数据结构上机实验题报告

题目：集合的并、交和差运算

姓名：王小龙　　班级：2020211310　　学号：2020211502　　提交日期：2021.11.10

一　题目

编制了一个能执行演示集合的并、交和差运算的程序。

输入两个集合和其中的元素，输出这两个集合的交集，并集,差集和补集。

二　程序设计

使用含有 next 指针和 char 字符的结构体

下面分析算法

Creatlist(LinkList& L)//创建集合　　时间复杂度 o(n),空间复杂度 o(n)

Print_Sq(SqList L)//输出函数　　时间复杂度 o(n),空间复杂度 o(1)

Insertlist(LinkList& L, char c)//插入函数 时间复杂度 o(n),空间复杂度 o(1)

Sortlist(LinkList& L)//排序函数

时间复杂度 o(n2),空间复杂度 o(1)

bing(SqList La,SqList Lb,SqList &Lc)//求集合的并集

时间复杂度 o(n2),空间复杂度 o(n)

jiao(SqList La,SqList Lb,SqList &Lc)//求两个集合的交集

时间复杂度 o(n2),空间复杂度 o(n)

cha(SqList La,SqList Lb,SqList &Lc)//求两个集合的差集

时间复杂度 o(n2),空间复杂度 o(n)

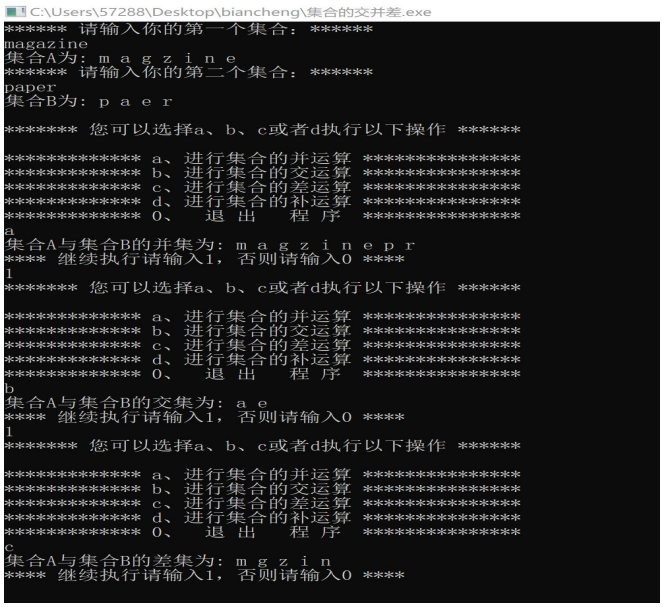
buji(SqList La,SqList Lb,SqList &Lc,SqList &Ld)//求第一个集合对第二个集合的补集

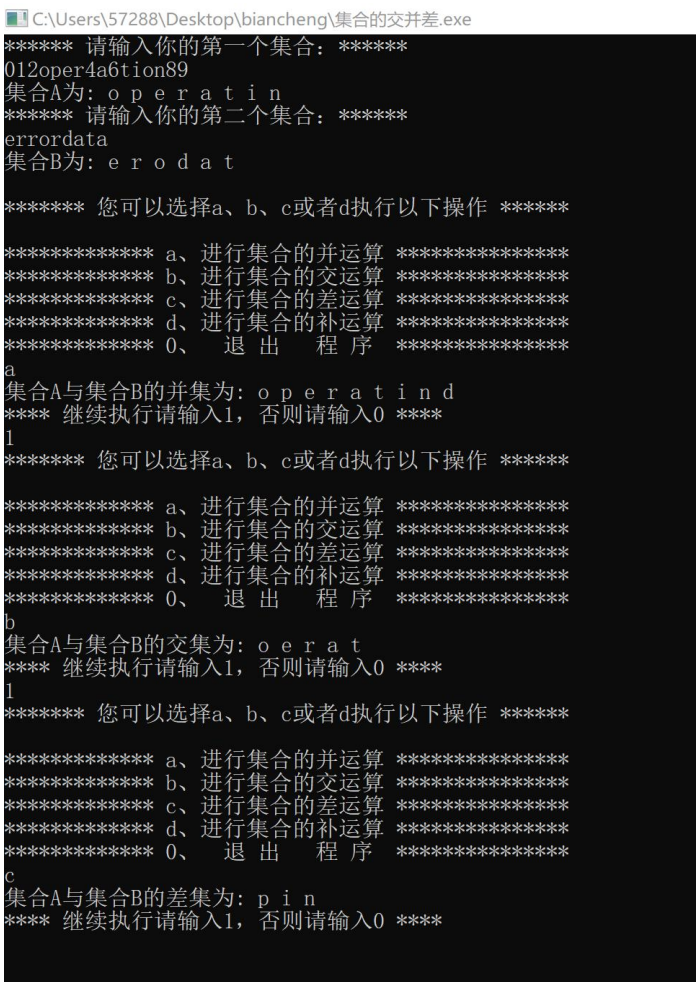
时间复杂度 o(n2),空间复杂度 o(n)

