

Algoritmi elementari

I. Algoritmi care prelucreaza cifrele unui numar

- Spargerea in cifre a numarului n si prelucrarea cifrelor de la dreapta la stanga

```
while (n) {
    // Extrager ultima cifra
    int lastDigit = n % 10;

    // Eliminare ultima cifra din numar
    n = n / 10;
}
```

- Spargerea in cifre a numarului n si prelucrarea cifrelor de la stanga la dreapta

```
int positions = 1;
// Aflam numarul de cifre din numarul initial
while ((positions * 10) <= n) {
    positions = positions * 10;
}

// Cat timp contorul ce contine numarul de cifre > 0
while (positions != 0) {
    // Extragem prima cifra
    int firstDigit = n / positions;

    // Taiem din n prima cifra
    n = n % p;

    // Taiem din p o pozitie, deoarece am extras numarul
    positions = positions / 10;
}
```

IMPORTANT: Spargerea in cifre a unui numar distrugere valoarea initiala a acestuia! Este necesara crearea unei copii a lui n, inainte de a se efectua extragerea cifrelor, asta in cazul in care mai avem nevoie de valoarea initiala.

Exemple de algoritmi ce folosesc spargerea pe cifre:

1. Construirea oglinditului si verificare n daca este palindrom

```
int isPalindrom(int n) {
    // Facem o copie lui N deoarece avem nevoie de valoarea initiala
```

```

int copieN = n;
// Initializam oglindit la 0
int oglindit = 0;

while (copieN > 0) {
    // Extragem ultima cifra din numar
    int ultimaCifra = copieN % 10;

    // Il adaugam la oglindit, de aceea avem nevoie
    // de inmultirea cu 10, deoarece ne mutam o pozitie mai la stanga
    (unitati, zeci, sute, etc)
    oglindit = oglindit * 10 + ultimaCifra;

    // Taiem ultima cifra din copie
    copieN = copieN / 10;
}

// Daca oglinditul este egal cu valoarea initiala
// inseamna ca avem un palindrom
return oglindit == n;
}

```

2. Media aritmetica a cifrelor nenule

```

float medieNumereNenule(int n) {
    // Declaram suma float ca sa evitam truncherea la int
    float suma = 0.0;
    int contorCifreNenule = 0;
    while (n > 0) {
        int ultimaCifra = n % 10;
        // Daca ultima cifra este mai mare ca 0
        // Actualizam suma si contorul de cifre nenule
        if (ultimaCifra > 0) {
            suma = suma + ultimaCifra;
            contorCifreNenule++;
        }

        n = n / 10;
    }
    // In cazul in care avem cel putin o cifra nenula
    // Calculam media
    if (contorCifreNenule > 0) {
        return suma / contorCifreNenule;
    }
    // Altfel inseamna ca avem un numar = 0
    else {
        return 0;
    }
}

```

3. Cifra maxima din n

```

int cifraMaxima(int n) {
    int maxim = 0;
    while (n > 0) {
        int lastDigit = n % 10;
        if (lastDigit > maxim) {
            maxim = lastDigit;
        }

        n = n / 10;
    }

    return maxim;
}

```

4. Eliminarea cifrelor impare din n

```

int eliminareCifreImpare(int n) {
    // Varibila pentru a tine numarul nou format prin eliminarea
    // cifrelor impare
    int numarNou = 0;
    // Variabila pentru a contoriza pozitia unitatilor, zecilor, etc.
    int pozitie = 1;
    while (n > 0) {
        int ultimaCifra = n % 10;
        // Daca ultima cifra este para, o extragem si o adaugam la numarul
        // nou
        if (ultimaCifra % 2 == 0) {
            // Numarul nou devine valoarea precedenta +
            // ultimaCifra inmultita cu pozitia
            numarNou = numarNou + ultimaCifra * pozitie;
            pozitie = pozitie * 10;
        }

        // Eliminare ultima cifra
        n = n / 10;
    }

    return numarNou;
}

```

- NOTA: Daca se doreste eliminarea cifrelor pare, tot ce trebuie sa facem va fi sa modificam conditia din:
 - `ultimaCifra %2 == 0` in `ultimaCifra %2 != 0`

