普通高等学校计算机基础教育“十二五”规划教材

创新系列

软件技术基础

张选芳　李廷元　付茂洺　主　编
何元清　王　欣　高大鹏　戴　蓉　副主编
刘晓东　陈华英　主　审

理论讲解全面，重点突出

内容简明清晰，图文并茂

结构设置合理，实例丰富

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

普通高等学校计算机基础教育“十二五”规划教材·创新系列

软件技术基础

张选芳　李廷元　付茂洛　主　编
何元清　王　欣　高大鹏　戴　蓉　副主编
刘晓东　陈华英　主　审

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

内 容 简 介

本书按照教育部高等学校计算机基础课程教学指导委员会提出的“三个层次五门课”系列课程体系设置的第二层次的一门基础理论课程大纲编写，系统介绍了计算机软件技术基础的基本内容，包括数据结构、计算机操作系统、软件工程、数据库技术。

本书内容丰富、重点突出，体系结构和内容选取强调基础性和实用性，符合理工科学生的认知规律，各章配有习题，以便学生练习使用。

本书可作为高等院校理工科非计算机专业学生“软件技术基础”课程教材，也可供科技人员及计算机爱好者阅读。

图书在版编目（CIP）数据

软件技术基础 / 张选芳，李廷元，付茂洺主编. -- 北京：中国铁道出版社，2013.8
普通高等学校计算机基础教育“十二五”规划教材. 创新系列
ISBN 978-7-113-17149-0

Ⅰ. ①软… Ⅱ. ①张… ②李… ③付… Ⅲ. ①软件－高等学校－教材 Ⅳ. ①TP31

中国版本图书馆 CIP 数据核字（2013）第 195981 号

书　　名：软件技术基础
作　　者：张选芳　李廷元　付茂洺　主编

策　　划：李金阳　孟　欣　　**读者热线：400-668-0820**
责任编辑：孟　欣
封面设计：刘　颖
责任印制：李　佳

出版发行：中国铁道出版社（100054，北京市西城区右安门西街 8 号）
网　　址：http://www.51eds.com
印　　刷：北京尚品荣华印刷有限公司
版　　次：2013 年 8 月第 1 版　　2013 年 8 月第 1 次印刷
开　　本：787mm×1092mm　1/16　　**印张：18**　　**字数：434 千**
印　　数：1～3 000 册
书　　号：ISBN 978-7-113-17149-0
定　　价：35.00 元

普通高等学校计算机基础教育“十二五”规划教材
创新系列

编审委员会

主　编：张选芳　李廷元　付茂洺

副主编：何元清　王　欣　高大鹏　戴　蓉

编　委：（按姓氏音序排列）

陈华英　戴　敏　华　漫　黄海洋　李尚俊
林瑞春　刘期建　路　晶　罗银辉　马　婷
潘　磊　宋　劲　宋海军　徐国标　严　宏
张　欢　张建学　张中浩　张娅岚　郑　涛
钟　晓　周　敏　朱建刚　庄　夏

FOREWORD 前言

本书是按照教育部高等院校计算机基础课程教学指导委员会提出的“三个层次五门课”系列课程体系设置的第二层次的一门基础理论课程大纲编写而成。通过本书的学习，学生会对计算机软件设计所需的基本知识和技巧有一个全面的认识，为软件设计开发工作打下坚实的基础。

为了满足各种水平读者的需要，本书内容力求由浅入深，通俗易懂，简明扼要，注重实用技术。全书着重介绍了数据结构、计算机操作系统、软件工程和数据库技术等方面的基础理论知识。

全书分四章。第1章数据结构，主要讲述算法与数据结构的基本概念及常用的典型数据结构与算法，包括链表、队列、栈、数组等线性数据结构，二叉树、哈夫曼树等树形数据结构和简单的图形数据结构。在算法方面，结合数据结构讲述了查找与排序算法。第2章计算机操作系统，主要介绍操作系统的几大管理功能：处理器管理、存储管理、作业管理、设备管理与文件管理。第3章软件工程，介绍软件工程的概念、常用开发模型以及新型软件工程技术。第4章数据库技术，主要介绍数据库的基本概念与技术，包括数据库的基础知识、数据库的数据模型、结构化查询语言、数据库设计以及新型数据库技术。

