字符串类

在上一讲，我们认识了字符和字符数组，但是在 C++中，提供了一种更强有力的字符处理工具——string 类。

string 类是 C++标准库的一个重要的部分，主要用于字符串处理，同时 C++的算法库对 string 也有着很好的支持，而且 string 还和 c 语言的字符串之间有着良好的接口。可能这段话你没看懂，不过没关系，等学到后面再回过头来看就明白了。

string 类的定义和初始化

想使用 string 首先要在头文件当中加入< string > ，注意，不是<cstring>，如下：
#include<string>

string 的声明

string s; //声明一个 string 对象
string ss[10]; //声明一个 string 对象的数组

string 的初始化

初始化是指在声明 string 变量时同时给它赋值。

```
#include <iostream>
#include <string>
using namespace std;
int main ( ){
    string str;  //默认初始化是一个空字符串

    string s0 = "Hello world";   // 给 s0 赋值为"Hello world"
    cout<<s0<<endl;

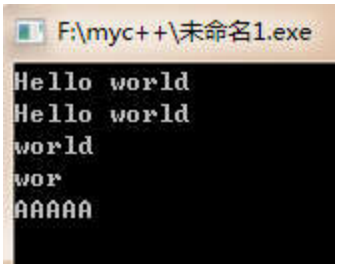
    string s1(s0);        //调用复制构造函数生成 s1，s1 为 s0 的复制品
    cout<<s1<<endl;

    string s2(s0,6);      //将 s0 内，开始于位置 6 的部分当作 s2 的初值
    cout<<s2<<endl;

    string s3(s0,6,3);  //将 s0 内，开始于 6 且长度顶多为 3 的部分作为 s3 的初值
    cout<<s3<<endl;

    string s6(5,'A');  //生成一个字符串，包含 5 个'A'字符
    cout<<s6<<endl;
}
```

输出结果如下：



string 类的输入和输出

string 类型的读写就像其它类型的读写一样，使用 cin，cout。不过值得注意的是，使用 cin 读入 string 类型，忽略开头的(制表符、换行符、空格)，当再次碰到空字符就停止（并不会读取空字符）。

例 1：给出一段文本，由空格和字母构成，空格之间的字母组成一个单词。请给单词编号输出。
输入样例：
Hello world
Today
输出样例：
1 Hello
2 world
3 Today
【分析】
本题可以使用 cin 读入，cin 可以自动忽略空格直接读取单词，将读取单词存到字符串变量中。但是 cin 一次只能读一个单词，所以我们可以使用 while 语句循环读入单词，直到文件结束，可以使用 while(cin>>s)的用法。记得在控制窗口中使用 ctrl+z 模拟文件结束符。
【代码】

```
#include<string>
#include<iostream>
using namespace std;
int main() {
    string s;
    int tot=0;
    while (cin>>s) {  //自动忽略空格读取单词
        cout<<++tot<<' '<<s<<endl;   //输出字符串变量 s
    }
}
```

使用 cin 不能读入空格，如果我们要整行读入，包含行中的空格，应该怎么办呢？

getline 可以整行读入字符串，包括空格，但是会舍弃回车。格式为

getline(cin,s);

cin 指的是读入流，一般情况下我们直接写 cin 即可，s 是字符串，即我们读入的东西要存放的字符串。注意与 cin.getline()的区别。getline()函数属 string 头文件，cin.getline()属于 iostream 头文件，getline()函数用于读入到字符串类，cin.getline()用于读入到字符数组。

