**实验1: 顺序表及其应用——顺序表的基本操作及有序表的合并（2学时）**

**【实验目的】**

1、帮助学生熟练掌握线性表的顺序存储结构（顺序表）

2、帮助学生熟练掌握顺序表的基本操作

3、帮助学生熟练掌握顺序表的简单应用，实现两个有序表的合并操作。

**【实验内容及要求】**

1.实现顺序表的各种基本操作

（1）创建有n个整型元素的顺序表:数据可以由键盘输入，或者随机生成。

int InitSqlist(Sqlist &L,int n);

（2）将一个无序顺序表L变为非递减有序的有序表

int sortSqlist(Sqlist &L);

（3）在显示器上显示一个顺序表：每行10个元素

int displaySqlist(Sqlist L);

2.实现两个有序顺序表的合并。

（1）问题描述：已知两个有序的顺序表L1和L2，表中元素值非递减有序，请编程将其合并为一个新的顺序表L3，且L3依然非递减有序；

（2）问题描述：已知两个有序的顺序表L1和L2，表中元素值非递减有序，请编程将其合并为一个新的顺序表L3，且L3递增有序；（可选）

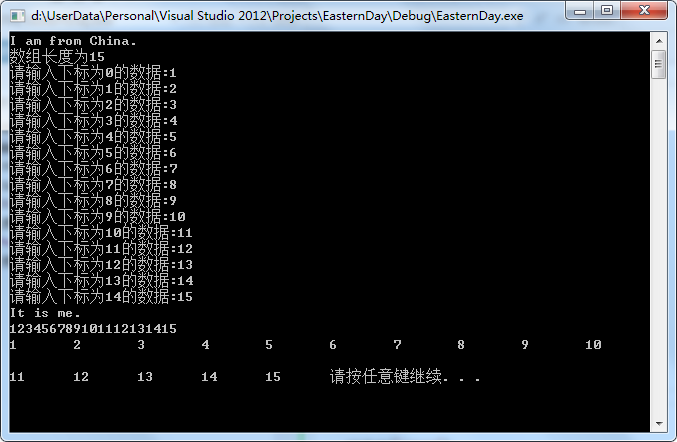
int mergeSqlist(Sqlist &L3,Sqlist L1,Sqlist L2);

3.设计测试用例，测试基本操作和合并操作的正确性。并以word文档的方式保留测试用例，和运行结果截图和说明。

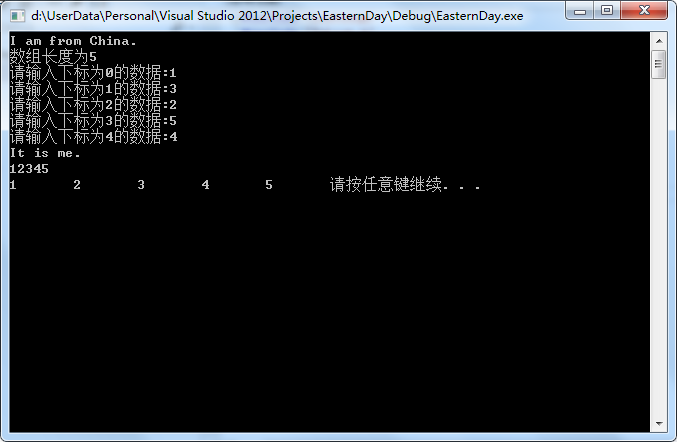
4.实验提交方式：将源程序和word文档一起打包压缩，压缩文件名为“学号+姓名.rar”，并于18:00分前发送邮箱147380431@qq.com

【程序测试】

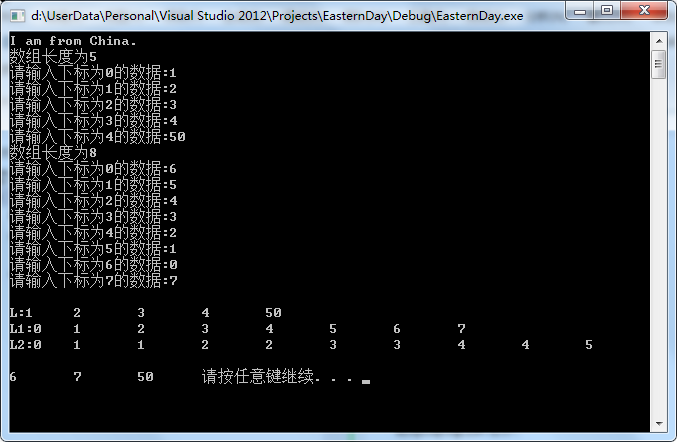
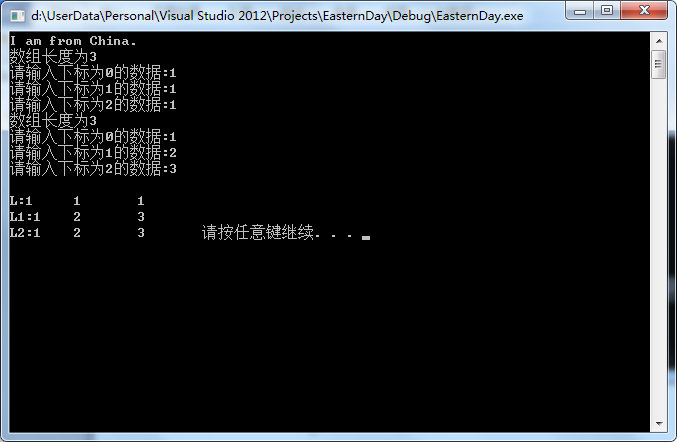
1、显示和生成数组并输入



