# 题目

请你编写一个程序来计算两个日期之间隔了多少天。

日期以字符串形式给出，格式为 YYYY-MM-DD，如示例所示。

示例 1：

输入：date1 = "2019-06-29", date2 = "2019-06-30"

输出：1

示例 2：

输入：date1 = "2020-01-15", date2 = "2019-12-31"

输出：15

提示：

给定的日期是 1971 年到 2100 年之间的有效日期。

# 分析

要计算两个日期的间隔天数，核心思路是将两个日期统一转换为“距离某个固定起始日期的总天数”，再通过差值的绝对值得到结果。这种方法无需直接处理日期加减的复杂逻辑，通过“基准天数”简化计算，且能确保精度。

解题思路：

1、固定起始基准：选择1971-01-01作为起始日期（题目日期范围包含此日期），所有目标日期都转换为“从该起始日到目标日的总天数”。

2、日期解析：将输入的字符串日期（YYYY-MM-DD）拆分为年、月、日三个整数。

3、总天数计算：

- 年份贡献：计算从起始年（1971）到目标年的前一年的总天数，需区分闰年（366天）和平年（365天）。

- 月份贡献：计算目标年中从1月到目标月的前一个月的总天数，2月需根据闰年调整（29天/28天）。

- 日期贡献：直接加上目标日的天数。

4、计算间隔：两个日期的总天数之差的绝对值，即为间隔天数。

关键工具：闰年判断

闰年需满足以下任一条件：

- 能被 400 整除；

- 能被 4 整除但不能被 100 整除。

代码：

class Solution {

public:

int daysBetweenDates(string date1, string date2) {

// 核心：将两个日期转为“距离1971-01-01的总天数”，返回差值绝对值

return abs(calculateTotalDays(date1) - calculateTotalDays(date2));

}

private:

// 辅助函数：判断某一年是否为闰年

bool isLeapYear(int year) {

if (year % 400 == 0) {

return true; // 能被400整除是闰年

}

if (year % 100 == 0) {

return false; // 能被100整除但不能被400整除，不是闰年

}

return year % 4 == 0; // 能被4整除且不满足前两个条件，是闰年

}

// 辅助函数：计算日期date距离1971-01-01的总天数

int calculateTotalDays(string date) {

// 1. 解析日期字符串：YYYY-MM-DD → 年、月、日

int year = stoi(date.substr(0, 4)); // 提取前4个字符作为年

int month = stoi(date.substr(5, 2)); // 提取第5-6个字符作为月（substr(起始索引, 长度)）

int day = stoi(date.substr(8, 2)); // 提取第8-9个字符作为日

// 2. 定义平年各月天数（索引0未使用，对应1-12月）

int monthDays[] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

int total = 0;

// 3. 累加“1971年”到“目标年的前一年”的总天数

for (int y = 1971; y < year; ++y) {

total += isLeapYear(y) ? 366 : 365;

}

// 4. 累加“目标年1月”到“目标月的前一个月”的总天数

for (int m = 1; m < month; ++m) {

if (m == 2) { // 2月需判断闰年

total += isLeapYear(year) ? 29 : 28;

} else {

total += monthDays[m];

}

}

// 5. 累加目标日的天数

total += day;

return total;

}

};

复杂度分析

- 时间复杂度：O(1)。年份循环最多遍历2100-1971=129次，月份循环最多遍历12次，均为常数级操作，与输入无关。

- 空间复杂度：O(1)。仅使用固定数量的变量和数组（monthDays长度固定），无额外动态空间开销。