**Prompt A (ngắn)**

Viết hàm C kiểm tra số nguyên tố, có chú thích từng bước, đầu vào là số nguyên dương n.

**Kết quả AI (phiên bản A)**

#include <stdio.h>

// Hàm kiểm tra số nguyên tố: trả về 1 nếu n là số nguyên tố, 0 ngược lại

int isPrime(int n) {

if (n < 2) return 0; // loại bỏ các số < 2

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

if (n % i == 0) return 0; // nếu có ước thì không phải số nguyên tố

}

return 1;

}

int main() {

int n;

scanf("%d", &n);

printf("%d\n", isPrime(n)); // in 1 nếu nguyên tố, 0 nếu không

return 0;

}

**Prompt B (dài, chi tiết)**

Viết chương trình C có hàm kiểm tra xem một số nguyên dương n có phải là số nguyên tố hay không.

Yêu cầu:

- Viết chú thích chi tiết từng bước (giải thích nguyên lý).

- Hàm nên tối ưu (kiểm tra đến căn bậc hai).

- Thêm kiểm tra đầu vào (n phải là số nguyên dương).

- In ra thông điệp dễ hiểu cho người dùng.

- Viết thêm ví dụ test trong phần comment.

**Kết quả AI (phiên bản B)**

#include <stdio.h>

#include <stdlib.h>

// Hàm kiểm tra số nguyên tố

// Nguyên lý: số n (>1) là nguyên tố nếu không có ước d (2 <= d <= sqrt(n))

// Trả về 1 nếu là nguyên tố, 0 nếu không

int isPrime(int n) {

if (n <= 1) return 0; // 0 và 1 không phải nguyên tố

if (n == 2) return 1; // 2 là nguyên tố

if (n % 2 == 0) return 0; // loại nhanh các số chẵn >2

for (int i = 3; i \* i <= n; i += 2) { // chỉ kiểm tra các số lẻ

if (n % i == 0) return 0;

}

return 1;

}

int main() {

int n;

printf("Nhap mot so nguyen duong n: ");

if (scanf("%d", &n) != 1) {

printf("Nhap khong hop le. Ket thuc.\n");

return 1;

}

if (n < 0) {

printf("Vui long nhap so nguyen DUONG.\n");

return 1;

}

if (isPrime(n))

printf("%d la so nguyen to.\n", n);

else

printf("%d khong phai la so nguyen to.\n", n);

/\* Ví dụ test:

n = 2 -> nguyên tố

n = 15 -> không nguyên tố

n = 17 -> nguyên tố

\*/

return 0;

}

**So sánh kết quả AI sinh code — điểm khác nhau chính**

1. **Phạm vi/độ đầy đủ**
   * Phiên bản A: chỉ cung cấp hàm + main() rất ngắn, in 1/0. Phù hợp nếu em chỉ muốn hàm tiện dùng trong mã khác.
   * Phiên bản B: chương trình hoàn chỉnh, có kiểm tra đầu vào, thông báo rõ ràng cho người dùng, kèm ví dụ test.
2. **Độ chi tiết của chú thích**
   * A: chú thích ngắn, mô tả trực tiếp chức năng.
   * B: chú thích chi tiết hơn, giải thích nguyên lý (tại sao kiểm tra đến sqrt(n), tối ưu hóa bằng loại trừ số chẵn).
3. **Tối ưu và an toàn**
   * A: dùng vòng lặp bước 1..sqrt(n) (thực ra dùng i\* i <= n) — đủ nhưng chưa tối ưu cho mọi trường hợp.
   * B: thêm tối ưu (loại các số chẵn trước, lặp qua các lẻ từ 3), kiểm tra đầu vào và xử lý lỗi nhập, nên an toàn hơn.
4. **Tính thân thiện với người học**
   * A: phù hợp cho người biết cơ bản, muốn hàm ngắn.
   * B: phù hợp cho người mới vì có hướng dẫn, thông báo rõ lỗi nhập, ví dụ test.
5. **Khả năng tái sử dụng**
   * A: hàm nhỏ gọn, dễ nhúng vào module khác.
   * B: cũng có hàm tái sử dụng, nhưng phần main() và thông báo hướng đến chương trình độc lập.