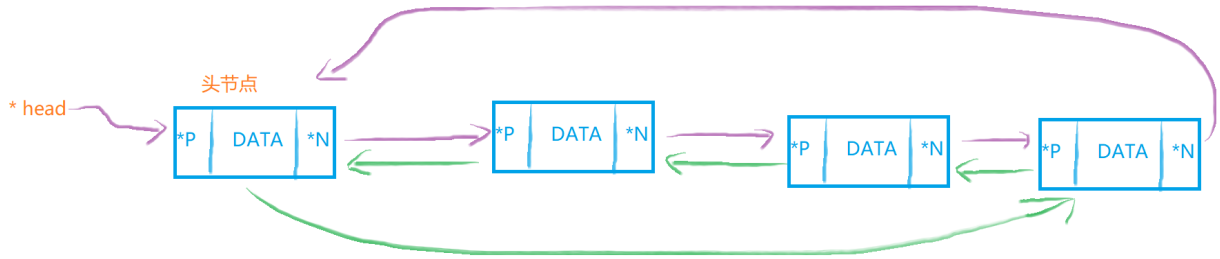
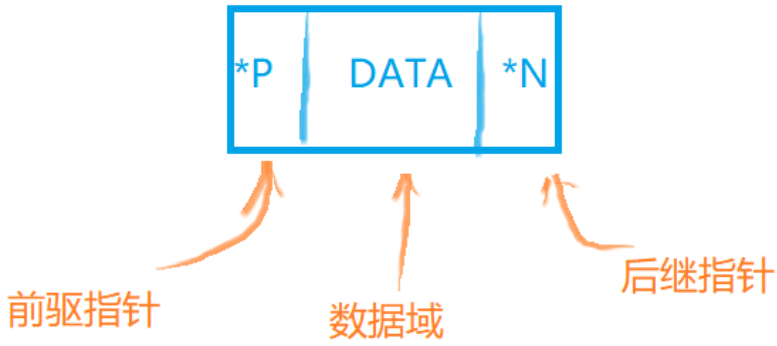
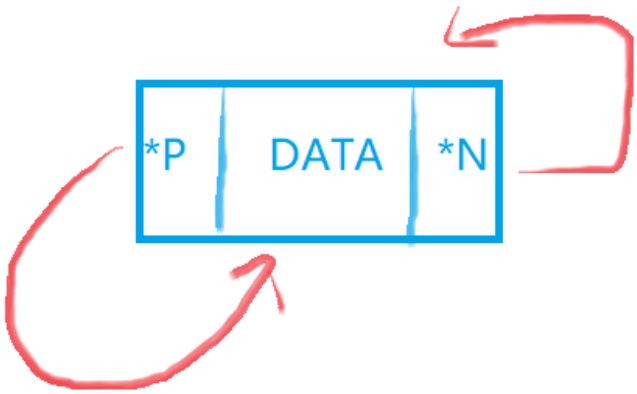


节点设计：



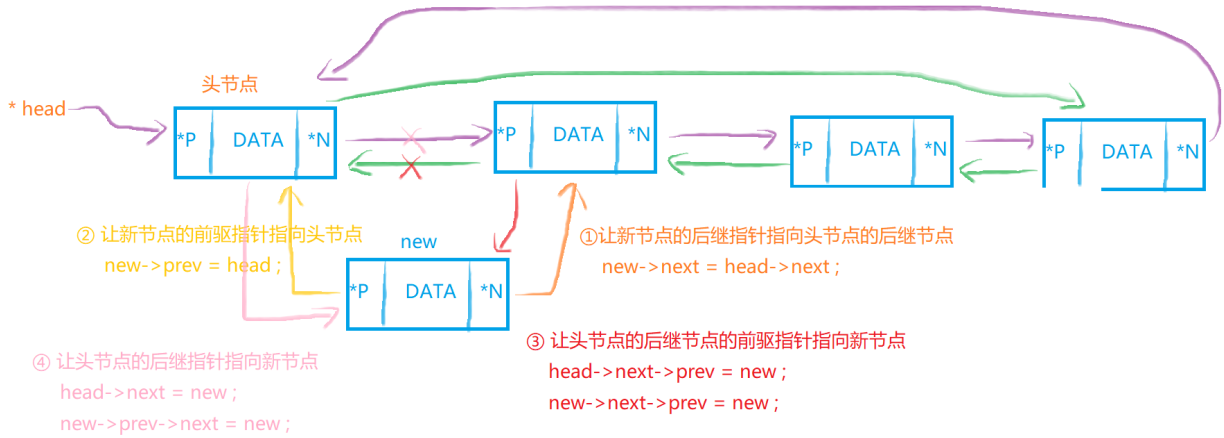

```
// 节点设计
typedef struct node
{
    Data_Type  data;         // 数据域
    struct node *prev ;      // 前驱指针
    struct node *next ;      // 后继指针
}Node , *P_Node ;
```

初始化：



```c

P_Node new_node_init( Data_Type  data )
{
    P_Node new = calloc(1 , sizeof(Node));
    if( new == NULL )
    {
        fprintf(stderr , "内存申请异常: %s\n" , strerror(errno));
        return NULL ;
    }

    new->data = data ;
    new->next = new->prev = new ;

    return new ;
}
```

