



授课时间: 2025.08.07









PART-01 栈

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

PART-02

队列

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

PART-03

map

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

PART-04

提示

TEACHING COURSEWARE









stack (栈):后进先出的容器

1. 为什么需要stack?

在很多场景中,我们需要"最后添加的元素最先被处理"的逻辑,例如:

函数调用的递归实现(调用栈)

表达式求值(后缀表达式)

括号匹配校验

撤销操作(如文本编辑器的 Ctrl+Z)

stack封装了这种逻辑,避免手动实现时的繁琐和错误。

2. 基本操作

使用stack需包含头文件〈stack〉,核心成员函数如下:

操作 函数示例 说明

入栈(添加元素) s. push(10); 在栈顶添加元素

出栈(移除元素) s. pop(); 移除栈顶元素(不返回值, 需先判断非空)

访问栈顶元素 s. top(); 返回栈顶元素(需先判断非空)

判断栈是否为空 s. empty(); 空返回true, 否则false

获取元素个数 s. size(); 返回栈中元素数量



假设一个表达式有英文字母(小写)、运算符(+、-、\*、/)和左右小(圆)括号构成,以 @ 作为表达式的结束符。请编写一个程序检查表达式中的左右圆括号是否匹配,若匹配,则输出 YES;否则输出 NO。表达式长度小于 255, 左圆括号少于 20 个。

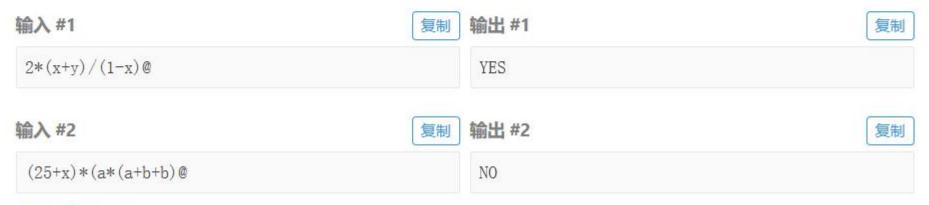
# 输入格式

一行: 表达式。

# 输出格式

一行: YES 或 NO。

# 输入输出样例



# 说明/提示

表达式长度小于255, 左圆括号少于20个。





通过栈解决括号匹配问题。 核心思路是:遇到左括号时入栈,遇到右 括号时检查栈顶是否有对应的左括号(有 则出栈,无则匹配失败)。 最终栈为空则表示所有括号完全匹配。



```
#include<bits/stdc++.h>
using namespace std;
int main() {
   stack(char) st; // 定义一个字符栈, 用于存储左括号
  char c; // 用于读取表达式中的每个字符
  // 循环读取表达式字符, 直到遇到结束符'@'
  while (cin >> c && c != '@') {
      if (c == '(') {
        // 遇到左括号,入栈
        st.push(c);
      } else if (c == ')') {
        // 遇到右括号时,检查栈中是否有对应的左括号
        if (st.empty()) {
           // 栈为空,说明没有匹配的左括号,直接输出NO并结束程序
           cout << "NO" << endl;
           return 0;
         } else {
           // 栈非空,弹出栈顶的左括号(表示匹配成功一对)
           st.pop();
```





队列 02 TEACHING COURSEWARE **TFACH** 

# 队列



queue (队列): 先进先出的容器 (FIF0)

1. 为什么需要queue?

当需要 "最先添加的元素最先被处理" 时, queue是最佳选择, 例如:

任务调度 (先提交的任务先执行)

消息队列(先发送的消息先处理)

广度优先搜索(BFS)

queue封装了这种顺序逻辑,简化了"排队"场景的实现。

2. 基本操作

使用queue需包含头文件〈queue〉,核心成员函数如下:

操作 函数示例 说明

入队(添加元素) q. push(10); 在队尾添加元素

出队(移除元素) q. pop(); 移除队头元素(不返回值,需先判断非空)

访问队头元素 q. front(); 返回队头元素(需先判断非空)

访问队尾元素 q. back(); 返回队尾元素(需先判断非空)

判断队列是否为空 q.empty(); 空返回true, 否则false

获取元素个数 q. size(); 返回队列中元素数量





## 题目描述

■ 复制 Markdown 【3展开 ② 进入 IDE 模式

n 个人围成一圈,从第一个人开始报数,数到 m 的人出列,再由下一个人重新从 1 开始报数,数到 m 的人再出圈,依次类推,直到所有的人都出圈,请输出依次出圈人的编号。

