21 世纪全国高职高专计算机系列实用规划教材

数据结构(C 语言版)

主　编　夏　燕　张兴科
副主编　李笑雪　蔡　芸
参　编　张顺利　李琳琳

北京大学出版社
PEKING UNIVERSITY PRESS

内 容 简 介

本书系统地介绍了较为常用的数据结构，主要包括线性表、栈、队列，串、数组、树和二叉树、图、查找表和排序，并按照高职高专计算机专业本课程大纲要求，对每种数据结构算法实现进行阐述，并对各种算法的时间和空间性能作了简要分析。

本书内容详实，通俗易懂，理论的讲述够用为度，注重实践。每章都有要重点掌握的、在 C 语言环境下调试通过的上机实训题，章末有大量标准化习题和上机操作题，并且上机操作题还配有参考操作步骤，使读者对每章的学习从理论到实践都能得到进一步巩固。

本书可作为高职高专院校、技校、职高及社会办学相关课程的教材，也非常适合计算机相关专业初学者学习使用，特别是要参加专升本考试的读者，是很好的学习参考书。

图书在版编目(CIP)数据

数据结构(C 语言版)/夏燕，张兴科主编. —北京：北京大学出版社，2007.7

(21 世纪全国高职高专计算机系列实用规划教材)

ISBN 978-7-301-12409-3

Ⅰ. 数… Ⅱ. ①夏… ②张… Ⅲ. ①数据结构—高等学校：技术学校—教材②C 语言—程序设计—高等学校：技术学校—教材 Ⅳ. TP311.12 TP312

中国版本图书馆 CIP 数据核字(2007)第 083359 号

书　　名：数据结构(C 语言版)

著作责任者：夏　燕　张兴科　主编

责 任 编 辑：李彦红

标 准 书 号：ISBN 978-7-301-12409-3/TP・0904

出　版　者：北京大学出版社

地　　址：北京市海淀区成府路 205 号　100871

网　　址：http://www.pup.cn　http://www.pup6.com

电　　话：邮购部 62752015　发行部 62750672　编辑部 62750667　出版部 62754962

电 子 邮 箱：pup_6@126.com

印　刷　者：

发　行　者：北京大学出版社

经　销　者：新华书店

787 毫米×1092 毫米　16 开本　20.75 印张　471 千字

2007 年 7 月第 1 版　2007 年 7 月第 1 次印刷

定　　价：28.00 元

前　言

数据结构是计算机相关专业重要的专业基础课，是计算机程序设计的重要理论技术基础。通过该课程的学习，不仅可以使学生掌握数据结构的基本内容、典型算法和使用方法，而且能够训练学生应用数据结构和算法进行具体应用问题的程序设计能力，为学生后续课程的学习打下良好的基础。

本书主要是针对高职高专院校、职高和技校学生撰写的。教材的编写以高职高专计算机专业本课程教学大纲为主要依据，结合专升本考试大纲的要求，讨论了数据的逻辑结构与存储结构，以及相关算法，还讨论了常用的几种查找方法和排序算法，帮助读者轻松掌握数据结构的基础知识与算法实现。

全书共 10 章，内容包括数据结构概述、线性表、栈、队列、串、数组、树和二叉树、图、查找表和排序。每种数据结构都有较完整的数据类型定义、在计算机内的存储表示和算法实现，并运用大量实例进行讲解，每章都配有要求重点掌握、在 C 语言环境下调试通过的上机实训题，使读者通过上机实训来加深对数据结构课程的理解和掌握。

本书每章后面配有大量标准化习题和上机操作题，并且上机操作题还配有参考操作步骤，通过练习，使读者对每章的学习从理论到实践都能得到进一步巩固。同时还可以到北京大学出版社第六事业部网站(www.pup6.com)下载本书所有习题参考答案和上机操作题 C 语言程序。

