数据结构实验报告一

学号：201930310012 姓名：张金宇

#include <string.h>

#include<ctype.h>

#include<malloc.h> /\* malloc()等 \*/

#include<limits.h> /\* INT\_MAX等 \*/

#include<stdio.h> /\* EOF(=^Z或F6),NULL \*/

#include<stdlib.h> /\* atoi() \*/

#include<io.h> /\* eof() \*/

#include<math.h> /\* floor(),ceil(),abs() \*/

#include<process.h> /\* exit() \*/

#include<Windows.h>

#define TRUE 1

#define FALSE 0

#define OK 1

#define ERROR 0

#define INFEASIBLE -1

#define LIST\_INIT\_SIZE 100

#define LISTINCREMRNT 10

typedef int ElemType;

typedef int Status;

typedef int Boolean;

typedef struct

{

ElemType \* elem; //储存空间基地址

int length; // 记录当前链表长度

int listsize; //链表规模

} SqList;

Status InitList(SqList \*L)

{

(\*L).elem = (ElemType\*)malloc(LIST\_INIT\_SIZE\*sizeof(ElemType));

if (!(\*L).elem)

exit(OVERFLOW);

(\*L).length = 0;

(\*L).listsize = LIST\_INIT\_SIZE;

return OK;

}

Status DestroyList(SqList \*L)

{ /\* 操作结果：三元组T被销毁 \*/

free((\*L).elem);

(\*L).elem = NULL;

(\*L).length = 0;

return OK;

}

Status GetEle(SqList L, int i, int \*e)

{

if (i<1 || i>L.length)

return ERROR;

\*e = L.elem[i - 1];

return OK;

}

Status ListLength(SqList L)

{

return L.length;

}

Status ListInsert(SqList \*L, int i, int e)

{

int \*newbase;

int \*p, \*q;

if (i<1 || i>(\*L).length + 1)

return ERROR;

if ((\*L).length>(\*L).listsize)

{

newbase = (ElemType\*)realloc((\*L).elem, ((\*L).listsize + LISTINCREMRNT) \* sizeof(ElemType));

if (!newbase)

exit(OVERFLOW);

(\*L).elem = newbase;

(\*L).listsize += LISTINCREMRNT;

}

q = &((\*L).elem[i - 1]);

for (p = &(\*L).elem[(\*L).length - 1]; p >= q; --p)

{

\*(p + 1) = \*p;

}

\*q = e;

++(\*L).length;

return OK;

}

Status ListDelete(SqList \*L, int i, int \*e)

{

int \*p;

if (i<1 || i>(\*L).length)

return ERROR;

\*e = (\*L).elem[i - 1];

for (p = &(\*L).elem[i - 1]; p<&(\*L).elem[(\*L).length - 2]; p--)

{

\*p = \*(p + 1);

}

(\*L).length--;

return OK;

}

Status SearchMax(SqList L)

{

//ElemType \*e;

ElemType max;

ElemType i = 2;

max = L.elem[0];

while (i <= (L).length)

{

if (max<L.elem[i - 1])

max = L.elem[i - 1];

i++;

}

return max;

}

Status SearchMin(SqList L)

{

//ElemType \*e;

ElemType min;

ElemType i = 2;

min = L.elem[0];

while (i <= (L).length)

{

if (min>L.elem[i - 1])

min = L.elem[i - 1];

i++;

}

return min;

}

void MergeList(SqList La, SqList Lb, SqList \*Lc)

{

InitList(Lc);

int i = 1, j = 1, k = 0;

int La\_length = ListLength(La);

int Lb\_length = ListLength(Lb);

int ai, bj;

while ((i <= La\_length) && (j <= Lb\_length))

{

GetEle(La, i, &ai);

GetEle(Lb, j, &bj);

if (ai <= bj)

{

ListInsert(Lc, ++k, ai);

++i;

}

else

{

ListInsert(Lc, ++k, bj);

++j;

}

}

while (i <= La\_length)

{

GetEle(La, i++, &ai);

ListInsert(Lc, ++k, ai);

}

while (j <= Lb\_length)

{

GetEle(Lb, j++, &bj);

ListInsert(Lc, ++k, bj);

}

}

void main()

{

SqList L1, L2, L3;

//int e;

int i;

int Max, Min;

if (InitList(&L1) == 1 && InitList(&L2) == 1)

{

printf("顺序表初始化成功\n");

}

for (i = 1; i<10; i++)

ListInsert(&L1, i, 2 \* i);

for (i = 1; i<10; i++)

ListInsert(&L2, i, (2 \* i + 1));

Max = SearchMax(L1);

Min = SearchMin(L1);

printf("L1 max=%d,L1 min=%d\n", Max, Min);

Max = SearchMax(L2);

Min = SearchMin(L2);

printf("L2 max=%d,L2 min=%d\n", Max, Min);

MergeList(L1, L2, &L3);

for (i = 1; i <= L1.length; i++)

printf("%d ", L1.elem[i - 1]);

printf("\n");

for (i = 1; i <= L2.length; i++)

printf("%d ", L2.elem[i - 1]);

printf("\n");

for (i = 1; i <= L3.length; i++)

printf("%d ", L3.elem[i - 1]);

system("pause");

return;

}

1.寻找顺序表中的最大值最小值，关键是遍历一次顺序表。先定义一个最大值、最小值变量，把顺序表中第0个元素赋给这个变量。  
2.在找最大值时，从第1个元素开始比较，如果大于MAX，则把这个值赋给MAX。最小值同理，如果小于MIN，则把此值赋给最小值。  
3.在找最大值最小值时，还可以先把这个顺序表从小到大进行排序，然后进行遍历，用t标记最后一个元素的位置，MAX即为最后一个元素，MIN为第0个元素。