**实验报告一**

题目：顺序表及其应用——顺序表的基本操作及有序表的合并

班级：计科一班 姓名：元雨暄 学号：1725111037 完成日期：2018.9.19

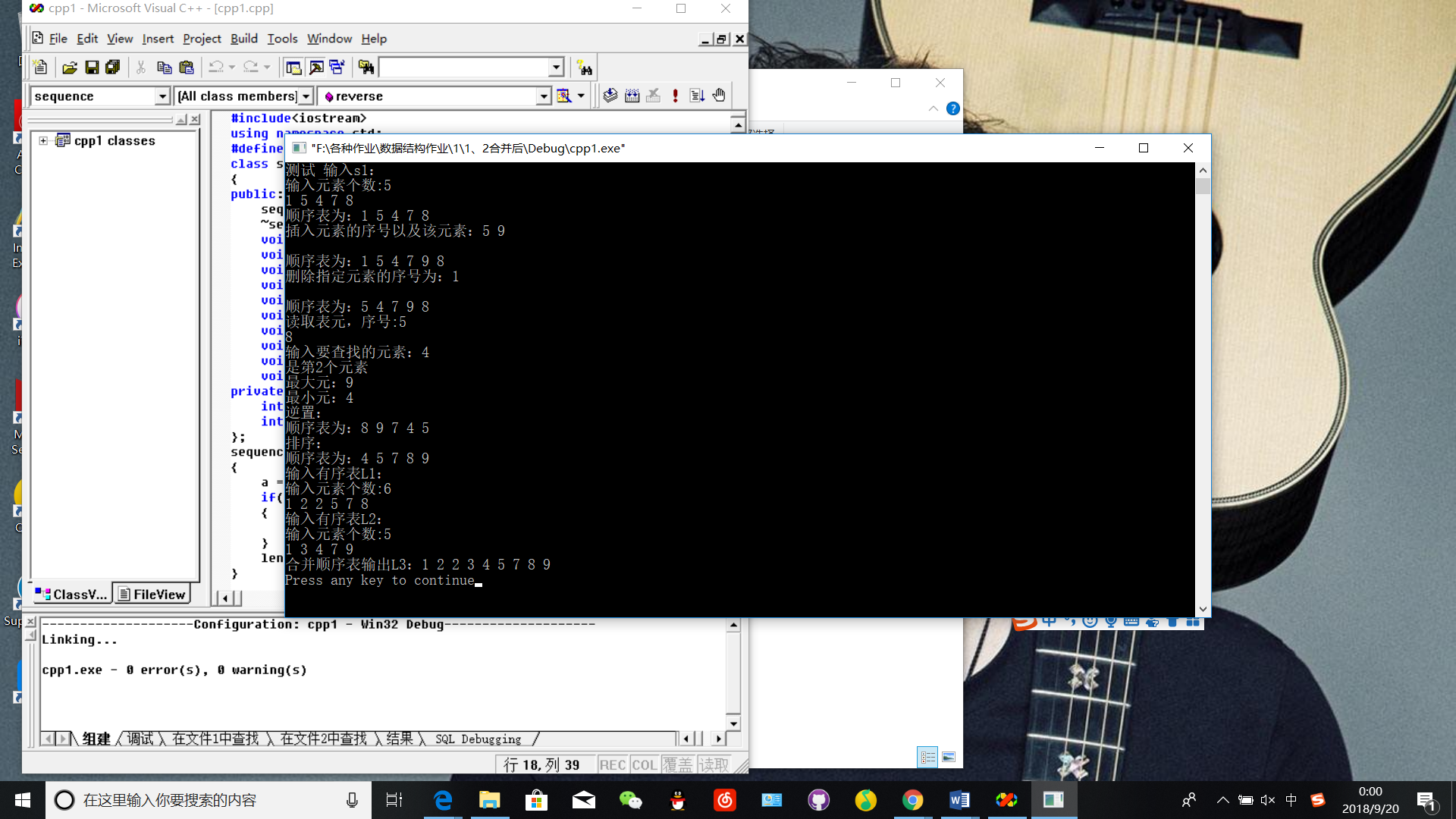
**一、需求分析**

输入形式为键盘输入，输入为整数型数据，输入L1和L2两个顺序表，输出形式为合并后的顺序表，测试时还需如输入相应数据以完成对顺序表的基本操作。

