

קורס יסודות התכנות בשפת C

פרק 10

מבני נתונים - מבנים Struct's



ד"ר שייקה בילו

יועץ ומרצה בכיר למדעי המחשב וטכנולוגית מידע
מומחה למערכות מידע חינוכיות, אקדמיות ומנהליות

כתובות – איך מתייחסים אליהן

2

- שפת C מאפשרת לנו לדעת מהי הכתובת בזיכרון שבה נשמר משתנה מסוים.
- מיקום הכתובת בזיכרון מאפשר גישה ישירה לכתובות בזיכרון דבר המייעל את עבודת המחשב.
- הפעולה & נותנת את הכתובת של המשתנה.
 - למשל &i היא הכתובת בזיכרון של המשתנה i.
 - בדוגמא הקודמת &i החזיר לנו את הכתובת 0x7fff9575c054.

כתובות – איך ניגשים אליהן

3

- הפעולה & נותנת את הכתובת של משתנה הצמוד לה.
 - למשל &i היא הכתובת בזיכרון של המשתנה i.
 - בדוגמא הקודמת &i היה נותן 0x7fff9575c054.
- הפעולה * ניגשת לערך שנמצא בכתובת מסוימת.
 - למשל (0x7fff9575c054)* זה הערך שנמצא בכתובת 0x7fff9575c054.
 - (&ch)* זה הערך של המשתנה ch (כי זה הערך שנמצא בכתובת של ch).

חזרה כתובות בזיכרון - פעולות

4

- מציאת הכתובת ניתן לבצע באמצעות: $\&i$
- גישה למשתנה לפי הכתובת ניתן לבצע עם הפעולה: *
- אופן הכתיבה הנ"ל מזמין לנו אפשרות לשנות כמה ערכים של משתנים בפונקציה.
- כל כתיבה ישירה בזיכרון ללא שימוש בשם המשתנה היא מהירה יותר ומבצעת עדכון של ערך המשתנה באופן מהיר ומדויק יותר.

חזרה עבודה עם כתובות - מצביעים

5

- באופן כללי, כדי להגדיר משתנה מסוג מצביע, יש לרשום את סוג המשתנה שעליו מצביעים, ולהוסיף את סימן ה- * לפני שם המשתנה.

○ עוד דוגמאות:

double *ptr;

float *ptr;

int *ptr;

char *ptr

חזרה מצביעים – שתי נקודות נוספות

6

- במידה ויש צורך לסמן שמצביע לא מצביע למשתנה (למשל לפני שהתחלנו להשתמש בו), אזי מקובל לתת לו את הערך NULL.
- למשל:

```
int *ptr_to_int=NULL;
```

○ NULL הוא קבוע שערכו 0 שמוגדר ע"י **#define** בספרייה **stdlib.h**.

- קידוד ההדפסה ב- **printf** להדפסת כתובת בזיכרון הוא **“%p”**, למשל להדפסת כתובת המשתנה **i** נכתוב:

```
printf(“%p”, &i);
```

חזרה מצביעים – דוגמא

7

- נתונה בעיה:

יש לקבל שני ערכים הנמצאים בשני משתנים שונים ולבצע החלפה ביניהם.

- נתאר פונקציה שמקבלת כתובות של אותם שני משתנים מסוג **int**, ומחליפה בין הערכים שלהם תוך כדי שימוש במצביעים.

חזרה מצביעים – דוגמא

8

- עד עכשיו לא יכולנו לכתוב פונקציה להחלפת ערכים של שני משתנים מסוג `int`, כי שינוי משתנים מסוג `int` בפונקציה לא משפיע על ערכם המקורי, ופונקציה יכולה להחזיר רק ערך אחד.
- כעת כשיש לנו אפשרות להשתמש בכתובות המשתנים שבזיכרון, כלומר במצביעים נוכל לבצע את המשימה באמצעות פונקציית `swap` פשוטה.

חזרה: פונקציה להחלפה בין ערכי משתנים

9

```
void swap(int *first, int *second)
{
    int temp;
    temp=*first;
    *first=*second;
    *second=temp;
}

int main()
{
    int mispar1=10, mispar2=20;
    swap(&mispar1,&mispar2);
}
```

חזרה עוד על פונקציות ומצביעים

10

- נציין שהערך המוחזר מפונקציה יכול להיות גם מצביע, למשל:
`int *address_number(int *ptr1, int *ptr2);`
- אבל החזרת כתובת משתנה שהוגדר בתוך הפונקציה תגרום לשגיאה.
- הסיבה לכך היא שהמשתנים המוגדרים בתוך הפונקציה נעלמים עם סיומה.
- למשל הפונקציה הבאה תגרום לשגיאה:

```
int *give_pointer_to_zero()
{
    int i=0;
    return &i;
}
```

חזרה גישה לתאי-מערך לפי הכתובת

11

- אם הגדרנו למשל את המערך והמצביע:

(מצביע על התא הראשון במערך)

```
int arr [SIZE];
```

```
int *arr_ptr;
```

```
arr_ptr=arr;
```

- אז אפשר לגשת לתאי-המערך גם לפי הכתובת שלהם, כמו משתנים אחרים.
- תזכורת מצביע ללא הכוונה הוא מצביע חסר משמעות.

חזרה גישה לתאי-מערך לפי הכתובת

12

- 3 הפעולות הבאות עושות אותו דבר (מבצעות השמה של המספר 100 בתא מס' 5 במערך):

```
arr [5]=100;
```

```
*(arr+5)=100;
```

גישה לתא שנמצא 5 תאים אחרי התא הראשון

```
*(arr_ptr+5)=100;
```

- המחשב יודע כמה בתים להתקדם כשאנחנו מבקשים להתקדם ב-5 תאים קדימה.

- הגדרנו שטיפוס הערכים הוא **int** והוא יודע כמה בתים כל **int** תופס (כאמור, ערכי המערך שמורים בזיכרון ברצף).

חזרה גישה למערך לפי כתובת – דוגמא נוספת

13

- עבור הדפסת מערך, 3 האפשרויות הבאות עושות בדיוק אותו דבר:

```
int i, arr [SIZE];
```

```
int *ptr;
```

- הדפסה על-ידי גישה רגילה לתאי המערך

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

```
printf(“%d”, arr [i]);
```

- הדפסה על-ידי גישה לכל תא לפי הכתובת

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

```
printf(“%d”, *(arr+i));
```

שלו יחסית לתא הראשון

```
for (ptr=arr; ptr <= &arr [9]; ptr++)
```

```
printf(“%d”, *ptr);
```

- הדפסה ע"י קידום מצביע מכתובת התא הראשון עד כתובת התא האחרון

חזרה מערכים ופונקציות - הסבר

14

- כפי שהוסבר בעבר כאשר מעבירים מערך לפונקציה, השינויים שנעשה בפונקציה ישפיעו על המערך המקורי.
- הדבר נובע מכך שלפונקציה מועברת הכתובת בזיכרון של המערך המקורי, והיא פועלת ישירות על ערכיו (ולא על העתק שלהם).
- למשל, כפי שאמרנו, הפונקציה הבאה תאפס מערך שמועבר אליה:

```
void zero_arr (int arr[ ], int size)
{
    int i;
    for(i=0 ; i< SIZE; i++)
        arr[i]=0;
}
```

חזרה מערכים ופונקציות - הסבר

15

- אמרנו בעבר שכשמעבירים מערך לפונקציה שינויים שנעשה בפונקציה ישפיעו על המערך המקורי.
- זה נובע מכך שלפונקציה מועברת הכתובת בזיכרון של המערך המקורי, והיא פועלת ישירות על ערכיו (ולא על העתק שלהם).
- כפי שאמרנו, הפונקציה הבאה תאפס מערך מועבר:

```
void zero_arr (int *arr)
{
    int i;
    for(i=0 ; i< SIZE; i++)
        arr[i]=0;
}
```

נציין שאם פונקציה מצפה לקבל מצביע, אפשר להעביר אליה מערך מהסוג הזה, כי בשני המקרים מה שמועבר הוא כתובת בזיכרון.

16

שאלות על השעור הקודם?

נושאי הפרק מבנים

17

```
typedef struct {  
    int lengthInSeconds;  
    int yearRecorded;  
} Song;
```

song1

lengthInSeconds: 213
yearRecorded: 1994

song2

lengthInSeconds: 248
yearRecorded: 1988

טיפוס משתנה חדש - מבנה:

- הגדרה ואתחול מבנים
- כתיבה מקוצרת של הגדרת מבנה
- פעולות על מבנים
- מבנים ופונקציות
- מבנים ומצביעים
- מבנה בתוך מבנה
- מערכים של מבנים

טיפוסי משתנים בשפת C

18

- הכרנו במהלך הקורס את טיפוסי המשתנים הבסיסיים שניתן לייצג בשפת C:

1. מספר שלם, ארוך/קצר/ללא סימן. **(int)**
2. ממשי. **(float, double)**
3. תו. **(char)**
4. מצביע. **(int *ptr)**
5. מחרוזת. **(char str[];)**
6. מערך. **(int arr[])**

- כשכתבנו תוכניות לפתרון בעיות כלשהן, יצגנו את נתוני התוכנית ע"י משתנים מהטיפוסים האלה.

טיפול משתנים בשפת C

19

- כידוע, בבעיות מציאותיות יכולים להיות אובייקטים מורכבים, שלא בהכרח מתאימים לייצוג ע"י מספר, תו, או אפילו מערך.
- למשל בבעיות גיאומטריות מדובר תמיד על נקודות, עד ישרים, על סוגי מצולעים, וכדומה.
- בבעיות מנהליות, לדוגמא: מנהלת-האוניברסיטה מטפלת בפקולטות, מרצים קורסים, סטודנטים, מבחנים, ספרייה, בטחון, משאבי אנוש, וכו'.
- כאשר כותבים תוכנות לפתרון בעיות מציאותיות, נוח מאד שיש טיפוסים משתנים היכולים לייצג את האובייקטים המציאותיים האלה.

טיפול משתנים בשפת C

20

- למשל, לטיפול במספרים מרוכבים היינו רוצים טיפוס המייצג מספר מרוכב על כל המשתמע מכך.
- לפתרון בעיות גיאומטריות היינו רוצים טיפוס המתאר נקודה במישור וטיפוסים המתארים מצולעים מסוגים שונים.
- במערכת המידע של האוניברסיטה נשאף לטיפול סימנים משתנים עבור סטודנט, מרצה, חוג, קורס, התמחות, עבודה מעשית, תרגול וכו' (שכל אחד מהם כולל כמה שדות-מידע רלוונטיים).

