# 数组：实现整数的数字反转

题目来源：Leetcode 7：https://leetcode-cn.com/problems/reverse-integer/

## 暴力解法：逆序输出

### Java代码

```
    /**
     * 解法一，暴力解法 思路：
     *  1.整数转字符串，再转字符数组
     *  2.反向遍历字符数组，并将元素存储到新数组中
     *  3.将新数组转成字符串，再转成整数输出
     *
     * 注意事项：
     *  边界问题
     *      数组索引越界
     *      数值溢出边界：溢出则返回0
     *  细节问题
     *      首位不为0
     *      符号处理
     * @param x 指定整数
     * @return 反转后的整数，或0
     */
    public int reverse(int x) {
        if (x == Integer.MIN_VALUE || x == Integer.MAX_VALUE) {
            // 整数类型的最小值的绝对值 比 最大值的绝对值 大1
            return 0;
        }
        int sign = x > 0 ? 1 : -1; // 符号
        x = x < 0 ? -x : x; // 无论正负，都当成正数

        // 1.整数转字符串，再转字符数组
        String str = String.valueOf(x);
        char[] chars = str.toCharArray();
        // 2.反向遍历字符数组，并将元素存储到新数组中
        int len = chars.length;
        char[] array = new char[len];
//            for (int i = len - 1; i >= 0; i--) { // 遍历原数组
//                array[len - 1 - i] = chars[i];
//            }
        for (int i = 0; i < len; i++) { // 遍历新数组
            array[i] = chars[len - 1 - i];
        }
        // 3.将新数组转成字符串，再转成整数输出
//            Integer result = Integer.valueOf(String.valueOf(array));
        long value = Long.valueOf(String.valueOf(array));
        boolean b = value > Integer.MAX_VALUE || value < Integer.MIN_VALUE;
        int result = b ? 0 : (int)value; // 数值越界：溢出则返回0
        return result * sign;
    }
```

# 优化解法：首尾交换

## Java代码

```java
/**
 * 解法二，优化解法 思路：
 *  1.整数转字符串，再转字符数组
 *  2.交换首位(start)和末位(end)数字
 *  3.循环操作：依次交换第二(start++)和倒数第二个(end--)
 *  直到数组剩下1个或0个元素
 *  4.将原数组转成字符串，再转成整数输出
 *
 * 注意事项：
 *  边界问题
 *      数组索引越界：数组长度为偶数，反转完成标志为start>end;
 *          为奇数时反转完成标志为start==end
 *      数值溢出边界：溢出则返回0
 *  细节问题
 *      首位不为0
 *      符号处理
 * @param x 指定整数
 * @return 反转后的整数，或0
 */
public int reverse(int x) {
    if (x == Integer.MIN_VALUE || x == Integer.MAX_VALUE) {
        // 整数类型的最小值的绝对值 比 最大值的绝对值 大1
        return 0;
    }
    int sign = x > 0 ? 1 : -1; // 符号
    x = x < 0 ? -x : x; // 无论正负，都当成正数

    // 1.整数转字符串，再转字符数组
    String str = String.valueOf(x);
    char[] chars = str.toCharArray();

    // 2.交换首位(start)和末位(end)数字
    // 3.循环操作：依次交换第二(start++)和倒数第二个(end--)
    int start = 0, end = chars.length - 1;
    while (start < end) { // 反转完成的标志：start >= end
        // 交换两端等距离的元素
        char temp = chars[start];
        chars[start] = chars[end];
        chars[end] = temp;

        start++;
        end--;
    }
    // 4.将原数组转成字符串，再转成整数输出
    long value = Long.valueOf(String.valueOf(chars));
    boolean b = value > Integer.MAX_VALUE || value < Integer.MIN_VALUE;
    int result = b ? 0 : (int)value;

    return result * sign;
}
```

