

Programozási nyelvek 1

Szathmáry László
Debreceni Egyetem
Informatikai Kar

2. előadás

- ismerkedés a C programozási nyelvvel

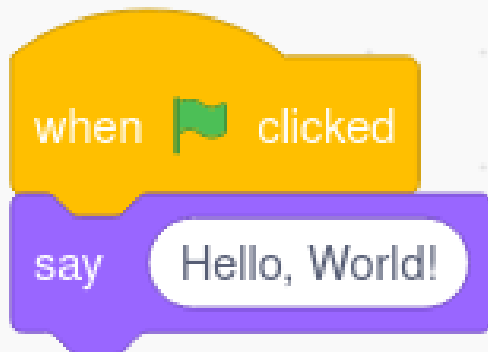
(utolsó módosítás: 2020. febr. 20.)

2019-2020, 2. félév



Hello, World!

```
1  #include <stdio.h>
2
3  int main( )
4  {
5      printf( "Hello, world!\n" );
6
7      return 0;
8  }
```



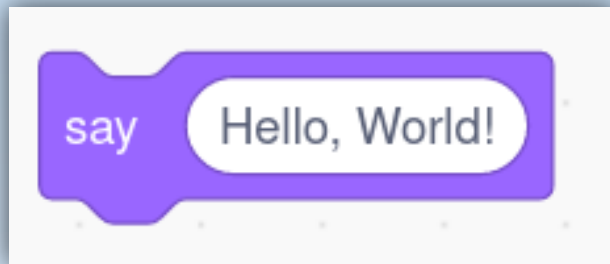
```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("Hello, world!\n");
6
7      return 0;
8  }
```

Számos új fogalommal fogunk találkozni:

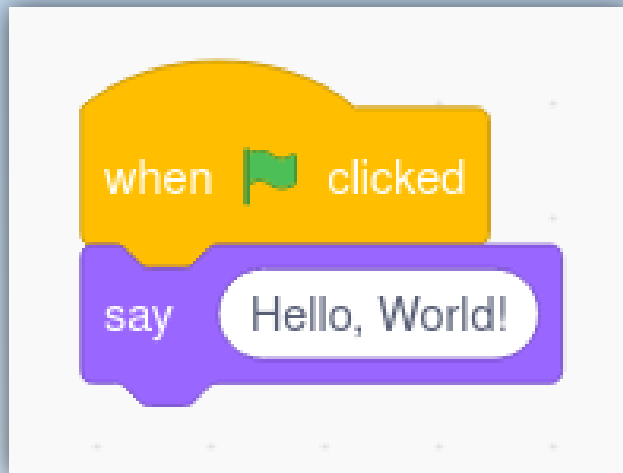
- változó
- konstans
- elágazás
- ciklus
- eljárás / függvény
- kifejezés
- ...

when  clicked

```
1  int main()  
2  {  
3  
4  }
```



```
printf("Hello, World!\n");
```



```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("Hello, world!\n");
6
7      return 0;
8  }
```

Blue arrows point from the logo to the code: one to line 1, one to the opening brace of main, one to the printf statement, one to the return statement, and one to the closing brace of main.

