**作业 HW1\* 实验报告**

1. **涉及数据结构和相关背景**

本次实验基于线性表的相关知识，运用基本的线性表操作（如线性表的初始化、赋值、求长度等）进行对应的程序设计，实现题目要求的功能。具体为：1.一维静态数组及其操作；2.链表及其操作。

**2. 实验内容**

**2.1 轮转数组**

**2.1.1 问题描述**

给定一个整数顺序表nums，将顺序表中的元素向右轮转 k 个位置，其中 k 是非负数。

**2.1.2 基本要求**

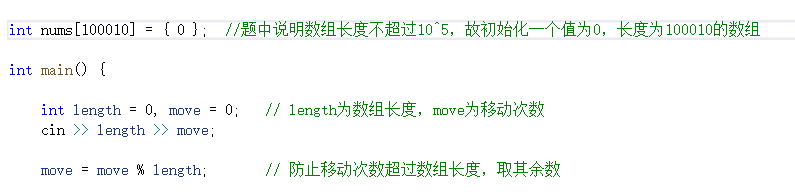
输入：第一行两个整数n和k，分别表示nums的元素个数n，和向右轮转k个位置；

第二行包括n个整数，为顺序表nums中的元素。

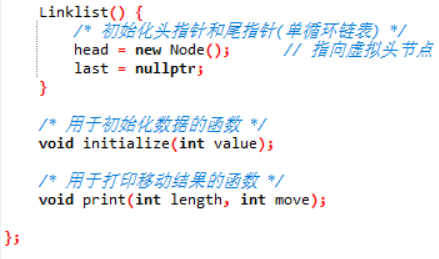
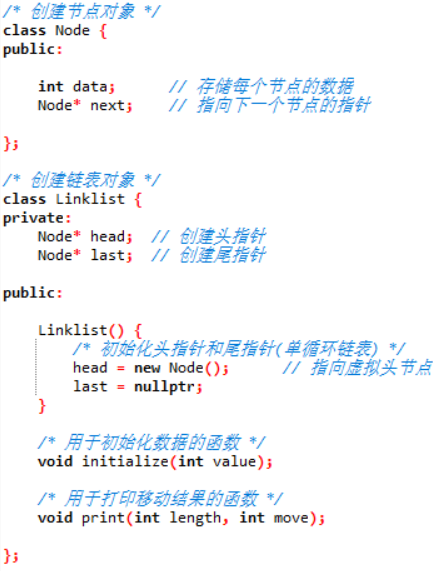
输出：轮转后的顺序表中的元素。

**2.1.3 数据结构设计**

方法一&二：定义并初始化一个数组：



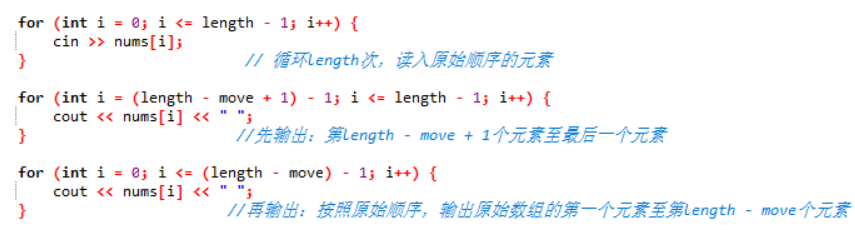
方法三：建立一个单循环链表



**2.1.4功能说明（函数、类）**

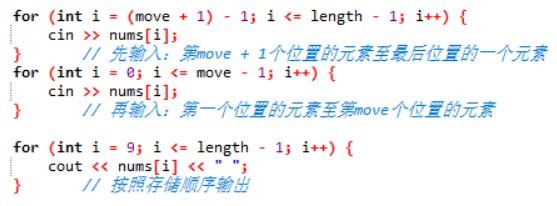
**方法一**：建立一维静态数组，按照原始数据的顺序读入数据。在输出时，先从倒数第k个元素输出到最后一个元素，再从第一个元素输出到第n-k个元素。

