2013

考研

计算机学科专业基础综合

辅导讲义

主编：崔巍

副主编：蒋本珊　孙卫真　白龙飞

- 考纲要求提纲挈领
- 考点精讲层次分明
- 常考点明确重难点
- 例题设置紧扣大纲

北京航空航天大学出版社
BEIHANG UNIVERSITY PRESS

2013

考研

计算机专业基础综合

辅导讲义

主编：崔巍

副主编：蒋本珊　孙卫真　白龙飞

- 考纲要求提纲挈领
- 考点精讲层次分明
- 常考点明确重难点
- 例题设置紧扣大纲

北京航空航天大学出版社
BEIHANG UNIVERSITY PRESS

内 容 简 介

本书严格依据最新考研大纲的要求编写，分为四部分：第一部分为数据结构，第二部分为计算机组成原理，第三部分为操作系统，第四部分为计算机网络。每章内容包括考纲要求、考点精讲、常考点三个部分。在考纲要求中明确本章的主要知识点，阐述清晰；在考点精讲中对相关课程考纲的各个知识点进行集中讲解和提炼，以帮助考生有针对性的复习，并选择了典型例题及部分真题进行分析，方便考生对每部分知识的考核方式有所把握，加强考生的应试能力；常考点部分更是编者在研究历年真题的命题思路和统考的命题规律的基础上，列出本章重点考核的知识点及考查形式。本书适合所有计算机专业考研学子。

图书在版编目(CIP)数据

2013考研计算机学科专业基础综合辅导讲义 / 崔巍主编. -- 北京 : 北京航空航天大学出版社，2012.7

ISBN 978-7-5124-0772-5

Ⅰ. ①2… Ⅱ. ①崔… Ⅲ. ①电子计算机－研究生－入学考试－自学参考资料 Ⅳ. ①TP3

中国版本图书馆CIP数据核字(2012)第058230号

2013考研计算机学科专业基础综合辅导讲义

主　编　崔　巍

副主编　蒋本珊　孙卫真　白龙飞

策划编辑　谭　莉

责任编辑　杨国龙

*

北京航空航天大学出版社出版发行

北京市海淀区学院路37号(邮编100191)　http://www.buaapress.com.cn

发行部电话：(010)82317024　传真：(010)82328026

读者信箱：bhpress@263.net　邮购电话：(010)82316936

北京时代华都印刷有限公司印装　各地书店经销

*

开本：787×1 092　1/16　印张：26.0　字数：666千字

2012年7月第1版　2012年7月第1次印刷

ISBN 978-7-5124-0772-5　定价：49.80元

前　言

通过对2009—2012年连续四年全国硕士研究生入学统一考试计算机学科专业基础综合考试真题的研究，编者发现：计算机学科专业基础综合考查的重点是考生对专业基础知识、基本理论、基础方法的掌握水平及分析问题、解决问题的能力。所以，考生在复习备考此门课程时应将精力放在基本概念、基本原理与基本方法的融会贯通上，并力求熟练运用所学知识分析、判断和解决有关理论问题与实际问题。这也是编者编写本书的依据之一。

《考研计算机学科专业基础综合辅导讲义》一书自出版后受到全国各地考生的一致好评与推崇，为了帮助备考2013年计算机专业的考生在深入了解最新考试命题规律的基础上更好地把握计算机学科专业基础综合的复习要点，编者对最新的《全国硕士研究生入学统一考试计算机学科专业基础综合考试大纲》规定的考试内容和考试要求进行了深入分析，并结合多年计算机统考的命题理念和多年来对这些课程的潜心研究编写此书，以帮助同学们迅速抓住考试重点、掌握难点。

全书分为四个部分：第一部分为数据结构，第二部分为计算机组成原理，第三部分为操作系统，第四部分为计算机网络。每章内容包括考纲要求、考点精讲、常考点三个部分。在考纲要求中明确本章的主要知识点，阐述清晰；在考点精讲中对相关课程考纲的各个知识点进行集中讲解和提炼，以帮助考生有针对性的复习，并选择了典型例题及部分真题进行分析，方便考生对每部分知识的考核方式有所把握，加强考生的应试能力；常考点部分更是编者在研究历年真题的命题思路和统考的命题规律的基础上，列出本章重点考核的知识点及考查形式。

