简明数据结构

主　编　刘渝妍
副主编　向　毅　柴世红

重庆大学出版社

内容简介

本书对数据结构的有关知识做了全面系统的介绍,内容包括:绪论,线性表,栈和队列,串和数组,树,图,查找,内部排序,参考文献等。

本书在内容组织上力求概念清晰,注重数据结构的实际应用。对算法设计做了详细、通俗的讲解,每章有小结和适量的习题。

本书可供高职高专计算机相关专业的学生使用,也可作为高等院校非计算机专业数据结构课程的教材或参考书。

图书在版编目(CIP)数据

简明数据结构/刘渝妍主编. —重庆:重庆大学出版社,2004.7
(高职高专计算机系列教材)
ISBN 7-5624-3139-6

Ⅰ.简… Ⅱ.刘… 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(2004)第 041862 号

简明数据结构

主编 刘渝妍

副主编 向 毅 柴世红

责任编辑:谭 敏 版式设计:谭 敏

责任校对:任卓惠 责任印制:张立全

*

重庆大学出版社出版发行

出版人:张鸽盛

社址:重庆市沙坪坝正街 174 号重庆大学(A 区)内

邮编:400030

电话:(023)65102378 65105781

传真:(023)65103686 65105565

网址:http://www.cqup.com.cn

邮箱:fxk@cqup.com.cn(市场营销部)

全国新华书店经销

重庆大学建大印刷厂印刷

*

开本:787×1 092 1/16 印张:14.25 字数:356 千

2004 年 6 月第 1 版 2004 年 6 月第 1 次印刷

印数:1—5 000

ISBN 7-5624-3139-6/TP·482 定价:19.50 元

前言

当前，我国高职教育得到了迅速发展，但其教材建设严重滞后，在这种情况下，编写适应高等职业教育培养技术应用性人才要求的，真正具有高职特色的教材十分必要而且迫切。在重庆大学出版社的大力支持下，我们组织了一批具有经验且长期在教学一线工作的教师，编写了《简明数据结构》一书。

本书对数据结构的有关知识做了系统全面地介绍，在内容组织上力求概念清晰，注重数据结构的实际应用。本书以C语言做为算法描述语言来讲述数据结构，结合高职高专教育的实际，以理论够用为度进行教学内容组织，强调理论与实验密切结合。内容包括绪论、线性表、栈、队列、串、数组、树、二叉树、图、查找、排序等。全书以结构化程序设计的思想对各种数据结构的算法做了详细、通俗的讲解，给出了每章相关知识在实际应用中的例子。由于“数据结构”是一门实践性环节较强的专业基础课，所以在每一章的最后，都设计了相关的实验内容，每章实验的部分项目都给出了C语言源程序，所有程序均在Tourbo c 2.0环境下调试通过。对于实验，要求读者在给出的源程序基础上，自行进行扩展，完成其余的实验项目。除此之外，每章最后还备有适量的习题，以供读者练习。

本书由刘渝妍担任主编，向毅、柴世红担任副主编。其中第1章、第2章由刘渝妍编写，第3章、第4章由向毅编写，第5章由向毅和廖继红编写，第6章由任照富编写，第7章由刘渝妍和梁富毫编写，第8章由柴世红编写。由于编者水平有限，编写时间较短，书中不妥之处，欢迎读者批评指正。

编者

2004年2月

目录

第 1 章 绪 论

用计算机解决任何问题都离不开程序设计,在程序开发过程中通常需要做到如下两点:高效地描述数据和设计一个好的算法,且该算法最终可用程序来实现。要想高效地描述数据,必须具备数据结构领域的知识;而要想设计一个好的算法,则需要算法设计领域的知识。数据结构主要研究数据的逻辑结构、存储结构及相应运算的算法并对算法的效率进行分析。

1.1 引 言

自 1946 年第一台计算机问世以来,计算机已深入到人类社会的各个领域,计算机加工处理的对象也由纯粹的数值数据发展到字符、表格、图像、声音等各种具有一定结构的数据。数据(Data) :是指能够输入到计算机中,并能被计算机存储、识别和处理的符号集合。

由于数据的表示方法和组织形式直接关系到程序对数据的处理效率,而系统程序和许多应用程序的规模很大,结构相当复杂,处理的对象又多为非数值型数据。因此,单凭程序员的经验和技巧已难以设计出效率高、可靠性强的程序。这就要求人们对计算机程序加工的对象进行系统的研究,即研究数据的特性以及数据之间存在的关系——数据结构(Data Structure) 。

下面用一个简单的例子来说明数据之间是存在关系的。

例 1.1　设有一个同学通讯录如表 1.1 所示。

表 1.1　同学通讯录

姓名	性别	电话号码	住址
张文	男	12345	新闻小区
李阳	男	24689	金康小区
王平	女	13579	正义路 23 号
王英	女	23456	青年路 85 号
吕强	男	32145	园丁小区
胡明	男	67549	东风路 32 号
……	……	……	……

