给定一个数组，将数组中的元素向右移动 k 个位置，其中 k 是非负数。

示例 1:

输入: [1,2,3,4,5,6,7] 和 k = 3

输出: [5,6,7,1,2,3,4]

解释:

向右旋转 1 步: [7,1,2,3,4,5,6]

向右旋转 2 步: [6,7,1,2,3,4,5]

向右旋转 3 步: [5,6,7,1,2,3,4]

示例 2:

输入: [-1,-100,3,99] 和 k = 2

输出: [3,99,-1,-100]

解释:

向右旋转 1 步: [99,-1,-100,3]

向右旋转 2 步: [3,99,-1,-100]

说明:

尽可能想出更多的解决方案，至少有三种不同的方法可以解决这个问题。

要求使用空间复杂度为 O(1) 的原地算法。

===========================================================

class Solution {

public:

void rotate(vector<int>& nums, int k) {

vector<int> tem(nums); // 把nums复制到tem中

for (int i = 0; i < nums.size(); ++i)

{

nums[(i + k) % nums.size()] = tem[i]; // -3%7=-3，因此正数的余数才为正数

}

}

};

34 / 34 个通过测试用例

状态：通过

执行用时：20 ms

Vector复制方法（assign函数；vector<int> 复制品(源);）

#include<iostream>

#include<vector>

using namespace std;

int main()

{

vector<int> A(3,100);

vector<int> B(5,200);

A.assign(B.begin(),B.end()); // 把B的内容复制给A，AB独立，互不影响

int i;

for(i=0;i<A.size();i++)

{

cout<<A[i]<<endl;

}

for(i=0;i<B.size();i++)

{

cout<<B[i]<<endl;

}

A[0] = 666;

cout << A[0] << "|" << B[0] << endl;

vector<int> C(B); // 第二种，把B的内容复制给C,BC独立，互不影响

for(i=0;i<C.size();i++)

{

cout<<C[i]<<endl;

}

C[0] = 666;

cout << C[0] << "|" << B[0] << endl;

cout << -3%7 << endl;

return 0;

}

===========================================================

翻转字符的方法，思路是先把前n-k个数字翻转一下，再把后k个数字翻转一下，最后再把整个数组翻转一下：

1 2 3 4 5 6 7

4 3 2 1 5 6 7

4 3 2 1 7 6 5

5 6 7 1 2 3 4

class Solution {

public:

void rotate(vector<int>& nums, int k) {

if (nums.empty() || (k %= nums.size()) == 0) return;

int n = nums.size();

reverse(nums.begin(), nums.begin() + n - k);

reverse(nums.begin() + n - k, nums.end());

reverse(nums.begin(), nums.end());

}

};

===========================================================

由于旋转数组的操作也可以看做从数组的末尾取k个数组放入数组的开头，所以我们用STL的push\_back和erase可以很容易的实现这些操作。

push\_back：在vector尾部加入一个数据。

erase：从指定容器删除指定位置的元素或某段范围内的元素

class Solution {

public:

void rotate(vector<int>& nums, int k) {

if (nums.empty() || (k %= nums.size()) == 0) return;

int n = nums.size();

for (int i = 0; i < n - k; ++i) {

nums.push\_back(nums[0]);

nums.erase(nums.begin()); // 删除nums的第一个元素

}

}

};

===========================================================

通过不停的交换某两个数字的位置来实现旋转，数组改变过程如下：

1 2 3 4 5 6 7

5 2 3 4 1 6 7

5 6 3 4 1 2 7

5 6 7 4 1 2 3

5 6 7 1 4 2 3

5 6 7 1 2 4 3

5 6 7 1 2 3 4

class Solution {

public:

void rotate(vector<int>& nums, int k) {

if (nums.empty()) return;

int n = nums.size(), start = 0;

while (n && (k %= n)) {

for (int i = 0; i < k; ++i) {

swap(nums[i + start], nums[n - k + i + start]);

}

n -= k;

start += k;

}

}

};

<http://www.cnblogs.com/grandyang/p/4298711.html> [LeetCode] Rotate Array 旋转数组