本书具有以下特点：

1. 考纲要求提纲挈领。每一章以“考纲要求”开始，以便考生了解该章知识的考试要求，整体把握复习侧重点。

2. 复习要点层次分明。“考点精讲”部分均逐层展开，脉络清楚，利于考生建立知识框架。

3. 明确重点考核内容。“常考点”部分简单明了地列出计算机学科专业基础综合考试中各章的考核重点及考查形式。

4. 内容讲述注重基础。知识点讲解以基础为中心，重视在基础中体现能力，充分体现大纲精神。

5. 例题设置紧扣大纲。为使考生充分掌握相关知识要点及考试出题规律而设的例题均围绕大纲要求编制。

参与本书编写的教师均为国家重点院校的长期从事计算机科学与技术学科相应本科生及研究生课程教学的一线教授和副教授，在相关课程中均具有十年以上的教学经历，并先后编写过多本教材和教学参考书。本书数据结构部分由崔巍老师编写，计算机组成原理部分由蒋本珊老师编写，操作系统部分由孙卫真老师编写，计算机网络部分由白龙飞老师编写。全书由崔巍老师统稿。

另外，编者作为计算机专业课的授课教师，在此也为准备参加2013年研究生入学考试计算机专业统考的同学给出一些复习建议。专业课的复习可分为以下三个阶段：

第一阶段：基础复习阶段（开始复习—2012 年 6 月）。这一阶段需要对“数据结构”、“计算机组成原理”、“操作系统”、“计算机网络”的教材仔细阅读一遍，了解四门课程的内容，理解每一个知识点，弄清每门课程的内在逻辑结构、重点章节等。这一阶段的复习要注意全面性。

第二阶段：强化提高阶段（2012 年 7 月—2012 年 11 月上旬）。这一阶段针对优秀的考研参考书进行深入复习，加强知识点的前后联系，建立整体框架结构。分清、整理、掌握重点和难点，完成参考书配有的习题，加深解题思路，提升解题速度。并且针对历年真题，整理真题答案，弄清每一道题属于教材中的哪一章、哪个知识点。通过做真题要了解考试形式、考试重点、题型设置和难易程度等内容，揣摩命题思路。这一阶段的复习要注意系统性。

第三阶段：冲刺阶段（2012 年 11 月中下旬—考前）。这一阶段总结所有重点知识点，包括重点概念、理论和模型等，查漏补缺。温习复习笔记和历年真题，分析真题的出题思路，预测本年度可能考查的内容和出题思路。多做模拟试卷，进一步归类整理总结。最后全面回顾知识点、易考题目及答案，准备应考。这一阶段的复习要注意目的性。

本书的编者为了更好地帮助考生复习，针对计算机专业课考试共编写了以下五本辅导教材：2013 考研计算机学科专业基础综合辅导讲义》、《2013 考研计算机学科专业基础综合考试大纲同步练习》、《2013 考研计算机学科专业基础综合考点速记手册》、《2013 考研计算机学科专业基础综合历年真题名师详解及 100 知识点聚焦》、《2013 考研计算机学科专业基础综合全真模拟试卷及精析》，其中《辅导讲义》、《同步练习》、《考点速记手册》这三本教材适用于考生在复习的各个阶段（基础阶段、强化阶段、冲刺阶段）中使用，《历年真题名师详解及 100 知识点聚焦》、《全真模拟试卷及精析》这两本适用于考生在复习的强化及冲刺阶段中使用。

在本书的编写过程中，参考了一些相关的书籍和资料，在此向这些书的作者表示深深的谢意。在编写、修改和出版本书的过程中，我们本着对考生高度负责的态度，精益求精，但由于编者水平有限，时间也比较仓促，尽管经过反复校对与修改，书中难免还存在错漏和不妥之处，敬请广大读者和专家批评指正，以便再版完善。

