目录

[面试题58 - II. 左旋转字符串（基础题，但是不熟练，字符串、数组、指针） 1](#_Toc40219377)

# [面试题58 - II. 左旋转字符串](https://leetcode-cn.com/problems/zuo-xuan-zhuan-zi-fu-chuan-lcof/)（基础题，但是不熟练，字符串、指针）

C：

1. **char**\* reverseLeftWords(**char**\* s, **int** n){
2. **int** l=strlen(s);//字符串长度
3. **char**\* result=(**char**\*)malloc(**sizeof**(**char**)\*(l+1));
4. **for**(**int** i=n%l;i<l;++i){//取余未免n超出范围
5. \*result=s[i];
6. ++result;
7. }
8. **for**(**int** i=0;i<n%l;++i){
9. \*result=s[i];
10. ++result;
11. }
12. \*result='\0';
13. **return** result-l;
14. }

C++:

1. **class** Solution {
2. **public**:
3. string reverseLeftWords(string s, **int** n) {
4. string temp(s,0,n%s.length());
5. string result(s,n%s.length(),s.length()-n%s.length());
6. result+=temp;
7. **return** result;
8. }
9. };

[面试题05. 替换空格](https://leetcode-cn.com/problems/ti-huan-kong-ge-lcof/)（基础题，但不熟练：指针、字符串）

C：

1. **char**\* replaceSpace(**char**\* s){
2. // int l=strlen(s);
3. // char\*result=malloc(sizeof(char)\*3\*(l+1));
4. // int cnt=0;
5. // for(int i=0;i<l;++i){
6. //     if(s[i]==' '){
7. //         \*result='%';
8. //         ++result;
9. //         \*result='2';
10. //         ++result;
11. //         \*result='0';
12. //         cnt+=3;
13. //     }
14. //     else{
15. //         \*result=s[i];
16. //         ++cnt;
17. //     }
18. //     ++result;
19. // }
20. // \*result='\0';
21. // return result-cnt;
23. //it could be simplied like below
24. **char**\*result=malloc(**sizeof**(**char**)\*3\*(strlen(s)+1));
25. **int** i=0;
26. **while**(\*s!='\0'){
27. **if**(\*s==' '){
28. result[i++]='%';
29. result[i++]='2';
30. result[i++]='0';
31. }
32. **else**{
33. result[i++]=\*s;
34. }
35. s++;
36. }
37. result[i]='\0';
38. **return** result;
39. }

#### [面试题03. 数组中重复的数字](https://leetcode-cn.com/problems/shu-zu-zhong-zhong-fu-de-shu-zi-lcof/)（数组，利用C++简单，但C不简单）

C++:

1. **class** Solution {
2. **public**:
3. **int** findRepeatNumber(vector<**int**>& nums) {
4. //another solution, used STL adjacent\_find
5. sort(nums.begin(),nums.end());
6. /\*iterator adjacent\_find(iterator first, iterator last)
7. 返回第一次出现连续两个元素相等的位置\*/
8. **return** \*adjacent\_find(nums.begin(),nums.end());
10. //the simplest solution
11. // sort(nums.begin(),nums.end());
12. // int i;
13. // for(i=0;i<nums.size();++i){
14. //     if(nums[i]==nums[i+1]){
15. //         break;
16. //     }
17. // }
18. // return nums[i];
19. }
20. };

书中解法：

从头到尾依次扫描这个数组中的每个数字。当扫描到下标为 i 的数字时，首先比较这个数字（用m表示）是不是等于i。如果是，则接着扫描下一个数字；如果不是，则再拿它和第m个数字进行比较。如果它和第m个数字相等，就找到了一个重复的数字（该数字在下标为i和m的位置都出现了）；如果它和第m个数字不相等，就把第i个数字和第m个数字交换，把m放到属于它的位置。接下来再重复这个比较、交换的过程，直到我们发现一个重复的数字。

1. **int** findRepeatNumber(**int**\* nums, **int** numsSize){
2. **for**(**int** i = 0; i < numsSize;) {
3. **if**(nums[i] == i) {
4. i++;
5. }
6. **else** {
7. **int** m = nums[i];
8. **if**(m == nums[m]) {
9. **return** m;
10. }
11. **else** {
12. nums[i] = nums[m];
13. nums[m] = m;
14. }
15. }
16. }
17. // 必须有一个一定能执行的返回语句，不然会报错
18. **return** -1;
19. }

标记法：

1. **int** findRepeatNumber(**int**\* nums, **int** numsSize){
2. unsigned **char** remark[numsSize];
3. memset(remark, 0, numsSize);
4. **for**(**int** i=0; i<numsSize; ++i){
5. **if**(remark[nums[i]]) **return** nums[i];
6. remark[nums[i]] = 1;
7. }
8. **return** -1;
9. }