5. Dublarea aparitiilor cifrelor pare din n

```

int dublareAparitiiCifrePare(int n) {
    // Variabila pentru a tine numarul nou. Practic recreem numarul, si
    // dublam cifra in cazul in care aceasta este para.
    int numarNou = 0;
    // Variabila pentru a contoriza pozitia unitatilor, zecilor, etc.
    int pozitie = 1;
    while (n > 0) {
        int ultimaCifra = n % 10;
        // Daca ultima cifra este para
        if (ultimaCifra % 2 == 0) {
            // Noul numar are valoare ultimaCifra * pozite + vechea
            // valoare
            numarNou = numarNou + ultimaCifra * pozitie;
            // Pozitia se muta mai la stanga
            pozitie = pozitie * 10;
        }
        // In cazul in care ultima cifra nu este para
        // Tot adaugam ultima cifra la numarul nou. Insa daca este para,
        // aceasta este adaugata de 2 ori.
        numarNou = numarNou + ultimaCifra * pozitie;

        // Incrementam pozitia
        pozitie = pozitie * 10;

        // Eliminam ultima cifra
        n = n / 10;
    }

    return numarNou;
}

```

6. Numararea cifrelor pare existente in n

```

int numarareCifrePare(int n) {
    // Initializare contor pentru cifre pare
    int contorCifrePare = 0;
    while (n > 0) {
        // Extragem ultima cifra para
        int ultimaCifra = n % 10;
        if (ultimaCifra % 2 == 0) {
            contorCifrePare++;
        }

        n = n / 10;
    }

    return contorCifrePare;
}

```

7. Cifra de control a unui numar n se obtine calculand suma cifrelor lui n apoi repetand procesul cu cifrele sumei obtinute anterior pana cand se obtine un numar format dintr-o singura cifra, numita **cifra de control**.

- De exemplu, pentru $n = 8579$ se obtin pe rand sumele:
 - $8 + 5 + 7 + 9 = 29$
 - $2 + 9 = 11$
 - $1 + 1 = 2$ (**cifra de control**)
- Algoritmul eficient ca timp de executie se bazeaza pe observatia ca cifra de control a unui numar respecta urmatoarea relatie:
 - $\text{cifraControl}(n) =$
 - 0, daca $n = 0$
 - 9, daca $n \% 9 = 0$ si $n \neq 0$
 - $n \% 9$, altfel

```
int cifraControl(int n) {
    while (n > 9) {
        int sumCifra = 0;
        while (n != 0) {
            int ultimaCifra = n % 10;
            sumCifra = sumCifra + ultimaCifra;
            n = n / 10;
        }
        n = sumCifra;
    }
    return n;
}
```

II. Divizibilitate. Algoritmi care prelucreaza divizorii proprii/improprii/primi ai unui numar.

- Cand vorbim de divizori, trebuie sa aducem in vedere urmatoarele 3 tipuri de divizori:
 1. Divizori improprii:
 - Prin divizori improprii intelegem setul cuprins din numerele 1 si n unde n este numarul pentru care dorim sa aflam divizorii
 2. Divizori proprii
 - Restul divizorilor care nu aparțin setului cu divizori improprii
 3. Divizori primi
 - Divizorii primi sunt divizorii lui n , care la randul lor au ca divizori doar pe 1 si pe ei insisi

Algoritmi fundamentali pentru divizori

1. Afisare divizori proprii ai lui n

```
void afisareDivizori(int n) {
    for (int i = 1; i <= n / 2; i++) {
```

```

    // Daca n se divide exact la i
    // inseamna ca i este divizor al lui `n`
    if (n % i == 0) {
        cout << i << endl;
    }
}
}

```

2. Afisare divizori primi n

```

void afisareDivizorPrimi(int n) {
    int divizor = 2;
    while (n > 1) {
        int contorDivizori = 0;
        while (n % divizor == 0) {
            n = n / divizor;
            contorDivizori += 1;
        }

        if (contorDivizori != 0) {
            cout << divizor << endl;
        }

        divizor = divizor + 1;
    }
}

```

DE RETINUT:

- N sunt numar impar de divizor daca este patrat perfect (de exemplu 36, etc.)
- N sunt exact 3 divizori daca este patrat perfect de numar prim (de exemplu 9, 25, etc.)
- N este perfect daca este egal cu suma divizorilor mai mici decat el insusi.
- A si B sunt numere prietene daca A este egal cu suma divizorilor lui B, mai mici decat B, iar B este egal cu suma divizorilor lui A, mai mici decat A.

