全国高等职业教育教材丛书

数据结构导论

陈有祺　辛运帏　　编著

南开大学出版社
天　津

图书在版编目(CIP)数据

数据结构导论 / 陈有祺，辛运帏编著. —天津：南开大学出版社，2001.11
（全国高等职业教育教材丛书）
ISBN 7-310-01579-7

Ⅰ. 数… Ⅱ. ①陈…②辛… Ⅲ. 数据结构—高等教育：职业教育—教材 Ⅳ. TP311.12

中国版本图书馆 CIP 数据核字(2001)第 041344 号

出版发行 南开大学出版社
地址：天津市南开区卫津路 94 号
邮编：300071 电话：(022)23508542
出版人 肖占鹏
承 印 天津宝坻第二印刷厂印刷
经 销 全国各地新华书店
版 次 2001 年 11 月第 1 版
印 次 2001 年 11 月第 1 次印刷
开 本 787mm×1092mm 1/16
印 张 12.75
字 数 315 千字
印 数 1 — 5000
定 价 18.00 元

高等职业教育教材编审委员会名单

主任委员：

乔丽娟

委　　员：（以姓氏笔画为序）

丁桂芝　王松岭　边奠英　刘凤桐

李占伦　李维祥　吴功宜　赵雅兴

徐宝强　徐娟敏　葛洪贵

序

全国高等教育自学考试指导委员会副主任

中国职业技术教育学会副会长　　王明达

中国高等教育大众化目标的实现必然伴随着高等教育形式和结构多样化的变革。单纯以学术水平为追求目标的高等教育无法满足社会对于多种专门人才的需求，因此要大力发展高等职业教育，培养社会需要的各类专门人才，以适应我国经济和社会发展的要求。

什么是高等职业教育?职业教育的特征不在于办学形式，主要体现在培养目标上。培养生产、服务、管理第一线的实用型人才的教育即为职业教育。按照专业所需接受教育的年限达到相当于普通高等教育学习年限的职业教育即为高等职业教育。

高等职业教育如何实现培养实用人才的目标?首要的就是专业设置。既然培养的是生产第一线的实用型人才，所设专业就一定是直接与社会生产、生活相联系的，社会生产、生活中最必需的。这与普通高等学校开设专业的思路有着本质的区别。其次是教学内容的安排和教学计划的制定。接受高等职业教育的学生其学习内容必须是成熟的技术和管理规范，教学计划、课程设置应该按照职业岗位群的职业能力要求来确定，而不应从学科体系出发。再次，为使学生毕业就能基本顶岗工作，要求增大实习训练所占的比例，在校期间就基本完成上岗前的实践训练。为了保证实践训练得到社会认可，要实行学历证书与职业资格证书“双证书”制度，同时要求双师型教师任教。只有按部就班实现以上要求的高等职业教育才会被社会认同，也才会有生命力。

办出特色是高等职业教育生命力的源泉。学生毕业即能顶岗是职业教育区别于其他教育的一个突出特点。要想做到这一点，一方面学习理论知识要以“必需”和“够用”为度，让学生掌握基本理论和知识；另一方面要全方位开辟实习基地，保证充足的实训时间。高等职业教育的水准主要是通过专业设置、课程内容，以及实训能力的培养体现的。

为落实第三次全教会“完善自学考试制度”、“大力发展高等职业教育”的改革思路，1999年全国高等教育自学考试指导委员会决定在天津市开展高等教育自学考试职业技术专业的试验工作。

天津市高等教育自学考试委员会在深入调查研究的基础上，从职业岗位群的知识技能需求出发，以能力本位教育(CBE)为理论依托，设计了 12 个职业技术专业，于 2000 年面向社会开考。

高等教育自学考试开考职业技术专业的试验，在完善高等教育自学考试专业建设、拓展自学考试教育功能方面，在探索开放式教育、培养应用型高级人才方面，在职业教育课程体系方面，以及在实践技能考核的研究、管理方面，对于我国高等教育自学考试制度的完善和高等职业教育的发展都具有重要意义。

