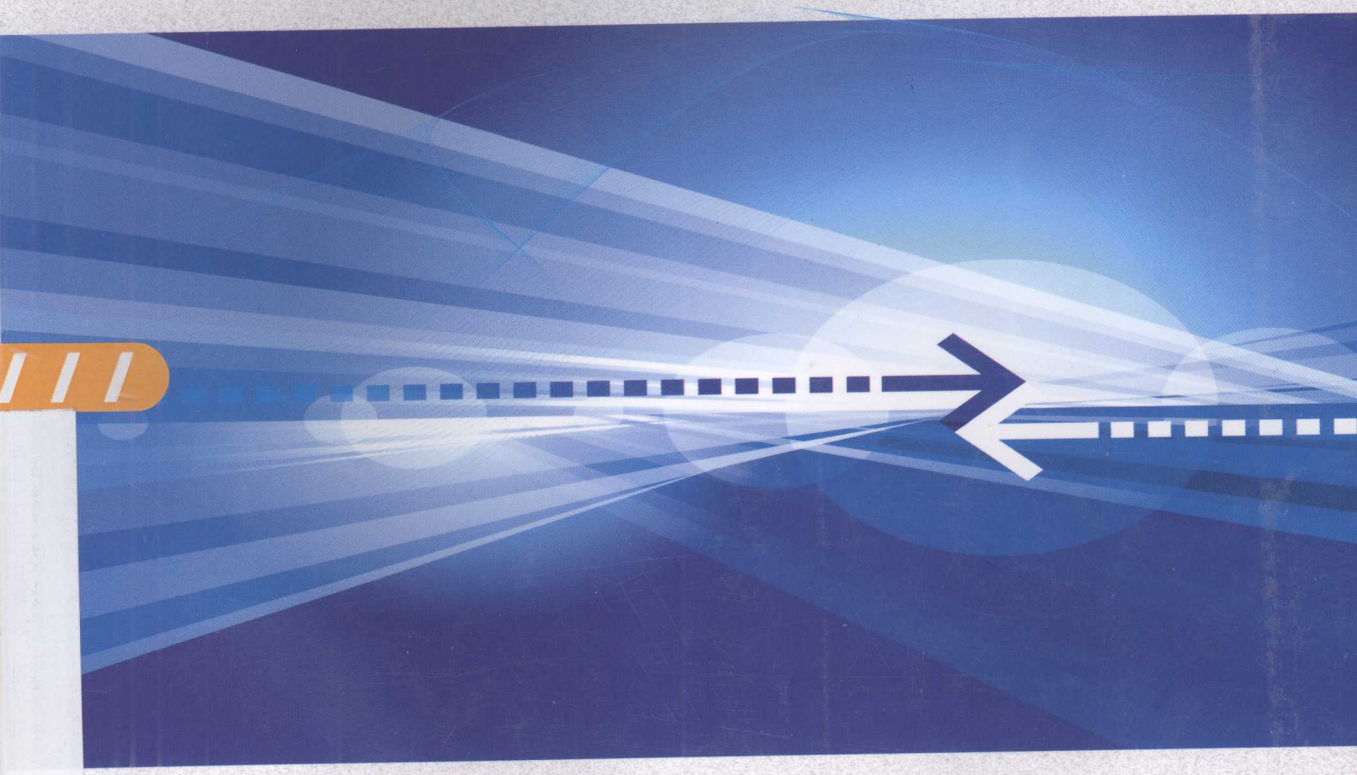
普通高等教育“十二五”规划教材

数据结构项目化教程

SHUJU JIEGOU XIANGMUHUA JIAOCHENG

主编 叶茂功 代文征



国防工业出版社
National Defense Industry Press

013069658

TP311.12-43
177

数据结构项目化教程

主审　孔繁民
主编　叶茂功　代文征
参编　邵开丽　刘寒冰　郑良仁

北京航空航天大学
★ 藏书 ★
图书馆

TP311.12-43
177

国防工业出版社
·北京·

北航　C1677793

01308358

内容简介

本书是为应用类高等学校学生学习“数据结构”课程编写的。书中介绍了数据处理领域中常见的数据结构及典型运算和相关综合应用项目。主要内容包括:数据结构的基本概念,线性表的顺序存储实现、链式存储实现及施加在其上的典型运算,树与二叉树的存储及施加在其上的运算的实现,图的存储及施加在其上的运算的实现,查找技术,排序技术。

