

# מבנים



## קרן כליף

# ביחידה זו נלמד:

---

- מהו מבנה (struct)
- typedef
- איתחול מבנה
- השמת מבנים
- השוואת מבנים
- העברת מבנה לפונקציה
- מבנה בתוך מבנה
- מערך של מבנים
- גודל מבנה בזיכרון

# מבנה (struct) - מוטיבציה

□ אם נרצה לשמור בתוכנית שלנו 3 תאריכים, נצטרך להגדיר 9 משתנים:

```
int day1, month1, year1, day2, month2...
```

□ אם היינו רוצים מערך שיכיל אוסף של תאריכים, כנראה שהיינו מגדירים 3 מערכים, כאשר הראשון היה מחזיק את הימים, השני את החודשים והשלישי את השנים

□ כדי להעביר נתוני מבנה לפונקציה, צריך להעביר 3 פרמטרים..

□ היינו רוצים להמציא טיפוס חדש שיכיל את כל הנתונים של תאריך. לטיפוס זה יהיו 3 שדות: יום, חודש ושנה

□ טיפוס חדש שאנחנו יוצרים נקרא "מבנה" (struct)

# מבנה - דוגמא

```
#include <stdio.h>
```

שם הטיפוס החדש

```
struct Date  
{  
    int day;  
    int month;  
    int year;  
};
```

הגדרת מבנה המכיל אוסף של שדות

Date: d.day	23	1000
d.month	8	
d.year	2008	

```
void main()  
{
```

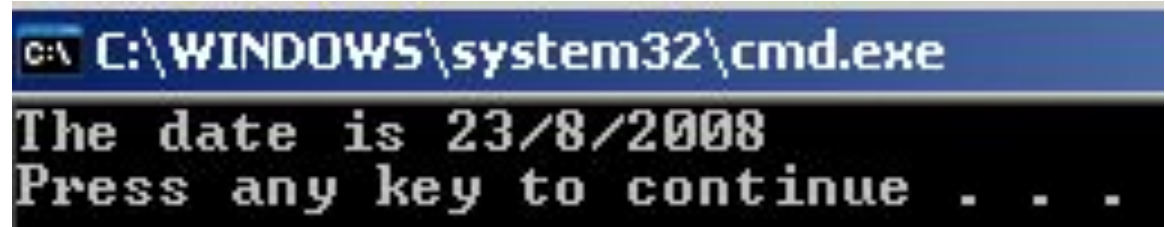
```
    struct Date d;  
    d.day = 23;  
    d.month = 8;  
    d.year = 2008;
```

הגדרת משתנה מטיפוס המבנה

פניה לשדות המשתנה תהיה באמצעות נקודה

```
    printf("The date is %d/%d/%d\n", d.day, d.month, d.year);
```

```
}
```



```
C:\WINDOWS\system32\cmd.exe  
The date is 23/8/2008  
Press any key to continue . . .
```

## מבנה - תחביר

□ כדי לייצר מבנה חדש נגדיר אותו בתחילת הקובץ (לפני ההצהרות על הפונקציות)

```
struct <שם המבנה>
{
    <type1> <name1>;
    <type2> <name2>;
};
```

□ פנייה לשדה של משתנה של מבנה היא ע"י

<שם המשתנה>.<שם השדה>

# typedef

עד כה ראינו כל מיני טיפוסים: int, double, char, float ועוד

- טיפוסים אלו נקראים "טיפוסים פרימיטיביים"

ניתן לתת שם נוסף לטיפוס ע"י הפקודה typedef

- תחביר הפקודה:

```
typedef <שם קיים> <שם חדש>
```

- דוגמא:

```
typedef unsigned long ulong
```

- כעת נוכל להגדיר משתנה מטיפוס זה באחת משתי הדרכים:

```
unsigned long id;  ulong id;
```

# מבנה – שימוש ב- typedef

---

```
#include <stdio.h>
```

```
struct Date  
{  
    int day;  
    int month;  
    int year;  
};
```

```
typedef struct Date date_t;
```

```
void main()  
{  
    date_t d;  
    d.day = 23;  
    d.month = 8;  
    d.year = 2008;  
  
    printf("The date is %d/%d/%d\n", d.day, d.month, d.year);  
}
```