יצירת טיפוסים משתנים חדשים

21

- שפת C מאפשרת לנו להגדיר טיפוסים משתנים חדשים שיתאימו לבעיה שאנחנו רוצים לפתור.
- טיפוסים הנתונים האלה יכולים להכיל בתוכם כמה שדות כאשר כל שדה יכול להיות מטיפוס אחר עבור אחסון מידע מסוג שונה.
- הטיפוס החדש הזה נקרא "מבנה" (structure) והפקודה שתשמש אותנו להגדיר אותו בשפת C נקראת: **struct**.
- בדרך-כלל המבנה יכול כלול כמה משתנים פנימיים, הנקראים "שדות" והם מוגדרים כפי שלמדנו להגדיר משתנים רגילים מכול טיפוס מוכר וידוע.

הגדרת מבנה חדש – דוגמא

22

- נקודה במישור מיוצגת על-ידי שתי קואורדינטות, X ו- Y .
- הגדרת מבנה שייצג נקודה תיראה למשל כך:

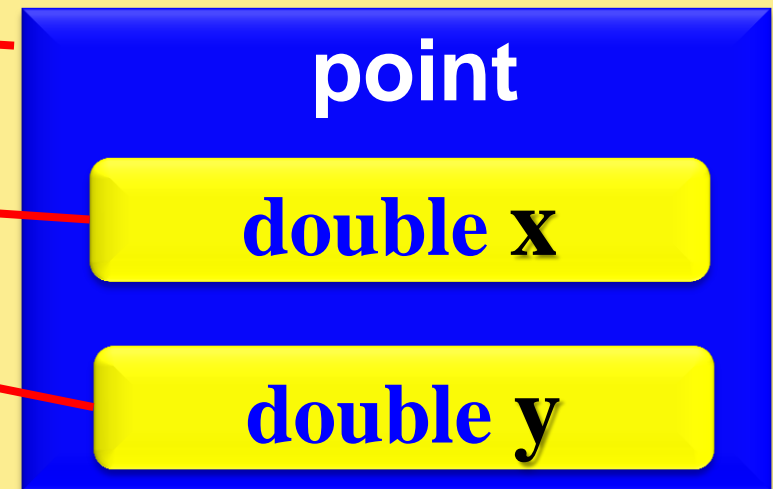
struct point ←

{

double x; ←

double y; ←

};



- ההגדרה הזו לא תופיע בתוך פונקציה, אלא בתחילת הקובץ (בד"כ מייד אחרי שורות ה-`#include` וה-`#define`).
- כעת ניתן להגדיר בתוכנית משתנים מהסוג הזה, כפי שנבחן בהמשך.

הגדרת מבנה חדש – המשך הדוגמא

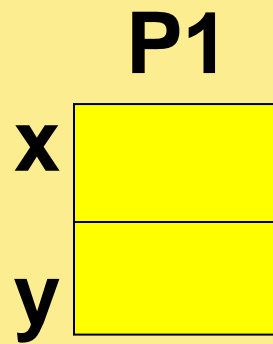
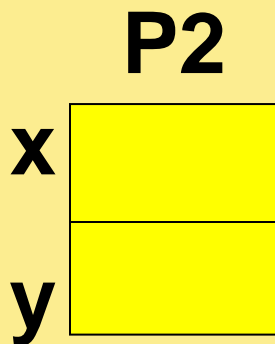
23

- דוגמא להגדרת משתנים מהטיפוס החדש:

```
int main()
{
    struct point P1,P2;
    return 0;
}
```

שני משתנים
חדשים מטיפוס
point

- בכך הגדרנו שתי נקודות (שני משתנים מסוג מבנה point), שלכל אחת מהן יש שני שדות.



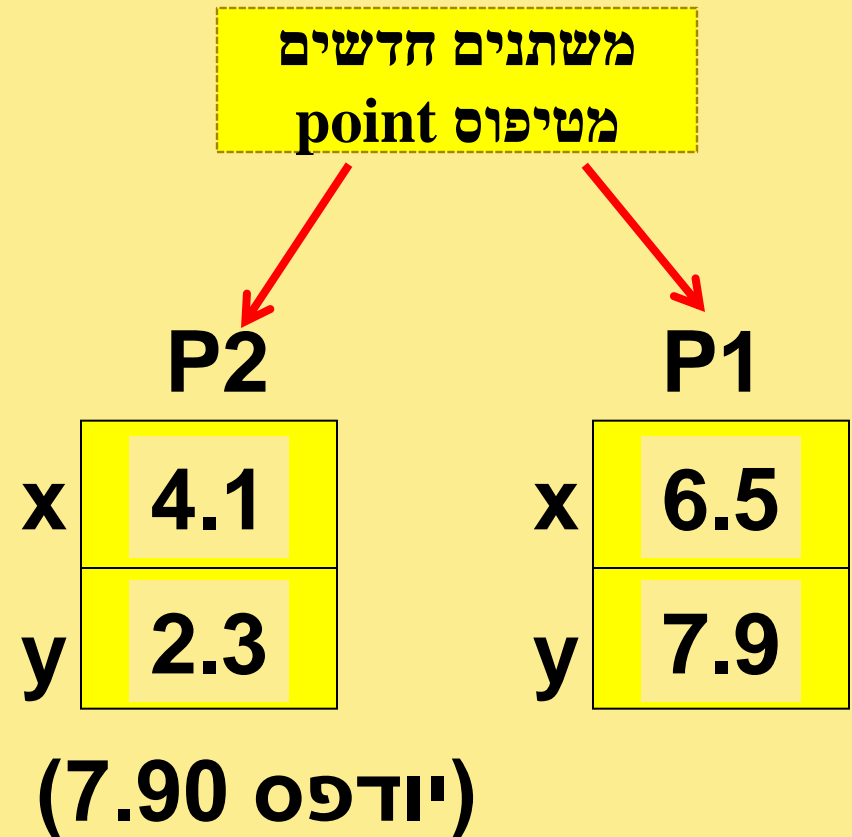
מפת הזיכרון של
שני המשתנים
הכוללים שני שדות

הגדרת מבנה חדש – המשך הדוגמא

24

- ניתן לגשת לכל אחד מהשדות של מבנה ולכתוב/לקרוא אליו/ממנו.
- למשל:

```
int main()
{
    struct point P1,P2;
    P1.x = 6.5;
    P1.y = 7.9;
    P2.x = 4.1;
    P2.y = 2.3;
    printf("%.2lf\n", P1.y);
    return 0;
}
```



הגדרת מבנים Syntax

25

- הגדרת מבנה (טיפוס משתנה חדש):

שם הטיפוס החדש **struct**

```
{  
    שם-שדה טיפוס;  
    .....  
    שם-שדה טיפוס;  
};
```

- הגדרת משתנה מהסוג החדש שהגדרנו:

- שם משתנה שם הטיפוס החדש **struct**

- פנייה לשדה של משתנה שהוא מבנה נעשית ע"י פניה למשתנה וממנו לשדה הפנימי:

variable.member

הגדרת מבנה חדש - הדוגמא הקודמת

26

```
#include <stdio.h>
struct point {
    double x, y;
};
int main()
{
    struct point P1,P2;
    P1.x = 6.5;
    P1.y = 7.9;
    P2.x = 4.1;
    P2.y = 2.3;
    printf("%.2lf\n", P1.y);
    return 0;
}
```

	P2
x	4.1
y	2.3

	P1
x	6.5
y	7.9

אתחול מבנים

27

- גם משתנים מטיפוס חדש אפשר לאתחל בשורת ההגדרה. למשל בהתייחס לדוגמא הקודמת:

```
struct point P1 = {6.5,7.9};
```

השדה x של המשתנה P1 יאותחל ל-6.5.

והשדה y של המשתנה P1 יאותחל ל-7.9.

- באופן כללי, האתחול הוא לפי סדר השדות בהגדרת המבנה, כלומר האיבר הראשון ברשימת האתחול יכנס לשדה הראשון, השני לשני, וכן הלאה (אם האתחול הוא חלקי, השאר יאותחל באפסים).

סיכום ביניים - מבנים

28

person

char name[SIZE]

int age

char address[SIZE]

int class

- מהם מבנים
- הגדרת מבנים
- אתחול מבנים

29

שאלות?

פעולות על מבנים

30

- השמה מתבצעת בצורה הרגילה:

$P1 = P2;$

- התוכן של $P2$ מועתק לתוך $P1$. (העתקה של שדה אחר שדה)
- אם אחד השדות הוא מערך אז גם הוא מועתק (שינוי שלו ב- $P1$ לא ישפיע על $P2$).

- פעולות השוואה ($==$ והאחרות) ופעולות חשבון אינן פועלות עבור מבנים (זה לא יעבור קומפילציה).

- על מנת לבצע פעולות השוואה בין מבנים נשתמש בפונקציות.
- עבור כל פעולה נכתוב פונקציה רלוונטית.

דוגמא: פונקציה להשוואת מבני point

31

```
int equal(struct point p, struct point q)
{
    return ((p.x == q.x) && (p.y == q.y));
}
```

אם הנקודות זהות הפונקציה תחזיר את הערך 1,
אחרת הפונקציה תחזיר את הערך 0

דוגמא לשימוש במבנים

32

- פונקציה המחשבת מרחק בין שתי נקודות:

```
double dist(struct point p, struct point q)
{
    double dis;
    dis=(p.x - q.x)*(p.x - q.x)+(p.y-q.y)*(p.y-q.y);
    return sqrt(dis);
}
```

תזכור חשובה: כדי להשתמש בפונקציה `sqrt()` יש להוסיף בתחילת הקובץ את הספרייה: `#include <math.h>`

דוגמא לשימוש במבנים – מציאת מרחק בין שתי נקודות

33

```
#include <stdio.h>
struct point
{
    double x, y;
};

double dist(struct point p, struct point q)
{
    double distance;
    distance=(p.x-q.x)*(p.x-q.x)+(p.y-q.y)*(p.y-q.y);
    return sqrt(distance);
}
```

דוגמא לשימוש במבנים – מציאת מרחק בין שתי נקודות

34

```
int main()
{
    struct point p,q;
    printf("Enter x & y to first point\n");
    scanf("%lf%lf", &p.x,&p.y);
    printf("Enter x & y to second point\n");
    scanf("%lf%lf", &q.x,&q.y);
    printf("Their distance is %.2lf\n", dist(p,q));
    return 0;
}
```

דוגמא לשימוש במבנים – מה עושה התוכנית הבאה?

