2011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

2011年
计算机学科
专业基础考研辅导

张丽芬 主编
张丽芬 钟宏 蒋本珊 薛静锋 编著

符合2011年最新大纲
历届考研真题解析

专家强力推荐
考研辅导书

人民邮电出版社
POSTS & TELECOM PRESS

2011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

2011年
计算机学科
专业基础考研辅导

张丽芬 主编
张丽芬 钟宏 蒋本珊 薛静锋 编著

人民邮电出版社
北京

图书在版编目（CIP）数据

2011年计算机学科专业基础考研辅导 / 张丽芬主编
张丽芬等编著. -- 北京 : 人民邮电出版社, 2010.7
ISBN 978-7-115-23243-4

Ⅰ. ①2… Ⅱ. ①张… 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(2010)第111345号

内 容 提 要

本书以“2011 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考试大纲”为依据，严格按照大纲的范围，针对数据结构、计算机组成原理、操作系统和计算机网络 4 门课程中涉及的知识要点进行集中讲解，选择了历年各个高校研究生入学考试中的典型试题进行分析，介绍相关的知识点、解题思路和算法；通过剖析 2009 年、2010 年实考试卷，帮助参加 2011 年计算机学科硕士研究生入学考试的考生进行复习和总结。

本书可作为计算机科学与技术学科硕士研究生入学考试的辅导用书，也可作为高等院校学生学习相关课程的教学参考书。

2011 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

2011 年计算机学科专业基础考研辅导

◆ 主　　编　张丽芬
编　　著　张丽芬　钟　宏　蒋本珊　薛静锋
责任编辑　邹文波

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
中国铁道出版社印刷厂印刷

◆ 开本：787×1092　1/16
印张：38.25　　2010 年 7 月第 1 版
字数：944 千字　　2010 年 7 月北京第 1 次印刷

ISBN 978-7-115-23243-4

定价：72.00 元

读者服务热线：(010)67170985　印装质量热线：(010)67129223
反盗版热线：(010)67171154

前言

2008年7月，教育部发布了“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考试大纲”（以下简称为大纲），决定对全国硕士研究生入学统一考试计算机科学与技术学科的初试科目进行调整，调整后报考计算机学科硕士研究生初试科目中政治理论、外国语和数学为全国统考，计算机学科专业基础综合考试为全国联合命题考试。自2009年起，计算机科学与技术学科硕士研究生入学专业基础综合考试已经进行了两年，从这两年实施情况看，已经开始对不少高校的计算机专业本科教学产生了积极的影响。

按照国家硕士研究生入学考试改革的思路，硕士研究生的选拔分为初试和复试两个阶段。在初试阶段，要侧重考查考生的基本素质、一般能力和学科基本素质。专业基础综合考试既要符合国家对高层次人才选拔的要求，又要符合我国高校本科教学和学生本科阶段的学习情况，以考查通识、基础、核心教学内容为原则，以考查进入研究生学习必备的专业基础知识、基本理论以及相应能力为重点。计算机学科专业基础综合考试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。

在大纲中规定了专业课150分的考试内容包括数据结构、操作系统、计算机组成原理和计算机网络4门课程。其中，数据结构占45分，计算机组成原理占45分，操作系统占35分，计算机网络占25分。

在大纲中没有指定教材或相关参考书目，按照我们国家公布的同类研究生入学考试大纲也从来不指定教材或相关参考书目。2008年起，一些学生在准备专业课程考试时产生了不少困惑和疑问；在实际的教学过程中，也有不少学生向任课教师提过关于研究生入学考试的问题，作为相应课程的主讲教师，我们也非常想给自己的学生以切实而有效的帮助。在不断回答学生提问的过程中，我们开始关注研究生入学考试，于2008年9月完成了《2009年计算机学科专业基础考研辅导》一书，并于2009年8月修订完成了《2010年计算机学科专业基础考研辅导》一书。

2009年元月全国进行了首次计算机学科联合命题考试，2010年元月全国进行了第二次计算机学科联合命题考试，为便于学生针对2011年进行计算机学科专业课程进行考研复习，我们根据公布的大纲和2009年、2010年已经进行的实考题目，结合多年的教学实践，分别听取了参加2009年和2010年入学考试的考生对2009版和2010版辅导书提出的修改意见，编写了2011版的辅导书。