衷心地希望本书能帮助考生在考试中取得理想的成绩！圆梦 2013！

编　者

2012 年 7 月

目　录

第 1 部分

数据结构

第1章 绪 论

考纲要求

本章在考试大纲中并未明确指出，但是自2011年起连续两年真题均在本章出现一道选择题，主要考查算法时间复杂度的分析。

考点精讲

1.1 基本概念【理解】

1. 数据结构：是指相互之间存在着一种或多种关系的数据元素的集合。

数据结构是一个二元组 Data_Structure =(D,R)，其中，D是数据元素的有限集，R是D上关系的有限集。

2. 逻辑结构：是指数据之间的相互关系，通常分为四类结构。

(1) 集合结构：结构中的数据元素除了同属于一种类型外，别无其他关系。

(2) 线性结构：结构中的数据元素之间存在一对一的关系。

(3) 树型结构：结构中的数据元素之间存在一对多的关系。

(4) 图状结构：结构中的数据元素之间存在多对多的关系。

3. 存储结构：是指数据结构在计算机中的表示，又称为数据的物理结构，通常由四种基本的存储方法实现。

(1) 顺序存储方式。数据元素顺序存放，每个存储结点只含一个元素。这种方式存储位置反映数据元素间的逻辑关系。这种方式存储密度大，但有些操作(如插入、删除)效率较差。

(2) 链式存储方式。每个存储结点除包含数据元素信息外还包含一组(至少一个)指针。指针反映数据元素间的逻辑关系。这种方式不要求存储空间连续，便于动态操作(如插入、删除等)，但存储空间开销大(用于指针)，另外不能折半查找等。

(3) 索引存储方式。除数据元素存储在一组地址连续的内存空间外，还需建立一个索引表，索引表中索引指示存储结点的存储位置(下标)或存储区间端点(下标)。

(4) 散列存储方式。通过散列函数和解决冲突的方法，将关键字散列在连续的、有限的地址空间内，并将散列函数的值解释成关键字所在元素的存储地址。其特点是存取速度快，只能按关键字随机存取，不能顺序存取，也不能折半存取。

1.2 算法和算法分析【熟练掌握】

1. 算法是对特定问题求解步骤的一种描述，是指令的有限序列。其中每一条指令表示一个或多个操作。

算法具有的特性为：(1) 有穷性；(2) 确定性；(3) 可行性；(4) 输入；(5) 输出。

算法和程序十分相似，但又有区别。程序不一定具有有穷性，程序中的指令必须是机器可执行的，而算法中的指令则无此限制。算法代表了对问题的解，而程序则是算法在计算机上的特定的实现。一个算法若用程序设计语言来描述，则它就是一个程序。

2. 算法的时间复杂度：以基本运算的原操作重复执行的次数作为算法的时间度量。一般情况下，算法中基本运算次数 T(n)是问题规模 n(输入量的多少，称之为问题规模)的某个函数 f(n)，记作：T(n) = O(f(n))。也可表示 T(n) =m(f(n))，其中 m 为常量。记号“O”读作“大 O”，它表示随问题规模 n 的增大，算法执行时间 T(n)的增长率和 f(n)的增长率相同。

注意：有的情况下，算法中基本操作重复执行的次数还随问题的输入数据集不同而不同。

常见的渐进时间复杂度有：$O(1)<O(\log_2 n)<O(n)<O(n\log_2 n)<O(n^2)<O(n^3)<O(2^n)<O(n!)<O(n^n)$。

3. 算法的空间复杂度：是对一个算法在运行过程中临时占用的存储空间大小的量度。只需要分析除输入和程序之外的辅助变量所占额外空间。

原地工作：若所需额外空间相对于输入数据量来说是常数，则称此算法为原地工作，空间复杂度为 O(1)。

常考点

2009—2012 年全国考研试题本章出题情况

年 份	单项选择题/分值	综合应用题/分值	小计/分值	考核知识点
2009	0 题	0 题	0 分	
2010	0 题	0 题	0 分	
2011	0 题	1 题×2 分	2 分	算法的时间复杂度分析
2012	0 题	1 题×2 分	2 分	算法的时间复杂度分析

