**ניפוי שגיאות (Debug)**

מאת

**ישראל אברמוביץ**

**מבוא**

מטרת הקובץ היא לתת לתלמידים כלי יעיל לבדיקה של נכונות האלגוריתם, לתיקון שגיאות וללימוד דרך טעויות.

**החזון: לתת לתלמידים כלים ללמוד לבד ...**



**דוא"ל להערות ותוספות**: Israel.abramovich@gmail.com

**תודה ליוסי ירון למני עבודי ולמשתתפי הקורס "ידע פדגוגי תכני" (של ד"ר עפרה ברנדס)**

בדף העבודה יש תירגול בסביבת העבודה לשפת C# (Visual Studio 2015) אך היא מתאימה גם לשפת Java וגם לסביבות עבודה אחרות.

שגיאות הידור (קומפילציה)

המחשב לפני או בזמן ביצוע הידור (קומפילציה), יציין שגיאה כאשר משתנה לא נמצא בטווח ההכרה

|  |  |
| --- | --- |
|  | **מה מקור שגיאות הקומפילציה הבאות?**  תשובה: השמה של double ל int |
|  | תשובה: בפעולה Absolut() תחום הגדרתו או הצהרתו של המשתנה x מותנת בקיום התנאי, ובפעולה הראשית Main() תחום ההכרה של המשתנה i הוא רק בתוך הלולאת for  שימו לב לקווים האדומים מתחת למשתנים עם שגיאות הקומפילציה |

שגיאות בתפעול סביבת העבודה

# ריצה על כל התוכנית ועצירה בסוף

**בעיה**: לאן מסך ה Console (מסך הקלט/פלט) נעלם?

אם מריצים את התוכנית ע"י אחת האפשרויות:

* + לחיצה על מקש: <F5>
  + לחיצה על 
  + Debug->Start Debugging

התוכנית לא עוצרת בסיום ההרצת התוכנית, ולכן אם בסיום התוכנית יש פלט, מכיוון שהתוכנית נסגרת הוא "נעלם"

פתרון ביניים

הוסיפו לתוכנית שורת קלט בסוף התוכנית, כדי שהתוכנית תעצור בשורה האחרונה ולא תסתיים

|  |
| --- |
| int x,y;  Console.WriteLine("Please enter number");  x = int.Parse(Console.ReadLine());  y = x + 1;  Console.WriteLine("x= " + x + " y= " + y);  Console.WriteLine("Please press on <ENTER> key");  Console.ReadLine(); |

תרגול:

הריצו שוב את התוכנית, עם אחת האפשרויות:

* + לחיצה על מקש: <F5>
  + לחיצה על 
  + Debug->Start Debugging

הפעם התוכנית תעצור בשורת הקלט ותחכה להקלדת <ENTER>

**פתרון נוסף:**

יש לבחור אחת מהאפשרויות:

* + לחיצה על שני המקשים: <Ctrl> + <F5>
  + Debug->Start Without Debugging

מריץ את כל התוכנית, ועוצר בסופה. כך ניתן לבדוק אם הפלט של התוכנית נכון.

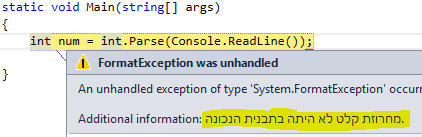
|  |
| --- |
| int x,y;  Console.WriteLine("Please enter number");  x = int.Parse(Console.ReadLine());  y = x + 1;  Console.WriteLine("x= " + x + " y= " + y); |

תרגול:

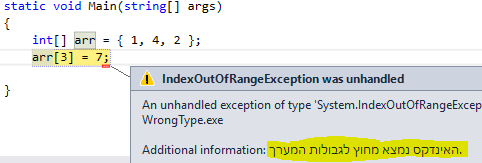
העתיקו את התוכנית לסביבת העבודה, הריצו אותה הכניסו קלט ובידקו את הפלט בחלון ה Console (מסך הקלט/פלט).

