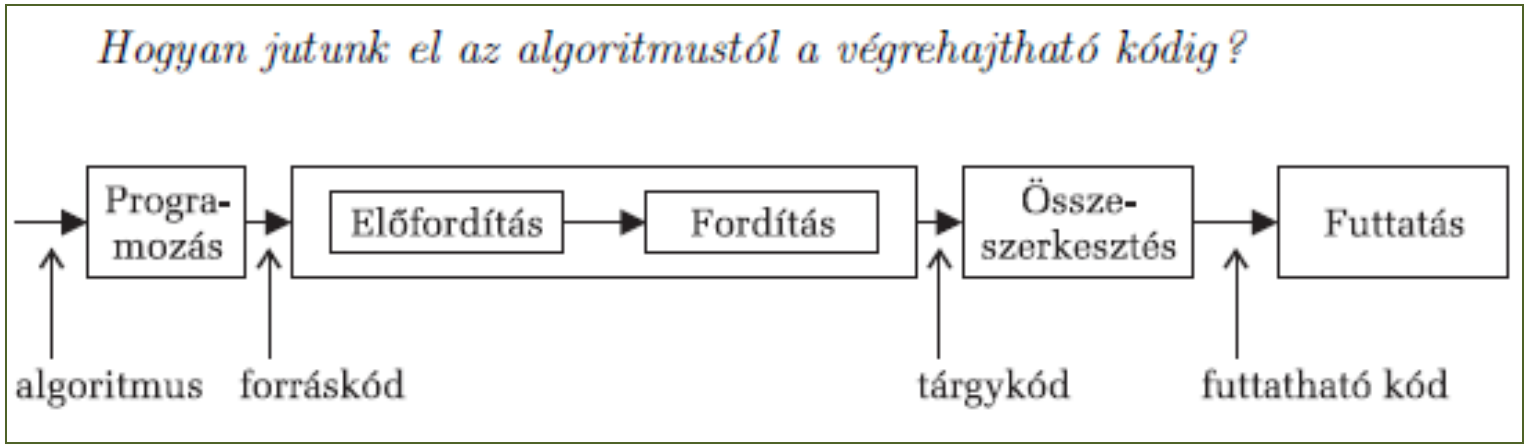
Programozás C nyelven
FELÜLNÉZETBŐL
elhullatott MORZSÁK

Sapientia EMTE

2015-16

Felülnézet – 1



- Feltételes fordítás
  - `#if, #else, #elif, #endif,`
  - `#ifdef, #ifndef`
  - `stb.`

