …… 212
二、实训内容 …… 212
三、算法描述 …… 212
实训题五 二叉树的遍历和应用 …… 218
一、实训目的 …… 218
二、实训内容 …… 218
三、算法描述 …… 218
实训题六 哈夫曼树的建立及应用 …… 226
一、实训目的 …… 226
二、实训内容 …… 226
三、算法描述 …… 226
实训题七 图的邻接矩阵和遍历 …… 228
一、实训目的 …… 228
二、实训内容 …… 229
三、算法描述 …… 229
实训题八 图的邻接表和遍历 …… 231
一、实训目的 …… 231
二、实训内容 …… 231
三、算法描述 …… 231
实训题九 查找 …… 234
一、实训目的 …… 234
二、实训内容 …… 234
三、算法描述 …… 234
实训题十 排序 …… 238
一、实训目的 …… 238
二、实训内容 …… 238
三、算法描述 …… 238
参考文献 …… 246

第一部分　习题及解答

第 1 章　绪论

1.1　基本概念

1.1.1　数据结构

数据结构（Data Structure）：是指相互之间存在一种或多种特定关系的数据元素所组成的集合。具体来说，数据结构包含三个方面的内容，即数据的逻辑结构、数据的存储结构和对数据所施加的运算（也称操作）。

这三个方面的关系为：

（1）数据的逻辑结构独立于计算机，是数据本身所固有的。

（2）存储结构是逻辑结构在计算机存储器中的映像，必须依赖于计算机。

（3）运算是指所施加的一组操作的总称。运算的定义直接依赖于逻辑结构，但运算的实现必须依赖于存储结构。

数据结构从逻辑结构上可以分为：线性结构、非线性结构。

数据结构具体可以表示为：集合、线性表、栈、队列、串、多维数组、广义表、树、二叉树、图形结构。

1.1.2　存储方式

数据结构从存储结构上可以分为如下几种：

（1）顺序存储（向量存储）

所有元素存放在一片连续的存储单元中，逻辑上相邻的元素存放到计算机内存仍然相邻（只需存放元素的值，而不需存放元素之间的关系）。

（2）链式存储

所有元素存放在可以是不连续的存储单元中，但元素之间的关系可以通过地址确定，逻辑上相邻的元素存放到计算机内存后不一定是相邻的（也可能是相邻的）。

（3）索引存储

使用该方法存放元素的同时，还应建立附加的索引表，索引表中的每一项都称为索引项，索引项的一般形式是：（关键字，地址），其中的关键字能唯一标识一个结点的数据项。而地址是存储关键字的具体位置。

