全国计算机等级考试——二级公共基础知识辅导讲义

第一章　数据结构与算法

1.1 算法

1、算法是指解题方案的准确而完整的描述。换句话说，算法是对特定问题求解步骤的一种描述。

*：算法不等于程序，也不等于计算方法。程序的编制不可能优于算法的设计。

2、算法的基本特征

（1）可行性。针对实际问题而设计的算法，执行后能够得到满意的结果。

（2）确定性。每一条指令的含义明确，无二义性。并且在任何条件下，算法只有唯一的一条执行路径，即相同的输入只能得出相同的输出。

（3）有穷性。算法必须在有限的时间内完成。有两重含义，一是算法中的操作步骤为有限个，二是每个步骤都能在有限时间内完成。

（4）拥有足够的情报。算法中各种运算总是要施加到各个运算对象上，而这些运算对象又可能具有某种初始状态，这就是算法执行的起点或依据。因此，一个算法执行的结果总是与输入的初始数据有关，不同的输入将会有不同的结果输出。当输入不够或输入错误时，算法将无法执行或执行有错。一般说来，当算法拥有足够的情报时，此算法才是有效的；而当提供的情报不够时，算法可能无效。

*：综上所述，所谓算法，是一组严谨地定义运算顺序的规则，并且每一个规则都是有效的，且是明确的，此顺序将在有限的次数下终止。

3、算法复杂度主要包括时间复杂度和空间复杂度。

（1）算法时间复杂度是指执行算法所需要的计算工作量，可以用执行算法的过程中所需基本运算的执行次数来度量。

（2）算法空间复杂度是指执行这个算法所需要的内存空间。

1.2 数据结构的基本概念

1、数据结构是指相互有关联的数据元素的集合。

2、数据结构主要研究和讨论以下三个方面的问题：

（1）数据集合中各数据元素之间所固有的逻辑关系，即数据的逻辑结构。

数据的逻辑结构包含：1）表示数据元素的信息；2）表示各数据元素之间的前后件关系。

（2）在对数据进行处理时，各数据元素在计算机中的存储关系，即数据的存储结构。

数据的存储结构有顺序、链接、索引等。

1）顺序存储。它是把逻辑上相邻的结点存储在物理位置相邻的存储单元里，结点间的逻辑关系由存储单元的邻接关系来体现。由此得到的存储表示称为顺序存储结构。

2）链接存储。它不要求逻辑上相邻的结点在物理位置上亦相邻，结点间的逻辑关系是由附加的指针字段表示的。由此得到的存储表示称为链式存储结构。

3）索引存储：除建立存储结点信息外，还建立附加的索引表来标识结点的地址。

*：数据的逻辑结构反映数据元素之间的逻辑关系，数据的存储结构（也称数据的物理结构）

批注 [wx1]: 这是因为：在编写程序时要受到计算机系统运行环境的限制，程序通常还要考虑很多与方法和分析无关的细节问题。

批注 [wx2]: 前后件关系：一般情况下，在具有相同特征的数据元素集合中，各个数据元素之间存在某种关系（即联系），这种关系反映了该集合中的数据元素所固有的一种结构。在数据处理领域中，通常把数据元素之间这种固有的关系简单地用前后件关系（即直接前驱与直接后继关系）来描述。

是数据的逻辑结构在计算机存储空间中的存放形式。同一种逻辑结构的数据可以采用不同的存储结构，但影响数据处理效率。

（3）对各种数据结构进行的运算。

3、数据结构的图形表示

一个数据结构除了用二元关系表示外，还可以直观地用图形表示。在数据结构的图形表示中，对于数据集合 D 中的每一个数据元素用中间标有元素值的方框表示，一般称之为数据结点，并简称为结点；为了进一步表示各数据元素之间的前后件关系，对于关系 R 中的每一个二元组，用一条有向线段从前件结点指向后件结点。

4、**数据结构分为两大类型：线性结构和非线性结构**。

（1）线性结构（非空的数据结构）条件：**1）有且只有一个根结点；2）每一个结点最多有一个前件，也最多有一个后件。**

批注 [wx3]: 在数据结构中，没有前件的结点称为根结点。

*：常见的线性结构**有线性表、栈、队列和线性链表**等。

（2）非线性结构：不满足线性结构条件的数据结构。

*：常见的**非线性结构有树、二叉树和图**等。

1.3 线性表及其顺序存储结构

1、线性表由一组数据元素构成，数据元素的位置只取决于自己的序号，元素之间的相对位置是线性的。线性表是由 n(n≥0)个数据元素组成的一个有限序列，表中的每一个数据元素，除了第一个外，有且只有一个前件，除了最后一个外，有且只有一个后件。线性表中数据元素的个数称为线性表的长度。线性表可以为空表。

*：线性表是一种存储结构，它的存储方式：**顺序和链式**。

2、线性表的顺序存储结构具有两个基本特点：（1）线性表中所有元素所占的存储空间是连续的；（2）线性表中各数据元素在存储空间中是按逻辑顺序依次存放的。

*：由此可以看出，在线性表的顺序存储结构中，其前后件两个元素在存储空间中是紧邻的，且前件元素一定存储在后件元素的前面，可以通过计算机直接确定第 i 个结点的存储地址。

3、顺序表的插入、删除运算

（1）顺序表的插入运算：在一般情况下，要在第 i（1≤i≤n）个元素之前插入一个新元素时，首先要从最后一个（即第 n 个）元素开始，直到第 i 个元素之间共 n-i+1 个元素依次向后移动一个位置，移动结束后，第 i 个位置就被空出，然后将新元素插入到第 i 项。插入结束后，线性表的长度就增加了 1。

*：顺性表的插入运算时需要移动元素，在等概率情况下，平均需要移动 n/2 个元素。

（2）顺序表的删除运算：在一般情况下，要删除第 i（1≤i≤n）个元素时，则要从第 i+1 个元素开始，直到第 n 个元素之间共 n-i 个元素依次向前移动一个位置。删除结束后，线性表的长度就减小了 1。

*：进行顺性表的删除运算时也需要移动元素，在等概率情况下，平均需要移动（n-1）/2 个元素。插入、删除运算不方便。

1.4 栈和队列

1、栈及其基本运算

栈是限定在一端进行插入与删除运算的线性表。

在栈中，允许插入与删除的一端称为栈顶，不允许插入与删除的另一端称为栈底。栈顶元素总是最后被插入的元素，栈底元素总是最先被插入的元素。即栈是按照**“先进后出”或“后进先出”**的原则组织数据的。

批注 [A4]: 第 8 题

栈具有记忆作用。

栈的基本运算：1）插入元素称为入栈运算；2）删除元素称为退栈运算；3）读栈顶元素是将栈顶元素赋给一个指定的变量，此时指针无变化。

栈的存储方式和线性表类似，也有两种，即顺序栈和链式栈。

2、队列及其基本运算

队列是指允许在**一端（队尾）进入插入，而在另一端（队头）进行删除**的线性表。尾指针（Rear）指向队尾元素，头指针（front）指向排头元素的前一个位置（队头）。

队列是“先进先出”或“后进后出”的线性表。

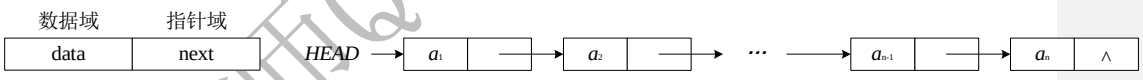
队列运算包括：1）入队运算：从队尾插入一个元素；2）退队运算：从队头删除一个元素。循环队列及其运算：所谓循环队列，就是将队列存储空间的最后一个位置绕到第一个位置，形成逻辑上的环状空间，供队列循环使用。在循环队列中，用队尾指针 rear 指向队列中的队尾元素，用排头指针 front 指向排头元素的前一个位置，因此，从头指针 front 指向的后一个位置直到队尾指针 rear 指向的位置之间，所有的元素均为队列中的元素。

批注 [A5]: 第 24 套

1.5 线性链表

1、线性表顺序存储的缺点：（1）插入或删除的运算效率很低。在顺序存储的线性表中，插入或删除数据元素时需要移动大量的数据元素；（2）线性表的顺序存储结构下，线性表的存储空间不便于扩充；（3）线性表的顺序存储结构不便于对存储空间的动态分配。

