**实验6：链表+文件1**

**姓名\_\_\_\_\_\_\_\_\_\_\_\_\_班级\_\_\_\_\_\_\_\_\_\_\_\_\_学号\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

* **请阅读此说明：实验6满分100分，做完实验后请按要求将代码和截图贴入该文档。然后将此文档、源代码文件（.hpp, .cpp）打包上传到学习通。**

**1、（总分60分）面向对象的链表类可以设计成：**

**typedef int DataType;**

**class node{**

**public:**

**node(DataType d,node\* ptr=nullptr) {data=d; next=ptr; }**

**DataType data;**

**node\* next;**

**};**

class linked\_list{

public:

linked\_list(); //无参构造，初始化一个空的链表（若定义为带哨兵的链表则默认包含一个结点）

linked\_list(const linked\_list& other); //拷贝构造

linked\_list& operator=(const linked\_list& right); //赋值重载

int list\_size() const; //求链表数据集中的数据规模

//集合并交叉也可以考虑设计为类外函数

linked\_list operator+(const linked\_list& right); //链表集合并

linked\_list operator-(const linked\_list& right);//链表集合差

linked\_list intersectionSet(const linked\_list& right); //链表集合交

node\* find(DataType value); //查找1 返回包含value值的结点地址，找不到返回空指针

bool find(DataType value,node\*& pre,node\*&p);

//查找2： 找到返回真：p为目标点，pre为前驱点; 找不到返回假：p和pre均为nullptr

void add\_front(DataType value); //添加到首

void add\_tail(DataType value);//添加到尾

void add\_pos\_after(DataType value, node \* pos); //添加到指定位置之后

void add\_pos\_before(DataType value,node \*pos);//添加到指定位置之前

void Delete(DataType value);//删除指定值

void delete\_pos\_after(node\* pos);//删除指定位置之后

void delete\_pos\_before(node\* pos);//删除指定位置之前

void delete\_all (DataType value);//删除所有包含value的结点

void delete\_repeat(); //将链表调整为无重复元素的集合

void reverse();//逆置链表

void sort();//升序排列当前链表

void display();//遍历链表，使用逗号间隔输出元素

~linked\_list(); //回收链表内的所有结点

private:

node\* head,\*tail; //表首、表尾

int size; //有效数据个数

}; //可以考虑设计带哨兵的链表，哨兵结点的信息可以代替size 存储表中结点的个数

* **实验要求：参考链表高级篇的讲义。补充完整我们描述的这个单向链表类。并通过给出的测试程序linked\_list\_demo。**

**在实现类的成员函数的过程中注意体会类成员函数和 课堂讨论过程化设计时的封装的独立函数 之间的区别。**

**可以为这个链表添加一个类外定义的普通函数，来完成有序表的合并操作：**

**linked\_lsit mergeSortedList(const linked\_list& L1, const linked\_list& L2 );**

**\*（选做）也可以使用模板类来描述这个单向链表类，则所有的定义系列变为模板：**

**template<class DataType>**

**class node{**

**public:**

**node(DataType d, node<DataType>\* ptr=nullptr) {data=d; next=ptr; }**

**DataType data;**

**node<DataType>\* next;**

**};**

**template<class DataType>**

class linked\_list{

public:

linked\_list(); //无参构造，初始化一个空的链表（若定义为带哨兵的链表则默认包含一个结点）

linked\_list(const **linked\_list<DataType>**& other); //拷贝构造

**linked\_list<DataType>**& operator=(const **linked\_list<DataType>**& right); //赋值重载

int list\_size() const; //求链表数据集中的数据规模

//集合并交叉也可以考虑设计为类外函数

**linked\_list<DataType>** operator+(const **linked\_list<DataType>**& right); //链表集合并

**linked\_list<DataType>** operator-(const **linked\_list<DataType>** & right);//链表集合差

**linked\_list<DataType>**  intersectionSet(const **linked\_list<DataType>**& right); //链表集合交

**node<DataType>**\* find(DataType value); //查找1 返回包含value值的结点地址，找不到返回空指针

bool find(DataType value, **node<DataType>**\*& pre,**node<DataType>**\*&p);

