南昌航空大学实验报告

二0 年 月 日

课程名称： 数据结构 实验名称： 线性表的顺序存储结构

班级学号： 202082 姓名： 王潇 同组人： 无

指导教师评定： 签名：

1. 需求分析

要解决的问题：构造顺序表（以回车符结束），过滤顺序表使顺序表不含重复字符和非小写字母，求两个顺序表的并集，求顺序表的补集。

1. 输入形式与输入值得范围：构造输入：任意非回车符（\n换行符的字符；功能选择输入：整形的1 2 3 4；
2. 输出的形式：以集合的形式输出字符型
3. 功能：构建顺序表，输出顺序表，合并顺序表，求顺序表的补集，过滤顺序表重复与非小写字母的元素
4. 测试数据：

输入数据：（由于输入可任意，没有错误的输入数据）

构建顺序表1：abccdeffg3;

构建顺序表2： efgedasdas;

输出数据：

输出顺序表1：abcdefg ;

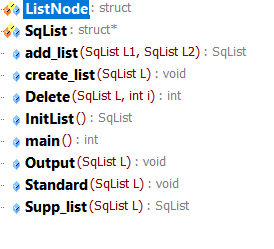
输出顺序表2：efgdas;

顺序表1和2的并集：aebfcg ds；

顺序表1补集：hijklmnopqrstuvwxyz

顺序表2补集：bchijklmnopqrtuvwxyz

1. 概要设计
2. 抽象函数定义：



1. 各函数功能：

add\_list(SqList L1,SqList L2)---顺序表并集

Create\_list(SqList L)---往L中输入数据

Delete（SqList L,int i）---删除L中位置为i的元素

InitList()----构造空的线性表

Output（SqList L）---输出L中数据

Standard(SqList L)---将L中的非法字符（不是小写或则重复的）滤去

Supp\_list(SqList L)----求L的补集

调用关系：

Stardars()函数调用Delete()函数；

add\_list()函数调用InitList()函数；

Supp\_list()函数调用InitList（）函数；

1. 详细设计
2. 伪代码：（这里写了每一个函数的伪代码，标红的是这一函数的作用）

typedef struct ListNode{

char elem[MAXSIZE];

int length;

}\*SqList;

创建空顺序表

SqList InitList(){

SqList L;

L=(SqList)malloc(MAXSIZE\*sizeof(SqList));

L->length=0;

return L;

}

create\_list(SqList L){

scanf("%c",&x);

int i=0;

while(x != '\n'){

L->elem[i++]=x;

L->length++;

scanf("%c",&x);

}

}

删除顺序表中特定位置的元素

Void Delete(SqList L,int i){

if(L->length == 0){

exit("顺序表为空，无法删除");

}

if(i<0||i>(L->length-1)){

exit("删除元素的位置有误");

}

if(i>=0 && i<=L->length-1){

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

L->elem[k] = L->elem[k+1];//元素前移

}

L->length--;

}

}

标准化顺序表（没有非小数元素和重复元素）

void Standard(SqList L){

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

//非法字符

if(L->elem[i]<'a' || L->elem[i]>'z'){

Delete(L,i);//调用

i--;

}

}

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

for(int j=i+1;j<L->length;j++){

if(L->elem[i]==L->elem[j]){

Delete(L,j);

j--;//删除的时候这个位置之后全部往前移一个位置，我们要对下标减一重新便利

}

}

}

}

输出顺序表的元素

void Output(SqList L){

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

printf(L->elem[i]);

}

}

求两个顺序表的并集的顺序表

SqList add\_list(SqList L1,SqList L2){

SqList L3 = InitList();//调用 创建一个新的空顺序表

if(L1->length+L2->length > MAXSIZE){

exit("越界");

}

int i=0,j=0;

while((L1->elem[i]!='\0')&&(L2->elem[i]!='\0')){

L3->elem[j++]=L1->elem[i];

L3->elem[j++]=L2->elem[i];

i++;

L3->length+=2;

}

if(L1->elem[i]='\0'){

for(int j=i;j<L2->length;j++){

L3->elem[j++]=L2->elem[j];

L3->length++;

}

}

if(L2->elem[i]=='\0'){

for(int j=i;j<L1->length;j++){

L3->elem[j++]=L1->elem[j];

L3->length++;

}

}

for(int i=0;i<L3->length;i++){ //删除重复元素

for(int j=i+1;j<L3->length;j++){

if(L3->elem[j]==L3->elem[i]){

Delete(L3,j);

j--;