35

```
#include <stdio.h>

#define SIZE 81

struct quadric
{
    double num;
    int mis;
    char ch;
    char str[SIZE];
};
```

quadric

double num

int mis

char ch

char str[SIZE]

דוגמא לשימוש במבנים – מה עושה התוכנית הבאה?

36

```
int main()
{
    struct quadric list;
    flushall();
    printf("Enter 4 parameters: double, integer,
           char, string:\n");
    scanf("%lf %d %c %s"
          ,&list.num,&list.mis,&list.ch,&list.str);
    printf("x=%.2lf, y=%d, c=%c, s=%s\n"
          ,list.num, list.mis, list.ch,list.str);
    return 0;
}
```

שאלות?

פונקציות ומבנים

38

- איך מבני הנקודות מועברים לפונקציה?
- הם מועברים על ידי העתקת הערכים של השדות שלהם לפרמטרים של הפונקציה.
 - כמו במשתנים רגילים (call by value), כמו עבור `int`.
 - שינוי הנקודה בפונקציה לא ישנה את הנקודה ב- `main`.
- פונקציה יכולה להחזיר גם מבנה, כמו כל משתנה אחר (גם במקרה הזה הערכים יועתקו).
- מערך של מבנים הנשלח לפונקציה מטופל כמערך רגיל.

מערכים במבנים – נקודה לתשומת-לב

39

- אם אחד השדות של המבנה הוא מערך, ומעבירים את המבנה הזה לפונקציה, אז כל המערך מועתק (לא רק כתובת המערך מועברת, ממש כמו טיפול במשתנה רגיל).
- במקרה בו נבצע שינוי בתוך פונקציה למערך שהוא שדה של מבנה, אז לא תהיה לזה השפעה על המערך המקורי.
- למשל אם נגדיר את המבנה הבא כאשר SIZE הוא 10:

```
struct array_10  
{  
    int arr[SIZE];  
};
```

משתנה חדש המוגדר
כמערך ומשמש כשדה
פנימי בתוך מבנה:
int_array_10

מערכים ומבנים – נקודה לתשומת-לב

40

- הפונקציה הבאה לא תשפיע על המשתנה שמועבר אליה:

```
void zero_arr (struct array_10 A)
```

```
{  
    int i;  
    for(i=0; i<SIZE; i++)  
        A.arr[i]=0;  
}
```

הערך המאופס הוא ההעתק של המערך, כי מערך בתוך מבנה מועתק בהעברה לפונקציה. השמת הערך 0 מתבצעת לתוך השדה הרלוונטי בתוך המשתנה A.array.

- כדי לשנות את הנתונים של המערך המקורי נשתמש במצביע, כפי שנראה בהמשך כי אחרת הערכים לא ישמרו במערך.
- אפשרויות אחרות הן: או שנעביר לפונקציה מצביע למבנה או שנגדיר במבנה מצביע במקום מערך.

מבנים - פעולות

41

- השמה בין מבנים מותרת (מתבצעת העתקה שדה-שדה)
- פעולות השוואה ופעולות חשבון לא מותרת כיוון שלא ניתן לבצע ללא שימוש בפונקציות ייעודיות לנושא.
- ניתן להעביר מבנים לפונקציה ולהחזיר ממנה מבנה, והם מועתקים אליה וממנה, כמו טיפוס-משתנה רגיל (כמו **int**)
- מערך בתוך מבנה מועתק וכל שינוי בו לא נשמר לאחר היציאה מהפונקציה.
- מערך של מבנים מועבר לפונקציה וכל שינוי בו נשמר לאחר היציאה מהפונקציה.

מבנים – כתיבה מקוצרת

42

- שמו של הטיפוס החדש שהגדרנו הוא **struct point**
- אפשר לחסוך את כתיבת המילה **struct** באמצעות הפקודה **typedef**, שמאפשרת לתת שם חדש לטיפוס-משתנה:

```
struct point
```

```
{
```

```
    double x, y;
```

```
};
```

```
typedef struct point Point;
```

טיפוס קיים

שם חדש

מבנים – כתיבה מקוצרת

43

- אפשר לכתוב את שתי השורות יחד באופן הבא:

```
typedef struct point
{
    double x, y;
} Point;
```

- בזכות ה- **typedef** נוכל לכתוב כעת את המילה **Point** במקום לכתוב את ההגדרה המלאה **struct point** (זה שקול לחלוטין).
- למשל:

```
double dist(Point p, Point q)
```

מבנים – כתיבה מקוצרת

44

- אפשר גם לכתוב את הגדרת המבנה באופן הבא:

```
typedef struct  
{  
    double x, y;  
} Point;
```

- בזכות ה- **typedef** נוכל לכתוב כעת את המילה **Point** במקום לכתוב את ההגדרה המלאה **struct point** (זה שקול לחלוטין).
- למשל:

```
double dist(Point p, Point q)
```

עוד על typedef

45

- נציין שהפקודה **typedef** יכולה לשמש גם כדי לתת שם חדש לטיפוס קיים שאיננו מבנה.
- למשל אם נרצה לתת שם מיוחד ל-

unsigned int

- אפשר לכתוב:

```
typedef unsigned int UINT;
```

שם קיים שם חדש

- עכשיו בכל מקום בתוכנית שנרצה נוכל להשתמש ב-**UINT** כדי להגדיר **unsigned int**.
- **unsigned int ui;** שקול ל- **UINT ui;**

עוד על typedef

46

```
typedef struct
{
    double real, image;
} Complex;
int main()
{
    Complex c={5.2,7.1};
    Complex *pc;
    pc = &c;
}
```

```
struct complex
{
    double real, image;
};
typedef struct complex
    complex_t;
int main()
{
    complex_t c={5.2,7.1};
    complex_t *pc;
    pc = &c;
}
```

מבנים – סיכום ביניים

47

- שימוש במבנים מאפשר הגדרת טיפוסי משתנים חדשים שקרובים לעולם האמיתי, מכווני הבעיה שמבקשים לפתור.
- בשימוש במבנים לא ניתן להשתמש בפעולות החשבון וההשוואה הרגילות- יש להגדיר פונקציות עבורן בהתאם לצורך.
- בשימוש במבנים ניתן להשתמש כמו בטיפוסים רגילים מבחינת העברה לפונקציה (למעט העובדה שמערך בתוך מבנה מועתק ולא מועבר המערך המקורי).

48

שאלות?

מבנים ומצביעים

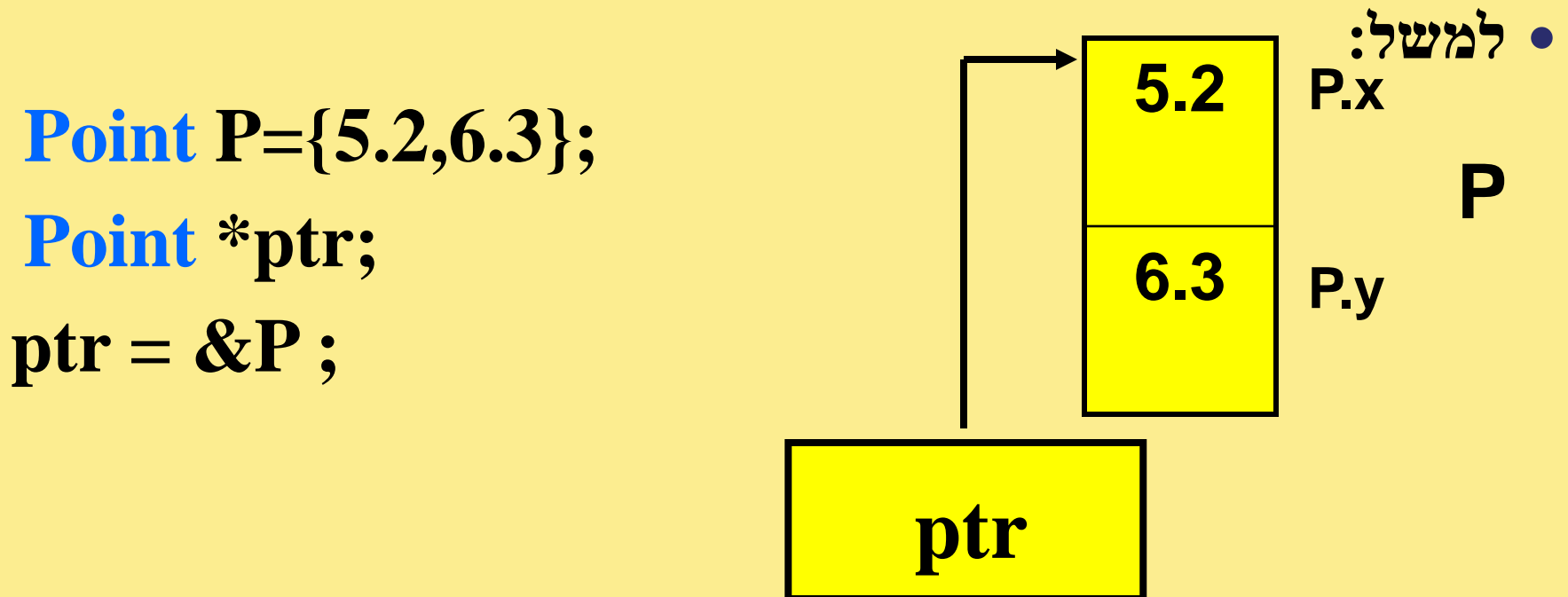
49

- אם המבנה שלנו מורכב מהרבה מאוד משתנים, אז העתקת כל המבנה לתוך הפרמטרים של הפונקציה דורשת זמן וזיכרון רבים.
- כמו-כן, כמו במשתנים רגילים, לפעמים נשאף לשנות יותר ממבנה אחד מתוך הפונקציה (ולא רק להחזיר מבנה אחד).
- כדי לטפל בשני הנושאים האלה נוכל לשלוח לפונקציה מצביעים למבנים, כמו ששלחנו מצביעים למשתנים מסוגים אחרים.
- בקריאה לפונקציה ישוכפל רק המצביע לכל מבנה, ובעזרת המצביע נוכל לשנות את המבנה המקורי.

