普通高等教育“十一五”国家级规划教材
高职高专计算机类专业规划教材

数据结构与算法实用教程

高佳琴　主编

SHUJU JIEGOU YU SUANFA SHIYONG JIAOCHENG

机械工业出版社
CHINA MACHINE PRESS

普通高等教育“十一五”国家级规划教材
高职高专计算机类专业规划教材

数据结构与算法实用教程

主　编　高佳琴
副主编　蒋　晖　乔明中
参　编　申燕萍　周国华
主　审　陆　兵

机械工业出版社

本书在简要回顾了基本的C程序设计概念的基础上，逐步引入与程序设计相关的算法与数据结构等基本概念，系统地介绍了顺序表、链表、队列与栈、树、图等基本数据结构，以及递归、查找与排序等多种算法。

本书的理论知识点涵盖了“全国计算机等级考试”及“中国计算机技术与软件专业技术资格考试”中程序员级的程序设计及算法基础。全部算法用C语言书写，并配有结构化流程图，结构清晰、重点难点突出、通俗易学，具有较好的可读性与可移植性。

全书共10章，每章都配有丰富的、类型多样的习题，并且提供了体现各主题基本任务的上机实验题。

本书是一本实践性、应用性很强的有关数据结构与常用算法的教材，可作为高职高专软件技术及相关专业的“数据结构”课程教材，对于软件技术从业人员也是一本很好的参考书。

图书在版编目(CIP)数据

数据结构与算法实用教程/高佳琴主编. —北京：机械工业出版社，2008.6

普通高等教育“十一五”国家级规划教材.

高职高专计算机类专业规划教材

ISBN 978-7-111-24128-7

Ⅰ.数… Ⅱ.高… Ⅲ.①数据结构-高等学校：技术学校-教材②算法分析-高等学校：技术学校-教材

Ⅳ.TP311.12

中国版本图书馆CIP数据核字（2008）第070665号

机械工业出版社（北京市百万庄大街22号 邮政编码100037）

策划编辑：王玉鑫

责任编辑：刘子峰 版式设计：张世琴 责任校对：程俊巧

封面设计：马精明 责任印制：邓 博

北京京丰印刷厂印刷

2008年7月第1版·第1次印刷

184mm×260mm·10.75印张·259千字

0 001—4 000册

标准书号：ISBN 978-7-111-24128-7

定价：19.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

销售服务热线电话：(010) 68326294

购书热线电话：(010) 88379639 88379641 88379643

编辑热线电话：(010) 68354423

封面无防伪标均为盗版

前　言

数据结构是计算机程序设计的重要基础，是计算机各专业的核心课程。本书以浅显易懂的文字和大量的例题对数据结构的相关主题进行讲解，对问题的解决方法与流程做了详尽的图形剖析，辅以结构化流程图与C语言程序代码进行算法描述，从而增进读者对问题的理解，帮助读者更好地理解和灵活地运用相关知识点。

本书共分10章，对应的主题分别为数据结构概论、顺序表、链表、栈与队列、字符串与数组、递归、树、查找、排序、图及常用算法介绍。

本书的主要特点有：

1. 从应用入手，首先介绍每一个主题的实用性，然后再分解成相关知识点进行剖析。

2. 涵盖了“全国计算机等级考试（二、三、四级）”及“中国计算机技术与软件专业技术资格考试”中程序员级别的考核内容。

3. 本书对算法的描述充分体现了健壮性及规范性，注重培养学生逐步形成良好的编程习惯。

4. 理论知识的阐述由浅入深，加强课程间的联系（特别是与C语言）。将抽象的理论阐述尽量用简明的形式化语言描述，省略了一些复杂的理论推导和数学证明。

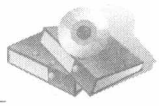
5. 本书附录B给出了《数据结构与算法》课程标准，该标准从教学内容、教学方法、考核方式等方面进行具体的阐述，对该课程教学和学习有较好的指导作用。