例 2：给出一段文本，请加上行号后输出。
输入样例：
Hello world
Today
输出样例：
1 Hello world
2 Today
【分析】
本题需要读入整一行，一行之中可能包含空格，所以不用用 cin 直接读入，应该使用 getline()函数进行读入。
【代码】

```
#include<string>
#include<iostream>
using namespace std;
int main() {
    string s;
    int tot=0;
    while (getline(cin,s)) {
        cout<<++tot<<' '<<s<<endl;
    }
    return 0;
}
```

string 类的操作符

string 类定义了非常丰富的操作符，方便我们的操作。我们可以直接用 ==、>、<、>=、<=、和!=比较字符串，也可以用+或者+=操作符连接两个字符串，用=给字符串变量赋值，并且可以用[]获取特定的字符。

```
#include <iostream>
#include <string>
using namespace std;
int main ( ){
    string s1="oi";
    string s2="class";
    if(s1>s2){ //直接比较两个字符串的大小
        s1+=s2;  //连接字符串，等价于 s1=s1+s2;
```

```
9.          cout<<s1; //输出“oiclass”
10.     }
11. }
```

注意：字符串“+”操作符的左右操作数必须有一个为字符串类型。

```
string s1= ”hello”;
string s2=”world”;
string s3=s1+”,”;                  //合法
string s4=”hello ”+” world”;   //非法，’+’号两边没有字符串类型
string s5=s1+”,”+”world”;      //合法
string s6=”hello”+”,” + s2;       //非法，第一个’+’号非法
```

我们可以像字符数组一样访问 string 类型的各个字符，利用[]操作符实现。特别注意，string 类的字符存储也是从下标 0 开始的。

```
string str=“2+2=4”;
cout<<str[1]<<endl;      //输出’+’
str[1]=’*’;      //修改字符，现在 str[1]变成了’*’
```

例 3：输入 n 个单词，将每个单词转化成大写字母后按 ASCII 值从小到大排序输出。

输入样例：

```
4
Cat dog hello apple
```

输出样例：

```
APPLE
CAT
DOG
HELLO
```

【分析】

本题是上一讲的例题，在这里使用 string 类解决，跟字符数组比较优劣。

【代码】

```
1.  #include<iostream>
2.  #include<string>
3.  using namespace std;
4.  string s[100];
5.  int n;
6.  int main ( ){
7.      cin>>n;
8.      for(int i=1;i<=n;i++){
9.          cin>>s[i];
10.         //以下循环将字符串转成大写
11.         for(int j=0;j<s[i].size();j++){ //枚举每个字符
12.             if(s[i][j]>='a'&&s[i][j]<='z'){ //如果该字符是小写字母
13.                 s[i][j]=s[i][j]-'a'+'A';  //转成大写字母
```

oiClass

```
            }
        }
    }
    for(int i=1;i<n;i++){
        for(int j=i+1;j<=n;j++){
            if(s[i]>s[j]){ //直接比较两个字符串大小
                s[i].swap(s[j]); //交换两个字符串，swap()函数见后面介绍
            }
        }
    }
    for(int i=1;i<=n;i++){
        cout<<s[i]<<endl;
    }
}
```

string 类的常用函数

string 类的函数有一大堆，可以看文末附表，在此介绍几个常用的函数。

1、 size()

获得字符串变量的长度。

```
string s=”oiclass”;
cout<<s.size();  //输出 7
```

2、 find()

find()函数主要是查找一个字符串是否在调用的字符串中出现过，大小写敏感。如果找到，该函数返回找到的位置，如果找不到，返回一个叫 npos 的常量。

```
string s=”oiclass”;
cout<<s.find(“oi”);  //查找字符串，输出 0；第 0 个位置找到
cout<<s.find(‘a’);   //查找字符，输出 4
if(s.find(“io”)==s.npos){  //查找不到的判断
    cout<<”can’t find”<<endl;
}
```

3、 erase()

erase()函数执行删除操作，返回删除后的字符串。erase()函数需要两个参数 pos 和 len，其中 pos 为起始位置，pos 以及后面 len-1 个字符串都删除。

```
string s=”oiclass”;
cout<<s.erase(2,5);  //从第 2 个位置开始删除，一共删除 5 个字符，输出”oi”。
```

4、 insert()

insert()函数执行插入操作，返回插入之后的字符串。insert()函数有多种参数形式，如 s.insert(pos,str)表示在 s 的 pos 位置插入字符串 str，s.insert(pos,str,a,n)表示在 s 的 pos 位置

