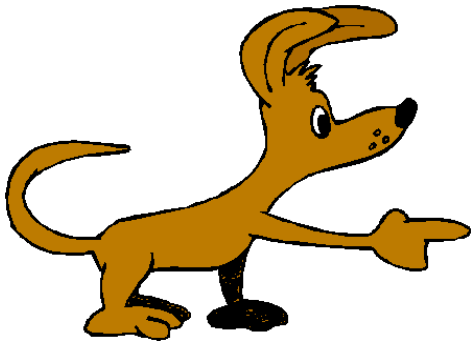
Programozás C nyelven (8. ELŐADÁS)

Sapientia EMTE

2020-21

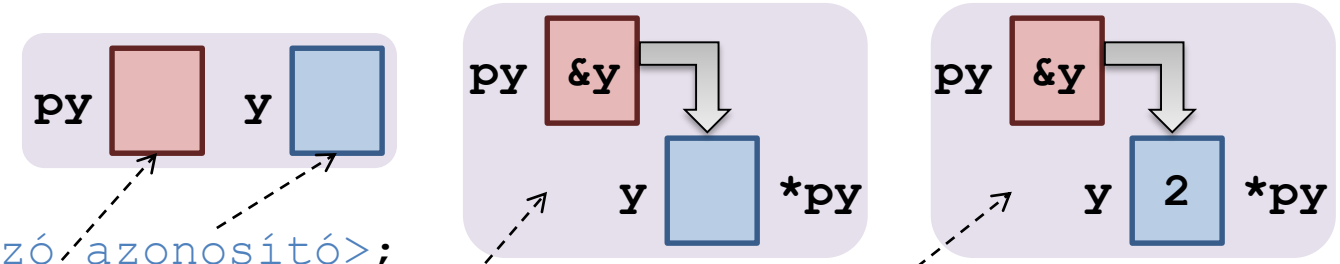
Mutatók (POINTER-ek)



MUTATÓ – típus

A mutatók (pointer-ek) változó-*címeket* tárolhatnak

Azt mondjuk, hogy a pointer *mutat* az illető című változóra



Szintaxis:

```
<típus> <változó_azonosító>;
<típus> *<pointer_változó_azonosító>;
<pointer_változó_azonosító> = &<változó_azonosító>;
*<pointer_változó_azonosító> = <érték>;
```

Hivatkozás egy változó címére:

```
&<változó_azonosító>
```

Hivatkozás a címzett változóra:

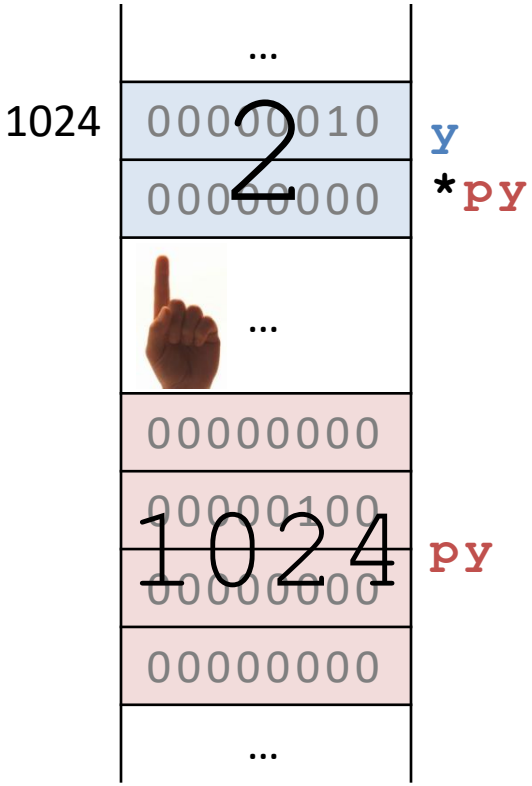
```
*<pointer_változó_azonosító>
```

Példák:

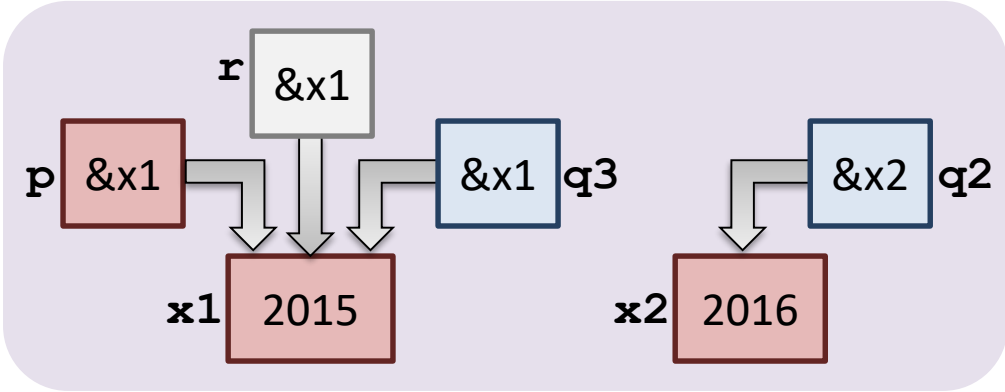
```
int *p; // p:int-pointer; egy int-változó címét tárolhatja
double *q1, x, *q2; // sizeof(x)=?, sizeof(q1)=?
short y=2, *py=&y; // y ⇔ *py
long a=13, *pa=&a, **ppa=&pa; // a ⇔ *pa ⇔ **ppa
```

