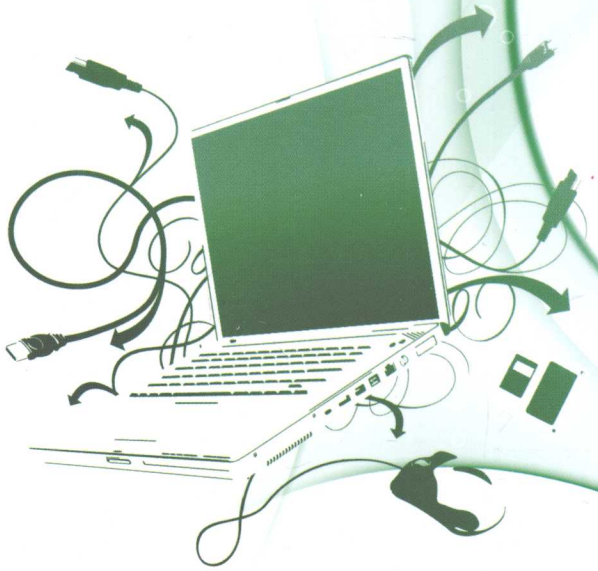
全国高等教育“十二五”精品教材

# 计算机软件技术基础

JI SUAN JI RUAN JIAN JI SHU JI CHU

主编 王海燕



航空工业出版社

全国高等教育“十二五”精品教材

# 计算机软件技术基础

主　编　王海燕

副主编　罗　静　赵永熹

航空工业出版社

北　京

## 内 容 提 要

本书针对非计算机专业的大学生、研究生以及科技工作者与研究人员对计算机软件应用技术的需要，介绍了计算机软件技术的基础知识、方法与实用技术。主要内容包括：基本数据结构及其运算、查找与排序技术、数据库设计技术、软件的设计与开发技术和操作系统原理等基础理论知识。每章都配有一定数量的习题。

本书内容丰富，通俗易懂，实用性强，可作为非计算机专业计算机软件基础课程的教材，也可作为广大从事计算机应用工作的科技人员的参考书。

**图书在版编目（CIP）数据**

计算机软件技术基础 / 王海燕主编. -- 北京 : 航空工业出版社，2012.2

ISBN 978-7-80243-919-1

Ⅰ. ①计… Ⅱ. ①王… Ⅲ. ①软件－技术 Ⅳ. ①TP31

中国版本图书馆 CIP 数据核字(2012)第 019626 号

计算机软件技术基础

Jisuanji Ruanjian Jishu Jichu

航空工业出版社出版发行

（北京市安定门外小关东里 14 号 100029）

发行部电话：010-64815615 010-64978486

北京市科星印刷有限责任公司印刷 全国各地新华书店经售

2012 年 2 月第 1 版 2012 年 2 月第 1 次印刷

开本：787×1092 1/16 印张：21.25 字数：530 千字

印数：1－3000 定价：45.00 元


计算机技术已深入到人类生活的各个角落，它与其他学科紧密结合，推动着各学科的飞速发展。作为新世纪的大学生，应该具备计算机的基础知识和应用能力。我国各大高校计算机专业都要求学生系统掌握数据结构、数据库技术、软件工程和操作系统的相关知识，每部分知识会独立设课，分别讲解。对于非计算机专业的学生，要求既熟悉自己所学的专业，又要掌握计算机的应用知识，利用计算机作为工具解决本领域中的任务，因此有必要在非计算机专业中进行计算机教育。

非计算机专业中的计算机教育，无论目的、内容、教学体系、教材、教学方法等各方面都与计算机专业有很大的不同，绝不能照搬计算机专业的模式和做法。根据教学需要，我们组织长期在第一线从事高校计算机软件技术基础教育的教师，编写了本书。本书主要针对非计算机专业的学生（包括广大科技人员）对于计算机软件技术的需要，将有关软件基础知识以及应用技术介绍给读者。本书并不是简单地将计算机专业的各门课程内容的简单组合，而是根据软件应用技术的需要，将它们有机地结合在一起，为读者提供软件开发中所需要的软件知识和技术，在编写方法上，既注重概念的严谨和清晰，又注重采用读者容易理解的方法阐明看似深奥难懂的问题。

全书共分 4 大部分 13 章：

第 1 部分：数据结构（第 1-5 章），介绍数据结构的基本概念，线性表、栈、队列、数组、二叉树、图等数据结构及有关算法，以及查找和排序的方法及相应算法。

第 2 部分：数据库系统（第 6-8 章），介绍数据库系统的基本概念，关系数据库系统的定义，关系代数，关系数据库标准语言 SQL，关系规范化理论以及数据库应用系统的设计方法。

