目录

全国计算机等级考试二级公共基础知识考纲

考试内容

一、 基本数据结构与算法

1. 算法的基本概念；算法复杂度的概念和意义（时间复杂度与空间复杂度）。
2. 数据结构的定义；数据的逻辑结构与存储结构；数据结构的图形表示；线性结构与非线性结构的概念。
3. 线性表的定义；线性表的顺序存储结构及其插入与删除运算。
4. 栈和队列的定义；栈和队列的顺序存储结构及其基本运算。
5. 线性单链表、双向链表与循环链表的结构及其基本运算。
6. 树的基本概念；二叉树的定义及其存储结构；二叉树的前序、中序和后序遍历。
7. 顺序查找与二分法查找算法；基本排序算法（交换类排序，选择类排序，插入类排序）。

二、 程序设计基础

1. 程序设计方法与风格。
2. 结构化程序设计。
3. 面向对象的程序设计方法，对象，方法，属性及继承与多态性。

三、 软件工程基础

1. 软件工程基本概念，软件生命周戎概念，软件工具与软件开发环境。
2. 结构化分析方法，数据流图，数据字典，软件需求规格说明书。
3. 结构化设计方法，总体设计与详细设计。
4. 软件测试的方法，白盒测试与黑盒测试，测试用例设计，软件测试的实施，单元测试、集成测试和系统测试。
5. 程序的调试，静态调试与动态调试。

四、 数据库设计基础

1. 数据库的基本概念：数据库，数据库管理系统，数据库系统。
2. 数据模型，实体联系模型及 E-R 图，从 E-R 图导出关系数据模型。
3. 关系代数运算，包括集合运算及选择、投影、连接运算，数据库规范化理论。
4. 数据库设计方法和步骤：需求分析、概念设计、逻辑设计和物理设计的相关策略。

考试方式

公共基础的考试方式为笔试，与 C 语言（VisualBASIC、Visual FoxPro、Java、Access、Visual C++）的笔试部分合为一张试卷。公共基础部分占全卷的 30 分。公共基础知识有 10 道选择题和 5 道填空题。

第一章 数据结构与算法

一、内容要点

（一）算法

1．算法的基本概念

算法是指解题方案的准确而完整的描述。即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，没有二义性，同时该规则将在有限次运算后可终止。

1）算法的基本特征

（1）可行性

由于算法的设计是为了在某一个特定的计算工具上解决某一个实际的问题而设计的，因此，它总是受到计算工具的限制，使执行产生偏差。

如：计算机的数值有效位是有限的，当大数和小数进行运算时，往往会因为有效位数的影响而使小数丢失，因此，在算法设计时，应该考虑到这一点。

（2）确定性

算法的设计必须是每一个步骤都有明确的定义，不允许有模糊的解释，也不能有多义性。

例如，一个实际的问题，小宝和萍萍共有 12 个苹果，小宝比萍萍多 4 个，请问小宝和萍萍各有几个苹果？这个问题，我们可以立一个方程 $\begin{cases} x+y=12 \\ x-y=4 \end{cases}$ 来求解，要求 x 和 y 的值，公式是正确的，但如何让计算能够进行计算，我们的算法不能把公式直接输进去，而应该设计出解题的步骤和过程。

即设计的算法是计算工具所能够正常解决问题的过程。

（3）有穷性

算法的有穷性，即在一定的时间是能够完成的，即算法应该在计算有限个步骤后能够正常结束。

例如，在数学中的无穷级数，在计算机中只能求有限项，即计算的过程是有穷的。

（4）拥有足够的情报

算法的执行与输入的数据和提供的初始条件相关，不同的输入或初始条件会有不同的输出结果，提供准确的初始条件和数据，才能使算法正确执行。

2）算法的基本要素

一是数据对象的运算和操作，二是算法的控制结构。

（1）算法中对数据的运算和操作

算法实际上是按解题要求从环境能进行的所有操作中选择合适的操作所组成的一组指令序列。即算法是计算机所能够处理的操作所组成的指令序列。

（2）算法的控制结构

算法的功能不仅取决于所选用的操作，而且还与各操作之间的顺序有关。

在算法中，操作的执行顺序又称算法的控制结构，一般的算法控制结构有三种：顺序结构、选择结构和循环结构。

在算法描述是，有相关的工具对这三种结构进行描述，常用的描述工具有：流程图、N-S 结构图和算法描述语言等。

3）算法设计的基本方法

为用计算机解决实际问题而设计的算法，即是计算机算法。

通常的算法设计有如下几种：

（1）列举法

列举法的基本思想是，根据提出的问题，列举出所有可能的情况，并用问题中给定的条件检验哪些是满足条件的，哪些是不满足条件的。列举法通常用于解决“是否存在”或“有哪些可能”等问题。

例如，我国古代的趣味数学题：“百钱买百鸡”、“鸡兔同笼”等，均可采用列举法进行解决。

使用列举法时，要对问题进行详细的分析，将与问题有关的知识条理化、完备化、系统化，从中找出规律。

（2）归纳法

归纳法的基本思想是，通过列举少量的特殊情况，经过分析，最后找出一般的关系。归纳是一种抽象，即从特殊现象中找出一般规律。但由于在归纳法中不可能对所有的情况进行列举，因此，该方法得到的结论只是一种猜测，还需要进行证明。

（3）递推

递推，即是从已知的初始条件出发，逐次推出所要求的各个中间环节和最后结果。其中初始条件或问题本身已经给定，或是通过对问题的分析与化简而确定。

递推的本质也是一种归纳，递推关系式通常是归纳的结果。

例如，裴波那契数列，是采用递推的方法解决问题的。

（4）递归

在解决一些复杂问题时，为了降低问题的复杂程序，通常是将问题逐层分解，最后归结为一些最简单的问题。这种将问题逐层分解的过程，并没有对问题进行求解，而只是当解决了最后的问题那些最简单的问题后，再沿着原来分解的逆过程逐步进行综合，这就是递归的方法。

递归分为直接递归和间接递归两种方法。如果一个算法直接调用自己，称为直接递归调用；如果一个算法 A 调用另一个算法 B，而算法 B 又调用算法 A，则此种递归称为间接递归调用。

（5）减半递推技术

减半递推即将问题的规模减半，然后，重复相同的递推操作。

例如，一元二次方程的求解。

（6）回溯法

有些实际的问题很难归纳出一组简单的递推公式或直观的求解步骤，也不能使用无限的列举。对于这类问题，只能采用试探的方法，通过对问题的分析，找出解决问题的线索，然后沿着这个线索进行试探，如果试探成功，就得到问题的解，如果不成功，再逐步回退，换别的路线进行试探。这种方法，即称为回溯法。

如人工智能中的机器人下棋。

2．算法复杂度

算法的复杂度包括时间复杂度和空间复杂度。

1）时间复杂度

即实现该算法需要的计算工作量。算法的工作量用算法所执行的基本运算次数来计算。

同一个问题规模下，如果算法执行所需要的基本次数取决于某一特定输入时，可以用以

下两种方法来分析算法的工作量：

算法工作量=f(n)

（1）平均性态

用各种特定输入下的基本运算次数的加权平均值来度量算法的工作量。

设 x 是某个可能输入中的某个特定输入，p(x)是 x 出现的概率，t(x)是算法在输入为 x 时所执行的基本运算次数，则算法的平均性态定义为：

$$A(n)=\sum_{x\in D_n} p(x)t(x)$$

D_n表示当规模为 n 时，算法执行时所有可能输入的集合。

（2）最坏情况复杂度

指在规模为 n 时，算法所执行的基本运算的最大次数。它定义为：

$$W(n)=\max_{x\in D_n}\{t(x)\}$$

例如，在具有 n 个元素的数列中搜索一个数 x。

平均性态：$A(n)=\sum_{i=1}^{n+1} p_i t_i=\sum_{i=1}^{n}\frac{q}{n}i+(1-q)n=\frac{(n+1)q}{2}+(1-q)n$

即该数在数列中任何位置出现的数列是相同的，也有可能不存在，存在的概率为 q。

如果有一半的机会存在，则概率 q 为 1/2，平均性态：

$$A(n)=\frac{(n+1)\times\frac{1}{2}}{2}+(1-\frac{1}{2})n\approx\frac{3}{4}n$$

如果查找的元素一定在数列中，则每个数存在的概率即为 1，则平均性态为：

$$A(n)=\frac{n+1}{2}\approx\frac{n}{2}$$

最坏情况分析：即要查找的元素 X 在数列的最后或不在数列中，显然，它的最坏情况复杂度为：$W(n)=\max\{t_i\,|\,1\le i\le n+1\}=n$

2）算法的空间复杂度

指要执行该算法所需要的内存空间。算法所占用的内存空间包括算法程序所占的空间、输入的初始数据所占的存储空间以及算法执行过程中所需要的额外空间，如执行过程中工作单元以及某种数据结构所需要的附加存储空间等。

（二）数据结构的基本概念

1．概念

数据结构是指相互有关联的数据元素的集合。它包括以下两个方面：

- 表示数据元素的信息
- 表示各数据之间的前后件关系

1）数据的逻辑结构

是指反映数据元素之间的逻辑关系的数据结构。

数据的逻辑结构有两个要素：

- 数据元素的集合，记作 D
- 数据之间的前后件关系，记作 R

则数据结构 B=（D，R）

2）数据的存储结构

数据的逻辑结构在计算机存储空间中的存放形式称为数据的存储结构，或数据的物理结构。

即数据存储时，不仅要存放数据元素的信息，而且要存储数据元素之间的前后件关系的信息。

通常的数据存储结构有顺序、链接、索引等存储结构。

2．数据结构的图形表示

数据结构的图形表示有两个元素：

- 中间标有元素值的方框表示数据元素，称为数据结点
- 用有向线段表示数据元素之间的前后件关系，即有向线段从前件结点指向后件结点

注意：在结构图中，没有前件的结点称为根结点，没有后件的结点称为终端结点，也称叶子结点。

3．线性结构与非线性结构

如果一个数据元素都没有，该数据结构称为空数据结构；在空数据结构中插入一个新的元素后数据结构变为非空数据结构；将数据结构中的所有元素均删除，则该数据结构变成空数据结构。

如果一个非空的数据结构满足如下条件，则该数据结构为线性结构：

- 有且只有一个根结点
- 每一个结点最多只有一个前件，也最多只有一个后件

线性结构又称线性表。

注意：在线性结构表中插入或删除元素，该线性表仍然应满足线性结构。

如果一个数据结构不满足线性结构，则称为非线性结构。

（三）线性表及其顺序存储结构

1．基本概念

线性表是最常用的数据结构，它由一组数据元素组成。

注意：这里的数据元素是一个广义的数据元素，并不仅仅是指一个数据。如，矩阵、学生记录表等。

非空线性表的结构特征：

- 有且只有一个根结点，它无前件
- 有且只有一个终端结点，它无后件
- 除根结点和终端结点之外，所有的结点有且只有一个前件和一个后件。线性表中结点的个数称为结点的长度 n。当 n=0 时，称为空表。

2．顺序存储结构

顺序存储结构的特点：

- 线性表中所有的元素所占的存储空间是连续的
- 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

通常，顺序存储结构中，线性表中每一个数据元素在计算机存储空间中的存储地址由该元素在线性表中的位置序号唯一确定。

线性表的顺序存储结构下的基本运算：

- 在指定位置插入一个元素
- 删除线性表中的指定元素
- 查找某个或某些特定的元素
- 线性表的排序
- 按要求将一个线性表拆分为多个线性表
- 将多个线性表合并为一个线性表
- 复制线性表
- 逆转一个线性表

3．线性表的基本操作

1）顺序表的插入运算

在顺序存储结构的线性表中插入一个元素。

注意：找到插入位置后，将插入位置开始的所有元素从最后一个元素开始顺序后移。另外，在定义线性表时，一定要定义足够的空间，否则，将不允许插入元素。

2）顺序表的删除运算

在顺序在存储结构的线性表中删除一个元素。

注意：找到删除的数据元素后，从该元素位置开始，将后面的元素一一向前移动，在移动完成后，线性表的长度减 1

（四）栈和队列

1．栈及其基本运算

1）栈

栈是一种特殊的线性表，它是限定在一端进行插入和删除的线性表。它的插入和删除只能在表的一端进行，而另一端是封闭的，不允许进行插入和删除操作。

在栈中，允许插入和删除操作一端称为栈顶，不允许插入和删除操作的一端则称为栈底。栈顶的元素总是最后被插入的元素，也是最先被删除的元素。它遵循的原则是：先进后出或后进先出。

