



计算机等级(水平)考试系列教材

计算机软件技术基础

孙家启 周鸣争 杨明 郭兴众 李明

安徽大学出版社

计算机等级(水平)考试系列教材

高级顾问

计算机软件技术基础

孙家启 周鸣争 杨明
郭兴众 李明

编著

861-863

主任:孙家启

委员:(按姓氏笔画)王忠仁 石

仲红 朱

吴国风 张国平

姚合生 聂会星 谢荣伟

图书技术部

图书技术部

孙家启 周鸣争 杨明 郭兴众 李明



编著

千814

幅印为1

西照

本开

1998年11月

册0002-1 覆印

封面设计 封面设计

安徽大学出版社

ISBN 7-81052-103-2/TP·14 定价 21.00元

安徽省出版集团出版, 安徽出版集团印刷

内容简介

本书选择数据结构、操作系统、数据库系统、软件工程和计算机网络这五门计算机软件课程内容作为主要内容,从计算机应用角度出发系统地、深入浅出地介绍了软件技术的基础知识。

本书兼顾全国高等学校计算机考试需要,其内容覆盖了四级偏软(三级B)考试范围的主体部分,所以它又是一本考试指导书——计算机等级(水平)考试系列教材之一。

本书除适用于高等学校非计算机专业本科生、研究生、计算机辅修专业学生外,也可以作为高等专科学校、职工大学、成人教育计算机应用专业的教学用书,同时还可供从事计算机应用的广大工程技术人员参考。

计算机软件技术基础

孙家启 周鸣争 杨明 郭兴众 李明 编著

安徽大学出版社出版发行

(合肥市肥西路3号 邮码 230039)

肥西县印刷有限责任公司印刷 新华书店经销

开本 787×1092 1/16 印张 17 字数 416 千

1998年11月第1版 1998年11月第1次印刷

印数 1—5000 册

责任编辑 陈琦 封面设计 孟献辉

ISBN 7-81052-163-2/TP·14 定价 21.00 元

如有印装质量问题,请与出版社联系调换

高级顾问

韦 穗 杨善林 陈国良

编委会名单

主 任:孙家启

委 员:(按姓氏笔划)王忠仁 石 竹 孙家启

仲 红 朱 武 朱学勤

吴国风 张国平 周鸣争

姚合生 聂会星 谢荣传

秘书长:聂会星

编写说明

为了支持计算机基础教育改革与建设,促进计算机基础课程教学与水平考试向纵深发展,我们组织编写了计算机等级(水平)考试系列教材。这套教材囊括了计算机入门知识、高级语言程序设计、计算机软件技术基础、微型计算机原理及应用等方面内容,涵盖计算机水平考试的一、二、四级(全国等级考试的一、二、三级),因而具有广泛的适应性。这套教材所具有的突出特点是:紧扣计算机基础教育大纲(即计算机水平考试大纲),兼具普通教材与考试辅导材料的双重功能;立意创新,内容简炼,大量针对性极强的习题和典型例题分析是其它教材所少见;编写人员都是教学、科研第一线有着丰富教学与实践经验的教师,他们深谙相关知识点的张弛取舍。我们还聘请了三位知名专家担任高级顾问,这诚然为本系列教材添色增辉。

本系列教材的先期版本现已问世,第一辑各册预计在1998年底全部出齐。由于计算机技术的发展比人们想象的还要快,所以我们今后还将不断调整教材内容使之与当时发展相适应,以便教材以更新更好的面目呈现在读者面前。

本系列教材编写目的明确,它特别适合于作为普通高校非计算机专业的本、专科教学用教材或成教、夜大、函大计算机专业的教材,也可供各地计算机水平考试考点使用,还可供广大计算机自学者、工程技术人员参考。

编写委员会

一九九七年十月

孙家庆 周

安徽

(合肥)

肥西县印刷

开本 787×

1998年11月

责任编辑 陈 峰 封面设计 王淑珍

ISBN 7-81052-163-2/TP·14 定价 21.00元

如有印装质量问题,请与出版社联系调换

前言

随着计算机技术的发展和普及,各行各业迫切需要大量具有专业知识和计算机应用知识的较高水平的复合型人才。高等学校非计算机专业本科生、研究生仅仅掌握计算机应用基础知识和一、二门程序设计语言,毕业后已不能适应这一形势。这就要求他们能够比较系统地,着重应用能力的提高,学习计算机软件技术基础的知识,为今后上岗从事计算机应用打下良好的基础。

