

湖南大学

数据结构

课程实验报告

题 目： 线性表的实现

学生姓名 吴多智

学生学号 201626010520

专业班级 软件 1605

完成日期 2017.10.25

一、需求分析

1.问题描述

基于数组或链表实现线性表，并实现个小 demo，测试其功能。

2.输入数据

由于要测试 demo 功能（添加、插入、删除）。所以输入的数据主要就是要添加的数据、要插入的数据及坐标和要删除数据的坐标。

3.输出数据

本例输出数据主要为添加、插入和删除后的结果。详情看测试样例设计。

4.测试样例设计

4.1 测试添加的数据：4 1 2 3 4

输出：size: 4

元素为：1 2 3 4

测试插入的数据（前者为数据，后者为插入的下标）： 0 0

输出 size: 5

元素为：0 1 2 3 4

测试删除的数据（删除的下标）： 0

输出 size:4

元素为：1 2 3 4

注：本例主要测试在线性表头部插入与删除

4.2 测试添加的数据：4 1 2 3 4

输出：size: 4

元素为：1 2 3 4

测试插入的数据（前者为数据，后者为插入的下标）： 0 4

输出 size: 5

元素为：1 2 3 4 0

测试删除的数据（删除的下标）： 4

输出 size:4

元素为：1 2 3 4

注：本例主要测试在线性表尾部插入与删除

4.3 测试添加的数据：4 1 2 3 4

输出：size: 4

元素为：1 2 3 4

测试插入的数据（前者为数据，后者为插入的下标）： 9 1

输出 size: 5

元素为：1 9 2 3 4

测试删除的数据（删除的下标）： 2

输出 size:4

元素为：1 9 3 4

注：本例主要测试在线性表中部插入与删除

4.4 测试添加的数据：4 1 2 3 4

输出：size: 4

元素为：1 2 3 4

测试插入的数据（前者为数据，后者为插入的下标）： 9 -1

输出 Index Error

size: 4

元素为: 1 2 3 4

测试删除的数据（删除的下标）: -1

输出 Index Error

size:4

元素为: 1 2 3 4

注：本例主要测试插入和删除坐标不合法时的情况

4.5 测试添加的数据: 4 1 2 3 4

输出: size: 4

元素为: 1 2 3 4

测试插入的数据（前者为数据，后者为插入的下标）: 9 100

输出 Index Error

size: 4

元素为: 1 2 3 4

测试删除的数据（删除的下标）: 100

输出 Index Error

size:4

元素为: 1 2 3 4

注：本例主要测试插入和删除坐标不合法时的情况

二、概要设计

（这个部分一般要撰写三个方面的内容）

1. 抽象数据类型

由于要编写一个线性表的类，但我们事先并不知道需要存储的数据类型是什么，所以采用模版类来完成线性表类

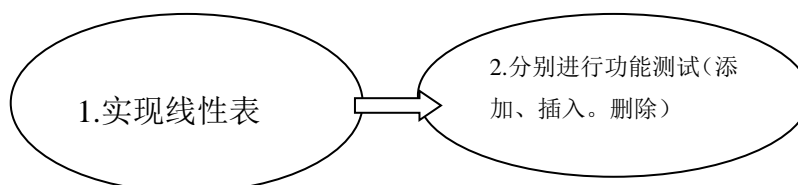
ADT:

```
class Link{
    void add(E e);
    void insert(E e,int position);
    E get(int position);
    E del(int position);
    void clear();
    void reverse();
    bool IsEmpty() {return head->next == NULL ? true : false;};
    int getLength(){return size;}
}
```

2.算法的基本思想

本例仅是实现线性表，无具体算法。

3.程序的流程



三、详细设计

(这个部分一般要撰写四个方面内容)

1. 物理数据类型

本实验基于链表实现线性表，需要具备线性表的基本功能，如添加、插入、查找、删除。

2. 输入和输出的格式

2.1、测试添加：

输入： example 4 1 2 3 4 其中，第一个整数为要添加的节点数，其余的为要添加的数据

输出： Size : 5 (此时现象表大小)

元素 : 1 2 3 4 (添加后线性表中的元素)

2.2、测试插入：

输入： Example 2 3 前者为要插入的数据，后者为下标

输出：输出插入后的结果。

2.3、测试删除

输入： example 4 要删除的节点下标

输出：输出删除后的结果

3. 算法的具体步骤

/*

节点类

*/

```
template <typename E> class Node{
```

```
public:
```

```
    E data;
```

```
    Node<E> * next;
```

```
    Node(){
```

```
        next = NULL;
```

```
    }
```

```
    Node(E e){
```

```
        data = e;
```

```
        next = NULL;
```

```
    }
```

```
};
```