总体时间复杂度 o(n2),空间复杂度 o(n)

三　程序测试运行

编译和运行环境为 vc++6.0;





四　编程工作总结:

本次实验使我巩固了以前的知识并在此基础上还对数据结构的特点和算法有了更深的理解，使我在这门课程的实际应用上也有了一个提高。

五　程序源代码（需 c++运行）

```
#include<stdio.h>
#include<stdlib.h>
#define TRUE 1
#define FALSE 0
#define OK 1
#define ERROR 0
#define OVERFLOW -1
#define LISTSIZE 100 //初始表空间大小
#define addlength 20 //表长增量
typedef int Status;
typedef char ElemType;
typedef struct{
ElemType *elem;
int length;
int listsize;
}SqList;
```

```
SqList La,Lb,Lc,Ld;

Status InitList_Sq(SqList &L){
    L.elem = (ElemType *)malloc(LISTSIZE * sizeof(ElemType));
    if(!L.elem) exit(-1);
    L.length = 0;
    L.listsize = LISTSIZE;
    return OK;
}
Status ListInsert_Sq(SqList &L,int i,ElemType e){
    ElemType *newbase,*p,*q;
    if(i < 1 || i > L.length + 1)
        return ERROR;
    if(L.length >= L.listsize){
        newbase = (ElemType *)realloc(L.elem,(L.listsize +addlength) * sizeof(ElemType));
    if(!newbase) exit(-1);
    L.elem = newbase;
    L.listsize +=addlength;
    }
    q = &(L.elem[i - 1]);
    for(p = &(L.elem[L.length - 1]); p >= q; --p)
    *(p + 1) = *p;
    *q = e;
    L.length++;
    return OK;
}
void CreateList_Sq(SqList &L){
    ElemType ch;
    int n=0,j;
    while((ch) != '\n'){
        scanf("%c",&ch);
        for(j = 0; j < L.length; j++)
        if(ch == L.elem[j]){
            n=1;
            break;
        }
        else
            n=0;
            if(!n && ch != '\n')
            ListInsert_Sq(L,L.length+1,ch);
    }
}
```

```
Status Equal(ElemType a,ElemType b){
	if(a == b)
	return 1;
	else
	return 0;
}

int LocateElem_Sq(SqList L,ElemType e,Status(* compare)(ElemType,ElemType)){
	ElemType *p;
	int i;
	i = 1;
	p = L.elem;
	while(i <= L.length && !(* compare)(*p++,e)) ++i;
	if(i <= L.length) return i;
	else return 0;
} //该函数的时间复杂度为 O(n)


void Print_Sq(SqList L){
	int i;
	for(i = 0; i < L.length; i++)
	{
	if(L.elem[i]>='a'&&L.elem[i]<='z')
		{
			printf("%2c",L.elem[i]);
		}
	}
	if(L.length == 0)
		printf("该集合为空集");

}

/**求集合的并集的函数**/
void bing(SqList La,SqList Lb,SqList &Lc){
	int i;
	ElemType elem;
	Lc.length=0;
	for(i = 0; i < La.length; i++)
	Lc.elem[Lc.length++]=La.elem[i];
	for(i = 1; i <= Lb.length; i++){
		elem = Lb.elem[i-1];
		if(!LocateElem_Sq(La,elem,Equal))
		ListInsert_Sq(Lc,Lc.length+1,elem);
    }
```

```
}

/**求集合的交集的函数**/