# שימוש ב- typedef מקוצר למבנים

```
#include <stdio.h>
```

```
struct Date  
{  
    int day;  
    int month;  
    int year;  
};
```

```
typedef struct Date date_t;
```

```
void main()  
{
```

```
    date_t d;  
    d.day = 23;  
    d.month = 8;  
    d.year = 2008;
```

```
    printf("The date is %d/%d/%d\n", d.day, d.month, d.year);
```

```
}
```

```
typedef struct  
{  
    int day;  
    int month;  
    int year;  
} date_t;
```

```
struct Date  
{  
    int day;  
    int month;  
    int year;  
} typedef date_t;
```



# איתחול מבנה

```
struct Date
{
    int day, month, year;
} typedef date_t;
```

ניתן לאתחל מבנה בשורת ההגדרה  
כמו שמאתחלים מערך:

```
void main()
{
    date_t d= {23, 8, 2003};
    printf("The date is %d/%d/%d\n", d.day, d.month,d.year);
}
```

הערך הראשון יושם בשדה הראשון, הערך השני בשדה השני וכו'

יש לוודא התאמה בין טיפוס השדה לערך

השמת ערכים שלא בשורת ההגדרה ניתן לבצע רק שדה-שדה ע"י השמה

# איתחול מבנה - דוגמא

```
#include <stdio.h>
```

```
struct Student  
{
```

```
    char name[20];
```

```
    float age;
```

```
    long id;
```

```
} typedef student_t;
```

```
void main()  
{
```

```
    student_t stud = {"momo", 23.5, 111111};
```

```
    printf("The student's details: name=%s, age=%.2f, id=%ld\n",  
           stud.name, stud.age, stud.id);
```

```
}
```

אין מניעה שאחד משדות המבנה יהיה מערך

```
C:\WINDOWS\system32\cmd.exe  
The student's details:  
name=momo, age=23.50, id=111111  
Press any key to continue . . .
```

# איתחול מבנה המכיל מערך - דוגמא

```
#include <stdio.h>
```

```
#define SIZE 3
```

```
struct course
{
    int grades[SIZE];
    int code;
} typedef course_t;
```

```
void main()
```

```
{
    course_t c = { {90,80,85}, 10801};
    int i;
```

```
    printf("%d is the course code and the grades are: ", c.code);
    for (i=0 ; i < SIZE ; i++)
        printf("%d ", c.grades[i]);
    printf("\n");
}
```

course_t: c.grades	<b>90</b>	<b>1000</b>
	<b>80</b>	
	<b>85</b>	
c.code	<b>10801</b>	
int: i	<b>???</b>	<b>1016</b>

```
10801 is the course code and the grades are: 90 80 85
Press any key to continue . . .
```

# השמת מבנים

---

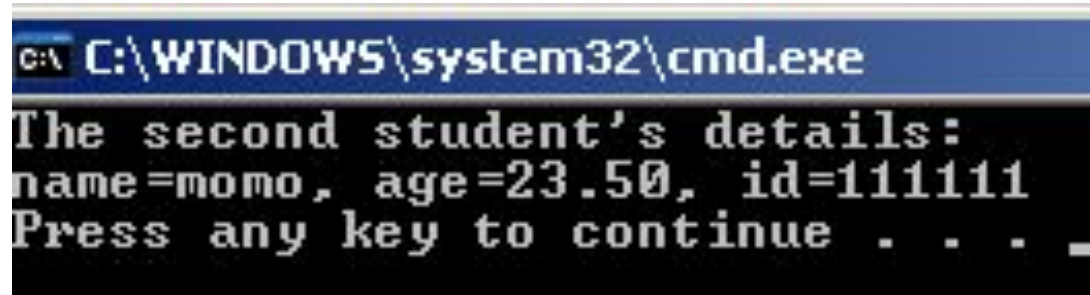
- השמה בין מבנים מעתיקה שדה-שדה
- אם אחד השדות הוא מערך, מועתק באופן אוטומטי איבר-איבר
- בניגוד להשמה בין מערכים שאינם בתוך מבנה (צריך בלולאה להעתיק איבר-איבר)