本书内容简明清晰、重点突出、实例丰富、图文并茂，并结合每章内容给出了习题，以达到通过练习巩固每章所学知识的目的。

本书由张选芳、李廷元、付茂洺任主编，何元清、王欣、高大鹏、戴蓉任副主编，刘晓东、陈华英任主审。其中，李廷元编写了第1章，付茂洺、高大鹏编写了第2章，王欣编写了第3章，何元清、戴蓉编写了第4章，最终由张选芳老师统编全稿。

本书在编写和出版过程中得到了许多老师的热情支持和帮助，在此对他们一并表示诚挚的谢意！

计算机技术的发展日新月异，作者水平有限，加之时间仓促，不足和疏漏在所难免，恳请同行和读者不吝赐教。

编者

2013年7月

CONTENT 目录

第1章 数据结构

从第一台电子计算机诞生到现在的 60 多年中，计算机的应用领域从单纯的科学计算扩展到情报检索、信息管理、工业生产、数据库应用等方面，处理对象也从原来的数值计算发展到对非数值数据的处理，而非数值数据不仅数据量大，而且类型繁多复杂，因此需要首先考虑这些数据量大且关系复杂的数据如何组织的问题，这正是数据结构的研究内容。

本章首先介绍数据结构的基本概念，然后重点介绍了数据结构中的线性结构和非线性结构，最后介绍了数据的查找和排序。

1.1 数据结构的基本概念

计算机软件离不开程序设计，而程序设计的两大基础是算法和数据结构，“程序=算法+数据结构”的概念已经广为人知。计算机软件加工处理的对象是数据，而数据具有一定的组织结构，因此数据结构就是研究数据的逻辑结构、存储结构及其运算的一门科学。

1.1.1 数据结构的研究内容及其重要性

数据结构是组织和访问数据的系统方法。自从 1946 年第一台计算机 ENIAC 问世以来，计算机已经在各个领域得到了广泛的应用。用计算机处理问题，首先需要把客观对象抽象为某种形式的数据，然后设计对这些数据进行处理的算法，由计算机执行设计好的算法，最后获得问题的处理结果。著名计算机科学家 Niklaus Wirth 在其经典著作《数据结构+算法=程序》中，认为程序其实就是数据在特定的表示方式和结构的基础上对抽象算法的具体描述，说明了数据结构的重要性。

在计算机应用的早期，计算机主要应用于科学计算，要解决的问题侧重于数值计算，处理的对象相对较为简单，如对整数、实数进行算术运算、逻辑运算等，此时用一些变量或数组等足以表示要解决的问题。但随着计算机从早期的数值计算扩展到非数值计算领域，如管理信息系统、受控热核聚变、人工智能模拟、工业过程实时控制、卫星遥感遥测及气象等领域，数据的处理对象日益复杂和多样化，处理的数据呈海量式增长，有关数据结构的研究内容已经成为编译系统、操作系统、数据库管理系统及其他系统程序和一些应用系统的重要基础。

1968 年，美国唐纳德·克努特（Donald E. Knuth）教授出版了其名著《计算机程序设计艺术》第一卷《基本算法》，首次系统地阐述了数据结构的主要内容，即数据的逻辑结构、存储结构以及对数据进行操作的各种算法。到 20 世纪 70 年代中期和 80 年代初，各种有关数据结构的著作大量问世，我国于 20 世纪 70 年代末开始设置数据结构的相关课程，现在数据结构不仅

是计算机科学与技术专业的核心课程，同时也是很多计算机应用相关专业的一门重要的选修课或必修课，因此掌握数据结构的知识对于我们进一步进行高效率的计算机程序开发非常重要。

1.1.2 数据结构的基本概念和术语

① 数据（Data）：数据是信息的载体，是描述客观事物的数、字符，以及所有能输入到计算机中、能被计算机程序识别和处理的符号的集合。数据又可以分为数值性数据和非数值性数据两大类。数值性数据如整数、实数、复数等，一般用于工程和科学计算等；非数值性数据如字符、字符串、文字、图形、语音等。

② 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。

③ 数据项（Data Item）：是具有独立含义的最小标识单位。有时，一个数据元素可由若干数据项组成。如整数集合中，10 就可以称为是一个数据元素。又如在一个关系型数据库中，一个记录可称为一个数据元素，而这个元素中的某一字段就是一个数据项。