天津市高等教育自学考试委员会将根据职业技术专业试验工作的需要陆续出版有关考试

课程的教材。教材编撰者多为具有职业教育经验的学科专家和职业教育专家，他们根据职业教育的专业培养目标重新整合了学科知识体系，尽力体现理论知识必需、够用的原则。当然，由于认识水平的局限和时间的紧迫，这些教材还需要继续完善提高。尽管如此，这迈出的第一步是十分可贵的。我深信，高等教育自学考试职业技术专业的试验工作一定能取得成功。

2001 年 1 月于北京

前　言

21 世纪计算机科学技术的发展将更加迅速,计算机的应用将更为广泛，社会对各类计算机人才的需求更是不断增长。《数据结构导论》作为高等教育自学考试计算机技术与应用专业重要的基础课教材，今天和广大读者见面了。我们希望这本教材在培养计算机人才方面能起到它应有的作用。数据结构导论这门课程在计算机技术与应用专业的教学中起着“承前启后”的作用。学生们在初步掌握了计算机基础知识和一种程序设计语言之后（主要是 C 语言），学习了本课程就可以明显地提高编程水平和解决实际问题的能力。另一方面，这门课程又是学习后继课程的基础，如数据库、操作系统等课程都需要本课程学到的有关概念和方法。

本教材共分 7 章。第 1 章主要介绍数据结构的基本概念和算法的思想。第 2 章引入最基本的数据结构——线性表，给出了线性表的定义、存储结构和基本运算，还介绍了特殊的线性表——串的有关运算。第 3 章介绍几种常用的线性表——栈、队列和数组，分别给出了它们的定义、基本运算和应用实例。第 4 章引入树型结构的概念，它是一种应用十分广泛的非线性结构，其中重点介绍了二叉树和一些应用实例。第 5 章引入了比树更复杂的结构——图结构，给出了它的各种存储结构和基本运算，详细讨论各种特殊图的典型应用。第 6 章讨论了查找问题，在各种结构上给出了查找方法，分析了查找的效率。第 7 章讨论了排序的概念和各种方法，并对它们的性能进行了分析比较。

本书是针对高等教育自学考试人员编写的，因此在取材上力求少而精，只选择最基本、最实用的内容来讲述，对于抽象的概念和复杂的公式推导都尽量避免。在叙述上力求深入浅出，通过大量例子和图表来讲明有关内容，对于重要算法都用 C 语言详细描述，并加上许多注解。书中各章最后都附有习题，这些习题从最基本的到较难的都有，基本上反映了本课程的要求，希望读者认真练习，有条件的读者最好能做一定数量的上机实习。

虽然编者希望把本书写好，但由于水平和经验所限，书中难免有错误和不妥之处，恳请同行专家和广大读者批评指正。

编　者

2001 年 5 月

通讯地址：300071　天津南开大学信息技术科学学院

E-mail 地址：陈有祺　chenyq@nankai.edu.cn

辛运帏　xinyw@robot.nankai.edu.cn

目　录

第 1 章 绪论

内容提要和学习指导

数据结构（data structure）是一门随着计算机科学的发展而逐渐形成的学科，目前已成为计算机类各专业的基础课之一。本章的目的就是对数据结构的基本概念和基本内容作一概要介绍，为以后各章的学习打下基础。具体来说，包括数据结构的发展历史，数据结构的基本概念和基本术语，关于算法的概念和算法的描述，以及计算复杂性的度量等方面的问题。

通过本章的学习，要求读者准确理解数据结构和算法的基本概念，初步掌握计算复杂性的概念和分析方法。

1.1 数据结构的发展历史

为了使读者能更清楚地理解数据结构究竟是什么，现在讲一讲数据结构的发展历史。

大家知道，在 20 世纪 40 年代电子数字计算机刚诞生时，它所能处理的数据是非常简单的，只能直接对二进制数进行计算，可以说谈不上什么结构。后来虽然可以处理十进制整数和十进制实数，也不过是从 0 到 9 这十个数字组成的序列，或者再加上一个小数点，其结构也是非常简单的，没有专门研究它的必要。

