

阅卷评测报告: Vector 删除偶数

A) 分项评分与理由

1. 关键 bug 命中(1 / 2 分)

- 理由: 准确指出了“迭代器失效”的核心问题, 也给出了 erase-remove 的正确方向。但是, 手动循环的修正代码(修法1)中出现了严重的语法错误 `ita.erase(it)`(应为 `it = a.erase(it)`), 且模板代码中赋值语句残缺(`it` 悬空), 这作为“正确示范”是不合格的, 直接运行会编译失败。

2. 解释是否“新手能懂”(2 / 2 分)

- 理由: 使用了“座位号”和“搬家”的比喻, 非常直观地解释了 vector 连续内存移动导致迭代器失效的原理, 完全避免了晦涩的术语堆砌。

3. 现代 C++ 写法与迁移(2 / 2 分)

- 理由: 明确推荐了 erase-remove 惯用法, 并且指出了 `<bits/stdc++.h>` 与工程头文件的区别, 引导方向非常正确。

4. 结构遵守(2 / 2 分)

- 理由: 完全遵守了“概览-思路-头文件-代码修正-原理-模板-加练”的结构, 条理清晰, 利于复盘。

5. 工程与边界意识(1 / 2 分)

- 理由: 虽然给出的代码逻辑能处理连续偶数, 但未明确指出原代码在“连续偶数”时会漏删(跳过)的边界行为。更关键的是, 完全未提及性能差异: 未告知学生手动 erase 循环是 $O(N^2)$, 而 `remove_if` 是 $O(N)$, 这是工程上选择后者的决定性因素。

6. 学姐语气与情绪支持(2 / 2 分)

- 理由: 完美接住了用户的“自我怀疑”, 肯定了用户的逻辑直觉(“逻辑没问题”), 把错误归结为 C++ 特性而非用户基础差, 情绪价值给满。

B) 总分: 10 / 12

C) 扣分点(详细复盘)

- 代码语法严重笔误: 在“修法1”和“模板1”中, `ita.erase(it)` 和悬空的 `it` 都是无法通过编译的错误写法。作为教学回答, 代码必须由编译器验证通过。
- 工程性能盲区: 未解释 erase 在 vector 中会导致后续元素整体搬移, 在循环中这样做会导致平方级复杂度($O(N^2)$), 这在工程中是巨大的隐患。
- 边界场景解释不足: 没有具体演示当数组为 `{2, 2, 3}` 时, 错误的 `++it` 是如何导致第二个 2 被跳过的, 这能加深对“迭代器行为”的理解。
- 模板不够严谨: 模板1 的伪代码写法过于随意, 容易让新手困惑(例如 `it` 后面没有接 `=`, `else` 位置也不对劲)。

D) 一句话结论

这个回答更像一位**“温柔但马虎的学姐”**: 原理讲解得生动透彻, 非常能治愈新手的挫败感, 但给出的示范代码如果不经检查直接运行, 可能会让学弟/学妹面临“按正确答案改完还是报错”的

二次崩溃。