

C++方向编程题答案

第四周

day20

题目ID: 36836-字符串反转

链接: <https://www.nowcoder.com/practice/e45e078701ab4e4cb49393ae30f1bb04?tpId=37&&tqId=21235&rp=1&ru=/activity/oj&qru=/ta/huawei/question-ranking>

【题目解析】:

本题题意明确

【解题思路】:

字符串反转, 需要交换首尾字符, 设置首尾两个位置start, end, 每次交换首尾字符, 然后start++, end--, 直到start, end相遇, 反转完成。

【示例代码】

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

string reverseString(string s) {
    if (s.empty())
        return s;
    size_t start = 0;
    size_t end = s.size() - 1;
    while (start < end)
    {
        swap(s[start], s[end]);
        ++start;
        --end;
    }
    return s;
}

int main()
{
    string s;
    getline(cin, s);
    cout<<reverseString(s)<<endl;
    return 0;
}
```

题目ID: 36899-公共字符串计算

链接: <https://www.nowcoder.com/practice/98dc82c094e043ccb7e0570e5342dd1b?tpId=37&&tqId=21298&rp=1&ru=/activity/oj&qru=/ta/huawei/question-ranking>

【题目解析】：

本题题意明确

【解题思路】：

求最大公共子串，使用递推实现 假设 $x(i)$: 字符串第 i 个字符 $y(j)$: 字符串第 j 个字符 $dp[i][j]$: 以 $x(i), y(j)$ 结尾的最大子串长度 比如: $x: "abcde"$ $y: "bcdae"$ $dp[2][1]$: 以 $x(2), y(1)$ 结尾的最大子串长度 即: x 遍历到 "abc", y 遍历到 "bc", 都以字符 'c' 结尾时最大公共子串为 "bc" 故: 当计算 $dp[i][j]$ 时, 首先看 $x(i), y(j)$ 的值: (1) : $x(i) == y(j)$ 当前两个字符串结尾的字符相等, $dp[i][j] = dp[i-1][j-1] + 1$ 即个它的长度加1 (2) : $x(i) != y(j)$ 当前两个字符串结尾的字符不相等, 说明没有以这连个字符结尾的公共子串 即 $dp[i][j] = 0$ (3) : $dp[0][j]$ 和 $dp[i][0]$ 表示以某个子串的第一个字符结尾, 最大长度为1 如果 $x(0) == y(j)$ 或者 $x(i) == y(0)$, 则长度为1, 否则为0 当 dp 中的所有元素计算完之后, 从中找打最大的值输出

【示例代码】

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

int main()
{
    int max = 0;    //max初值.
    string str1, str2;
    while (cin >> str1 >> str2)
    {
        int len1 = str1.size();
        int len2 = str2.size();
        int max = 0;
        //所有值初始化为0
        vector<vector<int>> dp(len1, vector<int>(len2, 0));
        //计算dp
        for (int i = 0; i < len1; i++)
        {
            for (int j = 0; j < len2; j++)
            {
                //如果当前结尾的字符相等, 则在dp[i-1][j-1]的基础上加1
                if (str1[i] == str2[j])
                {
                    if (i >= 1 && j >= 1)
                        dp[i][j] = dp[i - 1][j - 1] + 1;
                    else
                        //dp[i][0] or dp[0][j]
                        dp[i][j] = 1;
                }
                //更新最大值
                if (dp[i][j] > max)
                    max = dp[i][j];
            }
        }
        cout << max << endl;
    }
}
```

```
return 0;  
}
```

比特科技整理