到 20 世纪 50 年代以后，各种高级程序语言纷纷出现，所能描述的数据类型也逐渐增多。例如，在 FORTRAN 语言中允许使用复数类型的量，实际上，是用两个有序的实数（X,Y）来表示一个复数 X+iY，其中第一个实数表示复数的实部，第二个实数表示复数的虚部，这就有点“结构”的雏形了。另外，在 BASIC、FORTRAN 等语言中，都允许使用数组，其中最简单的是一维数组，就是同类型元素的一个有限序列，这当然是一种很典型的结构。当计算机的应用领域由数值计算扩充到非数值计算之后，各种新的结构应运而生，其中最重要的是在 COBOL、PL/1 等语言中出现的记录类型，它是把许多不同类型的数据组合起来构成一个新的类型。例如，一个职工的基本情况可以包括姓名、性别、出生年月、工龄、工资等项内容，这些项目合起来就构成一个职工的记录。如表 1.1 中的每一行就是一个记录，所有各行合起来就构成一维数组。

表 1.1 数据结构示例

张卫民	男	1960.04	21 年	1078.35
林 竹	女	1978.05	5 年	734.50
李 伟	男	1952.11	25 年	1235.60
刘文成	男	1970.07	10 年	945.20
王小燕	女	1965.12	16 年	1056.45

另外，大量的字符处理需要有字符串（简称为串）类型的量，SNOBOL 语言就是对于串运算最方便的高级程序语言。后来，LISP 语言又定义一种带有层次性的表结构以及相应

的运算。这种表已经是一种相当复杂的非线性结构，实际上是一种树型结构，这是我们在本书中将要着重讨论的内容。

20 世纪 60 年代中期以后，在数据结构的发展史上发生了几个重要的事件。首先，在美国计算机界出现了信息结构这一名称（后来改为数据结构）。然后在 1968 年，由美国计算机协会（ACM）颁发了建议性的计算机教学计划，其中规定数据结构作为一门独立的课程。

同年，世界著名的计算机科学家 D.E.Knuth 教授的巨著《计算机程序设计的技巧》第一卷《基本算法》出版，该书全面系统地论述了数据的逻辑结构和存储结构，并且给出了典型的各种算法，为数据结构奠定了理论基础。从 20 世纪 60 年代末到 70 年代初，出现了大型的程序和大规模的文件系统，结构程序设计成为程序设计方法学的主要内容，人们对数据结构越来越重视，认为程序设计的实质就是对要处理的问题选择一种好的数据结构，并在此结构上施加一种好的算法。著名的计算机科学家 N.Wirth 写的《算法+数据结构=程序》一书正是体现了这种观点。本书也正是紧紧围绕数据结构和算法这两个问题进行讨论的。

1.2 数据结构的基本概念和术语

数据 是指能输入到计算机中并能被计算机处理的一切对象。这里必须强调指出，所谓数据绝不能仅仅理解为整数或实数这种狭义的“数”，必须作广义的理解。例如，一个用某种程序语言编写的源程序、一篇文章、一张地图、一幅照片、一首歌曲等等，都可以视为“数据”。今后随着计算机的发展，还将不断扩大数据的范围。

数据元素 是数据的基本单位。由于数据的范围非常广泛，因此基本单位也是可大可小的。小到可以是一个字符，大到可以是一个国家的地图或一本书等等。对于较大的单位，还可以由称为“数据项”的较小单位组成。如一本书的目录卡片就可以包括书名、作者名、出版社名和出版日期等数据项。

数据对象 是具有相同性质的数据元素的集合，是数据的一个子集。因为计算机不可能同时处理一切类型的数据，总是对特定的问题处理一种或几种对象。例如用计算机求素数这种问题只涉及“整数”这种数据对象。

数据结构 是彼此具有一定关系的数据元素的集合。事实上，这些关系反映了客观世界事物之间的联系。由于客观事物存在着各种不同的联系形式，所以反映在数据关系上也就各不相同。一般来说，有四类基本结构：（1）集合 这个结构中的数据元素之间除了“同属于一个集合”这一关系外，没有其他关系。（2）线性结构 这个结构中数据元素存在着依次排列的先后次序关系。（3）树型结构 这个结构中数据元素之间存在着层次关系或分支关系。（4）图结构 这个结构中数据元素之间相互连接成网状。图 1.1 形象地表示出这四类基本结构中数据元素之间的关系。

