Programozás C nyelven (3. ELŐADÁS)

Sapientia EMTE

2020-21


EMPIRE

- **Classic Empire - A turn Based Wargame**
  - Classic Empire is a real time, multi-player, Internet-based game, featuring military, diplomatic, and economic goals. Empire is a game that is played against human opponents over a computer network, usually the Internet. It is possible for a game to last from a few hours to many months.


```
void pf_check(void)
{
   int i, uid, c;

   for (i = 0; i < pf_nheap; i++) {
      uid = pf_heap[i].uid;
      assert(0 <= uid && uid < WORLD_SZ());
      assert(pf_map[uid].heapi == i);
      assert(pf_map[uid].visit == pf_visit);
      assert(pf_map[uid].cost <= pf_heap[i].cost);
      c = 2 * i + 1;
      assert(c >= pf_nheap || pf_heap[i].cost <= pf_heap[c].cost);
      c++;
      assert(c >= pf_nheap || pf_heap[i].cost <= pf_heap[c].cost);
   }

   for (uid = 0; uid < WORLD_SZ(); uid++) {
      assert(pf_map[uid].visit <= pf_visit + 1);
      if (pf_map[uid].visit == pf_visit) {
         i = pf_map[uid].heapi;
         assert(0 <= i && i < pf_nheap && pf_heap[i].uid == uid);
      }
   }
}
```

```
while ((opt = getopt(argc, argv, "2:krs:uhv")) != EOF) {
                switch (opt) {
                case '2':
                   auxfname = optarg;
                   break;
                case 'k':
                   send_kill = 1;
                   break;
                case 'r':
                   restricted = 1;
                   break;
                case 's':
                   port = strdup(optarg);
                   colon = strrchr(port, ':');
                   if (colon) {
                                  *colon = 0;
                                  host = port;
                                  port = colon + 1;
                   }
                   break;
                case 'u':
                   utf8 = eight_bit_clean = 1;
                   break;
                case 'h':
                   print_usage(argv[0]);
                   exit(0);
                case 'v':
                   printf("%s\n\n%s", version, legal);
                   exit(0);
                default:
                   fprintf(stderr, "Try -h for help.\n");
                   exit(1);
                }
}
```

EMLÉKEZTETŐ / KIEGÉSZÍTŐ

- **VÁLTOZÓK:**
  - `<típus> <azonosító>;`
    - `char, int, float, double`
    - `bool`
- **IN/OUT**
  - `scanf / printf`
  - `freopen("…","r",stdin);`
- **SZEKVENCIA**
- **ELÁGAZÁS**
  - `if-else`
  - `< > ? < > : < >`
  - `switch`

Aritmetikai operátorok

`+, -, *, /, %`

Összetett operátorok

`+=, -=, *=, /=, %=`

```
x = x + 3; ⇔ x += 3;
i = i - 1; ⇔ --i;
++i vs. i++
```

Összehasonlítási operátorok

`==, !=, <, <=, >, >=`

Logikai operátorok

`!, &&, ||`

Négy (n) szám összege!

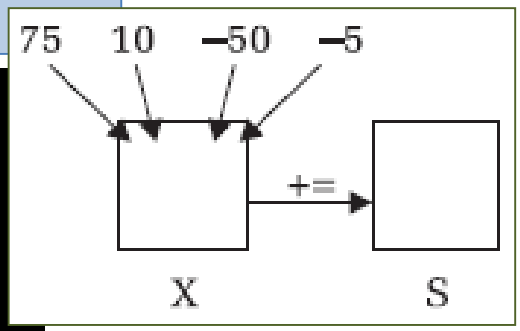
WATCH	
x	s
?	0
75	75
10	85
-50	35
-5	30

```
int x1,x2,x3,x4,s;
scanf("%i%i%i%i",&x1,&x2,&x3,&x4);
s = x1 + x2 + x3 + x4;
printf("s= %i", s);
```

?

```
int x,s=0;
scanf("%i",&x); s += x;
scanf("%i",&x); s += x;
scanf("%i",&x); s += x;
scanf("%i",&x); s += x;
printf("s= %i", s);
```

```
75
10
-50
-5
s = 30_
```



Összeg-számítás

```
int i,x,s=0;
for ( i=1 ; i<=4 ; ++i ){
    scanf("%i",&x); s += x;
}
printf("s= %i", s);
```

```
int i,n,x,s=0;
scanf("%i",&n);
for ( i=1 ; i<=n ; ++i ){
    scanf("%i",&x); s += x;
}
printf("s= %i", s);
```