שגיאות זמן ריצה (לפני Debug)

# קלט של משתנה מסוג לא מתאים



# חריגה מגודל מערך



ניפוי שגיאות (Debug) בזמן ריצה

# עצירה לאחר ביצוע כל שורה בלי כניסה לפעולות (Step Over)

מצב ניפוי שגיאות זה, משמש לריצה על כל אחת משורות התוכנית, כאשר המחשב עוצר לאחר ביצוע של כל שורה, במצב זה אם בשורת הפקודה יש קריאה לפעולה, הקריאה לפעולה מתבצעת אך המחשב לא "נכנס" לפעולה אלא ממשיך לשורה הבאה.

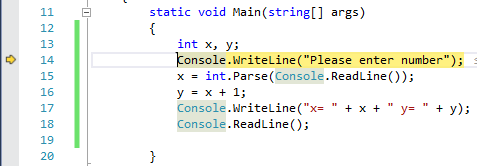
יש לבחור אחת מהאפשרויות:

* + לחיצה על מקש: <F10>
  + Debug->Step Over

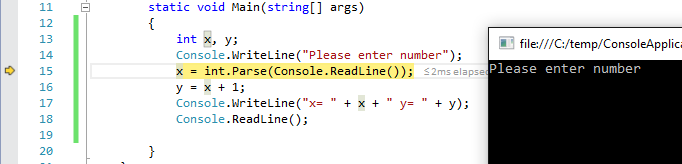
המחשב מבצע שורה בתוכנית ועוצר בשורה הבאה.

**הערה**: יש לשים לב שהשורה שמסומנת בצהוב מתבצעת רק כאשר המחשב עובר לשורה הבאה

|  |
| --- |
| int x,y;  Console.WriteLine("Please enter number");  x = int.Parse(Console.ReadLine());  y = x + 1;  Console.WriteLine("x= " + x + " y= " + y);  Console.ReadLine(); |



בדוגמה הנ"ל שורה 14 תתבצע ושורת הפלט "Please enter number" תודפס רק כאשר המחשב יסמן את שורה 15



# עצירה לאחר ביצוע כל שורה עם כניסה לפעולות (Step Into)

מצב ניפוי שגיאות זה, משמש לריצה על כל אחת משורות התוכנית, כאשר המחשב עוצר לאחר ביצוע של כל שורה, במצב זה אם בשורת הפקודה יש קריאה לפעולה, הקריאה לפעולה מתבצעת **והמחשב "נכנס" לפעולה** (בניגוד למצב Step Over שבה המחשב לא נכנס לפעולה) , בסיום הפעולה המחשב ממשיך לשורה שלאחר הקריאה לפעולה.

יש לבחור אחת מהאפשרויות:

* + לחיצה על מקש: <F11>
  + Debug->Step Into

מבצע שורה בתוכנית ועוצר.

|  |
| --- |
| public static int Pow(int i)  {  // ט.כ הפעולה מקבלת מספר שלם  // ט.י הפעולה מחזירה את הערך בריבוע  i = i \* i;  return i;  }  static void Main(string[] args)  {  for (int i = 2; i <= 10; i++)  {  Console.WriteLine(Pow(i));  }  } |

הריצו את התוכנית הנ"ל תוך שימוש ב Step Into ותוך שימוש ב Step Over

האם הפלט זהה?

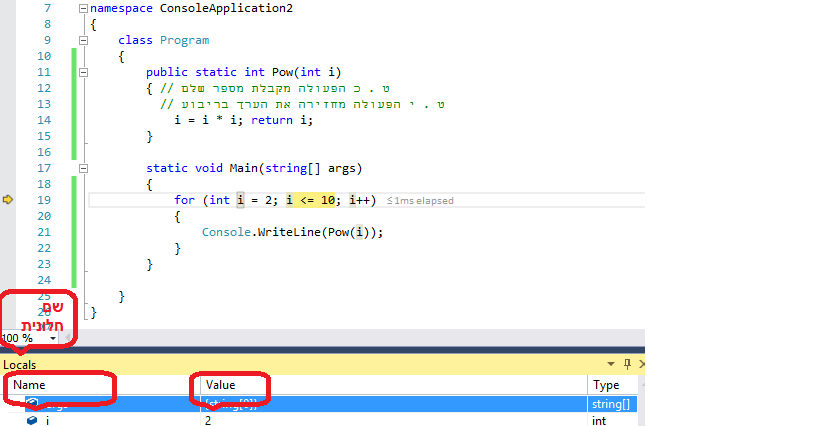
מה היתרון בשימוש בכל אחת מההרצות?