通讯录表格就是一个数据,它是由各位同学的信息(数据元素)构成的。现要求写一个算法,当给定一个姓名时,该算法能够查出此人的住址。

如果通讯录中的姓名是随意排列的,其次序没有任何规律。那么给定一个姓名时,只能对通讯录从头开始逐个与给定的姓名进行比较,顺序查对,直至找到所给定的姓名或搜索到表尾为止。这种方法相当浪费时间,效率很低。

这个算法的设计,将完全依赖于通讯录中同学姓名及相应的住址是如何构成的,以及计算机是怎样存储通讯录中的信息。如果对通讯录进行适当的组织,如:

方法一,将姓名按字典顺序排列,则可采用折半查找的方法进行查找,从而减少比较次数,提高查找效率。

方法二,按同学的性别将通讯录拆分成两个子表,如表 1.2 所示。

表 1.2　同学通讯录

子表 1			
姓名	性别	电话号码	住址
张文	男	12345	新闻小区
李阳	男	24689	青年路 85 号
吕强	男	32145	园丁小区
胡明	男	67549	东风路 32 号
……	……	……	……

子表 2			
姓名	性别	电话号码	住址
王平	女	13579	金康小区
王英	女	23456	正义路 23 号
……	……	……	……

现在如果要查找某个男同学的住址,只需到子表 1 中进行查找,而不必去查看子表 2 的姓名,避免了对所有同学进行扫描,从而提高了查找效率。

由此可得出以下结论:

结论 1. 数据元素之间是有联系的。如表 1.1 中,数据元素之间是一个接一个的联系。这些联系常常影响算法的选择和效率。

结论 2. 数据元素之间是有结构的。如表 1.1 的结构是线性结构。

结论 3. 数据的运算效率是与数据在计算机中的存储方式有关的。如表 1.1 和表 1.2,它们的查找效率是不一样的。

结论 4. 在某种结构上可定义一组运算。如例 1 中的查找就是一种运算。若要在通讯录中插入(删除)一位同学的有关数据,则插入(删除)也是一种运算。

这个例子说明了数据中各个数据元素不是孤立存在的,它们之间存在着一定的结构关系。要编写出一个“好”的程序,就必须分析数据元素的特性及数据元素之间存在的关系。

数据结构实质上是研究数据及数据之间关系的一门学科,其研究的内容如图 1.1 所示。图中用虚线框起的部分就是数据结构要研究的内容。即:

(1) 如何表示这些数据元素的结构关系(逻辑结构)。

(2) 怎样在计算机中存储数据元素(存储结构)。

(3) 对每种逻辑结构进行相适应的处理(运算)。

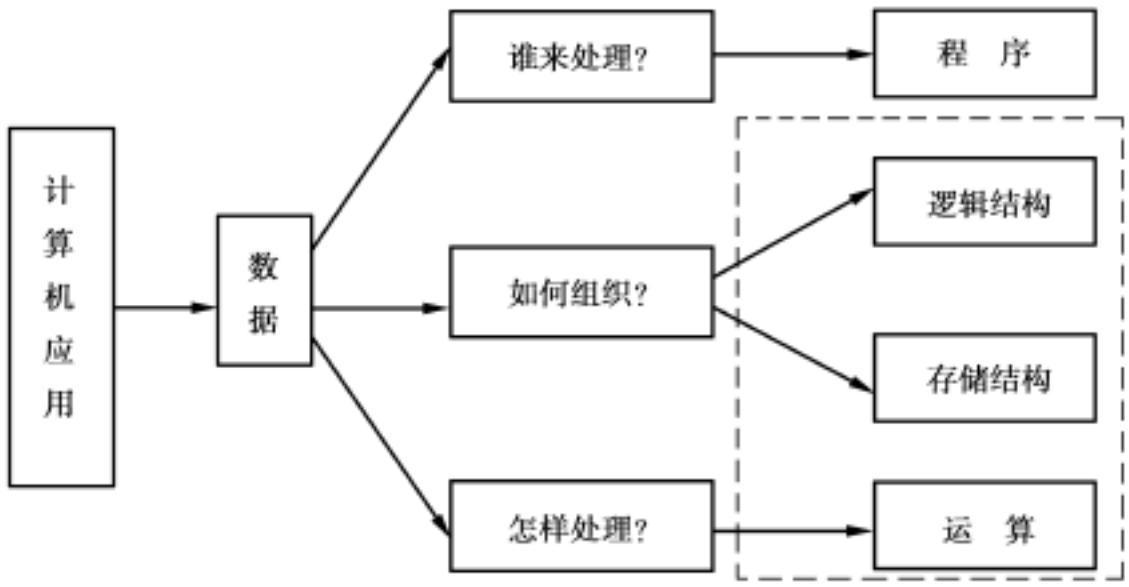

图 1.1　数据结构的研究内容

1.2　逻辑结构和数据结构

1.2.1　数据元素和数据项

数据元素(Data Element) 是数据的基本单位, 具有完整明确的实际意义。在很多情况下, 由于它还可以分割成若干个数据项(字段) , 故不是组成数据的最小单位。在后续内容中, 根据需要数据元素也被称为结点、顶点或记录。