Felülnézet – 2

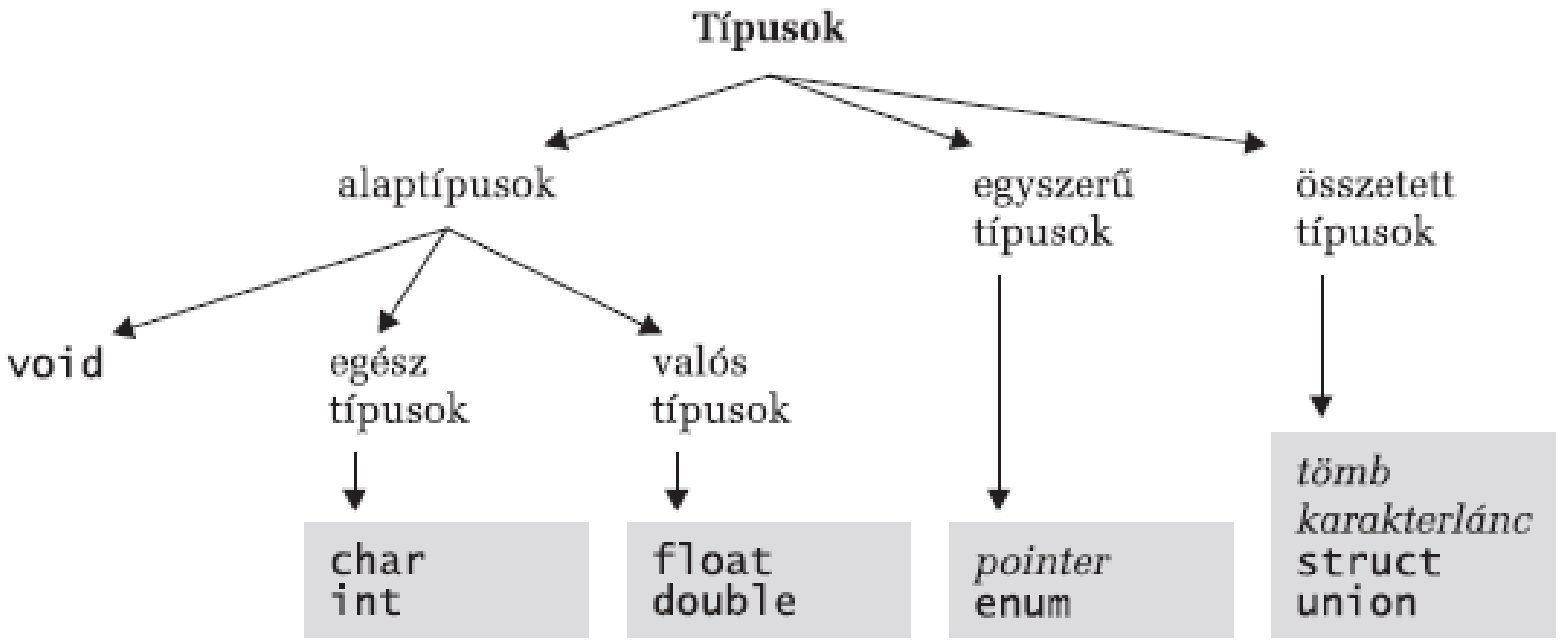
Egy C program lehetséges szerkezete:

```
<preprocesszor direktívák>
  #include direktívák
  #define direktívák
<globális definiálások>
  típusdefiniálások (typedef)
  változó definiálások
  sajátfüggvény deklarálások (prototípusok)
<függvénydefiníciók>
  main függvény
  sajátfüggvények
```

- `#include`: hatására a preprocesszor beépíti, logikailag, a header-állományt a forráskódba

Felülnézet – 3

Az alábbi ábra átfogó képet nyújt a C nyelv adattípusairól:



Típusmódosítók az alaptípusokhoz: `signed`, `unsigned`, `short`, `long`.

Módosító jelzők: `const`, `volatile`, `typedef`

?

Változók (1)

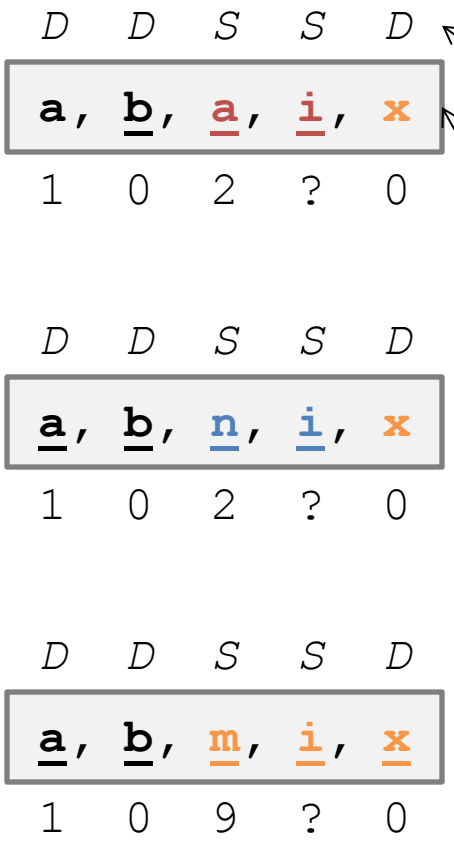
- azonosító, típus, cím, érték…
- **memóriaosztály** (hol kerül eltárolásra?)
  - adatszegmens/regiszter/STACK/HEAP
- **láthatósági tartomány** (mely programsorokból érhető el? hol definiáltuk?)
  - blokk/modul(állomány) szintű
- **élettartam** (meddig létezik? a tárolási osztálya határozza meg)
  - statikus/lokális/dinamikus

adatszegmensen | regiszter/STACK | HEAP

Változók (2)