第 3 部分：软件工程（第 9-12 章），介绍软件工程的形成，软件生存周期的各个阶段，软件开发和管理的相关知识。

第 4 部分：操作系统（第 13 章），介绍操作系统的基本概念、基本原理，操作系统的 5 大功能及实现技术。

本书内容丰富，通俗易懂，实用性强，便于自学，书中所有算法程序（C 语言描述）均上机调试通过。本书可作为非计算机专业的大学生或研究生的软件课程教材，也可作为广大从事计算机应用工作的科技人员的参考书。

本书由上海电力学院王海燕任主编，罗静和赵永熹任副主编。由于编者水平有限，书中存在的缺点和不足之处，敬请读者批评指正。请将意见发送至邮箱 wanghaiyan@shiep.edu.cn 中，谢谢！

编　者

2012 年 2 月

# 目　录

# 第1章 概 论

随着计算机（特别是微型计算机）技术和应用的飞速发展，其应用领域已经延伸到人们生活的各个方面。教学、商业、金融、工矿企业、科学研究、军事、邮电、通信，甚至家庭生活中都离不开计算机，借助计算机，许多行业都成倍提高了生产力和生产效率。计算机所处理的数据对象也从数值扩展到非数值。面对不同的数据处理对象、不同的需求，数据的织织形式、存储及运算必须有不同的方法，才能进行有效的处理。因此只有努力研究数据的内在结构，才能寻找到某些规律性的方法。此外，对于数据结构的深入研究，各应用领域都有广泛的需求。

## 1.1 数据结构的基本概念

**数据**（data）是信息的载体，所有能输入到计算机中并被计算机程序处理的描述客观事物的符号总称。如整数、实数、字符、文字、声音、图形、图像等都是数据。

**数据元素**（data element）是数据的基本单位。数据元素一般由一个或多个数据项组成。一个数据元素包含多个数据项时，常称为记录（record），也称结点（node）。

**数据项**（data item）是有独立含义的数据最小单位，也称域（field）。

什么是数据结构呢？在任何问题中，构成数据的数据元素并不是孤立存在的，它们之间必然存在着一定的关系以表达不同的事物及事物之间的联系。它是信息的一种组织方式，是数据按照某种组织关系联系起来的一批数据，其目的是为了提高算法的效率，然后用一定的存储方式存储到计算机中，并且它通常与一组算法的集合相对应，通过这组算法集合可以对数据结构中的数据进行某种操作。

简单说来，**数据结构**是一门研究非数值计算的程序设计问题中计算机的操作对象以及它们之间的关系和操作的学科。它不仅是一般程序设计的基础，而且是设计和实现编译程序、操作系统、数据库系统及其他系统程序和大型应用程序的重要基础。

数据结构是在整个计算机科学与技术领域上广泛使用的术语。它用来反映一个数据的内部构成，即一个数据由哪些成分数据构成，以什么方式构成，呈什么结构。数据结构有逻辑上的数据结构和物理上的数据结构之分。逻辑上的数据结构反映成分数据之间的逻辑关系，而物理上的数据结构反映成分数据在计算机内部的存储安排。数据结构是数据存在的形式，数据结构作为一门学科主要研究数据的各种逻辑结构和存储结构，以及对数据的各种操作。因此，主要包括三个方面的内容：

- 数据的逻辑结构；
- 数据的存储结构；
- 数据的运算。

通常，算法的设计取决于数据的逻辑结构，算法的实现取决于数据的存储结构。

### 1.1.1 数据的逻辑结构

数据的逻辑结构是指数据元素之间的逻辑关系，即从逻辑关系上描述数据。它与数据的存储无关，是独立于计算机的。前面已经介绍过数据的逻辑结构，下面用数学方法给出逻辑结构的定义：

设 D 表示数据元素的集合，R 表示 D 上关系的集合（即 R 反映了 D 中各元素的前驱、后继关系），则一个数据结构 Data-structure 可以表示为：

$$\text{Data-structure} = (D, R)$$

可见数据结构由两部分构成：

（1）表示各数据元素的信息 D；

（2）表示各数据元素之间关系的信息 R。

结点（数据元素）如果没有前趋，则称该结点为根结点或起始结点；结点如果没有后继，则称此结点为终端结点。例如，假设 a,b 是 D 中的两个数据元素，则二元组<a , b>，表示 a 是 b 的前趋，b 是 a 的后继。

