```
/*
给出两个 非空 的链表用来表示两个非负的整数。其中，它们各自的位数是按照 逆序 的方式存储的，并且它们的每个节点只能存储 一位 数字。

如果，我们将这两个数相加起来，则会返回一个新的链表来表示它们的和。

您可以假设除了数字 0 之外，这两个数都不会以 0 开头。

示例：

输入：(2 -> 4 -> 3) + (5 -> 6 -> 4)
输出：7 -> 0 -> 8
原因：342 + 465 = 807

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/add-two-numbers
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：

- 首先想到的方法当然是先把链表转成数，相加之后在转成链表，但用C语言一定会溢出，果断放弃。
- 模拟实际的加法：
  - 由于链表顺序和数字顺序是"左对齐"的，所以可以直接从链表头部开始处理；
  - 建立一个尾巴指针$tail_node$指向返回链表的尾巴节点，由于第一个节点加入时，需要在$tail_node$的$next$节点增加，所以先申请一个头节点结构体变量$head$，先让$tail$指向$head$,然后数据运算完以后，直接返回$head$的next指针.

方法一：C_模拟加法

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


struct ListNode* addTwoNumbers( struct ListNode* l1 ,
                                struct ListNode* l2
                              )
{

    struct ListNode     head            =   {0,NULL}    ;
    struct ListNode*    tail_node       =   NULL        ;
    struct ListNode*    temp            =   NULL        ;
    int                 carry_flag      =   0           ;
```

```
    int                 state_words     =   0           ;

    head.next    = NULL;
    tail_node    = &head;


    state_words  = (l1!=NULL)? 1 : 0;
    state_words += (l2!=NULL)? 2 : 0;
    while(state_words)
    {
        switch(state_words)
        {
            case 1: // (l1 != NULL) && (l2 == NULL)
                temp        =   (struct ListNode*)malloc(sizeof(struct
ListNode))       ;
                temp->next  =   NULL
       ;
                temp->val   =   l1->val + carry_flag
       ;
                carry_flag  =   temp->val / 10
       ;
                temp->val   %=  10
       ;
                l1          =   l1->next
       ;

                tail_node->next =   temp;
                tail_node       =   temp;

            break;
            case 2: // (l1 == NULL) && (l2 != NULL)
                temp        =   (struct ListNode*)malloc(sizeof(struct
ListNode))       ;
                temp->next  =   NULL
       ;
                temp->val   =   l2->val + carry_flag
       ;
                carry_flag  =   temp->val / 10
       ;
                temp->val   %=  10
       ;
                l2          =   l2->next
       ;

                tail_node->next =   temp;
                tail_node       =   temp;
            break;

            case 3: // (l1 != NULL ) && (l2 != NULL)
                temp        =   (struct ListNode*)malloc(sizeof(struct
ListNode))       ;
                temp->next  =   NULL
       ;
                temp->val   =   l1->val + l2->val + carry_flag
       ;
                carry_flag  =   temp->val / 10
       ;
```

```
                temp->val   %=  10
        ;
                l1 = l1->next;
                l2 = l2->next;

                tail_node->next =   temp;
                tail_node       =   temp;
            break;

            default: // case 0: (l1 == NULL) && (l2 == NULL)
            break;
        }
        state_words  = (l1!=NULL)? 1 : 0;
        state_words += (l2!=NULL)? 2 : 0;
    }

    if(carry_flag !=0)
    {
        temp            =   (struct ListNode*)malloc(sizeof(struct ListNode))
    ;
        temp->next      =   NULL
   ;
        temp->val       =   1
    ;
        tail_node->next =   temp
   ;
        tail_node       =   temp
   ;
    }

    return head.next;
}


/*
执行结果:
通过
显示详情
执行用时 :36 ms, 在所有 C 提交中击败了11.83% 的用户
内存消耗 :8.8 MB, 在所有 C 提交中击败了95.06%的用户
*/
```

方法一_plus:C_模拟加法，精简代码

```
struct ListNode* addTwoNumbers( struct ListNode* l1 ,
                                struct ListNode* l2
                              )
{

    struct ListNode     head            =   {0,NULL}    ;
    struct ListNode*    tail_node       =   NULL        ;
    struct ListNode*    temp            =   NULL        ;
    int                 carry_flag      =   0           ;
    int                 state_words     =   0           ;
```

```
    head.next    = NULL;
    tail_node    = &head;

    state_words  = (l1!=NULL)? 1 : 0;
    state_words += (l2!=NULL)? 2 : 0;
    while(state_words)
    {
        temp            =   (struct ListNode*)malloc(sizeof(struct ListNode))
    ;
        temp->next      =   NULL
   ;
        switch(state_words)
        {
            // (l1 != NULL) && (l2 == NULL)
            case 1:
                temp->val       =   l1->val + carry_flag            ;
                l1              =   l1->next                        ;
            break;
            // (l1 == NULL) && (l2 != NULL)
            case 2:
                temp->val       =   l2->val + carry_flag            ;
                l2              =   l2->next                        ;
            break;
            // (l1 != NULL ) && (l2 != NULL)
            case 3:
                temp->val       =   l1->val + l2->val + carry_flag  ;
                l1              =   l1->next                        ;
                l2              =   l2->next                        ;
            break;
             // case 0: (l1 == NULL) && (l2 == NULL)
            default:
            break;
        }

        carry_flag      =   temp->val / 10
;
        temp->val       %=  10
;

        tail_node->next =   temp;
        tail_node       =   temp;

        state_words     =   (l1!=NULL)? 1 : 0;
        state_words     +=  (l2!=NULL)? 2 : 0;
    }

    if(carry_flag !=0)
    {
        temp            =   (struct ListNode*)malloc(sizeof(struct ListNode))
    ;
        temp->next      =   NULL
   ;
        temp->val       =   1
    ;
        tail_node->next =   temp
   ;
        tail_node       =   temp
   ;
```

```
    }

    return head.next;
}


/*
执行结果:
通过
显示详情
执行用时 :36 ms, 在所有 C 提交中击败了11.83% 的用户
内存消耗 :8.8 MB, 在所有 C 提交中击败了95.33%的用户
*/
```

AlimyBreak
2019.08.19