Óbudai Egyetem
Kandó Kálmán Villamosmérnöki Kar

C programozási nyelv
Utasítások II.

Dr. Schuster György

2017. december 28.

Ciklusok

Ismétlődő feladatok esetén használjuk,

Ciklusok

Ismétlődő feladatok esetén használjuk, paramétrek változhatnak,

Ciklusok

Ismétlődő feladatok esetén használjuk, paramétrek változhatnak, **de** a kérdéses programrészlet struktúrája nem.

Ciklusok

Ismétlődő feladatok esetén használjuk, paramétrek változhatnak, **de** a kérdéses programrészlet struktúrája nem.
Egy ciklus lehet:

Ciklusok

Ismétlődő feladatok esetén használjuk, paramétrek változhatnak, **de** a kérdéses programrészlet struktúrája nem.
Egy ciklus lehet:

- elöltesztelő.

Ciklusok

Ismétlődő feladatok esetén használjuk, paramétrek változhatnak, **de** a kérdéses programrészlet struktúrája nem.
Egy ciklus lehet:

- elöltesztelő. Ekkor a ciklusban maradás feltételét a ciklus elején vizsgáljuk.

Ciklusok

Ismétlődő feladatok esetén használjuk, paramétrek változhatnak, **de** a kérdéses programrészlet struktúrája nem.
Egy ciklus lehet:

- elöltesztelő. Ekkor a ciklusban maradás feltételét a ciklus elején vizsgáljuk.
  Előfordulhat, hogy a program be sem lép a ciklusba.

Ciklusok

Ismétlődő feladatok esetén használjuk, paramétrek változhatnak, **de** a kérdéses programrészlet struktúrája nem.
Egy ciklus lehet:

- elöltesztelő. Ekkor a ciklusban maradás feltételét a ciklus elején vizsgáljuk.
  Előfordulhat, hogy a program be sem lép a ciklusba.
- hátultesztelő. Ekkor a ciklus törzse legalább egyszer lefut.

Ciklusok

Ismétlődő feladatok esetén használjuk, paramétrek változhatnak, **de** a kérdéses programrészlet struktúrája nem.
Egy ciklus lehet:

- elöltesztelő. Ekkor a ciklusban maradás feltételét a ciklus elején vizsgáljuk.
  Előfordulhat, hogy a program be sem lép a ciklusba.
- hátultesztelő. Ekkor a ciklus törzse legalább egyszer lefut.

A **ciklustörzs:**

Ciklusok

Ismétlődő feladatok esetén használjuk, paramétrek változhatnak, **de** a kérdéses programrészlet struktúrája nem.
Egy ciklus lehet:

- elöltesztelő. Ekkor a ciklusban maradás feltételét a ciklus elején vizsgáljuk.
  Előfordulhat, hogy a program be sem lép a ciklusba.
- hátultesztelő. Ekkor a ciklus törzse legalább egyszer lefut.

A **ciklustörzs:** egy logikai blokk,

Ciklusok

Ismétlődő feladatok esetén használjuk, paramétrek változhatnak,
de a kérdéses programrészlet struktúrája nem.
Egy ciklus lehet:

- elöltesztelő. Ekkor a ciklusban maradás feltételét a ciklus elején vizsgáljuk.
  Előfordulhat, hogy a program be sem lép a ciklusba.
- hátultesztelő. Ekkor a ciklus törzse legalább egyszer lefut.

A **ciklustörzs:** egy logikai blokk,
amely lehet egyetlen pontosvesszővel (`;`) lezárt kifejezés,

Ciklusok

Ismétlődő feladatok esetén használjuk, paramétrek változhatnak,
de a kérdéses programrészlet struktúrája nem.
Egy ciklus lehet:

- elöltesztelő. Ekkor a ciklusban maradás feltételét a ciklus elején vizsgáljuk.
  Előfordulhat, hogy a program be sem lép a ciklusba.
- hátultesztelő. Ekkor a ciklus törzse legalább egyszer lefut.

A **ciklustörzs:** egy logikai blokk,
amely lehet egyetlen pontosvesszővel (`;`) lezárt kifejezés,
vagy (`{ }`) operátor pár közé írt kifejezések.

`for` ciklus

Elöltesztelő ciklus.

`for` ciklus

Elöltesztelő ciklus. Szerkezete:

for ciklus

Elöltesztelő ciklus. Szerkezete:

```
for(kif1;kif2;kif3) ciklustorzs
```

`for` ciklus

Elöltesztelő ciklus. Szerkezete:

```
for(kif1;kif2;kif3) ciklustorzs
```

`kif1` egyszer hajtódik végre.

`for` ciklus

Elöltesztelő ciklus. Szerkezete:

```
for(kif1;kif2;kif3) ciklustorzs
```

`kif1` egyszer hajtódik végre.

`kif2` relációs jellegű.

`for` ciklus

Elöltesztelő ciklus. Szerkezete:

```
for(kif1;kif2;kif3) ciklustorzs
```

`kif1` egyszer hajtódik végre.

`kif2` relációs jellegű.
Ha igaz, a ciklus
folytatódik.

`for` ciklus

Elöltesztelő ciklus. Szerkezete:

```
for(kif1;kif2;kif3) ciklustorzs
```

`kif1` egyszer hajtódik végre.

`kif2` relációs jellegű.
Ha igaz, a ciklus
folytatódik.

`kif3` a ciklustörzs után hajtódik
végre.

`for` ciklus

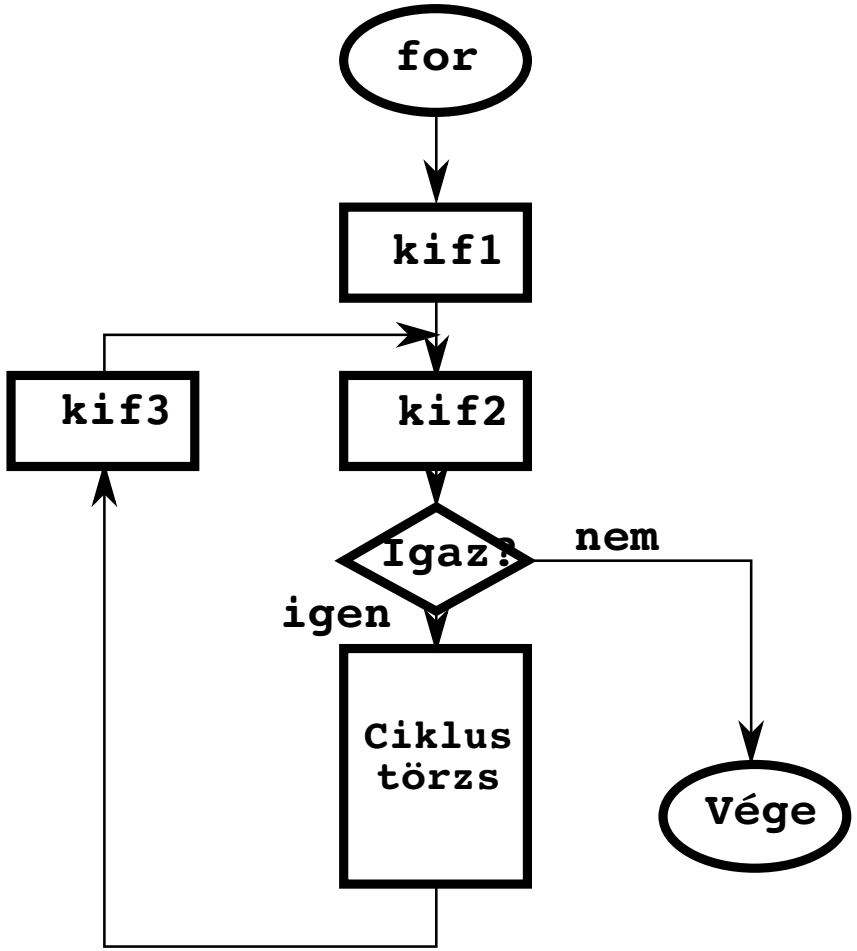
Elöltesztelő ciklus. Szerkezete:

```
for(kif1;kif2;kif3) ciklustorzs
```

kif1 egyszer hajtódik végre.