本书正是为了顺应这一发展形势而选择了数据结构、操作系统、数据库系统、软件工程和计算机网络五个部分。为了使非计算机专业本科生、研究生在有限的学时内系统地学到其中最重要、最基本、最实用部分,其内容组织上从计算机应用角度出发,将各有关篇章中要求计算机专业知识较多且难于理解的有关内容删除了,而保留了各篇章的系统性和主要精髓。本书兼顾全国高等学校计算机考试需要,其内容覆盖了四级偏软(三级B)考试范围主体部分,所以它又是一本考试指导用书——计算机等级(水平)考试系列教材之一。

本书共分五篇。第一篇由李明编写,第二篇由周鸣争编写,第三篇由郭兴众编写,第四篇由杨明编写,第五篇由孙家启编写。全书由孙家启修改定稿。编写过程中,得到了本系列教材高级顾问们的悉心指导,参阅了国内外有关著作,安徽大学出版社为本书出版付出了极大努力,编者在此一并表示衷心的感谢。

由于水平有限,加之时间仓促,错误与疏漏之处难免,恳请读者批评指正。

编者

一九九八年十月

目 次

第一篇 数据结构

第一章 概论	(1)
1.1 数据结构的基本概念	(1)
1.2 算法描述和算法分析	(2)
第二章 线性表	(5)
2.1 线性表及其基本操作	(5)
2.2 线性表的存储结构	(5)
2.3 栈	(10)
2.4 队列	(13)
2.5 串	(15)
第三章 数组和广义表	(16)
3.1 数组的概念	(16)
3.2 矩阵的压缩存储	(17)
3.3 广义表的概念	(20)
第四章 树和二叉树	(22)
4.1 树的概念	(22)
4.2 二叉树及其性质	(23)
4.3 二叉树的存储结构	(24)
4.4 二叉树的遍历和线索	(25)
4.5 树和森林	(27)
4.6 哈夫曼树及其应用	(29)
第五章 图	(32)
5.1 图的概念	(32)
5.2 图的存储结构	(34)
5.3 图的遍历	(36)
5.4 图的应用	(37)
第六章 查找	(41)
6.1 基本概念	(41)
6.2 线性表的查找	(41)
6.3 树表的查找	(44)
第七章 排序	(47)
7.1 基本概念	(47)

7.2	插入排序	(47)
7.3	交换排序	(49)
7.4	选择排序	(51)
7.5	归并排序	(51)
	第一篇习题	(52)
第二篇 操作系统		
	第一章 绪论	(56)
1.1	什么是操作系统	(56)
1.2	操作系统类型	(56)
1.3	操作系统的功能与特性	(58)
	第二章 CPU 管理	(60)
2.1	作业的管理	(60)
2.2	进程管理	(64)
	第三章 存储器管理	(80)
3.1	存储器管理的目的与功能	(80)
3.2	单一连续区管理	(81)
3.3	分区管理	(81)
3.4	段式与段页式管理	(87)
3.5	覆盖与交换技术	(89)
3.6	虚拟存储器	(90)
	第四章 文件管理	(91)
4.1	基本概念	(91)
4.2	文件目录管理	(92)
4.3	文件的结构	(94)
4.4	文件的存取与保护	(95)
4.5	文件的使用	(98)
	第五章 设备管理	(100)
5.1	引言	(100)
5.2	数据传送控制方式	(101)
5.3	设备分配	(102)
5.4	设备驱动程序	(105)
	第二篇习题	(105)
第三篇 数据库系统		
	第一章 概述	(107)
1.1	发展概况	(107)
1.2	数据库系统及其特点	(108)
1.3	数据模型	(109)
1.4	数据库系统的结构	(113)
	第二章 关系数据库	(116)

