2013年
计算机专业基础
综合考试备考一本通

2013NIAN
JISUANJI ZHUANYE JICHU
ZONGHE KAOSHI BEIKAO
YIBENTONG

全国硕士研究生入学考试计算机专业基础联考命题研究组◎组编
吴敏　俞露　葛武滇◎主编

◎书中章节与最新官方指定相关教程一致。利于考生分类复习，专项攻克，同时也便于考生更好地理解和掌握考试的题型、分值、内容、范围及难度，便于考生把握命题规律，快速提升应试能力。

◎考点辅导。用考生易于理解的方
罗列知识点。设置有提示、注意

◎典型例题分析。对2009～2012年计算机统考真题进行分类解析；研究各类名校试题，从中选取典型考题进行分类解析，并将题目穿插在相应的章节中。

机械工业出版社
CHINA MACHINE PRESS

2013年计算机专业基础综合考试备考一本通

全国硕士研究生入学考试
计算机专业基础联考命题研究组　组编
吴敏　俞露　葛武滇　主编

机械工业出版社

本书以最新考试大纲为依据，从考点出发，精解典型例题（最新4次统考真题及全国60所高校近几年考题），以达到综合辅导、一本通关之功效。内容包括：数据结构、操作系统、计算机组成原理、计算机网络4部分。每章分为考点辅导和典型例题分析两个板块。本书章节安排与官方指定考试教程一致，利于考生分类复习，专项攻克。书中试题分类科学、分析细致、解答完整，并给出了点评与拓展，所总结的解题方法不仅有仿效的价值，还可开拓思路。

本书特别适合广大应试考生考前综合复习使用，也可作为各类研究生入学考试培训班的辅助资料，还可作为高等院校师生的教学参考书。

图书在版编目（CIP）数据

2013年计算机专业基础综合考试备考一本通 / 吴敏主编.
—北京：机械工业出版社，2012.9
ISBN 978-7-111-39818-9

Ⅰ.①2… Ⅱ.①吴… 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223883号

机械工业出版社（北京市百万庄大街22号 邮政编码100037）
策划编辑：吉 玲 责任编辑：吉 玲 刘丽敏
封面设计：鞠 杨 责任印制：李 妍
北京振兴源印务有限公司印刷
2012年10月第1版第1次印刷
184mm×260mm · 27.5印张 · 910千字
标准书号：ISBN 978-7-111-39818-9
定价：62.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务		网络服务	
社服务中心：	（010）88361066	教材网：	http://www.cmpedu.com
销售一部：	（010）68326294	机工官网：	http://www.cmpbook.com
销售二部：	（010）88379649	机工官博：	http://weibo.com/cmp1952
读者购书热线：	（010）88379203	封面无防伪标均为盗版	

前　言

随着改革开放和现代化建设事业的需要，特别是“科教兴国”、“知识经济”等战略性措施日益广泛实施，国家机关、企事业单位及各行各业对高素质、高学历人才的需求量越来越大。同时，随着高等教育的大众化，本科人才越来越多，相当一部分大学毕业生不易找到理想工作，很多人希望取得更高的学历，以增强自己的竞争实力，因此，近年来“考研热”持续升温。研究生入学考试现已成为国内影响最大、参加人数最多的国家级选拔高层次人才的水平考试。

1．编写目的

2009 年，教育部对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，删除了以往考研中经常出现的判断题，填空题等。而考试的范围加大到了四科，各科的分值相对降低了。广大应试考生需要相关的辅导书以熟悉统考的题型、分值、难度。为了更好地引导考生在较短时间内掌握解题要领，并顺利通过研究生入学考试，我们组织了一批具有多年教学经验的一线教师，根据最新考试大纲编写了这本综合类辅导书。

2．本书特色

（1）本书章节安排与最新官方指定教程一致，利于考生分类复习，专项攻克，同时也便于考生更好地理解和掌握考试的题型、分值、内容、范围及难度，便于考生把握命题规律，快速提升应试能力。

（2）考点辅导：对考点的讲解、归纳总结，从考生的角度出发，用考生易于理解的方式进行讲解，而不是简单的知识点的罗列。设置有提示、注意等特色版块。

（3）典型例题分析：①对 2009～2012 年计算机统考真题进行分类解析；②研究各类名校试题，从中选取典型考题进行分类解析，并将题目穿插在相应的章节中。

3．本书阅读指南

本书系统、全面地分析了近几年计算机专业课考研题目的解题思路，并给出了翔实的参考答案，读者可以充分地了解各个学校考研题目的难度，查缺补漏，有针对性地提高自己的水平。本书共分 24 章，主要内容如下。

