

## Reprezentarea în virgulă flotantă

Un număr real în virgulă flotantă are o reprezentare de forma:  $x = \pm a_0.a_1a_2\dots a_{n-1} * b^e$  cu  $0 \leq a_i \leq b$  unde:

- $a_0.a_1a_2\dots a_{n-1}$  se numește **mantisă**,
- $b$  reprezintă **baza** în care este reprezentat numărul și
- $e$  este exponentul.

Pentru ca reprezentarea în virgulă flotantă să fie unică, **numerele flotante se normalizează astfel încât  $a_0 \neq 0$ .**

**Observație:** Normalizarea modifică doar *modul de reprezentare a numărului*, nu și valoarea sa.

### Exemplu 1:

Fie  $x_1 = 0.005 * 10^1$  scris în baza 10 are reprezentarea scrisă în virgulă flotantă:

$$x_1 = \begin{matrix} 0 & . & 0 & 0 & 5 & * & 10^1 \\ a_0 & & a_1 & a_2 & a_3 & & \text{baza}^{\text{exponent}} \end{matrix}$$

În acest exemplu  $n=4$ , se scrie *normalizat* sub forma  $x_1 = 5.000 * 10^{-2}$

În notația științifică numerele reale se notează sub forma:

$$\pm \text{mantisă} * \text{baza}^{\text{exponent}}$$

Scrierea valorilor reale sub forma  $x = \pm 0.fm * 10^e$  este o scriere cu mantisă subunitară în baza 10.

Scrierea valorilor reale sub forma  $x = \pm 1.fm * 2^e$  este o scriere a numărului în baza 2 cu mantisă între 1 și 2.  $fm$  reprezintă *partea fracționară a mantisei*.

Reprezentarea unui număr zecimal în virgulă flotantă se face astfel:

$$25_{(10)} = 11001_{(2)} = 1.1001 * 2^4$$

unde:

- **mantisă** este 1.1001
- **fracția mantisei** este  $fm = 1001$
- **baza** = 2
- **exponentul**  $e = 4$

### Exercițiu 1:

Reprezentați următoarele numere zecimale în baza 2 în virgulă flotantă:

$$x_1 = 176$$

$$x_2 = 125$$

$$x_3 = 41$$

Scrieți pentru fiecare reprezentare *forma normalizată* a numerelor, precum și *mantisă*, *fracția mantisei*, *baza* și *exponentul*.



Transformarea unui **număr real** din baza 10 în baza 2 se face astfel:

$$x=228.15$$

$$x=228+0.15$$

$$228=11100100_{(2)}$$

$$0.15 \cdot 2 = 0.30 = \mathbf{0} + 0.30$$

$$0.3 \cdot 2 = 0.60 = \mathbf{0} + 0.60$$

$$0.6 \cdot 2 = 1.20 = \mathbf{1} + 0.20$$

$$0.2 \cdot 2 = 0.40 = \mathbf{0} + 0.40$$

$$0.4 \cdot 2 = 0.80 = \mathbf{0} + 0.80$$

$$0.8 \cdot 2 = 1.60 = \mathbf{1} + 0.60$$

$$228:2=114 \text{ și rest } \mathbf{0}$$

$$114:2=57 \text{ și rest } \mathbf{0}$$

$$57:2=28 \text{ și rest } \mathbf{1}$$

$$28:2=14 \text{ și rest } \mathbf{0}$$

$$14:2=7 \text{ și rest } \mathbf{0}$$

$$7:2=3 \text{ și rest } \mathbf{1}$$

$$3:2=1 \text{ și rest } \mathbf{1}$$

$$1 \text{ rest } \mathbf{1}$$

citire

$$x = 11100100 . 00100110011001...$$

Forma normalizată a lui  $x$  este:  $1.11001000010011001... \cdot 2^7$

**Exemplu 3:** Reprezentarea numărului  $-14.25$  în virgulă flotantă, în simplă precizie se face astfel:

**Pas 1)** Se scrie numărul fără semn  $14.25$  în binar:

$$14 : 2 = 7 \text{ (rest 0)}$$

$$7 : 2 = 3 \text{ (rest 1)}$$

$$3 : 2 = 1 \text{ (rest 1)}$$

$$1 \text{ (rest 1)}$$

$$0.25 \cdot 2 = 0 + 0.50$$

$$0.50 \cdot 2 = 1 + 0.00$$

$$0.00 \cdot 2 = 0 + 0.00$$

$$14.25_{(10)} = 1110.0100000..._{(2)}$$

**Pas 2)** Se scrie numărul obținut în binar în formă normalizată:

$$1.11001000000 \cdot 2^3 \text{ (} fm = 11001000000... \text{)}$$

**Pas 3)** Se determină valoarea caracteristicii  $C$  după formula **Exponent\_bază + 127**

$$\text{Exponent\_bază} = 3, C = 3 + 127 = 130$$

**Pas 4)** Se transformă  $C$  în binar:  $130_{(10)} = 10000010_{(2)}$

**Pas 5)** Se determină bitul de semn al mantisei: **1** (deoarece numărul este negativ)

**Pas 6)** Reprezentăm numărul în simplă precizie:

Bit																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																
31	30																																23	22																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																														

### Exercițiu 2:

a) Să se reprezinte în simplă precizie numerele  $(n_i)_{i=1:5}$  în baza 2, în virgulă flotantă:

$$n_1=228.15$$

$$n_2=-27.25$$

$$n_3=135.25$$

$$n_4=0.1$$

$$n_5=-1.2$$

- b) Numărul  $x_1=0.002 \cdot 10^1$  scris în baza=10, cu reprezentarea în *virgulă mobilă*, se scrie normalizat, pentru  $n=4$ , sub forma:  $x_1 = 2.000 \cdot 10^{-2}$ . Scrieți normalizat, pentru  $n=4$ , următoarele numere:

$$x_2=0.003 \cdot 10^3$$

$$x_3=0.200$$

$$x_4=0.009$$

$$x_5=0.050$$

- c) Pentru  $b=2$  și  $n=14$ , numărul  $x=0.1_{(10)} \cong 1.1001100110011 \cdot 2^{-4}$ , dar pentru numerele:

- $y=0.25_{(10)}$  când  $b=8$  și  $n=14$  și
- $z=0.2_{(10)}$  când  $b=16$  și  $n=6$

Transformarea unui număr subunitar  $x$ , scris în baza 10, în numărul  $\tilde{x} = 0.c_1c_2c_3 \dots c_{n-1}$  scris în baza 2  
 $x = 0.1_{(10)}$

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
x	0.1	0.2	0.4	0.8	1.6	1.2	0.4	0.8	1.6	1.2	0.4	0.8	1.6	1.2
c[i]	0	0	0	0	1	1	0	0	1	1	0	0	1	1

Poziția punctului zecimal

Adică  $0.1_{(10)} = 0.0001100110011$

Transformarea unui număr subunitar  $x$ , scris în baza 10, în numărul  $\tilde{x} = 0.c_1c_2c_3 \dots c_{n-1}$  scris în baza 2 se face conform algoritmului:

**conversie** (x, c, n)

i:=0; c<sub>0</sub>:=0;

repeat

    i:=i+1

    x:=x\*2

    c[i]:=[x]

