**1. Lý Thuyết Đồng Dư**

Trong lập trình có những bài toán có kết quả rất lớn và tăng nhanh trong quá trình tính toán, ví dụ như giai thừa, số Fibonacci, Lũy thừa.

Tốc độ của ngôn ngữ C/C++ hoàn toàn có thể tính được 1000000 giai thừa trong chưa tới một giây, tuy nhiên việc lưu trữ kết quả của bài toán lại rất khó khăn vì C/C++ không hỗ trợ xử lý số nguyên lớn. Ngay cả bạn sử dụng Python hay Java để xử lý số nguyên lớn thì việc phải tính toán các giá trị này sẽ rất mất thời gian do phải xử lý các số rất lớn.

Cách dễ dàng hơn là chia dư kết quả cho một số định trước

**Công Thức Đồng Dư :**

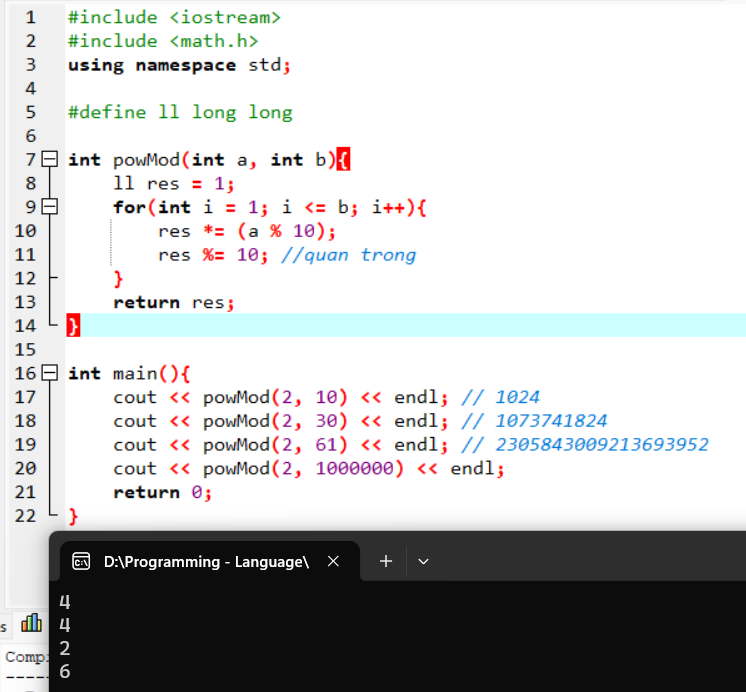
* (A + B) % C = ((A % C) + (B % C)) % C
* (A - B) % C = ((A % C) - (B % C)) % C
* (A \* B) % C = ((A % C) \* (B % C)) % C
* (A / B) % C = ((A % C) \* (B-1 % C)) % C

4 công thức trên bạn có thể áp dụng cho 2 số hoặc N số đều được, trong đó 3 công thức đầu dễ hiểu và có thể thực hành ngay, công thức thứ 4 liên quan tới Nghịch đảo mô đun nên các bạn có thể chuyển tới mục này ở bài học bên dưới để đọc trước.

Ví dụ (13 + 19) % 5 = ((13 % 5) + (19 % 5)) % 5 = (3 + 4) % 5 = 2

Lý do cần dùng công thức đồng dư là vế trái của 4 công thức trên không phải lúc nào cũng có thể tính được, nó có thể tràn giới hạn của số long long trong C/C++

**Ví dụ 1** : Tính ab chia dư cho 10, (1 ≤ a, b ≤106)



**Nhận xét :**

Để hàm powMod hoạt động đúng bạn cần bổ sung thêm 2 phần :

* Thay vì nhân a vào kết quả thì chỉ cần nhân (a % 10), phần này là dựa theo lý thuyết
* Chia dư kết quả cho 10 sau mỗi lần nhân, phần này là rất quan trọng vì trong quá trình nhân thì biến res có thể bị tràn dữ liệu long long dẫn tới kết quả sai.

Ở đây chắc bạn sẽ thắc mắc về phần thứ 2 khi ta phải chia dư kết quả cho 10 sau mỗi lần nhân vì nó không có giống trong mục lý thuyết cung cấp. Trong mục lý thuyết khi phát triển công thức thứ 3 cho N số thì đáng lẽ ta phải nhân N số dư của các số này với nhau thôi chứ không được chia dư trong quá trình nhân phải không?