# חלונית Locals

בזמן שמתחילים את ההרצה (שורה אחר שורה) נפתח חלון בשם Locals אשר מראה את הערכים של המשתנים המקומיים.

כתבו טבלת מעקב לתוכנית הבאה:

|  |
| --- |
| public static int Pow(int i)  {  // ט.כ הפעולה מקבלת מספר שלם  // ט.י הפעולה מחזירה את הערך בריבוע  i = i \* i;  return i;  }  static void Main(string[] args)  {  for (int i = 2; i <= 10; i++)  {  Console.WriteLine(Pow(i));  }  } |



שאלה: האם הערכים של משתנה i בתוכנית הראשית זהים לערכים של משתנה i בפעולה ? במידה ויש הבדל, מה לדעתכם הסיבה להבדל.

בתוכנית יש 2 משתנים שונים שלשניהם קוראים i

הריצו שוב את התוכנית תוך שימוש ב Step Into ועקבו בכל פעם אחרי הערך של המשתנה i בחלונית Locals, התוכנית תראה בכל פעם את הערך של המשתנה המקומי i שנמצא בטווח ההכרה.

הערה:

כדי לבדוק את הקוד בזמן ריצה (ולבדוק תיקון אפשרי) ניתן בחלונית Locals לשנות את ערכי המשתנים.

תרגול:

להריץ תוכנית עם 2 לולאות for אחת מתחת לשניה ולעקוב אחרי ערך מונה הלולאה

# חלונית Watch

נתון קטע תוכנית:

הקלט הינו של 3 מספרים שלמים שונים מטיפוס int

|  |
| --- |
| Console.WriteLine("Enter 3 DIFFERENT numbers");  int A = int.Parse(Console.ReadLine());  int B = int.Parse(Console.ReadLine());  int C = int.Parse(Console.ReadLine());  if (A < B)  if (B < C) Console.WriteLine("2");  else if (A < C) Console.WriteLine("3");  else Console.WriteLine("1");  else if (B > C) Console.WriteLine("II");  else if (A > C) Console.WriteLine("III");  else Console.WriteLine("I"); |

בצעו מעקב אחרי הקוד עם טבלת מעקב ידנית (ללא Debugger):

צריך עמודה לכל אחד מהמשתנים, לכל אחד מהתנאים הבוליאנים ולפלט

יש להכניס את הקלט **15,5,10** משמאל לימין.