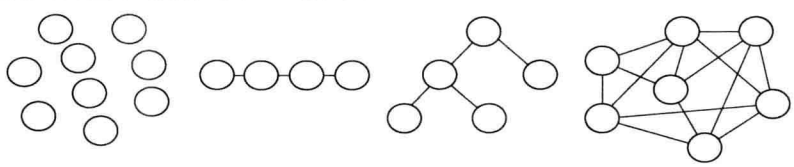
④ 数据对象（Data Object）：数据的子集。数据对象是具有相同性质的数据成员（数据元素）的集合。如整数数据对象 N={0,1,2,…}，英文字母数据对象 LETTER={'A','B',…,'Z'}。

⑤ 数据类型（Data Type）：是一个值的集合以及在这些值上定义的一组操作的总称。

⑥ 结构（Structure）：数据元素相互之间的关系称为结构，有以下四种类型的基本结构。

- 集合：结构中的数据元素之间除了“同属于一个集合”的关系外，别无其他关系。
- 线性结构：结构中的数据元素之间存在着一对一的关系。
- 树形结构：结构中的数据元素之间存在着一对多的关系。
- 图状结构或网状结构：结构中的数据元素之间存在着多对多的关系。

以上 4 种类型的基本结构如图 1-1 所示。



（a）集合　（b）线性结构　（c）树形结构　（d）网状结构

图 1-1　四种类型的基本结构

⑦ 数据结构（Data Structure）：相互之间存在着一种或多种特定关系的数据元素的集合。

数据结构的定义虽然没有统一的标准，但是它包括以下 3 方面的内容：逻辑结构、存储结构和对数据的操作。为了增加对数据结构的认识，举一个实例，如表 1-1 所示。

表 1-1　学生成绩表

学　号	姓　名	语　文	数　学	物　理
121001	王强	87	90	96
121002	李一龙	69	91	89
121003	张映月	87	79	71
121004	何一端	84	88	68
…	…	…	…	…

表1-1 是一个班级学生的成绩表（也称为成绩数据库），同时，它也构成一个数据结构。它由很多记录（数据元素）组成，每个元素又由多个数据列（字段、数据项）组成。那么这张表的逻辑结构是怎么样的呢？我们分析数据结构都是从结点（其实也就是元素、记录、顶点，虽然在各种情况下所用名字不同，但指的是同一个东西）之间的关系来分析的，对于这个表中的任一个记录（结点），它只有一个直接前驱和一个直接后继（前驱、后继就是前相邻、后相继的意思），整个表只有一个开始结点和一个终端结点。所有这些就构成了这个表的逻辑结构，亦即逻辑结构就是数据元素之间的逻辑关系。

数据的逻辑结构通常分两大类：线性结构和非线性结构。

① 线性结构：其特征是若结构为非空集，则该结构有且只有一个开始结点和一个终端结点，并且所有结点都最多只有一个直接前驱和一个直接后继。线性表就是一个典型的线性结构。

② 非线性结构：其逻辑特征是该结构中一个数据元素可能有多个直接前驱和直接后继。非线性结构中最普遍的就是图结构。

数据的存储结构是指数据的逻辑结构在计算机存储空间中的存放形式。在数据的存储结构中，不仅要存放各数据元素的信息，还需要存放各数据元素之间的前后间关系的信息。数据的存储结构有时也称为数据的物理结构，它是逻辑结构在计算机存储器里的物理实现。

表 1-1 中的数据元素在计算机中可以采取不同的存储方式：①将数据元素连续存放在一片内存单元中；②将数据元素随机地存放在不同的内存单元中，再用指针将这些数据元素链接在一起。这两种存储方式就形成两种不同的存储结构。

常用的数据存储结构有以下 4 种基本形式：

（1）顺序存储

该方法把逻辑上相邻的数据元素存储在物理位置上相邻的存储单元中，数据元素间的逻辑关系由存储单元的邻接关系来体现。由此得到的存储表示称为顺序存储结构（Sequential Storage Structure），通常借助程序语言的数组来实现。顺序存储方法主要应用于线性的数据结构。非线性的数据结构也可通过某种线性化的方法实现顺序存储。