- Globális / Lokális / Statikus változók

```
int a = 1, b;
void f1(int);
void f2(int);
int main(){
    int a = 2, i;
    ...
    f1(a); f2(9);
}
void f1(int n){
    int i;
    ...
}
void f2(int m){
    int i;
    static int x;
    ...
}
```



D: Adatszegmens
S: Stack

Hol léteznek?

Melyikek léteznek?

Melyikek láthatók?

Mennyi az értékük?

Változók (2a)

- **Globális változók**
  - minden függvényen kívül definiáljuk;
  - létrejönnek már a programfutás előtt, az adatszegmensen, és léteznek a teljes programfutás alatt;
  - implicit statikus élettartamúak;
  - implicit inicializálva vannak nullával;
  - láthatósági tartományuk a modul (állomány), amelyben definiáltuk őket, és elérhetőek bárhol, a definiálásuk helyétől lefele az állomány végéig;
  - láthatósági tartományuk kiterjeszthető más modulokra is (`extern`).

```
1. állomány

int a = 10; /* definíció */
void nyomtat(); /* deklaráció */
main()
{
    nyomtat(); return 0;
}
```

```
2. állomány

#include <stdio.h>
extern int a; /* deklaráció */
void nyomtat() /* definíció */
{
    printf("%d", a);
}
```

Változók (2b)

- **Lokális változók**
  - definiálásuk valamely blokkban történik, és láthatósági tartományuk az illető blokk területe, anélkül hogy kiterjeszthető lenne;
  - általában a STACK-en jönnek létre, de ha definiálásukat megelőzi a `register` kulcsszó, akkor lehetőség szerint a mikroprocesszor valamelyik regiszterében;
  - tárhelyfoglalás számukra programfutás közben történik, amikor a programfutás a definiálásukhoz érkezik;
  - az illető blokk befejeztével felszámolódnak.

```
#include <stdio.h>
int f ()
{
    int b = 3, a = b;
    return ++a;
}
```

```
main()
{
    printf ("a = %d\n", f());
    printf ("a = %d", f());
    return 0;
}
```

Változók (2c)

- **Statikus változók**
  - létrejönnek már a programfutás előtt, az adatszegmensen, és léteznek a teljes programfutás alatt;
  - implicit inicializálva vannak nullával;
  - láthatósági tartományuk nem terjeszthető ki;
  - ha minden függvényen kívül definiáltuk őket, akkor láthatósági tartományuk modulszintű (külső statikus változó)[miben különböznek a globális változóktól?];
  - ha valamely blokkban definiáltuk őket, akkor blokkszintű a láthatósági tartományuk (belső statikus változó) [miben különböznek a lokális változóktól?].

```
#include <stdio.h>
static double a;
main()
{
 ++a;
 printf("a=%.21f",a);
 return 0;
}
```

```
#include <stdio.h>
main()
{
 {static double a;}
 ++a;
 printf("a=%21f",a);
 return 0;
}
```

```
#include <stdio.h>
int f ()
{static int a;
return ++a;}
main()
{
 printf("a=%d\n",f();
 printf("a=%d",f());
 return 0;
}
```

