

המעבד הראשוני

```
#define name replacement-text
```

```
#define Area(r) Pi*r*r          שגיאה!!!
```

```
x=Area(s+1);
```

```
x=Area(s++);
```

```
double sqrarg;
```

```
#define SQR(a) (sqrarg=(a),sqrarg*sqrarg)
```

```
SQR(1.)*SQR(2.) => 1.
```

Numerical
Recipes

```
#define SQR(a) \
```

```
((sqrarg=(a)) == 0.0 ? 0.0 : sqrarg*sqrarg)
```

Formatted Input/Output

קלט/פלט ערוך

```
int printf(char* format, ...);
```

```
a=Pi;
```

```
printf("Result = %-12.5e\n",a);
```

מתחיל משמאל

אורך מינימלי (כולל רווחים)

conversion character

אות המרה

מספר ההמרות =
מספר המשתנים

דיוק

Formatted Input/Output

קלט/פלט ערוך

אותיות מיוחדות

<code>\n</code>	newline
<code>\t</code>	tab
<code>\\</code>	<code>\</code>
<code>\'</code>	<code>'</code>
<code>\"</code>	<code>"</code>
<code>%%</code>	<code>%</code>

אותיות המרה

<code>d,i</code>	int
<code>f</code>	<code>[-]m.ddd</code>
<code>e,E</code>	<code>[-]m.ddd e/E ±xx</code>
<code>g,G</code>	<code>%f, or %e/E</code>

Formatted Input/Output

קלט/פלט ערוך

```
double a=Pi;  
printf("Result = %12.5e!\n",a);  
printf("Result = %12.5E!\n",a);  
printf("Result = %10.10f!\n",a);
```

Result = 3.14159e+00 !

Result = 3.14159E+00!

Result = 3.1415926536!

Formatted Input/Output

קלט/פלט ערוך

```
int scanf(char* format, ...);
```

```
double a,b;
```

```
printf("%lg %lg",&a,&b);
```

אין משמעות למרווחים

conversion character אות המרה

= מספר ההמרות
מספר המשתנים

קלט/פלט ערוך

מדלג על רווחים/שורות חדשות בקלט

```
int i,j;  
scanf("%i %i",&i,&j);  
printf("%i %i\n",i,j);
```

45 76

45

4.5

45 76

76

45 76

4 134514033

אותיות המרה

d,i

int

e,f,g

float

ld,li

long

le,lf,lg

double

מציביעים: Pointers

```
int i;  
&i;  
int *point;  
point=&i;
```

הגדרנו משתנה:

את הכתובת בזיכרון מסמנים:

ניתן לשמור את הכתובת בתוך משתנה:

```
int *
```

משתנה שהוא מצביע ל-`int` מגדירים בעזרת:

הפעולה `&` מחזירה את הכתובת של הארגומנט.

הפעולה `&i` נקראת הפנייה (referencing) של `i`.

ההיפך (הצבעה, de-referencing) הוא `*`, שנותן את

```
i=*point;
```

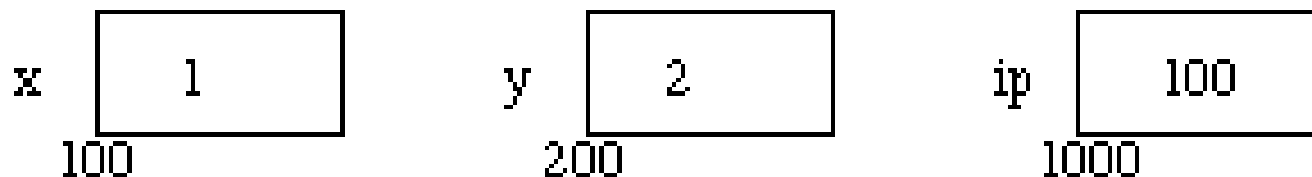
הארגומנט שאליו מצביעים:

אזהרה: `*` משמש למכפלה, להגדרת מצביעים, וגם לפעולת הצבעה.

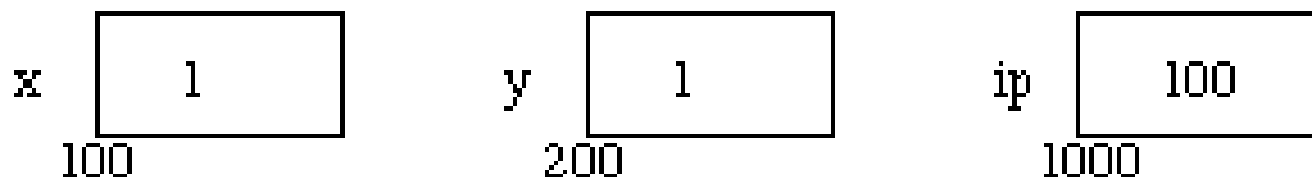
```
int x = 1, y = 2;  
int *ip;
```

מצביעים ברמת הזכרון (הנסתרת)