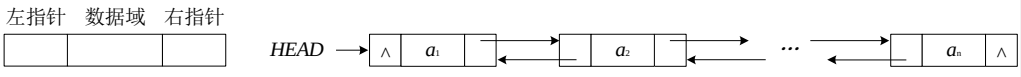
2、线性链表：线性表的链式存储结构称为线性链表，是一种物理存储单元上非连续、非顺序的存储结构，数据元素的逻辑顺序是通过链表中的指针链接来实现的。因此，在链式存储方式中，每个结点由两部分组成：一部分用于存放数据元素的值，称为数据域；另一部分用于存放指针，称为指针域，用于指向该结点的前一个或后一个结点（即前件或后件），如下图所示：



(a) 结点结构　　(b) 一个非空的线性链表示意图

线性链表分为单链表、双向链表和循环链表三种类型。

在单链表中，每一个结点只有一个指针域，由这个指针只能找到其后件结点，而不能找到其前件结点。因此，在某些应用中，对于线性链表中的每个结点设置两个指针，一个称为左指针，指向其前件结点；另一个称为右指针，指向其后件结点，这种链表称为双向链表，如下图所示：



(a) 结点结构　　(b) 一个非空的双向链表示意图

3、线性链表的基本运算

（1）在线性链表中包含指定元素的结点之前插入一个新元素。

*：在线性链表中插入元素时，不需要移动数据元素，只需要修改相关结点指针即可，也不会出现“上溢”现象。

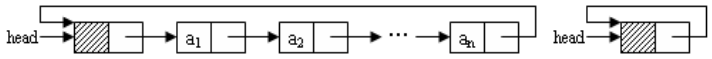
批注 [wx6]: 当为一个线性表分配顺序存储结构后，如果出现线性表的存储空间已满，但还需要插入新的元素时，就会发生“上溢”现象。

（2）在线性链表中删除包含指定元素的结点。

*：在线性链表中删除元素时，也不需要移动数据元素，只需要修改相关结点指针即可。
（3）将两个线性链表按要求合并成一个线性链表。
（4）将一个线性链表按要求进行分解。
（5）逆转线性链表。
（6）复制线性链表。
（7）线性链表的排序。
（8）线性链表的查找。
*：线性链表不能随机存取。

批注 [wx7]: 在链表中，即使知道被访问结点的序号 i，也不能像顺序表中那样直接按序号 i 访问结点，而只能从链表的头指针出发，顺着链域逐个结点往下搜索，直至搜索到第 i 个结点为止。因此，链表不是随机存储结构。

4、循环链表及其基本运算
在线性链表中，其插入与删除的运算虽然比较方便，但还存在一个问题，在运算过程中对于空表和对第一个结点的处理必须单独考虑，使空表与非空表的运算不统一。为了克服线性链表的这个缺点，可以采用另一种链接方式，即循环链表。
与前面所讨论的线性链表相比，循环链表具有以下两个特点：1）在链表中增加了一个表头结点，其数据域为任意或者根据需要来设置，指针域指向线性表的第一个元素的结点，而循环链表的头指针指向表头结点；2）循环链表中最后一个结点的指针域不是空，而是指向表头结点。即在循环链表中，所有结点的指针构成了一个环状链。
下图 a 是一个非空的循环链表，图 b 是一个空的循环链表：



(a)非空表　　(b)空表

循环链表的优点主要体现在两个方面：一是在循环链表中，只要指出表中任何一个结点的位置，就可以从它出发访问到表中其他所有的结点，而线性单链表做不到这一点；二是由于在循环链表中设置了一个表头结点，在任何情况下，循环链表中至少有一个结点存在，从而使空表与非空表的运算统一。
*：循环链表是在单链表的基础上增加了一个表头结点，其插入和删除运算与单链表相同。但它可以从任一结点出发来访问表中其他所有结点，并实现空表与非空表的运算的统一。

1.6 树与二叉树
1、树的基本概念
树是一种简单的非线性结构。在树这种数据结构中，所有数据元素之间的关系具有明显的层次特性。
在树结构中，每一个结点只有一个前件，称为父结点。没有前件的结点只有一个，称为树的根结点，简称树的根。每一个结点可以有多个后件，称为该结点的子结点。没有后件的结点称为叶子结点。
在树结构中，一个结点所拥有的后件的个数称为**该结点**的度，所有结点中最大的度称为树的度。树的最大层次称为树的深度。
2、二叉树及其基本性质
（1）什么是二叉树
二叉树是一种很有用的非线性结构，它具有以下两个特点：1）非空二叉树只有一个根结点；2）每一个结点最多有两棵子树，且分别称为该结点的左子树与右子树。
*：根据二叉树的概念可知，二叉树结点的度可以为 0（叶结点）、1（只有一棵子树）或 2（有 2 棵子树）。
（2）二叉树的基本性质

性质 1　在二叉树的第 k 层上，最多有 $2^{k-1}(k\geq 1)$个结点。

性质 2　深度为 m 的二叉树最多有个 2^m-1个结点。

性质 3　在任意一棵二叉树中，度数为 0 的结点（即叶子结点）总比度为 2 的结点多一个。

批注 [A8]: 第 4 题，第 23 题

性质 4　具有 n 个结点的二叉树，其深度至少为 $[\log_2 n]+1$，其中 $[\log_2 n]$ 表示对 $\log_2 n$ 取的整数部分。

3、满二叉树与完全二叉树

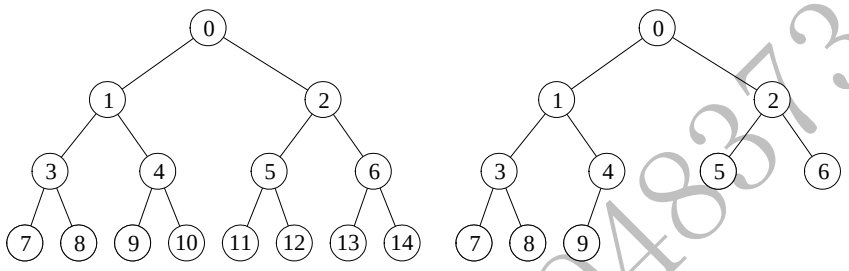
满二叉树：除最后一层外，每一层上的所有结点都有两个子结点。

完全二叉树：除最后一层外，每一层上的结点数均达到最大值；在最后一层上只缺少右边的若干结点。

*：**根据完全二叉树的定义可得出：度为 1 的结点的个数为 0 或 1。**

批注 [A9]: 第 19 题

下图 a 表示的是满二叉树，下图 b 表示的是完全二叉树：



(a) 满二叉树　　(b) 完全二叉树

完全二叉树还具有如下两个特性：

性质 5　具有 n 个结点的完全二叉树**深度为** $[\log_2 n]+1$。

性质 6　设完全二叉树共有 n 个结点，如果从根结点开始，按层序（每一层从左到右）用自然数 1，2，…，n 给结点进行编号，则对于编号为 k（k=1，2，…，n）的结点有以下结论：

①若 k=1，则该结点为根结点，它没有父结点；若 k>1，则该结点的父结点的编号为 INT(k/2)。

②若 2k≤n，则编号为 k 的左子结点编号为 2k；否则该结点无左子结点（显然也没有右子结点）。

③若 2k+1≤n，则编号为 k 的右子结点编号为 2k+1；否则该结点无右子结点。

4、二叉树的存储结构

在计算机中，二叉树通常采用链式存储结构。

与线性链表类似，用于存储二叉树中各元素的存储结点也由两部分组成：数据域和指针域。但在二叉树中，由于每一个元素可以有两个后件（即两个子结点），因此，用于存储二叉树的存储结点的指针域有两个：一个用于指向该结点的左子结点的存储地址，称为左指针域；另一个用于指向该结点的右子结点的存储地址，称为右指针域。

*：一般二叉树通常采用链式存储结构，对于满二叉树与完全二叉树来说，可以按层次进行顺序存储。

批注 [wx10]: 这样，不仅节省了存储空间，又能方便地确定每一个结点的父结点与左右子结点的位置，但顺序存储结构对于一般的二叉树不适用。

5、二叉树的遍历（所谓的前中后指的是，父结点访问的顺序）

二叉树的遍历是指不重复地访问二叉树中的所有结点。二叉树的遍历可以分为以下三种：

（1）**前序遍历**（DLR）：若二叉树为空，则结束返回。否则：首先访问根结点，然后遍历左子树，最后遍历右子树；并且，在遍历左右子树时，仍然先访问根结点，然后遍历左子树，最后遍历右子树。**（从上往下）**

（2）**中序遍历**（LDR）：若二叉树为空，则结束返回。否则：首先遍历左子树，然后访问根结点，最后遍历右子树；并且，在遍历左、右子树时，仍然先遍历左子树，然后访问根结点，最后遍历右子树。