本书建议课时为64学时，上机时数为16学时。

本书由常州轻工职业技术学院高佳琴副教授任主编，参与编写的老师包括蒋晖、乔明中、申燕萍、周国华，陆兵老师对全部书稿进行了审阅。由于编者水平有限，错误之处在所难免，恳请广大读者批评指正。

编　者

目　录



第1章 概 述

本章介绍数据结构与算法课程的任务和性质，数据结构的相关术语，算法的定义、特征、表示及分析方法，并对数据结构与算法的C语言基础进行了简要回顾。

本章要点：

1）数据、数据元素、数据结构、数据的逻辑结构与物理结构的概念以及逻辑结构与物理结构间的关系。

2）算法的定义、特性，算法的时间复杂度和空间复杂度分析。

3）C语言指针的定义、指针的基本操作、动态分配函数等。

本章难点：

1）数据的逻辑结构和物理结构的关系。

2）算法的时间复杂性和空间复杂性分析。

自1946年2月第一台电子计算机ENIAC诞生以来，计算机的发展早已超出了人们的预料。计算机的应用不再局限于科学计算，而是更多地应用于数据处理、信息管理、实时控制等非数值计算的各个领域。

用计算机解决任何问题都离不开程序设计。程序设计的步骤如图1-1所示。

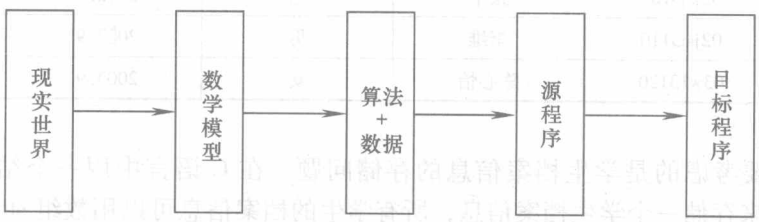


图1-1 计算机解决问题的一般步骤

1）从具体问题分析入手找出解决该问题的方法（数据模型）。

2）设计解决该问题的具体步骤（算法）。

3）选择程序设计语言和数据类型，编写代码（源程序），源程序经编译后得到可直接运行的程序（目标程序）。

程序设计在本质上是将一种数据形式转换成另一种数据形式的过程，因此除了要设计解决问题的算法外，还必须解决数据的组织形式及存储方法。所谓数据就是信息在计算机中的符号表示，是计算机处理信息的基本单位。无论是科学计算、数据处理、信息管理、实时控制，还是数据库技术都是对数据进行加工的过程。

为了编制“好”的程序，必须分析程序处理的数据特性及数据之间的关系，并设计一个好的算法。数据结构和算法是相辅相成的，即人们常说的“程序 = 数据结构 + 算法”。数据结构就是这样一门讨论“非数值应用问题中数据的表示、数据之间的关系及操作在计算机中的表示和实现”的学科。

数据结构是软件技术及相关专业的一门专业基础核心课程，要求学生必须具有一定的程序设计基础，能较熟练地运用高级语言（建议用 C 语言与 C ++ 语言）进行程序设计。它是操作系统、数据库原理、软件工程等专业课的前导课程。

1.1 什么是数据结构

本节先从一个简单的学生档案管理系统入手，引入数据结构的相关概念。

【例 1-1】 学生档案管理系统中的信息表示和存储。

问题描述：学生档案管理系统的主要功能包括：输入、修改、插入、删除、查找学生档案，并进行数据的统计（如统计男、女生比例等）。

分析：用计算机来处理学生档案管理系统首先要解决学生档案信息的归类和存储问题。最直观的方法是把学生档案信息用一个二维表（表 1-1）来表示。表中的一行表示一个学生的相关信息，用计算机专业术语可称为一个记录或一个结点。如果将一个学生的档案信息作为一个整体来考虑，以学号作为区分学生的标识符，那么所有学生的信息可以抽象地视为若干个学生的档案信息依次排列而成。这是一种最简单也是最常用的逻辑数据结构——线性表。

