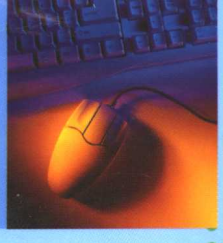
教育部高职高专规划教材
Jiaoyubu Gaozhi Gaozhuan Guihua Jiaocai



数据结构

陈 雁 主编

1.12

高等教育出版社

教育部高职高专规划教材

数 据 结 构

陈 雁 主 编

高 等 教 育 出 版 社

内 容 提 要

本书是教育部高职高专规划教材，是“计算机应用与维护专业系列教材”之一。

本书主要内容包括：数据结构的基本概念、线性表、栈、队列、树、图、查找、排序等，最后部分介绍了一些实用算法和面向程序设计中数据结构的描述方法。本书每章配有习题及上机实习题。

本书可作为高等职业学校、高等专科学校、成人高等学校以及本科院校二级职业技术学院计算机类专业或信息类相关专业的教材。也可作为非计算机专业学生选修课或辅修课的教材，还可作为计算机应用人员和工程技术人员的自学参考书。

图书在版编目(CIP)数据

数据结构/陈雁主编. —北京：高等教育出版社，
2001 (2003 重印)
教育部高职高专规划教材
ISBN 7-04-009950-0

Ⅰ.数… Ⅱ.陈… Ⅲ.数据结构-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(2001)第 035045 号

责任编辑 李 慧 **责任校对** 李 慧 **封面设计** 杨立新 **责任印制** 陈伟光

数据结构
陈 雁 主编

出版发行	高等教育出版社	**购书热线**	010-64054588
社 址	北京市西城区德外大街 4 号	**免费咨询**	800-810-0598
邮政编码	100011	**网 址**	http://www.hep.edu.cn
总 机	010-82028899		http://www.hep.com.cn
经 销	新华书店北京发行所		
印 刷	北京外文印刷厂		
开 本	787×1092 1/16	**版 次**	2001 年 9 月第 1 版
印 张	11	**印 次**	2003 年 6 月第 4 次印刷
字 数	260 000	**定 价**	21.00 元(含光盘)

本书如有缺页、倒页、脱页等质量问题，请到所购图书销售部门联系调换。

版权所有 侵权必究

出版说明

教材建设工作是整个高职高专教育教学工作中的重要组成部分。改革开放以来，在各级教育行政部门、学校和有关出版社的共同努力下，各地已出版了一批高职高专教育教材。但从整体上看，具有高职高专教育特色的教材极其匮乏，不少院校尚在借用本科或中专教材，教材建设仍落后于高职高专教育的发展需要。为此，1999 年教育部组织制定了《高职高专教育基础课程教学基本要求》(以下简称《基本要求》)和《高职高专教育专业人才培养目标及规格》(以下简称《培养规格》)，通过推荐、招标及遴选，组织了一批学术水平高、教学经验丰富、实践能力强的教师，成立了“教育部高职高专规划教材”编写队伍，并在有关出版社的积极配合下，推出一批“教育部高职高专规划教材”。

“教育部高职高专规划教材”计划出版 500 种，用 5 年左右时间完成。出版后的教材将覆盖高职高专教育的基础课程和主干专业课程。计划先用 2～3 年的时间，在继承原有高职、高专和成人高等学校教材建设成果的基础上，充分汲取近几年来各类学校在探索培养技术应用性专门人才方面取得的成功经验，解决好新形势下高职高专教育教材的有无问题；然后再用 2～3 年的时间，在《新世纪高职高专教育人才培养模式和教学内容体系改革与建设项目计划》立项研究的基础上，通过研究、改革和建设，推出一大批教育部高职高专教育教材，从而形成优化配套的高职高专教育教材体系。

“教育部高职高专规划教材”是按照《基本要求》和《培养规格》的要求，充分汲取高职、高专和成人高等学校在探索培养技术应用性专门人才方面取得的成功经验和教学成果编写而成的，适用于高等职业学校、高等专科学校、成人高校及本科院校举办的二级职业技术学院和民办高校使用。