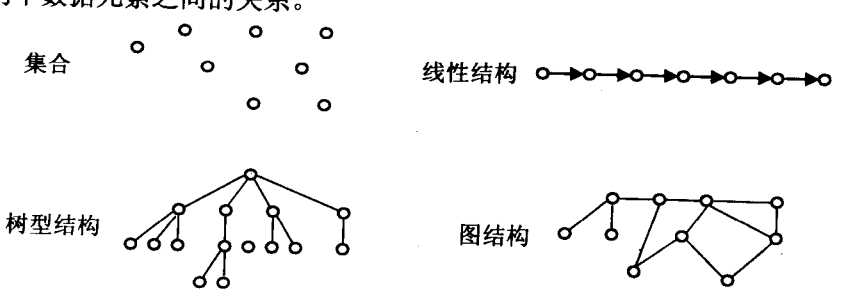

图 1.1 四类基本结构示意图

从另一个角度讲，数据结构又分为逻辑结构和物理结构两个方面。逻辑结构是指数据元素之间的关联方式，如上面所举出的各种基本结构就是逻辑结构。物理结构又称存储结构，是指在计算机中如何具体存储数据的各种逻辑结构。同一种逻辑结构可以由几种不同的物理结构来实现。在某种意义上，研究物理结构更为重要。

在引入上述基本概念之后，我们可以对数据结构这一学科的研究范围简单地概括如下：数据结构研究数据的逻辑结构和物理结构，并在这种结构上定义相关的运算，设计并实现相应的算法，分析算法的效率，以达到提高程序质量的目的。

1.3 运算、算法和算法分析

1.3.1 运算

从上一节关于数据结构的描述可知，运算和结构是紧密地联系在一起的。当然，在不同的结构上可以进行的运算是不同的。例如，在复数上只能定义加、减、乘、除和取模等运算，而在图或树型结构上可以进行的运算则复杂得多。一般来说，根据操作的效果，可以将运算分为以下两种基本类型：

（1）加工型运算　其操作改变了原结构中数据元素的个数或数据元素的内容。

（2）引用型运算　其操作不改变结构中数据元素的个数和元素的内容，只从结构中提取某些信息作为运算的结果。

在各种不同的结构上，通常的基本运算主要有：

①插入　加工型运算，在原结构中指定位置增添新的数据元素。

②删除　加工型运算，从原结构中删掉某个指定的数据元素。

③查找　引用型运算，从结构中找出满足某些条件的数据元素的位置。

④读取　引用型运算，从结构中找出满足某些条件的数据元素的内容。

⑤更新　加工型运算，改变结构中某个数据元素的内容。

通过这些基本运算，可以构成各种复杂的运算。实现任何运算，都要给出算法。有关算法的问题，我们在下两节讨论。

1.3.2 算法及其描述

算法是计算机科学的一个基本概念，也是程序设计的一个核心概念。为了弄清楚这个重要概念，我们从定义和实例两方面来讲述。

算法　是对某个问题求解步骤的一种描述，是“指令”的有限序列。所谓“指令”，不见得是一条机器指令，而是意义明确的一次操作。例如，“将两个数对调位置”，“求 x 的平方根”等等，都可以认为是一条指令。此外，一个算法还要具有以下五个特性：

