Testul de primalitate – verificarea unui număr natural dacă este prim

Pseudocod verificarea unui număr natural dacă este prim:

Pas 1. Presupunem ca *n* este **prim** (variabila prim primește valoarea 1)

Pas 2. Daca *n*<2 atunci *n* nu este prim si variabila **prim** primește valoarea 0

Pas 3. Parcurgem mulțimea divizorilor proprii posibili si verificam divizibilitatea lui *n* cu aceștia.

Daca exista un divizor propriu, atunci *n* nu este prim

variabila **prim** primește valoarea 0.

Pas 4. Prelucram numărul *n* în funcție de valoarea variabilei prim.

Implementare C++	Explicații

`prim=1;`
`if(n<2)`
`  prim=0;`
`for (d=2; d*d<=n; d++)`
`  if(n%d==0)`
`  {`
`  prim=0;`
`  break;`
`  }`
`if(prim==1)`
`  //prelucrare nr prim`
	Reducerea numărului de operații (micșorarea timpului de execuție in cazul numerelor mari si, eventual prime):
	Utilizarea instrucțiunii break determina ieșirea forțată din bucla for la întâlnirea unui divizor
	Parcurgerea divizorilor pana la cel mai mare pătrat perfect mai mic egal cu n

Exemplu

Testarea primalității lui n = 1.000.003 (număr prim) prin:

- parcurgerea divizorilor pana la n/2
- se vor face 500.000 de comparări

parcurgerea divizorilor pana când d*d>n

- se vor face aproximativ 1.000 de comparări (10002=1.000.000) deci, de 500 de ori mai puține

Si asta doar pentru un număr!!

Dacă ați avea de verificat 100.000 de numere mai mici sau egale cu 1.000.000.000?

Prelucrarea divizorilor primi și a exponenților acestora (din descompunerea în factori primi)

Pseudocod descompunerea în factori primi a lui n:

Pas 1. Pornim cu *d*=2 (cel mai mic factor prim posibil)

Pas 2. Cât timp *n* nu a ajuns la 1 sau nu e număr **prim**, împărțim *n* la *d* cât timp e posibil si numărăm împărțirile făcute (determinăm astfel exponentul factorului prim)

Pas 3. Trecem la următorul divizor (îl creștem pe d cu 1) și reluăm Pas 2.

Implementare C++

```
d=2;
while (d*d<=n){
      e=0;
      while (n%d==0){
            n=n/d;
            e++;
      }
      if(e>0) //prelucrare d si e
      d++;
}
if (n>1) {
        n este ultimul factor prim
        si apare la puterea 1 }
```

Calcularea numărului de divizori ai lui n, n>0

P1. Prin parcurgerea divizorilor proprii și numărarea lor, la care se adaugă 2 divizori (1 și n, dacă n>1) sau 1 (dacă n=1)

```
nrdiv=0;
for (d=2; d<=n/2; d++)
     if(n%d==0) nrdiv++;
if(n==1)
     nrdiv++;
     else
     nrdiv=nrdiv+2;
```

OBS: pentru n<1.000.000.000, numărul de operații de verificare a divizibilității este foarte mare

P2. Prin formula lui Euler bazată pe descompunerea în factori primi a lui n

Dacă în urma descompunerii în factori primi se obține produsul $d_1e_1*d_2e_2*...d_ke_k$ unde d_1, d_2, ...d_k sunt factorii primi iar e_1, e_2,... exponenții acestora, numărul divizorilor lui *n* se calculează după formula:

$$nrdiv = (1+e_1)* (1+e_2)*...* (1+e_k)$$

Exemplu

$n = 72 = 2^3*3^2$

nrdiv = (1+3)*(1+2) = 4*3 = 12

divizorii sunt: 1 2 3 4 6 8 9 12 18 24 36 72

$n = 87120 = 2^4*3^2*5^1*11^2$

nrdiv = (1+4)*(1+2)*(1+1)*(1+2) = 5*3*2*3 = 90

Implementare C++

```
nrdiv=1; d=2;
while (d*d<=n)
{
      e=0;
      while (n%d==0)
      {
            n=n/d;
            e++;
       }
      if(e>0)
       nrdiv=nrdiv*(1+e);
       d++;
}
if(n>1)
      nrdiv=nrdiv*(1+1);
```

Probleme

1. Se citește din fișierul *perechi.in* un număr natural n. Sa se afișeze în fișierul *perechi.out* toate perechile de numere naturale (x,y) până la n (1<x<y<=n), pentru care x este divizorul lui y , numerele x si y au aceeași cifra a unităților. Exemplu

perechi.in	*perechi.out*
30	(2,12),(2,22),(4,24),(5,15),(5,25), (10,20),(10,30)

2. Se citește de la tastură un număr natural nenul n de cel mult 4 cifre. Să se verifice dacă suma divizorilor lui n este un număr prim. Dacă da, se va afișa suma divizorilor săi, altfel se va scrie numărul acestora.

Date de intrare	Date de ieșire
n=30	8 *(1+2+3+5+6+10+15+30=72 nu e prim deci se va afisa 8, numarul divizorilor)*

3. Se citesc din fișierul puteri.in n și k (1<n<10.000, 1<k<10) apoi un șir de n valori naturale nenule de cel mult 9 cifre, separate prin câte un spațiu. Să se scrie în puteri.out, numerele din șir care sunt puteri ale lui k. Rezultatele se vor scrie pe prima linie, separate prin spațiu. Dacă nu există în șirul dat nicio putere a lui k se va scrie 0.

puteri.in	*puteri.out*
n=6, k=2 14 8 512 24 128 1024	8 512 128 1024

Prelucrarea divizorilor unui număr natural nenul

Orice n, număr natural nenul poate avea **divizori improprii, proprii sau primi.**

Divizorii IMPROPRII: 1 și n (dacă n ≠ 1).

Observăm că 1 este singurul număr care are doar un divizor.

Divizori PROPRII: numerele: 2,3,4...,n/2 (n%d==0)

Divizori PRIMI: numerele prime: 2,3,...,n/2,n (n%d==0)

Prelucrarea divizorilor proprii

Parcurgerea crescătoare sau descrescătoare a mulțimii divizorilor proprii posibili și verificarea divizibilității lui *n* cu fiecare dintre ei

```
for (d=2; d<=n/2; d++)
    if(n%d==0)
    //prelucreaza d

sau
for (d=n/2; d>=2; d--)
  if(n%d==0)       // prelucreaza d
```

Orice număr natural pătrat perfect are un număr impar de divizori