Bitenkénti operátorok használata C nyelven

Dr. Varga Imre

Debreceni Egyetem
Informatikai Kar
Informatikai Rendszerek és Hálózatok Tanszék

2026.06.11

Létre szeretnél hozni egy bizonyos bitmintát?

Használj **hexadecimális** számokat!

Felhasználás:

Például **maszk**ok megadása.

Elvárt érték:

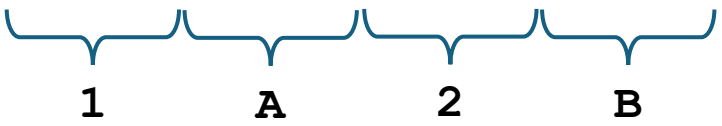
```
0001101000101011
```

Szükséges kód:

```
short int N = 0x1A2B;
// decimal: 6699
// octal: 015053
```

Matematikai háttér:

Minden hexadecimális számjegy 4 bitet jelent.



Be szeretnél állítani néhány bitet, úgy, hogy a többi ne változzon? Használj bitenkénti **vagy** műveletet!

Kiindulási érték:

`0101001100001111`

Elvárt érték:

`0101001101101111`

Szükséges kód:

```
short int N = 0x530F;
short int M = 0x0060;
// 0000 0000 0110 0000
N = N|M;
```

Matematikai háttér:

$x \vee F = x$ változatlan

$x \vee T = T$ beállítódik

Törölni szeretnél néhány bitet, úgy, hogy a többi ne változzon? Használj bitenkénti **és** műveletet!

Kiindulási érték:

0101001100001111

Elvárt érték:

0101000001101111

Szükséges kód:

```
short int N = 0x530F;
short int M = 0x0300;
// 0000 0011 0000 0000
N = N&M;
```

Matematikai háttér:

$x \wedge F = F$ törlődik

$x \wedge T = x$ változatlan

Ellentétesre szeretnél állítani néhány bitet, úgy, hogy a többi ne változzon? Használj bitenkénti **kizáró vagy** műveletet!

Kiindulási érték:

0101001100001111

Elvárt érték:

0010100001101111

Szükséges kód:

```
short int N = 0x530F;
short int M = 0x7800;
// 0111 1000 0000 0000
N = N^M;
```

Matematikai háttér:

$x \oplus F = x$ változatlan

$x \oplus T = \bar{x}$ változik

Ellentétesre szeretnél állítani minden bitet?

Használj bitenkénti **tagadás** műveletet!

Kiindulási érték:

`0101001100001111`

Elvárt érték:

`1010110011110000`

Szükséges kód:

```
short int N = 0x530F;
N = ~N; //0xACF0
```

Matematikai háttér:

$\neg x = \bar{x}$ minden bit változik

A teljes bitmintát el akarod tolni, úgy, hogy ne jelenjenek meg új 1-es bitek? Használj bitenkénti **logikai shift** műveletet!

Kiindulási érték:

1010100110001101

Elvárt értékek:

0000001010100110

1100011010000000

Szükséges kód:

```
unsigned short int N,A,B;
N = 0xA98D;
A = N>>6;
B = N<<7;
```

Matematikai háttér:

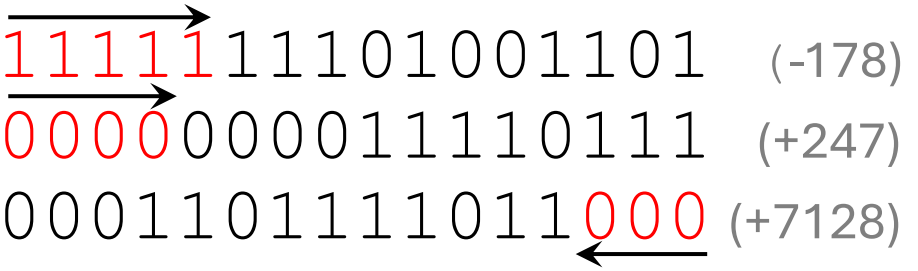
Minden bit elmozdul.
A megüresedő bitpozíciókba 0 bit kerül, ha az eltolás előjel nélküli értéken történik.

A teljes bitmintát el akarod tolni, úgy, hogy az szorzást/osztást eredményezzen? Használj bitenkénti **aritmetikai shift** műveletet!

Kiindulási értékek:

1110100110100110 (-5721)
0000111101111100 (+3964)
0000001101111011 (+891)

Elvárt értékek:



Szükséges kód:

```
short int X,Y,Z;
X=0xE9A6>>5; // -5721/32
Y=0x0F7C>>4; // +3964/16
Z=0x027B<<3; // +891*8
```

Matematikai háttér:

Minden bit elmozdul. Balról előjelbit másolatok kerülnek be, jobbról 0-k. Jobbra tolásnál: az előjel változatlan, a hányados egészre csonkolt.

Összetett példa

Kiindulási érték:

Elvárt érték:

Szükséges kód:

```
unsigned short int N = 0x5B4D;
N = N|0xC000;
N = N&0x1800;
N = N^0x0100;
N = ((N&0x0070)>>4) | ((N&0x0007)<<4) | (N&0xFF88);
```