本书由昆明冶金高等专科学校的夏燕、山东信息职业技术学院的张兴科担任主编，河南财经学院的李笑雪、北京交通职业技术学院的蔡芸担任副主编，郑州纺织专科学校的张顺利、淄博职业技术学院的李琳琳担任参编。夏燕编写了第 6 章、第 9 章，张兴科编写了第 7 章，李笑雪编写了第 2 章、第 3 章和第 4 章，蔡芸编写了第 10 章，张顺利编写了第 8 章，李琳琳编写了第 1 章和第 5 章。

郑州牧业高等专科学校的黄贻彬老师、洛阳大学的张红霞老师在本书编写过程中提供了很大帮助，在此深表感谢。

由于时间仓促，编者水平有限，书中错误和疏漏之处在所难免，恳请广大读者批评指正。

编者

2007 年 5 月

目　录

第 1 章　概　　论

教学提示：“数据结构”是研究数据在程序设计中操作对象及其之间关系与操作的学科，是介于数学、计算机硬件和计算机软件三者之间的一门核心课程，属于计算机专业的核心基础课程。“数据结构”是数据的组织、存储和运算的总和；是数据按照某种关系组织起来，同时用一定的存储方式存储到计算机中。

教学要求：

(1) 理解数据结构及相关的概念。

(2) 掌握各种逻辑结构的特点。

(3) 掌握算法的定义、特性及用类 C 语言描述算法的规则。

(4) 掌握评价算法优劣的标准：时间复杂度、空间复杂度的定义及表示。

1.1　数据结构的概念

随着计算机科学技术的快速发展，尤其是计算机的软、硬件技术飞速发展，计算机的应用领域也越来越广泛，从最初的科学计算逐步发展到人类活动的各个领域。

计算机处理的数据对象从单纯的数值计算，扩展到图像、视频、音频等具有不同结构的非数值数据的处理。针对复杂的处理对象，程序设计必须更多的注意到数据和数据之间存在的关系。合理的数据组织形式与良好的程序设计方法相互结合是有效地解决问题的前提。通常情况下，精心选择的数据结构可以产生更高的运行效率或者存储效率的算法。这就是“数据结构”这门学科形成和发展的原因。

1.1.1　什么是数据结构

随着计算机应用领域的不断扩大，计算机处理的对象更多的是非数值计算问题，它们的数学模型无法用数学方程来进行描述，此时就必须建立相应的数据结构来进行描述，分析问题中所用到的数据是如何组织的，研究数据之间的关系如何，进而为解决这些问题设计出合适的数据结构。

下面请看三个具体例子。

【例 1.1】　学生档案信息。

将学生信息组织成如表 1-1 所示的花名册。花名册中每个学生的信息由编号、姓名、性别、出生年月、籍贯等项目组成，占表的一行，一行信息称为一个结点，则表中的结点和结点之间是一种简单的线性关系，这就是上述花名册表的线性逻辑结构。当用计算机对上述花名册表中的数据进行运算时，就要考虑那些结点在计算机中的存储表示，即存储结构。另外，还必须考虑如何进行结点的插入、删除、修改、检索或查找，这就涉及数据的运算。

表 1-1 学生档案信息

编 号	姓 名	性 别	出 生 年 月	籍 贯
01	王蔷	女	1981.5	山东
02	万名化	男	1982.1	河北
03	赵力新	男	1981.6	浙江
04	孙仁	女	1983.1	辽宁
05	周立	男	1982.9	江苏
…	…	…	…	…

