第 7 章　软件技术基础理论

教学目标：

- 数据结构
  ①了解数据结构的基本概念。
  ②掌握线性结构及其操作。
  ③了解树、二叉树的概念及操作。
  ④了解图的基本概念及操作。
  ⑤掌握常用的排序和查找算法。
- 软件工程
  ①了解软件及软件危机的概念。
  ②了解软件工程学的相关知识。
  ③掌握软件生存周期。
- 数据库技术
  ①了解数据库涉及的基本概念。
  ②理解 DBMS 的功能。
  ③理解数据模型的概念。
  ④掌握关系数据库及其相关知识。
- 操作系统
  ①了解操作系统的基本概念。
  ②了解操作系统的类型。
  ③了解操作系统的基本特征。
  ④了解操作系统的基本功能。

7.1　数据库技术

7.1.1　数据库系统概述

1. 数据库的地位

数据库技术产生于 20 世纪 60 年代末，是数据管理的最新技术，是计算机科学的重要分支。数据库技术是信息系统的核心和基础，它的出现极大地促进了计算机应用向各行各业的渗透。数据库的建设规模、数据库信息量的大小和使用频度已成为衡量一个国家信息化程度的重要标志。

2. 4 个基本概念

（1）数据。数据（Data）是数据库中存储的基本对象。

数据的定义：是描述事物的符号记录。

数据的种类：文字、图形、图像、声音等。

数据的特点：数据与其语义是不可分的。

例如，学生档案中的学生记录：(李明,男,1985,江苏,计算机系,2003)。

在上例中，数据的形式不能完全表达其内容，需要对数据进行解释——数据的语义。

在上述记录数据中，加上如下的语义：

(学生姓名,性别,出生年月,籍贯,所在系别,入学时间)，解释：李明是个男大学生，1985 年出生，江苏人，2003 年考入计算机系。

当然，还可以给出另一个解释和语义，请同学们自己完成。

人们收集并抽取出一个应用所需要的大量数据之后，应将其保存起来以供进一步加工处理，进一步抽取有用信息。

（2）数据库。数据库（Database，DB）是长期存储在计算机内、有组织的、可共享的大量数据的集合。

数据库的特征

- 数据按一定的数据模型组织、描述和存储。
- 可为各种用户共享。
- 冗余度较小。
- 数据独立性较高。
- 易扩展。

（3）数据库管理系统。数据库管理系统（Database Management System，DBMS）是位于用户与操作系统之间的一层数据库管理软件。

DBMS 的用途：科学地组织和存储数据、高效地获取和维护数据。

DBMS 的主要功能如下：

- 数据定义功能：提供数据定义语言（DDL），定义数据库中的数据对象。
- 数据操纵功能：提供数据操纵语言（DML），操纵数据实现对数据库的基本操作（查询、插入、删除和修改）。
- 数据库的运行管理：保证数据的安全性、完整性、多用户对数据的并发使用、发生故障后的系统恢复、数据库的建立和维护功能（实用程序）、数据库数据批量装载、数据库转储、介质故障恢复、数据库的重组织、性能监视等。

（4）数据库系统。数据库系统（Database System，DBS）是指在计算机系统中引入数据库后的系统构成。

在不引起混淆的情况下常常把数据库系统简称为数据库。一般来讲，数据库系统由数据库、数据库管理系统（及其开发工具）、应用系统、数据库管理员（和用户）构成。

7.1.2 数据模型

1. 数据模型的基本概念

在数据库中用数据模型这个工具来抽象、表示和处理现实世界中的数据和信息。通俗地讲数据模型就是现实世界的模拟。数据模型应满足三方面要求：

- 能比较真实地模拟现实世界。
- 容易为人所理解。
- 便于在计算机上实现。

数据模型分成两个不同的层次：

- 概念模型：也称信息模型，它是按用户的观点来对数据和信息建模。
- 数据模型：主要包括网状模型、层次模型、关系模型等，它是按计算机系统的观点对数据建模。

客观对象的抽象过程：两步抽象。

- 现实世界中的客观对象抽象为概念模型。
- 把概念模型转换为某一 DBMS 支持的数据模型。

2. 数据模型的组成要素

数据模型由数据结构、数据操作、数据的约束条件三要素组成。

（1）数据结构。所研究的对象类型的集合，它用来描述系统的集合结构，规定了数据模型的静态特性。数据结构规定了如何把基本的数据项组织成较大的数据单位，以描述数据的类型、内容、性质和数据之间的关系。在数据库中，通常按照数据结构的类型来命名数据模型。

（2）数据操作。对数据库中各种对象（型）的实例（值）允许执行的操作及有关的操作规则，数据操作的类型有：检索、更新（包括插入、删除、修改）。

（3）数据的约束条件。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存储规则，用以限定符合数据模型的数据库状态以及状态的变化，以保证数据的正确、有效、相容。

7.1.3　概念模型

概念模型用于信息世界的建模，是现实世界到机器世界的一个中间层次，它是数据库设计的有力工具。概念模型是数据库设计人员和用户之间进行交流的语言。

对概念模型的基本要求是要有较强的语义表达能力，能够方便、直接地表达应用中的各种语义，知识简单、清晰、易于用户理解。

概念模型中涉及的基本术语包括实体（Entity）、属性（Attribute）、码（Key）、域（Domain）、实体型（Entity Type）、实体集（Entity Set）、联系（Relationship）。

（1）实体。客观存在并可以相互区分的事物称为实体。它是信息世界的基本单位。实体既可以是人，也可以是物；既可以是实际对象，也可以是抽象对象；既可以是事物本身，也可以是事物之间的联系。

（2）属性。实体所具有的某一特性称为属性。一个实体可以由若干个属性来刻画。

（3）码（关键字）。唯一标识实体的属性或属性的组合称为码或关键字。

（4）域。属性的取值范围称为域。

（5）实体型。用实体名及其属性名集合来抽象和刻画同类实体。

（6）实体集。同型实体的集合称为实体集。

7.1.4　联系及联系的类型

实体型间的联系可以分为三类：一对一联系（1:1）、一对多联系（1:n）、多对多联系（m:n）。

（1）一对一联系（1:1）。如果对于实体集 A 中的每一个实体，实体集 B 中至多有一个实体与之联系，反之亦然，则称实体集 A 与实体集 B 具有一对一联系，记为 1:1。

班级与班长之间的联系：一个班级只有一个正班长，一个班长只在一个班里任职，则：班长与班级之间具有 1:1 的联系。

（2）一对多联系（1:n）。如果对于实体集 A 中的每一个实体，实体集 B 中有 n 个实体（n≥0）与之联系，反之，对于实体集 B 中的每一个实体，实体集 A 中至多只有一个实体与之联系，则称实体集 A 与实体集 B 有一对多联系，记为 1:n。

班级与学生之间的联系：一个班级中有若干名学生，每个学生只在一个班级里学习，则：班级和学生之间具有 1:n 的联系。

（3）多对多联系（m:n）。如果对于实体集 A 中的每一个实体，实体集 B 中有 n 个实体（n

≥0）与之联系，反之，对于实体集 B 中的每一个实体，实体集 A 中有 m 个实体（m≥0）与之联系，则称实体集 A 与实体集 B 具有多对多联系，记为 m:n。

课程与学生之间的联系：一门课程同时有若干个学生选修，一个学生可以同时选修多门课程，则课程与学生之间具有 m:n 的联系。

实体之间存在什么样的联系是一个语义范畴内的问题，也就是说两个相同的实体在不同的环境中可能存在不同类型的联系。例如在上述例子中，如果现实中是这样的情况：一门课程只能有一个学生选修，一个学生只能选修一门课程，则课程与学生之间具有的关系为 1:1（当然，现实中并没有这样的规定，这里只是一种假设）。

7.1.5 概念模型的表示方法——实体－联系方法

概念模型的表示方法最常用的是实体－联系方法（E-R 方法），这是 P.P.S.Chen 于 1976 年提出的。该方法是用 E-R 图来描述某一组织的概念模型。这里我们简单介绍 E-R 图的要点，有关如何认识、分析现实世界，从中抽取实体、实体间的联系及建立概念模型的方法请参阅其他参考书籍。

E-R 图要点介绍：

（1）用矩形表示实体型，矩形框内写明实体名。

（2）用椭圆形表示属性，并用无向边将其与相应的实体连接起来。

例如，学生实体具有学号、姓名、性别、年龄、系等属性，则用 E-R 图表示如图 7-1 所示。

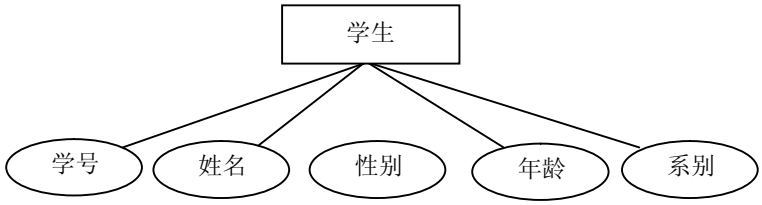


图 7-1　学生实体及其属性

（3）联系。

联系本身：用菱形表示，菱形框内写明联系名，并用无向边分别与有关实体连接起来，同时在无向边旁标上联系的类型（1:1、1:n 或 m:n）。

联系的属性：联系本身也是一种实体型，也可以有属性。如果一个联系具有属性，则这些属性也要用无向边与该联系连接起来。

下面用 E-R 图来表示学生和课程的概念模型。

学生和课程管理涉及的实体有：

- 学生：属性有学号、姓名、性别、年龄、系别。
- 课程：属性有课程号、课程名、学分、出版社、定价。

实体间的联系有（语义描述）：

- 每个学生的学号不能重复；每门课程的课程号不能重复。可以得到学生实体和课程实体的主码（关键字）分别为学号和课程号。
- 一个学生可以选修多门课程，一门课程可以被多个学生选修。可以得到学生和课程之间具有的联系类型为 m:n。