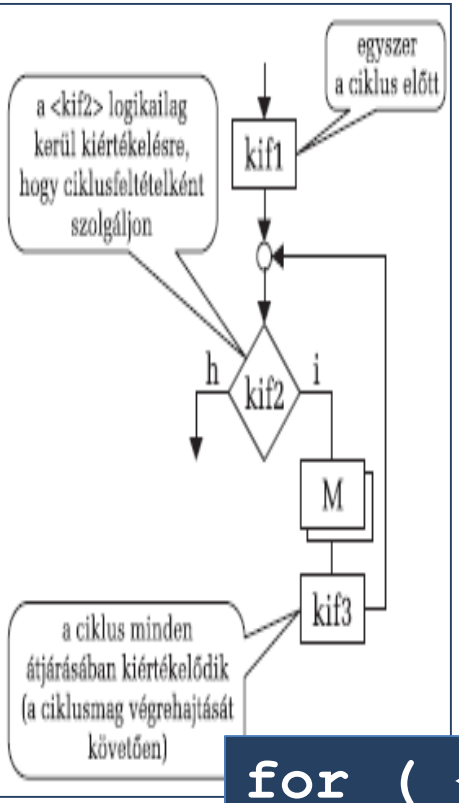
`for` – ciklus

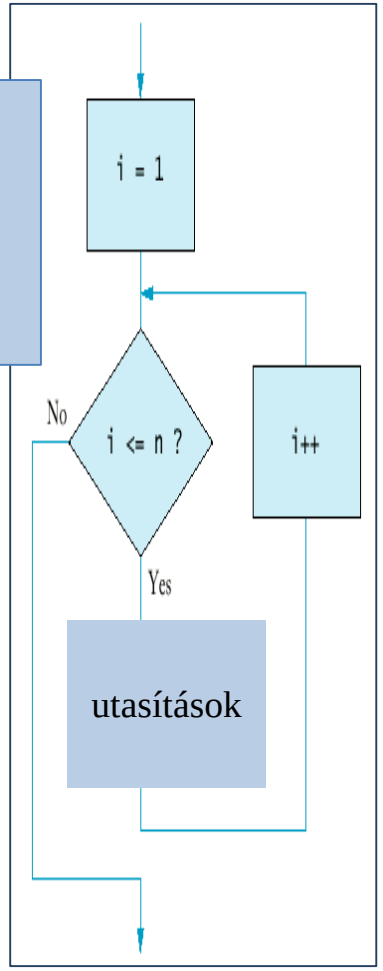
i ← 1, 2, …, n

```
for ( i=1 ; i<=n ; ++i ){
   <utasítások>
}
```

i ← 0, 1, …, n-1

```
for ( i=0 ; i<n ; ++i ){
   <utasítások>
}
```






```
for ( <kif1> ; <kif2> ; <kif3> ){
   <ciklus mag>
}
```

Számjegy-szorzat! [100 .. 999]

WATCH	
x	p
?	1
275	1
27	5
2	35
0	70

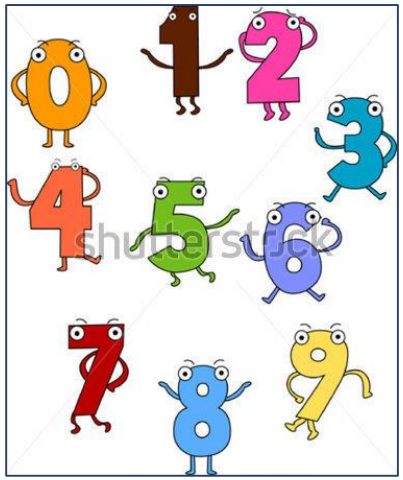
?

```
int x,p;
scanf("%i",&x);
p = x/100 * x/10%10 * x%10;
printf("p= %i", p);
```

275 275 275

```
int x,p=1;
scanf("%i",&x);
p *= x%10; x /= 10;
p *= x%10; x /= 10;
p *= x%10; x /= 10;
printf("p= %i", p);
```

```
275
p = 70_
```