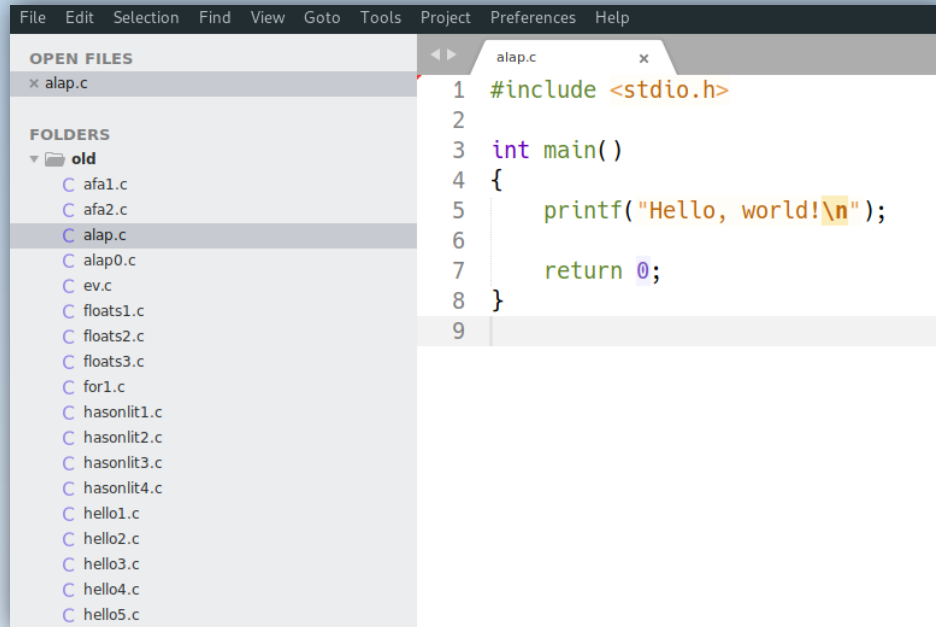
Programozási környezet

Programfejlesztéshez kell egy **kényelmes** és **hatékony** programozási környezet:

- operációs rendszer (Linux, Windows, Mac OS)
- a kiválasztott operációs rendszer (nálunk: Linux) ismerete
 - terminál használata
 - legfontosabb Linux parancsok ismerete
 - tájékozódás a fájlrendszerben
 - könyvtár- és fájlműveletek
 - egy hatékony fájlmenedzser ismerete (pl. Midnight Commander)
 - egy hatékony szövegszerkesztő ismerete (pl. Visual Studio Code, Sublime Text)

Mindezekről készítettem oktatóvideókat: <http://bit.ly/2tt1Ylf>

Fordítás / futtatás



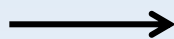
```
File Edit Selection Find View Goto Tools Project Preferences Help
OPEN FILES
x alap.c
FOLDERS
old
  C afa1.c
  C afa2.c
  C alap.c
  C alap0.c
  C ev.c
  C floats1.c
  C floats2.c
  C floats3.c
  C for1.c
  C hasonlit1.c
  C hasonlit2.c
  C hasonlit3.c
  C hasonlit4.c
  C hello1.c
  C hello2.c
  C hello3.c
  C hello4.c
  C hello5.c
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("Hello, world!\n");
6
7     return 0;
8 }
9
```

```
[18:35:48] ~/work $ gcc hello.c
[18:35:50] ~/work $ ./a.out
Hello, world!
[18:35:53] ~/work $
```

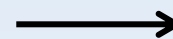
terminál

editor (szövegszerkesztő)

forráskód



fordító



gépi kód

Egész érték beolvasása a billentyűzetről

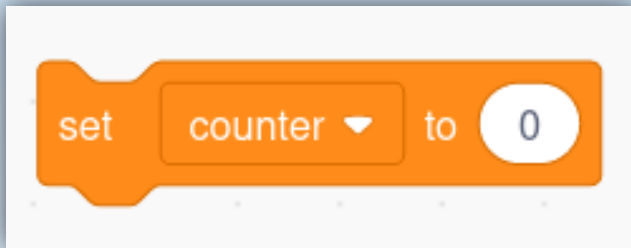
egysoros
megjegyzés

```
1 // egész (int) beolvasása / kiírása
2
3 #include <stdio.h>
4
5 int main()
6 {
7     // egész érték beolvasása
8     int ev;
9     printf("Milyen evet írunk?\n");
10    scanf("%d", &ev);
11
12    printf("Mar %d van? Hogy telik az ido!\n", ev);
13
14    return 0;
15 }
```

változó deklarálása

„placeholder”
(helykitöltő)

Változó deklarálása



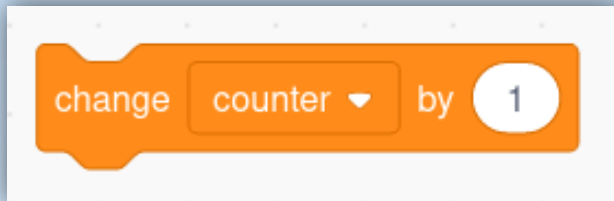
értékkadás
„legyen egyenlő”

```
int counter = 0;
```

típus név érték

Deklarálás: változó **létrehozása**. A változó számára memóriaterület foglalódik le.