- 每个学生选修一门课程对应有一个成绩。

则 E-R 图如图 7-2 所示。

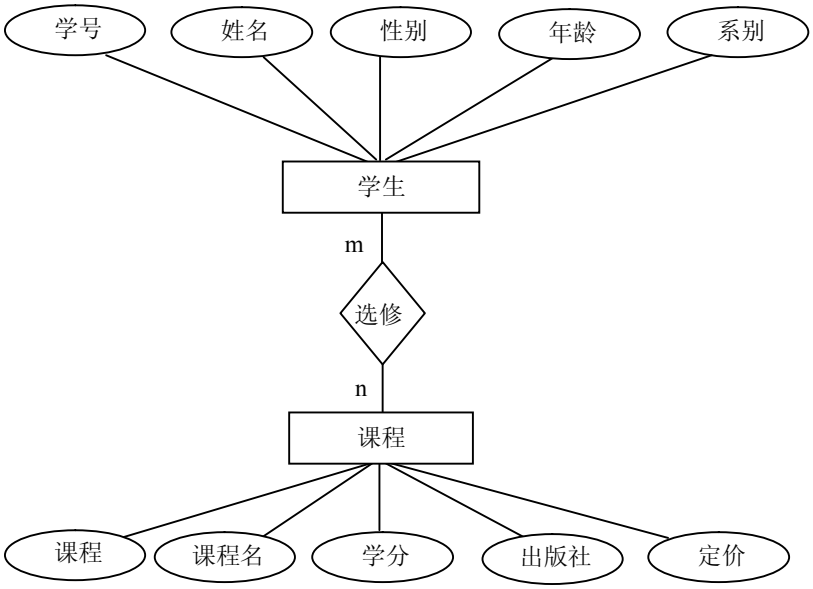


图 7-2　学生—课程的 E-R 图

7.1.6　4 种常用数据模型

当前，实际数据库系统中所支持的主要数据模型有：层次模型、网状模型、关系模型、面向对象模型。

1. 层次模型

满足下面两个条件的基本层次联系的集合为层次模型：

（1）有且只有一个节点没有双亲节点，这个节点称为根节点。

（2）根以外的其他节点有且只有一个双亲节点。

图 7-3 所示就是一个层次模型。

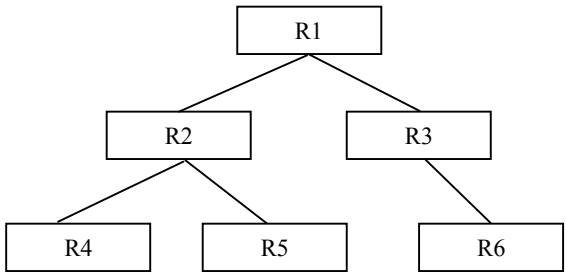


图 7-3　层次模型

2. 网状模型

满足下面两个条件的基本层次联系的集合为网状模型：

（1）允许一个以上的节点无双亲。

（2）一个节点可以有多于一个的双亲。

图 7-4 所示就是一个网状模型。

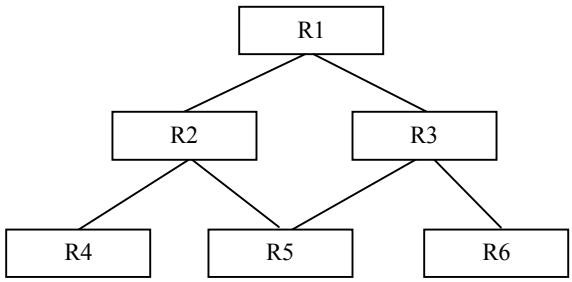


图 7-4　网状模型

3. 关系模型

关系模型是数据模型中最重要的模型。自 20 世纪 80 年代以来，计算机厂商新推出的数据库管理系统几乎都是支持关系模型的。许多非关系系统的产品也都加上了关系接口。在数据库领域中当前的研究工作也都是以关系方法为基础的。

在关系模型中，数据在用户观点下的逻辑结构是一张二维表，例如表 7-1 所示的学生基本情况表。

表 7-1　学生基本情况表

学号	姓名	性别	出生年月	……
200901001	李　军	男	1985-12-01	……
200901002	王小二	男	1984-06-01	……
200901003	李小弗	男	1984-11-12	……
200901004	李　玲	女	1985-12-12	……
……	……	……	……	……

关系模型具有以下特点：

（1）概念单一。无论是实体还是实体间的联系都用关系来表示，数据结构简单、清晰，用户易懂易用。

（2）关系必须是规范化的关系。所谓规范化是指关系模型中，每一个关系模式要满足一定的要求或者称为规范条件，规范条件很多，本书中不做详细讨论。但基本的要求是每一个分量是一个不可分的数据项，即不允许表中还包含表。

（3）在关系模型中，用户对数据的查询操作只是从原来的表中得到一张新的表，这表达了 3 个观点：

- 在用户眼中，无论是原始数据还是结果数据都是同一种数据结构——二维表。
- 关系模型中数据操作是集合操作，即操作对象和结果是若干元组的集合。
- 关系模型把存取路径向用户隐藏起来，用户只要指出“干什么”或“找什么”，不必详细说明“怎么干”或“怎么找”，大大提高了数据的独立性，提高了用户效率。

由于关系模型概念简单、清晰，用户易懂易用，有严格的数学基础及在此基础上发展的关系数据理论，简化了程序员的工作和数据库开发建立的工作，因而关系模型诞生以后发展迅速，成为深受用户欢迎的数据模型。

7.1.7　关系数据库基础知识

关系数据库是依照关系模型设计的若干二维表文件的集合。关系数据库应用数学方法来处理数据库中的数据，20 世纪 80 年代后，关系数据库系统成为最重要、最流行的数据库系统。

1. 关系术语

关系是建立在数学集合概念基础之上的，由行和列表示的二维表。

（1）关系：一个关系对应一张表。

（2）元组：表中的一行即为一个元组。

（3）属性：表中的一列即为一个属性，给每一个属性起一个名称即属性名。

（4）主码：表中的某个属性组，它可以唯一确定一个元组。

（5）域：属性的取值范围。

（6）分量：元组中的一个属性值。

（7）关系模式：对关系的描述。关系名(属性 1,属性 2,…,属性 n)，例如学生(学号,姓名,年龄,性别,系,年级)。

2. 关系数据结构

单一的数据结构——关系：现实世界的实体以及实体间的各种联系均用关系来表示。

数据的逻辑结构——二维表：从用户角度，关系模型中数据的逻辑结构是一张二维表。

3. 关系操作

（1）常用的关系操作。

查询：选择、投影、连接、除、并、交、差。

数据更新：插入、删除、修改。

（2）关系操作的特点。集合操作方式，即操作的对象和结果都是集合。

4. 关系的 3 类完整性约束

对如下的 3 个关系模式：

学生(学号,姓名,性别,年龄,系别)，描述学生基本信息。

课程(课程号,课程名,学分,出版社,定价)，描述课程基本信息。

选课(学号,课程号,成绩)，描述学生选修课程的成绩信息。

在 3 个关系模式中，加上下划线标志的属性为该关系模式的主码。如果某一个属性不是关系模式的主码，但它是另一个关系的主码，则称为外码。如选课关系模式中，学号不是主码，但学号在学生关系模式中是主码，因此选课关系中的学号被称为外码。

（1）实体完整性：要求关系的关键字的值不能为空，也不能重复，通常由关系系统自动支持。例如，在学生关系模式中，学号不能取空值，也不能重复。

关系模型必须遵守实体完整性规则的原因如下：

1）实体完整性规则是针对基本关系而言的。一个基本表通常对应现实世界的一个实体集或多对多联系。

2）现实世界中的实体和实体间的联系都是可区分的，即它们具有某种唯一性标识。

3）相应地，关系模型中以主码作为唯一性标识。

4）主码中的属性即主属性不能取空值。空值就是“不知道”或“无意义”的值。主属性取空值，就说明存在某个不可标识的实体，即存在不可区分的实体，这与第 2）点相矛盾，因此这个规则称为实体完整性。

（2）参照完整性：指外码的取值必须参照主码，也就是说外码究竟可以取哪些值，要受取主码集合的限制。如选课关系中的学号的取值必须来自于学生关系中的学号取值集合。课程号的取值也受到课程关系中课程号值的集合的限制。

（3）域完整性：它是针对某一应用环境的完整性约束条件，主要反映了某一具体应用所涉及的数据应满足的要求。

7.2 数据结构

在计算机系统软件和应用软件中都要用到各种数据结构，因此，要进行高质量的程序设计和软件开发，仅掌握几种计算机语言而缺乏数据结构方面的知识是不够的，难以应付各种复杂的问题。

现实世界的事物及其相互关系是十分复杂的，数据元素之间的相互关系往往无法用数学方程式来进行描述。因此，解决此类问题的关键已不再是分析数学和计算方法，而更重要的是设计出合适的数据结构，才能有效地解决问题。为了能够用计算机分析、解决各种各样的问题，就必须研究客观事物以及它们的逻辑联系在计算机内的表达和存储的模型，研究建立在这个模型之上的相应运算和处理的实现。

7.2.1 数据结构的基本概念

数据是描述客观事物并能被计算机加工处理的符号的集合。数据元素是数据的基本单位，即数据集合中的个体。有些情况下也把数据元素称为节点、记录等。一个数据元素可由一个或多个数据项组成。数据项是有独立含义的数据最小单位，有时也把数据项称为域、字段等。

数据结构（Data Structure）是指数据元素的组织形式和相互关系。数据结构一般包括以下三方面内容：

（1）数据的逻辑结构。数据的逻辑结构从逻辑上抽象地反映数据元素间的结构关系，它与数据在计算机中的存储表示方式无关。

数据的逻辑结构有两大类：

1）线性结构。线性结构的逻辑特征是：有且仅有一个始端节点和一个终端节点，并且除两个端点节点外的所有节点都有且仅有一个前趋节点和一个后继节点。线性表、堆栈、队列、数组、串等都是线性结构。

2）非线性结构。非线性结构的逻辑特征是：一个节点可以有多个前趋节点和后继节点。如树型结构、图等。

（2）数据的物理结构。数据的物理结构是逻辑结构在计算机存储器里的映像，也称为存储结构。

数据的存储结构可用以下 4 种基本存储方法实现：

1）顺序存储方法：把逻辑上相邻的节点存储在物理位置上相邻的存储单元里，节点之间的逻辑关系由存储单元的邻接关系来体现。由此得到的存储结构称为顺序存储结构。

2）链式存储方法：不要求逻辑上相邻的节点在物理位置上也相邻，节点之间的逻辑关系是由附加的指针字段表示的。由此得到的存储结构称为链式存储结构。

3）索引存储方法：在存储节点信息的同时，还建立附加的索引表，索引表中的每一项称为索引项。索引项由关键字和地址组成，关键字是能唯一标识一个节点的数据项，而地址一般是指示节点所在存储位置的记录号。

4）散列存储方法：根据节点的关键字直接计算出该节点的存储地址。

用不同的存储方法对同一种逻辑结构进行存储映像，可以得到不同的存储结构。4 种基本的存储方法也可以组合起来对数据逻辑结构进行存储映像。

（3）数据的运算。数据的运算是指对数据施加的操作。它是定义在数据的逻辑结构上的，但运算的具体实现要在物理结构上进行。数据的每种逻辑结构都有一个运算的集合，常用的运算有检索、插入、删除、更新、排序等。

7.2.2　数据结构的意义

数据结构实际上就是研究以下问题：

- 数据元素间的逻辑关系是什么？
- 适宜采用什么样的存储结构？
- 有哪些基本运算？怎样实现？

实际上也就是数据结构的 3 个层次：数据的逻辑结构、数据的存储结构和数据操作相关算法集合。

“例 7-1”已知学生基本情况表由学号、姓名、性别、出生年月等信息组成，设计一个查询程序，要求能根据指定的学生姓名进行查询，能检索出某出生年月的学生信息。

显然，该程序的设计直接依赖于数据行（记录）是怎样组织和存储的，即依赖于这 n 个数据元素的结构。如果这些数据没有任何规律地存放在计算机中，只能用逐一比较的方法顺序查找。这种查找方式对数量不大的对象或许是可行的，但对于大数据量的查询就难以实行了。学生基本情况如表 7-2 所示。

