QUANGUO SHUOSHIYANJIUSHENG RUXUEKAOSHI JISUANJIZHUANYE

全国硕士研究生入学考试计算机专业

统考考前辅导教程

—数据结构

段卫华　刘志鹏◎编著

- **考点辅导** 略去与统考大纲无关的数据结构知识点，突出常考知识与核心知识，对考点、重点、难点内容进行解释与讲述，让考生掌握问题的本质
- **典型例题分析** 针对新大纲中只有单项选择题和综合应用题的特点，精选常考题型与往年考试真题进行解析，增强考生的解题能力
- **同步练习** 对介绍的知识点给出一定数量的习题，便于考生复习与检查
- **同步练习答案** 考生通过对参考答案的分析和领会，可进一步加深对所学内容的理解，旨在达到启发解题思路的目的

清华大学出版社

全国硕士研究生入学考试计算机专业统考考前辅导教程——数据结构

段卫华　刘志鹏　编著

清华大学出版社
北　京

内 容 简 介

本书根据《2009 年全国硕士研究生入学统一考试计算机科学与技术学科联考计算机学科专业基础综合考试大纲》中的要求，参考全国著名高等院校近几年研究生入学考试计算机专业课试题编写而成。全书共 7 章，主要讨论了线性表、栈、队列、树和二叉树以及图等基本类型的数据结构及其应用，分析了查找和排序的各种实现方法。最后一章根据大纲要求编写了三套模拟试卷，并给出了参考答案。

本书章节安排与最新考试大纲同步，主要从考试大纲要求、考试要点、典型例题分析和同步练习训练等几方面对知识点加以系统阐释，可以帮助考生系统地理解和掌握考试大纲中的各个考点，通过实战练习提高考生的应试能力。

本书内容丰富、资料翔实、例题典型、讲解精当，特别适合参加全国研究生入学考试计算机统考的考生在考前复习使用，也可供大专院校计算机专业师生以及相应层次的计算机技术人员学习和参考。

图书在版编目(CIP)数据

全国硕士研究生入学考试计算机专业统考考前辅导教程——数据结构/段卫华，刘志鹏编著. —北京：清华大学出版社，2009.1
ISBN 978-7-302-18969-5

Ⅰ. 全… Ⅱ. ①段… ②刘… Ⅲ. ①电子计算机—研究生—入学考试—自学参考资料 ②数据结构—研究生—入学考试—自学参考资 Ⅳ. TP3

中国版本图书馆 CIP 数据核字(2008)第 180724 号

责任编辑：章忆文　宋延清
封面设计：杨玉兰
责任印制：杨　艳
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 刷 者：北京密云胶印厂
装 订 者：三河市溧源装订厂
经　　销：全国新华书店
开　　本：190×260　**印　张**：18.75　**字　数**：447 千字
版　　次：2009 年 1 月第 1 版　　**印　　次**：2009 年 1 月第1 次印刷
印　　数：1～5000
定　　价：29.80 元

前　言

教育部对 2009 年硕士研究生入学考试计算机科学与技术学科的初试科目进行了调整，其中计算机学科专业基础综合科目实行联合命题，由教育部考试中心和中国学位与研究生教育学会工科工作委员会组织实施，进行全国统一命题，由省级招生考试机构统一组织阅卷。新《考试大纲》对考试范围、方法和要求做出明确规定，是考试命题和考生准备考试的基本依据。

《考试大纲》给出了计算机专业基础综合考试试题的分布情况：卷面满分值为 150 分，包括数据结构、计算机组成原理、操作系统和计算机网络四大部分，数据结构和计算机组成原理各占 45 分，操作系统占 35 分，计算机网络占 25 分。统考中只有两种题型：单项选择题和综合应用题。其中选择题占 80 分，综合应用题占 70 分。从样题看来，针对操作系统和计算机网络的综合应用题相对较易，重点、难点仍是数据结构和计算机组成原理。

