הקצאות דינאמיות בשילוב מבנים

קרן כליף

ביחידה זו נלמד:

- דוגמאות משולבות למבנים והקצאות דינאמיות

הקצאת מערך של מבנים

אין שום שינוי פרט לטיפוס ב- calloc/malloc

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

struct Point
{
    int  x, y;
} typedef  point_t;

void main()
{
    int  size, i;
    point_t* points;

    printf("How many points? ");
    scanf("%d", &size);
    points = (point_t*)calloc(size, sizeof(point_t));
    if (!points)
    {
      printf("ERROR! Out of memory!\n");
      return;
    }

    printf("Points are: ");
    for (i=0 ; i < size ; i++)
      printf("(%d, %d) ", points[i].x, points[i].y);
    printf("\nPlease enter %d points: \n", size);
    for (i=0 ; i < size ; i++)
    {
      printf("Point #%d: ", i+1);
      scanf("%d %d", &points[i].x, &points[i].y);
    }
    printf("Points are: ");
    for (i=0 ; i < size ; i++)
      printf("(%d, %d) ", points[i].x, points[i].y);
    printf("\n");
    free(points)
}
```

```
C:\WINDOWS\system32\cmd.exe
How many points? 3
Points are: (0, 0) (0, 0) (0, 0)
Please enter 3 points:
Point #1: 1 1
Point #2: 2 2
Point #3: 3 3
Points are: (1, 1) (2, 2) (3, 3)
Press any key to continue . . .
```

דרכים שונות להגדרת מערך

1. מערך של Student בגודל הידוע בזמן קומפילציה

```
student_t  arr[3];
```

2. מערך של Student בגודל **שאינו** ידוע בזמן קומפילציה

```
scanf("%d", &size)
student_t*  arr = (student_t*)malloc(sizeof(student_t)*size);
```

3. מערך של מצביעים ל- Student בגודל הידוע בזמן קומפילציה

```
;student_t*  arr[3]
```

4. מערך של מצביעים ל- Student בגודל **שאינו** ידוע בזמן קומפילציה

```
scanf("%d", &size)
student_t**  arr = (student_t**)malloc(sizeof(student_t*)*size);
```

דרכים להגדרת מערך (1)

- מערך של Student בגודל הידוע בזמן קומפילציה

student_t arr[3];

  - במקרה זה כל איברי המערך נמצאים על ה- stack

student[]: arr[0]: name		
arr[0]: id		
arr[1]: name		
arr[1]: id		
arr[2]: name		
arr[2]: id		

- מימוש זה בזבזני במידה ולא נשתמש בכל איברי המערך
- יעיל מבחינת ביצועים (אין הקצאות דינאמיות)

דרכים להגדרת מערך (2)

- מערך של Student בגודל **שאינו** ידוע בזמן קומפילציה

```
scanf("%d", &size)
student_t* arr = (student_t*)malloc(sizeof(student_t)*size);
```

- במקרה זה רק כתובת ההתחלה של המערך נמצאת על ה-stack, בעוד המבנים עצמם נמצאים על ה- heap

student_t*: arr	**2220**	**1000**
int: size	**2**	**1004**

student_t: name	**"yoyo"**	**2220**
id	**111**	**2230**
student_t: name	**"gogo"**	**2236**
id	**222**	**2246**

- גם מימוש זה בזבזני במידה ולא נשתמש בכל איברי המערך
- יעיל מבחינת ביצועים (יש רק הקצאה אחת)

דרכים להגדרת מערך (3)

- מערך של מצביעים ל- Student בגודל הידוע בזמן קומפילציה