ADT 中具体函数的实现：

//添加节点

//主要步骤： 1、新建节点

2、尾节点 next 指向新节点

3、更新尾节点为新节点

```
template <typename E>
```

```
void MySingleLink<E>::add(E e){
```

```
    if(head == NULL){ //判断头结点是否为空
```

```
        head = new Node<E>(e);
```

```
        end = head;
```

```
        size++;
```

```
    }else{
```

```

        Node<E>* n = new Node<E>(e);
        end->next = n;
        end = n;
        size++;
    }
}
//插入
template <typename E>
void MySingleLink<E>::insert(E e,int position){
    if(position<0 || position>size){        //下标判断
        cout<<"Index Error.\n";
        return;
    }

    if(position==0){        //在头部插入
        Node<E>* n = new Node<E>(e);        //新建节点
        n->next = head;        //新节点 next 指向 head
        head = n;        //新节点为头结点
        size++;
    }else if(position==size){        //在尾部插入
        add(e);        //条用添加方法执行
    }else{        //在中间插入
        Node<E>* n = new Node<E>(e);        //新建节点
        Node<E>* temp = getNode(position-1);        //得到前驱节点
        n->next = temp->next;        //断开操作新节点 next 指向前驱节点的 next
        temp->next = n;        //前驱节点 next 指向新节点
        size++;
    }
}
//删除
template <typename E>
E MySingleLink<E>::del(int index){
    if(index<0 || index>=size){        //判断下标
        cout<<" Index error.\n";
        //exit;
        return -1000;        //由于 dev-c 使用 exit 会使程序退出
    }

    if(index == 0){        //删除头结点
        Node<E>* d = head;
        head = head->next;
        E value = d->data;
        delete d;
        size--;
        return value;
    }
}

```

```

    }else if(index==size-1){    //删除尾节点
        Node<E>* d = end;
        Node<E>* temp = getNode(index-1);
        end = temp;
        end->next = NULL;    //要记得讲为节点的指针变为 NULL
        E value = d->data;
        delete d;
        size--;
        return value;
    }else{    //删除中间节点
        Node<E>* temp = getNode(index-1);
        Node<E>* d = temp->next;
        temp->next = d->next;
        E value = d->data;
        delete d;
        size--;
        return value;
    }
}

```

4.算法的时空分析

基于链表实现的线性表中，添加功能只需要在尾部进行操作，时间复杂度为 $\theta(1)$ ；插入与删除功能在找到后只需要断开与连接，不需要移动其他元素，时间复杂度为 $\theta(1)$ ；但查找功能需要从头节点开始找，时间复杂度为 $\theta(n)$ 。

四、调试分析

1.调试方案设计

由于是基于链表实现的，需要对指针进行操作，故需要调试代码，测试每个功能是否如预期那样执行。

2.调试结果

刚开始时，插入功能的实现有问题，通过调试后发现是断开与连接的顺序反了，及时改正；同样，删除功能也用到了调试。

五、测试结果

```

C:\Users\Administrator\Desktop\SingleLinkedList.exe
本例是用链表实现的线性表。
测试添加功能：请输入添加数据的总数与要添加的数据，以空格隔开。如 4 1 2 3 4
4 1 2 3 4
Size :4
元素为： 1 2 3 4
测试插入功能：请输入需要插入的数据。前一个为数据，后一个为位置，以空格隔开。如 1
3
0 0
Size :5
元素为： 0 1 2 3 4
测试删除功能：请输入需要删除的下标。如 1
0
Size :4
元素为： 1 2 3 4
反转功能
反转后：元素为： 4 3 2 1

-----
Process exited after 64.22 seconds with return value 0
请按任意键继续. . .

```

```

C:\Users\Administrator\Desktop\SingleLinkedList.exe
本例是用链表实现的线性表。
测试添加功能：请输入添加数据的总数与要添加的数据，以空格隔开。如 4 1 2 3 4
4 1 2 3 4
Size :4
元素为： 1 2 3 4
测试插入功能：请输入需要插入的数据。前一个为数据，后一个为位置，以空格隔开。如 1
3
0 4
Size :5
元素为： 1 2 3 4 0
测试删除功能：请输入需要删除的下标。如 1
4
Size :4
元素为： 1 2 3 4
反转功能
反转后：元素为： 4 3 2 1

-----
Process exited after 32.17 seconds with return value 0
请按任意键继续. . .

