

ספר קורס

תכנות מיקרו-בקר

באמצעות שפת C#

מאת

BRK

חינוך טכנולוגי מתקדם

www.brk.co.il





1 יצירת פרויקט חדש ב MICROSOFT VISUAL STUDIO 2012

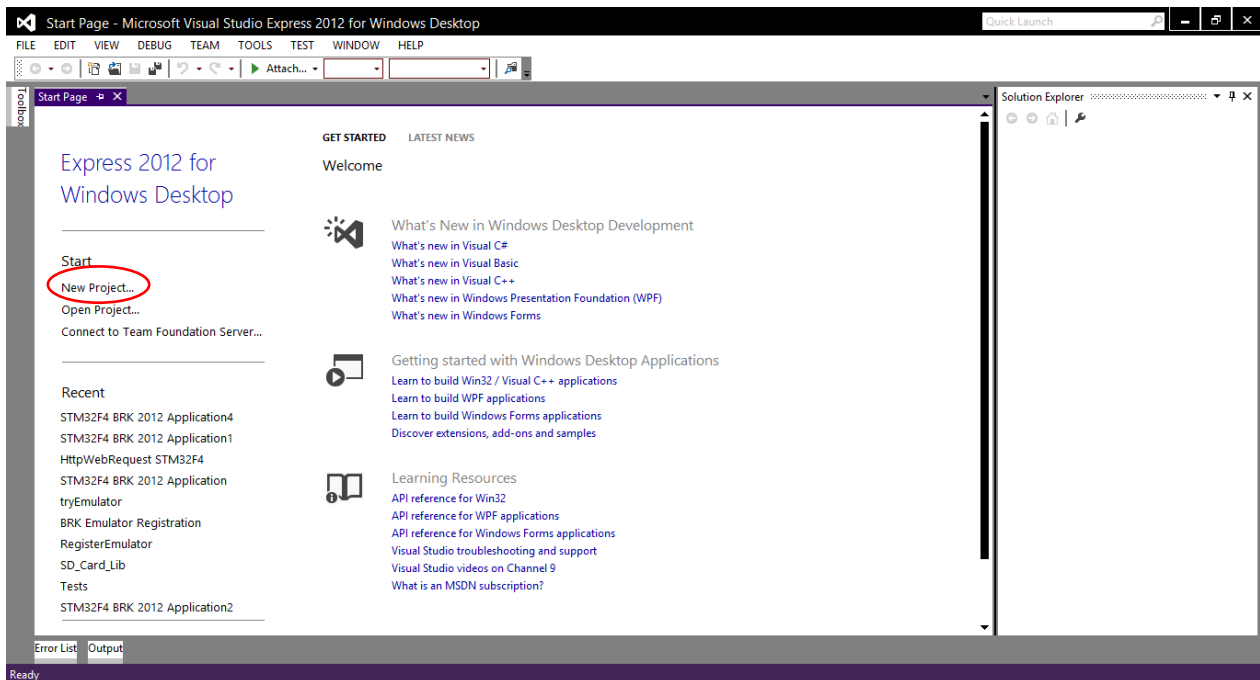
1. חברו את כרטיס הבקר למחשב ע"י חיבור ה USB.

2. וודאו שנדלקו לפחות 2 לדים אדומים ולד ירוק אחד ליד חיבורי ה USB.

3. במידה והלדים לא דולקים, בדקו את החיבורים של קבלי USB למחשב ותקינות יציאות ה USB.

4. על שולחן העבודה שבמחשב עליו אתם עובדים, תלחצו פעמיים עם העכבר על קיצור הדרך של Microsoft Visual Studio Express 2012 for Windows Desktop. במידה וקיצור דרך זה לא קיים שם, ניתן למצוא את התוכנה בתוך רשימת כל התוכניות המותקנות במחשב.

5. יפתח החלון המצולם באיור הבא:



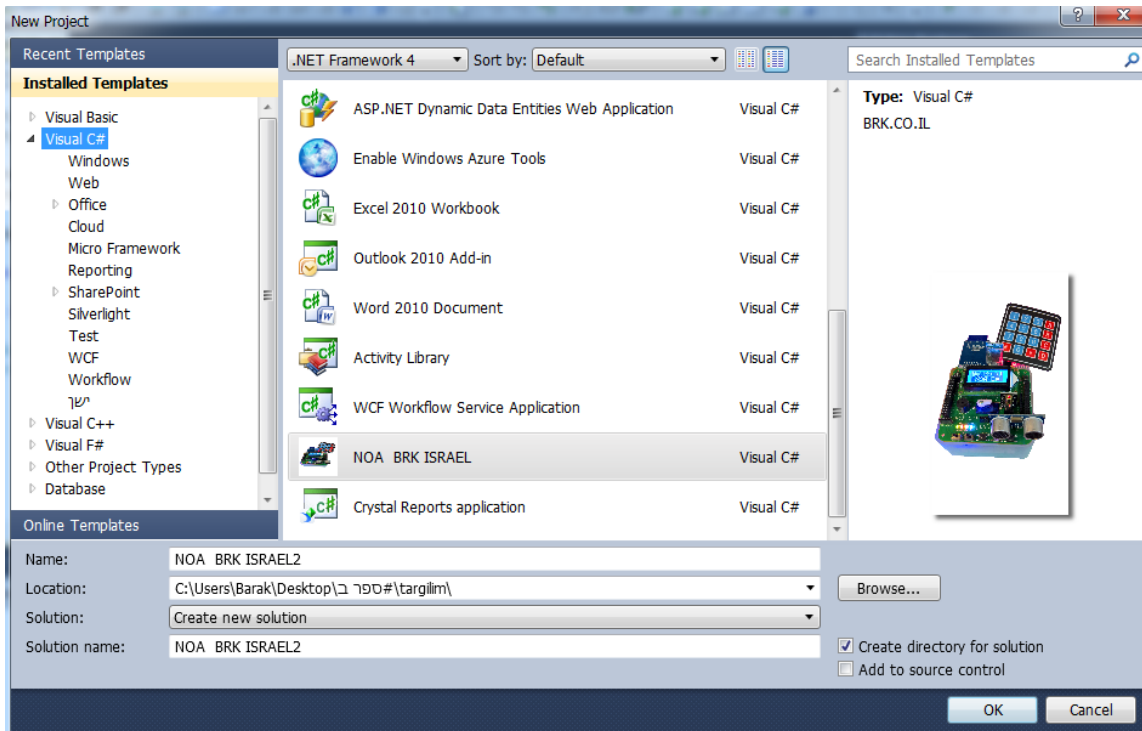
6. בלשונית Start Page שנפתחה בחלקו השמאלי של החלון, יש לבחור עם העכבר את האפשרות

.New Project

ספר קורס תכנות מיקרו-בקר באמצעות שפת C#

חברת BRK – חינוך טכנולוגי מתקדם

7. יפתח החלון הבא:



8. בחלקו השמאלי של החלון שיפתח יש לפתוח את התפריט Visual C# ולסמן את האפשרות NOA BRK ISRAEL כמתואר באיור.

9. בשדה Name שבחלקו התחתון של החלון יש לתת שם לתוכנית (הפרויקט). הקפידו לתת שמות הגיוניים לתוכניות המבטאות את מה שהיא תבצע. יש לתת שמות באנגלית בלבד.

10. בשדה Location שבחלקו התחתון של החלון הכניסו את המיקום במחשב בו ישמר הפרויקט אותה אתם יוצרים כרגע. ניתן להיעזר בלחצן Browse למציאת המיקום הרצוי.

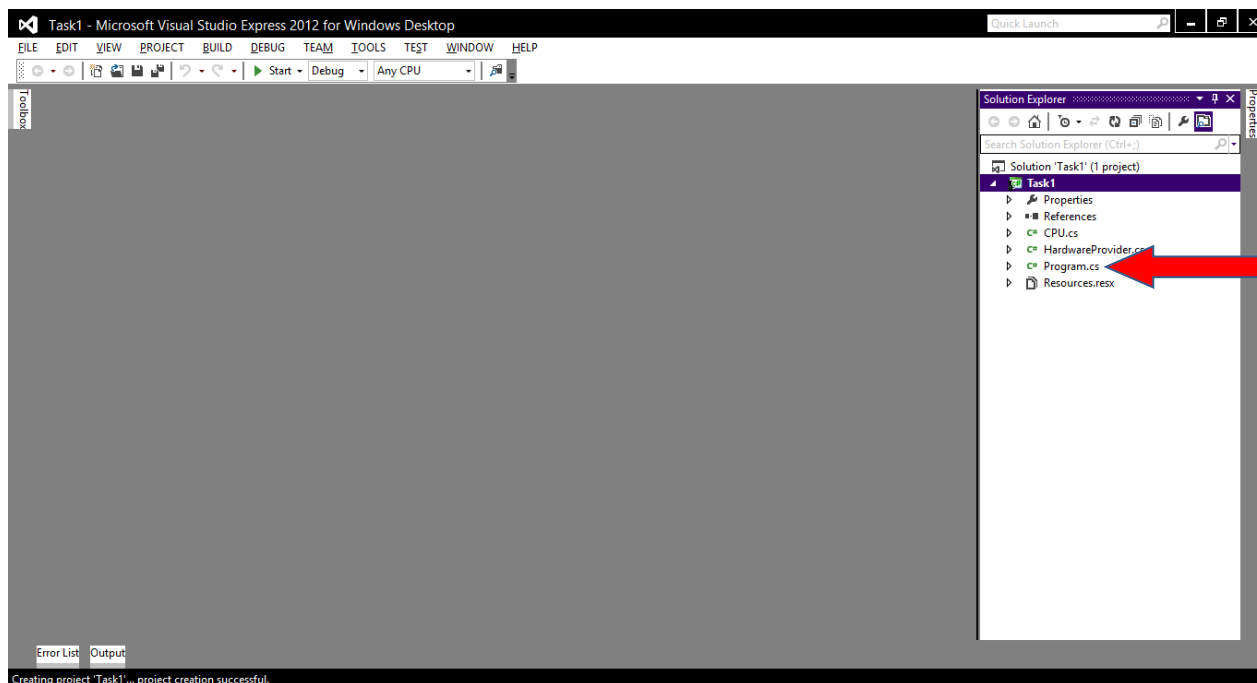
11. וודאו כי קיים סימן V בתיבת Create directory for solution

