

吴老师，骆学长和孙学长好，以下是我的作业。

-教材

P.186 题 15: 链表实现集合运算。要求: 不使用辅助表元, 实现功能函数 1. void **unionSet**(struct List \*s1, struct List \*s2, struct List \*result); 2. void **differenceSet**(struct List \*s1, struct List \*s2, struct List \*result); 3. void **intersectionSet**(struct List \*s1, struct List \*s2, struct List \*result)。测试要求: S1 = {2, 3, 5, 6}, S2 = {3, 4, 6, 8}。

- 上机指导书

实验七 6 思考题: 理论题。

- 补充题:

创建链表, 链表中的结点值依次为“10 20 30 40 50”, **Delete** 函数用于删除链表中所有值大于 min 且小于 max 的结点(表元), 同时释放被删结点空间。如设定 min=10 和 max=35, Delete 函数将使得链表中结点值被删为“10 40 50”。

[Chi-Shan0707/Homework-in-CS10004-Programming-by-yhchi](#)

代码仓库 ↑

T1.

核心思想: 不断比较, 不厌其烦地比较

```
#include <stdio.h>
#include <stdlib.h>

// 单向链表节点
struct Node
{
    int val;
    struct Node* next;
};

typedef struct Node Node;
struct List
{
    Node* lsthead;
};

typedef struct List List;
void convert(List* lst, int arr[], int n)
{
    lst->lsthead = NULL;
    if (n <= 0) return;
    Node *head = (Node *) malloc ( sizeof (Node) );
    head->val = arr[0];
    head->next = NULL;
    lst->lsthead = head;
```

```

Node* tail = head;
for (int i = 1; i < n; ++i)
{
    Node* cur = (Node*)malloc(sizeof(Node));
    cur->val = arr[i];
    cur->next = NULL;
    tail->next = cur;
    tail = cur;
}
}

void free_list(List* lst)
{
//记得整体清空

Node* head = lst->lsthead;
while(head)
{
    Node *tmp = head;
    head = head->next;
    free(tmp);
}
lst->lsthead = NULL;
//记得置空
}

void print_list(const char* name, List* lst)
{
printf("set %s = {\n", name);
Node* head = lst->lsthead;
int is_firstitem = 1;
while (head)
{
    if (!is_firstitem) printf(" , ");
    printf("%d", head->val);
    is_firstitem = 0;
    head = head -> next;
}
printf("}\n");
}

void append(List *lst, Node **tail_ptr,int val)
{
Node* new_node = (Node*)malloc(sizeof(Node));
new_node->val = val;
new_node->next = NULL;
if (lst->lsthead == NULL)
{
    lst->lsthead=new_node;
}
else
{
}
}

```

```

        (*tail_ptr)->next=new_node;
    }
    *tail_ptr = new_node;
}
// 2. 差集 S1 \ S2
void differenceSet(struct List *s1, struct List *s2, struct List *res) {

    Node* head1 = s1->lsthead;
    Node* head2 = s2->lsthead;
    Node* tail = NULL;
    free_list(res);
    res->lsthead = NULL;
    while(head1!=NULL)
    {
        int exist_in_s2=0;
        for(head2=s2->lsthead; head2!=NULL; head2=head2->next)
        {
            if(head2->val==head1->val)
            {
                exist_in_s2=1;
                break;
            }
        }
        if(!exist_in_s2)append(res, &tail, head1->val);
        head1=head1->next;
    }
}

void intersectionSet(struct List *s1, struct List *s2, struct List *res)
{
    Node* head1 = s1->lsthead;
    Node* head2 = s2->lsthead;
    Node* tail = NULL;
    free_list(res);
    res->lsthead = NULL;
    while(head1!=NULL)
    {
        int exist_in_both=0;
        for(head2=s2->lsthead; head2!=NULL; head2=head2->next)
        {
            if(head2->val==head1->val)
            {
                exist_in_both=1;
                break;
            }
        }
        if(exist_in_both)append(res, &tail, head1->val);
        head1=head1->next;
    }
}