插入 str 中从位置 a 到后面的 n 个字符，s.insert(pos,n,ch)表示在 s 的 pos 位置上面插入 n 个字符 ch。

string s=”oiclass”;

string str=”hello “;

cout<<s.insert(0,str); //输出”hello oiclass”

5、 replace()

replace()函数实现替换字符的操作，先删除后插入。replace()也有多重参数实现形式，s.replace(pos,n,str)表示将 s 的 pos 位置开始的 n 个字符替换为字符串 str。

string s=”oiclass”;

string str=”hello “;

cout<<s.replace(0,2,str); //输出”hello class”

6、 substr()

substr()函数用来提取字符串中的子串。s.substr(pos,n)返回字符串位置为 pos 后面的 n 个字符组成的串，s.substr(pos)返回一个 pos 到结尾的串

string s=”oiclass”;

cout<<s.substr(0,2); //输出 oi

string 类与数值转换

我们有时需要将数值类型转换成字符串类型，有时需要将字符串类型转换成数值类型。C++中并没有直接的函数可以转换（这句话不严谨，其实是有的，只是受编译器限制不能通用，但是为了避免讨论这些枝节问题，在此先略过了），不过我们可以借助字符串流轻松实现。

字符串流与我们的输入输出流（iostream）很相似，提供一个缓冲区进行读取写入。使用字符串流，我们首先需要引用头文件<sstream>，我们还需要定义一个 stringstream 类对象:

stringstream ss;

在这里，ss 就跟 cin 和 cout 的性质差不多。我们往 ss 对象写入数据可以用<<符号，从 ss 对象中读取数据可以用>>符号。

将数值转换为字符串

```
#include<iostream>
#include<string>
#include<sstream>  //引入头文件
using namespace std;
stringstream ss;  //定义流对象
string s;
int n,x;
int main ( ){
    cin>>n;
    for(int i=1;i<=n;i++){
        cin>>x;
```

oiClass

```
        ss.clear(); //重复读写需要清空缓冲区
        ss<<x;    //将整型数据 x 写入 ss 缓冲区
        ss>>s;    //从 ss 中读取数据，存储为字符串
        cout<<s<<endl;
    }
}
```

将字符串转换成数值

```
#include<iostream>
#include<string>
#include<sstream>  //引入头文件
using namespace std;
stringstream ss;  //定义流对象
string s;
int n,x;
int main ( ){
    cin>>n;
    for(int i=1;i<=n;i++){
        cin>>s;
        ss.clear(); //重复读写需要清空缓冲区
        ss<<s;    //将字符串 s 写入 ss 缓冲区
        ss>>x;    //从 ss 中读取数据，存储为整型
        cout<<x<<endl;
    }
}
```

string 类与字符数组的转换

字符数组转成 string

1、在 string 变量初始化时实现

```
1、 #include<iostream>
2、 #include<string>
3、 using namespace std;
4、 char ch[100]="oiclass";
5、 int main ( ){
6、     string s(ch);  //将 ch 字符数组赋值给字符串变量 s
7、     cout<<s<<endl;  //输出"oiclass"
8、     string s1(ch,2); //将 ch 的前 2 个字符赋值给字符串变量 s1
9、     cout<<s1<<endl;  //输出"oi"
10、 }
```

2、利用 assign()函数进行赋值

```
#include<iostream>
#include<string>
using namespace std;
char ch[100]="oiclass";
string s;
int main ( ){
    s.assign(ch);//将 ch 数组赋值给 s
    cout<<s<<endl;  //输出"oiclass"
    s.assign(ch,2); //将 ch 数组前 2 个字符赋值给 s
    cout<<s<<endl;  //输出"oi"
}
```

字符串转成字符数组

1、利用 copy()函数实现

copy(str,num,index)函数拷贝自己的 num 个字符到 str 中（从索引 index 开始）。返回值是拷贝的字符数

```
#include<iostream>
#include<string>
using namespace std;
char ch[100];
string s="oiclass";
int main ( ){
     s.copy(ch,s.npos,0); //s.npos 代表最多个数，即全串
     cout<<ch; //输出"oiclass"
}
```

