



普通高等教育“十三五”规划教材
计算机专业系列

数据结构 习题与实验指导

主 编 陈正铭 袁辉勇
副主编 刘 霖

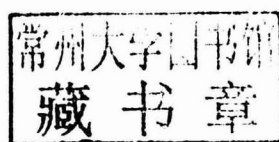


电子科技大学出版社

数据结构习题与实验指导

主 编 陈正铭 袁辉勇

副主编 刘 霖



 电子科技大学出版社

图书在版编目 (CIP) 数据

数据结构习题与实验指导 / 陈正铭, 袁辉勇主编

. —成都: 电子科技大学出版社, 2017. 7

ISBN 978 - 7 - 5647 - 4910 - 1

I. ①数… II. ①陈… ②袁… III. ①数据结构—教材 ②C 语言—程序设计—教材 IV. ①TP311.12②TP312.8

中国版本图书馆 CIP 数据核字 (2017) 第 183331 号

数据结构习题与实验指导

陈正铭 袁辉勇 主编

策划编辑 张 鹏

责任编辑 张 鹏

出版发行 电子科技大学出版社

成都市一环路东一段 159 号电子信息产业大厦九楼 邮编 610051

主 页 www.uestcp.com.cn

服务电话 028 - 83203399

邮购电话 028 - 83201495

印 刷 北京市彩虹印刷有限责任公司

成品尺寸 185mm × 260mm

印 张 25.25

字 数 450 千字

版 次 2017 年 7 月第 1 版

印 次 2017 年 7 月第 1 次印刷

书 号 ISBN 978 - 7 - 5647 - 4910 - 1

定 价 46.80 元

版权所有 侵权必究

前言

“数据结构”是计算机程序设计的重要理论基础，它不仅是计算机专业的核心课程，也是其他理工专业的热门选修课。在计算机的应用领域中，数据结构有着广泛的应用。市场上现有的“数据结构”教材多数侧重于原理的讲解，缺乏完整的算法实现、巩固性的习题与对应的实验指导，大多数学生在初学数据结构时，容易误解算法与程序之间的关系，经常会把书中的算法当作程序直接在编译器上进行运行测试。为解决此问题，编者根据 10 多年的教学经验与整理收集的课程资源，加上部分课程组独创的自测题与实验题目及指导，撰写了“数据结构”辅导书《数据结构习题与实验指导》。本书采用 C 语言作为数据结构和算法的描述语言，书中给出的每一个算法都是完整的，可在 VC6 ++ 6.0 上直接编译运行。考虑到专升本和其他相关考试的需要，在每章内容均包含详细的习题解析与适当的自测题，并配有自测题的答案，方便学生自学参考。

本书内容包括“数据结构”的重点知识点、典型习题解析、配套自测题、主要算法与实验指导（含参考源码），全书共分 8 章，可对应大多数的“数据结构”教材教学体系，第 1 章 绪论，介绍了数据结构的基本概念和算法分析的初步知识；第 2 章 线性表；第 3 章 栈和队列；第 4 章 串、数组和广义表；第 5 章 树和二叉树；第 6 章 图；第 7 章 查找；第 8 章 排序。本书第 1, 2, 3 章由陈正铭编写，第 4, 5, 6 章由袁辉勇编写，第 7, 8 章由刘霖编写，全书由陈正铭统一编排定稿，本书的编写得到戴经国教授的指导，得到数据结构课程组龙腾芳、刘晓樑、霍英等任课教师的组稿建议。书中所有程序由陈正铭整理编写，学生周浩、梁昭阳为程序做了验证性操作，并站在个人学习的角度对书稿的自测题提出了宝贵意见。学生王哲、胡静仪、钟智键等分别阅读了书稿中的部分章节，对书稿出现的一些编辑描述提出了质疑，在此一并表示衷心的感谢！

由于作者水平有限，书中难免会有不足和错误之处，敬请广大读者批评指正。

编者
2017 年 6 月

目 录

第1章 绪 论	1
1.1 本章知识点	1
1.2 习题解析	2
1.3 自测题	5
1.4 算法实现	8
1.5 实验1：预备实验——C 语言程序编程、调试的方法与结构体的应用	12
第2章 线性表	21
2.1 本章知识点	21
2.2 习题解析	23
2.3 自测题	31
2.4 算法实现	35
2.5 实验2：顺序表的实现	59
2.6 实验3：链表的实现	61
2.7 实验4：线性表及其应用——课堂经验值管理小程序	62
第3章 栈和队列	71
3.1 本章知识点	71
3.2 习题解析	72
3.3 自测题	86
3.4 算法实现	90
3.5 实验5：栈及其应用	118
3.6 实验6：队列的实现	129
第4章 串、数组和广义表	131
4.1 本章知识点	131
4.2 习题解析	135



