Euler totient ϕ(n)-phi(n)

Phi(n) este ca su un contor. Numara toate numerele care sunt relativ prime cu n. Un numar a este relativ prim cu n daca cmmdc(a, n) = 1.

Teorema lui Euler

$a^{\phi} \equiv 1 \bmod n$

Aplicatiile lui sunt in algebra abstracta si in algoritmul RSA utilizat in criptarea de securitate pe internet.

Exemplu :

Phi(8) = {**1**, 2 , **3** , 4 , **5**, 6 , **7**, 8}

Cmmdc(1,8) = 1 1 realtiv prim la 8

Cmmdc(3,8) = 1 3 relativ prim la 8

Cmmdc(5,8) = 1 5 relativ prim la 8

Cmmdc(7,8) = 1 7 relativ rpim la 8

Cate numere sunt relativ prime pornind de la 1 pana la < ca n ? , in acest caz n = 8

P(8) = 4 pentru sa sunt 4 numere relativ prime cu 8 pe intervalul 1….7

Exemplu :

Phi(Numar_Prim) = Numar_Prim – 1

Relatie :

Phi(A*B) = Phi(A)*Phi(B)

Daca N = NumarPrim1 * NumarPrim2

Phi(N) = Phi(NumarPrim1) * Phi(NumarPrim2)

= (NumarPrim1 – 1) * (NumarPrim2 -1)

Propietate utizata in algoritmul **RSA** : **Phi(pq) = (p-1)*(q-1)**

Pasi algoritm

1) Initializam rezultatul cu n

2) Luam in considerare fiecare numar 'p' (unde 'p' variaza de la 2 la sqrt(n)).

Daca p il diivide pe n, atunci fa urmatoarele

a) Scapati de toti multiplii de p de la 1 la n [toti multiplii de p vor avea cmmdc > 1]

b) Actualizeaza n impartindul in mod repetat la p.

3) Daca valoarea lui n redus este > 1, atunci scapa de toti multiplii de n din rezultat

1	Functia PHI(n):	
2	rezultat <- n	Initializeaza rezultatul cu n
3	pentru p<-2 pana la p*p	Luati in considerare toti factorii primi ai lui n si scapati de multiplii din rezultat
4	daca n % p = 0 :	Verifica daca p este factor prim
5	cat_timp n % p :	Daca este factor prim actualizeaza n si rezultat
6	n <- n / p	
7	rezultat <- rezultat – rezultat / p	
8	daca n > 1 :	Daca n are un factor prim mai mare decat sqrt(n) (Poate exista cel mult un singur astfel de factor)
9	rezultat <- rezultat – rezultat / p	
10	returneaza rezultat	

Algoritm implementat in c++

```
#include <iostream>
using namespace std;
class Functia_PHI{
private:
   int n;
public:
   int PHI(int n);
   void Afisare_PHI(int numar_functii);
};
int Functia_PHI::PHI(int n){
```

```
    int rezultat = n;
    for(int p = 2 ; p*p < n; p++){
        if(n % p == 0){
            while(n % p){
                n = n/p;
            }
         rezultat = rezultat - rezultat/p;
        }
    }
    if(n > 1){
        rezultat = rezultat - rezultat/n;
    }

return rezultat;
}

void Functia_PHI::Afisare_PHI(int numar_functii){
    for (int i = 1; i <= numar_functii; i++){
        cout<<"phi["<<i<<"]="<<PHI(i)<<endl;
    }
}
int main(){
Functia_PHI obj;
obj.Afisare_PHI(10);// PHI(1).....numar_functii_de_phi
    return 0;
}
```