这样做的时间复杂度为O(n)，空间复杂度为O(1)



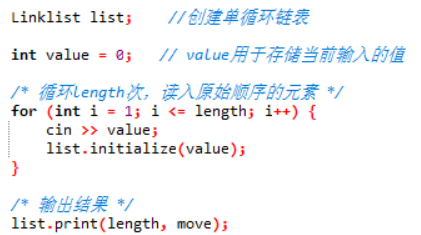
**方法二**:建立一维静态数组，存储时，先将第一个数据到第n-k个数据存储在数组的第k+1个位置到第n个位置；再将第n-k+1个数据到第n个元素存储在数组的第1个位置到第k个位置

这样做的时间复杂度为O(n)，空间复杂度为O(1)



**方法三**：建立一个单循环链表，按照原始数据的顺序读入数据，输出时找到第n-k+1个节点，从其正向输出一圈(即输出到第n个节点后，从第一个节点输出到第n-k个节点)

这样做的时间复杂度位O(n)，空间复杂度位O(1)



**2.1.5 调试分析（遇到的问题和解决方法）**

调试过程中首先输入样例中给出的测试数据，查看程序的输出是否一致，确保一定的输出准确性；

再使用一些特殊的边界数据进行测试。如本题中可输入k>n的数据，来检验是否能够正确轮转输出。

**2.1.6 总结和体会**

本题虽然思路比较简单，但是要求运用至少三种方法进行程序设计，这就对思维要求更高，同时还要联系课堂所学知识，对思考题进行解答。总之，做完此次题目，我觉得我的思维更开阔，同时对课堂所学的理解也更深刻了。

**2.2 求级数**

**2.2.1 问题描述**

求级数：A+2A2+3A3+……+NAN =IMG_256的整数值

**2.2.2 基本要求**

输入：若干行，在每一行中给出整数N和A的值，（1<=N<=150，0<=A<=15）

对于20%的数据，有 1<=N<=12，0<=A<=5；对于40%的数据，有1<=N<=18，0<=A<=9

对于100%的数据，有1<=N<=150，0<=A<=15

输出：对于每一行，在一行中输出级数的值

**2.2.3 数据结构设计**

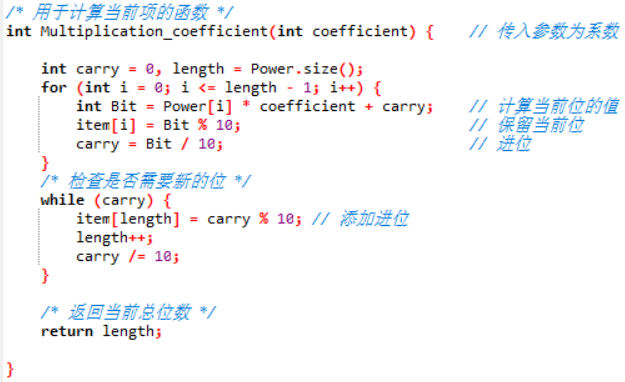
****

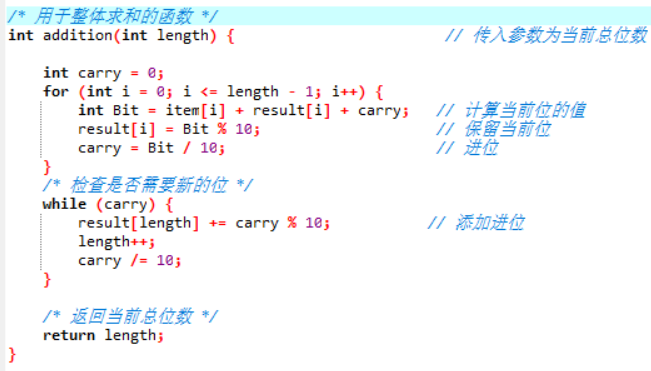
**2.2.4功能说明（函数、类）**

此处设计了三个函数（1.计算幂的函数；2.计算当前项的函数；3.求和函数）以便后续调用

****

某些项求幂后会超过long long型的数据范围，所以改用数组存储每一位的值，防止数据溢出产生的错误。

****

****

**2.2.5 调试分析（遇到的问题和解决方法）**

数据测试的过程中发现，某些项求幂后会超过long long型的数据范围，所以我修改了第一次用long long型存储当前项的方法，改用数组存储每一位的值，防止数据溢出产生的错误。