【例 1.2】 家族族谱。

家庭父子的关系，一父可对应多个子女，就不再如【例 1.1】所示——简单的一对一关系。将其用图示表示，如图 1.1 所示。

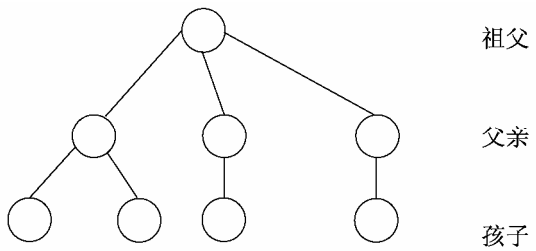


图 1.1 家族父子关系

参照图 1.1，要查找父亲的孩子，就不再是一对一的关系，同一个父亲可以找到多个孩子。同样用计算机进行处理时既要考虑存储问题，又要考虑增加、删除、查询等运算。

【例 1.3】 城市交通图。

众所周知，城市之间的交通联系比前两例要更复杂。也就是说数据元素之间出现了多对多的情况。用顶点表示城市，连线表示通路，如图 1.2 所示。

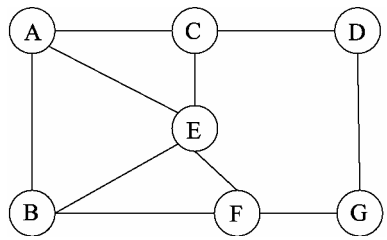


图 1.2 城市交通图

图 1.2 中，为表示城市交通图，可采用一种称为图的结构来表示实际的交通网络，此时，这个交通网络图就表示一个数据结构。在这个数据结构中，结点之间的关系可以是任意的，图中任意两个结点之间都可以相关。同样，必须考虑构造后将这张图存入计算机中，这就要涉及图的存储结构问题，以及图的运算。

从以上例子可见，要描述诸如此类的非数值计算问题的数学模型，则要涉及一些诸如表、图还有树之类的数据结构。

数据结构课程实际上是研究非数值计算问题的程序设计中计算机的操作对象以及它们之间的关系和运算操作等的一门学科。在计算机科学技术中，数据结构不仅是一般程序设计(特别是非数值计算的程序设计)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。

1.1.2　基本术语

下面介绍几个与数据结构相关的基本术语。

1. 数据

数据是人们对现实世界客观事物的符号化描述。因此，从整数、实数、字符、文字到声音、图形、图像都是数据。在计算机领域中，数据是指能够被计算机识别、存储、处理的一切信息符号。数据是计算机程序加工的“原料”。在程序设计语言中常说由程序处理数据，但程序也是数据。如 C 语言源程序由编译程序经过编译生成二进制的目标程序，对编译程序来说，源程序就是要处理的数据。

2. 数据元素

数据元素是数据整体中的相对独立的单位，即是数据的基本单位。通常在计算机程序中作为一个整体进行处理。数据元素一般由数据项组成，数据项(item)是具有独立含义的最小标识单位。

如表 1-1 所示，学生档案信息数据库中，一个学生信息记录可称为一个数据元素，而这个数据元素中的字段：学号、姓名、性别等就是一个个的数据项。

3. 数据对象

数据对象是具有相同特征的数据元素的集合，是数据的一个子集。数据对象与数据间的关系是部分与整体的关系。如整数数据对象是集合 N={0,±1,±2,…}，字母字符数据对象是集合 C={'A','B','C',…,'Z'}，表 1-1 所示的学生档案信息表也可看作一个数据对象。

4. 数据结构

数据结构是指相互之间存在一种或多种特定关系的数据元素集合，是带有结构的数据元素的集合，它指的是数据元素之间的相互关系，即数据的组织形式。

数据结构一般包括以下三个方面的内容：

(1) 数据元素之间的逻辑关系，有时也称为数据的逻辑结构。

(2) 数据元素及其关系在计算机内存中的表示(又称为映像)，称为数据的物理结构，又称为数据的存储结构，它包括数据元素的表示和数据元素之间关系的表示。

(3) 数据的运算及实现，即对数据元素可以施加的操作及其这些操作在相应的存储结构上的实现。

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描述。数据的逻辑结构通常有 4 类：集合结构、线性结构、树形结构、图形结构或称网状结构。

