Baze de numeratie

Transformarea din baza 10 în baza b

Cum transformăm un număr oarecare `n` din baza `10` într-o bază oarecare, `b`? Algoritmul de transformare este foarte asemănător cu cel de determinare a cifrelor în baza `10`. De fapt, este chiar identic, doar că baza nu este `10`, ci `b`. Mai exact:

- împărțim numărul `n` la `b`. Obținem un cât și un rest;
- împărțim câtul la `b`. Obținem un cât și un rest;
- împărțim noul cât la `b` și obținem un cât și un rest;
- continuăm împărțirile până când obținem câtul `0`;
- resturile obținute, scrise în ordinea inversă obținerii, reprezintă scrierea în baza `b` a lui `n`.

Exemplu: să transformăm numărul `24` din baza `10` în baza `2`. Efectuăm împărțirile:

24:2 =12 rest 0

12:2 = 6 rest 0

6:2 =3 rest 0

3:2 =1 rest 1

1:2 = 0 rest 1

Scriem resturile în ordine inversă și obținem: `24(10) = 11000(2)`. Să observăm că am obținut cifrele în ordinea inversă față de poziția lor în număr!!

Transformarea din baza b în baza 10

Pentru transformarea numărului $C_kC_{k-1}\dots C_1C_0$ din baza `b` în baza `10` folosim formula:

$$C_kC_{k-1}\dots C_1C_0 = C_k * b^k + C_{k-1} * b^{k-1} + \dots + C_1 * b^1 + C_0 * b^0$$

în care operațiile de adunare, înmulțire și ridicare la putere se fac în baza `10`. De fapt această formulă este echivalentă cu reprezentarea zecimală a numerelor, de exemplu:

$$253_{(10)} = 2 \cdot 10^2 + 5 \cdot 10^1 + 3 \cdot 10^0$$

Exemplu:

$$\begin{aligned} 11000_{(2)} &= 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 \\ &= 1 \cdot 16 + 1 \cdot 8 \\ &= 16 + 8 \\ &= 24 \end{aligned}$$

Transformarea din baza b în baza d

De regulă, pentru transformarea dintr-o bază oarecare `b` într-o bază oarecare `d` se folosește ca “bază de manevră” baza `10`. Mai exact:

- se dă un număr `x` în baza `b`
- se transformă numărul `x` din baza `b` în baza `10` și se obține un număr `y`
- se transformă numărul `y` din baza `10` în baza `d` și se obține rezultatul final

Transformări intre baze puteri ale lui 2

Un caz particular îl constituie transformările între baza 2 și altă bază care este putere a lui 2. De exemplu, să analizăm reprezentările în bazele 2 și 16 ale numărului $2018_{(10)}$

2018(10):

- $2018_{(10)} = 7E2_{(16)}$
- $2018_{(10)} = 11111100010_{(2)}$

Vom proceda astfel, pentru transformarea din baza 2 în baza 16 – algoritmul este similar și pentru alte baze, putere al ui 2:

- numărul în baza 2 se împarte în grupe de câte 4 cifre. Prima grupă poate fi incompletă.
- transformăm fiecare grupa în baza 16, de la sfârșit spre început, obținând câte o cifră.
- scriem rezultatul în baza 16.

Cum facem transformarea inversă, din baza 16 în baza 2?

- transformăm fiecare cifră a numărului din baza 16 în baza 2. Vom obține pentru fiecare cifră un șir cu cel mult 4 biți.
- dacă un șir conține mai puțin de 4 biți, îl completăm cu zerouri nesemnificative, cu excepția primului grup (corespunzător primei cifre)
- scriem șirurile de biți în ordine, obținând reprezentarea în baza 2

Operatii pe biti

Operatii pe biti, tabelul de adevar:

AND		
A	B	Result
0	0	0
0	1	0
1	0	0
1	1	1

OR		
A	B	Result
0	0	0
0	1	1
1	0	1
1	1	1

XOR		
A	B	Result
0	0	0
0	1	1
1	0	1
1	1	0

NOT	
A	Result
0	1
1	0

Pe langa cele 4 operatii prezentate anterior, mai avem 2 operatii:

- Shift bits to the left:

  x << i ⇔ in reprezentarea lui x in baza 2 se muta i biti la stanga

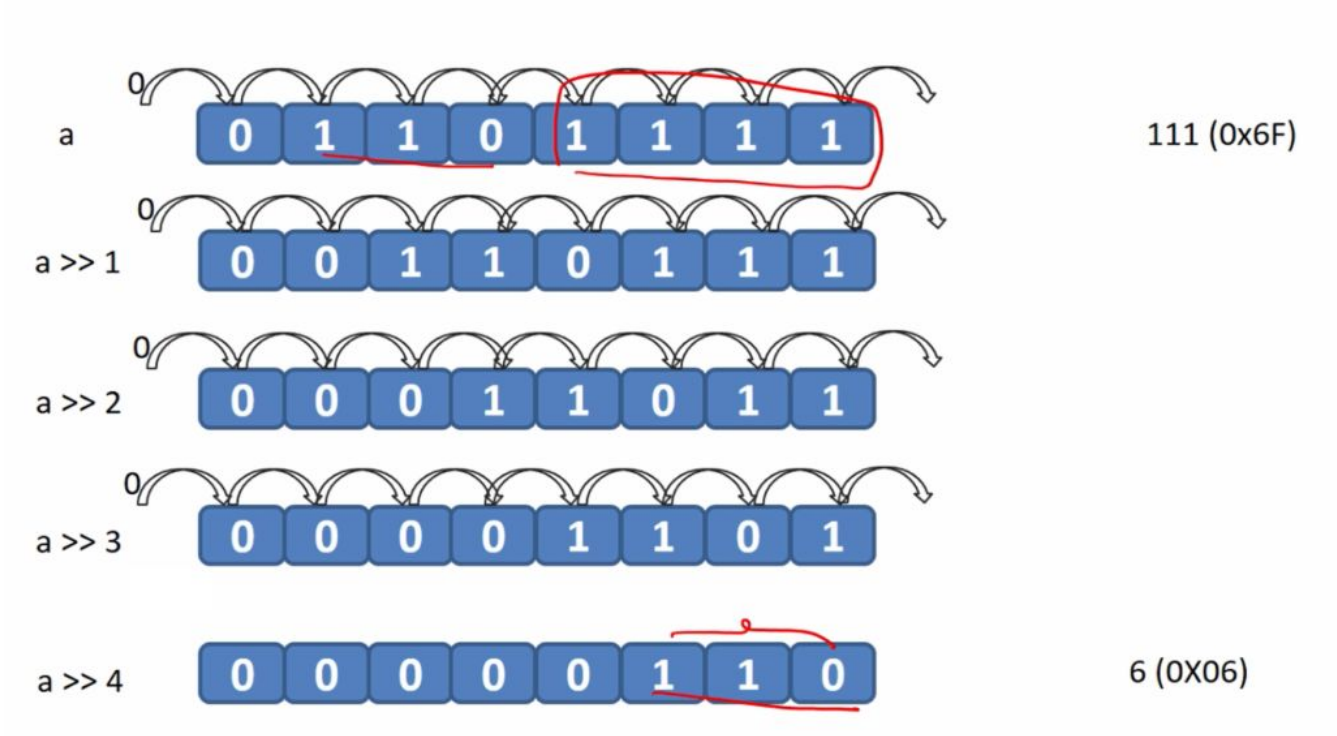
BITWISE OPERATORS

<< Shift Left

SYNTAX	BINARY FORM	VALUE
x = 7;	00000111	7
x=x<<1;	00001110	14
x=x<<3;	01110000	112
x=x<<2;	11000000	192

- Shift bits to the right:

  x >> i ⇔ in reprezentarea lui x in baza 2 se muta i biti la dreapta



Observatii importante:

Fie x un numar natural.

- Puterile de 2 au un singur bit de 1 in reprezentarea lor. (i.e. 2^k, are doar bitul k 1)
- Pentru a **verifica** bitul de la pozitia i din reprezentarea lui x in baza 2 folosim: **x & (2 ^ i)** (daca x & (2 ^ i) este egal cu 0, atunci bitul i este 0, altfel bitul i este 1)
- Pentru a **seta (il fac 1 daca nu era)** bitul de la pozitia i din reprezentarea lui x in baza 2 folosim: **x = x | (2 ^ i)**
- Pentru a **inversa** bitul de la pozitia i din reprezentarea lui x in baza 2 folosim: x = **x ^ (2 ^ i)**
- Shiftarea la dreapta cu i biti este echivalenta cu **impartirea** numarului la **(2 ^ i)**
- Shiftarea la stanga cu i biti este echivalenta cu **inmultirea** numarului cu (**2 ^ i)**
- Astfel, pentru a obtine repede valorea 2^i, folosim **(1 << i) ⇔ 1 * (2 ^ i)**
- **x ^ x = 0, x ^ 0 = x, x ^ y = y ^ x, (x ^ y) ^ z = x ^ (y ^ z)**

Probleme si exercitii

1. Interschimbati 2 variabile folosind doar operatii pe biti.
2. Se citesc n - 1 numere naturale distincte cu valori intre 1 si n. Gasiti numarul natural x (x <= n) care nu a fost citit.
3. https://www.pbinfo.ro/probleme/426/bazab
4. https://www.pbinfo.ro/probleme/1300/hex
5. https://www.pbinfo.ro/probleme/2560/bits
6. https://www.pbinfo.ro/probleme/2973/cate3cifre