Mutatók a memóriában

```
short y=2, *py=&y;
```



Tanulás PÉLDÁKon




```
long x1,x2,*p;
float y,*q1,*q2,*q3;
p = &x1; // OK
q1 = &y; // OK
q2 = &x2; // nincs implicit típus-konverzió
q2 = (float*)(&x2); // CAST-elés
x1 = 2015; printf("%li\n%li\n", x1, *p);
x2 = 2016; printf("%li\n%f\n", x2, *q2);
```

```
void *r;
r = p; printf("%li\n", *(long*)r);
q3 = (float*)r; printf("%f\n", *q3);
```

```
2015
2015
2016
2.82502e-042
2015
2.82362e-042
_
```

Műveletek POINTER-ekkel

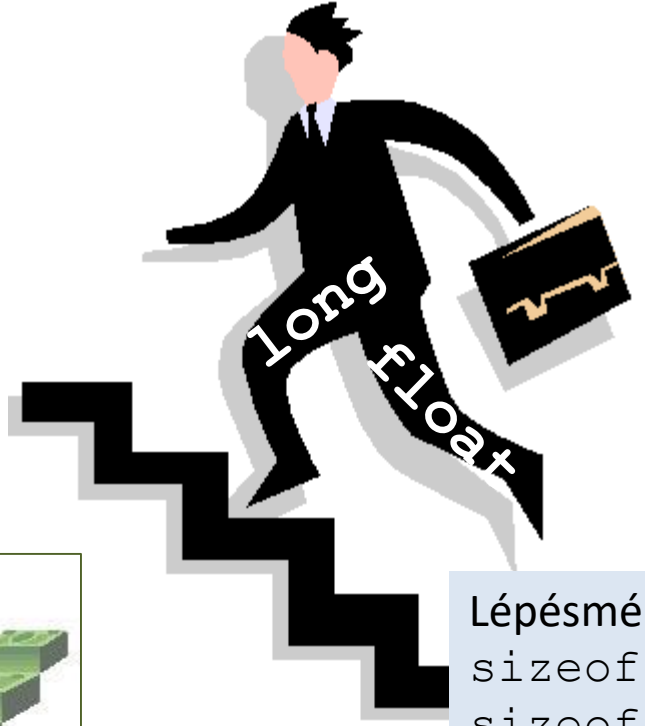
- **<pointer> + <egész_érték>**
  ++ <pointer>
- **<pointer> - <egész_érték>**
  -- <pointer>