数据项(Data Item) 是组成数据元素的最小单位。也称为字段或域。

例如, 在同学通讯录中, 一条记录(张文、男、12345、新闻小区) 是一个数据元素, 是由“ 张文 ”、“ 男 ”、“ 12345 ”、“ 新闻小区 ”这 4 个数据项组成的, 它具有实际意义; 而单独一个数据项如“ 12345 ”是没有实际意义的。

数据、数据元素和数据项反映了数据组织的 3 个层次。数据由若干个数据元素构成, 而数据元素又可由若干个数据项构成。

1.2.2　数据对象

数据对象(Data Object) 是性质相同的数据元素组成的集合, 是数据的一个子集。

例如, 整数数据对象的集合可表示为 N = { 0, ±1, ±2, …} ; 字母字符数据对象的集合可表示为 C = {‘ A ’,‘ B ’, …,‘ Z ’} 。一个班级的成绩表也可以看做是一个数据对象。

1.2.3　数据类型

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。当一个数据的类型确定后, 就确定了该数据占内存的字节数、数据取值的范围和在其上可进行的操作运算。

例如, 高级语言中用到的 16 位带符号的整数数据类型, 是指由 - 32 768 ～+ 32 767 中值构成的集合, 及一组操作(加、减、乘、除、乘方等) 的总称。

1.2.4 数据的逻辑结构

逻辑关系: 数据元素之间的关联方式(邻接关系)。例如, 在表 1.1 中, 第一条记录与第二条记录是邻接的即相互关联的, 因此这两个数据元素之间有邻接的逻辑关系。但第一条记录与第三条记录不相邻接即相互不关联, 因此它们之间没有邻接的逻辑关系。

逻辑结构: 数据元素之间逻辑关系的整体。

实质上, 数据的逻辑结构就是数据的组织形式。根据数据元素之间关系的不同特性, 通常有 4 种基本的逻辑结构, 如图1.2 所示。

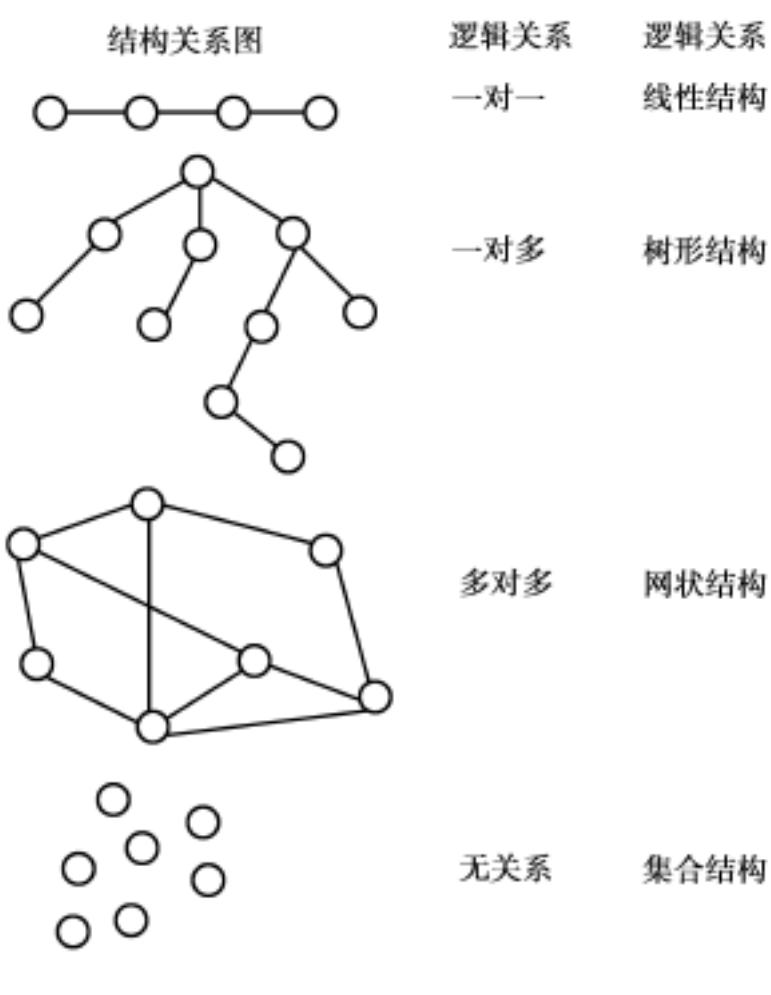

图 1.2 4 种基本逻辑结构示意图

线性结构中结点按逻辑关系依次排列成一条“ 锁链 ”; 树形结构有分支、层次特性, 其形态有点像自然界中的树; 网状结构中各结点按逻辑关系互相缠绕, 任何两个结点都可以相邻接; 集合结构中任何两个结点都没有逻辑关系, 组织形式松散。

