```
/*
给定一个排序数组和一个目标值，在数组中找到目标值，并返回其索引。如果目标值不存在于数组中，返回它将会被按顺序插入的位置。

你可以假设数组中无重复元素。

示例 1:

输入: [1,3,5,6], 5
输出: 2

示例 2:

输入: [1,3,5,6], 2
输出: 1

示例 3:

输入: [1,3,5,6], 7
输出: 4

示例 4:

输入: [1,3,5,6], 0
输出: 0

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/search-insert-position
著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。
*/
```

分析:

- 在按升序排列的数组中找插入位置，从低向高遍历即可，只要找到第一个大于等于目标数据的索引返回即可；
- 暴力遍历的时间复杂度是$O(n)$，方法一；
- 对于有序数组查询位置，还可以使用二分法，时间复杂度是$O(\log(n))$，方法二。

---

方法一:C_Solution,暴力遍历.

```c
int searchInsert(int* nums, int numsSize, int target)
{
    int i = 0;

    if(target <= nums[0])
    {
```

```
        return 0;
    }

    if(target > nums[numsSize-1])
    {
        return numsSize;
    }

    for(i=1;i<numsSize;i++)
    {
        if(target<=nums[i])
        {
            return i ;
        }
    }

    return -1; /* Anti warnning ,若在这里返回,那输入一定有问题.*/
}

/*
执行结果：
通过
显示详情
执行用时 :8 ms, 在所有 C 提交中击败了89.76% 的用户
内存消耗 :7.1 MB, 在所有 C 提交中击败了79.28%的用户
*/
```

---

方法二:C_Solution,二分法

```
int searchInsert(int* nums, int numsSize, int target)
{
    int i           =   0;
    int ret_val     =   0;
    int left        =   0;
    int right       =   numsSize - 1;
    int mid         =   0;
    int temp1       =   0;

    if(target <= nums[0])
    {
        return 0;
    }
    if(target > nums[numsSize-1])
    {
        return numsSize;
    }

    while(left<=right)
    {
        mid     = left + (right-left)/2; /*防止加法溢出*/
        temp1   = nums[mid];
```

```
        if( temp1 > target)
        {
            right = mid - 1;
            continue;
        }
        else if(temp1==target)
        {
            return mid;
        }
        else
        {
            if(nums[mid+1]>=target)
            {
                return mid + 1;
            }
            else
            {
                left = mid + 1;
            }
        }
    }
    return -1;/* Anti warnning ,若在这里返回,那输入一定有问题.*/
}

/*
执行结果：
通过
显示详情
执行用时 :4 ms, 在所有 C 提交中击败了99.87% 的用户
内存消耗 :7.3 MB, 在所有 C 提交中击败了66.60%的用户
*/
```

---

AlimyBreak
2019.07.19