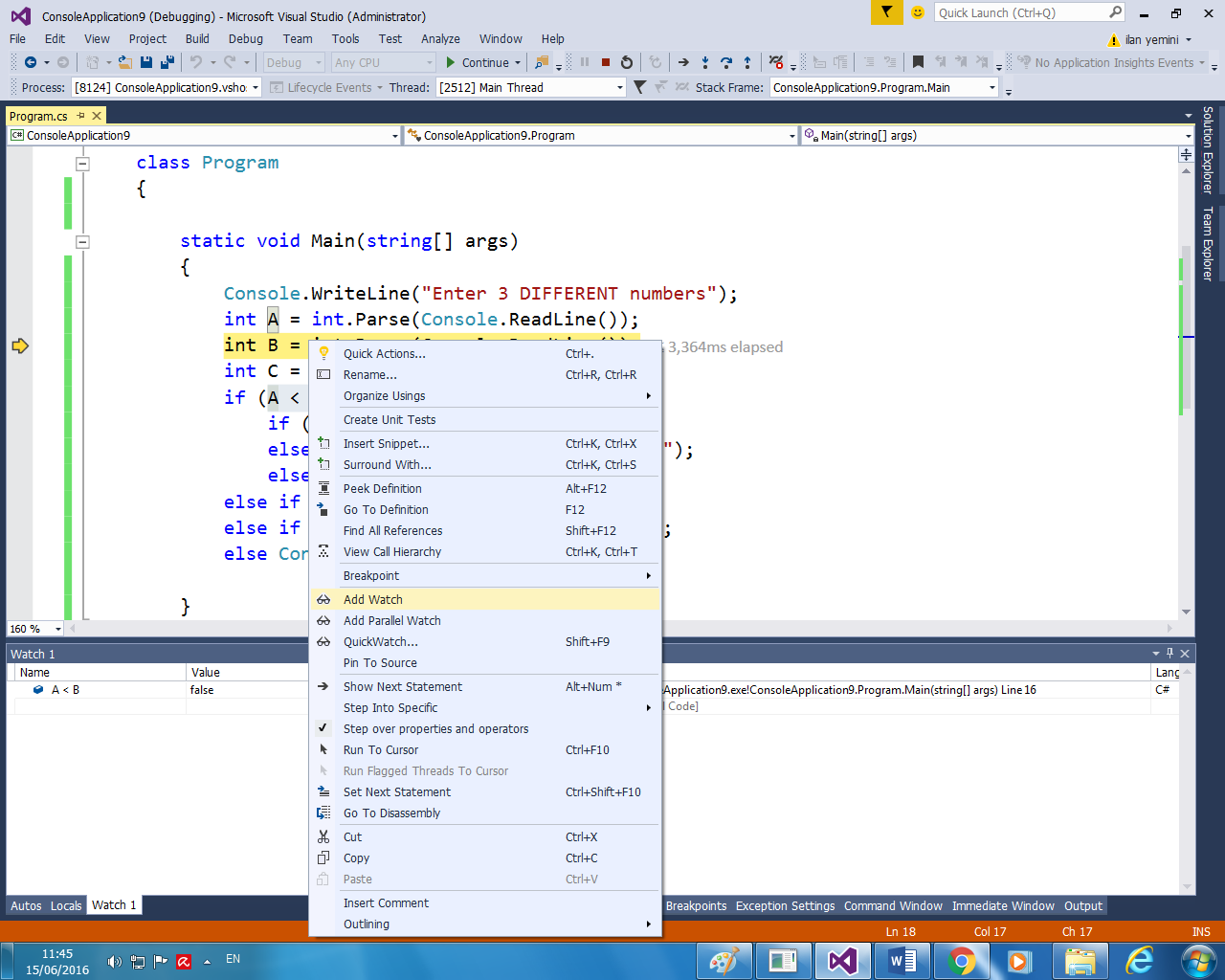
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | A < B | B < C | A < C | B>C | A>C | פלט |
| 15 |  |  |  |  |  |  |  |  |
|  | 5 |  |  |  |  |  |  |  |
|  |  | 10 |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

כדי שהמחשב יעזור לנו לעקוב אחר הקוד, נרצה לראות את הערך של כל אחד מהתנאים הבוליאנים, לשם כך ניתן להשתמש ב Watch

כדי לבצע זאת צריך לסמן תנאי בוליאני מסויים, ואז ע"י לחיצה על המקש הימני בעכבר לבחור 'Add Watch'

לדוגמה: סומן התנאי : A < B

הערה 1: ניתן להשתמש בדרך זו בסימון התנאי רק לאחר תחילת הרצת התוכנית (לדוגמה להריץ שורה בודד ב (Step Over)



ראו בחלונית Watch את הערך הבוליאני של התנאי הבוליאני A < B.

הערה 2: ניתן לרשום תנאי בוליאני או כל חישוב אחר (לדוגמה A+10) ישירות בתא פנוי בחלונית ה Watch

1. בצעו שוב מעקב אחרי הקוד עם טבלת מעקב, הפעם ניתן להעזר ב debugger ובחלונית watch

מה הפלט עבור סדרה A=15 B=5 C=10?

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | A < B | B < C | A < C | B>C | A>C | פלט |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

האם הפתרון זהה לטבלת המעקב הידנית?