מבנים ומצביעים

50

- בדיוק כמו טיפוסים-משתנים שהכרנו בעבר, אפשר להגדיר מצביע לטיפוס שהוא מבנה (כתובת ההתחלה של מבנה היא כתובת השדה הראשון שלו).



```
Point P={5.2,6.3};  
Point *ptr;  
ptr = &P ;
```

מצביעים ומבנים – גישה לשדות המבנה

51

- בדוגמא זו, כדי להגיע לשדות של P דרך ptr שמצביע על P, אפשר להשתמש ב- * כרגיל:

$P.x = 3.5;$ שקול ל- $(*ptr).x = 3.5;$

$P.y = 7.4;$ שקול ל- $(*ptr).y = 7.4;$

- יש צורך בסוגריים כי אחרת לנקודה יש קדימות על פני ה- *.

מצביעים ומבנים – גישה לשדות המבנה

52

- בדוגמא זו, כדי להגיע לשדות של P דרך ptr שמצביע על P, אפשר להשתמש ב- * כרגיל:

$P.x = 5.2;$ שקול ל- $(*ptr).x = 5.2;$

$P.y = 8.9;$ שקול ל- $(*ptr).y = 8.9;$

- אבל בדרך-כלל נשתמש לצורך זה בסימון מקוצר: חץ ->

$P.x = 5.2;$ שקול ל- $ptr \rightarrow x = 5.2;$

$P.y = 8.9;$ שקול ל- $ptr \rightarrow y = 8.9;$

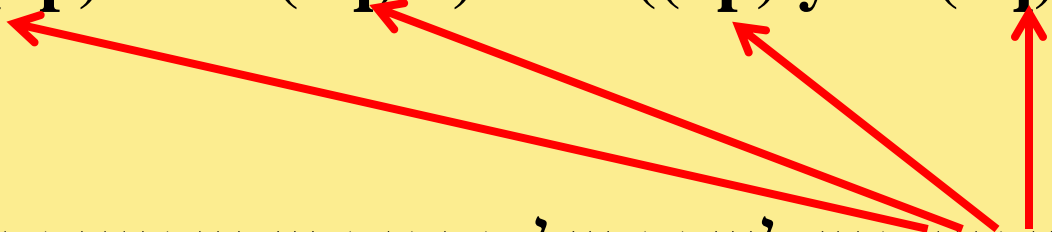
- כלומר משמעות החץ היא גישה לשדה במבנה שמצביעים עליו.

מצביעים ומבנים - דוגמא

53

- הקריאה לפונקציה ע"י `equal(&P1,&P2)`
- פונקציית ההשוואה כשמועברים מצביעים:

```
int equal( struct Point *p, struct Point *q)
{
    return (((*p).x == (*q).x) && ((*p).y == (*q).y));
}
```



- הגנישה עם מצביע היא לשדה של המבנה שמצביעים עליו.

מצביעים ומבנים - דוגמא

54

- הקריאה לפונקציה ע"י `equal(&P1,&P2)`
- פונקציית ההשוואה כשמועברים מצביעים:

```
int equal( struct Point *p, struct Point *q)
{
    return ((p->x == q->x) && (p->y == q->y));
}
```

- הגישה עם חץ `->` היא לשדה של המבנה שמצביעים עליו.

מצביעים ומבנים - דוגמא

55

- הקריאה לפונקציה ע"י `equal(P1, P2)`

```
int equal(struct Point p, struct Point q)
```

```
{
```

```
    return ( (p.x == q.x) && (p.y == q.y));
```

```
}
```

- הגישה עם סימן ה- נקודה . היא ישירות לשדה של מבנה

מצביעים ומבנים – עוד דוגמא

56

- נחשב את המרחק כשמועברים מצביעים ולא המבנים עצמם:

```
double dist(struct Point *p, struct Point *q)
{
    double dis,dx,dy;
    dx = p->x - q->x;
    dy = p->y - q->y;
    dis = dx*dx + dy*dy;
    return sqrt(dis);
}
```

- איך תיראה הקריאה ב-main?

```
printf("The distance is %g\n", dist(&P1,&P2));
```


מבנים Syntax - שימוש במצביעים

57

- הגדרת מבנה:

```
typedef struct
```

```
{
```

```
    double real;
```

```
    double image;
```

```
}Complex;
```

ה- members של המבנה



- הגדרת משתנים מסוג זה וקישור המצביע:

```
Complex c;
```

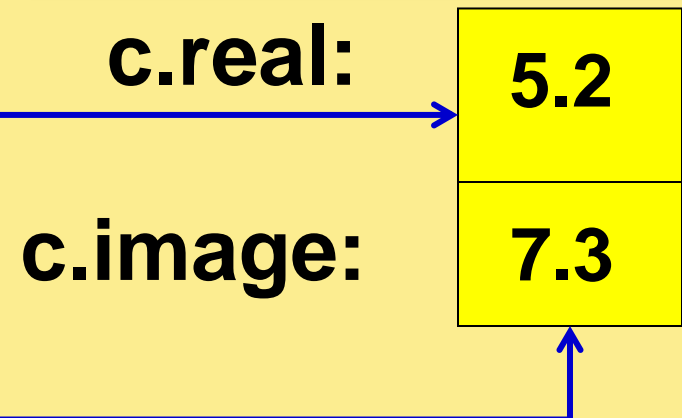
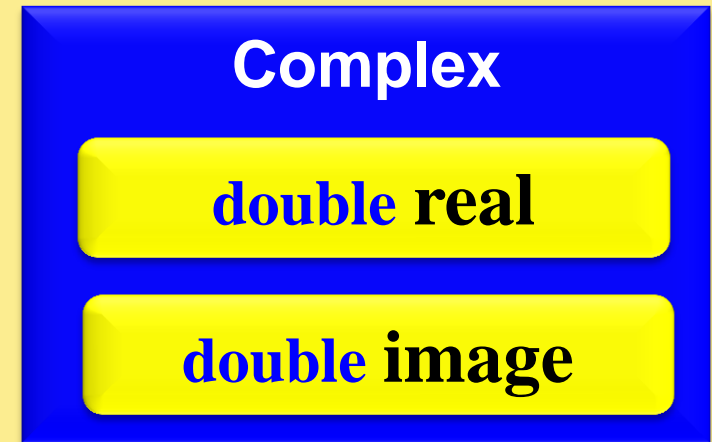
```
Complex *pc;
```

```
pc = &c;
```

מבנים Syntax - שימוש במצביעים

58

```
typedef struct {  
    double real, image;  
}Complex;  
  
int main()  
{  
    Complex c;  
    Complex *pc;  
    c.real = 5.2;  
    c.image = 7.3;  
}
```



חזרה - מבנים Syntax - כתובות ומצביעים

59

```
typedef struct {  
    double real, image;  
}Complex;
```

```
int main()  
{
```

```
    Complex c;
```

```
    Complex *pc;
```

```
    c.real = 5.2;    c.image = 7.1;
```

```
    pc = &c; → pc → 0x7fff9255c05c
```

```
    pc->real = 3.2;
```

```
    pc->image = 4.3;
```

```
}
```

c → 0x7fff9255c05c

c.real: 5.2

c.image: 7.1

pc → 0x7fff9255c05c

pc.real: 3.2

pc.image: 4.3

סיכום מבנים Syntax - כתובות ומצביעים

60

```
typedef struct  
{  
    double real, image;  
}Complex;
```

Complex

double real

double image

- לגישה ישירה ל- members של c נשתמש בנקודה (.).
- לגישה ישירה ל- members של pc נשתמש בחץ ימינה וסימן

המינוס (>-).

- משמעות הסימנים זהה ומאפשרת גישה ישירה לתוך השדה הפנימי של המבנה.

שאלות?

הגדרת המבנים והכרזה על פונקציות

62

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

```
typedef struct {
```

```
    double real, image;
```

```
}Complex;
```

```
void ScanComplex(Complex *ptr);
```

```
void PrintComplex (Complex n);
```

Complex

double real

double image

התוכנית הראשית

63

```
int main()
{
    Complex x;
    ScanComplex(&x);
    PrintComplex(x);
    return 0;
}
```

פונקציות הטיפול במבנים

64

```
void ScanComplex(Complex *ptr)
{
    printf("Enter two doubles real && image:\n");
    scanf("%lf%lf", &(ptr->real), &(ptr->image));
}
```

```
void PrintComplex (Complex n)
{
    printf("%.2lf + %.2lf = %.2lf\n", n.real,
    n.image, n.real+n.image);
}
```

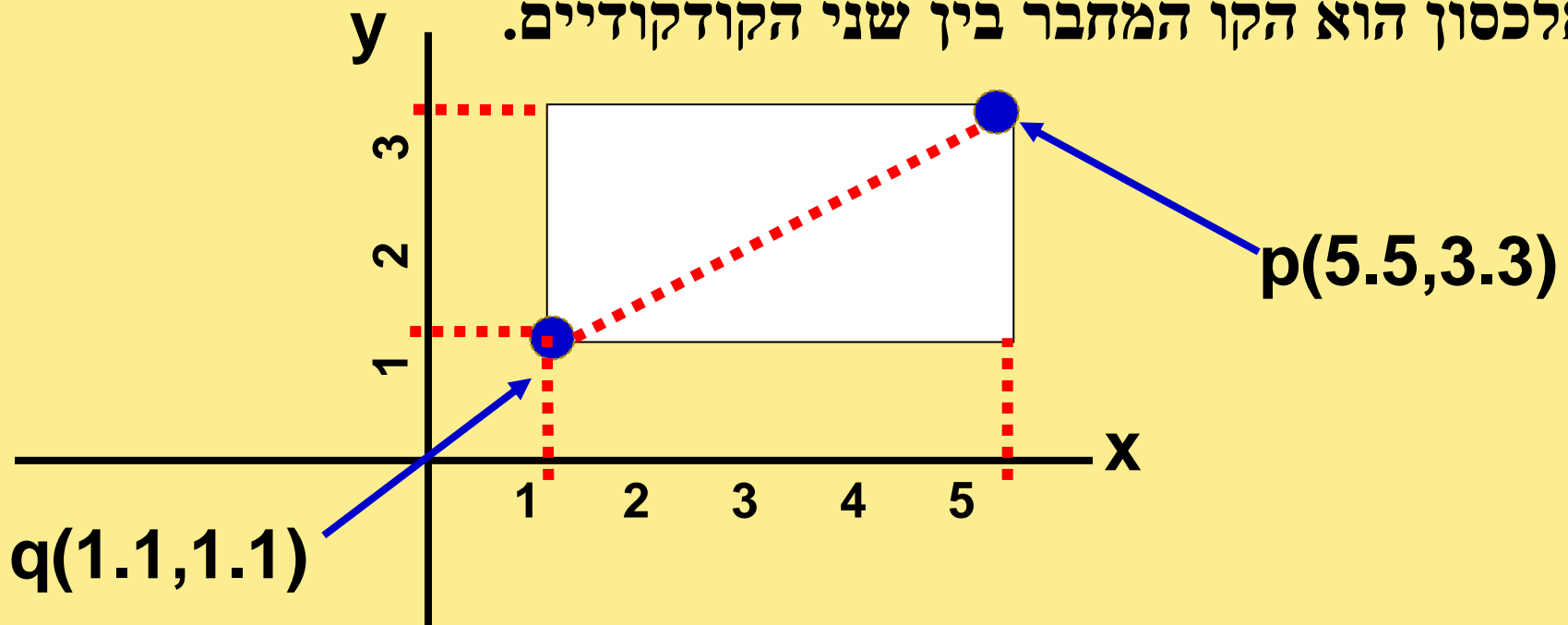

65

שאלות?

