# 第3章 表、栈和队列

## 1 抽象数据类型（ADT）

是一些操作的集合（并、交、查找、测定大小、取余），是数学的抽象、模块化设计的扩充。

## 2.表ADT

大小为0的是空表。有操作：Empty、PrintList、MakeEmpty、Insert、Delete、FindKth（返回k位置的元素）、Find（返回关键词首次出现的位置）、Next、Precious（上一个）。

### 2.1表的简单数组实现

连续存储，数组的大小估计得大一些，表的大小必须事先已知。Find、PrintList、Insert、Delete需要线性时间，FindKth需要常数时间。

### 2.2链表

不连续存储，最后一个单元的Next指针指向NULL（ASCI C规定NULL为0），第0个单元的元素为空，只存指针，作为表头/哑节点。因为：①不存在从所给定义出发在表的前面插入元素的真正显性的方法，②从表的前面实行删除时一个特殊情况。

Find、PrintList、FindKth需要线性时间，Delete、Insert需要常数时间。Insert需要一次malloc 命令调用从系统得到的一个新单元，并执行两次指针调整。Delete需要free命令释放动态空间，执行一次指针调整。

Find：while(p!=NULL && p->Next->Element != X)

p = p->Next;

### 2.3常见错误

1.变量初始化失败：无论何时只要确定一个指向，那么必须保证该指针不是NULL，比如：p->Next。P不可以为NULL，p->next可以为NULL。

2.何时使用或何时不使用malloc来获取一个新的单元:malloc需要包含stdlib.h。如果从未对一个链表进行过删除操作，那么调用malloc的次数应该等于表的大学，有表头再加1。Malloc(sizeof(ptrToNode))是合法的，但是它并不给结构体分配足够的空间，只给指针分配一个空间。

3.当空间不需要时，可以使用free命令通知系统来收回它，free(p)收回:p在指向的地址没变。

### 2.4双（向）链表

增加了空间需求，使得插入和删除的开销增加一倍，因为有更多的指针需要定位。简化了删除操作，因为不再被迫使用一个指向前驱元的指针来访问一个关键字。

### 2.5循环链表

可以有表头，也可以无表头。

### 2.6例子

#### 2.6.1一元多项式ADT

1.使用结构体

typedef struct{

int CoeffArray[MaxDegree+ 1];//系数数组，MaxDegree是最大系数

int HighPower;//最高幂次

}\*Polynomial;

2.使用单链表:以次数递减的顺序排序

typedef struct Node{

int Coefficient;//系数

int Exponent;//指数

PtrToNode Next;