אם לא מה הייתה השגיאה?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. תנו דוגמה לקלט אשר עבורו יהיה הפלט: '1' A = \_\_\_ , B = \_\_\_ ,C =\_\_\_
2. תנו דוגמה לקלט אשר עבורו יהיה הפלט: 'I' A = \_\_\_ , B = \_\_\_ ,C =\_\_\_
3. תנו דוגמה לקלט אשר עבורו יהיה הפלט: '2' A = \_\_\_ , B = \_\_\_ ,C =\_\_\_
4. תנו דוגמה לקלט אשר עבורו יהיה הפלט: 'II' A = \_\_\_ , B = \_\_\_ ,C =\_\_\_
5. תנו דוגמה לקלט אשר עבורו יהיה הפלט: '3' A = \_\_\_ , B = \_\_\_ ,C =\_\_\_
6. תנו דוגמה לקלט אשר עבורו יהיה הפלט: 'III' A = \_\_\_ , B = \_\_\_ ,C =\_\_\_

# ריצה של התוכנית ועצירה בסמן

כדי להגדיר לתוכנית לרוץ עד הסמן, יש לעמוד עם הסמן על השורה שבה רוצים שהתוכנית תעצור, ניתן להגיע למיקום על ידי החיצים, או על ידי לחיצה על מקש עכבר שמאלי

יש לבחור אחת מהאפשרויות:

* + לחיצה על שני המקשים: <Ctrl> + <F10>
  + לחיצה על מקש עכבר ימני, ואז לבחור בתפריט: Run To Cursor

המחשב מריץ את התוכנית, ועוצר במיקום של הסמן.

דוגמה לבדיקת פתרון:

התוכנית הבאה אמורה להגריל מספר בין 0 ל 99 כולל ולשחק את המשחק 7 בום, כלומר היא אמורה ולהדפיס את המספר שמקיים את החוקים (אחת הספרות שווה 7 או המספר מתחלק ב 7) ואז לסיים את התוכנית.

|  |
| --- |
| for (int i = 0; i < 100; i++)  {  if ((i % 10 == 7) && (i / 10 == 7) && (i % 7 == 0))  {  Console.WriteLine("BOOM i= " + i);  }  } |

יש בתוכנית שגיאה...

הריצו את התוכנית עד שתגיע לשורה המסומנת בצהוב וגלו את השגיאה.

אם עדיין התשובה לא ברורה, עצרו את התוכנית בשורת התנאי:

if ((i % 10 == 7) && (i / 10 == 7) && (i % 7 == 0))

שנו את הערך של משתנה i (בחלונית Local ) ובדקו כל אחד מהתנאים הבוליאנים (בחלונית Watch)

# ריצה של התוכנית ועצירה במספר נקודות עצירה (BreakPoint)

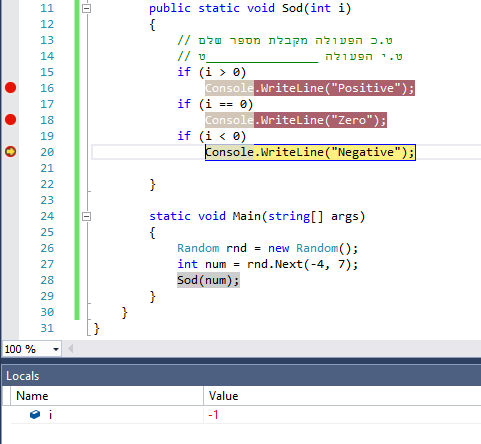
כדי להגדיר נקודות עצירה, יש לעמוד עם הסמן על השורה שבה רוצים להוסיף את נקודת העצירה. ואז יש לבחור אחת מהאפשרויות:

* + לחיצה על מקש: <F9>
  + Debug->Toggle BreakPoint
  + לחיצה על עכבר ימני, ואז לבחור Breakpoint -> Insert Breakpoint

כדי להריץ את התוכנית עד לנקודת העצירה, יש לבחור אחת מהאפשרויות:

* + לחיצה על מקש: <F5>
  + Debug->Start Debugging

מריץ את כל התוכנית, ועוצר (כאשר המחשב מגיע) באחת מנקודות העצירה.



# הפניות לעצמים

**שאלה 1**

נתונה המחלקה המייצגת מעגל :

public class Circle

{

private double radiuos ; //רדיוס המעגל

private string color ; // צבע המעגל

}

א . נתון קטע תוכנית בשפת C# המוגדר בתוכנית הראשית ונתונים שלושה עצמים.