(1) 集合结构：数据元素之间组织形式松散，其结构为离散型，可以认为是数据元素间无逻辑关系，只是在同一个集合中，如图 1.3 所示。

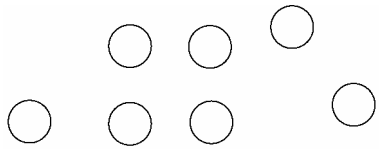


图 1.3 集合结构

(2) 线性结构：数据元素之间是一对一的关系。【例 1.1】中，各学生记录按学号排列之间的关系，可用图 1.4 来表示。很容易看出，每个数据元素有且只有一个前驱元素(除第一个元素之外)和一个后继元素(最后一个元素除外)。也就是说，每个元素都有唯一的前驱和后继(除第一个和最后一个元素外)，元素关系可以记为 1:1 的关系。

图 1.4 线性结构

(3) 树形结构：数据元素之间是一对多的关系。【例 1.2】中，父子关系，可用图 1.5 表示，像一棵倒挂的树。在树中，最上面的结点(数据元素)称为根结点，最下面的结点(数据元素)为叶子结点。可以看出，每个结点最多只能有一个父亲，但可以有多个孩子。即每个数据元素(根结点除外)有且只有一个前驱，但可以有多个后继。元素关系可以记为 1:m 的关系。

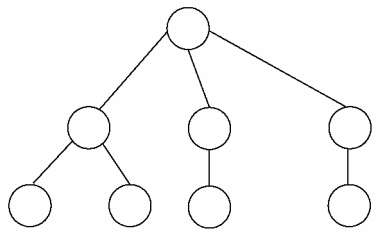


图 1.5 树形结构

(4) 图形结构：数据元素之间是多对多的关系。如【例 1.3】中的城市交通图，可以看出，每个城市有多个前驱或后继。即图形结构特点是每个数据元素可以有多个前驱和后继。元素关系可以记为 m:n 的关系。

2) 数据的存储结构

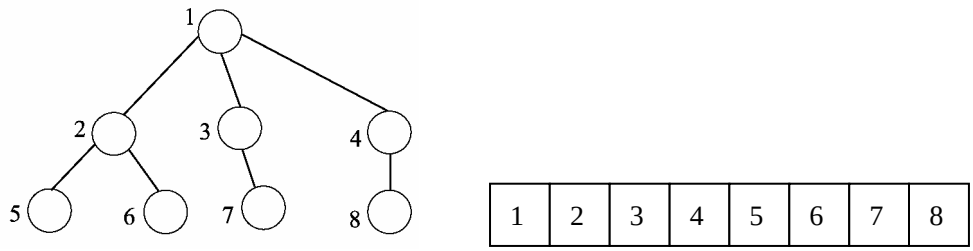
数据的存储结构是指逻辑结构用计算机语言的实现，就是数据元素及其关系如何存入计算机的问题。数据的存储结构可按以下四种基本存储方法而得到。

(1) 顺序存储方法。

顺序存储方法是把逻辑上相邻的结点存储在物理位置上相邻的存储单元中，结点间的逻辑关系由存储单元的相邻关系而体现。由此得到的存储表示称为顺序存储结构，通常可借助计算机语言的数组来描述。

【例 1.4】 顺序存储一棵树。

按照树的结构从上到下，从左到右给每个结点编号，在存储器中按编号进行顺序存储，如图 1.6 所示。



(a) 逻辑结构　　(b) 顺序存储结构

图 1.6　顺序存储结构

(2) 链状存储方式。

不要求逻辑上相邻的结点在物理位置上也相邻，数据元素可以存储在任意位置。为了实现数据元素之间逻辑关系的存储，必须通过一些附加的手段来存储这种相互关系，由此得到的存储表示称为链状存储结构。一般可以用指针来实现，通常可借助计算机程序语言的指针类型或者游标来描述。