（3）**后序遍历**（LRD）：若二叉树为空，则结束返回。否则：首先遍历左子树，然后遍历右子树，最后访问根结点，并且，在遍历左、右子树时，仍然先遍历左子树，然后遍历右子树，最后访问根结点。

1.7 查找技术

查找：根据给定的某个值，在查找表中确定一个其关键字等于给定值的数据元素。

查找结果：（查找成功：找到；查找不成功：没找到。）

平均查找长度：查找过程中关键字和给定值比较的平均次数。

1、顺序查找

基本思想：从表中的第一个元素开始，将给定的值与表中逐个元素的关键字进行比较，直到两者相符，查到所要找的元素为止。否则就是表中没有要找的元素，查找不成功。

在平均情况下，利用顺序查找法在线性表中查找一个元素，大约要与线性表中一半的元素进行比较，**最坏情况下需要比较 n 次**。

顺序查找一个具有 n 个元素的线性表，其平均复杂度为 O（n）。

下列两种情况下只能采用顺序查找：

1）如果线性表是无序表（即表中的元素是无序的），则不管是顺序存储结构还是链式存储结构，都只能用顺序查找。

2）即使是有序线性表，如果采用链式存储结构，也只能用顺序查找。

2、二分法查找

思想：先确定待查找记录所在的范围，然后逐步缩小范围，直到找到或确认找不到该记录为止。

前提：必须在具有顺序存储结构的有序表中进行。

查找过程：

1）若中间项（中间项 mid=(n-1)/2，mid 的值四舍五入取整）的值等于 x，则说明已查到；

2）若 x 小于中间项的值，则在线性表的前半部分查找；

3）若 x 大于中间项的值，则在线性表的后半部分查找。

特点：比顺序查找方法效率高。**最坏的情况下，需要比较 $\log_2 n$ 次。**

*：二分法查找只适用于顺序存储的线性表，且表中元素必须按关键字有序（升序）排列。

批注 [wx11]: 允许相邻元素值相等。

对于无序线性表和线性表的链式存储结构只能用顺序查找。在长度为 n 的有序线性表中进行二分法查找，其时间复杂度为 O（$\log_2 n$）。

1.8 排序技术

排序是指将一个无序序列整理成按值非递减顺序排列的有序序列，即是将无序的记录序列调整为有序记录序列的一种操作。

1、交换类排序法（方法：冒泡排序，快速排序）。

2、插入类排序法（方法：简单插入排序，希尔排序）。

3、选择类排序法（方法：简单选择排序，堆排序）。

总结：各种排序法比较：**以下的表是重点**

类别	排序方法	基本思想	时间复杂度
交换类	冒泡排序	相邻元素比较，不满足条件时交换	n(n-1)/2
	快速排序	选择基准元素，通过交换，划分成两个子序列	$O(n\log_2 n)$
插入类	简单插入排序	待排序的元素看成为一个有序表和一个无序表，将无序表中元素插入到有序表中	n(n-1)/2
	希尔排序	分割成若干个子序列分别进行直接插入排序	$O(n^{1.5})$
选择类	简单选择排序	扫描整个线性表，从中选出最小的元素，将它交换到表的最前面	n(n-1)/2
	堆排序	选建堆，然后将堆顶元素与堆中最后一个元素交换，再调整为堆	$O(n\log_2 n)$

批注 [A12]: 第 10 题，第 13 题，第 15 题，第 16 题。这里快速排序也为 n(n-1)/2

本章应考点拨：本章内容在笔试中会出现 5-6 个题目，是公共基础知识部分出题量比较多的一章，所占分值也比较大，约 10 分。

第二章　程序设计基础

2.1 程序设计风格

程序设计的风格主要强调:**“清晰第一，效率第二”**。可读性。主要应注重和考虑下述一些因素：

（1）源程序文档化。

1）符号名的命名。符号名能反映它所代表的实际东西，应有一定的实际含义。

2）**程序的注释。分为序言性注释和功能性注释**。

序言性注释：位于程序开头部分，包括程序标题、程序功能说明、主要算法、接口说明、程序位置、开发简历、程序设计者、复审者、复审日期及修改日期等。

功能性注释：嵌在源程序体之中，用于描述其后的语句或程序的主要功能。

3）视觉组织。利用空格、空行、缩进等技巧使程序层次清晰。

（2）数据说明。1）数据说明的次序规范化；2）说明语句中变量安排有序化；3）使用注释来说明复杂数据的结构。

（3）语句的结构。1）在一行内只写一条语句；2）程序编写应优先考虑清晰性；3）程序编写要做到清晰第一，效率第二；4）在保证程序正确的基础上再要求提高效率；5）避免使用临时变量而使程序的可读性下降；6）避免不必要的转移；7）尽量使用库函数；8）避免采用复杂的条件语句；9）尽量减少使用“否定”条件语句；10）数据结构要有利于程序的简化；11）要模块化，使模块功能尽可能单一化；12）利用信息隐蔽，确保每一个模块的独立性；13）从数据出发去构造程序；14）不要修补不好的程序，要重新编写。

（4）输入和输出。1）对输入数据检验数据的合法性；2）检查输入项的各种重要组合的合法性；3）输入格式要简单，使得输入的步骤和操作尽可能简单；4）输入数据时，应允许使用自由格式；5）应允许缺省值；6）输入一批数据时，最好使用输入结束标志；7）在以交互式输入/输出方式进行输入时，要在屏幕上使用提示符明确提示输入的请求，同时在数据输入过程中和输入结束时，应在屏幕上给出状态信息；8）当程序设计语言对输入格式有严格要求时，应保持输入格式与输入语句的一致性；给所有的输出加注释，并设计输出报表格式。

2.2 **结构化程序设计（面向过程的程序设计方法）**

1、**结构化程序设计方法的主要原则可以概括为：自顶向下，逐步求精，模块化，限制使用 goto 语句。尽量少使用 goto 语句**

（1）自顶向下。程序设计时，应先考虑总体，后考虑细节；先考虑全局目标，后考虑局部目标。不要一开始就过多追求众多的细节，先从最上层总目标开始设计，逐步使问题具体化。

（2）逐步求精。对复杂问题，应设计一些子目标作过渡，逐步细化。

（3）模块化。一个复杂问题，肯定是由若干稍简单的问题构成。模块化是把程序要解决的总目标分解为分目标，再进一步分解为具体的小目标，把每个小目标称为一个模块。

（4）限制使用 goto 语句。

2、**结构化程序的基本结构：顺序结构，选择（分支）结构，重复（循环）结构。**

1）顺序结构。一种简单的程序设计，即按照程序语句行的自然顺序，一条语句一条语句地执行程序，它是最基本、最常用的结构。

2）选择结构。又称分支结构，包括简单选择和多分支选择结构，可根据条件，判断应该选择哪一条分支来执行相应的语句序列。

3）重复结构。又称循环结构，可根据给定的条件，判断是否需要重复执行某一相同的或类似的程序段。

批注 [wx13]: “清晰第一，效率第二”是当今主导的程序设计风格。

批注 [wx14]: 信息隐蔽是指采用封装技术，将程序模块的实施细节隐藏起来，使模块接口尽量简单。即指在设计和确定模块时，使得一个模块内包含的信息（过程或数据），对于不需要这些信息的其它模块来说，是不能访问的。

仅仅使用顺序、选择和循环三种基本控制结构就足以表达各种其他形式结构，从而实现任何单入口/单出口的程序。

2.3 面向对象的程序设计

客观世界中任何一个事物都可以被看成是一个对象，面向对象方法的本质就是主张从客观世界固有的事物出发来构造系统，提倡人们在现实生活中常用的思维来认识、理解和描述客观事物，强调最终建立的系统能够映射问题域。也就是说，系统中的对象及对象之间的关系能够如实地反映问题域中固有的事物及其关系。

面向对象方法的主要优点：（1）与人类习惯的思维方法一致；（2）稳定性好；（3）可重用性好；（4）易于开发大型软件产品；（5）可维护性好。

批注 [wx15]: 软件的重用是指在不同的软件开发过程中重复使用相同或相似软件的过程。

***：面向对象的程序设计主要考虑的是提高软件的可重用性。**

对象是面向对象方法中最基本的概念，可以用来表示客观世界中的任何实体，对象是实体的抽象。面向对象的程序设计方法中的对象是系统中用来描述客观事物的一个实体，是构成系统的一个基本单位，由一组表示其静态特征的属性和它可执行的一组操作组成。对象是属性和方法的封装体。

