Contents

第 1 章　数据结构与算法

第 2 章　程序设计基础

第 3 章　软件工程基础

第 4 章　数据库设计基础

第 1 章　数据结构与算法

1.1　算法的复杂度

1. 算法的基本概念

利用计算机算法为计算机解题的过程实际上是在实施某种算法。

（1）算法的基本特征

算法一般具有 4 个基本特征：可行性、确定性、有穷性、拥有足够的情报。

（2）算法的基本运算和操作

算法的基本运算和操作包括：算术运算、逻辑运算、关系运算、数据传输。

（3）算法的 3 种基本控制结构

算法的 3 种基本控制结构是：顺序结构、选择结构、循环结构。

（4）算法基本设计方法

算法基本设计方法：列举法、归纳法、递推、递归、减半递推技术、回溯法。

（5）指令系统

所谓指令系统指的是一个计算机系统能执行的所有指令的集合。

2. 算法复杂度

算法复杂度包括时间复杂度和空间复杂度。注意两者的区别，无混淆，见表 1-1。

表 1-1　算法复杂性

名称	描述
时间复杂度	执行算法所需要的计算工作量
空间复杂度	执行这个算法所需要的内存空间

1.2　数据结构

1.2.1　逻辑结构和存储结构

1. 数据结构的基本概念

（1）数据结构

指相互有关联的数据元素的集合。

（2）数据结构研究的 3 个方面

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；

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；

③ 对各种数据结构进行的运算。

2. 逻辑结构

数据的逻辑结构是对数据元素之间的逻辑关系的描述，它可以用一个数据元素的集合和定义在此集合中的若干关系来表示。数据的逻辑结构有两个要素：一是数据元素的集合，通常记为 D；二是 D 上的关系，它反映了数据元素之间的前后件关系，通常记为 R。一个数据结构可以表示成：B=(D,R)

其中，B 表示数据结构。为了反映 D 中各数据元素之间的前后件关系，一般用二元组来表示。

例如，如果把一年四季看作一个数据结构，则可表示成：B =(D,R)

D ={春季,夏季,秋季,冬季}

R ={(春季,夏季),(夏季,秋季),(秋季,冬季)}

3. 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（也称数据的物理结构）。

由于数据元素在计算机存储空间中的位置关系可能与逻辑关系不同，因此，为了表示存放在计算机存储空间中的各数据元素之间的逻辑关系（即前后件关系），在数据的存储结构中，不仅要存放各数据元素的信息，还需要存放各数据元素之间的前后件关系的信息。

一种数据的逻辑结构根据需要可以表示成多种存储结构，常用的存储结构有顺序、链接等存储结构。

顺序存储方式主要用于线性的数据结构，它把逻辑上相邻的数据元素存储在物理上相邻的存储单元里，结点之间的关系由存储单元的邻接关系来体现。

链式存储结构就是在每个结点中至少包含一个指针域，用指针来体现数据元素之间逻辑上的联系。

1.2.2 线性结构和非线性结构

根据数据结构中各数据元素之间前后件关系的复杂程度，一般将数据结构分为两大类型：线性结构与非线性结构。

（1）如果一个非空的数据结构满足下列两个条件：

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。

则称该数据结构为线性结构。线性结构又称线性表。在一个线性结构中插入或删除任何一个结点后还应是线性结构。栈、队列、串等都为线性结构。

如果一个数据结构不是线性结构，则称之为非线性结构。数组、广义表、树和图等数据结构都是非线性结构。

（2）线性表的顺序存储结构具有以下两个基本特点：

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。

元素 ai 的存储地址为：ADR(ai)=ADR(a1)+(i-1)k，ADR(a1)为第一个元素的地址，k 代表每个元素占的字节数。

（3）顺序表的运算有查找、插入、删除 3 种。

1.3 栈

1. 栈的基本概念

栈（stack）是一种特殊的线性表，是限定只在一端进行插入与删除的线性表。

在栈中，一端是封闭的，既不允许进行插入元素，也不允许删除元素；另一端是开口的，允许插入和删除元素。通常称插入、删除的这一端为栈顶，另一端为栈底。当表中没有元素时称为空栈。栈顶元素总是最后被插入的元素，从而也是最先被删除的元素；栈底元素总是最先被插入的元素，从而也是最后才能被删除的元素。

栈是按照“先进后出”或“后进先出”的原则组织数据的。例如，枪械的子弹匣就可以用来形象的表示栈结构。子弹匣的一端是完全封闭的，最后被压入弹匣的子弹总是最先被弹出，而最先被压入的子弹最后才能被弹出。