kif2 relációs jellegű.
Ha igaz, a ciklus folytatódik.

kif3 a ciklustörzs után hajtódik végre.



Példa

Példa

Számoljuk ki a számok összegét 1-10 ig.

Példa

Számoljuk ki a számok összegét 1-10 ig.

```
#include <stdio.h>
int main(void)
    {
    int i,j;
    j=0;
```

Példa

Számoljuk ki a számok összegét 1-10 ig.

```
#include <stdio.h>
int main(void)
    {
    int i,j;
    j=0;
    for(i=1;i<=10;i++)
```

Példa

Számoljuk ki a számok összegét 1-10 ig.

```
#include <stdio.h>
int main(void)
    {
    int i,j;
    j=0;
    for(i=1;i<=10;i++)
       {
       j=j+i;
       }
```

Példa

Számoljuk ki a számok összegét 1-10 ig.

```
#include <stdio.h>
int main(void)
    {
    int i,j;
    j=0;
    for(i=1;i<=10;i++)
       {
       j=j+i;
       }
    printf("%i\n",j);
    return 0;
    }
```

Tipikus hiba

Tipikus hiba

```
for(...;...;...);
```

Tipikus hiba

```
for(...;...;...);
   {
   ⋮
   }
```

Ekkor a ciklustörzs **csak a piros ;-ig tart!**

Tipikus hiba

```
for(...;...;...);
   {
   ⋮
   }
```

Ekkor a ciklustörzs **csak a piros ;-ig tart!**

Nincs hibaüzenet fordításkor.

Tipikus hiba

```
for(...;...;...);
   {
   :
   }
```

Ekkor a ciklustörzs **csak a piros ;-ig tart!**

Nincs hibaüzenet fordításkor.

A program fut.

Tipikus hiba

```
for(...;...;...);
   {
   ⋮
   }
```

Ekkor a ciklustörzs **csak a piros ;-ig tart!**

Nincs hibaüzenet fordításkor.

A program fut.

ROSSZUL!!!

Hmm, lehet így is

Hmm, lehet így is

```
#include <stdio.h>
int main(void)
    {
    int i,j;
```

Hmm, lehet így is

```
#include <stdio.h>
int main(void)
    {
    int i,j;
    for(i=1,j=0;i<=10;j+=i++);
```

Hmm, lehet így is

```
#include <stdio.h>
int main(void)
    {
    int i,j;
    for(i=1,j=0;i<=10;j+=i++);
    printf("%i\n",j);
    return 0;
    }
```

Az első kifejezés a `i=1,j=0`

Hmm, lehet így is

```
#include <stdio.h>
int main(void)
    {
    int i,j;
    for(i=1,j=0;i<=10;j+=i++);
    printf("%i\n",j);
    return 0;
    }
```

Az első kifejezés a `i=1,j=0`, mert a vessző nem zárja le a kifejezést.

Hmm, lehet így is

```
#include <stdio.h>
int main(void)
    {
    int i,j;
    for(i=1,j=0;i<=10;j+=i++);
    printf("%i\n",j);
    return 0;
    }
```

Az első kifejezés a `i=1,j=0`, mert a vessző nem zárja le a kifejezést.
A második kifejezés változatlan.

Hmm, lehet így is

```
#include <stdio.h>
int main(void)
    {
    int i,j;
    for(i=1,j=0;i<=10;j+=i++);
    printf("%i\n",j);
    return 0;
    }
```

Az első kifejezés a `i=1,j=0`, mert a vessző nem zárja le a kifejezést.
A második kifejezés változtalan.
A ciklustörzs üres, amit a `;` zár le.

Hmm, lehet így is

```
#include <stdio.h>
int main(void)
    {
    int i,j;
    for(i=1,j=0;i<=10;j+=i++);
    printf("%i\n",j);
    return 0;
    }
```

Az első kifejezés a `i=1,j=0`, mert a vessző nem zárja le a kifejezést.
A második kifejezés változatlan.
A ciklustörzs üres, amit a `;` zár le.
A harmadik kifejezés a `j+=i++`

Hmm, lehet így is

```
#include <stdio.h>
int main(void)
    {
    int i,j;
    for(i=1,j=0;i<=10;j+=i++);
    printf("%i\n",j);
    return 0;
    }
```

Az első kifejezés a `i=1,j=0`, mert a vessző nem zárja le a kifejezést.
A második kifejezés változtalan.
A ciklustörzs üres, amit a `;` zár le.
A harmadik kifejezés a `j+=i++`, először `i` értéke adódik `j` -hez

Hmm, lehet így is

```
#include <stdio.h>
int main(void)
    {
    int i,j;
    for(i=1,j=0;i<=10;j+=i++);
    printf("%i\n",j);
    return 0;
    }
```

Az első kifejezés a `i=1,j=0`, mert a vessző nem zárja le a kifejezést.
A második kifejezés változtalan.
A ciklustörzs üres, amit a `;` zár le.
A harmadik kifejezés a `j+=i++`, először `i` értéke adódik `j` -hez, majd `i` inkrementálódik.

Hmm, lehet így is

```
#include <stdio.h>
int main(void)
    {
    int i,j;
    for(i=1,j=0;i<=10;j+=i++);
    printf("%i\n",j);
    return 0;
    }
```

Az első kifejezés a `i=1,j=0`, mert a vessző nem zárja le a kifejezést.
A második kifejezés változtalan.
A ciklustörzs üres, amit a `;` zár le.
A harmadik kifejezés a `j+=i++`, először `i` értéke adódik `j` -hez, majd `i` inkrementálódik.
Van `;` a `for( )` után.

while

while

A `while` utasítás szintén előltesztelő ciklust hoz létre.

while

A `while` utasítás szintén előltesztelő ciklust hoz létre.

```
while(kif) ciklustorzs
```

while

A `while` utasítás szintén előltesztelő ciklust hoz létre.

```
while(kif) ciklustorzs
```

Amíg a kifejezés igaz a ciklus fut,

while

A `while` utasítás szintén előltesztelő ciklust hoz létre.

```
while(kif) ciklustorzs
```

Amíg a kifejezés igaz a ciklus fut,
amint a kifejezés hamis lesz a ciklus befejeződik.

while

A `while` utasítás szintén előltesztelő ciklust hoz létre.

```
while(kif) ciklustorzs
```

Amíg a kifejezés igaz a ciklus fut,
amint a kifejezés hamis lesz a ciklus befejeződik.

