```
/*
删除链表中等于给定值 val 的所有节点。

示例:

输入: 1->2->6->3->4->5->6, val = 6
输出: 1->2->3->4->5

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/remove-linked-list-elements
著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。
*/
```

分析：

- 将节点移除单链表的情况有两种，移除链表头部节点，移除中部和尾部节点。
  - 移除链表头部节点：由于新的头节点要返回，原有头节点要移除（释放），所以先保存原有头节点，再更新头节点，再释放原有头节点的内存空间。

    ```
    temp_pointer = head;
    head=head->next;
    free(temp_pointer);
    ```

  - 移除中部和尾部节点：当头结点不为空时，需要一个指针保存记录当前节点和当前节点的前驱节点，**当发现当前节点应当删除时**，应该让前驱节点的后继指向当前节点的后继，并释放当前节点，随后更新当前节点为前驱节点的后继，**当发现当前节点不应当删除时**，就直接把当前节点更新给前驱节点，把当前节点的后继节点更新给当前节点，直到当前节点指向$NULL$,到达链表尾部擦停止。**更新顺序都是先更新前驱节点再更新当前节点。**

    ```
    front=head;
    cur = front->next;
    while(cur!=NULL)
    {
        if(cur->val == val)
        {
            front->next = cur->next;
            free(cur);
            cur = front->next;
        }
        else
        {
            front = cur;
            cur = front->next;
        }
    }
    ```

方法一：C_solution

```c
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     struct ListNode *next;
 * };
 */
struct ListNode* removeElements(struct ListNode* head, int val)
{
    struct ListNode* cur    = NULL;
    struct ListNode* front  = NULL;


    /*首先删除链表首val节点,保证head节点的值不是val*/
    while(  (head!=NULL)
          &&(head->val==val)
    )
    {
        cur     =   head;
        head    =   head-> next;
        free(cur);
    }

    if(head!=NULL)
    {

        front   = head;
        cur     = front->next;
        while(cur!=NULL)
        {
            if(cur->val == val)
            {
                front->next = cur->next;
                free(cur);
                cur = front->next;
            }
            else
            {
                front = cur;
                cur = front->next;
            }
        }
    }
    return head;
}
/*
```

```
执行结果:
通过
显示详情
执行用时 :16 ms, 在所有 C 提交中击败了95.57% 的用户
内存消耗 :9.4 MB, 在所有 C 提交中击败了87.50%的用户
*/
```

AlimyBreak
2019.07.21