面向二十一世纪高等院校非计算机专业通用教材

软件技术基础

RUANJIAN JISHU JICHU

孙淑霞 肖阳春 赵仕波 彭 舰 ◎编著

- 重点介绍数据结构的基本概念和算法
- 详细阐述软件开发的设计、测试、维护和管理
- 全面讲解操作系统的处理机、设备、存储和文件管理

浦东电子出版社
Pudong ePress

软件技术基础

孙淑霞　肖阳春　赵仕波　彭 舰　　编著

本书配有光盘，需要的读者请到 http://210.34.51.1/tractate/index.asp 网页上申请，或到“网络与光盘检索实验室”联系。

浦东电子出版社
PEP Pudong ePress

内 容 简 介

《软件技术基础》介绍了计算机软件技术基础的三个部分：数据结构、操作系统、软件工程。为了更好地掌握数据结构部分的算法，学以致用，在数据结构后面给出了相应的实验内容，并且每章后面都附有习题，以便学生检查和巩固所学知识。

光盘以多媒体形式全面讲解了中文 Windows 98 的基本操作，常见的几种数据结构，如线性表、栈、队列、树、图等的基本概念和基本算法，并举例说明了典型的几种查找算法和排序算法的实现思想和实现过程，最后介绍了软件设计方法、软件测试和软件维护。本光盘着重让读者了解软件技术的基础知识、软件开发的步骤，以及软件的后期维护和管理。

《软件技术基础》注重实用性，内容简明扼要，可作为理工科非计算机专业大学本科生的教材，也可以作为从事计算机应用等工作的工程和科技人员参考。

书　　名：软件技术基础
文本著作者：孙淑霞　肖阳春　赵仕波　彭　舰
CD 制作者：海搏多媒体制作中心
责任编辑：董继菡
出 版 者：浦东电子出版社
地　　址：上海浦东郭守敬路 498 号上海浦东软件园内　201203
电话：021-38954510，38953321，38953323（发行部）
发 行 者：浦东电子出版社成都发行部
电话：028-85410679　85410306（传真）　邮编：610064
成都市望江路 29 号（四川大学内）
经　　销：各地新华书店、软件连锁店
排　　版：四川中外科技文化交流中心排版制作中心
CD 生产者：湖南省远景光电实业有限公司
文本印刷者：四川省科学技术情报研究所印刷厂
开本/规格：787×1092 毫米　16 开本　12 印张　200 千字
版次/印次：2002 年 9 月第一版　2002 年 9 月第一次印刷
印　　数：0001——4000 册
本 版 号：ISBN 7-900360-48-4/TP.20
定　　价：19.00 元（盘配书）

技术支持热线：（028）85412516

前　言

为了满足新世纪对理工科非计算机专业学生知识结构的需要，本教材是按照国家教委高教司颁发的“工科非计算机专业计算机基础教学指南”中有关软件技术基础课程的教学要求进行编写的。

学习《软件技术基础》课程的目的是帮助非计算机专业的学生在掌握了程序设计技术的基础上，进一步奠定进行应用软件开发的软件基础，从而达到扩大应用软件的开发队伍和提高应用软件人员素质的目的。

本教材突出教学重点，力图用较短篇幅系统介绍计算机软件技术基本知识，注重基础，强调实用，内容包括数据结构、操作系统和软件工程三部分。

本教材第 1 章是数据结构的内容，包括线性表、非线性结构、查找和排序算法，以及实验项目，要求学生在学习数据结构的内容时按要求完成每一个实验项目的内容。第 2 章是操作系统的内容，包括操作系统概述、处理机管理、存储器管理、作业管理、设备管理和文件管理。第 3 章是软件工程的内容，包括了软件生存期和生存期模型、软件需求分析、软件设计、软件编码、软件测试和软件维护等内容。每一章后面都给出了相应的习题，学生通过习题可进行自我检查和巩固所学的知识。

光盘以多媒体形式全面讲解了中文 Windows 98 的基本操作，常见的几种数据结构，如线性表、栈、队列、树、图等的基本概念和基本算法，并举例说明了典型的几种查找算法和排序算法的实现思想和实现过程，最后介绍了软件设计方法、软件测试和软件维护。本光盘着重让读者了解软件技术的基础知识、软件开发的步骤，以及软件的后期维护和管理。