A feltételviszgálat a ciklustörzs előtt kerül vérehajtásra.

while

A `while` utasítás szintén előltesztelő ciklust hoz létre.

```
while(kif) ciklustorzs
```

Amíg a kifejezés igaz a ciklus fut,
amint a kifejezés hamis lesz a ciklus befejeződik.

A feltételviszgálat a ciklustörzs előtt kerül vérehajtásra.
Tehát a `while` segítségével **előltesztelő** ciklust hozhatunk létre.

while

A `while` utasítás szintén előltesztelő ciklust hoz létre.

```
while(kif) ciklustorzs
```

Amíg a kifejezés igaz a ciklus fut,
amint a kifejezés hamis lesz a ciklus befejeződik.

A feltételviszgálat a ciklustörzs előtt kerül vérehajtásra.
Tehát a `while` segítségével **előltesztelő** ciklust hozhatunk létre.

A ciklustörzs megegyezik a `for` -nál látottakkal.

Példa

Példa

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

Példa

```
#include <stdio.h>
int main(void)
    {
    int i=1;
```

Példa

```
#include <stdio.h>
int main(void)
    {
    int i=1;
    int j=0;
```

Példa

```
#include <stdio.h>
int main(void)
    {
    int i=1;
    int j=0;
    while(i<=10)
```

Példa

```
#include <stdio.h>
int main(void)
    {
    int i=1;
    int j=0;
    while(i<=10)
       {
       j=j+i;
       i++;
       }
```

Példa

```
#include <stdio.h>
int main(void)
    {
    int i=1;
    int j=0;
    while(i<=10)
       {
       j=j+i;
       i++;
       }
    printf("%i\n",j);
    return 0;
    }
```

Példa

```
#include <stdio.h>
int main(void)
    {
    int i=1;
    int j=0;
    while(i<=10)
       {
       j=j+i;
       i++;
       }
    printf("%i\n",j);
    return 0;
    }
```

```
#include <stdio.h>
int main(void)
    {
    int i=1;
    int j=0;
```

Példa

```
#include <stdio.h>
int main(void)
    {
    int i=1;
    int j=0;
    while(i<=10)
       {
       j=j+i;
       i++;
       }
    printf("%i\n",j);
    return 0;
    }
```

```
#include <stdio.h>
int main(void)
    {
    int i=1;
    int j=0;
    while(i<=10) j+=i++;
```

Példa

```
#include <stdio.h>
int main(void)
    {
    int i=1;
    int j=0;
    while(i<=10)
       {
       j=j+i;
       i++;
       }
    printf("%i\n",j);
    return 0;
    }
```

```
#include <stdio.h>
int main(void)
    {
    int i=1;
    int j=0;
    while(i<=10) j+=i++;
    printf("%i\n",j);
    return 0;
    }
```

Tipikus hiba

Tipikus hiba

Már megint az a fránya pontosvessző!

Tipikus hiba

Már megint az a fránya pontosvessző!

```
while(kif);
```

Tipikus hiba

Már megint az a fránya pontosvessző!

`while(kif);` ⇐ Eddig tart a ciklustörzs.

{

⋮

}

A program végtelen ciklusba kerül,

Tipikus hiba

Már megint az a fránya pontosvessző!

`while(kif);` ⇐ Eddig tart a ciklustörzs.

{

⋮

}

A program végtelen ciklusba kerül, mert a feltételben szereplő változó nem változik

Tipikus hiba

Már megint az a fránya pontosvessző!

`while(kif);` ⇐ Eddig tart a ciklustörzs.

{

⋮

}

A program végtelen ciklusba kerül, mert a feltételben szereplő változó nem változik (egy szálas programban).

`do - while` szerkezet

`do - while` szerkezet

Hátultesztelő ciklust lehet vele létrehozni.

do - while szerkezet

Hátultesztelő ciklust lehet vele létrehozni.

```
do ciklustorzs while(kif);
```

`do - while` szerkezet

Hátultesztelő ciklust lehet vele létrehozni.

```
do ciklustorzs while(kif);
```

A ciklus addig fut, amíg a `while` -ban lévő kifejezés igaz.

`do - while` szerkezet

Hátultesztelő ciklust lehet vele létrehozni.

```
do ciklustorzs while(kif);
```

A ciklus addig fut, amíg a `while` -ban lévő kifejezés igaz.

A ciklustörzs legalább egyszer lefut.

`do - while` szerkezet

Hátultesztelő ciklust lehet vele létrehozni.

```
do ciklustorzs while(kif);
```

A ciklus addig fut, amíg a `while` -ban lévő kifejezés igaz.

A ciklustörzs legalább egyszer lefut.

A ciklustörzs megegyezik a `for` -nál látottakkal.

Példa

Példa

Hosszan

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

Példa

Hosszan

```
#include <stdio.h>
int main(void)
    {
    int i=1;
```

Példa

Hosszan

```
#include <stdio.h>
int main(void)
    {
    int i=1;
    int j=0;
```

Példa

Hosszan

```
#include <stdio.h>
int main(void)
    {
    int i=1;
    int j=0;
    do
```

Példa

Hosszan

```
#include <stdio.h>
int main(void)
    {
    int i=1;
    int j=0;
    do
       {
       j=j+i;
       i++;
       }
    while(i<=10);
```

Példa

Hosszan

```
#include <stdio.h>
int main(void)
    {
    int i=1;
    int j=0;
    do
       {
       j=j+i;
       i++;
       }
    while(i<=10);
    printf("%i\n",j);
    return 0;
    }
```

Példa

Hosszan

```
#include <stdio.h>
int main(void)
    {
    int i=1;
    int j=0;
    do
       {
       j=j+i;
       i++;
       }
    while(i<=10);
    printf("%i\n",j);
    return 0;
    }
```

Röviden

```
#include <stdio.h>
int main(void)
    {
    int i=1;
    int j=0;
    do
```

Példa

Hosszan

```
#include <stdio.h>
int main(void)
    {
    int i=1;
    int j=0;
    do
       {
       j=j+i;
       i++;
       }
    while(i<=10);
    printf("%i\n",j);
    return 0;
    }
```

Röviden

```
#include <stdio.h>
int main(void)
    {
    int i=1;
    int j=0;
    do j+=i++; while(i<=10);
```

Példa

Hosszan

```
#include <stdio.h>
int main(void)
    {
    int i=1;
    int j=0;
    do
       {
       j=j+i;
       i++;
       }
    while(i<=10);
    printf("%i\n",j);
    return 0;
    }
```

Röviden

```
#include <stdio.h>
int main(void)
    {
    int i=1;
    int j=0;
    do j+=i++; while(i<=10);
    printf("%i\n",j);
    return 0;
    }
```

Példa

Hosszan

```
#include <stdio.h>
int main(void)
    {
    int i=1;
    int j=0;
    do
       {
       j=j+i;
       i++;
       }
    while(i<=10);
    printf("%i\n",j);
    return 0;
    }
```

Röviden

```
#include <stdio.h>
int main(void)
    {
    int i=1;
    int j=0;
    do j+=i++; while(i<=10);
    printf("%i\n",j);
    return 0;
    }
```

Tipikus hiba: lemarad a `;` `while` mögül**!!!**

`break` utasítás

`break` utasítás

Ciklusok esetén a `break` szerepe az, hogy ahol a program végrehajtja az **aktuális** ciklusból a program kilép.

