

**课程名称：数据结构与算法**

**第一次上机实验报告**

**姓 名： 刘瑞欣**

**学 号： U202016818**

**班 级： 财务管理2001班**

**任课教师： 刘干**

**目录**

[0. 将单链表逆置，要求不改变结点地址 3](#_Toc131015907)

[0.1 算法一：头插法 3](#_Toc131015908)

[0.2 算法二：递归算法 3](#_Toc131015909)

[1. 找单链表中点 4](#_Toc131015910)

[1.1 算法一：快慢指针 4](#_Toc131015911)

[1.2 算法二：计算链表长度 5](#_Toc131015912)

[2. 找单链表倒数第K个点 6](#_Toc131015913)

[3. 删除单链表倒数第K个点 7](#_Toc131015914)

[4. 判断单链表是否有环，如果有，找出交点 8](#_Toc131015915)

[5. 判断两个单链表是否相交，如果相交，找出交点 9](#_Toc131015916)

[5.1 算法一：长短判断 9](#_Toc131015917)

[5.2 算法二：暴力拦截 11](#_Toc131015918)

[6. 对于有序单链表，删除重复节点（保留一个/无保留） 12](#_Toc131015919)

[6.1 算法一：有保留 12](#_Toc131015920)

[6.2 算法二：无保留 13](#_Toc131015921)

[7. 约瑟夫问题 14](#_Toc131015922)

[8. 合并两个升序单链表(保留重复点)，合并后为升序 15](#_Toc131015923)

[9. 合并两个升序单链表(保留重复点)，合并后为降序 16](#_Toc131015924)

[9.1 算法一：升序再逆置 16](#_Toc131015925)

[9.2 算法二：头插法 17](#_Toc131015926)

[10. 判断一个单链表是否对称 18](#_Toc131015927)

[10.1 算法一：直接比较 18](#_Toc131015928)

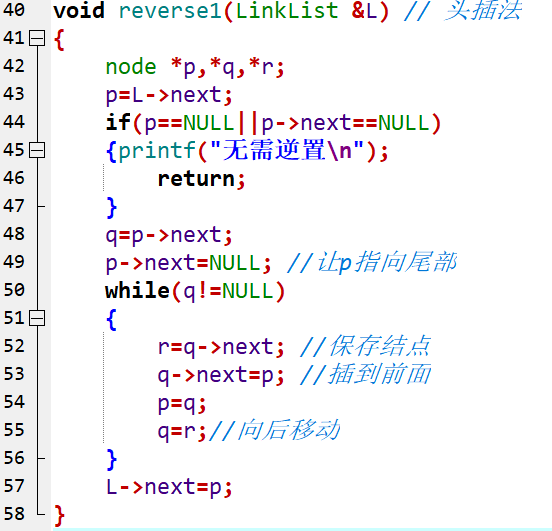
[10.2 算法二：利用数组 20](#_Toc131015929)