- **<pointer_1> - <pointer_2>**

- **<pointer_1> == <pointer_2>**
  <, >, <=, >=, ==, !=

<pointer_1> és <pointer_2> azonos típusúak

Lépésméret:
`sizeof(char)=1`



Lépésméret:
`sizeof(long)=4`
`sizeof(float)=4`

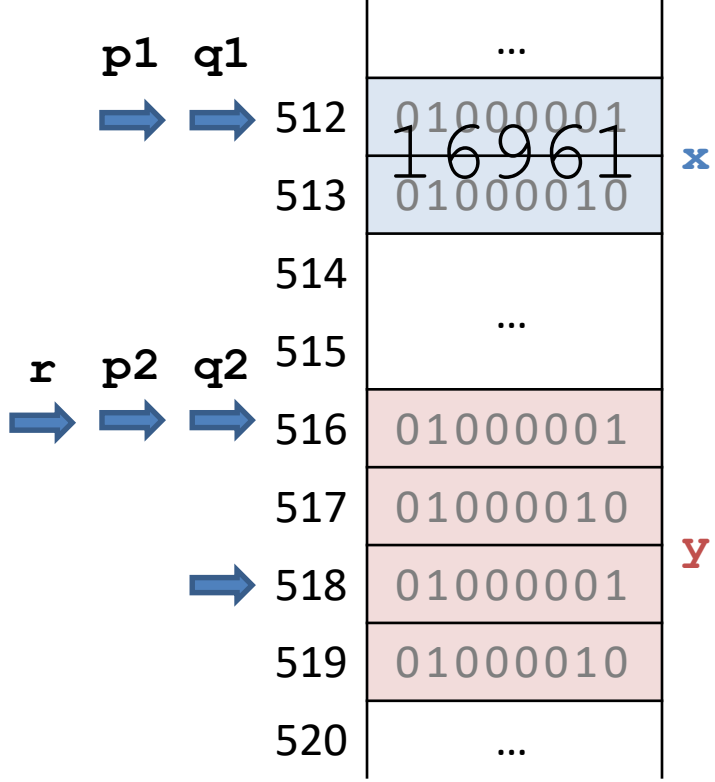
Lépésméret:
`sizeof(short)=2`

Lépésméret:
`sizeof(double)=8`

PÉLDÁK

```
short x=16961;
long y=1111573057;
char *p1, *p2;
short *q1, *q2;
long *r;

p1 = (char*)&x;
p2 = (char*)&y;
q1 = &x;
q2 = (short*)&y;
r = &y;
```




```
AB
16961
16961
16961
A
B
516
520_
```

```
printf("%c%c\n", *p1, *(p1+1));
printf("%h\n", *q2);
printf("%h\n", *(q2+1));
++q2;
printf("%h\n", *q2);
printf("%c\n", *(char*)q2);
printf("%c\n", *((char*)q2+1));
printf("%p\n%p", r, (r+1));
```

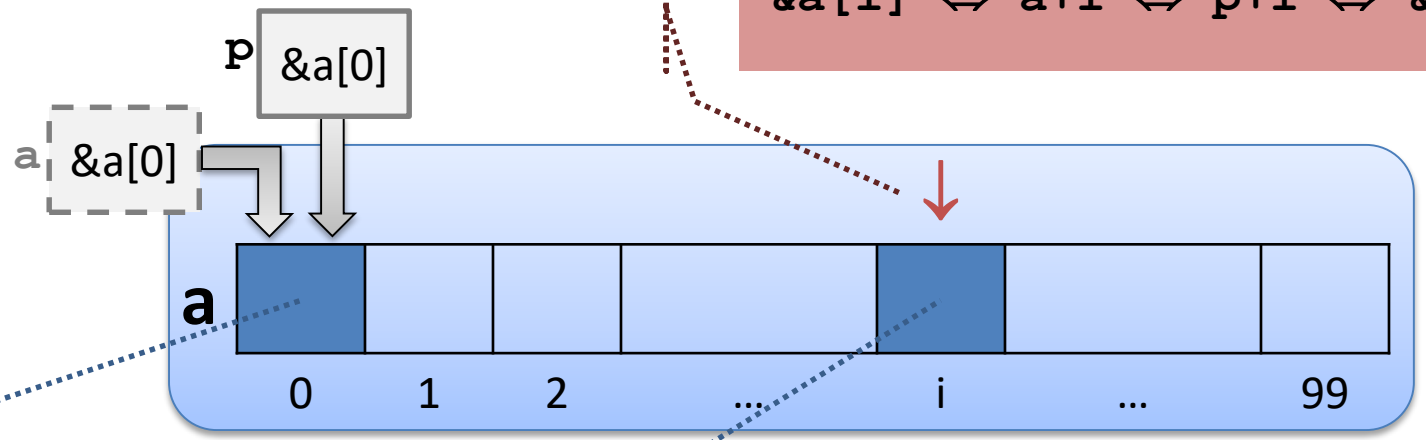
TÖMBök és POINTERek (1)

- Minden 1d-tömb neve ***tekinthető*** pointer-*konstansnak*
  - amely a 0. elem címét tárolja
- Minden pointer-*változó* ***használható*** tömbnévként is, amennyiben tömbre mutat
  - vagyis indexelhető: []

TÖMBök és POINTERek (2)

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

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



Hivatkozás a 0. elemre:
`a[0] ⇔ *a ⇔ *p ⇔ p[0]`

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

TÖMBök és POINTERek (3)

```
int a[100], *p, n, i;
...
scanf("%i", &n);
for( p=a ; (p-a)<n ; ++p ){
     printf("%i ", *p);
}
...
for( p=a,i=0 ; i<n ; ++i ){
     printf("%i ", p[i]); //*(p+i),a[i],*(a+i)
}
...
printf("%i\n", sizeof(a));
printf("%i\n", sizeof(p));

++a; // pointer-konstans*(a+i);
```

```
400
4

_
```

Összefoglalás

- Pointer definiálás
  - **<típus> *<pointer_változó_azonosító>;**
- Hivatkozás egy változó címére:
  - **&<változó_azonosító>**
- Hivatkozás a címzett változóra:
  - ***<pointer_változó_azonosító>**
- Műveletek POINTER-ekkel
  - **<pointer> + <egész_érték>**
  - **Pointer-ek különbsége**
  - **Pointer-ek összehasonlítása**
- TÖMBök és POINTERek
  - 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]**