2、数组排序（选择排序）



3、合并

【程序代码】

//SqList.h

#include <iostream>

using namespace std;

//宏定义

#define MAXSIZE 100

#define Status int

#define ElemType int

#define OK 1

#define ERR 0

#define OVERFLOW -1

//定义SqList结构体

typedef struct SqList

{

int Elem[MAXSIZE]; //数组

int Length; //长度

int ListSize; //总长度

} SqList;

//所有函数定义

//课上要求

Status InitList(SqList &L,int n); //创建有n个整型元素的顺序表:数据可以由键盘输入，或者随机生成。

Status SortSqList(SqList &L); //将一个无序顺序表L变为非递减有序的有序表

Status displaySqList(SqList L); //展示数组，十个一行

Status mergeSqList(SqList La, SqList Lb, SqList &Lc); //合并

Status merge\_SqList(SqList La, SqList Lb, SqList &Lc); //合并递增有序

//初始化SqLst

Status InitList(SqList &L,int n=0)

{

if(n!=0) cout << "数组长度为" << n << endl;

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

cout << "请输入下标为" << i << "的数据:";

cin >> L.Elem[i];

}

//在这里还需要判断是否内存溢出

//初始化数组长度

L.Length = n;

L.ListSize = MAXSIZE;

return OK;

}

//将一个无序顺序表L变为非递减有序的有序表

Status SortSqList(SqList &L){

int mix,temp;

for(int i=0;i<L.Length-1;i++) //每次循环数组，找出最小的元素，放在前面，前面的即为排序好的

{

mix=i; //假设最小元素的下标

for(int j=i+1;j<L.Length;j++) //将上面假设的最小元素与数组比较，交换出最小的元素的下标

if(L.Elem[j]<L.Elem[mix])

mix=j;

//若数组中真的有比假设的元素还小，就交换

if(i!=mix)

{

/\*

temp=L.Elem[i];

L.Elem[i]=L.Elem[mix];

L.Elem[mix]=temp;

\*/

L.Elem[i] = L.Elem[i] + L.Elem[mix];

L.Elem[mix] = L.Elem[i] - L.Elem[mix];

L.Elem[i] = L.Elem[i] - L.Elem[mix];

}

}

return OK;

}

//在显示器上显示一个顺序表：每行10个元素

Status displaySqList(SqList L){

for(int i =0;i<L.Length;i++){

cout << L.Elem[i] << "\t";

if(i%10 == 9) cout << endl;

}

return OK;

}

//合并

Status mergeSqList(SqList La, SqList Lb, SqList &Lc){

int i=0, j=0;

InitList(Lc);

Lc.Length=La.Length+Lb.Length;//初始化Lc

int k =0;

while(i<La.Length&&j<Lb.Length){

if(La.Elem[i]<=Lb.Elem[j]){

Lc.Elem[k]=La.Elem[i];

i++;

k++;

}else{

Lc.Elem[k]=Lb.Elem[j];

j++;

k++;

}

}

while(i<La.Length){

Lc.Elem[k]=La.Elem[i];

i++;

k++;

}

while(j<Lb.Length){

Lc.Elem[k]=Lb.Elem[j];

j++;

k++;

}

return OK;

}

Status merge\_SqList(SqList La, SqList Lb, SqList &Lc){

int i=0, j=0;

InitList(Lc);

int k =0;

while(i<La.Length&&j<Lb.Length){

if(k>0&&Lc.Elem[k-1]==La.Elem[i]){

i++;

}

else if(k>0&&Lc.Elem[k-1]==Lb.Elem[j])

{

j++;

}

else if(La.Elem[i]<Lb.Elem[j]){

Lc.Elem[k]=La.Elem[i];

i++;

k++;

}

else{

Lc.Elem[k]=Lb.Elem[j];

j++;

k++;

}

}

while(i<La.Length){

Lc.Elem[k]=La.Elem[i];

i++;

k++;

}

while(j<Lb.Length){

Lc.Elem[k]=Lb.Elem[j];

j++;

k++;

}

Lc.Length = k;

return OK;

}

//Main.cpp

#include <iostream>

#include "SqList.h"

using namespace std;

int main()

{

//注释

cout << "I am from China." << endl;

SqList L; //定义SqList

InitList(L, 3); //初始化SqList

SortSqList(L); //排序

SqList L1; //定义SqList

InitList(L1, 3); //初始化SqList

SortSqList(L1); //排序

SqList L2; //定义SqList

InitList(L2); //初始化SqList

cout << endl;

cout << "L:";

displaySqList(L);

cout << endl;

cout << "L1:";

displaySqList(L1);

cout << endl;

//mergeSqList(L,L1,L2);

merge\_SqList(L,L1,L2);

cout << "L2:";

displaySqList(L2);

system("pause");

return 0;

}