

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);
```

העברת מבנים לפונקציות

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

```
void add(struct comp *a , struct comp *b ,  
        struct comp *c)  
{  
    c->real=a->real+b->real;  
    c->imag=a->imag+b->imag;  
}
```

```
struct comp a={1,0},b={0,1},c[10];
```

```
add(&a,&b,c);  
printf("%g\n",c[0].real+c[0].imag);
```

מבנים בתוך מבנים

```
struct comp3D { struct comp x, y, z; };
```

```
struct comp3D point;
```

```
point.x.real=1.;
```

```
point.x.imag=0.;
```

```
point.y.real=1.;
```

```
point.y.imag=1.;
```

```
add(&point.x,&point.y,&point.z);
```

```
printf("%g %g\n",point.z.real,point.z.imag);
```

ו. -> מופעלים משמאל לימין:

```
struct comp3D point,*pp;
```

```
pp=&point;
```

```
pp->x.real=1.;
```

```
point.x.imag=0.;
```

```
(point.y).real=1.;
```

```
(pp->y).imag=1.;
```

```
struct comp3D point,*pp=&point;
```

מערכים של מצביעים

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

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

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

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

```
jptr = &b;
```

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

```
jptr=&no;    -> warning: assignment from  
            incompatible pointer type  
**jptr=4;
```

```
printf("%i\n",no[0]);
```

```
** (jptr+1)=7;
```

```
printf("%i\n",no[10]);
```

מערכים של מצביעים

```
int no[20], (*jptr)[10];
```

```
jptr=&no;    -> warning: assignment from  
            incompatible pointer type  
**jptr=4;
```

```
printf("%i\n",no[0]);
```

```
** (jptr+1)=7;
```

```
printf("%i\n",no[10]);
```

```
*( *jptr+1)=8;
```

```
printf("%i\n",no[1]);
```

מצביע כללי- כתובת
ללא סוג/גודל משתנה:

```
void *vptr;
```

```
vptr=&no;
```

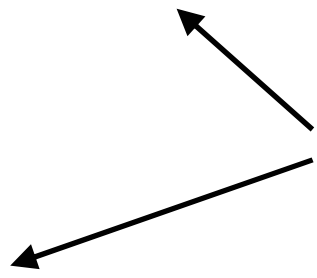
```
jptr=vptr;
```

מצביעים לפונקציות

```
double trapez(double (*func) (double), double a, double b)
{
    double trapez(double (func) (double), double a, double b)
    double h=b-a;
    return h*((*func)(a)+(*func)(b))/2.;
    return h*(func(a)+func(b))/2.;
}

printf("%g\n",trapez(sqrt,1.,2.));
```

עובד גם:



גודל של משתנה

sizeof object

גודל = מספר הבייטים

sizeof(type name)

(unsigned int)

```
printf("%i %i %i %i %i %i\n", sizeof(char), sizeof(int), sizeof(long),  
    sizeof(float), sizeof(double), sizeof(long double), sizeof(char*),  
    sizeof(double*));
```

1 4 4 4 8 12 4 4

```
struct comp3D { struct comp x, y, z; };
```

```
struct comp3D point;
```

```
float grades[2][4];
```

```
printf("%i %i %i\n", sizeof grades, sizeof point, sizeof(struct comp));
```

32 24 8

הקצאת זיכרון באופן דינאמי (Dynamic memory allocation)

הקומפיילר עצמו מקצה זיכרון באופן סטטי.

```
#include <stdlib.h>
```

```
void *malloc(size_t number_of_bytes);
```



stdlib.h: unsigned int

```
int i, j, *ip;
```

```
scanf("%i",&j);
```

```
ip = (int *) malloc(j*sizeof(int));
```

```
for (i=0; i<j; i++)
```

```
    ip[i]=10;
```

:5 הרצאה

cast: הטלה

(type-name) expression

```
int round(double x) {  
    return (int) ((x < 0) ? (x-.5) : (x+.5)); }  
}
```


הקצאת זיכרון באופן דינאמי (Dynamic memory allocation)

```
#include <stdlib.h>
```

```
void *malloc(size_t number_of_bytes);
```

```
void free(void* p);
```

```
ip = (int *) malloc(j*sizeof(int));
```

```
for (i=0; i<j; i++) ip[i]=10;
```

```
free(ip); ← לשימוש רק אחרי malloc
```