一、问题分析

1.1 处理的对象（数据）

多组字符串

1.2 实现的功能

给定一个字符串，求该字符串中包含的最长回文子串。

1.3 结果显示

输出起始位置最小的位置编号(从 0 开始)以及回文子串的长度。

1.4 样例求解过程

【样例输入】

```
2
12abcbats
qwerabccbaw
```

【样例输出】

```
2 5
4 6
```

求解过程

遍历字符串，将字符串的每个字符和每连续两个字符之间作为中心向左右拓展回文子串。

字符串1：12abcbats 以c为中心，向左右拓展，最长回文子串为abcba，开始下标为2，长度为5

字符串2：qwerabccbaw 以7、8下标中间为中心，向左右拓展，最长回文子串为abccba,开始下标为4，长度为6

二、数据结构和算法设计

1.算法思想的设计

采用中心扩展法。

由于回文字符串分为奇数长度字符和偶数长度字符，奇数常数字符为字符串中央元素，偶数长度字符串中心为中央两字符之间。于是采用遍历整个字符串和两字符间的间隙共2*n-1个可能的中心位置，以中心为起点向左右拓展回文字符串。

当且仅当遇到第一个最左边元素不等于最右边元素，本次中心扩展结束，上次循环的结果即为本次扩展的最长回文子串。

综合完整遍历的每个中心扩展的最长回文子串即可得到最长回文子串。

2.关键功能的算法步骤

2.1 循环2*n-1次，并通过循环变量计算出最开始时子串最左边下标和最右边下标

2.2 保证下标不越界的情况下判断最左边元素和最右边元素是否相等，相等则左右均向外扩展一格

2.3 计算出拓展出的子串长度，如果大于已记录最长长度则覆盖，并记下此时最左边元素下标

```
string s;

cin>>s;

int n=s.length(),ans=0;

int maxl=0;

for(int i=0; i<2*n; i++) {

    int l=i/2,r=(i+1)/2;

    while(l>=0&&r<n&&s[l]==s[r])

    {

        l--;

        r++;

    }

    if(r-l-1>ans)

    {

        ans=r-l-1;

        maxl=l+1;

    }

}

cout<<maxl<<" "<<ans<<endl;
```

三、算法性能分析

1.时间复杂度

$O(n^2)$，n为字符串的长度，可能的回文中心有2*n-1个，每个回文中心最多会向外拓展O(n)次

2.空间复杂度

O(1)，没有开辟新的数组空间

3.总结

中心扩展算法容易理解和实现，是对暴力枚举的一种优化，同时也应用了动态规划状态转移的思想

4.不足之处

中心扩展法没有时间复杂度O(n)的Manacher 算法效率高。