מבנה בתוך מבנה

66

- שדות של מבנה יכולים להיות בעצמם מבנים אחרים.
- לדוגמא, מבנה שמתאר מלבן במישור שמקביל לצירים.
- מלבן שמקביל לצירים נקבע על-ידי שני קודקודים נגדיים.
- אלכסון הוא הקו המחבר בין שני הקודקודים.

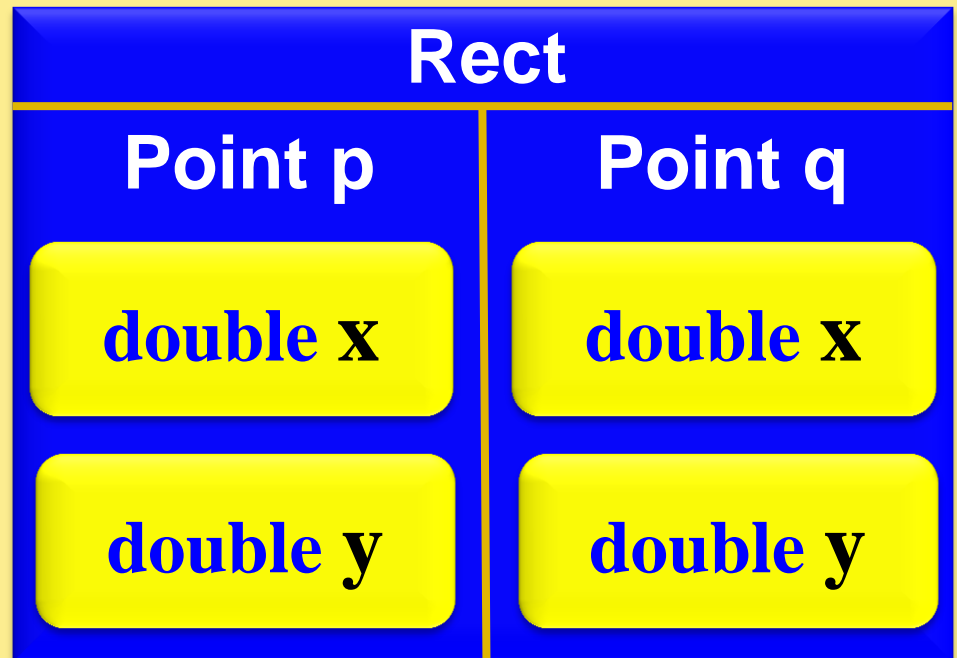


מבנה בתוך מבנה

67

- ברור יותר להגדיר ע"י 2 נקודות.
- נדגים כאן גם את הרישום המקוצר עם **typedef**:

```
typedef struct rect{  
    Point p;  
    Point q;  
}Rect;
```



הערה: יש חובה להגדיר את המבנה **Point** לפני המבנה **Rect**

מבנה בתוך מבנה – אתחול מבנים

68

- גם מבנה שיש בתוכו מבנה אפשר לאתחל בשורת ההגדרה:

Rect r = { {5.2,6.3} , {7.5,2.5} };

p q

- ערך האתחול הראשון משויך לשדה הראשון, השני לשני, וכן הלאה (ובאתחול חלקי השאר מאותחל לאפסים).

- אפשר כמובן גם לאתחל כל שדה ישירות אחרי ההגדרה:

r.p.x = 5.2;

r.p.y = 6.3;

r.q.x = 7.5;

r.q.y = 2.5;

שאלות?

מערך של מבנים

70

- כמו טיפוסים משתנים אחרים, אפשר להגדיר מערך של מבנים.
- מערך זה יוגדר ויכיל בתוכו שדות שהם מבנים הכוללים תתי שדות.
- למשל, אפשר להגדיר מערך של נקודות בגודל 20 (מוגדר ב-**SIZE**):

```
int main()
{
    Point arr[SIZE];
    ...
    return 0;
}
```

מערך של מבנים - דוגמא

71

- נכתוב פונקציה המקבלת מערך של מלבנים, ומחזירה מצביע למלבן עם האלכסון הארוך ביותר.

- שם הפונקציה יהיה `MaxRect`

- הפרמטרים שהפונקציה תקבל יהיו:

- מערך של מלבנים, כלומר הכתובת של המלבן הראשון במערך.

- גודל המערך.

- הערך המוחזר מהפונקציה יהיה המצביע למלבן אשר האלכסון שלו הוא הארוך ביותר.

תכנון הפונקציה MaxRect

72

- הפונקציה תעבור על פני כל איברי המערך ותחשב לכל מלבן את אורך האלכסון שלו על פי נקודות הקיצון שלו.
- הפונקציה תשמור את האורך המקסימלי שנמצא עד עכשיו במשתנה `max` ואת אינדקס המלבן שבו הוא נמצא במשתנה `index`.
- אם האלכסון הנוכחי גדול יותר מהמקסימלי שהיה עד עכשיו, נעדכן את המקסימום ואת האינדקס.
- בסוף יוחזר מצביע באמצעות האינדקס ששמרנו.

מערך של מלבנים בזיכרון

73

המלבנים נשמרים ברצף בזיכרון

כל נקודה מכילה שני

שדות מטיפוס

double

double x
double y
double x
double y

כל מלבן מכיל שני

שדות מטיפוס נקודה

Point

Point p
Point q

Rect

Rect

Rect

Rect

Rect

Rect

Rect

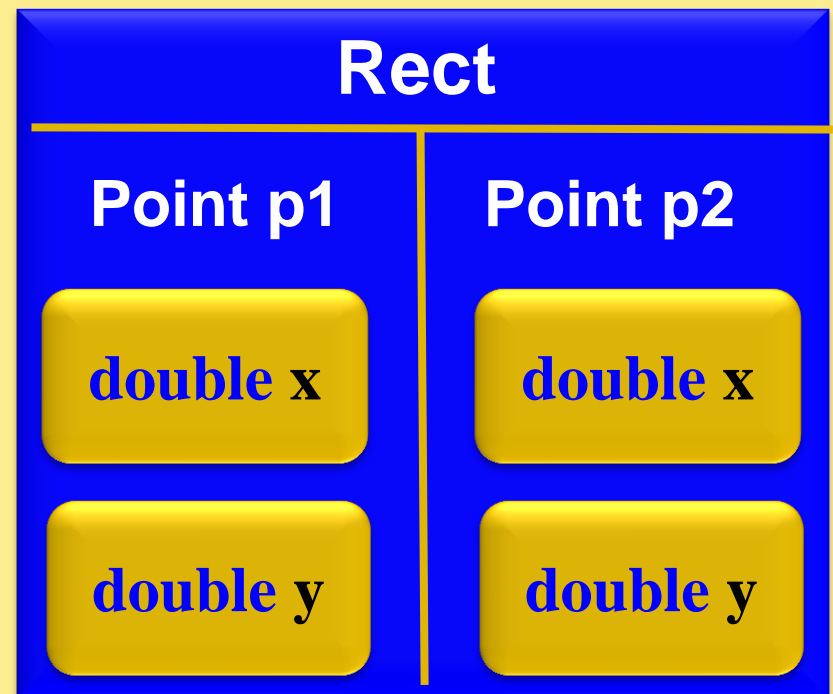
שימוש של מבנה בתוך מבנה – אלכסון של מלבן

74

- מבנה שהוגדר יכול לשמש אותנו גם כ-member של מבנה אחר. מבנה של נקודה יכול לשמש אותנו בבואנו להגדיר מבנה של משולש.

```
typedef struct point{  
    double x, y;  
}Point;
```

```
typedef struct rect {  
    point p1;  
    point p2;  
}Rect;
```

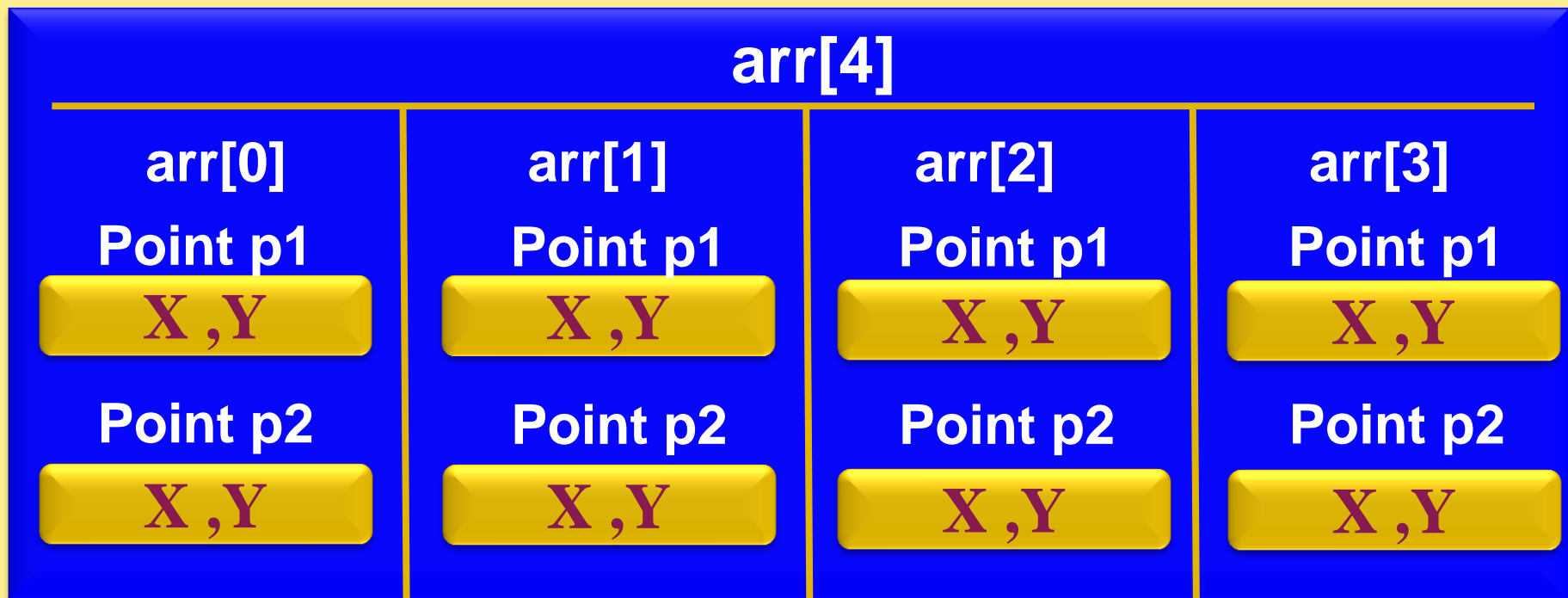


שימוש במערך של מבנים בתוך מבנה – אלכסון של מלבן

75

- נגדיר מערך של מבנה של נקודות אלכסוני מלבנים:

Rect arr[**SIZE**];



אתחול מערך של מבנים

76

- גם ניתן לאתחל מערך של מבנים בשורת ההגדרה, על-ידי כתיבת ערכי אתחול לכל מבנה לפי הסדר.
- חובה להקפיד על סדר האתחול, לכל שדה יש להקפיד להכניס ערך המתאים לטיפוס אותו שדה כפי שהוגדר במבנה.
- לדוגמא עבור מערך של מלבנים:

Rect arr[**SIZE**] = {{{ 3,4},{5,6}},{{ 4,7},{9,10}}}}

77

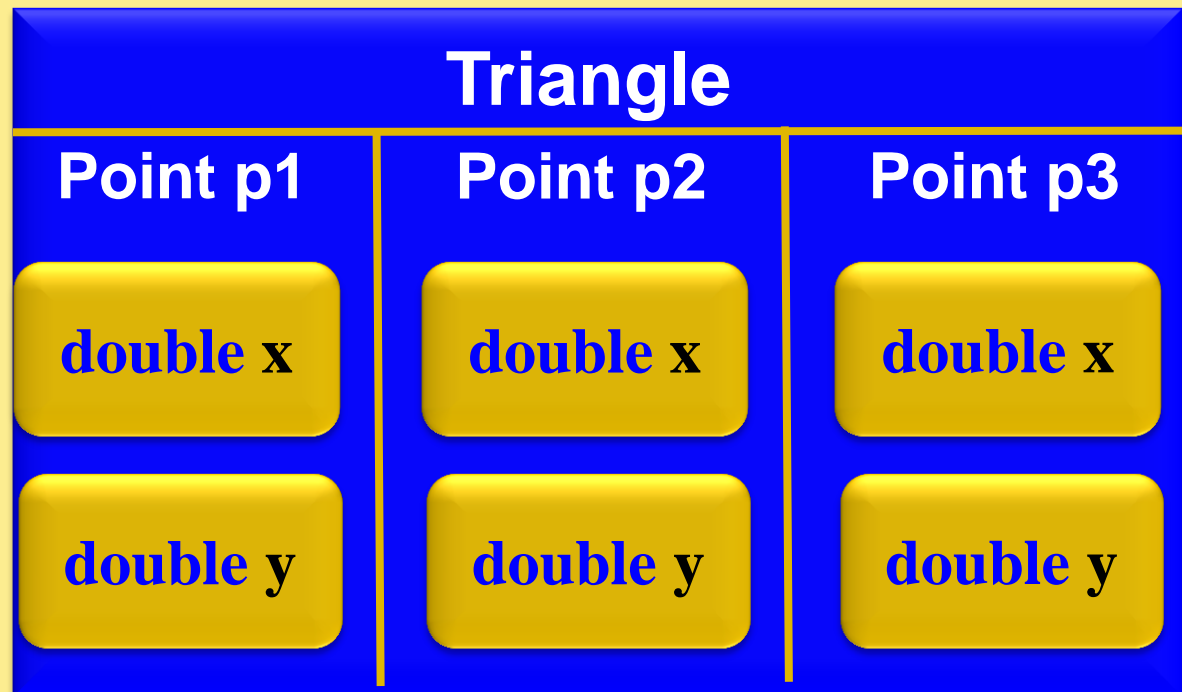
שאלות?

שימוש נוסף של מבנה בתוך מבנה –משולש

78

- מבנה שהוגדר יכול לשמש אותנו גם כ-member של מבנה אחר.
- מבנה של נקודה יכול לשמש אותנו בבואנו להגדיר מבנה של משולש.

```
typedef struct  
{  
    double x, y;  
} Point;  
typedef struct {  
    point p1;  
    point p2;  
    point p3;  
} Triangle;
```



שימוש נוסף של מבנה בתוך מבנה –משולש

79

- מבנה של נקודה יכול לשמש אותנו בבואנו להגדיר מבנה של משולש.
- t1 היא נקודה אחת, כלומר קודקוד המכילה בתוכה שני ערכים, אחד עבור x והשני עבור y.

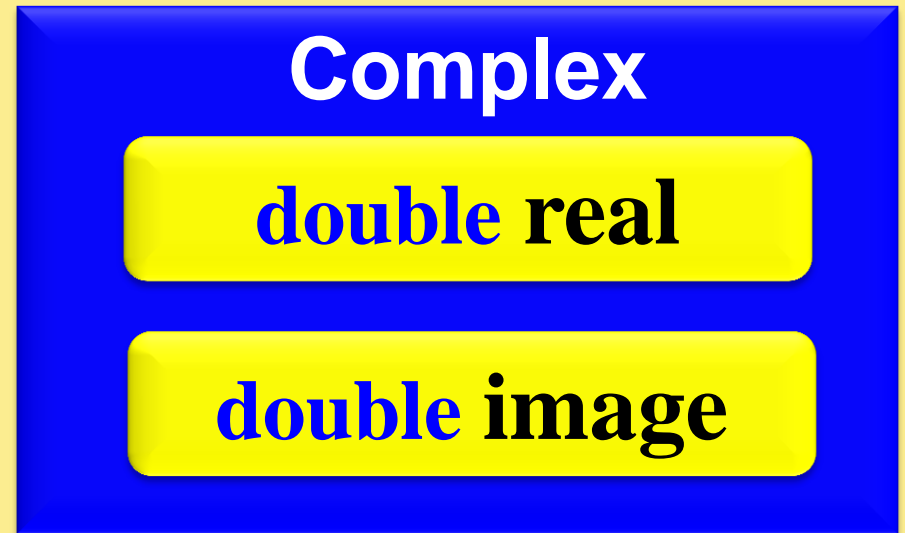
```
typedef struct{  
    point p1;  
    point p2;  
    point p3;  
} Triangle ;  
void main(){  
    Triangle t1;  
    t1.p2.x = 7.3;  
    t1.p2.y = 8.2;  
}
```

מבנה של מספר מורכב

80

- נגדיר מבנה של מספר מורכב:

```
typedef struct  
{  
    double real;  
    double image;  
} Complex;
```



מערך של מבנים של מספר מורכב

81

- נגדיר מערך של מבנה של מספר מורכב:

Complex arr[**SIZE**];

arr[4]			
arr[0]	arr[1]	arr[2]	arr[3]
real	real	real	real
5	2	7	1
image	image	image	image
7	1	2	8

מערכים ומבנים

82

- מאחר ומבנים מגדירים סוג חדש של משתנים הרי שכמו לסוגים רגילים נרצה ליצור מערכים עבור סוגים אלו.

Complex arr[SIZE]=
{{5,7},{2,1},{7,2},{1,8}}
arr[1].real = 9;
arr[1].image = 2;
arr[3].real = 8;
arr[3].image = 3;

arr		
real:	→	5
image:	→	7
real:	→	2
image:	→	1
real:	→	7
image:	→	2
real:	→	1
image:	→	8

מערך של מבנים של מספר מורכב

83

- נגדיר מערך של מבנה של מספר מורכב:

Complex arr[SIZE];

arr[4]			
arr[0]	arr[1]	arr[2]	arr[3]
real	real	real	real
5	9	7	8
image	image	image	image
7	2	2	3

מערכים ומבנים אחרי ביצוע הפקודות

84

Complex arr[SIZE]=

{{5,7},{2,1},{7,2},{1,8}}

arr[1].real = 2;

arr[1].image = 9;

arr[3].real = 3;

arr[3].image = 8;

arr[size]

real:

image:

real:

image:

real:

image:

real:

image:

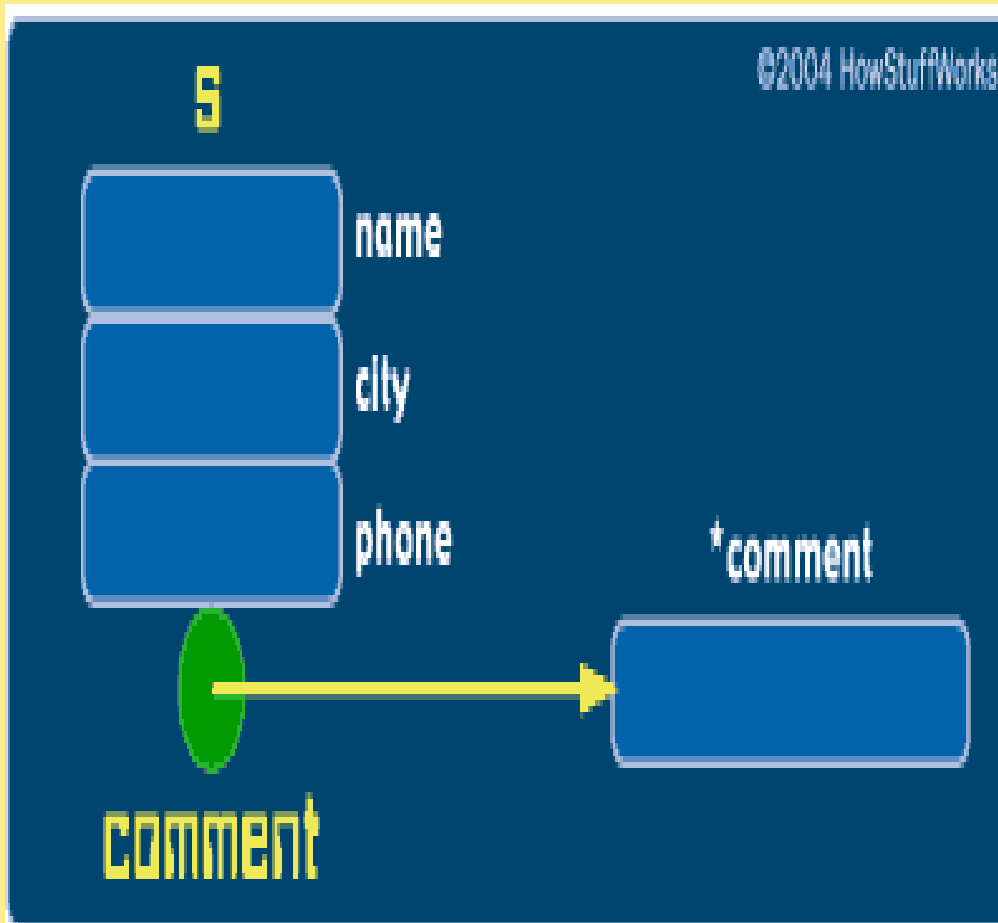
5	0
7	
2	1
9	
7	2
2	
3	3
8	

85

שאלות?

