**1. Số Chính Phương**

Số chính phương là số bình phương của một số nguyên, ví dụ như các số 0, 1, 4, 25, 49, 100..

Phương pháp :

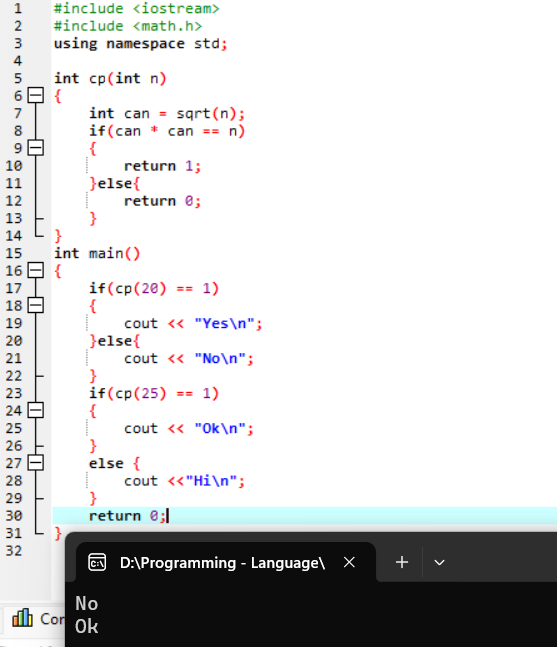
1. Tính căn bậc 2 của N và lưu căn vào số nguyên với mục đích loại bỏ phần thập phân của căn
2. So sánh tích của căn với chính nó với số N, nếu bằng nhau thì N là số chính phương

Giải thích : Khi N là số chính phương thì tính căn bậc 2 của N và lưu vào số nguyên sẽ không làm thay đổi giá trị của biến căn vì phần thập phần bằng 0, ngược lại N không phải là số chính phương thì khi tính căn bậc 2 nhưng lưu vào số nguyên sẽ làm mất phần thập phân có ý nghĩa dẫn đến thay đổi giá trị của biến căn N và vì thế khi ta nhân ngược lại căn với căn sẽ không ra được N.

Ví dụ :

N = 20, can = √N = 4.472 = 4, can \* can = 16 != 20 nên 20 không phải là số chính phương

N = 25, can = √N = 5.000 = 5, can \* can = 25 == 25 nên 25 là số chính phương



**Giải thích:**

- Dòng 5: Hàm nhận số nguyên n và kiểm tra xem n phải là số chính phương không.

- Dòng 7: Tính căn bậc 2 của n và lưu vào biến can.

- Dòng 8: Kiểm tra nếu bình phương của can bằng n, tức là n là số chính phương (ví dụ: 25 là số chính phương vì căn bậc hai của 25 là 5, và 5 \* 5 = 25).

- Dòng 17: if(cp(20) == 1): Gọi hàm cp với giá trị 20. Vì 20 không phải là số chính phương (sqrt(20) không phải là số nguyên), hàm cp sẽ trả về 0, nên in ra No.

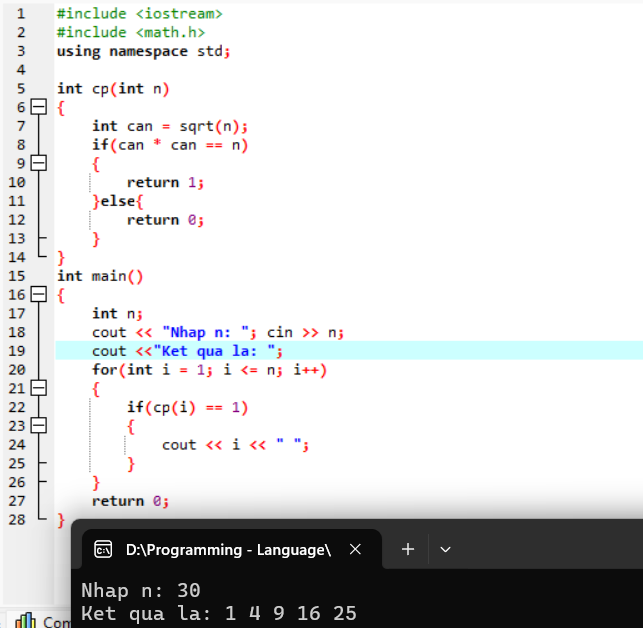
- Dòng 23: if(cp(25) == 1): Gọi hàm cp với giá trị 25. Vì 25 là số chính phương (sqrt(25) là 5), hàm cp trả về 1, nên in ra Ok.