III Primalitate. Testarea primalitatii unui numar n

- Un numar este prim, daca acesta are ca divizori doar pe 1 si pe el insusi.

```

int estePrim (int n) {
    int result = 1;
    // Cum un numar trebuie sa aiba pe 1 si pe el insusi ca divizori
    // daca n < 2 atunci rezultatul va fi fals
    if (n < 2) {
        result = 0;
    }
    else {
        // Iteram de la 2 pana la radical din n
    }
}

```

```

        for (int i = 2; i * i <= n; i++) {
            // Daca n se imparte exact la i
            // Ins-eamna ca nu este prim
            if (n % i == 0) {
                result = 0;
                break;
            }
        }

        return result;
    }
}

```

IV Cel mai mare divizor comun. Cel mai mare multiplu comun

1. Algoritmul lui Euclid bazat pe impartiri succesive. Varianta eficientă!

```

int cmmdc(int a, int b) {
    while (b != 0) {
        int rest = a % b;
        a = b;
        b = rest;
    }
    return a;
}

```

2. Algoritmul lui Euclid bazat pe scaderi succesive. Mai puțin eficient

```

int cmmdc(int a, int b) {
    while (b != a) {
        if (a > b) {
            a = a - b;
        }
        else {
            b = b - a;
        }
    }
    return a;
}

```

Cel mai mic multiplu comun se poate determina folosind formula de calcul:

- CMMMC(a, b) = $a * b / \text{cmmdc}(a, b)$
- Dacă numerele a și b sunt multiple între ele, $\text{cmmdc}(a, b) = 1$.

V. Siruri Recurrente. Sirul lui Fibonacci.

- Sirul lui Fibonacci este sirul cu proprietatea ca oricare numar, este suma ultimelor doua numere. Primele 2 numere din sir sunt **0** si **1**.
 - De exemplu, primele 6 numere sunt:
 - **0, 1, 1, 2, 3, 5**

1. Algoritm pentru generarea primilor n termeni din sirul Fibonacci

```
void afisareSirFibonacii(int n) {
    int f1 = 0;
    int f2 = 1;
    cout << f1 << " ";
    for (int i = 1; i < n; i++) {
        cout << f2 << " ";
        int next = f1 + f2;
        f1 = f2;
        f2 = next;
    }
}
```

2. Algoritm pentru generarea primilor n termeni pari din sirul Fibonacci

```
void afisareSirPareFibonacii(int n) {
    int f0 = 1;
    int f1 = 1;
    int f2;
    while (n > 0) {
        f2 = f0 + f1;
        if (f2 % 2 == 0) {
            cout << f2 << " ";
            n = n - 1;
        }
        f0 = f1;
        f1 = f2;
    }
}
```

VI. Baze de enumeratie. Conversii intre baza **10** si baza **b**, $2 \leq b \leq 9$, pentru un numar **n**

1. Din 10 in **b**:

- Conversia din baza 10 in baza **b**, este realizata prin impartiri succesive la **b**, cat timp $n > 0$.
- Resturile, in ordine inversa obtinerii lor, formeaza numarul in baza **b**, egal cu numarul **n**, in baza 10.

2. Din **b** in 10:

- Conversia din baza **b** in baza 10, este realizata prin inmultire cu puterile bazei **b**.
- Fiecare cifra a numarul in baza **b**, de la dreapta la stanga, este inmultita cu **b** la puterea **x**, unde $x = [0, \text{len}]$, unde **len** este lungimea numarului in baza **b**.

- Exemplu: convertim numarul **1001011** din baza 2 in baza 10:
 - $1 * 2^0 + 1 * 2^1 + 0 * 2^2 + 1 * 2^3 + 0 * 2^4 + 0 * 2^5 + 1 * 2^6 = 75$

Exercitii

PROBLEME PROPUSE

1. Se consideră programul pseudocod alăturat. S-a notat cu $x\%y$ restul împărțirii numerelor întregi x și y și cu $[z]$ partea întreagă a numărului real z .
- a) Care este valoarea afişată pentru $a = 1372$?

citește a (număr natural)
 $i \leftarrow 0$
 $a \leftarrow a \% 10$

- b) Scrie cea mai mică valoare de 3 cifre care se poate citi pentru a , astfel încât valoarea afişată de algoritm alăturat să fie maximă.
- c) Scrie un algoritm echivalent cu algoritmul dat în care structura **cât timp... execută** să fie înlocuită cu o structură repetitivă cu test final.
- d) Scrie programul C/C++ corespunzător algoritmului dat.

cât timp ($a > 1$) și ($a < 10$) execută
 $i \leftarrow i + 1$
 $a \leftarrow a * a$
 ■
 scrie $i * a$

