程序设计基础实验

The Basic Experiments of Programming Design

重庆工程学院通识学院

教 师： 王润生

2022, Spring

Outlines

二、链表创建

二、链表的遍历

三、链表的查询

四、链表的删除

链表创建

什么是链表？

线性表
- Array-Based List基于数组实现
- Linked List基于链接实现(链表)

A list is a finite, ordered sequence of data item

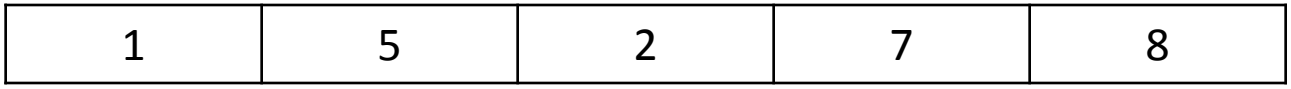
线性表是一个有限有序的数据序列

$<a_0, a_1, \ldots, a_{n-1}>$

链表创建

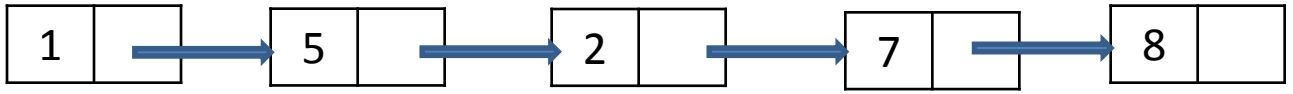
线性表<1,5,2,7,8>

Array-Based List数组实现



空间是连续的

Linked List链接实现（链表）

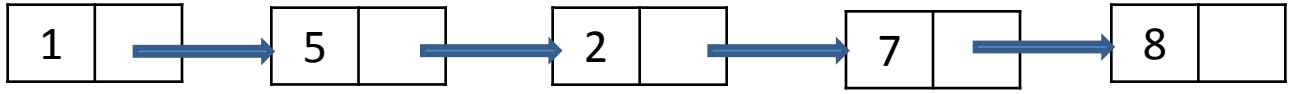


空间可以不连续

结点的值	下一个结点的地址

链表创建

链表中的结点



结点的值	下一个结点的地址

```
// 定义节点类型
struct Node
{
    // 结点的值
    int val;
    // 下一个结点的地址
    struct Node *next;
};
```

结点链接起来就是链表

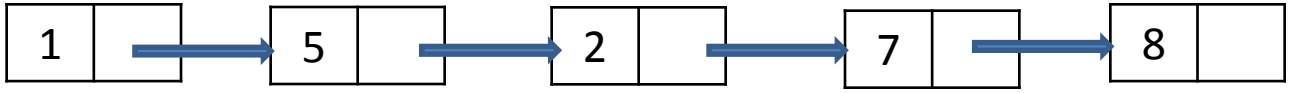
通过指针

即1结点里面存放5结点的地址

(1有5的地址，那我们是不是可以通过1去找到5，对吧？)

// 通常说1结点指向了5结点

链表创建




```c
#include<stdio.h>

struct Node // 定义一个Node类型
{
    int val; // 结点的值
    struct Node *next; // 下一个结点的地址
};

int main()
{
    struct Node n1,n2,n5,n7,n8; // 使用Node类型创建5个结点

    n1.val=1; // 结点值的初始化
    n2.val=2;
    n5.val=5;
    n7.val=7;
    n8.val=8;

    //链接 把n2的地址放在n1里面
    n1.next=&n2; // n1->n2
    n2.next=&n5; // n1->n2->n5
    n5.next=&n7; // n1->n2->n5->n7
    n7.next=&n8; // n1->n2->n5->n7->n8
    n8.next=NULL;// n1->n2->n5->n7->n8->NULL

    return 0;
}
```

