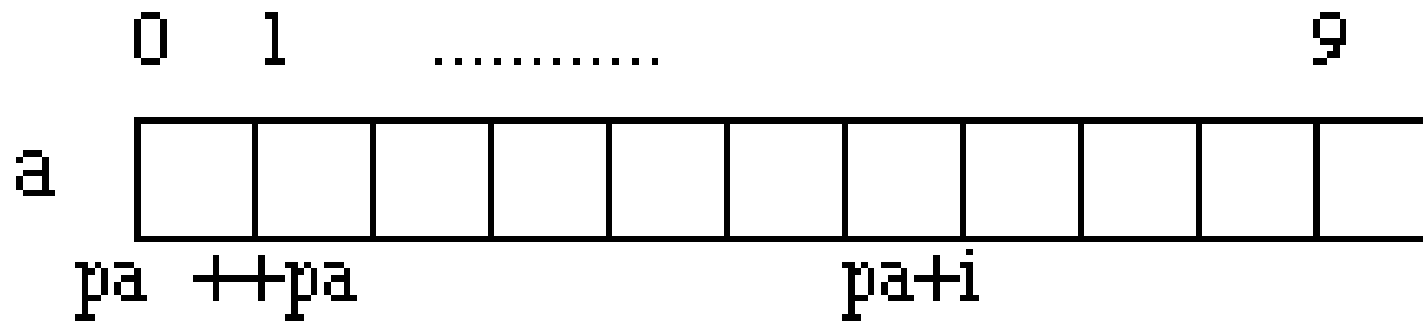


# מערכים ומצביעים

ב-C יש קשר קרוב בין מערכים למצביעים. מערך נותן סדרה רצופה של מקומות בזכרון.

```
int a[10], x;  
int *pa;  
pa = &a[0]; /* pa: pointer to address of a[0] */  
x = *pa; /* x = contents of pa (a[0] in this case) */
```



# מערכים ומצביעים

|                              |   |
|------------------------------|---|
| <code>pa=a;</code>           | אפשר פשוט לכתוב:                        |
| <code>pa=&amp;a[0];</code>   | במקום:                                  |
| <code>*(a+i);</code>         | ואת <code>a[i]</code> אפשר לכתוב בצורה: |
| <code>*(pa+i) ⇔ pa[i]</code> | ז"א, שתי הצורות שקולות:                 |

אבל, מצביע ומערך הם שונים:

מצביע הוא משתנה. אפשר לעשות `pa=a` ו- `pa++`.  
מערך קשור לכתובת קבועה. `a=pa` ו- `a++` הם משפטים  
אסורים.

# מערכים ופונקציות

כששולחים מערך לפונקציה, בעצם הכתובת נשלחת.

```
void allones(int *pa, int n)
{
    int i;
    for ( i=0 ; i<n ; i++)
    {
        *pa = 1;
        ++pa;
    }
}
```

```
int a[20];
allones(a,20);

for(i=0 ; i<n ; i++)
    *(pa+i)=1;

for(i=0 ; i<n ; i++)
    pa[i]=1;
```

## מערכים דו ממדים ומערכים של מצביעים

```
int grades [2] [4] = { { 80, 85, 97, 90},  
                      {75,76,78,90} };
```

|             | עמ' 0 | עמ' 1 | עמ' 2 | עמ' 3 |
|-------------|-------|-------|-------|-------|
| שורה 0 - דן | 80    | 85    | 97    | 90    |
| שורה 1 - רם | 75    | 76    | 78    | 90    |

```
type name_arr[length] [length] [length]...;
```

# פניה לאיבר מסוים

grades[1][3] = 79; --- יש לציין את כל מימדי המערך.

## פעולות

```
int i,j;  
for (i=0; i<2; i++)  
    for (j=0; j<4; j++)  
        grades[i][j] *= 1.05;
```

```
int *ip;  
ip= &grades[0][0];  
*(ip+5) = 2;
```

# מערכים של מצביעים

```
int no[20], x, y;
```

```
int *iptr[10];
```

```
iptr[0] = &x;
```

```
iptr[1]= &no[3];
```

```
iptr[2] =no+5;
```

```
iptr[3] = &y;
```

```
iptr[4]=iptr[3];
```

```
iptr[5]=&x;
```

```
*iptr[5]=17;
```

```
int *i1,i2,*i3,i4,i5[10];
```

```
int (*jptr)[10], b[10];
```

```
jptr =&b;
```

```
(*jptr)[5]=1;
```

# איתחול של מבנה

```
struct human {  
    char f_initial,l_initial;  
    int age;  
    float height; };
```

שיטה ראשונה - איתחול מלא:

```
struct human TR = {'T','R', 18, 1.82};
```

## איתחול של מבנה (2)

שיטה שנייה – כל רכיב בנפרד:

```
int i;
```

```
struct human TR;
```

```
TR.f_initial = 'T';
```

```
TR.i_initial = 'R';
```

```
TR.age = 18;
```

```
TR.height = 1.82;
```

```
i=TR.age;
```



# מצביעים למבנה

```
typedef struct human Human;
```

```
HUMAN TR;
```

```
HUMAN *h_ptr = &TR;
```

```
h_ptr->f_initial = 'T';
```

```
h_ptr->i_initial = 'R';
```

```
h_ptr->age = 18;
```

```
h_ptr->height = 1.82;
```

# העברת מבנים לפונקציות

```
HUMAN makehuman(char fi, char li, int age, float h)
{
    HUMAN temp;
    temp.f_initial=fi;
    temp.l_initial=li;
    temp.age=age;
    temp.height=h;

    return temp;
}
```

TR=makehuman('T','R',18,1.82);

# העברת מבנים לפונקציות

```
struct comp { float real;  
              float imag; };
```

```
struct comp add(struct comp a , struct comp b)  
{  
    struct comp c;  
    c.real=a.real+b.real;  
    c.imag=a.imag+b.imag;  
    return c;  
}
```

```
struct comp a={1,0},b={0,1},c=add(a,b);
```