```
student_t* arr[3];
```

- במקרה זה יש מערך של 3 כתובות על ה- stack, והמבנים עצמם יוקצו דינאמית על ה- heap בעת הצורך, או יצביעו למבנים קיימים

student_t*[]: arr	2200	1000
	5400	
	NULL	

student_t: name	"yoyo"	2200
id	111	
student_t: name	"gogo"	5400
id	222	

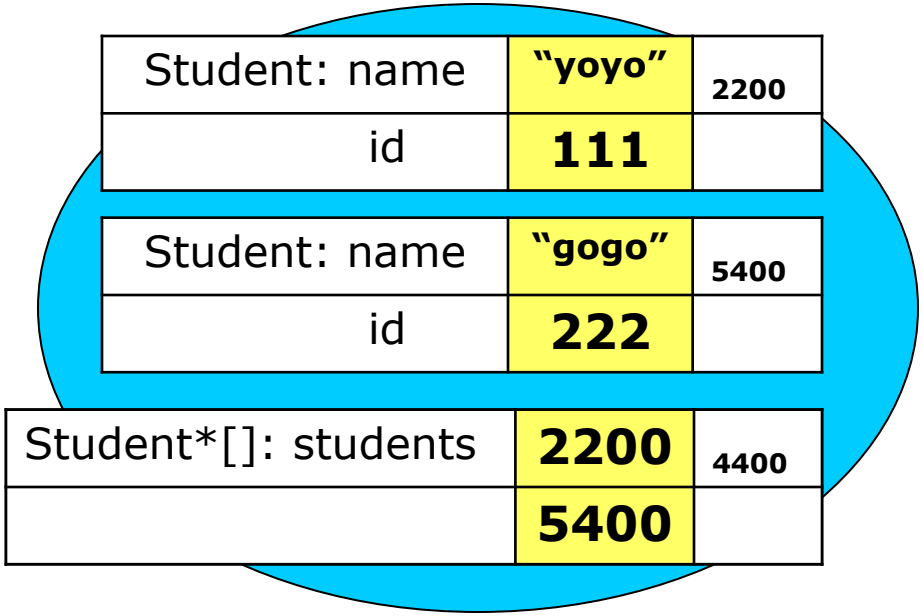
- מימוש זה אופטימלי מבחינת מקום, כלומר נקצה מקום לנתוני המבנה רק בעת הצורך. פחות יעיל מבחינת ביצועים (כל אחד מהאיברים מוקצה דינאמית)

דרכים להגדרת מערך (4)

- מערך של מצביעים ל- Student בגודל **שאינו** ידוע בזמן קומפילציה

```
scanf("%d", &size);
student_t**  arr = (student_t**)malloc(sizeof(student_t*)*size);
```

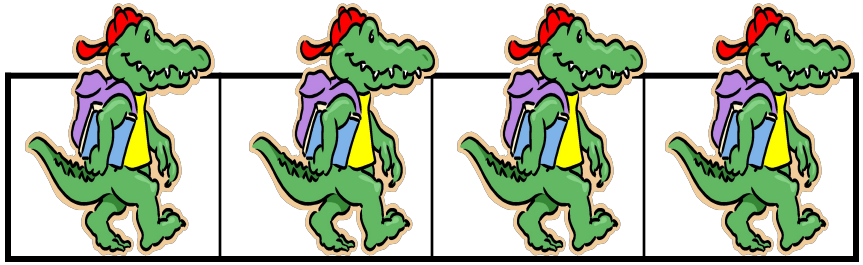
- על ה- stack תהיה רק כתובת ההתחלה של מערך הכתובות
- מערך הכתובות יוקצה על ה- heap שכן גודלו אינו ידוע בזמן קומפילציה



Student**: arr	**4400**	**1000**
int: size	**2**	**1004**

מערך מבנים בתוך מבנה

- בדוגמא הבאה יש לנו את המבנה "כיתה" שמכיל מערך של MAX_STUDNETS סטודנטים
- בכל איבר במערך יהיו נתונים של סטודנט
- כלומר, גודל המערך קבוע - כמות הסטודנטים המקסימלית בכל כיתה זהה



דוגמא – כיתה עם סטודנטים

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

#define MAX_STUDENTS 10

struct Student
{
    char name[10];
    int id;
} typedef student_t;

struct Class
{
    char  teacherName[10];
    int   registeredStudents;
    student_t  students[MAX_STUDENTS];
}typedef class_t;
```

```
Enter name and id for each student:
Enter another student? y
Student #1: gogo 111
Enter another student? y
Student #2: momo 222
Enter another student? y
Student #3: yoyo 333
Enter another student? n
The teacher is Keren and the 3 students are:
  1- Name: gogo Id: 111
  2- Name: momo Id: 222
  3- Name: yoyo Id: 333
Press any key to continue . . .
```