# השמת מבנים – דוגמאת student

```
#include <stdio.h>
```

```
struct Student  
{  
    char name[6];  
    float age;  
    long id;  
} typedef student_t;
```

```
void main()  
{  
    student_t s1 = {"momo", 23.5, 111111};  
    student_t s2;  
  
    s2 = s1;  
  
    printf("The second student's details:\nname=%s, age=%.2f, id=%ld\n",  
        s2.name, s2.age, s2.id);  
}
```



```
C:\WINDOWS\system32\cmd.exe  
The second student's details:  
name=momo, age=23.50, id=111111  
Press any key to continue . . .
```

student_t: s1.name[6]	<b>"momo"</b>	<b>100 0</b>
s1.age	<b>23.5</b>	
s1.id	<b>111111</b>	
student_t: s2.name[6]	<b>"momo"</b>	<b>101 6</b>
s2.age	<b>23.5</b>	
s2.id	<b>111111</b>	

# נשים לב..

□ כאשר פונים לאחד משדות המבנה מתייחסים אליו כמו למשתנה מטיפוס השדה

■ למשל, אם היינו רוצים לבצע השמה לאחד משדות המבנה שהוא מחרוזת, היינו צריכים להשתמש ב- `strcpy`

```
void main()
{
    student_t s1;

    strcpy(s1.name, "momo");
    s1.age = 23.5;
    s1.id = 111111;
```

```
struct Student
{
    char name[6];
    float age;
    long id;
} typedef student_t;
```

```
printf("The second student's details:\nname=%s, age=%.2f, id=%ld\n",
s1.name, s1.age, s1.id);
}
```

# העברת מבנה לפונקציה

---

□ כאשר מעבירים מבנה לפונקציה, מועבר העתק שלו, בדיוק כמו העברת פרמטר ממשתנה פרימיטיבי

- המשתנה מועבר by value
- אם נשנה את אחד משדות המבנה בפונקציה, השינוי לא ישפיע על המשתנה המקורי
- (כדי שהפונקציה תשנה את המשתנה המקורי, יש להעבירה by pointer, כלומר להעביר את כתובת המבנה)

# העברת מבנה לפונקציה - דוגמא

## פונקציה המדפיסה תאריך

העברת העתק של המבנה לפונקציה

```
struct Date
```

```
{  
    int day, month, year;  
} typedef date_t;
```

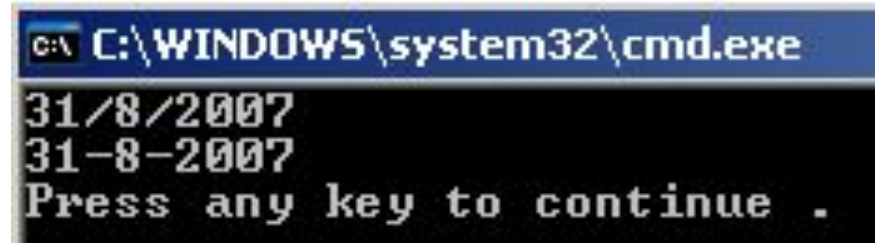
```
void printDate(date_t d, char delimiter)
```

```
{  
    printf("%d%c%d%c%d", d.day, delimiter, d.month, delimiter, d.year);  
}
```

```
void main()
```

```
{  
    date_t d1 = {31, 8, 2007};
```

```
    printDate(d1, '/');  
    printf("\n");  
    printDate(d1, '-');  
    printf("\n");  
}
```



```
C:\WINDOWS\system32\cmd.exe  
31/8/2007  
31-8-2007  
Press any key to continue .
```

date_t: d1.day	31	1000
d1.month	8	
d1.year	2007	

הזיכרון של ה-main

date_t: d.day	31	2000
d.month	8	
d.year	2007	
char: delimiter	'-'	2012

הזיכרון של printDate



# פונקציה המחזירה את הפרש הדקות בין 2 זמנים

(הנחה: הראשון גדול מהשני)

```
#include <stdio.h>
```

```
struct Clock
```

```
{
    int hour, minute;
} typedef clock_t;
```

```
int diffMinutes(clock_t c1, clock_t c2)
```

```
{
    int minutes1 = c1.hour*60 + c1.minute;
    int minutes2 = c2.hour*60 + c2.minute;
    return minutes1 - minutes2;
}
```

```
void main()
```

```
{
    clock_t c1 = {12, 30};
    clock_t c2 = {11, 15};
    int diff = diffMinutes(c1, c2);

    printf("The diff is %d minutes\n", diff);
}
```

```
C:\WINDOWS\system32\cmd.exe
The diff is 75 minutes
Press any key to continue . . .
```

clock_t: c1.hour	<b>12</b>	<b>2000</b>
c1.minute	<b>30</b>	
clock_t: c2.hour	<b>11</b>	<b>2008</b>
c2.minute	<b>15</b>	
int: minutes1	<b>750</b>	<b>2016</b>
int: minutes2	<b>675</b>	<b>2020</b>

