**C语言代码练习（4）--顺序表**

**顺序表的逆置：O(1)**

**一般情况下，逆置复杂的为O(n)，在有些情景下，可以写出满足实际复杂度为O(1)的逆置算法。（双向链表中）**

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

using namespace std;

#define MaxSize 20

typedef struct{

int data[MaxSize];

int length;

}SqList;

//初始化为一个长度为n的随机顺序表，便于测试

void InitList(SqList &L, int n) {

int i;

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

int a = rand()%10;

L.data[i] = a;

}

L.length = n;

}

//初始化为一个长度为n的升序顺序表，便于测试

void InitListUp(SqList &L, int n) {

int i;

L.data[0] = 1;

for(i = 1; i < n ; i++) {

int a = rand()%10;

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

}

L.length = n;

}

//将顺序表打印出来

void PrintList(SqList &L) {

int\* list = L.data;

cout<<"顺序表：";

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

cout<<\*(list++)<<"";

}

cout<<"；表长："<<L.length<<endl;

}

//将顺序表的后（L.length-n）个元素移动到前 n个元素前面

//也可以考虑后面元素不动，前面元素部分放在后面，这样会涉及到顺序表地址的一些问题

void ExchangeList(SqList &L, int n) {

int\* temp = new int[n];

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

if(i<n)

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

if(i<(L.length-n))

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

else

L.data[i] = temp[i-L.length+n];

}

}

//在升序顺序表中查找x，如果能找到，则交互x与其后继元素位置；否则，插入x，并保持顺序表仍然有序。

bool FindX(SqList &L, int x) {

//判断x是否在升序顺序表值的范围内，如果没有，直接插入

if(x < L.data[0]) { //将x插入到有序顺序表的第一位

if(L.length == MaxSize)

return false;

for(int i = L.length; i > 0; i--)

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

L.data[0] = x;

L.length++;

} else if(x > L.data[L.length-1]) { //将x插入到顺序表的最后一位

if(L.length == MaxSize)

return false;

L.data[L.length] = x;

L.length++;

} else if(L.data[L.length-1] == x)

return false;

else { //x介于最小值和最大值中间

int p = 0, q = L.length-1;

while((L.data[(p+q)/2]!=x)&&(q!=p+1)) { //折半查找

if(L.data[(p+q)/2]>x)

q = (p+q)/2;

else

p = (p+q)/2;

}

if(L.data[(p+q)/2] == x) { //查找到==x的元素了

p = (p+q)/2;

int temp = L.data[p];

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

L.data[p+1] = temp;

} else {

if(L.data[p] == x) {

int temp = L.data[p];

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

L.data[p+1] = temp;

} else if (L.data[q] == x) {

int temp = L.data[q];

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

L.data[q+1] = temp;

} else { //将x插入p和q之间

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

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

}

L.data[q] = x;

L.length++;

}

}

}

return true;

}

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

SqList L;

InitList(L, 15);

PrintList(L);

ExchangeList(L,5);

PrintList(L);

InitListUp(L, 15);

PrintList(L);

FindX(L, 0);

PrintList(L);

FindX(L, 55);

PrintList(L);

FindX(L, 33);

PrintList(L);

FindX(L,55);

PrintList(L);

return 0;

}