例 1.2 通讯录中相邻数据元素之间存在着一对一的关系, 即线性结构。

例 1.3 设某大学教师管理机构如图 1.3 所示。相邻结点之间存在着一对多的关系, 即树形结构。

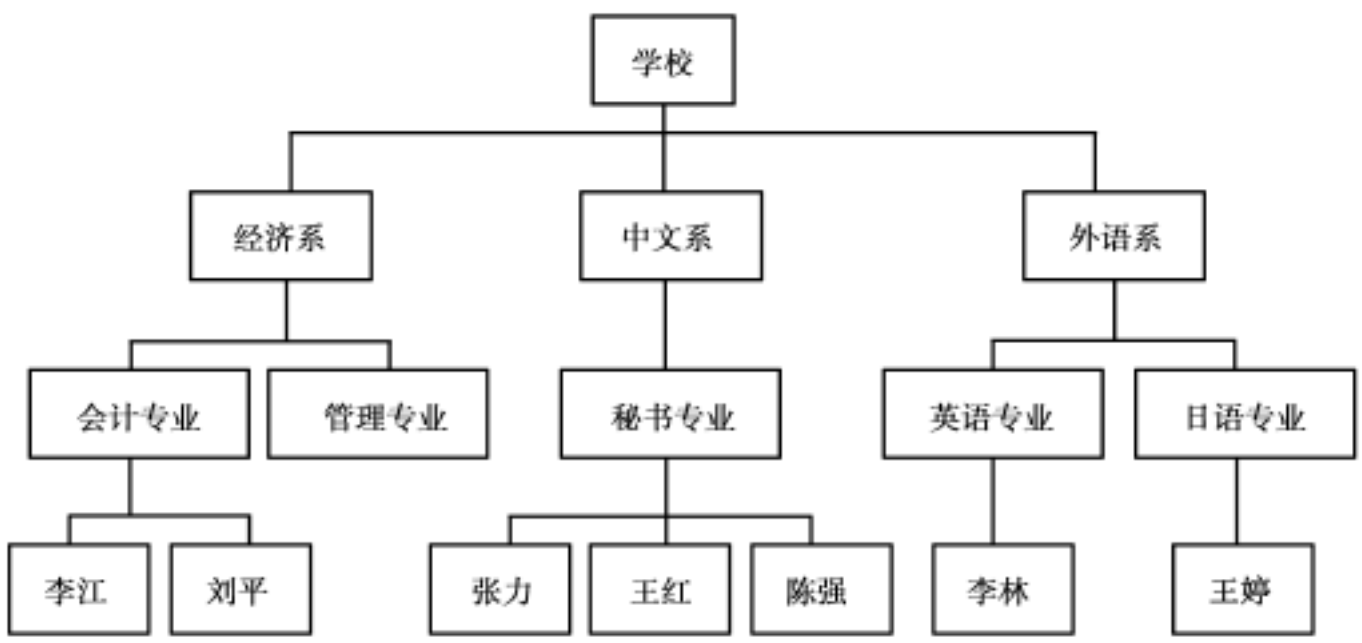

图 1.3 教师管理机构

例 1.4 在如图 1.4 所示的交通咨询网中, 相邻结点之间存在着一对多的关系, 即网状结构。

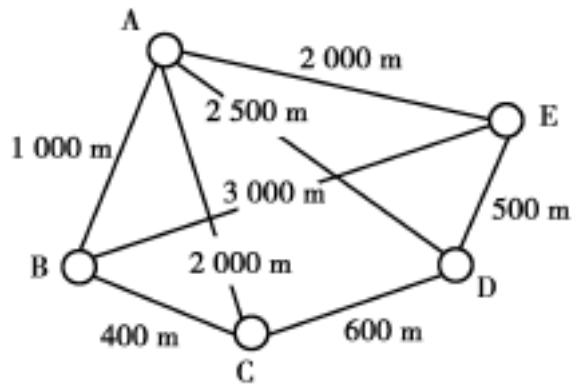

图 1.4　交通网

例 1.5　计算机专业的同学参加一次活动, 同学之间除了“ 同属于计算机专业 ”的关系外, 别无其他关系, 即集合结构。又例如, 红色、绿色、蓝色也组成一个颜色集合。

为什么“ 同学 ”与“ 颜色 ”这些不相同的数据可以有相同的逻辑结构?

