# 题目

给出一个 32 位的有符号整数，你需要将这个整数中每位上的数字进行反转。

**示例 1:**

输入: 123

输出: 321

**示例 2:**

输入: -123

输出: -321

**示例 3:**

输入: 120

输出: 21

**注意:**

假设我们的环境只能存储得下32位的有符号整数，则其数值范围为[−231, 231−1]。请根据这个假设，如果反转后整数溢出那么就返回0。

# 分析

要解决整数反转并处理溢出的问题，我们需要逐步提取原整数的每一位数字，构建反转后的整数，并在过程中检查是否溢出。

解题思路：

1、提取数字并构建反转数：

- 通过循环取原整数的末位数字（x % 10），并将其作为反转数的新末位（rev = rev \* 10 + 末位数字）。

- 每次提取后，将原整数除以10（x /= 10），直到原整数变为0。

2、溢出检查：

- 32位有符号整数的范围是 [-2^31, 2^31 - 1]（即 [-2147483648, 2147483647]）。

- 在构建反转数时，每次更新前需检查是否即将溢出：

- 若 rev > INT\_MAX / 10，则 rev \* 10 必然溢出。

- 若 rev == INT\_MAX / 10 且末位数字 > 7，则 rev \* 10 + 末位数字会超过 INT\_MAX（2147483647 的末位是7）。

- 若 rev < INT\_MIN / 10，则 rev \* 10 必然溢出。

- 若 rev == INT\_MIN / 10 且末位数字 < -8，则 rev \* 10 + 末位数字会小于INT\_MIN（-2147483648 的末位是-8）。

代码：

class Solution {

public:

int reverse(int x) {

int rev = 0;

while (x != 0) {

// 提取x的末位数字

int pop = x % 10;

x /= 10;

// 检查正溢出

if (rev > INT\_MAX / 10 || (rev == INT\_MAX / 10 && pop > 7)) {

return 0;

}

// 检查负溢出

if (rev < INT\_MIN / 10 || (rev == INT\_MIN / 10 && pop < -8)) {

return 0;

}

// 更新反转数

rev = rev \* 10 + pop;

}

return rev;

}

};

解释：

- 提取末位数字：通过 x % 10 得到末位数字，再用 x /= 10 移除末位，循环直到x为0。

- 溢出检查：在每次更新rev前，通过与 INT\_MAX / 10 和 INT\_MIN / 10 比较，提前判断是否会溢出，避免直接计算导致的溢出问题。

- 时间复杂度：O(log|x|)，其中|x|是输入整数的绝对值，循环次数为数字的位数（最多10位）。

- 空间复杂度：O(1)，仅使用常数个变量。