**2. Bài Tập Số Chính Phương**

Bài 1. In ra các số chính phương từ 1 tới N

Đếm các số chính phương từ 1 tới N các bạn làm tương tự

Cách 1. Thuật toán ngây thơ



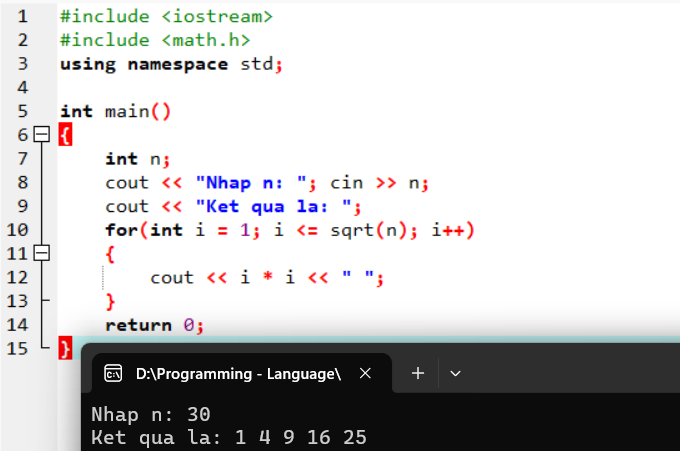
Giải thích:

- Dòng 22: if(cp(i) == 1): Kiểm tra nếu i là số chính phương (tức cp(i) trả về 1).

Cách 2 : Thuật toán tối ưu

Phân tích : Để liệt kê các số chính phương từ 1 tới n ta gọi số chính phương đó có dạng i2 trong đó i là một số nguyên

Từ đó ta có bất phương trình 1 ≤ i2 ≤ n => 1 ≤ i ≤ √n, vậy để in ra các số chính phương từ 1 tới n bạn chỉ cần duyệt các số i từ 1 tới √n và in ra bình phương của i là được.



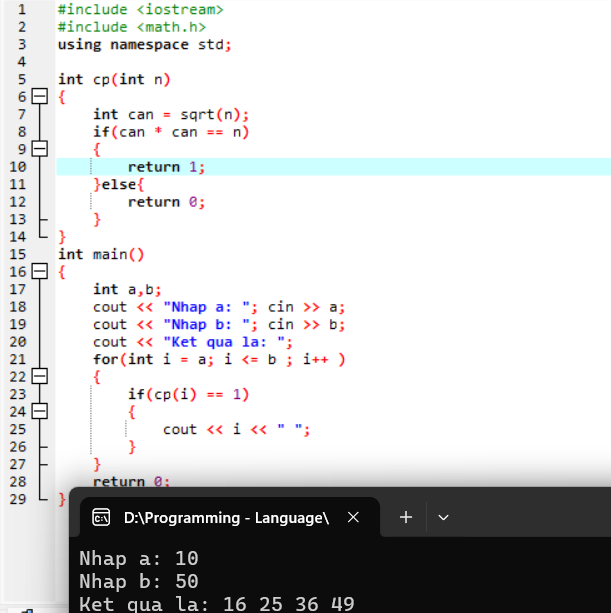
Giải thích:

- Dòng 10: for(int i = 1; i <= sqrt(n); i++): Vòng lặp chạy từ 1 đến sqrt(n), duyệt qua các giá trị mà khi bình phương lên vẫn nhỏ hơn hoặc bằng n.