סיכום

86



- הגדרה ואתחול
- פעולות על מבנים
- מבנים ופונקציות
- מבנים ומצביעים
- מבנה בתוך מבנה
- מערכים של מבנים
- מבנים מורכבים

תרגיל מספר 1

87

- כתוב פונקציה המקבלת מערך של 5 מספרים מרוכבים.
- הפונקציה תבדוק את המספרים ותאתר את האיבר בעל הערך המוחלט המקסימלי.
- הפונקציה תחזיר את האינדקס, המיקום, של האיבר בעל הערך המוחלט המקסימלי.
- כתוב תוכנית המפעילה את הפונקציה, כולל פונקציית קליטת הנתונים ופונקציית הדפסת המערך.
- השלם המשימות באופן עצמאי.

פתרון

88

```
#include <stdio.h>
#include <math.h>
#include <string.h>
#define SIZE 5
typedef struct    // הגדרת מבנה
{
    double real;
    double image;
}Complex;
```


פתרון

89

```
int ScanComplex(Complex *arr)
{
    int i;
    printf("Enter %d peers of numbers:\n", SIZE);
    for(i=0; i< SIZE; i++)
    {
        printf("Enter numbers for point %d\n",i);
        scanf("%lf%lf",&arr[i].image,&arr[i].real);
    }
}
```

פתרון

90

```
int PrintComplex(Complex *arr)
{
    int i;
    printf("array %d numbers:\n", SIZE);
    for(i=0; i< SIZE; i++)
        printf("index=%d image=%.2lf real=%.2lf\n",
            i,arr[i].image,arr[i].real);
}
```

פתרון

91

```
int MaxAbs(Complex *arr)
{
    int i,maxIn; double maxAbs=-1;
    char status[SIZE];
    for(i=0;i< SIZE;i++)
    {
        if (maxAbs < abs(arr[i].image))
        {
            maxAbs = abs(arr[i].image);  maxIn = i;
            strcpy(status,"image");
        }
    }
}
```

פתרון

92

```
if (maxAbs < abs(arr[i].real))  
{  
    maxAbs = abs(arr[i].real);  
    maxIn = i;  
    strcpy(status,"real");  
}  
  
}  
  
printf("Max abs is %.2lf\n",maxAbs);  
printf("Max abs is in status: %s\n",status);  
return maxIn;  
}
```

פתרון

93

```
int main()
{
    Complex arr[SIZE];
    ScanComplex(arr);
    PrintComplex(arr);
    printf("Max value in index: %d\n",MaxAbs(arr));
    return 0;
}
```

94

שאלות?

מבנים – עוד דוגמא

95

- מבנים יכולים כאמור להכיל שדות מסוגים שונים.
- דוגמא למבנה מורכב המייצג נתוני סטודנט.
- במבנה שלושה שדות: מספר ת.ז. שם, ממוצע ציונים.

```
#define SIZE 20
typedef struct
{
    int id;
    char name[SIZE];
    double average;
}Student;
```

מבנים - סטודנט

96

- נוכל למשל להגדיר מערך של הסטודנטים בכיתה:

```
# define CLASS_SIZE 50
```

```
int main()
```

```
{
```

```
    Student class[CLASS_SIZE];
```

```
    ...
```

```
    ...
```

```
}
```

(נניח ש- **CLASS_SIZE** הוא קבוע

שמכיל את מס' הסטודנטים בכיתה - 50)

- נכתוב לדוגמא פונקציה שמדפיסה את שמות הסטודנטים המצטיינים ומחזירה את מספרם

דוגמא: פונקציה למציאת המצטיינים

97

```
int top_students(Student students[ ])
{
    int i, cnt = 0;
    for(i = 0; i < CLASS_SIZE; i++)
    {
        if (students[i].average > 90)
        {
            printf("%s\n", students[i].name);
            cnt++;
        }
    }
    return cnt;
}
```

כאמור מחזירים את מספר המצטיינים

תרגיל מספר 2

98

- כתבו פונקציה המקבלת שני מספרים מרוכבים אשר כל אחד מהם מכיל שני מספרים פנימיים.
- על הפונקציה לחבר את שני המספרים, כלומר את ארבעת המספרים הפנימיים שלה ולהחזיר את הסכום.
- כתבו פונקציה נוספת המחשבת את הערך המוחלט של המספר המרוכב (השורש של מכפלת מרחקי הנקודות שבו) ומחזירה את הערך הזה.

פתרון - כללי


99

```
double AbsComplex(Complex mis)
{
    return sqrt(mis.real * mis.real + mis.image * mis.image);
}

Complex Add(Complex mis1, Complex mis2)
{
    Complex sum;
    sum.real = mis1.real + mis2 .real;
    sum.image = mis1.image + mis2.image;
    return sum;
}

x = Add(x,y);
ab = AbsComplex(x);
```

העברה לפי ערך!



פתרון – שימוש במשתנים

100

```
double AbsComplex(Complex mis)
{
return sqrt(mis.real *mis.real + mis.image *mis.image);
}

Complex Add(Complex mis1, Complex mis2)
{
    Complex sum;
    sum.real = mis1.real + mis2 .real;
    sum.image = mis1.image + mis2.image;
    return sum;
}
```

פתרון – שימוש במצביעים

101

```
double AbsComplex(Complex* pa)
{
    return sqrt(pa->real*pa->real + pa->image*pa->image);
}

Complex Add(Complex *pmis1, Complex *pmis2)
{
    Complex sum;
    sum.real = pmis1->real + pmis2->real;
    sum.image = pmis1->image + pmis2->image;
    return sum;
}
```

מבנים - סיכום

102

- מטרת המבנים היא לאפשר למתכנת להגדיר טיפוסים משתנים חדשים שמתאימים ספציפית לבעיה מציאותית אותה התוכנית מיועדת לפתור.
- השימוש במבנים יחד עם מערכים ומצביעים מאפשר הגדרת מערכים מורכבים של מבני נתונים בעלי שדות פנימיים עם טיפוסים שונים ומורכבים כאחד.
- מטרת המילה **typedef** היא לאפשר למתכנת לתת שמות חדשים ובעלי משמעות לטיפוסים משתנים קיימים. באמצעות ההגדרה המקוצרת טיפס הנתונים, המבנה החדש, קל להגדרה ולכתיבה כי אין צורך בהוספת המילה **struct** לפניו.

מבנים - סיכום

103

- במידה ונכתוב תוכנית שמטרתה היא לחשב ערכים באמצעות מספרים מרוכבים $a+b$ הכי נוח שיהיה לנו טיפוס ייחודי כמו מספר ממשי (**double**) והטיפוס יוגדר עבור מספר מרוכב בלבד, השימוש במבנים בשפת C מאפשר הגדרה זו.
- אפשר להשתמש במבנים כמו בטיפוסים רגילים מבחינת הגדרת מערך של מבנים, מצביעים על מבנה, העברה לפונקציה. חשוב לזכור ששדות שהם מערכים מועתקים ולא נשמרים המקוריים.
- לא ניתן להשתמש בפעולות החשבון וההשוואה הרגילות על מבנים - יש להגדיר פונקציות עבורן בהתאם לצורך (רק פעולת ההשמה מוגדרת, ובה מתבצעת גם העתקת מערכים).

תרגיל כיתה

104

1. נתונה הגדרת מבנה טיל (Missile) הבאה:

```
#define SIZE 20
typedef struct missile{
    char model[SIZE];
    int range, quantity;
    float price;
}Missile;
```

המבנה מתאר מידע על טילים בסוללת טילים, כאשר:
model – דגם הטיל, range – טווח הטיל בקילומטרים, quantity –
כמות טילים השמישים בסוללה במצבור אחד, יש מצב בו יהיה מודל
טיל בכמה מחסנים ובכמויות שונות ו-price – מחיר כל טיל בודד.

תרגיל כיתה

105

ידוע כי מאגר הטילים בסוללה מחולק ל- 20 מצבורים תת קרקעיים.

עליך לכתוב פונקציה המקבלת כפרמטרים את מערך הטילים מטיפוס missile ואת גודלו וכן פרמטר המציין את דגם הטיל הנבדק.

הפונקציה תבצע חיפוש ותציג דוח לפי כל דגם טיל. הדוח מבוסס על המערך של מצבורי הטילים בסוללה ותציג בסיום החיפוש את כמות הטילים מכל דגם המבוקש שנמצאו בכל המצבורים והעלות של כמות טילים אלה. בסיום תציג את עלות כל הטילים בסוללה.

תרגיל כיתה

106

2. בקניון "מיני אריאלי" מנוהלות פעולות הקניה במערכת ממוחשבת. בסוף כל שבוע מוזרמים הנתונים של אותו השבוע למחשב המרכזי לשלב ניתוח והסקת המסקנות.

פרטי רכישה כוללים את הנתונים הבאים:

- יום בשבוע בו התבצעה הרכישה – מספר שלם בין 1-7
 - קוד החנות – מספר שלם בין 1-5
 - פדיון יומי – מספר ממשי המציין פדיון יומי לחנות
- הגדר את מבנה הנתונים שמאפשר לאחסן נתוני רכישות.