链接

结点链接起来就是链表

Outlines

一、链表创建

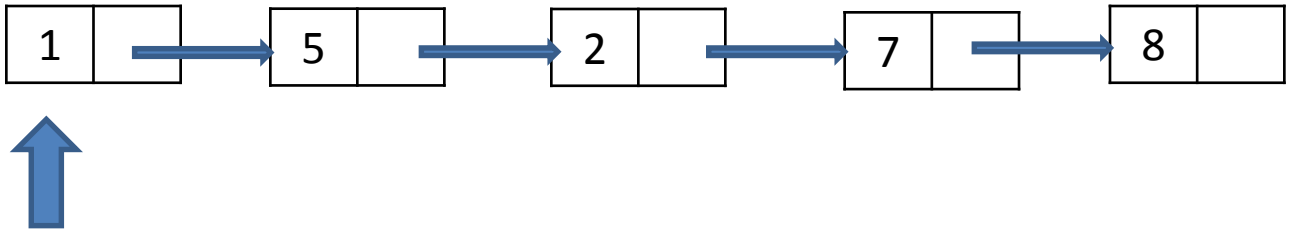
二、链表的遍历

三、链表的查询

四、链表的删除

链表遍历

从头到尾打印结点的值



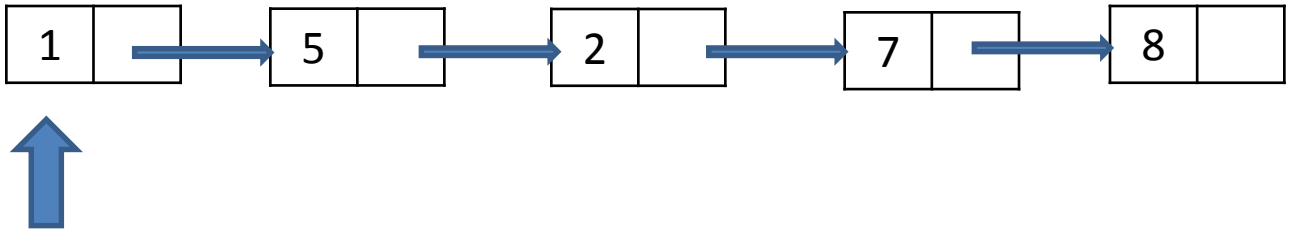
curNode

```
//链接 把n2的地址放在n1里面
n1.next=&n2; // n1->n2
n2.next=&n5; // n1->n2->n5
n5.next=&n7; // n1->n2->n5->n7
n7.next=&n8; // n1->n2->n5->n7->n8
n8.next=NULL;// n1->n2->n5->n7->n8->NULL

// 定义一个当前指针指向头结点n1
strcut Node *curNode=&n1;
```

链表遍历

从头到尾打印结点的值



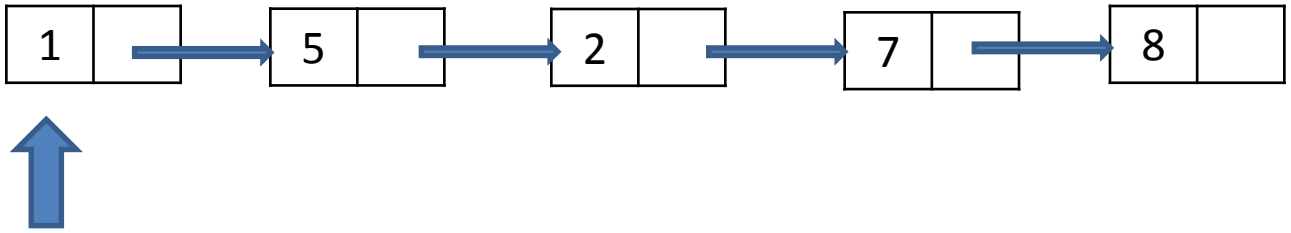
curNode

```
printf("%d ",(*curNode).val);
```