比较已经公开的2009年、2010年和2011年大纲可以看出，每年的大纲均有微小的变化，但考试的核心内容没有明显的变化，2011版大纲对各门课程的考查目标进行了更准确的说明，对考核的具体知识点进行了细微的调整，对于考生的复习思路和重点没有太大影响。

2011版大纲中明确指出了研究生入学统一考试的考查目标，要求考生比较系统地掌握专业基础课程的基本概念、基本原理和基本方法，能够运用所学的基本原理和基本方法分析、判断和解决有关的理论问题和实际问题。根据这一明确的考查目标，根据使用过2009年版、2010年版辅导书考生的意见，在2011年的版本中，我们保持了原来版本的基本结构，在各章的后面专门增加了“精选试题解析”一节。

通过分析2011版大纲和2009年、2010年的实际考题，全书在例题选择时放弃了一些过难、过偏的题目，也放弃了一些偏重概念记忆的题目，更多地选择了一些反映基本知识和基本概念，但注重考察理解和综合应用能力的题目；同时我们也没有试图编写一本“厚”的习题集，因为从教多年的经验告诉我们，“题海”战术在考研复习过程中只能是事倍功半。

2011版的辅导书仍然按课程分为4篇，每篇中的章节设置按照大纲的顺序进行组织，每章中首先对相关课程考纲中的知识点进行归纳，并针对要点进行集中讲解，然后结合历年各个高校研究生入学考试中的典型试题进行分析，介绍相关的知识点和题目要点，以帮助考生进行有针对性的复习，并通过精选试题练习开阔考生思路。由于国内各个高校采用的专业教材各不相同，所以不少技术名词也不相同，在编写本书时，作者也面临技术名词问题，最后决定，对于大纲中出现的技术名词均以大纲为准，在大纲中未出现过的名词，采用国内多数教材使用的名词，并同时注明原文。

根据广大读者的要求，在2011版中我们集中给出了全部选择题目的答案和精选试题练习题目的答案。在附录中，我们对2009年、2010年的研究生入学考试实际试卷给出了参考答案，对全部试题进行了逐个分析和讲解，试题的主观题部分没有采用考试中心公布的参考答案，而是选用了我们自己编写的答案，目的是为了开阔考生的思路。

参加本书编写工作的老师均为我国“985”高校中长期从事计算机科学与技术学科相应本科生课程教学的一线教授或副教授，在相关课程中均具有10年以上的教学经历，并先后编写过多本相关教材和教学参考书，对相关课程有着更深一层的认识。本书是这些教师多年教学经验和长期跟踪并研究硕士研究生专业入学考试的经验总结。

本书由张丽芬主编，第1篇数据结构部分由钟宏编写，赵小林审校；第2篇计算机组成原理部分由蒋本珊编写，马忠梅审校；第3篇操作系统部分由张丽芬编写，刘美华审校；第4篇计算机网络部分由薛静锋编写，李志强审校；后面研究生试题的参考答案由诸位老师合作完成。由于编者水平有限，书中难免有错误或不当之处，恳请广大读者批评指正。

编　者

2010年6月于北京

目　录

第1篇　数据结构

第2篇 计算机组成原理

第3篇 操作系统

第 4 篇 计算机网络

第 1 篇
数据结构

“数据结构”是计算机学科专业基础的重要组成部分之一，所涉及的基础知识、基本理论、基本方法是从事计算机学科研究和研究生学习阶段必须掌握的基本内容。在计算机科学与技术学科硕士研究生入学专业联考的 150 分中占 45 分。

对“数据结构”部分的考核基本要求包括：理解数据结构的基本概念；掌握数据的逻辑结构、存储结构及其差异，以及各种基本操作的实现。在掌握基本的数据处理原理和方法的基础上，能够对算法进行基本的时间复杂度与空间复杂度分析。能够选择适当的数据结构和方法进行问题求解；具备采用 C 或 C++或 Java 语言设计并实现算法的能力。