表 1-1 学生档案表

	学号	姓名	性别	入学年份	籍贯
第一条记录⟶	02jsj3101	张平	男	2002.9	江苏徐州
第二条记录⟶	02jsj3110	李维	男	2002.9	安徽黄山
第三条记录⟶	03w13120	陈心怡	女	2003.9	江苏无锡

下一步需要考虑的是学生档案信息的存储问题。在 C 语言中用一个结构体类型名为 student 的变量来存储一个学生档案信息，所有学生的档案信息可以用数组 stu 存储也可用链表存储。

学生档案信息管理中的类型定义及变量定义如下：

```
/*定义学生信息*/
typedef struct
{   char no[10];
    char name[10];
    char sex[2];
    char enrolldate[8];
    char address[20];
}student;
/*定义学生数组*/
```

```
student stu[50];
/*定义链表*/
typedef struct
{ student info;
  student *next;
}studnode;
```

在数组 stu 中，学生信息在内存中是顺序存放的，即第 i(1≤i≤50)个学生的信息 stu[i]存储在第 i-1 个学生的信息 stu[i-1]后，而在第 i+1 个学生的信息 stu[i+1]前。这种将存储顺序与逻辑顺序保持一致的存储结构就是顺序存储结构，如图 1-2 所示。

而在用链表存储信息时，信息在内存中存储的顺序与逻辑顺序不要求一致。它是通过为每一条记录增加一个存储下一个学生信息地址的信息项来表示学生的次序，这就是链式存储结构，如图 1-3 所示。

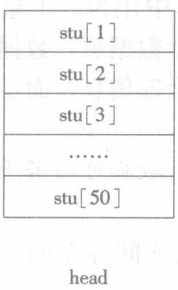


图 1-2　顺序存储结构

地址	学生档案信息	下一个地址
1000	02j sj 3110, ……	1220
1104	……	
1200	02j sj 3101, ……	1000
1220	02j sj 3120, ……	1104

图 1-3　链式存储结构

学生档案信息系统必须给用户提供查询、插入、删除、更新学生档案信息等基本操作功能，每一个功能都需要设计一个算法来实现。对于不同的存储结构，选择的算法也不同。

例如，若查找学号为 02jsj3120 的学生的姓名，对于数组 stu 而言，可以从 stu[1]开始比较，如果 stu[1].no 不等于 02jsj3120，则将 stu[2].no 与 02jsj3120 进行比较，依此类推直到找到 stu[k].no 等于 02jsj3120，返回 stu[k].name。如果用链式存储方式进行信息存储，则应先从箭头所指的头指针位置 1200 开始查找，若 head→no 不等于 02jsj3120，则从 head→next 域中得到下一个学生信息的地址 1000，再从 1000 开始的内存中读出学号信息，与 02jsj3120 进行比较，依次查找直到某结点的 no 等于 02jsj3120，返回 name 域的值。

结论：同一种数据结构，可以有不同的存储形式；对应于不同的存储形式，同一运算的实现算法也可能不同。

数据结构主要研究非数值应用问题中数据之间的逻辑关系和对数据的操作方式，同时还研究如何将这些具有逻辑关系的数据按一定的存储方式存放在计算机内。分析数据之间的逻

辑关系和确定数据在计算机内的存储结构是程序设计前两个必须完成的任务。

1.2 基本概念和术语

（1）数据　指所有能输入到计算机中并能被计算机程序处理的符号的总称。

注意：将现实世界中的信息转换成计算机输入和处理的数据除数值外，还有字符串、表格、图像甚至声音等。

（2）数据元素　在计算机程序中通常作为一个整体进行考虑和处理的基本数据单位。一个数据元素可以由若干个数据项组成，也可以只由一个数据项组成。数据元素又被称为元素、结点或记录。

（3）数据项　是不可分割的、具有独立意义的最小数据单位，数据项有时也被称字段或域。

学生档案信息表中每一行记录了一个学生的档案信息，在数据操作中作为一个整体考虑，对应为一个数据元素。这个记录中包含有学号、姓名、性别等若干个数据项。数据操作的基本单位是数据元素，如学生的插入或删除一定是对应于一个学生的全部信息，而不是对应于其中的某个数据项。

