实验内容：集合的并、交和算差运算

基本内容：

编写一个能完成集合的交、并和差运算的程序。要求:

(1)集合的元素限定为小写字母字符 [‘a’..’z’]。

(2)分别用数组和单向链表表示集合。

选作内容:

(1)集合的元素判定和子集判定运算。

(2)求集合的补集。

(3)集合的混合运算表达式求值。

(4)集合的元素类型推广到其他类型，甚至任意类型。

代码：

#define \_CRT\_SECURE\_NO\_WARNINGS 1

#include <stdio.h>

#include <stdlib.h>

#define MAX 26

#define TYPE char //元素类型

typedef struct Node

{

TYPE data;

struct Node\* next;

}Node, \*Setlist;

void printflist(Setlist set)

{

Node\* p = set;

while(p!=NULL)

{

printf("%c ", p->data);

p = p -> next;

}

}

void initlist(Setlist\* set)

{

\*set = NULL;

}

void addtolist(Setlist \*set, TYPE elem)

{

Node\* newNode = (Node\*)malloc(sizeof(Node));

newNode->data = elem;

newNode->next = NULL;

if (\*set== NULL)

{

\*set = newNode;

}

else

{

Node\* tail = \*set;

while (tail->next != NULL)

{

tail = tail->next;

}

tail->next = newNode;

}

}

void deletefromlist(Setlist\* set, TYPE elem) {

Node\* save = NULL;

Node\* p = \*set;

while (p != NULL)

{

if (p->data == elem)

{

if (save == NULL)

{

\*set = p->next;

}

else

{

save->next = p->next;

}

free(p);

return;

}

save = p;

p = p->next;

}

}

int searchinlist(Setlist set, TYPE elem)

{

Node\* p = set;

while (p != NULL)

{

if (p->data == elem)

{

return 1;

}

p = p->next;

}

return 0;

}

Setlist intersectionlist(Setlist set1, Setlist set2)

{

Setlist result;

initlist(&result);

Node\* p = set1;

while (p != NULL)

{

if (searchinlist(set2, p->data))

{

addtolist(&result, p->data);

}

p = p->next;

}

return result;

}

Setlist unionlist(Setlist set1, Setlist set2)

{

Setlist result;

initlist(&result);

Node\* p = set1;

while (p != NULL)

{

addtolist(&result, p->data);

p = p->next;

}

p = set2;

while (p != NULL)

{

if (!searchinlist(result, p->data))

{

addtolist(&result, p->data);

}

p = p->next;

}

return result;

}

Setlist differenceList(Setlist set1, Setlist set2)

{

Setlist result;

initlist(&result);

Node\* p = set1;

while (p != NULL)

{

if (!searchinlist(set2, p->data))

{

addtolist(&result, p->data);

}

p = p->next;

}

return result;

}

void initarray(TYPE set[])

{

int i;

for (i = 0; i < MAX; i++)

{

set[i] = 0;

}

}

void addtoarray(TYPE set[], TYPE elem)

{

set[elem - 'a'] = 1;

}

void deletefromarray(TYPE set[], TYPE elem)

{

set[elem - 'a'] = 0;

}

int searchinarray(TYPE set[], TYPE elem)

{

return set[elem - 'a'];

}

void printfarray(TYPE set[])

{

int i;

for (i = 0; i < MAX; i++)

{

if (set[i])

{

printf("%c ",'a'+i);

}

}

}

void intersectionarray(TYPE set1[], TYPE set2[], TYPE result[])

{

int i;

for (i = 0; i < MAX; i++)

{

result[i] = set1[i] && set2[i];

}

}

void unionarray(TYPE set1[], TYPE set2[], TYPE result[])

{

int i;

for (i = 0; i < MAX; i++)

{

result[i] = set1[i] || set2[i];

}

}

void differencearray(TYPE set1[], TYPE set2[], TYPE result[])

{

int i;

for (i = 0; i < MAX; i++)

{

result[i] = set1[i] && !set2[i];

}

}

int main()

{

Setlist listset1, listset2, listresult;

TYPE arrayset1[MAX], arrayset2[MAX], arrayresult[MAX];

initlist(&listset1);//初始化链表

initlist(&listset2);

initarray(&arrayset1);//初始化数组

initarray(&arrayset2);

printf("Please enter the number of elements in set 1 (linked list)：");//加上元素

int a;

TYPE c;

scanf("%d",&a);

for (int i = 1; i <= a; i++)

{

printf("The %d element is：", i);

scanf(" %c", &c);

addtolist(&listset1,c);

printf("\n");

}

printf("Please enter the number of elements in set 2 (linked list)：");

TYPE d;

int b;

scanf("%d", &b);

for (int i = 1; i <= b; i++)

{

printf("The %d element is：", i);

scanf(" %c", &d);

addtolist(&listset2,d);

printf("\n");

}

printf("listset1:");//打印链表1

printflist(listset1);

printf("\n");

printf("listset2:");//打印链表2

printflist(listset2);

printf("\n");

listresult = intersectionlist(listset1, listset2);//交集

printf("List intersection result:");

printflist(listresult);

printf("\n");

free(listresult);

listresult = unionlist(listset1, listset2);//并集

printf("List union result:");

printflist(listresult);

printf("\n");

free(listresult);

listresult = differenceList(listset1, listset2);//差集

printf("List difference result:");

printflist(listresult);

printf("\n");

free(listresult);

printf("Please enter the number of elements in set 1 (arraylist)：");//加上元素

int k;

TYPE e;

scanf("%d", &k);

for (int i = 1; i <= k; i++)

{

printf("The %d element is：", i);

scanf(" %c", &e);

addtoarray(&arrayset1, e);

printf("\n");

}

printf("Please enter the number of elements in set 2 (arraylist)：");

TYPE f;

int j;

scanf("%d", &j);

for (int i = 1; i <= j; i++)

{

printf("The %d element is：", i);

scanf(" %c", &f);

addtoarray(&arrayset2, f);

printf("\n");

}

printf("arrayset1:");

printfarray(arrayset1);

printf("\n");

printf("arrayset2:");

printfarray(arrayset2);

printf("\n");

intersectionarray(arrayset1, arrayset2, arrayresult);//j交集

printf("array intersection result:");

printfarray(arrayresult);

printf("\n");

unionarray(arrayset1, arrayset2, arrayresult);//并集

printf("array union result:");

printfarray(arrayresult);

printf("\n");

differencearray(arrayset1, arrayset2, arrayresult);//cha差集

printf("array difference result:");

printfarray(arrayresult);

printf("\n");

return 0;

}

分析：

1. 若要改变元素的类型在开头定义修改即可。
2. 代码的拓展性强，补集等其他运算通过添加函数即可。
3. 函数以及其他各种的变量的命名容易分辨，不会看了前面，忘了后面。

不足：

1. 在函数的后面我只释放了listresult的空间，其他链表的空间我没有释放。

简单加个指针遍历释放即可。

1. 没有过多的注释，但是各个函数名可表达函数的主体意思即可。
2. 每次运行只能运算一次，若想多次运行计算可使用while和switch函数。