Utasítások, kifejezések

I. SZEKVENCIA
- <kifejezés>;(1)
  - értékadás

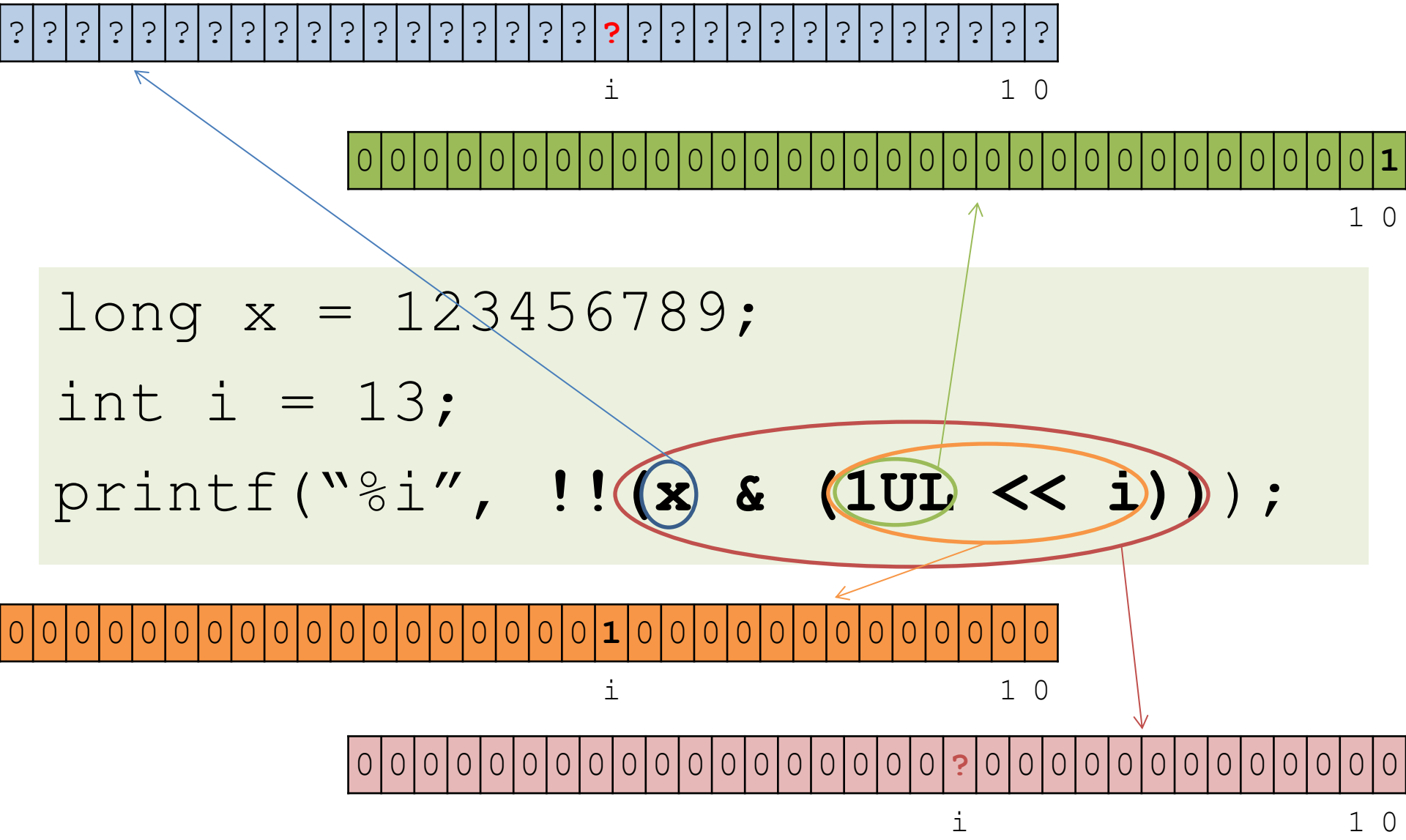
II.ELÁGAZÁS
- if-else (2)
- switch (3)

III.CIKLUSOK
- for (4)
- while (5)
- do-while (6)
  - break(7), continue(8), goto(9)

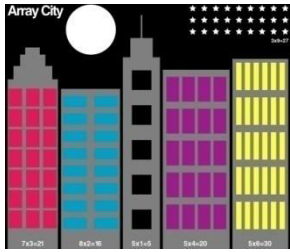
```
//olvasás 0 végjelig
int x;
while( scanf("%i", &x), x ){
    …
}
```

```
//olvasás állományvégig
char c;
while( fscanf(fin, "%c", &c) != EOF ){
    …
}
```

