

# 设计并实现链表

## 单链表的设计

### 结点的定义与设计

```
struct Node
{
    int val;          //数据域
    Node* next;       //后继节点
    Node(int newVal):val(newVal),next(nullptr) {}
};
```

### 类的设计

```
//手撕单向链表
class LinkedList
{
private:
    struct Node
    {
        int val;          //数据域
        Node* next;       //后继节点
        Node(int newVal):val(newVal),next(nullptr) {}
    };
    Node* head;          //头指针应该置空
    int size;
public:
    LinkedList();
    ~LinkedList();
    void addAtTail(int newVal); //添加到尾节点之后
    void addAtSomewhere(int index,int newVal); //添加到指定位置
    void printList();          //遍历输出链表的所有值
    int get(int index);         //获得指定位置的链表值
    void deleteByIndex(int index); //删除指定位置的元素
    int length();               //返回单向链表的长度 不包含头部节点
    void mySort();              //单向链表排序
};
```

## 操作实现

### 构造函数

```
LinkedList::LinkedList()
{
    cout<<"---初始化链表---"<<endl;
    this->head=new Node(1e9);
    this->size++;
}
```

## 析构函数

```
LinkedList::~LinkedList()
{
    Node* workNode=head;
    while(head!=nullptr)
    {
        head=head->next;
        delete workNode;
        workNode=head;
    }
    this->size=0;
}
```

## 添加到尾部

```
void LinkedList::addAtTail(int newVal)
{
    //添加到尾节点后面
    Node* workNode=new Node(0);
    Node* newNode=new Node(newVal);
    workNode=head;
    for(int i=0;i<size;i++)
    {
        if(workNode->next!=nullptr)
        {
            workNode=workNode->next;
        }
    }
    workNode->next=newNode;
    this->size++;
}
```

## 添加到指定位置

```
void LinkedList::addAtSomewhere(int index,int newVal)
{
    //插入元素到指定位置
    Node* workNode=head;    //工作指针 用于游历链表
    int count=0;            //计数器
    while(workNode!=nullptr && count<index-1)
    {
        //定位到传入索引的前一个结点
        workNode=workNode->next;
        count++;
    }
    if(workNode==nullptr) //如过找不到
    {
        throw "传入索引位置错误";
    }
    else
```

```

{
    Node* newNode=new Node(newVal);
    newNode->next=workNode->next;
    workNode->next=newNode;
}
}

```

## 遍历链表

```

void LinkedList::printList()
{
    //遍历链表
    Node* workNode=new Node(0);
    workNode=head->next;
    while(workNode!=nullptr)
    {
        cout<<workNode->val<<" ";
        workNode=workNode->next;
    }
    cout<<endl;
}

```

## 获得指定位置的元素值

```

int LinkedList::get(int index)
{
    //获得指定位置的数据
    Node* workNode=head;
    int count=0; //累加器初始化
    while(workNode!=nullptr && count<index)
    {
        workNode=workNode->next;
        count++;
    }
    if(workNode==nullptr)
    {
        throw "索引错误";
    }
    else
    {
        return workNode->val;
    }
}

```

## 删除指定位置的元素

```

void LinkedList::deleteByIndex(int index)
{
    //删除指定位置的元素
    Node* workNode=head;
    Node* temp=new Node(0); //存储要被删除的结点
    int count=0;

```

```

while(workNode!=nullptr && count<index-1)
{
    //定位到前一个结点
    workNode=workNode->next;
    count++;
}
if(workNode==nullptr)
{
    throw "位置错误";
}
else
{
    temp=workNode->next;    //清除内存
    workNode->next=workNode->next->next;
}
delete temp;
}

```

记得最后要删除结点、释放内存

## 获得链表长度（实际含有的元素个数）

```

int LinkedList::length()
{
    Node* workNode=head;
    int length=0;
    while(workNode->next!=nullptr)
    {
        workNode=workNode->next;
        length++;
    }
    return length;
}

```

## 排序

```

void LinkedList::mySort()
{
    //排序算法
    //大致思路 先将所有元素存储到vector中 再使用vector的排序算法进行排序
    //遍历链表
    Node* workNode=new Node(0);
    vector<int> v1;
    workNode=head->next;
    while(workNode!=nullptr)
    {
        v1.push_back(workNode->val);
        workNode=workNode->next;
    }
    sort(v1.begin(),v1.end());
    workNode=head->next;
    int count=0;
    while(workNode!=nullptr)
    {

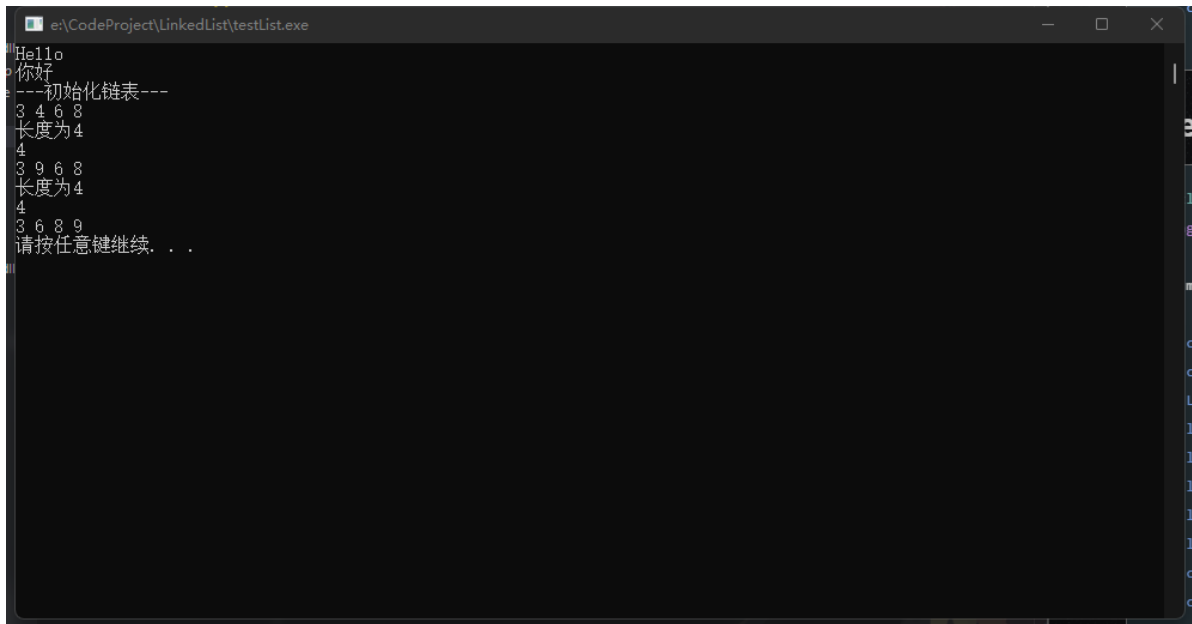
```

```
        workNode->val=v1[count];  
        workNode=workNode->next;  
        count++;  
    }  
    cout<<v1.size()<<endl;  
}
```

algorithm库内置排序算法

```
sort(vector.begin(),vector.end());
```

完整代码见仓库



The screenshot shows a Windows command prompt window titled "e:\CodeProject\LinkedList\testList.exe". The output of the program is as follows:

```
Hello  
你好  
---初始化链表---  
3 4 6 8  
长度为4  
4  
3 9 6 8  
长度为4  
4  
3 6 8 9  
请按任意键继续. . .
```