把字符串转换成整数

写一个函数 StrToInt，实现把字符串转换成整数这个功能。不能使用 atoi 或者其他类似的库函数。

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。

当我们寻找到的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组合起来，作为该整数的正负号；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，形成整数。

该字符串除了有效的整数部分之后也可能会存在多余的字符，这些字符可以被忽略，它们对于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。

在任何情况下，若函数不能进行有效的转换时，请返回 0。

说明：

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 $[-2^{31}, 2^{31} - 1]$。如果数值超过这个范围，请返回 INT_MAX $(2^{31} - 1)$ 或 INT_MIN (-2^{31}) 。

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 (-2^{31}) 。

```
class Solution {
public:
    bool isdigit(char str)
    {
        if(str>'0'&&str<'9')
            return true;
        else
            return false;
    }
    int strToInt(string str) {
        int flag=1;
        int i=0;
        //const char* p=str;
        if(str.size()==0)
        {
            return 0;
        }
        if(str[0]=='\0')
        {
            return 0;
        }
        while(str[i]==' ')
        {
            i++;
        }
        if(str[i]=='+')
        {
            flag=1;
```

```
            i++;
        }
        else if(str[i]=='-')
        {
            flag=-1;
            i++;
        }
        long long res=0;
        //对非空和非+-号的有效数字进行处理
        for(; i<str.size(); i++)
        {
            if(str[i]>='0' && str[i]<='9')  //判断是否为有效数字
            {
                //res = res*10 + (str[i] - '0');  //***字符相减转换成 int
                //if(res > INT_MAX && flag) return INT_MIN;
                //if(res > INT_MAX && flag) return INT_MAX;
                res = res*10 + flag*(str[i] - '0');  //***字符相减转换成 int
                if(res > INT_MAX) res=INT_MAX;
                if(res < INT_MIN) res=INT_MIN;
            }
            else        //碰到非有效数字就退出
                break;
        }

        return res;
    }
};

/*
class Solution {
public:
    int strToInt(string str) {
        if(str.size()==0) return 0;
        int i = 0;
        long ans = 0;
        bool negative = false;

        //跳过前面空格，判断正负
        while(str[i]==' ') i++;
        if(str[i]=='-')
        {
            negative = true;
            i++;
        }
```

```
        else if(str[i]=='+')
                i++;

        //对非空和非+-号的有效数字进行处理
        for(; i<str.size(); i++)
        {
            if(str[i]>='0' && str[i]<='9')  //判断是否为有效数字
            {
                ans = ans*10 + (str[i] - '0');  //***字符相减转换成 int
                if(ans > INT_MAX && negative) return INT_MIN;
                if(ans > INT_MAX && !negative) return INT_MAX;
            }
            else        //碰到非有效数字就退出
                break;
        }

        return negative? -ans:ans;
    }
};

*/
```