```
/*
 给定字符串J 代表石头中宝石的类型，和字符串 S代表你拥有的石头。 S 中每个字符代表了一种你拥有的石头的类型，你想知道你拥有的石头中有多少是宝石。

J 中的字母不重复，J 和 S中的所有字符都是字母。字母区分大小写，因此"a"和"A"是不同类型的石头。

示例 1:

输入: J = "aA", S = "aAAbbbb"
输出: 3

示例 2:

输入: J = "z", S = "ZZ"
输出: 0

注意:

    S 和 J 最多含有50个字母。
     J 中的字符不重复。

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/jewels-and-stones
著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。
*/
```

分析:

- 方法一:依次从J选取元素，在J中进行查找统计累加.时间复杂度$O(mn)$
- 方法二:由于所有字母是可枚举的,可以先统计一遍S中所有字母的长度,然后遍历J的元素进行累加起来即可，时间复杂度$O(m+n) \rightarrow O(\max m,n)$,空间复杂度:128*2,$O(1)$

方法一:C_暴力法

```
int countChar(char*S , char ch)
{
    int ret_val =   0;
    int i       =   0;
    while(S[i])
    {
        if(S[i]==ch)
        {
            ret_val++;
        }
        i++;
    }

    return ret_val;
}

int numJewelsInStones(  char*   J   ,
                        char*   S
```

```
                    )
{
    int ret_val = 0;
    int i = 0;
    while (J[i])
    {
        ret_val += countChar(S,J[i]);
        i++;
    }
    return ret_val;
}
/*
执行结果：
通过
显示详情
执行用时 :8 ms, 在所有 C 提交中击败了21.55% 的用户
内存消耗 :6.6 MB, 在所有 C 提交中击败了91.34%的用户
*/
```

方法二:C_枚举计数法

```
int numJewelsInStones(char* J, char * S)
{
    short int count_ch[128] = {0,};
    int i                   = 0;
    int ret_val             = 0;
    //memset(count_ch,0,sizeof(count_ch));
    while (S[i])
    {
        count_ch[S[i]] ++;
        i++;
    }
    i = 0;
    while (J[i])
    {
        ret_val += count_ch[J[i]];
        i++;
    }
    return ret_val;
}

/*
执行结果：
通过
显示详情
执行用时 :4 ms, 在所有 C 提交中击败了71.55% 的用户
内存消耗 :6.8 MB, 在所有 C 提交中击败了80.74%的用户
*/
```

AlimyBreak
2019.08.31