【例 1.5】 链式存储——线性表(A,B,C)。

如图 1.7 所示，开始元素位置 1006，通过元素 A 的地址(指针)可以找到元素 B，依次类推可以找到下一个元素。

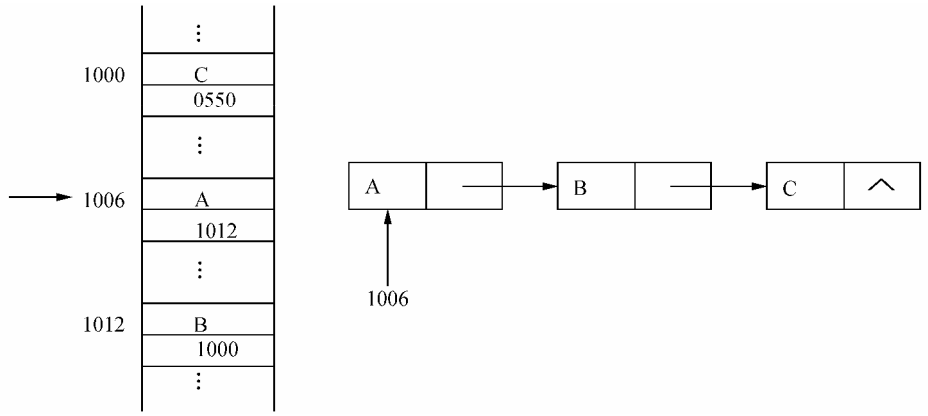


图 1.7　链式存储结构

(3) 索引存储方法。

该方法通常是在存储结点信息的同时，建立附加的索引表。索引表一般有稠密索引和稀疏索引两种。

(4) 散列存储方法。

该方法是依据结点的关键字直接计算出该结点的存储地址，而后将结点按某种方式存入该地址的一种存储方法。

数据的逻辑结构和物理结构是密切相关的，解决不同的问题，面对的数据的逻辑结构是不同的，设计的算法要依据逻辑结构，但算法如何实现又取决于数据的存储结构。

5. 数据的运算

数据的运算是指定义在数据的逻辑结构上的一组操作的集合。例如，检索(查找)、插

入、删除、定位、修改、排序等。要注意的是，这些运算实际上是在逻辑结构上对抽象数据所施加的一系列抽象的操作，运算的实现依赖于所选取的存储结构，是依赖于不同的计算机程序设计语言的。

1.2 数 据 类 型

数据类型是一组性质相同的值集合以及定义在这个集合上的一组操作的总称。数据类型中定义了两个集合，即该类型的取值范围，以及该类型中可允许使用的一组运算。例如高级语言中的数据类型就是已经实现的数据结构的实例。在高级语言如 C 语言中，整型类型可能的取值范围是-32768～+32767，可用的运算符集合为加、减、乘、除、乘方、取模。

按“值”的不同特性，高级程序语言中的数据类型可分为两大类：一类是非结构的原子类型。原子类型的值是不可分解的，如 C 语言中的标准类型(整型、实型和字符型)及指针。另一类是结构类型，结构类型的值是由若干成分按某种结构组成的，因此是可以分解的，并且它的成分可以是非结构的，也可以是结构的。例如数组的值由若干分量组成，每个分量可以是整数，也可以是数组等。数据类型指由系统定义的、可直接使用且可构造的数据类型。

因此，数据类型不仅是数据对象的集合还有在这些数据对象上的操作集合。

抽象数据类型是指一个数学模型及定义在该数学模型上的一组操作。抽象数据模型的定义取决于它的一组逻辑特性，与其在计算机内部如何表示和实现无关。

抽象数据类型的范畴更加广泛，不局限于已固有的数据类型，还包括了用户自定义的数据类型。因此，在抽象数据定义时定义三元组：D 为数据对象，S 为数据对象间的关系，O 为数据集上所完成的基本操作集——(D,S,O)。

在现代软件设计和开发中，大量使用抽象数据类型。

抽象数据类型的定义格式：

```
ADT 抽象数据类型名{
    数据对象的定义
    数据关系的定义
    基本操作
}ADT 抽象数据类型名
```

线性表的抽象数据类型描述：

```
ADT  Linear_list {
数据元素   所有 ai 属于同一数据对象,i=1,2,…,n      (n≥1)
逻辑结构   所有数据元素 ai 存在次序关系(ai,ai+1),a1 无前驱,an 无后继.
线性表的基本操作 /* 设 L 为 Linear_list 类型的线性表 */
InitList(L);        /* 建立一个空的线性表 L */
Length(L);          /* 求线性表 L 的长度 */
GetElement(L,i);    /* 取线性表 L 中的第 i 个元素 */
Locate(L,x);        /* 确定元素 x 在线性表 L 中的位置 */
Insert(L,I,x);      /* 在线性表 L 中的第 i 个位置处插入数据元素 x */
Delete(L,i);        /* 删除表 L 中第 i 个位置的元素 */
…
}
```