//查找2： 找到返回真：p为目标点，pre为前驱点; 找不到返回假：p和pre均为nullptr

void add\_front(DataType value); //添加到首

void add\_tail(DataType value);//添加到尾

void add\_pos\_after(DataType value, **node<DataType>** \* pos); //添加到指定位置之后

void add\_pos\_before(DataType value, **node<DataType>**\*pos);//添加到指定位置之前

void Delete(DataType value);//删除指定值

void delete\_pos\_after(**node<DataType>**\* pos);//删除指定位置之后

void delete\_pos\_before(**node<DataType>**\* pos);//删除指定位置之前

void delete\_all (DataType value);//删除所有包含value的结点

void delete\_repeat(); //将链表调整为无重复元素的集合

void reverse();//逆置链表

void sort();//升序排列当前链表

void display();//遍历链表，使用逗号间隔输出元素

~linked\_list(); //回收链表内的所有结点

private:

**node<DataType>**\* head,\*tail; //表首、表尾

int size; //有效数据个数

};

template<class DataType>

**linked\_list<DataType> mergeSortedList(const linked\_list<DataType>& L1, const linked\_list<DataType>& L2 );**

**或者：**

**template<class T>**

**T mergeSortedList(const T& L1, const T& L2 );**

* **测试程序：**

**//测试程序：**

**#include<iostream>**

**#include "linkedList.hpp"**

**using namespace std;**

**Int main()**

**{ linked\_list a1,a2, b,c; //若设计的类为模板，则采用linked\_list<int> a1,a2,b,c;**

**DataType data; //若设计的类为模板，则采用int data;**

**//正向和逆向建链测试**

**//输入2 ,6, 7, 3, 5, 9,12, 4 ,0**

**while(cin>>data){**

**if(data==0) break; //输入0结束**

**a1.add\_front(data);**

**a2.add\_tail(data);**

**}**

**a1.display(); //逆向链 4，12，9，5，3，7，6，2**

**a2.display(); //正向链2，6，7，3，5，9，12，4**

**//链表转置测试**

**//输入2 ,16, 3, 8, 15, 4, 9, 7 ,0**

**while(cin>>data) {**

**if(data==0) break; //输入0结束**

**b.add\_tail(data);**

**}**

**b.display(); //原始链表2,16,3,8,15,4,9,7**

**b.reverse();**

**b.display(); //转置结果7,9,4,15,8,3,16,2**

**c=a1+b; //测试集合并**

**c.display(); //4,12,9,5,3,7,6,2,15,8,16**

**c=a1-b; //测试集合差（属于a1且不属于b的数据）**

**c.display(); //12，5，6**

**c=a1.intersectionSet(b); //测试集合交**

**c.display(); //4,9,3,7,2**

**a1.sort(); //测试升序排序**

**a1.display(); //2,3,4,5,6,7,9,12**

**//思考需要降序排序如何做？**

**b.add\_tail(8); b.add\_tail(16); b.add\_tail(3); b.add\_front(3); b.add\_front(16);**

**b.display(); //16,3,7,9,4,15,8,3,16,2,8,16,3**

**b.delete\_all(16); //删除所有的16**

**b.display();//3,7,9,4,15,8,3,2,8,3**

**b.delete\_repeat(); //将b调整为无重复集合**

**b.display();//3,7,9,4,15,8,2**

**node\* pos=b.find(15); //若使用模板，则采用node<int>\* pos=b.find(15);**

**b. add\_pos\_after(18,pos);**

**b.display(); //3,7,9,4,15,18,8,2**

**b.add\_pos\_before(23,pos);**

**b.display();//3,7,9,4,23,15,18,8,2**

**b.delete\_pos\_after(pos);**

**b.display();//3,7,9,4,23,15,8,2**

**b.delete\_pos\_before(pos);**

**b.display();//3,7,9,4,15,8,2**

**b.Delete(7);**

**b.display();//3,9,4,15,8,2**

**b.Delete(8);**

**b.display();//3,9,4,15,2**

**b.sort();**

**b.display();//2,3,4,9,15**

**a2=mergeSortedList(a1,b);**

**a2.display(); //2,3,4,5,6,7,9,12,15**

**return 0;**

**}**

**实验解答：**

**❶黏贴链表类的实现代码：（标注你使用了基本单向链表 or带哨兵的单向链表？）**

**❷黏贴程序测试运行结果窗口（运行结果截屏）：**

**2、文件部分实验：学会ASCII文件的基本输入输出流读写（40分）**

**（1）写一个程序实现：随机生成两批数据，每批10个整数，范围自定。将第一组数据随机数写入 data1.txt；第二组随机书写入data2.txt （20分）**

**（2）写一个程序实现：读取data1.txt和data2.txt的共20个整数有序插入到数组a，再将数组a的结果输出到屏幕同时写入文件data3.txt存档。 （20分）**

**实验解答：**

**❶黏贴（1）的实现代码：**

**❷黏贴（2）的实现代码：**