```

```
}
```

```
void unionSet(struct List *s1, struct List *s2, struct List *res)
{
    Node* head1 = s1->lsthead;
    Node* head2 = s2->lsthead;
    Node* tail = NULL;
    free_list(res);
    res->lsthead = NULL;
    head2=s2->lsthead;
    while(head2!=NULL)
    {
        append(res, &tail, head2->val);
        head2=head2->next;
    }
    head2=s2->lsthead;
```

```
while(head1!=NULL)
{
    int exist_in_s2=0;
    for(head2=s2->lsthead; head2!=NULL; head2=head2->next)
    {
        if(head2->val==head1->val)
        {
            exist_in_s2=1;
            break;
        }
    }
    if(!exist_in_s2)append(res, &tail, head1->val);
    head1=head1->next;
}
```

```
int main() {
    List S1, S2, U, D, I;
    int a1[] = {2, 3, 5, 6};
    int a2[] = {3, 4, 6, 8};
    convert(&S1, a1, sizeof(a1)/sizeof(a1[0]));
    convert(&S2, a2, sizeof(a2)/sizeof(a2[0]));
    U.lsthead = D.lsthead = I.lsthead = NULL;
```

```
print_list("S1", &S1);
print_list("S2", &S2);
```

```
unionSet(&S1, &S2, &U);
print_list("S1 ∪ S2", &U);
```

```
differenceSet(&S1, &S2, &D);
```

```
print_list("S1\\S2", &D);
```

```
intersectionSet(&S1, &S2, &I);
print_list("S1nS2", &I);
```

```
// 释放
free_list(&S1);
free_list(&S2);
free_list(&U);
free_list(&D);
free_list(&I);
return 0;
}
```

```
C SetLinkT1.C M × C union.c
code_pack > C SetLinkT1.C > ...
41 void free_list(List* lst)
42 {
43     Node* head = lst->lsthead;
44     while(head)
45     {
46         Node *tmp = head;
47         head = head->next;
48         free(tmp);
49     }
50     lst->lsthead = NULL;
51     //记得置空
52 }
53
54 void print_list(const char* name, List* lst)
55 {
56     printf("set %s = ", name);
57     Node* head = lst->lsthead;
58     int is_firstitem = 1;
```

问题 输出 调试控制台 终端 端口 2 + × cppdbg: SetLinkT1 [ ] [ ]

```
set S1 = {2, 3, 5, 6}
set S2 = {3, 4, 6, 8}
set S1US2 = {3, 4, 6, 8, 2, 5}
set S1\S2 = {2, 5}
set S1nS2 = {3, 6}
[1] + Done "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm}
0<"/tmp/Microsoft-MIEngine-In-vreuigwc.4a0" 1>"/tmp/Microsoft-MIEngine-Out-tr2wg11
m.mv5"
chi_shan@localhost:/mnt/d/FDU_1/CS10004 Programming$
```

C DelteLinkMinMaxT3.c M C union.c X

code\_pack > C union.c > main()

```
7 int main()
9     /*
23     u.c[0] = 'A';
24     u.c[1] = 'a';
25     printf("%c - %c\n", u.c[0], u.c[1]);
26     printf((char [15])"\nu.u.c[1] = %c"
27     printf("\nu.u.c[1] = %c", u.c[1]);
28     return 0;
29 */
30 u.i = 24897
31 u.c[0] = A
32 u.c[1] = a
33 */
34 //u.i=(unsigned char)u.c[1]×256+(unsigned char)u.c[0]=97×256+65=24897,
35
36 /*
37 If "union" is replaced by "struct", u.i u.c[0] u.c[1] will have their own places
because struct isn't stingy but very generous !
38 The output will be:
39 0
40 A
41 a
42 */
43 }
44
```