Circle cir1 = new Circle ( 13.0 , "Red" ) ;

Circle cir2 = new Circle ( ) ;

Circle cir3 = new Circle ( cir2) ;

cir1

Radius color

13.0 Red

cir2

Radiuos color

1.0 Blue

cir3

Radius color

1.0 Blue

ממשו את ה הפעולות הבונות במחלקה Square כך שקטע התוכנית ירוץ .

**לכל אחת מהתכונות יש פעולה קובעת ומאחזרת (Get, Set)**

ב. כתבו למחלקה Circle את הפעולה הפנימית ToString()

ג. בתוכנית הראשית הוגדרו גם

Circle cir4 = cir3 ;

cir2.SetRadiuod(5.6) ;

cir3.SetRadiuod(6.6) ;

cir4.SetRadiuod(7.6) ;

console.WritrLine(cir2);

console.WritrLine(cir3);

console.WritrLine(cir4);

לימוד כלי ניפוי השגיאות (Debug)

א . כתבו את המחלקה

public class Circle

{

private double radiuos ; //רדיוס המעגל

private string color ; // צבע המעגל

}

ב . צרו בתוכנית הראשית את 3 העצמים,

וממשו את הפעולות הבונות במחלקה Circle כך שקטע התוכנית ירוץ .

Circle cir1 = new Circle ( 13.0 , "Red" ) ;

Circle cir2 = new Circle ( ) ;

Circle cir3 = new Circle ( cir2) ;

cir1

Radiuos color

13.0 Red

cir2

Radiuos color

1.0 Blue

cir3

Radiuos color

1.0 Blue

ג. כתבו למחלקה Circle את הפעולה הפנימית ToString()

ד. הוסיפו לתוכנית הראשית את קטע הקוד:

Circle cir4 = cir3;

cir2.SetRadiuod(5.6);

cir3.SetRadiuod(6.6);

cir4.SetRadiuod(7.6);

Console.WritrLine(cir2);

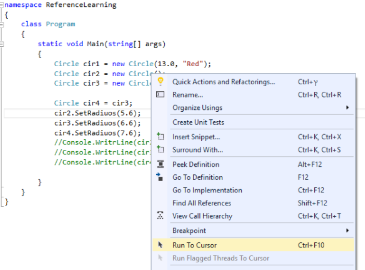
Console.WritrLine(cir3);

Console.WritrLine(cir4);

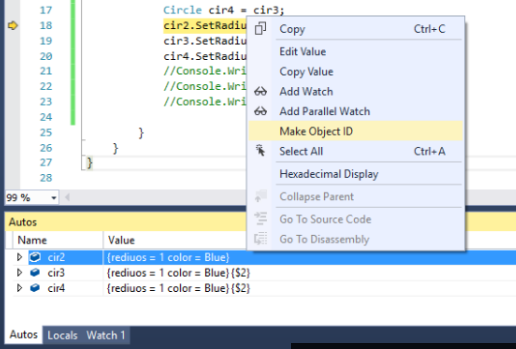
ה. בסביבת העבודה

לרוץ עד השורה שלאחר יצירת ההפניות (שורה: cir2.SetRadiuos(5.6);)

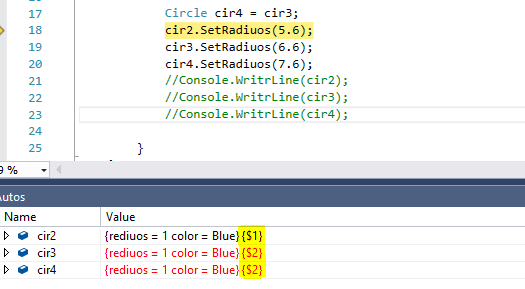
ללחוץ על עכבר ימני ולבחור 'Run To Cursor' (רוץ עד הסמן)



בחלונית Autos / Local / Watch הוסיפו לכל עצם Object ID, ההוספה מתבצעת על ידי עמידה על העצם עם העכבר, לחיצה על המקש הימני ובחירה ב 'Make Object ID'



ההפניות $1 ו $2 מציינות הפניות לעצמים



מה אנו למדים מההפניות? $1 ו $2 ?