这是由于逻辑结构的特性决定的。

(1) 逻辑结构与数据元素本身的形式、内容无关。例如, 在表 1.1 中, 若增加一个数据项“ 工作单位 ”, 就得到另一张表, 但由于新表的所有数据元素仍是按“ 一个接一个排列 ”的, 其逻辑结构与原表的逻辑结构相同。

(2) 逻辑结构与所含数据元素个数无关。例如, 表 1.1 中增加或删除一个同学的信息, 所得的表格仍为线性结构。

(3) 逻辑结构与数据元素间的相对位置无关。例如, 将表 1.1 中的所有数据元素按姓名的字典顺序重新排列后, 得到另一张表, 仍是线性结构。

上述逻辑结构是按数据元素间的关系来划分的, 也可按逻辑结构的特征来分类, 将数据的逻辑结构分为两大类: 线性结构和非线性结构。

线性结构的逻辑特征是: 若结构是非空集, 则有且仅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趋和一个直接后继。例如, 线性表是一个线性结构。

非线性结构的逻辑特征是: 一个结点可能有多个直接前趋和直接后继。例如, 树和图是非线性结构。

1.2.5　数据结构

数据结构(Data Structure) : 是带有结构的数据元素的集合。所谓结构就是数据元素之间的关系, 即描述数据元素之间的运算及运算规则。

数据结构的抽象描述: 用集合的形式进行描述, 数据结构是一个二元组:

DS = (D, R)

其中: D = { a_1, a_2, …, a_n } 是数据元素的集合, R = { r_1, r_2, …, r_m } 是 D 上关系的集合。

例 1.6　设有一个数据结构(D, R) , 其中数据元素集合 D = { a_1, a_2, a_3, a_4, a_5 }, 及其上的关系 R = {〈a_1, a_2〉,〈a_2, a_3〉,〈a_3, a_4〉,〈a_4, a_5〉}, 则它的逻辑结构可描述如下:

$$a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow a_4 \rightarrow a_5$$

例 1.7　设有一个数据结构 DS = (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〉}, 则它的逻辑结构用图描述如图 1.5 所示。

例 1.8　设有一个数据结构 DS = (D, R) , 其中 D = {1, 2, 3, 4}, R = {(1, 2) , (1, 3) , (1, 4) , (2, 3) , (2, 4) , (3, 4) }, 则它的逻辑结构用图描述如图 1.6 所示。

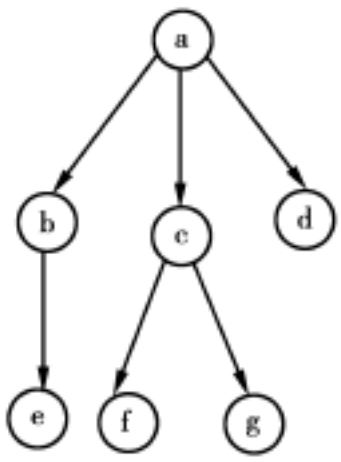

图 1.5　树形结构抽象描述示意图

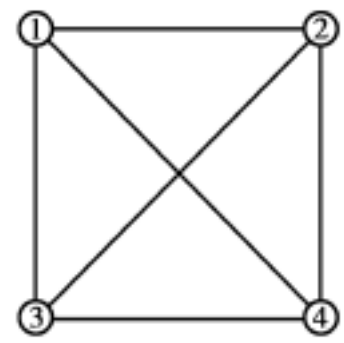

图 1.6　图形结构抽象描述示意图

数据结构包含三个方面的内容,即数据的逻辑结构、存储结构和对数据所施加的运算。这三个方面的关系是:

(1)数据的逻辑结构独立于计算机,是数据本身所固有的。

(2)存储结构是逻辑结构在计算机存储器中的映像,必须依赖于计算机。

(3)运算是指所施加的一组操作总称。运算的定义直接依赖于逻辑结构,但运算的实现必依赖于存储结构。

运算是逻辑结构上的操作,此操作不改变逻辑结构的类型。由于运算只能描述处理的功能,不包括处理的步骤和方法,因此有关运算的实现,即算法设计将在 1.4 节进行介绍。

例 1.9　试举一个数据结构的例子,叙述其逻辑结构、存储结构、运算 3 个方面的内容。

解　例如表 1.1,记录了同学的通讯录,它是以同学的信息为一行组成的表。这个表就是一个数据结构。每个记录(有姓名,性别,电话号码,住址等数据项)就是一个数据元素,对于整个表来说,只有一个开始结点(它的前面无记录)和一个终端结点(它的后面无记录),其他的结点则各有一个也只有一个直接前趋和直接后继。这就确定了这个表的逻辑结构是线性结构。