教育部高等教育司

2000 年 4 月 3 日 2

前　言

本书是教育部高职高专规划教材，是依据教育部《高职高专教育数据结构课程教学基本要求》编写的。本书系统地介绍了各种数据结构的特点、存储结构和有关算法，并采用流行的C语言描述算法。主要内容包括数据结构的基本概念、线性表、栈、队列、树、图、查找、排序等，最后部分介绍了一些实用算法和面向对象程序设计中数据结构的描述方法，以便学生进一步学习和提高。

本教材力求体现高职高专的特点，本着理论够用，注重实用的原则，对传统的数据结构课程的教学内容进行了调整，对一些不常用的数据结构和算法进行了简化和忽略；对常用的数据结构和算法除进行详细介绍外，还在主要章节增加了应用实例，引导学生在理解基本内容的基础上，学习数据结构的使用。为便于教学，每章后面配有习题和上机实习题，并有专门章节进行数据结构程序设计指导，介绍算法书写、上机实习规范、程序调试技巧、实习报告整理等，还专门配有演示主要算法和应用实例的软件。全书概念表述清楚、简洁，内容由浅入深，强调实践环节，利于教学与自学。

本书可作为高等职业学校、高等专科学校、成人高等学校及本科院校举办的二级职业技术学院计算机类专业或信息类相关专业的教材，也可作为非计算机专业学生选修课或辅修课的教材，还可作为计算机应用人员和工程技术人员的自学参考书。

本教材第1章至第5章由顾小晶撰写；第6、7章由陈雁撰写；第8、9章由曾海撰写；本书附带的光盘由顾成喜制作。全书由陈雁统稿。

南京大学陈佩佩副教授在百忙中抽时间担任主审并提出了许多宝贵的修改建议，高等教育出版社给予了热情的帮助与指导。编写过程中，作者得到了所在单位苏州市职业大学的领导和同行的支持。在此一并表示衷心的感谢。

由于编者水平有限，书中难免有一些不足之处，请读者批评指正。

作者

2001年3月于苏州

此为试读，需要完整PDF请访问：www.ertongbook.com

目　录

第1章 绪 论

计算机科学是一门研究信息表示、组织和处理的科学，而信息的表示和组织直接关系到处理信息时的效率。随着计算机产业的迅速发展和计算机应用领域的不断扩大，早期计算机主要应用于科学计算，而现在更多地用于控制、管理和数据处理等方面。随之而来的，便是处理的数据量越来越大，数据的类型越来越多，数据的结构越来越复杂。因此，针对实际问题，如要编制出一个高效率的处理程序，就需要合理地组织数据，建立合适的数据结构，设计一个好的算法，提高程序执行的效率。“数据结构”这门学科就是在此背景下逐步形成和发展起来的。

在这一章里我们将引出数据、数据结构和算法等基本概念，并介绍评价算法的一般方法。

1.1 数据结构的基本概念和术语

1.1.1 引 言

在程序设计课程的学习中，我们知道用计算机解题的基本方法是：分析问题，确定数据模型；设计相应的算法；编写程序；反复调试程序直至得到正确的结果。有些问题的数据模型可以用具体的代数方程、矩阵等来表示。然而，更多的实际问题是无法用数学方程来表示的。下面给出三个简单的例子加以说明。

表 1-1-1 是一个学生基本情况表。表中有 30 个记录，按学号顺序排列，它们之间存在一对一的关系，这是一种线性结构，其主要操作有查找、修改、插入或删除等。

表 1-1-1 学生基本情况表

学 号	姓 名	性 别	班 级	…
9905001	李力	男	99101	
9905002	杜军	男	99101	
9905003	程霄寒	女	99101	
⋮	⋮	⋮	⋮	
9905030	方勇	男	99101	