结论：数据、数据元素、数据项实际上反映了数据组织的三个层次：数据可由若干个数据元素构成，而数据元素又可以由一个或若干个数据项组成。

（4）数据逻辑结构　是指数据元素之间的抽象关联方式。数据元素之间存在的一种或多种特定的关系被称为数据的逻辑结构。通常用二元组表示：

Data structure = (D,R)

其中 D 表示数据元素的集合，R 表示 D 上元素的二元关系。用关系代数表示：

$D=\{d_i \mid 1\leqslant i\leqslant n\}$, $R=\{r_j \mid 1\leqslant j\leqslant m\}$

其中 r_j 表示数据元素之间存在关系，在关系代数中用有序偶表示：

$r_j=<d_k,d_t>, d_k\in D \wedge d_t\in D$

在 r_j 中称有序偶中第一个数据元素 d_k 是第二个数据元素 d_t 的前趋，而 d_t 称为 d_k 的后继。

人们常借助于图形直观地描述数据的逻辑结构，用小圆圈表示数据元素（结点），用直线表示数据元素之间的关系。

【例 1-2】 画出数据结构 DS1 的逻辑结构表示图。

问题描述：有一种数据结构 DS1 = (D1,R1)，其中 D = {A,B,C,D}，R1 = {<B,C>,<A,B>,<C,D>}，画出其逻辑结构表示图。

分析：数据结构 DS1 对应的逻辑结构如图 1-4 所示。

数据逻辑结构的类型可分为线性结构和非线性结构两种。

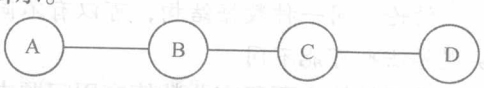


图 1-4　例 1-2 的逻辑结构表示图

1）线性结构。数据元素之间存在一对一的关

系，如图1-5a所示。其特点是除第一个元素和最后一个元素外，其他元素都有且只有一个直接前趋和直接后继。这是一种最常见，也是比较简单的数据结构。

线性结构的应用：仓库管理、教材管理等系统中处理的数据都是线性结构。

2）非线性结构。非线性结构指数据元素之间存在一对多或多对多的关系。它又可以细分为树形结构和图形结构。

①树形结构是指该结构中的数据元素之间存在一对多的关系，如图1-5b所示。其特点是该结构中除了有一个被称为根的结点没有前趋外，其余元素有且只有一个直接前趋，可以有多个后继。

树形结构的应用：操作系统的文件管理（资源管理器）、企业的组织机构等处理的数据都是树形结构。

②图形结构（网状结构）是最复杂的数据结构，数据元素之间存在多对多联系，如图1-5c所示。其特点是该结构中任何元素都可以有多个直接前趋，也可以有多个后继。

图形结构的应用：全国铁路网、高速公路网等数据是图形结构。

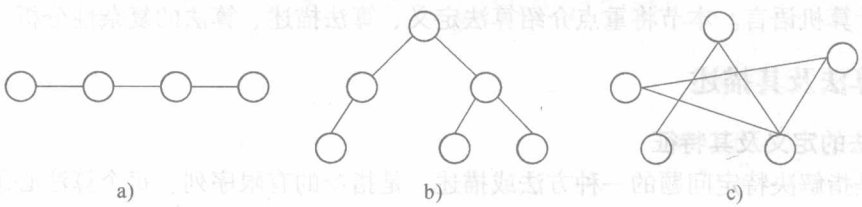


图1-5 三种基本逻辑结构

a）线性结构 b）树形结构 c）图形结构

【例1-3】 有一种数据结构K=(D,R),其中D={a,b,c,d,e,f,g},R={<a,b>,<a,c>,<a,d>,<b,e>,<c,f>,<c,g>}，画出其逻辑结构图，并指出其逻辑结构的类型。

