**头文件：sequlist.h**

#pragma once

#include<stdio.h>

#include<stdlib.h>

#define MAXSISE 100

typedef int datatype;

typedef struct {

datatype a[MAXSISE];

int size;

}sequence\_list;

//置空表——顺序表的初始化

void init(sequence\_list \*slt)

{

slt->size = 0;

}

//顺序表后部进行插入操作

void append(sequence\_list \*slt, datatype x)

{

if (slt->size == MAXSISE)

{

printf("顺序表是满的");

exit(1);

}

else {

slt->a[slt->size] = x;

slt->size++;

}

}

//

void display(sequence\_list \*slt) {

int i;

if (!slt->size)printf("当前顺序表为空");

else {

for (i = 0; i < slt->size; i++)

printf("%5d", slt->a[i]);

}

}

int empty(sequence\_list \*slt) {

return (slt->size == 0 ? 1 : 0);

}

int find(sequence\_list \*slt, datatype x){

int i = 0;

while (i < slt->size && slt->a[i] != x) {

i++;

}

return (i < slt->size ? i : -1);

}

datatype get(sequence\_list \*slt, int i) {

if (i == 0) {

printf("当前顺序表为空"); exit(1);

}

else {

return slt->a[i];

}

}

void insert(sequence\_list \*slt,datatype x,int position) {

int i;

if (slt->size == MAXSISE)

{

printf("当前顺序表是满的！没法插入"); exit(1);

}

if (position<0 || position>slt->size)

{

printf("指定插入的为位置不存在"); exit(1);

}

for (i = slt->size; i > position; i--)

{

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

}

slt->a[position] = x;

slt->size++;

}

void dele(sequence\_list \*slt, int position) {

int i;

if (slt->size == 0) { printf("当前顺序表为空"); exit(1); }

if (position<0 || position>slt->size) {

printf("当前顺序表中删除的位置不存在"); exit(1);

}

for (i = position; i < slt->size; i++)

{

slt->a[i] = slt->a[i + 1];

}

slt->size--;

}

//将小于a[i]的放在数据之前，大于a[i]的放在数据之后，不用排序（从小到大）

void fenlei(sequence\_list\* slt) {

int i, j, k, m, n;

printf("输入分类的数据位置：");

scanf\_s("%d", &i);

if (i>slt->size) {

printf("此数据位置无数据");

return;

}

m = slt->a[i];//数据储存

for (k = i; k > 0; k--) {

slt->a[k] = slt->a[k - 1];

}

slt->a[0] = m;

//上述为将位于中间的数据放在开头

for (j = 1; j < slt->size; j++) {

if (slt->a[j] < m) {

n = slt->a[j];

for (k = j; k > 0; k--) {

slt->a[k] = slt->a[k - 1];

}

slt->a[0] = n;

}

}

}

//排序算法

void paixu(sequence\_list\* slt) {

int j, i;

datatype m;

i = slt->size;

while (i--) {

for (j = 1; j < slt->size; j++) {

if (slt->a[j - 1] > slt->a[j])

{

m = slt->a[j - 1];

slt->a[j - 1] = slt->a[j];

slt->a[j] = m;

}

}

}

}

//合并两个已经排序了的数组，并输出一个排序的数组

void hebing(sequence\_list \* slt, sequence\_list \* p, sequence\_list\* q) {

int i = 0, j = 0, k = 0;//i为slt,j是p,k是q

paixu(slt);

paixu(p);

int n;

n = slt->size;

while(n-- >0)

{

if (slt->a[i] < p->a[j])

q->a[k++] = slt->a[i++];

else

q->a[k++] = p->a[j++];

}

while(i < slt->size) {

q->a[k++] = slt->a[i++];

}

while (j < p->size){

q->a[k++] = p->a[j++];

}

q->size = i+j;

}

//倒置数据

void daoxu(sequence\_list\* slt) {

int i;

datatype m;

printf("%d", slt->size);

for (i = 0; i <=( slt->size-1)/2; i++) {

m = slt->a[slt->size - i-1];

slt->a[slt->size - i-1 ] = slt->a[i];

slt->a[i] = m;

}

}

//插入数据后依然有序

void charu(sequence\_list\* slt,datatype x) {

int i;

paixu(slt);

for (i = 0; i < slt->size; i++) {

if (x < slt->a[i]) {

insert(slt, x, i);

break;

}

}

}

**主函数：**

#include"sequlist.h"

int main() {

sequence\_list\* slt = (sequence\_list\*)malloc(sizeof(sequence\_list));

sequence\_list\* p = (sequence\_list\*)malloc(sizeof(sequence\_list));

sequence\_list\* q = (sequence\_list\*)malloc(sizeof(sequence\_list));

int i, y;

int choose;

int place;

datatype m;

init(slt);

for (i = 0; i < 10; i++) {

slt->a[i] = i + 11;

slt->size++;

}

for (i = 0; i < 10; i++) {

p->a[i] = i + 1;

p->size++;

}

while (1) {

printf("\n请输入行为:\n");

scanf\_s("%d", &choose);

printf("\n");

if (choose == 1) {

printf("在结尾处插入数据");

scanf\_s("%d", &m);

append(slt, m);

}

else if (choose == 2) {

printf("当前数据已打印");

display(slt);

printf("\n");

}

else if (choose == 3) {

printf("判断当前顺序表是否为空(1为空，0为非空)");

printf("%d",empty(slt));

printf("\n");

}

else if (choose == 4) {

printf("请输入顺序表中x的数值:(-1为该值不存在)");

scanf\_s("%d", &y);

i = find(slt, y);

printf("\nx的节点位置为：");

printf("%d", i);

printf("\n");

}

else if (choose == 5) {

printf("请输入插入的数据");

scanf\_s("%d", &m);

printf("请输入插入的数据位置");

scanf\_s("%d", &place);

insert(slt, m, place);

printf("\n");

}

else if (choose == 6) {

printf("请输入删除的数据位置");

scanf\_s("%d", &place);

dele(slt, place);

printf("\n");

}

else if (choose == 7) {

fenlei(slt);

}

else if (choose == 8) {

printf("进行从小到大的排序");

paixu(slt);

}

else if (choose == 9) {

printf("此时p数组的值为：");

for (i = 0; i < 10; i++)

{

printf("%5d", p->a[i]);

}

hebing(slt, p, q);

printf("\n合并后的的数值为:");

for (i = 0; i < q->size; i++) {

printf("%5d", q->a[i]);

}

}

else if (choose == 10) {

printf("进行倒置数据");

daoxu(slt);

}

else if (choose == 11) {

printf("进行输入插入数据:");

scanf\_s("%d", &m);

charu(slt, m);

}

else if (choose == 12) {

break;

}

else {

printf("无此选择，请重新输入");

printf("\n");

}

}

return 0;

}**输出（截图）：**











