Óbudai Egyetem
Kandó Kálmán Villamosmérnöki Kar

C programozási nyelv
Operátorok

Dr. Schuster György

2017. december 27.

Operátor csoportok

Operátor csoportok

1. elválasztó perátorok,

Operátor csoportok

1. elválasztó perátorok,
2. értékadó operátorok,

Operátor csoportok

1. elválasztó perátorok,
2. értékadó operátorok,
3. aritmetikai operátorok,

Operátor csoportok

1. elválasztó perátorok,
2. értékadó operátorok,
3. aritmetikai operátorok,
4. relációs operátorok,

Operátor csoportok

1. elválasztó perátorok,
2. értékadó operátorok,
3. aritmetikai operátorok,
4. relációs operátorok,
5. logikai operátorok,

Operátor csoportok

1. elválasztó perátorok,
2. értékadó operátorok,
3. aritmetikai operátorok,
4. relációs operátorok,
5. logikai operátorok,
6. bit operátorok,

Operátor csoportok

1. elválasztó perátorok,
2. értékadó operátorok,
3. aritmetikai operátorok,
4. relációs operátorok,
5. logikai operátorok,
6. bit operátorok,
7. egyéb operátorok.

1. elválasztó operátorok

1. elválasztó operátorok

{ } valamilyen logikai blokkot fog közre,

1. elválasztó operátorok

{ } valamilyen logikai blokkot fog közre, ez lehet:

- függvény törzs,

1. elválasztó operátorok

{ } valamilyen logikai blokkot fog közre, ez lehet:

- függvény törzs,
- ciklus törzs,

1. elválasztó operátorok

{ } valamilyen logikai blokkot fog közre, ez lehet:

- függvény törzs,
- ciklus törzs,
- igaz - hamis ág,

1. elválasztó operátorok

{ } valamilyen logikai blokkot fog közre, ez lehet:

- függvény törzs,
- ciklus törzs,
- igaz - hamis ág,
- valamilyen szerkezet határolója

1. elválasztó operátorok

`{ }` valamilyen logikai blokkot fog közre, ez lehet:

- függvény törzs,
- ciklus törzs,
- igaz - hamis ág,
- valamilyen szerkezet határolója (pl. `switch - case` szerkezet),

`[ ]` tömbelemek indexelésénél használjuk,

1. elválasztó operátorok

{ } valamilyen logikai blokkot fog közre, ez lehet:

- függvény törzs,
- ciklus törzs,
- igaz - hamis ág,
- valamilyen szerkezet határolója (pl. `switch - case` szerkezet),

[ ] tömbelemek indexelésénél használjuk, később

1. elválasztó operátorok

{ } valamilyen logikai blokkot fog közre, ez lehet:

- függvény törzs,
- ciklus törzs,
- igaz - hamis ág,
- valamilyen szerkezet határolója (pl. `switch - case` szerkezet),

[ ] tömbelemek indexelésénél használjuk, később

( ) végrehajtási sorrendet megváltoztató operátorpár,

1. elválasztó operátorok

`{ }` valamilyen logikai blokkot fog közre, ez lehet:

- függvény törzs,
- ciklus törzs,
- igaz - hamis ág,
- valamilyen szerkezet határolója (pl. `switch - case` szerkezet),

`[ ]` tömbelemek indexelésénél használjuk, később

`( )` végrehajtási sorrendet megváltoztató operátorpár, pl. `d=a+b*c;` `d=(a+b)*c;`

1. elválasztó operátorok

`{ }` valamilyen logikai blokkot fog közre, ez lehet:

- függvény törzs,
- ciklus törzs,
- igaz - hamis ág,
- valamilyen szerkezet határolója (pl. `switch - case` szerkezet),

`[ ]` tömbelemek indexelésénél használjuk, később

`( )` végrehajtási sorrendet megváltoztató operátorpár, pl. `d=a+b*c;` `d=(a+b)*c;`

`,` kifejezéseket elválasztó operátor,

1. elválasztó operátorok

`{ }` valamilyen logikai blokkot fog közre, ez lehet:

- függvény törzs,
- ciklus törzs,
- igaz - hamis ág,
- valamilyen szerkezet határolója (pl. `switch - case` szerkezet),

`[ ]` tömbelemek indexelésénél használjuk, később

`( )` végrehajtási sorrendet megváltoztató operátorpár, pl. `d=a+b*c; d=(a+b)*c;`

`,` kifejezéseket elválasztó operátor, pl. `a=b+c,d+e;`

1. elválasztó operátorok

`{ }` valamilyen logikai blokkot fog közre, ez lehet:

- függvény törzs,
- ciklus törzs,
- igaz - hamis ág,
- valamilyen szerkezet határolója (pl. `switch - case` szerkezet),

`[ ]` tömbelemek indexelésénél használjuk, később

`( )` végrehajtási sorrendet megváltoztató operátorpár, pl. `d=a+b*c; d=(a+b)*c;`

`,` kifejezéseket elválasztó operátor, pl. `a=b+c,d+e;`

`;` kifejezéseket lezáró operátor,

1. elválasztó operátorok

`{ }` valamilyen logikai blokkot fog közre, ez lehet:

- függvény törzs,
- ciklus törzs,
- igaz - hamis ág,
- valamilyen szerkezet határolója (pl. `switch - case` szerkezet),

`[ ]` tömbelemek indexelésénél használjuk, később

`( )` végrehajtási sorrendet megváltoztató operátorpár, pl. `d=a+b*c; d=(a+b)*c;`

`,` kifejezéseket elválasztó operátor, pl. `a=b+c,d+e;`

`;` kifejezéseket lezáró operátor, pl. `a=b+c;`

2. értékadó operátorok

2. értékadó operátorok

- = értékadó operátor,

2. értékadó operátorok

= értékadó operátor,
Balérték szabály

2. értékadó operátorok

- = értékadó operátor,
  Balérték szabály
  `a=b+c;` **helyes!**

2. értékadó operátorok

- = értékadó operátor,
  Balérték szabály
  `a=b+c;` **helyes!**
  `b+c=a;` **HELYTELEN!!**