根据数据元素之间关系的不同特性，数据的逻辑结构又可分为两大类：

（1）线性数据结构

线性结构的逻辑特征是有且仅有一个开始结点和一个终端结点，并且所有结点都最多只有一个直接前趋和一个直接后继。

（2）非线性数据结构

非线性结构的逻辑特征是一个结点可能有多个直接前趋和直接后继。

### 1.1.2 数据的存储结构

数据的逻辑结构是从逻辑上来描述数据元素之间关系的，是独立于计算机的。然而讨论数据结构的目的是为了在计算机中实现对它的处理，因此还需要研究数据元素之间的关系在计算机中的表示方式，这就是数据的存储结构。数据存储结构是数据的逻辑结构在计算机存储器中的实现。

数据的存储结构可用以下四种基本的存储方法得到：

**1．顺序存储结构**

顺序存储结构是把逻辑上相邻的数据结点存储在物理上相邻的存储单元中，结点之间的逻辑关系由存储单元的邻接关系来体现。由此得到的存储表示称为顺序存储结构，通常顺序存储结构是借助于程序语言的数组来描述的。

这种存储方式主要用于线性数据结构，某些非线性数据结构也可以通过某种线性化的方法采用顺序方式存储，例如完全二叉树等，具体方法将在后面第 3 章中介绍。

**2．链式存储结构**

链式存储结构可以把逻辑上相邻的两个元素存放在物理上不相邻的存储单元中。即可用一组任意的存储单元来存储数据元素，这些存储单元可以是连续的，也可以是不连续的。

结点间的逻辑关系是由附加的指针字段表示的。由此得到的存储表示称为链式存储结构，通常要借助程序语言的指针类型来描述。

**3．索引存储结构**

索引存储结构是在存储结点信息的同时，还建立附加的索引表。索引表中的每一项称为索引项，索引项的一般形式是：（关键字，地址），关键字是能唯一标识一个结点的数据项。若每个结点在索引表中都有一个索引项，则该索引表称为稠密索引。若一组结点在索引表中只对应一个索引项，则该索引表称之为稀疏索引。稠密索引中索引项地址指出结点所在的存储位置，而稀疏索引中索引项的地址则指示一组结点的起始存储位置。

**4．散列存储结构**

该方法的基本思想是根据结点的关键字直接计算出该结点的存储地址。

把结点 a 中的某数据项（也称为关键字）a.key 作为自变量，通过一个称为散列函数的 Hash(a.key)的计算规则，确定出该结点的实际存储单元地址，即：

$$Loc(a) = Hash(a.key)$$

其中，Loc(a)为结点 a 的地址。

一般的数据结构都可以采用上述基本存储方式之一存储，或通过基本存储方式组合存储。一个数据结构可以采用不同的存储方式，到底选择什么存储方式，取决于数据运算的方便性和存储资源的可利用性。

对给定的数据结构，一旦建立了存储结构，则结构中某结点 a 及存放该结点的存储单元就融为一体，为叙述方便，有时就没有很严格的区分。例如，当我们说到结点 a 的地址时，应该是指存储结点 a 的存储单元的地址，说到结点 a 的指针域时，应该是指存储结点 a 的存储单元的指针域等。

### 1.1.3 数据的运算

数据的运算是定义在数据逻辑结构上的操作，每种数据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。数据运算定义在数据的逻辑结构上，而具体实施必须依赖于数据的存储结构。也就是说，当数据采用不同的存储结构，同样的运算其算法也不相同。

## 1.2 算法描述

算法是对特定问题求解步骤的一种描述。或者说算法是为求解某问题而设计的步骤序列，是指令的有限序列。求解同样的问题，不同的人写出的算法是不同的。算法的执行效率除了与数据结构的优劣有关外，还与程序设计者的水平有关，下面给出算法应该具有的几个特征：

① **有穷性**：一个算法必须在执行有穷步后结束，且每一步都能在有限的时间内完成。

② **确定性**：算法中每一条指令必须有确切的含义，读者理解时不会产生二义性。并且，在任何条件下，算法只有唯一的一条执行路径，即对于相同的输入只能得到相同的输出。