（2）链式存储

该方法不要求逻辑上相邻的数据元素在物理位置上也相邻，数据元素间的逻辑关系由附加的指针字段表示。由此得到的存储表示称为链式存储结构（Linked Storage Structure），通常借助于程序语言的指针类型来实现。

（3）索引存储

该方法通常在存储数据元素信息的同时，还建立附加的索引表。索引表由若干索引项组成。若每个数据元素在索引表中都有一个索引项，则该索引表称为稠密索引（Dense Index）。若一组数据元素在索引表中只对应一个索引项，则该索引表称为稀疏索引（Spare Index）。索引项的一般形式是(关键字,地址)。关键字是能唯一标识一个数据元素的数据项。稠密索引中索引项的地址表示数据元素所在的存储位置；稀疏索引中索引项的地址表示一组数据元素的起始存储位置。

（4）散列存储方法

该方法的基本思想是：根据数据元素的关键字直接计算该数据元素的存储地址。

以上 4 种基本存储方法，既可单独使用，也可组合起来对数据结构进行存储。同一逻辑结构采用不同的存储方法，可以得到不同的存储结构。选择何种存储结构表示相应的逻辑结构，视具体要求而定，主要考虑运算方便及算法的时间和空间要求。

除了逻辑结构和物理结构外，数据结构的第三个内容是对数据的操作（运算），如一张表格，怎么样才能进行查找、增加、修改、删除记录等操作呢？这也就是数据的运算，它不仅仅是指加减乘除这些算术运算，在数据结构中，这些运算常常涉及算法问题。

数据结构不仅存储数据，还包括对数据的操作。这些操作的设计和实现需要使用算法，算法（Algorithm）是对特定问题求解步骤的一种描述，由有限的指令序列组成，可完成某项特定的任务。为了保证正确地处理数据，学习数据结构必然要学习算法。要理解数据结构本身，重要的是理解实现数据结构操作的算法。算法和数据结构是相辅相成、缺一不可的两个方面。数据结构是算法处理的对象，也是设计算法的基础。一个具体问题的数据在计算机中往往可以采用多种不同的数据结构来表示；一个计算过程的实现又常常有多种可用的算法。因此，选择什么样的算法和数据结构就成为实现程序过程中最重要的一个课题。

一个算法具有 5 个重要的特性：

① 有穷性：一个算法必须在执行有穷步之后结束。

② 确定性：算法的每一个步骤，必须是确切地定义的。

③ 输入：一个算法有零个或多个输入。

④ 输出：一个算法有一个或多个输出。

⑤ 可行性：一个算法的每一个步骤都应当能有效地执行，并且在执行有限的步骤之后能得到确定的结果。

在使用算法对特定的问题进行求解时，往往涉及对算法的分析。算法分析（Algorithm Analysis）是指对算法的执行时间和所需内存空间的估算。同一问题的求解往往可以使用不同的算法。通过算法分析可以比较两个算法的效率高低。对于算法所需时间和空间的估算，一般不需要将算法写成程序，在实际的计算机上运行。

① 算法的时间复杂度：执行一个算法所花费的时间代价。当要解决的问题的规模以某种单位由 1 增至 n 时，对应算法所耗费的时间也以某种单位由 $f(1)$增至 $f(n)$。此时称该算法的时间复杂度为 $f(n)$。

② 算法的空间复杂度：执行一个算法所需占用的空间代价。当要解决的问题的规模以某种单位由 1 增至 n 时，对应算法所需占用的空间也以某种单位由 $g(1)$增至 $g(n)$。此时称该算法的空间复杂度为 $g(n)$。

③ 问题的规模：指的是算法要解决的问题所要处理的数据量的大小。例如，对 n 个记录进行排序，n 就是问题的规模。再例如，求 n 阶矩阵的转置矩阵，n 也可以看做是问题的规模。时间单位一般规定为一个简单语句（如赋值语句、条件判断语句等）所用的时间。空间单位一般规定为一个简单变量（如整型、字符型、浮点型等）所占用的存储空间大小。

要全面地分析一个算法，应考虑该算法在最坏情况下的代价、最好情况下的代价、平均情况下的代价。然而，要全面准确地分析每一个算法是相当困难的，一般考虑这个算法在最坏情况下的代价。在多数情况下，只要得到一个估计值就足够了。