数据结构是考试中的重点内容。为了帮助考生全面系统地理解和掌握考试大纲中数据结构部分的各个考点，我们依据《2009 年全国硕士研究生入学统一考试计算机科学与技术学科联考计算机学科专业基础综合考试大纲》，参考了全国著名高等院校近几年的研究生入学考试计算机专业课试题，同时结合编者的工作经验编写了这本考前辅导教程。

本书定位于考生考前复习使用，主要从应试的角度出发，在编写的过程中以“考点讲解、试题分析”为主线，以“辅导与训练并重，习题与分析结合”为编写原则；在内容上不求完整性、系统性，而是将“针对性”视为重中之重，把与考试无关的内容大刀阔斧地略去，引导考生花较少的时间顺利通过考试。本书每章包括 4 个板块：考点辅导、典型例题分析、同步练习、同步练习答案。

- ☑ 考点辅导：略去与统考大纲无关的数据结构知识点，突出常考知识与核心知识，对考点、重点、难点内容进行解释与讲述，让考生掌握问题的本质。
- ☑ 典型例题分析：针对新大纲中只有单项选择题和综合应用题的特点，精选常考题型和全国著名 60 所高校近 3 年来的考题进行解析，分析解题思路，找出解题方法，让考生掌握解题方法与技巧，增强考生的解题能力。
- ☑ 同步练习：对介绍的知识点给出一定数量的习题，加强考生对知识点的理解。
- ☑ 同步练习答案：考生通过对参考答案的分析和领会，可进一步加深对所学内容的理解，旨在达到启发解题思路的目的。

全书共分 7 章，章节安排与最新考试大纲同步，主要从考试大纲要求、考试要点、典型例题分析和同步练习训练等几方面对知识点加以系统地阐述和解释。

第 1 章主要讨论线性表的定义和基本操作及其顺序、链式存储结构实现，并在此基础上给出线性表的应用。

第 2 章主要讨论栈和队列的基本概念、存储结构和应用，以及特殊矩阵的压缩存储。

第 3 章主要讨论树的概念，二叉树的定义、存储结构和遍历方法，线索二叉树、二叉排序树和平衡二叉树的构造，树与森林的存储结构、遍历方法，以及树的应用等。

第 4 章主要讨论图的概念、存储结构、遍历方法以及图的基本应用和复杂度分析。

第 5 章主要讨论查找的基本概念、顺序查找法、折半查找法、B-树、散列表，以及查找算法的分析和应用。

第 6 章主要讨论排序的基本概念，插入排序、气泡排序、简单选择排序、希尔排序、快速排序、堆排序、二路归并排序和基数排序等各种排序算法，以及各种排序算法的比较和应用。

第 7 章根据大纲的要求编写了三套模拟试卷，并给出了参考答案。

本书由段卫华、刘志鹏编著。另外感谢骆健、郭剑、邱丽姣、阚德涛、王珊珊、陈芳、郭沛仪、李勇智、张凌云、谢波等同志的关心和帮助。

由于作者水平有限，书中难免有不当之处，恳请广大读者批评指正。任何批评和建议请发至：book21press@126.com。

最后预祝广大考生在研究生入学考试中取得理想的成绩。

编　者

目　录

第1章 线 性 表

本章大纲要求

- 线性表的定义和基本操作
- 线性表的实现
  1. 顺序存储结构
  2. 链式存储结构
  3. 线性表的应用

重点考点提示

根据对最新考试大纲和各大高校历年试卷的分析可知，本章考核内容约占数据结构部分的15%。主要考核以下几个方面：

- 线性表两种存储结构各自的特点及比较。
- 链式存储结构中指针的操作。
- 链式存储结构中单链表、带表头结点的单链表、循环链表、双向循环链表等结构的应用场合。

1.1 线性表的定义和基本操作

1.1.1 考点辅导

考点1 线性表的基本的定义

线性表是n(n≥0)个元素 $a_0, a_1, \ldots, a_{n-1}$ 的有序集合，记为$(a_0, a_1, \ldots, a_{n-1})$。其中n是线性表中元素的个数，称为线性表的长度。n=0时称为空表。

