@TOC

左旋字符串中的k个字符

例如:

> ABCD左旋一个字符得到BCDA
>
> ABCD左旋两个字符得到CDAB

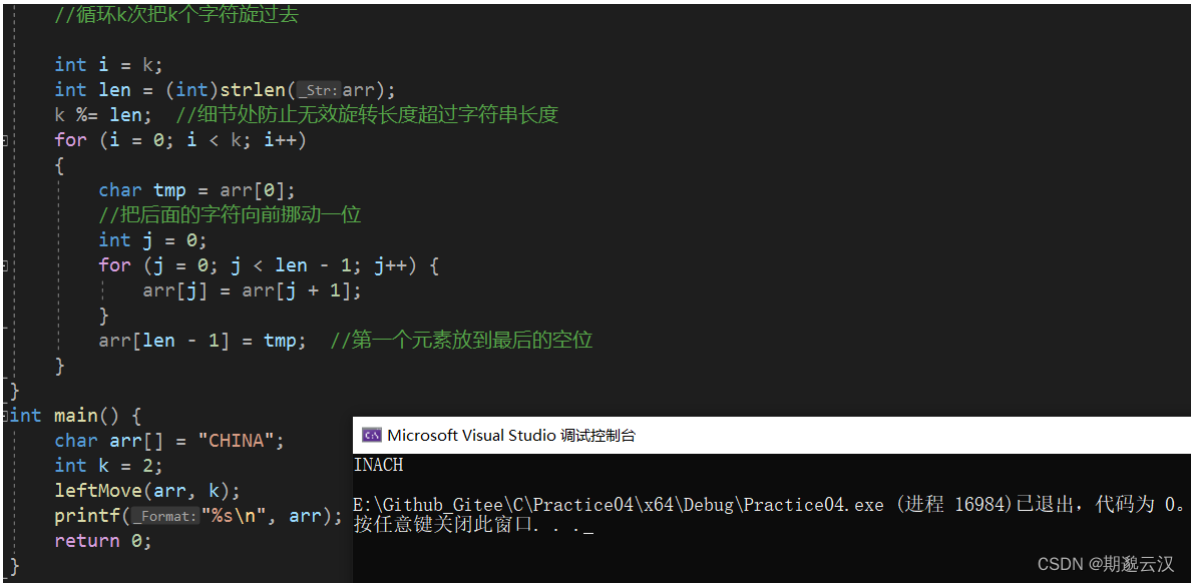
这个问题我们首先考虑移动字符串中的字符，左旋即把字符串左边的字符旋转到末尾，那么旋转k个字符，那我们就移动k次字符串。

考虑该函数的参数:

> 一个是要把目标字符串地址传入函数，
>
> 另外就是旋转的字符个数k

```
void leftMove(char arr[], int k) {
    //循环 k次把 k个字符旋过去
    int i = k;
    int len = (int)strlen(arr);

    k %= len;  //细节处防止无效旋转长度超过字符串长度
    for (i = 0; i < k; i++)
    {
        char tmp = arr[0];  //拿到最左边的元素
        //把后面的字符向前挪动一位
        int j = 0;
        for (j = 0; j < len - 1; j++) {
            arr[j] = arr[j + 1];
        }
        arr[len - 1] = tmp;  //第一个元素放到最后的空位
    }
}
```

测试一下:



这里使用双重for循环，外层控制几轮旋转，也就是我需要旋转几个字符。内层for循环来实现把最左边的字符放到字符串末尾，先将arr[0]拿出来保存到tmp，再逐个挪动后面剩下的字符向前位移，循环结束后再将tmp放到字符串末尾。这样即可实现字符串旋转。

更巧妙的实现思路

考虑逆序方法

这种方法的思路是：**前k个逆序，后len - k个逆序，再整体逆序**

例如：还是”CHINA“

要旋转两个字符“CH”，那么先把CH逆序得到HC，再把后面的剩下的字符逆序为ANI，最后把整个字符串逆序得到 INACH

注意：这里逆序都是在同样的字符串上原地逆序

```
#include <assert.h>
void reverse(char* left, char* right) {
    assert(left && right);  //断言两个指针非空
    while (left < right) {
        char tmp = *left;
        *left = *right;
        *right = tmp;
        left++;
        right--;
    }
}
void Left_move(char arr[], int k) {
    int len = (int)strlen(arr);
    reverse(arr, arr + k - 1);
    reverse(arr + k, arr + len - 1);
    reverse(arr, arr + len - 1);
}
```

结果是显然的：

```c
void reverse(char* left, char* right) {
    assert(left && right);  //断言两个指针非空
    while (left < right) {
        char tmp = *left;
        *left = *right;
        *right = tmp;
        left++;
        right--;
    }
}
void Left_move(char arr[], int k) {
    int len = (int)strlen(_Str:arr);
    reverse(left:arr, right:arr + k - 1);
    reverse(left:arr + k, right:arr + len - 1);
    reverse(left:arr, right:arr + len - 1);
}
int main() {
    char arr[] = "abcde";
    int k = 2;  //左旋三个字符
    Left_move(arr, k:3);
    printf(_Format:"%s\n", arr);
    return 0;
}
```

```
Microsoft Visual Studio 调试控制台
deabc

E:\Github_Gitee\C\Practice04\x
按任意键关闭此窗口. . ._
```