2、利用 strcpy()函数实现

```
#include<iostream>
#include<string>
#include<cstring>
using namespace std;
char ch[100];
string s="oiclass";
int main ( ){
    strcpy(ch,s.c_str());//s.c_str()获得字符串 s 的 c 式地址
    cout<<ch; //输出"oiclass"
}
```

string 函数列表

函数名	描述
begin	得到指向字符串开头的 Iterator
end	得到指向字符串结尾的 Iterator
rbegin	得到指向反向字符串开头的 Iterator
rend	得到指向反向字符串结尾的 Iterator
size	得到字符串的大小
length	和 size 函数功能相同
max_size	字符串可能的最大大小
capacity	在不重新分配内存的情况下，字符串可能的大小
empty	判断是否为空
operator[]	取第几个元素，相当于数组
c_str	取得 C 风格的 const char* 字符串
data	取得字符串内容地址
operator=	赋值操作符
reserve	预留空间
swap	交换函数
insert	插入字符
append	追加字符
push_back	追加字符
operator+=	+= 操作符
erase	删除字符串
clear	清空字符容器中所有内容
resize	重新分配空间
assign	和赋值操作符一样
replace	替代
copy	复制字符串到字符数组
find	查找
rfind	反向查找
find_first_of	查找包含子串中的任何字符，返回第一个位置
find_first_not_of	查找不包含子串中的任何字符，返回第一个位置
find_last_of	查找包含子串中的任何字符，返回最后一个位置
find_last_not_of	查找不包含子串中的任何字符，返回最后一个位置
substr	得到字串
compare	比较字符串
operator+	字符串链接
operator==	判断是否相等
operator!=	判断是否不等于
operator<</td>	判断是否小于
operator>>	从输入流中读入字符串
operator<<	字符串写入输出流
getline	从输入流中读入一行

==========网站练习==========

最长单词

【题目描述】
编写程序，根据给出的一个结束于‘.’的英文句子，找出其中最长的含有字母‘a’的单词。
【输入】
输入为 1 行，是一个英文句子，结束于句点‘.’。句子中的单词由一个或几个空格隔开。
【输出】
输出文件只有 1 行，只有一个数据，显示找出的最长的含有字母‘a’的单词。如果有多个这样的单词，只显示其中的第一个；若没有含字母‘a’的单词，则显示“NO”。
【样例输入】
Her name is Lilan and she is a happy student.
【样例输出】
Lilan
【分析】
本题可以用 cin 读入每个单词，利用字符串函数 find()查找单词中是否出现’a’。注意最后一个单词需要删除’.’号。
【代码】

```
#include<iostream>
#include<string>
using namespace std;
string s,str;
int len;
int main(){
    while(cin>>s){
        if(s.find('a')!=s.npos&&s.size()>len){
            str=s;
            len=s.size();
        }
    }
    if(str.find('.')!=str.npos){  //如果单词出现’.’，则删除
        str.erase(str.size()-1,1);
    }
    cout<<str<<endl;
}
```

牛棚回声

【题目描述】
奶牛们非常享受在牛栏中牟叫，因为她们可以听到她们牟声的回音。虽然有时候并不能完全

听到完整的回音。Bessie 曾经是一个出色的秘书，所以她精确地纪录了所有的牟叫声及其回声。她很好奇到底两个声音的重复部份有多长。

输入两个字符串（长度为 1 到 80 个字母），表示两个牟叫声。你要确定最长的重复部分的长度。两个字符串的重复部分指的是同时是一个字符串的前缀和另一个字符串的后缀的字符串。我们通过一个例子来理解题目。考虑下面的两个牟声：

moyooyoxyzooo

yzoooqyasdfljkamo

第一个串的最后的部份"yzooo"跟第二个串的第一部份重复。第二个串的最后部分"mo"跟第一个串的第一部分重复。所以"yzooo"跟"mo"都是这 2 个串的重复部分。其中，"yzooo"比较长，所以最长的重复部分的长度就是 5。