2. értékadó operátorok

- `=` értékadó operátor,
  Balérték szabály
  `a=b+c;` **helyes!**
  `b+c=a;` **HELYTELEN!!**
- `?:` feltételes értékadás,

2. értékadó operátorok

`=` értékadó operátor,

Balérték szabály

`a=b+c;` **helyes!**

`b+c=a;` **HELYTELEN!!**

`?:` feltételes értékadás,

```
ered = kif1 ?  kif2 :  kif3
```

2. értékadó operátorok

`=` értékadó operátor,
Balérték szabály
`a=b+c;` **helyes!**
`b+c=a;` **HELYTELEN!!**

`?:` feltételes értékadás,
`ered = kif1 ? kif2 : kif3`
ha `kif1` igaz, akkor az `ered` `kif2` lesz,

2. értékadó operátorok

`=` értékadó operátor,
Balérték szabály
`a=b+c;` **helyes!**
`b+c=a;` **HELYTELEN!!**

`?:` feltételes értékadás,
`ered = kif1 ? kif2 : kif3`
ha `kif1` igaz, akkor az `ered` `kif2` lesz,
ha `kif1` hamis, akkor az `ered` `kif3` lesz.

3. aritmetikai operátorok

`+, -, *, /` klasszikus aritmetikai műveletek operátorai,

3. aritmetikai operátorok

`+, -, *, /` klasszikus aritmetikai műveletek operátorai,

`%` maradékképző operátor,

3. aritmetikai operátorok

`+, -, *, /` klasszikus aritmetikai műveletek operátorai,

`%` maradékképző operátor, példa:

3. aritmetikai operátorok

`+, -, *, /` klasszikus aritmetikai műveletek operátorai,

`%` maradékképző operátor, példa:

```
int a=31,b=4,c;
c=a%b;
```

3. aritmetikai operátorok

`+, -, *, /` klasszikus aritmetikai műveletek operátorai,

`%` maradékképző operátor, példa:

```
int a=31,b=4,c;
c=a%b;
```

`c` értéke: 3

3. aritmetikai operátorok

`+, -, *, /` klasszikus aritmetikai műveletek operátorai,

`%` maradékképző operátor, példa:

```
int a=31,b=4,c;
c=a%b;
```

`c` értéke: 3

Figyelem! Csak egész jellegű változókon használható!

3. aritmetikai operátorok

`+, -, *, /` klasszikus aritmetikai műveletek operátorai,

`%` maradékképző operátor, példa:

```
int a=31,b=4,c;
c=a%b;
```

`c` értéke: 3

Figyelem! Csak egész jellegű változókon használható!

– egyoperandusú mínusz operátor,

3. aritmetikai operátorok

`+, -, *, /` klasszikus aritmetikai műveletek operátorai,

`%` maradékképző operátor, példa:

```
int a=31,b=4,c;
c=a%b;
```

`c` értéke: 3

Figyelem! Csak egész jellegű változókon használható!

– egyoperandusú mínusz operátor, példa:

3. aritmetikai operátorok

`+, -, *, /` klasszikus aritmetikai műveletek operátorai,

`%` maradékképző operátor, példa:

```
int a=31,b=4,c;
c=a%b;
```

`c` értéke: 3

Figyelem! Csak egész jellegű változókon használható!

`-` egyoperandusú mínusz operátor, példa:

```
a=-b;
```

3. aritmetikai operátorok

`+, -, *, /` klasszikus aritmetikai műveletek operátorai,

`%` maradékképző operátor, példa:

```
int a=31,b=4,c;
c=a%b;
```

`c` értéke: 3

Figyelem! Csak egész jellegű változókon használható!

`-` egyoperandusú mínusz operátor, példa:

```
a=-b;
```

Különböző típusú változók esetén a művelet a magasabb tárolási osztályban hajtódik végre.

4. relációs operátorok

4. relációs operátorok

Két változó összehasonlítását végzi.

4. relációs operátorok

Két változó összehasonlítását végzi.
Az eredmény lehet:

4. relációs operátorok

Két változó összehasonlítását végzi.
Az eredmény lehet:

- igaz,

4. relációs operátorok

Két változó összehasonlítását végzi.
Az eredmény lehet:

- igaz,
- hamis.

4. relációs operátorok

Két változó összehasonlítását végzi.
Az eredmény lehet:

- igaz,
- hamis.

  - $<$ kisebb operátor,

4. relációs operátorok

Két változó összehasonlítását végzi.
Az eredmény lehet:

- igaz,
- hamis.

`<` kisebb operátor,

`<=` kisebb - egyenlő operátor,

4. relációs operátorok

Két változó összehasonlítását végzi.
Az eredmény lehet:

- igaz,
- hamis.

`<`	kisebb operátor,
`<=`	kisebb - egyenlő operátor,
`==`	egyenlő operátor,

4. relációs operátorok

Két változó összehasonlítását végzi.
Az eredmény lehet:

- igaz,
- hamis.

$<$	kisebb operátor,
$<=$	kisebb - egyenlő operátor,
$==$	egyenlő operátor,
$!=$	nem egyenlő operátor,

4. relációs operátorok

Két változó összehasonlítását végzi.
Az eredmény lehet:

- igaz,
- hamis.

`<`	kisebb operátor,
`<=`	kisebb - egyenlő operátor,
`==`	egyenlő operátor,
`!=`	nem egyenlő operátor,
`>=`	nagyobb - egyenlő operátor,

4. relációs operátorok

Két változó összehasonlítását végzi.
Az eredmény lehet:

- igaz,
- hamis.

`<`	kisebb operátor,
`<=`	kisebb - egyenlő operátor,
`==`	egyenlő operátor,
`!=`	nem egyenlő operátor,
`>=`	nagyobb - egyenlő operátor,
`>`	nagyobb operátor.

4. relációs operátorok

Két változó összehasonlítását végzi.
Az eredmény lehet:

- igaz,
- hamis.

`<`	kisebb operátor,
`<=`	kisebb - egyenlő operátor,
`==`	egyenlő operátor,
`!=`	nem egyenlő operátor,
`>=`	nagyobb - egyenlő operátor,
`>`	nagyobb operátor.

Használatát lásd később az utasításoknál.