本书通俗易懂,案例丰富,针对数据结构的运算公式化明显,适合各类计算机专业学生使用,本书可作为学习数据结构课程的教材,也可作为自学教材及各类培训班的教材。

图书在版编目(CIP)数据

数据结构项目化教程/叶茂功,代文征主编. —北京:国防工业出版社,2013.9
ISBN 978-7-118-08990-5

Ⅰ.①数... Ⅱ.①叶... ②代... Ⅲ.①数据结构-教材 Ⅳ.①TP311.12

中国版本图书馆 CIP 数据核字(2013)第 211969 号

※

国防工业出版社出版发行
(北京市海淀区紫竹院南路 23 号 邮政编码 100048)
北京奥鑫印刷厂印刷
新华书店经售
*
开本 787×1092 1/16 印张 13¼ 字数 301 千字
2013 年 9 月第 1 版第 1 次印刷 印数 1—4000 册 定价 29.00 元

(本书如有印装错误,我社负责调换)

国防书店:(010)88540777 发行邮购:(010)88540776
发行传真:(010)88540755 发行业务:(010)88540717

前　言

数据结构是高校计算机、信息类专业的基础与核心课程，数据结构主要研究数据的逻辑结构、存储结构以及对数据施加的各种运算。通过对该课程的学习，学生能够在实际应用中对事务进行分析，建立合适的逻辑结构和存储结构，并选择和使用较好的数据处理方法，以编写出相应的算法，最后在计算机系统上调试、运行和实现算法。

本书介绍的是数据结构学科成熟而实用的知识，摈弃了那些深奥难懂而又过时少用的内容；在写法上力求条理清楚、层次分明、内容连贯、循序渐进、简明扼要，便于阅读和自学；在各种运算方法和算法的分析上，力求细致、生动、深入、透彻，便于理解。

本书共9章。第1章概要介绍数据的逻辑结构和存储结构。第2～4章主要介绍线性表、栈、队列、数组等具体线性数据结构的定义、存储结构，以及相应运算的方法和算法。第5章和第6章主要介绍树、二叉树、霍夫曼树、二叉排序树等具体树结构的定义、存储结构，以及建立、遍历、查找等运算的方法和算法。第7章主要介绍图结构的定义、图的存储结构，以及图的建立、遍历、最小生成树、拓扑排序等运算的方法和算法。第8章主要介绍顺序、二分、索引、散列等查找的方法和算法。第9章主要介绍选择、插入、堆、交换、归并等排序的方法和算法。

本书给出的所有算法和程序都采用C语言在WINTC运行环境下调试通过，在一定程度上确保算法是正确和有效的。

本书配有实践教材，在实践教材中每章的后面都有丰富的各种类型的练习题，这些措施有利于学生作为教材或教学辅导书使用。

在本书的构思和编写过程中得到了教务处李高申处长的热情帮助，得到了孔繁民的认真指导，得到了软件教研室同事的真诚帮助，在此谨向他们表示衷心感谢！

由于编者水平有限，不足之处在所难免，敬请同行和广大读者指正。

叶茂功

2011年6月

目　录

此为试读,需要完整PDF请访问: www.ertongbook.com

第1章 绪 论

计算机科学是一门研究数据表示和数据处理的科学。数据是计算机可以直接处理的对象。无论是进行科学计算或数据处理、过程控制，还是对文件的存储和检索及数据库技术等，都是对数据进行加工处理的过程。因此，要设计出一个结构好、效率高的程序，必须研究数据的特性与数据间的相互关系及其对应的存储表示，并利用这些特性、关系设计出相应的算法和程序。

1.1 数据结构的概念

“数据结构”是计算机专业基础课，是十分重要的核心课程。计算机系统软件和应用软件都要用到各种类型的数据结构。因此，要想更好地运用计算机来解决实际问题，仅掌握几种计算机程序设计语言是难以应付众多复杂的课题的。要想有效地使用计算机、充分发挥计算机的性能，必须学习和掌握好数据结构的有关知识。