**2.2.6 总结和体会**

基于调试过程中发现的问题，对程序进行多次优化：1.计算过程模拟竖式乘法，先计算每一位的值，再进位；2.由于该题的前一项与后一项的次数仅差一位，所以每次更新幂的计算结果以便下一项计算节约时间

优化过程中提高了自己发现问题、解决问题的能力。

**2.3 扑克牌游戏**

**2.3.1 问题描述**

对于一个扑克牌堆，定义以下4种操作命令：1) 添加2) 抽取3)反转4)弹出

初始时牌堆为空。输入n个操作命令（1 ≤ n ≤200），执行对应指令。所有指令执行完毕后打印牌堆中所有牌花色和数字（从牌堆顶到牌堆底），如果牌堆为空，则打印NULL。

对于20%的数据，n<=20，有Append、Pop指令；

对于40%的数据，n<=50，有Append、Pop、Revert指令；

对于100%的数据，n<=200，有Append、Pop、Revert、Extract指令。

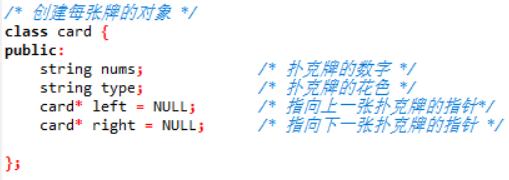
**2.3.2 基本要求**

输入：第一行输入一个整数n，表示命令的数量。接下来的n行，每一行输入一个命令。

输出：输出若干行，每次收到Pop指令后输出一行，最后将牌堆中的牌从牌堆顶到牌堆底逐一输出（花色和数字），若牌堆为空则输出NULL

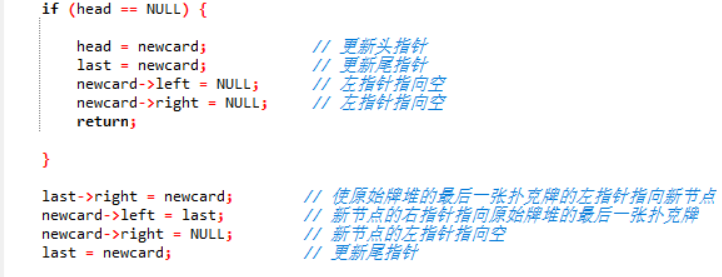
**2.3.3 数据结构设计**

定义两个类，4个函数（对应四种操作）

****

****

**2.3.4功能说明（函数、类）**

****

关于Append操作的解决：

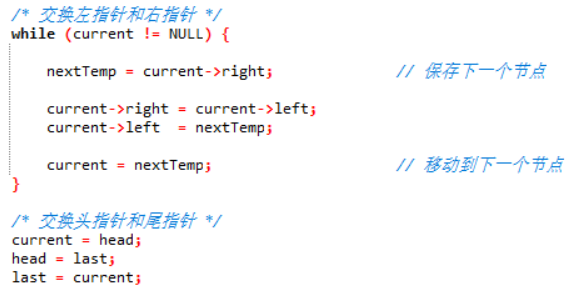
设置尾指针，便于判断牌底的扑克牌，插入新的扑克牌后，更新尾指针

****

关于Extract操作的解决：

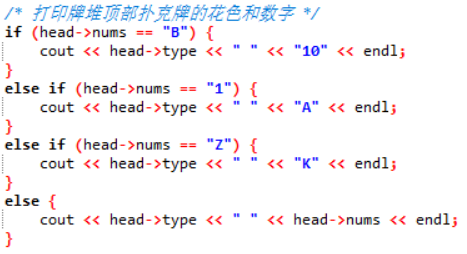
先遍历链表，将所有符合花色的扑克牌都移动至牌顶(先不排序)