5. logikai operátorok

5. logikai operátorok

Relációs jellegű kifejezéseket hoz logikai kapcsolatba, illetőleg módosít.

5. logikai operátorok

Relációs jellegű kifejezéseket hoz logikai kapcsolatba, illetőleg módosít.

- `&&` logikai és operátor,

5. logikai operátorok

Relációs jellegű kifejezéseket hoz logikai kapcsolatba, illetőleg módosít.

- `&&` logikai és operátor,
- `||` logikai vagy operátor,

5. logikai operátorok

Relációs jellegű kifejezéseket hoz logikai kapcsolatba, illetőleg módosít.

- `&&` logikai és operátor,
- `||` logikai vagy operátor,
- `!` logikai nem operátor,

5. logikai operátorok

Relációs jellegű kifejezéseket hoz logikai kapcsolatba, illetőleg módosít.

- `&&` logikai és operátor,
- `||` logikai vagy operátor,
- `!` logikai nem operátor,

A művelet addig megy, amíg az eredmény nem biztos.

5. logikai operátorok

Relációs jellegű kifejezéseket hoz logikai kapcsolatba, illetőleg módosít.

- `&&` logikai és operátor,
- `||` logikai vagy operátor,
- `!` logikai nem operátor,

A művelet addig megy, amíg az eredmény nem biztos.
Példa:

5. logikai operátorok

Relációs jellegű kifejezéseket hoz logikai kapcsolatba, illetőleg módosít.

- `&&` logikai és operátor,
- `||` logikai vagy operátor,
- `!` logikai nem operátor,

A művelet addig megy, amíg az eredmény nem biztos.

Példa:

```
kif1 && kif2 && kif3
```

5. logikai operátorok

Relációs jellegű kifejezéseket hoz logikai kapcsolatba, illetőleg módosít.

- `&&` logikai és operátor,
- `||` logikai vagy operátor,
- `!` logikai nem operátor,

A művelet addig megy, amíg az eredmény nem biztos.

Példa:

```
kif1 && kif2 && kif3
```

Ha `kif1` igaz, akkor `kif2` is végre hajtódik.

Ha ekkor `kif2` is igaz, akkor `kif3` is végrehajtásra kerül.

5. logikai operátorok

Relációs jellegű kifejezéseket hoz logikai kapcsolatba, illetőleg módosít.

- `&&` logikai és operátor,
- `||` logikai vagy operátor,
- `!` logikai nem operátor,

A művelet addig megy, amíg az eredmény nem biztos.

Példa:

```
kif1 && kif2 && kif3
```

Ha `kif1` igaz, akkor `kif2` is végre hajtódik.

Ha ekkor `kif2` is igaz, akkor `kif3` is végrehajtásra kerül.

Ha már `kif1` hamis, akkor a többi nem hajtódik végre.

6. bit operátorok

6. bit operátorok

Csak egész jellegű változókon hajtható végre!!

6. bit operátorok

Csak egész jellegű változókon hajtható végre!!

- ~ egyes komplemens képzés,

6. bit operátorok

Csak egész jellegű változókon hajtható végre!!

- ~ egyes komplemens képzés, példa:

6. bit operátorok

Csak egész jellegű változókon hajtható végre!!

- ~ egyes komplemens képzés, példa:

```
char a,b=0b00110101;
a=~b;
```

6. bit operátorok

Csak egész jellegű változókon hajtható végre!!

- `~` egyes komplemens képzés, példa:

```
char a,b=0b00110101;
a=~b;
```

A művelet:

6. bit operátorok

Csak egész jellegű változókon hajtható végre!!

- ~ egyes komplemens képzés, példa:

```
char a,b=0b00110101;
a=~b;
```

A művelet:

```
b: 00110101
a: 11001010
```

6. bit operátorok

Csak egész jellegű változókon hajtható végre!!

- `~` egyes komplemens képzés, példa:

```
char a,b=0b00110101;
a=~b;
```

A művelet:

```
b: 00110101
a: 11001010
```

- `&` bitenkénti és kapcsolat,

6. bit operátorok

Csak egész jellegű változókon hajtható végre!!

- `~` egyes komplemens képzés, példa:

```
char a,b=0b00110101;
a=~b;
```

A művelet:

```
b: 00110101
a: 11001010
```

- `&` bitenkénti és kapcsolat, példa:

6. bit operátorok

Csak egész jellegű változókon hajtható végre!!

- ~ egyes komplemens képzés, példa:

```
char a,b=0b00110101;
a=~b;
```

A művelet:

```
b: 00110101
a: 11001010
```

- & bitenkénti és kapcsolat, példa:

```
char a,b=0b00110101,c=0b00001111;
a=b & c;
```

6. bit operátorok

Csak egész jellegű változókon hajtható végre!!

- ~ egyes komplemens képzés, példa:

```
char a,b=0b00110101;
a=~b;
```

A művelet:

```
b: 00110101
a: 11001010
```

- & bitenkénti és kapcsolat, példa:

```
char a,b=0b00110101,c=0b00001111;
a=b & c;
```

A művelet:

6. bit operátorok

Csak egész jellegű változókon hajtható végre!!

- ~ egyes komplemens képzés, példa:

```
char a,b=0b00110101;
a=~b;
```

A művelet:

```
b: 00110101
a: 11001010
```

- & bitenkénti és kapcsolat, példa:

```
char a,b=0b00110101,c=0b00001111;
a=b & c;
```

A művelet:

```
b: 00110101
c: 00001111
a: 00000101
```

6. bit operátorok

- bitenkénti vagy kapcsolat,

6. bit operátorok

- bitenkénti vagy kapcsolat, példa:

6. bit operátorok

- bitenkénti vagy kapcsolat, példa:

```
char a,b=0b00110101,c=0b00001111;
a=b | c;
```

6. bit operátorok

- bitenkénti vagy kapcsolat, példa:

```
char a,b=0b00110101,c=0b00001111;
a=b | c;
```

A művelet:

6. bit operátorok

- bitenkénti vagy kapcsolat, példa:

```
char a,b=0b00110101,c=0b00001111;
a=b | c;
```