הזיכרון של diffMinutes

clock_t: c1.hour	<b>12</b>	<b>1000</b>
c1.minute	<b>30</b>	
clock_t: c2.hour	<b>11</b>	<b>1008</b>
c2.minute	<b>15</b>	
int: diff	<b>75</b>	<b>1016</b>

הזיכרון של ה-main

# דוגמא

```
struct Clock
{
    int hour, minute;
} typedef clock_t;
```

clock_t: c.hour	<b>12</b>	<b>2000</b>
c.minute	<b>50</b>	
clock_t: nc.hour	<b>13</b>	<b>2008</b>
nc.minute	<b>20</b>	
int: minutes	<b>30</b>	<b>2016</b>

הזיכרון של addMinutes

```
clock_t addMinutes(clock_t c, int minutes)
{
    clock_t newClock;
    newClock.minute = c.minute + minutes;
    newClock.hour = c.hour + newClock.minute/60;
    newClock.minute %= 60;
    newClock.hour %= 24;

    return newClock;
}
```

```
void printClock(clock_t c)
{
    if (c.hour < 10)
        printf("0");
    printf("%d:", c.hour);
    if (c.minute < 10)
        printf("0");
    printf("%d", c.minute);
}
```

	<b>1</b>	
clock_t: c.hour	<b>2</b>	<b>300</b>
הזיכרון של printClock		
c.minute	<b>0</b>	

clock_t: c1.hour	<b>12</b>	<b>1000</b>
c1.minute	<b>50</b>	
clock_t: c2.hour	<b>13</b>	<b>1008</b>
c2.minute	<b>20</b>	
int: toAdd	<b>30</b>	<b>1016</b>

הזיכרון של ה-main

```
void main()
{
    clock_t c1, c2;
    int toAdd;

    printf("Enter time and minutes to add:");
    scanf("%d %d %d", &c1.hour,
                &c1.minute, &toAdd);
    printf("The time is: ");
    printClock(c1);
```

```
    c2 = addMinutes(c1, toAdd);
    printf("\nThe new time is: ");
    printClock(c2);
    printf("\n");
}
```

```
Enter time and minutes to add:
12 50 30
The time is: 12:50
The new time is: 13:20
```

# דוגמאות פלט נוספות

אתם מוזמנים להריץ על "יבש" להבנת האלגוריתם

```
C:\WINDOWS\system32\cmd.exe
Enter time and minutes to add: 12 5 20
The time is: 12:05
The new time is: 12:25
Press any key to continue . . .
```

```
C:\WINDOWS\system32\cmd.exe
Enter time and minutes to add: 13 55 20
The time is: 13:55
The new time is: 14:15
Press any key to continue . . .
```

```
C:\WINDOWS\system32\cmd.exe
Enter time and minutes to add: 23 50 30
The time is: 23:50
The new time is: 00:20
Press any key to continue . . .
```

# פניה לשדה של מצביע למבנה

```
struct Date
{
    int day, month, year;
} typedef date_t;
```

```
void main()
{
    date_t d = {29, 3, 2008};
    date_t* pD = &d;

    printf("The year is %d\n", d.year);
    printf("The year is %d\n", pD->year);
    printf("The year is %d\n", (*pD).year);
}
```

date_t: d.day	29	1000
d.month	3	
d.year	2008	
date_t*: pD	1000	1012

כאשר יש מצביע למבנה נפנה  
לשדותיו באמצעות חץ: ->

1000  
המשתנה  
פניה לשדה d.year

# העברת מבנה לפונקציה כמצביע

---

- כאשר מעבירים מבנה לפונקציה, מועבר העתק שלו, בדיוק כמו העברת פרמטר ממשתנה פרימיטיבי
- כדי שהפונקציה תשנה את המשתנה המקורי, יש להעבירה by pointer, כלומר להעביר את כתובת המבנה
- כאשר מעבירים לפונקציה מבנה by value המכיל מערך, עותק של המערך מועבר, ולא רק כתובת ההתחלה שלו, ולכן שינוי הערך בפונקציה לא ישפיע על המערך המקורי

