# 实验一

## 1. 问题定义及需求分析

### 1.1 问题描述

##### 集合的并交差运算

##### 设计一个能演示集合的并、交、差运算程序。

### 1.2 实验要求

##### （1）采用顺序表或链表等数据结构。

##### （2）集合的元素限定为数字和小写英文字母。

### 1.3 输入

#### 1.3.1 输入的形式

##### 外部输入字符串。

#### 1.3.2 输入值的范围

##### 数字和小写英文字母。

### 1.4 输出形式

##### 字符集。

### 1.5 程序的功能

##### 计算两个集合的交、并、差。

### 1.6 测试数据

#### 1.6.1 输入

##### ab1

##### ac2

#### 1.6.2 输出

##### 集合A：a b 1

##### 集合B：a c 2

##### 集合A并B：a b 1 c 2

##### 集合A交B：a

##### 集合A-B：c 2

## 2.概要设计

### 2.1 抽象数据类型定义

##### ADT List

##### {

##### Data 字符类型；

##### 数据元素操作void init(SqList\* s,char\* str)//将字符串存进线性表内

##### int exist(SqList\* s,char c)//判断c是否在S的数据内

##### int add(SqList\* s,char c)//向线性表S内加入c字符

##### void print(SqList\* s)//打印S内的数据元素

##### void get\_union(SqList\* a,SqList\* b,SqList\* t)//求并集

##### void get\_intersection(SqList\* a,SqList\* b,SqList\* t)//求交集

##### void get\_diff(SqList\* a,SqList\* b,SqList\* t)//求集合差

##### }

### 2.2 主程序流程

### 2.3 模块调用关系

## 3.详细设计

### 3.1 定义数据类型及存储结构

typedef struct  
{  
 char Data[max\_element\_count];//把集合用字符串存起来  
 int count;//储存这个集合里元素的个数  
}SqList;

### 3.2 每个函数及操作的代码

void init(SqList\* s,char\* str) //将字符串存进线性表内  
{  
 s->count=0; //初始化长度为0  
 for(int i=0;i<strlen(str);++i)  
 {  
 if(!exist(s,str[i])) //判断元素是否已存储在集合内  
 add(s,str[i]); //添加元素  
 else  
 printf("\nERROR: duplicate element:%c\n",str[i]);  
 }  
}  
int exist(SqList\* s,char c)  
{  
 for(int i=1;i<=s->count;++i) //遍历顺序表中元素  
 {  
 if(s->Data[i]==c)   
 return 1; //已存在则返回0  
 }  
 return 0;  
}  
int add(SqList\* s,char c)  
{  
 if(s->count+1>=max\_element\_count) //判断顺序表是否已满  
 return ERROR;  
 ++s->count; //表长加一  
 s->Data[s->count]=c; //插入元素  
 return OK;  
}  
void print(SqList\* s)  
{  
 printf("\n集合中共有%d个元素：\n ",s->count);  
 for(int i=1;i<=s->count;++i)  
 printf("%c ",s->Data[i]);  
 printf("\n");  
}  
void get\_union(SqList\* a,SqList\* b,SqList\* t)  
{  
 init(t,""); //初始化一个空的集合  
 for(int i=1;i<=a->count;++i)   
 add(t,a->Data[i]); //将集合a中所有元素加入  
 for(int i=1;i<=b->count;++i)  
 {  
 if(!exist(t,b->Data[i]))   
 add(t,b->Data[i]); //如果b中元素不重复则加入  
 }  
}  
void get\_intersection(SqList\* a,SqList\* b,SqList\* t)  
{  
 init(t,"");  
 for(int i=1;i<=a->count;++i) //遍历a中元素  
 {  
 if(exist(b,a->Data[i])) //如果b中也存在  
 add(t,a->Data[i]); //加入新集合  
 }  
}  
void get\_diff(SqList\* a,SqList\* b,SqList\* t)  
{  
 init(t,"");  
 for(int i=1;i<=a->count;++i) //遍历a中元素   
 {  
 if(!exist(b,a->Data[i])) //如果b中不存在  
 add(t,a->Data[i]); //加入新集合  
 }  
}

