

חומרים שהוכנו על-ידי משתתפי קורס מורים מובילים תשע"ד

ניתן להשתמש בחומרים לצורך הוראה בלבד.

לא ניתן לפרסם את החומרים או לעשות בהם כל שימוש מסחרי

ללא קבלת אישור מראש מצוות הפיתוח

מערך שיעור בנושא: מערך של עצמים

כתיבה ועריכה:

ולרי פקר

הרציונל של מערך השיעור: המטרה העיקרית של שיעור זה היא פיתוח הבנה וצורך בשימוש אוסף עצמים. השיעור בנוי כפעילות קבוצתית ודיון בכיתה.

מטרות השיעור:

- התלמידים מכירים משיעורים קודמים בעיות אלגוריתמיות שונות עם עצמים שונים ומגוונים. הבעיות היו שונות זו מזו בכמות העצמים שהשתמשו בהן. למשל, הזזה "במקביל" על המסך של שלושה צבים.

- התלמידים מכירים שימוש במערכים של טיפוסים נתונים בסיסיים.

- התלמידים מכירים בעיות אשר בפתרון נחוץ מערך ובעיות אשר ניתן לפתור גם ללא מערך.

- כאשר למטרת פתרון הבעיה יש לשמור מספר גדול של נתונים נחוץ מערך.

- כאשר לא ידוע מראש מספר הנתונים והוא תלוי בקלט לתוכנית נחוץ מערך.

- ניתן להגדיר מערך מכל טיפוס שהוא, בפרט מערך של עצמים ממחלקות שהגדרנו אנחנו או מחלקות מוכנות מראש למשל, מערך עצמים מסוג Turtle.

- כל מה שתלמידים מכירים על מערכים של טיפוסים בסיסיים מתאים גם למערכים של עצמים. למשל, הזזה "במקביל" על המסך של מספר גדול של צבים. הצבים ניתן לייצוג כמערך עצמים מסוג Turtle.

שיעור זה מתאים לאחר שהתלמידים למדו: עבודה עם עצמים גרפיים מוכנים, שימוש בעצם מטיפוס Turtle, ביצוע מותנה if, ביצוע חוזר לולאות for ו-while, עבודה עם עצם Random, מערכים של טיפוסים נתונים בסיסיים, מושגי יסוד בעבודה עם מערכים: מציין (אינדקס), אורך (Length), גישה [], חישוב ערכי קיצון במערך (מינימום ומקסימום).

נקודות עיקריות בשיעור הקודם: שימוש במערכים של טיפוסים נתונים בסיסיים וחישוב ערכי קיצון במערך (מינימום ומקסימום).

נקודות עיקריות בשיעור הנוכחי:

- בניית מערך עצמים (השימוש בפעולה new).

- איתחול מערך עצמים (יצירת איברי המערך באמצעות פעולה new).

- שימוש בלולאה כדי לגרום לכמה עצמים במערך לבצע אותה פעולה או פעולה דומה.

שיעורי הבית שניתנו לקראת השיעור הנוכחי: תרגילים בנושא חישוב ערכי קיצון במערך.

קשיים צפויים במימוש השיעור:

- הבנה שהגדרת מערך מטיפוס מחלקה מסוימת נעשה כהגדרת כל מערך מטיפוס אחר.

- הבנה שמערך של עצמים הוא למעשה מערך של הפניות לעצמים.

- איתחול מערך של עצמים – בניית עצם מסוג מחלקה בכל תא המערך.

- עיבוד פשוט של מערך העצמים – פניה לאיבר המערך כפניה לעצם.