图 1-1-1 表示的是某高校的专业设置情况。在图 1-1-1 中可以把一所高校名称看成树根，把下设的若干个系名看成它的树枝中间结点，把每个系的若干个专业方向看成树叶，这就形成一个树形结构。树形结构通常用来表示结点的分层组织，结点之间是一对多的关系。对于树形结构的主要操作是遍历、查找、插入或删除等。

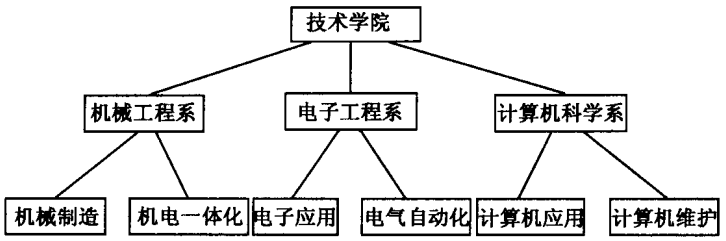


图 1 - 1 - 1　树型结构示例

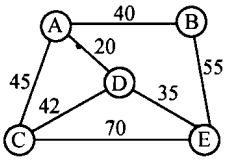


图 1 - 1 - 2　图型结构示例

图 1 - 1 - 2 是一个描述若干个城镇之间的公路网。图中每个顶点代表一个城镇，边表示城镇之间的道路。显然在图 1 - 1 - 2 中各个顶点之间的关系更加复杂，它们是一种多对多的关系。具有这种关系的结构称之为图形结构。在实际应用中假设从某个原料产地把原料运往各加工地，需要制定一个运输方案使得运输费用最省。

类似的还有图书档案类问题、棋类对弈问题、通信网络问题等。对于这些非数值问题的描述都是如上述的表、树和图之类的数据结构，这类数据结构中元素和元素之间都存在着相互的关系。因此，数据结构是一门研究非数值计算的程序设计中计算机的操作对象以及它们之间的关系和操作等的学科。

1968 年，美国的 D.E.Kunth 教授开创了数据结构的最初体系，他的名著《计算机程序设计技巧》较为系统地阐述了数据的逻辑结构和存储结构及其操作。随着计算机科学的飞速发展，应用领域的不断扩大，到 20 世纪 80 年代初期，数据结构的基础研究日臻成熟，成为一门完整的学科。

“数据结构”是计算机专业的一门专业基础课。它为操作系统、数据库原理、编译原理等后继专业课程的学习奠定了基础。数据结构涉及到各方面的知识，如计算机硬件范围的存储装置和存取方法；计算机软件范围中的文件系统，数据的动态存储与管理，信息检索；数学范围的许多算法知识，还有一些综合性的知识，如编码理论、算子关系、数据类型、数据表示、数据运算、数据存取等。因此，数据结构是数学、计算机硬件、软件三者之间的一门核心课程。

1.1.2　数据结构的有关概念及术语

数据(Data)是描述客观事物的数字、字符以及所有能输入到计算机中并被计算机程序处理的符号的集合。计算机输入和输出的数据除了数字以外，还有字符串，即用英文、汉字或其他语种字母组成的词组、语句，以及表示图形、声音、光和电的符号等。

数据元素(Data Element)是数据的基本单位，在计算机程序中通常作为一个整体进行考虑和处理。例如，图 1 - 1 - 1 中的“树”中的一个专业就是一个数据元素。数据元素可以由一个或多个数据项组成，例如，表 1 - 1 - 1 中每个学生的信息作为一个数据元素，每个数据元素由学号、姓名、性别、班级等七个数据项组成。数据项(Data Item)是有独立含义的数据的最小单位。

数据对象(Data Object)是性质相同的数据元素的集合，是数据的一个子集。例如，整数数据对象是集合{0，±1，±2，…}，字母字符数据对象是集合{‘A’，‘B’，…，‘Z’}。