③ **可行性**：一个算法必须是可行的，即算法中描述的操作都是可以通过已经实现的基本运算执行有限次来实现。

④ **输入**：一个算法应该有零个或多个输入。

⑤ **输出**：一个算法应该有一个或多个输出。

算法的含义与程序十分相似，但二者是有区别的。一个程序不一定满足有穷性，而且程序中的指令必须是机器可以执行的，算法中的指令则无此限制。但是一个算法若用机器可执行的语言来书写，则它就是一个程序。

一个算法可以用自然语言、数学语言或约定的符号语言来描述。本书用 C 语言来描述算法。

# 1.3 算法分析

同一个问题，可以写出多个算法，怎样判断一个算法的优劣呢？下面通过介绍几个有关算法效率的指标，对算法的效率进行简单评价。

讨论算法效率的前提是算法必须是正确的，即输入正确的数据能得到正确的结果。此外，主要考虑如下两点：

① **时间复杂度**：依据算法编制成程序后，在计算机上运行时所消耗的时间；

② **空间复杂度**：依据算法编制成程序后，在计算机执行过程中所需要的最大存储空间。

## 1.3.1 时间复杂度

要确定实现算法在运行时所花的时间和所占用的存储空间，最直接的方法就是测试，即将依据算法编制的程序在计算机上运行，所得到的结果就是算法运行时所花的时间。这种方法有时也称为事后统计的方法。同一算法在不同档次的计算机上运行所花的时间肯定不同，这取决于计算机系统的速度。另外一种方法就是事前分析估算的方法，通过对算法中不同语句序列的分析，得出算法中所有语句执行次数的相对大小，从而判断算法的运行时间长短。这只是一个相对概念，不是绝对大小。

假定知道算法中每一条语句执行一次所花的平均时间，则有：

算法运行所花的时间=语句执行一次所花的时间×语句执行次数

其中语句执行一次所花的时间取决于计算机系统中硬件、软件等环境因素，同一个算法用不同的语言、不同的编译程序、在不同的计算机上运行，效率均不同，所以使用绝对时间单位衡量算法效率不合适。而一个算法中语句的执行次数一般来说是确定的，因此，对于事前分析估算方法，我们讨论的目标集中在确定语句的执行频度上，即把算法的**语句执行频度**作为衡量一个算法时间复杂度的依据。在实际分析中，关注的是**频度的数量级**，即按重复执行次数最多的语句确定算法的时间复杂度。

一般地，我们将算法求解问题的输入量（或初始数据量）称为问题的**规模**，并用一个整数 n 表示。通常把一个程序的运行时间定义成一个 T(n)，当讨论一个程序的运行时间 T(n)时，注重的不是 T(n)的具体值，而是它的增长率。T(n)的增长率与算法中数据的输入规模是紧密相关的。而数据输入规模往往用算法中的某个变量的函数来表示，通常用 f(n)表示。随着数据输入规模的增大，f(n)的增长率与 T(n)的增长率相近，因此 T(n)同 f(n)在数量级上是一致的，引入符号“O”来表示这种数量级，算法的时间复杂度记作：

$$T(n) = O(f(n))$$

它表示随问题规模 n 的增大，算法执行时间的增长率和函数 f(n)的增长率相同，称作算法的渐近时间复杂度，简称时间复杂度。按数量级递增次序排列，常见的几种时间复杂度有：$O(1)$，$O(\log_2 n)$，$O(n)$，$O(n\log_2 n)$，$O(n^2)$，$O(n^3)$，这里 n 表示问题的规模。

【例 1-1】求两个 n 阶方阵的乘积 c=a*b，其算法程序段如下：

```
（1）for(i=1;i<=n;i++)                          n+1
（2）for(j=1;j<=n;j++)                          n(n+1)
（3）    {c[i][j]=0;                            n^2
（4）     for(k=1;k<=n;k++)                     n^2(n+1)
（5）          c[i][j]=c[i][j]+a[i][k]*b[k][j]; n^3
         }
```

其中右边列出的是各语句的频度。语句（1）的循环控制变量 i 要增加到 n，测试 i>=n 成立才会终止，故它的语句执行频度是 n+1，但是它的循环体却只能执行 n 次。语句（2）作为语句（1）循环体内的语句应执行 n 次，但语句（2）本身要执行 n+1，所以语句（2）的频度是 n（n+1）。同理，可得语句（3）（4）（5）的频度。该算法中所有语句执行频度之和（即算法的时间消耗）为：

$$T(n)= 2n^3+3n^2+2n+1= O(n^3)$$

【例 1-2】分析以下程序段的时间复杂度

```
for ( i = 0;  i<n;  i ++)
    for ( j = 0;  i<m;  j++)
        A[i][j]=0;
```