Bit-műveletek: <<, >>, &, |, ^, ~



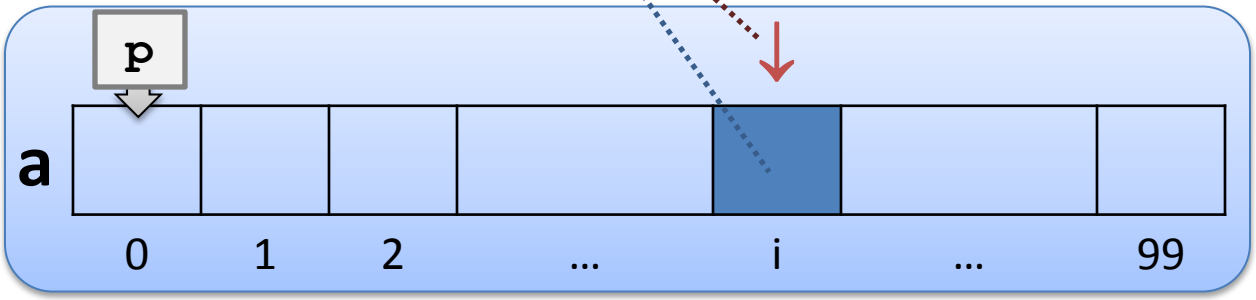
Pointerek és tömbök
Dinamikus helyfoglalás




```
int a[100], *p = a;
```

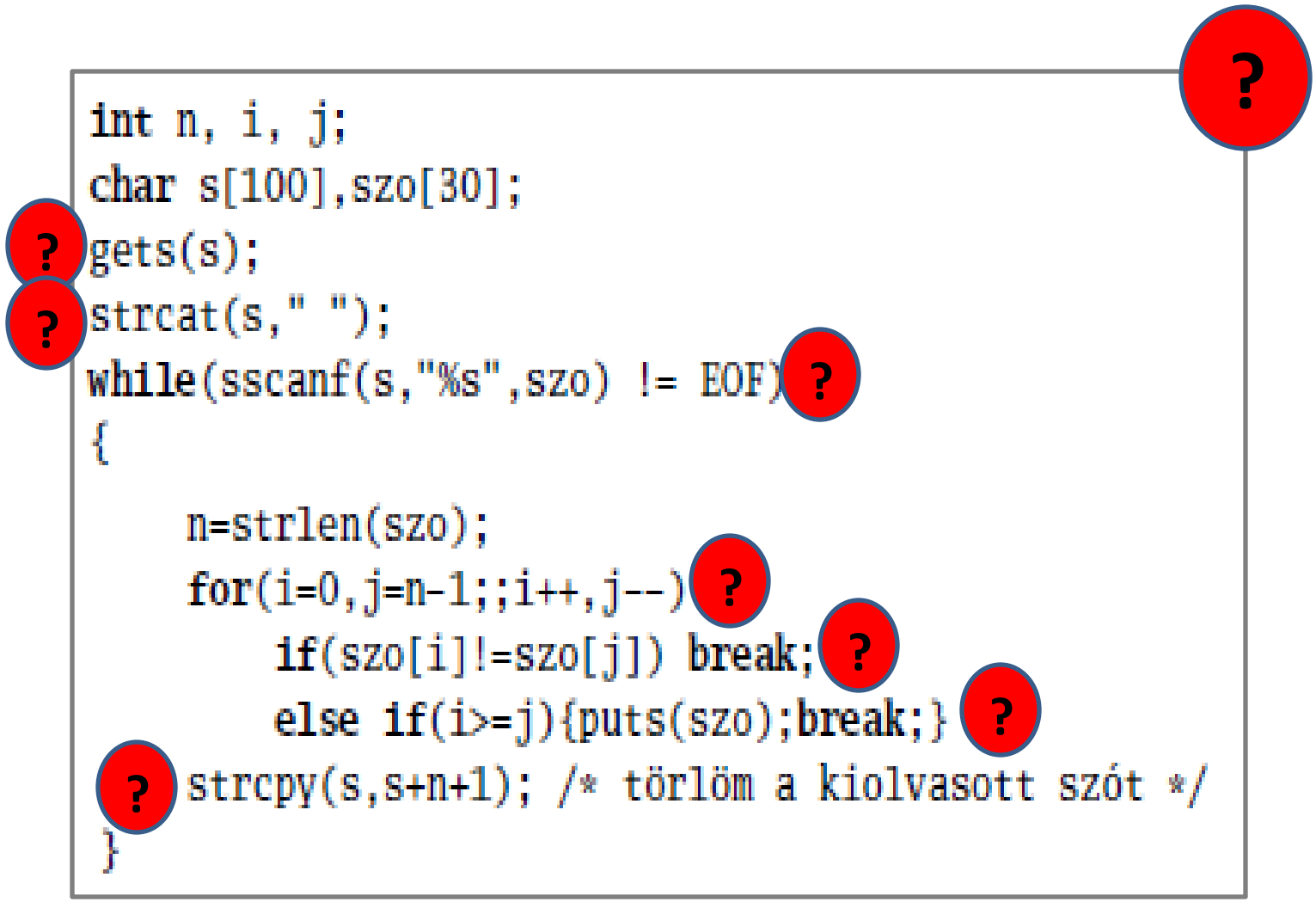
Hivatkozás az i. elemre:
`a[i] ⇔ *(a+i) ⇔ *(p+i) ⇔ p[i]`