1.3 算　　法

1.3.1 算法的描述

1. 算法的概念

做任何事情都是有一定步骤的。例如，购买某种商品，总是要先选好商品，然后填写购物单，付款，提货。这些都需要一定顺序，并且是缺一不可的步骤。因此简单地说，算法就是为解决某一问题所采取的方法和步骤。在计算机领域中，算法实质上是针对所处理问题的需要，在数据的逻辑结构和物理结构基础上，在有限步骤内解决这一问题所采用的一组指令序列。

2. 算法的五个重要特性

1) 有穷性

一个算法应包含有限的(指在合理的范围之内)操作步骤，保证执行有限步骤后能够结束。

2) 确定性

算法中的每一步执行的操作都必须是确定的，也就是说，算法的含义应当是唯一的，不能产生“歧义性”。比如，相同的输入，只能产生相同的输出结果。

3) 可行性

算法可以在有限的时间内完成，并能实现。

4) 输入

所谓输入指执行算法时可以从外界得到必要的消息(数据)。一个算法可以有零个或多个输入。

5) 输出

算法的目的就是要“得到”结果，一个算法得到的结果就是算法的输出，没有输出的算法是无实际意义的，因此，一个算法要有一个或多个输出。

3. 算法的描述

通过一定的符号或者语言将解决问题的方法和步骤描述出来就是算法的描述。描述算法有多种方法，常用的有：

(1) 用自然语言描述算法。

(2) 用流程图描述算法。

(3) 用 N-S 流程图描述算法。

(4) 用伪代码描述算法。

(5) 用计算机语言描述算法。

本书在进行算法描述时，采用伪代码和“类 C 语言”来进行描述，忽略了 C 语言的语法规则的细节，同时，做了一定的扩充和修改，使得描述出的算法更加清晰、明确，便于理解和分析。以下对其进行简明分类介绍：

(1) 定义常量及类型。

```
#define TRUE 1
#define FALSE 0
#define OK 1
#define ERROR 0
#define OVERFLOW -1
```

(2) 数据元素被约定为 ElemType 类型，需要根据具体情况，自行定义该数据类型。

(3) 算法描述函数的形式：

```
函数类型 函数名(函数参数表)
{
  语句序列;
}
```

为了简化函数的书写，提高描述的清晰度，一般规定除函数参数表中的参数需要说明数据类型外，函数中使用的局部变量可以不做变量说明，必要时给出相应的注释即可。另外，在书写算法时，应该养成对重点语句段落添加注解的良好习惯。

(4) 在算法描述中可以使用的赋值语句形式有：

```
简单赋值 变量名 = 表达式;
串联赋值 变量名1 = 变量名2 = … = 变量名n = 表达式;
成组赋值 (变量名1,…,变量名n)=(表达式1,…,表达式n);
结构赋值 结构名1 = 结构名2;
         结构名 =(值1,值2,…,值n);
条件赋值 变量名 = 条件表达式 ? 表达式1:表达式2;
```

(5) 在算法描述中可以使用的选择结构语句形式有：

```
条件语句1: if (表达式) 语句;
条件语句2: if (表达式) 语句;
           else 语句;
多分支语句: switch (表达式) {
           case 值1:语句序列1;break;
           case 值2:语句序列2;break;
           ...
           case 值n:语句序列n;break;
           default:语句序列n+1;
           }
```

