综合实验.h

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define OK 1

typedef char elemtype;

typedef int Status;

typedef struct LNode {

elemtype data;

struct LNode \*next;

} LNode,\*LinkList;

Status InitList(LinkList &L) {

L=(LinkList)malloc(sizeof(LNode));

if(!L) exit(-1);

L->next=NULL;

return OK;

}

Status ListInsert\_L(LinkList &L,int i,elemtype e) {

LinkList p=L,q;

int j=0;

for(; p&&j<i-1; j++)

p=p->next;

if(p&&j>i-1) return -1;

q=(LinkList)malloc(sizeof(LNode));

q->data=e;

q->next=p->next;

p->next=q;

}

Status ListPritnt\_L(LinkList &L) {

LinkList p=L;

for(p=p->next; p; p=p->next) {

printf("%c ",p->data);

}

printf("\n");

}

Status ListLength(LinkList &L) {

int i=0;

LinkList p=L->next;

while(p) {

p=p->next;

i++;

}

return i;

}

Status ListSort\_L1(LinkList &L) { //交换结点的冒泡排序

LinkList p,q,p\_prior;

int i,j,n;

n=ListLength(L);

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

p=L->next;

q=p->next;

p\_prior=L;

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

if((p->data)>(q->data)) {

p\_prior->next=q;

p->next=q->next;

q->next=p;

//交换后的更新结点情况和没交换时不同

p\_prior=q;

q=p->next;

} else {

p\_prior=p;

p=q;

q=q->next;

}

}

}

p=q=p\_prior=NULL;

}

Status ListSort\_L2(LinkList &L) { //直接交换内部元素

LinkList p,q;

elemtype temp;

int i,j,n;

n=ListLength(L);

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

for(j=0,p=L->next,q=p->next; j<n-i; p=q,q=q->next,j++) {

if((p->data)>(q->data)) {

temp=p->data;

p->data=q->data;

q->data=temp;

}

}

p=q=NULL;

}

void BubbleSort(struct LNode \* head) {

//https://www.cnblogs.com/darkchii/p/7302412.html

struct LNode \* p, \* q, \* tail;

tail = NULL;

while((head->next->next) != tail) {

p = head;

q = head->next;

while(q->next != tail) {

if((q->data) > (q->next->data)) {

p->next = q->next;

q->next = q->next->next;

p->next->next = q;

q = p->next;

}

q = q->next;

p = p->next;

}

tail = q;

}

}

LinkList ListMerge\_L(LinkList &L1,LinkList &L2) { //L1,L2有序情况的合并

LinkList p=L1->next,q=L2->next;

LinkList L3;

int i=1;

InitList(L3);

while(p&&q) {

if(p->data<q->data) {

ListInsert\_L(L3,i++,p->data);

p=p->next;

} else if(p->data==q->data) {

ListInsert\_L(L3,i++,p->data);

p=p->next;

q=q->next;

} else {

ListInsert\_L(L3,i++,q->data);

q=q->next;

}

}

while(p) {

ListInsert\_L(L3,i++,p->data);

p=p->next;

}

while(q) {

ListInsert\_L(L3,i++,q->data);

q=q->next;

}

return L3;

}

LinkList ListIntersect\_L(LinkList &L1,LinkList &L2) {

LinkList p=L1->next,q=L2->next;

LinkList L;

int i=1,flag=0;

InitList(L);

while(p) {

while(q) {

if(p->data==q->data) {

flag=1;

break;

}

q=q->next;

}

if(flag) {

ListInsert\_L(L,i++,q->data);

}

p=p->next;

q=L2->next;

flag=0;

}

return L;

}

LinkList ListDifferent\_L(LinkList &L1,LinkList &L2) {//就是把交运算改一下

LinkList p=L1->next,q=L2->next;

LinkList L;

int i=1,flag=1;

InitList(L);

while(p) {

while(q){

if(p->data==q->data) {

flag=0;

break;

}

q=q->next;

}

if(flag) {

ListInsert\_L(L,i++,p->data);

}

p=p->next;

q=L2->next;

flag=1;

}

return L;

}

综合实验.cpp

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define OK 1

typedef char elemtype;

typedef int Status;

typedef struct LNode {

elemtype data;

struct LNode \*next;

} LNode,\*LinkList;

Status InitList(LinkList &L) {

L=(LinkList)malloc(sizeof(LNode));

if(!L) exit(-1);

L->next=NULL;

return OK;

}

Status ListInsert\_L(LinkList &L,int i,elemtype e) {

LinkList p=L,q;

int j=0;

for(; p&&j<i-1; j++)

p=p->next;

if(p&&j>i-1) return -1;

q=(LinkList)malloc(sizeof(LNode));

q->data=e;

q->next=p->next;

p->next=q;

