

# Mutatók és vektorok kapcsolata

Kormányos Andor

Komplex Rendszerek Fizikája Tanszék

2021. szeptember 22.

A C-ben a mutatók és a vektorok között szoros kapcsolat van. Minden művelet, ami egy tömb indexelésével elvégezhető, megoldható mutatókkal is.

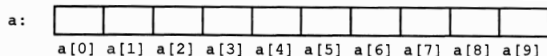
# Mutatók és vektorok

A C-ben a mutatók és a vektorok között szoros kapcsolat van. Minden művelet, ami egy tömb indexelésével elvégezhető, megoldható mutatókkal is.

Definiáljunk egy 10 elemű vektort:

```
double a[10];
```

A vektor elemei egy adott címtől kezdve folytonosan helyezkednek el a memóriában



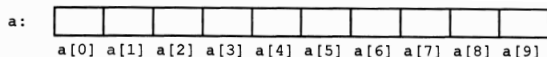
# Mutatók és vektorok

A C-ben a mutatók és a vektorok között szoros kapcsolat van. Minden művelet, ami egy tömb indexelésével elvégezhető, megoldható mutatókkal is.

Definiáljunk egy 10 elemű vektort:

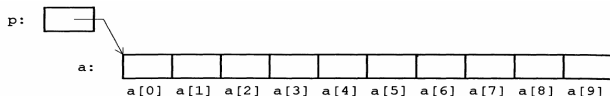
```
double a[10];
```

A vektor elemei egy adott címtől kezdve folytonosan helyezkednek el a memóriában



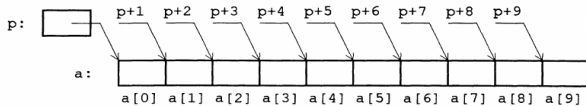
Definiálhatunk egy pointert és ráállíthatjuk az **a** vektor 0-ik elemének címére:

```
1 double a[10];  
2 double *p;  
3  
4 p = &a[0];  
5 // p most az a tömb 0-ik elemere mutat
```



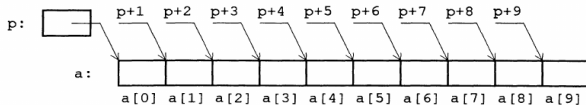
# Mutatók és vektorok

- ha a `p = &a[0]` paranccsal az `a` vektor 0-ik elemének címére állítottuk a `p` mutatót  $\Rightarrow$  `p+i` kifejezéssel az `a` vektor  $i$ -ik elemének címére mutatunk



# Mutatók és vektorok

- ha a `p = &a[0]` paranccsal az `a` vektor 0-ik elemének címére állítottuk a `p` mutatót  $\Rightarrow$  `p+i` kifejezéssel az `a` vektor  $i$ -ik elemének címére mutatunk



- tömb neve és 0-ik indexű elemének a címe igazából szinonímák, ezért

```
1 double a[10];  
2 double *p;  
3  
4 p = &a[0];
```

mellett használhatjuk

```
1 double a[10];  
2 double *p;  
3  
4 p = a; // p most az a tömb 0-ik elemere mutat
```

Miután a `p` pointert ráállítottuk egy `a` vektor első elemének a címére, a következők egyenértékűek:

<code>*(p + i)</code>	$\Leftrightarrow$	<code>p[i]</code>	az <code>i</code> -ik elem értéke
<code>p + i</code>	$\Leftrightarrow$	<code>&amp;p[i]</code>	az <code>i</code> -ik elem címe

**Fontos:** a  $p$  mutató egy változó, értéke változtatható, az  $a$  vektor viszont egy konstans tömb.

```
1 double a[10];
2 double *p;
3
4 p = &a[0];
5
6 p[5]=p[5]+1; // miután p-t raallitottuk a vektor első elemere
7             // a vektor i-ik elemének értékére p[i]-vel is
8             // hivatkozhatunk
9
10 *(p++)=5; // a mutató most a[1] címére mutat és értéke 5
11
12 (*p)++; // a mutató által kijelölt memóriacímen lévő
13         // értéket megnöveltük
14
15
16 a++; // helytelen, hibauzenetet kapunk
17
18 a[5]++; // helyes, az a vektor 5-ik elemét növeltük
```



## A pointer és a mutatott típus mérete

- a pointer memóriacímre mutat, de tisztában van a mutatott típus méretével
- ezt figyelembe veszi, amikor a következő elem memóriacímét kérdezzük le
- a mutatott típus méretétől függ, hogy hány byte-t ugrik