本教材可作为理工科非计算机专业大学本科生的教材，讲授学时为 30～40 学时。数据结构部分的程序用 C 语言编写，因此要求读者应有 C 语言编程的基础。

本教材第 1 章由孙淑霞和彭舰编写，第 2 章由赵仕波编写，第 3 章由肖阳春编写。全书由孙淑霞统稿并修改定稿。由于编者水平有限，教材中难免存在缺点和错误，敬请读者批评指正。

编　者

目　　录

第 1 章　数据结构

数据结构是计算机科学的重要基础，已经发展为一门专门的学科。数据结构是软件的基础，数据结构的知识对设计和实现计算机的系统软件和应用软件是必不可少的。因此，学习数据结构有助于提高软件设计和编制程序的水平，同时也为学习其它软件课程打下了良好的基础。本章主要介绍数据结构的基本概念和一些常用的数据结构，数据的逻辑结构与存储结构，以及在数据结构上进行的两类重要操作：查找与排序。

1.1　数据结构与算法

1.1.1　数据结构的基本概念

1．数据

什么是数据？数字、字符、图形图像以及其它特殊的符号都是数据，它是信息的具体表示形式。给数据下一个定义，数据就是计算机能够识别、存储和处理的描述客观事物的符号，即数据就是计算机化的信息。

数据元素是数据的基本单位，有时也把它称为结点、记录、表目等。一个数据元素可以是一个单独的符号，也可以由多个数据项组成。数据项是有独立含义的数据最小单位，有时也把它称作域、字段等。例如，可以将一个班的学生信息用表 1-1 表示。它的数据元素是由姓名、学号、性别、出生年月、专业、特长等六个数据项组成的。

表 1-1　学生信息表

姓名	学号	性别	出生年月	专业	特长
李红	99011101	女	1983.4	应用数学	舞蹈
张强	99011102	男	1984.1	应用数学	田径
王江	99011103	男	1983.7	应用数学	管乐
刘宏	99011104	男	1984.5	应用数学	书法
……	……	……	……	……	……

表 1-1 是由具有相同性质的数据元素构成的一个集合，称为数据对象。

2．数据结构

什么是数据结构？表 1-1 就可以称为一个数据结构，表中的每一行为一个数据元素，每一个数据元素又由六个数据项组成，这些数据元素之间存在着一定的联系。数据结构是研究这些数据的集合，根据数据的性质、数据元素之间的关系，研究在计算机中如何表示、存储和运算这些数据的技术。研究数据结构就是要研究以下三方面的内容：

（1）数据的逻辑结构

数据的逻辑结构是数据间关系的描述，它只抽象地反映数据元素间的逻辑关系，不考虑

其在计算机中的存储方式。

数据的逻辑结构分为线性结构（如线性表、栈、队列、数组和串）和非线性结构（如树、二叉树和图），它们具有不同的逻辑特征。线性结构中的每个元素，除第一个外，都仅有一个直接前趋，同时除最后一个外，都仅有一个直接后继。非线性结构中的数据元素，或者有一个直接前趋（根元素除外）和多个直接后继（树形结构），或者有多个直接前趋和多个直接后继（图或网状结构）。

（2）数据的存储结构

数据的存储结构是数据元素在计算机存储器中的表示，即数据的物理结构，它是数据的逻辑结构在计算机存储器里的实现。数据的存储结构主要有顺序存储结构和链式存储结构。

（3）数据的运算

数据元素间不同的逻辑结构关系有着不同的运算方法，而运算的具体实现要在存储结构上进行。常用的运算有：

①插入　在数据结构的指定位置上添加新的数据元素。

②删除　删去数据结构中某个指定的数据元素。

③更新　改变数据结构中某个数据元素的值。

④查找　在数据结构中寻找能够满足指定条件的数据元素。

⑤排序　在保持数据元素个数不变的前提下，把数据元素按指定的顺序重新排列。

⑥遍历　在数据结构中，从第一个数据元素开始，依次访问每个元素。

3．数据的存储方式

（1）顺序存储

顺序存储方式是把逻辑上相邻的数据元素存储在物理位置上相邻的存储单元里，元素间的逻辑关系由存储单元的邻接关系来体现。这种存储方式主要用于线性的数据结构，如线性表、数组等。