表 7-2　学生基本情况表

学号	姓名	性别	出生年月	……
200901001	李　军	男	1985-12-01	……
200901002	王小二	男	1984-06-01	……
200901003	李小弗	男	1984-11-12	……
200901004	李　玲	女	1985-12-12	……
……	……	……	……	……

特点：

- 每个学生的信息占一行，所有学生的信息按学号顺序依次排列构成一张表格。

- 表中每个学生的信息依据学号的大小存在着一种前后关系，这就是我们所说的线性结构。
- 对它的操作通常是插入某个学生的信息、删除某个学生的信息、更新某个学生的信息、按条件检索某个学生的信息等。

这张学生信息表可称为一个数据结构，它是一个线性表结构，表中的每一行为一个数据元素（它是数据结构的基本单位），它由学号、姓名、性别以及出生年月等数据项组成。

说明：

- 表中数据元素的逻辑关系是：对表中任一数据元素，与它相邻且在它前面的数据元素（也称为直接前趋）最多只有一个；与表中任一数据元素相邻且在它后面的数据元素（也称为直接后继）也最多只有一个。表中第一个元素没有直接前趋，故称为起始数据元素，最后一个数据元素没有直接后继，故称为终点数据元素。
- 该表的存储方式（指计算机存储器如何表示这种关系）：表中的数据元素是顺序地邻接在一片连续的单元之中，即顺序存储。
- 应针对表元素的逻辑顺序性和存储的顺序性设计高效合理的算法，实现查找、删除、插入等操作。

“例 7-2”输出 n 个对象的全排列可以使用如图 7-5 所示的形式描述。

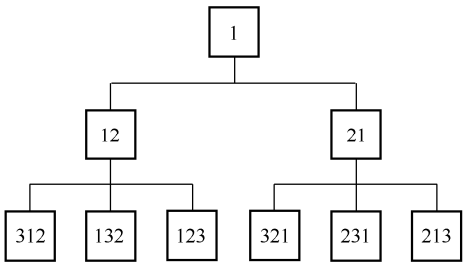


图 7-5　3 个对象的全排列过程

特点：

- 在求解过程中，所处理的数据之间具有层次关系，这就是我们所说的树型结构。
- 对它的操作有：建立树型结构，输出最低层节点内容等。

“例 7-3”在制定教学计划时，需要考虑各门课程的开设顺序（如表 7-3 所示）。有些课程需要先导课程，有些课程则不需要，而有些课程又是其他课程的先导课程。

表 7-3　课程信息表

课程编号	课程名称	先导课程
C1	程序设计基础	C1
C2	离散数学	C1、C2
C3	数据结构	C1
C4	汇编语言	C3、C4
C5	算法分析与设计	C11
C6	计算机组成原理	C3、C5

续表

课程编号	课程名称	先导课程
C7	编译原理	C3、C6
C8	操作系统	
C9	高等数学	C9
C10	线性代数	C9
C11	普通物理	C9、C10、C1
C12	数值分析	

课程先后关系的图形描述形式如图 7-6 所示。

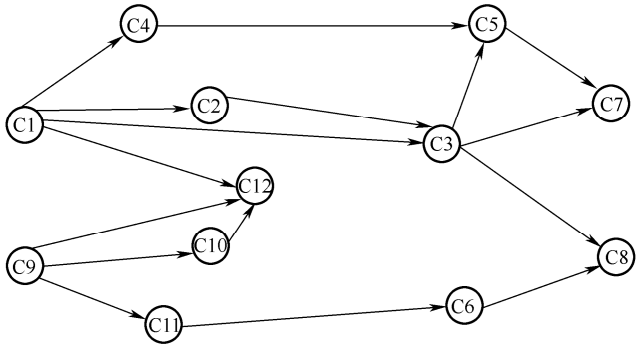


图 7-6　课程开设先后关系

特点：

- 课程之间的先后关系用图结构描述。
- 通过实施创建图结构，按要求将图结构中的顶点进行线性排序。

7.2.3　算法分析

1. 算法的性质

简单地讲，算法就是解题的方法。算法必须满足以下准则：

- 有穷性：一个算法的执行步骤必须是有限的。
- 确定性：算法中的每一个操作步骤的含义必须明确。
- 可行性：算法中的每一个操作步骤都是可以执行的。
- 输入：一个算法一般都要求有一个或多个输入量（个别的算法不要求输入量）。这些输入量是算法所需的初始数据。
- 输出：一个算法至少产生一个输出量，它是算法对输入量的执行结果。

2. 算法的描述

算法可以用文字、符号或图形描述，常用的描述方法有：

- 自然语言：用人的语言描述，该方法易于理解，但容易出现歧义。
- 流程图：用一组特定的几何图形来表示算法，这是最早的算法描述工具。
- N-S 图：用矩形框描述算法，一个算法就是一个矩形框。

- 伪代码：用介于高级语言和人的自然语言之间的文字、符号来描述算法，可以十分容易地转化为高级语言程序。
- PAD 图：全称为问题分析图，使用树型结构描述算法。

3. 算法性能分析

求解同一个问题，可以有多种不同的算法，那么如何衡量一个算法的好坏呢？显然，首先算法应是正确可行的；其次，通常还要考虑如下三方面的问题：

- 执行算法所耗费的时间。
- 执行算法所占用的存储空间。
- 算法是否易于理解、易于编码、易于调试。

7.2.4 线性表

1. 线性表的定义

定义：n（n≥0）个数据元素的有限序列，记作(a_1,…a_{i-1},a_i, a_{i+1},…,a_n)，其中，a_i 是表中的数据元素，n 是表长度。

特点：

- 同一线性表中元素具有相同特性。
- 相邻数据元素之间存在序偶关系。
- 除第一个元素外，其他每一个元素有且仅有一个直接前趋。
- 除最后一个元素外，其他每一个元素有且仅有一个直接后继。

2. 线性表的存储结构

（1）顺序存储结构——顺序表。

1）定义：将线性表中的元素相继存放在一个连续的存储空间中。

存取方式：随机存取。

顺序存储结构示意图如下：

0	1	2	3	4	5
45	89	90	67	48	80

在顺序表中，数据元素按逻辑次序依次放在一组地址连续的存储单元里。由于逻辑上相邻的元素存放在内存的相邻单元里，所以顺序表的逻辑关系蕴含在存储单元的邻接关系中。在高级语言中，可以直接用数组实现。

设顺序表中的每个元素占用 k 个存储单元，索引号为 1 的数据元素 a_1 的内存地址为 LOC(a_1)，则索引号为 i 的数据元素 a_i 的内存地址为：

$$LOC(a_i)=LOC(a_1)+(i-1)*k$$

显然，顺序表中每个元素的存储地址是该元素在表中索引号的线性函数。只要知道某元素在顺序表中的索引号，就可以确定其在内存中的存储位置。所以说，顺序表的存储结构是一种随机存取结构。

顺序表的特点如下：

- 物理上相邻的元素在逻辑上也相邻。
- 可随机存取。

- 存储密度大，空间利用率高。
- 对顺序表进行插入、删除等操作，但运算效率低，需要大量的数据元素移位。

2）顺序表的插入运算。顺序表的插入运算是指在表的第 i 个（1≤i≤n+1）位置上，插入一个新的数据元素 y。若插入位置 i=n+1，即插入到表的末尾，那么只要在表的末尾增加一个单位存储空间即可；但是若 1≤i≤n，则必须将表中第 i 个到第 n 个节点向后移动一个位置，共需移动 n-i+1 个节点。

在有 n 个元素的顺序表的第 i 个位置上插入一个元素需要移动 n-i+1 个元素。如果在第 i 个位置上插入一个元素的概率是 p_i，且在每个位置上插入概率相等，都是 1/(n+1)，则插入时的平均移动次数为：

$$M=\frac{1}{n+1}\sum_{i=1}^{n}(n-i+1)=\frac{n}{2}$$

因此，顺序表上插入运算的平均时间复杂度是 O(n)。

3）删除运算。顺序表的删除运算是指将表的第 i 个（1≤i≤n）数据元素删去。当 i=n 时，即删除表尾节点时，操作较为简单；但 1<i≤n-1 时，则必须将表中第 i+1 个到第 n 个共 n-i 个节点向前移动一个位置。

在有 n 个元素的顺序表的第 i 个位置删除一个元素需要移动 n-i 个元素。如果在第 i 个位置上删除一个元素的概率是 p_i，且在每个位置上删除的概率相等，都是 1/n，则删除的平均移动次数为：

$$M=\frac{1}{n}\sum_{i=1}^{n}(n-i)=\frac{n-1}{2}$$

因此，顺序表上删除运算的平均时间复杂度也是 O(n)。

3. 线性表的链式存储结构——链表

线性表的顺序存储结构在存储数据元素时需要占用连续的存储空间，另外在对顺序表进行某些操作（如插入、删除）时需要移动大量的数据元素。因此提出了线性表的另外一种存储结构——链表。

根据链表的结构，可分为单链表、单循环链表、双向链表、双向循环链表等几种形式，本书中只介绍单链表，其余内容请参阅相关资料。

定义：用一组地址任意的存储单元存放线性表中的数据元素，如图 7-7 所示。

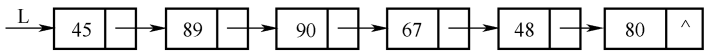


图 7-7　头指针为 L 的单链表

在线性表中，每个元素节点除存储自身的信息外，还要用指针域额外存储一个指向其直接后继的信息（即后继的存储位置——地址）。对链表的访问总是从链表的头部开始，根据每个节点中存储的后继节点的地址信息顺链进行的。当每个节点只有一个指针域时，称为单链表。

单链表中，插入删除一个数据元素，仅仅需要修改该节点的前一个和后一个节点的指针域，非常简便，但要访问表中的任意元素，都必须从头指针开始，顺链查找，无法随机访问。

7.2.5 栈与队列

栈与队列是两种特殊的线性表，即它们的逻辑结构与线性表相同，只是其插入、删除运算仅限制在线性表的一端或两端进行。

1. 栈

栈是一种先进后出的线性表，是一种仅限于在表的一端进行插入和删除运算的线性表，通常插入、删除的这一端称为栈顶，另一端称为栈底。当表中没有元素时称为空栈。栈的逻辑结构如图 7-8 所示。

2. 队列

队列是一种先进先出的线性表，它只允许在线性表的一端进行数据元素的插入操作，而在另一端才能进行数据元素的删除操作。其中，允许插入的一端称为队尾，允许删除的一端称为队头，队列的逻辑结构如图 7-9 所示。

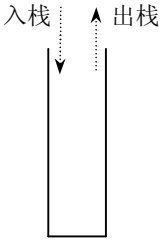


图 7-8 栈

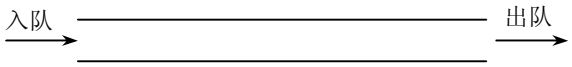


图 7-9 队列

7.2.6 树

1. 树结构

树是有 n（n≥0）个节点的有限集合。如果 n = 0，称为空树；如果 n>0，则：

- 有且仅有一个特定的称之为根（Root）的节点，它只有直接后继，没有直接前趋。
- 当 n>1 时，除根以外的其他节点划分为 m（m>0）个互不相交的有限集 T1、T2、…、Tm，其中每个集合本身又是一棵树，并且称为根的子树（SubTree）。