属性即对象所包含的信息，它在设计对象时确定，一般只能通过执行对象的操作来改变。

操作描述了对象执行的功能，操作也称为方法或服务。操作是对象的动态属性。

***：一个对象由对象名、属性和操作三部分组成。**

对象的基本特点：标识惟一性，分类性，多态性，封装性，模块独立性好。

（1）标识惟一性。指对象是可区分的，并且由对象的内在本质来区分，而不是通过描述来区分。

（2）分类性。指可以将具有相同属性的操作的对象抽象成类。

（3）多态性。指同一个操作可以是不同对象的行为。

（4）封装性。从外面看只能看到对象的外部特性，即只需知道数据的取值范围和可以对该数据施加的操作，根本无需知道数据的具体结构以及实现操作的算法。对象的内部，即处理能力的实行和内部状态，对外是不可见的。从外面不能直接使用对象的处理能力，也不能直接修改其内部状态，对象的内部状态只能由其自身改变。

*：信息隐蔽是通过对象的封装性来实现的。

（5）模块独立性好。对象是面向对象的软件的基本模块，它是由数据及可以对这些数据施加的操作所组成的统一体，而且对象是以数据为中心的，操作围绕对其数据所需做的处理来设置，没有无关的操作。从模块的独立性考虑，对象内部各种元素彼此结合得很紧密，内聚性强。

类是指具有共同属性、共同方法的对象的集合。所以类是对象的抽象，对象是对应类的一个实例。

消息是一个实例与另一个实例之间传递的信息。消息的组成包括：（1）接收消息的对象的名称；（2）消息标识符，也称消息名；（3）零个或多个参数。

*：在面向对象方法中，一个对象请求另一个对象为其服务的方式是通过发送消息。

继承是指能够直接获得已有的性质和特征，而不必重复定义他们。继承分单继承和多重继承。单继承指一个类只允许有一个父类，多重继承指一个类允许有多个父类。

***：类的继承性是类之间共享属性和操作的机制，它提高了软件的可重用性。**

多态性是指同样的消息被不同的对象接受时可导致完全不同的行动的现象。

本章应考点拨：本章在考试中会出现约 1 个题目，所占分值大约占 2 分，是出题量较小的一章。本章内容比较少，也很简单，掌握住基本的概念就可以轻松应对考试了，所以在这部分丢分，比较可惜。

第三章　软件工程基础

3.1 软件工程基本概念

1、软件的相关概念

计算机软件是包括程序、数据及相关文档的完整集合。

软件的特点包括：1）软件是一 f 种逻辑实体，而不是物理实体，具有抽象性；2）软件的生产与硬件不同，它没有明显的制作过程；3）软件在运行、使用期间不存在磨损、老化问题；4）软件的开发、运行对计算机系统具有依赖性，受计算机系统的限制，这导致了软件移植的问题；5）软件复杂性高，成本昂贵；6）软件开发涉及诸多的社会因素。

批注 [wx16]: 软件的这个特点使它与其它工程对象有着明显的差异。人们可以把它记录在存储介质上，但却无法看到软件本身的形态，必须通过观察、分析、思考、判断，才能了解它的功能、性能等特性。

批注 [wx17]: 许多软件的开发和运行涉及软件用户的机构设置，体制问题以及管理方式等，甚至涉及到人们的观念和心理，软件知识产权及法律等问题。

2、软件危机与软件工程

软件工程源自软件危机。所谓软件危机是泛指在计算机软件的开发和维护过程中所遇到的一系列严重问题。具体的说，在软件开发和维护过程中，软件危机主要表现在：

1）软件需求的增长得不到满足。用户对系统不满意的情况经常发生。

2）软件开发成本和进度无法控制。开发成本超出预算，开发周期大大超过规定日期的情况经常发生。

3）软件质量难以保证。

4）软件不可维护或维护程度非常低。

5）软件的成本不断提高。

6）**软件开发生产率的提高跟不上硬件的发展和应用需求的增长。**

总之，可以将软件危机可以归结为成本、质量、生产率等问题。

软件工程是应用于计算机软件的定义、开发和维护的一整套方法、工具、文档、实践标准和工序。软件工程的目的就是要建造一个优良的软件系统，它所包含的内容概括为以下两点：

1）软件开发技术，主要有软件开发方法学、软件工具、软件工程环境。

2）软件工程管理，主要有软件管理、软件工程经济学。

软件工程的主要思想是将工程化原则运用到软件开发过程，**它包括 3 个要素：方法、工具和过程。**方法是完成软件工程项目的技术手段；工具是支持软件的开发、管理、文档生成；过程支持软件开发的各个环节的控制、管理。

软件工程过程是把输入转化为输出的一组彼此相关的资源和活动。

3、软件生命周期（诞生）

软件生命周期：软件产品从提出、实现、使用维护到停止使用退役的过程。

软件生命周期分**为软件定义、软件开发及软件运行维护**三个阶段：

1）软件定义阶段：包括制定计划和需求分析。

制定计划：确定总目标；可行性研究；探讨解决方案；制定开发计划。

需求分析：对待开发软件提出的需求进行分析并给出详细的定义。

2）软件开发阶段：

软件设计：分为概要设计和详细设计两个部分。

软件实现：把软件设计转换成计算机可以接受的程序代码。

软件测试：在设计测试用例的基础上检验软件的各个组成部分。

3）软件运行维护阶段：软件投入运行，并在使用中不断地维护，进行必要的扩充和删改。

*：软件生命周期中所花费最多的阶段是软件运行维护阶段。

4、软件工程的目标和与原则

（1）软件工程目标：在给定成本、进度的前提下，开发出具有有效性、可靠性、可理解性、可维护性、可重用性、可适应性、可移植性、可追踪性和可互操作性且满足用户需求的产品。

（2）软件工程需要达到的基本目标应是：付出较低的开发成本；达到要求的软件功能；取得较好的软件性能；开发的软件易于移植；需要较低的维护费用；能按时完成开发，及时交付使用。

（3）软件工程原则：抽象、信息隐蔽、模块化、局部化、确定性、一致性、完备性和可验证性。

1)抽象：抽象是事物最基本的特性和行为，忽略非本质细节，采用分层次抽象，自顶向下，逐层细化的办法控制软件开发过程的复杂性。

2）信息隐蔽：采用封装技术，将程序模块的实现细节隐蔽起来，使模块接口尽量简单。

3）模块化：模块是程序中相对独立的成分，一个独立的编程单位，应有良好的接口定义。模块的大小要适中，模块过大会使模块内部的复杂性增加，不利于模块的理解和修改，也不利于模块的调试和重用；模块太小会导致整个系统表示过于复杂，不利于控制系统的复杂性。

4）局部化：保证模块间具有松散的耦合关系，模块内部有较强的内聚性。

5）确定性：软件开发过程中所有概念的表达应是确定、无歧义且规范的。

6）一致性：程序内外部接口应保持一致，系统规格说明与系统行为应保持一致。

7）完备性：软件系统不丢失任何重要成分，完全实现系统所需的功能。

8）可验证性：应遵循容易检查、测评、评审的原则，以确保系统的正确性。

5、软件开发工具与软件开发环境

(1)软件开发工具

软件开发工具的完善和发展将促使软件开发方法的进步和完善，促进软件开发的高速度和高质量。软件开发工具的发展是从单项工具的开发逐步向集成工具发展的，软件开发工具为软件工程方法提供了自动的或半自动的软件支撑环境。同时，软件开发方法的有效应用也必须得到相应工具的支持，否则方法将难以有效的实施。

（2）软件开发环境

软件开发环境（或称软件工程环境）是全面支持软件开发全过程的软件工具集合。

计算机辅助软件工程（CASE，Computer Aided Software Engineering）将各种软件工具、开发机器和一个存放开发过程信息的中心数据库组合起来，形成软件工程环境。它将极大降低软件开发的技术难度并保证软件开发的质量。

3.2 结构化分析方法

结构化方法的核心和基础是结构化程序设计理论。

1、需求分析

需求分析方法有：1）结构化需求分析方法；2）面向对象的分析方法。

*：需求分析的任务就是导出目标系统的逻辑模型，解决“做什么”的问题。

*：需求分析一般分为需求获取、需求分析、编写需求规格说明书和需求评审四个步骤进行。

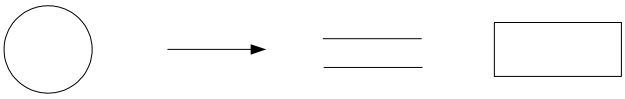
2、结构化分析方法

结构化分析方法是结构化程序设计理论在软件需求分析阶段的应用。

结构化分析方法的实质：着眼于数据流，自顶向下，逐层分解，建立系统的处理流程，以数据流图和数据字典为主要工具，建立系统的逻辑模型。

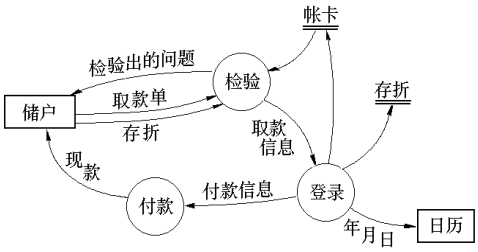
结构化分析的常用工具：1）数据流图（DFD）；2）数据字典（DD）；3）判定树；4）判定表。DD 为了解释 DFD