顺序存储结构的主要特点有：

①数据元素中只有自身信息域，没有连接信息域，因此，存储密度大，存储空间利用率高。

②可以通过计算直接确定数据结构中第 i 个元素的存储地址。假设 L_0 为第一个元素的存储地址，n 为每个元素所占用的存储单元个数，那么，第 i 个元素的存储地址 L_i 为：

$$L_i=L_0+(i-1)\times n$$

③插入、删除操作会进行大量数据元素的移动。

（2）链式存储

链式存储方式就是在每个数据元素中至少包括一个指针域，用指针来体现数据元素之间逻辑上的联系。这种方式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，元素间的逻辑关系由附加的指针字段来表示。

链式存储结构的主要特点有：

①数据元素中除自身信息外，还有表示连接信息的指针域，因此比顺序存储结构的存储密度小，存储空间利用率低。

②逻辑上相邻的元素物理上不必邻接，可用于线性表、树、图等多种逻辑结构的存储表

示。

③插入、删除操作灵活方便，不必移动元素，只要改变元素中的指针值即可。

（3）索引存储

索引存储方式通常是在存储元素信息的同时，建立一张附加的索引表，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）

关键字是能惟一标识一个元素的数据项。如果每个元素在索引表中都有一个索引项，索引项的地址指示元素所在的存储位置，则该索引表称为稠密索引；如果一组元素在索引表中只对应一个索引项，索引项的地址指示一组元素的起始存储位置，则该索引表称为稀疏索引。

（4）散列存储

散列存储方式的基本思想是由元素的关键字决定元素的存储地址。即以关键字值 k 为自变量，通过某一散列函数计算出函数值 h(k)来，这个值就是元素的存储地址，将元素存入该地址中。

1.1.2　算法

数据的运算是定义在数据结构上的操作，而运算的实现步骤是用算法来描述的。计算机解题一般可分解成若干操作步骤，通常把完成某一任务的操作步骤称为求解该问题的算法，即算法是解决某一特定类型问题的有限运算序列。对任何一类问题来说，算法就是解决该类问题的办法或步骤。

1．算法的特性

算法具有以下特点：

（1）确定性

算法中的每一个规则、每一个操作步骤都应当是确定的，不允许存在多义性和模棱两可的解释。

（2）有穷性

一个算法必须在执行有限步骤后结束。也就是说，任何算法都必须在有限的时间内完成。如果一个算法要执行上千年才结束，该算法也就失去了它的实用价值。因此，有穷性还隐含了算法的执行时间应该合理的涵义。

（3）输入

在执行算法过程中，从外界获得的信息就是输入。一个算法有零个、一个或多个输入量。

（4）输出

一个算法所得到的结果就是该算法的输出。一个算法必须有一个或多个输出，否则该算法就没有实际意义了。

（5）可执行性

算法的每一步操作都应该是可执行的。例如，当 B=0 时，A/B 就无法执行，不符合可执行性的要求。

2．算法的种类

按照不同的应用领域，计算机的算法可以分为两大类：数值计算算法和非数值计算算法。

数值计算算法主要用于科学计算，其目的是为了求数值解。例如，求方程的根有二分法、

迭代法或牛顿法；计算数值积分有梯形法、辛普生法和柯特斯法。在这类算法中，算术运算占主要地位，所用的数据结构比较简单。

非数值计算算法主要用于数据管理、实时控制以及人工智能等领域。在这类算法中，逻辑判断通常占主导地位，算术运算则居于相对次要的地位。算法处理的内容也从单纯的数值运算扩展到了对数据、图形和字符信息的综合处理。一般来说，这类算法所使用的数据结构也比较复杂。

3．算法的描述

算法的描述就是用文字或图形把算法表示出来。常用的描述方法有：自然语言、流程图、N-S 结构流程图、伪代码和 PAD 图。

（1）自然语言

自然语言就是人们日常使用的语言，可以是汉语、英语或其它语言。用自然语言描述算法通俗易懂，但往往要用一段较冗长的文字才能表达清楚要进行的操作，容易出现“歧义性”，即往往要根据上下文才能正确判断出它的含义，不太严格。如果用自然语言描述的算法是顺序执行的，还比较容易理解，当算法中包含了判断和转移等步骤时，用自然语言描述就不容易理解了。

