最长公共子串

■ 题目描述

题目标题:
计算两个字符串的最大公共字串的长度，字符不区分大小写
详细描述:
接口说明
原型:
int getCommonStrLength(char * pFirstStr, char * pSecondStr);
输入参数:
    char * pFirstStr //第一个字符串
    char * pSecondStr//第二个字符串

输入描述:

> 输入两个字符串

输出描述:

> 输出一个整数

```
#include<iostream>
#include<vector>
#include<string>
using namespace std;

//动态规划
int getCommonStrLength(string pFirststr,string pSecondstr)
{
    vector<vector<int>> res(pFirststr.size(),vector<int>(pSecondstr.size())
);
    int maxlen=0;
    for(int i=0;i<pFirststr.size();i++)
    {
        for(int j=0;j<pSecondstr.size();j++)
        {
            if(pFirststr[i]==pSecondstr[j])
            {
                if(i==0||j==0)
                    res[i][j]=1;
                else
                    res[i][j]=res[i-1][j-1]+1;
            }
            else
                res[i][j]=0;
            if(res[i][j]>maxlen)
```

```
                maxlen=res[i][j];
            }
        }
    }
    return maxlen;
}

int main()
{
    string str1,str2;
    while(cin>>str1>>str2)
    {
        if(str1.size()>str2.size())
            swap(str1,str2);
        cout<<getCommonStrLength(str1,str2)<<endl;
    }
    return 0;
}
```

查找两个字符串a,b中的最长公共子串。若有多个，输出在较短串中最先出现的那个。
注：子串的定义：将一个字符串删去前缀和后缀（也可以不删）形成的字符串。请和“子序列”的概念分开！

本题含有多组输入数据！

输入描述:

> 输入两个字符串

输出描述:

> 返回重复出现的字符

```
#include<iostream>
#include<vector>
#include<string>
using namespace std;

string find_str(string str1,string str2)
{
    vector<vector<int>> res(str1.size(),vector<int>(str2.size()));
    int maxlen=0;
    int maxend=0;
    for(int i=0;i<str1.size();i++)
```

```
    {
        for(int j=0;j<str2.size();j++)
        {
            if(str1[i]==str2[j])
            {
                if(i==0||j==0)
                {
                    res[i][j]=1;//相同位置给 1
                }
                else
                {
                    res[i][j]=res[i-1][j-1]+1;//赋值 1  并计长度
                }
            }
            else
                res[i][j]=0;
            if(res[i][j]>maxlen)
            {
                //找到更长的公共子串,进行更新 maxlen，maxend
                maxlen=res[i][j];
                maxend=i;
            }
        }
    }
    return str1.substr(maxend-maxlen+1,maxlen);
}

int main()
{
    string str1,str2;
    while(cin>>str1>>str2)
    {
        if(str1.size()>str2.size())
            swap(str1,str2);//str1 中放较短的字符串
        cout<<find_str(str1,str2)<<endl;
    }
    return 0;
}
```