A művelet:

```
b: 00110101
c: 00001111
a: 00111111
```

- bitenkénti kizáró vagy kapcsolat,

6. bit operátorok

- bitenkénti vagy kapcsolat, példa:

```
char a,b=0b00110101,c=0b00001111;
a=b | c;
```

A művelet:

```
b: 00110101
c: 00001111
a: 00111111
```

- bitenkénti kizáró vagy kapcsolat, példa:

6. bit operátorok

- bitenkénti vagy kapcsolat, példa:

```
char a,b=0b00110101,c=0b00001111;
a=b | c;
```

A művelet:

```
b: 00110101
c: 00001111
a: 00111111
```

- bitenkénti kizáró vagy kapcsolat, példa:

```
char a,b=0b00110101,c=0b00001111;
a=b ^ c;
```

6. bit operátorok

- bitenkénti vagy kapcsolat, példa:

```
char a,b=0b00110101,c=0b00001111;
a=b | c;
```

A művelet:

```
b: 00110101
c: 00001111
a: 00111111
```

- bitenkénti kizáró vagy kapcsolat, példa:

```
char a,b=0b00110101,c=0b00001111;
a=b ^ c;
```

A művelet:

6. bit operátorok

- bitenkénti vagy kapcsolat, példa:

```
char a,b=0b00110101,c=0b00001111;
a=b | c;
```

A művelet:

```
b: 00110101
c: 00001111
a: 00111111
```

- bitenkénti kizáró vagy kapcsolat, példa:

```
char a,b=0b00110101,c=0b00001111;
a=b ^ c;
```

A művelet:

```
b: 00110101
c: 00001111
a: 00111010
```

6. bit operátorok

- `<<` balrasiftelés operátor,

6. bit operátorok

- `<<` balrasiftelés operátor, példa:

6. bit operátorok

`<<` balrasiftelés operátor, példa:

```
char a,b=0b00011000,c=2;
a=b << c;
```

6. bit operátorok

`<<` balrasiftelés operátor, példa:

```
char a,b=0b00011000,c=2;
a=b << c;
```

A művelet:

6. bit operátorok

<< balrasiftelés operátor, példa:

```
char a,b=0b00011000,c=2;
a=b << c;
```

A művelet:

```
b: 00011000 <-00
a: 01100000
```

6. bit operátorok

<< balrasiftelés operátor, példa:

```
char a,b=0b00011000,c=2;
a=b << c;
```

A művelet:

```
b: 00011000 <-00
a: 01100000
```

>> jobbrasiftelés operátor,

6. bit operátorok

<< balrasiftelés operátor, példa:

```
char a,b=0b00011000,c=2;
a=b << c;
```

A művelet:

```
b: 00011000 <-00
a: 01100000
```

>> jobbrasiftelés operátor, példa:

6. bit operátorok

`<<` balrasiftelés operátor, példa:

```
char a,b=0b00011000,c=2;
a=b << c;
```

A művelet:

```
b: 00011000 <-00
a: 01100000
```

`>>` jobbrasiftelés operátor, példa:

```
char a,b=0b00011000,c=2;
a=b >> c;
```

6. bit operátorok

<< balrasiftelés operátor, példa:

```
char a,b=0b00011000,c=2;
a=b << c;
```

A művelet:

```
b: 00011000 <-00
a: 01100000
```

>> jobbrasiftelés operátor, példa:

```
char a,b=0b00011000,c=2;
a=b >> c;
```

A művelet:

6. bit operátorok

`<<` balrasiftelés operátor, példa:

```
char a,b=0b00011000,c=2;
a=b << c;
```

A művelet:

```
b: 00011000 <-00
a: 01100000
```

`>>` jobbrasiftelés operátor, példa:

```
char a,b=0b00011000,c=2;
a=b >> c;
```

A művelet:

```
b: 00-> 00011000
a:      00000110
```

7. egyéb operátorok

`rekurzív` bármilyen eredményt előállító kétoperandusú operátor esetén alkalmazható forma.

7. egyéb operátorok

`rekurzív` bármilyen eredményt előállító kétoperandusú operátor esetén alkalmazható forma. Példa:

7. egyéb operátorok

`rekurzív` bármilyen eredményt előállító kétoperandusú operátor esetén alkalmazható forma. Példa:
`a=a+b;` helyett írható: `a+=b;`

7. egyéb operátorok

`rekurzív` bármilyen eredményt előállító kétoperandusú operátor esetén alkalmazható forma. Példa:
`a=a+b;` helyett írható: `a+=b;`
Figyelem hibalehetőség!!

7. egyéb operátorok

`rekurzív` bármilyen eredményt előállító kétoperandusú operátor esetén alkalmazható forma. Példa:

`a=a+b;` helyett írható: `a+=b;`

Figyelem hibalehetőség!!

`a=b-a;` esetén nem alkalmazható!!

7. egyéb operátorok

`rekurzív` bármilyen eredményt előállító kétoperandusú operátor esetén alkalmazható forma. Példa:
`a=a+b;` helyett írható: `a+=b;`
Figyelem hibalehetőség!!
`a=b-a;` esetén nem alkalmazható!!
Leggyakrabban használt formák:

7. egyéb operátorok

`rekurzív` bármilyen eredményt előállító kétoperandusú operátor esetén alkalmazható forma. Példa:

`a=a+b;` helyett írható: `a+=b;`

Figyelem hibalehetőség!!

`a=b-a;` esetén nem alkalmazható!!

Leggyakrabban használt formák:

`a<<=b;` balra shiftelés,

7. egyéb operátorok

`rekurzív` bármilyen eredményt előállító kétoperandusú operátor esetén alkalmazható forma. Példa:

`a=a+b;` helyett írható: `a+=b;`

Figyelem hibalehetőség!!

`a=b-a;` esetén nem alkalmazható!!

Leggyakrabban használt formák:

`a<<=b;` balra shiftelés,

`a>>=b;` jobbra shiftelés,

7. egyéb operátorok

`rekurzív` bármilyen eredményt előállító kétoperandusú operátor esetén alkalmazható forma. Példa:

`a=a+b;` helyett írható: `a+=b;`

Figyelem hibalehetőség!!

`a=b-a;` esetén nem alkalmazható!!