# 0. 将单链表逆置，要求不改变结点地址

## 0.1 算法一：头插法

**算法思想：**首先将头节点置空，将要移动的节点的下一个结点进行保存，再把要移动的结点插到第一的位置，通过循环，每次往后推一个，直到为空

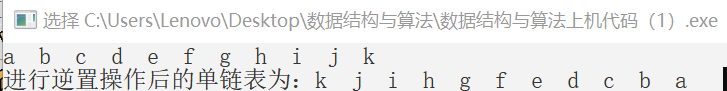
**子函数代码：**



**时间复杂度：**O(n)

**空间复杂度：**O(1)

**运行结果：**



## 0.2 算法二：递归算法

**算法思想：**设置递归出口，每次将一个结点拿出向后推，自己调用自己，一直推到递归出口停止，再把拿出来的结点连接起来。

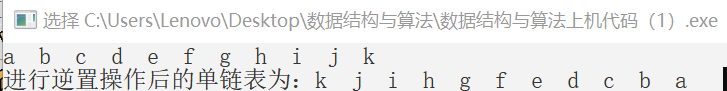
**子函数代码：**



**时间复杂度：**O(n)

**空间复杂度：**O(n)

**运行结果：**

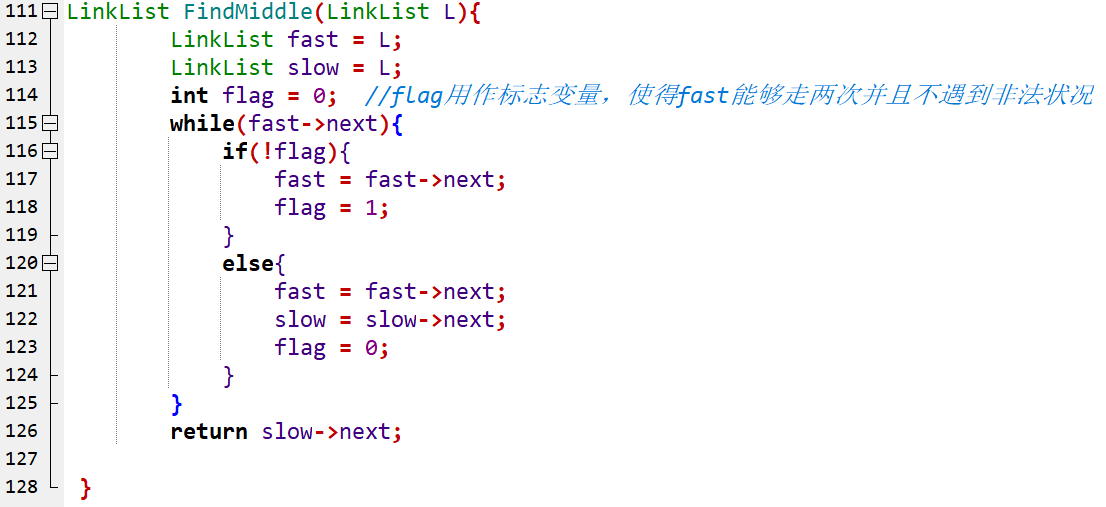


1. 找单链表中点

## **1.1 算法一：快慢指针**

**算法思想：**用双指针，都从头结点出发，快指针一次走两步，慢指针一次走一步，当快指针走到尾结点时慢指针刚好走到一半，即中点，本算法利用哈希判断法来实现。

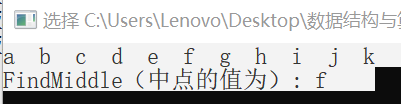
**子函数代码：**



**时间复杂度：**O(n)

**空间复杂度：**O(1)

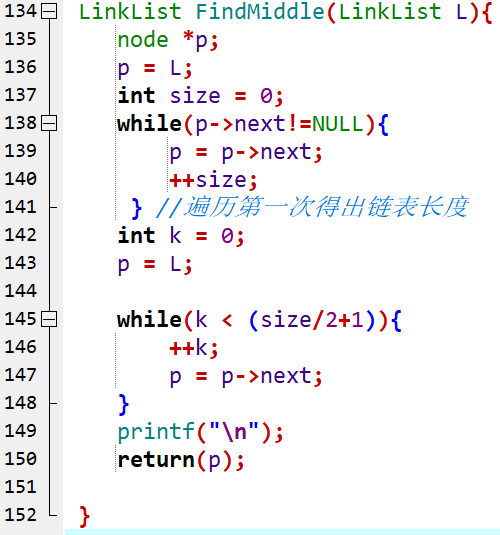
**运行结果：**



## **1.2 算法二：计算链表长度**

**算法思想：**把单链表先遍历一遍得到表长size，然后在遍历第二遍，到size/2+1就停住，则是中点，如果是偶数列则返回中间偏后的那个。

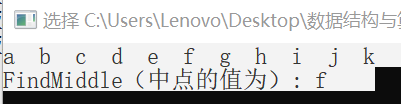
**子函数代码：**



**时间复杂度：**O(n)

**空间复杂度：**O(1)

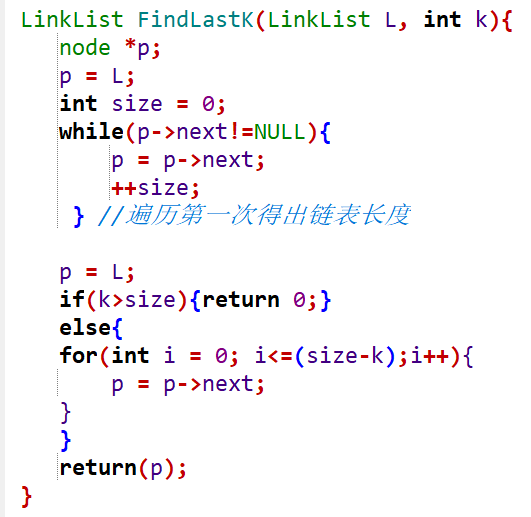
**运行结果：**



# 2. 找单链表倒数第K个点

**算法思想：**把单链表先遍历一遍得到表长size，然后再遍历第二遍，到size-k就停住，即找到倒数第K个点。

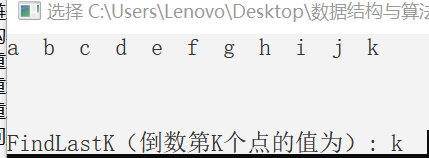
**子函数代码：**



**时间复杂度：**O(n)

**空间复杂度：**O(1)

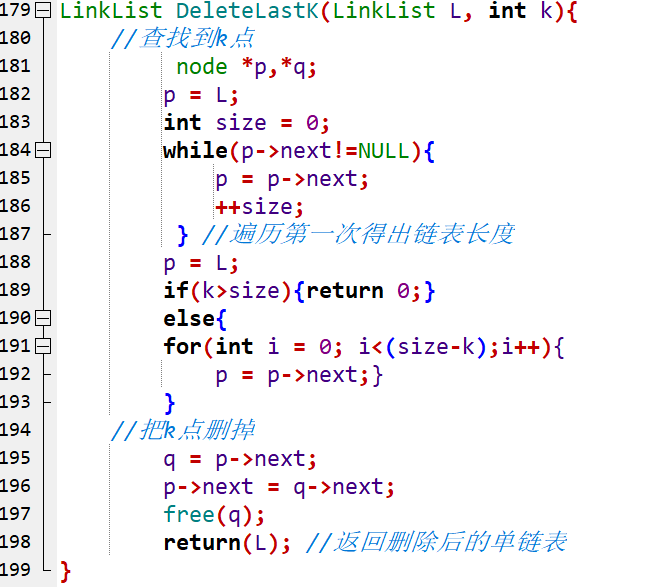
**运行结果：**这里主函数里输入的是FindLastK (L1, 1)，即找的是倒数第一个点