(6) 在算法描述中可以使用的循环结构语句形式有：

```
for 循环语句 :for (表达式1;循环条件表达式;表达式2) 语句;
while 循环语句: while (循环条件表达式)
                语句;
do-while 循环语句: do {语句序列;
                   } while (循环条件表达式);
```

(7) 在描述算法中可以使用的结束语句形式有：

```
函数结束语句: ① return 表达式;
```

② return;
case 或循环结束语句： break;
异常结束语句： exit(异常代码);

(8) 在算法描述中可以使用的输入/输出语句形式有：

输入语句： scanf([格式串],变量名 1,…,变量名 n);
输出语句： printf([格式串],表达式 1,…,表达式 n);
方括号([ ])：中的内容是可以省略的部分。

(9) 在算法描述中使用的注释格式为：

//文字序列 或 /* 文字序列 */

(10) 在算法描述中可以使用的扩展函数有：

求最大值 max(表达式 1,…,表达式 n):这个函数返回参数表中 n 个表达式计算结果中的最大值。
求最小值 min(表达式 1,…,表达式 n):这个函数返回参数表中 n 个表达式计算结果中的最小值。

1.3.2　算法性能分析

对于同一问题的解决，不仅仅只有一种算法。从解决同一问题的多种算法中选择出较为合适的一种，或者进行算法分析和评价后知道如何对现有的算法进行改进，从而设计出更好的算法。也就是说算法分析的任务是对设计出的每一个具体的算法，利用数学工具，讨论各种复杂度，以探讨某种具体算法适用于哪类问题，或某类问题宜采用哪种算法。算法的复杂度分时间复杂度和空间复杂度。

那么如何来评价算法的好坏呢？也就是算法设计的要求，主要从以下几方面进行考虑。

(1) 算法的正确性：要求算法能够正确地执行预先规定的功能，并达到所希望达到的性能。

(2) 算法的易理解性：为了便于人们进行阅读和理解，易于调试和修改。

(3) 算法的稳健性：算法中拥有对输入数据、打开文件、读取文件记录、分配内存空间等操作的结果检测，并通过与用户对话的形式做出相应的处理选择。

(4) 算法的效率：算法的效率主要指时间和空间上的。

算法的时间效率指算法变成程序后在计算机上的运行时间，它大致等于计算机进行一次简单操作与算法中进行简单操作次数的乘积。

算法的空间效率指在算法的执行过程中，所占据的辅助空间数量。辅助空间就是指算法代码本身和输入/输出数据所占据的空间和算法临时开辟的存储空间单元。在有些算法中，占据辅助空间的数量与所处理的数据量有关，而有些则无关。在设计算法时，应该注意空间效率。

时间复杂度：在运行算法时所耗费的时间为 f(n)(即 n 的函数)。

空间复杂度：实现算法所占用的空间为 g(n)(也为 n 的函数)。

称 O(f(n))和 O(g(n))为该算法的复杂度。

一个算法在计算机上运行所花费的时间及在计算机存储器上所占用的存储空间是进行算法评价的主要方面。

1. 算法的时间效率

算法的运行时间的分析，主要是算法复杂度的分析方法及其运用。

首先了解一下几个概念。一个是时间复杂度，一个是渐近时间复杂度。前者是某个算法的时间耗费，它是该算法所求解问题规模 n 的函数，而后者是指当问题规模趋向无穷大时，该算法时间复杂度的数量级。

当评价一个算法的时间性能时，主要标准就是算法的渐近时间复杂度，因此，在算法分析时，往往对两者不予区分，经常是将渐近时间复杂度 T(n)=O(f(n))简称为时间复杂度，其中的 f(n)一般是算法中频度最大的语句频度(语句频度指的是语句重复执行的次数)。也就是说，成某一数量级，T(n)可以看作关于 Of(n)函数，此函数可看作取 f(n)的数量级的函数。

常见的时间复杂度，按数量级递增排列依次为：常数阶 O(1)、对数阶 $O(\log_2 n)$、线性阶 O(n)、线性对数阶 $O(n\log_2 n)$、平方阶 On^2、立方阶 On^3、k 次方阶 On^k、指数阶 $O2^n$。