线性表的基本特征清晰地反映了线性表数据结构的特点，其基本特征为在一个非空线性结构中，有且只有一个称为第一个的元素；有且只有一个称为最后一个的元素；第一个元素无前驱，最后一个元素无后继；其余每个元素均有唯一前驱和唯一后继。

从数学的角度看，线性表可以用一个二元组来表示：Linear_list=(D, R)，其中，D是数据元素的集合：$D=\{a_0, a_1, \ldots, a_{n-1}\}$；R是数据元素间关系的集合：$R=\{<a_{i-1}, a_i> | a_{i-1}, a_i \in D\}$。

考点2 线性表的基本操作

线性表的基本操作如下。

(1) CreateList(L)：构造一个空的线性表L，即表的初始化。

(2) Length(L)：求线性表L中的结点个数，即求表长。

(3) GetElem(L, pos)：取线性表L中的第pos个数据元素，这里要求$0 \leqslant pos < n$。

(4) Locate(L, x)：定位函数。在L中查找值为x的数据元素，并返回该数据元素在L中的位置。若L中有多个数据元素的值和x相同，则返回首次找到的数据元素位置；若L中没有数据元素的值为x，则返回一个特殊值表示查找失败。

(5) Insert(L, pos, x)：在线性表L的第pos个位置上插入一个值为x的新数据元素，使得原下标为pos到n-1处的数据元素依次向后移动。其中$0 \leqslant pos \leqslant n$，而n是原表L的长度。插入后，表L的长度加1。要特别注意插入的位置pos是否满足条件。

(6) Delete(L, pos)：删除线性表L的pos处的元素，使得原下标为pos+1到n-1的元素依次向

前移动。这里 0≤pos<n，而 n 是原表 L 的长度。删除后表 L 的长度减 1。

(7) Replace(L, pos, x)：将线性表中下标为 pos 处的元素更新为 x。这里 0≤pos<n，而 n 是原表 L 的长度。

(8) IsEmpty(L)：判断空表的函数。若 L 为空，则返回为真；否则返回为假。

(9) Clear(L)：将 L 清空。

1.1.2 典型例题分析

【例题 1】(东北大学)线性表是具有 n 个_____的有限序列(n≥0)。

A. 整数　　B. 字符　　C. 数据元素　　D. 数据项

分析：考查对线性表概念的理解。数据元素是数据结构逻辑上有意义的基本单位，一个数据元素一般由若干数据项组成。

答案：C

【例题 2】(江苏大学)线性表是_____。

A. 一个有限序列，可以为空　　B. 一个有限序列，不能为空

C. 一个无限序列，可以为空　　D. 一个无序序列，不能为空

答案：A

【例题 3】(江苏大学)下面_____是一个线性表。

A. 由 n 个实数组成的集合　　B. 由 100 个字符组成的序列

C. 由所有整数组成的序列　　D. 邻接表

分析：A 是一个集合而不是线性表，C 不是一个有限序列。D 显然不是线性表，邻接表是图的一种存储结构。

答案：B

【例题 4】(南京邮电大学)以下与数据的存储结构无关的术语是_____。

A. 顺序表　　B. 链表　　C. 散列表　　D. 栈

分析：常见的数据结构，从存储实现上分为顺序存储结构、链式存储结构、索引存储结构和散列存储结构。其中选项 A 顺序表是顺序存储结构，选项 B 链表是链式存储结构，选项 C 散列表是散列存储结构，三者均与存储结构相关。而选项 D 是一种逻辑结构，与具体的存储结构无关。