堆栈指针总是指向栈顶元素的。

2）栈的顺序存储及其运算

在栈的顺序存储空间 S（1：m）中，S（bottom）通常为栈底元素，S（top）为栈顶元素。Top=0 表示栈空；top=m 表示栈满。

1）入栈运算

即在栈的顶部插入一个新元素。操作方式是：将栈顶指针加 1，再将元素插入至指针所

指的位置。

2）退栈运算

退栈运算即将栈顶元素取出并赋给一个指定的变量。操作方式是：先将栈顶元素赋给指定的变量，再将栈顶指针减 1。

3）读栈顶元素

将栈顶元素赋给某一指定变量，但栈顶指针不变。

2．队列及其基本运算

1）队列

队列即是允许在一端进行插入，而在另一端进行删除的线性表。允许插入的一端称为队尾，通常用一个尾指针指向队尾；允许删除的一端称为队首，通常用一个队首指针指向排队元素的前一个位置。

队列遵循的规则是：先进先出或后进后出

2）循环队列及其运算

队列的顺序存储结构一般采用循环队列的形式。

循环队列，即是次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，从排头指针 front 指向的后一个位置到队尾指针 rear 指向的位置之间所有的元素均为队列中的元素。

循环队列的初始状态为空，即 rear=front=m。这里 m 即为队列的存储空间。

循环队列的基本运算：入队运算和退队运算。

入队运算：每进行一次入队运算，队尾指针加 1。当队尾指针 rear=m+1 时，即表示队列空间的尾部已经放置了元素，则下一个元素应该旋转到队列空间的首部，即 rear=1

退队运算：每退队一个元素，排头指针加 1。当排头指针 front=m+1 时，即排头指针指向队列空间的尾部，退队后，排头指针指向队列空间的开始，即 front=1。

在队列操作时，循环队列满时，front=rear，队列空时，也有 rear=front，即在队列空或满时，排头指针和队尾指针均指向同一个位置。要判断队列空或满时，还应增加一个标志，s 值的定义：

$$s=\begin{cases}0 & \text{表示队列空}\\ 1 & \text{表示队列满}\end{cases}$$

判断队列空与队列满的条件下：

队列空的条件：s=0

队列满的条件：s=1、front=rear

（1）入队运算

即在队尾加入一个新元素。这个运算有两个基本操作：首先，将队尾指针加 1，即 rear=rear+1，当 rear=m+1 时，置 rear=1，然后，将新元素插入到队尾指针指向的位置。

当循环队列非空（s=1），且 front=rear 时，队列满，不能进行入队操作。此情况称“上溢”。

（2）退队操作

即将队首的元素赋给一个指定的变量。该运算也有两个基本操作：首先，将排头指针加 1，即 front=front+1，当 front=m+1 时，置 front=1，然后，将排头指针指向的元素赋给指

定的变量。

当循环队列为空（s=0）时，不能进行退队运算。此种情况称为“下溢”。

（五）线性链表

1．基本概念

前面的线性表均是采用顺序存储结构及在顺序存储结构下的运算。

1）顺序存储的优点：

- 结构简单
- 运算方便

2）顺序存储结构的缺点：

- 要在顺序存储的线性表中插入一个新元素或删除一个元素时，为了保证插入或删除后的线性表仍然为顺序存储。在插入或删除元素时，需要移动大量的数据元素，因此运算效率较低。
- 如果一个线性表分配顺序存储空间后，如果出现线性表的存储空间已满，但还需要插入元素时，会发生“上溢”错误。
- 在实际应用时，可能有多个线性表同时使用存储空间，这样给存储空间的分配带来问题，有可能使有的队列空间不够或过多造成浪费。

基于上述情况，对于大的线性表或元素变动频繁的大线性表不宜采用顺序存储结构，而应采用链式存储结构。

3）链式存储结构

假设每一个数据结点对应一个存储单元，该存储单元称为存储结点，简称结点。

在链式存储方式中，要求每一个结点由两部分组成：一部分用于存放数据元素，你为数据域；另一部分用于存放指针，称为指针域。该指针用于指向该结点的前一个或后一个结点。

在链式存储结构中，存储数据结构的存储空间可以不连续，各数据结点的存储顺序与数据元素之间的逻辑关系不一致，而数据元素之间的逻辑关系是由指针域来确定的。

链式存储结构既可以用于线性结构，也可用于非线性结构。

4）线性链表

线性表的链式存储结构称为线性链表。

将存储空间划分成若干的小块，每块占用若干个字节，这些小块称为存储结点。

将存储结点分为两个部分，一部分用于存储数据元素的值，称为数据域；另一部分用于存储元素之间的前后件关系，即存放下一个元素在存储序号（即存储地址），即指向后件结点，称为指针域。

在线性链表中用一个专门的指针 HEAD 指向线性链表中第一个数据元素的结点（即存放第一个元素的地址）。线性表中最后一个元素没有后件，因此，线性链表中的最后一个结点的指针域为空（用 Null 或 0 表示），表示链终结。

在线性链表中，各元素的存储序号是不连续的，元素间的前后件关系与位置关系也是不一致的。在线性链表中，前后件的关系依靠各结点的指针来指示，指向表的第一个元素的指针 HEAD 称为头指针，当 HEAD=NULL 时，表示该链表为空。

对于线性链表，可以从头指针开始，沿着各结点的指针扫描到链表中的所有结点。

这种线性链表称为线性单链表，即可以从表头开始向后扫描链表中的所有结点，而不能从中间或表尾结点向前扫描位于该结点之前的元素。

这种链表结构的缺点是不能任意地对链表中的元素按不同的方向进行扫描。在某些应用时，如果对链表中的元素设置两个指针域，一个为指向前件的指针域，称为左指针（LLink），一个为指向后件的指针域，称为右指针（RLink）。则这种链表是双向链表。

5）带链的栈

带链的栈即是用来收集计算机存储空间中的所有空闲的存储结点，这种带链的栈称为可利用栈。

当需要存储结点时，即从可利用的栈的顶部取出栈顶结点；当系统要释放一个存储结点时，将该结点空间放回到可利用栈的栈顶。

即在计算机中所有空闲的空间，均可以以结点的方式链接到可利用栈中，随着其他线性链表中结点的插入与删除，可利用栈处于动态变化之中，即可利用栈经常要进行退栈和入栈操作。

6）带链的队列

队列也是线性表，也可利用链式存储结构来进行保存。

2．线性链表的基本运算

线性链表包括的基本运算：

- 在链表中包含指定元素的结点之前插入一个新元素
- 在链表中删除包含指定元素的结点
- 将两个线性链表按要求合并成一个线性链表
- 将一个线性链表按要求进行分解
- 逆转线性链表
- 复制线性链表
- 线性链表的排序
- 线性链表的查找

1）线性链表中查找指定的元素

在线性链表中查找元素 X：从头指针指向的结点开始往后沿指针进行扫描，直到后面已没有结点或下一个结点的数据域为 X 为止。

元素的查找，经常是为了进行插入或删除操作而进行的，因此，在查找时，往往是需要记录下该结点的前一个结点。

2）线性链表的插入

线性链表的插入即在链式存储结构的线性表中插入一个新元素。

在线性链表中包含元素 x 的结点之前插入新元素 b，插入过程：

（1）从可利用栈中取得一个结点，设该结点号为 p，即取得的结点的存储序号存放在变量 p 中。并置结点 p 的数据域为插入的元素值 b。

（2）在线性链表中寻找包含元素 x 的前一个结点，该结点的存储序号为 q。

（3）将结点 p 插入到结点 q 之后。具体的操作：首先，使结点 p 插入到结点 q 之后（即结点 q 的后件结点），然后，使结点 q 的指针域 内容改为指向结点 p。

线性链表的插入操作，新结点是为来自于可利用栈，因此不会造成线性表的溢出。同样，

由于可利用栈可被多个线性表利用，因此，不会造成存储空间的浪费，大家动态地共同使用存储空间。

3）线性链表的删除

线性链表的删除，即是在链式存储结构下的线性表中删除指定元素的结点。

操作方式：

（1）在线性表中找到包含指定元素 x 的前一个结点 p

（2）将该结点 p 后的包含元素 x 的结点从线性链表中删除，然后将被删除结点的后一个结点 q 的地址提供给结点 p 的指针域，即将结点 p 指向结点 q。

（3）将删除的结点送回可利用栈。

从以上的删除操作可见，删除一个指定的元素，不需要移动其他的元素即可实现，这是顺序存储的线性表所不能实现的。同时，此操作还可更有效地利用计算机的存储空间。

3．循环链表及其基本操作

在线性链表中，虽然对数据元素的插入和删除操作比较简单，但由于它对第一个结点和空表需要单独处理，使得空表与非空表的处理不一致。

循环链表，即是采用另一种链接方式，它的特点如下：

（1）在循环链表中增加一个表头结点，其数据域为任意或根据需要来设置，指针域指向线性表的第一个元素的结点。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的，而是指向表头结点。在循环链表中，所有结点的指针构成一个环状链。

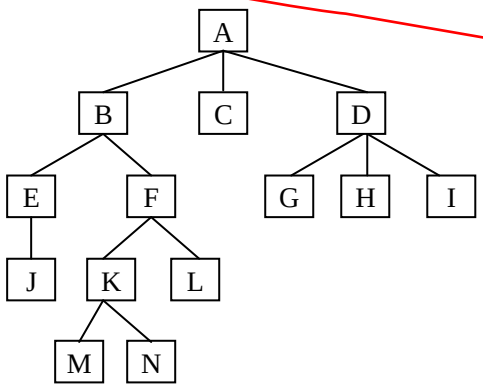
在循环链表中，只要指出表中任何一个结点的位置，均可以从它开始扫描到所有的结点，而线性链表做不到，线性链表是一种单向的链表，只能按照指针的方向进行扫描。

循环链表中设置了一个表头结点，因此，在任何时候都至少有一个结点，因此空表与非空表的运算相统一。

（六）树与二叉树

1．树的基本概念

树是一种简单的非线性结构。在树结构中，数据元素之间有着明显的层次结构。在树的图形表示中，用直线连接两端的结点，上端点为前件，下端点为后件。



在树结构中，每一个结点只有一个前件，称为父结点。如 A 即为结点 B、C、D 的父结点。

没有父结点的结点只有一个，称为根结点。如上图所示，结点 A 即为根结点。

每一个结点可以有多个后件，它们均称为该结点的子结点。如结点 G、H、I 是结点 D

的子结点。

没有后件的结点，称为叶子结点。上图中，叶子结点有：J、M、N、L、C、G、H、I。

在树结构中，一个结点所拥有的后件结点个数称为该结点的度。例如，结点 D 的度为 3，结点 E 的度为 1 等，按此原则，所有叶子结点的度均为 0。

在树中，所有结点中最大的度称为该树的度。上图所示的树中，所有结点中最大的度是 3，所以该树的度为 3。

树分层，根结点为第一层，往下依次类推。同一层结点的所有子结点均在下一层。如上图：A 结点在第 1 层，B、C、D 结点在第 2 层；E、F、G、H、I 在第 3 层；J、K、L 在第 4 层；M、N 在第 5 层。

树的最大层次称为树的深度。上图树的深度为 5。

在树中，某结点的一个子结点为根构成的树称作该结点的子树。叶子结点没有子树。

在计算机中，可以用树来表示算术表达式。原则如下：

（1）表达式中每一个运算符在树中对应一个结点，称为运算符结点

（2）运算符的每一个运算对象在树中为该运算符结点的子树（在树中的顺序为从左到右）

（3）运算对象中的单变量均为叶子结点

树在计算机中用多重链表表示。多重链表中的每个结点描述了树中对应结点的信息，而每个结点中的链域（即指针域）个数将随着树中该结点的度而定义。

如果在树中，每一个结点的子结点的个数不相同，因此在多重链中各结点的链域个数也不相同，会导致算法太复杂。因此，在树中，常采用定长结点来表示树中的每一个结点，即取树的度作为每个结点的链域的个数。这样，管理相对简化了，但会造成空间的浪费，因为有许多的结点存在空链域。

2．二叉树及其基本性质

1）二叉树的定义

二叉树的特点：

- 非空二叉树只有一个根结点
- 每一个结点最多只有两个子结点，且结点分左右。则一个结点最多可以有两棵子树，分别称为左子树和右子树

在二叉树中，每一个结点的度最大为 2，即二叉树的度为 2。在二叉树中，任何的子树也均为二叉树。

在二叉树中，每一个结点的子树被分为左子树和右子树。在二叉树中，允许某一个结点只有左子树或只有右子树。如果一个结点既没有左子树，也没有右子树，则该结点为叶子结点。

2）二叉树的性质

性质 1：在二叉树的第 k 层上，最多有 2^{k-1}（k≥1）个结点。

性质 2：深度为 m 的二叉树最多有 2^m-1 个结点。

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

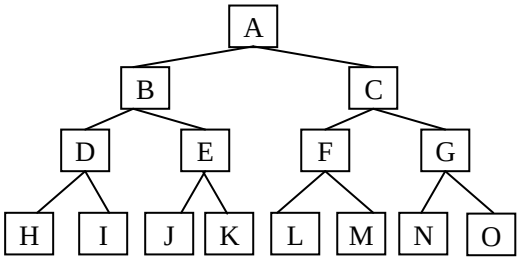
性质 4：具有 n 个结点的二叉树，其深度至少为$[\log_2 n]+1$，其中$[\log_2 n]$表示 $\log_2 n$ 的整数部分。

