DP问题：最长回文串问题

例题：1040 Longest Symmetric String (25分)

Given a string, you are supposed to output the length of the longest symmetric sub-string. For example, given `Is PAT&TAP symmetric?`, the longest symmetric sub-string is `s PAT&TAP s`, hence you must output `11`.

Input Specification:

Each input file contains one test case which gives a non-empty string of length no more than 1000.

Output Specification:

For each test case, simply print the maximum length in a line.

Sample Input:

```
Is PAT&TAP symmetric?
```

Sample Output:

```
11
```

思路：

1. 算法笔记说他是最好理解的DP问题。的确，我也这么觉得，非常好理解。
2. 初始化一个DP数组，`dp[i][j]`是指`string`中起点为`i`终点为`j`的子字符串是否为回文串
3. 首先初始化`dp[i][i]`为`true`，很好理解，字符串长度只有1的当然是回文串。
4. 再然后根据输入的`string`判断`dp[i][i+1]`，即长度为2的子串是否为回文串。往后的递推会利用这个结果。
5. 然后根据长度进行循环判断。将`len`初始化为2，即从长度为3开始判断，直到`len`增长到`string.size()-1`为止。
6. 递推思路是：先查看`dp[i+1][j-1]`是否为回文串，只要上一个串不是回文串，长度+2之后的回文串也不可能是回文串
7. 若是，`dp[i][j]=string[i]==string[j]?true:false`
8. 由此得到递推方程：

$$dp[\,i\,][\,j\,]=\begin{cases} string[\,i\,]==string[\,j\,], dp[\,i+1\,][\,j-1\,]==true; \\ false, dp[\,i+1\,][\,j-1\,]==false; \end{cases}$$

9. 然后写代码吧~

参考代码

```cpp
#include <iostream>
#include <vector>
#include <string>
using namespace std;
int main()
{
    string str;
    int maxLen = 0;
    getline(cin, str);
    vector<vector<bool>> dp(str.size());
    for (int i = 0; i < str.size(); i++)
        dp[i].resize(str.size());
    for (int i = 0; i < str.size(); i++)
        dp[i][i] = true;
    for (int i = 0; i < str.size() - 1; i++)
        if (str.at(i) == str.at(i + 1))
            dp[i][i + 1] = true;
    for(int len=2;len<str.size();len++)
    {
        int i = 0, j = i + len;
        while(j<str.size())
        {
            if (dp[i + 1][j - 1] && str.at(i) == str.at(j))
            {
                dp[i][j] = true;
                maxLen = len;
            }
            i++, j++;
        }
    }
    cout << maxLen + 1 << endl;
    return 0;
}
```