掌握好“数据结构”这门课程的知识，对于学习计算机专业的其他课程，如操作系统、编译原理、数据库管理系统、软件工程、人工智能等都是十分必要的。

1.1.1 学习数据结构的目的

在计算机发展的初期，人们使用计算机的主要目的是处理数值计算问题。当使用计算机来解决一个具体问题时，一般需要经过下列几个步骤：首先要从该具体问题抽象出一个适当的数学模型，然后设计或选择一个解此数学模型的算法，最后编出程序进行调试、测试，直至得到最终的解答。

由于当时所涉及的运算对象是简单的整型、实型或布尔类型数据，所以程序设计者的主要精力是集中于程序设计的技巧上，而无需重视数据结构。随着计算机应用领域的扩大和软、硬件的发展，非数值计算问题显得越来越重要。据统计，当今处理非数值计算性问题占用了90%以上的机器时间。这类问题涉及的数据结构更为复杂，数据元素之间的相互关系一般无法用数学方程式加以描述。因此，解决这类问题的关键不再是数学分析和计算方法，而是要设计出合适的数据结构，才能有效地解决问题。

教学计划编排问题。一个教学计划包含许多课程，在教学计划包含的许多课程之间，有些必须按规定的先后次序进行，有些则没有次序要求。即有些课程之间有先修和后续的关系，有些课程可以任意安排次序。这种各个课程之间的次序关系可用一个称为图的数据结构来表示，如图1.1所示。

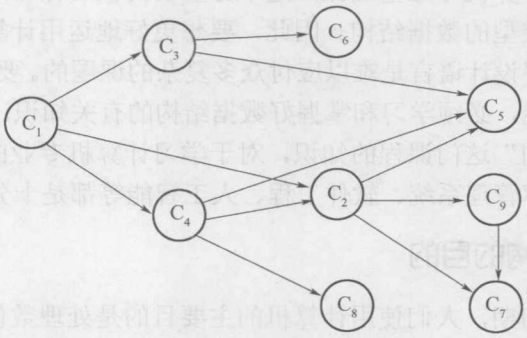
描述非数值计算问题的数学模型不再是数学方程，而是如表、树、图之类的数据结构。因此，数据结构课程主要是研究非数值计算的程序设计问题中所出现的计算机操作

对象以及它们之间的关系和操作的学科。

学习数据结构的目的是为了了解计算机处理对象的特性，将实际问题中所涉及的处理对象在计算机中表示出来并对它们进行处理，通过算法训练来提高学生的思维能力，通过程序设计的技能训练来促进学生的综合应用能力和专业素质的提高。

课程编号	课程名称	先修课程
C_1	计算机导论	无
C_2	数据结构	C_1，C_4
C_3	汇编语言	C_1
C_4	C 程序设计语言	C_1
C_5	计算机图形学	C_2，C_3，C_4
C_6	接口技术	C_3
C_7	数据库原理	C_2，C_9
C_8	编译原理	C_4
C_9	操作系统	C_2

(a)



(b)

图 1.1　教学计划编排问题的数据结构

(a) 教学计划；(b) 表示课程之间优先关系的有向图。

1.1.2　基本概念和术语

在系统地学习数据结构知识之前，先对一些基本概念和术语赋予确切的含义。

1．数据

数据(Data)是信息的载体，它能够被计算机识别、存储和加工处理。它是计算机程序加工的原料，应用程序处理各种各样的数据。计算机科学中，数据就是计算机加工处理的对象，它可以是数值数据，也可以是非数值数据。数值数据是一些整数、实数或复数，主要用于工程计算、科学计算和商务处理等；非数值数据包括字符、文字、图形、图像、语音等。

2．数据元素

数据元素(Data Element)是数据的基本单位。在不同的条件下，数据元素又可称为元

素、结点、顶点、记录等。例如，学生信息检索系统中学生信息表中的一个记录、教学计划编排问题中的一个顶点等，都称为一个数据元素。

有时，一个数据元素可由若干个数据项(Data Item)组成，例如，学籍管理系统中学生信息表的每一个数据元素就是一个学生记录。它包括学生的学号、姓名、性别、籍贯、出生年月、成绩等数据项。这些数据项可以分为两种：一种称为初等项，如学生的性别、籍贯等，这些数据项是在数据处理时不能再分割的最小单位；另一种称为组合项，如学生的成绩，它可以再划分为数学、物理、化学等更小的项。通常，在解决实际应用问题时是把每个学生记录当作一个基本单位进行访问和处理的。

