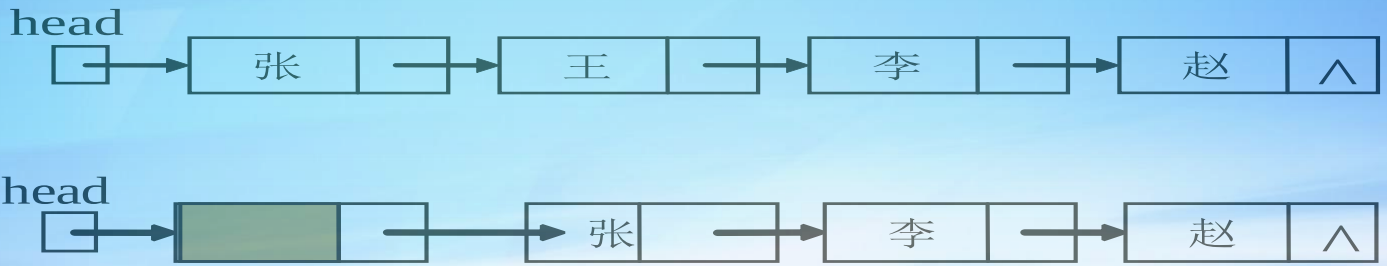
链表（上）

《数据结构》

网络工程教研中心　陈卫卫



稀疏多项式求和问题

- $P=8+10x^2+5x^{100}$
- $Q=2x^2+9x^{17}-5x^{100}+15x^{130}$
- P+Q=?

P

ceof	exp
8	0
10	2
5	100

Q

ceof	exp
2	2
9	17
-5	100
15	130

P+Q

ceof	exp
8	0
12	2
9	17
15	130

稀疏多项式求和问题

- 在原表P上进行稀疏多项式相加，若采用顺序存储，效率较低，如何改进算法？
- $P=8+10x^2+5x^{100}$
- $Q=2x^2+9x^{17}-5x^{100}+15x^{130}$