第 1 章主要介绍线性表的定义、基本操作、线性表的实现。

第 2 章主要介绍栈和队列的基本概念，栈和队列的顺序存储结构，栈和队列的链式存储结构，栈和队列的应用，以及特殊矩阵的压缩存储。

第 3 章主要介绍树的概念，二叉树，树、森林，以及树的应用。

第 4 章主要介绍图的概念，图的存储及基本操作，图的遍历，图的基本应用及其复杂度分析。

第 5 章主要介绍查找的基本概念，顺序查找法，折半查找法，B-树，散列（Hash）表及其查找，查找算法的分析及应用。

第 6 章主要介绍排序的基本概念，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，基数排序，各种内部排序算法的比较，内部排序算法的应用。

第 7 章主要介绍计算机发展历程：计算机系统层次结构，包括计算机硬件的基本组成、计算机软件的分类、计算机的工作过程和计算机性能指标。

第 8 章主要介绍数制与编码，定点数的表示和运算，浮点数的表示和运算，算术逻辑单元（ALU）。

第 9 章主要介绍存储器的分类，存储器的层次化结构，半导体随机存取存储器，只读存储器，主存储器与 CPU 的连接，双口 RAM 和多模块存储器，高速缓冲存储器（Cache），虚拟存储器。

第 10 章主要介绍指令格式，指令的寻址方式，CISC 和 RISC 的基本概念。

第 11 章主要介绍 CPU 的功能和基本结构，指令执行过程，数据通路的功能和基本结构，控制器的功能和工作原理，指令流水线。

第 12 章主要介绍总线概述，总线仲裁，总线操作和定时，总线标准。

第 13 章主要介绍 I/O 系统基本概念，外部设备，I/O 接口，I/O 方式。

第 14 章主要介绍操作系统的概念、特征、功能和提供的服务，操作系统的发展与分类，操作系统的运行环境。

第 15 章主要介绍进程与线程，处理器调度，进程同步，死锁。

第 16 章主要介绍内存管理基础，虚拟内存管理。

第 17 章主要介绍文件系统基础，文件系统实现，磁盘组织与管理。

第 18 章主要介绍 I/O 管理概述，I/O 核心子系统。

第 19 章主要介绍计算机网络的基本概念，计算机网络体系结构与参考模型。

第 20 章主要介绍物理层的功能，通信基础知识，传输介质和物理层设备。

第 21 章主要介绍数据链路层的功能，组帧，差错控制，流量控制与可靠传输机制，介质访问控制，局域网，广域网，数据链路层设备。

第 22 章主要介绍网络层的功能，路由算法，IPv4，IPv6，路由协议，IP 组播，移动 IP，网络层设备。

第 23 章主要介绍传输层提供的服务，UDP，TCP。

第 24 章主要介绍网络应用模型，DNS 系统，FTP，电子邮件，WWW 的概念与组成结构。

4．读者对象

本书特别适合于希望在较短时间内取得较大收获的计算机专业应试考生，也可作为计算机学科专业研究生入学考试培训班的辅助教材，以及高等院校师生的教学参考书。

5．关于作者

本系列图书由全国硕士研究生入学考试计算机专业基础联考命题研究组组编，他们长期从事这方面的教学和研究工作，积累了丰富的经验，对考研颇有研究。本书由吴敏、俞露、葛武滇主编，参与本书组织、编写、审校和资料收集等工作的有（排名不分先后）：张伍荣、王珊珊、周海霞、王国全、许勇、许娟、何光明、何杨光、吴婷、张建林、李千目、李海、杨明、杨萍、汪志宏、陈玉旺、陈智、范荣钢、姚昌顺、赵传申、骆健、钱阳勇、童爱红等。

由于编者水平有限，书中错误及不妥之处在所难免，请广大师生批评指正，以便今后逐步完善和提高。

作 者

目　录

第 1 章　线性表

本章大纲要求

- 线性表的定义与基本操作
- 线性表的实现
- 特殊矩阵的压缩存储

重点考点提示

根据对最新考试大纲和历年真题的分析可知，本章考核内容约占数据结构部分的 10%。主要考核以下几个方面：

- 线性表的定义
- 线性表的基本操作
- 线性表的逆序与有序线性表的合并

1.1　线性表的定义与基本操作

1.1.1　考点辅导

考点 1　线性表的定义

线性表是最基本、最简单，也是最常用的一种数据结构。线性表中数据元素之间的关系是一对一的关系，即除了第一个和最后一个数据元素之外，其他数据元素都是首尾相接的。线性表的逻辑结构简单，便于实现和操作。因此，线性表在实际应用中是广泛采用的一种数据结构。线性表也可以抽象成如下定义。