}

Status ListPritnt\_L(LinkList &L) {

LinkList p=L;

for(p=p->next; p; p=p->next) {

printf("%c ",p->data);

}

printf("\n");

}

Status ListLength(LinkList &L) {

int i=0;

LinkList p=L->next;

while(p) {

p=p->next;

i++;

}

return i;

}

Status ListSort\_L1(LinkList &L) { //交换结点的冒泡排序

LinkList p,q,p\_prior;

int i,j,n;

n=ListLength(L);

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

p=L->next;

q=p->next;

p\_prior=L;

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

if((p->data)>(q->data)) {

p\_prior->next=q;

p->next=q->next;

q->next=p;

//交换后的更新结点情况和没交换时不同

p\_prior=q;

q=p->next;

} else {

p\_prior=p;

p=q;

q=q->next;

}

}

}

p=q=p\_prior=NULL;

}

Status ListSort\_L2(LinkList &L) { //直接交换内部元素

LinkList p,q;

elemtype temp;

int i,j,n;

n=ListLength(L);

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

for(j=0,p=L->next,q=p->next; j<n-i; p=q,q=q->next,j++) {

if((p->data)>(q->data)) {

temp=p->data;

p->data=q->data;

q->data=temp;

}

}

p=q=NULL;

}

void BubbleSort(struct LNode \* head) {

//https://www.cnblogs.com/darkchii/p/7302412.html

struct LNode \* p, \* q, \* tail;

tail = NULL;

while((head->next->next) != tail) {

p = head;

q = head->next;

while(q->next != tail) {

if((q->data) > (q->next->data)) {

p->next = q->next;

q->next = q->next->next;

p->next->next = q;

q = p->next;

}

q = q->next;

p = p->next;

}

tail = q;

}

}

LinkList ListMerge\_L(LinkList &L1,LinkList &L2) { //L1,L2有序情况的合并

LinkList p=L1->next,q=L2->next;

LinkList L3;

int i=1;

InitList(L3);

while(p&&q) {

if(p->data<q->data) {

ListInsert\_L(L3,i++,p->data);

p=p->next;

} else if(p->data==q->data) {

ListInsert\_L(L3,i++,p->data);

p=p->next;

q=q->next;

} else {

ListInsert\_L(L3,i++,q->data);

q=q->next;

}

}

while(p) {

ListInsert\_L(L3,i++,p->data);

p=p->next;

}

while(q) {

ListInsert\_L(L3,i++,q->data);

q=q->next;

}

return L3;

}

LinkList ListIntersect\_L(LinkList &L1,LinkList &L2) {

LinkList p=L1->next,q=L2->next;

LinkList L;

int i=1,flag=0;

InitList(L);

while(p) {

while(q) {

if(p->data==q->data) {

flag=1;

break;

}

q=q->next;

}

if(flag) {

ListInsert\_L(L,i++,q->data);

}

p=p->next;

q=L2->next;

flag=0;

}

return L;

}

LinkList ListDifferent\_L(LinkList &L1,LinkList &L2) {

LinkList p=L1->next,q=L2->next;

LinkList L;

int i=1,flag=1;

InitList(L);

while(p) {

while(q){

if(p->data==q->data) {

flag=0;

break;

}

q=q->next;

}

if(flag) {

ListInsert\_L(L,i++,p->data);

}

p=p->next;

q=L2->next;

flag=1;

}

return L;

}

int main() {

elemtype A[10]= {'c','a','e','h'};

elemtype B[10]= {'f','h','b','g','d','a'};

int i;

LinkList head1,head2,head3,head4,head5;

InitList(head1);

InitList(head2);

for(i=0; A[i]; i++) {

ListInsert\_L(head1,i+1,A[i]);

}

printf("原 集 合A：");

ListPritnt\_L(head1);

for(i=0; B[i]; i++) {

ListInsert\_L(head2,i+1,B[i]);

}

printf("原 集 合B：");

ListPritnt\_L(head2);

ListSort\_L1(head1);

ListSort\_L2(head2);

printf("有序集合A：");

ListPritnt\_L(head1);

printf("有序集合B：");

ListPritnt\_L(head2);

head3=ListMerge\_L(head1,head2);

printf("集合的并C：");

ListPritnt\_L(head3);

head4=ListIntersect\_L(head1,head2);

printf("集合的交C：");

ListPritnt\_L(head4);

head5=ListDifferent\_L(head1,head2);

printf("集合的差C：");

ListPritnt\_L(head5);

printf("完成者：林身月");

}