对符合花色的扑克牌的数据进行排序(不对链表的节点排序，而是对数据排序)，将排序后的数据赋值给每个链表节点

****

关于Revert操作的解决：

选择双链表结构，不移动链表的节点，而是依次交换每个节点左右指针，最后交换头尾指针

****

关于pop操作的解决：

设置头指针

1.判断链表是否为空

2.判断牌顶的扑克牌

每次弹出扑克牌，更新头指针

**2.3.5 调试分析（遇到的问题和解决方法）**

排序时的简化操作：关于扑克牌数值的比较——将A改为1，将10改为B，将K改为Z，这样可以直接用ASCll码进行排序。不过输出时记得改回。

**2.3.6 总结和体会**

使用改良方法时一定不要脱离原有题目！！比如我在简化操作的时候忘记把1改回A，导致程序一直执行错误。

**2.4 一元多项式的相加和相乘**

**2.4.1 问题描述**

实现多项式的相加和相乘运算。（输入保证是按照指数项递增有序的）

对于%15的数据，有1<=n,m<=15；对于%33的数据，有1<=n,m<=50

对于%66的数据，有1<=n,m<=100；对于100%的数据，有1<=n,m<=2050

**2.4.2 基本要求**

输入：第1行一个整数m，表示第一个一元多项式的长度

第2行有2m个整数，p1 e1 p2 e2 ...，中间以空格分割,表示第1个多项式系数和指数

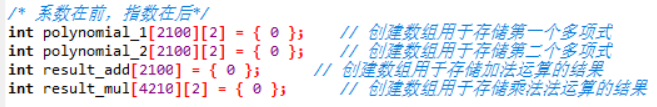
第3行一个整数n,表示第二个一元多项式的项数

第4行有2n个整数，p1 e1 p2 e2 ...，中间以空格分割,表示第2个多项式系数和指数

第5行一个整数，若为0,执行加法运算并输出结果，若为1，执行乘法运算并输出结果；若为2，输出一行加法结果和一行乘法的结果。

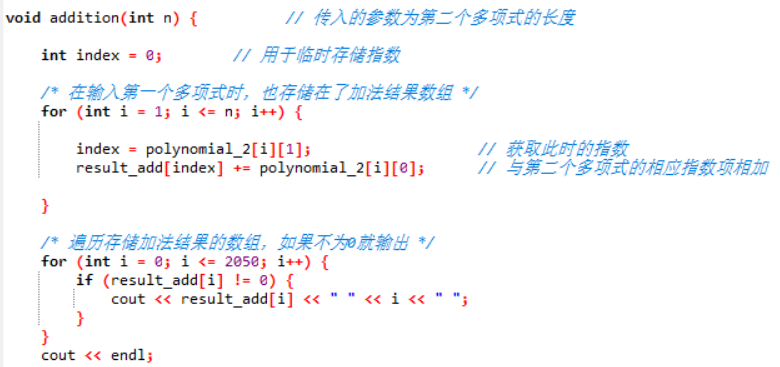
输出：运算后的多项式链表，要求按指数从小到大排列。当运算结果为0 0时，不输出。

