实验1: 线性表和链表

确保把类定义写在linearList.h中，类实现写在linearList.cpp中

给出的linearList.h中的函数声明不要改动，linearList.h中的内容可以增但不可以删，测试文件将根据已有的函数声明调用，linearList.cpp中的打印函数不要改动。

1. 完成顺序表的基本操作，class sequenceList

* 初始化顺序表sequenceList (const int&, const int&, float[]);//顺序表容量，初始化数组长度，初始化数组
* 销毁顺序表, ~ sequenceList ()
* 添加元素, bool addItem(const float&);//添加元素顺序表尾，成功返回true，失败返回false
* 插入元素, bool insertItem(const int& index, const float&);//插入元素到index位置，成功返回true，失败返回false
* 删除元素,int deleteItem(const float&)//返回删除位置，找不到返回-1
* 查找元素（按序号找）, bool locate(const int&, float& val)// 成功返回true，值放在val中，失败返回false
* 查找元素（按值找）, int locate(const float&)//返回找到位置，找不到返回-1
* 倒序排列元素（使用原空间），void reverse();
* 按序输出元素, void print()//元素个数:元素1, 元素2…

输入输出示例：

float i[3]={1.3,4.5,0.4};

float val;

sequenceList sl(50, 3, i); //顺序表容量，初始化数组长度，初始化数组值

sl.print(); // 3:1.3,4.5,0.4

sl.insertItem(3, 5.5);

sl.print(); // 4:1.3,4.5,0.4,5.5

sl.deleteItem(1.3);//返回0

sl.print();// 3: 4.5,0.4,5.5

sl.locate(9,val);//返回false

sl.locate(4.5);//返回0

sl.print();// 3: 4.5,0.4,5.5

sl.reverse();//

sl.print();// 3: 5.5,0.4,4.5

1. 实现有表头结点的链表的基本操作，class linkList

* 初始化链表, linkList(const int&, float[]);//初始化数组长度，初始化数组
* 销毁链表，~ linkList ()
* 插入元素(头插法)，bool headInsertItem(const float&);成功返回true，失败返回false
* 插入元素(尾插法），bool tailInsertItem(const float&);成功返回true，失败返回false
* 插入元素，int insertItem(const int& index, const float&); //插入元素到index位置，成功返回插入位置，错误返回-1
* 删除元素,int deleteItem(const float&)//返回删除位置，找不到返回-1
* 查找元素（按序号找）, bool locate(const int&, float& val)//成功返回true，值放在val中，失败返回false
* 查找元素（按值找）, int locate(const float&)//返回找到位置，找不到返回-1
* 递增排序，void ascendingOrder()//
* 倒序排列元素，void reverse();
* 按序输出元素, void print()//元素个数:元素1, 元素2…

输入输出示例：

float i[3]={1.3,4.5,0.4};

float val;

linkList ll(3, i);

ll.print(); // 3:1.3,4.5,0.4

ll.tailInsertItem(5.5);

ll.print(); // 4:1.3,4.5,0.4,5.5

ll.headInsertItem(7);

ll.print(); // 5:7,1.3,4.5,0.4,5.5

ll.deleteItem(1.3);//返回1

ll.print();// 4: 7,4.5,0.4,5.5

ll.locate(9,val);//返回false

ll.locate(4.5);//返回1

ll.print();// 4:7, 4.5,0.4,5.5

ll.reverse();//

ll.print();// 4: 5.5,0.4,4.5,7

ll. insertItem(1,10.2);

ll.print(); // 5: 5.5,10.2,0.4,4.5,7

1. 基于上面的链表实现，设计一个算法void merge(linkList& A, linkList& B)，读入A和B两个线性表，输入不需有序，输入后按元素值递增次序排列。编写程序将这两个单链表归并为一个按元素值递减次序排列的单链表，利用原来两个单链表的结点存放归并后的单链表,结果存在A线性表中。

输入输出示例：

float a[3]={1.3,4.5,0.4};

float b[4]={1.5,8.5,0.1,6.2};

linkList alist(3, a);

linkList blist(4, b);

alist.print();//3:1.3,4.5,0.4

blist.print();//4: 1.5,8.5,0.1,6.2

alist.ascendingOrder();

blist.ascendingOrder();

alist.print();//3:0.4,1.3,4.5

blist.print();//4:0.1,1.5,6.2,8.5

merge(alist, blist);

alist.print();//7:8.5,6.2,4.5,1.5,1.3,0.4,0.1