Egész típusú változó inkrementálása



```
counter = counter + 1;
```

a jobb oldal értékelődik ki előbb

Az előző oldalon a változót már létrehoztuk (deklaráltuk).
A típust így nem kell még egyszer megadnunk.

Egész típusú változó inkrementálása

```
counter = counter + 1;
```

Ugyanazt eredményezik:

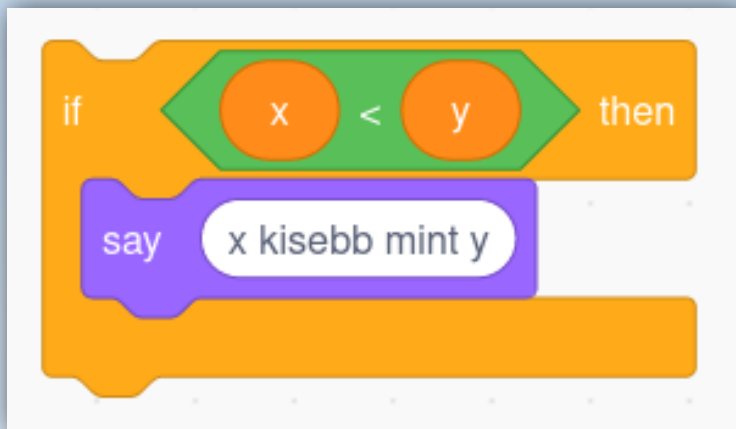
```
counter += 1;
```

```
counter++;
```

```
++counter;
```

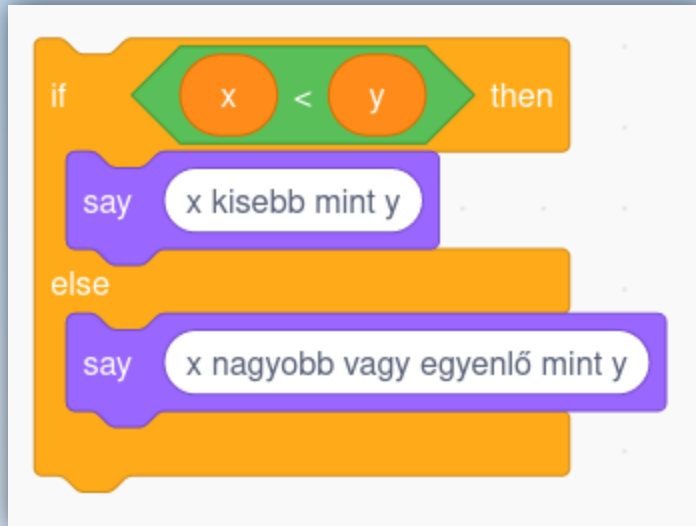
„syntactic sugar”