```

```

C:\Users\Administrator\Desktop\SingleLinkedList.exe
本例是用链表实现的线性表。
测试添加功能：请输入添加数据的总数与要添加的数据，以空格隔开。如 4 1 2 3 4
4 1 2 3 4
Size :4
元素为： 1 2 3 4
测试插入功能：请输入需要插入的数据。前一个为数据，后一个为位置，以空格隔开。如 1
3
9 1
Size :5
元素为： 1 9 2 3 4
测试删除功能：请输入需要删除的下标。如 1
2
Size :4
元素为： 1 9 3 4
反转功能
反转后：元素为： 4 3 9 1

-----
Process exited after 31.65 seconds with return value 0
请按任意键继续. . .

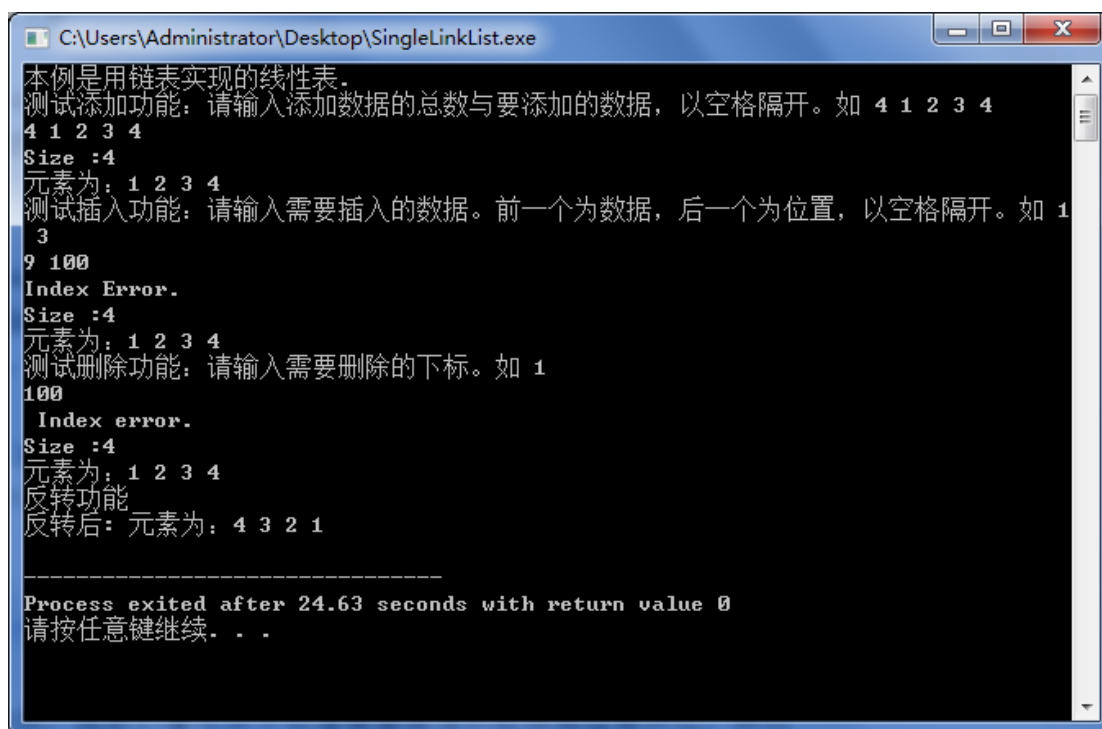
```

```

C:\Users\Administrator\Desktop\SingleLinkedList.exe
本例是用链表实现的线性表。
测试添加功能：请输入添加数据的总数与要添加的数据，以空格隔开。如 4 1 2 3 4
4 1 2 3 4
Size :4
元素为： 1 2 3 4
测试插入功能：请输入需要插入的数据。前一个为数据，后一个为位置，以空格隔开。如 1
3
9 -1
Index Error.
Size :4
元素为： 1 2 3 4
测试删除功能：请输入需要删除的下标。如 1
-1
Index error.
Size :4
元素为： 1 2 3 4
反转功能
反转后：元素为： 4 3 2 1

-----
Process exited after 27.33 seconds with return value 0
请按任意键继续. . .

```

```
C:\Users\Administrator\Desktop\SingleLinkedList.exe
本例是用链表实现的线性表。
测试添加功能：请输入添加数据的总数与要添加的数据，以空格隔开。如 4 1 2 3 4
4 1 2 3 4
Size :4
元素为： 1 2 3 4
测试插入功能：请输入需要插入的数据。前一个为数据，后一个为位置，以空格隔开。如 1
3
9 100
Index Error.
Size :4
元素为： 1 2 3 4
测试删除功能：请输入需要删除的下标。如 1
100
Index error.
Size :4
元素为： 1 2 3 4
反转功能
反转后：元素为： 4 3 2 1

-----
Process exited after 24.63 seconds with return value 0
请按任意键继续. . .
```

六、实验心得

通过对线性表的实现，自己对指针的使用有了进一步的掌握；同时，在基于数组和链表的线性表的实现，清楚的认识到了两者的不同点，各有各的优势，懂了了如何基于实际情况选用合适的实现方式。当需要频繁查找，应使用数组实现；当大小确定，且删除操作频繁时，应考虑使用链表实现。