题目描述

2.两数相加

给你两个 非空 的链表，表示两个非负的整数。它们每位数字都是按照 逆序 的方式存储的，并且每个节点只能存储 一位 数字。

请你将两个数相加，并以相同形式返回一个表示和的链表。

你可以假设除了数字 0 之外，这两个数都不会以 0 开头。

```
e.g.

输入：l1 = [2,4,3], l2 = [5,6,4]

输出：[7,0,8]

解释：342 + 465 = 807.
```

Solu:

```
#include <iostream>
using namespace std;

//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* addTwoNumbers(ListNode* l1, ListNode* l2)
    {
        ListNode* head = new ListNode(0);
        ListNode* current = head;
//head 和 current 指向同一块内存地址,因此修改current->next实质上就是在修改head->next;
这里相当于是借助current指针负责对链表的延长,借助head指针实现最终生成链表的从头访问.
        int carry = 0;
        while (l1 != nullptr || l2 != nullptr || carry != 0)
        {
            int sum = carry;
            if (l1 != nullptr)
```

```
            {
                sum += l1->val;
                l1 = l1->next;
            }
            if (l2 != nullptr)
            {
                sum += l2->val;
                l2 = l2->next;
            }
            carry = sum / 10;
            current->next = new ListNode(sum % 10);
//提取出个位数,并以此为内容创建一个新的节点,并使得当前节点指针中的next指针,指向该新创建的结点.(将原结点与新创建的结点连接起来,形成链表)
            current = current->next;
//将current指针移动到当前节点的next所指向的下一个结点.
        }
        ListNode* result = head->next;
        delete head;
        return result;
    }
};
```

comment:

1. 本题主要考查链表这一数据结构的基本应用.
2. 本题先创建一个虚拟的头结点,让指针先指向虚拟的头结点,在利=利用链表的尾指针最终指向NULL的特性,使用while循环遍历链表,循环遍历过程中要注意每一步节点中指针的更新.最终的结果链表是一步一步生成出来的,因此要不断使用new方法来生成新的存储有对应数据的结点,使得上一节点中的指针正确指向下一节点的位置,并更新移动指针指向下一节点.
3. 在上述生成链表的过程(不断生成新的存储有给定数据的结点,并将节点与节点之间连接起来)结束后,此时再通过虚拟头结点重新访问整个链表.将访问的结果存储到一个新的链表(result)中,并删除虚拟头结点,返回result作为本题的结果链表.
4. 有关于while循环条件中carry!=0这一条件的说明:即:即便l1和l2两个节点指针所访问的对应链表均已达到了nullptr,而carry的值仍不为0,则代表结果链表(result所访问的链表ListNode* result = head->next)仍未构建完成(需要添上carry项才算构建完成,否则为遗漏最高位的结果链表的情形.)