Az if utasítás



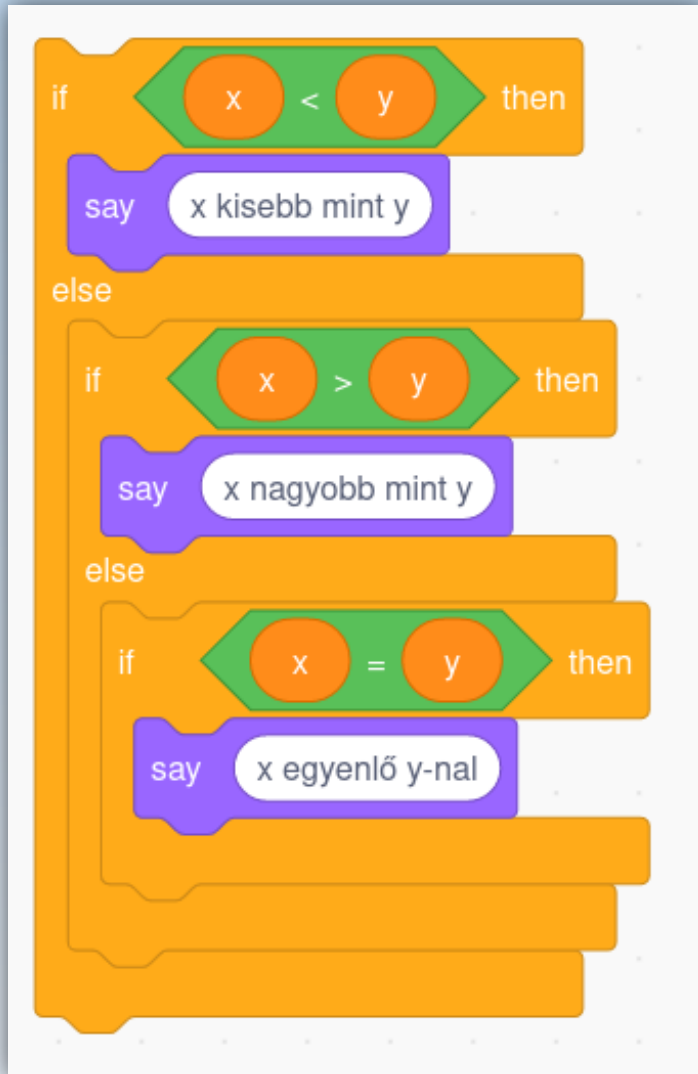
```
if (x < y)
{
    printf("x kisebb mint y\n");
}
```

Az if-else utasítás



```
if (x < y)
{
    printf("x kisebb mint y\n");
}
else
{
    printf("x nagyobb vagy egyenlo mint y\n");
}
```

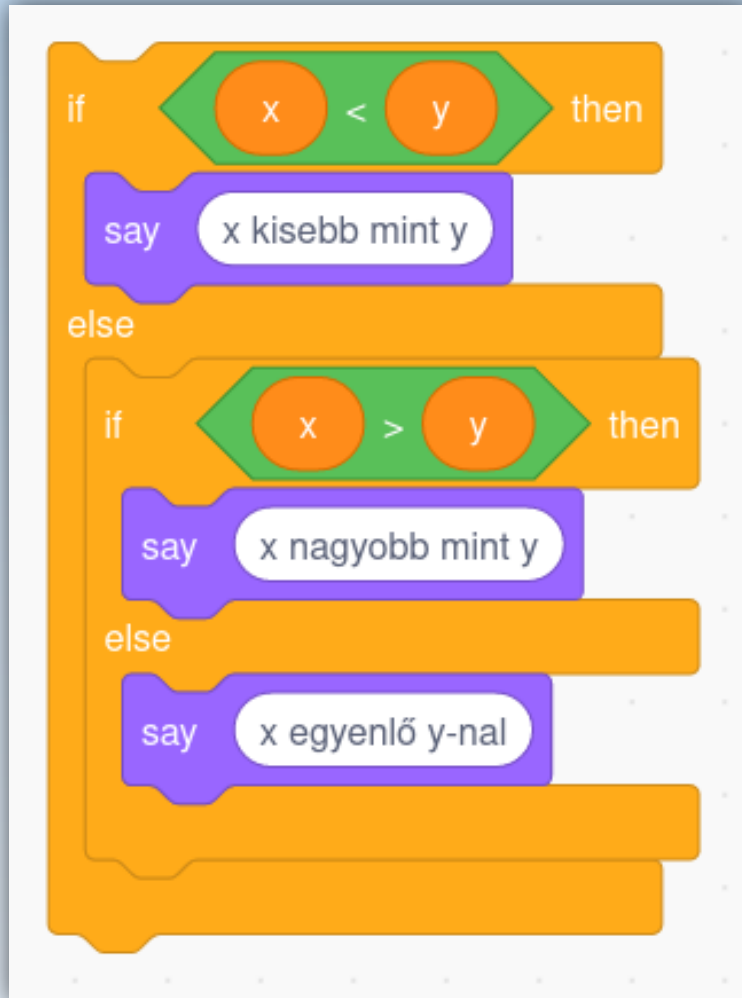
Az **else-if** utasítás (1. dia)



```
if (x < y)
{
    printf("x kisebb mint y\n");
}
else if (x > y)
{
    printf("x nagyobb mint y\n");
}
else if (x == y)
{
    printf("x egyenlo y-nal\n");
}
```