例如图 7-10 所示即为树结构。

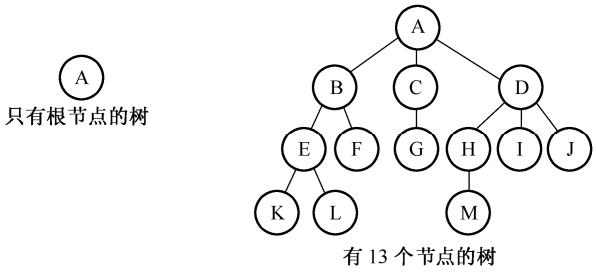


图 7-10 树结构

其中，A 是根；其余节点分成三个互不相交的子集：T1={B,E,F,K,L}、T2={C,G}、T3={D,H,I,J,M}，T1、T2、T3 都是根 A 的子树，且本身也是一棵树。

2. 树结构的重要术语和概念

- 节点：数据元素的内容及指向其子树根的分支统称为节点。
- 节点的度：节点的分支数。
- 终端节点（叶子）：度为 0 的节点。
- 非终端节点：度不为 0 的节点。
- 节点的层次：树中根节点的层次为 1，根节点子树的根为第 2 层，依此类推。
- 树的度：树中所有节点度的最大值。
- 树的深度：树中所有节点层次的最大值。
- 有序树、无序树：如果树中每棵子树从左向右的排列拥有一定的顺序，不得互换，则称为有序树，否则称为无序树。
- 森林：是 m（m≥0）棵互不相交的树的集合。

在树结构中，节点之间的关系又可以用家族关系描述，定义如下：

- 孩子、双亲：节点子树的根称为这个节点的孩子，而这个节点又被称为孩子的双亲。
- 子孙：以某节点为根的子树中的所有节点都被称为是该节点的子孙。
- 祖先：从根节点到该节点路径上的所有节点。
- 兄弟：同一个双亲的孩子之间互为兄弟。
- 堂兄弟：双亲在同一层的节点互为堂兄弟。

7.2.7　二叉树

二叉树结构也是非线性结构中很重要的一类，为有序树，它不是树的特殊结构。在二叉树中，每个节点最多只有两棵树，一个是左子树，一个是右子树。二叉树有 5 种基本形态如图 7-11 所示：它可以是空二叉树，根可以有空的左子树或空的右子树，或左子树、右子树皆为空；其次，二叉树是有序树（如图 7-12 所示），其节点子树要区分为左子树和右子树，即使某节点只有一棵子树的情况下，也要明确指出该子树是左子树还是右子树，而树中则无此区分。

图 7-11　二叉树的 5 种基本形态

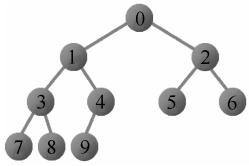


图 7-12　二叉树

二叉树有很多的重要性质：

【性质 1】在二叉树的第 i 层上至多有 2^{i-1} 个节点（i≥1）。[证明用归纳法]

证明：当 i=1 时，只有根节点，$2^{i-1}=2^0=1$。

假设对所有 j，i>j≥1，命题成立，即第 j 层上至多有 2^{j-1} 个节点。

由归纳假设第 i–1 层上至多有 2^{i-2} 个节点。

由于二叉树的每个节点的度至多为 2，故在第 i 层上的最大节点数为第 i–1 层上的最大节点数的 2 倍，即 $2*2^{i-2}=2^{i-1}$。

【性质 2】深度为 k 的二叉树至多有 2^k-1 个节点（k≥1）。

证明：由性质 1 可见，深度为 k 的二叉树的最大节点数为 ：

$$\sum_{i=1}^{K}(\text{第 K 层上的最大节点数})=\sum_{i=1}^{K}2^{i-1}=2^0+2^1+\cdots+2^{K-1}=2^K-1$$

【性质 3】对任何一棵二叉树 T，如果其叶节点数为 n0，度为 2 的节点数为 n2，则 n0=n2+1。

证明：若度为 1 的节点有 n1 个，总节点个数为 n，总边数为 e，则根据二叉树的定义，

$$n=n0+n1+n2 \qquad e=2n2+n1=n-1$$

因此，有 $2n2+n1=n0+n1+n2-1 \Rightarrow n2=n0-1 \Rightarrow n0=n2+1$

满二叉树和完全二叉树是两种特殊形式的二叉树。一棵深度为 k 且有 2^k-1 个节点的二叉树称为满二叉树。满二叉树的特点是每一层上的节点数都达到最大值，2^k-1 个节点是深度为 k 的二叉树所能具有的最大节点个数。

若一个二叉树至多有最下面的两层上的节点可以小于 2，并且最下一层上的节点都集中在该层最左边的位置上，则此二叉树称为完全二叉树。显然，满二叉树是完全二叉树，但完全二叉树不一定是满二叉树。满二叉树、完全二叉树及非完全二叉树如图 7-13 所示

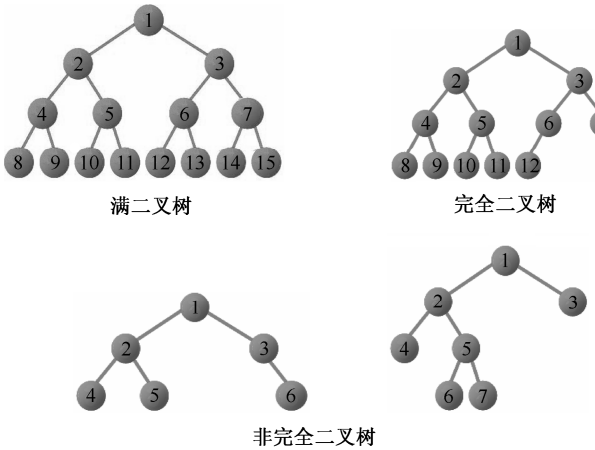


图 7-13　二叉树

【性质 4】具有 n（n≥0）个节点的完全二叉树的深度为 $\lfloor \log_2 n \rfloor+1$

证明：设完全二叉树的深度为 h，则根据性质 2 和完全二叉树的定义有

$$2^{h-1}-1<n\leqslant 2^h-1 \text{ 或 } 2^{h-1}\leqslant n<2^h$$

取对数 $h-1<\log_2 n\leqslant h$，又 h 是整数，因此有 $h=\lfloor \log_2 n \rfloor+1$

【性质 5】如将一棵有 n 个节点的完全二叉树自顶向下，同一层自左向右连续给节点编号 0, 1, 2, …, n–1，则有以下关系：

①若 i=0，则 i 无双亲。

②若 i>0，则 i 的双亲为⌊(i–1)/2⌋。

③若 2*i+1 < n，则 i 的左子女为 2*i+1，若 2*i+2 < n，则 i 的右子女为 2*i+2。

④若节点编号 i 为偶数，且 i!=0，则左兄弟节点为 i–1。

⑤若节点编号 i 为奇数，且 i!=n–1，则右兄弟节点为 i+1。

⑥节点 i 所在层次为⌊log2(i+1)⌋。

二叉树可以采用顺序存储结构和链式存储结构，本书不做介绍。

7.2.8　图结构

1. 图的概念

图是一种重要的、比树更复杂的非线性数据结构。在树结构中，某节点只能与其上层的一个节点（父节点）相联系，并且根节点还没有父节点，每个节点与同一层节点间没有任何横向联系；而在图结构中，数据元素之间的联系是任意的，每个元素可以和其他的元素相联系，从这个意义上来讲，树是一种特殊形式的图。

定义：图是由顶点集合（Vertex）及顶点间的关系集合组成的一种数据结构：

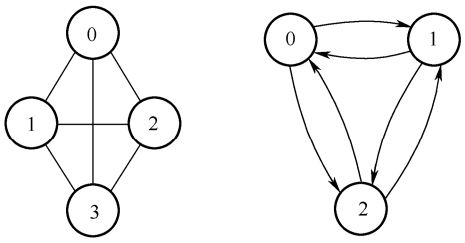
Graph=(V, E)

其中：

V = { x | x ∈ 某个数据对象} 是顶点的有穷非空集合；

E1 = {(x, y) | x, y ∈ V } 或 E2 = {<x, y> | x, y ∈ V && Path (x, y)}。E1 是顶点之间关系的有穷集合，也叫做边（Edge）集合，此时的图称为无向图。E2 表示从 x 到 y 的一条弧，且称 x 为弧尾，y 为弧头，这样的图称为有向图。

从上述定义可以看出，图包括一些点和边，故一个图 G 由点 V(G)和边 E(G)这两个集合组成。如图 7-14 所示，（a）为无向图 G1，（b）为有向图 G2。



（a）无向图　　（b）有向图

图 7-14　图结构

在无向图 G1 中：

G1 =(V1, E1)

V1 = {0,1,2,3}

E1 = {(0,1),(0,2),(0,3),(1,2),(1,3),(2,3)}

在无向图中，边没有方向：(0,3)也可以写成(3,0)。

在有向图 G2 中：

G2 = (V2, E2)

V2 = {0,1,2}

E2 = {<0,1>,<1,0>,<0,2>,<2,0>,<1,2>,<2,1>}

在有向图中，边有方向：<2,4>不能写成<4,2>。

2. 图的术语

- 有向图与无向图：在有向图中，顶点对<x, y>是有序的。在无向图中，顶点对(x, y)是无序的。
- 完全图：若有 n 个顶点的无向图有 n(n-1)/2 条边，则此图为无向完全图。有 n 个顶点的有向图有 n(n-1)条边，则此图为有向完全图。
- 邻接顶点：如果(u, v)是 E(G)中的一条边，则称 u 与 v 互为邻接顶点。
- 子图：设有两个图 G=(V, E)和 G′=(V′, E′)。若 V′⊆ V 且 E′⊆E，则称图 G′是图 G 的子图，如图 7-15 所示。

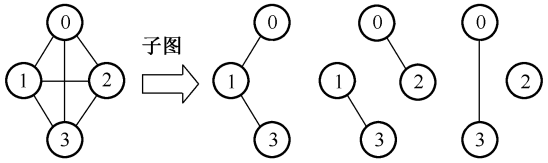


图 7-15　子图

- 权：某些图的边具有与它相关的数，称为权。这种带权图叫做网络。
- 顶点的度：一个顶点 v 的度是与它相关联的边的条数，记作 TD(v)。在有向图（如图 7-16 所示）中，顶点的度等于该顶点的入度与出度之和。