那么怎样把这个表中的数据存储到计算机里呢?用高级语言如何表示各结点之间的关系呢?是用一片连续的内存单元来存放这些记录(如用数组表示),还是随机存放各数据元素再用指针进行链接呢?这就是存储结构的问题。

有了这个表,就可对这张表中的记录进行查询、修改、删除等操作。对这个表可以进行哪些操作以及如何实现这些操作就是数据的运算问题了。

1.3　存 储 结 构

由于计算机在实际进行数据处理时,被处理的数据元素必须先存放到计算机的存储空间中,所以数据的存储也是数据结构研究的内容。

通常将数据的逻辑结构在计算机存储空间中的存放形式称为存储结构,又称为物理结构。在数据的存储结构中,不仅要存放各数据元素的信息,还要存放各数据元素之间相邻关系的信息。

存储结构包括 3 个方面的内容:
- 存储结点,一个结点存放一个元素;
- 存储逻辑结构(逻辑结构的机内表示);
- 附加设施(指针、索引表)。

由于数据元素之间的关系在计算机中可以有不同的表示方式,因此存储结构在计算机中有不同的存储方式。

(1) 顺序存储方式

把逻辑上相邻的结点存储在物理位置上相邻的存储单元里,结点间的逻辑关系由存储单元的邻接关系来体现。所有元素存放在一片连续的存储单元中,逻辑上相邻的元素存放到计算机内存仍然相邻。

(2) 链接存储方式

该方式不要求逻辑上相邻的结点在物理位置上亦相邻,结点间的逻辑关系是由附加的指针字段表示的。即所有元素存放在可以不连续的存储单元中,但元素之间的关系可以通过指针来确定,且逻辑上相邻的元素存放到计算机内存后不一定是相邻的。

(3) 索引存储方式

在存储结点信息的同时,还要建立附加的索引表,索引表中的每一项称为索引项,索引项的一般形式是:(关键字,地址),其中的关键字是能惟一标识一个结点的那些数据项,地址作为指向结点的指针。

(4) 散列存储方式

该方式的基本思想是根据结点的关键字构造散列函数,用函数的值来确定数据元素的存放地址。

一般来说,采用不同的存储方式,其数据处理的效率有所不同,因此在数据处理中选择合适的存储方式是相当重要的。

数据的逻辑结构和存储结构是密切相关的两个方面,任何一个算法的设计取决于选定的逻辑结构,而算法的实现依赖于采用的存储结构。

1.4　算法和算法分析

当程序员对问题的描述清楚后,想利用计算机来解决问题时,就要编写程序,但如果直接开始写程序可能有些复杂,所以一般都必须是先从算法入手。

1.4.1　算法

实现一个运算的方法和步骤称为此运算的算法。通俗地讲,算法就是一种解题的方法,是由若干条指令组成的有穷序列。

一个算法必须具有下列 5 个重要特性:

(1) 输入:一个算法具有 0 个或多个输入,这些输入取自于某个特定的对象集合;

(2) 输出:一个算法至少产生一个输出,这些输出是同输入有着某些特定关系的量,是算法执行完后的结果;

(3) 有穷性:一个算法必须在执行有限步之后结束,这就要求每条指令的执行次数必须是

有限的;

(4) 确定性: 算法中每条指令的含义都必须是明确的, 读者理解时不会产生二义性, 且在任何条件下, 算法只有惟一的一条执行路径, 对于相同的输入只能得出相同的输出;

(5) 可行性: 一个算法是能行的, 即算法中描述的操作都是可以通过已经实现的基本运算执行有限次来实现的。

1.4.2 算法描述

运算的实现要用算法来执行, 而算法需要用一种语言来描述。为了使算法的描述简单明确、一目了然、便于理解及掌握算法的思想和实质, 必须慎重地选择描述算法的语言工具。描述算法可以采用以下工具。

流程图: 一个算法可以用流程图的方式来描述, 输入、输出、判断、处理分别用不同的框图表示, 用箭头表示流程的流向。这是一种描述算法的较好方法, 目前在一些高级语言程序设计中仍然采用。

自然语言: 用日常生活中的自然语言(可以是中文形式, 也可以是英文形式) 也可以描述算法。

类语言: 为了方便, 通常不严格按照计算机程序设计语言的语法规定, 只使用其主要成分, 而忽略细节, 把这种描述称为用“ 伪语言 ”或“ 类语言 ”形式的描述算法。

在本教材中, 将采用类 C 语言来描述算法, 这样便于阅读、编写。类 C 语言来自对标准 C 语言的修改, 用类 C 语言描述的算法通常遵循如下规则:

(1) 预定义常量和类型

```
//函数结果状态代码
#define   TRUE          1
#define   FALSE         0
#define   OK            1
#define   ERROR         0
#define   OVERFLOW      -1
//Status 是函数的类型, 其值是函数结果状态代码
typedef int Status
```