2. Se consideră programul pseudocod alăturat. S-a notat cu $[z]$ partea întreagă a numărului real z .
- a) Care este valoarea afişată la citirea următorului sir de valori **5 37 205 199 30 86**?
- b) Scrie un sir de valori distințe, de cel mult două cifre pentru care algoritmul va afișa **9999**.
- c) Scrie un algoritm echivalent cu algoritmul dat în care prima structura **cât timp... execută** să fie înlocuită cu o structură repetitivă **pentru**.
- d) Scrie programul C/C++ corespunzător algoritmului dat.
3. Se consideră programul pseudocod alăturat. S-a notat cu n numărul de cifre ale numărului x .

citește n (număr natural nenul)
 $a \leftarrow 0$
 cât timp $n > 0$ execută
 citește x (număr natural nenul)
 $p \leftarrow 1$
 cât timp $p \leq [x/10]$ atunci
 ■ $p \leftarrow p * 10$
 $a \leftarrow a * 10 + [x/p]$
 $n \leftarrow n - 1$
 ■
 scrie a

numului dat.

corespunzător algo-

3. Se consideră programul pseudocod alăturat. S-a notat cu $x \% y$ restul împărțirii numerelor întregi x și y și cu $[z]$ partea întreagă a numărului real z .
- Ce va afișa algoritmul alăturat dacă se citesc valorile $a=12$ $b=24$?
 - Scrie toate perechile de valori de câte o cifră care pot fi citite pentru a și b , astfel încât ultima pereche afișată de algoritmul alăturat să fie $6\ 7$.
 - Scrie un algoritm echivalent cu algoritmul dat în care prima structura repetă... până când să fie înlocuită cu o structură repetitivă cu condiție inițială.
 - Scrie programul C/C++ corespunzător algoritmului dat.

Se consideră

 $n \leftarrow n-1$

scrie a

citește a, b (numere naturale nenule,
 $a < b$) $p \leftarrow a; u \leftarrow b$ cât timp $p < u$ execută $x \leftarrow p; y \leftarrow u$

repeta

 $z \leftarrow x \% y$ $x \leftarrow y$ $y \leftarrow z$ ■ până când $y = 0$ dacă $x = 1$ atunciscrie $p, ','; u, ',';$ $p \leftarrow p+1; u \leftarrow u-1$

numului dat.

corespunzător algo-

4. Se consideră programul pseudocod alăturat. S-a notat cu $x \% y$ restul împărțirii numerelor întregi x și y și cu $[z]$ partea întreagă a numărului real z .
- Ce va afișa algoritmul alăturat dacă se citesc valorile $a=10011$ și $b=2$?
 - Dacă pentru $b=8$, ce valoare poate fi citită pentru a , astfel încât algoritmul alăturat să afișeze **559**?
 - Scrie un algoritm echivalent cu algoritmul dat care să utilizeze o singură structură repetitivă.
 - Scrie programul C/C++ corespunzător algoritmului dat.

(Algoritmi elementari)

citește a, b (numere naturale > 0 ,
 $2 \leq b \leq 9$) $m \leftarrow 0; x \leftarrow 0$ cât timp $a > 0$ execută $p \leftarrow a \% 10$ pentru $i \leftarrow 1, m$ execută $p \leftarrow p * b$ $x \leftarrow x + p$ $a \leftarrow [a / 10]$ $m \leftarrow m + 1$

scrie x

5. Se consideră programul pseudocod alăturat. S-a notat cu $x \% y$ restul împărțirii numerelor întregi x și y și cu $[z]$ partea întreagă a numărului real z .
- Ce va afișa algoritmul alăturat dacă se citesc valorile **5 2 32 110 4 243 512**?
 - Pentru $n=3$, $a=5$ scrie un sir de valori distincte, de cel mult două cifre, ce pot fi citite pentru b , astfel încât algoritmul alăturat să afișeze 3.
 - Scrie un algoritm echivalent cu algoritmul dat în care structura **cât timp... execută** să fie înlocuită cu o structură repetitivă cu condiție finală.
 - Scrie programul C/C++ corespunzător algoritmului dat.

```

    citește n, a (număr natural nenul)
    x ← 0
    pentru i ← 1, n execută
        citește b (numere naturale
            0 < a ≤ b)
        p ← 1
        cât timp p < b execută
            p ← p * a
        x ← x + [b / p]
    scrie x