本章常考点：算法的性能分析，主要包括时间复杂度和空间复杂度的分析，是数据结构课程教学的基本要求，常以选择题的形式单独出现，另外在算法设计题中也会涉及。

第 2 章　线性表

考纲要求

（一）线性表的定义和基本操作

线性表的逻辑结构，是指线性表的数据元素间存在线性关系。主要是指：除第一及最后一个元素外，每个结点都只有一个前趋和一个后继。

（二）线性表的实现

1. 顺序存储结构

（1）线性表的顺序存储结构，靠元素存储的先后位置反映出数据元素的逻辑关系。

（2）用一维数组表示，给定下标，可以存取相应元素，属于随机存取的存储结构。

（3）线性表的顺序存储结构实现插入、删除、定位等运算的算法。

2. 链式存储结构

（1）线性表的链式存储结构，靠指针来反映数据元素的逻辑关系。

（2）链表的存取需要从头指针开始，顺链而行，不属于随机存取结构。

（3）几种常用链表的特点和相关算法设计：单链表、单循环链表、双向链表、双向循环链表的生成、检索、插入、删除、遍历、逆置、分解、归并等操作。

（4）从时间复杂度和空间复杂度的角度综合比较线性表在顺序和链式两种存储结构下的特点及其各自适用的场合。

3. 线性表的应用

运用顺序表和链表的特点解决复杂的应用问题。

考点精讲

2.1　线性表的定义【理解】

线性表是一种线性结构，在一个线性表中数据元素的类型是相同的，或者说线性表是由同一类型的数据元素构成的线性结构，定义如下：

线性表是具有相同数据类型的 $n(n\geqslant 0)$ 个数据元素的有限序列，通常记为：

$$(a_1, a_2, \cdots, a_{i-1}, a_i, a_{i+1}, \cdots, a_n)$$

其中，n 为表长，$n=0$ 时称为空表。

需要说明的是：a_i 为序号为 i 的数据元素（$i=1,2,\cdots,n$），通常将它的数据类型抽象为 ElemType，ElemType 根据具体问题而定。

2.2　线性表的实现

2.2.1　线性表的顺序存储结构【熟练掌握】

1. 顺序表

线性表的顺序存储是指在内存中用地址连续的一块存储空间顺序存放线性表的各元素，用这种存储形式存储的线性表称其为顺序表。因为内存中的地址空间是线性的，因此，用物理上的相邻实现数据元素之间的逻辑相邻关系是既简单又自然的。

设 a_1 的存储地址为 $Loc(a_1)$，每个数据元素占 d 个存储地址，则第 i 个数据元素的地址为：

$$Loc(a_i)=Loc(a_1)+(i-1)*d \qquad 1\leqslant i\leqslant n$$

这就是说只要知道顺序表首地址和每个数据元素所占地址单元的个数就可求出第 i 个数据元素的地址来，这也是顺序表具有按数据元素的序号随机存取的特点。

在程序设计语言中，一维数组在内存中占用的存储空间就是一组连续的存储区域，因此，用一维数组来表示顺序表的数据存储区域，如下描述。

线性表的静态分配顺序存储结构：

```
#define LISTSIZE 100                    //存储空间的最大分配量
typedef  struct{
    ElemType  elem[LISTSIZE];
    int  length;                        //当前长度
}SqList;
```

在线性表的静态分配顺序存储结构中，线性表的最多数据元素个数为 LISTSIZE，元素数量不能随意增加，这是以数组方式描述线性表的缺点。为了实现线性表最大存储数据元素数可随意变化，可以使用一个动态分配的数组来取代上面的固定长度数组，如下描述。

线性表的动态分配顺序存储结构：

```
#define LIST_INIT_SIZE 100              //存储空间的初始分配量
#define LISTINCREMENT 10                //存储空间的分配增量
typedef  struct{
    ElemType  *elem;                    //线性表的存储空间基址
    int  length;                        //当前长度
    int  listsize;                      //当前已分配的存储空间
}SqList;
```

2. 顺序表上基本运算的实现

(1) 顺序表的初始化

顺序表的初始化即构造一个空表，这是一个加工型的运算，因此，将 L 设为引用参数，首先动态分配存储空间，然后将 length 置为 0，表示表中没有数据元素。

```
int Init_SqList (SqList &L){
```

```
    L.elem=(ElemType * )malloc(LIST_INIT_SIZE * sizeof(ElemType));
    if (! L.elem) exit (OVERFLOW);          //存储分配失败
    L.length=0;
    L. listsize=LIST_INIT_SIZE;             //初始存储容量
    return OK;
}
```