# 3. 删除单链表倒数第K个点

**算法思想：**把单链表先遍历一遍得到表长size，然后在遍历第二遍，到size-k就停住，即找到倒数第K个点，设置一个q结点用来储存p结点的前置节点，方便删除。

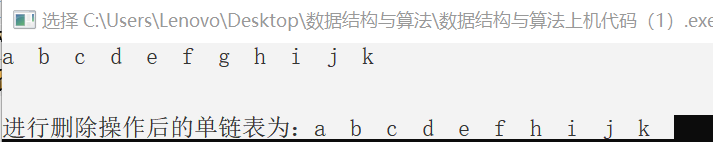
**子函数代码：**



**时间复杂度：**O(n)

**空间复杂度：**O(1)

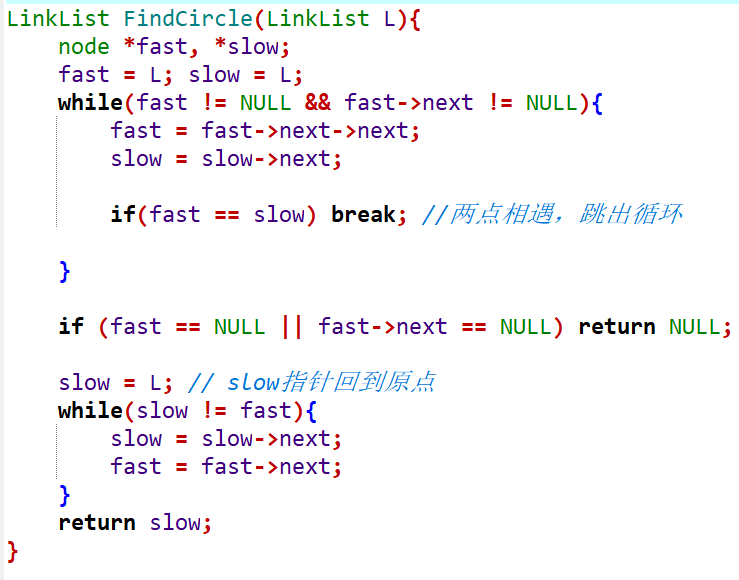
**运行结果：**这里主函数里输入的是DeleteLastK (L1, 5)，即删的是倒数第五个点



# 4. 判断单链表是否有环，如果有，找出交点

**算法思想：**利用快慢指针，设定两个指针，一个快指针（一次走两步），一个慢指针（一次走一步），如果有环，那么必然两个指针必然相遇；如果不存在环，fast会提前到达NULL。我们知道相遇点到连接点的距离等于头结点到连接点的距离。因此，在快慢指针相遇后，我们让慢指针重新指向头结点，快指针依旧指向相遇点，二者同时开始，每次向后移动一个节点，当快慢指针再次相遇时，相遇的位置即连接点。