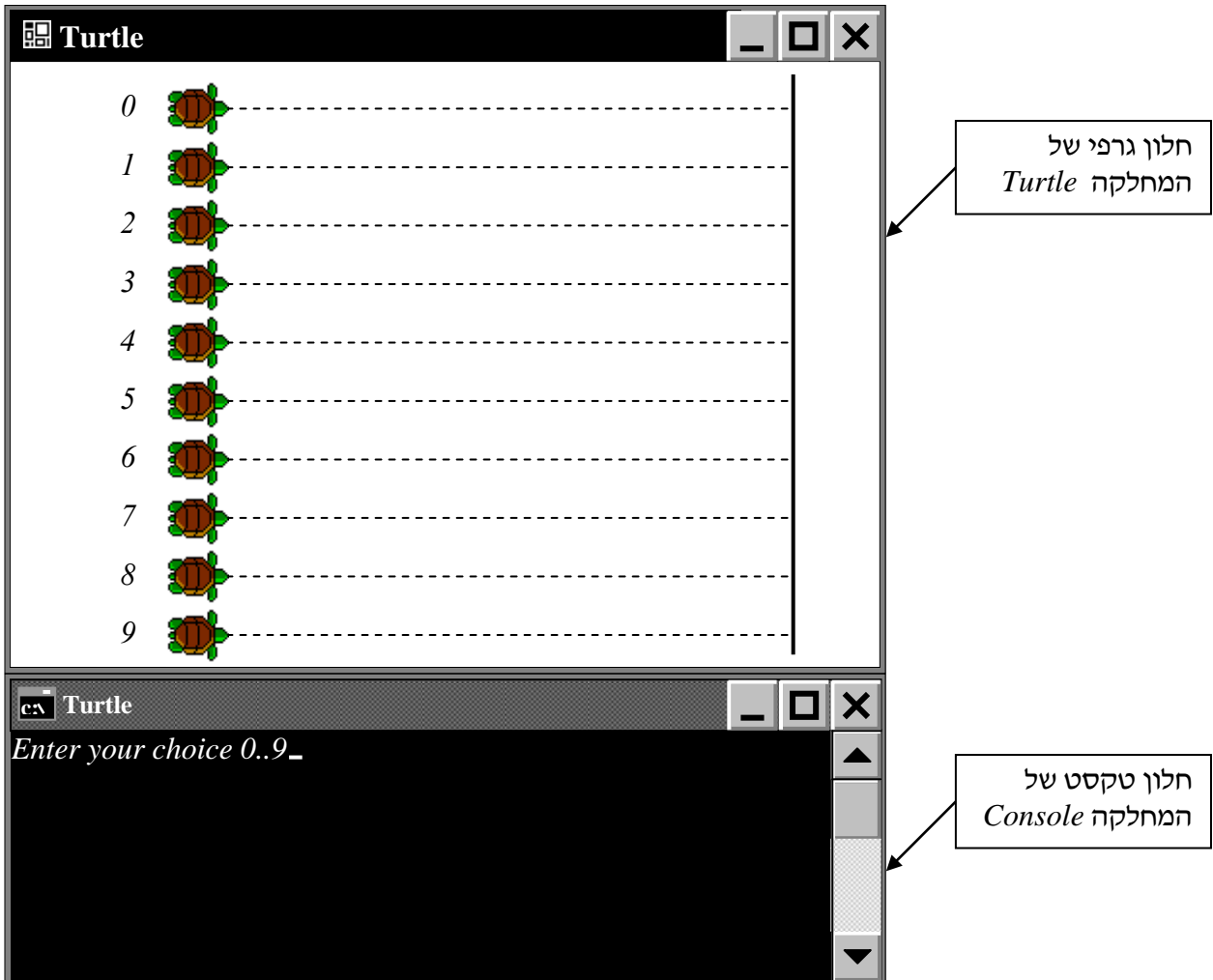
תיאור מהלך השיעור (45 דקות):

התלמידים מקבלים משימה:
משחק המדמה מרוץ של 10 צבים עם הימורים (המשתמש מהמר על מספר הצב שיגיע ראשון לקו הסיום).

המטרה היא להגיע לקצה המסלול בצד הימין של המסך. הצב שיגיע ראשון לקצה המסלול ינצח. בהתחלה 10 הצבים עומדים בשורה. מטרת המשחק – להגיע ראשון אל קו הסיום.