数据流图以图形的方式描绘数据在系统中流动和处理的过程，它反映了系统必须完成的逻辑功能，是结构化分析方法中用于表示系统逻辑模型的一种工具。



加工　　数据流　　存储文件　　源、潭

上图是数据流图的基本图形元素：
加工（转换）：输入数据经加工变换产生输出。
数据流：沿箭头方向传送数据的通道，一般在旁边标注数据流名。
存储文件（数据源）：表示处理过程中存放各种数据的文件。
源，潭：表示系统和环境的接口，属系统之外的实体。
画数据流图的基本步骤：自外向内，自顶向下，逐层细化，完善求精。
下图是一个数据流图的示例：



数据字典：对所有与系统相关的数据元素的一个有组织的列表，以及精确的、严格的定义，使得用户和系统分析员对于输入、输出、存储成分和中间计算结果有共同的理解。
*：**数据字典的作用是对数据流图中出现的被命名的图形元素的确切解释**。
*：数据字典是结构化分析方法的核心。
3、软件需求规格说明书（SRS）
软件需求规格说明书是需求分析阶段的最后成果，通过建立完整的信息描述、详细的功能和行为描述、性能需求和设计约束的说明、合适的验收标准，给出对目标软件的各种需求。
3.3 结构化设计方法
1、软件设计的基础
*：需求分析主要解决“做什么”的问题，而软件设计主要解决“怎么做”的问题。
从技术观点来看，软件设计包括软件结构设计、数据设计、接口设计、过程设计。
结构设计：定义软件系统各主要部件之间的关系。
数据设计：将分析时创建的模型转化为数据结构的定义。
接口设计：描述软件内部、软件和协作系统之间以及软件与人之间如何通信。
过程设计：把系统结构部件转换成软件的过程性描述。
从**工程角度来看，软件设计分两步完成，即概要设计和详细设计**。
概要设计：又称结构设计，将软件需求转化为软件体系结构，确定系统级接口、全局数据结构或数据库模式。
详细设计：确定每个模块的实现算法和局部数据结构，用适当方法表示算法和数据结构的细节。
软件设计的基本原理包括：抽象、模块化、信息隐蔽和模块独立性。
1）抽象。抽象是一种思维工具，就是把事物本质的共同特性提取出来而不考虑其他细节。
2）模块化。解决一个复杂问题时自顶向下逐步把软件系统划分成一个个较小的、相对独立但又不相互关联的模块的过程。
3）信息隐蔽。每个模块的实施细节对于其他模块来说是隐蔽的。
4）模块独立性。软件系统中每个模块只涉及软件要求的具体的子功能，而和软件系统中其他的模块的接口是简单的。
*：模块分解的主要指导思想是信息隐蔽和模块独立性。

模块的耦合性和内聚性是衡量软件的模块独立性的两个定性指标。

内聚性：是一个模块内部各个元素间彼此结合的紧密程度的度量。

*：按内聚性由弱到强排列，内聚可以分为以下几种：偶然内聚、逻辑内聚、时间内聚、过程内聚、通信内聚、顺序内聚及功能内聚。

耦合性：是模块间互相连接的紧密程度的度量。

*：按耦合性由高到低排列，耦合可以分为以下几种：内容耦合、公共耦合、外部耦合、控制耦合、标记耦合、数据耦合以及非直接耦合。

一个设计良好的软件系统应具有**高内聚、低耦合**的特征。大内高手

在结构化程序设计中，模块划分的原则是：模块内具有高内聚度，模块间具有低耦合度。

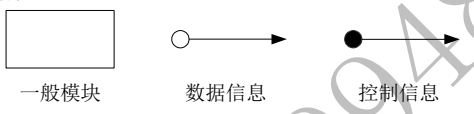
2、总体设计（概要设计）和详细设计

（1）总体设计（概要设计）

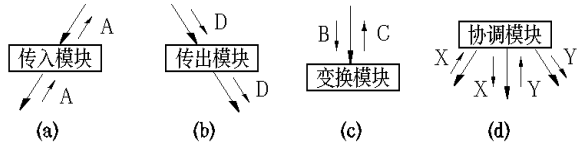
软件概要设计的基本任务是：1）设计软件系统结构；2）数据结构及数据库设计；3）编写概要设计文档；4）概要设计文档评审。

常用的软件结构设计工具是结构图，也称程序结构图。程序结构图的基本图符：

模块用一个矩形表示，箭头表示模块间的调用关系。在结构图中还可以用带注释的箭头表示模块调用过程中来回传递的信息。还可用带实心圆的箭头表示传递的是控制信息，空心圆箭心表示传递的是数据信息。



经常使用的结构图有四种模块类型：传入模块、传出模块、变换模块和协调模块。其表示形式如下图：



它们的含义分别是：

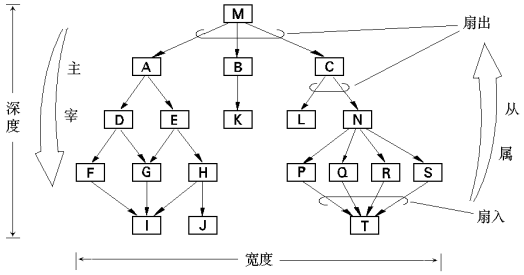
传入模块：从下属模块取得数据，经处理再将其传送给上级模块。

传出模块：从上级模块取得数据，经处理再将其传送给下属模块。

变换模块：从上级模块取得数据，进行特定的处理，转换成其他形式，再传送给上级模块。

协调模块：对所有下属模块进行协调和管理的模块。

程序结构图的例图及有关术语列举如下：



深度：表示控制的层数。

上级模块、从属模块：上、下两层模块 a 和 b，且有 a 调用 b，则 a 是上级模块，b 是从属模块。

宽度：整体控制跨度（最大模块数的层）的表示。
扇入：调用一个给定模块的模块个数。
扇出：一个模块直接调用的其他模块数。
原子模块：树中位于叶子结点的模块。
面向数据流的设计方法定义了一些不同的映射方法，利用这些方法可以把数据流图变换成结构图表示软件的结构。
数据流的类型：大体可以分为两种类型，变换型和事务型。
A、变换型：变换型数据处理问题的工作过程大致分为三步，即取得数据、变换数据和输出数据。变换型系统结构图由输入、中心变换、输出三部分组成。
B、事务型：事务型数据处理问题的工作机理是接受一项事务，根据事务处理的特点和性质，选择分派一个适当的处理单元，然后给出结果。
（2）详细设计
详细设计是为软件结构图中的每一个模块确定实现算法和局部数据结构，用某种选定的表达工具表示算法和数据结构的细节。
*：详细设计的任务是确定实现算法和局部数据结构，不同于编码或编程。
常用的过程设计（即详细设计）工具有以下几种：
图形工具：程序流程图、N-S（方盒图）、PAD（问题分析图）和 HIPO（层次图+输入/处理/输出图）。
表格工具：判定表。
语言工具：PDL（伪码）
数据流图中的箭头表示的是数据流，
3.4 软件测试
1、软件测试定义：使用人工或自动手段来运行或测定某个系统的过程，其目的在于检验它是否满足规定的需求或是弄清预期结果与实际结果之间的差别。
*：**软件测试的目的：尽可能地多发现程序中的错误**，不能也不可能证明程序没有错误。软件测试的关键是设计测试用例，一个好的测试用例能找到迄今为止尚未发现的错误。
2、软件测试方法：**静态测试和动态测试。**
静态测试：包括代码检查、静态结构分析、代码质量度量。不实际运行软件，主要通过人工进行。
动态测试：是基于计算机的测试，**主要包括白盒测试方法和黑盒测试方法**。
（1）白盒测试
白盒测试方法也称为结构测试或逻辑驱动测试。它是根据软件产品的内部工作过程，检查内部成分，以确认每种内部操作符合设计规格要求。
白盒测试的基本原则：保证所测模块中每一独立路径至少执行一次；保证所测模块所有判断的每一分支至少执行一次；保证所测模块每一循环都在边界条件和一般条件下至少各执行一次；验证所有内部数据结构的有效性。
***：白盒测试法的测试用例是根据程序的内部逻辑来设计的，主要用软件的单元测试，主要方法有逻辑覆盖、基本路径测试等。 白盒是可以看见。**
A、逻辑覆盖。逻辑覆盖泛指一系列以程序内部的逻辑结构为基础的测试用例设计技术。通常程序中的逻辑表示有判断、分支、条件等几种表示方法。
语句覆盖：选择足够的测试用例，使得程序中每一个语句至少都能被执行一次。
路径覆盖：执行足够的测试用例，使程序中所有的可能的路径都至少经历一次。
判定覆盖：使设计的测试用例保证程序中每个判断的每个取值分支（T 或 F）至少经历一次。
条件覆盖：设计的测试用例保证程序中每个判断的每个条件的可能取值至少执行一次。

