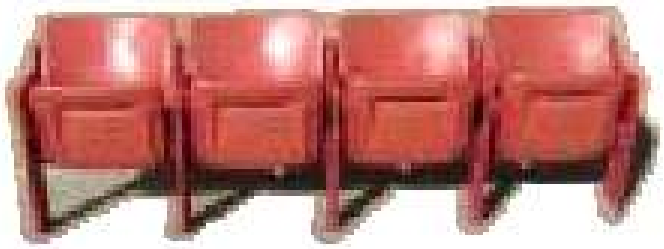
Egyméretű tömbök, vektorok

- *Vektorok definiálása*
- *Vektorok definiálása kezdőértékekkel*
- *Egyméretű tömbök és a mutatók*
- *Pointer-aritmetikai műveletek*
- *Példaprogram vektorhasználatra*
- *Szövegkezelés karaktervektorokkal*
- *Példaprogram karaktervektorokkal*



Vektorok definiálása

Az egydimenziós tömbök, vagy másnéven vektorok a memóriában egymást követően elhelyezkedő azonos típusú elemekből állnak.

A definiálás formája:

<típus> <azonosító> **[** <elemek_száma> **]** ;

Az <elemek_száma> *fordításkor kiértékelt egész konstans kifejezés.*

Egy adott tömbelemre a sorszámával (indexével) lehet hivatkozni, amely a 0 .. (<elemek_száma> -1) *tartományból választható.*

A tartományon kívüli indexmegadások nem okoznak hibajelzést és nehezen kideríthető hibákat eredményeznek. Emiatt célszerű az elemszámot (a vektor méretét) előre definiált konstanssal megadni.

(viszont e miatt a programvégrehajtás gyorsabb, mert nem kell ellenőrizni a túlcímzést)

Vektorok definiálása . .

Pl.:

```
...
#define  ESZ  8
float vektor[ ESZ ] ;
int i ;
for ( i = 0 ; i < ESZ ; i++)  vektor[i ] = 2*i ;
...
printf("%d", vector[i] );
```

Definiálhatunk saját tömbtípust is:

Pl.:

```
typedef unsigned int vektTip [ 7 ] ;
vektTip torpek_kora ;      /*  vektTip típusú vektor definiálása */
```

Vektorok definiálása kezdőértékekkel

A tömböket részben, vagy egészben feltölthetjük kezdőértékekkel definiáláskor. Ha ilyenkor az elemszám-megadás elmarad, a feltöltő elemek száma határozza meg a tömbméretet:

Pl.:

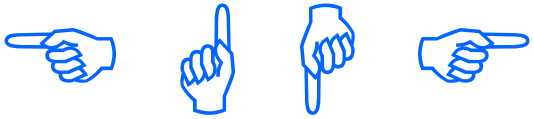
```
main()
{
    float vektor[ ] = {1.2, 2.3, 3.4, 4.5} ;
    float szumma = 0 ;
    int k, elemszam;
    elemszam = sizeof (vektor) / sizeof (float );
    for ( k = 0 ; k < elemszam; k++)
    {
    szumma += vektor[ k ];
    }
    printf("Szumma= %f", szumma);
}
```

Egyméretű tömbök és a mutatók

*A mutatók a C nyelvben az általuk mutatott adat típusának megfelelő típusúak. Ha típus nélküli mutatót akarunk definiálni, a **void** típust kell megadni.*

Pl.:

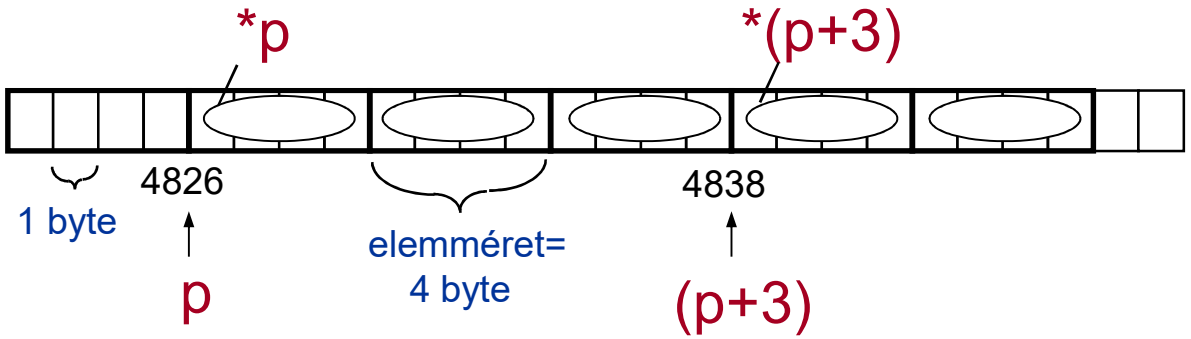
```
int * egeszremutato;
float * lebegopontosra_mutato;
double * mutdbl;
void * mutato;
```



Pointer-aritmetikai műveletek

A (nem void) pointerekre a következő aritmetikai műveletek vannak értelmezve, melyek kifejezetten tömbelem-mutatóknál alkalmazhatóak előnyösen:

p++	*inkrementálás*
p - -	*dekrementálás*
p + k	*egész kifejezés értékének hozzáadása*
p - k	*egész kifejezés értékének levonása*
p1 - p2	*két mutató kivonása.*



Pointer-aritmetikai műveletek . .

p++ hatására a p mutató értéke a mutatott típus méretének megfelelő számú byte-tal megnő, ezáltal a memóriában a következő vektorelemre mutat.

p- - hatására a p mutató a memóriában a megelőző vektorelemre mutat.

