Mô tả: Chọn hàm kiểm tra số nguyên tố. Viết ít nhất 2 phiên bản prompt (ngắn và dài), sau đó so sánh mã C do AI sinh ra và chỉ ra điểm khác biệt. (Đây là phiên bản 'tạo lại nhầm đề' — đề đã được sửa cho rõ ràng).

## Prompt ngắn

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

#include <stdio.h>  
#include <stdbool.h>  
  
// Hàm kiểm tra số nguyên tố  
bool laSoNguyenTo(int n) {  
 // Số nhỏ hơn 2 không phải số nguyên tố  
 if (n < 2) return false;  
 // Kiểm tra ước từ 2 đến căn bậc hai của n  
 for (int i = 2; i \* i <= n; i++) {  
 if (n % i == 0) return false; // Nếu tìm thấy ước, không phải số nguyên tố  
 }  
 return true; // Không tìm thấy ước → là số nguyên tố  
}  
  
int main() {  
 int n;  
 printf("Nhập số nguyên dương n: ");  
 scanf("%d", &n);  
 if (laSoNguyenTo(n))  
 printf("%d là số nguyên tố.\n", n);  
 else  
 printf("%d không phải số nguyên tố.\n", n);  
 return 0;  
}

## Prompt dài (chi tiết hơn)

Hãy viết một chương trình C hoàn chỉnh gồm một hàm kiểm tra xem một số nguyên dương n có phải là số nguyên tố không. Trong code hãy thêm chú thích chi tiết cho từng bước: giải thích tại sao kiểm tra n < 2, lý do chỉ cần kiểm tra đến căn bậc hai, và mô tả ý nghĩa của việc trả về true/false. Hàm phải nhận một tham số int n và trả về bool.

#include <stdio.h>  
#include <stdbool.h>  
  
// Hàm kiểm tra xem n có phải số nguyên tố hay không  
// - Trả về true nếu n là số nguyên tố, false nếu không phải  
bool kiemTraSoNguyenTo(int n) {  
 // Lý do: các số nhỏ hơn 2 (0 và 1) không thỏa điều kiện số nguyên tố  
 if (n < 2) {  
 return false;  
 }  
  
 // Lý do kiểm tra đến căn bậc hai: nếu n có ước a > sqrt(n) thì sẽ có ước b < sqrt(n)  
 // nên chỉ cần kiểm tra các ước <= sqrt(n)  
 for (int i = 2; i \* i <= n; i++) {  
 // Nếu tìm thấy ước i sao cho n % i == 0 thì n không phải số nguyên tố  
 if (n % i == 0) {  
 return false; // trả về false tức là không phải số nguyên tố  
 }  
 }  
  
 // Nếu không tìm thấy ước nào → n là số nguyên tố  
 return true;  
}  
  
int main() {  
 int n;  
 printf("Nhập một số nguyên dương: ");  
 scanf("%d", &n);  
  
 if (kiemTraSoNguyenTo(n)) {  
 printf("%d là số nguyên tố.\n", n);  
 } else {  
 printf("%d không phải số nguyên tố.\n", n);  
 }  
 return 0;  
}

So sánh kết quả mã sinh bởi AI

Phân tích các điểm khác biệt chính giữa hai mã trên:

|  |  |  |
| --- | --- | --- |
| Tiêu chí | Phiên bản ngắn | Phiên bản dài (chi tiết) |
| Tên hàm | laSoNguyenTo() | kiemTraSoNguyenTo() — tên mô tả rõ hơn |
| Chú thích | Ngắn, trực tiếp | Chi tiết, giải thích lý do ở mỗi bước |
| Độ rõ ràng cho người mới | Đủ cho người đã biết cơ bản | Thích hợp cho người mới học, lý giải nền tảng |
| Cấu trúc/logic | Tương đương — đều dùng kiểm tra đến i\*i <= n | Tương đương — logic giống phiên bản ngắn nhưng chú thích đầy đủ hơn |
| Tính tái sử dụng | Hàm đơn giản, có thể dùng lại | Hàm có tên rõ ràng và chú thích giúp tái sử dụng dễ hơn |
| Phong cách in/nhập | Ngắn gọn | Nhập/ in tương tự, nhưng lời nhắc dài dẫn đến mô tả rõ ràng hơn trong code |

## Kết luận

– Prompt ngắn cho kết quả nhanh, gọn. Prompt dài cho code có chú thích chi tiết hơn, phù hợp cho người mới.  
– Khi yêu cầu rõ ràng (ví dụ: 'giải thích lý do'), AI sẽ thêm phần chú thích giải thích logic và cơ sở toán học.  
– Lựa chọn prompt tuỳ mục tiêu: học nhanh (prompt ngắn) hay học sâu (prompt dài).