数据结构(Data Structure) 是指相互之间存在着一种或多种关系的数据元素的集合。严格

此为试读，需要完整PDF请访问：www.ertongbook.com

地讲,数据结构应包括两方面的内容:数据的逻辑结构和物理结构。数据的逻辑结构是指各数据元素之间的逻辑关系,是用户按使用需要建立起来的数据的组织形式,是独立于计算机的。例如,表 1-1-1 和图 1-1-1 表示的就是数据的逻辑结构。数据的物理结构又称为数据的存储结构,是指数据的逻辑结构在计算机内的表示方法,即存储形式。对机器语言来说,这种存储形式是具体的,但在高级语言的层次上,就可以借助于它的数据类型来描述这种存储形式的实现细节。常见的存储结构有顺序存储结构和链式存储结构。

数据结构课程主要研究数据的逻辑结构、相应的存储结构以及定义在它们之上的一组运算,并设计出相应的算法,同时还必须考虑执行算法时的时空效率。

数据类型(Data Type)是程序设计语言中所允许使用的变量种类。一个数据类型不仅定义了相应变量可以设定的值的集合,而且还规定了对变量允许进行的一组运算及其规则。例如,C 语言中的整型(int)这个数据类型,它的值集是[-32768,32767],主要运算有 +、-、*、/、%(取模运算)等。从硬件的角度看,它们的实现涉及到"位"、"字节"、"字"、"位的运算"等等。但对用户的来说,只需了解整数的各种运算的抽象特性,而不必了解计算机实现这些运算的细节,就可使用高级语言进行程序设计。所以,可以把数据类型看作是程序设计语言中已经实现了的数据结构。

抽象数据类型(Abstract Data Type,简称 ADT)是指基于一切逻辑关系的数据类型以及定义在这个类型之上的一组操作。抽象数据类型的定义取决于客观存在的一组逻辑特性,而与其在计算机内如何表示和实现无关,即不论其内部结构如何变化,只要它的数学特性不变,都不影响其外部使用。在某种意义上讲,抽象数据类型和数据类型实质上是一个概念。例如,整数类型就是一个简单的抽象数据类型。因此,"抽象"的意义在于数学特性的抽象。当然,抽象数据类型的范畴更广,不仅限于各种不同的计算机处理器中已定义并实现的数据类型,还包括设计软件系统时用户自己定义的复杂数据类型。所定义的数据类型的抽象层次越高,含有该抽象数据类型的软件复用程度就越高。

抽象数据类型由元素、结构和操作三部分组成。例如,线性表的抽象数据类型可描述如下:

ADT Linear_List

数据元素　所有 a_i 属于同一数据对象,$i=1,2,\cdots,n$　($n\geqslant 0$)

逻辑结构　所有数据元素 a_i 存在次序关系(a_i,a_{i+1}),a_1 无前驱,a_n 无后继。

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;
GetElem(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 中位置;
Insert(L,i,x);	在线性表 L 中第 i 个元素之前 (或之后)插入一个新元素 x;
Delete(L,i);	删除线性表 L 中的第 i 个元素;
⋮	

ADT 并不强调在哪一级工具语言中的实现,关注的是如何通过所定义的各种操作去完成某个算法任务。下面就是一个利用线性表的抽象数据类型实现其他算法的例子。

例 1.1　已知两个集合 A 和 B,现要求一个新的集合 $A=A-B$。

分析:假设用线性表 LA 和 LB 分别表示集合 A 和 B,只要将同在 LA 和 LB 中的元素从 LA 中删除即可。算法描述如下:

```
viod  Difference(Linear_List LA,Linear_List LB){
      n = Length(LB);
      for (i = 1;i <= n;i ++){
          x = GetElem(LB,i);
          k = Locate(LA,x);
          if(k ! = 0) Delete(LA,k); /* x 在 LA 和 LB 中 */
          }
    }/* Difference */
```

在这个例子中,尚未涉及线性表和它的操作在工具语言层的具体实现,仅通过线性表的抽象数据类型的 Length、GetElem、Locate 和 Delete 操作完成了求集合 $A - B$ 的算法。可见凭借抽象数据类型的支持,可以有效地研究问题,设计算法和分析算法。