(201)	2.1	关系模型的基本概念	(116)
(791)	2.2	关系代数	(119)
(801)	2.3	关系演算	(123)
(991)	2.4	关系规范化理论	(125)
(999)	第三章	关系数据库标准语言(SQL语言)	(132)
(991)	3.1	SQL语言特点	(132)
(005)	3.2	SQL数据定义	(133)
(105)	3.3	SQL数据操纵	(136)
(E05)	3.4	SQL数据控制	(143)
(503)	第四章	数据库设计	(145)
(506)	4.1	数据库设计的任务、方法与步骤	(145)
(705)	4.2	需求分析	(146)
(705)	4.3	概念结构设计	(147)
	4.4	逻辑结构设计	(151)
(905)	4.5	物理结构设计	(152)
(905)	第五章	数据库安全与保护	(153)
(515)	5.1	数据库的安全性和完整性	(153)
(215)	5.2	并发控制的基本概念	(154)
(215)	5.3	数据库恢复	(155)
(155)		第三篇习题	(156)
	第四篇	软件工程	
(555)	第一章	软件工程概论	(158)
(559)	1.1	软件危机	(158)
(555)	1.2	软件工程	(162)
(557)	第二章	软件计划	(164)
(755)	2.1	软件计划内容	(164)
(855)	2.2	可行性研究	(165)
(045)	第三章	软件需求分析	(167)
(545)	3.1	目标、任务和步骤	(167)
(745)	3.2	数据流分析	(168)
(523)	3.3	原型法	(172)
(825)	第四章	软件设计	(175)
(425)	4.1	软件设计过程	(175)
(524)	4.2	软件设计原理	(177)
(525)	4.3	结构化设计技术	(179)
(065)	第五章	编码	(189)
	5.1	程序设计语言	(189)
	5.2	程序设计途径	(192)
	第六章	软件测试	(196)

6.1	软件测试的目的与步骤	(196)
6.2	软件测试的方法	(197)
6.3	测试用例的设计	(198)
第七章	软件维护	(199)
7.1	软件维护	(199)
7.2	维护过程	(199)
7.3	可维护性	(200)
7.4	软件维护的管理	(201)
第八章	软件工程的文档编制	(203)
8.1	软件文档的作用和分类	(203)
8.2	文档编制的质量要求	(206)
8.3	文档管理和维护	(207)
	第四篇习题	(207)
第五篇	计算机网络	
第一章	计算机网络概述	(209)
1.1	计算机网络的发展	(209)
1.2	计算机网络的分类与功能	(212)
第二章	局域网	(215)
2.1	数据通信基本概念	(215)
2.2	局域网的拓扑结构	(221)
2.3	网络协议和网络体系结构	(224)
2.4	局域网体系结构	(226)
2.5	介质访问控制方法(协议)	(229)
2.6	网络互连	(232)
第三章	Novell 网络简介	(237)
3.1	Novell Netware 发展及版本	(237)
3.2	Novell 网络的系统组成	(238)
3.3	网络规划	(240)
3.4	Novell 网络的基本操作	(244)
3.5	NetWare386 文件系统管理	(247)
第四章	校园网	(253)
4.1	因特网(Internet)简介	(253)
4.2	中国教育与科研网(CERNet)概况	(254)
4.3	国内校园网实例	(254)
4.4	网络服务	(257)
	第五篇习题	(260)

第一篇 数据结构

第一章 概 论

自 1946 年问世以来,计算机的软硬件技术得到了飞速发展,同时它的应用领域也在迅速扩大,其加工处理的对象从数值型数据发展到字符、表格、图像、声音等复杂的具有一定结构的数据。为了设计出更“好”的程序,不仅要掌握计算机语言,还要研究数据的特征以及它们之间存在的关系,这就是数据结构这门学科形成和发展的背景。

数据结构是随着计算机发展而逐渐形成的一门学科。

1.1 数据结构的基本概念

1. 数据(data)

是信息的载体,是对客观事物的符号表示,它能够被计算机识别、存储和加工处理。

2. 数据元素(data element)

是数据的基本单位,通常在计算机中作为一个整体进行加工处理,有时,一个数据元素由若干个数据项组成,数据项是具有独立含义的最小单位。

3. 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。

4. 数据类型(data type)

是一个值的集合和定义在这个值集上的一组操作的总称,它可以分为原子数据和结构数据。数据处理(data processing)是对数据进行查找、插入、删除、合并、排序等的操作过程。

5. 数据结构(data structure)

是数据之间的相互关系,即数据的组织形式,它包括三方面的内容:

- (1)数据之间的逻辑关系,即数据的逻辑结构(logical structure);
- (2)数据元素及其关系在计算机中的存储方式,即数据的存储结构(storage structure);
- (3)数据的运算,即对数据施加的操作。

