

## Lab 1 建立并拆分单链表

姓名：阳焘

学号：2016213391

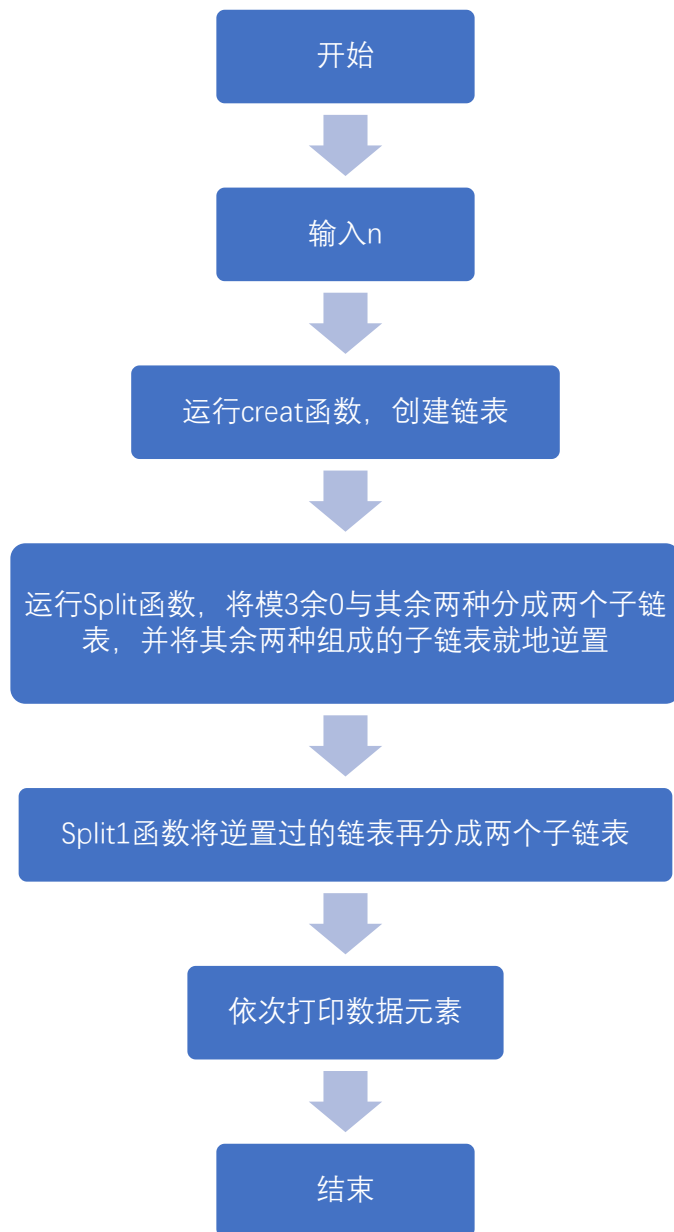
班级：2016215114

### 1. 问题描述：

- 输入N个整数顺序建立一个单链表，将该单链表拆分成三个子链表。第一个子链表存放了所有对3取模余数为0的数据元素，第二个子链表存刚了所有对3取模余数为1的数据元素，第三个子链表存刚了所有对3取模余数为2的数据元素。
- 输入
  - 第一行输入整数N
  - 第二行依次输入N个整数
- 输出
  - 第一行分别输出余数为0链表、余数为1与余数为2的元素个数
  - 第二行依次输出余数为0的子链表的所有数据元素
  - 第三行依次输出余数为1的子链表的所有数据元素
  - 第四行依次输出余数为2的子链表的所有数据元素

### 2. 程序结构

- 主程序函数 `main()`
- 建立链表函数 `struct node *Creat(int n)`  
函数参数：整型数字n，函数返回值：结构体类型指针
- 分离链表（分离模3余0与剩下两种）函数 `struct node *Split(struct node *head1)`  
函数参数：结构体类型指针head1，函数返回值：结构体类型指针
- 分离链表（分离模3余1与模3余2）函数 `struct node *Split1(struct node *head2)`  
函数参数：结构体类型指针head2，函数返回值：结构体类型指针



### 3. 源码

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

struct node
{
    int data;
    struct node *next;
};

struct node *Creat(int n)//函数申明，返回值是一个结构体类型的指针，参
                           数是整型的数字
{
    struct node *head, *p;//申请结构体指针
    int i;
    head=(struct node *)malloc(sizeof(struct node));//头指针
    head->next=NULL;//建立一个空链表
    for(i=1;i<=n;i++)
    {
        p=(struct node *)malloc(sizeof(struct node));
        scanf("%d",&p->data);//输入数据
        p->next=head->next;
        head->next=p;
    }
    return head;
}

//头插法输入

struct node *Split(struct node *head1)//分离模3余0和其他两种
{
    struct node *head2, *p, *q, *p1, *q1;
    int a=0;
    head2=(struct node *)malloc(sizeof(struct node));
```