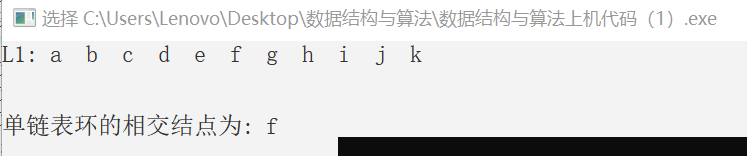
**子函数代码：**



**时间复杂度：**O(n)

**空间复杂度：**O(1)

**运行结果：**主函数中把L1的中点和尾部的点相连，创造环的节点，输出为f

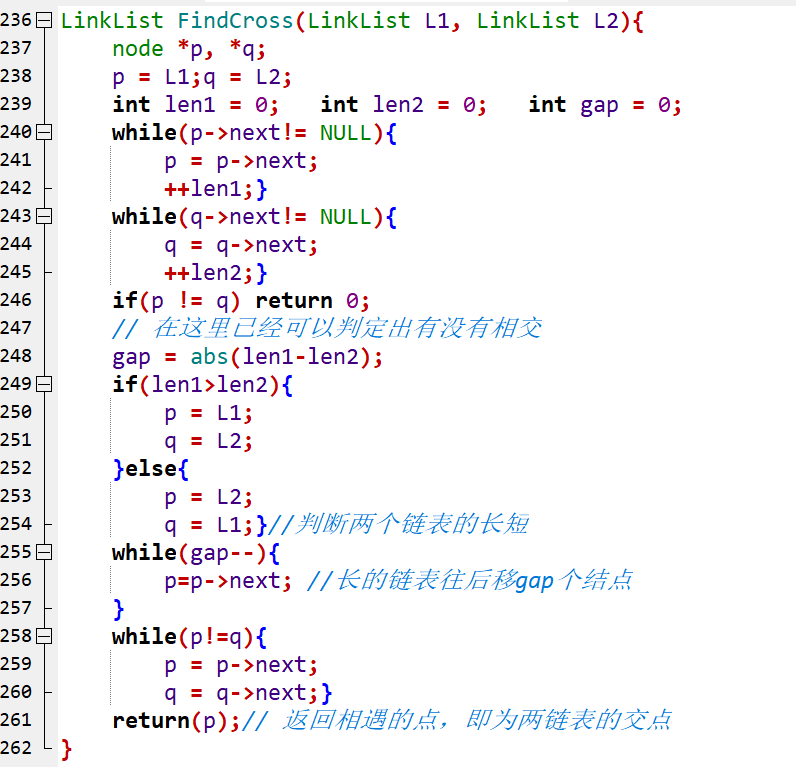


# 5. 判断两个单链表是否相交，如果相交，找出交点

## 5.1 算法一：长短判断

**算法思想：**如果两个链表相交，那么他们的尾结点必然相交，分别遍历到两个链表的尾部，并判断二者是否相同，两个链表必然有长短之差，设短单链表的长度为len1，长单链表的长度为len2，让长的往后移|len2-len1|，然后判断两个尾结点是不是相同就可以得到结点。

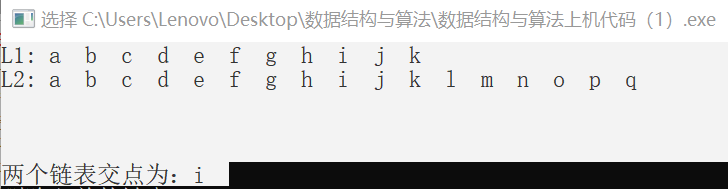
**子函数代码：**



**时间复杂度：**O(n)

**空间复杂度：**O(1)

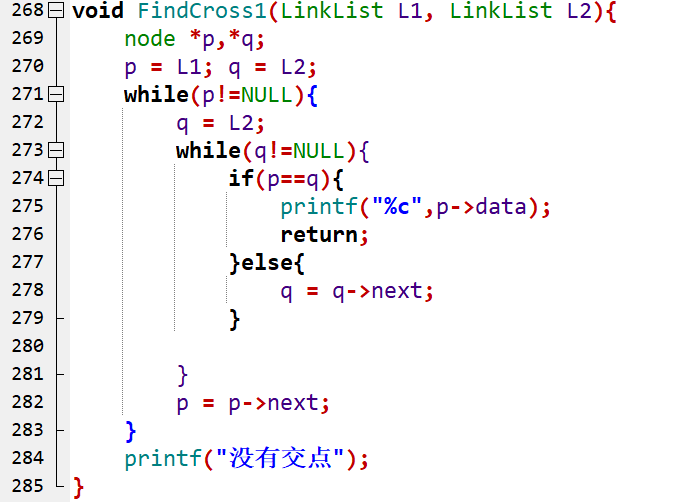
**运行结果：**这里是让L1的中点与L2的中点相连，创造相交。



## 5.2 算法二：暴力拦截

**算法思想：**L1的一个点与L2的所有点进行比较，第一个节点与L2的所有节点都不相同，就向后推，直到找到相同的点，或者走完。

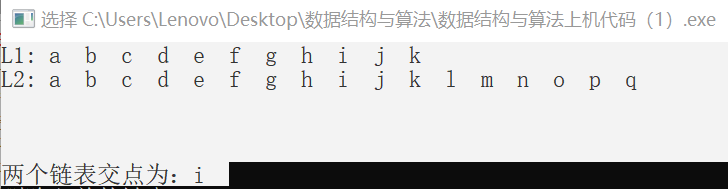
**子函数代码：**



**时间复杂度：**O(n²)

**空间复杂度：**O(1)

**运行结果：**这里是让L1的中点与L2的中点相连，创造相交。



# 6. 对于有序单链表，删除重复节点（保留一个/无保留）

## 6.1 算法一：有保留

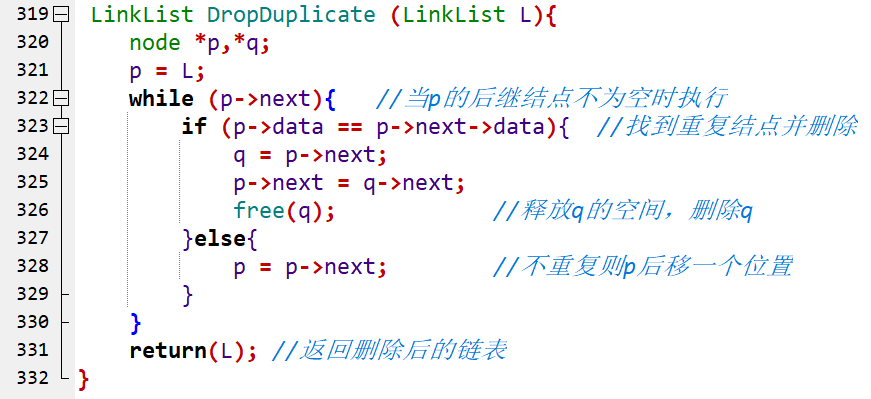
**算法思想：**用一个指针p指向起始结点，如果p所指的值等于后继结点的值，则删除后继节点；否则p就指向后继节点。重复以上过程，直到p的后继节点为空，然后删掉重复的部分。

