משתנים

משתנה הוא מעין "מחסנית זיכרון", שתפקידה לאחסן נתונים באופן זמני. משתנה רגיל יכול לאחסן ערך אחד בלבד. משתנה שמאחסן מספר ערכים נקרא '**מערך**' (ר' פרק 'שגיאה! מקור ההפניה לא נמצא.' בעמ' **שגיאה! הסימניה אינה מוגדרת.**)

תוכן ה"מחסנית" יכול להשתנות תוך כדי ריצת הקוד.

נהוג להכריז על שמות המשתנים בתחילת הקוד וכן על סוגם.

חוקים לשמות המשתנים:

- שם המשתנה חייב להתחיל ב**אות**.
- אין להוסיף רווחים לשם המשתנה.
- אין לתת למשתנה שם הזהה לשם המאקרו.
- אין לתת למשתנים שמות שמורים כגון print או save.

כדי להימנע מבעיות העלולות לצוץ בעקבות מתן שמות שמורים, תוכלו להוסיף לשם המשתנה את התחילית My, לדוגמה - MySum.

- מומלץ לתת שם המתאר באופן מדויק את משמעות המשתנה, לדוגמה - למשתנה שאמור לאחסן את מספר השורה האחרונה בעמודה מסוימת, מומלץ לקרוא LastRow.
- ניתן להשתמש בקו תחתי כדי "לאחד" שתי מילים, לדוגמה - Last_Row.

נהוג להשתמש בשילוב של אותיות קטנות וגדולות לצורך נוחות הקריאה.

- הקלדת משתנה באופן שגוי תגרום ליצירת משתנה חדש.
  לדוגמה - נניח שנתתם למשתנה את השם MyRng ובמהלך הקוד פניתם אליו בטעות בשם MyRange, משתנה חדש בשם זה יווצר באופן אוטומטי.
- לפני הרצת הקוד, מקבלים המשתנים ערך ראשוני:
  משתנים אשר הוכרזו כמשתנים מספריים מקבלים את הערך אפס, ואילו משתנים שהוכרזו כמשתני טקסט, יקבלו את הערך "empty".
- כאשר עוצרים את ריצת הקוד, ערך המשתנה מתאפס באופן אוטומטי, ואינו נשמר בזיכרון (כלומר, עם סיום ריצת הקוד - "מחסניות הזיכרון" מתרוקנות מתוכנן).

הערה:

מתן שם לא חוקי למשתנה יצבע את השורה באדום.

סוגי משתנים

קיים מספר רב של משתנים, אנו נתייחס רק לחשובים שביניהם:

משתנים מספריים

סוג המשתנה	גודל בבתים	טווח ערכים
Byte	1	0-255
Boolean	2	True or false
Integer	2	(-32,768)-(32,767)
Long	4	(-2,147,483,648)-(2,147,483,647)

משתנים נוספים

Double - מאפשר לאחסן בתוכו מספרים עשרוניים

String - מאפשר לאחסן בתוכו מחרוזות טקסט

Range - מאפשר לאחסן בתוכו טווחים

Date - מאפשר לאחסן בתוכו תאריכים

Variant - "משתנה על" - מאחסן בתוכו את כל סוגי המשתנים האחרים

באופן עקרוני, ולצורך חיסכון במקום, עדיף להשתמש בסוג המשתנה בעל טווח הערכים הקטן ביותר, אשר עדיין יכול להכיל את ערכי המשתנה.

שימו לב, אין חובה להכריז על משתנים, אולם אם לא נכריז, יקבע סוג המשתנה כ-VARIANT, אשר תופס מקום רב בזיכרון.

כדי לחייב את המשתמשים להכריז על משתנים במודול מסוים, ניתן להוסיף בראש המודול את הפקודה **Option Explicit**, שתמנע את ריצת הקוד במידה שקיימים משתנים שלא הוכרזו.

את הפקודה יש לכתוב בראש כל מודול בנפרד.

ההכרזה על המשתנים מתבצעת באמצעות הפקודה **Dim**

לדוגמה:

```
Dim LastRow As Integer
Dim Rng As Range
```

בקוד זה הגדרנו את המשתנה LastRow כמספר שלם (Integer)

ואילו את המשתנה Rng הגדרנו כטווח (Range)

מדוע כדאי לחייב את המשתמש להכריז על משתנים?

למרות שלא קיימת חובה להכריז על משתנים, הרי שאי הכרזה עליהם עלולה לגרום לעתים לבעיות, וזאת בשל העובדה שבכל פעם שהמהדר נתקל במשתנה שאינו מכיר, הוא יוצר משתנה חדש.