数据的逻辑结构是从逻辑关系上描述数据,是从具体问题抽象出来的数学模型,它独立于计算机;数据的存储结构是逻辑结构在计算机中的存储映象,它依赖于计算机语言,存储结构是具体的。数据的运算是定义在数据的逻辑结构上的,因而每种逻辑结构都有一个运算的集合,这些运算是在数据上所施加的一系列抽象操作,只知道“做什么”,而“怎么做”则建立在存储结构上。

综上所述,数据结构是研究程序设计出现的操作对象和它们之间的关系以及建立在

其上的各种操作的集合。

数据的逻辑结构有两大类：

(1)线性结构:其特征是有且仅有一个开始结点和终端结点,所有结点最多只有一个直接前驱和直接后继。

(2)非线性结构:其特征是一个结点可能有多个直接前驱和直接后继。

数据的存储结构常用的有两大类:

(1)顺序存储结构:其特征是将逻辑上相邻的结点,存储在物理位置也相邻的存储单元里,即逻辑相邻和物理相邻是吻合的。

(2)链式存储结构:其特征是逻辑相邻的结点不一定存储在相邻的存储单元中,通常用指针来描述并实现。

1.2 算法描述和算法分析

1.2.1 算法及其特征

算法是一种解题方法,是由若干条指令组成的有穷序列,是对特定问题求解步骤的一种描述,应具备以下五大特征:

(1)有穷性:算法应在执行有穷步之后结束。

(2)确定性:算法中的每一条指令必须有明确的含义。

(3)可行性:算法是可以实现的。

(4)输入:算法应有零个以上的输入。

(5)输出:算法至少要有有一个以上的输出。

1.2.2 算法描述

算法与程序是有区别的,它不要求用严格的计算机语言加以描述,它可以用自然语言、数学语言或计算机语言加以描述。

本篇采用类C语言加以描述,以下对其作简单说明:

1.赋值语句

变量 = 表达式;

变量 ++;

变量 --;

变量名 = 条件表达式? 表达式 T1; 表达式 T2;

2.条件语句

(1)if(表达式)语句;

(2)if(表达式)语句 1;

else 语句 2;

3.循环语句

(1)while(表达式)

{循环体语句;

(2)do{

循环体语句;

}while(表达式);

(3)for(表达式 1;表达式 2;表达式 3)

{循环体语句;

}

4. 开关语句

(1)switch(表达式){

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;

}

(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;

}

5. 输入输出语句

(1)scanf(格式串,变量 1,...,变量 n);

(2)printf(格式串,表达式 1,...,表达式 n);

6. 基本函数

(1)求最大值 max(表达式 1,...,表达式 n)

(2)求最小值 min(表达式 1,...,表达式 n)

(3)求绝对值 abs(表达式)

(4)判定文件结束 eof(文件变量)

(5)判定行结束 eoln(文件变量)

7. 函数形式

函数类型 函数名(函数参数表){

数据说明;

执行语句;

}

8. 结束语句

(1) return 函数结束

(2) break 结束

(3) typed exit datatype; 异常结束

9. 注释

/* 字符串 */

1.2.3 算法分析

1. 算法设计要求

求解一个问题,可以有许多不同的算法,而一个好的算法应达到以下目标:

(1)正确性:指在合理的输入数据下,在有限的运行时间内得出正确的结果。

(2)可读性:现代程序要求算法应具备良好的可读性,以便于人们理解和维护软件。

(3)健壮性:当算法有非法数据入侵时,能进行相应处理。

(4)时间和空间需求:执行时间短的算法效率高,好的算法则应追求高效和低存储空间,而这二者均与问题规模有关。如 100 个数的累加和 1000 个数的累加其时空特性是有差别的。

2. 算法效率度量

分析算法效率,目的在于通过对算法执行时间和占用空间的讨论,寻求改进算法的途径,以确定一个适合具体问题的最优算法。

一个算法所耗费的时间是算法中每条语句执行时间之和,而每条语句的执行时间为该语句的执行次数(也称频度)与该语句执行一次所需时间之乘积,由于语句执行一次所需时间与机器等因素有关,本篇只讨论语句执行的次数,选中算法中的某种基本操作,将其执行次数的多少作为算法的时间度量,称作算法的时间复杂度,记作 $T(n)$,其中 n 为问题的规模。