计算结果：P

ceof	exp
8	0
12	2
9	17
15	130

Q

ceof	exp
2	2
9	17
-5	100
15	130

学习目标和要求

1.准确地描述表的链式存储结构

2.写出单向链表的指定位置插入和删除的程序段

1. 表的链式存储—链表

- 链表是线性表的一种存储形式

链表的结点结构

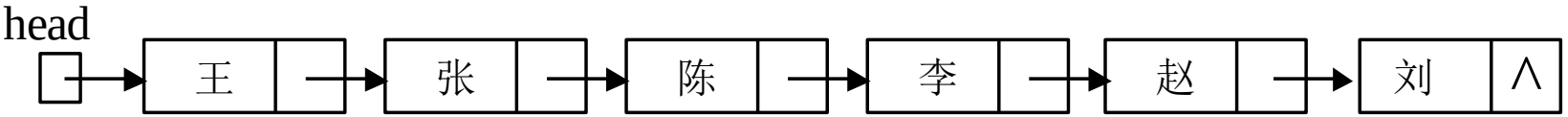
值域	链域

值域（数据域）：存储表元素值

链域（指针域）：存储后继结点的存储地址（指单向链表）

1. 表的链式存储—链表

链式存储结构图：



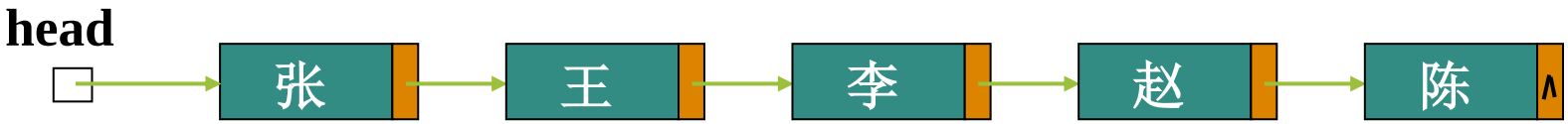
首指针（表头指针）：指向链表的第一个结点的指针变量。
其值为首结点的存储地址。

表尾结点（最后一个结点）的链域值为空（NULL）∧

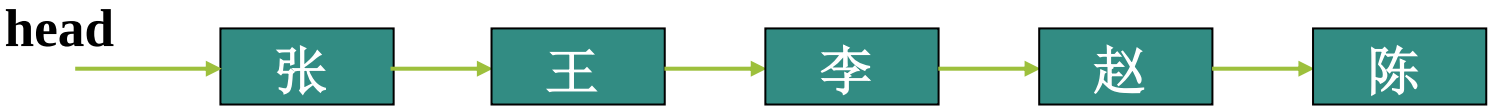
链表就是表头指针和一串相继链接的结点的总称

1. 表的链式存储—链表

链表的图示：



一般形式



简化形式

单向链表的结点结构的定义

链表的结点结构

值域	链域

```
typedef  struct  linkednode    //结点类型
{  int  data;                  //值域
   struct  linkednode  *next;  //链域
 }  snode, *ptr;          //结点类型名snode和指针类型名ptr
 ptr  head,p,q;                 // 定义指针类型变量
```

类型struct linkednode * 与类型ptr等价
都是指向snode的指针类型

2. 结点的产生和回收

编译预处理命令：#include <malloc.h>

通过调用文件malloc.h中的动态存储分配函数

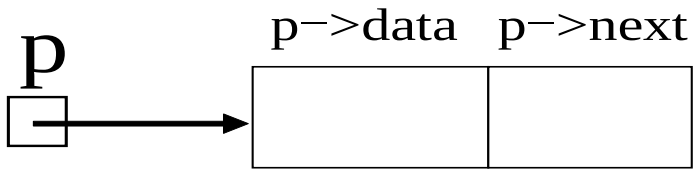
malloc产生结点（动态结点）

free回收结点

2. 结点的产生和回收

malloc()的典型用法：

p=(ptr)malloc(sizeof(snode));



含义：

产生一个结点——动态分配的变量

将结点的地址值转换成ptr类型

赋给指针变量p

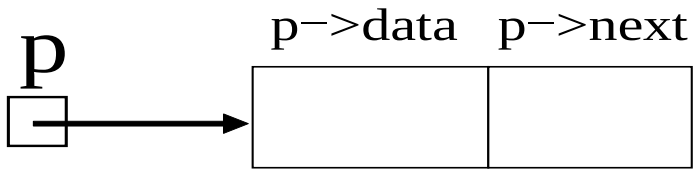
若分配失败，返回NULL

NULL是值为0的指针常量，无效地址

2. 结点的产生和回收

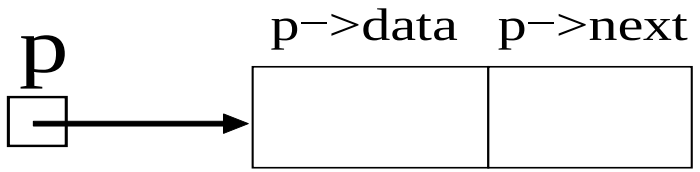
- **new的用法：**

p=new snode;



2. 结点的产生和回收

结点的引用：



p->data：结点的值域　等价于 (*p).data

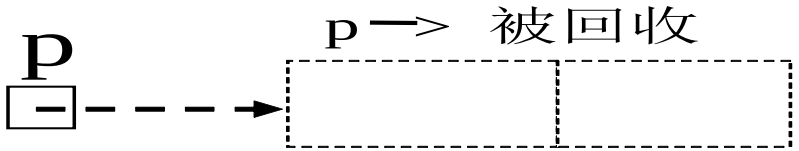
p->next：结点的链域　等价于 (*p).next

2. 结点的产生和回收

“废结点”的回收：

调用函数free(p)