] Node \*PtrToNode

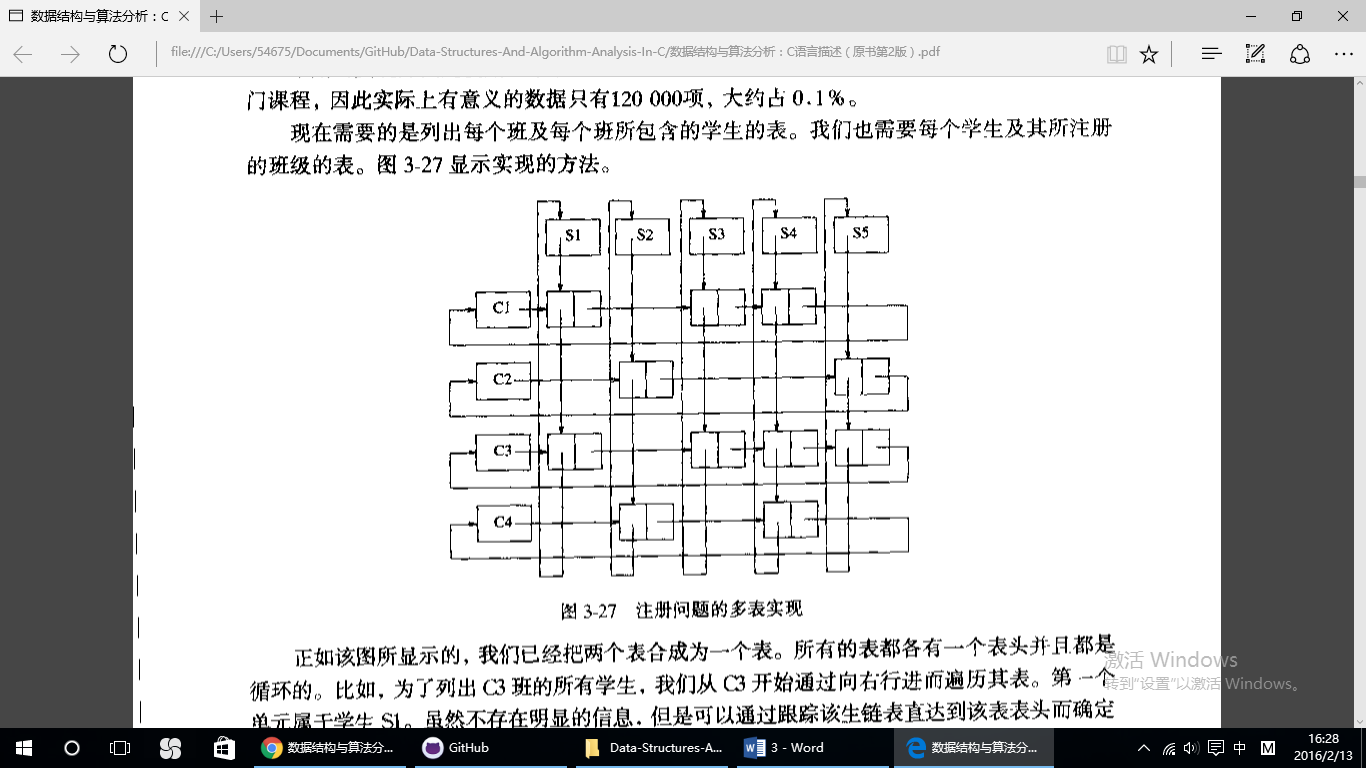
#### 2.6.2基数排序/卡式排序

桶式排序：对N个整数排序，范围从1-M。设置一个数组Count[M]，初始化为0，Count有M个桶。当Ai被读入时Count[Ai]增加1.最后扫描数组打印出排好序的表。花费O(M+N)。

基数排序是桶式排序的推广，有的数范围太大，桶的数量太多。采用多趟桶式排序，使用最低有效位优先的方式进行排序，每一位排序完后重新串联、对下一位进行排序。唯一可能出错的是两个数如果出自同一个桶但顺序却是错误的，不过前面各趟排序保证了当几个数进入一个桶的时候，它们是以顺序的顺序进入的。话费O(P(N+M))，P是排序的趟数。

#### 2.6.3多重表

假设：大学有40000名学生、2500门课程，需要生成2种类型报告：列出每个课程的学生、列出每个学生的课程。①使用二位数组，将有2500\*40000项。②我们使用多重表：二维表的实现，接近数组，但只留下了有用的项。课程从左往右遍历时，虽然不存在明显学生信息，但是可以通过跟踪该链表直达到该表表头而确定该生信息。



### 2.7链表的游标实现

需要链表又不能使用指针，将利用游标实现。指针链表的特性：①数组存储在一组结构体中，每隔结构体包含数据及指向下一个结构体的指针；②一个新的结构体可以通过调用malloc而从系统全局内存得到，并可通过free释放，游标指针也要符合这两个特性。

我们保留一个表（freelist），这个表由不在任何表中的单元构成，将用CursorSpace中的单元0作为表头。对于Next，0的值等价于NULL指针，CursorSpace初始化是一个简单的循环结构。为执行malloc功能，将在表头后面的第一个元素从freelist中删除；为了执行free功能，我们将该单元放在freelist的前端。从freelist删除的单元是刚刚由free放在那里的单元，最后被放在freelist的单元是被最先拿走的单元。