线性表（Linear List）是由 n（n≥0）个数据元素（结点）a_1，a_2，…，a_n 组成的有限序列，其中：

1）数据元素的个数 n 定义为表的长度（n=0 时称为空表）。

2）将非空的线性表（n>0）记做（a_1，a_2，…，a_n）。

3）数据元素 a_i（1≤i≤n）只是一个抽象符号，其具体含义在不同情况下可以不同。

考点 2　线性表的基本操作

常见的线性表的基本运算为：

MakeEmpty（L）：将线性表 L 变为空表。

Length（L）：返回线性表 L 的长度，即表中元素个数。

Get（L，i）：一个函数，函数值为表 L 中位置 i 处的元素（1≤i≤n）。

Prev（L，i）：取 i 的前驱元素。

Next（L，i）：取 i 的后继元素。

Locate（L，x）：一个函数，函数值为元素 x 在表 L 中的位置。

Insert（L，i，x）：在表 L 的位置 i 处插入元素 x。

Delete（L，p）：从表 L 中删除位置 p 处的元素。

IsEmpty（L）：如果表 L 为空表（长度为 0），则返回 true；否则，返回 false。

Clear（L）：清除表 L 中所有元素。

Init（L）：初始化线性表为空。

Traverse（L）：遍历表 L，输出所有元素。

Find（L，x）：查找并返回元素。

Update（L，x）：修改元素 x。

Sort（L）：对所有元素重新按给定的条件排序。

提　示

线性表具有如下结构特点。

均匀性：虽然不同数据表的数据元素可以是各种各样的，但同一线性表的各数据元素必定具有相同的数据类型和长度。

有序性：各数据元素在线性表中的位置只取决于它们的序号，数据元素之间的相对位置是线性的，即存在唯一的“第一个”和“最后一个”的数据元素。除了第一个和最后一个外，其他元素前面均只有一个数据元素（直接前驱）和后面均只有一个数据元素（直接后继）。

在实现线性表数据元素的存储方面，一般可用顺序存储结构和链式存储结构。在实际应用中，线性表都是以栈、队列、字符串、数组等特殊线性表的形式来使用的。由于这些特殊线性表具有各自的特性，因此掌握这些特殊线性表的特性，对于数据运算的可靠性和提高操作效率是至关重要的。另外，栈、队列和串又称为受限的线性结构。

1.1.2　典型例题分析

【例题 1】（清华大学）下列说法中，正确的是_____。

A．线性表中所有数据元素的数据类型必须相同

B．线性表的顺序存储表示属于静态结构，链式存储表示属于动态结构

C．二维数组是一种非线性结构

D．广义表是线性表的推广，所以广义表是一种线性结构

分析：考查线性表、栈、队列、广义表的基本概念。A 选项错误，线性表中所有数据元素的类型可以不同，但必须使用链表存储。B 选项错误，线性表的顺序存储可以是一维静态数组，也可以是动态数组。C 选项正确，二维数组中每个元素最多有两个直接前驱和两个直接后继。D 选项错误，广义表的表元素可以是子表，所以它是一种非线性结构。

解答：C

【例题 2】（武汉大学）_____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：此题考查线性表的基本概念。A 选项不能准确定位当前元素的前驱与后继。B 选项无法实现有序性，也就是说，知道当前字符，无法确定下一字符是什么。该题属于常规定义题，具体参考 1.1.1 节的提示部分。

解答：C

【例题 3】（江苏大学）线性表是 n 个_____的有限序列。

A．字符　　B．数据元素　　C．数据项　　D．信息项

分析：线性表（Linear List）是由 n（n≥0）个数据元素（结点）a_1，a_2，…，a_n 组成的有限序列。

解答：B

【例题 4】（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．一个无限序列，不能为空

解答：A

【例题 5】（天津大学）试述线性表逻辑结构的形式化定义。

解答：（1）存在唯一的“第一个”的数据元素。（2）存在唯一的“最后一个”的数据元素。（3）除

第一个结点外，每个结点都有一个前驱结点。（4）除最后一个结点外，所有的结点都有一个后继。

【例题6】（北京师范大学）线性表的基本运算包括哪些？简述线性表的链式存储结构的几种形式。

解答：查找、插入、删除。链式存储主要包括单链表、双向链表和循环链表等几种。

1.2 线性表的实现

1.2.1 考点辅导

考点1 基本操作实现

线性表的实现一般分为两种，顺序实现和链式实现。

顺序实现，即把线性表的结点按逻辑次序依次存放在一组地址连续的存储单元中。顺序表是用向量实现的线性表，向量的下标可以看做结点的相对地址。因此，顺序表的特点是逻辑上相邻的结点其物理位置亦相邻。

链接方式存储的线性表简称为链表（Linked List）。

链表的具体存储表示为：

1)用一组任意的存储单元来存放线性表的结点(这组存储单元既可以是连续的,也可以是不连续的)。

2）链表中结点的逻辑次序和物理次序不一定相同。为了正确地表示结点间的逻辑关系，在存储每个结点值的同时，还必须存储指示其后继结点的地址（或位置）信息（称为指针（Pointer）或链（Link））。