3．数据结构

数据结构(Data Structure)是指互相之间存在着一种或多种关系的数据元素的集合。数据元素之间不是孤立的，在它们之间都存在着这样或那样的关系，这种数据元素之间的关系称为结构。根据数据元素间关系的不同特性，通常有下列四类基本的结构。

(1) 集合结构。在集合结构中，数据元素间的关系是“属于同一个集合”。集合是元素关系极为松散的一种结构。

(2) 线性结构。该结构的数据元素之间存在着一对一的关系。

(3) 树形结构。该结构的数据元素之间存在着一对多的关系。

(4) 图形结构。该结构的数据元素之间存在着多对多的关系，图形结构也称为网状结构。图 1.2 为表示上述四类基本结构的示意图。

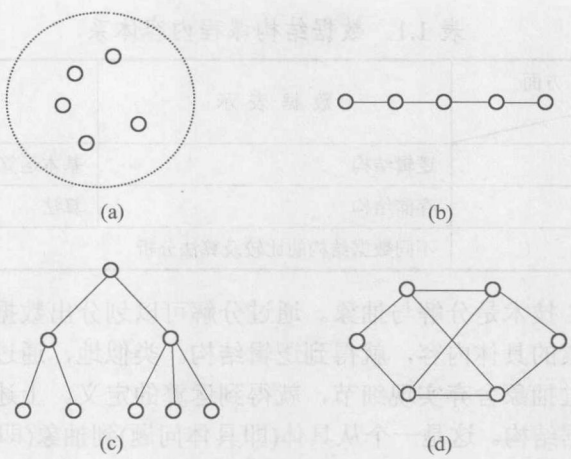


图 1.2　四类基本结构的示意图

(a) 集合结构；(b) 线性结构；(c) 树形结构；(d) 图形结构。

一个数据结构包含两个要素：一个是数据元素的集合；另一个是关系的集合。在形式上，数据结构通常可以采用一个二元组来表示。

数据结构的形式定义：数据结构是一个二元组，即

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

其中：D 是数据元素的有限集；R 是 D 上关系的有限集。

数据结构包括数据的逻辑结构和数据的物理结构。数据的逻辑结构可以看作是从

此为试读，需要完整PDF请访问：www.ertongbook.com

具体问题抽象出来的数学模型，它与数据的存储无关。研究数据结构的目的是为了在计算机中实现对它的操作，为此还需要研究如何在计算机中表示一个数据结构。数据结构在计算机中的标识(又称映像)称为数据的物理结构，或称存储结构。它所研究的是数据结构在计算机中的实现方法，包括数据结构中元素的表示及元素间关系的表示。

数据的存储结构常采用顺序存储或链式存储的方法。

顺序存储方法是把逻辑上相邻的元素存储在物理位置相邻的存储单元中，由此得到的存储表示称为顺序存储结构。顺序存储结构是一种最基本的存储表示方法，通常借助于程序设计语言中的数组来实现。

链式存储方法对逻辑上相邻的元素不要求其物理位置相邻，元素间的逻辑关系通过附设的指针字段来表示，由此得到的存储表示称为链式存储结构，链式存储结构通常借助于程序设计语言中的指针类型来实现。

除了通常采用的顺序存储方法和链式存储方法外，有时为了查找的方便还采用索引存储方法和散列存储方法。

1.1.3 数据结构课程内容体系

为了构造出好的数据结构及其实现，需考虑数据结构及其实现的评价与选择。因此，数据结构的内容包括三个层次的五个“要素”，见表 1.1。

表 1.1 数据结构课程内容体系

方面 / 层次	数据表示	数据处理
抽象	逻辑结构	基本运算
实现	存储结构	算法
评价	不同数据结构的比较及算法分析	

数据结构的核心技术是分解与抽象。通过分解可以划分出数据的三个层次；再通过抽象，舍弃数据元素的具体内容，就得到逻辑结构。类似地，通过分解将处理要求划分成各种功能，再通过抽象舍弃实现细节，就得到运算的定义。上述两个方面的结合使我们将问题变换为数据结构。这是一个从具体(即具体问题)到抽象(即数据结构)的过程。然后，通过增加对实现细节的考虑进一步得到存储结构和实现运算，从而完成设计任务。这是一个从抽象(即数据结构)到具体(即具体实现)的过程。熟练地掌握这两个过程是数据结构课程在专业技能培养方面的基本目标。