```

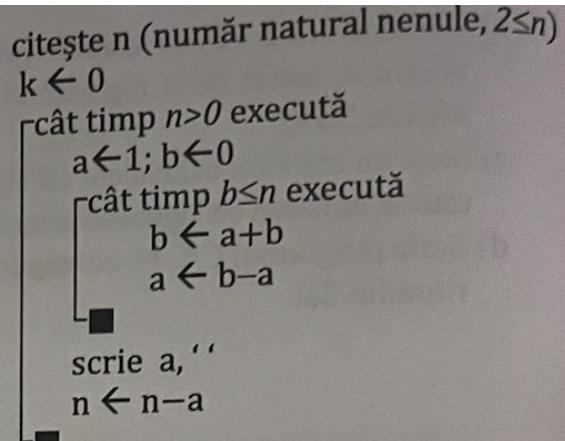
6. Se consideră programul pseudocod alăturat. S-a notat cu $x \% y$ restul împărțirii numerelor întregi x și y și cu $[z]$ partea întreagă a numărului real z .
- Ce va afișa algoritmul alăturat dacă se citesc valorile $x=2$ și $y=10$?
 - Scrie o pereche de valori de două cifre ce pot fi citite pentru x și y , astfel încât algoritmul alăturat să afișeze la sfârșit valoarea 1.
 - Scrie un algoritm echivalent cu algoritmul dat în care prima structură **cât timp... execută** să fie înlocuită cu o structură **pentru... execută**.
 - Scrie programul C/C++ corespunzător algoritmului dat.

```

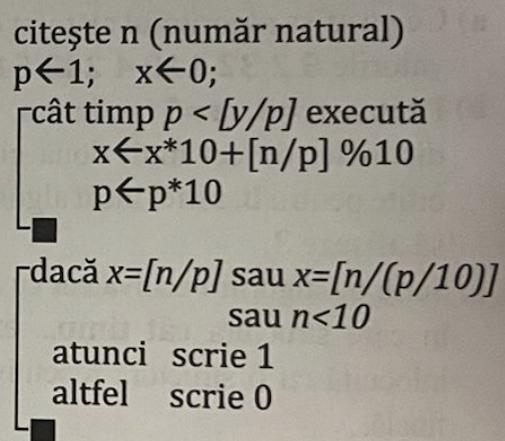
    citește x, y (numere naturale, 0 < x ≤ y)
    k ← 0
    cât timp y ≥ x execută
        d ← 2
        cât timp d * d ≤ y și y % d ≠ 0 execută
            d ← d + 1
        dacă d * d = y atunci
            scrie y, '
            k ← k + 1
        y ← y - 1
    scrie k

```

7. Se consideră programul pseudocod alăturat. S-a notat cu $x\%y$ restul împărțirii numerelor întregi x și y și cu $[z]$ partea întreagă a numărului real z .
- Ce va afișa algoritmul alăturat dacă se citește $n=120$?
 - Scrie cea mai mică valoare de trei cifre ce poate fi citită pentru n , astfel încât algoritmul alăturat să afișeze **o singură valoare**, pentru fiecare dintre acestea.
 - Scrie un algoritm echivalent cu algoritmul dat în care a doua structură **cât timp... execută** să fie înlocuită cu o structură repetitivă cu condiție finală.
 - Scrie programul C/C++ corespunzător algoritmului dat.



8. Se consideră programul pseudocod alăturat. S-a notat cu $x\%y$ restul împărțirii numerelor întregi x și y și cu $[z]$ partea întreagă a numărului real z .
- Ce va afișa algoritmul alăturat dacă se citește $n=32153$?
 - Câte numere de cel mult două cifre pot fi citite pentru n , astfel încât algoritmul alăturat să afișeze **1**?
 - Scrie un algoritm echivalent cu algoritmul dat în care structura **cât timp... execută** să fie înlocuită cu o structură repetitivă cu condiție finală.
 - Scrie programul C/C++ corespunzător algoritmului dat.



9. Se consideră programul pseudocod alăturat. S-a notat cu $x\%y$ restul împărțirii numerelor întregi x și y și cu $[z]$ partea întreagă a numărului real z .
- Ce va afișa algoritmul alăturat dacă se citește $n=4200$?
 - Scrie două valori distincte, de cel puțin trei cifre, ce pot fi citite pentru n astfel încât algoritmul alăturat să scrie o valoare egală cu n , pentru fiecare dintre ele.
 - Scrie un algoritm echivalent cu algoritmul dat în care a doua structură **cât timp... execută** să fie înlocuită cu o structură repetitivă cu condiție finală.
 - Scrie programul C/C++ corespunzător algoritmului dat.

```

citește n (număr natural nenul, n>1)
x ← 2; p ← 1
cât timp x≤n execută
    k←0;
    cât timp n%x=0 execută
        n←[n/x]
        k←k+1
    ■
    dacă k%2≠0 atunci
        p←p*x
    ■
    x←x+1
scrie p

```