- Dòng 12: Tính i \* i (bình phương của i) và in ra.

Bài 2. Liệt kê (đếm) các số chính phương từ a tới b.

Cách 1. Thuật toán ngây thơ

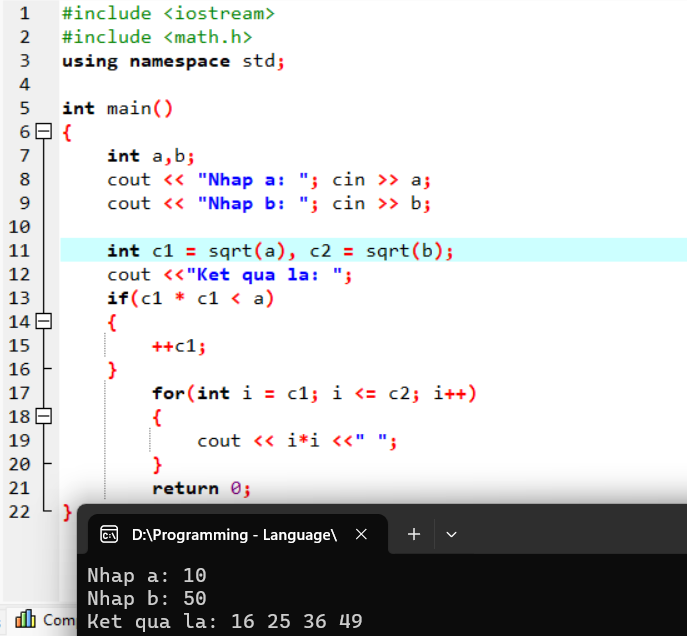


Cách 2 : Thuật toán tối ưu

Phân tích : Để liệt kê các số chính phương từ a tới b ta gọi số chính phương đó có dạng i2 trong đó i là một số nguyên

Từ đó ta có bất phương trình a ≤ i2 ≤ b => √a ≤ i ≤ √b, vậy để in ra các số chính phương từ a tới b bạn chỉ cần duyệt các số i từ √a tới √b và in ra bình phương của i là được.

Bạn cần chú ý cận √a vì nó bị làm tròn xuống nên có thể dẫn đến kết quả bị sai.



Giải thích:

- Dòng 11: int c1 = sqrt(a), c2 = sqrt(b);: Tính căn bậc hai của a và b, lưu kết quả vào c1 và c2.

- Dòng 13 và 15: if(c1 \* c1 < a): Kiểm tra nếu c1 bình phương nhỏ hơn a, tức là c1 chưa đủ lớn để tạo ra số chính phương >= a. Nếu đúng, tăng c1 lên 1 để bắt đầu từ số chính phương đầu tiên nằm trong khoảng từ a trở lên.

- Dòng 17 và 19:

+ for(int i = c1; i <= c2; i++): Vòng lặp từ c1 đến c2, duyệt qua các giá trị i mà khi bình phương sẽ tạo ra các số chính phương trong khoảng [a, b].

Giả sử người dùng nhập a = 10 và b = 50. Các bước diễn ra như sau:

1. **Tính c1 và c2:**
   * c1 = sqrt(10) xấp xỉ 3.16, nên c1 = 3. Do 3 \* 3 = 9 nhỏ hơn 10, c1 được tăng thêm 1 để trở thành 4.
   * c2 = sqrt(50) xấp xỉ 7.07, nên c2 = 7.
2. **Chạy vòng for:**
   * Lần 1: i = 4
     + 4 \* 4 = 16, in ra 16.
   * Lần 2: i = 5
     + 5 \* 5 = 25, in ra 25.
   * Lần 3: i = 6
     + 6 \* 6 = 36, in ra 36.
   * Lần 4: i = 7
     + 7 \* 7 = 49, in ra 49.
   * Khi i tăng lên 8, vòng lặp kết thúc vì 8 > 7.

+ cout << i \* i << " ";: In ra bình phương của i.