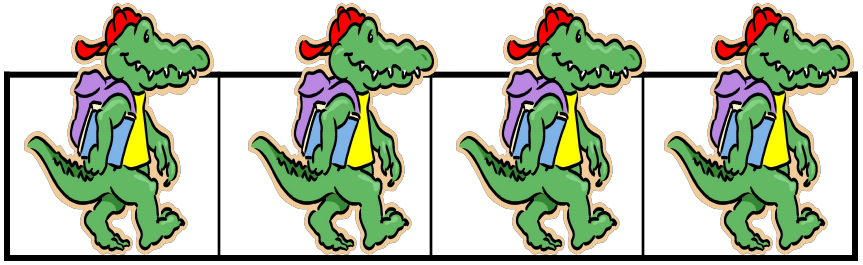
```
void printClass(class_t c)
{
    int i;
    printf("The teacher is %s and the %d students are:\n",
      c.teacherName,  c. registeredStudents);
    for (i=0 ; i < c. registeredStudents ; i++)
        printf("  %d- Name: %s\tId: %d\n",
        i+1, c.students[i].name,  c.students[i].id);
}

void main()
{
    class_t   c = {"Keren", 0};
    char      answer;
    int       fContinue = 1;

    printf("Enter name and id for each student:\n");
    do
    {
        printf("Enter another student? ");
        flushall();
        scanf("%c", &answer);
        if (answer == 'n' || answer == 'N')
        fContinue = 0;
        else
        {
        printf("Student #%d: ", c.registeredStudents+1);
        scanf("%s %d",
                c.students[c.registeredStudents].name,
                &c.students[c.registeredStudents].id);
        c.registeredStudents++;
        }
    } while (fContinue &&
            c.registeredStudents < MAX_STUDENTS);

    printClass(c);
}  // main
```

החיסרונות כאשר גודל המערך קבוע

- לא ניתן להגדיר כיתות בהן המספר המקסימלי של הסטודנטים שונה
  - למשל עבור קורסים עם קבוצות לימוד קטנות, או עבור שיעור שהוא תרגול
- יתכן ו- numOfRegistered קטן משמעותית מ- MAX_STUDENTS ואז יש ביזבוז רב של מקום
  - מבנה תופס יחסית הרבה מקום בזיכרון כי הוא מכיל אוסף של שדות

הקצאת מערך מבנים בתוך מבנה

- בדוגמא הבאה יש לנו את המבנה "כיתה" שיכול להכיל מערך של תלמידים
- בכל איבר במערך יהיו נתונים של סטודנט
- **מספר התלמידים המקסימלי אינו ידוע מראש וניתן ע"י המשתמש בזמן ריצה**
  - לכן מערך התלמידים שבתוך המבנה "כיתה" מוקצה דינאמית
- דוגמא, המשתמש בחר מערך בגודל 4, בכל איבר יהיו נתוני סטודנט



דוגמא – הקצאת סטודנטים בכיתה

(הקוד בלבד, כך שאפשר לראות אותו ☺)

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

struct Student
{
    char name[10];
    int id;
} typedef student_t;

struct Class
{
    char teacherName[10];
    int numOfStudents;
    student_t* students;
}typedef class_t;

void main()
{
    int i;
    class_t c = {"Keren"};
    printf("How many students? ");
    scanf("%d", &c.numOfStudents);
    c.students = (student_t*)calloc(c.numOfStudents ,
                    sizeof(student_t));
    // check if allocation succeeded..

    printf("Enter name and id for each student:\n");
    for (i=0 ; i < c.numOfStudents ; i++)
    {
      printf("Student #%d: ", i+1);
      scanf("%s %d", c.students[i].name,
                        &c.students[i].id);
    }

    printClass(c);
    free(c.students);
}
```

```
void printClass(class_t c)
{
    int i;
    printf("The teacher is %s and the %d students are:\n",
                c.teacherName,
                c.numOfStudents);
    for (i=0 ; i < c.numOfStudents ; i++)
      printf("  %d- Name: %s\tId: %d\n",
                i+1, c.students[i].name,
                c.students[i].id);
}
```

