1. 问题描述与分析

1.两个有序线性表中的数据元素类型为字符，根据原有的这两个表生成一个新的有序线性表，新表要求包含原表的所有数据元素。如若需要尽量利用原表的空间完成运算。

2.抽象数据类型定义

定义线性表的结构

typedef struct {

ElemType\* elem; //存储空间基址

int length; //当前长度

int listsize; //当前分配的存储容量

} SqList;

3输入值范围

由于要求元素字符类型是char 定义为ElemType 为char ，所以输入的数值为字符

1. 问题解决思路
2. 首先得创建初始化一个空的线性表，故设计了初始化的函数，然后应有输入的函数和输出函数来帮我们验证。以上为基本，题目要求是设计的Merge函数，在我的算法设计中，完成了利用原表的空间完成运算搬运。
3. 算法及算法分析
4. 结构体。C语言的大部分数据结构均匀结构体定义

2.

#include <stdio.h>

#include <stdlib.h>

#define TURE 1

#define FALSE 0

#define OK 1

#define ERROR 0

#define INFEASIBLE -1

#define OVERFLOW -2

#define list\_init\_size 80 //初始分配存储空间

#define listincreament 10 //存储空间分配增量

typedef char ElemType;

//定义线性表的结构

typedef struct {

ElemType\* elem; //存储空间基址

int length; //当前长度

int listsize; //当前分配的存储容量

} SqList;

//构造一个空的线性表（初始化）

int InitList\_Sq(SqList\* L) {

//动态开辟一维数组

L->elem = (char\*)malloc(list\_init\_size \* sizeof(ElemType));

//存储分配失败

if (!L->elem)

exit(OVERFLOW);

//空表长度初始化为0；

L->length = 0;

//存储的初始容量为初始分配空间

L->listsize = list\_init\_size;

return OK;

}

int Input\_Sq(SqList\* L, int n) {

int i;

char \*newbase;

if (n < 0)

return ERROR;

if (n > L->listsize) {

newbase = (char\*)malloc(listincreament \* sizeof(char));

if (!newbase)

exit(OVERFLOW);

L->elem = newbase;

L->listsize += listincreament;

}

printf("请输入元素：\n");

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

scanf(" %c", &L->elem[i]);

L->length++;

}

return OK;

}

void Merge(SqList La, SqList Lb, SqList \*Lc) {

ElemType\* pa = La.elem;

ElemType\* pb = Lb.elem;

int i, j;

i = 1;

j = 1;

ElemType\* pc;

Lc->listsize = Lc->length = La.length + Lb.length;

pc = Lc->elem = (ElemType\*)malloc(Lc->listsize \* sizeof(ElemType));

if (!Lc->elem) exit(OVERFLOW);

ElemType\* pa\_last = La.elem + La.length - 1;

ElemType\* pb\_last = Lb.elem + Lb.length - 1;

while (pa <= pa\_last && pb <= pb\_last) {

if (\*pa < \*pb)

{

\*pc++ = \*pa++;

++i;

}

if (\*pa == \*pb)

{

\*pc++ = \*pa++;

++pb;

Lc->length--;

}

if (\*pa > \* pb) {

\*pc++ = \*pb++;

++j;

}

}

while (pa <= pa\_last)\*pc++ = \*pa++;

while (pb <= pb\_last)\*pc++ = \*pb++;

}

int Output(SqList\*L) {

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

putchar(L->elem[i]);

printf(" ");

}

printf(" \n");

return OK;

}

int main(void) {

printf("有序输入四个元素（字符）的线性表1\n");

SqList ts;

InitList\_Sq(&ts);

printf("请输入四个元素\n");

Input\_Sq(&ts, 4);

Output(&ts);

printf("有序输入四个元素（字符）的线性表2，请输入四个元素\n");

SqList tpps;

InitList\_Sq(&tpps);

Input\_Sq(&tpps, 4);

Output(&tpps);

SqList dc;

InitList\_Sq(&dc);

Merge(ts, tpps, &dc);

printf("合并后如下");

Output(&dc);

return 0;

}

1. 时间复杂度和空间复杂度的分析

Input\_Sq Output 输入和输出函数的时间复杂度均为O（n) 空间复杂度为O（1）

InitList\_Sq

初始化线性表时间复杂度均为O（1) 空间复杂度为O（1）

Merge

归并函数时间复杂度不同情况下待定 空间复杂度为O（n+m）（n,m为两个表的长度）

3.测试数据见截图里