在具体程序设计时,需要通过工具语言来表示和实现 ADT。具体实现时依赖于所选择的高级语言的功能。一般来说,可以有以下几种不同的方法。一种是传统的面向过程的程序设计方法,也就是根据数据的逻辑结构选定合适的存储结构,根据所要求的操作设计出相应的算法(函数)。这也是本书所采用的方法。另一种是"包"、"模块"的设计方法:Ada 语言提供了"包"(package),Module - 2 类语言提供了"模块"(module)结构,TURBO PASCAL 语言提供了"单元"(unit)结构,用这类结构实现 ADT 比起第一种方法有一定的进步。第三种是面向对象的程序设计(Object _ Oriented Programming,简称 OOP),在面向对象的程序设计语言中,存储结构的说明和操作函数的说明被封装在一个整体结构中,这个整体结构称之为"类"(class),属于某个"类"的具体变量称之为"对象"(object)。OOP 与 ADT 的实现更加接近和一致。在本书的第 9 章将对 OOP 作进一步的介绍。

1.2 算法和算法描述

1.2.1 什么是算法

算法(algorithm)是对特定问题求解步骤的一种描述,它是指令或语句的有限序列。一个算法一般具有下列五个重要特性。

有穷性:一个算法必须总是在执行有穷步之后结束,即必须在有限时间内完成。

确定性:算法中的每一条指令必须有确切的含义,不能产生多义性。

可行性:算法中的每一条指令必须是切实可行的,即原则上是可以通过已经实现的基本运算执行有限次来实现的。

输入:一个算法有零个或多个输入,这些输入取自于特定对象的集合。

输出:一个算法有一个或多个输出,这些输出是同输入有某个特定关系的量。

在计算机科学研究中,算法与数据结构是相辅相成的。解决某一特定类型问题可以选定不同的数据结构,而且选择恰当与否直接影响算法的效率。反之,一种数据结构的优劣要由实现各种操作时的算法来体现。

1.2.2 算法描述工具——类 C 语言

算法需要用一种语言来描述。为了便于理解和掌握算法的思想和实质,本书采用类 C 语

言进行算法描述。类 C 语言实际上是对 C 语言的一种简化，保留了 C 语言的精华，忽略了 C 语言语法规则中的一些细节，这样描述出的算法清晰、直观、便于阅读和分析。

本书在用类 C 语言描述算法时作如下的约定：

1. 问题的规模用 MAXSIZE 表示。具体尺寸可以由用户预先定义，例如：

```
#define  MAXSIZE  100
```

2. 数据元素的类型约定为 ELEMTP。具体的类型可以由用户在使用时定义，例如：

```
typedef  int  ELEMTP;
```

3. 数据的存储结构用类型定义(typedef)描述，在书写算法之前进行说明。

4. 算法以函数形式描述：

类型标识符　函数名(形式参数表)

/* 算法说明 */

{ 语句序列 }

除了形参类型需要说明外，算法中其他变量的类型说明省略不写，这样使算法的处理过程更加突出明了。算法说明是一个完整算法不可缺少的部分，它可以包括算法的功能、数据的存储结构、形式参数的含义和输入输出属性等。

下面是书写算法的一般步骤。

设某班级有 n 个学生，设计算法将他们某一门课程的成绩按由高到低的顺序排序，并且顺序输出他们成绩和相应的学号。

分析：选用一维数组作存储 n 个学生的成绩和学号。每个数组元素有两个域：一个是学号域，一个是成绩域。

```
typedef struct{
   int  no;      /*学号域*/
   int  score;   /*成绩域*/
}ELEMTP;
```