(1) 有穷性　一个算法必须总是对合法的输入值执行有穷步之后结束，而每一步都必须在有穷时间内完成。

(2) 确定性　算法中每条“指令”的含义都必须非常确定，读者在理解时不会有二义性。另外，对相同的输入在任何时候执行都应得出相同的结果。

(3) 可行性　一个算法应是可行的，即算法中每一操作都能通过已知的一组基本操作来实现。

(4) 输入　一个算法可以有零个或多个输入，这些输入取自特定的数据对象的集合。

(5) 输出　一个算法有一个或多个输出，这些输出是算法对输入进行运算的结果。

【例 1.1】　问题：从 *a,b,c,d* 四个整数中选出最大的一个，将它平方后输出。

算法：　(1) 先从 *a,b* 两个数中选出最大的一个，设为 *x*；

(2) 再从 *c,d* 两个数中选出最大的一个，设为 *y*；

(3) 从 *x,y* 两个数中选出最大的一个，设为 *z*；

(4) 对 *z* 求平方，然后输出。

可以看出，以上算法是用自然语言描述的，因为问题很简单，所以每一个步骤都是明确的，也很容易实现。为了更加准确和便于在计算机上实现，在本书中绝大多数算法都用 C 语言描述，但也有一些算法用自然语言描述。在例 1.1 中的算法用 C 语言描述就是（略去某些变量的说明）：

```
void   max(int a,b,c,d)
{
    if (a>b)
        x=a;
    else x=b;              // 将 a,b 两数中较大的一个赋给 x
    if (c>d)
        y=c;
    else y=d;              // 将 c,d 两数中较大的一个赋给 y
    if (x>y)
        z=x;
    else z=y;              // 将 x,y 两数中较大的一个赋给 z，此时 z 为 a,b,c,d
                           // 四个数中最大的一个数
    w=z*z;
    printf("%d\n",w);      // 输出结果
}
```

1.3.3 算法分析和计算复杂性

一般来说，为了解决一个问题，可以有不同的算法。那么，如何评价这些算法，在解决同一个问题时，究竟选择什么样的算法，这就是算法分析所要讨论的内容。在本书中，不能对算法分析进行深入的讨论，但是也要涉及一些基本的概念。

首先，如何评价一个算法的优劣，通常遵循以下几个评价标准：

(1) 正确性　这是一个起码的标准，一个不能保证正确性的算法，根本不能称为算法。

(2) 易读性　算法应易于阅读和理解，以便于调试和修改。

(3) 健壮性　当输入的数据非法时，算法也能做出适当的反应或进行特殊处理，不至于得出莫名其妙的结果或死机。

(4) 高效率　算法的效率包括时间和空间两个方面，对于解决一个问题，理想的算法

是计算次数尽量少并且存储量尽量小。

在以上几个标准中，除正确性之外，算法的效率是最重要的标准。下面对这个问题再进一步加以讨论。

算法的效率也称为算法的复杂度（或计算复杂度），一个算法所需计算次数的多少称为时间复杂度，一个算法所需的辅助存储空间的大小称为空间复杂度。这两种度量都不能表示为绝对的量，而是表示为该算法所解决的问题中数据个数的一个函数关系。例如，我们设计一个对 n 个数排序的算法，这个算法当 n=10 和 n=10 000 时所用的时间是明显不同的，因此只能用一个 n 的函数 $f(n)$来刻画该算法的时间复杂度，$f(n)$的不同形式就能区别算法时间复杂度的高低。对于空间复杂度也用同样的表示方法。

为了说明如何对一个算法的复杂度进行分析，下面举一个大家熟悉的例子。

【例 1.2】 求两个 n 阶矩阵的乘积。用 C 语言写出算法如下：

```
(1)  for(i=0;i<n;i++)
(2)      for(j=0;j<n;j++)
(3)      {
(4)          C[i][j]=0;
(5)          for(k=0;k<n;k++)
(6)              C[i][j]=C[i][j]+A[i][k]*B[k][j];
(7)      }
```

让我们对上述算法的时间复杂度作一分析。根据 C 语言中 for 语句的语义，算法的第(1)行需执行 n 遍（每遍中具体的操作次数可从略），而第(2)行的 for 语句是包含在第(1)行的 for 语句之中的，因此它需要执行 n^2 遍；同样，第(4)行的赋值语句也执行 n^2 遍。可是第(5)行的 for 语句需要执行 n^3 遍，第(6)行的赋值语句也执行 n^3 遍。将各行的执行遍数加起来即为整个算法的时间复杂度

$$T(n)=2n^3+2n^2+n$$

它是 n 的一个三阶多项式，忽略低次项后，可写为

$$T(n)=O(n^3)$$

这里的 $O(n^3)$表示它是与 n^3 同阶的一个量，也可以认为它是 n^3 的某个常数倍。以后我们描述复杂度都用这种记法，如 $O(1)$、$O(n)$、$O(n^2)$分别表示常量阶、线性阶和平方阶。其他可能出现的复杂度还有对数阶 $O(\log n)$、指数阶 $O(2^n)$等等。不同阶的复杂度随着 n 的增长率是很不相同的，它们的变化如表 1.2 所示。

表 1.2　常见函数值的增长比较

n	$\log n$	n^2	n^3	2^n
10	1	100	1 000	1 024
20	1.3	400	8 000	1 048 576
50	1.7	2 500	125 000	$>10^{15}$
100	2	10 000	1 000 000	$>10^{30}$