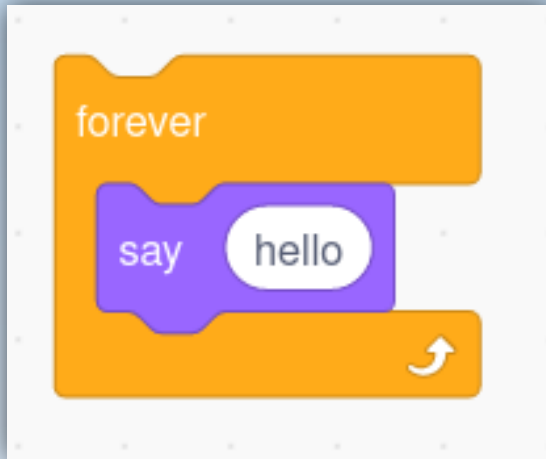
← Scratch-ben ez kicsit bonyolultabb

Az **else-if** utasítás (2. dia)



```
if (x < y)
{
    printf("x kisebb mint y\n");
}
else if (x > y)
{
    printf("x nagyobb mint y\n");
}
else
{
    printf("x egyenlo y-nal\n");
}
```


Végtelen ciklus **while** ciklussal



```
while (1)
{
    printf("hello\n");
}
```

C nyelv

Nincs külön logikai típus. A logikai igaz / hamis értéket egész számokkal adjuk meg:

0 – hamis

1 – igaz

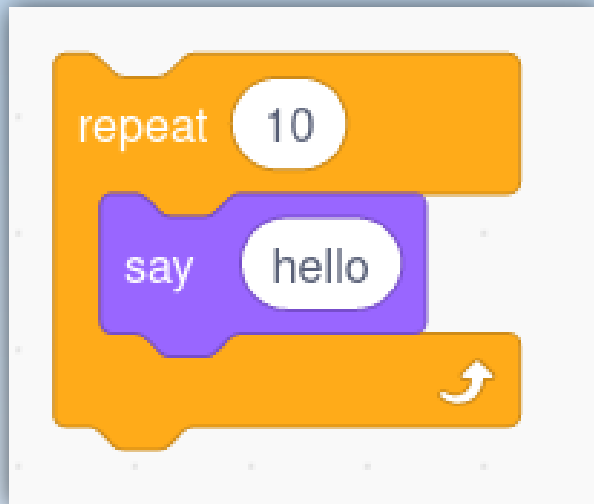
Más nyelvek

Sok más nyelvben van külön logikai típus (pl. `bool`). A logikai igaz / hamis értékeket nevesített konstanssal adjuk meg, pl.:

`True` – igaz

`False` – hamis

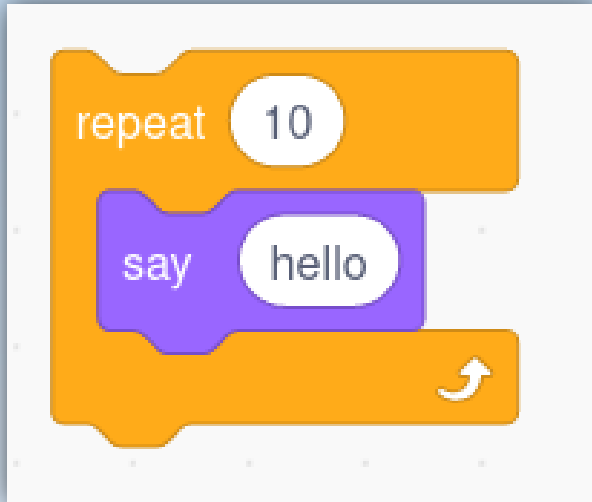
A **while** ciklus (1. dia)