算法描述：

算法 1.1

```
void sort( ELEMTP  S[MAXSIZE],int n)
/* 对数组 S 中的 n 个数据按由大到小的顺序排序并输出，n < MAXSIZE */
{for(i = 1;i < n;i ++)
   for(j = i;j <= n;j ++)
     if(S[i].score < S[j].score)
        {t = S[i];S[i] = S[j];S[j] = t;}
 for(i = 1;i <= n;i ++)
   printf(''%8d%8d%8d\n'',i,S[i].no,S[i].score);
}/*sort*/
```

1.3 算法分析技术初步

对于一个给定问题的求解，往往可以设计出若干个算法。那么如何评价这些算法的优劣呢？正确性是评价一个算法的首要条件。一个正确的算法是指在合法的数据输入下，能在有

限的运行时间内，得出正确的结果。此外，主要考虑执行算法所耗费的时间和执行算法所占用的存储空间。

1.3.1 时 间

一个算法的执行时间等于其所有语句执行时间的总和，而任一语句的执行时间为该语句的执行次数，也称语句频度(Frequency Count)与该语句执行一次所需时间的乘积。但当算法转换为程序之后，每条语句的执行时间取决于机器的硬件速度、指令类型、以及编译所产生的代码质量，这是很难确定的。因此，我们只是将算法中基本操作重复执行的次数作为算法执行时间的量度。

算法中基本操作重复执行的次数依据算法中最大语句频度来估算，它是问题规模 n 的某个函数 $f(n)$，算法的时间量度记作 $T(n)=O(f(n))$，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，称为算法的渐近时间复杂度(asymptotic time complexity)，简称时间复杂度。时间复杂度往往不是精确的执行次数，而是估算的数量级，它着重体现的是随着问题规模 n 的增大，算法执行时间的变化趋势。

例如，在下列三个程序段中：

```
(a) x = x + 1;
(b) for(i = 1;i <= n;i ++ ) x = x + 1;
(c) for(j = 1;j <= n;j ++ )
        for(k = 1;k <= n;k ++ ) x = x + 1;
```

语句 x = x + 1; 的频度分别为 1、n 和 n^2，则(a)的时间复杂度可记为 $O(1)$；在(b)中，赋值语句于 for 循环之中，故要执行 n 次，其执行时间和 n 成正比，时间复杂度应记为 $O(n)$；在(c)中，赋值语句要执行 n^2 次，其执行时间和 n^2 成正比，则时间复杂度应记为 $O(n^2)$。

现在来分析算法 1.1 的时间复杂度。算法中有一个二重循环，if 语句的执行频度为：

$$n+(n-1)+(n-2)+\cdots+3+2+1=n(n+1)/2$$

它的数量级为 $O(n^2)$。算法中输出语句的语句频度为 n，数量级为 $O(n)$。该算法的时间复杂度以 if 语句执行频度来估算(忽略输出部分)，则记为 $O(n^2)$。通常将这些时间复杂度分别称为常量阶、线性阶和平方阶，算法还可能呈现的时间复杂度有指数阶等。不同数量级时间复杂度的形状如图 1-3-1 所示。