从表中可以看出，对数阶的函数增长最慢，其次为多项式阶（n 的某个整数幂，当然幂次越低越好），而指数阶是增长最快的。事实上，一个指数函数的增长率达到惊人的程度。例如，对于 n=100，这个数在实际问题中不能算是太大的，可是 2 的 100 次幂竟然超过 10^{30}。这是一个什么概念呢？如果有一个算法需要 10^{30} 次运算，即使在每秒钟能执行一亿次运算

的计算机上运行，也要100万亿年才能完成！这在实际上当然是不可能实现的，因此，我们要避免使用那些计算复杂度达到指数阶的算法。

本章小结

本章主要内容包括以下几个方面：（1）介绍了贯穿全书的基本概念。例如数据、数据元素、数据项、数据对象、运算、算法等等，都必须准确地掌握。（2）给出了数据结构的定义和它的两个重要的方面——逻辑结构和物理结构，要求初步领会这些概念和四种基本数据结构的特点，并通过以后各章的学习逐步深入理解。（3）关于算法描述和计算复杂度。要求能用C语言描述简单的算法，建立计算复杂度的概念，理解选择复杂度低的算法的必要性。

至于数据结构的发展历史，主要是为了对数据结构从纵的方向加深了解，认识到它的重要性，对其中的年代和有关事件，不要求死记硬背。

练习题

一、选择题（将正确答案的号码填在横线上）

1.数据对象是具有相同性质的_____的集合。

① 数据结构　② 数据

③ 数据元素　④ 数据值

2. 算法的有穷性是指_____。

① 所需要的存储空间是有限的　② 所需要的执行时间是有限的

③ 输出的数据是有限的　④ 描述算法的语句个数是有限的

二、填空题（将正确的答案填在线的上方）

1. 数据的基本结构有_____，_____，_____ 和_____四类。

2. 在数据结构上的运算有_____ 和_____两种基本类型。

三、问答题

1. 数据结构的研究范围是什么？

2. 如何评价一个算法的优劣？其中哪些指标是最重要的？

3. 什么是数据的逻辑结构和存储结构？它们之间有什么关系？

4. 试设定若干个 n 值，比较两个函数 n^2 和 $60n*\log n$ 的增长趋势，并确定 n 大于何值时，函数 n^2 的值大于 $60n*\log n$ 的值。

四、实习题

1. 用C语言编写一个计算一元多项式

$$P_n(x)=a_0+a_1x+a_2x^2+\cdots+a_nx^n$$

的值 $P_n(x_0)$的程序。该问题可以采用不同的算法，试分析不同算法的时间复杂度（写为 n 的函数）。

2. 用 C 语言写出求数组 $A[n]$中元素的最大值和次最大值的程序，并分析在最坏情况下的时间复杂度。

3. 已知 k 阶斐波那契序列的定义为：

$f_0=0, f_1=0, \cdots, f_{k-2}=0, f_{k-1}=1;$

$f_n=f_{n-1}+f_{n-2}+\cdots+f_{n-k} \qquad (n=k,k+1,\cdots)$

试编写求 k 阶斐波那契序列第 m 项的程序（写为函数形式，其中 k 和 m 为参数），并分析算法的时间复杂度和空间复杂度。

第 2 章 线性表

内容提要和学习指导

线性表（linear list）是一种最基本、最常用的数据结构。本章将给出线性表的定义和在线性表上的基本运算，详细介绍线性表的两种基本存储结构，以及基本运算在这两种存储结构上的实现。为了使读者更好地掌握线性表的应用，本章还给出一个数据存储和运算的综合实例。另外，本章还简要介绍线性表的一种特殊形式——串（string），讨论了串的存储方式和它的简单运算。在学习本章的过程中，读者要认真领会线性表的基本概念，熟练掌握线性表的基本存储结构和基本运算在各种存储结构上的实现算法。同时，也要掌握串的存储方式和几种简单运算。

