**Code gốc (khó đọc, tên biến rối)**

#include <stdio.h>

int f(int a){

int j,k,h;

if(a<2)return 0;

for(j=2;j\*j<=a;j++){

if(a%j==0)return 0;

}

return 1;

}

int main(){

int i,n,s=0;

scanf("%d",&n);

for(i=1;i<=n;i++){

if(f(i)==1)s+=i;

}

printf("%d",s);

return 0;

}

**Prompt refactor**

“Hãy refactor đoạn code C dưới đây thành phiên bản ngắn gọn, dễ đọc, có chú thích.”

**Code refactor (phiên bản dễ hiểu, có chú thích)**

#include <stdio.h>

#include <stdbool.h>

// Hàm kiểm tra một số có phải là số nguyên tố hay không

bool isPrime(int num) {

if (num < 2) return false;

for (int i = 2; i \* i <= num; i++) {

if (num % i == 0) return false;

}

return true;

}

// Hàm chính: tính tổng các số nguyên tố từ 1 đến n

int main() {

int n;

printf("Nhap n: ");

scanf("%d", &n);

int sum = 0;

for (int i = 2; i <= n; i++) {

if (isPrime(i)) {

sum += i;

}

}

printf("Tong cac so nguyen to tu 1 den %d la: %d\n", n, sum);

return 0;

}

**So sánh và điểm cải thiện**

| **Tiêu chí** | **Code gốc** | **Code refactor** | **Điểm cải thiện** |
| --- | --- | --- | --- |
| Tên hàm/biến | f, j, k, h, s | isPrime, sum, num | Tên rõ nghĩa, dễ hiểu hơn |
| Kiểu dữ liệu logic | Dùng int | Dùng bool từ <stdbool.h> | Thể hiện đúng bản chất logic |
| Cấu trúc hàm | Gộp chung, không có chú thích | Có tách hàm, có chú thích | Dễ bảo trì, dễ đọc |
| Phạm vi biến | Có biến thừa (k, h) | Loại bỏ biến không dùng | Gọn, tối ưu |
| Giao diện nhập/xuất | Không có hướng dẫn người dùng | Có lời nhắc khi nhập | Thân thiện hơn |
| Phong cách code | Viết liền, không rõ ràng | Căn chỉnh, tuân thủ chuẩn code | Dễ theo dõi, chuyên nghiệp hơn |

**Kết luận**

Code sau khi refactor ngắn gọn, dễ hiểu và tuân thủ chuẩn lập trình C hơn.  
Việc đổi tên biến, thêm chú thích, và tổ chức lại cấu trúc giúp chương trình rõ ràng và dễ mở rộng trong tương lai.