（4）散列存储

通过构造散列函数，用函数的值来确定元素存放的地址（理论上不需用到比较）。

1.1.3 算法及评价

通俗地讲，算法就是一种解题的方法。更严格地说，算法是由若干条指令组成的有穷序列。

评价一种算法的好坏，主要考虑以下三个方面：

（1）执行算法所耗费的时间（时间复杂度）。

（2）执行算法所占用的内存开销（主要考虑占用的辅助存储空间，称为空间复杂度）。

（3）算法应该易于理解、易于调试、易于编码。

本书中，主要是讨论算法的时间频度和时间复杂度。

1.2 习题及解答

1.2.1 配套教材中的习题

1-1．对下列用二元组表示的数据结构，画出它们的逻辑结构图，并指出它们属于何种结构。

（1）A=(K,R)，其中：

K={a_1,a_2,a_3,…,a_n}

R={<a_i,a_{i+1}>,i=1,2,…,n-1}

（2）B=(K,R)，其中：

K={a,b,c,d,e,f}

R={<a,b>,<b,c>,<c,a>,<a,d>,<d,e>,<e,f>,<c,f>}

（3）C=(K,R)，其中：

K={1,2,3,4,5,6}

R={(1,2),(2,3),(2,4),(3,4),(3,5),(3,6),(4,5),(4,6)}

（4）D=(K,R)，其中：

K={48,25,64,57,82,36,75,43}

R={r1,r2,r3}

r1={<48,25>,<25,64>,<64,57>,<57,82>,<82,36>,<36,75>,<75,43>}

r2={<48,25>,<48,64>,<64,57>,<64,82>,<25,36>,<82,75>,<36,43>}

r3={<25,36>,<36,43>,<43,48>,<48,57>,<57,64>,<64,75>,<75,82>}

解：（1）的逻辑结构如图 1-1 所示。该结构为线性结构。

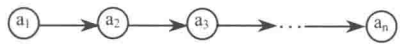

图 1-1 （1）的逻辑结构

（2）的逻辑结构如图 1-2 所示。该结构为图形结构。

（3）的逻辑结构如图 1-3 所示。该结构为图形结构。

（4）的逻辑结构如图 1-4 所示。该结构为图形结构。

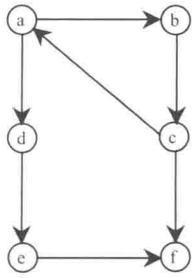

图 1-2　（2）的逻辑结构

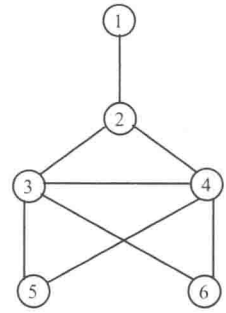

图 1-3　（3）的逻辑结构

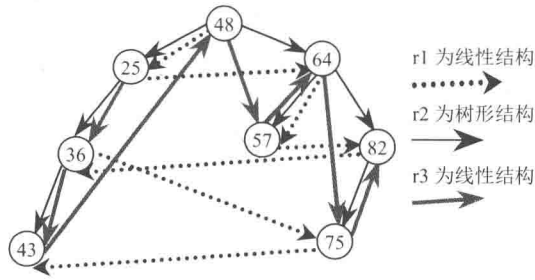

图 1-4　（4）的逻辑结构

1-2．简述概念：数据、数据元素、数据结构、逻辑结构、存储结构、线性结构、非线性结构。

解：

数据：是指能够输入到计算机中，并被计算机识别和处理的符号的集合。

数据元素：是组成数据的基本单位，是一个数据整体中相对独立的单位，但它还可以分割成若干个具有不同属性的项（字段），故其不是组成数据的最小单位。

数据结构：是指相互之间存在一种或多种特定关系的数据元素所组成的集合。具体来说，数据结构包含三个方面的内容，即数据的逻辑结构、数据的存储结构和对数据所施加的运算。

逻辑结构：是数据之间的内在关系，是数据本身所固有的，数据的逻辑结构独立于计算机。

存储结构：是逻辑结构在计算机存储器中的映像，必须依赖于计算机。

线性结构：元素之间存在一种一对一的线性关系的数据结构。

非线性结构：元素之间存在一对多或多对多的非线性关系的数据结构。

1-3．试举日常生活中的一个例子来说明数据结构三个方面的内容。

解：可以用日常生活中的一种物质——水，来说明数据结构的三个方面。

水由许多水分子组成，一个水分子由氢和氧两种元素组成，这与外界的环境无关，相当于数据的逻辑结构。

水在日常生活中有液态、气态、固态三种不同形态，相当于数据有三种存储结构。

对水进行升温、降温等操作相当于对数据所施加的操作。

1-4. 什么是算法？它的五个特性是什么？

解：通俗地讲，算法就是一种解题的方法。更严格地说，算法是由若干条指令组成的有穷序列，它必须满足下述条件（也称为算法的五大特性）：

（1）输入：具有0个或多个输入的外界量（算法开始前的初始量）。

（2）输出：至少产生一个输出，它们是算法执行完后的结果。

（3）有穷性：每条指令的执行次数必须是有限的。

（4）确定性：每条指令的含义都必须明确，无二义性。

（5）可行性：每条指令的执行时间都是有限的。

1-5. 设n为整数，分析下列程序中用#标明的语句的语句频度及时间复杂度。

```
（1）for (i=1; i<=n; i++)
        for (j=1; j<=n; j++)