分析：画出对应的逻辑结构图如图1-6所示。

从图1-6看出，除数据元素a没有前趋外，其他数据元素有且只有一个前趋；所有数据元素均有0个或多个后继，这是一种树形结构。

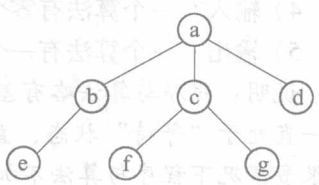


图1-6 例1-3逻辑结构图

（5）数据物理结构 数据在计算机存储器中的存放方式称为数据的物理结构，简称存储结构。

数据元素在计算机中主要有两种不同的存储方法：顺序存储结构和链式存储结构。例如在例1-1学生档案管理系统中学生档案信息就可以有两种存储形式，一种是以数组形式也就是顺序存储方式存储；另一种用指针类型存储即链式存储结构。

顺序存储的特点是在内存中开辟一组连续的空间（高级语言中的数组）来存放数据，数据元素之间的逻辑关系通过元素在内存中存放的相对位置来确定。

链式存储的特点是除了存储所涉及问题的相关信息外，还增加了指针域，记录下一个数据元素所对应的地址，通过指针反映数据元素之间的逻辑关系。

要点：数据的逻辑结构和物理结构是数据结构的两个密切相关的方面，同一种逻辑结构可以对应不同的存储结构。算法的设计取决于数据的逻辑结构，而算法的实现依赖于指定的

存储结构。

（6）数据类型　在用高级语言编写的程序中，所有的变量、常量或表达式都具有确定的数据类型。数据类型包含了数据的取值范围及基本操作运算，可以这样认为：数据类型是程序设计语言中已经实现了的数据结构。

例如：当 C 语言中的函数 strlen( )返回整型数据时，它就可以进行四则运算和取模运算，也可以通过%d 格式输出该函数的值。

数据类型可分为两类：一类是非结构的原子类型或简单数据类型，如 C 语言中的基本类型（整型、实型、字符型等）、指针类型和空类型；另一类是结构类型，它由原子类型组成，它的成分可以分解，如数组由若干相同类型的分量组成。

1.3　算法和算法分析

算法（algorithm）是程序设计的精髓，程序设计的实质就是构造解决问题的算法，并将其解释为计算机语言。本节将重点介绍算法定义、算法描述、算法的复杂性分析。

1.3.1　算法及其描述

1. 算法的定义及其特征

算法是指解决特定问题的一种方法或描述，是指令的有限序列。每个算法必须具有五大特性：

1）正确性。算法必须解决具体的问题，完成所期望的功能，给出正确的输出。

2）确定性。算法执行的每一步和下一步必须确定，不能有二义性。

3）有限性。一个算法必须由有限步组成。无限步组成的算法无法用计算机程序来实现，因此算法必须可以终止，不能进入死循环。

4）输入。一个算法有零个或多个输入。

5）输出。一个算法有一个或多个输出。

说明：程序与算法略有差异，程序可以不满足特征 3）。例如操作系统在用户未操作之前一直处于“等待”状态，直到出现新用户操作为止。由于大部分程序满足特征 3），因此在很多情况下程序与算法不加区别。

2. 算法的描述方法

算法可以用自然语言描述，但由于自然语言表达算法容易产生二义性，所以常使用专用的算法描述工具。常用的算法描述方法有三种。

（1）图形工具　用一些基本符号表示处理、输入、输出等操作，比较流行的框图有传统流程图和结构化流程图，如图 1-7 所示，其优点是直观、易懂。但传统流程图用来描述比较复杂的算法就显得不够方便，也不够清晰简洁。人们越来越习惯用结构化流程图来描述算法。