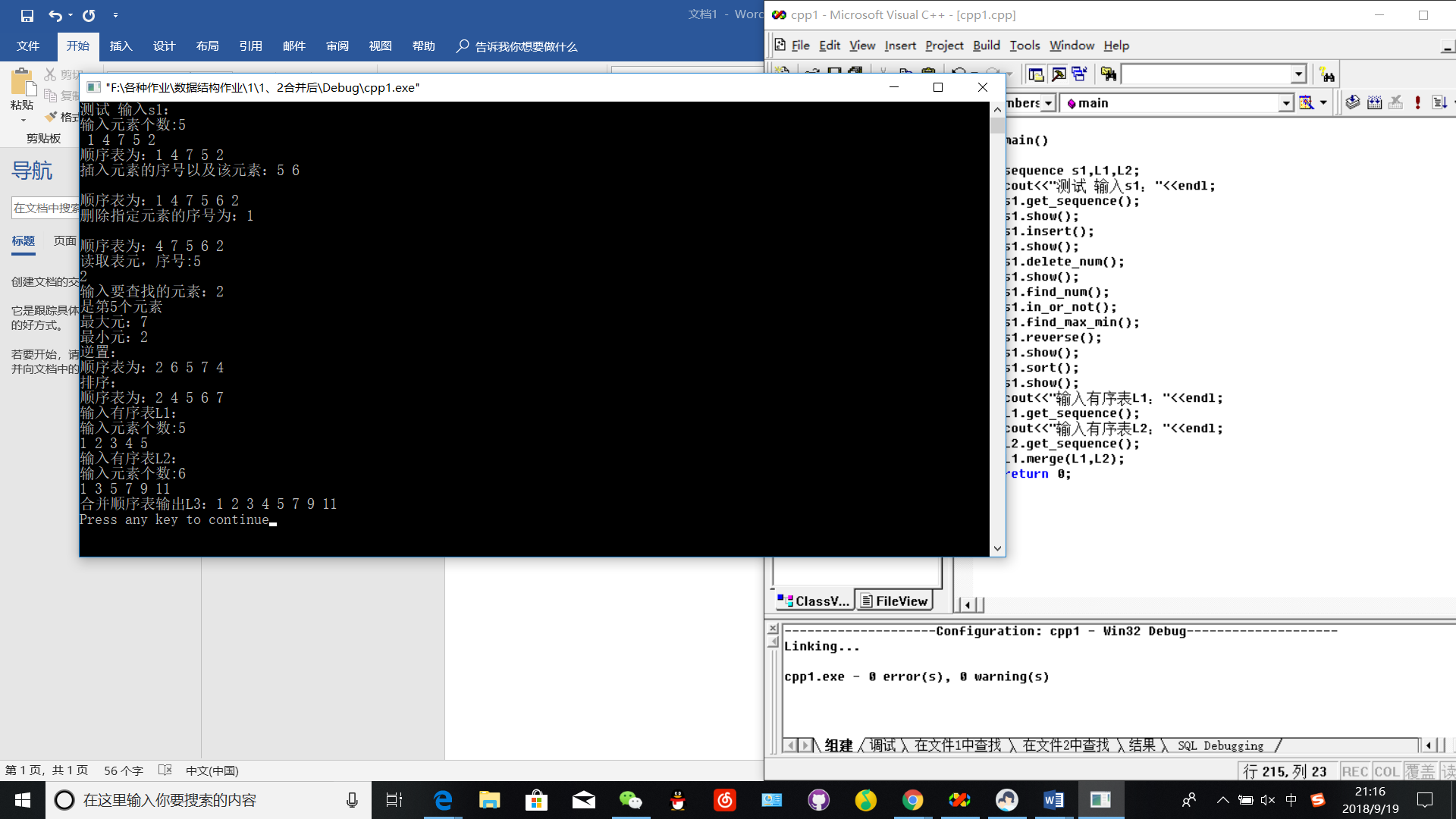
程序能够对一个顺序表进行创建顺序表、插入和删除指定序号的元素、读取表元、获取最大和最小值元素、查找元素、表元素的排序、表元素逆置、顺序表的输入和输出等基本操作。还能合并两个有序表。

测试数据：对于L1，L2输入后，若L1和L2里面各自都没有重复的，但是L1和L2里面有相同的，在L3中可变成无重复的，若L1或者L2自身有重复的则难以消除L3中重复的元素。