        {   c[i][j]=0;
            for (k=1; k<=n; k++)
            # c[i][j]=c[i][j]+a[i][k]*b[k][j];
        }
（2）int i=1, j=1;
     while (i<=n&&j<=n)   {# i=i+1; j=j+i; };
（3）int i=1;
     do {for (j=1; j<=n; j++)   # i=i+j
     } while (i<100+n);
（4）for (i=1; i<=n; i++)
        for (j=i; j>=1; j--)
           for (k=1; k<=j; k++)   # x=x+1;
```

解：（1）语句频度 $T(n)=n^3$，时间复杂度为：$O(n^3)$。

（2）语句频度 $T(n)=\lfloor\frac{\sqrt{8n+1}-1}{2}\rfloor$，时间复杂度为：$O(\lfloor\frac{\sqrt{8n+1}-1}{2}\rfloor)$。

（3）语句频度 $T(n)=\lceil\frac{198+2n}{n(n+1)}\rceil n$，时间复杂度为：$O(\lceil\frac{198+2n}{n(n+1)}\rceil n)$。

（4）语句频度 $T(n)=\frac{1}{2}[\frac{n(n+1)(2n+1)}{6}+\frac{n(n+1)}{2}]$，时间复杂度为：$O(n^3)$。

1-6. 设计出一个算法，将n个元素按升序排列，并分析出它的时间频度及时间复杂度。

解：

```
void sort(int a[ ],int n)
{
  int i,j,k,t;
  for(i=0;i<n-1;i++)
  {
    k=i;
    for(j=i+1;j<=n-1;j++)
    if(a[k]>a[j])
    k=j;
    if(k!=i)
    {
```

```
            t=a[k];
            a[k]=a[i];
            a[i]=t;
        }
    }
}
```

该算法的时间频度为 $T(n)=\frac{(n-1)(n+8)}{2}$，时间复杂度为 $O(n^2)$。

1-7. 计算 $\sum_{i=0}^{n}\frac{x^i}{i+1}$ 的值，用 C 语言函数写出算法，并求出它的时间复杂度。

解：

```
float sum(float x,int n)
{
  float s=1,p=1;
  int i;
  for( i=1;i<=n;i++)
  {
      p=p*x;
      s+=p/(i+1);
  }
  return s; }
```

该算法的时间复杂度为 O(n)。

1-8. 用 C 函数描述如何将一维数组中的元素逆置，并分析出时间频度及时间复杂度。

解：

```
void exchange(int a[ ],int n)
{int t;
  for(int i=0;i<=n/2-1;i++)
  {
    t=a[i];
    a[i]=a[n-i-1];
    a[n-i-1]=t;
  }
}
```

该算法的时间频度为 $T(n)=\frac{3n}{2}$，时间复杂度为 O(n)。

1-9. 将三个元素 X、Y、Z 按从小到大排列，用 C 语言函数描述算法，要求所用的比较和移动元素次数最少。

解：

```
void abc(int &x,int &y,int &z)
{
  if(x>y)
  {int t=x;x=y;y=t;}
  if(y>z)
```

```
    { int t=z;z=y;
      if(x<=t) y=t;
      else {y=x; x=t;}
    }
}
```

该算法最坏时仅需 3 次比较和 7 次移动。

1-10．用 C 语言中冒泡排序和选择排序两种方法，分别描述出 n 个元素 $a_1,a_2,a_3,\cdots,a_n$ 的升序排列算法，并分析两种方法的平均比较和移动元素次数。

解：

冒泡排序：

```
void bubblesort(int a[ ],int n)
{
  int i,j,t;
  for( i=0;i<n-1;i++)
    for( j=n-1;j>=i+1;j--)
      if(a[j]<a[j-1])
      {int t=a[j];a[j]=a[j-1];a[j-1]=t;}
}
```