第1章 线性表

1.1 知识要点扫描

1.1.1 线性表的定义和基本操作

1. 线性表的定义

线性表简单地说是具有相同特征的数据元素的有限序列。

线性表的抽象数据类型定义（ADT）是线性表逻辑结构的描述，它包括数据对象、关系的定义，与线性表相关操作的定义。

线性表的数据对象是具有相同性质的数据元素集合。

线性表的数据关系是数据元素按位置有序，即表中的数据元素有一个前后顺序关系。理解线性表定义的关键是数据元素在表中的位置关系。线性表中各个元素之间强调的是一种位序而不是数值大小的顺序，这既不同于无前后顺序关系的数据元素集合，也不同于按数据元素值大小排列的有序表。

2. 线性表的基本操作

每一种数据结构都有一组与之相关的基本操作。基本操作反映了该数据结构的“行为特征”。不同的应用，不同的存储方式，基本操作的种类和功能可能是不同的。尽管如此，每一种数据结构大都包含：构造、销毁（析构）操作、访问操作和维护操作。

与线性表相关的操作有：初始化线性表，撤销线性表，判/置表空，取表长，取前驱，取后继，取第 i 个元素/置第 i 个元素为 x，插入元素，删除元素，以某函数遍历线性表等。

掌握基本操作的关键是理解基本操作的功能以及它们对于数据抽象的作用和意义。另外需要注意每个操作调用的前提条件。

1.1.2 线性表的实现

1. 顺序存储结构

线性表存储结构表示也称为线性表存储结构的实现。同一种数据结构，可以有不同的存

储结构，如线性表可用顺序存储结构或链式存储结构存储。同样，由于高级程序设计语言提供了越来越丰富的数据类型，同一种存储结构可以有不同的实现方法。线性表的顺序存储通常有两种表示方式：静态数组方式和动态数组方式。在严蔚敏等编著的《数据结构》（C 语言版，清华大学出版社）书中，线性表的顺序存储结构采用的是动态数组方式。

掌握线性表的顺序存储结构的关键是数据类型的定义以及基本操作的实现方法。

学习线性表的顺序存储结构时，要注意顺序存储结构的随机存取特性。

在顺序存储下结点插入和删除的时空效率要低于在链式存储下结点插入和删除的时空效率，因为顺序存储下的插入和删除往往涉及大量的数据移动。

2．链式存储结构

线性表的链式存储结构有线性链表、循环链表、双向链表等。线性表链式存储结构的表示（链式存储的实现）有动态链表和静态链表两种方式。动态线性链表、动态循环链表、动态双向链表通常分别简称为线性链表、循环链表、双向链表。

掌握线性表的链式存储结构的关键是数据类型的定义以及基本操作的实现方法。

学习线性表的链式存储结构时，要注意指针的操作顺序以及存储资源的有效利用。

在线性表一章中，链表操作是复习的重点，建议先把程序设计语言中的指针操作等内容复习一下，因为链表的许多操作是在操作指针，只有对指针的运算相当清楚才可能准确把握链表的各种算法。对于指针部分，最基本的是要搞清何时表示地址，何时表示地址内的数据。

3．线性表的应用

线性表是计算机应用中常见的数据结构，掌握线性表应用的关键是如何用线性表表示实际问题中的数据，以及如何利用线性表的基本操作实现更复杂的数据处理。

在线性表的应用上，需特别注意针对不同的应用场合，采用不同的存储结构、不同形式的链表。应熟悉各种存储结构及各种链表的特点，针对不同的运算，它们在时间和存储上的效率不同。

1.2 典型例题分析

【例 1.1】在下列关于线性表的叙述中，正确的是（　　）。

A．每个元素有且仅有一个直接前趋，有且仅有一个直接后继

B．表中至少有一个元素

C．除第一个元素和最后一个元素之外，其他元素都有且仅有一个直接前趋，有且仅有一个直接后继

D．表中元素必须从大到小或从小到大排列

解答：C。

试题分析：此题考查学生对线性表定义的理解程度。线性表一般被定义为由若干个元素组成的有序序列，注意，线性表是位置有序而不是数据有序。

【例 1.2】在链式存储结构中，体现数据之间关系的是（　　）。

A．数据在内存的相对位置　　B．指示数据元素的指针

C．数据的存储地址　　D．指针

解答：B。

试题分析：此题属于大纲中链式存储结构的概念题。链式存储结构是线性表数据结构在计算机中的体现，而线性表中的数据关系主要是数据元素的前后次序关系，这个关系是通过指向数据元素的指针实现的。注意，在做选择题时应该尽量选择更准确的答案，B 比 D 更准确。

