1. Feladat: Mi lesz **a** és **b** értéke?

```
#include <stdio.h>

int main()
{
    int a = 1;
    int b;
    b = a++;
    printf("b = %d a = %d", b, a);
}
```

2. Feladat: Mi lesz a és b értéke, ha a **++** operátor az **a** változó előtt áll?

```
#include <stdio.h>

int main()
{
    int a = 1;
    int b;
    b = ++a;
    printf("b = %d a = %d", b, a);
}
```

3. Feladat: Mi lesz a **k** változó értéke?

```
#include <stdio.h>

int main()
{
    int i = 1, k = 2;
    int e;
    k *= -i-- * ++k;
    printf("k = %d", k);
}
```

4.a Feladat: Mi lesz az **e** változó értéke?

```
#include <stdio.h>

int main()
{
    int a = 1, b = 2, c = 3, d = 4;
    int e;
    e = !( a <= b && !(c == d));
```

```
    printf("e = %d", e);
}
```

4.b Feladat: Mi lesz az **f** változó értéke?

```
#include <stdio.h>

int main()
{
    int a = 5, b = 3, c = 8;
    int f;
    f = (a > b) || (c < a + b);
    printf("f = %d", f);
}
```

4.c Feladat: Mi lesz az **z** változó értéke?

```
#include <stdio.h>

int main()
{
    int x = 10, y = 7;
    int z;
    z = (x != y) && (y <= x) || !(x - y > 3);
    printf("z = %d", z);
}
```

4.d Feladat: Mi lesz az k változó értéke?

```
#include <stdio.h>

int main()
{
    int m = 4, n = 6;
    int k;
    k = !(m > n) && (m + n > 8) && (n - m == 2);
    printf("k = %d", k);
}
```

5. Logikai operátorok

```
#include <stdio.h>
main()
{
```

```
    // a = 5(00000101)
    // b = 9(00001001)
    unsigned char a = 5, b = 9;

    // 00000001 = 1
    printf("a & b = %d\n", a & b);

    // 00001101 = 13
    printf("a | b = %d\n", a | b);

    // 00001100 = 12
    // XOR akkor 1 ha vagy az egyik, vagy a másik bit 1
    printf("a ^ b = %d\n", a ^ b);

    // 11111010 = 250
    printf("~a = %d\n", a = ~a);

    // 00010010 = 18
    printf("b << 1 = %d\n", b << 1);

    // 00100100 = 36
    printf("b << 2 = %d\n", b << 2);

    // 00000100 = 4
    printf("b >> 1 = %d\n", b >> 1);
}
```

6. műveletek logikai operátorokkal

```
#include <stdio.h>

int main()
{
    // kapcsoljuk be egy int változó 4. bitjét
    int bit = 1 << 3;   // az 1.et eltoljuk hárommal balra, az eredmény:
00001000
    int val = 1;        // tetszőleges szam
    printf("%8b\n", val | bit);
    // kapcsoljuk ki egy int változó 8. bitjét
    int bit2 = 1 << 7;   // az 1.et eltoljuk héttel balra, az eredmény:
10000000
    int val2 = 255;      // tetszőleges szam
    printf("%8b\n", val2 & ~bit2);
    return 0;
}
```

7. az XOR-os titkosítás

```
#include <stdio.h>

int main()
{
    // ha egy számot XOR-ozunk egy tetszőleges számmal 2x-akkor visszakapjuk
az eredeti számot
    int value = 546356243;
    printf("%8b\n", value);
    int code = 112254534;
    int encrypted = value ^ code;
    printf("%8b\n", encrypted);
    printf("%8b\n", encrypted ^ code);
    return 0;
}
```

From:
https://edu.iit.uni-miskolc.hu/ - **Institute of Information Science - University of Miskolc**

Permanent link:
https://edu.iit.uni-miskolc.hu/tanszek:oktatas:szamitastechnika:operatorok

Last update: **2026/10/09 18:42**