冒泡排序的最小比较次数为 n-1，最小移动次数为 0，最大比较次数为 $\frac{n(n-1)}{2}$，最大移动次数为 $\frac{3n(n-1)}{2}$，平均比较次数为 $\frac{(n-1)(n+2)}{4}$，平均移动次数为 $\frac{3n(n-1)}{4}$。

选择排序：

```
void bubblesort(int a[ ],int n)
{
  int i,j,t;
  for(i=0;i<n-1;i++)
    for(j=n-1;j>=i+1;j--)
      if(a[j]<a[j-1])
      {int t=a[j];a[j]=a[j-1];a[j-1]=t;}
}
```

选择排序的最小比较次数为 $\frac{n(n-1)}{2}$，最小移动次数为 0，最大比较次数为 $\frac{n(n-1)}{2}$，最大移动次数为 3(n-1)，平均比较次数为 $\frac{n(n-1)}{2}$，平均移动次数为 $\frac{3n(n-1)}{2}$。

1-11．设计一个二次多项式 ax^2+bx+c 的抽象数据类型，假定起名为 AX2BXC，该类型的数据部分分为 3 个系数 a、b 和 c，操作部分为：

（1）初始化成员 a、b 和 c（假定用结构体 D 来定义 a、b、c）。

```
AX2BXC initAX2BXC(x,y,z);
```

（2）做两个多项式的加法运算，即将它们的系数相加，并返回相加的结果。

```
AX2BXC add(AX2BXC f1, AX2BXC f2);
```

（3）根据给定的 x 值，求多项式的值。

```
float value(AX2BXC f,float x);
```

（4）计算方程 $ax^2+bx+c=0$ 的两个实根，要求分有实根、无实根和不是二次方程这三种情况讨论，并返回不同的值，以便调用时做不同的处理。

```
int root(AX2BXC f, float &r1,float &r2);
```

（5）按照 ax**2+bx+c 的格式（x^2 用 x**2 表示）输出二次多项式，在输出时必须注意去掉系数为 0 的项，并且当 b 和 c 的值为负时，其前面不能出现加号。

```
void print(AX2BXC f);
```

请写出上面每一个操作的具体实现。

解：

抽象数据类型描述为：

```
ADT Ax2BxC is
Data: a,b,c 为 3 个实型数
Operation:
Ax2BxC initAx2BxC(float x,float y,float z);
Ax2BxC add(Ax2BxC f1, Ax2BxC f2);
float value(Ax2BxC f, float x);
int root(Ax2BxC f,float &r1,float &r2);
void print(Ax2BxC f);
End Ax2BxC
```

上面每一个操作的具体实现如下：

```
struct Ax2BxC
{   float a,b,c;
}D;
Ax2BxC initAx2BxC(float x,float y,float z)
{
    D.a=x;D.b=y;D.c=z;
    return D;
}

Ax2BxC add(Ax2BxC f1, Ax2BxC f2)
{
    f1.a=f1.a+f2.a;
    f1.b=f1.b+f2.b;
    f1.c=f1.c+f2.c;
    return f1;
}

float value(Ax2BxC f, float x)
{
    return f.a*x*x+f.b*x+f.c;
}

int root(Ax2BxC f,float &r1,float &r2)
{
    float r1,r2, d=f.b*f.b-4*f.a*f.c;
    int e;
```

```
    if(f.a!=0)
    {   if(d>=0)
        {   r1=(-f.b+sqrt(d))/(2*f.a);
            r2=(-f.b-sqrt(d))/(2*f.a);
            e=1;                          //有实根
        }
        else e=0;                         //无实根
    }
    else e=-1;                            //不是二次方程
    return e;
}

void print(Ax2BxC f)
{   if(f.a!=0)
    {   printf("%f x**2",f.a);
        if(f.b!=0)
        {   if(f.b>0)   printf("+");
            printf("%fx",f.b);
        }
        if(f.c!=0)
        {   if(f.c>0)   printf("+");
            printf("%f\n",f.c);
        }
    }
    else { if(f.b!=0)   printf("%fx",f.b);
           if(f.c!=0)
           {if(f.c>0) printf(" +");
            printf("%f\n",f.c);
           }
    }
}
```