2. 栈的顺序存储及其运算

栈的基本运算有 3 种：入栈、退栈与读栈顶元素。

① 入栈运算：在栈顶位置插入一个新元素；

② 退栈运算：取出栈顶元素并赋给一个指定的变量；

③ 读栈顶元素：将栈顶元素赋给一个指定的变量。

1.4 队列

1. 队列的基本概念

队列是只允许在一端进行删除，在另一端进行插入的顺序表，通常将允许删除的这一端称为队头，允许插入的这一端称为队尾。当表中没有元素时称为空队列。

队列的修改是依照先进先出的原则进行的，因此队列也称为先进先出的线性表，或者后进后出的线性表。例如：火车进遂道，最先进遂道的是火车头，最后是火车尾，而火车出遂道的时候也是火车头先出，最后出的是火车尾。若有队列：

$Q=(q_1,q_2,\cdots,q_n)$

那么，q_1 为队头元素（排头元素），q_n 为队尾元素。队列中的元素是按照 q_1，q_2，…，q_n 的顺序进入的，退出队列也只能按照这个次序依次退出，即只有在 q_1，q_2，…，q_{n-1} 都退队之后，q_n 才能退出队列。因最先进入队列的元素将最先出队，所以队列具有先进先出的特性，体现“先来先服务”的原则。

队头元素 q_1 是最先被插入的元素，也是最先被删除的元素。队尾元素 q_n 是最后被插入的元素，也是最后被删除的元素。因此，与栈相反，队列又称为“先进先出”（First In First Out，简称 FIFO） 或“后进后出”（Last In Last Out，简称 LILO）的线性表。

2. 队列运算

入队运算是往队列队尾插入一个数据元素；退队运算是从队列的队头删除一个数据元素。

队列的顺序存储结构一般采用队列循环的形式。循环队列 s=0 表示队列空；

s=1 且 front=rear 表示队列满。计算循环队列的元素个数：“尾指针减头指针”，若为负数，再加其容量即可。

1.5 链表

在链式存储方式中，要求每个结点由两部分组成：一部分用于存放数据元素值，称为数据域；另一部分用于存放指针，称为指针域。其中指针用于指向该结点的前一个或后一个结点（即前件或后件）。

链式存储方式既可用于表示线性结构，也可用于表示非线性结构。

（1）线性链表

线性表的链式存储结构称为线性链表。

在某些应用中，对线性链表中的每个结点设置两个指针，一个称为左指针，用以指向其前件结点；另一个称为右指针，用以指向其后件结点。这样的表称为双向链表。

在线性链表中，各数据元素结点的存储空间可以是不连续的，且各数据元素的存储顺序与逻辑顺序可以不一致。在线性链表中进行插入与删除，不需要移动链表中的元素。

线性单链表中，HEAD 称为头指针，HEAD=NULL（或 0）称为空表。

如果是双项链表的两指针：左指针（Llink）指向前件结点，右指针（Rlink）指向后件结点。

线性链表的基本运算：查找、插入、删除。

（2）带链的栈

栈也是线性表，也可以采用链式存储结构。带链的栈可以用来收集计算机存储空间中所有空闲的存储结点，这种带链的栈称为可利用栈。

1.6 二叉树

1.6.1 二叉树概念及其基本性质

1. 二叉树及其基本概念

二叉树是一种很有用的非线性结构，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和右子树。

在二叉树中，每一个结点的度最大为 2，即所有子树（左子树或右子树）也均为二叉树。另外，二叉树中的每个结点的子树被明显地分为左子树和右子树。

在二叉树中，一个结点可以只有左子树而没有右子树，也可以只有右子树而没有左子树。当一个结点既没有左子树也没有右子树时，该结点即为叶子结点。

例如，一个家族中的族谱关系如图 1-1 所示：

A 有后代 B，C；B 有后代 D，E；C 有后代 F。

典型的二叉树如图 1-1 所示：

详细讲解二叉树的基本概念，见表 1-2。

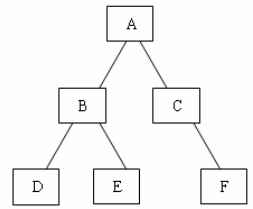

图 1-1　二叉树图

表 1-2　二叉树的基本概念

父结点（根）	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。例如，在图 1-1 中，结点 A 是树的根结点。
子结点和叶子结点	在树结构中，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。例如，在图 1-1 中，结点 D，E，F 均为叶子结点。
度	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。例如，在图 1-1 中，根结点 A 和结点 B 的度为 2，结点 C 的度为 1，叶子结点 D，E，F 的度为 0。所以，该树的度为 2。
深度	定义一棵树的根结点所在的层次为 1，其他结点所在的层次等于它的父结点所在的层次加 1。树的最大层次称为树的深度。例如，在图 1-1 中，根结点 A 在第 1 层，结点 B，C 在第 2 层，结点 D，E，F 在第 3 层。该树的深度为 3。
子树	在树中，以某结点的一个子结点为根构成的树称为该结点的一棵子树。