顶点 v 的入度是以 v 为终点的有向边的条数，记作 ID(v)；顶点 v 的出度是以 v 为始点的有向边的条数，记作 OD(v)。

顶点的出度：以顶点 v 为弧尾的弧的数目。

顶点的入度：以顶点 v 为弧头的弧的数目。

顶点的度（TD）=出度（OD）+入度（ID）

TD(B) =OD(B)+2ID(B) = 3

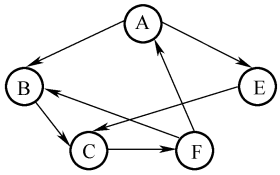


图 7-16　有向图

- 路径：在图 G=(V,E) 中，若从顶点 vi 出发，沿一些边经过一些顶点 vp1, vp2,…, vpm，到达顶点 vj，则称顶点序列(vi, vp1, vp2,…, vpm, vj)为从顶点 vi 到顶点 vj 的路径。它经过的边(vi, vp1)、(vp1, vp2)、…、(vpm, vj) 应是属于 E 的边。
- 路径长度：非带权图的路径长度是指此路径上边的条数。带权图的路径长度是指路径上各边的权之和。

- 简单路径：若路径上各顶点 v1,v2,...,vm 均不互相重复，则称这样的路径为简单路径。
- 简单回路：若路径上第一个顶点 v1 与最后一个顶点 vm 重合，则称这样的路径为回路或环。
- 连通图与连通分量：在无向图中，若从顶点 v1 到顶点 v2 有路径，则称顶点 v1 与 v2 是连通的。如果图中任意一对顶点都是连通的，则称此图是连通图。非连通图的极大连通子图叫做连通分量。
- 强连通图与强连通分量：在有向图中，若对于每一对顶点 vi 和 vj，都存在一条从 vi 到 vj 和从 vj 到 vi 的路径，则称此图是强连通图。非强连通图的极大强连通子图叫做强连通分量。
- 生成树：假设一个连通图有 n 个顶点和 e 条边，其中 n-1 条边和 n 个顶点构成一个极小连通子图，称该极小连通子图为此连通图的生成树。在极小连通子图中增加一条边，则一定有环。在极小连通子图中去掉一条边，则成为非连通图。

3. 图的存储结构

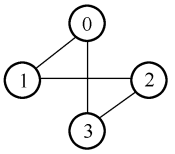
本书中仅介绍两种图的邻接矩阵和邻接表两种存储结构。

（1）邻接矩阵。在图的邻接矩阵表示中，有一个记录各个顶点信息的顶点表，还有一个表示各个顶点之间关系的邻接矩阵，如图 7-17 所示。

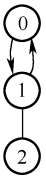
设图 A= (V,E)是一个有 n 个顶点的图，图的邻接矩阵是一个二维数组 A.Edge[n][n]，定义：

$$A.Edge[i][j]\begin{cases}1，若<i,j>\in E 或(i,j)\in E\\0，否则\end{cases}$$

基本思想：一个图由顶点集合、边集合（顶点偶对集合、反映顶点间关系）组成，因此，图的计算机存储只要解决两集合的表示即可。



$$A.Edge=\begin{bmatrix}0&1&0&1\\1&0&1&0\\0&1&0&1\\1&0&1&0\end{bmatrix}$$



$$A.Edge=\begin{bmatrix}0&1&0\\1&0&1\\0&0&0\end{bmatrix}$$

图 7-17　邻接矩阵

无向图的邻接矩阵是对称的；有向图的邻接矩阵可能是不对称的。

在无向图中，统计第 i 行（列）1 的个数可得顶点 i 的度。

在有向图中，统计第 i 行 1 的个数可得顶点 i 的出度，统计第 j 列 1 的个数可得顶点 j 的入度。

邻接矩阵是一种静态存储结构，当矩阵大小与顶点个数相符，与边（弧）数目无关，易产生稀疏矩阵，造成空间浪费。

（2）邻接表。为了克服邻接矩阵的不足，采用动态的链式存储结构来保存图信息，这就

是邻接表，如图 7-18 和图 7-19 所示为无向图和有向图的邻接表。

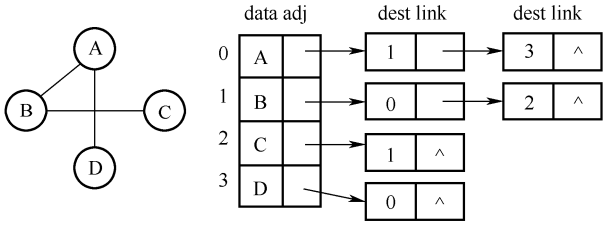


图 7-18　无向图的邻接表

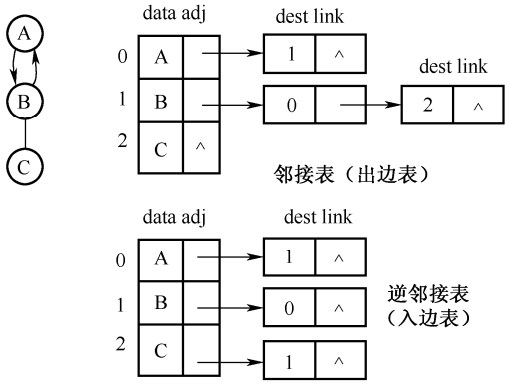


图 7-19　有向图的邻接表和逆邻接表

基本思想：

- 对每一个顶点建立一个单链表。
- 第 i 个单链表中存放顶点 i 的所有邻接顶点。
- 第 i 个单链表的头节点中，存放顶点 i 的信息 Vi。

7.2.9　线性表的查找

在数据结构中，数据的基本单位是数据元素，数据元素通常表现为记录、节点、定点等。一个数据元素由若干个数据项（或称为域）组成，用以区别数据元素中各个数据元素的数据项称为关键字（Key）。

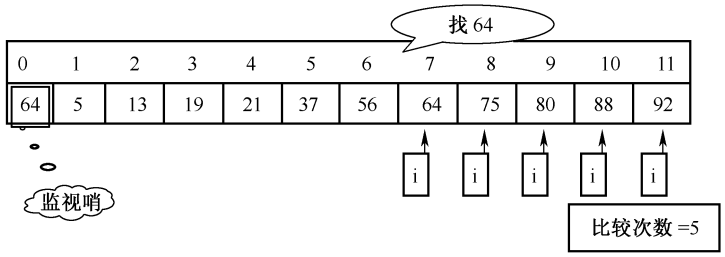
所谓查找（Search），又称检索，就是在一个含有 n 个数据元素的集合中，根据一个给定的值 k，找出其关键字的值等于给定值 k 的数据元素。若找到，则查找成功，输出该元素或该元素在集合中的位置；否则查找失败，此时或者输出查找失败信息，或者将给定值作为数据元素插入到集合中适当的位置。

线性表的查找，常用的有如下 3 种方法：顺序查找、二分法查找、分块查找。

1. 顺序查找

从第 1 个或第 n 个数据元素开始，逐个把数据元素的关键字值与给定值比较，若找到某数据元素的关键字值与给定值相等，则查找成功；若遍历整个线性表都未找到，则查找失败。

【例 7-4】在所给的线性表中查找：



比较次数：
查找第 n 个元素：　1
查找第 n-1 个元素：　2
……
查找第 1 个元素：　n
查找第 i 个元素：　n+1-i
查找失败：　n+1

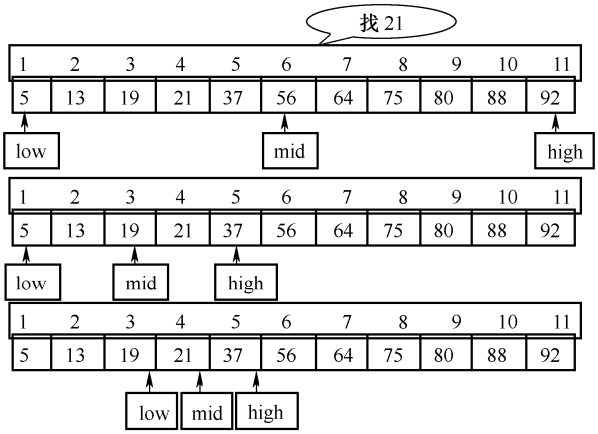
容易推导出，在长度为 n 的线性表中进行查找的平均查找长度为：(n+1)/2。

2. 二分法查找

顺序表的查找算法简单，但平均查找长度较大，不适用于表长较大的查找表。若以有序表表示静态查找表，则查找过程可以基于“折半”进行，这就是所谓的二分法查找。

二分法查找的基本思路是：由于查找表中的数据元素按关键字有序（假设为增序），则查找时不必逐个顺序比较，而先与中间数据元素的关键字比较。若相等，则查找成功；若不等，即把给定值与中间数据元素的关键字值比较，若给定值小于中间数据元素的关键字值，则在前半部分进行二分查找，否则在后半部分进行二分查找。这样，每进行一次比较，就将查找区间缩短为原来的一半。

【例 7-5】Key =21 的查找过程。



low 指示查找区间的下界；high 指示查找区间的上界；mid = (low+high)/2。

在长度为 n 的有序顺序表中进行二分查找的查找次数不超过⌊log2(n+1)⌋”次（其中⌊⌋代表取整）。因此，二分法查找具有效率高的特点。

3. 分块查找

分块查找时介于顺序查找与二分法查找之间的一种查找方法，又称索引顺序查找。它的基本思想是：

分块：将数据划分为若干数据块，数据在块内无序，但块间有序，也就是说，第一块内的最大数据比后继所有数据都小（假设按数据递增有序），后面的每一块内的所有数据都大于它前面的所有块的最大数据，同时又小于后继所有块内的所有数据。

查找：分两步进行。

（1）块间：建立一个各块最大关键字值表，将待查数据在该表中按二分法或顺序查找进行，通过块间查找确定数据所在的块。用二分法可以提高块间查找的效率。

（2）块内：在块内按顺序查找方式直接查找元素。

由于块间查找用了二分法，所以整个算法的效率要比顺序查找高，但事先要将数据进行分块，这在一定程度上增加了额外的时间开销。

【例 7-6】分块查找。待查序列为：22,12,13,8,9,20,33,42,44,38,24,48,60,58,74,57,86,53，查找关键字值为 38 的元素。

①先将该序列等分为三块：{22,12,13,8,9,20}　{33,42,44,38,24,48,60}　{58,74,57,86,53}

②建立一个顺序的各块最大关键字值表：{22,48,86}

③根据待查的关键字在各块最大关键字值表中进行查找，确定数据可能在的块。在{22,48,86}中查找 38，由于 22<38<86，确定数据在第二块中。

④在第二块中顺序查找关键字为 38 的元素，找到。

具体过程如图 7-20 所示。

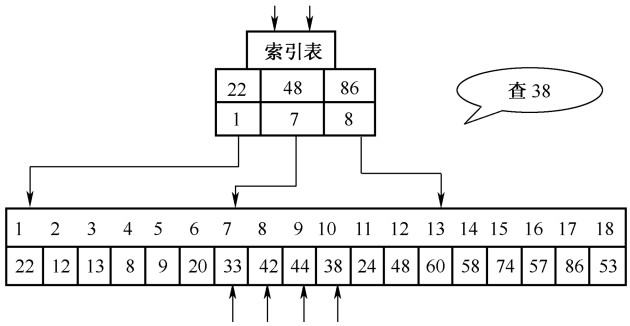


图 7-20　分块查找

7.2.10　内排序

排序又称为分类，它是数据处理中经常使用的一种运算，是将一组数据元素（记录）按其排序码进行递增或递减的运算操作。排序分内排序和外排序。

- 内排序：整个排序运算在内存中进行。
- 外排序：对外存储器中的数据进行排序操作。

1. 插入法排序

把 N 个数据元素的序列分成两部分，一个是已排好序的有序部分，另一个是未排好序的未排序部分；把未排好序的元素逐个与已排好序的元素比较，并插入到有序部分的合适位置，最后得到一个新的有序序列。