例:(1) $a=0$;

(2)for($i=1$; $i \leq n$; $i++$)

$s=s+x$;

(3)for($i=1$; $i \leq n$; $i++$)

for($j=1$; $j \leq n$; $j++$)

$s=s+x$;

它们的时间复杂度分别为 $O(1)$, $O(n)$, $O(n^2)$,即常量阶,线性阶,平方阶。

与时间复杂度类似,算法的空间复杂度为该算法所耗费的空间,它也与问题规模有关,记作 $S(n)$ 。

第二章 线性表

2.1 线性表及其基本操作

2.1.1 线性表的定义

线性表(linear list)是具有相同类型元素的有限序列,即由 $n(n \geq 0)$ 个数据元素 $a_1, a_2, \dots, a_n$ 组成的有限序列,其一般表示为 $(a_1, a_2, \dots, a_n)$ 。

其中, n 为线性表的长度,当 $n=0$ 时,则为空表; a_1 为表头; a_n 为表尾, a_{i-1} 为 a_i 的直接前驱, a_i 为 a_{i-1} 的直接后继。

例: $(A, B, C, D, \dots, X, Y, Z)$ 是由 26 个字母组成的线性表。

由此可知,一个非空线性表满足以下几个逻辑特征:

- (1) 有且仅有一个开始结点 a_1 , 无直接前驱。
- (2) 有且仅有一个终端结点 a_n , 无直接后继。
- (3) 内部结点 $a_2 \sim a_n$ 有且仅有一个直接前驱。
- (4) 内部结点 $a_1 \sim a_{n-1}$ 有且仅有一个直接后继。

2.1.2 线性表的基本运算

线性表的常见运算有:

- (1) $\text{setnull}(L)$ 置线性表 L 为空表。
- (2) $\text{length}(L)$ 求线性表的长度。
- (3) $\text{get}(L, i)$ 取线性表中第 i 个元素, $1 \leq i \leq N$ 。
- (4) $\text{locate}(L, x)$ 确定元素 x 在线性表中的位置,若不存在,结果为零。
- (5) $\text{insert}(L, x, i)$ 将元素 x 插在第 i 个元素位置上, $1 \leq i \leq N+1$ 。
- (6) $\text{delete}(L, i)$ 将线性表中第 i 个元素删除, $1 \leq i \leq N$ 。

2.2 线性表的存储结构

2.2.1 线性表的顺序存储结构及其基本运算

1. 线性表的顺序存储

将线性表中的元素按逻辑顺序依次存放在一片地址连续的存储单元中,即为顺序表,如图 1-2-1 所示,顺序表常用数组类型来表示,其描述形式如下:

```
typedef int datatype;
```

```
# define max 100;
```

```
typedef struct
```

```
{datatype data[max];
```

```
int last;
```

```
}slist;
```

设线性表中每个元素占用1个存储单元,用 $Loc(a_i)$ 表示第 i 个元素的存储位置,则有如下关系:

$$Loc(a_{i+1}) = Loc(a_1) + (i - 1) * l$$

其中 $Loc(a_1)$ 为第一个元素的地址,即基地址。

2. 顺序表上的基本运算

(1) 插入 $insert(L, x, i)$

指在顺序表的第 i 个 ($1 \leq i \leq N+1$) 位置上插入新元素 x , 则线性表: $(a_1, a_2, \dots, a_{i-1}, a_i, \dots, a_n)$ 变为线性表 $(a_1, \dots, a_{i-1}, x, a_i, \dots, a_n)$, 表长由 N 增至 $N+1$, 如图 1-2-2 所示。

由图 1-2-2 可见,若要在顺序表上插入元素 x , 首先将元素 a_n 至 a_i 从后向前依次顺序后移一个标准位置, 然后才可插入, 其算法如下:

```
int INSERT(L, x, i)
```

```
slist * L;
```

```
int i;
```

```
{ int j;
```

```
if((( * L).last) >= max - 1)
```

```
{ printf("error");
```

```
return NULL; } /* 表空间满 */
```

```
else
```

```
if((i < 1) || (i > (( * L).last) + 1))
```

```
{ printf("error");
```

```
return NULL; } /* 插入位置非法 */
```

```
else
```

```
{ for(j = ( * L).last; j >= i - 1; j--)
```

```
( * L).data[j + 1] = ( * L).data[j]; /* 元素后移 */
```

```
( * L).data[i - 1] = x; /* 插入 */
```

```
( * L).last = ( * L).last + 1; /* 修正表长 */
```

```
}
```

```
return(1);
```

```
}
```