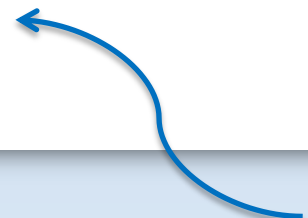
```
int i = 0;  
  
while (i < 10)  
{  
    printf("hello\n");  
}
```

Mi hiányzik innen?

A **while** ciklus (2. dia)



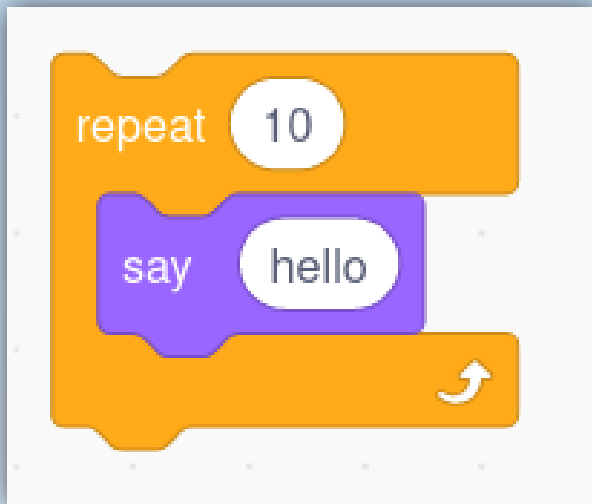
```
int i = 0;  
  
while (i < 10)  
{  
    printf("hello\n");  
    ++i;  
}
```



Ha nem figyelünk eléggé, akkor könnyen végtelen ciklus lesz a while ciklusból.

Végtelen ciklusba került program leállítása: Ctrl + C billentyűzetkombinációval.

A for ciklus



```
for (int i = 0; i < 10; ++i)
{
    printf("hello\n");
}
```

Ha előre tudjuk, hogy hányszor akarjuk lefuttatni a ciklust, akkor előnyösebb a **for** ciklust használni.

Jelen esetben: „írd ki **10-szer** a ‘hello’ szöveget”.

Gyakori típusok

típus	„placeholder” (formátumkód)
char	%c
int	%d
long	%ld
float	%f
double	%lf
...	...

Példa

```
char c = 'L';  
printf("%c\n", c);
```

Aritmetikai műveletek

aritmetikai operátorok	művelet	példa
+	összeadás	$10 + 2 = 12$
-	kivonás	$10 - 3 = 7$
*	szorzás	$2 * 2 = 4$
/	osztás	$5 / 2 = 2$ $5.0 / 2.0 = 2.5$
%	modulo (osztás utáni maradék)	$10 \% 3 = 1$ $17 \% 5 = 2$

Példák

Kérjük be a felhasználótól, hogy hány éves,
majd írassuk ki, hogy legalább hány napos.

Példák

Kérjük be a felhasználótól, hogy hány éves,
majd írassuk ki, hogy legalább hány napos.

```
1 // aritmetikai művelet (szorzás)
2
3 #include <stdio.h>
4
5 int main()
6 {
7     int ev;
8     printf("Hany éves vagy?\n");
9     scanf("%d", &ev);
10
11     int napok = ev * 365;
12     printf("Akkor legalabb %d napos vagy.\n", napok);
13
14     return 0;
15 }
```



Példák

Kérjük be a felhasználótól egy termék ÁFA-mentes árát.
Írassuk ki a termék ÁFÁ-val terhelt árát.