2. 二叉树基本性质

二叉树具有以下几个性质：

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）总是比度为 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[log2n]+1，其中[log2n]表示取 log2n 的整数部分。

3. 满二叉树与完全二叉树

满二叉树是指这样的一种二叉树：除最后一层外，每一层上的所有结点都有两个子结点。在满二叉树中，每一层上的结点数都达到最大值，即在满二叉树的第 k 层上有 2^{k-1} 个结点，且深度为 m 的满二叉树有 2^m-1 个结点。

完全二叉树是指这样的二叉树：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。

对于完全二叉树来说，叶子结点只可能在层次最大的两层上出现：对于任何一个结点，若其右分支下的子孙结点的最大层次为 p，则其左分支下的子孙结点的最大层次或为 p，或为 p+1。

完全二叉树具有以下两个性质：

性质 1：具有 n 个结点的完全二叉树的深度为[log2n]+1。

性质 2：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（显然也没有右子结点）；

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。

1.6.2 二叉树的遍历

在遍历二叉树的过程中，一般先遍历左子树，再遍历右子树。在先左后右的原则下，根据访问根结点的次序，二叉树的遍历分为三类：前序遍历、中序遍历和后序遍历。

（1）前序遍历

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，仍需先访问根结点，然后遍历左子树，最后遍历右子树。例如，对图 1-1 中的二叉树进行前序遍历的结果（或称为该二叉树的前序序列）为：A，B，D，E，C，F。

（2）中序遍历

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。例如，对图 1-1 中的二叉树进行中序遍历的结果（或称为该二叉树的中序序列）为： D，B，E，A，C，F。

（3）后序遍历

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。例如，对图 1-1 中的二叉树进行后序遍历的结果（或称为该二叉树的后序序列）为： D， E，B，F，C，A。

1.7 查找

1.7.1 顺序查找

查找是指在一个给定的数据结构中查找某个指定的元素。从线性表的第一个元素开始，依次将线性表中的元素与被查找的元素相比较，若相等则表示查找成功；若线性表中所有的元素都与被查找元素进行了比较但都不相等，则表示查找失败。

例如，在一维数组[21，46，24，99，57，77，86]中，查找数据元素 99，首

先从第 1 个元素 21 开始进行比较，比较结果与要查找的数据不相等，接着与第 2 个元素 46 进行比较，以此类推，当进行到与第 4 个元素比较时，它们相等，所以查找成功。如果查找数据元素 100，则整个线性表扫描完毕，仍未找到与 100 相等的元素，表示线性表中没有要查找的元素。

在下列两种情况下也只能采用顺序查找：

① 如果线性表为无序表，则不管是顺序存储结构还是链式存储结构，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。

1.7.2 二分法查找

二分法查找，也称拆半查找，是一种高效的查找方法。能使用二分法查找的线性表必须满足用顺序存储结构和线性表是有序表两个条件。

“有序”是特指元素按非递减排列，即从小到大排列，但允许相邻元素相等。下一节排序中，有序的含义也是如此。

对于长度为 n 的有序线性表，利用二分法查找元素 X 的过程如下：

步骤 1：将 X 与线性表的中间项比较；

步骤 2：如果 X 的值与中间项的值相等，则查找成功，结束查找；

步骤 3：如果 X 小于中间项的值，则在线性表的前半部分以二分法继续查找；

步骤 4：如果 X 大于中间项的值，则在线性表的后半部分以二分法继续查找。

例如，长度为 8 的线性表关键码序列为：[6，13，27，30，38，46，47，70]，被查元素为 38，首先将与线性表的中间项比较，即与第 4 个数据元素 30 相比较，38 大于中间项 30 的值，则在线性表[38，46，47，70]中继续查找；接着与中间项比较，即与第 2 个元素 46 相比较，38 小于 46，则在线性表[38]中继续查找，最后一次比较相等，查找成功。

顺序查找法每一次比较，只将查找范围减少 1，而二分法查找，每比较一次，可将查找范围减少为原来的一半，效率大大提高。

对于长度为 n 的有序线性表，在最坏情况下，二分法查找只需比较 log2n 次，

而顺序查找需要比较 n 次。

1.8 排序

1. 交换类排序法

（1）冒泡排序法

首先，从表头开始往后扫描线性表，逐次比较相邻两个元素的大小，若前面的元素大于后面的元素，则将它们互换，不断地将两个相邻元素中的大者往后移动，最后最大者到了线性表的最后。

然后，从后到前扫描剩下的线性表，逐次比较相邻两个元素的大小，若后面的元素小于前面的元素，则将它们互换，不断地将两个相邻元素中的小者往前移动，最后最小者到了线性表的最前面。