例如：　　free(p)；



释放后，变量p->不复存在

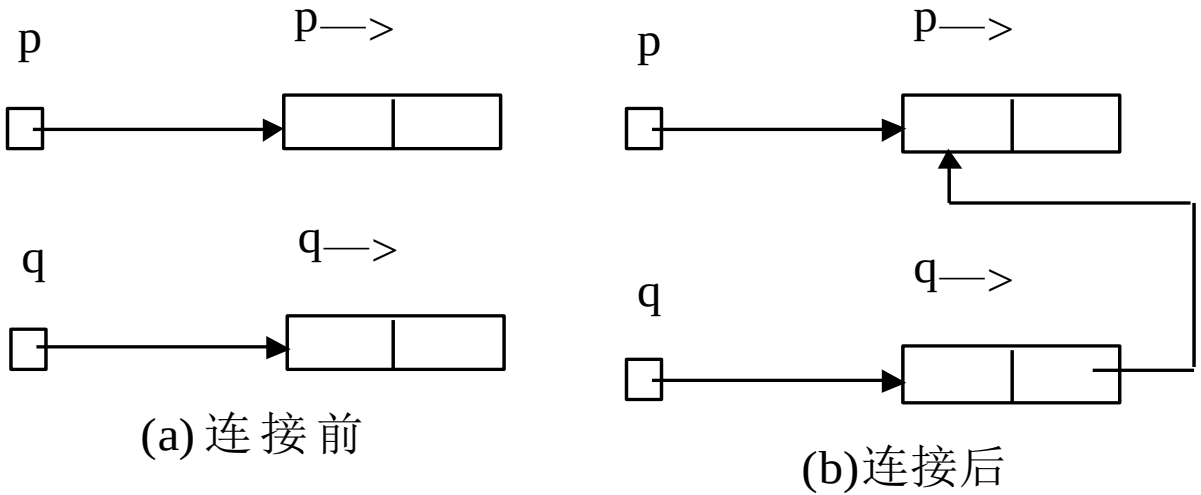
对p->的操作将变成“无意义”

```
delete p;
```

3. 结点的链接操作

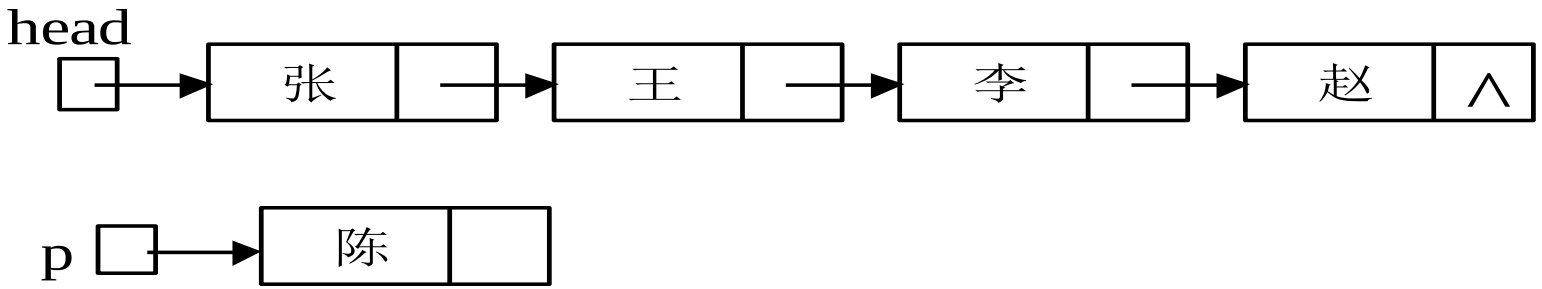
让前趋结点的链域指向后继结点

q->next=p;