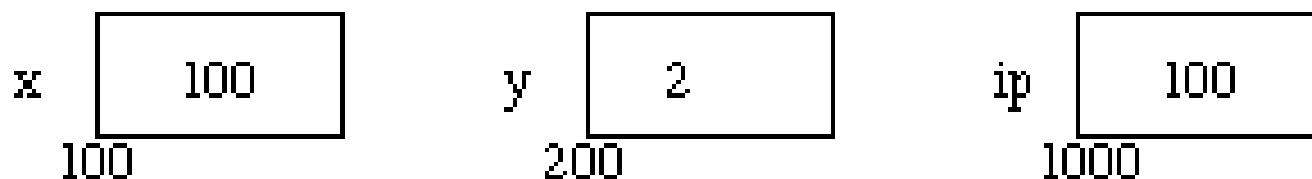
```
ip = &x;
```



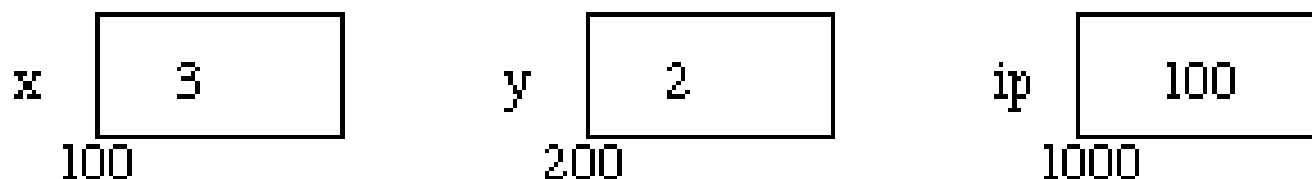
```
y = *ip;
```



```
x = ip;
```



```
*ip = 3
```



דוגמא עם פונקציות

```
fiddle(int x, int *y)
{
    printf("Starting fiddle: x = %d, y = %d\n", x, *y);
    x++;
    (*y)++;
    printf("Finishing fiddle: x = %d, y = %d\n", x, *y);
}
```

משנים לא את הערך של y עצמו, אלא את הערך
השמור במקום בזכרון ש- y מצביע עליו.

```
main()
{
    int i = 0, j = 0;
    printf("Starting main : i = %d, j = %d\n", i, j);
    printf("Calling fiddle now\n");
    fiddle(i, &j);
    printf("Returned from fiddle\n");
    printf("Finishing main : i = %d, j = %d\n", i, j);
}
```

יוצרים מצביע ל-int בעזרת פעולת &.

תוצאת הרצת התכנית:

Starting main : $i = 0, j = 0$

Calling fiddle now

Starting fiddle: $x = 0, y = 0$

Finishing fiddle: $x = 1, y = 1$

Returned from fiddle

Finishing main : $i = 0, j = 1$

הערך של i לא השתנה, אבל הערך של j שנשלח בעזרת מצביע, כן השתנה.

דוגמא: פונקציה להחלפת שני משתנים

- `swap(a, b)` לא יעבוד:
- `swap(&a, &b)` :

```
void swap(int *px, int *py)
{
    int temp;
    temp = *px;
    *px = *py;
    *py = temp;
}
```

חשוב לזכור:

כשמגדירים משתנה בתור מצביע, הוא עדיין לא מצביע לשום מקום. לכן:

```
int *ip;  
*ip = 100;
```

היא שגיאה שמונעת הרצה (גורמת ל-crash).
דוגמא לשימוש נכון:

```
int *ip, x;  
ip = &x;  
*ip = 100;
```

אריתמטיקה עם מצביעים

```
float x=0, y=0, *flp, *flq;  
flp=&x;  
flq=&y;  
*flp = *flp + 10;  
++*flp;  
(*flp)++;  
flq = flp;  
++flq;
```



פעולות על משתנה אחד (כמו *
או ++), בניגוד לשניים (כמו ++),
מבוצעות מימין לשמאל.

אריתמטיקה עם מצביעים

- לכל מצביע מציינים את סוג המשתנה שאליו הוא מצביע, בשביל שהמחשב ידע את כמות הזיכרון שבה מדובר.
- כשמגדילים את הערך של מצביע, הוא גדל ביחידה אחת בעלת גודל מתאים.
- ++ip מוסיף 2 בייטים לכתובת, אם מדובר במצביע ל-int.
- ++fp מוסיף 4 בייטים לכתובת, אם מדובר במצביע ל-float.

סיכום (ביניים)

- כל משתנה מקבל מקום בגודל מוגדר בזיכרון.
- משתנה מצביע מאחסן כתובת של משתנה. גם למצביע יש כתובת בזיכרון.
- הגדרת מצביע נעשית ע"י * לפני השם.
- אתחול מצביע בכתובת של משתנה מתבצע בעזרת פעולת & .
- ל- * תפקיד נוסף (אם אינו בשורת הגדרה) לציין בקשה להשתמש ב (או לשנות את) תוכן התא שכתובתו מאוחסנת במצביע.