答案：D

【例题 5】(中南大学)以下数据结构中，不是线性结构的是_____。

A. 队列　　B. 二叉树　　C. 稀疏矩阵　　D. 数组

分析：常见线性结构包括线性表、栈、队列、数组、稀疏矩阵等。二叉树是树形结构。

答案：B

【例题 6】(西北大学)链式存储在设计时，存储单元的地址_____。

A. 一定连续　　B. 一定不连续　　C. 不一定连续　　D. 部分连续，部分不连续

分析：常见的存储结构有顺序存储结构和链式存储结构。其中，顺序存储结构要求存储单元连续，而链式存储结构没有对存储单元的地址连续与否做要求。切莫简单地认为链式存储结构一定不连续。

答案：C

【例题 7】(武汉大学)设线性表有 n 个元素，以下操作中，_____在顺序表上实现比在链表上实现效率更高。

A. 输出第 i(1≤i≤n)个元素值　　B. 交换第 1 个元素与第 2 个元素的值

C. 顺序输出这 n 个元素的值　　D. 输出与给定值 x 相等的元素在线性表中的序号

分析：考查线性表两种存储结构——顺序表与链表的比较。

答案：A

1.1.3　同步练习

1. (上海大学)数据结构在计算机内存中的表示是指_____。

A. 数据的存储结构　　B. 数据结构

C. 数据的逻辑结构　　D. 数据元素之间的关系

2. (南京航空航天大学)下列选项中_____不是常见的线性结构。

A. 花名册　　B. 学生的高矮排序

C. 学校的行政机构划分　　D. 书中的页码

1.1.4　同步练习答案

1. **分析**：数据逻辑关系的描述是逻辑结构，数据在计算机内的存储表示是存储结构。数据结构研究的三个主要方面是逻辑结构、存储结构和运算。

答案：A

2. **答案**：C

1.2　线性表的实现

1.2.1　考点辅导

线性表的存储实现常见的有顺序存储结构和链式存储结构两种方法。

考点 1　顺序存储结构

实现顺序存储结构的方法是使用数组。数组把线性表的数据元素存储在一块连续地址空间的内存单元中，这样线性表中逻辑上相邻的数据元素在物理存储地址上也相邻。数据元素间的逻辑上的前驱、后继逻辑关系就表现在数据元素的存储单元的物理前后位置上，如图 1.1 所示。

a_0	a_1	a_2	a_3	…	a_{n-2}	a_{n-1}
0	1	2	3		n-2	n-1

图 1.1　顺序表存储结构示意图

数组的空间分配有静态分配和动态分配两种方式。静态分配存储空间的分配和释放由系统自动

完成，动态分配存储空间的申请和释放由程序员通过函数调用来完成。

提　示

不论静态还是动态，其功能都是向系统申请一块地址连续的空间。此处重点介绍静态分配方式。动态分配方式除了在创建和撤销操作方面与静态方式区别较大外，其他操作的实现与静态方式基本类似。

顺序表的结构体如下：

```
typedef struct list
{
    int Size, MaxSeqList;
    DataType Elements[MaxSize];
} SeqList;
```

其中，DataType 是顺序表中的数据类型，MaxSize 是初始时向系统申请数组的最大个数，MaxSeqList 是当前顺序表所能存放的最大个数(MaxSeqList≤MaxSize)，Size 表示当前顺序表中元素的个数(Size≤MaxSeqList)。

在顺序存储结构下，线性表的若干操作运算实现如下。

(1) 创建一个空的顺序表：

```
void CreateList(SeqList *lst, int maxsize)
{
    lst->Size = 0;
    lst->MaxSeqList = maxsize;
}
```

(2) 清空一个顺序表：

```
void Clear(SeqList *lst)
{
     lst->Size = 0;
}
```

(3) 判断是否为空：

```
BOOL IsEmpty(SeqList lst)
{
    return lst.Size==0;
}
```

(4) 判断是否为满：

```
BOOL IsFull(SeqList lst)
{
     return lst.Size==lst.MaxSeqList;
}
```