תרגיל כיתה

107

כתוב פונקציה:

המקבלת מערך גלובלי בגודל $DAYS \times SHOPS$ של מבני הנתונים מסעיף א' ומדפיסה את היום בשבוע בו סכום הכנסות הקניון היה הגבוה ביותר.

אם יש מספר ימים כאלה, הפונקציה תדפיס את כולם.

הפונקציה מחשבת, מוצאת ומחזירה את קוד החנות הרווחית ביותר בשבוע שחלף. ניתן להניח שקיימת רק חנות אחת כזו.

תרגיל כיתה

108

3. נתונה הגדרת מבנה Car הבאה:

```
#define SIZE 20
typedef struct car
{
    char model[SIZE];
    float price;
    int year;
}Car;
```

המתארת את הנתונים על מכונית בחברת מכירות כאשר:
model – שם הדגם, price – מחיר המכונית, year – שנת היצור.

תרגיל כיתה

109

עליך לכתוב פונקציה בשם `find_car` המחפשת ומדפיסה את נתוני כל המכוניות המתאימות לקונה פוטנציאלי.

הפונקציה מקבלת וקולטת את הפרמטרים הבאים:

מערך מבנים מסוג `Car` שמכיל מידע על כל המכוניות, הדגם המבוקש על ידי הלקוח, המחיר המירבי שהקונה מוכן לשלם, שנת יצור מינימלית.

הפונקציה תחפש במערך המכוניות את הדגם המבוקש בעל מחיר קטן או שווה למחיר המרבי ושנת יצור מאוחרת או שווה לשנת היצור המינימלית.

הפונקציה תחזיר את מספר המכוניות שפרטיהן הודפסו.

תרגיל כיתה

110

4. נתון המבנה Film בצורה הבאה:

```
#define SIZE 20
typedef struct film
{
    char title[SIZE], director[SIZE];
    int budget, year;
}Film;
```

המבנה מתאר מידע על סרט, כאשר:
– title – שם הסרט, director – שם הבמאי, budget –
תקציב הסרט, year – שנת יציאה לאור

תרגיל כיתה

111

נתון מערך allFilms בגודל N שמאחסן נתונים של N סרטים שונים.

עליך לכתוב פונקציה funFilm המקבלת את מערך המבנים allFilms מסוג Film בגודל N, שם במאי ושנת ההפצה. הפונקציה בודקת האם יש סרטים של אותו הבמאי עם תקציב מעל 25000000 ושנת ההפצה שלהם לאחר 2010.

אם כן, הפונקציה מדפיסה את נתוני הסרט: שם הסרט, התקציב ושנת היציאה לאור.

אם אף סרט לא מתאים, יש להדפיס הודעה בתוך הפונקציה. בנוסף לכך, הפונקציה מחשבת ומדפיסה את הממוצע תקציבי סרטים של אותו במאי ומחזירה ל-main את המספר הסרטים המתאימים.

תרגיל כיתה

112

5. נתון המבנה Student בצורה הבאה:

```
#define SIZE 20
typedef struct student
{
    char name[SIZE], department[SIZE+10];
    float grades[SIZE-10];
}Student;
```

המבנה מתאר מידע על סטודנט, כאשר:

name - מערך תווים שמאחסן שם של סטודנט במכללה,

department - שם המחלקה בה לומד הסטודנט, **grades** -

מעריך שמאחסן את ציוני הסטודנט ב- 10 קורסים אותם הוא לומד.

תרגיל כיתה

113

עליך לכתוב פונקציה **average** שמקבלת מערך בגודל **N** שמאחסן נתונים של **N** סטודנטים במכללה ומבצעת את המשימות הבאות:

תחשב את הממוצע של כל אחד מהסטודנטים הלומדים במחלקה "**Mechanical Eng**" ותדפיס את שם הסטודנט והממוצע שלו ולאחר מכן את מספר הסטודנטים הלומדים במחלקה זו.

תדפיס את שמות הסטודנטים במחלקה "**Mechanical Eng.**" בעלי ממוצע הגבוה ביותר והנמוך ביותר ואת הממוצעים שלהם. תחשב ותדפיס את הממוצע של כלל הסטודנטים במכללה בקורס חדו"א. ציון הקורס רשום במקום השני (בעל אינדקס 1) במערך הציונים.

תרגיל כיתה

114

6. נתונה הגדרת מבנה ספר בספרייה (Book) הבאה:

```
#define SIZE 20  
typedef struct book  
{  
    char name[30], author[20];  
    int copies, year;  
}Book;
```

המבנה מתאר מידע על ספרים בספריית השכונה, כאשר:

name – שם הספר, author – שם הסופר, copies – מספר העותקים של הספר בספרייה, year – שנת הוצאת הספר

תרגיל כיתה

115

כל פרטי הספרים בספרייה רשומים במערך בגודל 1000 המדמה את מאגר הספרים בספרייה, כאשר מספר התא במערך מסמן את מספר המדף בו מונחים עותקים של ספר מסוים.

עליך לכתוב פונקציה בשם `checkBook`, המקבלת כארגומנט מערך של מבנים מטיפוס `Book`, את גודל המערך ואת שם הספר המבוקש על ידי לקוח הספרייה. הפונקציה סורקת את מאגר הספרים הקיים במערך אחרי חיפוש הספר המבוקש.

אם הספר מצוי בספרייה הפונקציה מחזירה את מקום הספר, מספר המדף שהוא גם מספר האינדקס של הספר המבוקש במערך ומדפיסה את מספר העותקים של הספר הקיימים בספרייה. אם הספר אינו קיים בספרייה הפונקציה מחזירה 1-.

תרגיל כיתה

116

7. במתנ"ס שכונתי רשומים N ילדים בגילאים שונים לחוגים שלאחר צהריים. רשומה אחת מוגדרת ע"י המבנה הבא:

```
#define SIZE 20
```

```
typedef struct person
```

```
{
```

```
    char firstName[SIZE/2], lastName[SIZE];
```

```
    int ageMon, ageYear;
```

```
}Person;
```

שדות המבנה : firstName ו- lastName – שם ושם משפחה
של הילד ו- ageYear ו- ageMon – גיל של ילד: שנים ומספר חודשים.

תרגיל כיתה

117

עליך לכתוב פונקציה שמקבלת רשימת N ילדים במתנ"ס. הפונקציה תדפיס את הגיל של הילד הצעיר ביותר שמופיע ברשימה. (יש להתחשב גם במספר חודשים);

הפונקציה תחשב ותדפיס את הגיל הממוצע של הילדים שבמתנ"ס (יש להתחשב גם במספר חודשים);

הפונקציה תחשב ותדפיס את כל הנתונים של הילדים שגילם בטווח בין 10 ל-12 ותחזיר את מספרם (יש להתחשב גם במספר חודשים).

אם אין ילדים בגילאים אלה הפונקציה מדפיסה הודעה על כך.

תרגיל כיתה

118

8. כתוב תוכנית לניהול משרד שידוכים ממוחשב כשידוע שבמשרד קיימת כרטסת לקוחות הכוללת את הפרטים הבאים לכל לקוח:
- שם, מין, גיל ותחום התעניינות. כאשר מגיע לקוח חדש יש למצוא לו בני זוג מתאימים מבין לקוחות החברה, בן זוג מתאים פרושו:
- בן המין השני ללקוח החדש
 - הפרש הגילים בינו לבין הלקוח החדש לא עולה על 10 שנים
 - יש לו תחום התעניינות משותף עם הלקוח החדש.
- התוכנית צריכה לקלוט את הנתונים של כל הלקוחות ושל הלקוח החדש ולהדפיס רשימה של בני הזוג המתאימים ללקוח החדש או הודעה מתאימה במקרה שאין אף אחד מתאים ללקוח החדש.

תרגיל כיתה

119

9. נתונה הגדרת מבנה דירה (Apartment) הבאה:

```
#define SIZE 30
typedef struct apartment
{
    char address[SIZE];
    float rooms;
    double price;
}Apartment;
```

המתארת מידע על דירה למכירה במשרד תיווך, כאשר:
address – הכתובת של הדירה למכירה, rooms – מספר
החדרים בדירה, price – מחיר הדירה

תרגיל כיתה

120

עליך לכתוב פונקציה המקבלת כפרמטרים מערך של מבנים מסוג apart ואת גודל המערך.

כמו כן, הפונקציה מקבלת כפרמטרים את מספר החדרים הרצוי לקונה ואת המחיר המקסימלי שהקונה מוכן לשלם.

הפונקציה מדפיסה את כל המידע הקיים על כל הדירות שמחירן אינו עולה על המחיר המקסימלי ושמספר חדריהן הוא כמבוקש.

במידה ולא נמצאו במאגר דירות העונות על הקריטריונים הנדרשים, תודפס הודעה מתאימה.

הערך המוחזר הוא מספר הדירות שפרטיהן הודפסו.

תרגיל כיתה

121

10. בקופת חולים "הפועל" החליטו למחשב את נתוני הרופאים והפציינטים.

במערכת שהוקמה הוגדרו שני המבנים הבאים:

רופא

- מספר רשיון רופא (מספר שלם עד 8 ספרות)
- שם רופא (מחרוזת עד 50 תווים)
- התמחות (מחרוזת עד 50 תווים)
- וותק (מספר שלם)

תרגיל כיתה

122

פציינט

- מספר זיהוי (תעודת זהות, מספר שלם עד 9 ספרות)
- שם מלא (מחרוזת עד 50 תווים)
- רשימת הרופאים אצלם ביקר בשנה האחרונה (מערך של מספרים שלמים, שהם מספרי הרשיון של הרופאים אצלם ביקר)
- מספר הביקורים אצל רופאים בשנה האחרונה (מספר שלם, מייצג גם את אורך המערך הנ"ל)
- במערכת הוגדרו שני מערכים גלובליים:
מערך באורך 50 לייצוג נתוני הרופאים ומערך של 3000 נתוני הפציינטים.
- ניתן להניח כי המערכים מולאו בנתונים.

תרגיל כיתה

123

יש לבצע את המשימות הבאות:

- **להגדיר את מבני הנתונים הנ"ל.**
- **יש לכתוב פונקציה המחזירה את מספר הרישוי של הרופא בעל הוותק המקסימלי. הנח שקיים בדיוק אחד כזה.**
- **יש לכתוב פונקציה המדפיסה את שם הרופא שאצלו ביקרו הכי הרבה פציינטים בשנה האחרונה.**
- **אם יש מספר רופאים שלהם אותה כמות של מבקרים, הדפס את שמות כולם.**

שאלות?