Stepenovanje brojeva ne postoji kao ugradjena. Morate sami napisati rutinu koja to radi.

Prvo što pada na pamet je n puta množenje broja sa samim sobom. Algoritam je složenosti O(n).

Res = 1; // trazimo a na n-ti (stepen)

  f**or (int i = 1; i <=** Stepen; i++)

**r**es = res \* a;

Ali to nije najbolje, tehnikom podeli i pokori, moguće je izvesti stepenovanje u vremenu O(log2 n):

Res = 1; // trazimo a na n-ti (stepen)

**while** (Stepen){

**if** (Stepen%2==1) res = res\*a; // %x

      stepen = stepen / 2;

      a = a\*a // %x

  } // rezultat je u promenljivoj RES }

S obzirom da vrednost stepenovanog broja brzo raste, često se ova operacija traži u modularnoj aritmetici. Zbog toga je onih %x gde je x broj po kom modulu se traži rezultat.

**REKURZIVNO:**

Objašnjanje postupka:

Recimo da je potrebno izračunati 320 odnosno 321

| 3\*3\*3\*3\*3\*3\*3\*3\*3\*3\*3\*3\*3\*3\*3\*3\*3\*3\*3\*3 =  (3\*3\*3\*3\*3\*3\*3\*3\*3\*3) \* (3\*3\*3\*3\*3\*3\*3\*3\*3\*3) =  (3\*3\*3\*3\*3\*3\*3\*3\*3\*3)2 | 320 = 310 \* 310 = (310)2 {u samo jednom koraku, sveli smo problem na duplo manji } |
| --- | --- |
| 3\*3\*3\*3\*3\*3\*3\*3\*3\*3\*3\*3\*3\*3\*3\*3\*3\*3\*3\*3\*3 =  (3\*3\*3\*3\*3\*3\*3\*3\*3\*3) \* (3\*3\*3\*3\*3\*3\*3\*3\*3\*3) \* 3 =  (3\*3\*3\*3\*3\*3\*3\*3\*3\*3)2 \* 3 | 321 = 310 \* 310 \* 3 = (310)2 \* 3{u samo jednom koraku, sveli smo problem na duplo manji } |

U oba slučaja problem smo sveli na 310. Sada ako dalje nastavimo tako, problem svodimo na 35 pa na 32. Sve u svemu očas posla, izvrši se stepenovanje. Ovo naročito dolazi do izražaja kod velikih stepena. Recimo a1000000 iz 20 koraka se izračunava stepen na ovaj način. Običnim množenjem potrebno je 1000000 koraka.

Dole je Rekurzivna implementacija.

int M = ?????; // broj sa kojim kad delimo traži se ostatak

int A, B;

int NaStepen(int Osnova, int Stepen) // rekurzivno

long long Pom = 1LL;

If (Stepen) { // Ako je stepen nula, rešenje će ostati 1

Pom = NaStepen(Osnova, Stepen / 2); // %M;

Pom = Pom \* Pom; // %M;

if (Stepen%2==1) Pom := Pom \* Osnova; // %M;

}

return Pom;

};

int main() {

cin >> A >> B;

cout << NaStepen(A, B) << endl;

return 0;

}

[Video lekcija sa dizanjem broja na N-ti stepen](http://ocw.mit.edu/OcwWeb/Electrical-Engineering-and-Computer-Science/6-046JFall-2005/VideoLectures/detail/embed03.htm) i njegovim Big O objasnjenjem .(od 13. min do 17 min)

Zadaci u kojim se ovo koristi

1. Z-warmup (z-trening)
2. Z-product (z-trening)
3. SuperClimber (z-trening)
4. Surovi matematičar: “Cruel1” Zadatak 11: FEB09 USACO elite BRONZE competition.