4.3	自测题	142
4.4	算法实现	148
4.5	实验 7: 模式匹配算法的实现	154
第 5 章	树和二叉树	168
5.1	本章知识点	168
5.2	习题解析	173
5.3	自测题	183
5.4	算法实现	190
5.5	实验 8: 二叉树的递归遍历及应用	209
5.6	实验 9: 树及其应用——哈夫曼树	214
第 6 章	图	216
6.1	本章知识点	216
6.2	习题解析	219
6.3	自测题	230
6.4	算法实现	234
6.5	实验 10: 图的遍历算法	276
6.6	实验 11: 图及其应用	283
第 7 章	查 找	288
7.1	本章知识点	288
7.2	习题解析	291
7.3	自测题	301
7.4	算法实现	305
7.5	实验 12: 查找及其应用	322
第 8 章	排 序	327
8.1	本章知识点	327
8.2	习题解析	330
8.3	自测题	339
8.4	算法实现	342
8.5	实验 13: 内部排序算法的实现与比较	363
参考文献		397

第1章 绪论

1.1 本章知识点

数据结构主要研究非数值计算的问题，即如何合理地组织数据、高效地处理数据。它是一门研究非数值计算程序设计中的操作对象以及它们之间的关系和操作的学科，是一门综合性的计算机专业基础课，前驱课程为程序设计语言与离散数学，后续课程有操作系统、数据库等。结构化程序设计使得人们认为程序设计的实质就是对所处理的问题选择一种合适的数据结构，并在此基础上设计一种好的算法（程序 = 数据结构 + 算法）。

基本概念：

数据（Data）——客观事物的符号表示，是所有能输入到计算机中并被计算机程序存储、加工处理的符号的总称。

数据元素（Data element）——数据的基本单位，也称结点（node）或记录（record）。

数据项（Data item）——组成数据元素的、有独立含义的、不可分割的数据最小单位，也称域（Field）。

数据对象（Data Object）——性质相同的数据元素的集合，是数据的一个子集。

数据结构（Data Structure）——相互之间存在一种或多种特定关系的数据元素的集合。

逻辑结构（Logical structure）——数据元素间抽象化的相互关系，与数据的存储无关，独立于计算机，是从具体问题抽象出来的数学模型。有以下类型：

集合（Set）——数据元素间除“同属于一个集合”约束外，无其他关系；

线性结构（Linear structure）——一对一关系；

树形结构（Tree structure）——一对多关系；

图形结构（Graph structure）——多对多关系；

存储结构（Storage structure）——数据对象在计算机存储器中的存储表示，也称物理结构。主要有以下类型：

顺序存储结构（Sequential storage structure）——借助元素在存储器中的相对位置来表示数据元素间的逻辑关系；

链式存储结构（Linked storage structure）——借助指示元素存储地址的指针表示数据元素间的逻辑关系；

数据类型（Data type）——一个值的集合以及定义在这个值集合上的一组操作的

总称。

抽象数据类型 (Abstract data type) ——一般指由用户定义的, 用以表示应用问题的数学模型, 以及定义在该模型上的一组操作的总称 (数据对象 + 数据关系 + 基本操作);

类 C 语言 ——伪码的一种, 精选了 C 语言的一个核心子集, 同时做了若干扩充修改, 以增强语言的描述功能。

算法 (Algorithm) ——为了解决某类问题而规定的一个有限长的操作序列, 必须满足 5 个特征: 有穷性、确定性、可行性、输入、输出。评价算法优劣的标准有 (算法设计的要求): 正确性、可读性、健壮性、高效性。

