**Operatii pe biti**

1. Se consideră un număr natural n. Să se verifice dacă n este par sau impar.
2. Se citeşte un număr natural k <= 15. Să se afişeze valoarea 2k.
3. Se consideră un număr natural n. Să se determine câtul şi restul împărţirii lui n la 8. Generalizare: să se determine câtul şi restul împărţirii lui n la un număr care este putere a lui 2.

N=25 > 3 si 1

Cout<<(n>>3) – catul restul **cout<<(n&7)**

1 1 0 0 1 n=29 1 1 1 0 1 &

0 0 1 1 1 0 0 1 1 1

0 0 0 0 1 0 0 1 0 1 = 5

1. Se consideră un număr natural n. Să se verifice dacă n este sau nu o putere a lui 2.

8= 1000 &

7= 0111

1. Să se determine numărul de biţi de 1 din reprezentarea binară a lui n.
2. Se consideră un număr natural n. Să se afişeze reprezentarea lui n în baza 2.
3. Să se determine cel mai puţin semnificativ bit de 1 din reprezentarea binară a lui n..
4. Realizaţi un subprogram care identifică cea mai mare putere a lui 2 care îl divide pe n, număr natural nenul.

N=24 <= 2^3

N=15 => 2^0

1. Scrieti un program care rotunjeste un numar natural n la cea mai apropiata putere a lui 2.

N=125 => n=>128

N=24 => 32

N=32 => 32

1. Reprezentarea mulțimilor prin vector caracteristic implementat pe biți

Fie n un număr natural (n<VMAX).

O metodă eficientă de a reprezenta o submulțime S a mulțimii {0, 1, ..., n} este vectorul caracteristic.

Mai exact, asociem fiecărui element x din mulțimea {0, 1, ..., n} un bit. Bitul corespunzător elementului x

are valoarea 1 dacă x∈S, respectiv valoarea 0 dacă x∉S.

**Descrieți funcții care să implementeze operațiile elementare cu mulțimi (reuniune, intersecție,**

**diferență, incluziune, test de apartenență) folosind o astfel de reprezentare.**