OPERATORII LA NIVEL DE BIT

Operatorii la nivel de bit se aplica fiecarui bit din reprezentarea *operanzilor intregi*, spre deosebire de restul operatorilor care se aplica valorilor operanzilor.

Din aceasta categorie fac parte operatorii urmatori, care apar in ordinea descrescatoare a prioritatii:

~	complementare
>>	deplasare la dreapta
<<	deplasare la stanga
&	si
^	sau exclusiv
|	sau

Operatorul ~ transforma fiecare bit din reprezentarea operandului in complementarul sau -- bitii 1 in 0 si cei 0 in 1.
Operatorii &, ^, | realizeaza operatiile *si, sau exclusiv*, respectiv *sau* intre toate perechile de biti corespunzatori ai operanzilor. Daca b1 si b2 reprezinta o astfel de pereche, tabelul urmator prezinta valorile obtinute prin aplicarea operatorilor &, ^, |.

b1	b2	b1&b2	b1^b2	b1|b2
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	0	1

Din tabela de mai sus *se observa* ca aplicand unui bit:

- operatorul & cu 0, bitul este sters
- operatorul & cu 1, bitul este neschimbat
- operatorul | cu 1, bitul este setat
- operatorul ^ cu 1, bitul este complementat.

Exemplu:

```
char a,b;
```

a	b	~a	!a	~b	!b	a&b	a^b	a|b
00000000	00000001	11111111	1	11111110	0	00000000	00000001	00000001
11111111	10101010	00000000	0	01010101	0	10101010	01010101	11111111

In cazul operatorilor de deplasare, care sunt binari, primul operand este cel al carui biti sunt deplasati, iar al doilea indica numarul de biti cu care se face deplasarea -- deci numai primul operand este prelucrat la nivel de bit:

a<<n
a>>n

La *deplasarea la stanga* cu o pozitie, bitul cel mai semnificativ se pierde, iar in dreapta se completeaza cu bitul 0.
La *deplasarea la dreapta* cu o pozitie, bitul cel mai putin semnificativ se pierde, iar in stanga se completeaza cu un bit identic cu cel de semn.
Cu exceptia cazurilor cand se produce depasire, deplasarea la stanga cu n biti echivaleaza cu inmultirea cu 2 la puterea n.
Analog, deplasarea la dreapta cu n biti echivaleaza cu impartirea cu 2 la puterea n. Este indicat a se realiza inmultirile si impartirile cu puteri ale lui 2 prin deplasari, ultimele realizandu-se intr-un timp mult mai scurt.

Exemple:

- Cele doua secvente din tabela conduc la aceleasi rezultate:

```
int i;
```

`i*=8;`
`i/=4;`
`i*=10;`
	`i<<=3;`
	`i>>=2;`
	`i=i<<3+i<<1;`

- In tabela urmatoare apar valorile obtinute (in binar si zecimal) prin aplicarea operatorilor de deplasare:

```
char a;
```

a	a<<1	a<<2	a>>1	a<<2
00000001 1	00000010 2	00000100 4	00000000 0	00000000 0
00001110 14	00011100 28	00111000 56	00000111 7	00000011 3
11111111 -1	11111110 -2	11111100 -4	11111111 -1	11111111 -1
11011101 -35	10111010 -70	01110100 116(depasire)	11101110 -18	11110111 -9

- Mai jos este prezentata o secventa care construieste o masca ce contine n biti de 1 incepand cu pozitia p, spre dreapta (bitul cel mai putin semnificativ se considera ca fiind de pozitie 0)

```
int masca=0, unu=1;
int i,p=5,n=3;

unu<<=p;   // bitul 1 este pe pozitia p
for(i=1;i<=n;i++){
 masca|=unu;
 unu>>=1;
}
```