3）满二叉树与完全二叉树

（1）满二叉树

满二叉树的特点：

除最后一层外，每一层上的所有结点都有两个子结点。即在满二叉树中，每一层上的结点数都达到最大值，即在满二叉树上的第 k 层上有 2^{k-1} 个结点。如下即为一棵满二叉树。



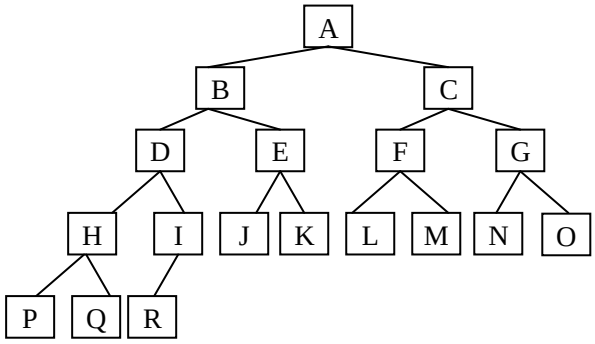
满二叉树

（2）完全二叉树

特点：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个结点。

即如果从根结点开始，对二叉树的结点自上而下、自左而右用自然数进行连续编号，则深度为 m、且有 n 个结点的二叉树，当且仅当其每一个结点都与深度为 m 的满二叉树中编号从 1 到 n 的结点一一对应，则是完全二叉树。

对于完全二叉树，叶子结点只能在层次最大的两层中出现；对于任何一个结点，若其右



完全二叉树

分支下的子树结点的最大层次为 p，则其分支下的子孙结点的最大层次为 p 或 p+1。

完全二叉树具有的性质：

性质 5：具有 n 个结点的完全二叉树的深度为$[\log_2 n]+1$

性质 6：设完全二叉树共有 n 个结点。如果从根结点开始，按层次（每一层从左到右）用自然数 1、2……、n 给结点编号，对于编号为 k（k=1,2,……n）的结点有如下结论：

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

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

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

3．二叉树的存储结构

二叉树的存储常采用链式存储结构。

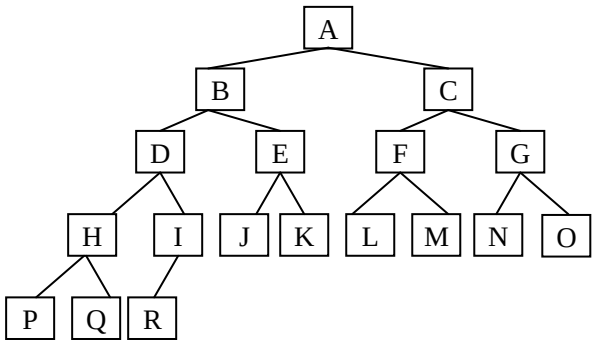
存储二叉树中各元素的存储结点由两个部分组成：数据域和指针域。在二叉树中，由于每个结点可有两个子结点，则它的指针域有两个：一个用于存储该结点的左子结点的存储地址，即称为左指针域；一个用于存储指向该结点的右子结点的存储地址，称为右指针域。

存储结构如下：

	Lchild	Value	Rchild
i	L(i)	V(i)	R(i)

即二叉树的存储结构中每一个存储结点都有两个指针域，因此，二叉树的链式存储结构也称为二叉树的链表。在二叉树在存储中，用一个头指针指向二叉树的根结点的存储地址。

如图所示的二叉树：



如果我们将该二叉树的所有结点顺序编号，顺序存放在存储空间里，则它们在内存空间中的存放方式是：

	i	L(i)	V(i)	R(i)
BT→	1	2	A	3
	2	4	B	5
	3	6	C	7
	4	8	D	9
	5	10	E	11
	6	12	F	13
	7	14	G	15
	8	16	H	17
	9	18	I	0
	10	0	J	0
	11	0	K	0
	12	0	L	0
	13	0	M	0
	14	0	N	0
	15	0	O	0
	16	0	P	0
	17	0	Q	0
	18	0	R	0

当然，对于满二叉树或完全二叉树而言，也可采用顺序存储的方式，但顺序存储的方式不适合其他的二叉树。

4．二叉树的遍历

二叉树的遍历即是不重复地访问二叉树的所有结点。

在遍历二叉树时，一般先遍历左子树，然后再遍历右子树。在先左后右的原则下，二叉树的遍历又可分为三种：前序遍历、中序遍历和后序遍历。

1）前序遍历

前序遍历即先访问根结点，然后遍历左子树，最后遍历右子树。在遍历左子树和遍历右子树时，依然是先遍历根结点，然后是左子树，再是右子树。

操作的具体方式：

- 若二叉树为空，则结束返回。
- 否则：访问根结点→前序遍历左子树→前序遍历右子树

如上图所示的完全二叉树，它的前序遍历结果是：A、B、D、H、P、Q、I、R、E、J、K、C、F、L、M、G、N、O

2）中序遍历

中序遍历，即先遍历左子树，然后访问根结点，最后是遍历右子树。

具体的操作方式：

- 若二叉树为空，则结束返回。
- 否则：中序遍历左子树→访问根结点 →中序遍历右子树

这里强调，在遍历左子树和右子树时，仍然要采用中序遍历的方法。

如上图所示的完全二叉树，它的中序遍历结果是：P、H、Q、D、R、I、B、J、E、K、A、L、F、M、C、N、G、O

3）后序遍历

后序遍历，即选遍历左子树，然后是遍历右子树，最后访问根结点。

具体的操作方式：

- 若二叉树为空，则结束返回。
- 否则：前序遍历左子树→前序遍历右子树→访问根结点

如上图所示的完全二叉树，它的后序遍历结果是：P、Q、H、R、I、D、J、K、E、B、L、M、F、N、O、G、C、A

（七）查找技术

查找即是指在一个给定的数据结构中查找某个指定的元素。

1．顺序查找

顺序查找又称顺序搜索。一般是在线性表中查找指定的元素。

基本操作方法是：

从线性表的第一个元素开始，与被查元素进行比较，相等则查找成功，否则继续向后查找。如果所有的元素均查找完毕后都不相等，则该元素在指定的线性表中不存在。

顺序查找的最好情况：要查找的元素在线性表的第一个元素，则查找效率最高；如果要查找的元素在线性表的最后或根本不存在，则查找需要搜索所有的线性表元素，这种情况是最差情况。

对于线性表而言，顺序查找效率很低。但对于以下的线性表，也只能采用顺序查找的方法：

- 线性表为无序表，即表中的元素没有排列不是按大小顺序进行排列的，这类线性表不管它的存储方式是顺序存储还是链式存储，都只能按顺序查找方式进行查找
- 即使是有序线性表，如果采用链式存储，也只能采用顺序查找方式

例如，现有线性表：7、2、1、5、9、4，要在序列中查找元素 6，查找的过程是：

- 整个线性表的长度为 5
- 查找计次 n=1，将元素 6 与序列的第一个 7 元素进行比较，不等，继续查找
- n=2，将 6 与第二个元素 2 进行比较，不等，继续
- n=3，将 6 与第三个元素 1 进行比较，不等，继续
- n=4，将 6 与第四个元素 5 进行比较，不等，继续
- n=5，将 6 与第五个元素 9 进行比较，不等，继续
- n=6，将 6 与第六个元素 4 进行比较，不等，继续
- n=7，超出线性表的长度，查找结束，则该表中不存在要查找的元素。

2．二分查找

二分查找只适用于顺序存储的有序表。此处所述的有序表是指线性中的元素按值非递减排列（即由小到大，但允许相邻元素值相等）。

二分查找的方法如下：

将要查找的元素与有序序列的中间元素进行比较：

- 如果该元素比中间元素大，则继续在线性表的后半部分（中间项以后的部分）进行查找
- 如果要查找的元素的值比中间元素的值小，则继续在线性表的前半部分（中间项以前的部分）进行查找

这个查找过程一直按相同的顺序进行下去，一直到查找成功或子表长度为 0（说明线性表中没有要查找的元素）

有序线性表的二分法查找，条件是必须这个有序线性表的存储方式是顺序存储的。它的查找效率比顺序查找要高得多，它的最坏情况的查找次数是 $\log_2 n$ 次，而顺序查找的最坏情况的查找次数是 n 次。

当然，二分查找的方法也支持顺序存储的递减序列的线性表。

有非递减有序线性表：1、2、4、5、7、9，要查找元素 6。查找的方法是：

- 序列长度为 n=6，中间元素的序号 m=[(n+1)/2]=3
- 查找计次 k=1，将元素 6 与中间元素即元素 4 进行比较，不等，6>4
- 查找计次 k=2，查找继续在后半部分进行，后半部分子表的长度为 3，计算中间元素的序号：m=3+[(3+1)/2]=5，将元素与后半部分的中间项进行比较，即第 5 个元素中的 7 进行比较，不等，6<7
- 查找计次 k=3，继续查找在后半部分序列的前半部分子序列中查找，子表长度为 1，则中间项序号即为 m=3+[(1+1)/2]=4，即与第 4 个元素 5 进行比较，不相等，继续查找的子表长度为 0，则查找结束

（八）排序技术

排序即是将一个无序的序列整理成按值非递减顺序排列的有序序列。在这里，我们讨论的是顺序存储的线性表的排序操作。

1．交换类排序法

交换类排序法，即是借助于数据元素之间的互相交换进行排序的方法。

1）冒泡排序法

冒泡排序法即是利用相邻数据元素之间的交换逐步将线性表变成有序序列的操作方法。

操作过程如下：

- 从表头开始扫描线性表，在扫描过程中逐次比较相邻两个元素的大小，若相邻两个元素中前一个元素的值比后一个元素的值大，将两个元素位置进行交换，当扫描完成一遍时，则序列中最大的元素被放置到序列的最后。
- 再继续对序列从头进行扫描，这一次扫描的长度是序列长度减 1，因为最大的元素已经就位了，采用与前相同的方法，两两之间进行比较，将次大数移到子序列的末尾。
- 按相同的方法继续扫描，每次扫描的子序列的长度均比上一次减 1，直至子序列的长度为 1 时，排序结束。

例如，有序列 5、2、9、4、1、7、6，将该序列从小到大进行排列。

采用冒泡排序法，具体操作步骤如下：

序列长度 n=7

原序列	5	2	9	4	1	7	6
第一遍（从前往后）	5←→	2	9	4	1	7	6
	2	5	9←→	4	1	7	6
	2	5	4	9←→	1	7	6
	2	3	4	1	9←→	7	6
	2	5	4	1	7	9←→	6
第一遍结束后	2	5	4	1	7	6	9
第二遍（从前往后）	2	5←→	4	1	7	6	9
	2	4	5←→	1	7	6	9
	2	4	1	5	7←→	6	9
	2	4	1	5	6	7	9
第二遍结束后	2	4	1	5	6	7	9
第三遍（从前往后）	2	4←→	1	5	6	7	9
	2	1	4	5	6	7	9
第三遍结束	2	1	4	5	6	7	9
第四遍（从前往后）	2←→	1	4	5	6	7	9
	1	2	4	5	6	7	9
第四遍结束	1	2	4	5	6	7	9
最后结果	1	2	4	5	6	7	9

扫描的次数，最多需要扫描 n-1 次，如果序列已经就位，则扫描结束。测试是否已经就位，可设置一个标志，如果该次扫描没有数据交换，则说明数据排序结束。

2）快速排序法

冒泡排序方法每次交换只能改变相邻两个元素之间的逆序，速度相对较慢。如果将两个不相邻的元素之间进行交换，可以消除多个逆序。

快速排序的方法是：

从线性表中选取一个元素，设为 T，将线性表后面小于 T 的元素移到前面，而前面大于 T 的元素移到后面，结果将线性表分成两个部分（称为两个子表），T 插入到其分界线的位置处，这个过程称为线性表的分割。对过对线性表的一次分割，就以 T 为分界线，将线性表分成前后两个子表，且前面子表中的所有元素均不大于 T，而后面的所有元素均不小于 T。

再将前后两个子表再进行相同的快速排序，将子表再进行分割，直到所有的子表均为空，则完成快速排序操作。

在快速排序过程中，随着对各子表不断的进行分割，划分出的子表会越来越多，但一次

又只能对一个子表进行分割处理，需要将暂时不用的子表记忆起来，这里可用栈来实现。

对某个子表进行分割后，可以将分割出的后一个子表的第一个元素与最后一个元素的位置压入栈中，而继续对前一个子表进行再分割；当分割出的子表为空时，可以从栈中退出一个子表进行分割。