A k egész kifejezés hozzáadása után a k elemmel előrébb levő, a k levonása után pedig a k elemmel hátrébb lévő elemre mutat.

Két mutató különbsége a két mutatott elem elemsorszámának különbségét adja, ami a két mutatott memóriacím byte-okban vett különbségének a mutatott elem méretével való osztásával adódik.

Pointer-aritmetikai műveletek . .

A tömbök és a mutatók rokonságának bemutatására tekintsük a következő példát:

```
int vekt[ 6 ] , * p ;       /* p definiálatlan helyre mutat */
int elso, masodik, harmadik;
p = &vekt[ 0 ];             /*   vagy   p = vekt    */
elso = vekt[ 1 ];           /*   vagy   p[ 1 ]      */
masodik = vekt[ 2 ];        /*   vagy   *(vekt+2)   */
harmadik = vekt[ 3 ];       /*   vagy   *(p + 3)    */
```

*A C nyelvben a vekt[k] és a *(vekt + k) kifejezések egyenértékűek.*

A vekt tömbnév és a p mutató közti eltérés, hogy amíg p egy változó, a tömbnév egy konstans mutató, igy nem is adható neki érték.

Példaprogram vektorhasználatra

1. alapalgoritmus: összegzés

```
#include <time.h>
#include <stdio.h>
int vekt [ 20 ];
main()
{
    int i, szum;
    srand(time(NULL));
    for ( i = 0; i < 20; i ++)
    {
        vekt [ i ] = rand() % 31;
    }
   for ( szum = 0, i = 0; i < 20; i ++)
   {
        szum += vekt [ i ];
   }
    printf("Átlag= %f",szum / 20.);
}
```

Szövegkezelés karaktervektorokkal

A C nyelv nem ismer önálló szöveges típust, a szövegek tárolását karaktervektorokkal oldja meg.

A karaktervektorok között nincs közvetlen értékátadás, a sztringek átadását karaktermásoló ciklussal kell megoldani.

~~*szoveg1 = szoveg2;*~~

Egy karakterfüzér tárolására kijelölt memóriaterület hossza eltérhet az aktuálisan benne tárolt szöveg hosszától. A szöveg hosszának tárolása helyett a C nyelv a szöveg végét jelöli a \0 karakterrel (0 ASCII kód). A karaktervektor-változó definiálásakor ezt a karaktert is számba kell venni a méret megadásánál.

Szövegkezelés karaktervektorokkal . .

Példák:

```
char str20[21];
char s_teli[ 8]     = {'M', 'o', 'n', 'i', 't', 'o', 'r', '\0' };
char s_min[ ]       = {'E', 'g', 'é', 'r', '\0'};
char s_nagy[ 22 ] = "Joystick";
char s_smart[ ]     = "Modem";
```

A két utolsó esetben a \0 karaktert a fordító helyezi el a szövegek végén.

Amennyiben mutatóval definiáljuk a karakter vektort, ügyelni kell a mutatónak történő értékadással, mert felülírhatjuk az eredeti stringre mutató értéket.

Példaprogram karaktervektorokkal

A sztringek kezelésére vonatkozó függvények részletes ismertetése előtt tekintsünk egy Julianus dátumot számító példaprogramot:

```
/* Hányadik_nap  */
#include <stdio.h>
#include <conio.h>
#include <string.h>     /* Szövegkezelő függvények  */
#include <stdlib.h>     /* exit() függvény  */
void main()
{
 char datum[9];        /* szövegváltozó deklarálása */

 int ev,ho,nap,hiba,hanyadik;

const unsigned int hoelejek[12]=
    {0,31,59,90,120,151,181,212,243,273,304,334};
```

Példaprogram karaktervektorokkal . .

```
printf("A program megadja, hogy az adott ");
printf("datumu nap az ev hanyadik napja.\n");
printf("A datum (ee.hh.nn): ");
gets(datum);
/* három sztring-re tagol: */
datum[2] = '\0';
datum[5] = '\0';
ev = atoi(datum);    /* ev==0, ha sikertelen */
ho = atoi(datum+3);  /* sztringet konvertál egésszé */
nap = atoi(datum+6);
if ( !(ev && ho && nap) ) /* valamelyik == 0 */
{
    printf("A dátumkonvertálás sikertelen !");
    exit(0);
}
hanyadik = hoelejek[ho - 1] + nap;
if (ev % 4 == 0 && ho > 2 )
{
    hanyadik++;
}
printf("\nA 20%s. ev %d. napja.", datum, hanyadik);
}
```