פיתוח משימה בשלבים:

- שלב 1 – בניית ממשק:



לצורך הדמיית מרוץ של 10 צבים נשתמש במערך עצמים מסוג Turtle:

- בפעולה הראשית יש להכריז על מערך של 10 צבים.
- בפעולה הראשית יש לאתחל את מערך הצבים (יצירת 10 צבים)
- אחרי איתחול המערך, כל צבים שנוצרו נמצאים במרכז, פניהם צפונה. יש למקם אותם כמו באיור ולסובב ימינה.

- שלב 2 – קליטת ניחוש המשתמש.

- שלב 3 – הדמיית התחרות.

- כל צב זז משמאל לימין במהירות אקראית המשתנה כל צעד.
- כל הצבים זזים "במקביל". כשהצב הראשון עבר את קו הסיום המרוץ מסתיים.
- שלב 4 – קביעת המנצח.
 - אחרי שהצבים עצרו מתבצעת בדיקה מה מספרו של הצב ש"ניצח" (חיפוש מקסימום).
 - בשלבים הבאים ניתן לבדוק מי מהצבים הגיע למקום שני ושלישי.
- שלב 5 – בדיקת המנצח בין המהמרים.
 - בדיקה האם ההימור על הצב המנצח, שנבחר בתחילת המשחק, אכן היה נכון.
 - הבדיקה תלווה בהודעה מתאימה.
- שלב 6 – שיפור
 - מה צריך לשנות בתכנית בהנחה שיתכן שכמה צבים הגיעו ראשונים אל קו הסיום? חיפוש מספריהם הסידוריים של המנצחים.
 - מה צריך לשנות בתכנית כאשר לא ידוע מראש גודל המערך של צבים (צבים שמתתפים במרוץ) והוא תלוי בקלט לתוכנית?

התכנית הראשית – מרוץ צבים

```

namespace TestTurtleLib
{
    class Program
    {
        static void Main(string[] args)
        {
            Turtle[] turtles = new Turtle[10];
            int i, j;
            Turtle t = new Turtle();
            t.SetVisible(false);
            t.TurnRight(90);
            t.MoveForward(400);
            t.TurnLeft(90);
            t.MoveForward(200);
            t.TailDown();
            t.MoveBackward(400);

            Random rnd = new Random();
            for (i = 0; i < turtles.Length - 1; i++)
            {
                turtles[i] = new Turtle();
                turtles[i].MoveForward(100 - i * 30);
                turtles[i].TurnRight(90);
            }
        }
    }
}

```

```

        turtles[i].MoveBackward(400);
        //turtles[i].SetDelay(1);
    }
    int choice;
    Console.WriteLine("Enter your choice 0..{0}", turtles.Length - 1);
    choice = int.Parse(Console.ReadLine());
    t.SetDelay(1);
    double d;

    bool stop = false;
    while(!stop)
    {
        for (i = 0; i < turtles.Length - 1; i++)
        {
            d = rnd.Next(1, 5);
            d = rnd.NextDouble()*2;
            turtles[i].MoveForward(d);
            if (turtles[i].GetX() >= 400)
                stop = true;
        }
        t.MoveForward(0);
    }
    int turtleWin = 0;
    double x, maxX = 0;
    for (i = 0; i < turtles.Length - 1; i++)
    {
        x = turtles[i].GetX();
        if (x > maxX)
        {
            maxX = x;
            turtleWin = i;
        }
    }
    if (choice == turtleWin)
        Console.WriteLine("Your are winner");
    else
        Console.WriteLine("Turtle {0} is win", turtleWin);
    }
}
}

```

דפי מעבדה: מרוץ צבים

סיכום ביניים: ראינו בכיתה כיצד מערך של עצמים עוזר לנו לטפל בכמויות גדולות של עצמים מאותו סוג. במעבדה הזו, אתם תשתמשו במערך של עצמים כדי לבנות תכנית של מרוץ צבים.

שלב א: בניית מערך עצמים (השימוש בפעולה *new*)

בשלב הראשון, אנחנו נבנה מערך עצמים ונבין גם את ההבדל בין השלב של הגדרת מערך העצמים לבין השלב של אתחול המערך.