这个过程直到栈为空为止，说明所有子表为空，没有子表再需分割，排序就完成。

2．插入类排序法

1）简单插入排序

插入排序，是指将无序序列中的各元素依次插入到已经有序的线性表中。

插入排序操作的思路：在线性表中，只包含第 1 个元素的子表，作为该有序表。从线性表的第 2 个元素开始直到最后一个元素，逐次将其中的每一个元素插入到前面的有序的子表中。

该方法与冒泡排序方法的效率相同，最坏的情况下需要 n(n-1)/2 次比较。

例如，有序列 5、2、9、4、1、7、6，将该序列从小到大进行排列。

采用简单插入排序法，具体操作步骤如下：

序列长度 n=7

5	2	9	4	1	7	6
	↑j=2					
2	5	9	4	1	7	6
		↑j=3				
2	5	9	4	1	7	6
			↑j=4			
2	4	5	9	1	7	6
				↑j=5		
1	2	4	5	9	7	6
					↑j=6	
1	2	4	5	7	9	6
						↑j=7
插入排序后的结果	1	2	4	5	6	7	9

2）希尔排序法

希尔排序法的基本思想：

将整个无序序列分割成若干小的子序列分别进行插入排序。

子序列的分割方法：将相隔某个增量 h 的元素构成一个子序列，在排序的过程中，逐次减小这个增量，最后当 h 减小到 1 时，再进行一次插入排序操作，即完成排序。

增量序列一般取 $h_t=n/2^k(k=1,2,\cdots,[\log_2 n])$，其中 n 为待排序序列的长度。

3．选择类排序法

1）简单选择排序法

基本思路：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，然后对后面的子表采用相同的方法，直到子表为空为止。

对于长度为 n 的序列，需要扫描 n-1 次，每一次扫描均找出剩余的子表中最小的元素，然后将该最小元素与子表的第一个元素进行交换。

例如，有序列 5、2、9、4、1、7、6，将该序列从小到大进行排列。

采用简单选择排序法，具体操作步骤如下：

原序列	5	2	9	4	1	7	6
第一遍扫描	1	2	9	4	5	7	6
第二遍扫描	1	2	9	4	5	7	6
第三遍扫描	1	2	4	9	5	7	6
第四遍扫描	1	2	4	5	9	7	6
第五遍扫描	1	2	4	5	6	7	9
第六遍扫描	1	2	4	5	6	7	9
排序结果	1	2	4	5	6	7	9

2）堆排序法

堆排序法属于选择类排序方法。

堆的定义：具有 n 个元素的序列（$h_1, h_2, \cdots, h_n$），当且仅当满足

$\begin{cases} h_i \geq h_{2i} \\ h_i \geq h_{2i+1} \end{cases}$ 或 $\begin{cases} h_i \leq h_{2i} \\ h_i \leq h_{2i+1} \end{cases}$（I=1, 2, …, n/2）时称之为堆。

本节只讨论满足前者条件的堆。

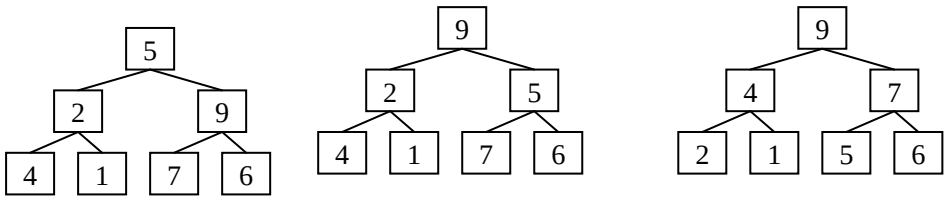
由堆的定义看，堆顶元素（即第一个元素）必为最大项。

可以用一维数组或完全二叉树来表示堆的结构。

用完全二叉树表示堆时，树中所有非叶子结点值均不小于其左右子树的根结点的值，因此堆顶（完全二叉树的根结点）元素必须为序列的 n 个元素中的最大项。

例如，有序列 5、2、9、4、1、7、6，将该序列从小到大进行排列。

利用堆排序法将该序列进行排序。



无序堆　　调整根结点　　调整子树的根结点

建堆的过程

操作方式即：先将无序堆的根结点 5 与左右子树的根结点 2、9 进行比较，5<9，将 5 与 9 进行交换；整后，对左右子树进行堆调整，左子树的根结点 2 小于其左叶子结点 5，调整；右子树的根结点 5 小于其左右子结点 7 和 6，根据堆的要求，将 5 与 7 进行调整。

根据堆的定义，可以得到堆排序的方法：

（1）首先将一个无序序列建成堆

（2）然后将堆顶元素（序列中的最大项）与堆中最后一个元素交换（最大项应该在序列的最后）。

三、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

一、内容要点

（一）程序设计方法与风格

程序设计方法：主要经过了面向过程的结构化程序设计和面向对象的程序设计方法。

程序设计风格，是指编写程序时所表现出来的特点、习惯和逻辑思路。通常，要求程序设计的风格应强调简单和清晰，必须是可以读的，可以理解的。

要形成良好的程序设计的风格，应考虑如下因素：

1．源程序文档化

（1）符号名的命名：符号名的命名要具有一定的实际含义，便于对程序的理解，即通常说的见名思义；

（2）程序注释：正确的程序注释能够帮助他人理解程序。注释一般包括序言性注释和功能性注释；

（3）视觉组织：为了使程序一目了然，可以对程序的格式进行设置，适当地通过空格、空行、缩进等使程序层次清晰。

2．数据说明方法

（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的数据结构。

3．语句的结构

（1）在一行内只写一条语句；
（2）程序的编写应该优先考虑清晰性；
（3）除非对效率有特殊的要求，否则，应做到清晰第一，效率第二；
（4）首先保证程序的正确，然后再要求速度；
（5）避免使用临时变量使程序的可读性下降；
（7）尽量使用库函数，即尽量使用系统提供的资源；
（8）避免采用复杂的条件语句；
（9）尽量减少使用“否定”条件的条件语句；
（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）对所有的输入输出数据都要检验数据的合法性；
（2）检查输入项的各种重要组合的合理性；
（3）输入格式要简单，以使得输入的步骤和操作尽可能简单；

（4）输入数据时，应允许自由格式；

（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）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。

（二）结构化程序设计

1．结构化程序设计的原则

结构化程序设计方法的主要原则：自顶而下、逐步求精，模块化，限制使用 goto 语句。

1）自顶而下

程序设计时，应先考虑总体，后考虑细节；先考虑全局，后考虑局部目标。即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为过渡，逐步细化。

3）模块化

一个复杂问题，都是由若干个稍简单的问题构成的。模块化即是将复杂问题进行分解，即将解决问题的总目标分解成若干个分目标，再进一步分解为具体的小目标，把每一个小目标称作一个模块。

4）限制使用 goto 语句

goto 语句可以提高效率，但对程序的可读性、维护性都造成影响，因此应尽量不用 goto 语句。

2．结构化程序设计的基本结构与特点

结构化程序设计是程序设计的先进方法和工具，采用结构化程序设计可以使程序结构良好、易读、易理解、易维护。

1）顺序结构

顺序结构即是顺序执行的结构，是按照程序语句行的自然顺序，一条一条语句地执行程序。

2）选择结构

选择结构又称分支结构，它包括简单选择和多分支选择结构。程序的执行是根据给定的条件，选择相应的分支来执行。

3）重复结构

重复结构又称循环结构，根据给定的条件，决定是否重复执行某一相同的或类似的程序段。利用重复结构可以大量简化程序行。

3．结构化程序设计原则和方法的应用

1．使用程序设计语言中的顺序、选择、循环等有限的控制结构表示程序的控制逻辑；
2．选用的控制结构只允许有一个入口和一个出口；
3．程序语句组成容易识别的块，每块只有一个入口和一个出口；
4．复杂结构应该用嵌套的基本控制结构进行组合嵌套来实现；
5．语言中所有没有的控制结构，应该采用前后一致的方法来模拟；
6．严格控制 goto 语句的使用：
（1）用一个非结构化的程序设计语言去实现一个结构化的构造；
（2）若不使用 goto 语句会使功能模糊；
（3）在某种可以改善而不是损害程序可读性的情况下。

（三）面向对象的程序设计

1．关于面向对象方法

面向对象方法的本质，是主张从客观世界固有的事物出发来构造系统，提倡用人类在现实生活中常用的思维方法来认识、理解和描述客观事物，强调最终建立的系统能够反映问题域，即系统中的对象以及对象之间的关系能够如实地反映问题域中固有事物及其关系。

面向对象的优点：

1）与人类习惯的思维方法一致

传统的程序设计方法是以算法作为核心，将程序与过程相互独立。

面向对象方法和技术是以对象为核心，对象是由数据和容许的操作组成的封装体，与客观实体有直接的对应关系。对象之间通过传递消息互相联系，以实现模拟世界中不同事物之间的联系。

2）稳定性好

面向对象方法基于构造问题领域的对象模型，以对象为中心构造软件系统。它的基本方法是用对象模拟问题领域中的实体，以对象间的联系刻画实体间的联系。

3）可重用性好

软件的重用性是指在不同的软件开发过程中重复使用相同或相似的软件元素的过程。

4）易于开发大型软件产品

在使用面向对象进行软件开发时，可以把大型产品看作是一系列本质上相互独立的小产品来处理，降低了技术难度，也使软件开发的管理变得容易。

5）可维护性好

（1）利用面向对象的方法开发的软件稳定性比较好
（2）用面向对象的方法开发的软件比较容易修改
（3）用面向对象的方法开发的软件比较容易理解
（4）易于测试和调试

2．面向对象方法的基本概念

1）对象

在面向对象程序设计方法中，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，它由一组表示其静态特征的属性和它执行的一组操作组成。

对象的基本特点：

（1）标识的唯一性

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和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征，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。

（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，对象内容各种元素彼此相结合得很紧密，内聚性强。

2）类和实例

将属性、操作相似的对象归为类。具有共同的属性、共同的方法的对象的集合，即是类。

类是对象的抽象，它描述了属于该对象的所有对象性质，而一个对象则是其对应类的一个实例。

3）消息

消息是一个实例与另一个实例之间传递的信息，它请求对象执行某一处理或回答某一个要求的信息，它统一了数据流和控制流。

消息只包含传递者的要求，它告诉接受者需要做哪些处理，并不指示接受者怎样去完成这些处理。

4）继承

继承是使用已有的类定义作为基础建立新类的定义技术。已有的类可当作基类来引用，则新类相应地可作为派生类来引用。

继承即是指能够直接获得已有的性质和特征，而不必重复定义它们。

5）多态性

对象根据所接受的消息而做出动作，同样的消息被不同的对象接受时可导致完全不同的行动，该现象称为多态性。

在面向对象技术中，多态性是指子类对象可以像父类对象那样使用，同样的消息可以发送给父类对象也可以发送给子类对象。

多态性机制增加了面向对象软件系统的灵活性，减少了信息冗余，而且显著提高了软件的可重用性可扩充性。

三、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

一、学习目标与要求

1．了解软件工程的基本概念；
2．了解软件工程过程与软件的生命周期，以及软件工程的目标和原则；
3．了解利用结构化分析法进行软件工程中的需求分析的方法，并了解需求分析的方法和需要完成的任务；
4．了解数据流图的使用方法；
5．了解如何利用结构化设计方法进行软件设计，并了解软件设计的一些常用用工具；
6．了解软件测试的目的和方法，以及软件测试的准则，了解常用的软件测试方法的区别和各自的功能与特点；
7．了解程序调试的方法和原则。

二、内容要点

（一）软件工程基本概念

1．软件定义与软件特点

1）软件的定义

与计算机系统的操作有关的计算机程序、规程、规则，以及可能有的文件、文档及数据。

2）软件的特点

（1）软件是一种逻辑实体，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；但为了适应硬件、环境以及需求的变化要进行修改，会导致一些错误的引入，导致软件失效率升高，从而使得软件退化；
（4）软件的开发、运行对计算机系统具有依赖性，受到计算机系统的限制，这导致了软件移植的问题；
（5）软件复杂性高，成本昂贵。软件开发需要投入大量、高强度的脑力劳动，成本高，风险大；
（6）软件开发涉及诸多的社会因素。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。

3）软件的分类

按功能分，可分为：

- 应用软件：为解决特定领域的应用而开发的软件

- 系统软件：是计算机管理自身资源，提高计算机使用效率并为计算机用户提供各种服务的软件
- 支撑软件（或工具软件）：介于系统软件和应用软件之间，协助用户开发软件的工具性软件，包括辅助和支持开发和维护应用软件的工具软件

2．软件危机与软件工程

1）软件危机

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。它主要表现在：
（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。开发的成本超预算和开发周期的超期经常出现；
（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）软件成本不断提高；
（6）软件开发生产率的提高赶不上硬件的发展和应用需求的增长。