Példák

Kérjük be a felhasználótól egy termék ÁFA-mentes árát.
Írassuk ki a termék ÁFÁ-val terhelt árát.

```
1  #include <stdio.h>
2
3  int main()
4  {
5      float ar;
6      printf("Ar AFA nelkul?\n");
7      scanf("%f", &ar);
8
9      printf("Teljes ar: %.2f\n", ar * 1.27);
10
11     return 0;
12 }
```



Példák

Kérjünk be a felhasználótól egy egész számot,
majd írassuk ki, hogy a számos páros-e vagy páratlan.

Példák

Kérjünk be a felhasználótól egy egész számot,
majd írassuk ki, hogy a számos páros-e vagy páratlan.

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int szam;
6      printf("szam: ");
7      scanf("%d", &szam);
8
9      if (szam % 2 == 0)
10     {
11         printf("paros\n");
12     }
13     else
14     {
15         printf("paratlan\n");
16     }
17
18     return 0;
19 }
```

Példák

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int x = 2;
6      int y = 3;
7
8      if (x < y)
9      {
10         printf("x kisebb mint y\n");
11     }
12     else if (x > y)
13     {
14         printf("x nagyobb mint y\n");
15     }
16     else
17     {
18         printf("x egyenlo y-nal\n");
19     }
20
21     return 0;
22 }
```

Két érték összehasonlítása.

Házi feladat:

Módosítsuk úgy a programot, hogy az x és y értékeket a felhasználótól kérjük be.

Példák

```
1  #include <stdio.h>
2
3  int main()
4  {
5      // karakter beolvasása
6      char c;
7      printf("Akarod folytatni?\n");
8      scanf("%c", &c);
9
10     if (c == 'i' || c == 'I')
11     {
12         printf("folytatás...\n");
13     }
14     else if (c == 'n' || c == 'N')
15     {
16         printf("stop\n");
17     }
18     else
19     {
20         printf("nem értem :(\n");
21     }
22
23     return 0;
24 }
```

|| : logikai VAGY

&& : logikai ÉS

! : logikai tagadás

Példák

```
1  #include <stdio.h>
2
3  void hello()
4  {
5      printf("hello\n");
6  }
7
8  int main()
9  {
10     for (int i = 0; i < 3; ++i)
11     {
12         hello();
13     }
14
15     return 0;
16 }
```

Saját függvény írása, majd a függvény meghívása.

* a hello() valójában egy eljárás, hiszen nem ad vissza értéket

Példák

```
1  #include <stdio.h>
2
3  void hello(int n)
4  {
5      for (int i = 0; i < n; ++i)
6      {
7          printf("hello\n");
8      }
9  }
10
11 int main()
12 {
13     hello(5);
14
15     return 0;
16 }
```

Saját függvény ellátása paraméterrel.

A hívás helyén egy argumentum segítségével be tudom állítani, hogy a függvény hogyan fusson le.

Többször is meghívhatom, más-más argumentumokkal.

* a hello() valójában egy eljárás, hiszen nem ad vissza értéket

Példák

```
1  #include <stdio.h>
2
3  void hello(int n)
4  {
5      for (int i = 0; i < n; ++i)
6      {
7          printf("hello\n");
8      }
9  }
10
11 int main()
12 {
13     hello(2);
14     printf("-----\n");
15     hello(4);
16
17     return 0;
18 }
```

Házi feladat:

A 14. soron hogyan tudnánk javítani? Egy vízszintes elválasztó vonal kiíratását hogyan lehetne általánosítani?