【例 1-4】　分别用传统流程图和结构化流程图描述一个算法。

问题描述：分别用传统流程图和结构化流程图描述下列问题：给定两个正整数 m 和 n，求最大公约数。

分析：将数学中求最大公约数的辗转相除法的求解过程进行分解，用标准的流程图基本符号表示如图 1-8a、b 所示。

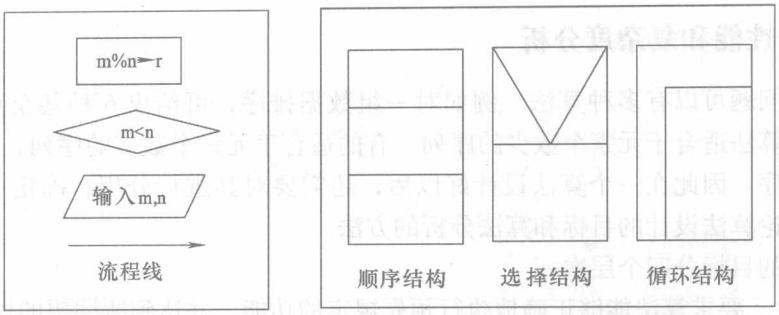


图 1-7　流程图

a）传统流程图　b）结构化流程图

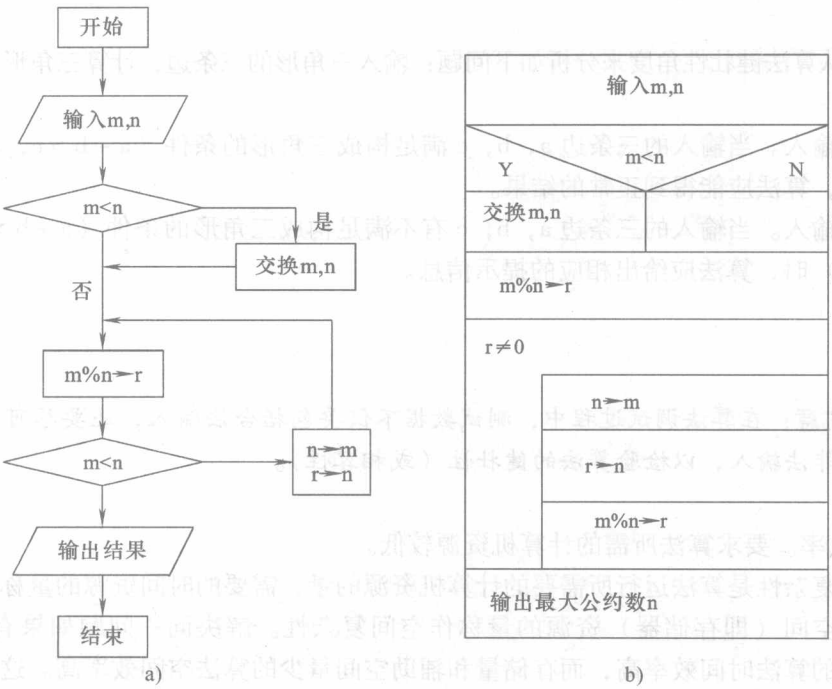


图 1-8　例 1-4 流程图

a）传统流程图　b）结构化流程图

（2）伪语言描述　伪语言与高级程序设计语言有些类似，有比较严格的外语法，如用 if …else…表示选择结构，用 while 表示循环结构，对内语法如变量定义等无明确要求。这种算法不能直接在计算机上运行，但专业设计人员经常使用伪语言来描述算法，它容易编写、阅读和统一格式。类 C 语言是目前应用较广的伪语言描述工具之一。

（3）C 语言编写的程序或函数　这是可在计算机上运行并获得结果的算法，使给定问题能在有限时间内被求解，通常这种算法也称程序。

说明：本书假设读者有一定的 C 语言编程经验，为便于对算法的理解和调试，所有算法的分析用结构化流程图描述，所有算法的实现用 C 语言编写。

1.3.2 算法性能和复杂度分析

解决一个问题可以有多种算法。例如对一组数据排序，可给出6种甚至更多种排序算法，有的排序算法适合于元素个数少的序列，有的适合于元素个数多的序列，有的则适合于基本有序的排序。因此在一个算法设计好以后，还需要对其进行分析，确定一个“好”的算法。下面讨论算法设计的目标和算法分析的方法。