批注 [wx18]: 单元是程序中最小的部分，由可以隐含的三部分组成：数据输入、加工和数据输出。

批注 [wx19]: PDL（伪码）：过程设计语言，它是用正文形式表示数据和处理过程的设计工具。

批注 [wx20]: 测试用例是指对一项特定的软件产品进行测试任务的描述，体现测试方案、方法、技术和策略。

判断-条件覆盖：设计足够的测试用例，使判断中每个条件的所有可能取值至少执行一次，同时每个判断的所有可能取值分支至少执行一次。

*：逻辑覆盖的强度依次是：语句覆盖<路径覆盖<判定覆盖<条件覆盖<判断-条件覆盖。

B、基本路径测试。其思想和步骤是，根据软件过程性描述中的控制流程确定程序的环路复杂性度量，用此度量定义基本路径集合，并由此导出一组测试用例，对每一条独立执行路径进行测试。

（2）黑盒测试

黑盒测试方法也称为功能测试或数据驱动测试。黑盒测试是对软件已经实现的功能是否满足需求进行测试和验证。

黑盒测试主要诊断功能不对或遗漏、接口错误、数据结构或外部数据库访问错误、性能错误、初始化和终止条件错误。

黑盒测试不关心程序内部的逻辑，只是根据程序的功能说明来设计测试用例，**主要方法有等价类划分法、边界值分析法、错误推测法等，**主要用软件的确认测试。

A、等价类划分法。这是一种典型的黑盒测试方法，它是将程序的所有可能的输入数据划分成若干部分（及若干等价类），然后从每个等价类中选取数据作为测试用例。

B、边界值分析法。它是对各种输入、输出范围的边界情况设计测试用例的方法。

C、错误推测法。人们可以靠经验和直觉推测程序中可能存在的各种错误，从而有针对性地编写检查这些错误的用例。

3、**软件测试过程一般按 4 个步骤进行：单元测试、集成测试、确认测试和系统测试。**

（1）单元测试

单元测试是对软件设计的最小单位——模块（程序单元）进行正确性检测的测试，目的是发现各模块内部可能存在的各种错误。

单元测试根据程序的内部结构来设计测试用例，其依据是详细设计说明书和源程序。单元测试的技术可以采用静态分析和动态测试。对动态测试通常以白盒测试为主，辅之以黑盒测试。

单元测试的内容包括：模块接口测试、局部数据结构测试、错误处理测试和边界测试。

*：在进行单元测试时，要用一些辅助模块去模拟与被测模块相联系的其他模块，即为被测模块设计和搭建驱动模块和桩模块。其中，驱动模块相当于被测模块的主程序，它接收测试数据，并传给被测模块，输出实际测试结果；而桩模块是模拟其他被调用模块，不必将子模块的所有功能带入。

（2）集成测试

集成测试是测试和组装软件的过程，它是把模块在按照设计要求组装起来的同时进行测试，主要目的是发现与接口有关的错误。

集成测试的依据是概要设计说明书。

集成测试所涉及的内容包括：软件单元的接口测试、全局数据结构测试、边界条件和非法输入的测试等。

集成测试通常采用两种方式：非增量方式组装与增量方式组装。

非增量方式组装：也称为一次性组装方式。首先对每个模块分别进行模块测试，然后再把所有模块组装在一起进行测试，最终得到要求的软件系统。

增量方式组装：又称渐增式集成方式。首先对一个个模块进行模块测试，然后将这些模块逐步组装成较大的系统，在组装的过程中边连接边测试，以发现连接过程中产生的问题。最后通过增殖逐步组装成要求的软件系统。增量方式组装又包括自顶向下、自底向上、自顶向下与自底向上相结合等三种方式。

（3）确认测试

确认测试的任务是验证软件的有效性，即验证软件的功能和性能及其他特性是否与用户的

要求一致。

确认测试的主要依据是软件需求规格说明书。

确认测试主要运用黑盒测试法。

（4）系统测试

系统测试的目的在于通过与系统的需求定义进行比较，发现软件与系统定义不符合或与之矛盾的地方。

系统测试的测试用例应根据需求分析规格说明来设计，并在实际使用环境下来运行。

系统测试的具体实施一般包括：功能测试、性能测试、操作测试、配置测试、外部接口测试、安全性测试等。

3.5 程序的调试

程序调试的任务是诊断和改正程序中的错误，主要在开发阶段进行，调试程序应该由编制源程序的程序员来完成。

程序调试的基本步骤：（1）错误定位；（2）纠正错误；（3）回归测试。

*：软件的调试后要进行回归测试，防止引进新的错误。

软件调试可分为静态调试和动态调试。静态调试主要是指通过人的思维来分析源程序代码和排错，是主要的调试手段，而动态调试是辅助静态调试。

对软件主要的调试方法可以采用：

（1）强行排错法。主要方法有：通过内存全部打印来排错；在程序特定部位设置打印语句；自动调试工具。

（2）回溯法。发现了错误，分析错误征兆，确定发现“症状”的位置。一般用于小程序。

（3）原因排除法。是通过演绎、归纳和二分法来实现的。

1）演绎法。根据已有的测试用例，设想及枚举出所有可能出错的原因作为假设；然后再用原始测试数据或新的测试，从中逐个排除不可能正确的假设；最后，再用测试数据验证余下的假设确定出错的原因。

2）归纳法。从错误征兆着手，通过分析它们之间的关系来找出错误。大致分四步：收集有关的数据；组织数据；提出假设；证明假设。

3）二分法。在程序的关键点给变量赋正确值，然后运行程序并检查程序的输出。如果输出结果正确，则错误原因在程序的前半部分；反之，错误原因在程序的后半部分。

本章应考点拨：本章在笔试中一般占 8 分左右，约 3 道选择题，1 道填空题，是公共基础部分比较重要的一章。从出题的深度来看，本章主要考察对基本概念的识记，有少量对基本原理的理解，没有实际运用，因此考生在复习本章时，重点应放在基本概念的记忆和基本原理的理解上。

批注 [wx21]: 注意与软件测试区分。

批注 [wx22]: 这是因为修改程序可能带来新的错误，重复进行暴露这个错误的原始测试或某些有关测试，以确认该错误是否被排除、是否引进了新的错误。

第四章　数据库设计基础

4.1 数据库系统的基本概念

1、数据、数据库、数据管理系统

（1）数据：实际上就是描述事物的符号记录。

数据的特点：有一定的结构，有型与值之分。数据的型给出了数据表示的类型，如整型、实型、字符型等。而数据的值给出了符合给定型的值，如整型（INT）值 15。

（2）数据库（DB）：是数据的集合，具有统一的结构形式并存放于统一的存储介质内，是多种应用数据的集成，并可被各个应用程序所共享。

数据库存放数据是按数据所提供的数据模式存放的，具有集成与共享的特点，亦即是数据库集中了各种应用的数据，进行统一的构造和存储，而使它们可被不同应用程序所使用。

（3）数据库管理系统（DBMS）：一种系统软件，负责数据库中的数据组织、数据操纵、数据维护、控制及保护和数据服务等，是数据库的核心。

数据库系统：由数据库 DB，数据库管理系统 DBMS（如 VFP，ACCESS），数据库管理员，数据库应用系统 DBAS，硬件系统，软件系统组成

数据库系统的核心是：DBMS

数据库管理系统功能：

1）数据模式定义。数据库管理系统负责为数据库构建模式，也就是为数据库构建其数据框架。

2）数据存取的物理构建。数据库管理系统负责为数据模式的物理存取与构建提供有效的存取方法与手段。

3）数据操纵。数据库管理系统为用户使用数据库中的数据提供方便，它一般提供如查询、插入、修改以及删除数据的功能。此外，它自身还具有做简单的算术运算及统计的能力，而且还可以与某些过程性语言结合，使其具有强大的过程性操作能力。

