C++方向编程题答案

第三周

day13

> **题目ID：36898-参数解析**
>
> 链接：https://www.nowcoder.com/practice/668603dc307e4ef4bb07bcd0615ea677?tpId=37&&tqId=21297&rp=1&ru=/activity/oj&qru=/ta/huawei/question-ranking
>
> **【题目解析】：**
>
> 本题考察string的运用
>
> **【解题思路】：**
>
> 本题通过以空格和双引号为间隔，统计参数个数。对于双引号，通过添加flag，保证双引号中的空格被输出。
>
> **【示例代码】**

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string str;
    while (getline(cin, str))
    {
        int count = 0;
        //首先计算参数数量
        for (int i = 0; i < str.size(); i++)
        {
            if (str[i] == ' ')
                count++;
            //如果是双引号，向后遍历，直到下一个双引号结束
            if (str[i] == '"')
            {
                do
                {
                    i++;
                } while (str[i] != '"');
            }
        }
        //以空格计算个数，空格数量比参数个数少1
        cout << count + 1 << endl;
//用flag表示是否包含双引号，0表示有双引号
//双引号中的空格要打印出来
//用异或改变flag的值，两个双引号可以使flag复原
        int flag = 1;
        for (int i = 0; i < str.size(); i++)
        {
   //有双引号，flag通过异或变为0，下一次再遇到双引号，flag置为1
            if (str[i] == '"')
```

```
                flag ^= 1;
    //双引号和普通空格不打印
            if (str[i] != ' ' && str[i] != '"')
                cout << str[i];
    //双引号中的空格要打印
            if (str[i] == ' ' && (!flag))
                cout << str[i];
    //遇到双引号之外的空格就换行
            if (str[i] == ' ' && flag)
                cout << endl;
        }
        cout << endl;
    }
    return 0;
}
```

题目ID:46574-跳石板

链接：https://www.nowcoder.com/practice/4284c8f466814870bae7799a07d49ec8?tpId=85&&tqId=29852&rp=1&ru=/activity/oj&qru=/ta/2017test/question-ranking

【题目解析】：

题目的意思是从N开始，最少需要累加几步可以变成指定的数字M，每次累加的值为当前值的一个约数。

【解题思路】：

将1 - M个石板看做一个结果数组stepNum，每个stepNum[i]储存着从起点到这一步最小的步数，其中0为不能到达。
从起点开始对stepNum进行遍历，先求i的所有约数（即从stepNum[i]能走的步数），然后更新那几个能到达的位置的最小步数。如果不能到达则更新为此时位置的最小步数 + 1，如果是能到达的就更新为min（已记录的最小步数，此处的最小步数 + 1）），遍历一遍后得到结果。

【示例代码】

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
//计算约数，求除了1和本身的约数
void divisorNum(int n, vector<int> &divNum)
{
    for (int i = 2; i <= sqrt(n); i++)
    {
        if (n%i == 0)
        {
            divNum.push_back(i);
            //非平方数时还有另一个数也要加入
            if (n / i != i)
                divNum.push_back(n / i);
        }
    }
}
int Jump(int N, int M)
{
    //储存的到达此stepNum点的步数，初始N为1步，从N到N为1步
    vector<int> stepNum(M + 1, 0);
```

```
    stepNum[N] = 1;

    for (int i = N; i < M; i++)
    {
        //N的所有约数，即为从本身这个点开始能走的数量
        vector<int> divNum;

        //0代表这个点不能到
        if (stepNum[i] == 0)
            continue;

        //求出所有能走的步数储存在divNum的容器中
        divisorNum(i, divNum);

        for (int j = 0; j < divNum.size(); j++)
        {
            //由位置i出发能到达的点为 stepNum[divNum[j]+i]
            if ((divNum[j] + i) <= M && stepNum[divNum[j] + i] != 0)
                stepNum[divNum[j] + i] = min(stepNum[divNum[j] + i],
stepNum[i] + 1);
            else if ((divNum[j] + i) <= M)
                stepNum[divNum[j] + i] = stepNum[i] + 1;
        }
    }

    if (stepNum[M] == 0)
        return -1;
    else
        //初始化时多给了一步，故需要减1
        return stepNum[M] - 1;
}

int main()
{
    int n, m;
    cin >> n >> m;
    cout << Jump(n, m) << endl;
    return 0;
}
```