说明：若要独立于计算机的软、硬件系统来分析算法的时间耗费，则设每条语句执行一次所需的时间均是单位时间，一个算法的时间耗费就是该算法中所有语句的频度之和。

下面通过例子来分析算法的时间复杂度。

【例 1.6】 分析下列程序段的时间复杂度。

```
for(i=0;i<n;i++)
s=s+i;                          // n 次
```

该算法中所有语句的频度之和为：n

说明：对于顺序执行语句频度之和为单一语句执行的次数的最大值。

所以：T(n)=O(n)

【例 1.7】 分析下列程序段的时间复杂度。

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

分析：该算法中，外层循环控制下，内层执行语句频度是 n，而内层循环控制下语句执行 k 次，所以语句频度之和为 n*k

```
T(n)=O(n*k)
```

【例 1.8】 求两个 n 阶方阵的乘积 C=A×B，其算法如下。

```
# define n 100        /* n 可根据需要定义,这里假定为 100 */
void MatrixMultiply(int A[a],int B [n][n],int C[n][n])
//右边列为各语句的频度
int i ,j ,k;
for(i=0;i<n;i++)                              ……(1)   /* n */
  for(j=0;j<n;j++)                            ……(2)   /* n (n+1) */
    {C[i][j]=0;                               ……(3)   /* n^2 */
     for(k=0;k<n;k++)                         ……(4)   /* n^2(n+1) */
       C[i][j]=C[i][j]+A[i][k]*B[k][j];}      ……(5)   /* n^3 */
```

该算法中所有语句的频度之和(即算法的时间耗费)为

$$f(n)=2n^3+3n^2+2n+1$$

分析：

语句(1)的循环控制变量 i 要增加到 n，测试到 i=n 成立才会终止，它的频度是 n+1。但是它的循环体却只能执行 n 次。语句(2)作为语句(1)循环体内的语句应该执行 n 次，但语句(2)本身要执行 n+1 次，所以语句(2)的频度是 n(n+1)。同理可得语句(3)，语句(4)和语句(5)的频度分别是 n^2，$n^2(n+1)$和 n^3。

算法 MatrixMultiply 的时间复杂度 $T(n)=O(n^3)$。

【例 1.9】　交换 i 和 j 的内容。

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

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

如果算法的执行时间不随着问题规模 n 的增加而增长，即使算法中有上千条语句，其执行时间也不过是一个较大的常数。此类算法的时间复杂度是 O(1)。

【例 1.10】　分析下列程序的时间复杂度。

```
x=0;  y=0;              ……     (1)
 for(k-1;k<=n;k++)      ……     (2)
    x++;                ……     (3)
 for(i=1;i<=n;i++)      ……     (4)
   for(j=1;j<=n;j++)    ……     (5)
     y++;               ……     (6)
```

一般情况下，对步进循环语句只需考虑循环体中语句的执行次数，忽略该语句中的步长加 1、终值判别、控制转移等成分。因此，以上程序段中频度最大的语句是(6)，其频度为 $f(n)=n^2$，所以该程序段的时间复杂度为 $T(n)=O(n^2)$。

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。

算法的时间复杂度不仅仅依赖于问题的规模，还与输入实例的初始状态(最坏情况和最好情况)有关，具体实例分析在第 10 章的排序中予以讨论。

2. 算法的空间效率

一个算法的空间效率是指在算法的执行过程中，所占据的辅助空间数量。辅助空间就是除算法代码本身和输入/输出数据所占据的空间外，算法临时开辟的存储空间单元。在有些算法中，占据辅助空间的数量与所处理的数据量有关，而有些却无关。后一种是较理想的情况。在设计算法时，应该注意空间效率。

3. 算法性能选择

一个占用的存储空间小、运行时间短、其他性能也好的算法，也就是面面俱到的算法是很难做到的。因为,上述要求有时相互抵触：要节约算法的执行时间往往要以牺牲更多的