2）软件工程

软件工程的定义：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。

软件工程包括 3 个要素：方法、工具和过程。

方法：完成软件工程项目的技术手段；

工具：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。

3．软件工程过程与软件生命周期

1）软件工程过程

软件工程过程把输入转化为输出的一组彼此相关的资源和活动。支持软件工程过程的两方面内涵：

（1）软件工程过程是指为获得软件产品，在软件工具支持下由软件工程师完成的一系列软件工程活动。它包括 4 种基本活动：

- P—软件规格说明。规定软件的功能及其运行时的限制；
- D—软件开发。产生满足规格说明的软件；
- C—软件确认。确认软件能够满足客户提出的要求；
- A—软件演进过程。为满足客户的变更要求，软件必须在使用的过程中演进。

（2）使用适当的资源（包括人员、硬软件工具、时间等），为开发软件进行的一组开发活动，在过程结束时将输入（用户要求）转化为输出（软件产品）。

软件工程过程是将软件工程的方法和工具综合起来，以达到合理、及时地进行计算机软件开发的目的。

2）软件生命周期

将软件产品从提出、实现、使用维护到停止使用退役的过程称为软件生命周期。即软件的生命周期就是软件产品从开始考虑其概念开始，到软件产品不能使用为止的整个时期都属于软件生命周期。一般包括可行性研究与需求分析、设计、实现、测试、交付使用以及维护等活动。这些活动可以有重复，执行时也可以有迭代。

生命周期的主要阶段：

- 软件定义
- 软件开发
- 软件维护

软件生命周期的主要活动阶段是：

（1）可行性研究与计划制定：确定待开发软件系统的开发目标和总的要求，给出它的功能、性能、可靠性以及接口等方面的可能方案，制定完成开发任务的实话计划；

（2）需要分析。对待开发软件提出的需求进行分析并给出详细的定义；

（3）软件设计。系统设计人员和程序设计人员给出软件的结构、模块的划分、功能的分配以及处理流程；

（4）软件实现。把软件设计转换成计算机可以接受的程序代码。即完成源程序的编码，编写用户手册、操作手册等面向用户的文档，编写单元测试计划；

（5）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，编写测试分析报告；

（6）运行和维护。将已交付的软件投入运行，并在运行使用中不断地维护，根据新提出的需求进行必要且可能的扩充和删改。

4．软件工程的目标与原则

1）软件工程的目标

软件工程的目标：在给定成本、进度的情况下，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。

软件工程需要达到的基本目标：

- 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

软件工程的理论和技术性研究的内容包括：软件开发技术和软件工程管理。

（1）软件开发技术

软件开发方法学、开发过程、开发工具和软件工程环境，其主体内容是软件开发方法学。软件开发方法学是根据不同的软件类型，按不同的观点和原则，对软件开发中应遵循的策略、原则、步骤和必须产生的文档资料都做出规定，从而使软件开发能够进入规范化和工程化的阶段。

（2）软件工程管理

软件工程管理：软件管理学、软件工程经济学、软件心理学等内容。

软件工程管理学包括：人员组织、进度安排、质量保证、配置管理、项目计划等。

软件工程经济学：是研究软件开发中成本的估算、成本效益分析的方法和技术，用经济学的基本原理事研究软件工程开发中的经济效益问题。

软件心理学：从个体心理、人类行为、组织行为和企业文化等角度来研究软件管理和软件工程。

2）软件工程的原则

（1）抽象。抽取事物取基本的特征和行为，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藏起来，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。块太大会使模块内部过渡复杂，不利于对模块的理解和修改，也不利于模块的调试和重用；模块太小会使程序结构过于复杂，难于控制；

（4）局部化。在同一个物理模块中集中逻辑上相互关联的计算资源，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。所有的概念表达应是确定的、无歧义且规范。

（6）一致性。包括程序、数据和文档的整个软件系统的各模块应使用已知的概念、符号和术语；程序内外部接口保持一致，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份，完全实现系统所需要的功能；

（8）可验证性。开发大型软件系统需要对系统自顶向下，逐层分解。

5．软件开发工具与软件开发环境

1）软件开发工具

早期的软件开发，最早使用的是单一的程序设计语言，没有相应的开发工具，效率很低，随着软件开发工具的发展，提供了自动的或半自动的软件支撑环境，为软件开发提供了良好的环境。

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成起来，形成软件工程环境。

（二）结构化分析方法

1．需求分析与需求分析方法

1）需求分析

软件需求分析是指用户对目标软件系统在功能、行为、性能、设计约束等方面的期望。需求分析的任务是发现需求、求精、建模和定义需求的过程。

（1）定义

软件需求分析是指用户对目标软件系统在功能、行为、性能、设计约束等方面的期望。

（2）需求分析阶段的工作

① 需求获取。需求获取的目的是确定对目标系统的各方面需求；

② 需求分析。对获取的需求进行分析和综合，最终给出系统的解决方案和目标系统的逻辑模型；

③ 编写需求规格说明书。为用户、分析人员和设计人员之间进行交流提供方便。

④ 需求评审。对需求分析阶段的工作进行复审，验证需求文档的一致性、可靠性、完事性和有效性。

2）需求分析方法

（1）结构化分析方法

包括：

- 面向数据流的结构化分析方法
- 面向数据结构的 Jackson 方法
- 面向数据结构的结构化数据系统开发方法

（2）面向对象的分析方法

从需求分析建立模型的特性分，需求分析方法又分为静态分析方法和动态分析方法。

2．结构化分析方法

1）关于结构化分析方法

结构化分析方法的实质是：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。

结构化分析的步骤：

- 通过对用户的调查，以软件需求为线索，获得系统的具体模型；
- 去掉模型的非本质因素，抽象出系统的逻辑模型；
- 根据计算机的特点分析当前系统与目标系统的差别，建立目标系统的逻辑模型；
- 完善目标系统交补充细节，写出目标系统的软件需求规格说明；
- 评审直到确认完全符合用户对软件的需求。

2）结构化分析的常用工具

（1）数据流图

数据流图从数据传递和加工的角度，来刻画数据流从输入到输出的移动变换过程。

数据流图下的图形元素：

○（圆），加工（转换）。输入数据经过加工变换产生输出

→（箭头），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

═ （平行的二条直线），存储文件（数据源）。表示处理过程中存放各种数据的文件。

□（长方形），源，潭。表示系统和环境的接口，属于系统之外的实体。

（2）数据字典

数据字典是结构化分析方法的核心。对数据流图中出现的被命名的图形元素的确切解释。通常包括：名称、别名、何处使用/如何使用、内容描述、补充信息等。

（3）判定树

利用判定树，对数据结构中的数据之间的关系进行描述，弄清楚判定条件之间的从属关系、并列关系、选择关系。

（4）判定表

在数据流图中的加工要依赖于多个条件的取值，即完成该加工的一组动作是由于某一组条件取值的组合而引发的情况。它与判定树是相似的，但更适宜于较复杂的条件组合。

3．软件需求规格说明书

是需求分析阶段的最后成果，是软件开发的重要文档之一。

1）作用

- 便于用户、开发人员进行理解和交流
- 反映用户问题的结构，可以作为软件开发工作的基础和依据
- 作为确认测试和验收的依据

2）内容

在软件计划中确定的软件范围加以展开，制定出完整的信息描述、详细的功能说明、恰当的检验标准以及其他与要求有关的数据。

3）特点

软件需求规格说明书是确保软件质量的措施，它的内涵是：

- 正确性
- 无歧义性
- 完整性
- 可验证性
- 一致性
- 可理解性
- 可修改性
- 可追踪性

（三）结构化设计方法

1．软件设计的基本概念

1）软件设计的基础

软件设计包括软件结构设计、数据设计、接口设计、过程设计。其中，结构设计是定义软件系统各主要部件之间的关系；数据设计是将分析时创建的模型转化为数据结构的定义；接口设计是描述软件内部、软件和协作系统之间以及软件与人之间如何通信；过程设计是把系统结构部件转换成软件的过程性描述。

软件设计的一般过程：软件设计是一个迭代的过程；先进行高层次的结构设计；后进行低层次的过程设计；穿插进行数据设计和接口设计。

2）软件设计的基本原理

（1）抽象

抽象的层次从概要设计到详细设计逐渐降低。在软件概要设计中的模块分层也是由抽象到具体逐步分析和构造出来的。

（2）模块化

模块是指把一个待开发的软件分解成若干小的简单的部分。

模块化是指解决一个复杂问题时自顶向下逐层把软件系统划分成若干模块的过程。

（3）信息隐蔽

在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他模块来说是不能访问的。

（4）模块独立性

独立性是指每个模块只完成系统要求的独立的子功能，并且与其他模块的联系最少且接口简单。

衡量软件的模块独立性的标准：

- 内聚性：一个模块内部各个元素间彼此结合的紧密程度的度量
- 耦和性：模块间相互连接的紧密程序的度量

3）结构化设计方法

即将软件设计成相对独立、单一功能的模块组成结构。

2．概要设计

1）概要设计的任务

① 设计软件系统结构

即将系统划分成模块以及模块的层次结构。

② 数据结构及数据库设计

数据设计是实现需求定义和规格说明过程中提出的数据对象的逻辑表示。

数据设计的具体任务是：

- 确定输入、输出文件的详细数据结构
- 结合算法设计，确定算法所必须的逻辑数据结构及其操作
- 确定对逻辑数据结构所必须的那些操作的程序模块，限制和确定各个数据设计决策的影响范围
- 需要与操作系统或调度程序接口所必须的控制表进行数据交换时，确定其详细的数据结构和使用规则
- 数据的保护性设计：防卫性、一致性、冗余性设计

③ 编写概要设计文档

需要编写的文档有：

- 概要设计说明书
- 数据库设计说明书
- 集成测试计划

④ 概要设计文档评审

需要评审的内容：设计部分是否完整地实现了需求中规定的功能、性能等要求，设计方案的可行性，关键的处理及内外部接口定义的正确性、有效性，各部分之间的一致性等

软件结构设计工具是结构图，描述软件系统的层次和分块结构关系，它反映了整个系统的功能实现以及模块与模块之间的联系与通讯，是未来程序中的控制层次体系。

结构图的元素：

- 矩形表示一个模块，在矩形内注明模块的功能和名字
- 箭头表示模块间的调用关系。带实心圆的箭头表示传递的是控制信息，带空心圆的箭头表示传递的是数据

结构图中常有的模块类型：

- 传入模块
- 传出模块
- 变换模块
- 协调模块

2）面向数据流的设计方法

① 数据流类型

- 变换型。将数据流分成三个部分：输入数据、中心变换和输出数据三个部分。
- 事务型。在事务中心接收数据，分析数据以确定它的类型，再选取一条活动的通路

② 面向数据流设计方法的实施要点与设计过程

3）设计的准则

- 提高模块的独立性
- 模块规模适中
- 深度、宽度、扇出和扇入适当
- 使模块的作用域在该模块的控制域内
- 应减少模块的接口和界面的复杂性
- 设计成单入口、单出口的模块
- 设计功能可预测的模块

3．详细设计

详细设计，即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，用某种工具表示算法和数据结构的细节。

常用的设计工具有：

- 图形工具：程序流程图，N-S，PAD，HIPO
- 表格工具：判定表
- 语言工具：PDL（伪码）

（四）软件测试

1．软件测试的目的

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否弄清预期的结果与实际结果之间的差别。

2．软件测试的准则

- 所有测试应追溯到需求
- 严格执行测试计划，排除测试的随意性
- 充分注意测试中的群集现象
- 程序员应避免检查自己的程序
- 穷举测试不可能
- 妥善保存测试计划、测试用例、出错统计和最终分析报告，为维护提供方便

3．软件测试技术与方法综述

1）静态测试与动态测试

静态测试包括：代码检查、静态结构分析、代码质量度量等。

动态测试是基于计算机的测试，根据软件需求设计测试用例，利用这些用例去运行程序，以发现程序错误的过程。

2）白盒测试方法与测试用例设计

白盒测试也称结构测试或逻辑驱动测试。

白盒测试的原则：保证所有的测试模块中每一条独立路径至少执行一次；保证所有的判断分支至少执行一次；保证所有的模块中每一个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

主要的方法有：逻辑覆盖（包括语句覆盖、路径覆盖、判定覆盖、条件覆盖和判断—条

件覆盖)、基本路径测试等

3）黑盒测试方法与测试用例设计

黑盒测试方法也称功能测试或数据驱动测试，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、界面错误、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。