线性表的顺序实现只需用一个数组，下面主要以链式存储来实现。

下面是链式存储实现的基本操作的源代码，主要实现了5个函数：链表的创建、链表长度的查询、链表的输出打印、链表的删除和链表的插入。

```
#include "stdafx.h"
#include <iostream>
#include <iomanip>
using namespace std;
#define DataType int
typedef struct Node
{
    struct Node *next;
    DataType data;
}Node;
//基本操作声明部分
    Node* create();
    int length(Node *head);
    void print(Node* head);
    Node* Delete(Node* head,DataType d_data);
    Node* Insert(Node* head,DataType newdata);
//基本操作实现部分
Node* create()
    {
        Node *head,*p,*q;
        head=new Node;
        head->data=0;
```

```
        head->next=NULL;
        p=head;
        DataType a;
        cout<<"Input data(0 marked end):"<<endl;
        while(cin>>a)
        {
            if(a)
            {
                q=new Node;
                q->data=a;
                q->next=NULL;
                p->next=q;
                p=q;
            }
            else break;
        }
        cout<<"Here is create() ending. "<<endl;
        p=head->next;
        delete head;
        head=p;
        return head;
    }
int length(Node *head)
    {
        Node *p=head;
        int count=0;
        while(p)
        {
            count++;
            p=p->next;
        }
        return count;
    }
void print(Node* head)
    {
        Node* p=head;
        cout<<"Now the list is: "<<endl;
        while(p)
        {
            cout<<p->data<<setw(6);
            p=p->next;
        }
        cout<<endl<<"Here is the print() ending!"<<endl;
    }
```

```
Node* Delete(Node* head,DataType d_data)
    {
        Node* p=head,*q=p;
        while(p && (p->data !=d_data))
        {
            q=p;
            p=p->next;
        }
        if(!p)
        {
            cerr<<"No the element:"<<d_data<<endl;
            return head;
        }
        if(p==head)
            head=p->next;
        else
            q->next=p->next;
        delete p;
        p=NULL;
        return head;
    }
Node* Insert(Node* head,DataType newdata)
    {
        Node* p=head,*q=head,*s;
        s=new Node;
        s->data=newdata;
        s->next=NULL;
        while(p && newdata>p->data)
        {
            q=p;
            p=p->next;
        }
        if(p==head)
        {
            s->next=head;
            head=s;
        }
        else
        {
            s->next=p;
            q->next=s;
        }
        return head;
    }
```

提 示

上面的插入排序中，是以有序表为基础进行插入的。另一种插入方法是在指定位置插入元素。实现时一定要先判断该指定位置是否存在。

考点 2 线性表的逆序

线性表中除了基本操作外，常考的另一种题目就是考查线性表的逆序。

此处要用到 3 个指针，第一个指针用来始终保存当前新序列头指针，第二个指针用来指向剩余线性表的第一个元素，第三个指针指向第二个指针的下一元素，用来判断剩余线性表中是否还有元素。实现代码如下：

```
Node* reverse_order(Node* head)
{
    if(head==NULL && head->next==NULL)
        return head;
    Node* p=head->next,*q,*r=head;
    while(p)
    {
        q=p->next;
        p->next=r;
        r=p;
        p=q;
    }
    head->next=NULL;
    head=r;
    return head;
}
```

考点 3 链表合并

链表的合并也是常考的应用型题目之一。常考的情况是：已知两个有序的线性链表，现在要合并成一个新的有序的链表，并且只用原来已分配的内存，而不再分配内存来存储新链表。

1）第一种情况，两链表均为从小到大的有序链表，现在将其合并为一个有序链表。

```
Node* merge_link(Node* head1,Node* head2)
{
    Node * head,* p,* p1=head1,* p2=head2,* q;
    if(head1 && head2)
    {
        if(head1->data>head2->data)
        {
            head=head2;
            p2=p2->next;
        }
        else
        {
            head=head1;
```

```
            p1=p1->next;
        }
        p=head;
    }
    else
    {
        if(head1)
            return head1;
        else
            return head2;
    }
    while(p1 && p2)
    {
        if(p1->data>p2->data)
        {
            p->next=p2;
            p=p2;
            p2=p2->next;
        }
        else
        {
            p->next=p1;
            p=p1;
            p1=p1->next;
        }
    }
    if(p1)
        p->next=p1;
    else
        p->next=p2;
    return head;
}
```

2）第二种情况，两链表均为从小到大的单一（无重复元素）有序链表，现在合并为一个有序链表，并且如有相同元素，则只保留一个，使得新链表仍为单一有序链表。

```
Node* merge_link_single(Node* head1,Node* head2)
{
    Node * head,* p,* p1=head1,* p2=head2,* q=NULL;
    if(head1 && head2)
    {
        if(head1->data>head2->data)
        {
            head=head2;
            p2=p2->next;
        }
```