**Aritmetică binară și algoritmi**

abcdef(2)=a⋅25+b⋅24+c⋅23+d⋅22+ e⋅21+f

unde cifrele sunt 0 sau 1.

DECI fiecare putere a lui 2 poate să apară sau nu.

Acest lucru este echivalent și cu unicitatea descompunerii unui număr natural în sumă de puteri ale lui 2.

Acest lucru ne dă totodată și o metodă mult mai rapidă (când e vorba de calcule fără mașini) de transformat din baza 10 în baza 2:

Ne luăm puteri ale lui 2 de la cea mai mare putere cuprinsă în număr către puteri mai mici și pe cele care intră în descompunere le scădem.

Ex:

x=169

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |

169-128=41

41-32=9

Obs: în baza 2 cea mai mare cifră posibilă este 1. Ea joacă rolul lui 9 (gen dacă în baza 10 după 9 urmează 10, în baza 2 după 1 urmează 10.

La fel - după 99 urmează 100, atunci în baza 2 după 11 urmează 100.

Analog după 111...1 urmează 1000...0 (atâția 0 câți de 1 aveam în numărul inițial). Acest lucru se traduce algebric ca:

(2k+2k-1+..+22+2+1) + 1 = 2k+1

Apoi, algebric, dacă în baza 10 înmulțirea cu 10 produce shiftarea cifrelor cu 1 poz. la stânga și adăugarea unui 0, în baza 2 lucrurile stau la fel (doar că e vorba de 10-le bazei 2, adică 10(2)=2(10) ).

Analog, împărțirea la 10 și obținerea câtului este echivalentă cu shiftarea la dreapta (și pierderea ultimei cifre)

**Operațiile pe biți**

● Shiftarea:

- la stânga se face prin a<<b. Semnificația: a se deplasează la stânga cu b biți. Atenție să puneți în paranteze, ca să nu avem probleme cu prioritățile operațiilor. a<<b este perfect echivalentă cu a⋅2b

- la dreapta se face prin a>>b. Semnificația: a se deplasează la dreapta cu b biți. a>>b este perfect echivalent cu a / 2b (câtul împărțirii întregi)

Obs:

1<<31 reprezintă INT\_MAX + 1 (pt. că INT\_MAX este cu toți biții 1)

Dacă reprezentăm pe int 1<<31 ne dă de fapt INT\_MIN)

Obs: Atenție la priorități - sunt mai scăzute decât ale operatorilor obișnuiți +, -, \*, / - din acest motiv e indicat să utilizați paranteze (chiar și dacă ele sunt în plus)

● negația: se neagă bit cu bit:

~valoare

(atenție, valorile int fiind reprezentate în cod complementar, rezultatul conține și bitul de semn și reprezentările aferente)

● conjuncția: se face operația "și" bit cu bit

v1 & v2

Ex: 23 & 17 = 10111 & 10001 = 10001 = 17

● disjuncția exclusivă: se face operația "xor" bit cu bit

v1 ^ v2

(xor este 1 doar din 0 și 1 sau 1 și 0, în rest e 0)

● disjuncția: se face operația "sau" bit cu bit

v1 | v2