Operatori logici pe biţi

1 Noţiuni teoretice

Operatorii logici pe biţi se aplică bit cu bit numai datelor de tip întreg, mai precis reprezentărilor interne ale acestora. În continuare ne referim la modul în care se comportă aceşti operatori în C++.

Operatorii logici pe biţi sunt: negaţia ($\sim$), conjuncţia (&),disjuncţia (|), disjuncţia exclusivă(^), deplasarea spre stânga << şi deplasare spre dreapta (>>).

Operatorul de negaţie este un operator unar care aplicat unui număr întreg x returnează un număr întreg a cărui reprezentarea internă se obţine din reprezentarea internă a lui x prin complementarea faţă de unu a fiecărui bit ($\sim 1 = 0$ şi $\sim 0 = 1$).

Exemplu:

Considerând *int* $x = 25$, reprezentarea internă a lui x (pe 2 octeţi) este: 0000000000011001. Atunci $\sim x$ returnează 1111111111100110.

Acesta reprezentare corespunde unui număr negativ. Pentru a determina valoarea absolută a acestuia urmăm paşii inverşi operaţiei de complementare: $1111111111100110 - 1 = 1111111111100101$ şi aplicând fiecărui bit operatorul $\sim$ obţinem 0000000000011010 care corespunde numărului 26.

Deci, dacă $x = 25$ atunci $\sim x = -26$.

În general pentru x număr întreg (cu semn) $\sim x = -x - 1$.

Observaţie:

Pentru *unsigned int* $x = 25$, reprezentarea internă a lui x pe 2 octeţi este tot 0000000000011001, iar $\sim x$ returnează 1111111111100110. De această dată însă numărul este interpretat ca unul natural şi valoarea corespunzătoare este $2^{16} - 1 - (2^4 + 2^3 + 1) = 65510$.

Operatorul de conjuncţie (&) este un opertor binar care returnează numărul întreg a cărui reprezentare internă se obţine prin conjunţia perechilor de biţi situaţi pe aceeaşi poziţie în reprezentările interne ale operanzilor.

(1&1 = 1, iar 0&0 = 1&0 = 0&1 = 0)

Exemplu: Dacă $x = 5$ şi $y = 12$ atunci reprezentările interne ale celor două numere sunt:

x : 0000000000000101 şi y = 0000000000001100, de unde

$x\&y$: 0000000000000100 care are corespondentul zecimal 4.

Operatorul de disjuncţie (|) este un operator binar care returnează numărul întreg a cărui reprezentare internă se obţine prin disjunţia biţilor situaţi pe aceeaşi poziţie în reprezentările interne ale operanzilor.

(1|0 = 0|1 = 1|1 = 1, iar 0|0 = 0)

Exemplu: Dacă $x = 5$ şi $y = 12$ atunci reprezentările interne ale celor două numere sunt:

x : 0000000000000101 şi y = 0000000000001100, de unde

$x|y$: 0000000000001101 care are corespondentul zecimal 13.

Operatorul de disjuncţie exclusivă (^) este un operator binar care returnează numărul întreg a cărui reprezentare internă se obţine prin disjunţia exclusivă a biţilor situaţi pe aceeaşi poziţie în reprezentările interne ale operanzilor.(1^1 = 0^0 = 0, iar 1^0 = 0^1 = 1)

Exemplu: Dacă $x = 5$ şi $y = 12$ atunci reprezentările interne ale celor două numere sunt:

x : 0000000000000101 şi y = 0000000000001100, de unde

$x\hat{}y$: 0000000000001001 care are corespondentul zecimal 9.

Operatorul de deplasare spre stânga (shift left) << este un operator binar care returnează numărul întreg a cărui reprezentare este obţinută din reprezentarea internă a primului operand prin deplasarea la stânga cu un număr de biţi egal cu al doilea operand. Biţii rămaşi liberi se completează cu 0.

Exemplu: Dacă $x = 5$ cu reprezentarea internă x : 0000000000000101, atunci $x << 2$ returnează 0000000000010100 adică valoarea 20.

Observaţie: Expresia $x << n$ are ca efect înmulţirea operandului x cu 2^n.

Operatorul de deplasare spre dreapta (shift right) >> este un operator binar care returnează numărul întreg a cărui reprezentare este obţinută din reprezentarea internă a primului operand prin deplasarea la drepta cu un număr de biţi egal cu al doilea operand. Biţii rămaşi liberi se completează cu bitul de semn al numărului iniţial.

Exemplu: Dacă $x = 12$ cu reprezentarea internă x : 0000000000001100, atunci $x >> 2$ returnează 0000000000000011 adică valoarea 3.

Observaţie: Expresia $x >> n$ are ca efect împărţirea întreagă a operandului x la 2^n.

Acţionând direct asupra reprezentării interne a operanzilor, operaţiile efectuate pe biţi sunt foarte performante (se execută foarte rapid).

2 Aplicaţii

1. Verificaţi dacă un număr natural n introdus de la tastatură este par sau impar.
   Indicaţie: Dacă n este par cel mai din dreapta bit este 0, altfel este 1. Astfel, se poate folosi conjuncţia dintre n şi 1.

2. Fiind dat de la tastatură un număr natural $k < 15$, afişaţi 2^k.
   Indicaţie: Folosiţi operatorul de deplasare la stânga.

3. Fiind dat un număr natural n să se determine câtul şi restul împărţirii acestuia la 8.
   Indicaţie: Câtul se obţine folosind deplasarea la dreapta, iar restul folosind conjuncţia dintre n şi 7.

4. Fiind dat un număr natural n să se verifice dacă n este o putere a lui 2.
   Indicaţie: n este o putere a lui 2 dacă şi numai dacă reprezentarea sa internă conţine un sigur bit egal cu 1. Dacă presupunem că acest bit ocupă poziţia k, atunci pentru $n - 1$ toţi biţii din dreapta lui k devin 1, iar bitul k şi toţi biţii din stânga sa devin 0. În consecinţă n este o putere a lui 2 dacă şi numai dacă $n\&(n - 1) == 0$.

5. Fiind dat un număr natural n să se afişeze reprezentarea lui n în baza 2.
   Indicaţie: Bitul de pe poziţia i este dat de expresia logică $(n >> i)\&1$.

6. Fiind dat n un număr natural ($n \leq 10000$) să se genereze toate numerele prime mai mici decât n.

Indicaţie: Se utilizează Ciurul lui Eratostene. Ideea acestuia este de a pune în "ciur" toate numerele de la 2 la n şi de "a le cerne" în aşa fel în cât în ciur să rămână doar numerele prime. Prin "a cerne numerele" se înţelege a elimina toţi multiplii de 2, apoi toţi multiplii de 3 ş.a.m.d.

Ciurul se reprezintă ca un vector cu maxim 10000 de componente care pot fi 0 sau 1, cu semnificaţia ciur[i]=1 dacă numărul i este în ciur şi 0 altfel.

Vectorul ciur este parcurs de la 2 (cel mai mic număr prim), până la $[\sqrt{n}]$, iar pentru orice i cu proprietatea că ciur[i]==1 eliminăm din ciur toţi multiplii de i. Iniţial toate numerele de la 2 la $n-1$ se consideră în ciur. La final se afişează elementele rămase în ciur care vor fi numerele prime mai mici ca n.

7. Scrieţi un program care afişează numărul cifrelor de 1 din reprezentarea internă a unui număr natural n.

8. Scrieţi un program care afişează cea mai mare putere a lui 2 care divide un număr natural dat n.