`break` utasítás

Ciklusok esetén a `break` szerepe az, hogy ahol a program végrehajtja az **aktuális** ciklusból a program kilép.

`break` utasítást csak úgy nem teszünk a ciklusba.

`break` utasítás

Ciklusok esetén a `break` szerepe az, hogy ahol a program végrehajtja az **aktuális** ciklusból a program kilép.

`break` utasítást csak úgy nem teszünk a ciklusba.

Miértis?

`break` utasítás

Ciklusok esetén a `break` szerepe az, hogy ahol a program végrehajtja az **aktuális** ciklusból a program kilép.

`break` utasítást csak úgy nem teszünk a ciklusba.

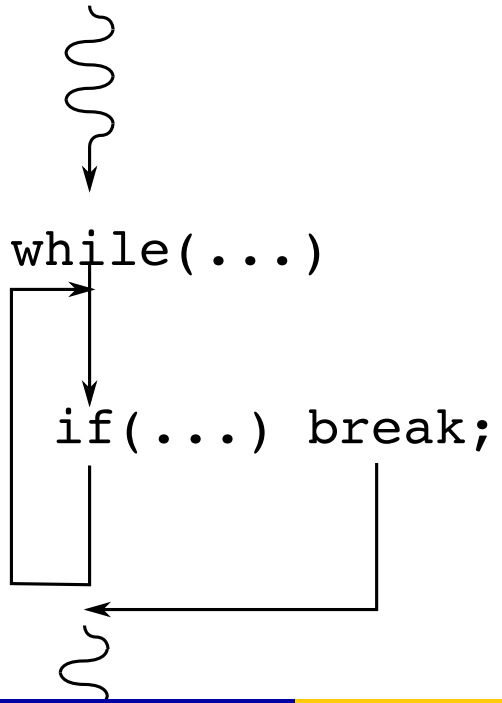
Miértis? Mert akkor mi a fenének írtunk ciklust!

`break` utasítás

Ciklusok esetén a `break` szerepe az, hogy ahol a program végrehajtja az **aktuális** ciklusból a program kilép.

`break` utasítást csak úgy nem teszünk a ciklusba.

Miértis? Mert akkor mi a fenének írtunk ciklust!

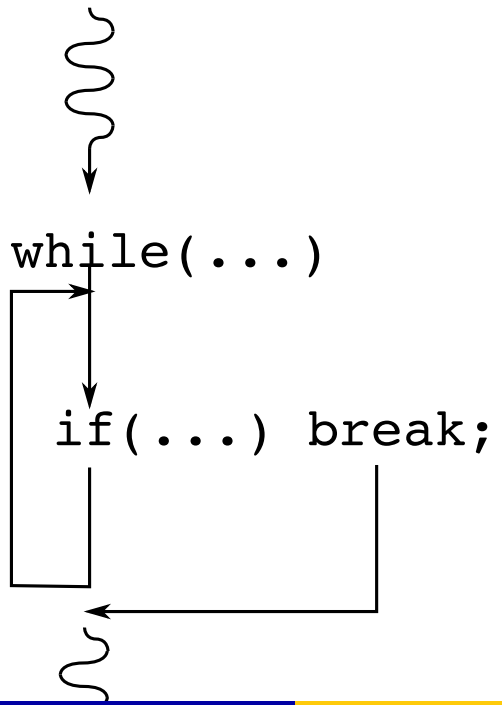


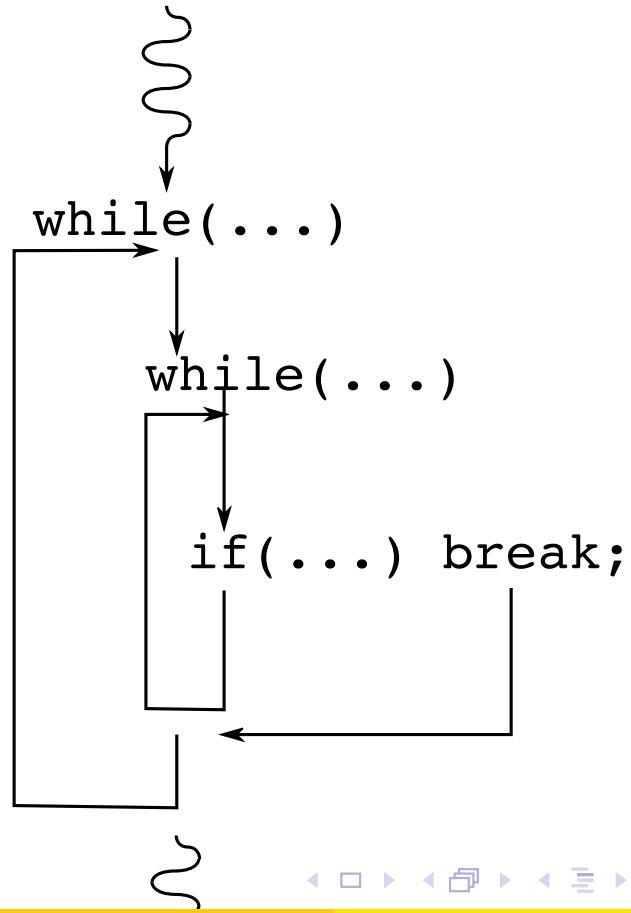
`break` utasítás

Ciklusok esetén a `break` szerepe az, hogy ahol a program végrehajtja az **aktuális** ciklusból a program kilép.

`break` utasítást csak úgy nem teszünk a ciklusba.

Miértis? Mert akkor mi a fenének írtunk ciklust!





Példa a `break` utasításra

Példa a `break` utasításra

```
#include <stdio.h>
int main(void)
    {
    int i;
    int j;
```

Példa a `break` utasításra

```
#include <stdio.h>
int main(void)
    {
    int i;
    int j;
    for(i=0;i<5;i++)
       {
       for(j=0;j<5;j++)
          {
          printf("%i %i\n",i,j);
```

Példa a `break` utasításra

```
#include <stdio.h>
int main(void)
    {
    int i;
    int j;
    for(i=0;i<5;i++)
       {
       for(j=0;j<5;j++)
          {
          printf("%i %i\n",i,j);
          if(j==2) break;
          }
       }
    return 0;
    }
```

Példa a `break` utasításra

```
#include <stdio.h>
int main(void)
    {
    int i;
    int j;
    for(i=0;i<5;i++)
       {
       for(j=0;j<5;j++)
          {
          printf("%i %i\n",i,j);
          if(j==2) break;
          }
       }
    return 0;
    }
```

A képernyőn ez látszik:

```
0 0
0 1
0 2
```

Példa a `break` utasításra

```
#include <stdio.h>
int main(void)
    {
    int i;
    int j;
    for(i=0;i<5;i++)
       {
       for(j=0;j<5;j++)
          {
          printf("%i %i\n",i,j);
          if(j==2) break;
          }
       }
    return 0;
    }
```

A képernyőn ez látszik:

```
0 0
0 1
0 2
1 0
1 1
1 2
```

Példa a `break` utasításra

```
#include <stdio.h>
int main(void)
    {
    int i;
    int j;
    for(i=0;i<5;i++)
       {
       for(j=0;j<5;j++)
          {
          printf("%i %i\n",i,j);
          if(j==2) break;
          }
       }
    return 0;
    }
```

A képernyőn ez látszik:

```
0 0
0 1
0 2
1 0
1 1
1 2
2 0
2 1
2 2
3 0
3 1
3 2
4 0
4 1
4 2
```

`continue` utasítás

`continue` utasítás

A szerepe az, hogy attól a ponttól, ahonnan a program végrehajta a ciklustörzs hátralevő részét kihagyja.