**子函数代码：**

先创造一个有重复的链表：



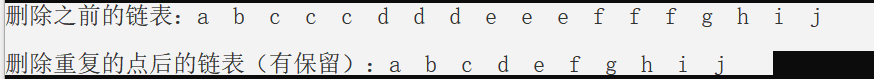
然后进行判断和删除操作：



**时间复杂度：**O（n）

**空间复杂度**：O（1）

**运行结果：**



## 6.2 算法二：无保留

**算法思想：**遍历单链表的时候，检查当前节点与下一点是否为相同值，如果相同，继续查找相同值的最大长度，然后指针改变指向。

**子函数代码：**



**时间复杂度：**O（n）

**空间复杂度**：O（1）

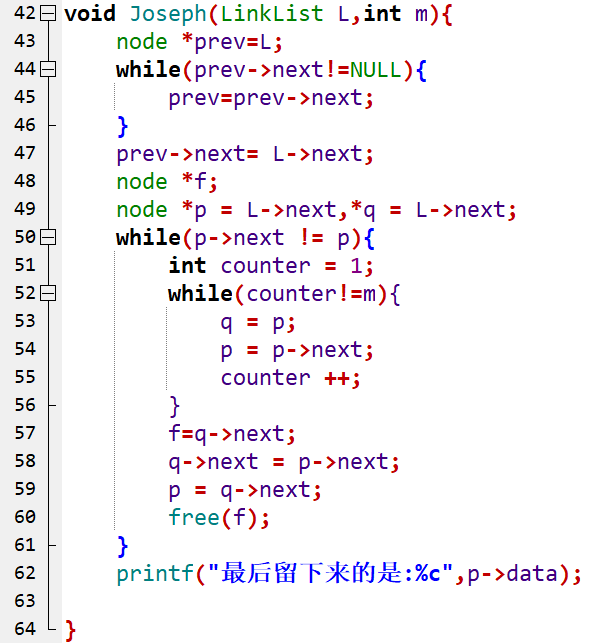
**运行结果：**



# 7. 约瑟夫问题

**算法思想：**利用循环单链表，利用循环单列表，设置两个指针，一个p，一个指向p的前驱结点，从第一个结点出发，报数数到m之后删掉该结点，然后往后走，直到最后p的后置结点为自身（即没有留下其他结点）

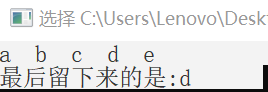