$\overline{d_n d_{n-1} \ldots d_1 d_0}$

$\overline{d_n d_{n-1} \ldots d_1 \not{d}_0}$

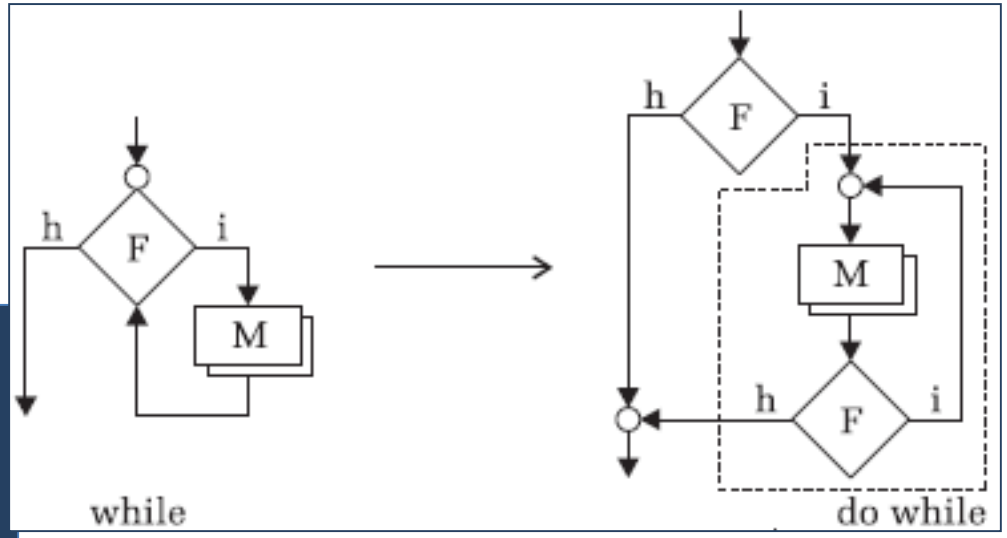
Szorzat-számítás

```
int x,p=1;
scanf("%i",&x);
while ( x>0 ){
    p *= x%10; x /= 10;
}
printf("p= %i", p);
```

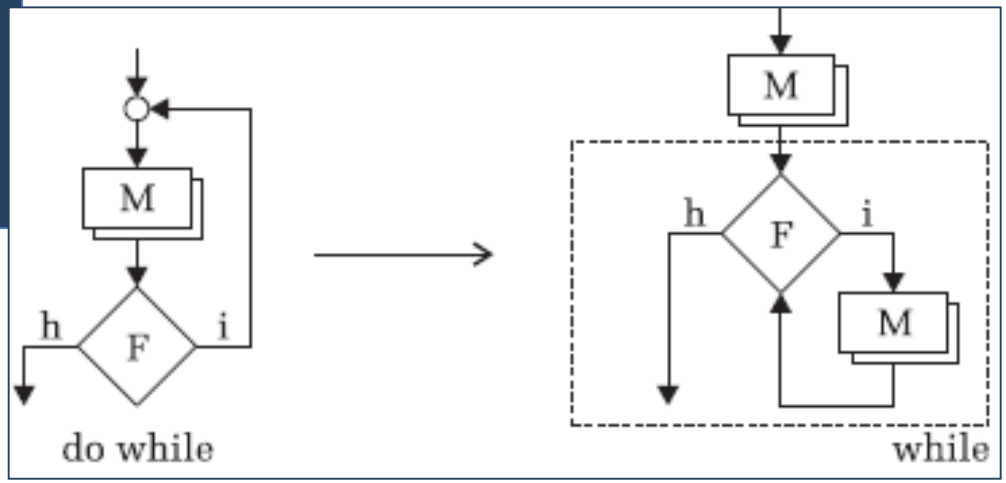
?

`while` / `do-while` – ciklusok

```
while ( <feltétel> ){
   <utasítások>
}
```




```
do {
   <utasítások>
} while (<feltétel>);
```



break / continue / goto

Természetes számokat olvastassunk be nulla végjelig, majd íratassuk ki az átlagukat.

Végtelen ciklus

Kiugortat a kurrens ciklusból

```
unsigned i;
double szam, osszeg = 0;
for ( i=0 ; ; ++i ){
   scanf("%lf", &szam);
   if ( !szam ) { break; }
   osszeg += szam;
}
if (i){ printf( "Atlag = %.2lf", osszeg/i ); }
else { printf( "Nem volt szam" ); }
```

Átlag-számítás

break / continue / goto

Egész számokat olvastassunk be nulla végjelig, majd íratassuk ki a pozitívak átlagát.