【例 1.3】静态链表与动态链表相比，其缺点是（　　）。

A．插入、删除时需移动较多数据　　B．有可能浪费较多存储空间

C．不能随机存取　　D．以上都不是

解答：B。

试题分析：此题属于大纲中链式存储结构的概念题。静态链表与动态链表都属于链式存储结构，不同之处是静态链表中的结点来源于一个静态数组。浪费较多存储空间是使用数组存储数据常见的问题，因为数组中的数据不可能总是满的。静态链表与动态链表都不能随机存取，所以不能选 C。

【例 1.4】在一个单链表中，已知指针 p 指向其中的某个结点，若在该结点前插入一个由指针 s 指向的结点，则需执行（　　）。

A．s->next = p->next; p->next = s;　　B．p->next = s; s->next = p;

C．r = p->next; p->next = s; s->next = r;　　D．仅靠已知条件无法实现

解答：D。

试题分析：此题考查学生对链式存储结构中基本操作实现的掌握程度。由于单链表的单向性，在某个结点前插入结点时，必须有一个指针指向该结点的前驱结点。所以仅靠此题的已知条件无法实现所要求的插入。

【例 1.5】假设单链表的类型定义如下：

```
typedef struct   LNode {
        ElemType        data;     // 数据域
        struct LNode    *next;    // 指针域
} LNode, *LinkList;
```

若 L 表示单链表的头指针，正确的表示是（　　）。

A．ElemType L;　　B．ElemType *L;　　C．LinkList *L;　　D．LinkList L;

解答：D。

试题分析：此题考查学生对链式存储结构实现的掌握程度。根据单链表的类型定义，LinkList 是指向结点类型的指针，通常在程序中用此类型来定义表示链表的头指针，所以此题应选 D。

【例 1.6】在线性表的顺序存储结构中，记录元素之间关系的是（　　）。

A．保存后继元素地址　　B．元素的存储顺序

C．保存左、右孩子地址　　D．保存后继元素的数组下标

解答：B。

试题分析：此题属于大纲中顺序存储结构的概念题。顺序存储结构是线性表数据结构在计算机中的体现，而线性表中的数据关系主要是数据元素的前后次序关系，在顺序存储结构

中，这个次序关系与数据元素的存储顺序是一致的，所以应选 B。

【例 1.7】在线性表的顺序存储结构中，为求出任一元素的存储地址，只要知道（　　）。

A．基地址　　B．一个元素占用的字节

C．向量大小　　D．基地址和一个元素占用的字节

解答：D。

试题分析：此题属于大纲中顺序存储结构的概念题。根据一维数组的随机存储特性，计算一个数据元素存储位置需要知道数据元素的序号、数组的基地址和一个元素占用的字节数，此题假设元素的序号是已知的，所以应选 D。

【例 1.8】在顺序存储结构下，在线性表第 i 个元素之前插入新元素，需要（　　）。

A．移动元素　　B．修改头指针　　C．修改指针　　D．申请新的结点空间

解答：A。

试题分析：此题考查学生对顺序存储结构中基本操作实现的掌握程度。需要注意的是移动元素是必须做的，而申请新的结点空间不是必需的。

【例 1.9】静态链表中指针表示（　　）。

A．内存地址　　B．元素的数组下标　C．元素的地址　　D．左、右孩子地址

解答：B。

试题分析：静态链表属于链式存储结构的一种。在静态链表中，结点是从某个特定数组中获得，其特点是用数组的下标代替链表中的地址指针，因此答案应选 B。

【例 1.10】在非空单向循环链表的某个元素之前插入新元素，需要（　　）。

A．向前移动元素　　B．向后移动元素

C．修改前驱元素结点指针域　　D．修改该元素结点的指针域

解答：C。

试题分析：在非空单向循环链表中，每个元素都有前驱结点，在某个元素之前插入新元素时，需要修改该元素前驱结点的指针域。

【例 1.11】判断带头结点的线性链表 L 是否为空的条件是（　　）。

A．L.elem=NULL　　B．L.length = 0　　C．L->next=NULL　D．L = NULL

解答：C。

试题分析：需要注意的是带头结点链表与不带头结点链表在判断表空的条件是不同的，头结点是链表中永远存在的首结点，无论线性表是空还是非空，因此答案应选 C。

【例 1.12】对于线性链表，在 p 所指向的结点后插入由 q 指向的新结点的语句序列是（　　）。

A．p->next = q; q->next = p->next;　　B．q = p->next; p->next = q;