    if x>=1 then

        x:={x}

until i=n

Programul MATLAB următor convertește numărul  $x$  din baza 10 în baza 2 cu exact  $n$  cifre:

```
function []=conversie2(x,n)
% conversie x din baza 10 in baza 2

if nargin~=2
    error('nr de argumente gresit! Sintaxa este conversie(nr_baza_10,nr_cifre)');
elseif n<0
    error('n reprezinta nr de cifre ale lui x si nu poate fi nr negativ');
end

nrNeg=0;
if x<0
    nrNeg=1; x=abs(x); % nrNeg = true daca avem un numar x negativ
end

vect=[]; % folosit pentru a scrie numarul din fata virgulei in baza 2
ParteInt=0; ParteFract=0; % ParteInt,ParteFract - variab folosite pentru test
```

```

% conversia pentru partea intreaga a lui x
if fix(x)>0 % verificam daca avem parte intreaga la numarul real x
    ParteInt=1; % ParteInt = true daca avem un numar x cu parte intreaga
    x1=fix(x); %retin in x1 partea intreaga a lui x
    while fix(x1) > 0
        rest=mod(x1,2); % retin restul impartirii la 2
        x2=fix(x1/2); % retin catul lui x1/2
        x1=x2; % retin noua valoare pt x1
        vect=[vect,rest]; % salvam in vect resturile împărțirii la 2
    end
    cifre_parte_intreaga=fliplr(vect); % simetrie de la st la dr
else
    cifre_parte_intreaga=0;
end

i=0; % indicele
c=[]; % vectorul cu cifre

% x - numar subunitar

% Transformarea unui numar subunitar x, scris in baza 10, in numarul
% x_tilda = 0,c(1),c(2),...,c(n-1), scris in baza 2 se face conform
% algoritmului de mai jos

if ( x-fix(x) > 0 ) % verificam daca avem zecimale la numarul real
    ParteFract=1; % ParteFract = true daca avem un numar x cu parte fractionara
    while (i~=n)
        i=i+1;
        parte_zecimala=x-fix(x);
        x_tilda=parte_zecimala;
        x_tilda=x_tilda*2;
        c=[c,fix(x_tilda)]; % vectorul cu cifre de 0 si 1 (cu partile intregi)
        x=x_tilda-fix(x_tilda);
    end
    cifre_parte_zecimala=c;
end

% afisare rezultat
% rezultatul este afisat exact cu n cifre
if n < length(cifre_parte_intreaga)
    fprintf('Atentie! n < %d (nr necesar de cifre)\n',
length(cifre_parte_intreaga));
    error('nu se poate scrie numarul normalizat! Dati nr de cifre mai mare!')
end

if(ParteInt==1) % exista parte intreaga
    fprintf('Conversia in baza 2 pentru n=%d este:\n',n);
    if nrNeg==1
        fprintf('-'); % pentru ca avem un numar negativ va adauga '-'
    end
    for i=1:length(cifre_parte_intreaga)
        fprintf('%d',cifre_parte_intreaga(i));
    end
else
    if nrNeg==1
        fprintf('-0'); % pentru ca avem un numar negativ va adauga '-'
    else
        fprintf('0'); % daca nu avem valori inainte de virgula
    end
end
end

```

```

if(ParteFract==1) % exista parte zecimala
    dimensiune_max = length(cifre_parte_zecimala)-length(cifre_parte_intreaga);
    if dimensiune_max > 0
        fprintf('.'); % va adauga '.' pentru punctul zecimal
        for i=1:dimensiune_max
            fprintf('%d',cifre_parte_zecimala(i));
        end
    else
        %completam daca este nevoie cu 0 pentru nr de cifre dat
        z=zeros(1,n-length(cifre_parte_intreaga));
        fprintf('%d',z);
    end
else % nu avem parte zecimala
    if n ~= length(cifre_parte_intreaga)
        % verificam daca este nevoie de punctul zecimal, in cazul nr intregi
        fprintf('.');
    end
    %completam daca este nevoie cu 0 pentru nr de cifre dat
    z=zeros(1,n-length(cifre_parte_intreaga));
    fprintf('%d',z);
end

fprintf('\n');

```

Rezultatul algoritmului pentru  $x=0.1$  și  $n=14$  este:

Apelul funcției se face prin:

```
>> conversie2(0.1,14) % Conversia lui x = 0.1 in baza 2 pentru n=14 cifre
```

