C++方向编程题答案

第二周

day12

> 题目ID：24951-二进制插入
>
> 链接：https://www.nowcoder.com/practice/30c1674ad5694b3f8f0bc2de6f005490?tpId=8&&tqId=11019&rp=1&ru=/activity/oj&qru=/ta/cracking-the-coding-interview/question-ranking

```
/*
           i    j
m:1024: 100000000 00
n:19  :     10011
要把m的二进制值插入m的第j位到第i位，只需要把n先左移j位，然后再进行或运算（|）即可。

10000000000
00001001100
--------------------
10001001100 ---->1100（十进制）
*/

class BinInsert {
public:
    int binInsert(int n, int m, int j, int i) {
        m <<= j;
        return n | m;
    }
};
```

> 36899-公共字串计算
>
> 链接：https://www.nowcoder.com/practice/98dc82c094e043ccb7e0570e5342dd1b?tpId=37&&tqId=21298&rp=1&ru=/activity/oj&qru=/ta/huawei/question-ranking

```
/*
    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
*/

#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;
}
```