(2) 插入运算

如图 1-2-1 所示,线性表的插入是指在表的第 i (i 的取值范围: $1 \leqslant i \leqslant n+1$)个位置上插入一个值为 x 的新元素,插入后使原表长为 n 的表:

$$(a_1, a_2, \cdots, a_{i-1}, a_i, a_{i+1}, \cdots, a_n)$$

成为表长为 n+1 表:

$$(a_1, a_2, \cdots, a_{i-1}, x, a_i, a_{i+1}, \cdots, a_n)$$

顺序表上完成这一运算则通过以下步骤进行:

① 将 $a_i \sim a_n$ 顺序向下移动,为新元素让出位置;(注意数据的移动方向:从后往前依次后移一个元素。)

② 将 x 置入空出的第 i 个位置;

③ 修改表长。

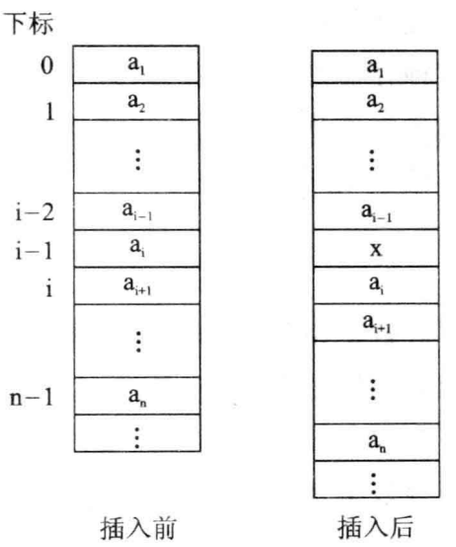

图 1-2-1 顺序表中的插入

```
int Insert_SqList (SqList &L,int i,ElemType x){
    if (i < 1 || i > L.length+1) return ERROR;        //插入位置不合法
    if (L.length >=L.listsize) return OVERFLOW;       //当前存储空间已满,不能插入
        //需注意的是,若是采用动态分配的顺序表,当存储空间已满时也可增加分配
    q=&(L.elem[i-1]);                                 //q 指示插入位置
    for (p=&(L.elem[L.length-1]); p >=q;--p)
      *(p+1)= *p;                                     //插入位置及之后的元素右移
    *q=e;                                             //插入 e
    ++L.length;                                       //表长增 1
    return OK;
}
```

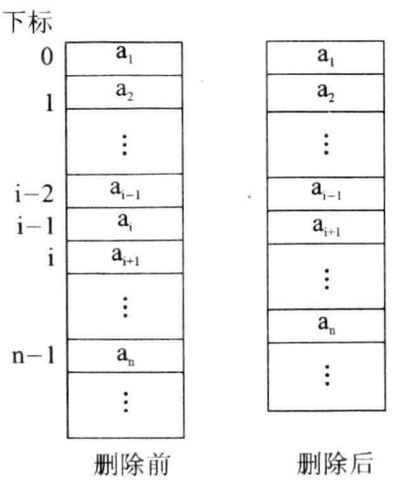

图 1-2-2 顺序表中的删除

顺序表上的插入运算,时间主要消耗在了数据的移动上,在第 i 个位置上插入 x,从 a_i 到 a_n 都要向下移动一个位置,共需要移动 n-i+1 个元素。

(3) 删除运算

如图 1-2-2 所示,线性表的删除运算是指将表中第 i (i 的取值范围为 : $1 \leqslant i \leqslant n$)个元素从线性表中去掉,删除后使原表长为 n 的线性表:

$$(a_1, a_2, \cdots, a_{i-1}, a_i, a_{i+1}, \cdots, a_n)$$

成为表长为 n-1 的线性表:

$$(a_1, a_2, \cdots, a_{i-1}, a_{i+1}, \cdots, a_n)$$

顺序表上完成这一运算的步骤如下：

① 将 a_{i+1}～a_n顺序向上移动；(注意数据的移动方向：从前往后依次前移一个元素。)

② 修改表长。

```
int Delete_SqList (SqList &L;int i) {
    if ((i < 1) || (i > L. length))   return ERROR;   //删除位置不合法
    p=&(L. elem[i-1]);                                //p 为被删除元素的位置
    e= * p;                                           //被删除元素的值赋给 e
    q=L. elem+L. length-1;                            //表尾元素的位置
    for (++p; p <=q;++p)
        *(p-1)= * p;                                  //被删除元素之后的元素左移
    --L. length;                                      //表长减 1
    return OK;
}
```