算法设计的目标分四个层次：

1）正确性。要求算法能够正确地执行预先规定的功能，并达到所期望的性能要求。

2）易读性。算法应易于阅读和理解，以便调试、修改和扩充。

3）健壮性。正确的输入能得到正确的输出，这是算法必须具有的特性之一。但当遇到非法输入时，算法应能做出反应或处理（如提示信息等），而不会产生异常中断、死机或不正确的结果。

【例1-5】 从算法健壮性角度来分析如下问题：输入三角形的三条边，计算三角形的面积。

分析：

①合法输入。当输入的三条边a，b，c满足构成三角形的条件（a+b>c，a+c>b，b+c>a）时，算法应能得到正常的结果。

②非法输入。当输入的三条边a，b，c有不满足构成三角形的条件（a+b>c，a+c>b，b+c>a）时，算法应给出相应的提示信息。

注意： 在算法调试过程中，测试数据不仅要包括合法输入，还要尽可能多地包含各种形式的非法输入，以检验算法的健壮性（或相容性）。

4）高效率。要求算法所需的计算机资源较低。

算法的复杂性是算法运行所需要的计算机资源的量，需要的时间资源的量称作时间复杂性，需要的空间（即存储器）资源的量称作空间复杂性。解决同一问题如果有多个算法，执行时间短的算法时间效率高，而存储量和辅助空间量少的算法空间效率高。这两者都和问题的规模有关。

一个算法的“规模”和“基本操作”要视具体算法而定。“规模”一般是指输入量的数目，比如在排序问题中，问题的规模可以是定义为被排序的元素数目。“基本操作”一般是指简单操作如赋值操作等。

人们经常把运行算法所需要的时间T写成输入规模n的函数，记作T(n)。若将每次比较的时间记为c，则T(n)=c*n表明了顺序检索数组中最大元素的算法的时间是随着n的增长而线性增长。在难以精确计算基本操作执行次数的情况下，只要求出它关于n的增长率即可，忽略所有的常数和低次项，用大O表示法来表示算法的时间复杂度。故本例的时间复杂度可简记为O(n)。

在一个没有循环的算法中基本运算次数与问题规模无关，记为O(1)，也称为常数阶。作为时间复杂度衡量的函数，还有平方阶n^2，指数阶2^n，对数阶$\log_2 n$等。

各种不同的数量阶之间的关系如下：

$O(1) < O(\log_2 n) < O(n) < O(n\log_2 n) < O(n^2) < O(n^3) < O(2^n)$

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，算法运行过程中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。

【例 1-6】 分析查找一维整数数组中最大元素的算法。该算法从数组中下标为 0 的元素开始，遍历数组中的所有元素，在遍历过程中，将当前最大元素保存在变量 large 中。

```
int largest (int *array,int n)
{   int large,i;
    large = array[0] ;
    for(i =1; i <n ;i ++)
       if (array[i] >large)   large = array[i];
    return large;
}
```

分析：数组 array 中存放有 n 个整数，无论哪种算法，数据 n 的值越大，执行的时间都越长，因此可将问题的规模定为数组中数据元素的个数 n。基本操作是“比较”操作，即将数组中的一个整数和现有的最大整数作比较。上述算法中比较操作的次数为 n－1。算法的时间复杂度为 O(n)。

由于本例中空间效率与数据量无关，因此空间复杂度为 O(1)。

1.4 C 语言基础

为便于学生对数据结构课程的理解，现将数据结构课程与 C 语言关系紧密的基本语法和主要函数作简要回顾。

1.4.1 数组

将相同类型的若干变量按有序的形式组织起来称为数组。一个数组可分解为多个变量，故数组是一种构造数据类型。数组元素在数组中的位置由序号（下标）确定，只有一个下标的数组称为一维数组。

1. 一维数组的定义

一维数组定义方式如下：

类型说明符 数组名[常量表达式];

例如：int score[50];