【例 1.1】求两个正整数 a 和 b 的最大公约数。

用自然语言描述的算法如下：

①用 a 除以 b，求 a/b 的余数 x_1；

②判断 x1 是否为 0，若 x1=0，则 b 为最大公约数；若 x1≠0，则继续执行③；

③用 b 除以 x1，求 b/x1 的余数 x2；

④判断 x2 是否为 0，若 x_2=0，则除数 x1 为最大公约数；若 x2≠0，则继续执行⑤；

⑤用 x1 除以 x2，求 x1/x2 的余数 x3；

执行与④中一样的判断，若成功，则找到最大公约数；若不成功，则继续执行下去，直到 x_{n-1}/x_n 的余数为零，则找到 x_n 为最大公约数。

由此可见，用自然语言描述算法比较繁琐。

（2）流程图

流程图是用图形来表示算法，即用一些几何图形的框来表示各种操作。它是使用最早的算法和程序描述工具。美国国家标准化协会 ANSI 规定的一些常用流程图符号(见图 1.1)已被大多数国家所接受。

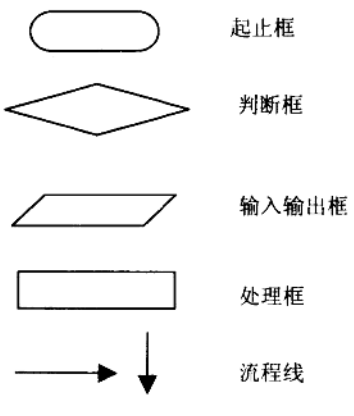


图 1.1　常用流程图符号

【例 1.2】用流程图描述例 1.1 的问题。

用流程图描述的例 1.1 如图 1.2 所示。

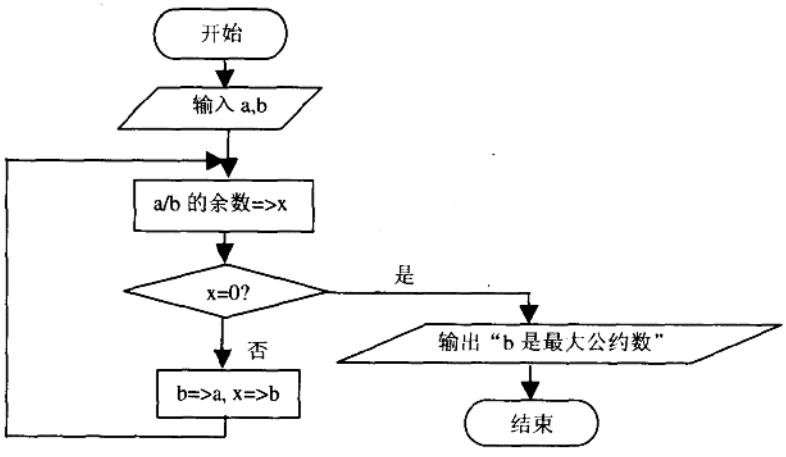


图 1.2　流程图

（3）N-S 结构流程图

N-S 的名称来源于英国的两位学者 I.Nassi 和 B.Shneiderman，他们在 1973 年提出了一种新的流程图形式。在这种流程图中完全去掉了流程线，全部算法都写在一个矩形框内，框内还可以包含其它的框。这就是 N-S 结构流程图，简称 N-S 图。

这种流程图适用于结构化程序设计，能清楚地显示出程序的结构。但当嵌套层数太多时，内层的方框将越画越小，从而影响图形的清晰度。

N-S 结构流程图用以下三种基本元素框来表示三种基本结构。

①顺序结构（见图 1.3）

②选择结构（见图 1.4）

图 1.4 表示的意义是：当条件 p 成立时执行 A 操作，条件 p 不成立时执行 B 操作。

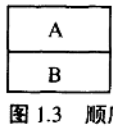


图 1.3　顺序结构

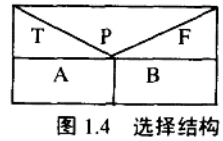
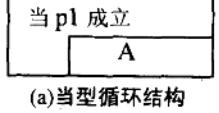
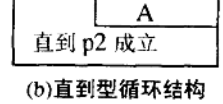