则该程序段的时间复杂度为 O (m*n)。

遇到多层循环时，要由内层向外层逐层分析。因此，当分析外层循环的运行时间时，内层循环的运行时间应该是已知的，这时可以把内层循环看成是外层循环的循环体的一部分。由此可见，当有若干个循环语句时，**算法的时间复杂度是由嵌套层数最多的循环语句中最内层语句的频度 f（n）决定的。**

【例 1-3】交换 a,b 的内容。

```
temp=i;
i=j;
j=temp;
```

以上三条单个语句的频率均为 1，该程序段的执行时间是一个与问题规模 n 无关的常数，因此算法的时间复杂度为常数阶，记为：T(n) = O(1)。

## 1.3.2 空间复杂度

一个算法的实现所占用的存储空间大致有这样三个方面：其一是指令、常数、变量所占用的存储空间；其二是输入数据所占用的存储空间；其三是算法执行时必需的辅助存储空间。前两种空间是计算机运行时所必须的。因此，把算法在执行时所需的辅助空间的大小作为分析算法空间复杂度的依据。

与算法时间复杂度的表示一致，也用辅助空间大小的数量级来表示算法的空间复杂度，

记为：S(n) = O ( f(n) )。它也是问题规模 n 的函数。

事实上，一个问题的算法实现，时间复杂度和空间复杂度往往是相互矛盾的。要降低算法的执行时间，可能要以使用更多的空间为代价；要节省空间，可能要以增加算法的执行时间为代价，两者很难兼顾。因此，只能根据具体情况有所侧重。

# 思考与练习

1．将下面程序的时间复杂度表示为 n 的函数。

（1）
```
int i=0，s1=0，s2=0;
  while（i++<n）
      { if (i%2)
        s1+=i;
       else s2+=i;
      }
```

（2）
```
for（i=0;i<=n;i++）
      for（j=0;j<=n;j++)
        s;
```

（3）
```
for(int i=0;i<m;i++)
    for(int j=0;j<n;j++)
        a[i][j]=i*j;
```

2．简述数据结构的分类。

3．简述下列术语：数据、数据元素、逻辑结构、存储结构、数据运算、线性结构、非线性结构、算法特性、时间复杂度。

# 第 2 章　线性数据结构

## 2.1　线性表

### 2.1.1　线性表的逻辑结构

线性表是由 n 个（n>=0）结构相同的数据元素 $a_1$，$a_2$，$a_3$，…，$a_i$，…，$a_n$ 组成的有限序列，记为（$a_1$，$a_2$，$a_3$，…，$a_i$，…，$a_n$）。这里的数据元素 $a_i$ 只是一个数据符号，其具体含义在不同情况下可以不同。数据元素的类型可以是高级语言提供的简单类型，或由用户定义的任何类型，如整型、实型、字符型、结构体类型等。

线性表可以用一个标识符来命名，如用 A 来命名线性表，则 A=($a_1$，$a_2$，$a_3$，…，$a_i$，…，$a_n$)。

数据元素的个数 n 称为线性表的长度，当 n=0 时，称线性表为空表，即该线性表不包含任何数据元素；当 n>0 时，线性表中的每一个数据元素都有一个确定的位置，即线性表中的数据元素在位置上是有序的。除第一个元素 $a_1$ 外，每一个数据元素有且仅有一个前趋元素，除最后一个元素 $a_n$ 外，每一个数据元素有且仅有一个后继元素。

线性表的例子不胜枚举：

【例 2-1】英文大写字母表（A，B，C，…，Z），是一个长度为 26 的线性表，B 的前趋是 A，B 的后继是 C。A 没有前趋，Z 没有后继。

【例 2-2】某学校的学生情况表如表 2-1 所示，每个学生的情况为一条记录，它由学号、姓名、年龄三个数据项组成。在表 2-1 中，一个数据元素由若干个数据项组成。在这种情况下，常把数据元素称为记录，含有大量记录的线性表又称为文件。

表 2-1　学生情况表

| 学号 | 姓名 | 年龄 |
|---|---|---|
| 00000001 | 王平 | 19 |
| 00000002 | 李林 | 20 |

线性表中结点之间的逻辑关系就是上述的邻接关系，由于该关系是线性的，因此线性表是一种线性结构。

### 2.1.2　线性表的顺序存储结构

1. 顺序表

在计算机内可以用不同的方式来表示线性表，其中最简单和最常用的方式是用一组地址