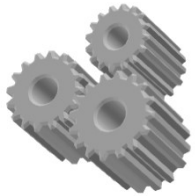
# Kifejezések, operandusok, operátorok

- *Kifejezések a C nyelvben*
- *Operátorok típusai, kiértékelési sorrendje*
  - ***Aritmetikai operátorok***
  - ***Értékadó operátorok***
  - ***Léptető operátorok***
  - ***Relációoperátorok***
  - ***Logikai operátorok***
  - ***A feltételes operátor***
  - ***A címe és a mutató operátor***
  - ***A sizeof operátor***
  - ***A vessző operátor***
  - ***Típuskonvertáló operátor***

# Kifejezések a C nyelvben

- **Kifejezés** *lehet* : *operátorokkal összekapcsolt operandusok, vagy egyetlen operandus. Egy kifejezés kiértékelésének eredménye általában egy érték.*
- **Operandus**, *amelyen az operátor hatása érvényesül, lehet : konstans, változó, függvény, vagy egy kifejezés.*
- **Operátorok**, *műveleti jelek* : *meghatározzák az operandusokkal végzendő műveleteket. Megkülönböztethetünk*
  - ***egy operandusra ható előrevetett (prefix) és hátravetett (postfix) operátorokat,***
  - ***közbeírt (infix) kétoperandusos operátorokat és***
  - ***létezik egy három operandusos operátor is.***



## Értékadó kifejezés

- *A C nyelv jellegzetessége, hogy értékadó kifejezés is létezik, amelyből, bármely más kifejezéshez hasonlóan, egy* **;** *utánaírásával utasítást kapunk. Az értékadó kifejezés alakja:*

***balérték = jobbérték***

*ahol a balérték (lvalue) egy tárolóhelyet jelölő kifejezés, mely egyértelműen beazonosítja a tárolóhely memóriacímét.*

Pl.: *valt1* , *vekt*[ *i* ] , *rekord.mezo* .

*A jobbérték (rvalue) egy kifejezés, amelynek kiértékelésével adódó értéket a balértéknek megfelelő tárolóhelyre kell eltárolni.*

Pl.: 0.625 , *const1*, *valt1*, *vekt*[ *i* ] + 7 .

- ***Az értékadó kifejezés értéke**: az eltárolt érték.*

## Operátorok típusai, kiértékelési sorrendje

( □ ) [ □ ] □.□ □ ->□

!□ -□ ++□ □++ --□ □-- &□ *□ (típus)□ sizeof □

□*□ □/□ □%□

□+□ □-□

□<□ □<=□ □>□ □>=□

□==□ □!=□

□&&□

□||□

□?□:□

□=□ □+=□ □-=□ □*=□ □/=□ □%=□

□,□

Kiértékelés: jobbról balra ill. balról jobbra haladva. □ = operandus

## Operátorok részletezése

**( )** *A zárójelek alkalmazásával az operátorok precedenciája által meghatározott kiértékelési sorrend módosítható:*

5+7*2 ⇒ 19 (5+7) * 2 ⇒ 24

**[ ]** *A tömbök indexének megadására szolgáló zárójelekről a tömbök témakörénél szólunk bővebben.*

**. ->** *A struktúratagokra való hivatkozás operátorait, a pontot és a nyílat a struktúrák témájánál részletezzük.*

## Aritmetikai operátorok

**-□ □+□ □ - □ □ * □ □ / □ □ % □**

**-□** *előjelváltás (negálás)*

Pl.: v = 6; w = -v; /* *w* értéke -6 lesz */

**□+□** *összeadás*

Pl.: **unsigned int** *v* ; /* 0 <= *v* <= 65535 */
*v* = 5+3; /* *v* értéke 8 lesz */
*v* = 65535 + 4; /* *v* értéke 3 lesz !! */

**□ - □** *kivonás*

Pl.: **unsigned int** *v* ; /* 0 <= *v* <= 65535 */
*v* = 125 - 43; /* *v* értéke 82 lesz */
*v* = 43 - 125; /* *v* értéke 65454 lesz! */

## Aritmetikai operátorok ..

-□ □+□ □ - □ □ * □ □ / □ □ % □

□ * □ *szorzás*

```
Pl.: unsigned int v;      /* 0 <= v <= 65535 */
     v = 257 * 256;       /* v értéke 256 lesz!! */
```

□ / □ *osztás, egész típusú operandusok esetén egészosztás*

```
Pl.: f = 42.6 / 5;        /* v értéke 8.52 lesz */
     v = 42 / 5;          /* v értéke 8 lesz */
```

□ % □ *osztás maradéka*

*egész jellegű (egészek, char és enum) értékekre értelmezett.*

```
Pl.: v = 42 % 5;          /* v értéke 2 lesz */
```