1.2 算法和算法分析

算法与数据结构的关系紧密，在算法设计时先要确定相应的数据结构，而在讨论某一种数据结构时也必然会涉及相应的算法。下面就从算法特性、算法描述、算法性能分析等三个方面对算法进行介绍。

1.2.1 算法特性

算法(Algorithm)是对特定问题求解步骤的一种描述，是指令的有限序列。其中每一条指令表示一个或多个操作。一个算法应该具有下列特性。

(1) 有穷性。一个算法必须在有穷步之后结束，即必须在有限时间内完成。

(2) 确定性。算法的每一步必须有确切的定义，无二义性。算法的执行对应着的相同的输入仅有唯一的一条路径。

(3) 可行性。算法中的每一步都可以通过已经实现的基本运算的有限次执行得以实现。

(4) 输入。一个算法具有零个或多个输入，这些输入取自特定的数据对象集合。

(5) 输出。一个算法具有一个或多个输出，这些输出同输入之间存在某种特定的关系。

算法的含义与程序十分相似，但又有区别。一个程序不一定满足有穷性。例如，操作系统，只要整个系统不遭破坏，它将永远不会停止，即使没有作业需要处理，它仍处于动态等待中。因此，操作系统不是一个算法。另外，程序中的指令必须是机器可执行的，而算法中的指令则无此限制。算法代表了对问题的解，而程序则是算法在计算机上的特定的实现。一个算法若用程序设计语言来描述，则它就是一个程序。

算法与数据结构是相辅相承的。解决某一特定类型问题的算法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，一种数据结构可以由各种算法的执行来体现。

要设计一个好的算法通常要考虑以下要求。

(1) 正确。算法的执行结果应当满足预先规定的功能和性能要求。

(2) 可读。一个算法应当思路清晰、层次分明、简单明了、易读易懂。

(3) 健壮。当输入不合法数据时，应能作适当处理，不至引起严重后果。

(4) 高效。有效使用存储空间和有较高的时间效率。

1.2.2 算法描述

算法可以使用各种不同的方法来描述。最简单的方法是使用自然语言。用自然语言来描述算法的优点是简单，便于人们对算法的阅读；缺点是不够严谨。也可以使用程序流程图、N-S 图等算法描述工具描述算法。其特点是描述过程简洁、明了。还可以直接使用程序设计语言来描述算法，直接使用程序设计语言并不容易，且不太直观，常常需要借助于注释才能使人看明白。

为了解决理解与执行两者之间的矛盾，人们常常使用伪码语言来描述算法。伪码语言介于高级程序设计语言和自然语言之间，它忽略高级程序设计语言中一些严格的语法规则与描述细节，比程序设计语言更容易被人理解和描述，而比自然语言更接近程序设计语言。

1.2.3 算法性能分析

当将一个算法转换成程序并在计算机上执行时，其运行所需要的时间取决于下列因素。

(1) 硬件的速度。

(2) 书写程序的语言。实现语言的级别越高，其执行效率就越低。

(3) 编译程序所生成目标代码的质量。代码优化较好的编译程序所生成的程序质量较高。

(4) 问题的规模。例如，求 10 以内的素数与求 1000 以内的素数其执行时间必然是不同的。

显然，在各种因素都不能确定的情况下，很难比较出算法的执行时间。也就是说，使用执行算法的绝对时间来衡量算法的效率是不合适的。为此，可将上述各种与计算机相关的软、硬件因素都确定下来，这样一个特定算法的运行工作量的大小就只依赖于问题的规模。