Leggyakrabban használt formák:

`a<<=b;` balra shiftelés,

`a>>=b;` jobbra shiftelés,

`a&=b;` bit ”törlés”,

7. egyéb operátorok

`rekurzív` bármilyen eredményt előállító kétoperandusú operátor esetén alkalmazható forma. Példa:

`a=a+b;` helyett írható: `a+=b;`

Figyelem hibalehetőség!!

`a=b-a;` esetén nem alkalmazható!!

Leggyakrabban használt formák:

`a<<=b;` balra shiftelés,

`a>>=b;` jobbra shiftelés,

`a&=b;` bit ”törlés”,

`a|=b;` bit ”rámaszkolás”,

7. egyéb operátorok

++ inkrementáló operátor,

7. egyéb operátorok

`++` inkrementáló operátor, működés:
`a++;` megfelel `a=a+1;` kifejezésnek.

7. egyéb operátorok

`++` inkrementáló operátor, működés:
`a++;` megfelel `a=a+1;` kifejezésnek.
Használható `a++;` és

7. egyéb operátorok

`++` inkrementáló operátor, működés:
`a++;` megfelel `a=a+1;` kifejezésnek.
Használható `a++;` és `++a;` formában.

7. egyéb operátorok

`++` inkrementáló operátor, működés:
`a++;` megfelel `a=a+1;` kifejezésnek.
Használható `a++;` és `++a;` formában. Mi a különbség?

7. egyéb operátorok

`++` inkrementáló operátor, működés:
`a++;` megfelel `a=a+1;` kifejezésnek.
Használható `a++;` és `++a;` formában. Mi a különbség?
Példa:

```
int i=5,j;

j=i++;
```

```
int i=5,j;

j=++i;
```

7. egyéb operátorok

`++` inkrementáló operátor, működés:
`a++;` megfelel `a=a+1;` kifejezésnek.
Használható `a++;` és `++a;` formában. Mi a különbség?
Példa:

```
int i=5,j;

j=i++;

 j: 5
 i: 6
```

```
int i=5,j;

j=++i;

 j: 6
 i: 6
```

7. egyéb operátorok

- `--` dekrementáló operátor,

7. egyéb operátorok

`--` dekrementáló operátor, működés:
`a--;` megfelel `a=a-1;` kifejezésnek.

7. egyéb operátorok

`--` dekrementáló operátor, működés:
`a--;` megfelel `a=a-1;` kifejezésnek.
Használható `a--;` és

7. egyéb operátorok

`--` dekrementáló operátor, működés:
`a--;` megfelel `a=a-1;` kifejezésnek.
Használható `a--;` és `--a;` formában.

7. egyéb operátorok

`--` dekrementáló operátor, működés:
`a--;` megfelel `a=a-1;` kifejezésnek.
Használható `a--;` és `--a;` formában. Mi a különbség?

7. egyéb operátorok

- `--` dekrementáló operátor, működés:
  `a--;` megfelel `a=a-1;` kifejezésnek.
  Használható `a--;` és `--a;` formában. Mi a különbség?
  Példa:

```
int i=5,j;

j=i--;
```

```
int i=5,j;

j=--i;
```

7. egyéb operátorok

- `--` dekrementáló operátor, működés:
  `a--;` megfelel `a=a-1;` kifejezésnek.
  Használható `a--;` és `--a;` formában. Mi a különbség?
  Példa:

```
int i=5,j;

j=i--;
```

```
 j: 5
 i: 4
```

```
int i=5,j;

j=--i;
```

```
 j: 4
 i: 4
```

7. egyéb operátorok

`(cast)` kényszerített típuskonverzió.

7. egyéb operátorok

`(cast)` kényszerített típuskonverzió. Adott műveletben, az adott helyen a kérédses változó típusát a kívánt típusra változtatja.

7. egyéb operátorok

`(cast)` kényszerített típuskonverzió. Adott műveletben, az adott helyen a kérédses változó típusát a kívánt típusra változtatja. Példa:

7. egyéb operátorok

`(cast)` kényszerített típuskonverzió. Adott műveletben, az adott helyen a kérédses változó típusát a kívánt típusra változtatja. Példa:

```
int i=5,j=6;
double d;

d=(double)i/j;
```

7. egyéb operátorok

`(cast)` kényszerített típuskonverzió. Adott műveletben, az adott helyen a kérédses változó típusát a kívánt típusra változtatja. Példa:

```
int i=5,j=6;
double d;

d=(double)i/j;
```

```
d: 0.83333333
```

7. egyéb operátorok

`(cast)` kényszerített típuskonverzió. Adott műveletben, az adott helyen a kérédses változó típusát a kívánt típusra változtatja. Példa:

```
int i=5,j=6;
double d;

d=(double)i/j;
```

```
d: 0.83333333
```

Tipikus hiba!!

7. egyéb operátorok

`(cast)` kényszerített típuskonverzió. Adott műveletben, az adott helyen a kérédses változó típusát a kívánt típusra változtatja. Példa:

```
int i=5,j=6;
double d;

d=(double)i/j;
```

```
d: 0.83333333
```

Tipikus hiba!!

```
d=(double)(i/j);
```

```
d: 0
```

7. egyéb operátorok

`sizeof()` egy változó, vagy egy típus méretét adja meg bájt méretben.

7. egyéb operátorok

`sizeof()` egy változó, vagy egy típus méretét adja meg bájt méretben. Példa:

```
int i,s;

s=sizeof(int);
```

7. egyéb operátorok

`sizeof()` egy változó, vagy egy típus méretét adja meg bájt méretben. Példa:

```
int i,s;

s=sizeof(int);
```

vagy

7. egyéb operátorok

`sizeof()` egy változó, vagy egy típus méretét adja meg bájt méretben. Példa:

```
int i,s;

s=sizeof(int);
```

vagy

```
s=sizeof(i);
```

7. egyéb operátorok

A további operátorok később krülnek ismertetésre.

- `&` címképző operátor (lásd tömbök),
- `*` indirdekciós operátor (lásd tömbök),
- `.` mezőhozzáférés operátor (lásd struktúrák),
- `->` indirekt mezőhozzáférés operátor (lásd struktúrák).