Végtelen ciklus

Befejezi a kurrens ciklusmag-átjárást

```
int i=0;
double szam, osszeg = 0;
while ( 1 ){
   scanf("%lf", &szam);
   if ( !szam ) { break; }
   if ( szam < 0 ) { continue; }
   osszeg += szam;
   ++i;
}
if (i){ printf( "Atlag = %.2lf", osszeg/i ); }
else { printf( "Nem volt szam" ); }
```

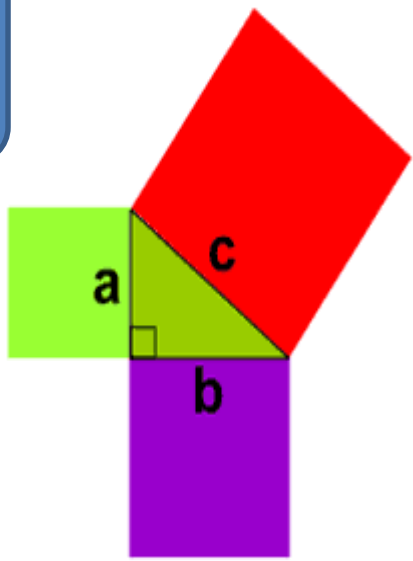
Átlag-számítás

break / continue / goto

Íratassuk ki a „legkisebb” kétszámjegyű, legfentebb 100 kerületű, pitágorászi számhármast.

```
#include <stdio.h>
#define K 100
int main(){
    unsigned short a, aa, b, bb, c, cc;
    for ( a = 10 ; a <= K/3; ++a ){
        aa = a * a;
        for ( b = a+1 ; b < K/2; ++b ){
            bb = b * b;
            for ( c = b+1 ; c < K/2; ++c ){
                cc = c * c;
                if (cc > aa + bb){ break; }
                if (a + b + c > K){ break; }
                if (cc == aa + bb){ goto cimke; }
            }
        }
    }
    cimke: printf("%hu %hu %hu\n", a, b, c);
    return 0;
}
```

Ugorj a címkézett utasításhoz



Van-e nulla érték n szám között?

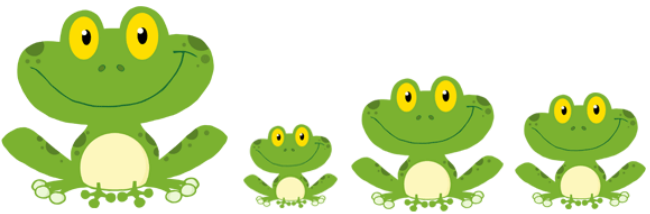
```
#include <stdio.h>
#include <stdbool.h>
int main(){
    freopen("szamsor.txt", "r", stdin);
    int x, n, i;
    scanf("%i", &n);
    bool b = false;
    for ( i = 1 ; i <= n ; ++i){
        scanf("%i", &x);
        if ( x == 0 ) {b = true; break;}
    }
    if ( b == true ) { printf("VAN"); }
    else { printf("NINCS"); }
    freopen("CON", "r", stdin);
    return 0;
}
```

szamsor.txt

```
10
22 3 -5  0 6 0 123 77 -2 512
```

```
VAN_
```

Létezés-ellenőrzés



N szám minimuma

WATCH

n	i	x	min
?	?	?	?
4	?	?	13
4	2	3	3
4	3	6	3
4	4	5	3
4	5	5	3

```
#include <stdio.h>
#include <stdbool.h>
int main(){
    int x, min, n, i;
    scanf("%i", &n);
    scanf("%i", &min);
    for ( i = 2 ; i <= n ; ++i){
        scanf("%i", &x);
        if (x < min) {min = x;}
    }
    printf("MIN = %i", min);
    return 0;
}
```

```
4
13
3
6
5
MIN = 3_
```

Minimum-keresés

ISMÉTLÉS

- **SZEKVENCIA**
- **ELÁGAZÁS**
- **CIKLUSOK**
  - `for`
  - `while`
  - `do - while`
  - `break / continue`

- Összeg / szorzat / átlag számítás technikája
- Számlálás technikája
- Létezés-ellenőrzés technikája
- Min-keresés technikája

- Számsorozat elemenkénti feldolgozása
  - adott elemszámú sorozat
  - végjelig olvasott sorozat
- Természetes szám számjegyenkénti feldolgozása