void jiao(SqList La,SqList Lb,SqList &Lc){
	int i;
	ElemType elem;
	Lc.length = 0;
	for(i = 1; i <= La.length; i++){
		elem = La.elem[i-1];
		if(LocateElem_Sq(Lb,elem,Equal))
		ListInsert_Sq(Lc,Lc.length+1,elem);
    }
}

/**求集合的差集函数**/
void cha(SqList La,SqList Lb,SqList &Lc){
	int i;
	ElemType elem;
	Lc.length = 0;
	for(i = 1; i <= La.length; i++){
		elem = La.elem[i-1];
		if(!LocateElem_Sq(Lb,elem,Equal))
		ListInsert_Sq(Lc,Lc.length+1,elem);
    }
}

/**求集合的补集函数**/
void buji(SqList La,SqList Lb,SqList &Lc,SqList &Ld){
	int i;
	ElemType elem;
	Ld.length = 0;
	bing(La,Lb,Lc);
	for(i = 1; i <= Lc.length; i++)
	{
		elem = Lc.elem[i-1];
		if(!LocateElem_Sq(La,elem,Equal))
		ListInsert_Sq(Ld,Ld.length+1,elem);
	}
}

void select(){
	char s;
	int l=1;
	InitList_Sq(La);
```

```
printf("****** 请输入你的第一个集合： ******\n");

CreateList_Sq(La);
printf("集合 A 为:");
Print_Sq(La); //实现表 LA 的操作
printf("\n");
InitList_Sq(Lb);
printf("****** 请输入你的第二个集合： ******\n");
CreateList_Sq(Lb);
printf("集合 B 为:");
Print_Sq(Lb); //实现表 LB 的操作
printf("\n\n");

InitList_Sq(Lc); //初始化表 LC 的操作
InitList_Sq(Ld); //初始化表 Ld 的操作
while(l){
printf("******* 您可以选择 a、b、c 或者 d 执行以下操作 ******\n\n");
printf("************* a、进行集合的并运算 ***************\n");
printf("************* b、进行集合的交运算 ***************\n");
printf("************* c、进行集合的差运算 ***************\n");
printf("************* d、进行集合的补运算 ***************\n");
printf("************* 0、  退 出  程 序 ***************\n");

scanf("%c",&s);
switch(s){
    case 'a':bing(La,Lb,Lc);
    printf("集合 A 与集合 B 的并集为:");
    Print_Sq(Lc); //实现表 LA 与表 LB 并集的操作
    printf("\n");
    break;
    case 'b' : jiao(La,Lb,Lc);
    printf("集合 A 与集合 B 的交集为:");
    Print_Sq(Lc); //实现表 LA 与表 LB 交集的操作
    printf("\n");
    break;
    case 'c' : cha(La,Lb,Lc);
    printf("集合 A 与集合 B 的差集为:");
    Print_Sq(Lc); //实现表 LA 与表 LB 差集的操作
    printf("\n");
    break;
    case 'd' : buji(La,Lb,Lc,Ld);
    printf("集合 A 的补集为:");
    Print_Sq(Ld); //实现表 LA 的补集操作
    printf("\n");
```

```
            break;
            case 'e' : exit(0);
            break;
            default    : printf("tenter data error!\n");
            printf("\n");
            }
            printf("****  继续执行请输入 1，否则请输入 0 ****\n");
            scanf("%d",&l);
            getchar();
        }
}

int main()
{
        select();
        return 0;
}
```