测试数据：1.错误的（L3输出重复）



2.正确的（L3输出正确且无重复）



**二、概要设计**

定义了一个类，用来实现顺序表的基本操作

ADT sequence

{

数据对象：D={ai | ai∈IntSet,i=1,2,…n,0<n<=100}

数据关系：R1={<ai-1,ai>|ai-1,ai∈D, i=2,…,n}

基本操作：

sequence()

操作结果：开辟一个空间用来存放数据

~sequence()

操作结果：用来删除空间

get\_sequence()

操作结果：用来输入顺序表

insert()

操作结果：用来插入元素

show()

操作结果：用来输出顺序表

delete\_num()

操作结果：用来删除某个序号的元素

find\_num()

操作结果：用来读取表中元素，输入序号，输出该序号的元素

in\_or\_not()

操作结果：用来判断元素是否在顺序表中

find\_max\_min()

操作结果：输出最大元最小元

sort()

操作结果：对顺序表由小到大排序

reverse()

操作结果：对顺序表逆序排序

merge(sequence&,sequence&)

操作结果：用来合并顺序表

包含两个模块

主程序模块：

int main()

{

初始化：

接受命令；

处理命令；

类模块：

实现顺序表的基本操作

各模块之间调用关系如下：

**主程序模块**

**类模块**

**三、详细设计**

Sequence类的基本操作如下

sequence()

//初始化，为列表开辟空间

~sequence()

//清除开辟的空间

void get\_sequence()

//键盘输入元素个数以及顺序表

void insert()

//输入元素的插入序号以及元素内容

void show()

//输出每个元素

void delete\_num()

//输入想要删除某个序号的元素，按序号找到需要删除的元素序号然后将该元素之后的每个元素往前移一位

void find\_num()

//输入序号，输出那个序号的元素

void in\_or\_not()

//输入一个数，遍历顺序表，检查是否有该元素

void find\_max\_min()

//遍历顺序表，输出最大最小元素

void sort()

//遍历顺序表，将列表按非递减顺序排序

void merge(sequence&,sequence&)

//合并两个顺序表，按非递减顺序排序，新的顺序表里面没有重复的元素

void reverse()

//将顺序表里的元素逆置，分成一半每一对元素互换

**函数调用关系图**

各函数独立调用

sequence();

~sequence();

void get\_sequence();

void insert();

void show();

void delete\_num();

void find\_num();

void in\_or\_not();

void find\_max\_min();

void sort();

void merge(sequence&,sequence&);

void reverse();

**四、调试分析**

（1）除去未定义未加上分号等，若L1或者L2中存在重复的元素，则在L3中难以去除，实现过程中本来写多个函数最后决定将其封装成类，以便每个顺序表都能进行基本操作且互相独立易分清楚。

（2）

时间复杂度：

get\_sequence()：O(n) （n为元素个数）

insert()：O(n)

void show()：O(n)

delete\_num()：O(n)

find\_num()：O(n)

in\_or\_not()：O(n)

find\_max\_min()：O(n)

sort()：O()

merge(sequence&,sequence&)：O()

reverse():O(n)

空间复杂度:

get\_sequence()：O(n) （n为元素个数）

insert()：O(1)

void show()：O(1)

delete\_num()：O(1)

find\_num()：O(1)

in\_or\_not()：O(1)

find\_max\_min()：O(2)

sort()：O(1)

merge(sequence&,sequence&)：O(n)