【输入】

前两行: 每一行是 1 个字符串表示奶牛的牟声或它的回声。

【输出】

第一行: 包含一个单独的整数表示输入的 2 个字符串中，一个字符串的前缀和另一个字符串的后缀的最长的重复部分的长度。

【样例输入】

abcxxxxabcxabcd

abcdxabcxxxxabcx

【样例输出】

11

【提示】

"abcxxxxabcx"是第一个字符串的前缀和第二个字符串的后缀。

【分析】

本题可以枚举字符串 1 的前缀和字符串 2 的后缀，判断是否相同。提取字符串的子串可以使用 substr()函数。

【代码】

```
#include<iostream>
#include<string>
using namespace std;
string s1,s2;
int len1=1,len2=1;
int main(){
    cin>>s1>>s2;
    for(int i=1;i<=min(s1.size(),s2.size());i++){
        //枚举 s1 的前缀和 s2 的后缀，比较是否相同
        if(s1.substr(0,i)==s2.substr(s2.size()-i,i)){
            len1=i;
        }
        //枚举 s2 的前缀和 s1 的后缀，比较是否相同
        if(s2.substr(0,i)==s1.substr(s1.size()-i,i)){
            len2=i;
        }
    }
    if(len1>len2)cout<<len1;
```

```
19.     else cout<<len2;
20. }
```

戴夫的信

【题目描述】
僵尸就要来进攻你的房子了，戴夫给你写了一封密信，但为了怕僵尸发现，他决定给信加密，并提前告诉你加密的的方法，加密的方法是在信里加入一些没有意义的字符或单词，然后加入两个修正符号“#”和“@”。修正的规则如下：
1，当你发现一个退格符"#"，则表示前一个字符无效；
2，如果发现一个退行符"@"，以表示该行中 “@”前的所有字符均无效。 如果已经在行首'#'符号无效。
现在请你翻译处加密前的信。
【输入】
第一行一个整数 n，代表信一共有多少行。
接下来 n 行字符串。
【输出】
输出加密前的信。
【样例输入】
3
whli##ilr#e (s#*s)
I am daifu@
    outcha@putchar(*s=#++);
【样例输出】
While (*s)

putchar(*s++);
【分析】
本题只需在给定字符串中找到’@’和’#’符号，按照对应规则进行删除，细节处理方面，注意’@’和’#’符号可能多次出现。另，文本有空格，需用 getline 语句。但是在输入整型后使用 getline，会读到第一行的换行符，所以多读了一次 getline()，用来过滤第一行的换行。
【代码】

```
1.  #include<iostream>
2.  #include<string>
3.  using namespace std;
4.  string s;
5.  int n;
6.  int main(){
7.      cin>>n;
8.      getline(cin,s); //过滤换行符
9.      for(int i=1;i<=n;i++){
10.         getline(cin,s);
11.         int len=s.rfind('@');// 反向查找最后一个'@'
```

oiClass

```
        s.erase(0,len+1);  //删除'@'之前的字符
        len=s.find('#');  //不断查找'#'
        while(len!=s.npos){
            if(len==0){   //特判开头位置的'#'
                s.erase(0,1);
            }else{
                s.erase(len-1,2);
            }
            len=s.find('#');
        }
        cout<<s<<endl;
    }
}
```

统计单词数

【题目描述】
一般的文本编辑器都有查找单词的功能，该功能可以快速定位特定单词在文章中的位置，有的还能统计出特定单词在文章中出现的次数。
现在，请你编程实现这一功能，具体要求是：给定一个单词，请你输出它在给定的文章中出现的次数和第一次出现的位置。注意：匹配单词时，不区分大小写，但要求完全匹配，即给定单词必须与文章中的某一独立单词在不区分大小写的情况下完全相同（参见样例 1），如果给定单词仅是文章中某一单词的一部分则不算匹配（参见样例 2）。
【输入】
第 1 行为一个字符串，其中只含字母，表示给定单词；
第 2 行为一个字符串，其中只可能包含字母和空格，表示给定的文章。
【输出】
只有一行，如果在文章中找到给定单词则输出两个整数，两个整数之间用一个空格隔开，分别是单词在文章中出现的次数和第一次出现的位置（即在文章中第一次出现时，单词首字母在文章中的位置，位置从 0 开始）；如果单词在文章中没有出现，则直接输出一个整数-1。
【输入样例 1】
To
to be or not to be is a question
【输入样例 2】
to
Did the Ottoman Empire lose its power at that time
【输出样例 1】
2 0
【输出样例 2】
-1
【输入输出样例 1 说明】
输出结果表示给定的单词 To 在文章中出现两次，第一次出现的位置为 0。
【输入输出样例 2 说明】
表示给定的单词 to 在文章中没有出现，输出整数-1。