【例 7-7】插入法排序（线性表长度 n=8）。

初始序列	[49]	38	65	97	76	7	27	49
第一轮（1）	[38	49]	65	97	76	7	27	49
第二轮（2）	[38	49	65]	97	76	7	27	49
第三轮（3）	[38	49	65	97]	76	7	27	49
第四轮（4）	[38	49	65	79	97]	7	27	49
第五轮（5）	[7	38	49	65	76	97]	7	49
第六轮（6）	[7	27	38	49	65	76	97]	49
第七轮（7）	[7	27	38	49	49	65	76	97]

2. 选择排序

每一轮排序中，将第 i 个元素与从序列第 i+1 到 n 的 n-i+1（i=1,2,3,n-1）个元素中选出的值最小的一个元素进行比较，若该最小元素比第 i 个元素小，则两者交换。从 i=1 开始，重复此过程，直到 i=n-1。

简单地说，通过交换位置，选最小的放在第一，次小的放在第二，依此类推，直到元素序列的最后为止。

【例 7-8】选择排序（线性表长度 n=8）。

初始序列	49	{38	65	97	76	7	27	49}
第一轮（1）	7	38	{65	97	76	49	27	49}
第二轮（2）	7	27	65	{97	76	49	38	49}
第三轮（3）	7	27	38	97	{76	49	65	49}
第四轮（4）	7	27	38	49	76	{97	65	49}
第五轮（5）	7	27	38	49	49	97	{65	76}
第六轮（6）	7	27	38	49	49	65	97	{76}
第七轮（7）	7	27	38	49	49	65	76	97

以第一、二轮为例：49 与 7 比较后互换，38 与 27 比较后互换……以后各轮依此类推。

注意：此序列中有两个元素具有相同关键字 49，经过排序，原来排在后面的一个 49 反而排在了前面，这种相同关键字值经过排序交换位置的现象，称为不稳定，这种排序方法被称为是不稳定的。选择排序就是一种不稳定的排序方法。

3. 冒泡排序

冒泡排序需要进行 n-1 轮排序过程。

第一轮：从 a_i 开始，两两比较 a_1,a_{i+1}（i=1,2,⋯,n–1）的大小，若 $a_1>a_{i+1}$，则交换 a_i 与 a_{i+1}。当第一轮完成时，最大元素将被交换到最后一位（第 n 位）。

第二轮：仍然从 a_1 开始，两两比较 a_i、a_{i+1}（i=1,2,⋯,n–2）的大小，注意此时的处理范围是从第一轮的整个序列 n 个数据元素比较 n-1 次（i=1,2,⋯,n–1），变成了 n–1 个数据元素比较 n–2 次（i=1,2,⋯,n–2）。当第二轮完成时，最大元素将被交换到次后一位（第 n–1 位）。

第n-1轮：只需要比较最初两个元素，就完成了整个线性表的排序。

【例7-7】冒泡排序过程（线性表长度n=7）。

初始状态：	[65	96	76	7	27	49	58]
第一轮（i=1,…,6）	[65	97	7	27	49	58]	97
第二轮（i=1,…,5）	[65	7	27	49	58]	76	97
第三轮（i=1,…,4）	[7	27	49	58]	65	76	97
第四轮（i=1,…,3）	[7	27	49]	58	65	76	97
第五轮（i=1,…,2）	[7	27]	49	58	65	76	97
第六轮（i=1）	[7]	27	49	58	65	76	97

此外，还有归并排序和快速排序两种方法。

归并是将两个或两个以上的有序表组合成一个新的有序表。将每个元素看成一个长度为 1 的子序列，把相邻子序列两两合并，得到一个新的子序列，如此重复，最后得到长度为 n 的一个新的有序序列。

快速排序是冒泡排序的改进，平均速度较快。基本思想如下：

- 任选一个元素 R_i（一般为第一个）作为标准。
- 调整各元素的位置，排在 R_i 前的元素的排序码都小于 R_i，而排在 R_i 后的元素的排序码都大于 R_i。本过程称为一次快排，由此确定了 R_i 在有序序列中的最后位置，同时将剩余元素分为两个子序列。
- 对两个子序列分别进行快速排序，又确定了两个元素在有序序列中的位置，并将剩余元素分为 4 个子序列。
- 重复此过程，直到各子序列的长度都为 1，排序结束。

4. 几种排序比较

（1）稳定性比较。

稳定的排序方法有：插入、归并、冒泡排序；不稳定的排序方法有：选择排序、快速排序。

（2）平均综合情况。归并排序速度较快；插入、冒泡排序速度较慢。

总之，各种排序法各有其优缺点。其选用依据如下：

- 其数据规模 n 大，内存允许，要求稳定，则选用归并排序。
- 其数据规模 n 较小，有稳定要求，则选用插入排序。
- 其数据规模 n 较小，对稳定不要求，则选用选择排序。
- 其数据规模 n 大，内存允许，对稳定不要求，则选用快速排序。

7.3 软件工程

7.3.1 概述

自 20 世纪 40 年代计算机问世以来，如何编制符合要求的程序一直是人们追求的目标。并且，随着计算机应用领域的扩大，人们对软件的需求量剧增，对软件的正确性提出了更高的要求，并迫切地需要缩短软件生产周期。但是，当时的软件编制还只是一种手工活动，过多地依赖于程序员的个人能力和技巧，这就导致了软件的生产周期长，可靠性及可维护性也很差。软件开发远远落后于硬件发展，远远满足不了社会的需求，从而爆发了一场“软件危机”。

软件工程作为一门学科的出现给软件界带来了一场变革，众多的研究者和实践者投身于软件工程领域，并取得了丰硕成果。虽然离预期目标仍有很大距离，并未能从根本上克服软件危机，但是软件工程的思想毕竟给人们带来希望的曙光。

软件工程的研究除计算机软件本身外，还涉及众多其他的领域，如管理科学、心理学、经济学、人体工程学等，因此，它是一门综合性学科。粗略来说，软件工程学科可划分为两大方面：一方面是基础性理论研究，主要目标是用形式化技术解决软件生产中所遇到的问题，如需求规格的描述、规格到系统的转换、系统测试、维护及理解等，主要为解决“做什么”的描述手段问题；另一方面是工程化技术研究，总结软件开发过程的规律，探讨软件开发过程的工程化因素、方法及工程支持，解决“怎么做”的问题。20 世纪 80 年代以来，计算机辅助软件工程（CASE）的研究已成为软件工程领域中的热点，其中包括 CASE 工具和 CASE 环境的研究，旨在应用计算机支持软件开发过程，改进软件开发行为，为开发人员提供软件开发平台和环境，以提高软件生产率并改善软件产品的质量。

7.3.2　软件

1. 软件的概念

软件是一系列按照特定顺序组织的计算机数据和指令的集合。一般来讲软件被划分为系统软件、应用软件和介于这两者之间的中间件。其中系统软件为计算机使用提供最基本的功能，但是并不针对某一特定应用领域。而应用软件则恰好相反，不同的应用软件根据用户和所服务的领域提供不同的功能。

软件并不只是包括可以在计算机上运行的程序，与这些程序相关的文档一般也被认为是软件的一部分。简单地说软件就是程序加文档的集合体。

2. 软件开发经历的三个阶段

第一个阶段是 20 世纪 50 年代到 60 年代，是程序设计阶段，基本是个体手工劳动的生产方式。这个时期，一个程序是为一个特定的目的而编制的，软件的通用性很有限的，软件往往带有强烈的个人色彩。早期的软件开发没有什么系统的方法可以遵循，软件设计是在某个人的头脑中完成的一个隐蔽的过程。而且，除了源代码往往没有软件说明书等文档，因此这个时期尚无软件的概念，基本上只有程序、程序设计概念，不重视程序设计方法，主要是用于科学计算，规模很小，采用简单的工具（基本上采用低级语言），硬件的存储容量小、运行可靠性差。

第二阶段是 20 世纪 60 年代到 70 年代，是软件设计阶段，为小组合作生产方式。在这一时期软件开始作为一种产品被广泛使用，出现了“软件作坊”。这个阶段，基本采用高级语言开发工具，开始提出结构化方法。硬件的速度、容量、工作可靠性有明显提高，而且硬件的价格降低。人们开始使用软件产品（可购买），从而建立了软件的概念。程序员数量猛增，但是开发技术没有新的突破，软件开发的方法基本上仍然沿用早期的个体化软件开发方式，软件需求日趋复杂，维护的难度越来越大，开发成本令人吃惊地高，开发人员的开发技术不适应规模大、结构复杂的软件开发，失败的项目越来越多。

第三阶段是从 20 世纪 70 年代至今，为软件工程时代，是工程化的生产方式。这个阶段的硬件向超高速、大容量、微型化以及网络化方向发展，第三、四代语言出现。数据库、开发工具、开发环境、网络、分布式、面向对象技术等工具和方法都得到应用。软件开发技术有了很大进步，但未能获得突破性进展，软件开发技术的进步一直未能满足发展的要求。软件的数量急剧膨胀，一些复杂的、大型的软件开发项目提了出来，在那个时代，很多的软件最后都得到了一个悲惨的结局。很多的软件项目开发时间大大超出了规划的时间表，一些项目导致了财产的流失，甚至某些软件导致了人员伤亡。同时软件开发人员也发现软件开发的难度越来越大，在软件开发中遇到的问题找不到解决的办法，使问题积累起来，形成了尖锐的矛盾，失败的软件开发项目却屡见不鲜，因而导致了软件危机。

7.3.3　软件危机

软件危机指的是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。概括来说，软件危机包含两方面问题：

（1）如何开发软件，以满足不断增长、日趋复杂的需求。

（2）如何维护数量不断膨胀的软件产品。

落后的软件生产方式无法满足迅速增长的计算机软件需求，从而导致软件开发与维护过

程中出现一系列严重问题的现象。

由于软件本身是一个逻辑实体，而非一个物理实体，因此软件是非实物性和不可见性的。而软件开发本身又是一个“思考”的过程，很难进行管理。开发人员以“手工作坊”的开发方式来开发软件，完全是按照各自的喜好和习惯进行的，没有统一的标准和规范可以遵循。因此，在软件的开发过程中，人们遇到了许多困难。有的软件开发彻底失败了，有的软件虽然开发出来了，但运行的结果极不理想，如程序中包含着许多错误，每次错误修改之后又会有一批新的错误取而代之。这些软件有的因无法维护而不能满足用户的新要求，最终失败了；有的虽然完成了，但是比原计划推迟了好几年，而且成本上大大超出了预算。