时间复杂度 (Time complexity) ——指程序运行从开始到结束所需要的时间度量 $T(n)$ 。但要精确地计算 $T(n)$ 是困难的, 因此引入渐进时间复杂度在数量上估计一个算法的执行时间, 使用大 O 记号表示的算法的时间复杂度, 称为算法的渐进时间复杂度。通常用 $O(1)$ 表示常数计算时间。常见的渐进时间复杂度有:

$$O(1) < O(\log_2 n) < O(n) < O(n \log_2 n) < O(n^2) < O(n^3) < O(2^n)$$

空间复杂度 (Space complexity) ——指程序运行从开始到结束所需的辅助存储量 $S(n)$ 。

1.2 习题解析

1. 简述下列概念: 数据、数据元素、数据项、数据对象、数据结构、逻辑结构、存储结构、抽象数据类型。

【解答】 请参考 1.1 本章知识点

2. 试举一个数据结构的例子, 叙述其逻辑结构和存储结构两方面的含义和相互关系。

【解答】 如一个位数为 200 位的长整数, 常用的程序设计语言都无对应的数据类型来定义它。采用数据结构的观点, 可以把该位数为 200 位的长整数的看作是由 200 个 1 位十进制整数构成的序列, 对应的逻辑结构就是线性结构, 存储结构可以采用顺序存储结构 (位数固定) 或链式存储结构 (位数不固定, 如 210 位的长整数)。

3. 简述逻辑结构的四种基本关系并画出它们的关系图。

【解答】 逻辑结构的四种基本关系为:

集合——数据元素间除“同属于一个集合”约束外, 无其他关系;

线性结构——一对一关系;

树形结构——一对多关系;

图形结构——多对多关系;

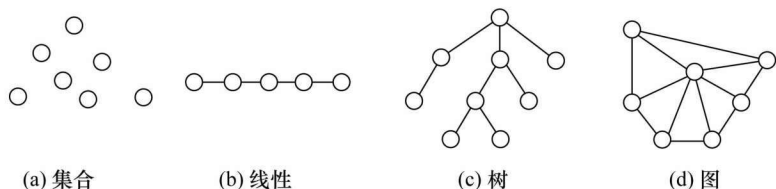


图 1-1 逻辑结构的四种基本形态

4. 存储结构由哪两种基本的存储方法实现?

【解答】存储结构可以由顺序存储结构和链式存储结构这两种基本的存储方法实现。

5. 选择题

(1) 在数据结构中,从逻辑上可以把数据结构分成()。

- A. 动态结构和静态结构
- B. 紧凑结构和非紧凑结构
- C. 线性结构和非线性结构
- D. 内部结构和外部结构

【解答】选 C。数据结构的逻辑结构除了可以分为:集合、线性结构、树形结构、图形结构,还可以分为线性结构和非线性结构,其中非线性结构包含:集合、树形结构、图形结构。

(2) 与数据元素本身的形式、内容、相对位置、个数无关的是数据的()。

- A. 存储结构
- B. 存储实现
- C. 逻辑结构
- D. 运算实现

【解答】选 C。存储结构是数据对象在计算机存储器中的存储表示,与数据元素本身的形式、内容、相对位置、个数均有关;存储实现实际也是存储结构的意思;运算实现即算法,也与问题规模即据元素的个数有关。

(3) 通常要求同一逻辑结构中的所有数据元素具有相同的特性,这意味着()。

- A. 数据具有同一特点
- B. 不仅数据元素所包含的数据项的个数要相同,而且对应数据项的类型要一致
- C. 每个数据元素都一样
- D. 数据元素所包含的数据项的个数要相等

【解答】选 B。相同特性指的就是数据元素本身所包含的数据项的类型与个数。

(4) 以下说法正确的是()。

- A. 数据元素是数据的最小单位
- B. 数据项是数据的基本单位
- C. 数据结构是带有结构的各数据项的集合
- D. 一些表面上很不相同的数据可以有相同的逻辑结构

【解答】选 D。数据元素是数据的基本单位,数据项是数据的最小单位,A 与 B 选项错误;数据结构是相互之间存在一种或多种特定关系的数据元素的集合,不是数据项的集合,C 选项错误。

(5) 以下与数据的存储结构无关的术语是()。

- A. 顺序队列
- B. 链表
- C. 有序表
- D. 链栈

【解答】选 C。存储结构指数据对象在计算机存储器中的存储表示,也称物理结构。主要有以下类型:顺序存储结构和链式存储结构。A 选项“顺序队列”为顺序存储结构;B 选项“链表”为链式存储结构;D 选项“链栈”为链式存储结构;C 选项“有序表”指的是按值大小排列而成的线性表,并未涉及数据的存储结构,因此选 C。

(6) 以下数据结构中,()是非线性数据结构

- A. 树
- B. 字符串
- C. 队
- D. 栈

【解答】选 A。非线性结构包含：集合、树形结构、图形结构。字符串、队、栈均为特殊的线性结构。

6. 试分析下面各程序段的时间复杂度。

```
(1) x = 90; y = 100;
    while(y > 0)
        if(x > 100) {x = x - 10; y -- ; }
        else x ++;
```