Hivatkozás az i. elem címére:
`&a[i] ⇔ a+i ⇔ p+i ⇔ &p[i]`



- `int a[10][10][10]; //a[i][j][k]`
- `int *b = (int*)malloc(n*m*sizeof(int));`
  `// b[i*m+j]`
- `int (*c)[3] = (int(*)[3])malloc(n*3*sizeof(int));`
  `// c[i][0], c[i][1], c[i][2]`

Karakterláncok: sscanf/sprintf



struct/union/enum

8.6. feladat. Olvassunk be a billentyűzetről n (n<=50) természetes számot az unsigned short tartományból, amelyeknek unsigned short típusúkénti belső ábrázolásuk személyek születési dátumainak kódjai, a következőképpen:

7 bit – az év utolsó két számjegye
4 bit – a hónap
5 bit – a nap

```
typedef struct
{
    unsigned ev:7;
    unsigned ho:4;
    unsigned nap:5;
} datum;
typedef union
{
    unsigned short kod;
    datum d;
} szemely;
```

```
szemely x;
scanf("%hu", &x.kod);
if( x.d.ev == 68 ){…}
```

```
enum{
    marcius = 3;
    junius = 6;
    szeptember = 9
    december = 12;
};
…
if(ho>=december || ho<marcius){…}
…
```