(5) 更新顺序表中的第 pos 个元素：

```
BOOL Replace(SeqList *lst, int pos, DataType x)
```

```
{
    if (pos<0 || pos >= lst->Size)
    {
        printf("Out of Bounds");
        return FALSE;
    }

    lst->Elements[pos] = x;
    return TRUE;
}
```

(6) 插入元素：

```
BOOL Insert(SeqList *lst, int pos, DataType x)
{
    int i;
    /* 判断是否已满 */
    if (IsFull(*lst))
    {
        printf("Overflow");
        return FALSE;
    }
    /* 判断pos的位置是否合法 */
    if (pos<0||pos > lst->Size)
    {
        printf("Out of Bounds");
        return FALSE;
    }
    /* 将原下标为[pos, lst->Size -1]元素依次向后移动 */
    for (i=lst->Size-1; i>=pos; i--)
        lst->Elements[i+1] = lst->Elements[i];
    /* 插入新元素，表长加1 */
    lst->Elements[pos] = x;
    lst->Size++;
    return TRUE;
}
```

(7) 删除元素：

```
BOOL Delete(SeqList *lst, int pos, DataType *x)
{
    int i;
    /* 判断是否为空 */
    if (IsEmpty(*lst))
    {
        printf("Underflow");
        return FALSE;
    }
    /* 判断pos的位置是否合法 */
    if (pos<0 || pos>lst->Size-1)
    {
```

```
        return FALSE;
    }
    *x = lst->Elements[pos];
    /* 将原下标为[pos+1, lst->Size-1]的元素依次向前移动 */
    for(i=pos+1; i<lst->Size; i++)
        lst->Elements[i-1] = lst->Elements[i];
    /* 表长减 1 */
    lst->Size--;
    return TRUE;
}
```

注 意

顺序表的操作中特别要注意插入和删除两个操作。插入和删除操作中都需要移动元素。在顺序表 pos 位置插入一个新元素的运算中，需要将下标为 pos 到 Size-1 的元素向后移动。当 pos=0 时需要移动 Size 个元素，这是最坏的情况；当 pos=Size 时，需要移动 0 个元素，这是最好的情况。设顺序中元素个数为 n，当在顺序表的任何一个位置插入元素的概率相等时，平均情况下需要移动元素次数为：

$$E_i = \frac{1}{(n+1)}\sum_{i=0}^{n}(n-i) = \frac{n}{2}$$

将顺序表 pos 处的元素删除的运算中，需要将下标为 pos+1 到 Size-1 的元素向前移动。当 pos=0 时需要移动 Size-1 个元素，这是最坏情况；当 pos=Size-1 时，需要移动 0 个元素，这是最好的情况。设顺序表中元素个数为 n，当在顺序表的任何一个位置插入元素的概率相等时，平均情况下需要移动元素次数为：

$$E_d = \frac{1}{n}\sum_{i=0}^{n-1}(n-i-1) = \frac{n-1}{2}$$

考点 2 线性表的链式存储结构

线性表的链式存储结构有单链表、带表头结点单链表、双向链表等具体实现。单链表的结构最为常用，这里重点讨论单链表的实现。单链表的结构如图 1.2(a)所示，每个结点的结构如图 1.2(b)所示。

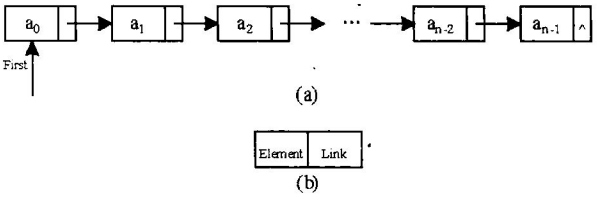

图 1.2 单链表的存储结构和结点结构示意图

单链表中每个结点的结构有两个域，Element 域存放结点元素，Link 域存放下一个结点地址。单链表结构中有两个域表头指针域 First 和表中元素个数 Size。

