```
/*
给你一份『词汇表』（字符串数组） words 和一张『字母表』（字符串） chars。

假如你可以用 chars 中的『字母』（字符）拼写出 words 中的某个『单词』（字符串），那么我们就认为你掌握了这个单词。

注意：每次拼写时，chars 中的每个字母都只能用一次。

返回词汇表 words 中你掌握的所有单词的 长度之和。



示例 1：

输入：words = ["cat","bt","hat","tree"], chars = "atach"
输出：6
解释：
可以形成字符串 "cat" 和 "hat"，所以答案是 3 + 3 = 6。

示例 2：

输入：words = ["hello","world","leetcode"], chars = "welldonehoneyr"
输出：10
解释：
可以形成字符串 "hello" 和 "world"，所以答案是 5 + 5 = 10。



提示：

    1 <= words.length <= 1000
    1 <= words[i].length, chars.length <= 100
    所有字符串中都仅包含小写英文字母

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/find-words-that-can-be-formed-by-characters
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:

- 利用map数据结构统计chars中的资源数量
- 利用map统计各个string需要的资源数量
- 对比资源数量是否满足

方法一:C++_map统计法

```
class Solution
{
    public:
        int countCharacters(vector<string>& words, string chars)
```

```
        {
            map<char,int>   mci             ;
            char            ch      =   0   ;
            int             i       =   0   ;
            int             j       =   0   ;
            int             ret_val =   0   ;
            int             flag    =   0   ;

            for(ch = 'a'; ch<='z';ch++)
            {
                mci[ch] = 0;
            }
            for(i=0;i<chars.size();i++)
            {
                mci[chars[i]]++;
            }


            for(i=0;i<words.size();i++)
            {
                map<char,int> mci_temp;
                for(j=0;j<words[i].size();j++)
                {
                    if(mci_temp.count(words[i][j]))
                    {
                        mci_temp[words[i][j]]++;
                    }
                    else
                    {
                        mci_temp[words[i][j]] = 1;
                    }
                }

                map<char,int>::iterator it = mci_temp.begin();
                flag = 1;
                while(it!=mci_temp.end())
                {
                    if(it->second > mci[it->first])
                    {
                        flag = 0;
                        break;
                    }
                    it++;
                }

                if(flag==1)
                {
                    ret_val += words[i].size();
                }
            }
            return ret_val;
        }
};

/*
执行结果：
通过
显示详情
```

```
63 执行用时 :344 ms, 在所有 cpp 提交中击败了10.01% 的用户
64 内存消耗 :53.7 MB, 在所有 cpp 提交中击败了100.00%的用户
65 */
```

AlimyBreak
2019.10.26