}

}

}

return L3;

}

求顺序表的补集

SqList Supp\_list(SqList L){

x='a';//创建一个全集顺序表

L3 = InitList();

for(int i=0;i<26;i++){

L3->elem[i]=x;

x++;

L3->length++;

}

for(int j=0;j<L->length;j++){//求L的补集

for(int k=0;k<26;k++){

if(L3->elem[k] == L->elem[j]){

Delete(L3,k);

k--;

}

}

}

return L3;

}

1. 各函数功能：（1）中伪代码部分标红处

调用关系：

Standard()函数调用Delete()函数与InitList()函数；

add\_list()函数调用InitList()函数与InitList（）函数；；

Supp\_list()函数调用Delete()函数

# 调试分析

(1)问题：

Standard（）与add\_list()函数的预期结果与实际结果有误差；

问题解决：

通过调试并且多次输入不同的值发现发现异常字符删除后，下标没有前移一次而是继续遍历导致有些字符会跳过；

解决办法：发现异常并且调用Delete()函数后，将下标前移一次；

（2）问题：在Initlist()函数中，开始没有去分配地址，而且编译也不会报错，但是编译后与预期不一样。

问题分析：在定义结构体中是char elem[max],应该每定义一次会自动分配内存。并且经过调试到哪里发现内存空间分配错误。去尝试着手动的分配地址给他

解决办法：手动分配地址

1. 部分函数实践复杂度分析：

Standard（）

函数中过滤重复字符和非法字符在一个循环中会降低实践复杂度（由原来的O(N\*2+n)降低为O(N\*2)）

Add\_list()

函数中对两个顺序表和过滤重复字符串可以直接在同一个便利里进行，边便利边过滤重复在加入到一个新的空顺序表中。

（4）心得：过去我不会使用debug,但程序一旦复杂起来很难观察出来，使用调试可以更加清晰你的思路并且发现你思路的漏洞并且找出程序错误。

# 用户说明

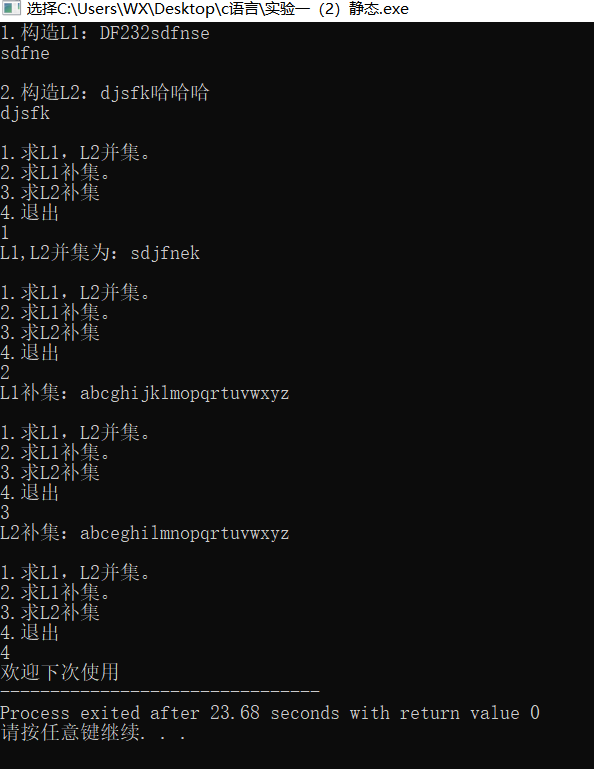
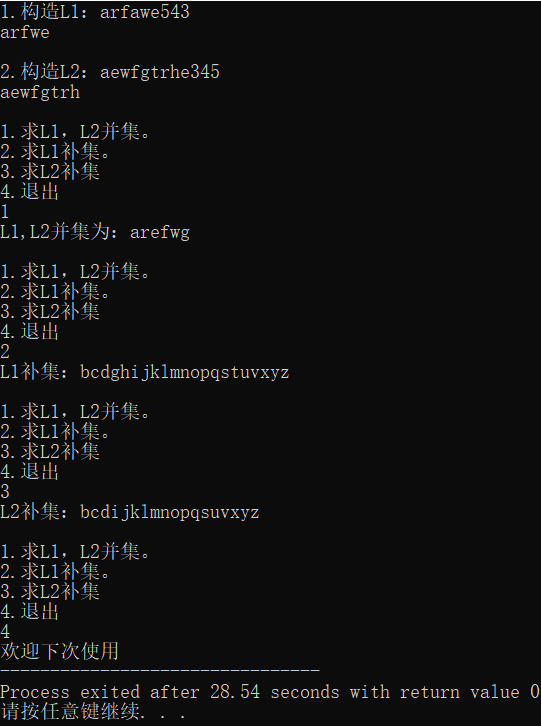
构造L1，构造L2：