注意：

1）数组名的命名规则与普通变量名相同。

2）常量表达式表示数组元素的个数，用方括号括起来。

3）定义了一维数组后，系统给该一维数组分配一组连续的存储空间，数组名表示该段存储空间的首地址。也就是说，数组类型是一种顺序存储结构。

2. 一维数组元素的引用

一维数组元素的引用方式如下：

数组名[下标]

例如：score[0]，score[i]，score[49]

说明：

1）数组不能整体引用，必须单个引用。

2）数组元素的下标可以是常量、变量或表达式。

3）C 程序规定，数组元素下标的下界为 0，上界为数组长度减 1。例如 score 数组的第一个元素是 score[0]，最后一个元素是 score[49]。

4）人们习惯使用单循环（for 循环结构），通过控制循环变量对一维数组进行访问。例如：

```
int i;
for(i=0;i<50;i++) scanf("%d",score[i]);
```

1.4.2 指针

指针是 C 语言中最重要的特征之一。指针是指某个对象（如简单变量、数组和函数）所占用的存储单元的地址。定义指针的目的是通过它访问内存单元。

利用指针变量可以方便地访问数组，动态地分配存储空间，指针是高级语言实现数据链式存储的唯一途径。

1. 指针变量定义

存储指针的变量称为指针变量，定义方式如下：

基类型 *指针变量名;

例如：int *p;

说明：

1）*表示其后定义的变量是一个指针变量。

2）基类型表示指针变量所指向的数据类型，即一个指针变量只能存储同一种数据类型变量的地址。

2. 指针引用

定义指针变量仅仅规定了其基类型，只有对其值进行初始化赋值，使其指向某个变量后才能引用。与指针相关的运算符有 & 与 *。

1）取址运算符 &，其作用是获取变量所占用的存储单元的首地址。

2）间接访问运算符 *，其作用是通过指针变量访问它所指向的变量。

例如：

```
int *p,i;                    /*定义 p 是基类型为整型的指针变量,i 是整型变量*/
i=3;                         /*给变量 i 赋初值 3*/
```

```
p = &i;                          /* 给指针变量 p 赋初值 */
printf("%d,%d",i, *p);           /* 对变量 i 的两种访问方式,对变量 i 直接访问,或通过指
                                    针变量 p 间接访问 */
```

3. 指针与数组

数组名代表数组首地址，属指针常量。用指针变量访问数组更加灵活、方便。

例如：int a[10]， *p = a;

说明：用指针变量 p 访问数组 a 比较方便，p+2 表示数组元素 a[2]的地址，*(p+2)与 a[2]等价。

4. 指针作为函数参数

指针作为函数参数的作用是将一个变量的地址传递到函数的指针变量形参，共享对应的实参的存储单元，因此形参值的改变将引起实参值的改变。

由于数组名代表指针常量，因此指针作函数参数，通常与数组的操作相关。例如：

函数定义语句为 void sort(int *a, int n)

函数调用语句为 sort(b, 10);

b 为整型数组名。在这里，形参为指针变量，实参为数组名。

【例 1-7】 用选择法对 10 个整数排序。

分析：在进行程序设计时把一个大的程序按照功能划分为若干个小的程序（模块），每个小的程序完成一个确定的功能。在这些小的程序之间建立必要的联系，互相协作完成整个程序要完成的功能。在 C 语言中，用函数来实现模块功能。本程序由两部分构成，主函数 main 和自定义函数 sort，通过指针进行参数传递，从而达到传递一批数据的目的。

```
void sort (int *b,int n)   /* 形参 b 为指针变量 */
{   int i,j,k,t;
    for(i=0;i<n-1;i++)
    {   k=i;
        for(j=i+1;j<n;j++)
        if(*(b+j)<*(b+k)) k=j;
        if(k!=i)
        {t=*(b+k);*(b+k)=*(b+i);*(b+i)=t;}
    }
}
void  main()
{   int a[10],i;
    for(i=0;i<10;i++)
    scanf("%d",&a[i]);
    sort(a,10);   /* 实参为数组名 */
    for(i=0;i<10;i++)
    printf("%d,",a[i]);
    printf("\n");
}
```