## 2021.9.10 课后编程训练题（array-based list）-陈鹏宇20204227

#### Alist.h

#ifndef ALIST\_H

#define ALIST\_H

#include "list.h"

#include <cassert>

#include <string>

#define defaultSize 1024

using namespace std;

template <typename E>

class AList : public List<E>

{

private:

int maxSize;

int listSize;

int curr;

public:

E\* listArray;

AList();

AList(const int n);

~AList();

void clear();

void insert(const E& it);

void append(const E& it);

E remove();

void moveToStart();

void moveToEnd();

void prev();

void next();

int length() const;

int currPos() const;

void moveToPos(int Pos);

const E& getValue() ;

void Assert(bool val, string s);

void setValue(const E& it); //新增setValue函数

void Swap(int i , int j); //新增Swap函数

};

#endif // ALIST\_H

#### List.h

#ifndef LIST\_H

#define LIST\_H

#include <cassert>

#include <cstring>

template <typename E>

class List

{

private:

void operator = (const List&) {}

List(const List&) {}

public:

List() {};

virtual ~List() {};

virtual void clear() = 0;

virtual void insert(const E& item) = 0;

virtual E remove() = 0;

virtual void moveToStart() = 0;

virtual void moveToEnd() = 0;

virtual void prev() = 0;

virtual void next() = 0;

virtual int length() const = 0;

virtual int currPos() const = 0;

virtual void moveToPos(int pos) = 0;

virtual const E& getValue() = 0;

};

#endif // LIST\_H

#### Alist.cpp

#include "alist.h"

#include "list.h"

#include <iostream>

#include <string>

using namespace std;

template<typename E>

void AList<E>::Assert(bool val, string s)

{

if (!val) {

cout << "Assertion Failed: " << s << endl;

exit(-1);

}

}

template<typename E>

AList<E>::AList(int size)

{

maxSize = size;

listSize = curr = 0;

listArray = new E[maxSize];

}

template<typename E>

AList<E>::AList()

{

maxSize = defaultSize;

listSize = curr = 0;

listArray = new E[maxSize];

}

template<typename E>

AList<E>::~AList()

{

delete[] listArray;

}

template<typename E>

void AList<E>::clear()

{

delete[] listArray;

listSize = curr = 0;

listArray = new E[maxSize];

}

template<typename E>

void AList<E>::insert(const E& it)

{

Assert(listSize < maxSize, "List capacity exceeded");

for (int i = listSize; i > curr; i--)

listArray[i] = listArray[i - 1];

listArray[curr] = it;

listSize++;

}

template<typename E>

void AList<E>::append(const E& it)

{

Assert(listSize<maxSize, "List capacity exceeded");

listArray[listSize++] = it;

}

template<typename E>

E AList<E>::remove()

{

Assert((curr >= 0) && (curr < listSize), "No element");

E it = listArray[curr];

for (int i = curr; i < listSize - 1; i++)

listArray[i] = listArray[i + 1];

listSize--;

return it;

}

template<typename E>

void AList<E>::moveToStart() { curr = 0; }

template<typename E>

void AList<E>::moveToEnd() { curr = listSize; }

template<typename E>

void AList<E>::prev() { if (curr != 0) curr--; }

template<typename E>

void AList<E>::next() { if (curr != listSize) curr++; }

template<typename E>

int AList<E>::length() const { return listSize; }

template<typename E>

int AList<E>::currPos() const { return curr; }

template<typename E>

void AList<E>::moveToPos(int pos)

{

Assert((pos >= 0) && (pos <= listSize), "Pos out of range");

curr = pos;

}

template<typename E>

const E& AList<E>::getValue()

{

Assert((curr >= 0) && (curr <= listSize), "No current element");

return listArray[curr];

}

template<typename E>

void AList<E>::setValue(const E& it)

{

Assert((curr >= 0) && (curr <= listSize), "No current element");

listArray[curr] = it;

}

template <typename E>

void AList<E>::Swap(int i , int j)

{

Assert((i >= 0) && (i <= listSize) && (j >= 0) && (i <= listSize), "Pos out of range");

E temp = listArray[i];

listArray[i] = listArray[j];

listArray[j] = temp;

}

#### List.cpp

#include "list.h"

#### 1. Write a function to reverses the order of the elements on the array-based list.

#include "alist.h"

#include "alist.cpp"

using namespace std;

template <typename E>

void myReverse(AList<E> &myList)

{

int n = myList.length();

int right = n-1 , left = 0;

int times = n/2;

for (myList.moveToStart() ; left < times ; right-- , left++) //只需交换n/2次

myList.Swap(left,right);

}

int main()

{

AList<int> myList;

for (myList.moveToStart(); myList.currPos() < 3; myList.next())

{

int m;

cin>>m;

myList.append(m);

}

myReverse(myList);

for (myList.moveToStart(); myList.currPos() < 3; myList.next())

cout << myList.getValue() << endl;

return 0;

}

文本

描述已自动生成

#### 2. Given two sorted array-based lists L1 and L2, Write a function to computer L1 ∩ L2.

#include "alist.h"

#include "alist.cpp"

using namespace std;

template <typename E>

void mySort(AList<E> &t) //从小到大排序

{

int len = t.length();

for(int i = 0 ; i < len-1 ; i++)

for(int j = 0 ; j < len-1-i ; j++)