Példák



Példák



Példák

```
1  #include <stdio.h>
2
3  int main()
4  {
5      // pénzérmék száma
6      int n;
7      printf("n: ");
8      scanf("%d", &n);
9
10     for (int i = 0; i < n; ++i)
11     {
12         printf("0");
13     }
14     printf("\n");
15
16     return 0;
17 }
```

Példák



Példák



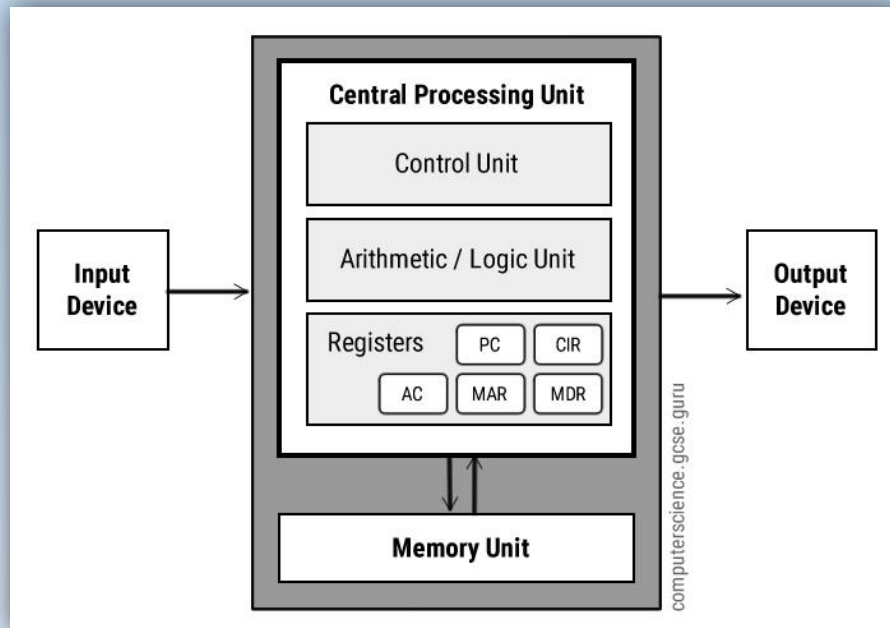
Példák

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int magassag = 10;
6      int szelesseg = 2;
7
8      for (int i = 0; i < magassag; ++i)
9      {
10         for (int j = 0; j < szelesseg; ++j)
11         {
12             printf("#");
13         }
14         printf("\n");
15     }
16
17     return 0;
18 }
```

Házi feladat:

Módosítsuk úgy a programot, hogy a magasságot és a szélességet a felhasználótól kérjük be.

Memória



Erőforrások: CPU, memória, háttértár.

A memória mérete **véges**. Az erőforrásokkal, így a memóriával is, hatékonyan kell gazdálkodni.

Példák

```
1  #include <stdio.h>
2
3  int main()
4  {
5      float x = 0.1;
6      float y = 0.2;
7
8      float result = x + y;
9      printf("Eredmeny: %.20f\n", result);
10
11     return 0;
12 }
```

Lásd:



IEEE 754 szabvány

<http://bit.ly/3bhmXlY>

Kimenet:

Eredmeny: 0.300000001192092895508

Fix méreten lebegőpontos számokat nem mindig lehet teljesen precízen tárolni!

Példák

Túlcsordulás

```
1  #include <stdio.h>
2  #include <unistd.h>
3
4  int main()
5  {
6      int i = 1;
7
8      while (1)
9      {
10         printf("%d\n", i);
11         i = i * 2;
12
13         sleep(1);
14     }
15
16     return 0;
17 }
```

67108864

134217728

268435456

536870912

1073741824

overflow1.c:11:11: runtime error: signed integer overflow: 1073741824 * 2 cannot be represented in type 'int'

-2147483648

0

0

0

0

Példák

Túlcsordulás



Y2K bug

1999



1900

Házi feladat

- A K & R-féle „C Bibliában” nézzék át azokat a részeket, amikről szó volt az előadáson.

Szorgalmi

- Haladjanak tovább a Linux operációs rendszerrel. Lejátszási lista: <http://bit.ly/31pRf7A> . Megtekintendő videók: 6, 7, 8, 9, 10. (A múlt héten az első öt videó volt feladva).
- Olvassanak utána a *Year 2038* problémának: <http://bit.ly/2vTNgVp> .