1

链表遍历

从头到尾打印结点的值



curNode

```
printf("%d ",(*curNode).val);
```

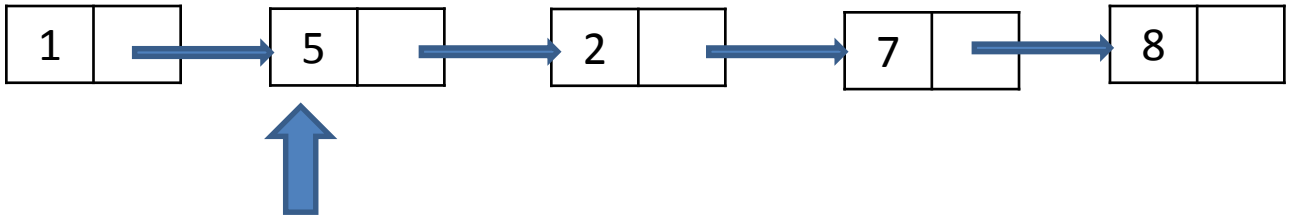
1

```
curNode=(*curNode).next;
```

移动当前指针到下一个结点

链表遍历

从头到尾打印结点的值

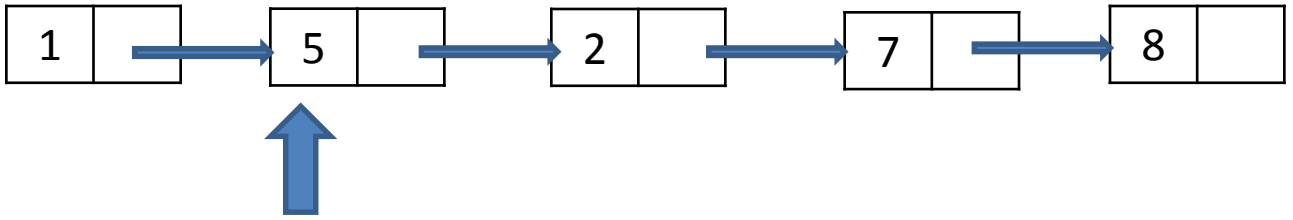



```
printf("%d ",(*curNode).val);
```

5

链表遍历

从头到尾打印结点的值




```
printf("%d ",(*curNode).val);
```

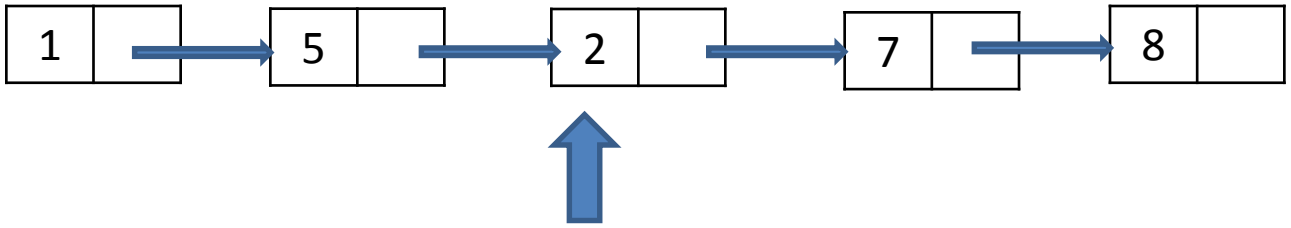
5

```
curNode=(*curNode).next;
```

移动当前指针到下一个结点

链表遍历

从头到尾打印结点的值

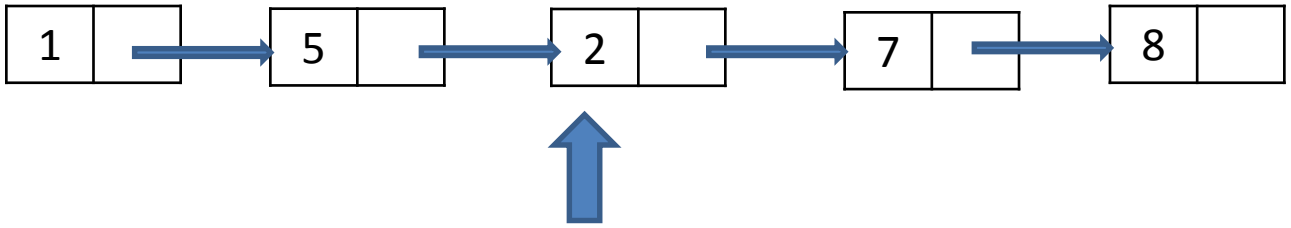



```
printf("%d ",(*curNode).val);
```