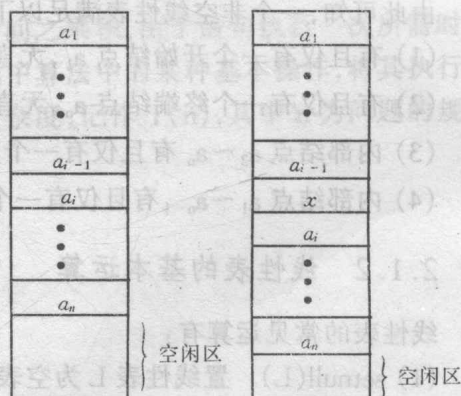


图 1-2-2 顺序表的插入

此算法中基本操作是将元素依次向后搬移, 插入位置越前, 移动元素越多, 最多移动 n 次, 则此算法时间复杂度近似为 $O(n)$, 即线性阶。

(2) 删除 $delete(L, i)$

将顺序表中第 i 个位置上的元素删除, 则线性表 $(a_1, \dots, a_{i-1}, a_i, \dots, a_n)$ 变成线性表 $(a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n)$

$\dots, a_{i-1}, a_{i+1}, \dots, a_n$), 表长由 N 减少到 $N-1$, 如图 1-2-3 所示。

由图 1-2-3 可见, 若要从顺序表中删除第 i 个元素, 须将元素 a_{i+1} 至 a_n 从前向后依次前移一个标准位置, 其算法如下:

```
int DELETE(L, i)
slist * L;
int i
{int j;
  if ((i < 1) || (i > (*L).last + 1))
  {printf("error");
   return null;} /* 删除位置非法 */
  else
  {for(j = i; j <= (*L).last; j++)
    (*L).data[j - 1] = (*L).data[j];
    (*L).Last = (*L).Last - 1;
  }
  return(1);
}
```

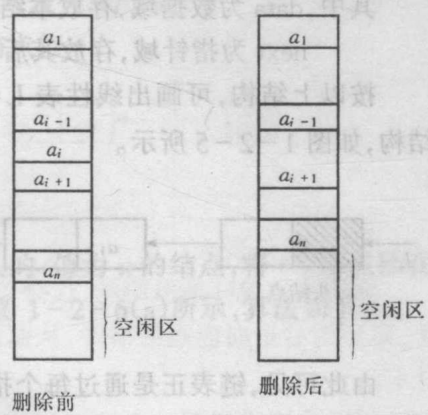


图 1-2-3 顺序表的删除

此算法中基本操作是将元素依次向前移动, 删除位置越前, 移动元素越多, 最多移动 $n-1$ 次, 则此算法的时间复杂度与插入算法相同, 近似为 $O(n)$, 也为线性阶。

2.2.2 线性表的链式存储结构及其基本运算

1. 线性表的链式存储的特点

线性表的顺序存储是用物理位置上的相邻来代表逻辑之间的相邻, 因而它有如下特点:

- (1) 不需要为表示元素之间的逻辑关系而增加额外空间。
- (2) 利用数组可以方便和随机存取线性表中任一元素。
- (3) 插入和删除须移动大量元素。
- (4) 对空间的要求较高, 它需要一片连续的空间, 空间利用率较低。
- (5) 表的扩充受到一定限制。

线性表的链式存储中, 各元素的物理位置不一定相邻。它们可以分散存储在内存中的各个地方, 它们之间通过一系列指针将它们链接在一起, 从而组成了一个链表, 克服了顺序表的缺点。

链表的特点如下:

- (1) 必须为表示元素之间的逻辑关系附设指针, 从而增加空间开销。
- (2) 存取链表中的某一元素比顺序表复杂。
- (3) 插入和删除无须移动元素。
- (4) 不需要连续的空间, 对空间的要求低, 空间的利用率较高。
- (5) 链表的扩充基本不受限制。

2. 单链表及其运算

(1) 单链表