在描述算法分析的结果时，通常采用 O 表示法：即某个算法的时间代价（或空间代价）为 $O(f(n))$，如果存在正常数 c 和 n_0。当问题的规模 $n \geqslant n_0$ 时，该算法的时间代价（或空间代价）为 $T(n) \leqslant c \cdot f(n)$，这时也称该算法的时间（或空间）代价的增长率为 $f(n)$。这种说法意味着，当 n 充分大时，该算法的复杂度不大于 $f(n)$的一个常数倍。

采用 O 表示法简化了时间复杂度和空间复杂度的度量，其基本思想是关注复杂性的数量级，而忽略数量级的系数，这样在分析算法的复杂度时，可以忽略零星变量的存储空间和个别语句

的执行时间，重点分析算法的主要代价。

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 $O(n^2)$、立方阶 $O(n^3)$、k 次方阶 $O(n^k)$、指数阶 $O(2^n)$。

理解了逻辑结构、存储结构以及对数据的操作三个问题，就可以理解数据结构的概念。

在不引起混淆的情况下，通常将数据的逻辑结构简称为数据结构，但要清楚的是，数据结构研究的不仅仅是数据的逻辑结构，还包含对数据的存储结构以及数据的操作的研究。

1.1.3　数据结构、数据类型和抽象数据类型

数据结构用来反映一个数据的内部构成，即一个数据由哪些数据元素构成，以什么方式构成，呈什么结构。数据结构包括逻辑上的数据结构和物理上的数据结构。逻辑上的数据结构反映数据元素之间的逻辑关系，物理上的数据结构反映数据元素在计算机内的存储方式。数据结构是数据存在的形式。

数据按照数据结构分类，具有相同数据结构的数据属同一类。同一类数据的全体称为一个数据类型。在高级程序设计语言中，数据类型用来说明一个数据在数据分类中的归属。它是数据的一种属性。这个属性限定了该数据的变化范围。为了解题的需要，根据数据结构的种类，高级语言定义了一系列的数据类型。不同的高级语言所定义的数据类型不完全相同。例如，C++语言所定义的数据类型如图 1-2 所示。

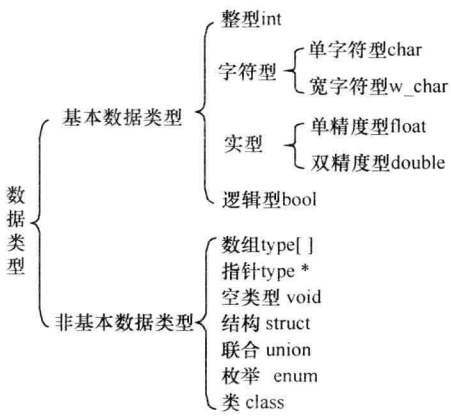


图 1-2　C++的数据类型

其中，基本数据类型对应于简单的数据结构，非基本数据类型对应于复杂的数据结构。在复杂的数据结构中，允许数据元素本身具有复杂的数据结构，因而，非基本数据类型允许复合嵌套。指针类型对应于数据结构中数据元素之间的关系，表面上属基本数据类型，实际上都指向复杂的数据元素即构造数据类型中的数据，因此这里没有把它划入基本数据类型中，而是把它划入非基本数据类型中。

数据结构反映数据内部的构成方式，常常用一个结构图来描述：数据中的每一项，数据元素被看做一个结点，并用方框或圆圈表示，数据元素之间的关系用带箭头的连线表示。如果数据元素本身又有它自身的结构，则结构出现嵌套。这里嵌套还允许是递归的嵌套。

指针数据的引入使构造各种复杂的数据结构成为可能。

按照数据结构中的数据元素之间的关系，数据结构有线性与非线性之分。在非线性数据结

构中又有层次与网状之分。由于数据类型是按照数据结构划分的，因此一类数据结构对应着一种数据类型。数据类型按照该类型中的数据所呈现的结构也有线性与非线性之分和层次与网状之分。一个数据变量在高级语言中的类型说明必须是该变量所具有的数据结构所对应的数据类型。

抽象数据类型（Abstract Data Type，ADT）是带有一些操作的数据元素的集合，是一种描述用户和数据间接口的抽象模型，ADT 的主要功能是简单而明确地描述数据结构的操作。此处的“抽象”意味着应该从与实现方法无关的角度去研究数据结构。抽象数据类型为用户提供了一种定义数据类型的手段，其关键的两个要素为数据的结构以及在该结构上相应的操作的集合。