{

E t1 , t2;

t.moveToPos(j) ; t1 = t.getValue();

t.moveToPos(j+1) ; t2 = t.getValue();

if(t1 > t2)

t.Swap(j,j+1);

}

}

template <typename E>

void myIntersection(AList<E> &t1 , AList<E> &t2 , AList<E> &ans) //此处的box只储存100位不同元素，可修改

{

int box[100]{0};

for(t1.moveToStart() ; t1.currPos() < t1.length() ; t1.next())

{

E temp = t1.getValue();

box[temp]++;

}

for(t2.moveToStart() ; t2.currPos() < t2.length() ; t2.next())

{

if(box[t2.getValue()] == 1)

ans.append(t2.getValue());

}

}

int main()

{

AList<int> a(4);

AList<int> b(5);

AList<int> ans;

for(a.moveToStart() ; a.currPos() < 4 ; a.next())

{

int temp ; cin >> temp;

a.append(temp);

}

for(b.moveToStart() ; b.currPos() < 5 ; b.next())

{

int temp ; cin >> temp;

b.append(temp);

}

myIntersection(a,b,ans);

mySort(ans);

for(ans.moveToStart() ; ans.currPos() < ans.length() ; ans.next())

cout<<ans.getValue()<<" ";

}

文本

描述已自动生成

#### 3. Assume there are two ascending ordered array-based lists L1 and L2, please merge L1 and L2 into a new descending array-based list L3. There will be no duplicate items in L3.

#include "alist.h"

#include "alist.cpp"

using namespace std;

template <typename E>

void myReverse(AList<E> &myList)

{

int n = myList.length();

int right = n-1 , left = 0;

int times = n/2;

for (myList.moveToStart() ; left < times ; right-- , left++) //n为偶数或奇数时，都只需交换n/2次

myList.Swap(left,right);

}

template <typename E>

void myUnion(AList<E> &t1 , AList<E> &t2 , AList<E> &ans) //此处的box只储存1-100，可修改

{

int box[100] {0};

for(t1.moveToStart() ; t1.currPos() < t1.length() ; t1.next())

{

E temp = t1.getValue();

box[temp]++;

}

for(t2.moveToStart() ; t2.currPos() < t2.length() ; t2.next())

{

E temp = t2.getValue();

box[temp]++;

}

for(int i = 0 ; i < 100 ; i++)

if(box[i])

ans.append(i);

}

int main()

{

AList<int> a(4);

AList<int> b(5);

AList<int> ans;

for(a.moveToStart() ; a.currPos() < 4 ; a.next())

{

int temp ; cin >> temp;

a.append(temp);

}

for(b.moveToStart() ; b.currPos() < 5 ; b.next())

{

int temp ; cin >> temp;

b.append(temp);

}

myUnion(a,b,ans);

myReverse(ans);

for(ans.moveToStart() ; ans.currPos() < ans.length() ; ans.next())

cout<<ans.getValue()<<" ";

}

文本

描述已自动生成

#### 4.有一个顺序表L，假设元素类型为整型，设计一个尽可能高效的算法，以第一个元素为分界线，将所有小于等于它的元素移到该元素的前面，将所有大于它的元素移到该元素的后面。

#include "alist.h"

#include "alist.cpp"

using namespace std;

template <typename E>

void sideSort(AList<E> &a)

{

int i = 0 ; int j = a.length()-1;

a.moveToPos(i);

int vue = a.getValue(); //标记值

while(j > i) //从后向前遍历

{

a.moveToPos(j) ; E t = a.getValue();

if(t == vue) //若与标记值相同，放在标记值位置i，标记i后移

{

a.remove();

a.moveToPos(i);

a.insert(vue);

i++;

continue;

}

else if(t < vue) //若小于标记值，放在表头，标记位置后移

{

a.remove();

a.moveToStart();

a.insert(t);

i++;

continue;

}

else

j--; //直到j = i，结束遍历；

}

}

int main()

{

AList<int> a(9);

for(a.moveToStart() ; a.currPos() < 9 ; a.next())

{

int temp ; cin >> temp;

a.append(temp);

}

sideSort(a);

for(a.moveToStart() ; a.currPos() < a.length() ; a.next())

cout<<a.getValue()<<" ";

}

文本

描述已自动生成

#### 5.有一个顺序表L，设计一个算法将L中所有奇数序号的元素移到所有偶数序号元素的前面，例如，L=<1,2,3,4,5,6,7>，移动后变为L=<1,7,3,5,4,6,2>.

#include "alist.h"

#include "alist.cpp"

using namespace std;

template <typename E>

template <typename E>

void Separate(AList<E> &a) //根据题目提示，可推测，从表首表尾同时遍历

{

int i = 0 ; int j = a.length();

while(i < j)

{

while(i < j)

{

E t1 ; a.moveToPos(j) ; t1 = a.getValue();

if(t1 % 2 == 0)

j--;

else

break;

}

while(i < j)

{

E t2 ; a.moveToPos(i) ; t2 = a.getValue();

if(t2 % 2 != 0)

i++;

else

break;

}

a.Swap(i,j) ; j-- ; i++;

}

}

int main()

{

AList<int> a(7);

for(a.moveToStart() ; a.currPos() < 7 ; a.next())

{

int temp ; cin >> temp;

a.append(temp);

}

Separate(a);

for(a.moveToStart() ; a.currPos() < a.length() ; a.next())

cout<<a.getValue()<<" ";

}文本

描述已自动生成