

# Programozási nyelvek 1

Szathmáry László  
Debreceni Egyetem  
Informatikai Kar

## 6. előadás

- random számok, nevesített konstansok
- tömb / sztring visszaadása függvényből

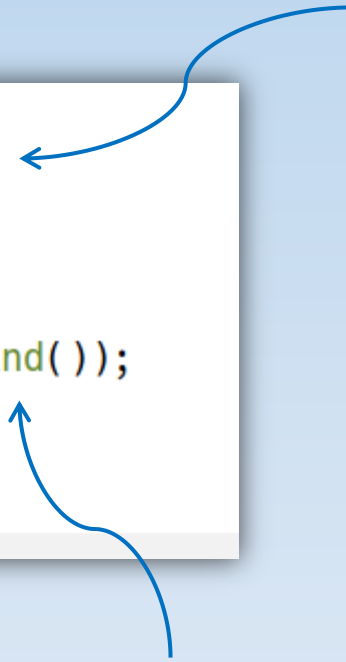
(utolsó módosítás: 2024. jan. 31.)

2023-2024, 2. félév



# Random számok

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      printf("%d\n", rand());
7
8      return 0;
9  }
10
```



# Random számok

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      srand(42);
7
8      for (int i = 0; i < 3; ++i)
9      {
10         printf("%d\t", rand());
11     }
12     puts("");
13
14     return 0;
15 }
16
```

*seed* érték

Ezzel inicializáljuk a véletlenszám-generátort.

# Random számok

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4
5  int main()
6  {
7      srand(time(NULL));
8
9      for (int i = 0; i < 3; ++i)
10     {
11         printf("%d\t", rand());
12     }
13     puts("");
14
15     return 0;
16 }
17
```

Minden indítás során  
más és más értékkel  
inicializáljuk a  
véletlenszám-generátort.

Következmény:

minden indítás során  
más és más  
véletlenszámokat  
kapunk

```
srand(42); // inicializálás valamilyen konstans értékkel
           // Mire jó? Egy kísérletet megismételhetővé tesz.
```

# Random számok

Véletlenszámok esetén gyakori igény, hogy egy adott intervallumból szeretnénk kapni egy random számot.

Pl.: generáljunk egy véletlen értéket a  $[70, 72]$  zárt intervallumból.

Fogalmazzunk precízen!

- **Generáljunk egy random számot 20 és 30 között.**
- Generáljunk egy random számot 20 és 30 között (a 30 is benne lehet).
- Generáljunk egy random számot a  $[20, 30]$  zárt intervallumon.
- **Generate a random number between 20 and 30.**
- Generate a random number between 20 and 30 (excl.).
- Generate a random number between 20 and 30 (incl.).

## Házi feladat:

- A) generáljunk ötöslottó számokat
- B) *finomítás*: a generált öt szám legyen **különböző**

# Random számok

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4
5  /*
6   *   Generáljunk egy véletlenszámot egy [also, felso]
7   *   zárt intervallumon.
8   */
9
10 int randint(int also, int felso)           // also: 70, felso: 72
11 {
12     int veletlen = rand();                 // pl. 3568
13     int intervallum = felso - also + 1;    // 72 - 70 + 1 = 3
14
15     veletlen = veletlen % intervallum;     // 3568 % 3 = 1
16     veletlen = also + veletlen;           // 70 + 1 = 71
17
18     return veletlen;                      // 71
19 }
20
21 int main()
22 {
23     srand(time(NULL));
24
25     printf("%d\n", randint(70, 72));
26
27     return 0;
28 }
29
```

# Nevesített konstansok

A nevesített konstans olyan programozási eszköz, amelynek három komponense van:

1. név
2. típus
3. érték

# Nevesített konstansok

```
#define FENYSEBESSEG_M_S 299792458
```



```
int main()  
{  
    int eredmeny = 2 * FENYSEBESSEG_M_S;  
  
    return 0;  
}
```



# Nevesített konstansok

```
#define FENYSEBESSEG_M_S 299792458
```

```
int main()  
{  
    int eredmeny = 2 * FENYSEBESSEG_M_S;  
  
    return 0;  
}
```

# Nevesített konstansok

```
int main()  
{  
    int eredmeny = 2 * 299792458;  
  
    return 0;  
}
```

Az előfeldolgozó által produkált kimenet (módosított C forráskód) megtekintése:

```
$ gcc -E input.c
```

# Nevesített konstansok

Egy másik módszer:

```
#include <stdio.h>
```

```
const int FENYSEBESSEG_M_S = 299792458;
```




```
int main()  
{  
    int eredmeny = 2 * FENYSEBESSEG_M_S;  
    printf("%d\n", eredmeny);  
  
    return 0;  
}
```

# Tömb visszaadása függvényből

1. Egy függvényből nem tudunk visszaadni egy *lokális* tömböt, mivel a függvény visszatérésekor felszabadul a tömb által lefoglalt memóriaterület.
2. Használhatnánk dinamikus memórafoglalást, de erről később lesz szó.
3. **Jelenlegi megoldás:** a tömböt a hívó oldalon deklarálom, s átadom egy másik fv.-nek (vagy eljárásnak), ami közvetlenül módosítja a tömböt. A meghívott fél befejeződése után látni fogja a hívó fél a módosításokat.

# Tömb visszaadása függvényből

```
1  #include <stdio.h>
2
3  void array_10(int n, int tomb[])
4  {
5      for (int i = 0; i < n; ++i)
6      {
7          tomb[i] = i + 1;
8      }
9  }
10
11 int main()
12 {
13     // Kell nekem egy tömb, ami egészeket tartalmaz 1-től 10-ig (a 10-et is).
14
15     int szamok[10];
16     int meret = 10;
17
18     array_10(meret, szamok);
19
20     for (int i = 0; i < meret; ++i)
21     {
22         printf("%d\t", szamok[i]);
23     }
24     puts("");
25
26     return 0;
27 }
28
```



# Sztring visszaadása függvényből

```
1  #include <stdio.h>
2
3  #define MAX 100
4
5  /*
6   Írjunk függvényt, ami visszaadja a "Bea" sztringet.
7   */
8
9  void insert_name(int n, char s[])
10 {
11     s[0] = 'B';
12     s[1] = 'e';
13     s[2] = 'a';
14     s[3] = '\0';
15 }
16
17 int main()
18 {
19     char szoveg[MAX];
20     int meret = MAX;
21
22     insert_name(meret, szoveg);
23
24     printf("%s\n", szoveg);
25
26     return 0;
27 }
```

manuális  
módszer



# Sztring visszaadása függvényből

```
1  #include <stdio.h>
2  #include <string.h>
3
4  #define MAX 100
5
6  /*
7   * Írjunk függvényt, ami visszaadja a "Bea" sztringet.
8   */
9
10 void insert_name(int n, char s[])
11 {
12     strcpy(s, "Bea");
13 }
14
15 int main()
16 {
17     char szoveg[MAX];
18     int meret = MAX;
19
20     insert_name(meret, szoveg);
21
22     printf("%s\n", szoveg);
23
24     return 0;
25 }
```

# Házi feladat

- A K & R-féle „C Bibliában” nézzék át azokat a részeket, amikről szó volt az előadáson.
- Juhász István jegyzetéből nézzék át azokat a fogalmakat, amikről szó volt az előadáson ([link](#)).