2

链表遍历

从头到尾打印结点的值




```
printf("%d ",(*curNode).val);
```

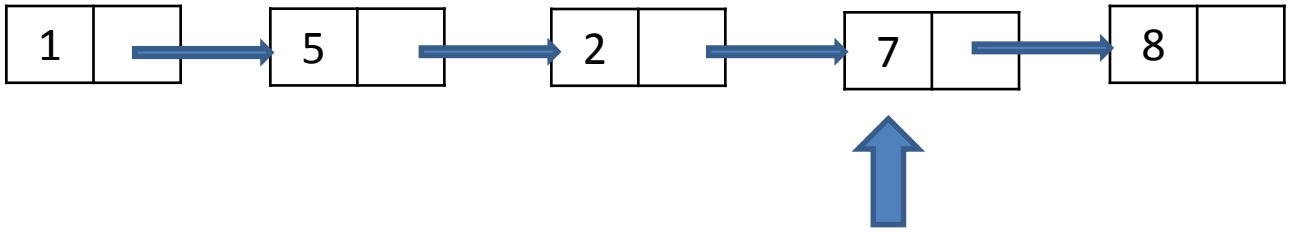
2

```
curNode=(*curNode).next;
```

移动当前指针到下一个结点

链表遍历

从头到尾打印结点的值

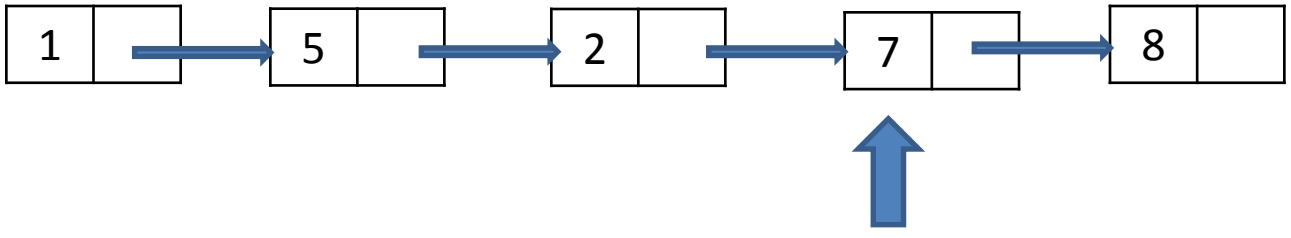



```
printf("%d ",(*curNode).val);
```

7

链表遍历

从头到尾打印结点的值




```
printf("%d ",(*curNode).val);
```

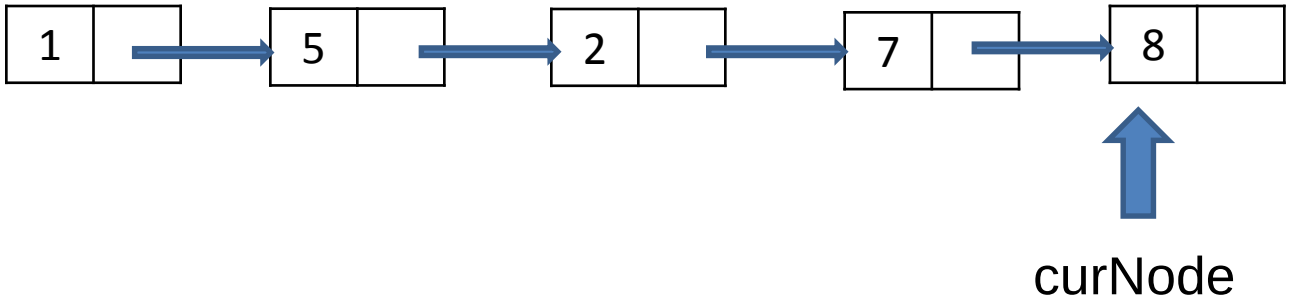
7

```
curNode=(*curNode).next;
```

移动当前指针到下一个结点

链表遍历

从头到尾打印结点的值

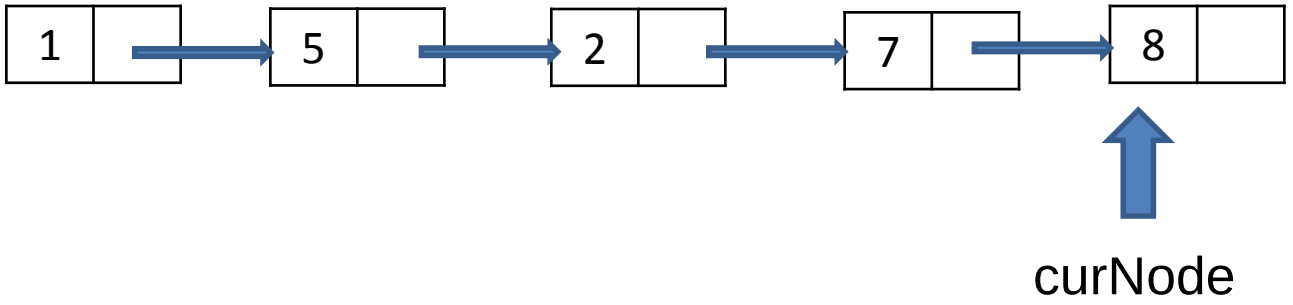



```
printf("%d ",(*curNode).val);
```