C．q->next = p->next; p->next = q;　　D．p = p->next; q->next = p;

解答：C。

试题分析：在单链表中进行插入运算时需特别注意指针修改顺序。选项 A 的语句虽然和选项 C 的语句一样，但顺序是错误的。

【例 1.13】若某线性表最常用的操作是在最后一个结点之后插入一个结点或删除最后一个结点，则最节省运算时间的存储方式是（　　）。

A．单链表　　B．给出表头指针的单循环链表

C．双向链表　　D．带头结点的双循环链表

解答：D。

试题分析：此题应属于大纲中线性表应用题，考查学生是否可以根据具体的应用场合，选择合适的数据结构实现方法。在链表中，插入或删除一个结点，需修改它的前趋或后继结点的指针域。如不特别指明，链表通常只给出第一个结点的地址——头指针。其他结点的地址只能从它的前趋或后继结点得到。4种选择中只有D能从第一个结点，经过最少的结点，获得最后一个结点“相邻”结点的地址。

【例1.14】若长度为n的线性表采用顺序存储结构，在其第i个位置（$1\leqslant i\leqslant n+1$）插入一个新元素算法的时间复杂度为（　　）。

A．$O(0)$　　B．$O(1)$　　C．$O(n)$　　D．$O(n^2)$

解答：C。

试题分析：此题考查学生是否了解顺序存储结构下，在其第i个位置（$1\leqslant i\leqslant n+1$）插入一个新元素算法的时间复杂度。在顺序表第$i$个位置（$1\leqslant i\leqslant n+1$）插入一个新元素，平均移动元素的个数为$n/2$，因此答案应是C。

【例1.15】在线性链表中，头指针指向的位置可能是（　　）。

A．第一个元素结点　B．头结点　　C．A和B都不是　D．A或B

解答：D。

试题分析：此题属于大纲中链式存储结构的概念题。在不带头结点的链表中，头指针指向第一个元素结点；在带头结点的链表中，头指针指向头结点。此题没有说明线性链表是否带头结点，因此答案应选D。

【例1.16】静态链表的类型定义如下：

```
typedef struct {
    ElemType    data;
    int         cur;
} component, SLinkList[ MAXSIZE ];
```

假设S是表示静态链表的数组（SlinkList S），若某个链表的第k个结点的下标是i，则第$k+1$个结点的数据是（　　）。

A．S[i + 1].data　　B．S[k + 1].cur.data　C．S[S[i].cur].data　D．S[S[k].cur].data

解答：C。

试题分析：在静态链表中，第k个结点是S[i]，其下一个结点（即第$k+1$个结点）的下标是S[i].cur，因此第$k+1$个结点是S[S[i].cur]，其包含的数据是S[S[i].cur].data。

【例1.17】若需要对线性表中保存的无序数据按关键字进行查找，查找效率最高的存储结构是（　　）。

A．顺序存储结构　　B．链式存储结构　C．静态链表　　D．都没有区别

解答：D。

试题分析：此题应属于大纲中线性表应用题，设计的应用场景为“按关键字进行查找”，考查学生是否能够根据具体的应用场合选择最合适的数据结构实现算法。因为此题中线性表中的元素是无序的，无论采用何种存储结构，都需要从头一个一个地进行比较，查找的时间复杂度都是$O(n)$，因此答案应选D。

【例 1.18】设有一非零整数数组 A[n]，试编写算法将 A 中的小于 0 的整数放在 A 的前边，将大于零的整数放在 A 的后边，要求不使用其他的辅助数据结构。

解答（参考算法）：

```
void Rearrange (int A[ ], int n )
{// 从数组 A 的两端交替地向中间扫描，i、j 分别为扫描指示器，初始时，i = 0,j = n-1。
    i = 0;  j = n-1;
    temp = A[0];
    while ( i < j ) {
       while ( ( i < j )&& (A[j]>0 ) )
          --j;
       A[i] = A[j];
       while ( ( i < j )&& (A[i]<0 ) )   ++ i;
       A[j] = A[i];
      }
      A[ i ] = temp;
}
```