Operátorok precedenciája

legerősebb	{} () [] -> .	b j
	! ++ -- + - (cast) * & sizeof()	j b
	* / %	b j
	+ -	b j
	>> <<	b j
	> < < = > =	b j
	== !=	b j
	&	b j
	^	b j
	|	b j
	&&	b j
	||	b j
	?:	j b
	= += -= ...	j b
leggyengébb	,	b j

Kérdések

Mi a szerepe a {} operátor párnak?

Kérdések

Mi a szerepe a {} operátor párnak?
Valamilyen logikai blokkot határol.

Kérdések

Mi a szerepe a {} operátor párnak?
Valamilyen logikai blokkot határol.
Mikor használjuk a `;` operátort?

Kérdések

Mi a szerepe a {} operátor párnak?
Valamilyen logikai blokkot határol.
Mikor használjuk a ; operátort?
Minden ”befejezett” kifejezés végén, de vannak kivételek.

Kérdések

Mi a szerepe a {} operátor párnak?
Valamilyen logikai blokkot határol.
Mikor használjuk a ; operátort?
Minden ”befejezett” kifejezés végén, de vannak kivételek.
Mit jelent a balérték szabály?

Kérdések

Mi a szerepe a {} operátor párnak?
Valamilyen logikai blokkot határol.
Mikor használjuk a ; operátort?
Minden ”befejezett” kifejezés végén, de vannak kivételek.
Mit jelent a balérték szabály?
A művelet során a bal oldalon lévő változó kap értéket a jobb oldalon lévő kifejezéstől.

Kérdések

Mi a szerepe a {} operátor párnak?
Valamilyen logikai blokkot határol.
Mikor használjuk a ; operátort?
Minden ”befejezett” kifejezés végén, de vannak kivételek.
Mit jelent a balérték szabály?
A művelet során a bal oldalon lévő változó kap értéket a jobb oldalon lévő kifejezéstől.
Lehet-e két különböző típusú változón aritmetikai műveletet végezni?

Kérdések

Mi a szerepe a {} operátor párnak?
Valamilyen logikai blokkot határol.
Mikor használjuk a ; operátort?
Minden ”befejezett” kifejezés végén, de vannak kivételek.
Mit jelent a balérték szabály?
A művelet során a bal oldalon lévő változó kap értéket a jobb oldalon lévő kifejezéstől.
Lehet-e két különböző típusú változón aritmetikai műveletet végezni?
Lehet abban az esetben, ha a változón lehetséges művelet. Ekkor az művelet a magasabb számábrázolási formában (tárolási osztályban) kerül végrehajtásra.

Kérdések

Mi a szerepe a {} operátor párnak?
Valamilyen logikai blokkot határol.
Mikor használjuk a `;` operátort?
Minden ”befejezett” kifejezés végén, de vannak kivételek.
Mit jelent a balérték szabály?
A művelet során a bal oldalon lévő változó kap értéket a jobb oldalon lévő kifejezéstől.
Lehet-e két különböző típusú változón aritmetikai műveletet végezni?
Lehet abban az esetben, ha a változón lehetséges művelet. Ekkor az művelet a magasabb számábrázolási formában (tárolási osztályban) kerül végrehajtásra.
Milyen szabály vonatkozik a `%` operátorra?

Kérdések

Mi a szerepe a {} operátor párnak?
Valamilyen logikai blokkot határol.
Mikor használjuk a `;` operátort?
Minden ”befejezett” kifejezés végén, de vannak kivételek.
Mit jelent a balérték szabály?
A művelet során a bal oldalon lévő változó kap értéket a jobb oldalon lévő kifejezéstől.
Lehet-e két különböző típusú változón aritmetikai műveletet végezni?
Lehet abban az esetben, ha a változón lehetséges művelet. Ekkor az művelet a magasabb számábrázolási formában (tárolási osztályban) kerül végrehajtásra.
Milyen szabály vonatkozik a `%` operátorra?
Csak egész jellegű változókra alkalmazható?

Kérdések

Mi a szerepe a {} operátor párnak?
Valamilyen logikai blokkot határol.
Mikor használjuk a `;` operátort?
Minden ”befejezett” kifejezés végén, de vannak kivételek.
Mit jelent a balérték szabály?
A művelet során a bal oldalon lévő változó kap értéket a jobb oldalon lévő kifejezéstől.
Lehet-e két különböző típusú változón aritmetikai műveletet végezni?
Lehet abban az esetben, ha a változón lehetséges művelet. Ekkor az művelet a magasabb számábrázolási formában (tárolási osztályban) kerül végrehajtásra.
Milyen szabály vonatkozik a `%` operátorra?
Csak egész jellegű változókra alkalmazható?
Mit csinála `!=` operátor?

Kérdések

Mi a szerepe a {} operátor párnak?
Valamilyen logikai blokkot határol.
Mikor használjuk a `;` operátort?
Minden ”befejezett” kifejezés végén, de vannak kivételek.
Mit jelent a balérték szabály?
A művelet során a bal oldalon lévő változó kap értéket a jobb oldalon lévő kifejezéstől.
Lehet-e két különböző típusú változón aritmetikai műveletet végezni?
Lehet abban az esetben, ha a változón lehetséges művelet. Ekkor az művelet a magasabb számábrázolási formában (tárolási osztályban) kerül végrehajtásra.
Milyen szabály vonatkozik a `%` operátorra?
Csak egész jellegű változókra alkalmazható?
Mit csinála `!=` operátor?
Akkor ad igaz értéket, ha a két operandusa nem egyenlő.

Kérdések

Mi a szerepe a {} operátor párnak?
Valamilyen logikai blokkot határol.
Mikor használjuk a `;` operátort?
Minden ”befejezett” kifejezés végén, de vannak kivételek.
Mit jelent a balérték szabály?
A művelet során a bal oldalon lévő változó kap értéket a jobb oldalon lévő kifejezéstől.
Lehet-e két különböző típusú változón aritmetikai műveletet végezni?
Lehet abban az esetben, ha a változón lehetséges művelet. Ekkor az művelet a magasabb számábrázolási formában (tárolási osztályban) kerül végrehajtásra.
Milyen szabály vonatkozik a `%` operátorra?
Csak egész jellegű változókra alkalmazható?
Mit csinála `!=` operátor?
Akkor ad igaz értéket, ha a két operandusa nem egyenlő.
Mire szolgálnak a logikai operátorok?