**子函数代码：**



**时间复杂度：O(n)**

**空间复杂度：O(1)**

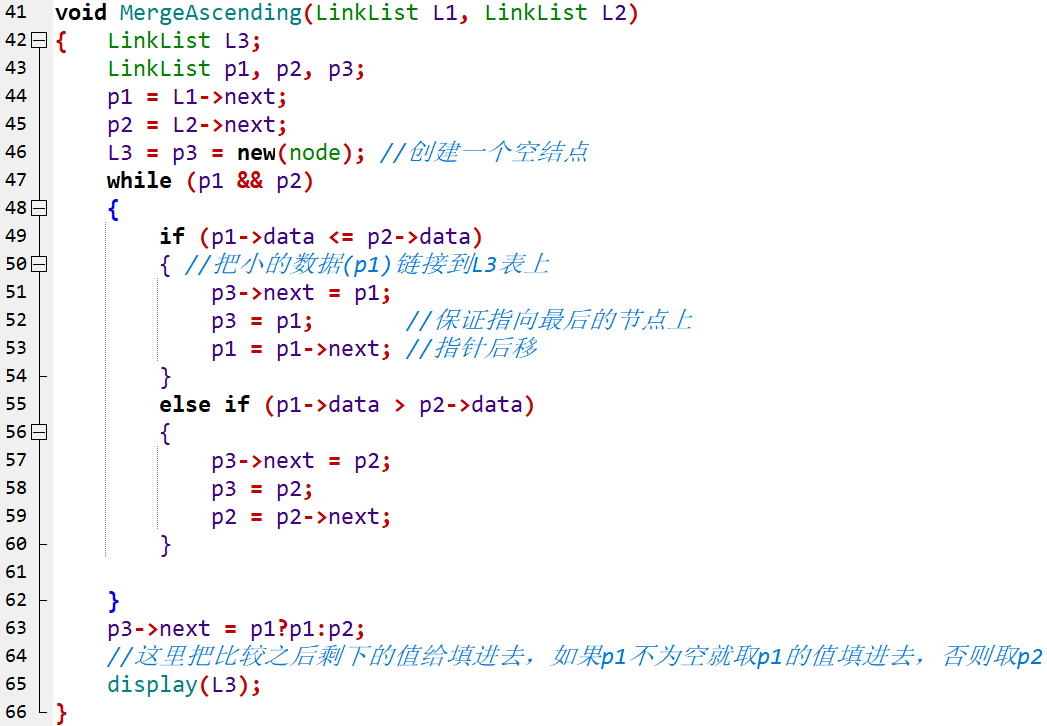
**运行结果：**输入为Joseph(L,3)，报数为3



# 8. 合并两个升序单链表(保留重复点)，合并后为升序

**算法思想：**创造新链表，利用尾插法，两个链表比较然后把按顺序排列的结点数据依次放进进新链表中。

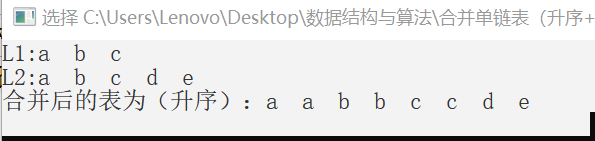
**子函数代码：**



**时间复杂度：O(n)**

**空间复杂度：O(n)**

**运行结果：**

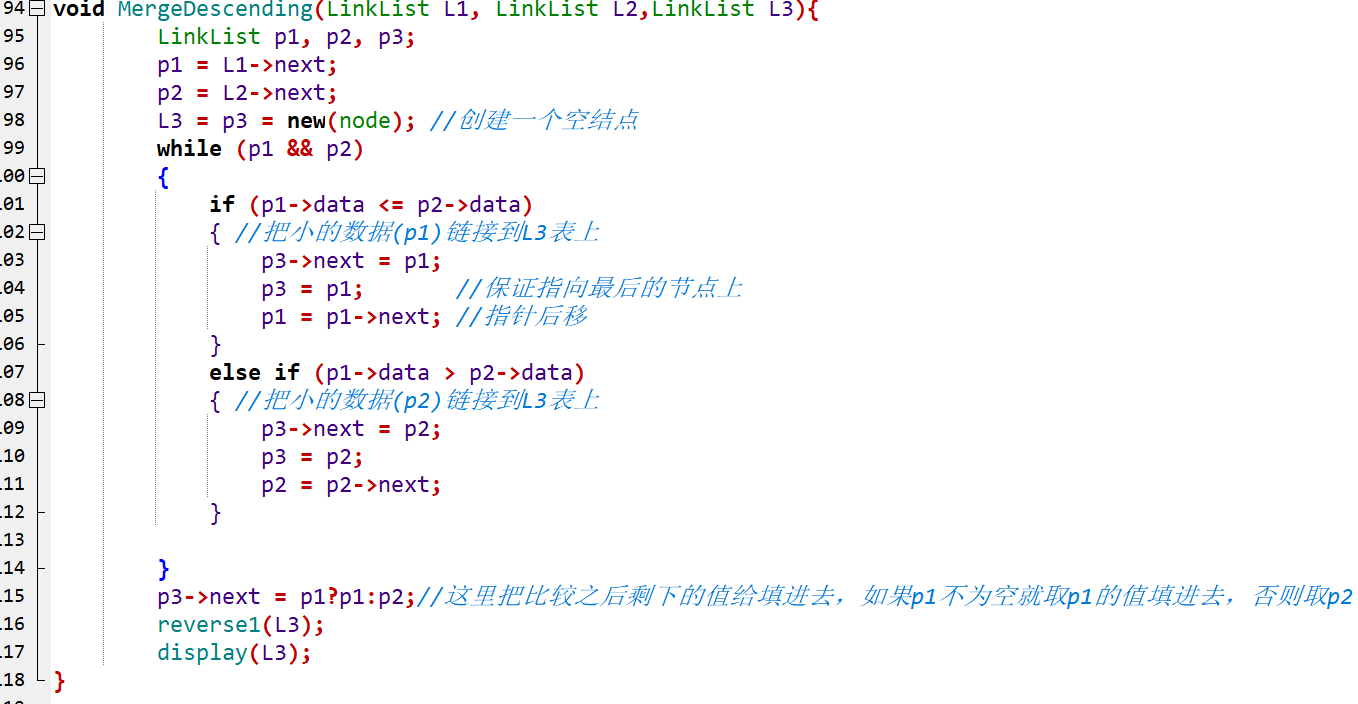


# 9. 合并两个升序单链表(保留重复点)，合并后为降序

## 9.1 算法一：升序再逆置

**算法思想：**先用合并为升序的算法（尾插法）算出升序的排列，然后直接用reverse函数逆置就是倒序的。