CSDN @期邈云汉

🌟 判断是否左旋

有这样的问题，判断一个字符串是否为另外一个字符串旋转之后的字符串

例如：

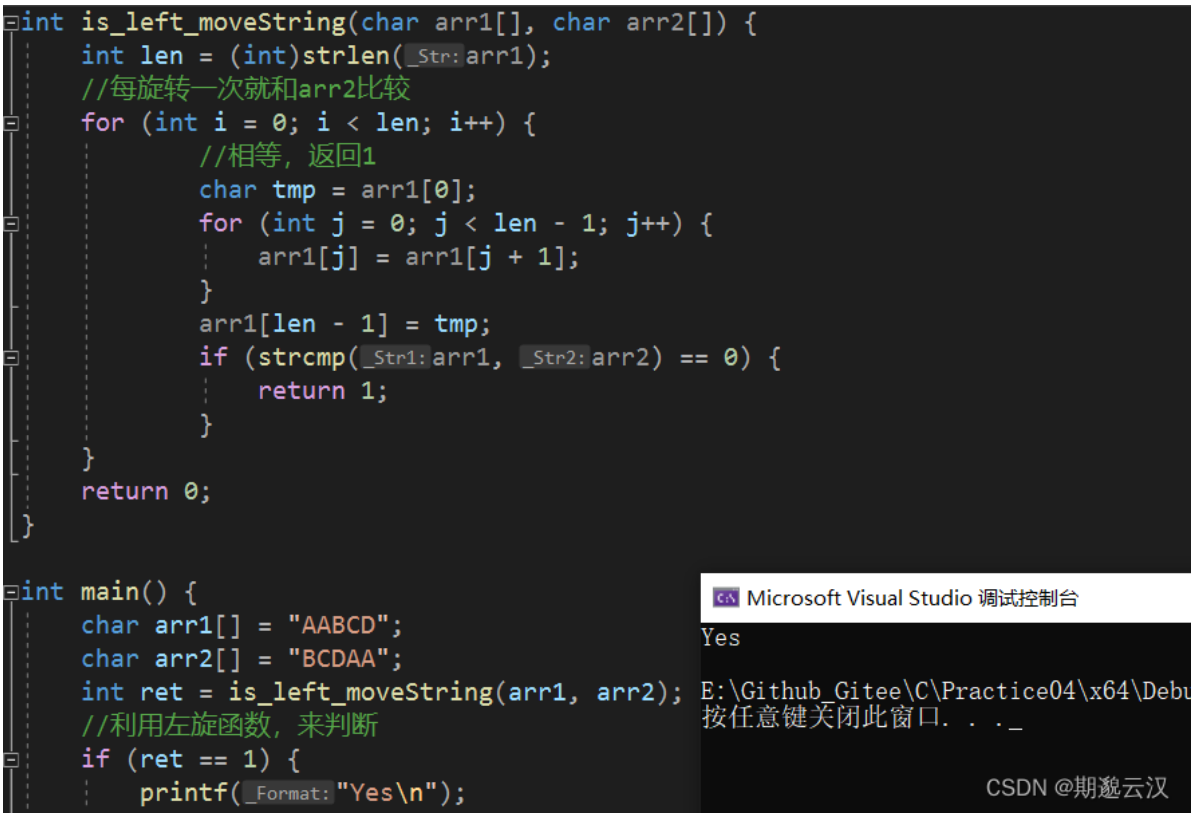
```
给定s1 = AABCD和s2 = BCDAA，返回1

给定s1 = abcd和s2 = ACBD，返回0
```

想到用上述的左旋函数来判断，那么相当于暴力变量，采用循环来将字符串旋转1 2 ....k次，每次结束判断两个字符串是否相等，相等就返回1，否则返回0。这样的思路是很直观的。

```c
#include <string.h>
int is_left_moveString(char arr1[], char arr2[]) {
    int len = (int)strlen(arr1);
    //每旋转一次就和arr2比较
    for (int i = 0; i < len; i++) {
            //相等，返回1
            char tmp = arr1[0];
            for (int j = 0; j < len - 1; j++) {
                arr1[j] = arr1[j + 1];
            }
            arr1[len - 1] = tmp;
            if (strcmp(arr1, arr2) == 0) {
```

```
                return 1;
            }
        }
        return 0;
    }
```

测试一下，结果显然：



考虑更巧妙的方法（自身追加）

验证下面的说法：

一个字符串后面追加自身，那么**新的字符串就可以包含原字符串所有旋转的情况**，再判断另一个字符串是不是新长串的子串，例如：**就考虑 CHINA**

那么自身追加后字符串变为CHINACHINA，仔细去数，发现确实这个题目思想本身可能就是自身追加后字符串的特性。

```
int up_judge_leftMove(char arr1[], char arr2[]) {
    int len1 = (int)strlen(arr1);
    int len2 = (int)strlen(arr2);
    if (len1 != len2) {
        //说明不可能是旋转得来的
        return 0;
    }

    strcat(arr1, arr1);
    char* ret = strstr(arr1, arr2);
    if (ret == NULL) {
        return 0;
    }
    else {
```

```
        return 1;
    }
}
int main() {
    char arr1[] = "CHINA";
    char arr2[] = "NACHI";

    int ret = up_judge_leftMove(arr1, arr2);
    if (ret == 1) {
        printf("Yes\n");
    }
    else {
        printf("No\n");
    }
}
```

验证一下结果：