## Értékadó operátorok

□ = □ □ += □ □-= □ □*= □ □/= □ □%= □

*A balérték és a jobbérték operandusok az értékadó operátorok egyikével összekapcsolva értékadó kifejezést alkotnak. Ebből következik, hogy a C nyelvben kifejezésekben is kaphatnak új értéket a változók.*

Pl.: *terulet* = (*magassag* = 6) * (*szelesseg* = 9);

*Ennél a nehezen olvasható programszöveget eredményező tömör megoldásnál hasznosabb a jobbról balra történő kiértékelésből eredő többszörös értékadás lehetősége:*

Pl.: *alfa* = *beta* = 90; ⇔ *alfa* = (*beta* = 90) ;

## Értékadó operátorok ..

□ += □ □-= □ □*= □ □/= □ □%= □

*A további összetett értékadó műveletek, melyek gyorsabb programot eredményeznek, az alábbi értékadások egyszerűsített formái:*

| | | |
|---|---|---|
| *valt = valt + kif ;* | ⇔ | *valt += kif ;* |
| *valt = valt - kif ;* | ⇔ | *valt -= kif ;* |
| *valt = valt * kif ;* | ⇔ | *valt *= kif ;* |
| *valt = valt / kif ;* | ⇔ | *valt /= kif ;* |
| *valt = valt % kif ;* | ⇔ | *valt %= kif ;* |

## Léptető operátorok

**++□ □++ --□ □--**

*A léptető operátorok változók (balértékek) értékének eggyel történő növelésére (inkrementálás, ++), illetve csökkentésére (dekrementálás, - -) alkalmasak.*

*++valt* ; ⇔ *valt* = *valt* + 1; ⇔ *valt*++;

*- -valt* ; ⇔ *valt* = *valt* - 1; ⇔ *valt*- -;

*Eredmény: könnyebben olvasható, gyorsabb program.*

## Léptető operátorok .. ++ ++ -- --

*Összetett kifejezésbe ágyazva az operátorokat, figyelembe kell vennünk azt, hogy hatásukat vagy a kifejezés kiértékelése előtt (prefix, előrevetett alak), vagy után (postfix, hátravetett alak) fejtik ki.*

Pl.:

| | | |
|---|---|---|
| `z = ++x * - -y ;` | ⇔ | `++x ; - -y ; z = x * y ;` |
| `z = x++ / y - -;` | ⇔ | `z = x / y ; x++; y- -;` |
| `z = x- - + ++y ;` | ⇔ | `++y ; z = x + y ; x- -;` |
| `z = -x- - - - -y ;` | ⇔ | `- -y ; z = -x - y ; x- -;` |
| `z += ++x + y++` | ⇔ | `++x ; z += (x+y); y++;` |
| `z -= - -x - y- -;` | ⇔ | `- -x ; z -= (x-y); y- -;` |

*A kifejezésbe ágyazott léptető operátorok mellékhatás(side effect)aként a kifejezés kiértékelése előtt, vagy után megváltozik a léptetett változók értéke. Használjunk szóközös tagolást, kétség esetén zárójelet!*

## Relációoperátorok

**< <= > >= == !=**

*A C nyelvben nincs logikai típus.*

*Bármely kifejezés szerepelhet logikai feltételben, ilyenkor a kifejezés nulla értéke a hamis, attól eltérő értéke az igaz értéknek felel meg. Ennek megfelelően a relációk teljesülése 1, hamis volta 0 értéket ad. Az egyenlőség relációjele az* **==** *, a nemegyenlő relációoperátor pedig a* **!=** *.*

Pl.: 25 < 3 ⇔ 0, hamis

45 != 12 ⇔ 1, igaz

## Logikai operátorok !□ □&&□ □||□

*A logikai tagadás operátora a* **!** *, az és művelet operátora az* **&&** *, a vagy műveleté a* **||** *. Igazságtábláik:*

| a | !a |
|---|---|
| 0 | 1 |
| 1 | 0 |

| a && b | b = 0 | b = 1 |
|---|---|---|
| a = 0 | 0 | 0 |
| a = 1 | 0 | 1 |

| a \|\| b | b = 0 | b = 1 |
|---|---|---|
| a = 0 | 0 | 1 |
| a = 1 | 1 | 1 |

*Relációkifejezések logikai operátorral történő összekapcsolásánál nem szükséges zárójelezni, mert a relációk magasabb precedenciájúak.* Pl.: 25 +3 < 45 && 12 > valt1

***Logikai kifejezések rövid, csak az egyértelmű döntéshez szükséges hosszban való kiértékelése megakadályozhatja a nem kiértékelt részben található művelet, vagy mellékhatású operátor működését (rövidzár).***

Pl.: a || (v = 3) .