可以用时间复杂度与空间复杂度来评价算法的优劣。

1．时间复杂度

时间复杂度是指算法转化成的程序从开始运行到结束所需要的时间。一个算法是由控制结构和原操作构成的，其执行时间取决于两者的综合效果。

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来说是基本运算的原操作，以该原操作重复执行的次数作为算法的时间度量。一般情况下，算法中原操作重复执行的次数是规模 n 的某个函数 T(n)。

许多时候要精确地计算 T(n)是困难的，引入渐近时间复杂度在数量上估计一个算法的执行时间，也能够达到分析算法的目的。

定义：如果存在两个正常数 a 和 n_0，使得对所有的 n，$n \geqslant n_0$，有

$$f(n) \leqslant ac(n)$$

则有

$$f(n) = O(c(n))$$

例如，一个程序的实际执行时间为 $T(n)=2.7n^3+3.8n^2+5.3$，则 $T(n)=O(n^3)$。

使用上述表示的算法的时间复杂度，称为**算法的渐近时间复杂度**。常见的渐近时间复杂度有 $O(1)<O(\log_2 n)<O(n)<O(n\log_2 n)<O(n^2)<O(n^3)<O(2^n)$，通常用 O(1)表示常数计算时间。

2．空间复杂度

空间复杂度是指算法转化成的程序从开始运行到结束所需的存储量。程序的一次运行是针对所求解问题的某一特定实例而言的。例如，排序程序的每次执行是对一组特定个数的元素进行排序。对该组元素的排序是排序问题的一个实例。元素个数可视为该实例的特征。

程序运行所需的存储空间包括以下两部分。

(1) 固定部分。这部分空间与所处理数据的大小和个数无关，或者称与问题的实例的特征无关。主要包括程序代码、常量、简单变量、定长成分的结构变量所占的空间。

(2) 可变部分。这部分空间大小与算法在某次执行中处理的特定数据的大小和规模有关。例如，10 个数据元素与 1000 个数据元素的排序算法所需的存储空间显然是不同的。

第2章　线性表

线性表是一种常见的数据结构。它有两种存储方式：顺序存储和链式存储。对其进行的主要操作有插入、删除、查找和排序等。

2.1　线性表的逻辑结构

2.1.1　线性表的定义

线性表是一种线性结构的数据结构，数据元素之间是一种线性关系，元素是顺序排列的。在线性表中，数据元素的类型是相同的。在实际生活中线性表的例子很多，如学生基本信息表、学生成绩表等。

线性表定义如下：

线性表是由n(n>=0)个具有相同数据类型的数据元素构成的有限序列，常记为

$$A=(a_1,\ a_2,\ \cdots,\ a_{i-1},\ a_i,\ a_{i+1},\ \cdots,\ a_n)$$

其中：A为线性表名；n为线性表的表长，当n＝0 时，称此线性表为空表。表中相邻元素之间存在着顺序关系，在线性表中有且仅有一个开始元素与一个终端元素。

2.1.2　线性表的基本操作

在第1章中提到，数据结构的运算是定义在逻辑结构层次上的，而运算的具体实现是建立在存储结构上的。换句话说，某数据结构的一种运算，针对此数据结构采用不同的存储结构，这个运算的实现的具体方法是不同的。

因此，下面的线性表的基本运算是定义在逻辑结构上的，每个运算的具体实现步骤(方法)只有在确定了线性表的存储结构之后才能完成。

线性表上的基本操作如下：

(1) 构建线性表：Create_List(L)。将线性表中的元素，采用一定的存储结构存放到计算机中，利用计算机语言来实现对线性表的操作。

(2) 输出线性表中元素操作：Out_List(L)。输出线性表中所有的数据元素。

(3) 求线性表的长度：Length_List(L)。求线性表中数据元素的个数。

(4) 简单查找：Check_List(L，x)，x是给定的一个数据元素。在表L中查找值为 x 的数据元素，其结果返回在L中首次出现的值为x的那个元素的序号或地址，称为查找成功;否则，在L中未找到值为 x 的数据元素，则返回一特殊值表示查找失败。

(5) 复杂查找操作：Checkall_List(L，x)，x是给定的一个数据元素。在表L中查找值为 x 的数据元素，其结果返回在L中所有的值为x的元素的序号或地址。