4）数据的完整性、安生性定义与检查。数据库中的数据具有内在语义上的关联性与一致性，它们构成了数据的完整性，数据的完整性是保证数据库中数据正确的必要条件，因此必须经常检查以维护数据正确。数据库中的数据具有共享性，而数据共享可能会引发数据的非法使用，因此必须要对数据正确使用做出必要的规定，并在使用时做检查，这就是数据的安全性。数据完整性与安全性的维护是数据库系统的基本功能。

5）数据库的并发控制与故障恢复。数据库是一个集成、共享的数据集合体，它能为多个应用程序服务，所以就存在着多个应用程序对数据库的并发操作。在并发操作中如果不加控制和管理，多个应用程序间就会相互干扰，从而对数据库中的数据造成破坏。因此，数据库管理系统必须对多个应用程序的并发操作做必要的控制以保证数据不受破坏，这就是数据库的并发控制。数据库中的数据一旦遭到破坏，数据库管理系统必须有能力及时进行恢复，这就是数据库的故障恢复。

6）数据的服务。数据库管理系统提供对数据库中数据的多种服务功能，如数据拷贝、转存、重组、性能监测、分析等。

（4）数据库管理员（DBA）：对数据库进行规划、设计、维护、监视等的专业管理人员。

（5）**数据库系统（DBS）：由数据库（数据）、数据库管理系统（软件）、数据库管理员（人员）、硬件平台（硬件）、软件平台（软件）五个部分构成的运行实体。**

（6）数据库应用系统：由数据库系统、应用软件及应用界面三者组成。

*：数据库技术的根本目标是解决数据的共享问题。

2、数据库系统的发展

数据库管理发展至今已经历了三个阶段：**人工管理阶段、文件系统阶段和数据库系统**阶段。下表是数据管理三个阶段的比较：

		人工管理阶段	文件系统阶段	数据库系统阶段
背景	应用背景	科学计算	科学计算、管理	大规模管理
	硬件背景	无直接存取存储设备	磁盘、磁鼓	大容量磁备盘
	软件背景	没有操作系统	有文件系统	有数据库管理系统
	处理方式	批处理	联机实时处理、批处理	联机实时处理、分布处理、批处理
特	数据的管理者	用户（程序员）	文件系统	数据库管理系统

点	数据面向的对象	某一应用程序	某一应用	现实世界
	数据的共享程度	无共享，冗余度极大	共享性差，冗余度大	共享性高，冗余度小
	数据的独立性	不独立，完全依赖于程序	独立性差	具有高度的物理独立性和一定的逻辑独立性
	数据的结构化	无结构	记录内有结构，整体无结构	整体结构化，用数据模型描述
	数据控制能力	应用程序自己控制	应用程序自己控制	由数据库管理系统提供数据安全性、完整性、并发控制和恢复能力

3、数据库系统的基本特点

（1）数据的高集成性。

（2）数据的**高共享性与低冗余性**。

*：数据库系统可以减少数据冗余，但**无法避免一切冗余**。

批注 [wx23]: 在一个集合中的重复数据称为数据冗余。

（3）数据独立性：数据独立性是数据与程序间的互不依赖性，即数据库中数据独立于应用程序而不依赖于应用程序。也就是说，数据的逻辑结构、存储结构与存取方式的改变不会影响应用程序。

数据独立性一般分为物理独立性与逻辑独立性两级。

1）物理独立性：物理独立性即是数据的物理结构（包括存储结构，存取方式等）的改变，如存储设备的更换、物理存储的更换、存取方式改变等都不影响数据库的逻辑结构，从而不致引起应用程序的变化。

2）逻辑独立性：数据库总体逻辑结构的改变，如修改数据模式、增加新的数据类型、改变数据间联系等，不需要相应修改应用程序，这就是数据的逻辑独立性。

（4）数据统一管理与控制。

数据统一管理与控制主要包含以下三个方面：

1）数据的完整性检查：检查数据库中数据的正确性以保证数据的正确。

2）数据的安全性保护：检查数据库访问者以防止非法访问。

3）并发控制：控制多个应用的并发访问所产生的相互干扰以保证其正确性。

4、数据库系统的内部结构体系

（1）数据库系统的三级模式：

1）概念模式：数据库系统中全局数据逻辑结构的描述，是全体用户（应用）公共数据视图。

批注 [wx24]: 视图是从一个或几个基本表（或视图）导出的表，它与基本表不同，是一个虚表。数据库中只存放视图的定义，而不存放视图对应的数据，这些数据仍然存放在原来的基本表中。

2）外模式：也称子模式或**用户模式**，它是用户的数据视图，也就是**用户所见到**的数据模式，它由概念模式推导而出。

3）内模式：又称物理模式，它给出了数据库**物理存储结构与物理存取方法**。内模式的物理性主要体现在操作系统及文件级上，它还未深入到设备级上（如磁盘及磁盘操作）。内模式对一般用户是透明的，但它的设计直接影响数据库的性能。

（2）数据库系统的两级映射：

1）概念模式/内模式的映射：实现了概念模式到内模式之间的相互转换。当数据库的存储结构发生变化时，通过修改相应的概念模式/内模式的映射，使得数据库的逻辑模式不变，其外模式不变，应用程序不用修改，从而保证数据具有很高的物理独立性。

2）外模式/概念模式的映射：实现了外模式到概念模式之间的相互转换。当逻辑模式发生变化时，通过修改相应的外模式/逻辑模式映射，使得用户所使用的那部分外模式不变，从而应用程序不必修改，保证数据具有较高的逻辑独立性。

4.2 数据模型

1、数据模型

（1）数据模型的概念：是数据特征的抽象，它从抽象层次上描述了系统的静态特征、动态行为和约束条件，为数据库系统的信息表示与操作提供一个抽象的框架。

（2）数据模型所描述的内容有三个部分，它们是数据结构、数据操作与数据约束。

1）数据结构：数据结构是所研究的对象类型的集合，包括与数据类型、内容、性质有关的对象，以及与数据之间联系有关的对象。它用于描述系统的静态特性。

2）数据操作：数据操作是对数据库中各种对象（型）的实例（值）允许执行的操作的集合，包括操作的含义、符号、操作规则及实现操作的语句等。它用于描述系统的动态特性。

3）数据的约束条件：数据的约束条件是一组完整性规则的集合。完整性规则是给定的数据模型中数据及其联系所具有的制约和依存规则，用以限定符号数据模型的数据库状态及状态的变化，以保证数据的正确、有效和相容。

（3）数据模型分为概念模型、逻辑数据模型和物理模型三类：

1）概念数据模型：简称概念模型，是对客观世界复杂事物的结构描述及它们之间的内在联系的刻画。概念模型主要有：E-R 模型(实体联系模型)、扩充的 E-R 模型、面向对象模型及谓词模型等。

2）逻辑数据模型：又称数据模型，是一种面向数据库系统的模型，该模型着重于在数据库系统一级的实现。逻辑数据模型主要有：层次模型、网状模型、关系模型、面向对象模型等。