软件开发的高成本与软件产品的低质量之间的尖锐矛盾，终于导致了软件危机的发生。具体地说，软件危机主要有以下几方面的表现：

- 软件的复杂性越来越高，“手工作坊”式的软件开发方式已经无法满足要求。
- 对软件成本和进度统计不准，费用实际超过预算。
- 开发周期过长。
- 软件质量难以保证，常被怀疑。
- 缺乏良好的软件文档。
- 现有的软件极难维护。
- 软件开发效率远跟不上计算机的发展要求。
- 用户往往对软件不满意。

为摆脱软件危机，1968 年北大西洋公约组织的计算机科学家在联邦德国召开的国际学术会议上第一次提出了“软件危机”（Software Crisis）这个名词。同时，讨论和制定摆脱“软件危机”的对策。在那次会议上第一次提出了软件工程（Software Engineering）这个概念，从此一门新兴的工程学科——软件工程学——为研究和克服软件危机应运而生。

7.3.4 软件工程学概述

（1）软件工程学的研究对象。软件工程学研究如何应用一些科学理论和工程技术来指导软件系统的开发与维护，使其成为一门严格的工程学科。

（2）软件工程学的基本目标。软件工程学的基本目标在于研究一套科学的工程方法，设计一套方便实用的工程系统，以达到在软件研制生产中的投资少、效率高、优质的目的。

（3）软件工程学的三要素：方法、工具和管理。

（4）软件生命周期。

一个软件项目从问题提出、定义、开发、使用、维护，直至被废弃，要经历一个漫长的时期，通常把这个时期称为软件生命周期（Software Life Cycle）。

软件工程是研究软件的研制和维护的规律、方法和技术的学科。贯穿于这一学科的基本线索是软件生命周期（也叫软件生存周期），它将告诉软件研制者与维护者“什么时候做什么、怎样做”。

7.3.5 软件生存周期

同任何事物一样，一个软件产品或软件系统也要经历孕育、诞生、成长、成熟、衰亡等阶段，一般称为软件生存周期（软件生命周期）。把整个软件生存周期划分为若干阶段，使得

每个阶段有明确的任务，使规模大、结构复杂和管理复杂的软件开发变得容易控制和管理。通常，软件生存周期包括可行性分析与开发项目计划、需求分析、设计（概要设计和详细设计）、编码、测试、维护等活动，可以将这些活动以适当的方式分配到不同的阶段去完成。

软件生命周期是软件从产生直到报废的生命周期，周期内有问题定义、可行性分析、总体描述、系统设计、编码、调试和测试、验收与运行、维护升级到废弃等阶段，这种按时间分程的思想方法是软件工程中的一种思想原则，即按部就班、逐步推进，每个阶段都要有定义、工作、审查、形成文档以供交流或备查，以提高软件的质量。但随着新的面向对象的设计方法和技术的成熟，软件生命周期设计方法的指导意义正在逐步减少。

软件生命周期的 6 个阶段：

（1）问题的定义及规划。此阶段是软件开发方与需求方共同讨论，主要确定软件的开发目标及其可行性。

（2）需求分析。在确定软件开发可行的情况下，对软件需要实现的各个功能进行详细分析。需求分析阶段是一个很重要的阶段，这一阶段做得好，将为整个软件开发项目的成功打下良好的基础。“唯一不变的是变化本身。”，同样需求也是在整个软件开发过程中不断变化和深入的，因此我们必须制定需求变更计划来应付这种变化，以保证整个项目的顺利进行。

（3）软件设计。此阶段主要根据需求分析的结果对整个软件系统进行设计，如系统框架设计、数据库设计等。软件设计一般分为总体设计和详细设计。好的软件设计将为软件程序编写打下良好的基础。

（4）程序编码。此阶段是将软件设计的结果转换成计算机可运行的程序代码。在程序编码中必须要制定统一、符合标准的编写规范，以保证程序的可读性、易维护性，提高程序的运行效率。

（5）软件测试。在软件设计完成后要经过严密的测试，以发现软件在整个设计过程中存在的问题并加以纠正。整个测试过程分单元测试、组装测试、系统测试三个阶段进行。测试的方法主要有白盒测试和黑盒测试两种。在测试过程中需要建立详细的测试计划并严格按照测试计划进行测试，以减少测试的随意性。

（6）运行维护。软件维护是软件生命周期中持续时间最长的阶段。在软件开发完成并投入使用后，由于多方面的原因，软件不能继续适应用户的要求。要延续软件的使用寿命，就必须对软件进行维护。软件的维护包括纠错性维护和改进性维护两个方面。

（1）瀑布模型。

1970 年温斯顿• 罗伊斯（Winston Royce）提出了著名的“瀑布模型”，直到 20 世纪 80 年代早期，它一直是唯一被广泛采用的软件开发模型。

瀑布模型的核心思想是按工序将问题化简，将功能的实现与设计分开，便于分工协作，即采用结构化的分析与设计方法将逻辑实现与物理实现分开。将软件生命周期划分为制定计划、需求分析、软件设计、程序编写、软件测试和运行维护 6 个基本活动，并且规定了它们自上而下、相互衔接的固定次序，如同瀑布流水，逐级下落。从本质来讲，它是一个软件开发架构，开发过程是通过一系列阶段顺序展开的，从系统需求分析开始直到产品发布和维护，每个阶段都会产生循环反馈，因此，如果有信息未被覆盖或者发现了问题，那么最好“返回”上一个阶段并进行适当的修改，开发进程从一个阶段“流动”到下一个阶段，这也是瀑布开发名称的由来。

瀑布模型是最早出现的软件开发模型，在软件工程中占有重要的地位，它提供了软件开

发的基本框架。其过程是从上一项活动接收该项活动的工作对象作为输入，利用这一输入实施该项活动应完成的内容，给出该项活动的工作成果，并作为输出传给下一项活动。同时评审该项活动的实施，若确认，则继续下一项活动；否则返回前面，甚至更前面的活动。对于经常变化的项目而言，瀑布模型毫无价值（采用瀑布模型的软件过程如图 7-21 所示）。

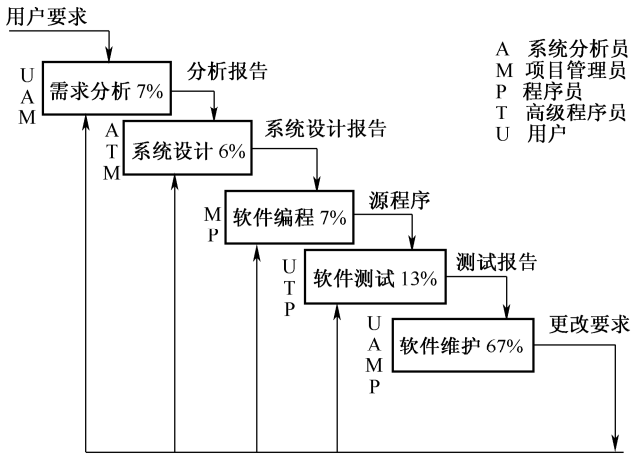


图 7-21　瀑布模型示意图

1）瀑布模型有以下优点：

①为项目提供了按阶段划分的检查点。

②当前一阶段完成后，你只需要去关注后续阶段。

③可在迭代模型中应用瀑布模型。

2）瀑布模型有以下缺点：

①在项目各个阶段之间极少有反馈。

②只有在项目生命周期的后期才能看到结果。

③通过过多的强制完成日期和里程碑来跟踪各个项目阶段。

（2）快速原型。

快速原型模型需要迅速建造一个可以运行的软件原型，以便理解和澄清问题，使开发人员与用户达成共识，最终在确定的客户需求基础上开发客户满意的软件产品。快速原型模型允许在需求分析阶段对软件的需求进行初步而非完全的分析和定义，快速设计开发出软件系统的原型，该原型向用户展示待开发软件的全部或部分功能和性能；用户对该原型进行测试评定，给出具体改进意见以丰富细化软件需求；开发人员据此对软件进行修改完善，直至用户满意认可之后，进行软件的完整实现及测试、维护。

1）快速原型模型的优点：克服瀑布模型的缺点，减少由于软件需求不明确带来的开发风险。

2）快速原形模型的缺点：所选用的开发技术和工具不一定符合主流的发展；快速建立起来的系统结构加上连续的修改可能会导致产品质量低下。

1. 软件定义（系统设计）阶段

软件定义，又称为系统分析。这个时期的任务是确定软件开发的总体目标，确定软件开发工程的可行性，确定实现工程目标应该采用的策略和必须完成的功能，估计完成该项工程需

要的资源和成本，制定出工程进度表。

软件定义可进一步划分为 3 个阶段，即问题定义、可行性研究和需求分析。

（1）问题定义。

问题定义阶段必须考虑的问题是“做什么”。正确理解用户的真正需求，是系统开发成功的必要条件。软件开发成员与用户之间的沟通，必须通过系统分析员对用户进行访问调查，扼要地写出对问题的理解，并在有用户参与的会议上认真讨论，澄清含糊不清的地方，改正理解不正确的地方，最后得到一份双方都认可的文档。在文档中，系统分析员要写明问题的性质、工程的预期目标以及工程的规模。

问题定义阶段是软件生命周期中最短的阶段，一般不超过 3 天。

（2）可行性研究。

可行性研究要研究问题的范围，并探索这个问题是否值得去解决，以及是否有可行的解决办法，可行性研究的结果为部门负责人决定是否继续这项工程提供了重要依据。

可行性论证的内容包括：技术可行性、经济可行性、操作可行性。

可行性论证是分析员在收集资料的基础上，经过分析，明确工程软件项目的目标、问题域、主要功能和性能要求，确定应用软件的支撑环境以及经济、制作和时间限制等方面的约束条件，并用高层逻辑模式（通常用数据流图）对各种可能方案进行可行性分析及成本/效益分析。如果该项目在技术和经济上均可行，可明确地写出开发任务的全面要求和细节，形成软件计划任务书，作为本阶段的工作总结。

（3）需求分析。

需求分析即系统分析，通常采用系统模型定义系统。在可行性分析的基础上，需求分析的主要任务是：明确用户要求软件系统必须满足的所有功能、性能和限制，也就是解决软件“做什么的问题”。系统分析员和用户密切配合，充分交流信息，得出经过用户确认的系统逻辑模型。