```
How many students? 2
Enter name and id for each student:
Student #1: yoyo 1111
Student #2: momo 2222
The teacher is Keren and the 2 students are
  1- Name: yoyo Id: 1111
  2- Name: momo Id: 2222
Press any key to continue . . .
```

דוגמא – הקצאת סטודנטים בכיתה

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

st
{

}

st
{

student_t* s
}typedef class_t:
```

int:i	???	3000
class_t: char[10]:c.teacherName	"keren"	3004
int: c.numOfStudents	2	
student_t*:c.students	2220	

הזיכרון של printClass

student_t: name	"yoyo"	2220
id	111	2230
student_t: name	"gogo"	2236
id	222	2246

זיכרון ה- heap

int:i	???	1000
class_t: char[10]:c.teacherName	"keren"	1004
int: c.numOfStudents	2	
student_t*:c.students	2220	

הזיכרון של ה- main

```
void printClass(class_t c)
{
    int i;
    printf("The teacher is %s and the %d students are:\n",
                c.teacherName,
                c.numOfStudents);
    for (i=0 ; i < c.numOfStudents ; i++)
      printf("  %d- Name: %s\tId: %d\n",
                i+1, c.students[i].name,
                c.students[i].id);
}

void main()
{
    int i;
    class_t c = { "keren" };
    printf("How many students? ");
    scanf("%d", &c.numOfStudents);
    c.students = (student_t*)calloc(c.numOfStudents ,
                sizeof(student_t));
    // check if allocation succeeded..

    printf("Enter name and id for each student:\n");
    for (i=0 ; i < c.numOfStudents ; i++)
    {
      printf("Student #%d: ", i+1);
      scanf("%s %d", c.students[i].name,
                &c.students[i].id);
    }

    printClass(c);
    free(c.students);
}
```

שינוי שם המורה בתוך הפונקציה לא ישנה את הנתון המקורי, אבל שינוי ערכים במערך הסטודנטים כן ישתנו.

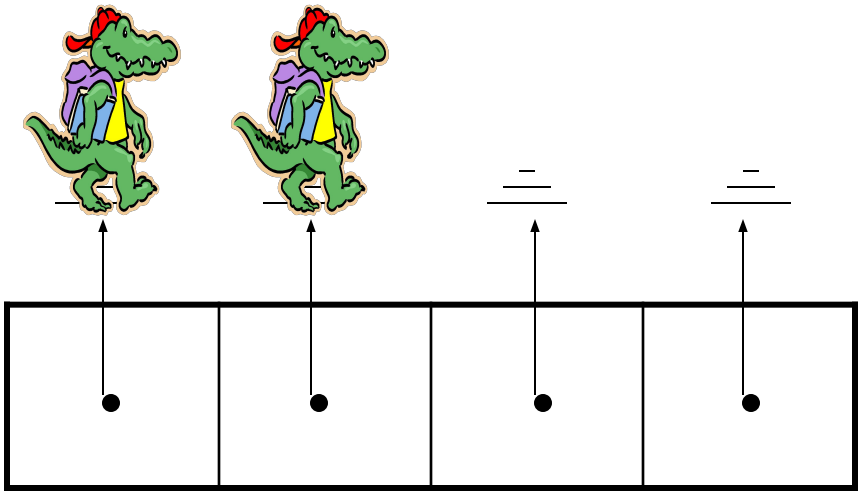
אבל מה אם נרצה להקצות מערך ולהשתמש רק בחלק מהאיברים?