头插数据



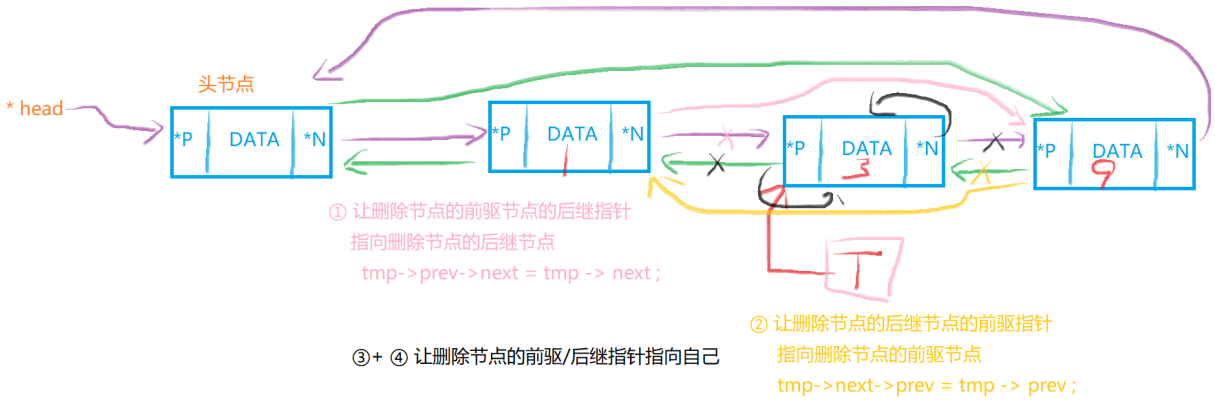

```c

bool add_2_list_head( P_Node head , P_Node new)
{
    if(head == NULL )
    {
        printf("链表头异常！！\n");
        return false ;
    }

```

```
    new->next = head->next ;
    new->prev = head ;

    head->next->prev = new ;
    head->next = new ;

    return true ;
}
```

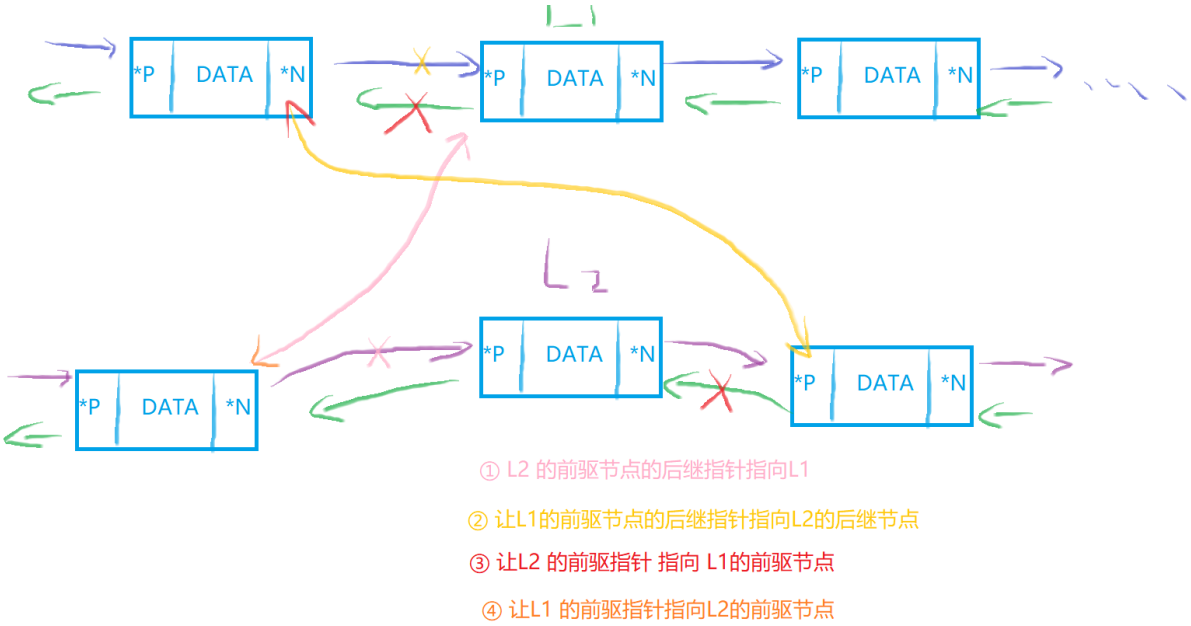
遍历显示

```

bool display_list( P_Node head)
{
    if(head == NULL || head->next == NULL )
    {
        printf("链表头异常！！\n");
        return false ;
    }

    P_Node tmp = head->next ;
    for ( ; tmp != head ; tmp=tmp->next )
    {
        printf("%d\n" , tmp->data) ;
    }

    return true ;
}
```

删除节点



连接两个双向循环链表：




```
1  void double_list_mix(P_Node L1 , P_Node L2 )
2  {
3      L2->prev->next = L1 ;
4      L1->prev->next = L2->next ;
5
6      L2->next->prev = L1->prev ;
7      L1->prev = L2->prev ;
8
9      L2->prev = L2->next = L2 ;
10 }
```

作业：

尝试自行编写双向循环链表代码实现以下功能：

1. 每插入一个新的数据进行判断，使得链表始终保持有序（升/降）
2. 实现查找删除操作（例如数据为正数， 输入负数则删除）
3. 实现移动节点功能

预习：

队列

内核链表？