oiClass

【数据范围】
1≤单词长度≤10。
1≤文章长度≤1,000,000。
【分析】
获取每个单词，判断这个单词跟给定的单词是否相同。问题难点在于，如果单词相同，需要返回单词在字符串中的位置，所有不能用 cin 读入，可以用逐字符读入的方式来记录位置。
【代码】

```
#include<iostream>
#include<string>
using namespace std;
string s,t;
int cnt,first=-1;
int main(){
    cin>>t;  //读入单词
    getline(cin,s);  //换行
    for(int i=0;i<t.size();i++){  //转换成大写
        if(t[i]>='a'&&t[i]<='z')t[i]-=32;
    }
    getline(cin,s);  //读入一整行
    for(int i=0;i<s.size();i++){  //转换成大写
        if(s[i]>='a'&&s[i]<='z')s[i]-=32;
    }
    s.insert(0," ");  //行首和行尾行尾加空格，便于查找单词判断
    s.insert(s.size()," ");
    int p=s.find(t);  //查找模式串
    while(p!=s.npos){
        if(s[p-1]==' '&&s[p+t.size()]==' '){ //如果字符的前后为空格，即为单词
            cnt++;
            if(first==-1)first=p-1;
        }
        p=s.find(t,p+t.size());  //继续往后查找
    }
    if(first==-1)cout<<first;
    else  cout<<cnt<<" "<<first;
}
```

牛的 RP

【题目描述】
天苍苍，野茫茫，JSZX 的菜鸟们来到 OI 牧场旅游，看到了好多好多的牛。OI 牧场所有的牛都觉得自己的 Rp 最高（简称 RP 牛），为此他们常争论不休。于是，他们让 JSZX 的菜菜们用最最朴素的方法找出这只 RP 牛。
经过讨论，最菜的 mmk 想出了最朴素的方法：

我们要以 cows 的名字为线索，来找出 RP 牛。
首先，得到 n 头牛的名字清单（每头牛的名字是一个仅包含小写字母的字符串，且这些牛的读写方式比较特殊—从右到左），然后对每头牛进行检验，检验按照牛的读写方式进行。规则如下：
1．Rp 牛的名字中必须有子串“jszxoier”
2．将名字中的每个“cow”的替换为“bird”。
3．计算 Rp 值：A 为名字中子串“r”的个数；
B 为名字中子串“p”的个数；
C 为名字中字串“rp”的个数；
Rp 值即为 5×A+5×B+20×C。
最后输出 RP 牛的名字，若有多个 RP 牛，则输出名字最短的那个。
假如你也是牛中一员，尽管你很不屑这样的水题，但是，你很想到 RP 牛那里分点 Rp，所以你决定解决这道题，并算出 RP 牛的 Rp 是多少。
【输入】
第一行，一个数 n（n<=3000）。
接下来的 n 行，每行一个字符串，长度<=300，数据保证存在 RP 牛。
【输出】
共两行
第一行为 RP 牛的名字
第二行为 RP 牛的 Rp 值
【样例输入】
8
reioxzsjzmy
mmk
jwc
zxf
jwc
wangwei
xcy
yuhc
【样例输出】
reioxzsjzmy
5
【分析】
本题按照题目意思模拟处理即可。细节注意的地方较多，要注意反序处理，要注意输出的字符串是原串，所以不能在原字符串上修改。
【代码】

```
#include<iostream>
#include<string>
using namespace std;
string s, str,tmp;
int maxrp,n;
int main(){
    cin>>n;
```

