初级篇

第一部分：链表之内功心法

链表之链结点：

```
typedef int dataType;
class node{
public:
   node(dataType d,node*ptr =nullptr){data=d; next=ptr;} //空指针也可以使用 NULL
   dataType data;
   node* next
}; //也可以考虑模板
或者
struct node{
dataType data;
struct node* next;
};
```

- 由一个个在物理位置上分离的结点通过指针关联构成的批数据表示，我们就称为链表了。
- 最基本的链表被称为单向链表。链表可以为空，也可以由若干链接点通过指针链接构成。它的第一个结点，是（链）表首；最后一个结点，是（链）表尾。
- 链表的管理：一般由一个指向第一个结点的指针来管理，我们叫这个指针为表首（表头）指针；有时候，单向链表也会引入指向链表最后一个结点的指针，我们叫这个指针为表尾指针。

第二部分：链表之横空出世

一、链表的建立（使用 head 来命名表首指针；使用 tail 来命名表尾指针）：

(1) 正向构建（新结点添加在表尾）

```
node* head=new node, *tail=head,*Nnode; //建立第一个结点，表首和表尾指示同一个结点
dataType xdata;//接受输入的临时变量
cin>>xdata;
head->data=xdata;
head->next=nullptr;
while(cin>>xdata) //使用 cin 构建循环，当输入为非 dataType 类型或者 Ctrl+Z 结束
{    Nnode=new node; //创建新结点空间
     Nnode->data=xdata;
     Nnode->next=nullptr;
     tail->next=Nnode; //将新结点挂接在表尾
     tail=Nnode; //新结点成为新表尾
}
```

//使用构造函数 改善 代码

node* head, *tail,*Nnode;

dataType xdata;//接受输入的临时变量
cin>>xdata;
tail=head=new node(xdata); //建立第一个结点，表首和表尾指示同一个结点

while(cin>>xdata) //使用 cin 构建循环，当输入为非 dataType 类型或者 Ctrl+Z 结束
{ Nnode=new node(xdata); //创建新结点空间
tail->next=Nnode; //将新结点挂接在表尾
tail=Nnode; //新结点成为新表尾
}

(2) 逆向构建（新结点添加在表首）

```
node* head=nullptr, *Nnode; //建立第一个结点，表首和表尾指示同一个结点
dataType xdata;//接受输入的临时变量
while(cin>>xdata) //使用 cin 构建循环，当输入为非 dataType 类型或者 Ctrl+Z 结束
{   Nnode=new node(xdata); //创建新结点空间
    Nnode->next=head;//将新结点放在表首
    head=Nnode; //新结点成为新首
}
```

二、链表的访问：

//while 版本

```
node* p=head; //从表首开始
while(p) //一直遍历到表尾
{
  cout<<p->data; //输出每个结点的数据域
  p=p->next;
}
```

//for 版本

```
node* p;
for(p=head; p!=nullptr; p=p->next)
      cout<< p->data<<' ';
```

***课堂扩展-链表的查找(假设要查找的值为 value)：**

```
node* p;
for(p=head; p && p->data!=value; p=p->next) ;
//循环结束后，如 value 存在，则 p 指向 value 所在的结点
```

第三部分：链表之门户管理

一、链表结点的添加(假设要添加的值为 value，添加在指针 p 指向的结点之后，若 p 为空，则添到表首)：

新结点建立：

node* Nnode=new node(value);

1）添加到表尾（根据正向构建）

```
node* tail;
if(head){//非空表，先找表尾
    for(tail=head; tail->next!=nullptr; tail=tail->next);

    tail->next=Nnode;
    tail=Nnode;
}
else //空表，新结点成为唯一结点
    tail=head=Nnode;
```

2）添加到表首（根据逆向构建）

```
Nnode->next=head;
head=Nnode;
```

3）添加到其他 pos 位置相关

```
//添加到 pos 之后，成为 pos 的后继
Nnode->next=pos->next;
pos->next=Nnode;

//添加到 pos 之前，成为 pos 的前驱
node* pre_pos;
if(pos!=head){
     //寻找 pos 的原前驱点
    for(pre_pos=head; pre_pos->next!=pos; pre_pos=pre_pos->next);
    pre_pos->next=Nnode;
    Nnode->next=pos;
}
else {   //pos 为表首，添加成为添加在首
     Nnode->next=head;
     head =Nnode;
}
```

二、链表结点的删除：(假设要删除的结点值为 value,则先找，后删除)

```
    //注意对查找工作的修改
    node *pos, *pre_pos;   //pre 为 p 的前驱点
    for( pre_pos=nullptr, pos=head; pos && pos->next!=value; pre_pos=pos, pos=pos->next) ;
    if(pos){ //待删除点存在
        if(pre_pos){//待删除点为中间结点
           pre_pos->next=pos->next;
         }
        else{//要删除头结点（无前驱）
```

```
            head=pos->next;
        }
        delete pos;
    }
    //若 pos 为空表示待删除点不存在，不需要后续处理
```

三、链表删除：

```
node* p=head;
while(head)
{   head=head->next;
    delete p;
    p=head;
}
```