图 1.4　选择结构



(a)当型循环结构



(b)直到型循环结构

图 1.5　N-S 结构流程图

③循环结构（见图 1.5）

图 1.5(a)是当型循环结构，它表示的意义是：当条件 p1 成立时反复执行 A 操作，直到条件 p1 不成立时为止。图 1.5(b)是直到型循环结构，它表示的意义是：反复执行 A 操作，直到条件 p2 成立时为止。

由于 N-S 图仅使用三种基本结构设计程序，因此使某些程序设计的实现变得繁琐和困难。

【例 1.3】用 N-S 图描述例 1.1 的问题（见图 1.6）。

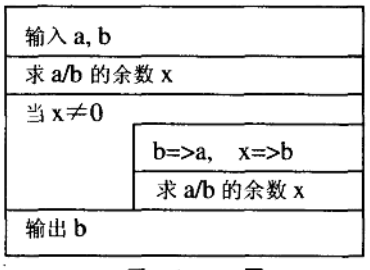


图 1.6 N-S 图

（4）伪代码

伪代码是用介于自然语言和计算机语言之间的文字和符号来描述算法。它不用图形符号，比较好懂，也就容易转化为高级语言程序。

【例 1.4】用伪代码描述例 1.1 中的问题

```
read   a, b
a/b=>a
do while x≠0
  b=>a
  x=>b
  a%b=>x
enddo
print b
```

当 x≠0 时，执行下面的操作：
把 b 的值赋给 a
把 x 的值赋给 b
把 a%b 的值赋给 x

（5）PAD 图

PAD 图的全名为“问题分析图(Problem Analysis Diagram)”，是由日本人二村良彦设计的一种描述工具。用 PAD 图描述的算法通常呈树形结构，图 1.7 是 PAD 图使用的各种符号。

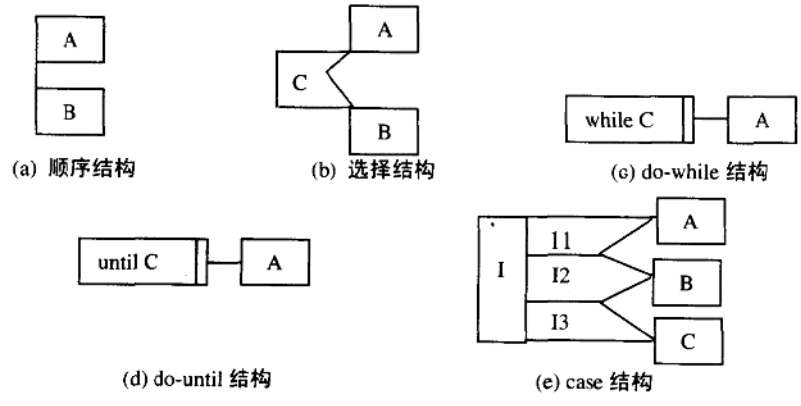


图 1.7 PAD 图

【例 1.5】用 PAD 图描述例 1.1 中的问题（见图 1.8）。

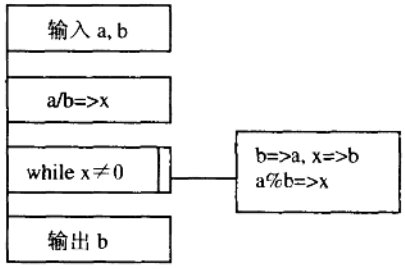


图 1.8　PAD 图

PAD 图不仅克服了传统流程图不能清晰地显示出程序控制结构的缺点，也不用像 N-S 图那样必须将全部算法约束在一个方框内，使其显得太拥挤，能清晰地显示出程序的各部分控制结构。

4．算法设计的要求

通常设计一个“好”的算法应该考虑达到如下目标：

（1）正确性　算法应该能满足具体问题的需求。

（2）可读性　算法主要是为了人的阅读、理解和交流，其次才是机器执行。

（3）健壮性　当输入数据非法时，算法也能适当地作出反应或进行处理，而不会产生莫名其妙的输出结果。

（4）效率与低存储量需求　效率指的是算法执行的时间，存储量需求是指算法执行过程中所需的最大存储空间。同一个问题如果有多种算法可以解决，执行时间短的算法效率高。而效率与低存储量需求都与问题的规模有关。