# פונקציה המקבלת זמן ומגדילה אותו

```
struct Clock
{
    int hour, minute;
} typedef clock_t;
```

העברת כתובת של  
המבנה לפונקציה,  
כדי שהפונקציה תוכל  
לשנות את המבנה

```
void addMinutes(clock_t* c, int minutes)
```

```
{
    c->minute += minutes;
    c->hour += c->minute/60;
    c->minute %= 60;
    c->hour %= 24;
}
```

נשים לב לשימוש בחץ  
מאחר ו- c הוא מצביע!

clock_t: c1.hour	<b>1</b>	<b>1000</b>
c1.minute	<b>5</b>	
int: toAdd	<b>75</b>	<b>1008</b>

הזיכרון של ה-main

```
void main()
{
```

```
    clock_t c1;
    int toAdd;
```

```
    printf("Enter time and minutes to add: ");
    scanf("%d %d %d", &c1.hour,
        &c1.minute, &toAdd);
```

```
    printf("The time is: ");
    printClock(c1);
```

```
    addMinutes(&c1, toAdd);
    printf("\nThe new time is: ");
    printClock(c1);
    printf("\n");
```

clock_t*: c	<b>1000</b>	300 0	
int: minutes	<b>75</b>		300 0
clock_t: c1.hour		<b>23</b>	
clock_t: c1.minute		<b>50</b>	

הזיכרון של  
addMinutes  
הזיכרון של  
printClock

```
void printClock(clock_t c)
```

```
{
    if (c.hour < 10)
        printf("0");
    printf("%d:", c.hour);
```

C:\WINDOWS\system32\cmd.exe

```
Enter time and minutes to add: 23 50 75
The time is: 23:50
The new time is: 01:05
Press any key to continue . . .
```

# אופרטורי יחס בין מבנים

□ בניגוד למשתנה פרימיטיבי, לא ניתן להשוות בין מבנים  
ע"י אופרטורי השוואה:  $=$ ,  $\neq$ ,  $<$ ,  $>$ ,  $\leq$ ,  $\geq$

□ סיבות:

1. יתכן ואחד משדות המבנה אינו ניתן להשוואה ע"י אופרטורים  
אלו, למשל מערך או מחרוזת

2. אם ישנם כמה שדות במבנה, לא ברור לפי מה נחליט לאיזה  
שדה חשיבות גדולה יותר (למשל השוואה בין סטודנטים:  
לפי ת.ז. או לפי שם?), וכמובן הקומפילר אינו יכול לנחש  
זאת

□ כדי להשוות בין מבנים עלינו לכתוב פונקציות מתאימות

□ גם לביצוע פעולות חשבון בין שני מבנים עלינו לכתוב  
פונקציות (האם מוגדרת פעולת החיבור בין 2  
סטודנטים? ועבור שני מערכים?)

# השוואה בין מבנים

□ מאחר ולא ניתן להשוות בין מבנים ע"י האופרטור == עלינו לכתוב פונקציה המשווה בין מבנים

```
struct Clock
{
    int hour, minute;
} typedef clock_t;
```

```
int equalClock(clock_t c1, clock_t c2)
{
    return (c1.hour == c2.hour &&
        c1.minute == c2.minute);
}
```