- החיסרון במימוש המחזיק מערך של סטודנטים, בין אם הוקצה דינאמית או לא, הוא כאשר לא מנצלים את כל איברי המערך, ואז יש ביזבוז זיכרון

- לכן נקצה **מערך של מצביעים למבנים** בגודל המקסימלי שאנחנו רוצים, וכל איבר שיהיה בשימוש יצביע למבנה שיוקצה דינאמית

הקצאת מערך של **מצביעים למבנים** בתוך מבנה

- בדוגמא הבאה יש לנו את המבנה "כיתה" שיכול להכיל אוסף של תלמידים
- מספר הסטודנטים המקסימלי **ידוע מראש** ויש מערך של **מצביעים** ל"סטודנט"
- בתחילה רשומים לכיתה 0 סטודנטים, וכל פעם נוסיף סטודנט נוסף לכיתה
- כל איבר יהיה מצביע ל"סטודנט". כל עוד לא נרשם סטודנט המצביע הוא NULL
  - דוגמא: כיתה שיכולים להיות בה מקסימום 4 סטודנטים
    - לאחר רישום סטודנט
    - לאחר רישום סטודנט



דוגמא – רישום סטודנטים לכיתה

(הקוד בלבד, כך שאפשר לראות אותו ☺)

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

#define MAX_STUDENTS  2

struct Student
{
    char   name[10];
    int    id;
} typedef student_t;

struct Class
{
    char   teacherName[10];
    int    registeredStudents;
    student_t* students[MAX_STUDENTS];
}typedef class_t;

void printClass(class_t c)
{
    int i;
    printf("The teacher is %s and the %d students are:\n", c.teacherName, c.registeredStudents);
    for (i=0 ; i < c.registeredStudents ; i++)
      printf("  %d- Name: %s\tId: %d\n", i+1, c.students[i]->name, c.students[i]->id);
}
```

מערך של מצביעים

```
void main()
{
    int i, fExit=0;
    char answer;
    class_t c = {"Keren", 0};

    do {
      if (c.registeredStudents == MAX_STUDENTS)
      {
            printf("Class is full!\n");
      break;
      }
      printf("Register a student (Y/N)? ");
      flushall();
      scanf("%c", &answer);
      if (answer == 'N')
          fExit = 1;
      else
      {
            c.students[c.registeredStudents] =
                                    (student_t*)calloc(1, sizeof(studen
            printf("Enter name and id: ");
      scanf("%s %d",
                    c.students[c.registeredStudents]->name,
                &c.students[c.registeredStudents]->id);
      c.registeredStudents++;
      }
    } while (fExit==0);
    printClass(c);
    for (i=0 ; i < c.registeredStudents ; i++)
      free(c.students[i]);
}
```

בדיקה אם יש מקום לסטודנט נוסף

פניה לאיבר הפנוי הבא בעזרת registeredStudents

פניה לשדות איברי המערך בעזרת -> (כי הם מצביעים)

דוגמא – רישום סטודנטים לכיתה

class_t: char[10]:c.teacherName	"Keren"	1000
int: c.registeredStudents	2	
student_t*[] :c.students	2200	
	5300	

הזיכרון **החלקי** של ה- main

student_t: name	"momo"	2200
id	1111	
student_t: name	"gogo"	5300
id	2222	

זיכרון ה- heap

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

#define MAX_STUDENTS 2

struct Stu
{
    int
    student_t                    ENTS];
}typedef class_t;

void printClass(class_t c)
{
    int i;
    printf("The teacher is %s and the %d students are:\n", c.teacherName, c.registeredStudents);
    for (i=0 ; i < c.registeredStudents ; i++)
      printf("  %d- Name: %s\tId: %d\n", i+1, c.students[i]->name, c.students[i]->id);
}
```

```
void main()
{
    int i, fExit=0;
    char answer;
    class_t c = {"Keren", 0};

    do {
```

