反转链表

定义一个函数，输入一个链表的头节点，反转该链表并输出反转后链表的头节点。

示例:

输入：1->2->3->4->5->NULL

输出：5->4->3->2->1->NULL

/*借助中间链表 通过调制链表 达到反转链表的目的

1）pre 指针指向已经反转好的链表的最后一个节点，最开始没有反转，所以指向 nullptr

2）cur 指针指向待反转链表的第一个节点，最开始第一个节点待反转，所以指向 head

3）nex 指针指向待反转链表的第二个节点，目的是保存链表，因为 cur 改变指向后，后面的链表则失效了，所以需要保存

接下来，循环执行以下三个操作

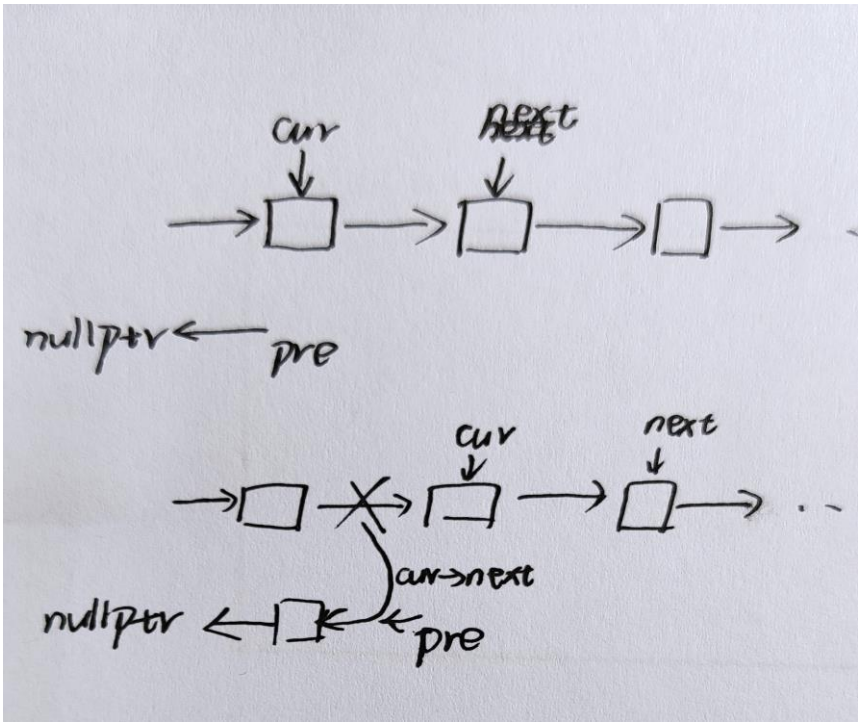
1）nex = cur->next，保存作用

2）cur->next = pre 未反转链表的第一个节点的下个指针指向已反转链表的最后一个节点

3）pre = cur， cur = nex；指针后移，操作下一个未反转链表的第一个节点

循环条件，当然是 cur != nullptr

循环结束后，cur 当然为 nullptr，所以返回 pre，即为反转后的头结点



```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */
class Solution {
public:
    ListNode* reverseList(ListNode* head) {
        ListNode* pre=nullptr;
        ListNode* cur=head;
        while(cur!=nullptr)
        {
            ListNode* next=cur->next;
            cur->next=pre;
            pre=cur;
            cur=next;
        }
        return pre;
    }
};
```