结点和单链表的描述如下：

```
typedef struct node
{
```

```
    DataType Element;          /* 数据元素 */
    struct node *Link;         /* 下一个结点地址 */
} Node;

typedef struct list
{
    Node *First;           /* 表头结点 */
    int Size;              /* 表中元素个数 */
} LinkList;
```

在单链表的存储结构下，线性表的若干操作运算实现如下。

(1) 创建一个空的单链表：

```
void CreateList(LinkList *lst)
{
    lst->First = NULL;
    lst->Size = 0;
}
```

(2) 清空一个单链表：

```
void Clear(LinkList *lst)
{
    Node *p = lst->First;
    while (p)
    {
        lst->First = p->Link;
        free(p);
        p = lst->First;
    }
    lst->Size = 0;
}
```

(3) 判断是否为空：

```
BOOL IsEmpty(LinkList lst)
{
    return lst.Size == 0;
}
```

(4) 判断是否为满：

```
BOOL IsFull(LinkList lst)
{
    BOOL result;
    Node *p = (Node*)malloc(sizeof(Node));
    if (p == NULL) result = TRUE;
    else
    {
        result = FALSE;
        free(p);
    }
```

```
    return result;
}
```

(5) 更新单链表中的第 pos 个元素：

```
BOOL Replace(LinkList *lst, int pos, DataType  x)
{
    Node *p = lst->First;
    int i;
    if (pos < 0 || pos >= lst->Size)
    {
        return FALSE;
    }
    for (i=0; i<pos; ++i)
    {
        p = p->Link;
    }
    p->Element = x;
    return TRUE;
}
```

(6) 插入元素：

```
BOOL Insert(LinkList *lst, int pos, DataType x)
{
    /* 两个指针 p 和 q */
    Node *p, *q = lst->First;
    int i;
    if (pos<0 || pos>lst->Size)
    {
        return FALSE;
    }
    /* 新生成一个结点 */
    p = (Node*)malloc(sizeof(Node));
    p->Element = x;
    p->Link = NULL;
    /* 根据 pos 的值判断是否需要修改头指针 First */
    if (pos==0)
    {
        /* 需修改头指针 First */
        p->Link = lst->First;
        lst->First = p;
    }
    else
    {
        /* 移动 q 到第 pos-1 个结点 */
        for (i=0; i<pos-1; ++i)
        {
            q = q->Link;
        }
        /* 将新生成 p 结点插入到 q 的后面 */
```

```
            p->Link = q->Link;
            q->Link = p;
        }
        lst->Size++;
        return TRUE;
}
```

(7) 删除元素：

```
BOOL Delete(LinkList *lst, int pos, DataType *x)
{
    Node *p, *q = lst->First;
    int i;
    if (IsEmpty(*lst))
    {
        return FALSE;
    }
    if (pos<0||pos>=lst->Size)
    {
        return FALSE;
    }
    if (pos==0)
        lst->First = lst->First->Link;
    else
    {
        /* q移动到要删除结点的前驱结点 */
        for (i=0; i<pos-1; ++i)
        {
            q = q->Link;
        }
        /* 修改指针 */
        p = q->Link;
        q->Link = p->Link;
    }
    /* 释放空间 */
    free(p);
    lst->Size--;
    return TRUE;
}
```

考点 3　线性表两种存储结构的比较

线性表有两种存储结构：顺序表和链式表。

(1) 顺序存储有 3 个优点。

① 方法简单，各种高级语言中都有数组，容易实现。

② 不用为表示结点间的逻辑关系而增加额外的存储开销。

③ 顺序表具有按元素序号随机访问的特点。

(2) 顺序存储也有两个缺点。

① 在顺序表中做插入、删除操作时，平均移动大约表中一半的元素，因此对规模较大的顺序