改进设想：除去L1和L2中的重复元素，使得L3中没有重复元素

**五、用户使用说明**

运行时只要根据文字提示向窗口输入相应的数据即可，首先是测试顺序表，1、先输入元素个数，查看是否超过最大长度，再依次输入各元素

2、执行插入元素的函数，输入序号检查是否超过长度，若没有，则输入元素，

3、删除指定序号的元素，先检查输入的序号有无在范围内，有则删除那个位置的元素，没有则弹出超过范围，

4、输入指定序号，读出相应序号的元素，先检查输入的序号有无在范围内，有则输出那个位置的元素，没有则弹出超过范围，

5、输出要查找的元素，若表中有则输出元素的序号，没有则跳过，

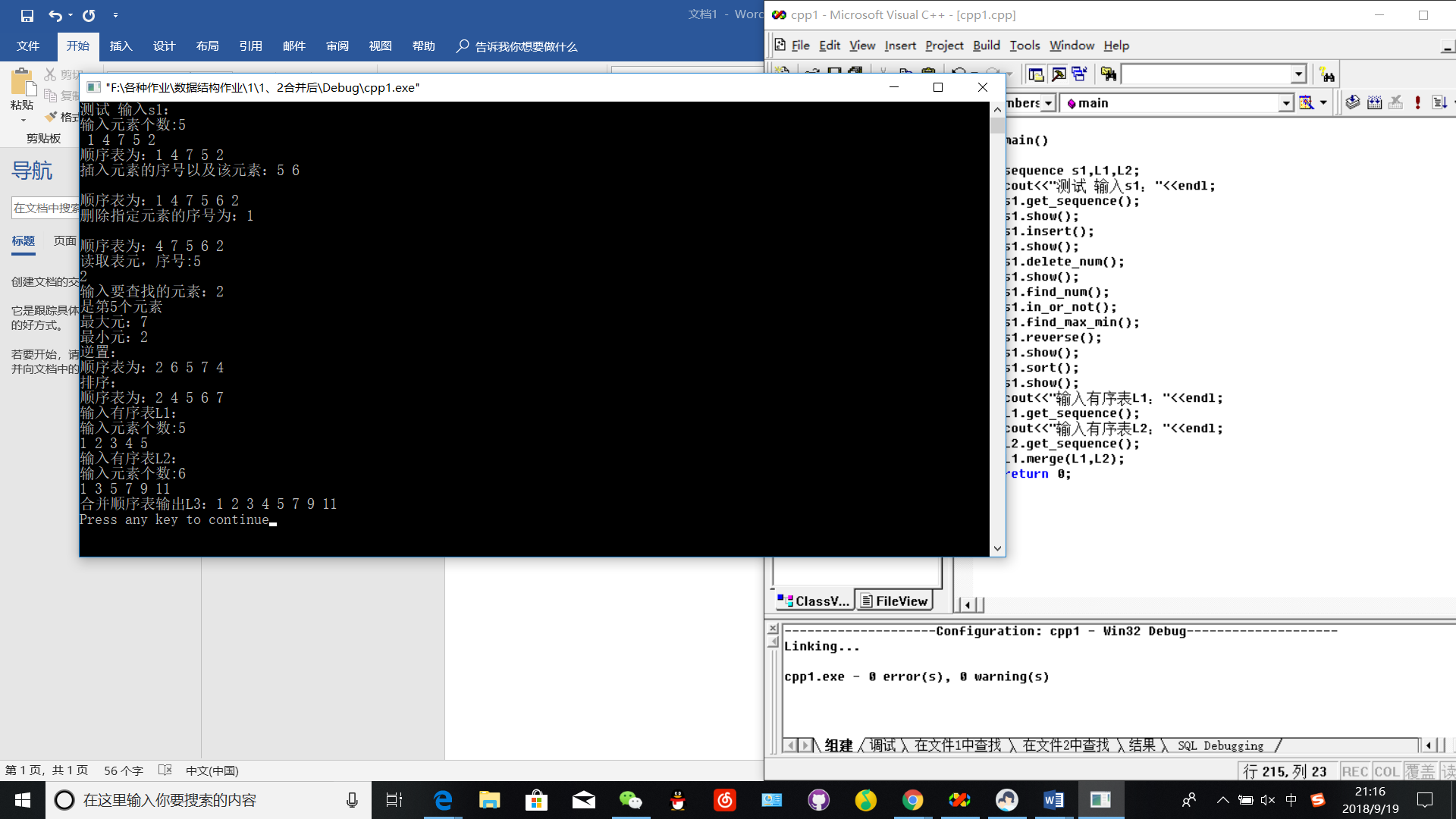
6、输出顺序表中的最大元素和最小元素，

7、将顺序表的元素逆置，

8、将顺序表排序，

9、输入L1和L2作顺序表，需要输入非递减序列。Merge函数将L1和L2合并成一个L3顺序表。

**六、测试结果**



**七、附录**

#include<iostream>

using namespace std;

#define MAX 100

class sequence

{

public:

sequence();

~sequence();

void get\_sequence(); //输入顺序表

void insert(); //插入元素

void show(); //输出顺序表

void delete\_num(); //删除制定序号的元素

void find\_num(); //输入指定序号，输出元素

void in\_or\_not(); //输入数字，检查是否在顺序表内，有则输出元素的序号

void find\_max\_min(); //输出顺序表的最大元素和最小元素

void sort(); //对顺序表进行排序

void merge(sequence&,sequence&);//将两个有序表合并成一个表并且输出

void reverse(); //对顺序表进行逆序排列

private:

int length;

int \*a;