לפניכם הקטע הבא במחלקה הראשית:

```
namespace TestTurtlesRace
{
    public class TurtlesRace
    {
        static void Main(string[] args)
        {
            Turtle[] turtles = new Turtle[10];
            Console.ReadLine();
        }
    }
}
```

בדוגמאות שעבדנו איתן בעבר, ראיתם תוכנית המשתמשת במחלקה *Turtle* ובמקרים האלה הופיעו תמיד שני חלונות: החלון הגרפי של המחלקה *Turtle* וחלון הטקסט של המחלקה *Console*. כמו בשרטוט הבא:



חלון טקסט של המחלקה *Console*

נסו לשער מה יוצג כאשר נריץ את התכנית וכמה חלונות יוצגו על המסך? אחרי שרשמתם את הניחוש שלכם, הקלידו את התכנית, הריצו את התכנית ובדקו מה ייוצג.

נסו להסביר מדוע רואים על המסך רק חלון של המחלקה *Console*.

כמה עצמים מטיפוס *Turtle* נוצרו בתוכנית הראשית? הסבירו.

עכשיו הוסיפו לתכנית את ההוראות:

```
Turtle[] turtles = new Turtle[10];
```

```
turtles[0] = new Turtle[10];
```

נסו לשער מה יוצג כאשר נריץ את התכנית?

הריצו את התכנית ובדקו אותה.
כמה עצמים מטיפוס *Turtle* נוצרו בתוכנית הראשית? הסבירו.
האם קיימים עצמים מטיפוס אחר (כלומר, עצמים שאינם צבים)?

כעת, שנו את התכנית והריצו אותה:

```
Turtle[] turtles = new Turtle[10];
int i;
for (i = 0; i < turtles.Length - 1; i++)
{
    turtles[i] = new Turtle[10];
}
```

כמה עצמים מטיפוס *Turtle* נוצרו בתוכנית הראשית? הסבירו.
למרות שנוצרו _____ עצמים על פני החלון הגרפי נראה רק עצם אחד. הסבירו מדוע.

שלב ב: אתחול מערך העצמים

לצורך המשמה של מרוץ צבים, תוך כדי אתחול יש ליצור 10 עצמים וגם למקם אותם.

```
Turtle[] turtles = new Turtle[10];
int i;
for (i = 0; i < turtles.Length - 1; i++)
{
    turtles[i] = new Turtle[10];
    // מקם את העצם turtles[i] לאחר שנוצר למקום המתאים
    // כוון את העצם turtles[i] לכיוון ימינה
}
```

שלב ג: הזזה של הצבים (הדמייה של המרוץ)

בשלב הזה נצטרך להזיז על החלון הגרפי את כל הצבים במקביל.
השלימו והקלידו את הלולאה – הדמיית התחרות

```
int j;
for (j = 0; i < 200; j++)
{
    for (i = 0; i < turtles.Length - 1; i++)
    {
        // הזז את העצם turtles[i] לכיוון ימינה מספר אקראי של צעדים
    }
}
```

תוספות אפשריות לשיפור המרוץ:

1) אפשר לשפר את הדמיית התחרות כך שכאשר הצב הראשון עובר את קו הסיום המרוץ מסתיים.

רמז: הפעולה *GetX* מחזירה את שיעור ה-*x* של צב.

2) אפשר לשפר את הדמיית התחרות על-ידי ציור קו הסיום לפני שמתחילה התחרות.

שלב ד: קליטת ניחוש המשתמש בתחילת המרוץ וקביעת המנצח בסוף.

אחרי שהצבים עוצרים, תתבצע בדיקה מה מספרו של הצב ש"ניצח" (רמז: חיפוש מקסימום).
הוסיפו לתכנית שלכם את המרכיב הזה.
בשלבים הבאים ניתן לבדוק מי מהצבים הגיע למקום שני ושלישי. הוסיפו לתכנית שלכם את המרכיב הזה.
מה צריך לשנות בתכנית בהנחה שיתכן שכמה צבים הגיעו ראשונים אל קו הסיום? (רמז: חיפוש מספריהם הסידוריים של המנצחים).
מה צריך לשנות בתכנית כאשר לא ידוע מראש גודל המערך של הצבים שמשתתפים במרוץ והוא תלוי בקלט לתוכנית?