对剩下的线性表重复上述过程，直到剩下的线性表变空为止，此时已经排好序。

在最坏的情况下，冒泡排序需要比较次数为 n(n-1)/2。

（2）快速排序法

任取待排序序列中的某个元素作为基准（一般取第一个元素），通过一次排序，将待排元素分为左右两个子序列，左子序列元素的排序码均小于或等于基准元素的排序码，右子序列的排序码则大于基准元素的排序码，然后分别对两个子序列继续进行排序，直至整个序列有序。

2. 插入类排序法

① 简单插入排序法，最坏情况需要 n(n-1)/2 次比较；

② 希尔排序法，最坏情况需要 $O(n^{1.5})$次比较。

3. 选择类排序法

① 简单选择排序法，最坏情况需要 n(n-1)/2 次比较；

② 堆排序法，最坏情况需要 O(nlog2n)次比较。

相比以上几种（除希尔排序法外），堆排序法的时间复杂度最小。

第 2 章　程序设计基础

2.1　程序设计的方法与风格

养成良好的程序设计风格，主要考虑下述因素：

（1）源程序文档化

① 符号名的命名：符号名的命名应具有一定的实际含义，以便于对程序功能的理解；

② 程序注释：在源程序中添加正确的注释可帮助人们理解程序。程序注释可分为序言性注释和功能性注释。语句结构清晰第一、效率第二；

③ 视觉组织：通过在程序中添加一些空格、空行和缩进等，使人们在视觉上对程序的结构一目了然。

（2）数据说明的方法

为使程序中的数据说明易于理解和维护，可采用下列数据说明的风格，见表 2-1。

表 2-1　数据说明风格

数据说明风格	详细说明
次序应规范化	使数据说明次序固定，使数据的属性容易查找，也有利于测试、排错和维护
变量安排有序化	当多个变量出现在同一个说明语句中时，变量名应按字母顺序排序，以便于查找
使用注释	在定义一个复杂的数据结构时，应通过注解来说明该数据结构的特点

（3）语句的结构程序

语句的结构程序应该简单易懂，语句构造应该简单直接。

（4）输入和输出

输入输出比较简单，这里就不作介绍。

2.2 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引入了工程思想和结构化思想，使大型软件的开发和编程得到了极大的改善。结构化程序设计方法的主要原则为：自顶向下、逐步求精、模块化和限制使用 goto 语句。

① 自顶向上：先考虑整体，再考虑细节；先考虑全局目标，再考虑局部目标；

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：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。

限制使用 goto 语句：在程序开发过程中要限制使用 goto 语句。

2. 结构化程序的基本结构

结构化程序的基本结构有三种类型：顺序结构、选择结构和循环结构。

① 顺序结构：是最基本、最普通的结构形式，按照程序中的语句行的先后顺序逐条执行；

② 选择结构：又称为分支结构，它包括简单选择和多分支选择结构；

③ 循环结构：根据给定的条件，判断是否要重复执行某一相同的或类似的程序段。循环结构对应两类循环语句：先判断后执行的循环体称为当型循环结构；先执行循环体后判断的称为直到型循环结构。

2.3 面向对象方法

面向对象方法涵盖对象及对象属性与方法、类、继承、多态性几个基本要素。

1. 对象

通常把对象的操作也称为方法或服务。

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。属性值应该指的是纯粹的数据值，而不能指对象。

操作描述了对象执行的功能，若通过信息的传递，还可以为其他对象使用。

对象具有如下特征：标识惟一性、分类性、多态性、封装性、模块独立性。

2. 类和实例

类是具有共同属性、共同方法的对象的集合。它描述了属于该对象类型的所有对象的性质，而一个对象则是其对应类的一个实例。

类是关于对象性质的描述，它同对象一样，包括一组数据属性和在数据上的一组合法操作。

3. 消息

消息是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要求的信息，它统一了数据流和控制流。

一个消息由三部分组成：接收消息的对象的名称、消息标识符（消息名）和零个或多个参数。

4. 继承

广义地说，继承是指能够直接获得已有的性质和特征，而不必重复定义它们。

继承分为单继承与多重继承。单继承是指，一个类只允许有一个父类，即类等级为树形结构。多重继承是指，一个类允许有多个父类。

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。

第 3 章　软件工程基础

3.1　软件工程基本概念

1. 软件定义与软件特点

软件指的是计算机系统中与硬件相互依存的另一部分，包括程序、数据和相关文档的完整集合。

程序是软件开发人员根据用户需求开发的、用程序设计语言描述的、适合计算机执行的指令序列。

数据是使程序能正常操纵信息的数据结构。文档是与程序的开发、维护和使用有关的图文资料。

可见，软件由两部分组成：

- 机器可执行的程序和数据；
- 机器不可执行的，与软件开发、运行、维护、使用等有关的文档。

根据应用目标的不同，软件可分应用软件、系统软件和支撑软件（或工具软件），见表 3-1。

表 3-1　软件的分类

名称	描述
应用软件	为解决特定领域的应用而开发的软件
系统软件	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）	支撑软件是介于两者之间，协助用户开发软件的工具性软件