引入抽象数据类型的目的是把数据类型的表示和数据类型上运算的实现与这些数据类型和运算在程序中的应用隔开，使它们相互独立。对于抽象数据类型的描述，除了必须描述它的数据结构外，还必须描述定义在它上面的运算（过程或函数）。抽象数据类型上定义的过程和函数以该抽象数据类型的数据所应具有的数据结构为基础。

下面的各节将讨论一些基本抽象数据类型。所谓基本只是相对而言，这些数据类型是最基本、最简单的，并且是实现其他抽象数据类型的基础。

在后面的内容中我们将了解到表、栈、队列、串、树、图等常见的基本抽象数据类型的 ADT 操作以及这些操作用不同数据描述方法的具体实现。

1.2 线 性 结 构

线性结构是数据结构中最简单且最常用的一种数据结构。线性结构的基本特点是数据元素有序并且是有限的。

线性结构包括以下以种常见的数据类型：线性表、栈、队列、数组、串等。

1.2.1 线性表

线性表（Linear List）是 n（$n\geq 0$）个相同类型的元素 $a_0,a_1,a_2,a_3,\cdots,a_n$ 所构成的有限线性序列，通常表示为$(a_0,a_1,a_2,a_3,\cdots,a_n)$，其中 n 为线性表的长度。a_i（$0\leq i\leq n$）是线性表中第 i 个序号的数据元素。a_i 是抽象表示符号，在不同的情况下含义不同。例如一个整数序列(7,8,9,10,11,12,34)是一个线性表，表中每一个元素 a_i 均为整数，表长为 7。

以上每个数据元素包括一个数据项，而 1.1.2 节中的学生成绩表数据，对于每一位学生 student1、student2、student3…每个数据元素均包括多个数据项。

1. 线性表的概念

简单地说，线性表就是 n 个数据元素的有限序列。线性表具有如下特点：存在唯一的被称为“第一个”的数据元素；存在唯一的被称为“最后一个”的数据元素；除第一个数据元素之外，线性表中的每个数据元素均只有一个前驱；除最后一个数据元素之外，线性表中的每个数据元素均只有一个后继。

在线性表中，一个元素可以由若干数据项组成，在这种情况下的数据元素称为记录，而含有大量记录的线性表又称为文件。线性表中元素的个数 n 定义为线性表的长度，$n\geq 0$。

同一个线性表中的元素必定具有相同特性，即属于同一数据对象，相邻数据元素之间存在

着序偶关系，即数据元素是“一个接一个地排列在一起”：

$$(a_0,a_1,\cdots,a_{i-1},a_i,a_{i+1},\cdots,a_n)$$

其中，a_{i-1} 为 a_i 的直接前驱，a_{i+1} 为 a_i 的直接后继，a_i 是第 i 个元素，称 i 为数据元素 a_i 在线性表中的位序。

线性表是相当灵活的一种数据结构，如长度可根据需要增减，元素也可以增删。

线性表的基本操作有以下几种：

① Initiate(L){初始化}：设定一个空的线性表。

② Length(L){求表长}：对给定的线性表 L，函数返回值为其数据元素的个数。

③ Get(L,i){取元素}：若给定的数据元素序号 i 满足 $1\leqslant i\leqslant$Length(L)，则函数返回值为给定线性表 L 的第 i 个元素 a_i，否则返回空元素。

④ Locate(L,x){定位}：对给定值，若线性表 L 中存在某个数据元素 a_i 等于 x，则函数返回索引号最小的 i 的值；若 L 中不存在等于 x 的数据元素，则函数返回一个特殊值（比如-1），以说明不存在的位置。

⑤ Insert(L,i,x){插入}：对给定的线性表 L，若索引号 i 满足 $1\leqslant i\leqslant$Length(L)+1，则在线性表的第 i 个位置上插入一个新的数据元素 x，使原来表长度为 n 的线性表变为表长为 n+1 的线性表，并使函数返回值为 true，否则函数返回值为 false。