`continue` utasítás

A szerepe az, hogy attól a ponttól, ahonnan a program végrehajta a ciklustörzs hátralevő részét kihagyja.
`for` esetén a 3. kifejezés végrehajtásra kerül.

`continue` utasítás

A szerepe az, hogy attól a ponttól, ahonnan a program végrehajta a ciklustörzs hátralevő részét kihagyja.
`for` esetén a 3. kifejezés végrehajtásra kerül. Ezután folyatódik a ciklus elölről.

`continue` utasítás

A szerepe az, hogy attól a ponttól, ahonnan a program végrehajta a ciklustörzs hátralevő részét kihagyja.
`for` esetén a 3. kifejezés végrehajtásra kerül. Ezután folyatódik a ciklus elölről.
`while` esetén a ciklus a ciklusfejtől újra kezdődik.

`continue` utasítás

A szerepe az, hogy attól a ponttól, ahonnan a program végrehajta a ciklustörzs hátralevő részét kihagyja.
`for` esetén a 3. kifejezés végrehajtásra kerül. Ezután folyatódik a ciklus elölről.
`while` esetén a ciklus a ciklusfejtől újra kezdődik. a `while` -ban lévő kifejezés kiértékelődik.

`continue` utasítás

A szerepe az, hogy attól a ponttól, ahonnan a program végrehajta a ciklustörzs hátralevő részét kihagyja.
`for` esetén a 3. kifejezés végrehajtásra kerül. Ezután folyatódik a ciklus elölről.
`while` esetén a ciklus a ciklusfejtől újra kezdődik. a `while` -ban lévő kifejezés kiértékelődik.
`do-while` esetén a program a **`while(kif)`** -re ugrik.

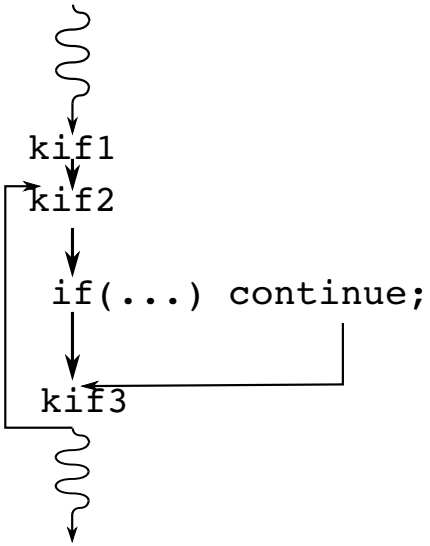
`continue` utasítás

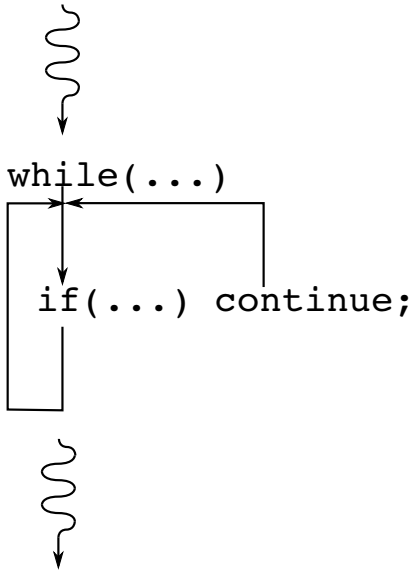
A szerepe az, hogy attól a ponttól, ahonnan a program végrehajta a ciklustörzs hátralevő részét kihagyja.
`for` esetén a 3. kifejezés végrehajtásra kerül. Ezután folyatódik a ciklus elölről.
`while` esetén a ciklus a ciklusfejtől újra kezdődik. a `while` -ban lévő kifejezés kiértékelődik.
`do-while` esetén a program a **`while(kif)`** -re ugrik. A `kif` értékétől függően folyatódik a ciklus.

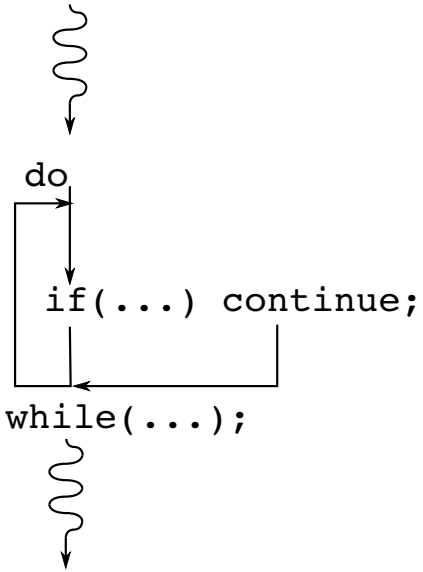
`continue` utasítás

A szerepe az, hogy attól a ponttól, ahonnan a program végrehajta a ciklustörzs hátralevő részét kihagyja.
`for` esetén a 3. kifejezés végrehajtásra kerül. Ezután folyatódik a ciklus elölről.
`while` esetén a ciklus a ciklusfejtől újra kezdődik. a `while` -ban lévő kifejezés kiértékelődik.
`do-while` esetén a program a **`while(kif)`** -re ugrik. A `kif` értékétől függően folyatódik a ciklus.

`continue` utasítás







Példa `continue` utasításra

Példa `continue` utasításra

```
#include <stdio.h>
int main(void)
    {
    int i;
    for(i=0;i<5;i++)
       {
       if(i==2) continue;
       printf("%i\n",i);
```

Példa `continue` utasításra

```
#include <stdio.h>
int main(void)
    {
    int i;
    for(i=0;i<5;i++)
       {
       if(i==2) continue;
       printf("%i\n",i);
       }
    return 0;
    }
```

Példa `continue` utasításra

```
#include <stdio.h>
int main(void)
    {
    int i;
    for(i=0;i<5;i++)
       {
       if(i==2) continue;
       printf("%i\n",i);
       }
    return 0;
    }
```

A képernyőn ez látszik:

```
0
1
```

Példa `continue` utasításra

```
#include <stdio.h>
int main(void)
    {
    int i;
    for(i=0;i<5;i++)
       {
       if(i==2) continue;
       printf("%i\n",i);
       }
    return 0;
    }
```

A képernyőn ez látszik:

```
0
1
3
```

Példa `continue` utasításra

```
#include <stdio.h>
int main(void)
    {
    int i;
    for(i=0;i<5;i++)
       {
       if(i==2) continue;
       printf("%i\n",i);
       }
    return 0;
    }
```

A képernyőn ez látszik:

```
0
1
3
4
```

goto utasítás

goto utasítás

Lehetőség szerint ne használjuk!!

goto utasítás

Lehetőség szerint ne használjuk!!