2. 软件工程

为了摆脱软件危机，提出了软件工程的概念。软件工程学是研究软件开发和维护的普遍原理与技术的一门工程学科。所谓软件工程是指采用工程的概念、原理、技术和方法指导软件的开发与维护。软件工程学的主要研究对象包括软件开发与维护的技术、方法、工具和管理等方面。

软件工程包括 3 个要素：方法、工具和过程，见表 3-2。

表 3-2　软件工程三要素

名称	描述
方法	方法是完成软件工程项目的技术手段
工具	工具支持软件的开发、管理、文档生成
过程	过程支持软件开发的各个环节的控制、管理

3.2　软件生命周期

1. 软件生命周期概念

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。软件生命周期分为 3 个时期共 8 个阶段，

- 软件定义期：包括问题定义、可行性研究和需求分析 3 个阶段；
- 软件开发期：包括概要设计、详细设计、实现和测试 4 个阶段；
- 运行维护期：即运行维护阶段。

软件生命周期各个阶段的活动可以有重复，执行时也可以有迭代，如图 3-1 所示。

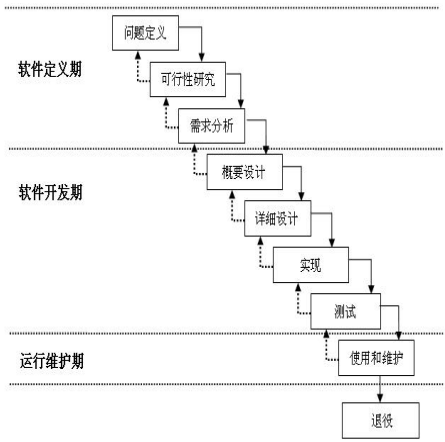

图 3-1　软件生命周期

2. 软件生命周期各阶段的主要任务

在图 3-1 中的软件生命周期各阶段的主要任务，见表 3-3。

表 3-3　软件生命周期各阶段的主要任务

任务	描述
问题定义	确定要求解决的问题是什么
可行性研究与计划制定	决定该问题是否存在一个可行的解决办法，指定完成开发任务的实施计划
需求分析	对待开发软件提出需求进行分析并给出详细定义。编写软件规格说明书及初步的用户手册，提交评审
软件设计	通常又分为概要设计和详细设计两个阶段，给出软件的结构、模块的划分、功能的分配以及处理流程。这阶段提交评审的文档有概要设计说明书、详细设计说明书和测试计划初稿
软件实现	在软件设计的基础上编写程序。这阶段完成的文档有用户手册、操作手册等面向用户的文档，以及为下一步作准备而编写的单元测试计划
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
运行维护	将已交付的软件投入运行，同时不断的维护，进行必要而且可行的扩充和删改

3.3　软件设计

3.3.1　软件设计基本概念

（1）按技术观点分

从技术观点上看，软件设计包括软件结构设计、数据设计、接口设计、过程设计。

① 结构设计定义软件系统各主要部件之间的关系；

② 数据设计将分析时创建的模型转化为数据结构的定义；

③ 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；

④ 过程设计则是把系统结构部件转换为软件的过程性描述。

（2）按工程管理角度分

从工程管理角度来看，软件设计分两步完成：概要设计和详细设计。

① 概要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；

② 详细设计确立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。

3.3.2 软件设计的基本原理

1. 软件设计中应该遵循的基本原理和与软件设计有关的概念

（1）抽象

软件设计中考虑模块化解决方案时，可以定出多个抽象级别。抽象的层次从概要设计到详细设计逐步降低。

（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

信息隐蔽是指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

模块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。模块的独立程度是评价设计好坏的重要度量标准。衡量软件的模块独立性使用耦合性和内聚性两个定性的度量标准。内聚性是信息隐蔽和局部化概念的自然扩展。一个模块的内聚性越强则该模块的模块独立性越强。一个模块与其他模块的耦合性越强则该模块的模块独立性越弱。

2. 衡量软件模块独立性使用耦合性和内聚性两个定性的度量标准

内聚性是度量一个模块功能强度的一个相对指标。内聚是从功能角度来衡量模块的联系，它描述的是模块内的功能联系。内聚有如下种类，它们之间的内聚度由弱到强排列：偶然内聚、逻辑内聚、时间内聚、过程内聚、通信内聚、顺序内聚、功能内聚。