**子函数代码：**



这里reverse1的代码是调用的前面第0题的函数。

**时间复杂度：**O(n)

**空间复杂度：**O(n)

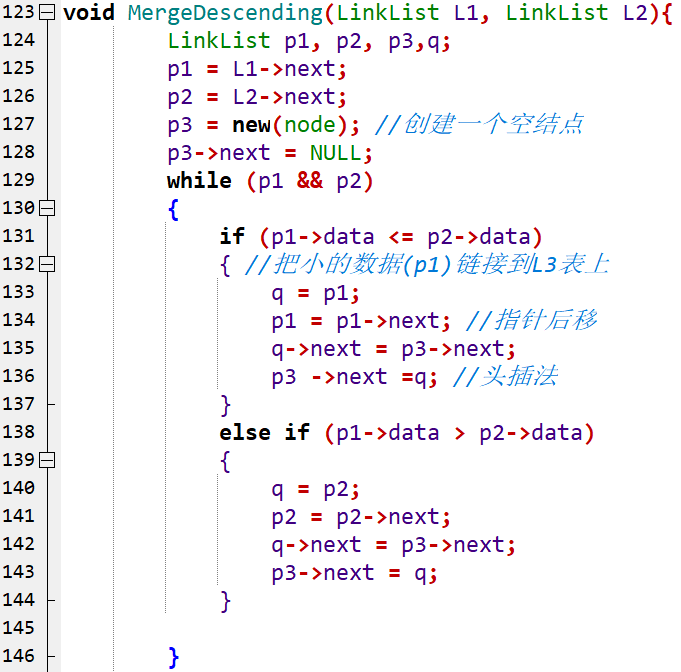
**运行结果：**

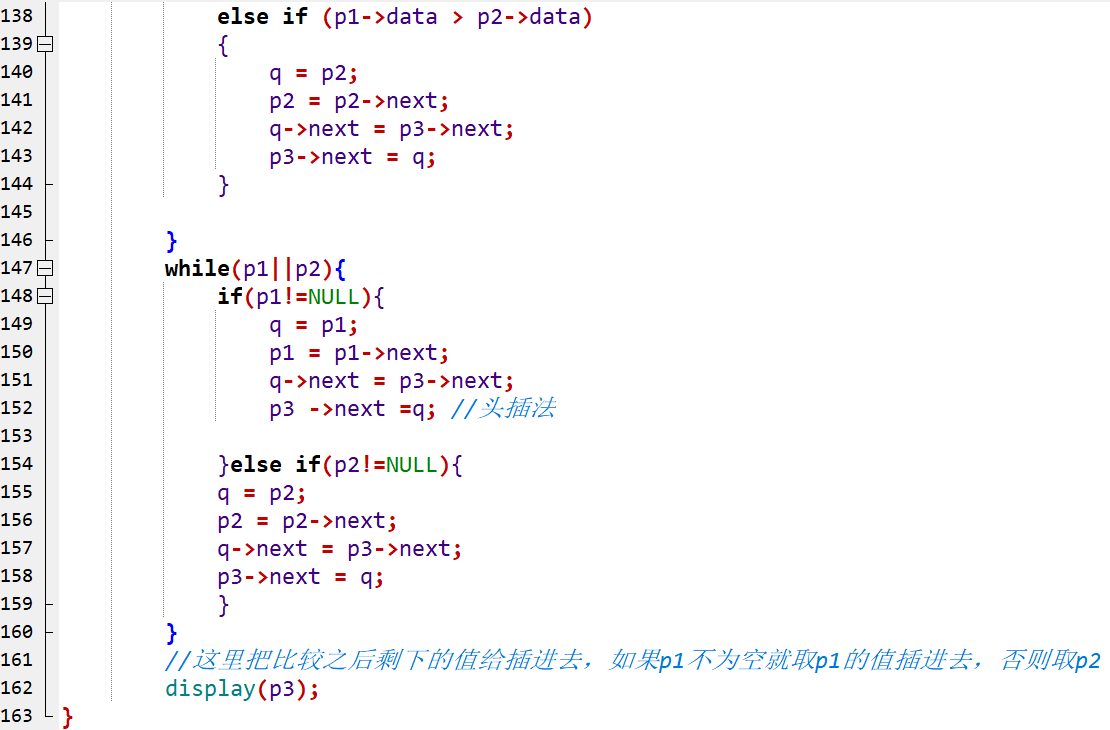


## 9.2 算法二：头插法

**算法思想：**创建新链表，用头插法，把比较后更小的那一个先插入，然后不断前插，创建出倒序单链表

**子函数代码：**





**时间复杂度：O(n)**

**空间复杂度：O(n)**

**运行结果：**



# 10. 判断一个单链表是否对称

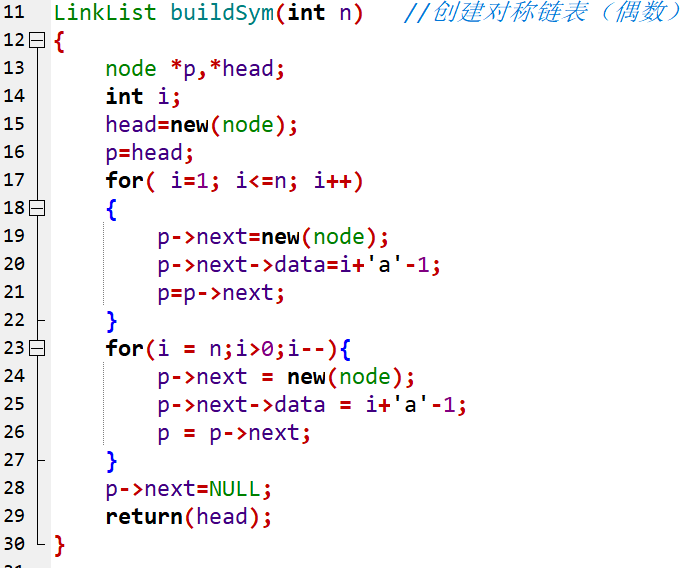