8

链表遍历

从头到尾打印结点的值




```
printf("%d ",(*curNode).val);
```

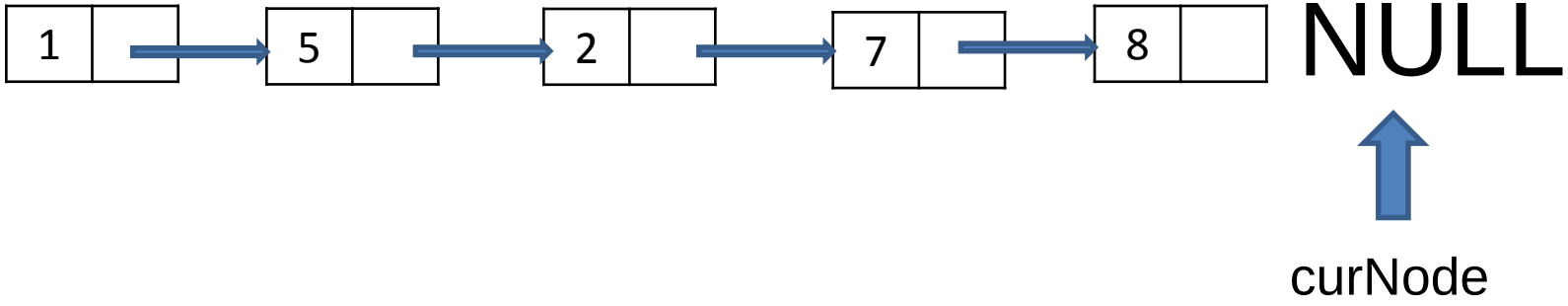
8

```
curNode=(*curNode).next;
```

移动当前指针到下一个结点

链表遍历

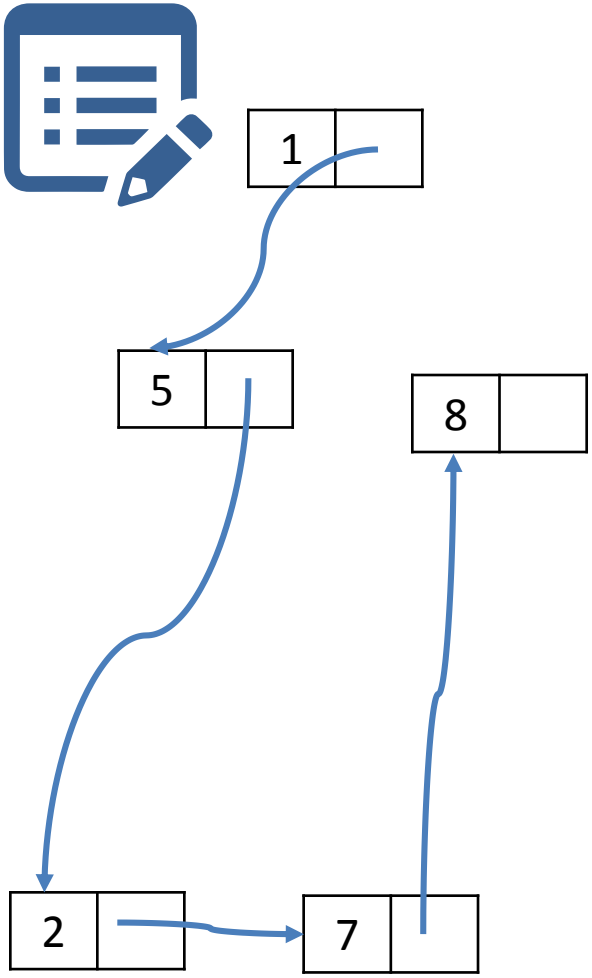
从头到尾打印结点的值



遍历结束

//注意到当前结点指向NULL时结束

链表遍历




```c
#include<stdio.h>

struct Node // 定义一个Node类型
{
    int val; // 结点的值
    struct Node *next; // 下一个结点的地址
};

int main()
{
    struct Node n1,n2,n5,n7,n8; // 使用Node类型创建5个结点

    n1.val=1; // 结点值的初始化
    n2.val=2;
    n5.val=5;
    n7.val=7;
    n8.val=8;

    //链接 把n2的地址放在n1里面
    n1.next=&n2; // n1->n2
    n2.next=&n5; // n1->n2->n5
    n5.next=&n7; // n1->n2->n5->n7
    n7.next=&n8; // n1->n2->n5->n7->n8
    n8.next=NULL;// n1->n2->n5->n7->n8->NULL

    // 定义一个当前指针指向头结点n1
    struct Node *curNode=&n1;
    while(curNode!=NULL)
    {
        printf("%d ",(*curNode).val); // 打印当前结点的值
        curNode=(*curNode).next; // curNode指向下一个结点
    }

    return 0;
}
```