Kérdések

Mi a szerepe a {} operátor párnak?
Valamilyen logikai blokkot határol.
Mikor használjuk a `;` operátort?
Minden ”befejezett” kifejezés végén, de vannak kivételek.
Mit jelent a balérték szabály?
A művelet során a bal oldalon lévő változó kap értéket a jobb oldalon lévő kifejezéstől.
Lehet-e két különböző típusú változón aritmetikai műveletet végezni?
Lehet abban az esetben, ha a változón lehetséges művelet. Ekkor az művelet a magasabb számábrázolási formában (tárolási osztályban) kerül végrehajtásra.
Milyen szabály vonatkozik a `%` operátorra?
Csak egész jellegű változókra alkalmazható?
Mit csinála `!=` operátor?
Akkor ad igaz értéket, ha a két operandusa nem egyenlő.
Mire szolgálnak a logikai operátorok?
Relációs jellegű kifejezéseket fűznek össze.

Kérdések

Mi a szerepe a {} operátor párnak?
Valamilyen logikai blokkot határol.
Mikor használjuk a `;` operátort?
Minden ”befejezett” kifejezés végén, de vannak kivételek.
Mit jelent a balérték szabály?
A művelet során a bal oldalon lévő változó kap értéket a jobb oldalon lévő kifejezéstől.
Lehet-e két különböző típusú változón aritmetikai műveletet végezni?
Lehet abban az esetben, ha a változón lehetséges művelet. Ekkor az művelet a magasabb számábrázolási formában (tárolási osztályban) kerül végrehajtásra.
Milyen szabály vonatkozik a `%` operátorra?
Csak egész jellegű változókra alkalmazható?
Mit csinála `!=` operátor?
Akkor ad igaz értéket, ha a két operandusa nem egyenlő.
Mire szolgálnak a logikai operátorok?
Relációs jellegű kifejezéseket fűznek össze.

Kérdések

Milyen szabály vonatkozik a logikai operátorokra?

Kérdések

Milyen szabály vonatkozik a logikai operátorokra?

Ha a kijelölt művelet eredménye biztos, akkor a kifejezést a program nem értékeli tovább.

Kérdések

Milyen szabály vonatkozik a logikai operátorokra?
Ha a kijelölt művelet eredménye biztos, akkor a kifejezést a program nem értékeli tovább.
Milyen változó típusokra alkalmazhatók a bit operátorok?

Kérdések

Milyen szabály vonatkozik a logikai operátorokra?
Ha a kijelölt művelet eredménye biztos, akkor a kifejezést a program nem értékeli tovább.
Milyen változó típusokra alkalmazhatók a bit operátorok?
Csak egész jellegű változókra.

Kérdések

Milyen szabály vonatkozik a logikai operátorokra?

Ha a kijelölt művelet eredménye biztos, akkor a kifejezést a program nem értékeli tovább.

Milyen változó típusokra alkalmazhatók a bit operátorok?

Csak egész jellegű változókra.

Mit csinál a `a+=b;` kifejezés?

Kérdések

Milyen szabály vonatkozik a logikai operátorokra?

Ha a kijelölt művelet eredménye biztos, akkor a kifejezést a program nem értékeli tovább.

Milyen változó típusokra alkalmazhatók a bit operátorok?

Csak egész jellegű változókra.

Mit csinál a `a+=b;` kifejezés?

`a=a+b;`

Kérdések

Milyen szabály vonatkozik a logikai operátorokra?

Ha a kijelölt művelet eredménye biztos, akkor a kifejezést a program nem értékeli tovább.

Milyen változó típusokra alkalmazhatók a bit operátorok?

Csak egész jellegű változókra.

Mit csinál a `a+=b;` kifejezés?

`a=a+b;`

Igaz-e, hogy `a-=b;` az `a=b-a;` ;

Kérdések

Milyen szabály vonatkozik a logikai operátorokra?

Ha a kijelölt művelet eredménye biztos, akkor a kifejezést a program nem értékeli tovább.

Milyen változó típusokra alkalmazhatók a bit operátorok?

Csak egész jellegű változókra.

Mit csinál a `a+=b;` kifejezés?

`a=a+b;`

Igaz-e, hogy `a-=b;` az `a=b-a;` ;

Nem mert a kivonás nem kommutatív. A helyes kifejezés `a=a-b;`

Kérdések

Milyen szabály vonatkozik a logikai operátorokra?
Ha a kijelölt művelet eredménye biztos, akkor a kifejezést a program nem értékeli tovább.
Milyen változó típusokra alkalmazhatók a bit operátorok?
Csak egész jellegű változókra.
Mit csinál a `a+=b;` kifejezés?
`a=a+b;`
Igaz-e, hogy `a-=b;` az `a=b-a;` ;
Nem mert a kivonás nem kommutatív. A helyes kifejezés `a=a-b;`
Mi az eredménye a `b=a++;` kifejezésnek végrehajtás után?

Kérdések

Milyen szabály vonatkozik a logikai operátorokra?
Ha a kijelölt művelet eredménye biztos, akkor a kifejezést a program nem értékeli tovább.
Milyen változó típusokra alkalmazhatók a bit operátorok?
Csak egész jellegű változókra.
Mit csinál a `a+=b;` kifejezés?
`a=a+b;`
Igaz-e, hogy `a-=b;` az `a=b-a;` ;
Nem mert a kivonás nem kommutatív. A helyes kifejezés `a=a-b;`
Mi az eredménye a `b=a++;` kifejezésnek végrehajtás után?
`b`változó értéke egyel kisebb, mint `a` változó értéke.

Kérdések

Milyen szabály vonatkozik a logikai operátorokra?

Ha a kijelölt művelet eredménye biztos, akkor a kifejezést a program nem értékeli tovább.

Milyen változó típusokra alkalmazhatók a bit operátorok?

Csak egész jellegű változókra.

Mit csinál a `a+=b;` kifejezés?