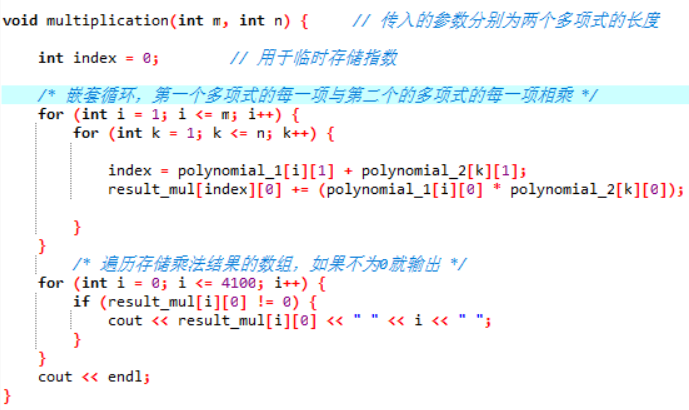
**2.4.3 数据结构设计**

****

**2.4.4功能说明（函数、类）**

2个函数

****

****

**2.4.5 调试分析（遇到的问题和解决方法）**

调试过程中发现时间复杂度有些高，因此采用以下方法——存储计算结果的数组用下标存储指数，避免了因为判断两个多项式是否存在某个指数而浪费时间。

**2.4.6 总结和体会**

该题目主要使用线性表结构(主要为一维数组)。通过此题了解到多项式数组存储指数和系数，可以避免因为遍历指数而浪费时间。是一个值得积累的算法思想。

**2.5学生信息管理**

**2.5.1 问题描述**

定义一个包含学生信息（学号，姓名）的的 顺序表，使其具有如下功能：

(1)根据指定学生个数，逐个输入学生信息；(2) 给定一个学生信息，插入到表中指定的位置；(3) 删除指定位置的学生记录；(4) 分别根据姓名和学号进行查找，返回此学生的信息；

(5) 统计表中学生个数。

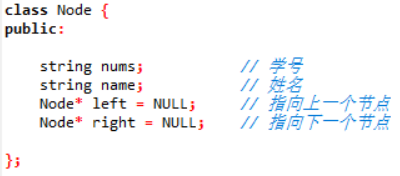
**2.5.2 基本要求**

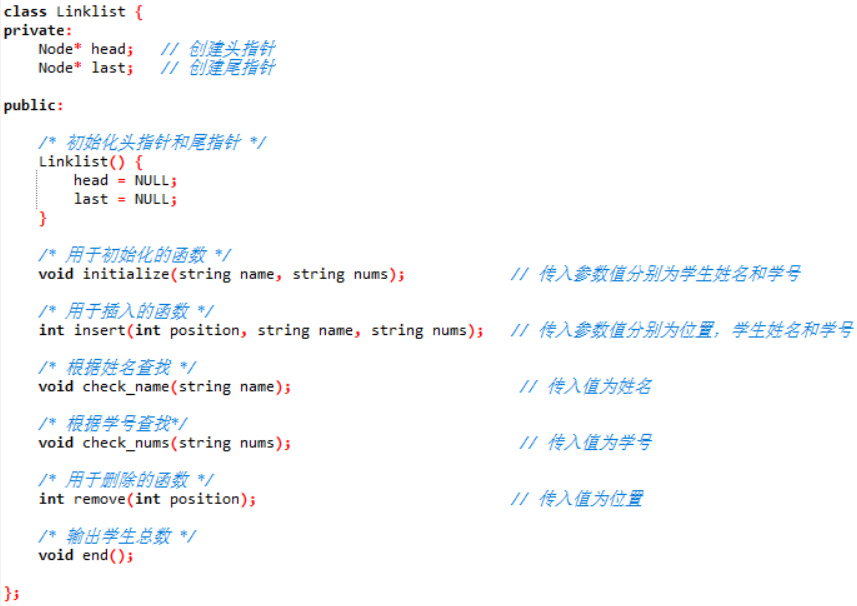
输入&输出：第1行是学生总数n

接下来n行是对学生信息的描述，每行是一名学生的学号、姓名，用空格分割；(学号、姓名均用字符串表示,字符串长度<100)  
接下来是若干行对顺序表的操作：(每行内容之间用空格分隔)