## 4.调试分析

### 4.1 遇到的问题及分析

#### 4.1.1 问题

##### 线性表长度可能不够，导致溢出。

#### 4.1.2 分析

##### 初始开辟的data数组足够大，或者使用动态内存。

### 4.2 算法时空分析

#### 4.2.1 时间

##### 时空分析：

##### void init(SqList\* s,char\* str)时间复杂度为O(n);

##### int exist(SqList\* s,char c) 时间复杂度为O(n);

##### int add(SqList\* s,char c)时间复杂度为O(n);

##### void print(SqList\* s)时间复杂度为O(n);

##### void get\_union(SqList\* a,SqList\* b,SqList\* t)时间复杂度为O(m\*n);

##### void get\_intersection(SqList\* a,SqList\* b,SqList\* t)时间复杂度为O(m\*n);

##### void get\_diff(SqList\* a,SqList\* b,SqList\* t) 时间复杂度为O(m\*n);

#### 4.2.2 空间

##### 空间复杂度是O(n)，n为A、B中元素个数总和。

#### 4.2.3 改进

##### 求交、并、差运算时都需要遍历集合内所有元素，复杂度较高，可以使用二叉平衡树将复杂度降低至O(logn)。

### 4.3 经验和体会

##### 顺序表使用起来比较简单，但长度不可随意变化，适用于大量访问元素，而不适用于大量增添和删除元素；和链表相比，顺序表在内存中存储地址连续。

## 5.使用说明

##### 第一步：根据提示输入集合A

##### 第二步：根据提示输入集合B

##### 第三步：程序自动输出集合A、B，AB的并集、交集、差集。

## 6.测试结果

## 

## 7.附录

### 7.1 个人负责的部分

##### 主函数、接口函数、集合的差集。

### 7.2 整个程序

#### 7.2.0 makefile

main : main.c sqlist.c  
 gcc -Wall -O2 -o main.bin main.c sqlist.c

#### 7.2.1 universal\_header.h

#ifndef UNIVERSAL\_H  
#define UNIVERSAL\_H  
  
#include<stdio.h>  
#include<stdlib.h>  
#include<string.h>  
#define max\_element\_count 100  
#define max\_str\_length 10  
#define ERROR 1  
#define OK 0  
  
#endif //UNIVERSAL\_H

#### 7.2.2.1 main.h

#ifndef MAIN\_H  
#define MAIN\_H  
  
#include"universal\_header.h"  
#include"sqlist.h"  
  
#endif //MAIN\_H

#### 7.2.2.2 main.c

int main()  
{  
 char tmp[max\_element\_count];  
 SqList set\_a,set\_b;  
 printf("\n----------输入----------\n");  
 printf("\n请输入集合A中的元素（每个元素为一个数字或小写字母，元素之间不加空格）：");  
 scanf("%s",tmp);  
 init(&set\_a,tmp);  
 printf("\n请输入集合B中的元素（每个元素为一个数字或小写字母，元素之间不加空格）：");  
 scanf("%s",tmp);  
 init(&set\_b,tmp);  
  
 printf("\n----------输出----------\n");  
 printf("\n集合A：");  
 print(&set\_a);  
 printf("\n集合B：");  
 print(&set\_b);  
 SqList set\_t;  
 get\_union(&set\_a,&set\_b,&set\_t);  
 printf("\n集合A并B：");  
 print(&set\_t);  
 get\_intersection(&set\_a,&set\_b,&set\_t);  
 printf("\n集合A交B：");  
 print(&set\_t);  
 get\_diff(&set\_a,&set\_b,&set\_t);  
 printf("\n集合A-B：");  
 print(&set\_t);  
 printf("\n-----------完成----------\n");  
  
 return 0;  
}

#### 7.2.3.1 sqlist.h

#ifndef SQLIST\_H  
#define SQLIST\_H  
  
#include"universal\_header.h"  
typedef struct  
{  
 char Data[max\_element\_count];  
 int count;  
}SqList;  
void init(SqList\*,char\*);  
int exist(SqList\*,char);  
int add(SqList\*,char);  
void print(SqList\*);  
void get\_union(SqList\*,SqList\*,SqList\*);  
void get\_intersection(SqList\*,SqList\*,SqList\*);  
void get\_diff(SqList\*,SqList\*,SqList\*);  
  
#endif //SQLIST\_H

#### 7.2.3.2 sqlist.c

#include"sqlist.h"  
void init(SqList\* s,char\* str) //将字符串存进线性表内  
{  
 s->count=0; //初始化长度为0  
 for(int i=0;i<strlen(str);++i)  
 {  
 if(!exist(s,str[i])) //判断元素是否已存储在集合内  
 add(s,str[i]); //添加元素  
 else  
 printf("\nERROR: duplicate element:%c\n",str[i]);  
 }  
}  
int exist(SqList\* s,char c)  
{  
 for(int i=1;i<=s->count;++i) //遍历顺序表中元素  
 {  
 if(s->Data[i]==c)   
 return 1; //已存在则返回0  
 }  
 return 0;  
}  
int add(SqList\* s,char c)  
{  
 if(s->count+1>=max\_element\_count) //判断顺序表是否已满  
 return ERROR;  
 ++s->count; //表长加一  
 s->Data[s->count]=c; //插入元素  
 return OK;  
}  
void print(SqList\* s)  
{  
 printf("\n集合中共有%d个元素：\n ",s->count);  
 for(int i=1;i<=s->count;++i)  
 printf("%c ",s->Data[i]);  
 printf("\n");  
}  
void get\_union(SqList\* a,SqList\* b,SqList\* t)  
{  
 init(t,""); //初始化一个空的集合  
 for(int i=1;i<=a->count;++i)   
 add(t,a->Data[i]); //将集合a中所有元素加入  
 for(int i=1;i<=b->count;++i)  
 {  
 if(!exist(t,b->Data[i]))   
 add(t,b->Data[i]); //如果b中元素不重复则加入  
 }  
}  
void get\_intersection(SqList\* a,SqList\* b,SqList\* t)  
{  
 init(t,"");  
 for(int i=1;i<=a->count;++i) //遍历a中元素  
 {  
 if(exist(b,a->Data[i])) //如果b中也存在  
 add(t,a->Data[i]); //加入新集合  
 }  
}  
void get\_diff(SqList\* a,SqList\* b,SqList\* t)  
{  
 init(t,"");  
 for(int i=1;i<=a->count;++i) //遍历a中元素   
 {  
 if(!exist(b,a->Data[i])) //如果b中不存在  
 add(t,a->Data[i]); //加入新集合  
 }  
}