CURS 06 – FP

Algoritmi numerici fundamentali

3. Testarea primalității unui număr:

Un număr este *prim* dacă se divide doar cu 1 și el însuși.
Un număr este *prim* dacă nu are divizori proprii (diferiți de 1 și el însuși).

Un număr care nu este prim se numește *număr compus*.

```
#include <stdio.h>

int main()
{
    int n, d;

    printf("n = ");
    scanf("%d", &n);

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

    if(d == n/2 + 1)
        printf("Numarul %d este prim!", n);
    else
        printf("Numarul %d este compus!", n);

    return 0;
}
```

Variantă:

```
#include <stdio.h>
int main()
{
    int n, d, prim;

    printf("n = ");
    scanf("%d", &n);

    //presupunem ca numarul n este prim
    prim = 1;
    for(d = 2; d <= n/2; d++)
        if(n % d == 0)
```

```
        {
            //am gasit un divizor propriu al numarului n,
            //deci presupunerea devine falsa
            prim = 0;
            break;
        }

    if(prim == 1)
        printf("Numarul %d este prim!", n);
    else
        printf("Numarul %d este compus!", n);

    return 0;
}
```

Observație: Toți divizorii proprii ai unui număr natural n sunt cuprinși între 2 și $\left[\frac{n}{2}\right]$.

Exemplu: $n = 15 \Rightarrow \mathcal{D}_{15} = \{1, 3, 5, 15\} \Rightarrow$ toți divizorii proprii, adică 3 și 5, sunt mai mici sau egali decât $[15/2] = 7$.

Teoremă: Dacă n este compus, atunci el are cel puțin un divizor propriu mai mic sau egal decât $\sqrt{n}$.

Exemplu: $n = 15 \Rightarrow d_1 = 3 < \sqrt{15} \approx 3.87$, dar $d_2 = 5 > \sqrt{15}$!

Demonstrație:
Pp. prin absurd faptul că n este compus, dar nu are niciun divizor propriu mai mic sau egal decât $\sqrt{n}$ => $n = d_1 * d_2$ și $d_1, d_2 > \sqrt{n}$ => $n = d_1 * d_2 > \sqrt{n} * \sqrt{n} = n \Rightarrow n > n$ (imposibil!!!).

```
#include <stdio.h>

int main()
{
    int n, d;

    printf("n = ");
    scanf("%d", &n);

    //d * d <= n este echivalenta cu d <= sqrt(n),
    //dar mult mai rapida!!!
    for(d = 2; d * d <= n; d++)
```

```
        if(n % d == 0)
            break;

    if(d * d > n)
        printf("Numarul %d este prim!", n);
    else
        printf("Numarul %d este compus!", n);

    return 0;
}
```

Variantă optimizată (testăm doar posibilii divizori impari):

```
#include <stdio.h>

int main()
{
    int n, d;

    printf("n = ");
    scanf("%d", &n);

    if(n < 2)
    {
        printf("Numarul %d nu este prim (primul numar prim este 2)!\n", n);
        return 0;
    }

    if(n == 2)
    {
        printf("Numarul 2 este prim!\n");
        return 0;
    }

    if(n % 2 == 0)
    {
        printf("Numarul %d este compus!\n", n);
        return 0;
    }

    //numarul n este impar si n >= 3, deci n
    //poate sa aiba doar divizori proprii impari
    for(d = 3; d * d <= n; d += 2)
        if(n % d == 0)
            break;


    if(d * d > n)
        printf("Numarul %d este prim!", n);
    else
        printf("Numarul %d este compus!", n);
```

```
    return 0;
}
```