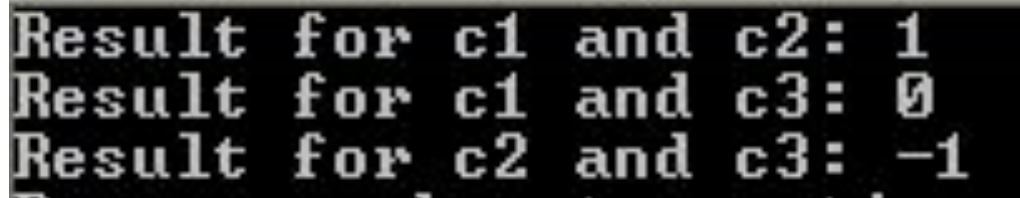


# יחס בין מבנים - דוגמא

```
struct Clock
{
    int hour, minute;
} typedef clock_t;
```

```
int compareClocks(clock_t c1, clock_t c2)
{
    if (c1.hour < c2.hour)        return -1;
    else if (c1.hour > c2.hour)    return 1;
    else // if (c1.hour == c2.hour)
    {
        if (c1.minute == c2.minute) return 0;
        else if (c1.minute < c2.minute) return -1;
        else return 1;
    }
}
```

```
void main()
{
    clock_t c1={12,30}, c2={11,30}, c3={12,30};
    printf("Result for c1 and c2: %d\n", compareClocks(c1, c2));
    printf("Result for c1 and c3: %d\n", compareClocks(c1, c3));
    printf("Result for c2 and c3: %d\n", compareClocks(c2, c3));
}
```



```
Result for c1 and c2: 1
Result for c1 and c3: 0
Result for c2 and c3: -1
```

# מבנה בתוך מבנה

□ אחד משדות המבנה יכול להיות מבנה אחר  
■ דוגמא:

```
struct Date
{
    int day, month, year;
} typedef date_t;
```

```
struct Student
{
    char    name[20];
    date_t birthday;
    float   average;
} typedef student_t;
```

□ היה אפשר גם כך, אבל עדיף  
שלא..

```
struct Student
{
    char name[20];
    int day, month, year;
    float average;
} typedef student_t;
```

# מבנה בתוך מבנה - דוגמא

```
#include <stdio.h>
```

```
struct Date  
{
```

```
    int day, month, year;  
} typedef date_t;
```

```
struct Student  
{
```

```
    char    name[20];  
    date_t  birthday;  
    float   average;  
} typedef student_t;
```

```
void main()  
{
```

```
    student_t  s;
```

```
    printf("Enter name of student and average:\n");  
    scanf("%s %f", s.name, &s.average);  
    printf("Please enter DOB: ");  
    scanf("%d %d %d", &s.birthday.day,  
          &s.birthday.month, &s.birthday.year);
```

```
    printf("The student's details: \n");  
    printf("Name: %s, Average: %.2f,  
          DOB: %d-%d-%d\n",  
          s.name, s.average, s.birthday.day,  
          s.birthday.month, s.birthday.year);
```

```
}
```

```
Enter name of student and average:  
gogo 99.5  
Please enter DOB: 28 8 1983  
The student's details:  
Name: gogo, Average: 99.50, DOB: 28-8-1983  
Press any key to continue . . . _
```

# מבנה בתוך מבנה ופונקציות - דוגמא

```
struct Date
{
    int day, month, year;
} typedef date_t;

struct Student
{
    char    name[20];
    date_t  birthday;
    float   average;
} typedef student_t;

void printDate(date_t d)
{
    printf("%d-%d-%d", d.day, d.month, d.year);
}

void printStudent(student_t s)
{
    printf("Name: %s, Average: %.2f, DOB: ",
           s.name, s.average);
    printDate(s.birthday);
    printf("\n");
}

void main()
{
    student_t s;

    printf("Enter name of student and average:\n");
    scanf("%s %f", s.name, &s.average);
    printf("Please enter DOB: ");
    scanf("%d %d %d", &s.birthday.day,
           &s.birthday.month, &s.birthday.year);

    printf("The student's details: \n");
    printStudent(s);
}
```

# יעילות

- כאשר מעבירים מבנה לפונקציה העתק שלו מועבר למחסנית של הפונקציה
- מבנה יכול להכיל המון שדות ולכן פעולת ההעתקה יכולה להיות "יקרה" מבחינת זמן
  - אנחנו כמובן לא נרגיש זאת
- לכן, אם מעבירים מבנה לפונקציה, מטעמי יעילות נעדיף להעביר מצביע למבנה, כדי לחסוך את ההעתקה
  - במקרה זה גם נעביר את המצביע כ- `const` כהצהרה על כך שהפונקציה לא תשנה את תוכנו