需求分析只是方案的设计。在这一阶段的工作中，为清晰地解释问题的本质，往往略去具体问题中的一些次要因素，只需抽象出反映该问题的系统逻辑模型。

系统逻辑模型是以后设计和实现目标系统的基础，必须准确而完整地体现用户的要求。

2. 软件开发阶段

软件开发，是实现前一个时期定义的软件。它包含 3 个阶段：总体设计、详细设计、编码。

（1）总体设计。总体设计也叫概要设计或初步设计。这个阶段必须回答的是“概括地说，应该如何解决这个问题”。最后得到软件说明书。

总体设计的目标是结果化分析的成果，由数据模型、功能模型、行为模型描述的软件需求，按一定的设计方法，完成数据设计、体系结构设计、接口设计和过程设计。

总体设计应遵循的一条主要原则就是程序模块化的原则。

（2）详细设计。总体设计阶段以比较抽象、概括的方式提出了问题的解决方法，而详细设计阶段的任务是把解法具体化，也就是回答“应该怎样具体地实现这个系统”。

详细设计亦即模块设计。它是在算法设计和结构设计的基础上，针对每个模块的功能、接口和算法定义，设计模块内部的算法过程及程序的逻辑结构，并编写模板设计说明。

（3）编码设计。这个阶段的任务是，根据详细设计的结果，选择一种适合的程序设计语言，把详细设计的结果翻译成程序源代码。

3. 软件测试

为了发现程序中的错误而执行程序的过程。测试的目的是尽可能地揭露和发现程序中隐藏的错误，好的测试方案是尽可能发现未发现的错误的测试方案；成功的测试是发现了至今为止尚未发现的错误的测试。因此，一般不由软件编写者测试程序，而由其他人组成的一个测试小组来进行。而且，就是通过了最严密的测试，仍可能存在未发现的错误。总之，测试只能发现错误，不能证明程序中没有错误。

（1）基本测试方法。

黑盒测试（功能测试）：是在程序接口进行的测试，根据规格说明书检查接口，而不考虑程序的内部结构和实现过程。

白盒测试（结构测试）：按照程序的内部逻辑实现来测试程序，了解程序的每条通路是否按预定要求正确实现。

（2）测试策略。

测试过程必须分步进行。

单元测试：着重测试每个单独模块，以确保其作为一个单元功能是正确的。单元测试大量使用白盒测试技术，检查模块的控制结构。

集成测试：把模块装配（集成）为一个完整的软件包，在装配的同时进行测试。集成测试主要使用黑盒色测试技术，要同时解决程序验证和构造两个问题。

4. 运行和维护

经过测试后的软件即可投入运行，在运行的过程中进行维护。软件维护的任务是使软件能够持久地满足用户的需求。

维护可分为 4 类：纠错性维护、适应性维护、完善性维护和预防性维护。

7.4 操作系统

7.4.1 操作系统的概念

操作系统（Operating System，OS）是计算机系统中最重要的系统软件，协调管理计算机的软硬件资源，以提高硬件的利用率；是用户与计算机之间的接口，为用户使用计算机提供操作的平台和环境，以便用户无需了解计算机硬件或系统软件的有关细节就能方便地使用计算机。如图 7-22 所示，从用户和资源两个方面描述了操作系统所处的地位。

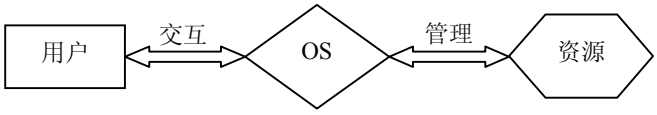


图 7-22 操作系统所处的地位

7.4.2 操作系统的类型

1. 批处理操作系统

基本特点是：多道，即允许外存中的多个作业队列进入内存，由 CPU 调度各作业交替运

行；成批，即作业的装入，运行及结果输出等都由系统自动实现，不允许用户干预。

2. 分时操作系统

如果说，推动多道批处理系统形成和发展的主要动力是提高资源利用率和系统吞吐量，那么，推动分时系统形成和发展的主要动力则是用户的需求。或者说，分时系统是为了满足用户需求所形成的一种新型 OS。它与多道批处理系统之间有着截然不同的性能差别。用户的需求具体表现在以下几个方面：

（1）人机交互。

（2）共享主机。

（3）便于用户上机。

多个用户对系统的资源进行时间上的分享（如图 7-23 所示），具体实现是将 CPU 的时间划分成一个一个的时间片，按某种策略分配给各个用户的进程使用，每个用户都似乎独占了 CPU 一样。

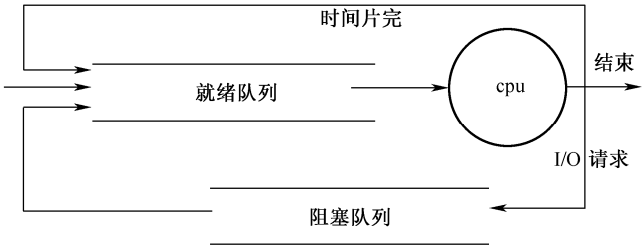


图 7-23　分时操作系统模型

其特点是：

多路性：同时响应多个终端的服务请求。

交互性：各终端用户可以通过终端、键盘、鼠标等输入/输出设备与系统交互，控制作业的运行，得到系统的服务。

独立性：用户各自独立地使用计算机。

及时性：对用户的任务及时接收和处理。

3. 实时操作系统

所谓“实时”，是表示“及时”，而实时系统（Real-Time System）是指系统能及时（或即时）响应外部事件的请求，在规定的时间内完成对该事件的处理，并控制所有实时任务协调一致地运行。

其特点是：

及时响应：实时任务必须在指定的时限内响应完成。

交互功能：实时系统仍然要求满足用户的实时交互要求。

高可靠性：实时系统往往用于工业、国防等对实时性要求高的场合，如温度控制、卫星发射等。因此，系统的高可靠性远比系统性能更重要。

4. 网络操作系统

计算机网络是将地理上分布的各数据处理系统或计算机系统互连，实现资源共享、信息交换和协作完成任务。网络操作系统是管理计算机网络，为用户提供网络资源共享、系统安全

及各种网络应用服务的操作系统。网络操作系统的基本特点是处理上的分布，也就是功能和任务上的分布以及系统管理的分布。网络对用户是不透明的，用户能感知并选择访问其中某节点上的资源。

5. 分布式操作系统

分布式系统是将地理上分布的各数据处理系统或计算机系统互连，实现资源共享、信息交换和协作完成任务。分布式系统要求一个统一的操作系统，以实现系统操作的统一性，其基本特点是处理上的分布，也就是功能和任务上的分布及系统管理的统一。分布式系统对用户是透明的，用户面对的是一个统一的操作系统，他无法也不必知道系统的内部实现。

7.4.3 操作系统的基本特征

现代操作系统普遍采用多道程序设计技术，所谓多道程序设计技术是为了提高计算机软硬件资源的利用率，允许在内存中同时安排多个作业（用户软件程序），各个作业共享系统资源，以并发的方式各自向前推进。由于多道程序共存于内存并且交替执行，有的程序正在计算，有的程序正在输入输出，于是 CPU 利用率、I/O 设备利用率以及内存利用率都大大提高。

由于多道程序设计技术的引入，使得操作系统具有如下基本特征：

（1）并发性。并行性和并发性是既相似又有区别的两个概念，并行性是指两个或多个事件在同一时刻发生；而并发性是指两个或多个事件在同一时间间隔内发生。在多道程序环境下，并发性是指在一段时间内，宏观上有多个程序在同时运行，但在单处理机系统中，每一时刻却仅能有一道程序执行，故微观上这些程序只能是分时地交替执行。倘若在计算机系统中有多个处理机，则这些可以并发执行的程序便可被分配到多个处理机上，实现并行执行，即利用每个处理机来处理一个可并发执行的程序，这样，多个程序便可同时执行。

（2）共享性。在操作系统环境下，所谓共享是指系统中的资源可供内存中多个并发执行的进程（线程）共同使用。由于资源属性的不同，进程对资源共享的方式也不同，目前主要有以下两种资源共享方式：

- 互斥共享：一段时间内只允许一个用户使用的资源，如打印机。
- 并发访问：一段时间内可以有多个进程同时使用某个资源。

并发与共享是操作系统最基本的两个特征，互为存在条件。

（3）虚拟性。操作系统中的所谓“虚拟”，是指通过某种技术把一个物理实体变为若干个逻辑上的对应物。物理实体（前者）是实的，即实际存在的；而后者是虚的，是用户感觉上的东西。相应地，用于实现虚拟的技术，称为虚拟技术。在 OS 中利用了多种虚拟技术，分别用来实现虚拟处理机、虚拟内存、虚拟外部设备和虚拟信道等。

（4）异步性。在多道程序环境下，允许多个进程并发执行，但只有进程在获得所需的资源后方能执行。在单处理机环境下，由于系统中只有一个处理机，因而每次只允许一个进程执行，其余进程只能等待。当正在执行的进程提出某种资源要求时，如打印请求，而此时打印机正在为其他某进程打印，由于打印机属于临界资源，因此正在执行的进程必须等待，且放弃处理机，直到打印机空闲，并再次把处理机分配给该进程时，该进程方能继续执行。可见，由于资源等因素的限制，使进程的执行通常都不是“一气呵成”，而是以“停停走走”的方式运行。

7.4.4　操作系统的功能

由于多道程序设计技术的引入，各并发进程相互合作及相互竞争资源，为了保证系统高效、有序地运行，从资源管理和用户接口的角度，操作系统的主要功能包括以下几个方面：

（1）处理机管理。如何分配与回收被多道程序所共享的 CPU（实际上就是如何对进程进行合理调度），以提高 CPU 利用率。

（2）内存管理。一方面，多个程序如何合理分配与回收内存空间，使内存利用率尽可能高；另一方面，各程序所占的内存空间，必须进行必要的存储保护，以防止作业信息被窃取或混淆。同时，又要满足合理的共享；必要时，还要利用外存空间进行内存扩充。

（3）I/O 设备管理。各种外部设备的分配、回收以及共享，以提高设备利用率。

（4）文件管理。随着磁介质存储技术的进步，磁带、磁盘等大容量辅助存储设备的使用，大量用户程序、数据的存储组织、存储保护、共享等一系列问题，构成了操作系统中文件管理的主要内容。

（5）操作系统的用户接口。用户接口是为方便用户使用计算机资源所建立的用户和计算机之间的联系。用户接口可分为三个部分：

1）命令接口：命令接口是用户利用操作系统命令组织和控制作业的执行或管理计算机系统。命令是在命令输入界面上输入，由系统在后台执行，并将结果反映到前台界面或者特定的文件内。

2）程序接口：程序接口由一组系统调用命令组成，这是操作系统提供给编程人员的接口。用户通过在程序中使用系统调用命令来请求操作系统提供服务。

3）图形接口：图形用户接口采用了图形化的操作界面，用非常容易识别的各种图标来将系统各项功能、各种应用程序和文件，直观、逼真地表示出来。用户可通过鼠标、菜单和对话框来完成对应程序和文件的操作。

习题七

1．解释数据、数据库、数据库系统、数据库管理系统。
2．数据库管理系统的功能是什么？
3．解释数据模型的概念。
4．绘制 E-R 图的方法是什么？并举例说明。
5．关系数据库涉及哪些基本术语？并回答关系数据库有哪些特征？
6．数据结构的主要研究内容是什么？为什么要研究数据结构？
7．数据结构主要的逻辑结构和物理结构有哪些？
8．比较顺序存储和链式存储的优缺点。
9．有哪些主要的查找和排序方法，各自的效率如何？
10．软件危机产生的原因是什么？
11．请详细阐述软件生命周期。
12．操作系统有哪些基本类型？
13．请详细阐述操作系统的基本特征和功能。