הדוגמה הבאה תעזור להבהיר את הדברים:

ברשותנו מוצר במחיר מסוים, ואנו רוצים להעלות את מחירו ב-10%. עלינו להכפיל את המחיר הישן ב-1.1, לפיכך כתבנו את הקוד הבא:

```
OldPrice = 30
NewPrice = price * 1.1
```

נתנו למשתנה OldPrice ערך התחלתי של 30.

לאחר מכן רצינו לקבל את ערכו החדש של המוצר במשתנה NewPrice, אולם בשל חוסר תשומת לב, במקום להשתמש במשתנה OldPrice, השתמשנו במשתנה Price.

משום שהמשתנה Price אינו מוכר, הוא נוצר באופן אוטומטי וקיבל את הערך 0, ולכן תוצאת הכפל היא 0, ולא 33 כמצופה.

לו היינו מחייבים את המשתמש להצהיר על משתנים, היינו מקבלים הודעת שגיאה, בעת שהמהדר היה נתקל במשתנה Price.

ניתן להכריז על משתנה ללא הכרזת סוג המשתנה:

```
Dim LastRow
```

במקרה זה תופעל בקרת האיכות, אשר תמנע מאתנו להשתמש במשתנים באופן שגוי, אולם סוג המשתנה יקבע כ-Variant ויגזול מקום רב בזיכרון.

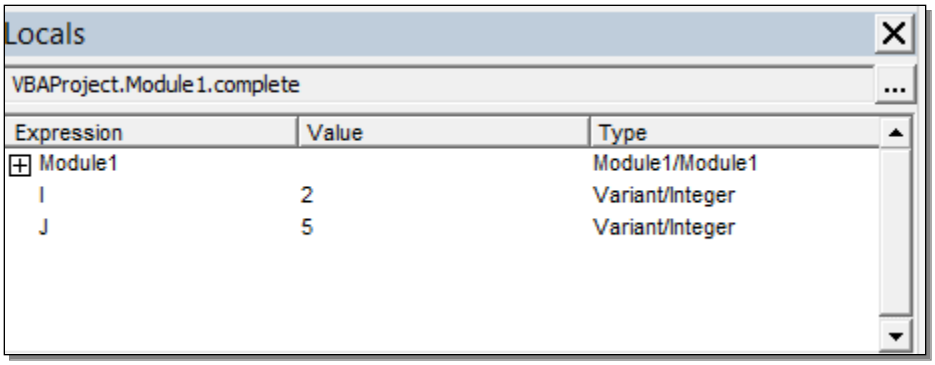
<u>צפייה במשתנים תוך כדי ריצת קוד</u>

בעת הרצת הקוד 'צעד אחרי צעד' ניתן לצפות בערכים אותם מקבלים המשתנים

חלונית Locals

מציגה רשימה של כל המשתנים בפרוצדורה.

להוספת החלונית בחרו בתפריט **View → Locals Window**



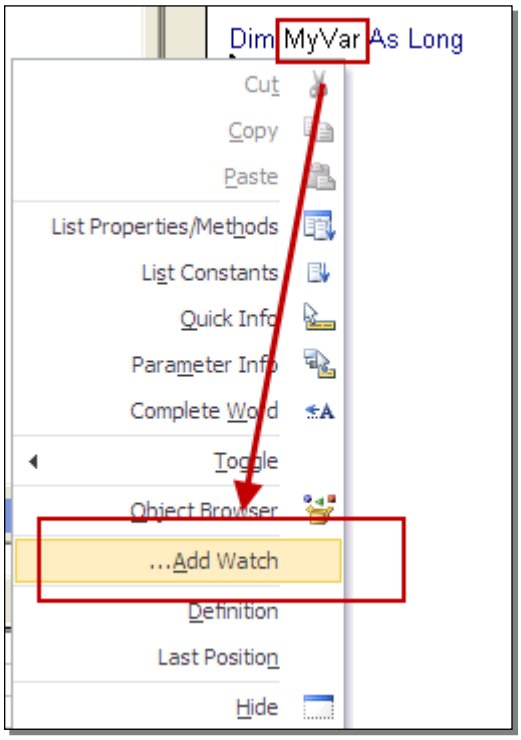
בזמן הפעלת הקוד 'צעד אחרי צעד' ניתן לראות את השתנותם תוך כדי ריצה.