(2) 数据结构的表示

数据元素类型约定为 ElemType, 碰到具体的应用时, 用户可根据实际定义其类型。

(3) 函数定义

基本运算的算法都用以下形式的函数描述:

```
函数类型 函数名( 函数参数表)
{  //算法说明
     语句序列
} //函数名
```

除了函数的参数需要说明类型外, 算法中使用的辅助变量可以不做变量说明。当函数返回值为函数结果状态代码时, 函数可定义为 Status 类型。

在形参表中有两种参数: 赋值参数只为运算提供输入值; 引用参数以 & 打头, 除可提供输

入值外, 还将返回操作结果。

例如, 有一个函数为 void Swap(&i, &j, k) 。则 i, j 为引用参数, k 为赋值参数。

(4) 赋值语句

简单赋值　变量名 = 表达式;

串联赋值　变量名 1 = 变量名 2 = … = 变量名 n = 表达式;

交换赋值　变量名←→变量名;

条件赋值　变量名 = 条件表达式? 表达式 T: 表达式 F;

成组赋值　(变量名 1, …, 变量名 n) = (表达式 1, …, 表达式 n) ;

结构名 = 结构名;

变量名[起始下标……终止下标] = 变量名[起始下标……终止下标] ;

(5) 选择语句

条件语句 1 if(表达式) 语句;

条件语句 2 if(表达式) 语句;
else 语句;

开关语句 1 switch
{ case 值 1: 语句序列 1; break;
…
case 值 n: 语句序列 n; break;
default: 语句序列 n + 1;
}

开关语句 2 switch
{ case 条件 1: 语句序列 1; break;
…
case 条件 n: 语句序列 n; break;
default: 语句序列 n + 1;
}

(6) 循环语句

for 语句　for(赋初值表达式序列; 条件; 修改表达式序列) 语句;

while 语句　while(条件) 语句;

do - while 语句　do {
语句序列;
} while(条件) ;

(7) 结束语句

函数结束语句　return 表达式;
return;

异常结束语句　exit(异常代码) ;

(8) 输入输出语句

输入语句　scanf([格式串] , 变量 1, …, 变量 n) ;

输出语句　printf([格式串] , 表达式 1, …, 表达式 n) ;

通常省略格式串。

(9) 注释

单行注释　　//文字序列

(10) 基本函数

求最大值　　max(表达式1, …, 表达式n);

求最大值　　min(表达式1, …, 表达式n);

求绝对值　　abs(表达式);

例 1.10　要从 n 个整数元素中查找出最大值, 请采用上述 3 种方法进行算法描述。

解　(1) 流程图描述如图 1.7 所示。

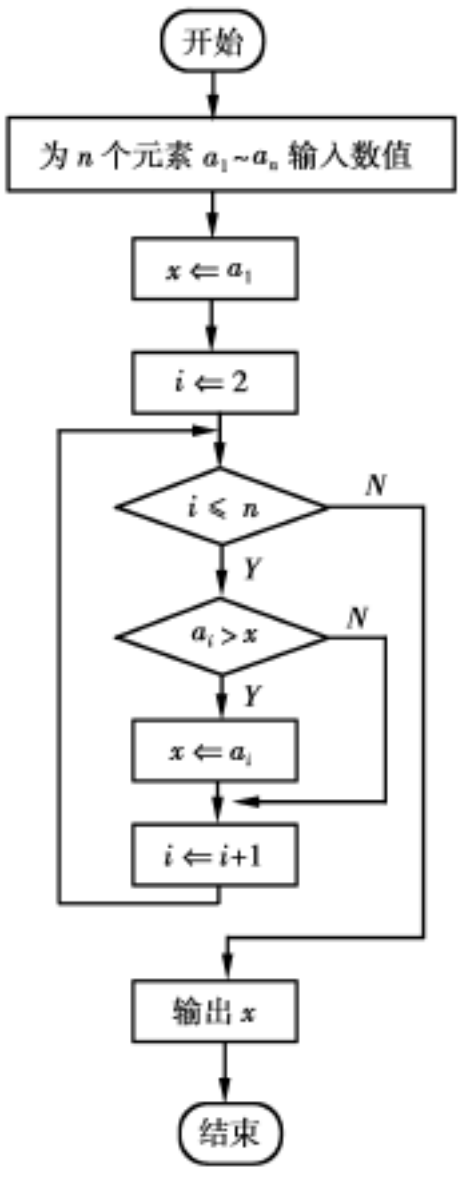

图 1.7　求 n 个元素的最大值

(2) 采用自然语言描述, 步骤如下:

1) 给 n 个整数元素 a_1 ~a_n 输入数值;

2) 把第一个元素 a_1 赋给用于保存最大值元素的变量 x;