Függvények

- Függvény-pointerek
  - `int (*fp)(char, float);`
  - ```
    void mysort(
      void *v,
      int nr,
      int size,
      int(*p_cmp)(const void*,const void*)
    )
    ```
- Változó paraméterszámú függvények
  - `void valt_par_fugg(int n, ...);`
- Parancssor argumentumok
  - `int main(int argc, char *argv[]){ … }`
    ```

Makrók

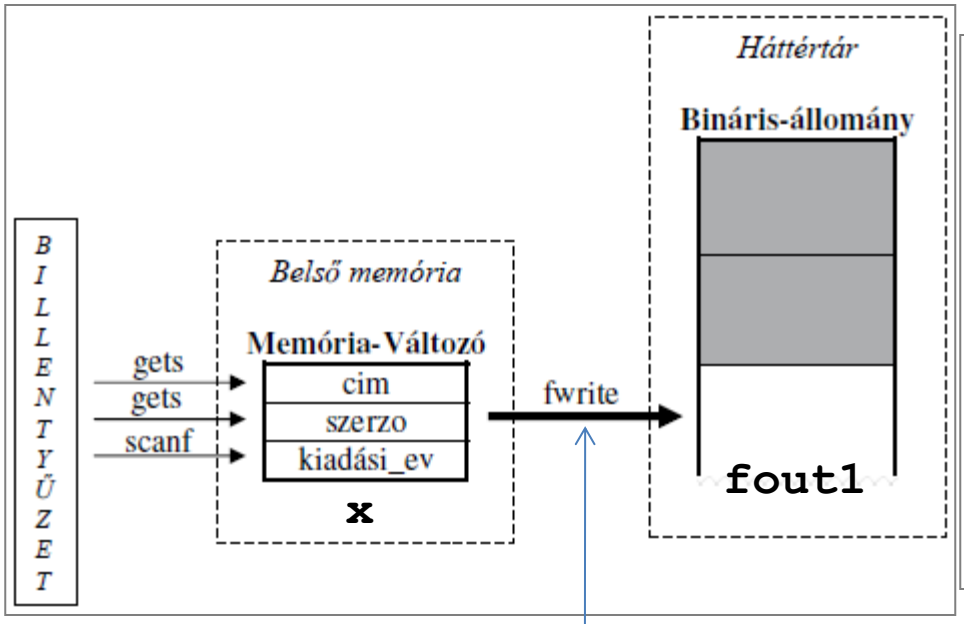
(Hogyan működik az alábbi program?)

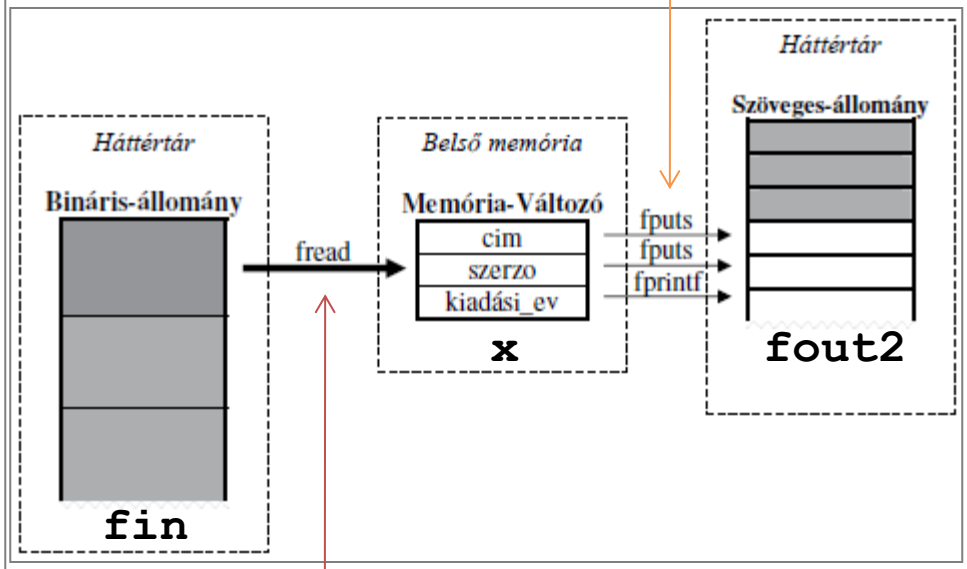
```
#include <stdio.h>
#define def_fn(name, operation)\
int name(int a, int b){\
   int c = a operation b;\
   printf("%d %s(%s) %d = %d\n",a,#name,#operation,b,c);\
   return c;\
}\
def_fn(plus,+); def_fn(minus,);
main(){
   plus(1,1);
   minus(1,1);
   return 0;
}
```

Állománykezelés
(könyvtárkezelő program)

```
typedef struct
{
    char cim[50],szerzo[50];
    int kiadasi_ev;
}KONYV;
```

```
fprintf(fout2, "%s\n%s\n%i\n",
        x.cim, x.szerzo, x.kiadasi_ev);
```






```
fread(&x,sizeof(KONYV),1,fin);
```

```
fwrite(&x,sizeof(KONYV),1,fout1);
```


VÉGE