# מבנה בתוך מבנה ויעילות - דוגמא

```
struct Date
{
    int day, month, year;
} typedef date_t;

struct Student
{
    char name[20];
    date_t birthday;
    float average;
} typedef student_t;

void printDate(const date_t* d)
{
    printf("%d-%d-%d", d->day, d->month, d->year);
}

void printStudent(const student_t* s)
{
    printf("Name: %s, Average: %.2f, DOB: ",
           s->name, s->average);
    printDate(&(s->birthday));
    printf("\n");
}

void main()
{
    student_t s;

    printf("Enter name of student and average:\n");
    scanf("%s %f", s.name, &s.average);
    printf("Please enter DOB: ");
    scanf("%d %d %d", &s.birthday.day,
              &s.birthday.month, &s.birthday.year);
    printf("Second student's details: \n");
    printStudent(&s);
}
```

# מערך של מבנים

□ נרצה לשמור במערך את אוסף הסטודנטים הרשומים בכיתה

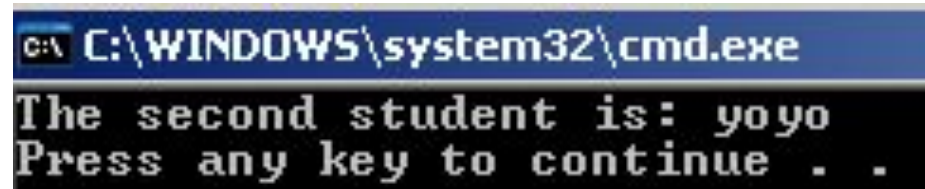
```
#define MAX_STUDENTS 3
```

```
struct Student  
{  
    char name[20];  
    float average;  
} typedef student_t;
```

```
void main()  
{  
    student_t students[MAX_STUDENTS] = { {"momo", 90.5},  
                                           {"yoyo", 85},  
                                           {"gogo", 78.6} };
```

הסטודנט השני במערך

```
printf("The second student is: %s\n", students[1].name);  
}
```



```
C:\WINDOWS\system32\cmd.exe  
The second student is: yoyo  
Press any key to continue . .
```

הסוגריים הכחולים עבור איתחול המערך,  
וכל זוג סוגריים ירוקים מאתחלים מבנה אחד בתוך המערך

# העברת מערך מבנים לפונקציה

```
#define MAX_STUDENTS 3
```

```
struct Student
```

```
{  
    char name[20];  
    float average;  
} typedef student_t;
```

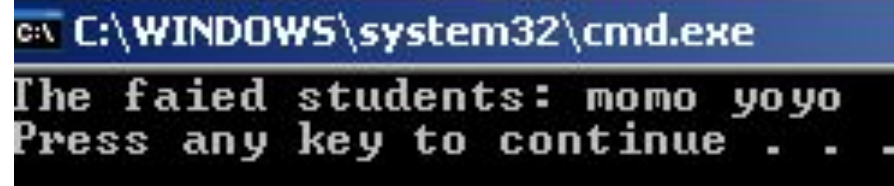
```
void printFail(const student_t students[], int size)
```

```
{  
    int i;  
    for (i=0 ; i < size ; i++)  
        if (students[i].average < 60)  
            printf("%s ", students[i].name);  
    printf("\n");  
}
```

```
void main()
```

```
{  
    student_t students[MAX_STUDENTS] = { {"momo", 50.5}, {"yoyo", 55}, {"gogo", 78.6} };  
  
    printf("The faied students: ");  
    printFail(students, MAX_STUDENTS);  
}
```

תזכורת: כאשר מעבירים מערך לפונקציה למעשה מעבירים את כתובת ההתחלה שלו, וכל שינוי ערכי המערך בפונקציה ישנה את ערכי המערך המקורי



C:\WINDOWS\system32\cmd.exe  
The faied students: momo yoyo  
Press any key to continue . . .