顺序表的删除运算与插入运算相同，其时间主要消耗在了移动表中元素上，删除第 i 个元素时，其后面的元素 a_{i+1}～a_n 都要向上移动一个位置，共移动了 n－i 个元素。

顺序表的插入与删除运算分析对比如表 1－2－1 所列。

表 1－2－1　顺序表插入和删除算法的分析

	插　入	删　除
基本操作	移动元素	移动元素
平均移动次数	$\frac{1}{n+1}\sum_{i=1}^{n+1}(n-i+1)=\frac{n}{2}$	$\frac{1}{n}\sum_{i=1}^{n}(n-i)=\frac{n-1}{2}$
时间复杂度	O(n)	O(n)
尾端操作	插入第 n+1 个元素，不移动	删除第 n 个元素，不移动

顺序表的插入、删除需移动大量元素，时间复杂度为 O(n)；但在尾端插入、删除效率高，时间复杂度为 O(1)。

(4) 按值查找

线性表中的按值查找是指在线性表中查找与给定值 x 相等的数据元素。在顺序表中完成该运算最简单的方法是：从第一个元素 a_1 起依次和 x 比较，直到找到一个与 x 相等的数据元素，则返回它在顺序表中的存储下标或序号(二者差 1)；或者遍历整个表都没有找到与 x 相等的元素，返回 ERROR。

```
int Locate_SqList (SqList L, ElemType x){
    i=0;
    while(i<=L. length-1 && L. elem[i]! =x)
        i++;
    if (i>L. length-1)   return ERROR;
    else return i;                    //返回的是存储位置
}
```

本算法的主要运算是比较，显然比较的次数与 x 在表中的位置有关，也与表长有关。当

$a_1=x$ 时，比较 1 次成功，当 $a_n=x$ 时，比较 n 次成功，按值查找的平均比较次数为(n+1)/2，时间性能为 O(n)。

2.2.2 线性表的链式存储结构【熟练掌握】

2.2.2.1 单链表

1. 链表表示

链表通过一组任意的存储单元(可以连续也可以不连续)来存储线性表中的数据元素。为建立起数据元素之间的线性关系，对每个数据元素 a_i，除了存放数据元素的自身的信息 a_i 之外，还需要和 a_i 一起存放其后继 a_{i+1} 所在的存储单元的地址，这两部分信息组成一个“结点”，结点的结构如图 1-2-3 所示。

其中，存放数据元素信息的称为数据域，存放其后继地址的称为指针域。因此，n 个元素的线性表通过每个结点的指针域拉成了一个“链”，称之为链表。因为每个结点中只有一个指向后继的指针，所以称其为单链表，如图 1-2-4 所示。

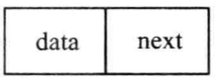

图 1-2-3 单链表结点结构

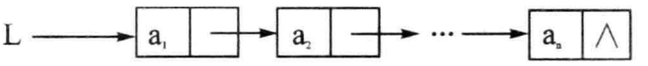

图 1-2-4 单链表示意图

线性表的单链表存储结构 C 语言描述下：

```
typedef struct LNode{
    ElemType      data;          //数据域
    struct LNode  * next;        //指针域
}LNode, * LinkList;
LinkList L;                      //L 为单链表的头指针
```

通常用“头指针”来标识一个单链表，如单链表 L、单链表 H 等，是指某链表的第一个结点的地址放在了指针变量 L、H 中，头指针为“NULL”则表示一个空表。

2. 单链表上基本运算的实现

(1) 建立单链表

● 头插法——在链表的头部插入结点建立单链表

链表与顺序表不同，它是一种动态管理的存储结构，链表中的每个结点占用的存储空间不是预先分配的，而是运行时系统根据需求而生成的，因此建立单链表从空表开始，每读入一个数据元素则申请一个结点，然后插在链表的头部。如图 1-2-5 展现了线性表：(1,2,3,4,5)在链表的头部插入结点建立链表的过程，因为是在链表的头部插入的，所以读入数据的顺序和线性表中的逻辑顺序是相反的。

```
LinkList  CreateListF ( ){
LinkList L=NULL;                 //空表
LNode * s;
int x;                           //设数据元素的类型为 int
scanf("%d",&x);
while (x! =flag) {
```