1-12．指出下列各算法的功能并求出其时间复杂度。

（1）

```
int prime(int n)
    {
      int i=2;
      int x=(int)sqrt(n);
      while(i<=x)
      {   if(n%i==0) break;
          i++;
      }
      if(i>x) return 1;
      else return 0;
    }
```

（2）

```
int sum1(int n)
    { int p=1,s=0;
      for(int i=1;i<=n;i++)
```

```
    { p*=i;
      s+=p;}
    return s;
  }
（3）int sum2(int n)
  { int s=0;
    for(int i=1;i<=n;i++)
    { int p=1;
      for(int j=1;j<=i;j++)
      p*=j;
      s+=p;}
    return s;
  }
（4）void print(int n)
  { int i;
     for(i=1;i<=n;i++)
     {
       if(i%2) printf("*");
       else continue;
       printf("#");
     }
  printf("$\n");}
（5）void print1(int n)
  { int i,k;
    long j;
    for(i=1;i<=n;i++)
    { j=i*i;
      if(j<10)
      {if(j==i)printf("%d %d\n",i,j);}
      else if(j<100)
      {if(j%10==i) printf("%d %d\n",i,j);}
      else if(j<1000)
      {if(j%100==i) printf("%d %d\n",i,j);}
      else if(j<10000)
      {if(j%1000==i) printf("%d %d\n",i,j);}
      else if(j<100000)
      {if(j%10000==i) printf("%d %d\n",i,j);}
      else
      {if(j%100000==i) printf("%d %d\n",i,j);}
    }
  }
（6）void matrix(int a[m][n],int b[n][l],int c[m][l])
  { int i,j,k;
    for(i=0;i<m;i++)
    for(j=0;j<l;j++)
    { c[i][j]=0;
```

```
            for(k=0;k<n;k++)
            c[i][j]+=a[i][k]*b[k][j];
        }
    }
（7）void xyx(int n)        //n<1000
    { int i,j,k,l;
      for (l=100;l<=n;l++)
      {
          i=l/100;
          j=l/10%10;
          k=l%10;
          if(k*100+j*10+i==l)
          printf("%d\n",l);
      }
    }
（8）int sum3(int n)
    { int i,s=0;
      for(i=1;i<=n;i++)
      if(i%2==1)
        s+=i;
      else s+=i*i;
      return s;
    }
```

解：

（1）功能：判断 n 是否为素数，时间复杂度为 $O(\sqrt{n})$。

（2）功能：求 $\sum_{i=1}^{n} i!$，时间复杂度为 O(n)。

（3）功能：求 $\sum_{i=1}^{n} i!$，时间复杂度为 $O(n^2)$。

（4）功能：输出 $\frac{n+1}{2}$ 组“*#”后，最后输出一个“$”，时间复杂度为 O(n)。

（5）功能：求 1000 以内的所有同构数（该数出现在它的平方的右边），时间复杂度为 O(n)。

（6）功能：求矩阵的乘积，时间复杂度为 O(m*n*l)。

（7）功能：求 1000 以内的回文数，时间复杂度为 O(n)。

（8）功能：求 1～n 中奇数的和与偶数的平方和，时间复杂度为 O(n)。

1.2.2 综合题

1-1. 将时间复杂度 O(1)、O(n)、$O(n^2)$、$O(n^3)$、$O(n\log_2 n)$、$O(\log_2 n)$、$O(2^n)$按递增顺序排列。

解：上述时间复杂度按递增顺序排列为 O(1)、$O(\log_2 n)$、O(n)、$O(n\log_2 n)$、$O(n^2)$、$O(n^3)$、$O(2^n)$。

1-2. 编程，从 n（最大为 100）个数据中求出最大值和最小值，并分析时间复杂度。