5．算法分析

算法分析主要是对算法效率的分析，分析算法所要占用的计算机资源，即运行时间和占用空间。运行时间是指一个算法在计算机上运行所花费的时间，采用时间复杂度来度量；占用空间是指计算机在存储器上所占用的存储空间，主要考虑在算法运行过程中临时占用的存储空间大小，用空间复杂度来度量。

（1）时间复杂度

算法执行时间需通过依据该算法编制的程序在计算机上运行时所消耗的时间来度量。为了便于比较同一问题的不同算法，通常的做法是从算法中选取一种对于所研究的问题(或算法类型)来说是基本运算的原操作(指固有数据类型的操作)，以该基本操作重复执行的次数作为算法的时间度量。

一般情况下，算法中基本操作重复执行的次数是问题规模 n 的某个函数 f(n)，算法的时间度量记作：

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

它表示随问题规模 n 的增大，算法执行时间的增长率和 f(n)的增长率相同，称作算法的渐进时间复杂度，简称时间复杂度。一般不必精确计算出算法的时间复杂度，只要大致计算出相应的数量级，如 O(1)、$O(\log_2 n)$、O(n)或 $O(n^2)$等。一般地，常用的时间复杂度有如下关

系：

$$O(1)\leqslant O(\log_2 n)\leqslant O(n)\leqslant O(n\log_2 n)\leqslant O(n^2)\leqslant O(n^3)\leqslant\cdots\leqslant O(n^k)\leqslant O(2^n)$$

一个算法中所有语句的频度（该语句重复执行的次数）之和构成了该算法的运行时间。在难以精确计算语句频度的情况下，只需求出它关于 n 的增长率或阶即可。

【例 1.6】分析下面程序的时间复杂度。

```
for(k=1;k<n;k++)
  {  y+=1;          /*   语句(1)   */
     for(j=0;j<=(2*n);j++)
        x+=1;       /*   语句(2)   */
  }
```

在上面的算法中，语句(1)的频度是 n-1，语句(2)的频度是$(n-1)(2n+1)=2n^2-n-1$。因此该程序段的时间复杂度为：$T(n)=2n^2-n-1=O(n^2)$。

解决同一类问题的不同算法，在执行时间上可能有很大的差异。例如，对于长度为 1024 的有序表；若要用顺序查找算法。最多要执行 1024 次比较操作；而用折半查找算法，最多只需进行 10 次比较操作。当数据量越大，不同算法在执行时间上的差异也就可能越大。执行时间的长短，是衡量算法效率最基本的依据。

（2）空间复杂度

空间复杂度用于度量算法所需的存储空间。记作：

$$S(n)=O(f(n))$$

其中 n 为问题的规模（或大小）。在计算机上执行的一个程序除了需要存储空间来寄存本身所用指令、常数、变量和输入数据外，还需要一些对数据进行操作的工作单元和存储一些为实现计算所需信息的辅助空间。若输入数据所占空间只取决于问题本身，和算法无关，则只需要分析除输入和程序之外的额外空间，否则应同时考虑输入本身所需空间(和输入数据的表示形式有关)。

随着计算机内存容量的增大，为节省少量内存单元去花费精力是不值得的。但当处理大容量的数组或矩阵运算时，用好的算法却可以节省大量的存储空间。

【例 1.7】在内存中存放一个 N×N 的单位矩阵（如图 1.9 所示），并打印出来。

单位矩阵是一个对角线元素全为 1，其余元素全为 0 的 N×N 阶矩阵。用不同的算法，所占用的存储空间也将不同。

$$A_{N\times N}=\begin{bmatrix}1&0&0&\cdots&0&0\\0&1&0&&0&0\\0&0&1&&0&0\\&\vdots&&\vdots&&\vdots\\0&0&0&&1&0\\0&0&0&\cdots&0&1\end{bmatrix}$$

图 1.9　N×N 单位矩阵

①直接法

这种方法是先生成矩阵，然后将其打印出来。算法如下：

```
for(i=0;i<N;i++)
    for(j=0;j<N;j++)
      if(i!=j)
        a[i][j]=0;
      else
        a[i][j]=1;
```