Használata veszélyes, nehezen felderíthető programhibákhoz vezet.

goto utasítás

Lehetőség szerint ne használjuk!!

Használata veszélyes, nehezen felderíthető programhibákhoz vezet.

Szerepe az, hogy a program feltétel nélkül az adott címkére ugrik.

goto utasítás

Lehetőség szerint ne használjuk!!

Használata veszélyes, nehezen felderíthető programhibákhoz vezet.

Szerepe az, hogy a program feltétel nélkül az adott címkére ugrik.

A címke:

goto utasítás

Lehetőség szerint ne használjuk!!

Használata veszélyes, nehezen felderíthető programhibákhoz vezet.

Szerepe az, hogy a program feltétel nélkül az adott címkére ugrik.

A címke:

`cimke:`

Nem lehet foglalt kulcsszó és ’:’ zárja le.

goto utasítás

Lehetőség szerint ne használjuk!!

Használata veszélyes, nehezen felderíthető programhibákhoz vezet.

Szerepe az, hogy a program feltétel nélkül az adott címkére ugrik.

A címke:

cimke:

Nem lehet foglalt kulcsszó és ':' zárja le.

A címke lehet a goto előtt és után.

goto utasítás

Lehetőség szerint ne használjuk!!

Használata veszélyes, nehezen felderíthető programhibákhoz vezet.

Szerepe az, hogy a program feltétel nélkül az adott címkére ugrik.

A címke:

`cimke:`

Nem lehet foglalt kulcsszó és ':' zárja le.

A címke lehet a `goto` előtt és után.

Tehát:

```
cimke:
    ⋮
   goto cimke;
```

`goto` utasítás

Lehetőség szerint ne használjuk!!

Használata veszélyes, nehezen felderíthető programhibákhoz vezet.

Szerepe az, hogy a program feltétel nélkül az adott címkére ugrik.

A címke:

`cimke:`

Nem lehet foglalt kulcsszó és ’`:`’ zárja le.

A címke lehet a `goto` előtt és után.

Tehát:

```
cimke:
    ⋮
  goto cimke;
```

vagy

```
  goto cimke;
    ⋮
cimke:
```

Kérdések

Mikor alkalmazunk ciklusokat?

Kérdések

Mikor alkalmazunk ciklusokat?

Ha sokszor kell elvégezni azonos feladatot.

Kérdések

Mikor alkalmazunk ciklusokat?

Ha sokszor kell elvégezni azonos feladatot.

Mit jelent, hogy a ciklus elltesztelő?

Kérdések

Mikor alkalmazunk ciklusokat?

Ha sokszor kell elvégezni azonos feladatot.

Mit jelent, hogy a ciklus elltesztelő?

Ha a ciklus feltétel kiértékelése a ciklustörzs előtt megtörténik. Ekkor előfordulhat, hogy a ciklus egyszer sem fut le.

Kérdések

Mikor alkalmazunk ciklusokat?

Ha sokszor kell elvégezni azonos feladatot.

Mit jelent, hogy a ciklus elltesztelő?

Ha a ciklus feltétel kiértékelése a ciklustörzs előtt megtörténik. Ekkor előfordulhat, hogy a ciklus egyszer sem fut le.

Mit jelent, hogy ciklus hátul tesztelő?

Kérdések

Mikor alkalmazunk ciklusokat?

Ha sokszor kell elvégezni azonos feladatot.

Mit jelent, hogy a ciklus elltesztelő?

Ha a ciklus feltétel kiértékelése a ciklustörzs előtt megtörténik. Ekkor előfordulhat, hogy a ciklus egyszer sem fut le.

Mit jelent, hogy ciklus hátul tesztelő?

Ha a ciklus feltétel kiértékelése a ciklustörzs után tötrénik. Ekkor a ciklustörzs legalább egyszer lefut.

Kérdések

Mikor alkalmazunk ciklusokat?
Ha sokszor kell elvégezni azonos feladatot.
Mit jelent, hogy a ciklus elltesztelő?
Ha a ciklus feltétel kiértékelése a ciklustörzs előtt megtörténik. Ekkor előfordulhat, hogy a ciklus egyszer sem fut le.
Mit jelent, hogy ciklus hátul tesztelő?
Ha a ciklus feltétel kiértékelése a ciklustörzs után tötrénik. Ekkor a ciklustörzs legalább egyszer lefut.
Mi lehet a ciklustörzs?

Kérdések

Mikor alkalmazunk ciklusokat?

Ha sokszor kell elvégezni azonos feladatot.

Mit jelent, hogy a ciklus elltesztelő?

Ha a ciklus feltétel kiértékelése a ciklustörzs előtt megtörténik. Ekkor előfordulhat, hogy a ciklus egyszer sem fut le.

Mit jelent, hogy ciklus hátul tesztelő?

Ha a ciklus feltétel kiértékelése a ciklustörzs után tötrénik. Ekkor a ciklustörzs legalább egyszer lefut.

Mi lehet a ciklustörzs?

Vagy egy kifejezés `;`-vel lezárva, vagy {} operátork közé zárt kifejezések.

Kérdések

Mikor alkalmazunk ciklusokat?

Ha sokszor kell elvégezni azonos feladatot.

Mit jelent, hogy a ciklus elltesztelő?

Ha a ciklus feltétel kiértékelése a ciklustörzs előtt megtörténik. Ekkor előfordulhat, hogy a ciklus egyszer sem fut le.

Mit jelent, hogy ciklus hátul tesztelő?

Ha a ciklus feltétel kiértékelése a ciklustörzs után tötrénik. Ekkor a ciklustörzs legalább egyszer lefut.

Mi lehet a ciklustörzs?

Vagy egy kifejezés `;`-vel lezárva, vagy {} operátork közé zárt kifejezések.

Milyen jellegű a ciklus feltétel kiértékelése?

Kérdések

Mikor alkalmazunk ciklusokat?
Ha sokszor kell elvégezni azonos feladatot.
Mit jelent, hogy a ciklus elltesztelő?
Ha a ciklus feltétel kiértékelése a ciklustörzs előtt megtörténik. Ekkor előfordulhat, hogy a ciklus egyszer sem fut le.
Mit jelent, hogy ciklus hátul tesztelő?
Ha a ciklus feltétel kiértékelése a ciklustörzs után tötrénik. Ekkor a ciklustörzs legalább egyszer lefut.
Mi lehet a ciklustörzs?
Vagy egy kifejezés `;`-vel lezárva, vagy {} operátork közé zárt kifejezések.
Milyen jellegű a ciklus feltétel kiértékelése?
Minden esetben igaz - nem igaz jellegű a kiértékelés. A C-ben a ciklusban maradás minden esetben igaz esetben történik.

Kérdések

Mikor alkalmazunk ciklusokat?

Ha sokszor kell elvégezni azonos feladatot.