```

head2->next=NULL;
p=head1->next;
head1->next=NULL;
q=p->next;
while(p!=NULL)
{
    if(p->data%3==0)
    {
        p->next=head1->next;
        head1->next=p;
        a++;
    }
    else
    {
        p->next=head2->next;
        head2->next=p;
    }
    p=q;
    if(q!=NULL)
        q=q->next;
}
printf("%d ",a);    //打印出模3余0

```

p1=head2->next; //把第一次分离后的head2（余1&余2）就地逆置，不然  
后面输出的余1、余2顺序会反

```

head2->next=NULL;
while(p1!=NULL)
{
    q1=p1;
    p1=p1->next;
    q1->next=head2->next;
    head2->next=q1;
}

```

```

    return head2;
}

struct node *Split1(struct node *head2)//把余1和余2分开
{
    struct node *head3, *p, *q;
    int b=0, c=0;
    head3=(struct node *)malloc(sizeof(struct node));
    head3->next=NULL;
    p=head2->next;
    head2->next=NULL;
    q=p->next;
    while(p!=NULL)
    {
        if(p->data%3==1)
        {
            p->next=head2->next;
            head2->next=p;
            b++;
        }
        if(p->data%3==2)
        {
            p->next=head3->next;
            head3->next=p;
            c++;
        }
        p=q;
        if(q!=NULL)
            q=q->next;
    }
    printf("%d %d\n",b,c); //打印出模3余1和模3余2
    return head3;
}

```

```
}
```

```
int main()
```

```
{
```

```
    int n;
```

```
    struct node *head, *head1, *head2, *p, *q, *r;
```

```
    printf("输入\n");
```

```
    scanf("%d",&n); //输入n值
```

```
    head=Creat(n); //创建最初的链表
```

```
    printf("输出\n");
```

```
    head1=Split(head); //分离模3余0与余1余2
```

```
    head2=Split1(head1); //分离模3余1与模3余2
```

```
    p=head->next; //head是模3余0链表的头指针
```

```
    while(p!=NULL)
```

```
    {
```

```
        if(p->next!=NULL)
```

```
            printf("%d ",p->data);
```

```
        else
```

```
            printf("%d\n",p->data);
```

```
        p=p->next;
```

```
    }
```

```
    q=head1->next; //head1是模3余1链表的头指针
```

```
    while(q!=NULL)
```

```
    {
```

```
        if(q->next!=NULL)
```

```
            printf("%d ",q->data);
```

```
        else
```

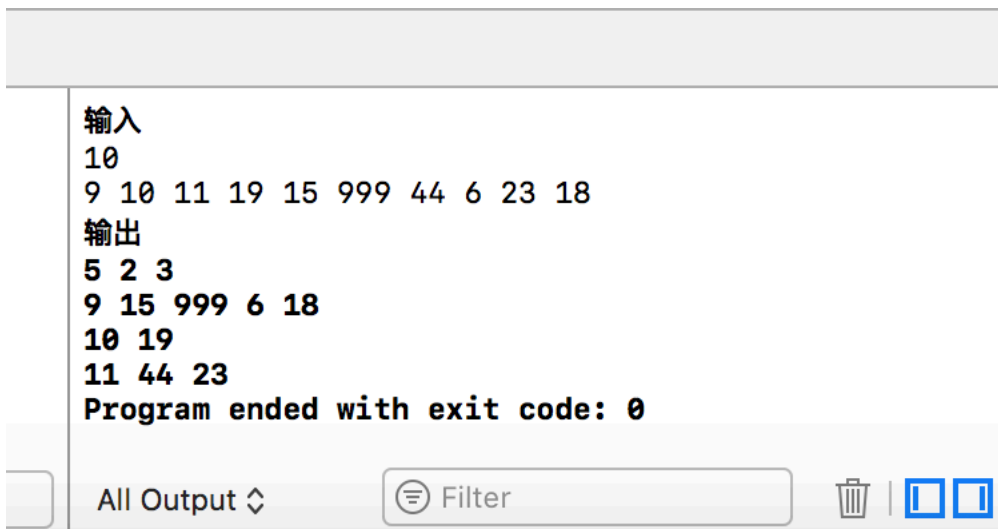
```
            printf("%d\n",q->data);
```

```
        q=q->next;
```

```
    }
```

```
r=head2->next;//head2是模3余2链表的头指针
while(r!=NULL)
{
    if(r->next!=NULL)
        printf("%d ",r->data);
    else
        printf("%d\n",r->data);
    r=r->next;
}
return 0;
}
```

#### 4. 测试结果



The screenshot shows a terminal window with the following content:

```
输入
10
9 10 11 19 15 999 44 6 23 18
输出
5 2 3
9 15 999 6 18
10 19
11 44 23
Program ended with exit code: 0
```

At the bottom of the window, there is a toolbar with the text "All Output" followed by a dropdown arrow, a "Filter" input field with a search icon, a trash icon, and two window icons.