String to Integer (atoi)

Implement `atoi` to convert a string to an integer.

Hint: Carefully consider all possible input cases. If you want a challenge, please do not see below and ask yourself what are the possible input cases.

Notes: It is intended for this problem to be specified vaguely (ie, no given input specs). You are responsible to gather all the input requirements up front.

Update (2015-02-10):
The signature of the `C++` function had been updated. If you still see your function signature accepts a `const char *` argument, please click the reload button to reset your code definition.

spoilers alert... click to show requirements for atoi.

Requirements for atoi:
The function first discards as many whitespace characters as necessary until the first non-whitespace character is found. Then, starting from this character, takes an optional initial plus or minus sign followed by as many numerical digits as possible, and interprets them as a numerical value.

The string can contain additional characters after those that form the integral number, which are ignored and have no effect on the behavior of this function.

If the first sequence of non-whitespace characters in str is not a valid integral number, or if no such sequence exists because either str is empty or it contains only whitespace characters, no conversion is performed.

If no valid conversion could be performed, a zero value is returned. If the correct value is out of the range of representable values, INT_MAX (2147483647) or INT_MIN (-2147483648) is returned.

Solution 1

I think we only need to handle four cases:

1. discards all leading whitespaces
2. sign of the number
3. overflow
4. invalid input

Is there any better solution? Thanks for pointing out!

```
int atoi(const char *str) {
    int sign = 1, base = 0, i = 0;
    while (str[i] == ' ') { i++; }
    if (str[i] == '-' || str[i] == '+') {
        sign = 1 - 2 * (str[i++] == '-');
    }
    while (str[i] >= '0' && str[i] <= '9') {
        if (base >  INT_MAX / 10 || (base == INT_MAX / 10 && str[i] - '0' > 7)) {
            if (sign == 1) return INT_MAX;
            else return INT_MIN;
        }
        base  = 10 * base + (str[i++] - '0');
    }
    return base * sign;
}
```

written by yuruofeifei original link here

Solution 2

```
public int myAtoi(String str) {
    int index = 0, sign = 1, total = 0;
    //1. Empty string
    if(str.length() == 0) return 0;

    //2. Remove Spaces
    while(str.charAt(index) == ' ' && index < str.length())
        index ++;

    //3. Handle signs
    if(str.charAt(index) == '+' || str.charAt(index) == '-'){
        sign = str.charAt(index) == '+' ? 1 : -1;
        index ++;
    }

    //4. Convert number and avoid overflow
    while(index < str.length()){
        int digit = str.charAt(index) - '0';
        if(digit < 0 || digit > 9) break;

        //check if total will be overflow after 10 times and add digit
        if(Integer.MAX_VALUE/10 < total || Integer.MAX_VALUE/10 == total && Integ
er.MAX_VALUE %10 < digit)
            return sign == 1 ? Integer.MAX_VALUE : Integer.MIN_VALUE;

        total = 10 * total + digit;
        index ++;
    }
    return total * sign;
}
```

written by lestrois original link here

Solution 3

```
int myAtoi(string str) {
    int ret = 0, sign = 1, i = str.find_first_not_of(' '), base = INT_MAX / 10;
    if (str[i] == '+' || str[i] == '-') sign = str[i++] == '+' ?: -1;
    while (isdigit(str[i])) {
        if (ret > base || (ret == base && str[i] - '0' > 7))
            return sign > 0 ? INT_MAX : INT_MIN;
        ret = 10 * ret + (str[i++] - '0');
    }
    return sign * ret;
}
```

written by houzi original link here