⑥ Delete(L,i){删除}：对给定的线性表，若索引号 i 满足 $1\leqslant i\leqslant$Length(L)，则把线性表中第 i 个元素 a_i 删除，使原来表长为 n 的线性表变成表长为 n-1 的线性表，并使函数返回值为 true，否则函数返回值为 false。

对于线性表的其他有关操作，如线性表的合并、排序等操作，都可以通过以上的基本操作来实现。

2. 线性表的抽象数据类型的定义

```
ADT List{
    数据对象: D={ai|ai∈Elemset, i=1,2,…,n,n≥0}
    数据关系: R1={<ai-1,ai>|ai-1,ai∈D,i=2,…,n}
    基本操作:
    InitList(&L)
    操作结果: 构造一个空的线性表 L
    DestroyList(&L)
    初始条件: 线性表已存在
    操作结果: 销毁线性表 L
    ClearList(&L)
    初始条件: 线性表已存在
    操作结果: 置线性表 L 为空表
    ListEmpty(L)
    初始条件: 线性表已存在
    操作结果: 若线性表 L 为空表, 则返回 Ture, 否则返回 False
    ListLenght(L)
    初始条件: 线性表已存在
    操作结果: 返回线性表 L 数据元素个数
    GetElem(L,i,&e)
    初始条件: 线性表已存在 (1≤i≤ListLenght(L))
```

操作结果：用 e 返回线性表 L 中第 i 个数据元素的值
locatElem(L,e,compare())
初始条件：线性表已存在，compare()是数据元素判定函数
操作结果：返回线性表 L 中第 1 个与 e 满足关系 compare()的数据元素的位序
PriorElem(L,cur_e,&pre_e)
初始条件：线性表已存在
操作结果：若 cur_e 是线性表 L 的数据元素，且不是第一个，则用 pre_e 返回它的前驱，否则操作失败，pre_e 无定义
NextElem(L,cur_e,&)
初始条件：线性表已存在
操作结果：若 cur_e 是线性表 L 的数据元素，且不是第最后一个，则用 next_e 返回它的后继，否则操作失败，next_e 无定义
ListInsert(&L,i,e)
初始条件：线性表已存在（1≤i≤ListLenght(L)+1）
操作结果：在线性表 L 中第 i 个数据元素之前插入新元素 e，L 长度加 1
ListDelete(&L,i,&e)
初始条件：线性表已存在（1≤i≤ListLenght(L)）
操作结果：删除线性表 L 中第 i 个数据元素，用 e 返回其值，L 长度减 1
ListTraverse(L,visit())
初始条件：线性表已存在
操作结果：依次对线性表 L 的每个数据元素调用 visit()函数，一旦 visit()失败，则操作失败

}ADT List

上述是线性表抽象数据类型的定义，其中只是一些基本操作，另外可以更复杂。如将两个线性表合并等。

3．线性表的顺序存储结构及其实现——顺序表

线性表的存储结构有多种类型，如顺序存储结构和链式存储结构，因此线性表可以采用使用顺序存储结构的顺序表和有序顺序表来实现，也可以采用链式存储结构的单链表、双链表和循环链表等来实现。

（1）顺序表的定义

① 顺序存储结构：即把线性表的数据元素按逻辑次序依次存放在一组地址连续的存储单元中的方法。

② 顺序表（Sequential List）：用顺序存储结构实现的线性表简称为顺序表（Sequential List）。按数据元素的值无序或有序存放，顺序表又可以分为无序顺序表和有序顺序表两种。

（2）数据元素的存储地址

设线性表中所有数据元素的类型相同，则每个数据元素所占用的存储空间大小亦相同。当把线性表的数据元素存入存储空间后，称这样的一种存储空间为一个“结点”。假设表中每个结点占用 c 个存储单元，并设表中第一个结点 a_1 的存储地址（简称基地址）是 $\mathrm{Loc}(a_1)$，那么结点 a_i 的存储地址 $\mathrm{Loc}(a_i)$可通过下式计算：

$$\mathrm{Loc}(a_i)=\mathrm{Loc}(a_1)+(i-1)\times c \quad (1\leqslant i\leqslant n)$$

在顺序表中，每个结点 a_i 的存储地址是该结点在表中的位置 i 的线性函数。只要知道基地址和每个结点的大小，就可在相同时间内求出任一结点的存储地址，因此顺序表是一种随机存取存储的结构。