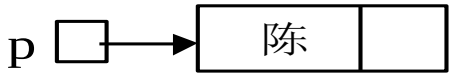
4. 指定位置的插入

基本操作：局部的修改结点的链域

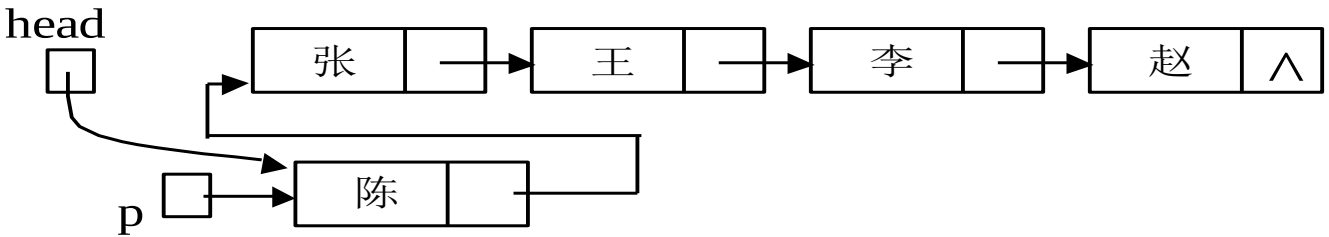


4. 指定位置的插入

（1）插在表头

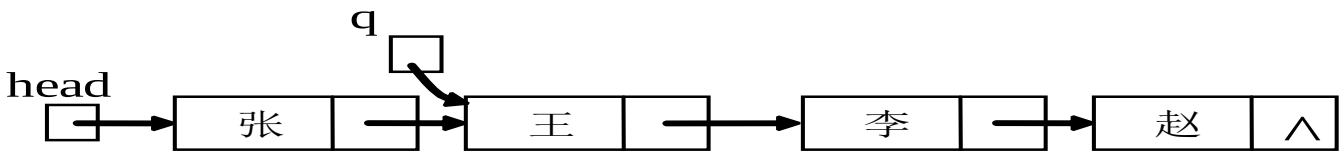



```
p->next=head;
head=p;
```

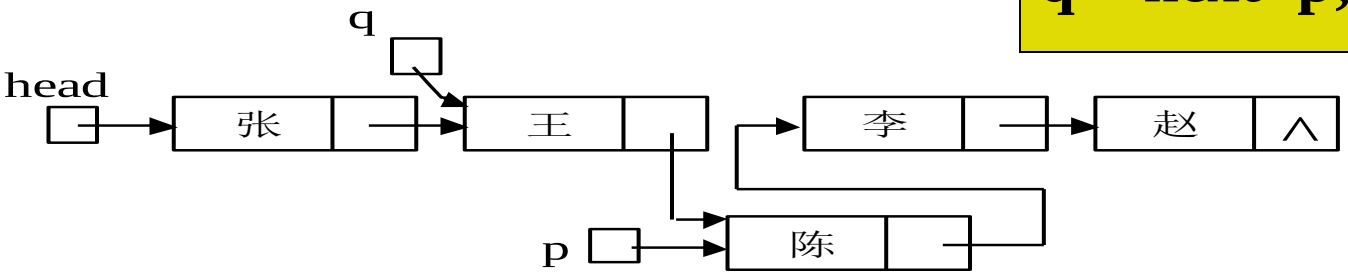


4. 指定位置的插入

（2）插在表中

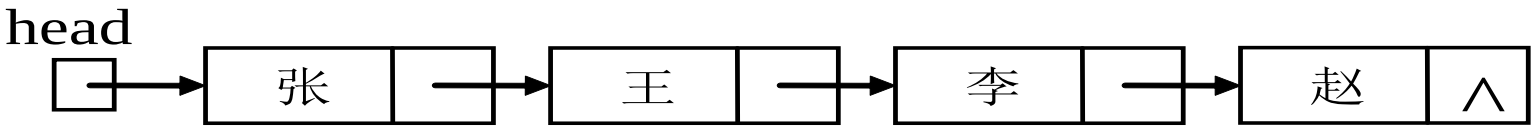



```
p->next=q->next;
q->next=p;
```



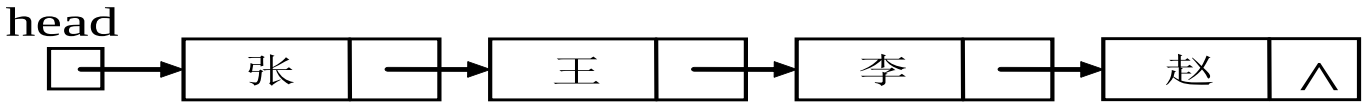
5. 指定位置的删除