# העברת מערך מבנים לפונקציה

□ בדיוק כמו מערך רגיל, מועברת כתובת ההתחלה...

```
void printStudents(const student_t students[], int size)
{
    int i;
    for (i=0 ; i < size ; i++)
        printf("Name: %s\tAverage: %f\n",
            students[i].name, students[i].average);
}

#define MAX 3

struct Student
{
    char name[20];
    float average;
} typedef student_t;

void readStudents(student_t students[], int size)
{
    int i;
    for (i=0 ; i < size ; i++)
    {
        printf("Enter name and average for student #%d:", i+1);
        scanf("%s%f", students[i].name, &students[i].average);
    }
}

void main()
{
    student_t students[MAX];

    readStudents(students, MAX);
    printf("All students: \n");
    printStudents(students, MAX);
}
```

עובדים על  
המערך המקורי

# גודל מבנה בזיכרון

□ גודל כל משתנה מסוג מבנה יהיה סכום גודלי השדות מעוגל כלפי מעלה לכפולה של 4

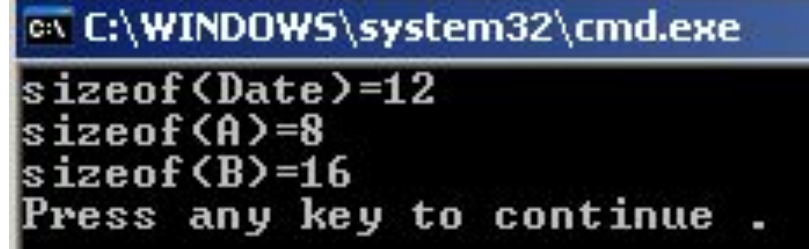
```
#include <stdio.h>

struct Date
{
    int day;
    int month;
    int year;
} typedef date_t;

struct A
{
    int num;
    char ch;
};

struct B
{
    char arr[10];
    int num;
};

void main()
{
    printf("sizeof(Date)=%d\n", sizeof(date_t));
    printf("sizeof(A)=%d\n", sizeof(struct A));
    printf("sizeof(B)=%d\n", sizeof(struct B));
}
```



```
C:\WINDOWS\system32\cmd.exe
sizeof(Date)=12
sizeof(A)=8
sizeof(B)=16
Press any key to continue .
```

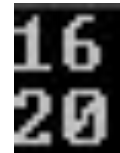
## גודל מבנה בזיכרון (2)

---

```
struct Date
{
    int    day, month, year;
    char  delimiter;
} typedef date_t;
```

```
struct Student
{
    char    name[3];
    date_t  birthday;
} typedef student_t;
```

```
void main()
{
    printf("%d\n", sizeof(date_t));
    printf("%d\n", sizeof(student_t));
}
```



16  
20

# תרגיל 1:

- הגדר את המבנה "נקודה" שנתוניו הם  $x$  ו-  $y$
- הגדר את המבנה "מרובע" שנתוניו הם הנקודה השמאלית העליונה והנקודה הימנית התחתונה
- כתוב `main`:

- הגדר "מרובע" ואתחל אותו להיות בפינה השמאלית במיקום  $(0, 0)$  ובפינה הימנית התחתונה במיקום  $(4, 3)$
- הגדר "מרובע" וקרא לתוכו נתונים מהמשתמש
- הדפס את נתוני שני המרובעים
- יש להקפיד כל מודלוריות (פירוק לפונקציות)

- הגדר 3 מרובעים וקרא לתוכם נתונים. הדפס את קאורדינטת ה-  $x$  של הנקודה השמאלית העליונה של המרובע השני.

# ביחידה זו למדנו:

---

- מהו מבנה (struct)
- typedef
- איתחול מבנה
- השמת מבנים
- השוואת מבנים
- העברת מבנה לפונקציה
- מבנה בתוך מבנה
- מערך של מבנים
- גודל מבנה בזיכרון

## תרגיל 2:

- הגדר את המבנה `Appear` ששדותיו הם `to` ומספר.
- כתוב פונקציה המקבלת מחרוזת ומערך מטיפוס `Appear` שגודלו 26 (כמספר האותיות באנגלית).

	0	1	2	...	24	25
ch	'A'	'B'	'C'	...	'Y'	'Z'
count	0	2	1		1	1

למשל המחרוזת: "cZb!yB"

- הפונקציה תחזיר שיש 4 אותיות שונות במחרוזת ותמלא את השדות במערך באופן הבא:
  - במקום ה- 0 את התו A ואת כמות המופעים שלו במחרוזת
  - במקום ה- 1 את התו B ואת כמות המופעים שלו במחרוזת
  - ...
  - במקום ה- 25 את התו Z ואת כמות המופעים שלו במחרוזת