C:\WINDOWS\system32\cmd.exe

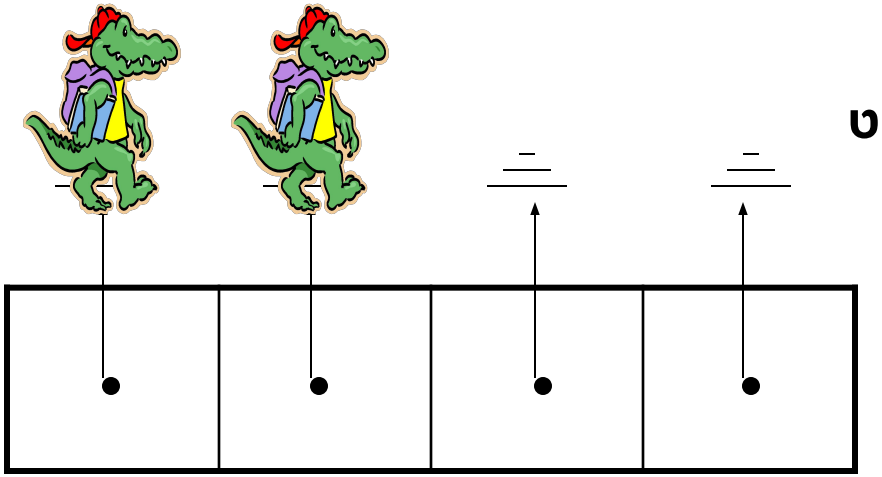
```
Register a student (Y/N)? Y
Enter name and id: momo 1111
Register a student (Y/N)? Y
Enter name and id: gogo 2222
Class is full!
The teacher is Keren and the 2 students are:
  1- Name: momo Id: 1111
  2- Name: gogo Id: 2222
Press any key to continue . . .
```

```
        else
        {
            c.students[c.registeredStudents] =
                                (student_t*)calloc(1, sizeof(student_t));
            printf("Enter name and id: ");
            nf("%s %d",
                    c.students[c.registeredStudents]->name,
                &c.students[c.registeredStudents]->id);
        c.registeredStudents++;
        }
    } while (fExit==0);
    printClass(c);
    for ( i=0 ; i < c.registeredStudents ; i++ )
      free(c.students[i]);
}
```

i=0

הקצאת מערך של מצביעים למבנים בתוך מבנה

- בדוגמא הבאה יש לנו את המבנה "כיתה" שיכול להכיל אוסף של תלמידים
- מספר הסטודנטים המקסימלי **אינו** ידוע מראש וניתן ע"י המשתמש בזמן ריצה
- בתחילה רשומים לכיתה 0 סטודנטים, וכל פעם נוסיף סטודנט נוסף לכיתה
- בכל איבר יהיה מצביע ל"סטודנט". כל עוד לא נרשם סטודנט המצביע הוא NULL
  - דוגמא: כיתה **שהמשתמש החליט** שיכולים להיות בה מקסימום 4 סטודנטים:
    - לאחר רישום סטודנט
    - לאחר רישום סטודנט



דוגמא – רישום סטודנטים לכיתה דינאמית (1)

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

struct Student
{
    char name[10];
    int id;
} typedef student_t;

struct Class
{
    char teacherName[10];
    int numOfStudents;
    int registeredStudents;
    student_t** students;
}typedef class_t;

void printClass(class_t c)
{
    int i;
    printf("The teacher is %s and the %d students are:\n",
            c.teacherName, c.registeredStudents);
    for (i=0 ; i < c.registeredStudents ; i++)
      printf("  %d- Name: %s\tId: %d\n",
            i+1, c.students[i]->name, c.students[i]->id);
}
```

מערך בגודל שאינו ידוע עדיין ושכל איבר בו יהיה מצביע:

- כוכבית אחת כי זהו מערך שמוצקה דינאמית, כדי להכיל את כתובת תחילת המערך
- כוכבית שניה כי כל איבר במערך הוא כתובת

דוגמא – רישום סטודנטים לכיתה דינאמית (2)

```
void main()
{
    int i, fExit=0;
    char answer;
    class_t c = {"Keren", 0};
    printf("How many max students? ");
    scanf("%d", &c.numOfStudents);
    c.students = (student_t**)malloc(sizeof(student_t*)*c.numOfStudents);
    // check if allocation succeeded