(6) 插入操作：Insert_List(L，i，x)。在线性表L的第 i 个位置上插入一个值为 x 的新元素，插入后表长为原表长加1。

(7) 简单删除操作：Delete_List(L，i)。在线性表L中删除第i个数据元素，删除后新表长为原表长减 1 。

(8) 复杂删除操作：Delete_List(L，i，k)。在线性表L中，从第i个数据元素开始，连续删除k个数据元素，删除后修改新表长。

说明：

(1) 某数据结构上的基本运算，不是它的全部运算，而是一些常用的基本运算，读者掌握了某一数据结构上的基本运算后，其他的运算可以通过基本运算来实现，也可以直接去实现。

(2) 在上面各操作中定义的线性表 L 仅仅是一个抽象在逻辑结构层次的线性表，尚未涉及它的存储结构，因此每个操作在逻辑结构层次上尚不能用具体某种程序语言写出具体算法，而算法的实现只有在存储结构确立之后。

2.2 线性表的顺序存储及运算实现

2.2.1 顺序表

线性表的顺序存储是指在计算机的内存中用一块连续的地址作为存储空间顺序存放线性表的各元素，这种存储形式的线性表称为顺序表。计算机的内存中的地址空间是连续的，因此，用物理上相邻的存储单元来实现逻辑上相邻的数据元素的存放是简单、可行的。

如图2.1所示，设 a_1 的存储地址为Loc(a_1)，每个数据元素占d个存储地址，则第i个数据元素的地址为

$$Loc(a_i)=Loc(a_1)+(i-1)*d \quad (1<=i<=n)$$

只要知道顺序表首地址和每个数据元素所占地址单元的个数就可求出第i个数据元素的地址，这是顺序表“按序号随机存取”特点的体现。

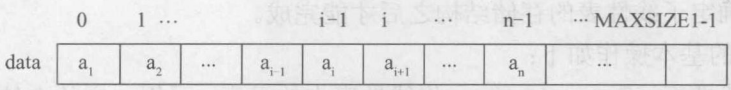


图 2.1 顺序存储示意图

在高级程序设计语言中，一维数组在内存中占用的存储空间是一组连续的存储区域，因而，用一维数组来实现线性表的数据元素的存放是再合适不过的。设用data[MAXSIZE]来表示存放线性表的区域，其中MAXSIZE是一个根据实际问题定义的足够大的整数，线性表中的数据元素从 data[0]开始依次存放，为了记录当前线性表中的实际元素个数，引入了一个变量 length 来存放线性表中数据元素的个数(表长)。因此，当表为空时length为0。

通常将data数组和length封装成一个结构体作为顺序表的类型：

```
typedef   struct
{  datatype  data[MAXSIZE];
   int     length;
    } SeqList;
```

定义一个顺序表的语句：SeqList　L；

这种线性表如图2.2所示。表长为L. length，线性表中的数据元素a_1～a_n分别存放在L.data[0] ～L.data[L. length-1]中。

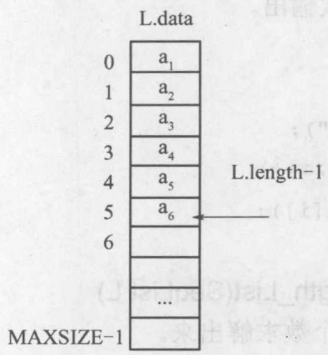


图 2.2　线性表的顺序存储示意图

2.2.2　顺序表上基本运算的实现

1. 顺序表的构建：Create_List ( )

顺序表的构建就是将线性表中的元素依次存放到data[MAXSIZE]数组中，并将数据元素的个数赋给length。算法如下(为了简单起见，假设数据元素的类型是int类型)：

```
#define MAXSIZE 100
/*定义顺序存储类型 */
typedef  struct
{int data[MAXSIZE];              /*存放线性表的数组*/
 int length;                     /* length是线性表的长度*/
}SeqList;
/* 构建顺序表 */
SeqList Create_List ( )
{SeqList L;
int i, n, x;
scanf("%d", &n);
for(i=1;i<=n;i++)
scanf("%d", & L.data[i-1]);
 L.length=n;
 return (L); }
```

设调用函数为主函数，主函数对构建函数的调用如下：

此为试读，需要完整PDF请访问：www.ertongbook.com