# 最优解：数学思维解法

## java代码

```java
/**
 * 最优解法 数学解法思路：
 *  1.尝试拿个位数字
 *      对10取模运算得到个位数字
 *  2.让每一位数字变成个位数字
 *      先除以10，再对10取模得到十位数字
 *      循环上述操作
 *  3.将每一位数字计算累加
 *      将上次累加结果*10 + 新数字
 *
 * 注意事项：
 *  边界问题：
 *      从低位到高位处理，最高位结束
 *          最高位 / 10 == 0
 *          最高位 % 10 == 最高位
 *      数值溢出边界：溢出则返回0
 *          用long类型存放，溢出int则返回0
 *          新整数补充最后一位前判断溢出
 *  细节问题：
 *      首位不为0
 *      符号处理
 * @param x 指定整数
 * @return 反转后的整数，或0
 */
public int reverse(int x) {
    if (x == Integer.MIN_VALUE || x == Integer.MAX_VALUE) {
        // 整数类型的最小值的绝对值 比 最大值的绝对值 大1
        return 0;
    }
    int sign = x > 0 ? 1 : -1; // 符号
    x = x < 0 ? -x : x; // 无论正负，都当成正数

    int result = 0; // 返回结果
    // 1.尝试拿个位数字：对10取模运算
    // 2.让每一位数字变成个位数字：先除以10，再对10取模得到十位数字
    int last = 0; // 末位
    while ((last = x % 10) != x) {
        // 3.将每一位数字计算累加：将上次累加结果*10 + 新数字
        result = result * 10 + last;
        x /= 10;
    }
    if (last != 0) { // 此时last是最高位，单独处理
        long re = result;
        re = re * 10 + last;
        if (re > Integer.MAX_VALUE || re < Integer.MIN_VALUE) {
            result = 0;
        } else {
            result = (int)re;
        }
    }
    return result * sign; // 返回前进行符号处理
}
```

## C++代码

```cpp
#include <iostream>
#include <limits.h>
using namespace std;

/**
 * 执行用时：0 ms, 在所有 C++ 提交中击败了 100.00% 的用户
 * 内存消耗：6.1 MB, 在所有 C++ 提交中击败了 10.95% 的用户
 */
class Solution {
public:
    int reverse(int x) {
        if (x == INT_MAX || x == INT_MIN) {
            // 整数类型的最小值的绝对值 比 最大值的绝对值 大1
            return 0;
        }
        int sign = x > 0 ? 1 : -1; // 符号
        x = x < 0 ? -x : x; // 无论正负，都当成正数

        int result = 0; // 返回结果
        // 1.尝试拿个位数字：对10取模运算
        // 2.让每一位数字变成个位数字：先除以10，再对10取模得到十位数字
        int last = 0; // 末位
        while ((last = x % 10) != x) {
            // 3.将每一位数字计算累加：将上次累加结果*10 + 新数字
            result = result * 10 + last;
            x /= 10;
        }
        if (last != 0) { // 此时last是最高位，单独处理
            long re = result;
            re = re * 10 + last;
            if (re > INT_MAX || re < INT_MIN) {
                result = 0;
            } else {
                result = (int)re;
            }
        }
        return result * sign; // 返回前进行符号处理
    }
};
```

## Python代码

```python

# 执行用时：32 ms, 在所有 Python3 提交中击败了 98.30% 的用户
# 内存消耗：13.5 MB, 在所有 Python3 提交中击败了 20.10% 的用户

def reverse(self, x: int) -> int:
        MAX_VALUE = (1 << 31) - 1  # 最大值
        MIN_VALUE = (1 << 31) * -1  # 最小值
        # 超出最值直接返回：0
```

```
        if x >= MAX_VALUE or x <= MIN_VALUE:
            return 0
        # -10到10之间，直接返回
        if -10 < x < 10:
            return x

        sign = 1 if x > 0 else -1  # 符号
        x = sign * x  # 无论正负，都当成正数

        result = 0
        while x != 0:  # Python3中整数类型都是int，没有long
            last = x % 10
            result = result * 10 + last
            x //= 10  # 取商，忽略小数部分

        if result >= MAX_VALUE or result <= MIN_VALUE:
            return 0
        return result * sign
```

## 测试用例

输入：123
输出：321

输入：-123
输出：-321

输入：120
输出：21

输入：-2147483648
输出：0

输入：2147483647
输出：0