黑盒测试方法主要有：等价类划分法（包括有效等价类和无效等价类)、边界值分析法、错误推测法、因果图等，主要用于软件确认测试。

4．软件测试的实施

1）单元测试

对模块进行测试，用于发现模块内部的错误

2）集成测试

测试和组装软件的过程，主要用于发现与接口有关的错误。

集成测试包括的内容：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。

集成测试分为：增量方式组装（包括自顶而下、自底而上、自顶向下和自底向上的混合增量方式）与非增量方式组装。

3）确认测试

验证软件的功能和性能及其他特征是否满足了需求规格说明中确定的各种需求，以及软件配置是否完全、正确。

4）系统测试

将经过测试后的软件，与计算机的硬件、外设、支持软件、数据和人员等其他元素组合在一起，在实际运行环境中进行一系列的集成测试和确认测试。

（五）程序的调试

1．基本概念

程序调试活动包括：根据错误的迹象确定程序中错误的确切性质、原因和位置；对程序进行修改，排除错误。

1）基本步骤

错误定位→修改设计和代码，以排除错误→进行回溯测试，防止引进新的错误。

2）程序调试的原则

（1）确定错误的性质和位置

- 分析与错误有关的信息
- 避开死胡同
- 调试工具只是一种辅助手段，只能帮助思考，不能代替思考
- 避免用试探法

（2）修改错误的原则

- 在出现错误的地方，有可能还有别的错误，在修改时，一定要观察和检查相关的代码，以防止其他的错误
- 一定要注意错误代码的修改，不要只注意表象，而要注意错误的本身，把问题解决
- 注意在修正错误时，可能代入新的错误，错误修改后，一定要进行回归测试，避免新的错误产生
- 修改错误也是程序设计的一种形式
- 修改源代码程序，不要改变目标代码

2．软件调试方法

1）强行排错法

通过内存全部打印来排错
在程序特定部位设置打印语句—即断点法
自动调试工具。

2）回溯法

适合小规模程序的排错。发现错误，分析错误表象，确定位置，再回溯到源程序代码，找到错误位置或确定错误范围。

3）原因排除法

原因排除法包括：演绎法、归纳法和二分法。

演绎法：是一种从一般原理或前提出法，经过排除和精化的过程来推导出结论的思考方法。

归纳法：从一种特殊推断出一般的系统化思考方法。其基本思想是从一些线索着手，通过分析寻找到潜在的原因，从而找出错误。

二分法：如果已知每个变量在程序中若干个关键点的正确值，则可以使用定值语句在程序中的某点附近给这些变量赋值，然后运行程序并检查程序的输出。

三、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。

第四章 数据库设计基础

一、内容要点

（一）数据库系统的基本概念

1．数据、数据库、数据库管理系统

1）数据

数据是指存储在某一种媒体上能够被识别的物理符号，即描述事物的符号记录。
数据是有结构的。首先，数据有型与值的区别，型即类型，值是符合指定类型的值。
数据的概念在数据处理领域中已经大大地拓宽了。数据不仅包括数字、字母、文字和其

他特殊字符组成的文本形式的数据，而且还包括图形、图像、动画、影像、声音等多媒体数据。但是使用最多、最基本的仍然是文字数据。

2）数据库

数据库（DataBase，DB），是存储在计算机存储设备上，结构化的相互关联的数据的集合。它不仅包括描述事物的数据本身，而且还包括相关事物之间的联系。

它用综合的方法组织和管理数据，具有较小的数据冗余，可供多个用户共享，具有较高的数据独立性，具有安全机制，能够保证数据的安全、可靠，允许并发地使用数据库，能有效、及时地处理数据，并能保证数据的一致性和完整性。

例如，某个学校的相关数据，如学生基本情况、选课情况、学籍管理等所涉及的相关数据的集合。

3）数据库管理系统

数据库管理系统（DataBase Management System，DBMS）是对数据库进行管理的系统软件，它的职能是有效地组织和存储数据、获取和管理数据，接受和完成用户提出的访问数据的各种请求。同时还能保证数据的安全性、可靠性、完整性、一致性，还要保证数据的高度独立性。

数据库管理系统主要功能包括以下几个方面：

（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及构建提供有效的存取方法和手段。

（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，一般提供查询、插入、修改和删除数据的功能，此外，还具有简单的算术运算和统计功能，还具有专长强大的程序控制功能。

（4）数据的完整性、安全性定义与检查

数据库中的数据具有内存语义上的关联性与一致性，即数据的完整性。数据的完整性是保证数据库中数据正确的必要条件。

（5）数据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，因此，当多个应用程序对数据库并发操作时，要保证数据不被破坏。

（6）数据的服务

数据库管理系统提供了对数据库中数据的多种服务，如数据拷贝、转存、重组、性能监测、分析等。

数据库管理系统提供的相应的数据语言包括如下：

1）数据定义语言（Data Definition Language，DDL）

D 用户通过它可以方便地对数据库中的相关内容进行定义。例如，对数据库、表、索引进行定义。

2）数据操纵语言（Data Manipulation Language，DML）

用户通过它可以实现对数据库的基本操作。例如，对表中数据的查询、插入、删除和修改。

3）数据控制语言（Data Control Language，DCL）

负责数据完整性、安全性的定义与检查以及并发控制、故障恢复等功能，包括系统初启程序、文件读写与维护程序、存取路径管理程序、缓冲区管理程序、安全性控制程序、完整性检查程序、并发控制程序、事务管理程序、运行日志管理程序、数据库恢复程序等。

目前流行的 DBMS 均为关系型数据库系统，发 ORACLE、Sybase 的 PowerBuilder 及 IBM 的 DB2、微软件的 SQLServer 等。还有一些小型的数据库，如 Visual FoxPro 和 Access 等。

4）数据库管理员

数据库的管理员（DataBase Administrator，DBA）：对数据库的规划、设计、维护、监视等进行管理。

主要工作如下：

（1）数据库设计

（2）数据库维护

（3）改善系统性能，提高系统效率

5）数据库系统

数据库系统（DataBase System，DBS）由如下几个部分组成：

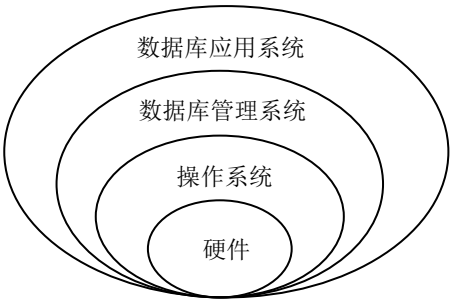
- 数据库（数据）
- 数据库管理系统（软件）
- 数据库管理员（人员）
- 系统平台（硬件平台和软件平台）

硬件平台包括：

- 计算机
- 网络

软件平台包括：

- 操作系统
- 数据库系统开发工具
- 接口软件



数据库系统的组成

6）数据库应用系统

数据库应用系统（DataBase Application System，DBAS）是数据库系统再加上应用软件及应用界面而构成的。它包括：

- 数据库
- 数据库管理系统
- 数据库管理员
- 硬件平台
- 软件平台
- 应用软件
- 应用界面

2．数据库系统的发展

随着计算机软硬件技术的发展，数据处理方法也经历了从低级到高级的发展过程，按照数据管理的特点可将其划分为人工管理、文件系统及数据库系统三个阶段。

1）人工管理阶段

在 20 世纪 50 年代，计算机主要用于数值计算。从当时的硬件看，外存只有纸带、卡片、磁带，没有直接存取设备；从软件看（实际上，当时还未形成软件的整体概念），没有操作系统以及管理数据的软件；从数据看，数据量小，数据无结构，由用户直接管理，且数据间缺乏逻辑组织，数据依赖于特定的应用程序，缺乏独立性。

2）文件系统阶段

是数据库系统发展的初级阶段，它提供了简单的数据共享和数据管理能力，但无法提供完整的、统一的、管理和数据共享的能力。

3）层次数据库与网状数据库阶段

20 世纪 60 年代末期，层次数据库与网状数据库开始发展，它们为统一管理和数据共享提供了支撑，即标志着数据库系统的真正来临。但它们有许多的不足，如受文件的物理影响较大，对数据库使用带来许多不便，数据结构复杂，不变于推广。

4）关系数据库系统阶段

关系数据库系统出现于 20 世界 70 年代，它的数据库结构简单，使用方便，逻辑性强物理性少，使用广泛。

由于应用的领域不同，它常分为：

- 工程数据库系统
- 图形数据库系统
- 图像数据库系统
- 统计数据库系统
- 知识数据库系统
- 分布式数据库系统
- 并行数据库系统
- 面向对象数据库系统

3．数据库系统的基本特点

1）数据的集成性

- 在数据库系统中采用统一的数据结构方式
- 在数据库系统中按照多个应用程序的需要组织全局的统一的数据结构，数据模式可建立全局的数据结构，也可建立数据间的语义联系从而构成一个内存紧密联系的数据整体
- 数据模式是多个应用程序共同的、全局的数据结构，而每个应用的数据则是全局结构中的一部分

2）数据的高共享性与低冗余性

数据的一致性是指系统中同一数据的不同出现应保持相同的值，而数据的不一致性是指同一数据在系统不同拷贝处有不同的值。减少数据的冗余性可以避免数据的不一致性。

3）数据的独立性

数据的独立性是指数据与程序间的互不依赖性。即数据的逻辑结构、存储结构与存取方

式的改变不会影响应用程序。

（1）物理独立性

即数据的物理结构（包括存储结构、存取方式）的改变，不会影响数据库的逻辑结构，即不会引起应用程序的变化。

（2）逻辑的独立性

4）数据统一管理与控制

- 数据库总体逻辑结构的改变，不需要相应修改应用程序。
- 数据完整性检查：检查数据库中数据的正确性以保证数据的正确
- 数据的安全性保护：检查数据库访问者以防非法访问
- 并发控制：控制多个应用程序的并发访问所发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

数据库系统的内部具有三级模式与二级映射。

1）数据库系统的三级模式

数据模式是数据库系统中数据结构的一种表示形式，它具有不同的层次与结构方式。

（1）概念模式

概念模式是数据库系统中全局数据逻辑结构的描述，是全体用户公共数据视图。概念模式主要描述数据的概念记录类型以及它们之间的关系，还包括一些数据间的语义约束。

（2）外模式

外模式又称子模式或用户模式，是用户的数据视图，即用户见到的数据模式。

概念模式给出系统全局的数据描述而外模式则给出每个用户的局部数据描述。

（3）内模式

内模式又称物理模式，它给出数据库物理存储结构与物理存储方法，如数据存储的文件结构、索引、集簇及 hash 等存取方式与存取路径，内模式的物理性主要体现在操作系统及文件级上。

内模式对一般的用户是透明的，但它的设计直接影响到数据库系统的性能。

模式的三个级别层次反映了模式的三个不同环境以及它们的不同要求，其中内模式处于最底层，它反映数据在计算机物理结构中的实际存储形式，概念模式牌中层，它反映了设计者的数据全局逻辑要求，而外模式处于最外层，通过两种映射由物理数据库映射而成它反映用户对数据的要求。

2）数据库系统的二级映射

数据库系统的三级模式是对数据的三个级别抽象，它把数据的具体物理实现留给物理模式，使得全局设计者不必关心数据库的具体实现与物理背景；通过两级映射建立了模式间的联系与转换，使得概念模式与外模式虽然并不物理存在，但也能通过映射获得实体。同时，两级映射也保证了数据库系统中数据的独立性。

两级模式的映射：

- 概念模式到内模式的映射：该映射给出概念模式中数据的全局逻辑结构到数据的物理存储结构间的对应关系
- 外模式到概念模式的映射：该映射给出了外模式与概念模式之间的对应关系

（二）数据模型

1．数据模型的基本概念

数据是现实世界符号的抽象，而数据模型是数据特征的抽象，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了一个抽象的框架。

数据模型描述的三个部分：数据结构、数据操作与数据约束。

（1）数据结构

描述数据的类型、内容、性质及数据间的联系等。

（2）数据操作

主要描述在相应的数据结构上的操作类型与操作方式。

（3）数据约束

主要描述数据结构内数据间的语法、语义联系，它们之间的制约与依存关系，以及数据动态变化的规则，以保证数据的正确、有效与相容。

逻辑数据模型又称数据模型，较为成熟的有：层次模型、网状模型和关系模型。

物理数据模型又称物理模型，是面向计算机物理表示的模型。

2．E-R 模型

1）E-R 模型的基本概念

E-R 模型（Entity—Relationship model），即实体联系模型。

（1）实体

在现实生活中客观存在且又能相互区别的事物，称为实体。

具有共性的实体可组成一个集合称为实体集。

（2）属性

属性是用来描述实体的特征。一个实体有许多个属性。

每个属性都可以有值，一个属性的取值范围称为该属性的值域或值集。

（3）联系

反映事物之间的关联称为联系。

实体集之间的联系有多种，就实体集个数而言，有：

- 两个实体集间的联系
- 多个实体集之间的联系
- 一个实体集内部的联系

两个实体集间的联系即实体集间的函数关系，有如下几种关系：

- 一对一的联系
- 一对多的联系
- 多对多的联系

2）E-R 模型三个基本概念之间的联系关系

（1）实体集与属性之间的联接关系

实体是概念世界中的基本单位，属性附属于实体，它本身并不构成独立性单位。

一个实体可以有若干个属性，实体与它所有属性构成了实体的一个完整描述。实体与属性间有一定的联系。

实体有型与值的区分，一个实体的所有属性的集合，称为实体型，而实体中属性值的集

合，即构成该实体的值。

（2）实体与联系

实体集之间通过联系建立联接关系。

3）E-R 模型的图示法

- 用矩形表示实体集，在矩形内部标出实体集的名称
- 用椭圆形表示属性，在椭圆上标出属性的名称
- 用菱形表示联系，在菱形上标出联系名
- 属性依附于实体，它们之间用无向线段联接
- 属性也依附于联系，它们之间用无向线段联接
- 实体集与联系之间的联接关系，通过无向线段表示