基本操作：局部的修改结点的链域

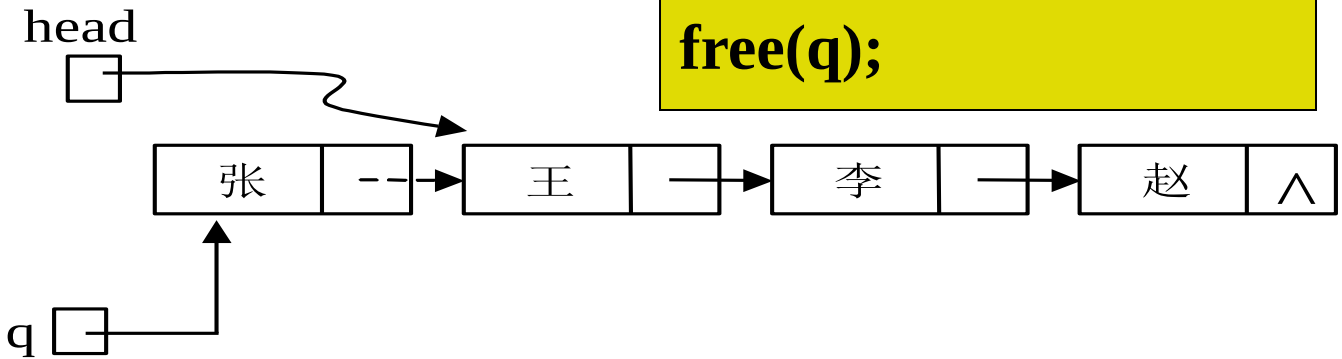


5. 指定位置的删除

（1）删除表头结点

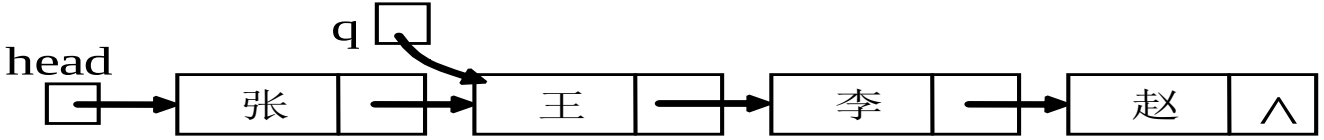



```
q=head;
head=head->next;
free(q);
```

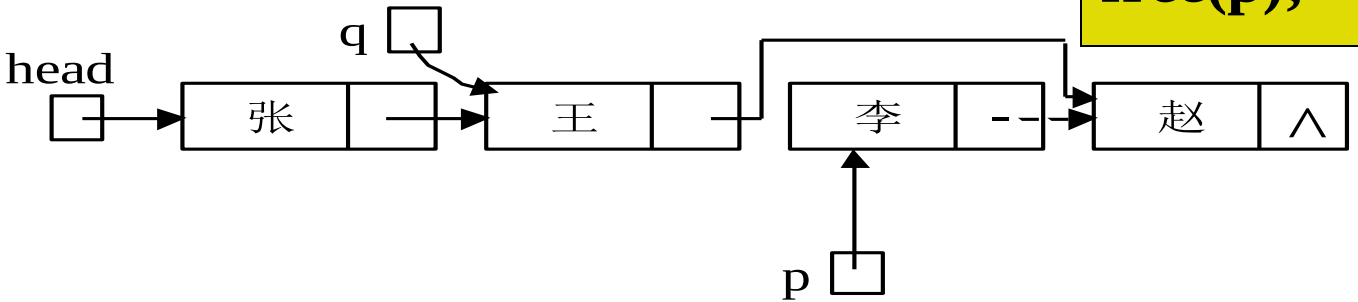


5. 指定位置的删除

（2）删除表中结点




```
p=q->next;
q->next=p->next;
free(p);
```



6. 链表的特点

特点：

（1）结点地址不连续

（2）插入/删除不移动结点，耗时为O(1)

（3）用于动态管理

核心：

（1）使用指向结点（结构类型）的指针

（2）执行期间，调用动态存储管理函数产生结点、回收结点