耦合性是模块之间互相连接的紧密程度的度量。耦合性取决于各个模块之间

接口的复杂度、调用方式以及哪些信息通过接口。耦合可以分为多种形势，它们之间的耦合度由高到低排列：内容耦合、公共耦合、外部耦合、控制耦合、标记耦合、数据耦合、非直接耦合。

在程序结构中，各模块的内聚性越强，则耦合性越弱。一般较优秀的软件设计，应尽量做到高内聚，低耦合，即减弱模块之间的耦合性和提高模块内的内聚性，有利于提高模块的独立性。

3.4 结构化分析方法

1. 结构化分析方法的定义

结构化分析方法就是使用数据流图（DFD）、数据字典（DD）、结构化英语、判定表和判定树的工具，来建立一种新的、称为结构化规格说明的目标文档。

结构化分析方法的实质是着眼于数据流、自顶向下、对系统的功能进行逐层分解、以数据流图和数据字典为主要工具，建立系统的逻辑模型。

2. 结构化分析方法常用工具

（1）数据流图（DFD）

数据流图是系统逻辑模型的图形表示，即使不是专业的计算机技术人员也容易理解它，因此它是分析员与用户之间极好的通信工具。

（2）数据字典（DD）

数据字典是对数据流图中所有元素的定义的集合，是结构化分析的核心。

数据流图和数据字典共同构成系统的逻辑模型，没有数据字典数据流图就不严格，若没有数据流图，数据字典也难于发挥作用。

数据字典中有 4 种类型的条目：数据流、数据项、数据存储和加工。

（3）判定表

有些加工的逻辑用语言形式不容易表达清楚，而用表的形式则一目了然。如果一个加工逻辑有多个条件、多个操作，并且在不同的条件组合下执行不同的操作，那么可以使用判定表来描述。

（4）判定树

判定树和判定表没有本质的区别，可以用判定表表示的加工逻辑都能用判定

树表示。

3. 软件需求规格说明书

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。它的特点是具有正确性、无歧义性、完整性、可验证性、一致性、可理解性、可修改性和可追踪性。

3.5 软件测试

3.5.1 软件测试的目的和准则

1. 软件测试的目的

Grenford.J.Myers 给出了软件测试的目的：

- 测试是为了发现程序中的错误而执行程序的过程；
- 好的测试用例（test case）能发现迄今为止尚未发现的错误；
- 一次成功的测试是能发现至今为止尚未发现的错误。

测试的目的是发现软件中的错误，但是，暴露错误并不是软件测试的最终目的，测试的根本目的是尽可能多地发现并排除软件中隐藏的错误。

2. 软件测试的准则

根据上述软件测试的目的，为了能设计出有效的测试方案，以及好的测试用例，软件测试人员必须深入理解，并正确运用以下软件测试的基本准则：

- 所有测试都应追溯到用户需求；
- 在测试之前制定测试计划，并严格执行；
- 充分注意测试中的群集现象；
- 避免由程序的编写者测试自己的程序；
- 不可能进行穷举测试；
- 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。

3.5.2 软件测试的方法和实施

1. 软件测试方法

软件测试具有多种方法，依据软件是否需要被执行，可以分为静态测试和动

态测试方法。如果依照功能划分，可以分为白盒测试和黑盒测试方法。

（1）静态测试和动态测试

① 静态测试包括代码检查、静态结构分析、代码质量度量等。其中代码检查分为代码审查、代码走查、桌面检查、静态分析等具体形式；

② 动态测试。静态测试不实际运行软件，主要通过人工进行分析。动态测试就是通常所说的上机测试，是通过运行软件来检验软件中的动态行为和运行结果的正确性。

动态测试的关键是使用设计高效、合理的测试用例。测试用例就是为测试设计的数据，由测试输入数据和预期的输出结果两部份组成。测试用例的设计方法一般分为两类：黑盒测试方法和白盒测试方法。

（2）黑盒测试和白盒测试

① 白盒测试。白盒测试是把程序看成装在一只透明的白盒子里，测试者完全了解程序的结构和处理过程。它根据程序的内部逻辑来设计测试用例，检查程序中的逻辑通路是否都按预定的要求正确地工作；

② 黑盒测试。黑盒测试是把程序看成一只黑盒子，测试者完全不了解，或不考虑程序的结构和处理过程。它根据规格说明书的功能来设计测试用例，检查程序的功能是否符合规格说明的要求。

2. 软件测试的实施

软件测试过程分 4 个步骤，即单元测试、集成测试、验收测试和系统测试。

单元测试是对软件设计的最小单位——模块（程序单元）进行正确性检验测试。单元测试的技术可以采用静态分析和动态测试。

集成测试是测试和组装软件的过程，主要目的是发现与接口有关的错误，主要依据是概要设计说明书。集成测试所设计的内容包括：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。集成测试时将模块组装成程序，通常采用两种方式：非增量方式组装和增量方式组装。

