```
/*
给定一个带有头结点 head 的非空单链表，返回链表的中间结点。

如果有两个中间结点，则返回第二个中间结点。



示例 1：

输入：[1,2,3,4,5]
输出：此列表中的结点 3 (序列化形式：[3,4,5])
返回的结点值为 3 。 (测评系统对该结点序列化表述是 [3,4,5])。
注意，我们返回了一个 ListNode 类型的对象 ans，这样：
ans.val = 3, ans.next.val = 4, ans.next.next.val = 5, 以及
ans.next.next.next = NULL.
示例 2：

输入：[1,2,3,4,5,6]
输出：此列表中的结点 4 (序列化形式：[4,5,6])
由于该列表有两个中间结点，值分别为 3 和 4，我们返回第二个结点。



提示：

给定链表的结点数介于 1 和 100 之间。

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/middle-of-the-linked-list
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:

- 方法一:两遍遍历法,先遍历一遍统计链表总长度,然后再从head开始遍历一半长度返回.
- 方法二:快慢指针标记法,设置两个指针,快指针每次向链表后端推进2个节点，慢指针每次向链表后端推进1个节点，直到快指针遇到链表末尾，返回慢指针的next节点即可.

方法一:C++_两遍遍历法

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */
class Solution
{
    public:
        ListNode* middleNode(ListNode* head)
        {
```

```
            ListNode* temp  = head;
            int       count = 0;

            while(temp)
            {
                count++;
                temp = temp->next;
            }

            temp = head;
            count = count / 2;
            while(count--)
            {
                temp = temp -> next;
            }
            return temp ;
        }
};

/*
执行结果:
通过
显示详情
执行用时 :4 ms, 在所有 cpp 提交中击败了72.23%的用户
内存消耗 :8.5 MB, 在所有 cpp 提交中击败了34.56%的用户
*/
```

方法二:C++_快慢指针标记法

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */
class Solution
{
    public:
        ListNode* middleNode(ListNode* head)
        {
            ListNode temp(0);
            temp.next = head;
            ListNode* slower    =   &temp;
            ListNode* faster    =   &temp;


            while(faster->next && faster->next->next)
            {
                slower = slower->next;
                faster = faster->next->next;
            }


            return slower->next;
```

```
28
29          }
30  };
31
32  /*
33  执行结果:
34  通过
35  显示详情
36  执行用时 :4 ms, 在所有 cpp 提交中击败了72.23%的用户
37  内存消耗 :8.5 MB, 在所有 cpp 提交中击败了24.13%的用户
38  */
```

AlimyBreak
2019.10.23