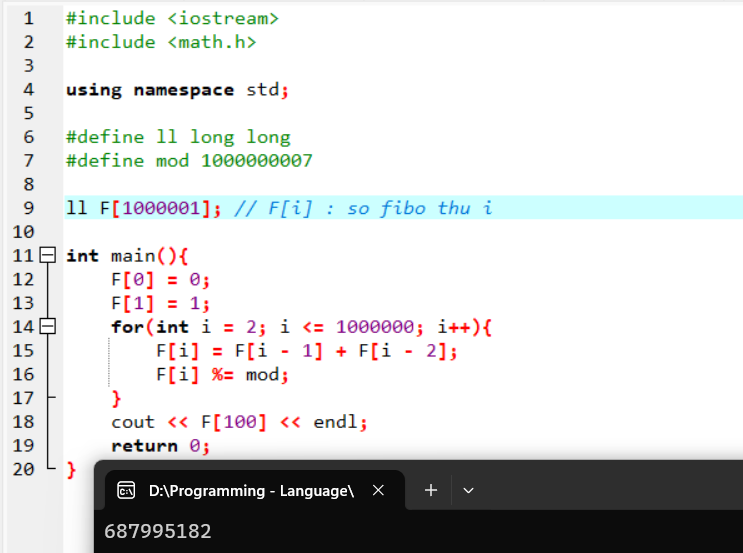
Ví dụ (13 \* 11 \* 8 \* 7) % 5 thì nếu chỉ nhân các số dư với nhau và cuối cùng mới đem chia dư cho 5 thì kết quả sẽ là (3 \* 11 \* 3 \* 2) % 5 = 198 % 5 = 3.

Giả sử bạn nhân 3 số đầu với nhau rồi thay vì giữ nguyên thì đem chia cho 5 ta sẽ có kết quả là (99 \*  2) % 5 = ((99 % 5) \* 2) % 5 = 8 % 5 = 3, rõ ràng là việc chia dư kết quả trong quá trình tính toán không ảnh hưởng gì tới kết quả cuối cùng.

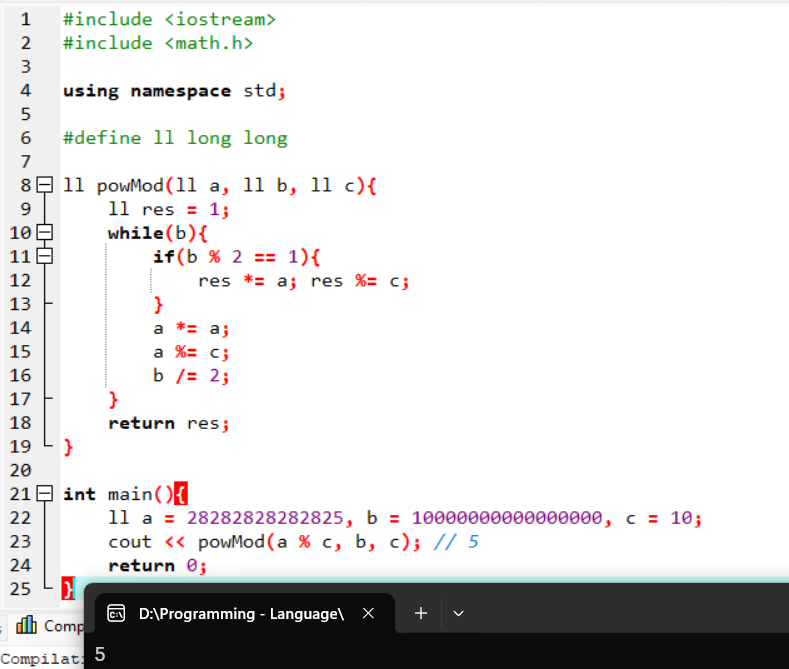
Hoặc bạn có thể coi cụm (13 \* 11 \* 8) là A, 7 là B và 5 là C trong công thức thứ 3 thì rõ ràng là là bạn chỉ cần tính số dư của A sau đó nhân với số dư của B là được. Ngoài ra cụm A cũng có thể chia nhỏ thành (13 \*  11) và 8, tiếp tục chia nhỏ ra thì bạn thấy việc chia dư trong quá trình tính toán là hoàn toàn đúng đắn.

**2. Một Số Bài Toán Lập Trình Với Đồng Dư**

**Bài toán 1**: In ra số Fibonacci thứ N (0 ≤ N ≤ 106) sau khi chia dư cho 109 + 7



**Bài toán 2**: Tính ab chia dư cho c ( 1 ≤ a, b ≤ 1016), ( 1 ≤ c ≤ 109)



**Bài toán 3** : Tính tổ hợp chập K của N chia dư cho 109 + 7 (0 ≤ K ≤ N ≤ 103)