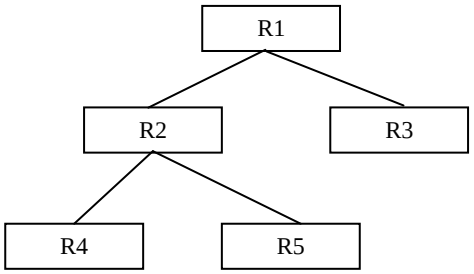
3．层次模型

若用图来表示，层次模型是一棵倒立的树。在数据库中，满足以下两个条件的数据模型称为层次模型：

● 有且仅有一个结点无父结点，这个结点称为根结点

● 其他结点有且仅有一个父结点

在层次模型中，结点层次从根开始定义，根为第一层，根的子结点为第二层，根为其子结点的父结点，同一父结点的子结点称为兄弟结点，没有子结点的结点称为叶结点。



层次模型

层次模型表示的是一对多的关系，即一个父节点可以对应多个子节点。这种模型的优点是简单、直观、处理方便、算法规范；缺点是不能表达含有多对多关系的复杂结构。

R1 是根节点，R2、R3 是 R1 的子结点，它们互为兄弟结点；R4、R5 为 R2 的子结节点，它们也互为兄弟节点；R3、R4、R5 是叶子结点。

其中，每一个节点都代表一个实体型，各实体型由上而下是 1:n 的联系。

支持层次模型的 DBMS 称为层次数据库管理系统，在这种数据库系统中建立的数据库是层次数据库。

层次数据模型支持的操作主要有：查询、插入、删除和更新。

4．网状模型

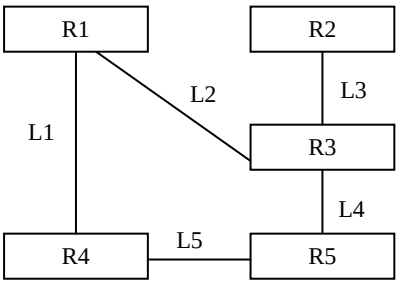
若用图来表示，网状模型是一个网络。在数据库中，满足以下两个条件的数据模型称为网状模型。

● 允许一个以上的结点无父结点

● 一个结点可以有一个以上的父结点

● 允许两个结点间有两种以上的联系，即允许结点间有复合链，用网络表示某种联系

由于在网状模型中子结点与父结点的联系不是唯一的，所以要为每个联系命名，并指出与该联系有关的父结点和子结点。



网状模型

网状模型的优点是可以表示复杂的数据结构，存取数据的效率比较高；缺点是结构复杂，每个问题都有其相对的特殊性，实现的算法难以规范化。

在抽象网状模型中，R1 与 R4 之间的联系被命名为 L1，R1 与 R3 之间的联系被命名为 L2，R2 与 R3 之间的联系被命名为 L3，R3 与 R5 之间的联系被命名为 L4，R4 与 R5 之间的联系被命名为 L5。R1 为 R3 和 R4 的父结点，R2 也是 R3 的父结点。R1 和 R2 没有父结点。

网状模型是一个不加任何条件限制的无向图。它没有层次模型那样需要满足严格的条件，相对比较灵活。

通常的操作方式是将网状模型分解成若干个二级树，即只有两个层次的树。

在网状模型标准中，基本结构简单二级树叫做系，系的基本数据单位是记录，它相当于 E-R 模型中的实体集；记录又可由若干数据项组成，它相当于 E-R 模型中的属性。

5．关系模型

1）关系的数据结构

关系模型是利用二维表来表示，简称表。

表头即属性的集合，在表中每一行存放数据，称为元组。

二维表要求满足的条件：

- 二维表中元组的个数有限
- 元组在二维表中的唯一性，在同一个表中不存在完全相同的两个元组
- 二维表中元组的顺序无关，可以任意调换
- 元组中的各分量不能再分解
- 二维表中各属性名唯一
- 二维表中各属性的顺序无关
- 二维表属性的分量具有与该属性相同的值域

键：能够唯一确定元组的属性或属性的组合。例如，在学生基本情况表中，可以用学号来唯一标识某个学生，即学号可以作为该表的键。键具有标识元组、建立元组间联系等重要作用。

在二维表中凡是能够唯一标识元组的最小属性集称为该表的键或码。二维表中可能有若干个键，称为候选码或候选健。从二维表的所有候选键中选取一个作为用户使用的键称为主键或主码。

外键：如果表中的一个字段不是本表的键或候选键，而是另外一个表的键或候选键，则称该字段为外键或外码。

表中一定有键。

在关系中一般支持空值，空值表示未知的值或不可能出现的值，一般用 NULL 表示。关系的主键中不允许出现空值，因为如主键为空值则失去了其元组标识的作用。

关系模式支持子模式，关系子模式是关系数据库模式中用户所见到的那部分数据模式描述。

2）关系操作

关系模型的数据操纵是建立在关系上的数据操纵，一般有查询、增加、删除和修改。

（1）数据查询

在一个关系中查询数据，操作方式是先定位，然后再操作。

在多个关系中查询数据，先将多个关系合并为一个关系，再在合并后的新关系中进行定位，然后再操作。

（2）数据删除

数据删除操作是在一个关系中删除元组的操作。操作方式也是先定位，然后再删除操作。

（3）数据插入

数据插入也是仅对一个关系的操作。即在指定的关系中插入一个或多个元组。

（4）数据修改

数据修改是在一个关系中修改指定的元组与属性。数据修改不是一个基本的操作，可分解为删除要修改的元组，再插入修改后的元组两个基本操作。

关系的基本操作：

- 关系的属性指定
- 关系的元组选择
- 两个关系合并
- 一个或多个关系的查询
- 关系中元组的插入
- 关系中元组的删除

3）关系中的数据约束

数据约束：实体完整性约束、参照完整性约束和用户定义的完整性约束。

（1）实体完整性约束

要求关系的主键中属性值不能为空值，主键的惟一决定元组的惟一性。

（2）参照完整性约束

关系之间相关联的基本约束，不允许关系引用不存在的元组。

（3）用户定义的完整性约束

用户根据具体的数据环境与应用环境具体设置约束。关系数据库系统提供完整性约束语言，用户利用该语言写出的约束条件，运行时由系统自动检查。

（三）关系代数

1．关系模型的基本操作

关系是由若干个不同的元组组成的，因此关系可看作元组的集合。N 元关系是一个 n 元有序组的集合。

设有一个 n 元关系 R，它有 n 个域，分别是 D_1、D_2、……、D_n，此时，它们的笛卡尔集是：

$$D_1 \times D_2 \times \cdots \times D_n$$

集合可看作是域的笛卡尔积的子集，$R \subseteq D_1 \times D_2 \times \cdots D_n$。

关系模型的四种操作是：

- 插入
- 删除
- 修改
- 查询

可将它们分解为六种基本操作：

- 关系的属性指定
- 关系的元组选择
- 两个关系的合并运算

- 关系的查询
- 关系元组的插入
- 关系元组的删除

2．关系模型的基本运算

1）插入

插入操作可看作是集合的并运算。即在原有的关系 R 中并入要插入的元组 R’，是这两个元组的并运算：$R \bigcup R'$

2）删除

在关系 R 中删除元组 R’，可看作是两个关系的差运算：R-R’

3）修改

修改关系 R 中的元组的内容的操作：先将要修改的元组 R’从关系 R 中删除，再将修改后的元组 R”插入到关系 R 中，即操作为：$(R-R') \bigcup R''$

4）查询

插入运算无法通过传统的集合运算来表示，需要专门的关系运算来实现。

（1）投影运算

投影运算，是从关系中指定若干个属性组合成一个新的关系的操作。投影操作后得到一个新的关系，其关系模式中包含的属性通常比原来的关系少，或者，与原来的关系具有不同的属性顺序。

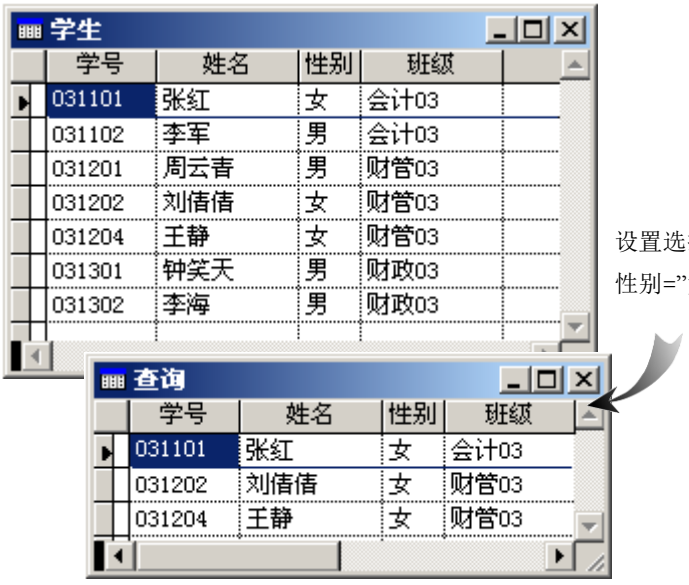
投影是从垂直的角度进行运算，即从列的角度进行运算，投影运算基于一个关系，是一个一元运算。

投影操作

（2）选择

选择，是从关系中查找满足条件的元组。选择的条件是通过逻辑表达式进行描述，逻辑表达式值为真的元组被选出。

选择是从行的角度进行的运算，即从水平方向进行元组的抽取。选择基于一个关系，得到的结果可以形成一个新的关系，它的关系模式与原关系相同，但是原关系的一个子集。例如，从学生表中查找女同学的信息。



选择操作

（3）笛卡尔积运算

两个关系的合并操作可以用笛卡尔积表示。设有 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 和关系 S 的笛卡尔积为 R×S，新关系是一个 n+m 元关系，元组个数是 p×q，由 R 和 S 的有序组合而成。

3．关系代数中的扩充运算

1）交运算

关系 R 与关系 S 经交运算后所得到的关系是既在 R 中又在 S 中的元组组成，记为 $R \cap S$。

2）除运算

如果将笛卡尔积运算看作乘运算的话，除运算即是它的逆运算。当关系 T=R×S 时，则可将运算写成：

T÷R＝S 或 T / R＝S

S 称为 T 除以 R 的商。T 能被除的充分与必要条件是：T 中的域包含 R 中的所有属性，T 中有一些域不出现在 R 中。

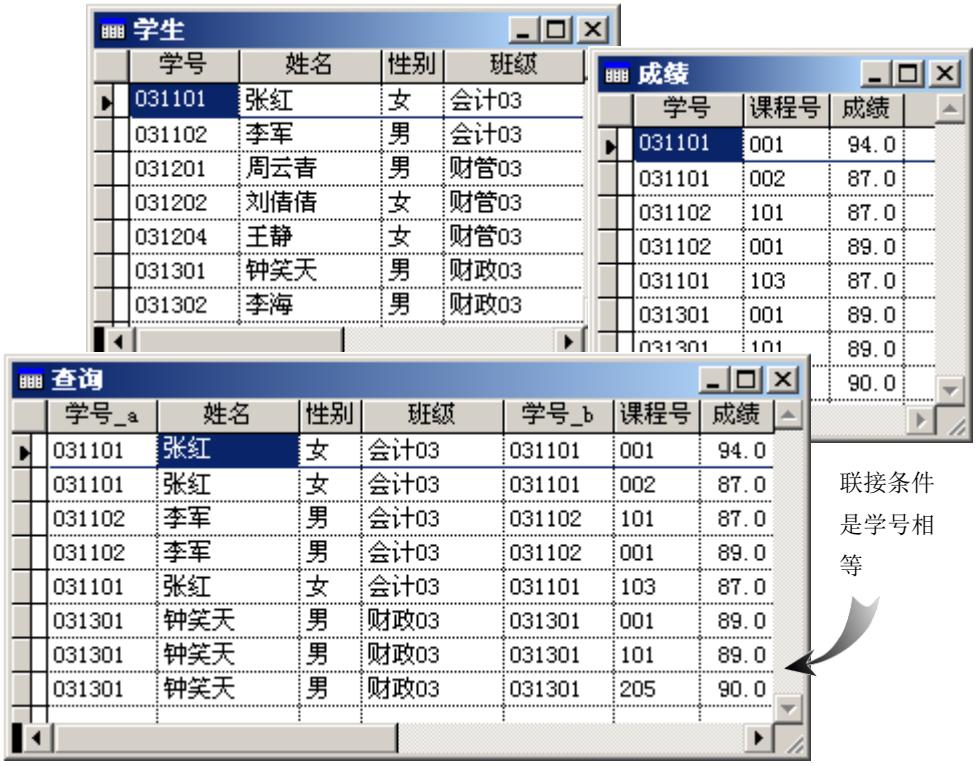
在除运算中 S 的域由 T 中那些不出现在 R 中的域所组成，对于 S 中任一有序组，由它与关系 R 中每个有序组所构成的有序组均出现在关系 T 中。