【解答】该程序段的各语句执行次数是固定的，与问题规模 n 无关，因此时间复杂度为： $O(1)$ 。

```
(2) for (i = 0; i < n; i ++ )
    for (j = 0; j < m; j ++ )
        a[i][j] = 0;
```

【解答】该程序段的 $a[i][j] = 0$ 语句执行了 $n * m$ 次，算法执行时间为 $T(n) = n * m$ ，因此时间复杂度为： $O(n * m)$ 。

```
(3) s = 0;
    for (i = 0; i < n; i ++ )
        for (j = 0; j < n; j ++ )
            s += B[i][j];
    sum = s;
```

【解答】该程序段的 $s = 0$ 和 $sum = s$ 各执行了一次， $s += B[i][j]$ 语句执行了 $n * n$ 次，算法执行时间为 $T(n) = 1 + n * n + 1$ ，因此时间复杂度为： $O(n * n)$ 即 $O(n^2)$ 。

```
(4) i = 1;
    while(i <= n)
        i = i * 3;
```

【解答】该程序段的 $i = 1$ 执行了一次， $i = i * 3$ 语句执行了 $\log_3 n$ 次，算法执行时间为 $T(n) = 1 + \log_3 n$ ，因此时间复杂度为： $O(\log_3 n)$ 。

```
(5) x = 0;
    for(i = 1; i < n; i ++ )
        for (j = 1; j <= n - i; j ++ )
            x ++;
```

【解答】该程序段的 $x = 0$ 执行了一次， $x ++$ ；语句执行了 $\sum_{i=1}^n \sum_{j=1}^{n-i} 1 = n(n-1)/2$ 次，算法执行时间为 $T(n) = 1 + n(n-1)/2$ ，因此时间复杂度为： $O(n^2)$ 。

```
(6) x = n; //n > 1
    y = 0;
    while(x >= (y + 1) * (y + 1))
        y ++;
```

【解答】该程序段的 $x = n; y = 0$; 各执行了一次, $y++$ 语句执行了 $\sqrt{n-1}$ 次, 算法执行时间为 $T(n) = 2 + \sqrt{n-1}$, 因此时间复杂度为: $O(\sqrt{n})$ 。

1.3 自测题

1. 填空题

- (1) _____ 是数据的基本单位, 在计算机程序中通常作为一个整体进行考虑和处理。
- (2) _____ 指的是解决问题的有限运算序列。
- (3) 从逻辑关系上讲, 数据结构主要分为_____、_____、_____和_____。
- (4) 数据的存储结构主要有_____和_____两种基本方法, 不论哪种存储结构, 都要存储两方面的内容: _____和_____。
- (5) 算法具有五个特性, 分别是 _____、_____、_____、_____、_____。
- (6) 算法的描述方法通常有_____、_____、_____和_____四种。
- (7) 在一般情况下, 一个算法的时间复杂度是_____的函数。
- (8) 设待处理问题的规模为 n , 若一个算法的时间复杂度为一个常数, 则表示成数量级的形式为_____, 若为 $n * \log_2 5n$, 则表示成数量级的形式为_____。
- (9) 算法在发生非法操作时可以作出处理的特性称为_____。

2. 选择题

- (1) 顺序存储结构中数据元素之间的逻辑关系是由 () 表示的, 链接存储结构中的数据元素之间的逻辑关系是由 () 表示的。
A. 线性结构 B. 非线性结构 C. 存储位置 D. 指针
- (2) 假设有如下遗产继承规则: 丈夫和妻子可以相互继承遗产; 子女可以继承父亲或母亲的遗产; 子女间不能相互继承。则表示该遗产继承关系的最合适的数据结构应该是 ()。
A. 树 B. 图 C. 线性表 D. 集合
- (3) 算法指的是 ()。
A. 对特定问题求解步骤的一种描述, 是指令的有限序列
B. 计算机程序
C. 解决问题的计算方法
D. 数据处理
- (4) 下面 () 不是算法所必须具备的特性。
A. 有穷性 B. 确切性 C. 高效性 D. 可行性
- (5) 算法分析的目的是 (), 算法分析的两个主要方面是 ()。
A. 找出数据结构的合理性 B. 研究算法中输入和输出的关系
C. 分析算法的效率以求改进 D. 分析算法的易读性和文档性
E. 空间性能和时间性能 F. 正确性和简明性

