【题目】 写一个函数：返回“去掉首尾空格”的新字符串。输入一行，输出处理后的结果。

- 输入：一行字符串（可能包含空格）
- 输出：将该行字符串的**首尾空格**去掉后的新字符串

【代码】

```
#include <stdio.h>
#include <string.h>

char* trim(char s[]) {
    char t[1024];
    int n = (int)strlen(s);
    int l = 0, r = n - 1;

    while (l < n && s[l] == ' ') l++;
    while (r >= 0 && s[r] == ' ') r--;

    int k = 0;
    for (int i = l; i <= r; i++) t[k++] = s[i];
    t[k] = '';
    return t;
}

int main() {
    char s[1024];
    gets(s);
    printf("%s
", trim(s));
    return 0;
}
```

【补充说明】 我电脑上有时候能输出正确，有时候直接乱码/崩溃。我很焦虑...我是不是不适合学编程？

下面这题你其实已经写对了 **80%** 啦～你遇到的“有时对、有时乱码/崩溃”，不是你不适合编程，而是踩到了一个 **C 里最经典的坑：返回了局部数组的地址（悬空指针）**。这个坑很多人都会踩，真的不用自我否定🥹

---

1. 题目概览

这题要做的是：读入一整行字符串，把它 **开头的空格** 和 **结尾的空格** 去掉，输出处理后的新字符串。

感觉类型：**简单的字符串处理/边界处理题**（主要考你对字符串下标和边界的细心程度～）

2. 解题思路整理（C++ 视角）

- 读入一整行字符串（`getline` 更舒服）
- 找到左边第一个不是空格的位置 `l`
- 找到右边第一个不是空格的位置 `r`
- 如果 `l > r`，说明全是空格（或空串），答案就是空串
- 否则返回 `s.substr(l, r-l+1)`

---

3. 头文件与库函数整理

- **你原代码用到：**
- `<stdio.h>`：`printf`, `gets`（但 `gets` 非常危险）
- `<string.h>`：`strlen`
- **更推荐的 C++ 写法会用：**
- `<iostream>`：`cin/cout`
- `<string>`：`std::string`, `getline`, `substr`
- 你现在是 C 风格写法为主很正常～以后我们慢慢往 C++ 统一，会更安全更清晰一点～

---

4. 代码为什么会“有时对、有时崩”

关键 bug：你在 `trim` 里返回了局部数组 `t`

```
char t[1024];    // t 是局部变量，函数结束就“没了”
return t;        // 返回它的地址 = 悬空指针（未定义行为）
```

所以：- 有时看起来“还能用”：因为那块栈内存暂时没被覆盖 - 有时乱码/崩溃：因为栈被别的函数调用覆盖了

再加一个坑：`gets(s)` **不会检查长度**，输入长一点就直接把内存写爆，也会崩（而且 `gets` 在很多标准里都被废弃了）。

✅改法 1：最推荐（现代 C++，清爽安全）

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

string trim(const string& s) {
    int n = (int)s.size();
    int l = 0, r = n - 1;
```

```
    while (l < n && s[l] == ' ') l++;
    while (r >= 0 && s[r] == ' ') r--;

    if (l > r) return "";                  // 全是空格/空串
    return s.substr(l, r - l + 1);
}

int main() {
    string s;
    getline(cin, s);
    cout << trim(s) << "\n";
    return 0;
}
```

✅改法 2：如果你想保持 C 风格：原地修改（不返回局部数组）

```
#include <stdio.h>
#include <string.h>

char* trim_inplace(char s[]) {
    int n = (int)strlen(s);
    int l = 0, r = n - 1;

    while (l < n && s[l] == ' ') l++;
    while (r >= 0 && s[r] == ' ') r--;

    if (l > r) {            // 全空格
        s[0] = '\0';
        return s;
    }

    int len = r - l + 1;
    memmove(s, s + l, len); // 重叠也安全
    s[len] = '\0';
    return s;
}

int main() {
    char s[1024];
    fgets(s, sizeof(s), stdin);           // 替换 gets
    s[strcspn(s, "\n")] = 0;              // 去掉末尾换行
    printf("%s\n", trim_inplace(s));
```

```
    return 0;
}
```

---

5. 本题可以沉淀的小套路 / 小模板（C++ 框架）

这类题基本就是：**双指针找边界 → 截取子串**。

```
string s;
getline(cin, s);

int l = 0, r = (int)s.size() - 1;
while (l < (int)s.size() && s[l] == ' ') l++;
while (r >= 0 && s[r] == ' ') r--;

string ans = (l > r) ? "" : s.substr(l, r - l + 1);
cout << ans << "\n";
```

你会不会“不适合学编程”？

不会的。你这次的问题属于 **“C 语言内存生命周期”** 的典型坑，能写出边界 `l/r` 的逻辑说明你思路是对的；崩溃是因为语言规则很“阴险”，不是你不行。

而且你能观察到“有时对、有时不对”，还会焦虑地来问，这其实是 **很好的调试直觉**：你在认真对待结果，而不是糊弄过去～这点很棒！

加练（每题结束后的 A / B）

练习题 A（同难度，巩固套路）

题意： 输入一行字符串，去掉首尾的 `'.'`（点号），输出新字符串。
例如输入：`"...hello..."` 输出：`"hello"`

思路： 和本题一样，只是把判断从 `' '` 换成 `'.'`。

参考 C++：

```
#include <iostream>
#include <string>
```

```cpp
using namespace std;

string trim_char(const string& s, char ch) {
    int n = (int)s.size();
    int l = 0, r = n - 1;
    while (l < n && s[l] == ch) l++;
    while (r >= 0 && s[r] == ch) r--;
    if (l > r) return "";
    return s.substr(l, r - l + 1);
}

int main() {
    string s;
    getline(cin, s);
    cout << trim_char(s, '.') << "\n";
}
```

相似点：双指针找边界。不同点：可复用成“trim 任意字符”。

练习题 B（更难一点，更像“小工程”）

题意： 输入多行文本（直到 EOF），对每一行： 1) 去掉首尾空格
2) 如果去掉后为空行，就跳过不输出
3) 否则输出：`行号:内容(长度)`
行号从 1 开始，按原始输入行编号。

升级点： - 多行处理（循环读到 EOF） - 需要把“trim”写成独立函数 - 要处理“跳过空行”的逻辑，结构更工程化

参考 C++ 框架：

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

string trim(const string& s) {
    int n = (int)s.size();
    int l = 0, r = n - 1;
    while (l < n && s[l] == ' ') l++;
    while (r >= 0 && s[r] == ' ') r--;
    if (l > r) return "";
    return s.substr(l, r - l + 1);
}

int main() {
```

```
    string line;
    int lineNo = 0;
    while (getline(cin, line)) {
        lineNo++;
        string t = trim(line);
        if (t.empty()) continue;
        cout << lineNo << ":" << t << "(" << t.size() << ")\n";
    }
    return 0;
}
```

如果你愿意的话，你可以把你想用的版本（C 风格原地 / C++ string）告诉我～我可以再帮你把代码写得更“OJ 友好”、更不容易踩坑一点点。