一．8．字符串转整数

（一）题目

请你来实现一个 atoi 函数，使其能将字符串转换成整数。

首先，该函数会根据需要丢弃无用的开头空格字符，直到寻找到第一个非空格的字符为止。接下来的转化规则如下：

如果第一个非空字符为正或者负号时，则将该符号与之后面尽可能多的连续数字字符组合起来，形成一个有符号整数。

假如第一个非空字符是数字，则直接将其与之后连续的数字字符组合起来，形成一个整数。

该字符串在有效的整数部分之后也可能会存在多余的字符，那么这些字符可以被忽略，它们对函数不应该造成影响。

注意：假如该字符串中的第一个非空格字符不是一个有效整数字符、字符串为空或字符串仅包含空白字符时，则你的函数不需要进行转换，即无法进行有效转换。

在任何情况下，若函数不能进行有效的转换时，请返回 0 。

提示：

本题中的空白字符只包括空格字符 ' ' 。

假设我们的环境只能存储 32 位大小的有符号整数，那么其数值范围为 [−231,  231 − 1]。如果数值超过这个范围，请返回  INT\_MAX (231 − 1) 或 INT\_MIN (−231) 。

（二）示例

示例 1:

输入: "42"

输出: 42

示例 2:

输入: " -42"

输出: -42

解释: 第一个非空白字符为 '-', 它是一个负号。

  我们尽可能将负号与后面所有连续出现的数字组合起来，最后得到 -42 。

示例 3:

输入: "4193 with words"

输出: 4193

解释: 转换截止于数字 '3' ，因为它的下一个字符不为数字。

示例 4:

输入: "words and 987"

输出: 0

解释: 第一个非空字符是 'w', 但它不是数字或正、负号。

因此无法执行有效的转换。

示例 5:

输入: "-91283472332"

输出: -2147483648

解释: 数字 "-91283472332" 超过 32 位有符号整数范围。

  因此返回 INT\_MIN (−231) 。

二．题解

（一）思路

1. 输入的字符串为str，如果str.length()为0，则return 0。

2. 首先我们要寻找第一个非空的字符，设置初始索引index为0，若没有找到，索引最后为str.length() ，说明该字符串为空字符串，返回0；若找到了非空的字符，则记录索引。

3. 设置flag符号位初始化为1，判断第一个非空字符是否是符号。若为+号，则索引index++；若为-号，则index++,同时flag=-1。

4. 设置一个最终结果ans=0，判断第一个非空字符是否是数字。

（1）若是数字，再判断当前的ans\*10再加上当前的数字是否会超出32位整数的最大或最小范围，这里要注意负数的情况，仔细看看代码中负数的编写。若没有超出，则将该数字添加到ans\*10中，若超出了范围，则返回最大值或者最小值。index索引照常++。

（2）若不是数字，则说明第一个非空字符是无效字符，此时ans=0，直接break退出循环。

5. 遍历字符串完毕，返回ans。

（二）代码实现

C++

#include<string>

#include<iostream>

using namespace std;

class Solution {

public:

int myAtoi(string str) {

unsigned long len = str.length();

//去除前面的空格

int index = 0;

while (index < len) {

if (str[index] != ' ')

break;

index++;

}

//字符串为空字符串

if (index == len) {

return 0;

}

//找到非空字符

//判断是否为符号

int flag = 1;

if (str[index] == '+')

index++;

else if(str[index] == '-'){

index++;

flag = -1;

}

//判断是否为数字

int ans = 0;

while (index < len) {

char curChar = str[index];

//无效字符

if (curChar < '0' || curChar > '9')

break;

//数字，判断溢出情况

if (ans > INT\_MAX / 10 || (ans == INT\_MAX / 10 && (curChar - '0') > INT\_MAX % 10))

return INT\_MAX;

if (ans < INT\_MIN / 10 || (ans == INT\_MIN / 10 && (curChar - '0') > -(INT\_MIN % 10)))

return INT\_MIN;

ans = ans \* 10 + flag \* (curChar - '0');

index++;

}

return ans;

}

};

Java

public class Solution {

    public int myAtoi(String str){

        int len = str.length();

        char[] charArray = str.toCharArray();

        int index = 0;

        //取出前面的空格

        while(index < len && charArray[index] == ' ')

            index++;

        //空白字符串

        if(index == len)

            return 0;

        //判断是否是符号

        int flag = 1;

        char firstCHar = charArray[index];

        if(firstCHar == '+')

            index++;

        else if(firstCHar == '-'){

            index++;

            flag = -1;

        }

        //判断是否是数字

        int ans = 0;

        while(index < len){

            char currChar = charArray[index];

            //无效字符，退出循环，返回0

            if(currChar < '0' || currChar > '9')

                break;

            //数字，判断溢出

            //最大值溢出

            if(ans > Integer.MAX\_VALUE / 10 || (ans == Integer.MAX\_VALUE / 10 && (currChar - '0') > Integer.MAX\_VALUE % 10)){

                return Integer.MAX\_VALUE;

            }

            if(ans < Integer.MIN\_VALUE/10 || (ans == Integer.MIN\_VALUE/10 &&  (currChar - '0') > -(Integer.MIN\_VALUE % 10))){

                return Integer.MIN\_VALUE;

            }

            //正常相加

            ans = ans \* 10 + flag \* (currChar - '0');

            index++;

        }

        return ans;

    }

}