链表遍历

两种等价写法

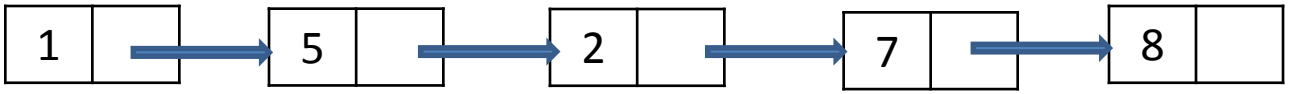
```
// 定义一个当前指针指向头结点n1
struct Node *curNode=&n1;
while(curNode!=NULL)
{
    printf("%d ",(*curNode).val); // 打印当前结点的值
    //printf("%d ",curNode->val); // 等价

    curNode=(*curNode).next; // curNode指向下一个结点
    //curNode=curNode->next; // 等价
}
```

Outlines

一、链表创建
二、链表的遍历
三、链表的查询
四、链表的删除

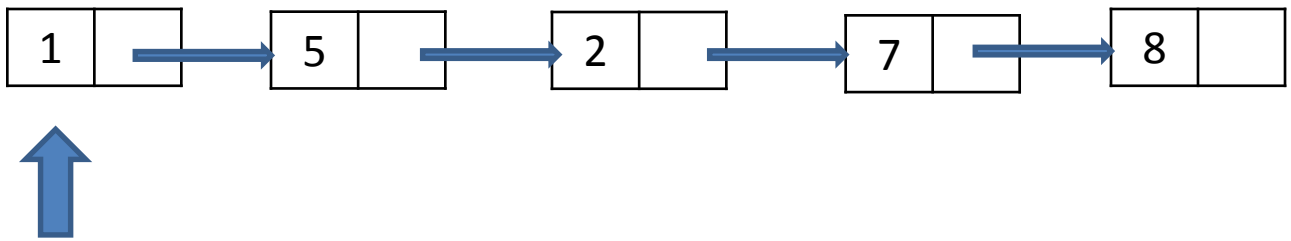
链表查询



查询值为5的结点

即遍历链表，判断当前结点的值是不是需要查询的值
如果是，则结束
如果不是，则继续向后查询直到遇见NULL

链表查询

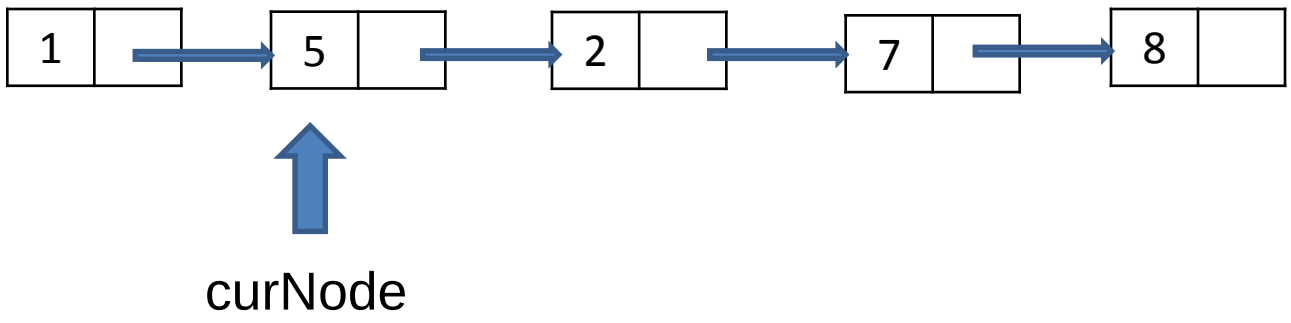


curNode

```
if(curNode->val==5)
{
    printf("欧耶！找到\n");
}
else
{
    curNode=curNode->next; // 继续寻找
}
```

继续寻找

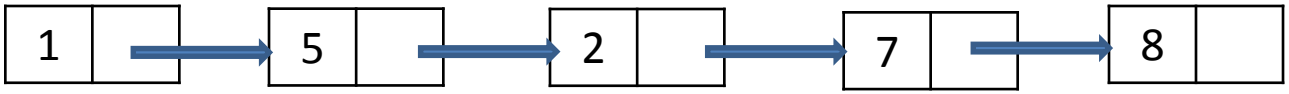
链表查询




```
if(curNode->val==5)
{
    printf("欧耶！找到\n");
}
else
{
    curNode=curNode->next; // 继续寻找
}
```