2.1 线性表的定义和基本运算

为了对线性表有一个具体印象，我们先举几个例子。

【例 2.1】（4，8，6，9，12，3）是一个线性表，其中数据元素是整数，按表中列出的顺序排列，共有 6 个数据元素。

【例 2.2】（A, B, C, D,…,X, Y, Z）是一个线性表，其中的数据元素是英文大写字母，按字母表的顺序排列，共有 26 个数据元素。

【例 2.3】 第 1 章中的表 1.1 也是一个线性表，其中数据元素是记录（表中的每一行），按行的顺序排列，共有 5 个数据元素。

综合上述例子，我们给出线性表的定义：

一个线性表是 n（$n\geq 0$）个同类型数据元素 $a_1,a_2,\cdots,a_n$ 的有限序列，记作

$(a_1,a_2,\cdots,a_n)$

从定义可知，线性表强调两个特点。一个是强调每个数据元素 $a_i(i=1,\cdots,n)$必须是同类型的数据，不能将不同性质的数据并列在一个表内。例如，如果是整数，则一切元素必须都是整数；如果是单个字符，则必须都是字符（不要求都是字母）；如果是记录，则必须都是含有相同数据项的记录。另一个是强调数据元素的有序性，数据元素在表中的位置决定了它的序号。按照这个排定的次序，元素 a_{i-1} 称为 a_i 的直接前趋；反之，a_i 称为 a_{i-1} 的直接后继 $(i=2,\cdots,n)$。因为 a_1 是表中第一个元素，所以它没有直接前趋；a_n 是表中最后一个元素，所以它没有直接后继。

表中数据元素的个数 n 称为线性表的长度。长度为 0 的表称为空表。

对于线性表，常用的运算有以下几种：

（1）初始化　INITIATE (L)，加工型运算。其结果是建立一个空表 L。

（2）求表长　LENGTH(L)，引用型运算。其结果是表 L 的长度，即表 L 中元素的个数。

（3）读表元　GET(L,i)，引用型运算。当 $1\leq i\leq$LENGTH(L)时，其结果是表 L 中第 i

个元素的值。

（4）按值查找　LOCATE(L,x)，引用型运算。若表 L 中存在一个或多个值为 x 的元素，则结果为这些元素在表中序号的最小值；否则，结果为 0。

（5）插入　INSERT(L,x,i)，加工型运算。其作用是在表 L 的第 $i(1\leqslant i\leqslant LENGTH(L)+1)$ 个位置插入新的元素 x，插入后表 L 的长度增加 1。

（6）删除　DELETE(L,i)，加工型运算。其作用是从表 L 中去掉第 $i(1\leqslant i\leqslant LENGTH(L))$ 个元素，删除后表 L 的长度减少 1。

（7）清除表　CLEAR(L)，加工型运算。其作用是将表 L 中全部元素删除，使 L 变为一个空表。

对线性表的其他运算还有将两个表合并，将表中元素按一定规则排序等。在本章，将主要讨论以上几种最基本的运算。

2.2 线性表的顺序存储结构

在计算机内，可以用不同的方式来表示线性表，其中最简单和最常用的方式是用一组地址连续的内存单元依次存储线性表中各元素。

设线性表每个元素需占用 r 个存储单元，则线性表

$$(a_1,a_2,\cdots,a_n)$$

的各个元素在内存中所占的位置如图 2.1 所示。

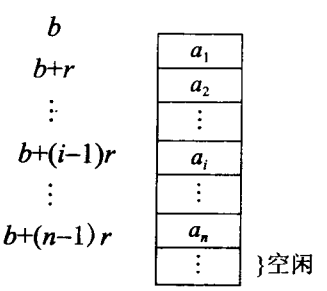

图 2.1 线性表的顺序存储结构

从图中可以看出，只要知道线性表的开始存储位置 b（也就是线性表第一个元素的开始地址），表中任一元素 a_i 的存储位置 LOC(a_i)就可用以下公式计算出来：

$$LOC(a_i)=LOC(a_1)+(i-1)*r$$

这种顺序存储结构，和许多程序语言中的一维数组的结构是非常相似的；但是我们的线性表允许插入、删除等运算，它的长度是不断变化的，为了适应这种情况，在 C 语言中要用一个结构变量来表示。

```
Struct sqlist
{
    datatype elem[maxlen];
    int     last;
};
```

这里 datatype 代表线性表元素的类型，此类型应根据实际问题再给出具体定义。maxlen 是