试题分析：此题应属于大纲中线性表的应用编程题，考查学生是否可以根据顺序存储结构的特点解决实际问题。学生解决此题时，有可能考虑对数据进行排序，但这样做的效率比较低，而完全有序的数据不是此题所要求的。此题实际要求的是数组的数据划分成两部分，前半部分小于 0，后半部分大于 0。而快速排序中的一次划分恰好可以完成此任务。此时的枢轴是 0，而不是数组中的第一个元素（但首先需要把第一个元素 A[0]放到 temp 变量中，这样才能进行后面的交换），这样的分割使负数都在数组的前边，正数都在数组的后边（最后不要忘了把 temp 的变量放回数组，此时分割点是空的，可以放任何元素）。

【例 1.19】设计算法将一个带头结点的单链表 Ha 分解为两个具有相同结构的链表 Hb、Hc，其中，Hb 表的结点为 Ha 表中值小于零的结点，而 Hc 表的结点为 Ha 表中值大于等于零的结点（链表 A 的元素类型为整型，要求 Hb、Hc 表利用 Ha 表的结点）。

解答（参考算法）：

```
void decompose( LinkList &Ha, LinkList &Hb, LinkList &Hc )
{  // Ha、Hb、Hc 分别为 3 个链表的头指针。
   Hb = (linklist)malloc(sizeof(Lnode));  Hb->next = null;  // 建 Hb 空表
   Hc = (linklist)malloc(sizeof(Lnode));  Hc->next = null;  // 建 Hc 空表
   p = Ha->next;
   while ( p )  {
      q = p->next;
      if  ( p->data < 0 ) {
         p->next = Hb->next;        // 将当前结点插到表 Hb 的表头
         Hb->next = p;
      }
      else {
         p->next = Hc->next;     //将当前结点插到表 Hc 的表头
         Hc->next = p;
      }
```

```
        p = q;
    }
}
```

试题分析：此题应属于大纲中线性表的应用编程题，考查学生是否可以解决有关链表运算的实际问题。算法的核心是对链表 Ha 进行一次扫描，对于链表 Ha 中的每个结点，根据其大小，插入到链表 Hb 或 Hc 中，插入的位置总是在表头结点后面，这样效率最高。

【例 1.20】试编写在带头结点的单链表中删除一个最小值结点的高效算法。

解答（参考算法）：

```
void delete( linklist  &L )
{  if ( !L->next )  return;
   premin = L;
   precur = L->next;
   while ( precur->next != NULL )  {
      if ( premin->next->data > precur->next->data )
         premin = precur;
      precur = precur->next;
   }
   pmin = premin->next;
   premin->next = pmin->next;
   free( pmin );
}
```

试题分析：算法的核心思想是从头至尾扫描单链表，找出值最小的结点。算法的关键是设置合适的指针。在算法中，premin 指向当前已扫描结点中值最小结点的前驱结点；precur 指向当前结点的前驱结点。开始时，premin 指向链表的头结点，表示第一个元素结点是最初的值最小的结点；precur 指向第一个元素结点，表示第二个元素结点是最初的当前结点。若当前结点值比当前最小值小，则 premin 等于 precur。继续扫描直至链表尾。最后 premin 指向结点值最小结点的前驱结点，从单链表中删除值最小结点。

需要注意的是，在算法的开始要考虑表空这一特殊情况。

【例 1.21】Ha 和 Hb 为两个带头结点链表的头指针，分别表示两个集合 A 和 B，两个链表的元素值递增有序排列。现要求另辟空间构成一个新的带头结点链表，其头指针为 Hc，链表的元素为 A 和 B 的交集，且也是递增有序。试编写建立新链表的算法。

解答（参考算法）：

```
void join( LinkList Ha, LinkList Hb, LinkList &Hc )
{
    Hc = ( Linklist ) malloc( sizeof( Lnode ) );  // 为 Hc 链表建立头结点
    pc = Hc;
    pa = Ha->next;
    pb = Hb->next;
    while ( ( pa! = NULL ) && ( pb! = NULL ) ) {    // 扫描单链表 Ha 和 Hb
       if ( pa->data < pb->data )
          pa = pa->next;
```