Mit jelent, hogy a ciklus elltesztelő?

Ha a ciklus feltétel kiértékelése a ciklustörzs előtt megtörténik. Ekkor előfordulhat, hogy a ciklus egyszer sem fut le.

Mit jelent, hogy ciklus hátul tesztelő?

Ha a ciklus feltétel kiértékelése a ciklustörzs után tötrénik. Ekkor a ciklustörzs legalább egyszer lefut.

Mi lehet a ciklustörzs?

Vagy egy kifejezés `;`-vel lezárva, vagy {} operátork közé zárt kifejezések.

Milyen jellegű a ciklus feltétel kiértékelése?

Minden esetben igaz - nem igaz jellegű a kiértékelés. A C-ben a ciklusban maradás minden esetben igaz esetben történik.

Milyen jellegű a **`for`** ciklus?

Kérdések

Mikor alkalmazunk ciklusokat?
Ha sokszor kell elvégezni azonos feladatot.
Mit jelent, hogy a ciklus elltesztelő?
Ha a ciklus feltétel kiértékelése a ciklustörzs előtt megtörténik. Ekkor előfordulhat, hogy a ciklus egyszer sem fut le.
Mit jelent, hogy ciklus hátul tesztelő?
Ha a ciklus feltétel kiértékelése a ciklustörzs után tötrénik. Ekkor a ciklustörzs legalább egyszer lefut.
Mi lehet a ciklustörzs?
Vagy egy kifejezés `;`-vel lezárva, vagy {} operátork közé zárt kifejezések.
Milyen jellegű a ciklus feltétel kiértékelése?
Minden esetben igaz - nem igaz jellegű a kiértékelés. A C-ben a ciklusban maradás minden esetben igaz esetben történik.
Milyen jellegű a `for` ciklus?
Elöltesztelő.

Kérdések

Mikor alkalmazunk ciklusokat?
Ha sokszor kell elvégezni azonos feladatot.
Mit jelent, hogy a ciklus elltesztelő?
Ha a ciklus feltétel kiértékelése a ciklustörzs előtt megtörténik. Ekkor előfordulhat, hogy a ciklus egyszer sem fut le.
Mit jelent, hogy ciklus hátul tesztelő?
Ha a ciklus feltétel kiértékelése a ciklustörzs után tötrénik. Ekkor a ciklustörzs legalább egyszer lefut.
Mi lehet a ciklustörzs?
Vagy egy kifejezés `;`-vel lezárva, vagy {} operátork közé zárt kifejezések.
Milyen jellegű a ciklus feltétel kiértékelése?
Minden esetben igaz - nem igaz jellegű a kiértékelés. A C-ben a ciklusban maradás minden esetben igaz esetben történik.
Milyen jellegű a **`for`** ciklus?
Elöltesztelő.
Mi a **`for`** kifejezéseinek működése?

Kérdések

Mikor alkalmazunk ciklusokat?
Ha sokszor kell elvégezni azonos feladatot.
Mit jelent, hogy a ciklus elltesztelő?
Ha a ciklus feltétel kiértékelése a ciklustörzs előtt megtörténik. Ekkor előfordulhat, hogy a ciklus egyszer sem fut le.
Mit jelent, hogy ciklus hátul tesztelő?
Ha a ciklus feltétel kiértékelése a ciklustörzs után tötrénik. Ekkor a ciklustörzs legalább egyszer lefut.
Mi lehet a ciklustörzs?
Vagy egy kifejezés `;`-vel lezárva, vagy {} operátork közé zárt kifejezések.
Milyen jellegű a ciklus feltétel kiértékelése?
Minden esetben igaz - nem igaz jellegű a kiértékelés. A C-ben a ciklusban maradás minden esetben igaz esetben történik.
Milyen jellegű a `for` ciklus?
Elöltesztelő.
Mi a `for` kifejezéseinek működése?
Az 1. kifejezés csak egyszer fut le. Ezért általában kezdeti értékadásra használják. A 2. kifejezés relációs jellegű, ez a ciklus feltétel kiértékelése. A 3. kifejezés a ciklustörzs után fut le.

Kérdések

Mikor alkalmazunk ciklusokat?
Ha sokszor kell elvégezni azonos feladatot.
Mit jelent, hogy a ciklus elltesztelő?
Ha a ciklus feltétel kiértékelése a ciklustörzs előtt megtörténik. Ekkor előfordulhat, hogy a ciklus egyszer sem fut le.
Mit jelent, hogy ciklus hátul tesztelő?
Ha a ciklus feltétel kiértékelése a ciklustörzs után tötrénik. Ekkor a ciklustörzs legalább egyszer lefut.
Mi lehet a ciklustörzs?
Vagy egy kifejezés `;`-vel lezárva, vagy {} operátork közé zárt kifejezések.
Milyen jellegű a ciklus feltétel kiértékelése?
Minden esetben igaz - nem igaz jellegű a kiértékelés. A C-ben a ciklusban maradás minden esetben igaz esetben történik.
Milyen jellegű a `for` ciklus?
Elöltesztelő.
Mi a `for` kifejezéseinek működése?
Az 1. kifejezés csak egyszer fut le. Ezért általában kezdeti értékadásra használják. A 2. kifejezés relációs jellegű, ez a ciklus feltétel kiértékelése. A 3. kifejezés a ciklustörzs után fut le.

Kérdések

Milyen jellegű a `while` ciklus?

Kérdések

Milyen jellegű a `while` ciklus?
Elöltesztelő.

Kérdések

Milyen jellegű a `while` ciklus?
Elöltesztelő.
Milyen jellegű a `while` utasítás argumentumának kiértékelése?

Kérdések

Milyen jellegű a `while` ciklus?
Elöltesztelő.
Milyen jellegű a `while` utasítás argumentumának kiértékelése?
Relációs jellegű.

Kérdések

Milyen jellegű a **`while`** ciklus?
Elöltesztelő.
Milyen jellegű a **`while`** utasítás argumentumának kiértékelése?
Relációs jellegű.
Mi a tipikus hiba **`for`** és **`while`** uatsítások esetén, ha van ciklustörzs?

Kérdések

Milyen jellegű a **`while`** ciklus?
Elöltesztelő.
Milyen jellegű a **`while`** utasítás argumentumának kiértékelése?
Relációs jellegű.
Mi a tipikus hiba **`for`** és **`while`** uatsítások esetén, ha van ciklustörzs?
Az argumentum után egy `;` kerül.

Kérdések

Milyen jellegű a **`while`** ciklus?
Elöltesztelő.
Milyen jellegű a **`while`** utasítás argumentumának kiértékelése?
Relációs jellegű.
Mi a tipikus hiba **`for`** és **`while`** uatsítások esetén, ha van ciklustörzs?
Az argumentum után egy `;` kerül.
Milyen jellegű ciklust hoz létre a **`do - while`** szerkezet?

Kérdések

Milyen jellegű a **`while`** ciklus?
Elöltesztelő.
Milyen jellegű a **`while`** utasítás argumentumának kiértékelése?
Relációs jellegű.
Mi a tipikus hiba **`for`** és **`while`** uatsítások esetén, ha van ciklustörzs?
Az argumentum után egy `;` kerül.
Milyen jellegű ciklust hoz létre a **`do - while`** szerkezet?
Hátultesztelő ciklust.