## A feltételes operátor

□ ? □ : □

*Az egyetlen háromoperandusos operátor az első operandusként megadott feltételkifejezés igaz (nem nulla), vagy hamis (nulla) értékétől függően a : előtti, vagy utáni kifejezés értékét adja a feltételes kifejezés értékeként. A feltételes kifejezés típusa a kettő közül a nagyobb pontosságúéval egyezik.*

Pl.: *x* > 0 **?** *y* **:** -*y*

*minab* = (*a* < *b*) **?** *a* **:** *b* ;

*Zárójelezéssel áttekinthetőbb formát kapunk.*

## A címe és a mutató operátor

&  *

*A címe **&** operátorból és általában egy balérték operandusból álló kifejezés értéke az operandus memóriabeli címe.*

*A mutató * operátorból és egy memóriacímet meghatározó pointerből álló kifejezés értéke a memóriacímen található objektum értéke.*

```
Pl.: int  valt1, *mut1 ;          /* mut1 mutató definiálása */
     valt1 = 28 ;                  /* értékadás a változónak */
     mut1 = &valt1;                /* értékadás a mutatónak */
     *mut1 = *mut1 + 2;            /* hivatkozás a mutatott változó
                                      értékére és a módosított érték
                                      eltárolása a mutatott objektumban */
```

## A sizeof operátor

**sizeof**

*A* sizeof *operátor operandusa egy egyszerű, vagy összetett változó neve, vagy egy zárójelek között megadott típusnév lehet. A kifejezés értéke a változó, vagy a típus memóriabeli helyfoglalását adja meg byte-okban.*

Pl.:
```
int x, x_merete, szohossz, helyfogl ;
unsigned int szo;
x_merete = sizeof x ;          /* x_merete  4 lesz */
szohossz = sizeof szo ;        /* szohossz  4 lesz */
helyfogl = sizeof (long int ); /* helyfogl  8 lesz */
```

## A vessző operátor

□,□

*A vesszővel elválasztott két operandusból álló kifejezés értéke és típusa a jobboldali operandus értékével és típusával egyezik. A vessző operátorral több kifejezést sorolhatunk fel és értékeltethetünk ki olyan helyen, ahol egyébként csak egy kifejezés állhatna. Előnyösen használható pl.* **for** *ciklusban kezdőértékadásra.*

Pl.: *kerulet* = ( *r* = 23.5 **,** 2 * *r* * PI );

*Megjegyzés: A deklarációkban, ill. függvényargumentum-felsorolásban alkalmazott vessző nem operátor, hanem elválasztó karakter.*

- **Típuskonvertáló operátor**
  - **Automatikus (implicit) típuskonverzió**

*A C nyelv típusai eltérhetnek az adatábrázolás pontosságában, értéktartományában, de azonos memóriafoglalású és értékkészletű típusok is lehetnek eltérő nevűek, mint pl. a* **short int** *és az* **int**, *vagy az alapértelmezett* **char** *és a* **signed char**. *Különösen érvényes ez az eltérő adattípusokra mutató pointerek esetében. Olyan eset is előfordulhat, amikor az egyik típus értékkészlete részhalmaza egy másik típus értékkészletének, pl. az* **int** *a* **long int** *típusnak. Az eltérő típusú értékek közötti műveletek elvégzése előtt a típusokat azonos típussá kell konvertálni. Ez a típusátalakítás az esetek nagy részében a beépített automatikus (implicit) típuskonverzió következtében észrevétlenül zajlik. A kisebb értéktartományú és pontosságú típusok automatikus konverziója probléma nélkül végbemegy, pl.* **float** *típusról* **double** *típusra. Azonban a konvertált adat pontossága, vagy akár értéke is romolhat, vagy definiálatlanná is válhat, pl.* **float** *típusról* **int**-*re történő átalakításnál.*

- **Típuskonvertáló operátor (típus)**
  - **Programozott (explicit ) típuskonverzió**

    *Szükséges lehet a típuskonverzió operátorával programozott típuskonverzió olyan esetben, ha*
    - *felül akarjuk bírálni az automatikus konverziót pl. a program gyorsítása céljából, vagy*
    - *a típuseltérés hibájának elkerülésére, ha egy függvény adott típusú paramétert vár, továbbá*
    - *programozott konverziót igényel eltérő típusra mutató pointerek címtartalmának átadása.*

    *Ilyen esetben az átalakítandó típusú kifejezésre alkalmazzuk a típuskonverzió operátorát :*

    (*típus*) *kifejezés*

    *alakban. Az eredményül kapott kifejezésérték a zárójelben megadott típusú lesz.*

- **Típuskonvertáló operátor .. (típus)**

Pl.:
```
int fok ;
double v ;
int * egeszremutato ;
char * karakterremutato ;
fok = 85;   v = sin( (double) fok );
egeszremutato = &fok ;
karakterremutato = (char*) egeszremutato ;
```