G. 可读性和文档性

H. 数据复杂性和程序复杂性

3. 判断题

- () (1) 算法的时间复杂度都要通过算法中的基本语句的执行次数来确定。
- () (2) 每种数据结构都具备三个基本操作：插入、删除和查找。
- () (3) 所谓数据的逻辑结构指的是数据之间的逻辑关系。
- () (4) 逻辑结构与数据元素本身的内容和形式无关。
- () (5) 基于某种逻辑结构之上的基本操作，其实现是唯一的。

4. 分析以下各程序段，并用大 O 记号表示其执行时间。

(1) $i = 1; k = 0$

while ($i < n - 1$)

{

$k = k + 10 * i;$

$i++;$

}

(2) $i = 1; k = 0;$

do

{

$k = k + 10 * i;$

$i++;$

}while($i \leq n$)

(3) $i = 1; j = 0$

while($(i + j) \leq n$)

 if($i > j$) $j++;$

 else $i++;$

(4) $y = 0;$

while($(y + 1) * (y + 1) \leq n$)

$y = y + 1;$

(5) for($i = 1; i \leq n; i++$)

 for($j = 1; j \leq i; j++$)

 for($k = 1; k \leq j; k++$)

$x++;$

5. 设有数据结构 (D, R) ，其中 $D = \{1, 2, 3, 4, 5, 6\}$ ， $R = \{(1, 2), (2, 3), (2, 4), (3, 4), (3, 5), (3, 6), (4, 5), (4, 6)\}$ 。试画出其逻辑结构图并指出属于何种结构。

6. 将下列函数按它们在 n 时的无穷大阶数，从小到大排列。

$n, n - n^3 + 7n^5, n \log n, 2^{n/2}, n^3, \log_2 n, n^{1/2} + \log_2 n, (3/2)^n, n!, n^2 + \log_2 n$

7. 对下列用二元组表示的数据结构，试分别画出对应的逻辑结构图，并指出属于何种结构。

(1) $A = (D, R)$ ，其中 $D = \{a_1, a_2, a_3, a_4\}$ ， $R = \{ \}$

(2) $B = (D, R)$ ，其中 $D = \{a, b, c, d, e, f\}$ ， $R = \{(a, b), (b, c), (c, d), (d, e), (e, f)\}$

(3) $C = (D, R)$ ，其中 $D = \{a, b, c, d, e, f\}$ ， $R = \{(d, b), (d, g), (b, a), (b, c), (g, e), (g, h)\}$

(4) $D = (D, R)$ ，其中 $D = \{1, 2, 3, 4, 5, 6\}$ ， $R = \{(1, 2), (1, 4), (2, 3), (2, 4), (3, 4), (3, 5), (3, 6), (4, 6)\}$

自测题参考答案

1. 填空题答案

- (1) 数据元素;
- (2) 算法;
- (3) 集合, 线性结构, 树结构, 图结构;
- (4) 顺序存储结构, 链接存储结构, 数据元素, 数据元素之间的关系;
- (5) 输入, 输出, 有穷性, 确定性, 可行性;
- (6) 自然语言, 程序设计语言, 流程图, 伪代码;
- (7) 问题规模;
- (8) $O(1)$, $O(n \log_2 n)$;
- (9) 健壮性

2. 选择题答案

- (1) C, D
- (2) B
- (3) A
- (4) C
- (5) C, E

3. 判断题答案

- (1) 错。时间复杂度要通过算法中基本语句执行次数的数量级来确定。
- (2) 错。如数组就没有插入和删除操作。此题注意是每种数据结构。
- (3) 对。
- (4) 对。
- (5) 错。基本操作的实现是基于某种存储结构设计的, 因而不是唯一的。

4. (1) $T(n) = O(n)$ 。

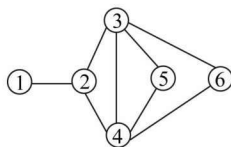
(2) $T(n) = O(n)$ 。

(3) $T(n) = O(n)$ 。

(4) $T(n) = O(\sqrt{n})$ 。

(5) $T(n) = O(n^3)$ 。

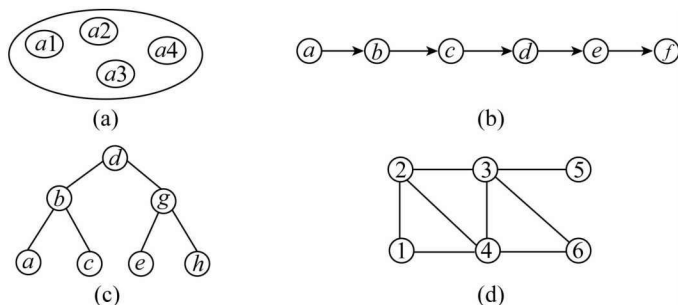
5. 其逻辑结构图如下图所示, 它是一种图结构。



6. $\log_2 n$, $n^{1/2} + \log_2 n$, n , $n \log_2 n$, $n^2 + \log_2 n$, n^3 , $n - n^3 + 7n^5$, $2^{n/2}$, $(3/2)^n$, $n!$