偶也！找到

链表查询



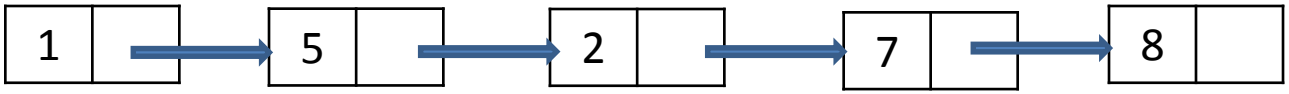
5

```
// 定义一个当前指针指向头结点n1
struct Node *curNode=&n1;
while(curNode!=NULL)
{
    if(curNode->val==5)
    {
        printf("欧耶！找到\n");
        break;
    }
    else
    {
        curNode=curNode->next; // 继续寻找
    }
}
```

Outlines

一、链表创建
二、链表的遍历
三、链表的查询
四、链表的删除

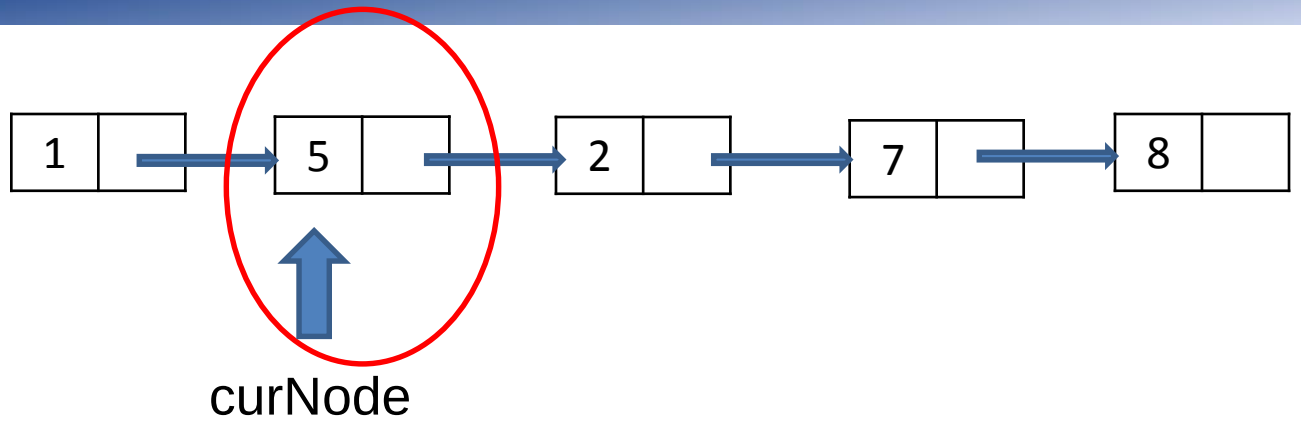
链表删除



删除值为5的结点

即遍历链表，判断当前结点的值是不是需要删除的值
如果是，则删除该结点
如果不是，则继续向后查询直到遇见NULL

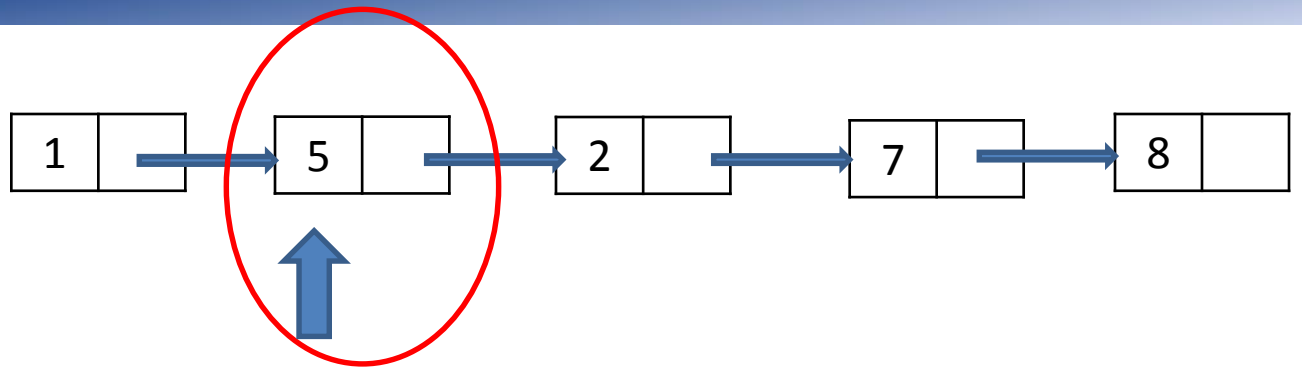
链表删除



5

1、找到该结点

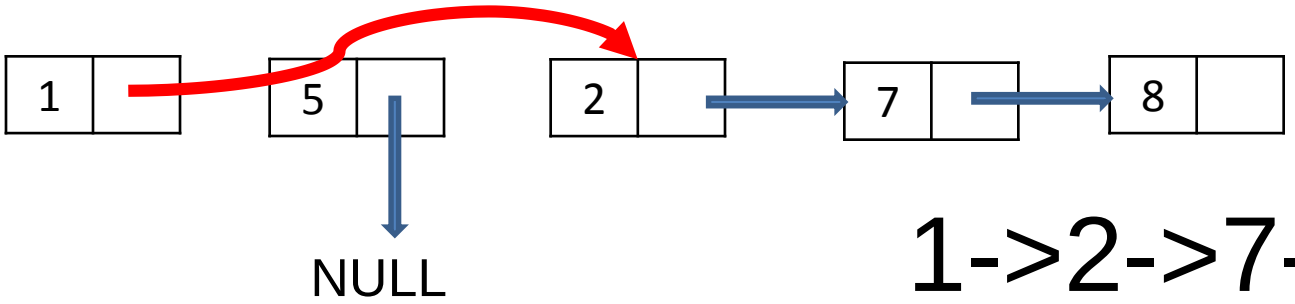
链表删除



5

2、(1)将当前结点前一个结点的next指针指向
当前结点的后一个结点
(2)将当前结点的next指针指向NULL

```
if(curNode->val==5)
{
    preNode->next=curNode->next;
    curNode->next=NULL;
}
```



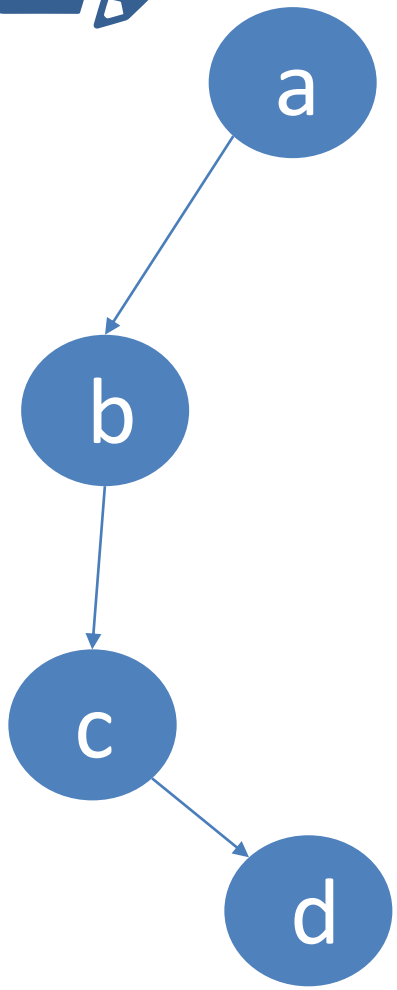
1->2->7->8

练习挑战

练习：编写代码创建如下abcd四个学生结点的链表


```
5  struct Student
6  {
7      int stuId; // 学生学号
8      int stuAge; // 学生年龄
9      char stuName[100]; // 学生姓名
10     float score; // 100分制
11     int weight; // kg
12     int height; // cm
13
14     struct Student *next; // 存放下一个节点的地址
15
16 };
```

练习挑战Answer




```c
#include<stdio.h>
#include<string.h>

struct Student // 1、定义Student结构体类型
{
    int stuId; // 学生学号
    struct Student *next; // 存放下一个节点的地址
};

int main()
{
    struct Student a,b,c,d; //定义学生节点a,b,c,d

    a.studId=2022; // 初始化节点a
    a.next=NULL;

    b.studId=2023; // 初始化节点b
    b.next=NULL;

    c.stuId=2024; // 初始化节点c
    c.next=NULL;

    d.stuId=2025; // 初始化节点d
    d.next=NULL;

    a.next=&b; // a节点指向b节点 a->b->NULL
    b.next=&c; // b节点指向c节点 a->b->c->NULL
    c.next=&d; // c节点指向d节点 a->b->c->d->NULL
    return 0;
}
```

实验ing

完成“简易学生信息管理系统”

1、上传代码
2、写实验报告

The End

Q&A