3）连接与自然连接运算

联接是关系的横向运算。联接运算将两个关系横向地拼接成一个更宽的关系，生成的新关系中有满足联接条件的所有元组。

联接运算通过联接条件来控制，联接条件中将出现两个关系中的公共属性，或者具有相同的域、可比的属性。

连接运算基于两个关系。如下图所示为联接运算的操作。



联接操作

在联接运算中，按字段值相等的为条件进行的联接运算，即为等值联接。上例即为等值联接的运算。

自然联接，是去掉重复属性的等值联接。自然联接是最常用的联接方式。如果上例进行的是自然联接，则联接后的关系如下图所示。

自然联接后的结果

（四）数据库设计与管理

1．数据库设计概述

数据库设计的基本任务是根据用户对象的信息需求、处理需求和数据库的支持环境（包括硬件、操作系统与 DBMS）设计出数据模式。

数据库设计的两种方法：

- 面向数据的方法：以信息需求为主，兼顾处理需求。

- 面向过程的方法：以处理需求为主，兼顾信息需求。

目前，面向数据的设计方法是数据库设计的主流方法。

数据库设计一般采用生命周期法，分为如下几个阶段：

- 需求分析阶段
- 概念设计阶段
- 逻辑设计阶段
- 物理设计阶段
- 编码阶段
- 测试阶段
- 运行阶段
- 进一步修改阶段

前四个阶段是数据库设计的主要阶段，重点以数据结构与模型的设计为主线。

2．数据库设计的需求分析

第一阶段：需求收集和分析，收集基本数据和数据流图。

主要的任务是：通过详细调查现实世界要处理的对象（组织、部门、企业等），充分了解原系统的工作概况，明确用户的各种需求，在此基础上确定新系统的功能。

对数据库的要求：

- 信息要求
- 处理要求
- 安全性和完整性的要求

数据字典是各类数据的集合，它包括五个部分：

- 数据项，即数据的最小单位
- 数据结构，是若干数据项有意义的集合
- 数据流，可以是数据项，也可以是数据结构，用来表示某一处理过程的输入或输出
- 数据存储，处理过程中存取的数据，通常是手工凭证、手工文档或计算机文件
- 处理过程

3．数据库概念设计

1）概念设计概述

（1）集中式模式设计法

根据需求由一个统一的机构或人员设计一个综合的全局模式。适合于小型或并不复杂的单位或部门。

（2）视图集成设计法

将系统分解成若干个部分，对每个部分进行局部模式设计，建立各个部分的视图，再以各视图为基础进行集成。比较适合于大型与复杂的单位，是现在使用较多的方法。

2）数据库概念设计的过程

（1）选择局部应用

根据系统情况，在多层的数据流图中选择一个适当层次的数据流图，将这组图中每一部分对应一个局部应用，以该层数据流图为出发点，设计各自的 E-R 图。

（2）视图设计

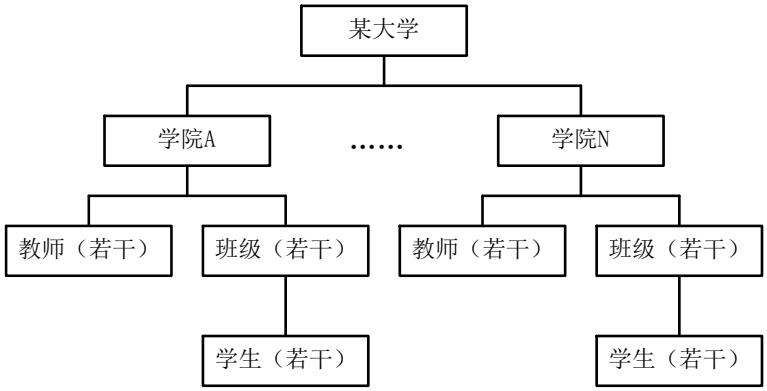
视图设计的三种次序：

- 自顶向下：先从抽象级别高且普遍性强的对象开始逐步细化、具体化和特殊化。

- 由底向上：先从具体的对象开始，逐步抽象，普遍化和一般化，最后形成一个完整的视图设计
- 由内向外：先从最基本与最明显的对象开始，逐步扩充至非基本、不明显的对象。

例：某大学由一名校长主管，学校下设多个学院，每个学院又的多个系；每个系有一名系主任，负责聘任教师；每个教师可以承担多门课，同一门课又可由多个教师承担；每个系有多个班级，每个班级有一定数量的学生；学生在校期间要学习多门课程，学习结束后，每门课程对应一个成绩。要求设计该大学的教学管理系统。

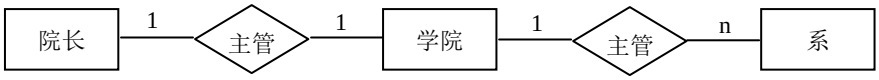
需求分析阶段，得到该学校的机构组织结构图如下图所示：



某大学的组织结构图

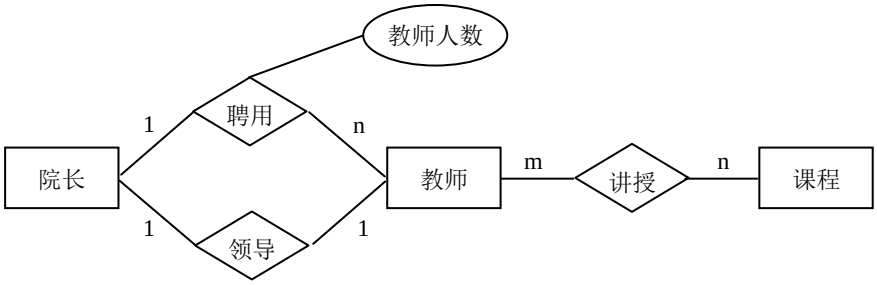
E-R 图设计

首先，设计“院长”、“学院”和“系”之间的联系。一个学院有一个院长，一个院长主管一个学校；一个系属于一个学院，一个学院有多个系。院长与学院的关系是一对一的联系，学院和系之间是一对多的联系。



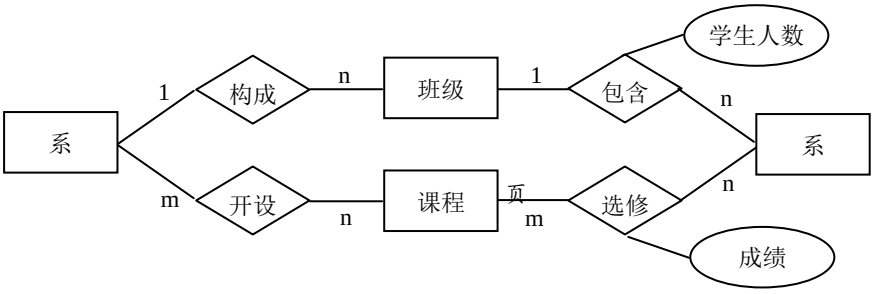
院长、学院、系之间的关系

在系里在，一个系会聘用多个教师，而一个教师只属于一个系，所以，系和教师之间的关系是一对多的联系；一门课可由多个教师讲授，同时，一个教师可讲授多门课，课程和教师之间的关系是多对多的联系。



系、教师、课程之间的联系

在系里，学生和课程之间的联系有，一个系有多个班，一个班只能属于一个系，它们之



系、课程和学生之间的联系

间的联系是一对多的联系；一个班有多个学生，同时，一个学生只属于一个班，所以，班级和学生之间的联系是一对多的联系；系和课程之间的联系，一个系可开设多门课，同时，一门课可被多个系开设，因此，课程和系之间的关系是多对多的联系；学生与课程之间，一个学生会选多门课，同时，一门课可被多个学生选取，因此，课程和学生之间的关系是多对多的联系。

逻辑设计

学院（学院编号，学院名，学院地址，院长编号）

院长（院长编号，院长姓名，联系电话，办公地址）

系（系编号，系名，联系电话，系地址，学院编号，系主任职工号）

教师（职工号，姓名，性别，学历，职称，工资，联系电话，系编号）

班级（班级编号，班级名称，学生人数，系名）

学生（学号，身份证号，姓名，性别，出生日期，民族，籍贯，班级名）

课程（课程编号，课程名称，学分）

开课（系编号，课程号）

授课（职工号，课程号）

选课（学号，课程号，成绩）

（3）视图集成

视图集成是将所有局部视图统一与合并成一个完整的数据模式。

视图集成的重点是解决局部设计中的冲突，常见的冲突主要有如下几种：

- 命名冲突：有同名异义或同义异名
- 概念冲突：同一概念在一处为实体而在另一处为属性或联系
- 域冲突：相同的属性在不同视图中有不同的域
- 约束冲突：不同的视图可能有不同的约束

视图经过合并生成 E-R 图时，其中还可能存在冗余的数据和冗余的实体间联系。冗余数据和冗余联系容易破坏数据库的完整性，给数据库维护带来困难。

对于视图集成后所形成的整体的数据库概念结构必须进行验证，满足下列要求：

- 整体概念结构内部必须具有一致性，即不能存在互相矛盾的表达
- 整体概念结构能准确地反映原来的每个视图结构，包括属性、实体及实体间的联系
- 整体概念结构能满足需求分析阶段所确定的所有要求
- 整体概念结构还需要提交给用户，征求用户和有关人员的意见，进行评审、修改和优化，最后定稿

4．数据库的逻辑设计

1）从 E-R 模型向关系模式转换

E-R 模型向关系模式的转换包括：

- E-R 模型中的属性转换为关系模式中的属性
- E-R 模型中的实体转换为关系模式中的元组
- E-R 模型中的实体集转换为关系模式中的关系
- E-R 模型中的联系转换为关系模式中的关系

转换中存在的一些问题：

- 命名与属性域的处理。名称不要重复，同时，要用关系数据库中允许的数据类型来描述类型

- 非原子属性处理。在 E-R 模型中允许非原子属性存在，但在关系模式中不允许出现非原子属性，因此，要将非原子属性进行转换。
- 联系的转换。通常联系可转换为关系，但有的联系需要归并到相关联的实体中

2）逻辑模式规范化及调整、实现

（1）规范

对关系做规范化验证。

（2）RDBMS

对逻辑模式进行调整以满足 RDBMS 的性能、存储空间等要求，包括如下内容：

- 调整性能以减少连接运算
- 调整关系大小，使每个关系数量保持在合理水平，从而可以提高存取效率
- 尽量采取快照，提高查询速度

3）关系视图设计

逻辑设计又称外模式设计。关系视图是关系模式基础上所设计的直接面向操作用户的视图。

关系视图的作用：

- 提供数据逻辑独立性
- 能适应用户对数据的不同需求
- 有一定数据保密功能

5．数据库的物理设计

物理设计的主要目标是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。

6．数据库管理

数据库管理包括：

1）数据库的建立

数据库建立包括：

- 数据模式的建立。数据模式由 DBA 负责建立，定义数据库名、表及相应的属性，定义主关键字、索引、集簇、完整性约束、用户访问权限、申请空间资源，定义分区等。
- 数据加载。在数据模式定义后可加载数据，DBA 可以编制加载程序将外界的数据加载到数据模式内，完成数据库的建立。

2）数据库的调整

在数据库库建立并运行一段时间后，对不适合的内容要进行调整，调整的内容包括：

- 调整关系模式与视图使之更适应用户的需求
- 调整索引与集簇使数据库性能与效率更佳
- 调整分区、数据库缓冲区大小以及并发度使数据库物理性能更好

3）数据库的重组

数据库运行一段时间后，由于数据的大量插入、删除和修改，使性能受到很大的影响，需要重新调整存贮空间，使数据的连续性更好，即通过数据库的重组来实现。

4）数据库的故障校复

保证数据不受非法盗用与破坏；保证数据的正确性。

5）数据安全性控制与完整性控制

一旦数据被破坏，要及时恢复。

6）数据库监控

DBA 需要随时观察数据库的动态变化，并在发生错误、故障或产生不适应情况时随时采取措施，并监控数据库的性能变化，必要时可对数据库进行调整。

三、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。