确认测试的任务是验证软件的功能和性能，以及其他特性是否满足了需求规格说明中确定的各种需求，包括软件配置是否完全、正确。确认测试的实施首先运用黑盒测试方法，对软件进行有效性测试，即验证被测软件是否满足需求规格

说明确认的标准。

系统测试是通过测试确认的软件，作为整个基于计算机系统的一个元素，与计算机硬件、外设、支撑软件、数据和人员等其他系统元素组合在一起，在实际运行（使用）环境下对计算机系统进行一系列的集成测试和确认测试。

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。

3.6 程序的调试

在对程序进行了成功的测试之后将进入程序调试（通常称 Debug，即排错）。

程序的调试任务是诊断和改正程序中的错误。调试主要在开发阶段进行。

程序调试活动由两部分组成，一是根据错误的迹象确定程序中错误的确切性质、原因和位置；二是对程序进行修改，排除这个错误。

程序调试的基本步骤：

① 错误定位。从错误的外部表现形式入手，研究有关部分的程序，确定程序中出错位置，找出错误的内在原因；

② 修改设计和代码，以排除错误；

③ 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，是主要的设计手段，而动态调试是辅助静态调试的。

主要的调试方法有：强行排错法、回溯法和原因排除法 3 种。

第 4 章　数据库设计基础

4.1　数据库的基本概念

数据是数据库中存储的基本对象，它是描述事物的符号记录。

数据库是长期储存在计算机内、有组织的、可共享的大量数据的集合，它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，所以数据库技术的根本目标是解决数据共享问题。

数据库管理系统（DBMS，Database Management System）是数据库的机构，它是一种系统软件，负责数据库中的数据组织、数据操作、数据维护、控制及保护和数据服务等。数据库管理系统是数据系统的核心。

为完成数据库管理系统的功能，数据库管理系统提供相应的数据语言：数据定义语言、数据操纵语言、数据控制语言。

4.2　数据库系统的发展和基本特点

1. 数据库系统的发展

数据管理技术的发展经历了 3 个阶段：人工管理阶段、文件系统阶段和数据库系统阶段。

关于数据管理三个阶段中的软硬件背景及处理特点，简单概括可见表 4-1。

表 4-1　数据管理三个阶段的比较

		人工管理阶段	文件管理阶段	数据库系统管理阶段
背景	应用目的	科学计算	科学计算、管理	大规模管理
	硬件背景	无直接存取设备	磁盘、磁鼓	大容量磁盘
	软件背景	无操作系统	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	分布处理、联机实时处理和批处理

特点	数据管理者	人	文件系统	数据库管理系统
	数据面向的对象	某个应用程序	某个应用程序	现实世界
	数据共享程度	无共享，冗余度大	共享性差，冗余度大	共享性大，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	由应用程序控制	应用程序控制	由 DBMS 提供数据安全性、完整性、并发控制和恢复

2. 数据库系统的特点

数据独立性是数据与程序间的互不依赖性，即数据库中的数据独立于应用程序而不依赖于应用程序。

数据的独立性一般分为物理独立性与逻辑独立性两种。

①物理独立性：当数据的物理结构（包括存储结构、存取方式等）改变时，如存储设备的更换、物理存储的更换、存取方式改变等，应用程序都不用改变。

②逻辑独立性：数据的逻辑结构改变了，如修改数据模式、增加新的数据类型、改变数据间联系等，用户程序都可以不变。

4.3 数据库系统的内部体系结构

1. 数据统系统的 3 级模式

①概念模式，也称逻辑模式，是对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。一个数据库只有一个概念模式；

②外模式，外模式也称子模式，它是数据库用户能够看见和使用的局部数据的逻辑结构和特征的描述，它是由概念模式推导而出来的，是数据库用户的数据

视图，是与某一应用有关的数据的逻辑表示。一个概念模式可以有若干个外模式；

③内模式，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。

内模式处于最底层，它反映了数据在计算机物理结构中的实际存储形式，概念模式处于中间层，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，它反映了用户对数据的要求。

2. 数据库系统的两级映射

两级映射保证了数据库系统中数据的独立性。

①概念模式到内模式的映射。该映射给出了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；

②外模式到概念模式的映射。概念模式是一个全局模式而外模式是用户的局部模式。一个概念模式中可以定义多个外模式，而每个外模式是概念模式的一个基本视图。

4.4 数据模型的基本概念

数据模型从抽象层次上描述了数据库系统的静态特征、动态行为和约束条件，因此数据模型通常由数据结构、数据操作及数据约束三部分组成。

数据库管理系统所支持的数据模型分为 3 种：层次模型、网状模型和关系模型。数据模型特点见表 4-2。

表 4-2 各种数据模型的特点

发展阶段	主要特点
层次模型	用树形结构表示实体及其之间联系的模型称为层次模型，上级结点与下级结点之间为一对多的联系
网状模型	用网状结构表示实体及其之间联系的模型称为网状模型，网中的每一个结点代表一个实体类型，允许结点有多于一个的父结点，可以有一个以上的结点没有父结点
关系模型	用二维表结构来表示实体以及实体之间联系的模型称为关系模型，在关系模型中把数据看成是二维表中的元素，一张二维表就是一个关系