oiClass

```
    for(int i=1;i<=n;i++){
        cin>>s;
        tmp=s;
        //规则 1
        if(tmp.find("reioxzsj")==tmp.npos)continue;
        //规则 2
        while(tmp.find("woc")!=tmp.npos){
            int pos=tmp.find("woc");
            tmp.replace(pos,3,"drib");
        }
        //规则 3
        int a=0,b=0,c=0;
        for(int i=tmp.size()-1;i>=0;i--){
            if(tmp[i]=='r')a++;
            if(tmp[i]=='p'){
                b++;
                if(tmp[i+1]=='r')c++;
            }
        }
        int rp=a*5+b*5+c*20;
        //比较
        if(rp>maxrp){
            maxrp=rp;
            str=s;
        }
        //相同取名字短的
        if(rp==maxrp){
            if(s.size()<str.size())str=s;
        }
    }
    cout<<str<<endl;
    cout<<maxrp<<endl;
}
```

漏洞和补丁

【题目描述】
由 FF 博士带领的团队要开始准备 ICG 模拟赛了，但他们发现机房的电脑有很多漏洞。
为了修补漏洞，只有下载各种补丁。 但是由于这些漏洞太过久远，不一定能找到能够修复漏洞的补丁。
以下有 n 个漏洞和 m 个补丁，用字符串表示。
若其中一个漏洞被一个补丁包含或包含一个补丁则认为这是一个可以被修复的漏洞。
(不区分大小写,一个补丁可以修复多个漏洞，一个漏洞只能被修复一次)
输出能被修复的漏洞总数。

【输入】
第一行，两个整数 n,m；
第 2 行到第 N+1 行,表示漏洞的名称；
接下来 M 行表示能下载到的各个补丁的名称。
【输出】
1 个整数，表示能被修复的漏洞总数。
【样例输入】
3 3
Abc
DGFYJJa
ICG2010
aBCd
YHDajfje
ICG
【样例输出】
2
【样例解释】
漏洞是
Abc
DGFYJJa
ICG2010
补丁是：
aBCd
YHDajfje
IcG
其中 1 号漏洞 Abc 被 aBCd 包含，可修复。3 号漏洞 ICG2010 包含 IcG.所以有 2 个漏洞可以被修复。
(可能会出现同名的补丁或漏洞。)
【数据范围】
对于每个给出的漏洞和补丁，长度不超过 256；
对于 100%的数据，m,n<=100
【分析】
枚举每个补丁，对于当前补丁，标记其能修复哪些漏洞，最后统计能被修复的漏洞个数。
【代码】

```
#include<iostream>
#include<string>
using namespace std;
string ld[101],s;
bool fix[101];
int n,m;
int main(){
    cin>>n>>m;
    //读入 n 个漏洞
    for(int i=1;i<=n;i++){
```

oiClass

```
        cin>>ld[i];
        //将字符串转成大写
        for(int j=0;j<ld[i].size();j++){
            if(ld[i][j]>='a'&&ld[i][j]<='z'){
                ld[i][j]=ld[i][j]-'a'+'A';
            }
        }
    }
    //读入 m 个补丁
    for(int i=1;i<=m;i++){
        cin>>s;
        //将字符串转成大写
        for(int j=0;j<s.size();j++){
            if(s[j]>='a'&&s[j]<='z'){
                s[j]=s[j]-'a'+'A';
            }
        }
        //对于当前补丁 s，枚举所有漏洞，判断可以修复哪些漏洞
        for(int j=1;j<=n;j++){
            if(ld[j].find(s)!=s.npos||s.find(ld[j])!=s.npos){
                fix[j]=true;
            }
        }
    }
    //统计能够被修复的漏洞的个数
    int cnt=0;
    for(int i=1;i<=n;i++){
        if(fix[i]==true)cnt++;
    }
    //输出结果
    cout<<cnt;
}
```