Kérdések

Milyen jellegű a **`while`** ciklus?
Elöltesztelő.
Milyen jellegű a **`while`** utasítás argumentumának kiértékelése?
Relációs jellegű.
Mi a tipikus hiba **`for`** és **`while`** uatsítások esetén, ha van ciklustörzs?
Az argumentum után egy `;` kerül.
Milyen jellegű ciklust hoz létre a **`do - while`** szerkezet?
Hátultesztelő ciklust.
Mi a tipikus hiba a **`do - while`** esetén?

Kérdések

Milyen jellegű a **`while`** ciklus?
Elöltesztelő.
Milyen jellegű a **`while`** utasítás argumentumának kiértékelése?
Relációs jellegű.
Mi a tipikus hiba **`for`** és **`while`** uatsítások esetén, ha van ciklustörzs?
Az argumentum után egy `;` kerül.
Milyen jellegű ciklust hoz létre a **`do - while`** szerkezet?
Hátultesztelő ciklust.
Mi a tipikus hiba a **`do - while`** esetén?
Nem kerül `;` a záró **`while`** után. De ez szintaktikai hiba könnyen észrevehető.

Kérdések

Milyen jellegű a **`while`** ciklus?
Elöltesztelő.
Milyen jellegű a **`while`** utasítás argumentumának kiértékelése?
Relációs jellegű.
Mi a tipikus hiba **`for`** és **`while`** uatsítások esetén, ha van ciklustörzs?
Az argumentum után egy `;` kerül.
Milyen jellegű ciklust hoz létre a **`do - while`** szerkezet?
Hátultesztelő ciklust.
Mi a tipikus hiba a **`do - while`** esetén?
Nem kerül `;` a záró **`while`** után. De ez szintaktikai hiba könnyen észrevehető.
Mi a szerepe a **`break`** utasításnak?

Kérdések

Milyen jellegű a **`while`** ciklus?
Elöltesztelő.
Milyen jellegű a **`while`** utasítás argumentumának kiértékelése?
Relációs jellegű.
Mi a tipikus hiba **`for`** és **`while`** uatsítások esetén, ha van ciklustörzs?
Az argumentum után egy `;` kerül.
Milyen jellegű ciklust hoz létre a **`do - while`** szerkezet?
Hátultesztelő ciklust.
Mi a tipikus hiba a **`do - while`** esetén?
Nem kerül `;` a záró **`while`** után. De ez szintaktikai hiba könnyen észrevehető.
Mi a szerepe a **`break`** utasításnak?
Az aktuális ciklust a program elhagyja.

Kérdések

Milyen jellegű a **`while`** ciklus?
Elöltesztelő.
Milyen jellegű a **`while`** utasítás argumentumának kiértékelése?
Relációs jellegű.
Mi a tipikus hiba **`for`** és **`while`** uatsítások esetén, ha van ciklustörzs?
Az argumentum után egy `;` kerül.
Milyen jellegű ciklust hoz létre a **`do - while`** szerkezet?
Hátultesztelő ciklust.
Mi a tipikus hiba a **`do - while`** esetén?
Nem kerül `;` a záró **`while`** után. De ez szintaktikai hiba könnyen észrevehető.
Mi a szerepe a **`break`** utasításnak?
Az aktuális ciklust a program elhagyja.
Mi a szerepe a **`continue`** utasításnak?

Kérdések

Milyen jellegű a **`while`** ciklus?
Elöltesztelő.
Milyen jellegű a **`while`** utasítás argumentumának kiértékelése?
Relációs jellegű.
Mi a tipikus hiba **`for`** és **`while`** uatsítások esetén, ha van ciklustörzs?
Az argumentum után egy `;` kerül.
Milyen jellegű ciklust hoz létre a **`do - while`** szerkezet?
Hátultesztelő ciklust.
Mi a tipikus hiba a **`do - while`** esetén?
Nem kerül `;` a záró **`while`** után. De ez szintaktikai hiba könnyen észrevehető.
Mi a szerepe a **`break`** utasításnak?
Az aktuális ciklust a program elhagyja.
Mi a szerepe a **`continue`** utasításnak?
A ciklustörzs azon részét, amely az utasítást követi kihagyja és a ciklus végére ugrik.

Kérdések

Milyen jellegű a **`while`** ciklus?
Elöltesztelő.
Milyen jellegű a **`while`** utasítás argumentumának kiértékelése?
Relációs jellegű.
Mi a tipikus hiba **`for`** és **`while`** uatsítások esetén, ha van ciklustörzs?
Az argumentum után egy `;` kerül.
Milyen jellegű ciklust hoz létre a **`do - while`** szerkezet?
Hátultesztelő ciklust.
Mi a tipikus hiba a **`do - while`** esetén?
Nem kerül `;` a záró **`while`** után. De ez szintaktikai hiba könnyen észrevehető.
Mi a szerepe a **`break`** utasításnak?
Az aktuális ciklust a program elhagyja.
Mi a szerepe a **`continue`** utasításnak?
A ciklustörzs azon részét, amely az utasítást követi kihagyja és a ciklus végére ugrik.
Mi történik **`for`** esetén a 3. kifejezéssel?

Kérdések

Milyen jellegű a **`while`** ciklus?
Elöltesztelő.
Milyen jellegű a **`while`** utasítás argumentumának kiértékelése?
Relációs jellegű.
Mi a tipikus hiba **`for`** és **`while`** uatsítások esetén, ha van ciklustörzs?
Az argumentum után egy `;` kerül.
Milyen jellegű ciklust hoz létre a **`do - while`** szerkezet?
Hátultesztelő ciklust.
Mi a tipikus hiba a **`do - while`** esetén?
Nem kerül `;` a záró **`while`** után. De ez szintaktikai hiba könnyen észrevehető.
Mi a szerepe a **`break`** utasításnak?
Az aktuális ciklust a program elhagyja.
Mi a szerepe a **`continue`** utasításnak?
A ciklustörzs azon részét, amely az utasítást követi kihagyja és a ciklus végére ugrik.
Mi történik **`for`** esetén a 3. kifejezéssel?
Végrehajtásra kerül.

Kérdések

Milyen jellegű a **`while`** ciklus?
Elöltesztelő.
Milyen jellegű a **`while`** utasítás argumentumának kiértékelése?
Relációs jellegű.
Mi a tipikus hiba **`for`** és **`while`** uatsítások esetén, ha van ciklustörzs?
Az argumentum után egy `;` kerül.
Milyen jellegű ciklust hoz létre a **`do - while`** szerkezet?
Hátultesztelő ciklust.
Mi a tipikus hiba a **`do - while`** esetén?
Nem kerül `;` a záró **`while`** után. De ez szintaktikai hiba könnyen észrevehető.
Mi a szerepe a **`break`** utasításnak?
Az aktuális ciklust a program elhagyja.
Mi a szerepe a **`continue`** utasításnak?
A ciklustörzs azon részét, amely az utasítást követi kihagyja és a ciklus végére ugrik.
Mi történik **`for`** esetén a 3. kifejezéssel?
Végrehajtásra kerül.