可任意输入，换行（回车键）确定，每一次输入后的输出是按要求过滤后的顺序表元素；

选择功能：

按提示输入1，2，3，4后按回车键，当按其他键或报错，这是你只需按提示再次输入一次选项；

# 测试结果（这里我用运行截图。）



# 附录

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#define MAXSIZE 100

typedef struct ListNode{

char \*elem;

int length;

int listsize;

}\*SqList;

//初始化

SqList create\_list(){

SqList L;

L=(SqList)malloc(sizeof(struct ListNode));

L->length=0;

return L;

}

//插入

int ListInsert(SqList L,int i,char e){

int k;

if(L->length==MAXSIZE){

return 0;

}

if(i<1||i>L->length+1){

return 0;

}

if(i<=L->length){

for(k=L->length-1;k>=i-1;k--){

L->elem[k+1]=L->elem[k];

}

}

L->elem[i-1]=e;

L->length++;

return 1;

}

//删除顺序表指定位置的元素

int Delete(SqList L,int i){

if(L->length == 0){

printf("顺序表为空，无法删除");

return 0;

}

if(i<1||i>(L->length)){

printf("删除元素的位置有误");

return 0;

if(i>=1 && i<=L->length){

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

L->elem[k-1] = L->elem[k];//元素前移

}

L->length--;

}

}

}

//自动滤去重复或者不法字符

void filter\_list(SqList L){

int i,j;

for(i=0;i<L->length;i++){//b不法字符

if(L->elem[i]<'A' || L->elem[i]>'Z');

Delete(L,i);

}

for(i=1;i<L->length;i++){

for(j=0;j<i;j++){//重复字符

if(L->elem[i] == L->elem[j]){

Delete(L,i);

}

}

}

}

//两顺序表并

SqList add\_twolist(SqList L1,SqList L2){

SqList L3=create\_list();

int k=0;

while(k<(L1->length+L2->length)){

for(int i=0;i<L1->length && i<L2->length;i++){

ListInsert(L3,++k,L1->elem[i]);

ListInsert(L3,++k,L1->elem[i]);

}

if(k<L1->length){

for(int j=k;k<L1->length;j++){

ListInsert(L3,++k,L1->elem[j]);

}

}

if(k<L2->length){

for(int j=k;k<L2->length;j++){

ListInsert(L3,++k,L2->elem[j]);

}

}

}

return L3;

}

//求补集

SqList Supp\_set(SqList L){

SqList L2 = create\_list();

char x='a';//构造一个全集出来

int i=1;

while('A' <= x && x <= 'Z') {

ListInsert(L2,++i,++x);

}

for(int j=0;j<L->length;j++){//求L的补集

for(int k=0;k<26;k++){

if(L2->elem[k] == L->elem[j]){

Delete(L2,++k);

}

}

}

return L2;

}

int main(){

int x;

char e;

printf("1.建立L1顺序表");

SqList L1 = create\_list();

printf("请顺序表的长度：");

scanf("%d",&x);

printf("请输入各字母：");

for(int i=1;i<=x;i++){

scanf("%c ",&e);

ListInsert(L1,i,e);

}

filter\_list(L1);

printf("2.建立L2顺序表");

SqList L2 = create\_list();

printf("请顺序表的长度：");

scanf("%d",&x);

printf("请输入各字母：");

for(int i=1;i<=x;i++){

scanf("%c ",&e);

ListInsert(L2,i,e);

}

filter\_list(L2);

int n;

do{

printf("1.L1,L2的并集\n2.L1的补集\n3.L2的补集\n4.退出");

scanf("%d",&n);

switch(n){

case 1:{

add\_twolist(L1,L2);

break;

}

case 2:{

SqList L1\_supp = Supp\_set(L1);

break;

}

case 3:{

SqList L2\_supp = Supp\_set(L2);

break;

}

case 4:{

printf("欢迎下次使用");

break;

}

default:printf("输入错误！");

}

}while(n!=4);

}

1. 本次代码是静态形式的，一次性将要的内存给分配好
2. 编译器是Dev-c++;
3. 代码中部分内容参照了《数据结构（c语言版）》和csdn中其他博主分享