注：全部数值 <= 10000，元素位置从1开始。 学生信息有重复数据（输入时未做检查），查找时只需返回找到的第一个。 每个操作都在上一个操作的基础上完成。

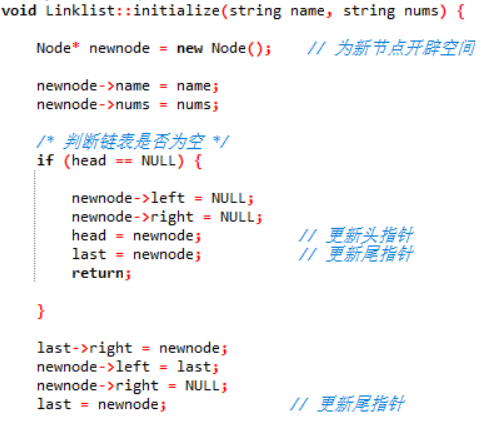
**2.5.3 数据结构设计**

****

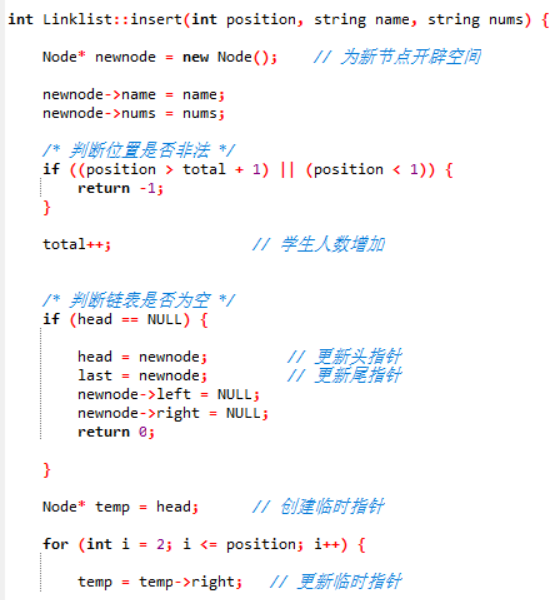
****

**2.5.4功能说明（函数、类）**

5个函数对应五种操作

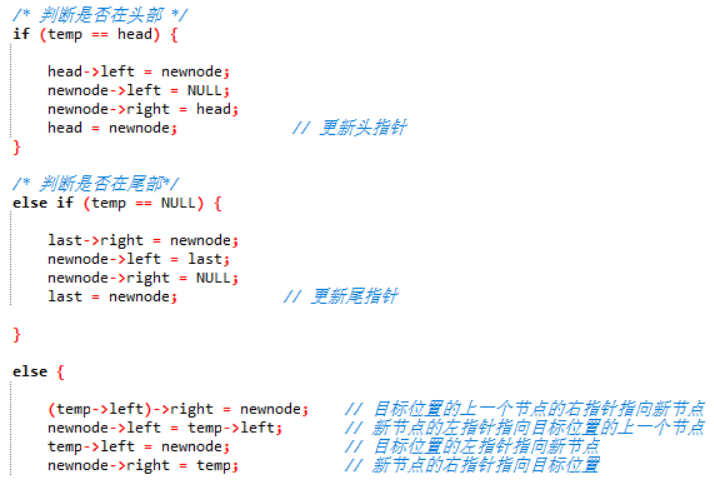


Insert

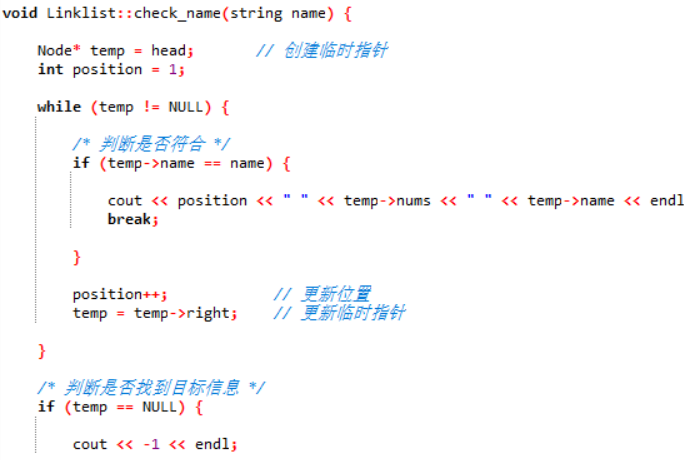


关于insert函数：

遍历链表，到达目标位置



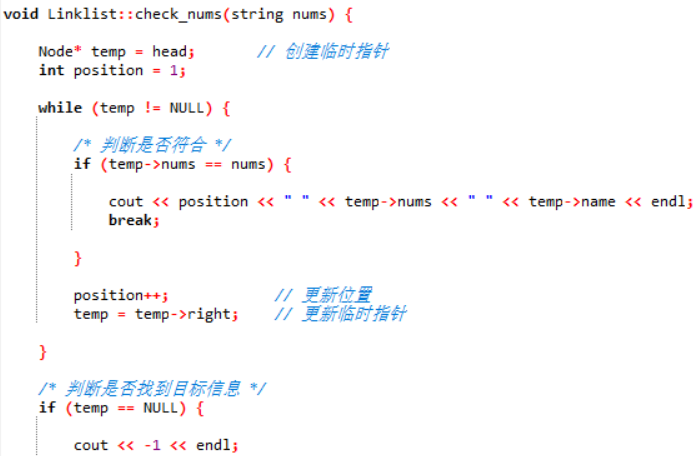
Check\_name



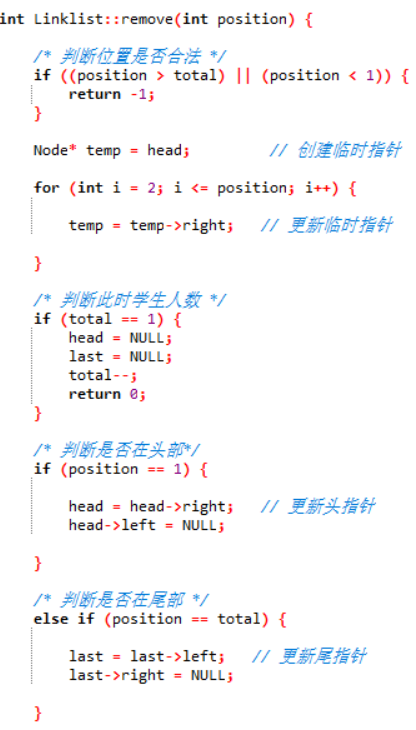