T3

```
#include <stdio.h>
#include <stdlib.h>
struct Node
{
    int val;
    struct Node* next;
};
typedef struct Node Node;
Node* convert(int arr[], int n)
{
    int i;
    Node* head = (Node*)malloc(sizeof(Node));
    Node *tail=head;
    if(n == 0) return NULL;
    head->val = arr[0];
```

```

head->next = NULL;
for(i=1;i<n;++i)
{
    Node* cur = (Node*)malloc(sizeof(Node));
    cur->val = arr[i];
    cur->next = NULL;
    tail->next = cur;
    tail = cur;
}
return head;
}

void print_list(const char* name, Node* head)
{
printf(" List %s = {      ", name);
int is_firstitem = 1;
while (head)
{
    if (!is_firstitem) printf(" -> ");
    printf("%d", head->val);
    is_firstitem = 0;
    head = head -> next;
}
printf("      }\n");
}

void free_list(Node* head) {
while (head)
{
    Node* tmp = head;
    head = head->next;
    free(tmp);
}
}

// Delete: 删除值大于 min 且小于 max 的结点, 同时释放空间
void Delete(Node** head_ptr, int min, int max) {
if (*head_ptr == NULL) return;
Node* cur = *head_ptr;
Node* prev = NULL;
while (cur) {
    if (cur->val > min && cur->val < max) {
        Node* to_del=cur;
        if (prev == NULL) {
            // 删除头结点, 比较麻烦
            *head_ptr=cur->next;//将问题留给下一个人
            cur = *head_ptr;
        }
        else
        {
            prev->next=cur->next;
            cur = prev->next;
        }
    }
}
}

```

```
        }
        free(to_del);
    }
    else
    {
        prev=cur;
        cur=cur->next;
    }
}

int main(void) {
    int arr[] = {10, 20, 30, 40, 50};
    Node* head = convert(arr, sizeof(arr)/sizeof(arr[0]));
}
```

```
/*
 创建链表，链表中的结点值依次为“10 20 30 40 50”，Delete 函数用于删除链表中所有值大于 min 且小于 max 的结点(表元)，同时释放被删除结
 点空间。如设定 min=10 和 max=35，Delete 函数将使得链表中结点值被删为“10 40 50”。
*/
int min = 10, max = 35;
print_list("Before operation(min=10, max=35)", head);
Delete(&head, min, max);

print_list("After operation(min=10, max=35)", head);

free_list(head);
return 0;
}
```

code\_pack > C DeltaLinkMinMaxT3.c M X C OddEvenSortT3.c U C union.c

code\_pack > C DeltaLinkMinMaxT3.c > main(void)

```
76 int main(void) {
77     int arr[] = {10, 20, 30, 40, 50};
78     Node* head = convert(arr, sizeof(arr)/sizeof(arr[0]));
79
80
81    /*
82     * 创建链表，链表中的结点值依次为“10 20 30 40 50”，Delete 函数用于删除链表中所有值
83     * 大于 min 且小于 max 的结点(表元)，同时释放被删结点空间。如设定 min=10 和
84     * max=35，Delete 函数将使得链表中结点值被删为“10 40 50”。
85     */
86     int min = 10, max = 35;
87     print_list("Before operation(min=10, max=35)", head);
88     Delete(&head, min, max);
89
90     print_list("After operation(min=10, max=35)", head);
91     free_list(head);
92     return 0;
93 }
```

问题 输出 调试控制台 终端 端口 2

+ v 运行 cppdbg:DeltaLinkMinMaxT3 [ ] [ ] ...

```
List Before operation(min=10, max=35) = { 10 -> 20 -> 30 -> 40 -> 50
List After operation(min=10, max=35) = { 10 -> 40 -> 50 }
[1] + Done "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm}
0<"/tmp/Microsoft-MIEngine-In-anbn154n.zj2" 1>/tmp/Microsoft-MIEngine-Out-yg322
3.a25"
chi_shan@localhost:/mnt/d/FDU_1/CS10004_Programming$
```