C语言练习代码（2）--顺序表

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

using namespace std;

#define MaxSize 10

typedef struct{

int data[MaxSize];

int length;

}SqList;

//初始化为一个长度为5的顺序表，便于测试

void InitList(SqList &L) {

int i;

for(i = 0; i < 5 ; i++) {

int a = rand()%10;

L.data[i] = a;

}

L.length = 5;

}

//将初始化后的顺序表打印出来

void printList(SqList &L) {

cout<<"顺序表：";

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

cout<<L.data[i]<<" ";

}

cout<<"；表长："<<L.length<<endl;

}

//向顺序表第p个位置中中插入元素e

bool Insert(SqList &L, int p,int e) {

if(p<1||p>L.length+1)

return false;

for(int i = L.length; i >= p; i--) {

L.data[i] = L.data[i-1];

}

L.data[p-1] = e;

L.length++;

return true;

}

//删除顺序表中第p个位置的值，并用e返回

bool DeleP(SqList &L, int p, int &e) {

if(p<1||p>L.length) {

e = -1;

return false;

}

e = L.data[p-1];

for(int i = p-1; i < L.length-1; i++) {

L.data[i] = L.data[i+1];

}

L.length--;

return true;

}

//删除最小值（假设唯一），并将其值用e返回

bool DeleMin(SqList &L, int &e) {

int i,p;

if(L.length ==0) {

e = -1;

return false;

}

e = L.data[0];

p = 0;

for(i = 1; i < L.length; i++) {

if(e > L.data[i]) {

e = L.data[i];

p = i;

}

}

for(i = p; i<L.length-1; i++) {

L.data[i] = L.data[i+1];

}

L.length--;

return true;

}

//获取顺序表中值为x的第一个位置

int GetPX(SqList &L, int x) {

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

if(x == L.data[i])

return i+1;

}

return -1;

}

//获取顺序表中值为s和t之间的第一个位置

int GetPX(SqList &L, int s, int t) {

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

if((L.data[i]>=s)&&(L.data[i]<=t))

return i+1;

}

return -1;

}

//删除顺序表中值为x的元素

void DeleX(SqList &L, int x) {

int num=0, p;

p = GetPX(L,x);

if(p!=-1) { //当p不为-1，即所删除元素在顺序表中存在的时候，进行下面处理

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

if(x == L.data[i])

num++;

}

for(int i = p-1, j = p; i < L.length-num; i++, j++) {

for(;(L.data[j] == x)&&(j<L.length); j++);

L.data[i] = L.data[j];

}

L.length = L.length-num;

}

}

//删除顺序表中满足s<=x<=t的元素x(从有序表中删除满足s<=x<=t的数，大致思想一样，比从顺序表中删除逻辑更简单，因为删除元素的位置是连续的)

bool DeleBetwST(SqList &L, int s, int t) {

if(L.length == 0)

return false;

if(s>t)

return false;

int num=0, p;

p = GetPX(L, s, t);

if(p!=-1) {

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

if((L.data[i]>=s)&&(L.data[i]<=t))

num++;

}

for(int i = p-1, j = p; (i < L.length-num)&&(j<L.length); i++, j++) {

for(;(L.data[j]>=s)&&(L.data[j]<=t)&&(j<L.length); j++);

L.data[i] = L.data[j];

}

L.length = L.length-num;

}

return true;

}

//将顺序表逆置

void Inverse(SqList &L) {

int temp, i;

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

temp = L.data[i];

L.data[i] = L.data[L.length-1-i];

L.data[L.length-1-i] =temp;

}

}

int main(int argc, char\*\* argv) {

SqList L;

InitList(L);

printList(L);

int e;

DeleMin(L, e);

cout<<"被删除的最小元素e："<<e<<"；删除后顺序表长："<<L.length<<endl;

Inverse(L);

printList(L);

DeleP(L, 3, e);

cout<<"被删除位置对应的的元素e："<<e<<endl;

printList(L);

Insert(L, 2, 7);

Insert(L, 2, 8);

Insert(L, 2, 9);

Insert(L, 2, 7);

Insert(L, 2, 8);

Insert(L, 2, 9);

printList(L);

DeleX(L,7);

printList(L);

Insert(L, 3, 5);

Insert(L, 2, 6);

printList(L);

DeleBetwST(L,3,5);

printList(L);

return 0;

}

