```
#include "stdafx.h"
#include "string.h"
#include "malloc.h"

/*
第4次作业：整数集合用单链表实现存储，实现如下操作：
（1） 初始化集合
（2）插入一个数到集合指定位置
（3）按值删除集合中的元素
（4）按值在集合中进行查找
（5）清空集合
（6）求两个集合的交集
（7）求两个集合的并集
（8）求两个集合的差集
（9）输出集合
*/

typedef struct node
{
    int data;
    struct node *next;
}lnode,*link;

typedef int status;

void Creat(link *l,int n);
void Init(link *L);
void Print(link L);
link GetElem(link l,int e);
void Clear(link *l);
status Insert(link *L,int i,int e);
int Delete(link *L,int e);
```

```
void sum(link la,link lb,link *lc);
void differ(link la, link lb, link *lc);
void Intersection(link la,link lb,link *lc);

//不带头结点的单链表按升序保存集合中的数据元素
int _tmain(int argc, _TCHAR* argv[])
{
    link la,lb,lc;
    Creat(&la,4);
    Creat(&lb,4);
    Init(&lc);
    Intersection(la,lb,&lc);
    Print(lc);
    Clear(&lc);
    sum(la,lb,&lc);
    Print(lc);
    Clear(&lc);
    differ(la, lb, &lc);
    Print(lc);
    /*link p = GetElem(la,4);
    Insert(&la,1,444);
    Insert(&la,0,1111);
    Insert(&la,7,555);
    Insert(&la,5,777);
    Print(la);
    Delete(&la,777);
    Print(la);
    Delete(&la,555);
    Print(la);
    Delete(&la,444);
    Print(la);
    Delete(&la,4);
```

```
    Print(la);
    Clear(&la);
    Print(la);
    */
    return 0;
}

void Init(link *L) //不带头结点的单链表的空表创建
{
    *L = NULL;
}

void Creat(link *l,int n)//不带头结点的单链表的创建
{
    link s;
    *l = NULL;

    for(int i=0;i<n;i++)
    {
        s = new lnode;
        scanf("%d",&s->data);
        s->next = NULL;
        s->next = *l;
        *l = s;
    }
}
void Intersection(link la,link lb,link *lc)//交集
{
    link pa,pb;
    link pc=*lc;
    link s;
```

```
    pa = la;
    pb = lb;

    while(pa && pb)
    {
        if(pa->data == pb->data)
        {
            s = new lnode;
            s->data = pa->data;
            s->next = NULL;
            if(*lc == NULL)
            {
                *lc = s;
                pc = s;
            }
            else
            {
                pc->next = s;
                pc = s;
            }
            pa = pa->next;
            pb = pb->next;
        }
        else if(pa->data<pb->data)
            pa = pa->next;
        else
            pb = pb->next;
    }
}

void sum(link la,link lb,link *lc) //并集
{
```

```
link pa = la;
link pb = lb;
link pc = *lc;
link s;

while(pa && pb)
{
    int e;
    if(pa->data < pb->data)
    {
        e = pa->data;
        pa = pa->next;
    }
    else if(pa->data > pb->data)
    {
        e = pb->data;
        pb = pb->next;
    }
    else
    {
        e = pa->data;
        pa = pa->next;
        pb = pb->next;
    }
    s = new lnode;
    s->data = e;
    s->next = NULL;
    if(*lc== NULL)
    {
        *lc = s;
        pc = s;
    }
```

```
        else
        {
            pc->next = s;
            pc = s;
        }
    }
    while(pa)
    {
        s = new lnode;
        s->data = pa->data;
        s->next = NULL;
        if(*lc== NULL)
        {
            *lc = s;
            pc = s;
        }
        else
        {
            pc->next = s;
            pc = s;
        }
        pa = pa->next;
    }

    while(pb)
    {
        s = new lnode;
        s->data = pb->data;
        s->next = NULL;
        if(*lc== NULL)
        {
            *lc = s;
```

```
            pc = s;
        }
        else
        {
            pc->next = s;
            pc = s;
        }
        pb= pb->next;
    }
}

void differ(link la, link lb, link *lc)//差集
{
    link pa = la;
    link pb = lb;
    link pc = *lc;
    link s;
    while(pa && pb)
    {
        if(pa->data<pb->data)
        {
            s = new lnode;
            s->data = pa->data;
            s->next;
            if(*lc == NULL)
            {   *lc = s;
                pc = s;
            }
            else{
                pc->next = s;
                pc = s;
            }
```

```
            pa = pa->next;
        }
        else if(pa->data>pb->data)
            pb = pb->next;
        else if(pa->data == pb->data)
        {
            pa = pa->next;
            pb = pb->next;
        }
    }
        while(pa)
    {
        s = new lnode;
        s->data = pa->data;
        s->next = NULL;
        if(*lc== NULL)
        {
            *lc = s;
            pc = s;
        }
        else
        {
            pc->next = s;
            pc = s;
        }
        pa = pa->next;
    }
}

void Print(link L)
{
    link p = L;
```

```
    while(p)
    {
        printf("%d ",p->data);
        p = p->next;
    }
    printf("\n");
}

void Clear(link *l)
{
    while(*l)
    {
        link p = *l;
        *l =  (*l)->next;
        free(p);
    }
}
link GetElem(link l,int e)//按值查找
{
    link p = l;
    while(p)
    {
        if(p->data == e)
            break;
        p = p->next;
    }
    return p;
}
status Insert(link *L,int i,int e)
{
    link s;
```

```
    if(i == 0)
        return 0;

    if(i==1)
    {
        s = new lnode;
        s->data = e;
        s->next = *L;
        *L = s;
        return 1;
    }
    link p = *L;
    int j=1;
    while(p && j<i-1)
    {
        p = p->next;
        j++;
    }
    if(!p)
        return 0;
    s = new lnode;
    s->data = e;
    s->next = p->next;
    p->next = s;
    return 1;
}

int Delete(link *L,int e)//按值删除
{
    link p;
    if(*L == NULL)
        return 0;
```

```
    if(*L && (*L)->data == e)
    {
        p = *L;
        *L = (*L)->next;
        delete p;
        return 1;
    }
    p = *L;
    while(p && p->next)
    {
        if(p->next->data == e)
            break;
        p = p->next;
    }

    link q = p->next;
    p->next = q->next;
    delete q;
    return 1;
}
```