## 10.1 算法一：直接比较

**算法思想：**直接比较，设立两个指针p和q，p都从头出发，p与q位置对称，循环比较数据，由于没有前驱节点，只能每循环一次将q放回原点再走，走到中点时停下来，过程中数据如果出现任何不相等即停止。

**子函数代码：**

先创建对称的单链表：

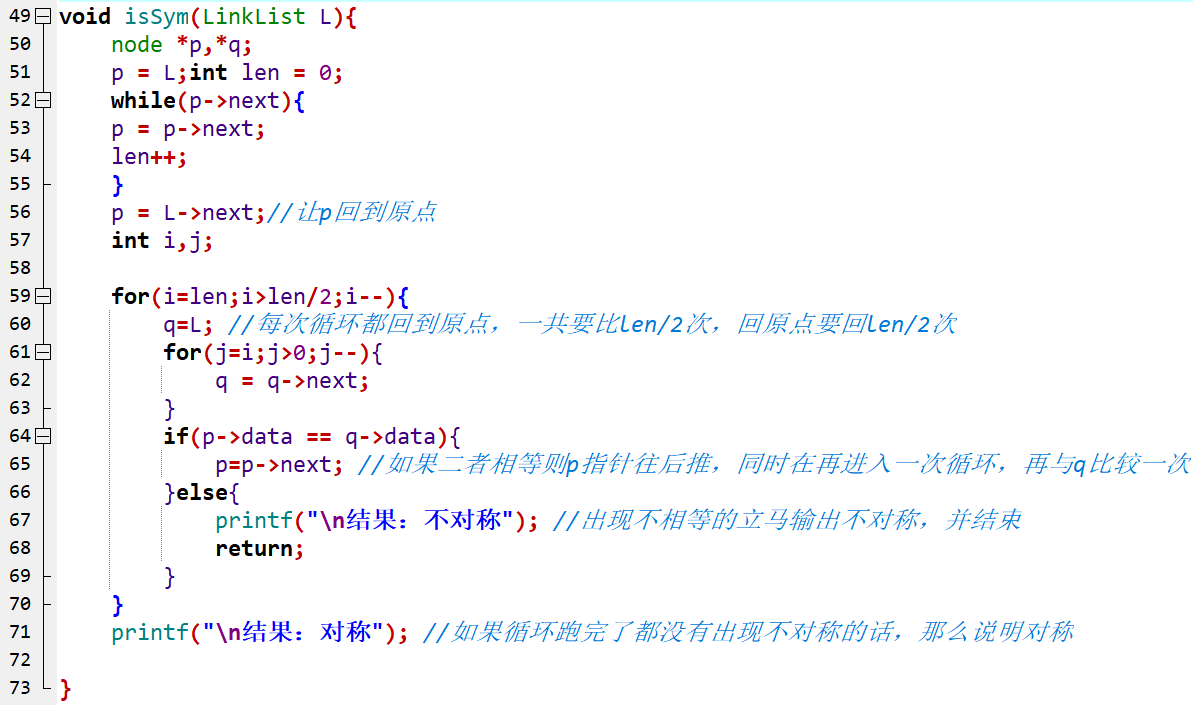
（1）偶数对称



（2）奇数对称



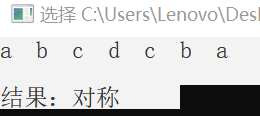
然后开始进行比较：



**时间复杂度：** O（n）

**空间复杂度：**O（1）

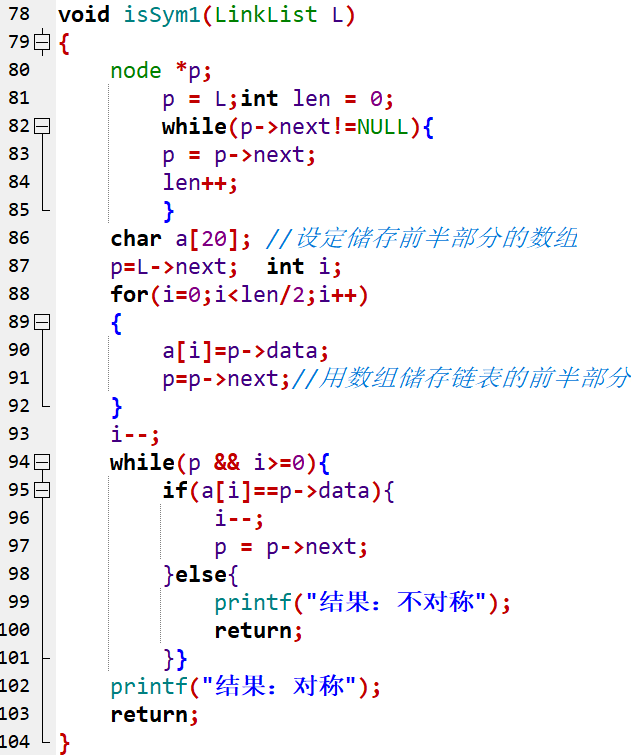
**运行结果：**



## 10.2 算法二：利用数组

**算法思想：**用数组，把前一半数据储存在数组里面，然后倒过来，从后往前和后半段的数据一一比较，如果没有出现不相等的情况则对称

**子函数代码：**



**时间复杂度：**O（n）

**空间复杂度：**O（1）

**运行结果：**