关于check函数

遍历链表，找到匹配的第一个节点

Check\_nums

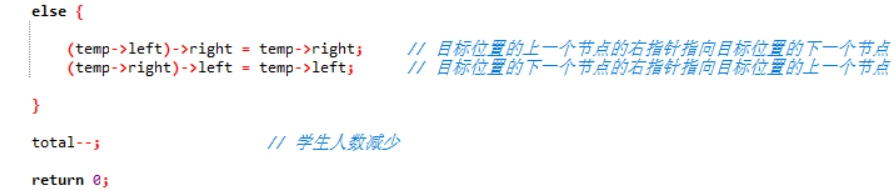


Remove



关于remove函数:

遍历链表，到达目标位置



**2.5.5 调试分析（遇到的问题和解决方法）**

调试时发现linux系统和windows系统的ASCll表好像有差异。oj为为linux系统，用getline读数据，换行符windows里是读\n，linux系统是读\r，所以在oj写程序时，尽量使用cin，这样不容易出错。

**2.5.6 总结和体会**

由于此题的操作主要涉及插入，删除的操作，所以采用链表结构。

**3. 实验总结**

经过此次实验作业，我熟悉了线性表的基本操作，并能运用到解决实际问题的实践中。同时，在独立调试解决问题的过程中，深感个人发现问题和与别人讨论问题的能力显著提高。