0.0001100110011

### Exercițiu 3:

- Converțiți numerele  $(2)_{16}$  și  $(1.F)_{16}$  din baza 16 în baza 10. Comparați numerele între ele.
- Determinați  $(a)_{10}$  pentru  $a=+(0.5374)_8 \cdot 8^6$

**Exercițiu 4:** Să se calculeze suma numerelor:  $x_1 = 4.0321 \cdot 10^2$ ,  $x_2 = 2.3901 \cdot 10^3$ ,  $x_3 = 1.5213 \cdot 10$ , considerând că ele sunt reprezentate sub formă normalizată într-un sistem de calcul cu virgulă mobilă având baza de numerație  $b=10$  și numărul de cifre  $n=6$ . Scrieți suma calculată sub formă normalizată.

**Exercițiu 5:** Completați tabelul:

Baza 10	15	0.12				
Baza 2			0.011	110		
Baza 16					B.C	D.F

**Exercițiu 6:** Să se modifice funcția *conversie2(x,n)* definită mai sus, astfel încât conversia unui număr zecimal să se poată face în orice bază, condiția este ca  $2 \leq \text{bază} < 10$ .

Cerințe:

- Modificați numele funcției în *conversie\_min10(x,n,baza)*.
- Programul va da un mesaj de eroare în cazul în care se va scrie o bază mai mare ca 9 sau mai mică decât 2.
- Se va da un mesaj de eroare dacă *baza* și *n* (nr. de cifre) nu sunt numere naturale (Indicație: Se verifică dacă *baza* este pozitivă și întreagă (se compară partea întreagă a bazei, adică **fix(baza)**, cu valoarea dată ca argument, **baza**).

Notă! Atenție la mesajele afișate. Acestea trebuie să indice baza corectă, cea dată în argumentul de intrare.

*Exemple de apel:*

```

Command Window
>> conversie2_min10(1023,17,20)
Error using conversie2_min10 (line 11)
baza nu e corect indicata. baza = [2,10]

>> conversie2_min10(1023,17,-5)
Error using conversie2_min10 (line 11)
baza nu e corect indicata. baza = [2,10]

>> conversie2_min10(1023,17,5.2)
Error using conversie2_min10 (line 13)
baza si nr de cifre trebuie sa fie nr.
naturale

>> conversie2_min10(1023,17,2,5)
Error using conversie2_min10 (line 13)
baza si nr de cifre trebuie sa fie nr.
naturale

```

```

>> conversie2_min10(1023,17,5)
Conversia in baza 17 pentru n=5 este:
13043.000000000000

>> conversie2_min10(1023,7,3)
Conversia in baza 7 pentru n=3 este:
1101220

>> conversie2_min10(-1023,7,8)
Conversia in baza 7 pentru n=8 este:
-1777.000

>> conversie2_min10(-102.3,7,8)
Conversia in baza 7 pentru n=8 este:
-146.2314

>> conversie2_min10(1023,7,2)
Atentie! n < 10 (nr necesar de cifre)
Error using conversie2_min10 (line 65)
nu se poate scrie numarul normalizat! Dati
nr de cifre mai mare!

```

**Exercițiu 7:** Să se modifice funcția  $conversie2(x,n)$  astfel încât conversia unui număr zecimal să se poată face în baza 16. Modificați numele funcției în  $conversie16(x,n)$ . Aici se va trata situația în care știm că 10 este A, 11 este B,..., 15 este F.

*Exemplu de apel:*

```

Command Window
>> conversie16(-2035.28,25)
Conversia in baza 16 pentru n=25 este:
-7F3.47AE147AE1400000000000

>> conversie16(2,25)
Conversia in baza 16 pentru n=25 este:
2.000000000000000000000000

>> conversie16(-0.768,25)
-0.C49BA5E353F7D00000000000

>> conversie16(-0.768,10)
-0.C49BA5E35

```