

递归习题

T E A C H I N G C O U R S E W A R E P O W P O I N T

授课时间：2025.08.01



目录

- PART-01 题目描述 TEACHING COURSEWARE
- PART-02 概念分析 TEACHING COURSEWARE
- PART-03 经典例题 TEACHING COURSEWARE
- PART-04 教学反思 TEACHING COURSEWARE

01

题目描述

TEACHING
COURSEWARE

TEACH

题目描述

题目描述

实现一个递归函数 $w(a, b, c)$ ，其计算规则如下：

1. 如果 $a \leq 0$ 或 $b \leq 0$ 或 $c \leq 0$ ，返回 1；
2. 如果 $a < b$ 并且 $b < c$ ，返回 $w(a, b, c - 1) + w(a, b - 1, c - 1) - w(a, b - 1, c)$ ；
3. 其他情况，返回 $w(a - 1, b, c) + w(a - 1, b - 1, c) + w(a - 1, b, c - 1) - w(a - 1, b - 1, c - 1)$ 。

给定一组整数 a, b, c ，满足：

- 输入仅包含一组测试用例；
- a, b, c 均为小于 10 的非负整数（即 $0 \leq a, b, c \leq 10$ ）。

请根据上述规则计算 $w(a, b, c)$ 的值并输出。

输入格式

一行，包含三个整数 a, b, c ，用空格分隔。

输出格式

一行，格式为： $w(a, b, c) = ans$ ，其中 ans 为计算结果。

输入样例

```
1 1 1
```

Copy

输出样例

```
w(1, 1, 1) = 2
```

Copy

提示

- 本题仅需处理一组测试用例，且输入范围被严格限制在 $[0, 10]$ ，直接按照递归定义实现即可。
- 注意判断条件的优先级：先判断是否满足规则1，再判断规则2，最后执行规则3。

02

知识回归

TEACHING
COURSEWARE

TEACH



知识回归

递归的概念

递归（Recursion）是一种算法思想，指函数通过调用**自身**来解决问题的方式。其核心是将一个复杂问题分解为与原问题结构**相同**但规模**更小**的子问题，直到子问题小到可以**直接解决**。

递归必须满足两个要素：

基准情况：存在至少一种无需递归就能直接解决的情况（**终止条件**），避免无限递归。

03

题目解答

TEACHING
COURSEWARE

TEACH

题目解答

递归函数逻辑：

规则 1：当 a 、 b 、 c 中任意一个小于等于 0 时，直接返回 1。这是递归的终止条件之一，用来处理最基础的简单情况。例如，当 a 为 0 时，不管 b 和 c 是什么，函数结果都是 1。

规则 2：当 $a < b$ 并且 $b < c$ 时，返回 $w(a, b, c - 1) + w(a, b - 1, c - 1) - w(a, b - 1, c)$ 。这里是一种特定条件下的递归组合，通过对 c 减 1 以及对 b 减 1 等操作，将问题分解为更小的子问题，然后根据这些子问题的结果计算当前问题的结果。

规则 3：在不满足规则 1 和规则 2 的其他情况下，返回 $w(a - 1, b, c) + w(a - 1, b - 1, c) + w(a - 1, b, c - 1) - w(a - 1, b - 1, c - 1)$ 。同样是把当前的 a 、 b 、 c 相关的问题，通过让 a 、 b 、 c 分别减 1 等方式，拆分成几个子问题，再利用子问题结果计算当前结果。

04

标准程序

TEACHING
COURSEWARE

TEACH

标准程序

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

// 递归实现题目要求的函数 w(a, b, c)
long long w(int a, int b, int c) {
    // 规则 1: 如果 a <= 0 或 b <= 0 或 c <= 0, 返回 1
    if (a <= 0 || b <= 0 || c <= 0) {
        return 1;
    }
    // 规则 2: 如果 a < b 并且 b < c, 返回 w(a, b, c - 1) + w(a, b - 1, c - 1) - w(a, b - 1, c)
    else if (a < b && b < c) {
        return w(a, b, c - 1) + w(a, b - 1, c - 1) - w(a, b - 1, c);
    }
    // 规则 3: 其他情况, 返回 w(a - 1, b, c) + w(a - 1, b - 1, c) + w(a - 1, b, c - 1) - w(a - 1, b - 1, c - 1)
    else {
        return w(a - 1, b, c) + w(a - 1, b - 1, c) + w(a - 1, b, c - 1) - w(a - 1, b - 1, c - 1);
    }
}

int main() {
    int a, b, c;
    // 输入三个整数 a, b, c
    cin >> a >> b >> c;
    // 计算并按照要求格式输出结果
    cout << "w(" << a << ", " << b << ", " << c << ") = " << w(a, b, c) << endl;
    return 0;
}
```