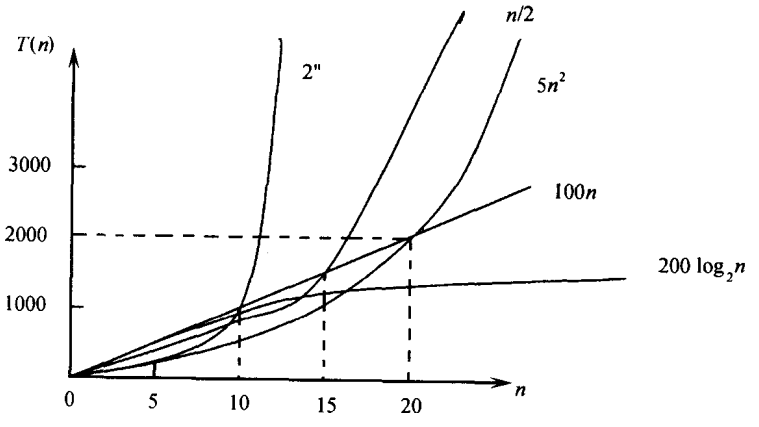


图 1-3-1 各种数量级的时间复杂度

从图中可见，随着问题规模的增大，其时间消耗也在增大，但它们的增长趋势明显不同。如果对于一个问题所设计的两种不同算法，算法 A 的时间复杂度为 $O(\log_2 n)$，算法 B 的时间复杂度为 $O(n^2)$。由图可知，随着问题规模 n 的增大，算法 B 所消耗时间迅速增大，而算法 A 增大趋向平缓，显然，算法 A 运行速度较快，可以认为算法 A 优于算法 B。

1.3.2 空　间

空间是指执行算法所需用的存储空间，包括算法本身所占用的存储空间、输入数据占用的存储空间以及算法在运行过程中的工作单元和实现算法所需的辅助空间。

空间复杂度(Space Complexity)可类似于时间复杂度的讨论。它与问题规模 n 的函数关系表示为：$S(n) = O(f(n))$。

习　题

1. 简述下列术语：

数据元素、数据、数据对象、数据结构、存储结构和算法。

2. 试写一算法，自大至小依次输出顺序读入的三个整数 x，y 和 z 的值。

3. 举出一个数据结构的例子，叙述其逻辑结构、存储结构、运算等三方面的内容。

4. 分析下列算法的的时间复杂度：

(1)
```
int prime(int n)
/* 判断 n 是否为素数,n≥0 */
{for (i = 2;i < sqrt(n);i ++)
    if(n % i == 0) return 0;
 return 1;
}
```

(2)
```
long sun(int n)
 /* n 为一个正整数 */
 {s = 0;
   for (i = 1;i <= n;i ++){
      for (p = 1,j = 1;j < i;j ++)
         p = p * j;
       s + = p;
      }
   return s;
 }
```

第2章　线性表和数组

在本章和下一章中，我们将讨论线性结构。线性结构的特点是，数据元素之间的关系是线性的。数据元素可以看成是排列在一条线上或一个环上。线性表(Linear _ List)是最简单且最常用的一种数据结构。本章讨论线性表的逻辑结构、存储结构以及相关的操作。数组在某种意义上也可看作是一种线性结构，因此在本章一起讨论。

2.1　线性表的逻辑结构

2.1.1　线性表的定义

线性表是 $n(n \geqslant 0)$个数据元素组成的有限序列。一般记作：

$$L=(a_1, a_2, \cdots a_i, \cdots, a_n)$$

表中的元素可以是一个数、一个符号或是由多个数据项组成的复杂信息，但同一线性表中的元素必须属于同一数据对象。例如，英文字母表(A,B,C,D,…,X,Y,Z)和表 2-1-1 的学生登记表都是线性表。

表 2-1-1　学生登记表

学　号	姓　名	性　别	年　龄	班　级	…
9905010	程晓珉	女	18	计 99	…
9905012	方正飞	男	19	计 99	…
9905013	刘津津	男	19	计 99	…
…	…	…	…	…	…

线性表的逻辑结构是通过元素之间的相邻关系体现的：a_1 为开始结点，a_n 为终端结点；a_{i-1}为 a_i 的直接前趋结点(predecessor)$(2 \leqslant i \leqslant n)$；$a_{i+1}$为 a_i 的直接后继结点(successor) $(1 \leqslant i \leqslant n-1)$。线性表中元素的个数$(n)$称为该表的长度。长度为零$(n=0)$的表称为空表。

2.1.2　线性表的基本操作

线性表是一种灵活的数据结构，其表长可根据不同的操作增长或缩短。对于线性表有以下几种常用的基本操作：

(1) InitList(L);　　建立一个空的线性表 L;

(2) GetElem(L,i);　取线性表 L 中的第 i 个元素;

(3) Length(L);　　求线性表 L 的长度;

(4) Locate(L,x);　确定元素 x 在线性表 L 中位置;

此为试读，需要完整PDF请访问：www.ertongbook.com