    do {
        if (c.registeredStudents == c.numOfStudents)
        {
      printf("Class is full!\n");
      break;
        }
        printf("Register a student (Y/N)? ");
        flushall();
        scanf("%c", &answer);
        if (answer == 'N')
            fExit = 1;
        else
        {
            c.students[c.registeredStudents] =
                        (student_t*)calloc(1, sizeof(student_t));
            printf("Enter name and id: ");
     scanf("%s %d", c.students[c.registeredStudents]->name,
                    &c.students[c.registeredStudents]->id);
     c.registeredStudents++;
        }
    } while (fExit==0);
    printClass(c);
    for (i=0 ; i < c.registeredStudents ; i++)
      free(c.students[i]);
    free(c.students);
}
```

קבלת מספר הסטודנטים המקסימלי מהמשתמש

הקצאת מערך של מצביעים לסטודנט

שיחרור כל אחד מאיברי המערך

שיחרור מערך המצביעים שגם הוקצה דינאמית
("כמו בדוגמא "מערך של מערכים)

השוואה בזכרון בין מערך מבנים למערך מצביעים

- נניח כי מבנה Student תופס 16 בתים בזכרון
- נניח כי יש כיתה עם פוטנציאל ל- 100 סטודנטים, אבל בפועל רשומים רק 40

מערך מצביעים	מערך מבנים	

ולסיום..

http://rlv.zcache.com/programmer_joke_superior_intelligence_ornament-r09ddf4dc720c4e978a03668525a88_x7s2y_8byvr_512.jpg

ביחידה זו למדנו:

- דוגמאות משולבות למבנים והקצאות דינאמיות

תרגיל 1:

- הגדר את המבנה Point אשר נתוניו הם x ו- y
- הגדר את המבנה Polygon אשר נתוניו הם מספר הקודקודים שלו וכן מערך לקודקודים

- הגדר ב- main 3 נקודות ואתחל את ערכיהן
- הגדר פוליגון ושאל את המשתמש לכמות קודקודיו
- 3 הקודקודים הראשונים יהיו 3 הנקודות שהוגדרו מקודם, ושאר הקודקודים יוגרלו
- הדפס את נתוני הפוליגון

תרגיל 2:

- הגדר את המבנה Person ששדותיו הם שם (הגודל אינו מוגבל) ות.ז.
- הגדר את המבנה Family שנתוניו הם אבא ואמא (מטיפוס Person), וכן מערך של 10 מצביעים ל-Person שייצג את הכמות המקסימלית של ילדים במשפחה
- כתוב פונקציות לקליטת הנתונים
- כתוב פונקציות להדפסת הנתונים
- כתוב main הבודק את התוכנית

- יש להקפיד על מודולריות!

תרגיל 3 (1):

- כתוב את המבנה Friend המכיל שם ואת המרחק ממנו הוא גר ממני (בק"מ)

- את אוסף החברים שלי נפריד לקבוצות לפי המרחק שבו הם גרים ממני:
  - חברים שניתן ללכת אליהם ברגל: גרים עד 2 ק"מ ממני
  - חברים שניתן לרכוב אליהם באופניים: גרים עד 5 ק"מ ממני
  - חברים שצריך לנסוע אליהם באוטו: גרים במרחק הגדול מ- 5 ק"מ ממני

תרגיל 3 (2):

- עבור מערך החברים הבא:

{toto", 2"}	{"koko", 4}	{yoyo", 1"}	{momo", 6"}	{"gogo", 3}

- נייצר את 3 המערכים הבאים:

NULL		NULL			NULL		

חברים שניתן ללכת אליהם

חברים שניתן לרכוב אליהם

חברים שצריך לנהוג אליהם

- ולבסוף נחזיר מערך שיכיל את הכתובות של מערכים אלו:

תרגיל 3 (3):

- כתוב פונקציה המקבלת מערך של חברים וגודלו
- הפונקציה תייצר מערך בגודל 3, שכל איבר בו יהיה כתובת של מערך של מצביעים לחברים לפי ההגדרות הנ"ל
- הפונקציה תמלא את המערכים המתאימים ותחזיר מערך זה
- שימו לב: מערכי החברים מכילים רק מצביעים לאיברים מהמערך המקורי שהתקבל כפרמטר (אין סיבה לשכפל את הנתונים).
- כדי לציין סיומו של מערך חברים שנוצר בפונקציה, יש לשים NULL באיבר האחרון

תרגיל 3 (4):

- הגדר ב- main מערך שאיבריו מטיפוס Friend, ושלח אותו לפונקציה שהגדרת
- הדפס את החברים לפי החלוקה לקבוצות