注意:本题和《深入浅出-基础篇》上例题的表述稍有不同。书上表述是给出淘汰 n-1 名小朋友,而该题是全部出圈。

## 输入格式

输入两个整数 n, m。

### 输出格式

输出一行 n 个整数,按顺序输出每个出圈人的编号。

## 输入输出样例



## 说明/提示

 $1 \le m, n \le 100$ 





队首元素代表当前需要报数的人; 若报数未到 m,则将队首元素移到队尾(模 拟 "下一个人继续报数"); 若报数到 m,则将队首元素移除(模拟 "淘汰"),并记录该元素。





```
#include <bits/stdc++.h>
using namespace std;
int main()
  queue(int) q; // 定义一个整数类型的队列,用于模拟围成一圈的人
  int n, m; // n表示总人数, m表示计数淘汰的间隔
  cin >> n >> m; // 从输入读取n和m的值
  int cnt = 1; // 计数器,用于记录当前报数的数字
  // 初始化队列: 将1到n的编号依次加入队列,模拟n个人围成一圈
  for(int i = 1; i <= n; i++)
     q.push(i); // 将编号i加入队列尾部
   // 当队列不为空时,继续循环(即还有人未被淘汰)
   while(q.size() > 0)
      for(int i = 1; i < m; i++)
        q.push(q.front());
        q.pop();
      cout << q.front() << " "; // 输出被淘汰的人的编号
                             // 将队首元素移除队列(淘汰)
        q.pop();
      // 如果当前报数等于m, 说明该淘汰队首的人了
  cout << endl; // 输出结束后换行
   return 0;
```









在 STL 中

map是一种关联容器,用于存储键值对(key-value)数据,并通过键(key)快速查找对应的值(value)。它的底层通常基于红黑树实现,保证了高效的插入、删除和查找操作。

为什么需要map?

在很多场景中,我们需要通过一个"标识"快速定位到对应的数据,例如:

字典(通过单词查找查找释义)

用户信息表(通过用户名查找用户详情)

统计单词出现出现次数(单词作为键,次数出现次数作为值)

如果用数组或vector存储这类数据,查找时需要遍历整个容器(时间复杂度 0 (n)),而map的查找效率可达 0 (log n),且能自动动根据键排序。





```
核心成员函数
操作 函数示例 说明
访问 / 修改元素 dict["apple"] = 5; 通过键访问值,若键不存在则自动插入
查找元素 auto it = dict. find("banana"); 查找键,返回迭代器(找到则指向该键值对,
否则指向 end())
删除元素 dict. erase("apple"); 通过键删除元素;也可通过迭代器删除
获取元素个数 dict. size(); 返回键值对数量
判断是否为空 dict. empty(); 空返回true,否则false
清空所有元素 dict. clear(); 清空 map
查询个数 dict. count("apple"); 返回指定键在 map 中出现的次数(0 或 1)
```





```
map的遍历:
for(auto x : dict)
{
    cout<<x.first<<" "<<x.second<<endl;
}
auto
```



出题是一件痛苦的事情!

相同的题目看多了也会有审美疲劳,于是我舍弃了大家所熟悉的 A+B Problem, 改用 A-B 了哈哈!

#### 题目描述

给出一串正整数数列以及一个正整数 C,要求计算出所有满足 A-B=C 的数对的个数(不同位置的数字一样的数对算不同的数对)。

#### 输入格式

输入共两行。

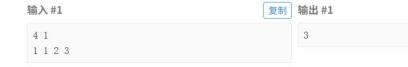
第一行,两个正整数 N, C。

第二行,N个正整数,作为要求处理的那串数。

#### 输出格式

一行,表示该串正整数中包含的满足 A-B=C 的数对的个数。

#### 输入输出样例



### 输出格式

一行,表示该串正整数中包含的满足 A-B=C 的数对的个数。

#### 输入输出样例



#### 说明/提示

对于 75% 的数据, $1 \le N \le 2000$ 。

对于 100% 的数据, $1 \le N \le 2 \times 10^5$ , $0 \le a_i < 2^{30}$ , $1 \le C < 2^{30}$ 。

2017/4/29 新添数据两组

```
#include<bits/stdc++.h>
using namespace std;
const long N=200010;
int a[N];
int main()
    int n,c;
    cin>>n>>c;
    map<long long, long long>mp;
    for(int i=1; i<=n; i++)
        cin>>a[i];
        mp[a[i]]++;
    long long cnt=0;
    for(int i=1;i<=n;i++)</pre>
        cnt+=mp[a[i] + c];
    cout<<cnt;
    return 0;
```





提示 04 TEACHING COURSEWARE **TFACH** 

# 代码