4.5 E-R模型

1. E-R 模型的基本概念

①实体：现实世界中的事物可以抽象成为实体，实体是概念世界中的基本单位，它们是客观存在的且又能相互区别的事物；

②属性：现实世界中事物均有一些特性，这些特性可以用属性来表示；

③码：唯一标识实体的属性集称为码；

④域：属性的取值范围称为该属性的域；

⑤联系：在现实世界中事物间的关联称为联系。

两个实体集间的联系实际上是实体集间的函数关系，这种函数关系可以有下面几种：一对一的关系、一对多或多对一关系、多对多关系。

2. E-R 模型的的图示法

E-R 模型用 E-R 图来表示。

①实体表示法：在 E-R 图中用矩形表示实体集，在矩形内写上该实体集的名字；

②属性表示法：在 E-R 图中用椭圆形表示属性，在椭圆形内写上该属性的名称；

③联系表示法：在 E-R 图中用菱形表示联系，菱形内写上联系名。

4.6 关系模型

关系模式采用二维表来表示，一个关系对应一张二维表。可以这么说，一个关系就是一个二维表，但是一个二维表不一定是一个关系。

- 元组：在一个二维表（一个具体关系）中，水平方向的行称为元组。元组对应存储文件中的一个具体记录；
- 属性：二维表中垂直方向的列称为属性，每一列有一个属性名；
- 域：属性的取值范围，也就是不同元组对同一属性的取值所限定的范围。

在二维表中惟一标识元组的最小属性值称为该表的键或码。二维表中可能有若干个健，它们称为表的侯选码或侯选健。从二维表的所有侯选键选取一个作为

用户使用的键称为主键或主码。表 A 中的某属性集是某表 B 的键，则称该属性值为 A 的外键或外码。

关系模型采用二维表来表示，二维表一般满足下面 7 个性质：

①二维表中元组个数是有限的——元组个数有限性；

②二维表中元组均不相同——元组的唯一性；

③二维表中元组的次序可以任意交换——元组的次序无关性；

④二维表中元组的分量是不可分割的基本数据项——元组分量的原子性；

⑤二维表中属性名各不相同——属性名唯一性；

⑥二维表中属性与次序无关，可任意交换——属性的次序无关性；

⑦二维表属性的分量具有与该属性相同的值域——分量值域的统一性。

关系操纵：数据查询、数据的删除、数据插入、数据修改。

关系模型允许定义三类数据约束，它们是实体完整性约束、参照完整性约束以及用户定义的完整性约束。

4.7 关系代数

1. 传统的集合运算

（1）投影运算

从关系模式中指定若干个属性组成新的关系称为投影。

投影是从列的角度进行的运算，相当于对关系进行垂直分解。经过投影运算可以得到一个新的关系，其关系模式所包含的属性个数往往比原关系少，或者属性的排列顺序不同。

（2）选择运算

从关系中找出满足给定条件的元组的操作称为选择。

选择是从行的角度进行的运算，即水平方向抽取记录。经过选择运算得到的结果可以形成新的关系，其关系模式不变，但其中的元组是原关系的一个子集。

（3）迪卡尔积

设有 n 元关系 R 和 m 元关系 S，它们分别有 p 和 q 个元组，则 R 与 S 的笛卡儿积记为：R×S。

它是一个 m+n 元关系，元组个数是 p×q。

2. 关系代数的扩充运算

（1）交

假设有 n 元关系 R 和 n 元关系 S，它们的交仍然是一个 n 元关系，它由属于关系 R 且由属于关系 S 的元组组成，并记为 R∩S，它可由基本运算推导而得：

$R \cap S = R - (R - S)$

4.8 数据库设计与原理

数据库设计中有两种方法，面向数据的方法和面向过程的方法：

面向数据的方法是以信息需求为主，兼顾处理需求；面向过程的方法是以处理需求为主，兼顾信息需求。由于数据在系统中稳定性高，数据已成为系统的核心，因此面向数据的设计方法已成为主流。

数据库设计目前一般采用生命周期法，即将整个数据库应用系统的开发分解成目标独立的若干阶段。它们是：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、运行阶段和进一步修改阶段。在数据库设计中采用前 4 个阶段。