简单方式，直接交换值

```cpp
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* swapPairs(ListNode* head) {

        ListNode* current = head;

        // 处理空链表和只有一个节点的情况
        if (current == nullptr || current->next == nullptr) {
            return head;
        }

        while (current != nullptr && current->next != nullptr) {
            // 交换当前节点和下一个节点的值
            int temp = current->val;
            current->val = current->next->val;
            current->next->val = temp;

            // 移动到下一个节点对
            current = current->next->next;
        }

        return head;
    }
};
```

如果是节点指针的移动

```cpp
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
```

```
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* swapPairs(ListNode* head) {
        if (head == nullptr || head->next == nullptr) {
            return head;
        }
        // 虚拟头节点
        ListNode* dummy = new ListNode(0);
        dummy->next = head;
        ListNode* temp = dummy;
        head = temp->next->next;
        // 进行节点对的交换
        while (temp->next != nullptr && temp->next->next != nullptr) {
            // 预先保存节点指针
            ListNode* a = temp->next;
            ListNode* b = temp->next->next;

            // 指针变换
            temp->next = b;     // 前一个节点指向b
            a->next = b->next; // a指向下一对的第一个节点
            b->next = a;        // b指向a

            // 更新temp
            temp = a; // a是下一对的前一个节点
        }
        return head; // 返回新的头节点
    }
};```
```