7. (1) 属于集合, 其逻辑结构图如 (a) 所示;

- (2) 属于线性结构，其逻辑结构图如 (b) 所示；
 (3) 属于树结构，其逻辑结构图如 (c) 所示；
 (4) 属于图结构，其逻辑结构图如 (d) 所示。



1.4 算法实现

【算法 1.1】求两个 n 阶矩阵的乘积

//求两个 3 阶矩阵的乘积

```
#include <iostream>
```

```
using namespace std;
```

```
void matproduct(int a[][3], int b[][3], int c[][3], int n)
```

```
{
```

```
    for(int i = 0; i < n; ++i)
```

```
        for(int j = 0; j < n; ++j)
```

```
        {
```

```
            c[i][j] = 0;
```

```
            for(int k = 0; k < n; ++k)
```

```
                c[i][j] = c[i][j] + a[i][k] * b[k][j];
```

```
        }
```

```
}
```

```
int main()
```

```
{
```

```
    int a[3][3] = {{1, 1, 1}, {2, 2, 2}, {3, 3, 3}};
```

```
    int b[3][3] = {{4, 4, 4}, {5, 5, 5}, {6, 6, 6}};
```

```
    int c[3][3] = {0};
```

```
    matproduct(a, b, c, 3);
```

```
    for(int i = 0; i < 3; ++i)
```

```
{
    for(int j = 0; j < 3; ++j)cout << c[i][j] << '\t';
    cout << endl;
}
return 0;
}
```

【算法 1.2】 交换 x 和 y 的值

```
//交换 x 和 y 的值
#include <iostream>
using namespace std;

void main()
{
    int x, y, temp;
    cout << "请输入 x 和 y 的值,用空格隔开:";
    cin >> x >> y;
    temp = x;
    x = y;
    y = temp;
    cout << "x = " << x << "   y = " << y << endl;
}
```

【算法 1.3】 变量计数之一

```
//变量计数之一
#include <iostream>
using namespace std;

void main()
{
    int x = 0, y = 0, n;
    cout << "请输入 n 的值:";
    cin >> n;
    for(int k = 1; k <= n; k++)
        x++;
    for(int i = 1; i <= n; i++)
        for(int j = 1; j <= n; j++)
            y++;
    cout << "x = " << x << "   y = " << y << endl;
}
```

【算法 1.4】 变量计数之二

```
//变量计数之二
#include <iostream>
using namespace std;

void main()
{
    int x = 0, n;
    cout << "请输入 n 的值:";
    cin >> n;
    for(int i = 1; i <= n; i++)
        for(int j = 1; j <= n; j++)
            for(int k = 1; k <= n; k++)
                x++;
    cout << "x = " << x << endl;
}
```

【算法 1.5】 顺序查找

```
//顺序查找
#include <iostream>
using namespace std;

int Search(int a[], int n, int e)
{
    for(int i = 0; i < n; i++)
        if(a[i] == e) return i + 1;
    return 0;
}

void main()
{
    int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    int e, pos;
    cout << "请输入要查找的元素的值:";
    cin >> e;
    pos = Search(a, 10, e);
    if (pos == 0) cout << "查找失败!" << endl;
    else cout << "元素 " << e << " 在第" << pos << "位置。" << endl;
}
```

【算法 1.6】复数的抽象数据类型实现

```
//复数的运算实现
#include <iostream>
using namespace std;

typedef struct Complex{
    float Realpart;
    float Imagepart;
}Complex;

void Creat(Complex &C, float x, float y)
{
    C. Realpart = x;
    C. Imagepart = y;
}

float GetReal(Complex C)
{
    return C. Realpart;
}

float GetImage(Complex C)
{
    return C. Imagepart;
}

Complex Add(Complex C1, Complex C2)
{
    Complex sum;
    sum. Realpart = C1. Realpart + C2. Realpart;
    sum. Imagepart = C1. Imagepart + C2. Imagepart;
    return sum;
}

Complex Sub(Complex C1, Complex C2)
{
```