חלונית Watch

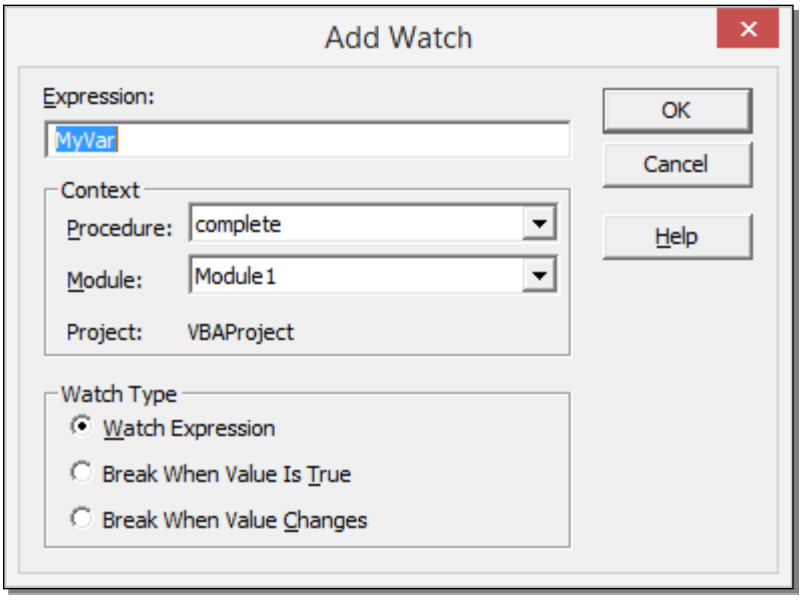
מאפשרת צפייה במשתנים נבחרים תוך כדי ריצת הקוד.

<u>הוספת משתנים לחלונית:</u>

1. לחצו לחיצה ימנית על המשתנה (מתוך הקוד)
2. בחרו באפשרות **Add Watch**

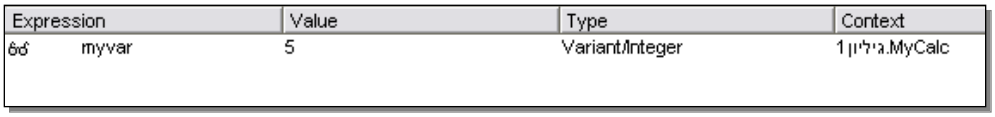


3. יפתח החלון הבא:



4. ודאו שסימנתם את המשתנה הנכון
5. לחצו על 

6. המשתנה יופיע בחלונית, כפי שניתן לראות באיור שלהלן:



העברת משתנים בין פרוצדורות

לפני שנלמד כיצד להעביר משתנים בין פרוצדורות, עלינו ללמוד כיצד לקרוא לפרוצדורה אחת מתוך פרוצדורה אחרת.

ההפניה מתבצעת באמצעות כתיבת שמו של הקוד האחר בתוך הקוד שלנו.

לדוגמה:

```
Sub CallMacro()

    Call MyMsg

End Sub

Sub MyMsg()

    MsgBox ("Hello")

End Sub
```

בדוגמה שלעיל, הרצת הקוד CallMacro תקרא לקוד MyMsg, אשר מציג תיבת טקסט ובה המילה Hello.

הערה: הפקודה Call עוזרת לנו להבין שמדובר בקריאה לפרוצדורה אחרת, אולם ניתן לוותר עליה ולכתוב את הקוד גם ללא השימוש בפקודה Call:

```
Sub CallMacro()

    MyMsg

End Sub
```

```
Sub MyMsg()

    MsgBox ("Hello")

End Sub
```

בנוסף, התוכנה מאפשרת לא רק לקרוא לקודים אחרים, אלא גם להעביר אליהם פרמטרים.

```
(General)
Sub CallMacro()

    MyStr = "Hello"
    MyMsg (MyStr)

End Sub
```

```
Sub MyMsg(MyStr)

    MsgBox MyStr

End Sub
```

בדוגמה שלעיל הזנו לתוך המשתנה MyStr את המחרוזת Hello.

לאחר מכן קראנו לפרוצדורה MyMsg, והעברנו אליה את הערך של המשתנה MyStr.

כדי שהפרוצדורה MyStr תדע שעליה לצפות לקבל ערך כלשהו, כתבנו את שם המשתנה בתוך הסוגריים שליד שם הפרוצדורה, כפי שהוצג בדוגמה שלעיל.

העברת פרמטרים מרובים

ניתן להעביר פרמטרים מרובים בין פרוצדורות. בפרוצדורת המקור נקיף כל פרמטר בסוגריים ונפריד ביניהם בפסיקים, ואילו בפרוצדורת היעד נכתוב את כולם בתוך אותם הסוגריים, מופרדים בפסיקים, כפי שתוכלו לראות בדוגמה הבאה:

```
Sub CallMacro()

    x = 3
    y = 4
    MyMsg (x), (y)

End Sub
```

```
Sub MyMsg(x, y)

    tmp = MsgBox(x & vbNewLine & y)

End Sub
```