`a=a+b;`

Igaz-e, hogy `a-=b;` az `a=b-a;` ;

Nem mert a kivonás nem kommutatív. A helyes kifejezés `a=a-b;`

Mi az eredménye a `b=a++;` kifejezésnek végrehajtás után?

`b` változó értéke egyel kisebb, mint `a` változó értéke.

Mi az eredménye a `b=++a;` kifejezésnek végrehajtás után?

Kérdések

Milyen szabály vonatkozik a logikai operátorokra?
Ha a kijelölt művelet eredménye biztos, akkor a kifejezést a program nem értékeli tovább.
Milyen változó típusokra alkalmazhatók a bit operátorok?
Csak egész jellegű változókra.
Mit csinál a `a+=b;` kifejezés?
`a=a+b;`
Igaz-e, hogy `a-=b;` az `a=b-a;` ;
Nem mert a kivonás nem kommutatív. A helyes kifejezés `a=a-b;`
Mi az eredménye a `b=a++;` kifejezésnek végrehajtás után?
`b`változó értéke egyel kisebb, mint `a` változó értéke.
Mi az eredménye a `b=++a;` kifejezésnek végrehajtás után?
`b`változó értéke egyenlő `a` változó értékével.

Kérdések

Milyen szabály vonatkozik a logikai operátorokra?
Ha a kijelölt művelet eredménye biztos, akkor a kifejezést a program nem értékeli tovább.
Milyen változó típusokra alkalmazhatók a bit operátorok?
Csak egész jellegű változókra.
Mit csinál a `a+=b;` kifejezés?
`a=a+b;`
Igaz-e, hogy `a-=b;` az `a=b-a;` ;
Nem mert a kivonás nem kommutatív. A helyes kifejezés `a=a-b;`
Mi az eredménye a `b=a++;` kifejezésnek végrehajtás után?
`b` változó értéke egyel kisebb, mint `a` változó értéke.
Mi az eredménye a `b=++a;` kifejezésnek végrehajtás után?
`b` változó értéke egyenlő `a` változó értékével.
Mire a szerepe a `sizeof` operátornak?

Kérdések

Milyen szabály vonatkozik a logikai operátorokra?
Ha a kijelölt művelet eredménye biztos, akkor a kifejezést a program nem értékeli tovább.
Milyen változó típusokra alkalmazhatók a bit operátorok?
Csak egész jellegű változókra.
Mit csinál a `a+=b;` kifejezés?
`a=a+b;`
Igaz-e, hogy `a-=b;` az `a=b-a;` ;
Nem mert a kivonás nem kommutatív. A helyes kifejezés `a=a-b;`
Mi az eredménye a `b=a++;` kifejezésnek végrehajtás után?
`b` változó értéke egyel kisebb, mint `a` változó értéke.
Mi az eredménye a `b=++a;` kifejezésnek végrehajtás után?
`b` változó értéke egyenlő `a` változó értékével.
Mire a szerepe a `sizeof` operátornak?
Megdaja egy típus, vagy egy változó méretét bájtban értelmezve.

Kérdések

Milyen szabály vonatkozik a logikai operátorokra?
Ha a kijelölt művelet eredménye biztos, akkor a kifejezést a program nem értékeli tovább.
Milyen változó típusokra alkalmazhatók a bit operátorok?
Csak egész jellegű változókra.
Mit csinál a `a+=b;` kifejezés?
`a=a+b;`
Igaz-e, hogy `a-=b;` az `a=b-a;` ;
Nem mert a kivonás nem kommutatív. A helyes kifejezés `a=a-b;`
Mi az eredménye a `b=a++;` kifejezésnek végrehajtás után?
`b` változó értéke egyel kisebb, mint `a` változó értéke.
Mi az eredménye a `b=++a;` kifejezésnek végrehajtás után?
`b` változó értéke egyenlő `a` változó értékével.
Mire a szerepe a `sizeof` operátornak?
Megdaja egy típus, vagy egy változó méretét bájtban értelmezve.
Mi a szerepe a `(double)a` kifejezésben a `(double)` cast operátornak?

Kérdések

Milyen szabály vonatkozik a logikai operátorokra?
Ha a kijelölt művelet eredménye biztos, akkor a kifejezést a program nem értékeli tovább.
Milyen változó típusokra alkalmazhatók a bit operátorok?
Csak egész jellegű változókra.
Mit csinál a `a+=b;` kifejezés?
`a=a+b;`
Igaz-e, hogy `a-=b;` az `a=b-a;` ;
Nem mert a kivonás nem kommutatív. A helyes kifejezés `a=a-b;`
Mi az eredménye a `b=a++;` kifejezésnek végrehajtás után?
`b` változó értéke egyel kisebb, mint `a` változó értéke.
Mi az eredménye a `b=++a;` kifejezésnek végrehajtás után?
`b` változó értéke egyenlő `a` változó értékével.
Mire a szerepe a `sizeof` operátornak?
Megdaja egy típus, vagy egy változó méretét bájtban értelmezve.
Mi a szerepe a `(double)a` kifejezésben a `(double)` cast operátornak?
Az adott helyen az a változó értékét `double` típussá konvertálja.

Kérdések

Milyen szabály vonatkozik a logikai operátorokra?
Ha a kijelölt művelet eredménye biztos, akkor a kifejezést a program nem értékeli tovább.
Milyen változó típusokra alkalmazhatók a bit operátorok?
Csak egész jellegű változókra.
Mit csinál a `a+=b;` kifejezés?
`a=a+b;`
Igaz-e, hogy `a-=b;` az `a=b-a;` ;
Nem mert a kivonás nem kommutatív. A helyes kifejezés `a=a-b;`
Mi az eredménye a `b=a++;` kifejezésnek végrehajtás után?
`b` változó értéke egyel kisebb, mint `a` változó értéke.
Mi az eredménye a `b=++a;` kifejezésnek végrehajtás után?
`b` változó értéke egyenlő `a` változó értékével.
Mire a szerepe a `sizeof` operátornak?
Megdaja egy típus, vagy egy változó méretét bájtban értelmezve.
Mi a szerepe a `(double)a` kifejezésben a `(double)` cast operátornak?
Az adott helyen az a változó értékét `double` típussá konvertálja.