3）物理数据模型：又称物理模型，它是一种面向计算机物理表示的模型，此模型给出了数据模型在计算机上物理结构的表示。

2、实体联系模型及 **E-R 图**

（1）E-R 模型的基本概念：

1）实体：现实世界中的事物。

2）属性：事物的特性。

3）联系：现实世界中事物间的关系。实体集的关系有一对一、一对多、多对多的联系。

E-R 模型三个基本概念之间的联接关系：1）实体集（联系）与属性间的联接关系；2）实体（集）与联系。

*：E-R 模型的基本成分是实体和联系。

（2）E-R 模型的图示法：

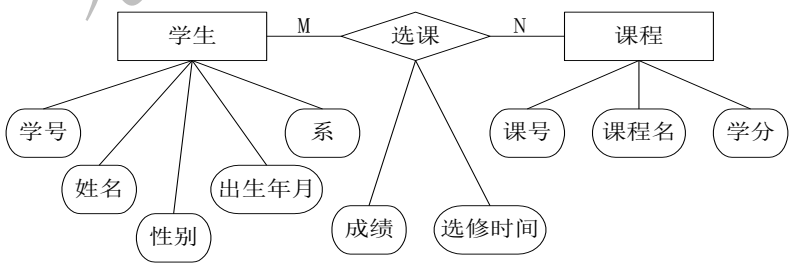
1）实体集：用矩形表示。

2）属性：用椭圆形表示。

3）联系：用菱形表示。

4）实体集与属性间的联接关系：用无向线段表示。

5）实体集与联系间的联接关系：用无向线段表示。

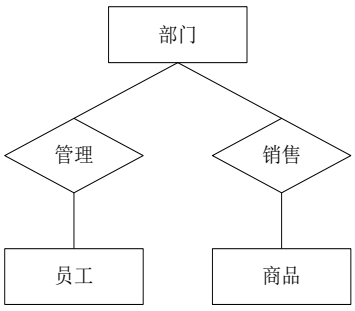


（3）数据库管理系统常见的**数据模型有层次模型、网状模型和关系模型三种。**

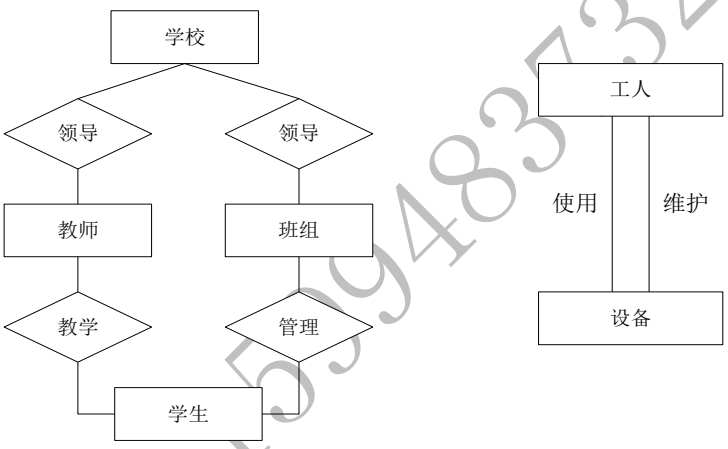
1）层次模型的基本结构是树形结构，具有以下特点：A、每棵树有且仅有一个无双亲结点，

批注 [wx25]: 层次模型是最早发展起来的数据库模型。

称为根；B、树中除根外所有结点有且仅有一个双亲。



2）网状模型是层次模型的一个特例，从图论上看，网状模型是一个不加任何条件限制的无向图。



(a)教学关系E-R图　　(b)工作与设备

3）**关系模型采用二维表来表示，简称表，由表框架及表的元组组成。一个二维表就是一个关系。**

二维表的表框架由 n 个命名的属性组成，n 称为属性元数。每个属性有一个取值范围称为值域。表框架对应了关系的模式，即类型的概念。在表框架中按行可以存放数据，每行数据称为元组，实际上，一个元组是由 n 个元组分量所组成，每个元组分量是表框架中每个属性的投影值。

学号	姓名	性别	出生年月	班级	籍贯
2007102	张洁然	男	07-07-88	07 动画 1 班	天津
2007203	李一明	男	05-01-87	07 播音 5 班	广西南宁
2007305	王　丽	女	04-09-88	07 管理 4 班	辽宁沈阳
2007406	刘　宏	男	10-11-88	07 新闻 3 班	江苏南京

*：同一个关系模型的任两个元组值不能完全相同。

主码：或称为关键字、主键，简称码、键，表中的一个属性或几个属性的组合、其值能唯一地标识表中一个元组的，称为关系的主码或关键字。例如，学生的学号。主码属性不能取空

值。

外部关键字：或称为外键，在一个关系中含有与另一个关系的关键字相对应的属性组称为该关系的外部关键字。外部关键字取空值或为外部表中对应的关键字值。例如，在学生表中含有的所属班级名字，是班级表中的关键字属性，它是学生表中的外部关键字。

（4）关系中的数据约束：

1）实体完整性约束：要求关系的主键中属性值不能为空值，因为主键是唯一决定元组的，如为空值则其唯一性就成为不可能的了。

2）参照完整性约束：关系之间相互关联的基本约束，不允许关系引用不存在的元组，即在关系中的外键要么是所关联关系中实际存在的元组，要么为空值。

3）用户定义的完整性约束：反映某一具体应用所涉及的数据必须满足的语义要求。例如某个属性的取值范围在 0—100 之间等。

3、从 E-R 图导出关系数据模型

数据库的逻辑设计的主要工作是将 E-R 图转换成指定 RDBMS（关系数据库管理系统）中的关系模式。首先，从 E-R 图到关系模式的转换是比较直接的，实体与联系都可以表示成关系，E-R 图中属性也可以转换成关系的属性。实体集也可以转换成关系。

4.3 关系代数

1、关系的数据结构

关系是由若干个不同的元组所组成，因此关系可视为元组的集合。n 元关系是一个 n 元有序组的集合。

关系模型的基本运算：1）插入；2）删除；3)修改；4）查询（包括投影、选择、笛卡尔积运算）。

2、关系操纵

关系模型的数据操纵即是建立在关系上的数据操纵，一般有查询、增加、删除和修改四种操作。

3、**集合运算及选择、投影、连接运算**

（1）并（∪）：关系 R 和 S 具有相同的关系模式，R 和 S 的并是由属于 R 或属于 S 的元组构成的集合。

（2）差（－）：关系 R 和 S 具有相同的关系模式，R 和 S 的差是由属于 R 但不属于 S 的元组构成的集合。

（3）交（∩）：关系 R 和 S 具有相同的关系模式，R 和 S 的交是由属于 R 且属于 S 的元组构成的集合。

（4）**广义笛卡尔积**（×）：设关系 R 和 S 的属性个数分别为 n、m，则 R 和 S 的广义笛卡尔积是一个有（n+m）列的元组的集合。每个元组的前 n 列来自 R 的一个元组，后 m 列来自 S 的一个元组，记为 R×S。

*：根据笛卡尔积的定义：有 n 元关系 R 及 m 元关系 S，它们分别有 p、q 个元组，则关系 R 与 S 经笛卡尔积记为 R×S，该关系是一个 n+m 元关系，元组个数是 p×q，由 R 与 S 的有序组组合而成。

例：有两个关系 R 和 S，分别进行并、差、交和广义笛卡尔积运算。

R

A	B	C
a1	b1	c1
a1	b2	c2
a2	b2	c1

(a)

S

A	B	C
a1	b2	c2
a1	b3	c2
a2	b2	c1

(b)

R∪S

A	B	C
a1	b1	c1
a1	b2	c2
a2	b2	c1
a1	b3	c2

(c)

R-S

A	B	C
a1	b1	c1

(d)

R∩S

A	B	C
a1	b2	c2
a2	b2	c1

(e)

R×S

R.A	R.B	R.C	S.A	S.B	S.C
a1	b1	c1	a1	b2	c2
a1	b1	c1	a1	b3	c2
a1	b1	c1	a2	b2	c1
a1	b2	c2	a1	b2	c2
a1	b2	c2	a1	b3	c2
a1	b2	c2	a2	b2	c1
a2	b2	c1	a1	b2	c2
a2	b2	c1	a1	b3	c2
a2	b2	c1	a2	b2	c1

(f)

（5）在关系型数据库管理系统中，基本的关系运算有选择、投影与联接三种操作：

1）选择：选择指的是从二维关系表的全部记录中，把那些符合指定条件的记录挑出来。
选的是行。

2）投影：投影是从所有字段中选取一部分字段及其值进行操作，它是一种纵向操作。
选的是列

3）联接：联接将两个关系模式拼接成一个更宽的关系模式，生成的新关系中包含满足联接条件的元组。
合并两个表
元组（行或记录）和属性（列或字段）

4.4 数据库设计方法和步骤

（1）数据库设计阶段包括：需求分析、概念分析、逻辑设计、物理设计。

（2）数据库设计的每个阶段都有各自的任务：

1）需求分析阶段：这是数据库设计的第一个阶段，任务主要是收集和分析数据，这一阶段收集到的基础数据和数据流图是下一步设计概念结构的基础。

2）概念设计阶段：分析数据间内在语义关联，在此基础上建立一个数据的抽象模型，即形成 E-R 图。

*：数据库概念设计的过程包括选择局部应用、视图设计和视图集成。

3）逻辑设计阶段：将 E-R 图转换成指定 RDBMS 中的关系模式。

4）物理设计阶段：对数据库内部物理结构作调整并选择合理的存取路径，以提高数据库访问速度及有效利用存储空间。

本章应考点拨：本章在考试中一般出现 2-4 个小题。本章内容概括性强，比较抽象，难于理解，因此建议考生在复习的时候，首先熟读讲义，其次对数据库系统的基本概念及原理等知识要注意理解、加强记忆。

批注 [wx26]: 关系型数据库管理系统（RDBMS）是引入基于关系型模型的一个数据库管理系统(DBMS)。这个系统必须满足以下最小标准：（1）对用户以关系显示数据（以表格形式显示）；（2）提供关系运算以表格形式维护这些数据。
*：VFP 是一种关系型数据库管理系统。