};

sequence::sequence()

{

a = new int[MAX];

if(!a)

{

exit(0);

}

length=0;

}

sequence::~sequence()

{

delete []a;

}

void sequence::get\_sequence()

{

int n,i;

cout<<"输入元素个数:";

cin>>n;

if(n>MAX || n<1)cout<<"超出范围！"<<endl;

for(i=0;i<n;i++)

cin>>a[i];

length=n;

}

void sequence::insert()

{

int index\_in,in,i;

cout<<"插入元素的序号以及该元素：";

cin>>index\_in>>in;

if(index\_in==100 || index\_in<1)cout<<"超出范围！"<<endl;

for(i=length;i>index\_in-2;i--)

{

a[i+1]=a[i];

}

a[index\_in-1]=in;

length++;

cout<<endl;

}

void sequence::delete\_num()

{

int index\_out,i;

cout<<"删除指定元素的序号为：";

cin>>index\_out;

if(index\_out>length || index\_out<1)cout<<"没有该序号的元素！"<<endl;

for(i=index\_out;i<length;i++)

a[i-1]=a[i];

length--;

cout<<endl;

}

void sequence::in\_or\_not()

{

int num,i;

cout<<"输入要查找的元素：";

cin>>num;

for(i=0;i<length;i++)

{

if(a[i]==num)

{

cout<<"是第"<<i+1<<"个元素"<<endl;

break;

}

if(a[i]==a[length] && a[i]!=num)

cout<<"没有该元素！"<<endl;

}

}

void sequence::show()

{

int i;

cout<<"顺序表为：";

for(i=0;i<length;i++)

cout<<a[i]<<" ";

cout<<endl;

}

void sequence::find\_num()

{

int index\_find;

cout<<"读取表元，序号:";

cin>>index\_find;

if(index\_find>length || index\_find<1)cout<<"超出范围！"<<endl;

else cout<<a[index\_find-1]<<endl;

}

void sequence::find\_max\_min()

{

int max\_num,min\_num,i;

max\_num=a[0];

min\_num=a[0];

for(i=0;i<length;i++)

{

if(max\_num<a[i])max\_num=a[i];

if(min\_num>a[i])min\_num=a[i];

}

cout<<"最大元："<<max\_num<<endl;

cout<<"最小元："<<min\_num<<endl;

}

void sequence::sort()

{

int i,temp,j;

cout<<"排序："<<endl;

for(i=1; i<length; i++)

{

for(j=0;j<length-i; j++)

{

if(a[j+1]<a[j])

{

temp = a[j];

a[j] = a[j+1];

a[j+1] = temp;

}

}

}

}

void sequence::reverse()

{

int temp,i;

cout<<"逆置：";

for(i=0;i<(length/2);i++)

{

temp=a[i];

a[i]=a[length-i-1];

a[length-i-1]=temp;

}

cout<<endl;

}

void sequence::merge(sequence &s1,sequence &s2)

{

int \*p1=s1.a,\*p2=s2.a;

int \*L3=new int[s1.length+s2.length];

int \*p3=L3;

int \*p1\_last,\*p2\_last;

p1\_last=p1+s1.length-1;

p2\_last=p2+s2.length-1;

int count=0,i;

while(p1<=p1\_last && p2<=p2\_last)

{

if(\*p1<\*p2)

{

\*p3++=\*p1++;

count++;

}

else if(\*p1>\*p2)

{

\*p3++=\*p2++;

count++;

}

else if(\*p1==\*p2)

{

\*p3++=\*p1++;

p2++;

count++;

}

}

while(p1<=p1\_last)

{

\*p3++=\*p1++;

count++;

}

while(p2<=p2\_last)

{

\*p3++=\*p2++;

count++;

}

cout<<"合并顺序表输出L3：";

for(i=0;i<count;i++)

cout<<L3[i]<<" ";

cout<<endl;

delete []L3;

}

int main()

{

sequence s1,L1,L2;

cout<<"测试 输入s1："<<endl;

s1.get\_sequence();

s1.show();

s1.insert();

s1.show();

s1.delete\_num();

s1.show();

s1.find\_num();

s1.in\_or\_not();

s1.find\_max\_min();

s1.reverse();

s1.show();

s1.sort();

s1.show();

cout<<"输入有序表L1："<<endl;

L1.get\_sequence();

cout<<"输入有序表L2："<<endl;

L2.get\_sequence();

L1.merge(L1,L2);

return 0;

}