生成矩阵

```
for(i=0;i<N;i++)
    { for(j=0;j<N;j++)
      printf("%d", a[i][j]);
    printf("\n");
    }
```

打印矩阵

②压缩存储法

这种方法是用一个数组专门存储对角线上的元素，零元素不存储。算法如下：

```
for(i=0;i<N;i++)
      x[i]=1;
```

生成打印矩阵

```
for(i=0;i<N;i++)
    {   for(j=0;j<N;j++)
          if(i= =j)
              y[j]=x[i];
          else
              y[j]=0;
for(j=0;j<N;j++)
    printf("%d",y[j]);
printf("\n");
}
```

打印矩阵

由此可见，当 N 很大时，两种算法所占的存储空间是相差很大的，前者正比于 N^2，而后者正比于 N。

6．算法与数据结构

算法与数据结构是计算机程序的两大基础，数据结构是为了研究数据运算而存在的；算法是为了实现数据运算，即实现数据的逻辑关系的变化，或者是在这个结构上得到一个新的信息而存在的。数据结构与算法的实质不仅表现在两者互为依存，还体现在提高计算机效率的作用上。数据结构直接影响计算机进行数据处理的效率，而算法的好坏也直接影响计算机的效率。

美国科学家 N.Wirth 在谈到算法与数据结构二者的联系时，明确地指出：

算法+数据结构=程序

这说明了算法与数据结构在编制程序中的地位及重要性。

1.2　线性表

本节主要介绍线性数据结构中的线性表、栈、队列、数组和串的基本概念、基本知识和基本算法。

1.2.1 线性表的逻辑结构

线性表是 n(n≥0)个具有相同属性的元素组成的有限序列。可表示为

$$(a_1,a_2,\ldots,a_{n-1},a_n)$$

其中，a_1，…，a_n 是元素，其下标表示元素的序号，代表元素在线性表中的位置。n 是表中数据元素的个数，定义为表的长度。当 n=0 时，线性表是空表。例如，（10，20，30，40，50，60）是一个整数序列的线性表。对下面的线性方程：

$A(x)=(a_1,a_2,\ldots,a_n)(x_1,x_2,\ldots,x_n)^T$

$(a_1,a_2,\ldots,a_n)$和$(x_1,x_2,\ldots,x_n)^T$分别构成了两个线性表。

1.2.2 线性表的顺序存储结构及其基本运算

1. 线性表的顺序存储结构

在计算机中可以用不同的方式来存储一个线性表。用一组地址连续的存储单元顺序存储线性表中的元素，称这种存储方法为顺序存储。用顺序存储结构存储的线性表称为顺序表。高级语言中的一维数组就是用顺序存储的线性表。

假设表中的每一个元素所占的存储空间是相同的，即占用 L 个存储单元，第 1 个元素 a_1 的开始存储位置为 $Loc(a_1)$，那么第 i 个元素的存储位置为 $Loc(a_i)$，可用下式求出：

$$Loc(a_i)= Loc(a_1)+(i-1)*L$$

这里的 $Loc(a_1)$称为线性表的起始地址或基地址。

以顺序存储结构存储的线性表具有的特点是：表中相邻的两个元素 a_i 和 a_{i+1} 具有相邻的存储位置。因此，只要确定了线性表的起始存储地址，就可以求出任何一个数据元素的存储地址。因此，顺序表是一种随机的存取结构。

2. 基本运算

对线性表可以进行各种运算，例如求表长、查找、存取或替换表中某个指定的元素等。下面只介绍线性表的插入和删除运算。

（1）插入运算

线性表的插入运算是指在表的第 i 个($1\leqslant i\leqslant n$)元素 a_i 之前(或之后)插入一个新的元素，如图 1.10 所示。

由于数据元素的物理顺序必须和元素的逻辑顺序相一致，因此，在长度为 n 的线性表中的元素 a_i 之后插入一个新元素 x 时，需要将原来表中 a_i 之后的 n－i 个元素向后移动一个位置，产生一个新的线性表，其长度为 n+1。

假设向每个位置插入的概率相等，均为 $\frac{1}{n+1}$，则在线性表中插入一个新元素平均需要移动的元素个数为：

$$\frac{1}{n}\sum_{i=0}^{n}(n-i)=\frac{n}{2}$$