3) 把表示下标的变量 i 赋初值 2;

4) 如果 i≤n 则向下执行, 否则输出最大值 x 后结束算法;

5) 如果 a_i > x 则将 a_i 赋给 x, 否则不改变 x 的值, 这使得 x 始终保存着当前比较过的所有元素的最大值;

6) 使下标 i 增 1, 以表示下一个元素; 转向 4) 继续执行。

(3) 采用类 C 语言描述如下:

```
void FindMax( int a[ ], int n)
{  //用 a[0] ~a[n - 1] 保存元素 a1 ~an
    for( i = 0; i < n; i++ ) scanf( a[i] ) ;
    x = a[0] ;          //把第一个元素 a[0] 赋给 x
    i = 1;              //把第二个元素 a[1] 的下标 1 赋给 i
    while( i < n)
    {
        if( a[i] > x) x = a[i] ;
        i + + ;
    }
    printf ( "%d", x) ;
} //FindMax
```

注意, 要使一个算法能在计算机上运行, 最终必须采用一种程序设计语言进行描述。算法不是程序, 程序是用某种程序设计语言对算法的具体实现。尽管算法的含义与程序十分相似, 但二者是有区别的。主要区别在于有穷性和描述方法。

程序可以是无穷的(死循环), 算法是有穷的;

程序是用程序设计语言描述, 在机器上可以执行。算法还可以用框图、自然语言等方式描述。一个算法若用计算机语言来书写, 则它就可以是一个程序。

1.4.3　算法分析

算法是解决问题的一种方法或一个过程, 一个问题可以有多种算法, 如何从中找出最有效

的算法? 怎样判断一个算法的好坏? 通常是从以下几个方面评价算法的性能:

正确性: 算法应当满足具体问题的需求, 包括对输入、输出和加工处理等明确无歧义的描述。

可读性: 算法应当便于调试和修改, 易于人们理解、阅读与交流。

健壮性: 当环境发生变化(如遇到非法输入) 时, 算法能做出适当的反应或进行处理, 不会产生不需要的运行结果。

高效性: 对同一个问题, 执行时间短、所需存储空间少的算法效率较高。

在数据结构课程中侧重于对高效性的分析, 时间和空间效率的巨大改进源于更好的数据结构或算法。由于算法的执行时间和所需的存储空间往往是相矛盾的, 两者难以兼得, 但现在硬件的价格越来越便宜, 使得存储空间的花费不断下降, 因此, 通常是用算法的执行时间为算法效率的主要衡量指标。如何确定一个算法的时空性能, 这项工作称为“ 算法分析 ”。

(1) 语句频度

一个算法的执行时间等于其所有语句执行时间之总和, 而任一语句的执行时间为该语句的执行次数与执行一次所需时间的乘积。要想精确地确定各种语句执行一次所需要的时间是十分困难的, 从理论上是不能算出来的, 必须上机运行测试才能知道。但不可能也没有必要对每个算法都上机测试, 只需知道哪个算法花费的时间多, 哪个算法花费的时间少就可以了。由于一个算法花费的时间与算法中语句的执行次数成正比例, 所以哪个算法中语句执行次数多, 它花费时间就多。

将算法中某个语句可能重复执行的最大次数称为该语句的语句频度。一个算法中所有语句的语句频度之和构成了该算法的运行时间, 也称为算法的语句频度, 记为 T(n) 。

例 1. 11　求下列语句①的语句频度

```
for( i = 1; i < = n; i ++ )
    for( j = 1; j < = i ; j ++ )
        x = x + 1;                    ①
```

解　该算法为一个二重循环, 语句①的执行次数为内、外循环次数相乘, 但内循环次数不固定, 与外循环有关, 因此, 语句①的语句频度为 $1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$

(2) 时间复杂度

在算法的语句频度 T(n) 中, n 称为问题的规模, 当 n 不断变化时, 语句频度也会不断变化。但要知道它变化时呈现什么规律, 在此, 引入了时间复杂度的概念。

设算法的语句频度为 T(n) , 函数 f(n) 是当 n 趋向无穷大时与 T(n) 同数量级的函数, 把 T(n) 表示成数量级的形式为: T(n) = O(f(n)) , 则称 O(f(n)) 为算法的时间复杂度。其中, 大写字母 O 为英文 Order(即数量级) 一词的第一个字母; n 为算法计算量或称为规模; f(n) 是运算时间随 n 增大时的增长率。

算法的时间复杂度不仅与问题规模有关, 还与输入实例的初态有关。但是我们总是考虑在最坏的情况下(在所有输入下的计算量的最大值) 的时间复杂度, 以保证算法的运行时间不会比它更长。

例 1. 12　在数组 a[1. . n] 中查找给定值 d 的算法中, 求语句①的语句频度及算法的时间复杂度。