ספר קורס תכנות מיקרו-בקר באמצעות שפת C#

חברת BRK – חינוך טכנולוגי מתקדם

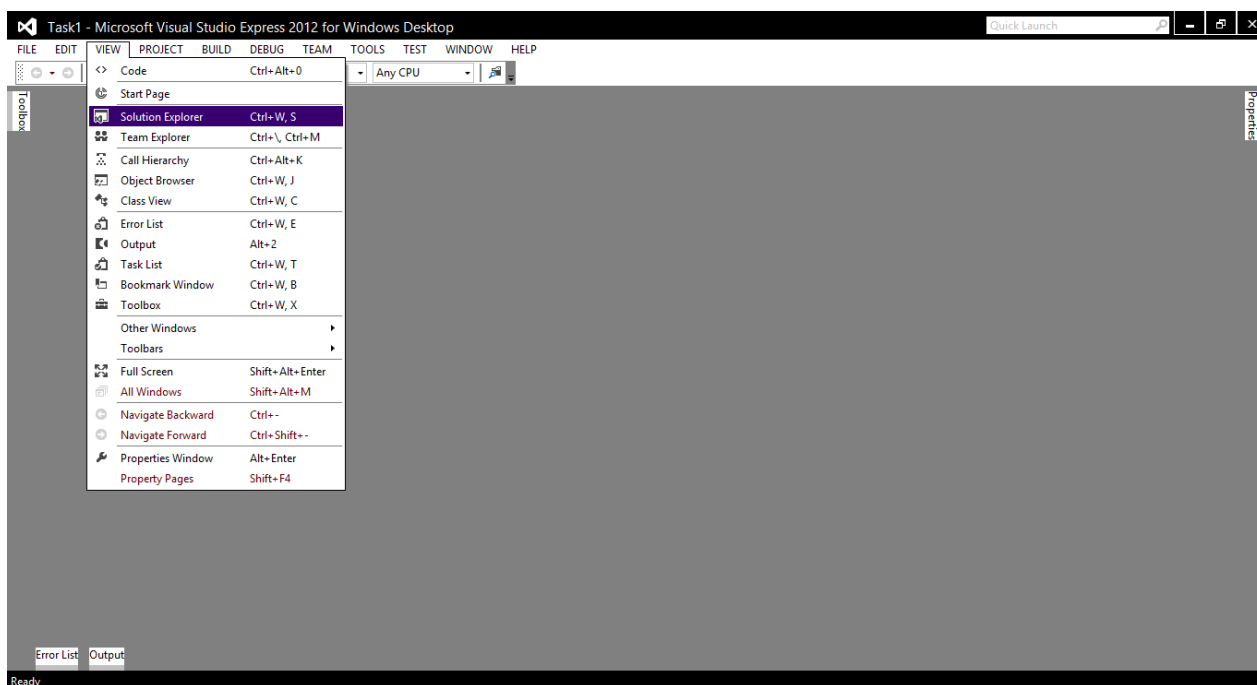
12. לחצו על הלחצן OK בתחתית החלון ליצירת הפרויקט.

13. יפתח החלון הבא:



14. הקישו הקשה כפולה על הקובץ Program.cs במופיע בחלונית Solution שבחלקו הימני של המסך. במידה והחלונית Solution לא מופיעה על המסך, ניתן לבחור אפשרות הצגה שלה מתוך תפריט

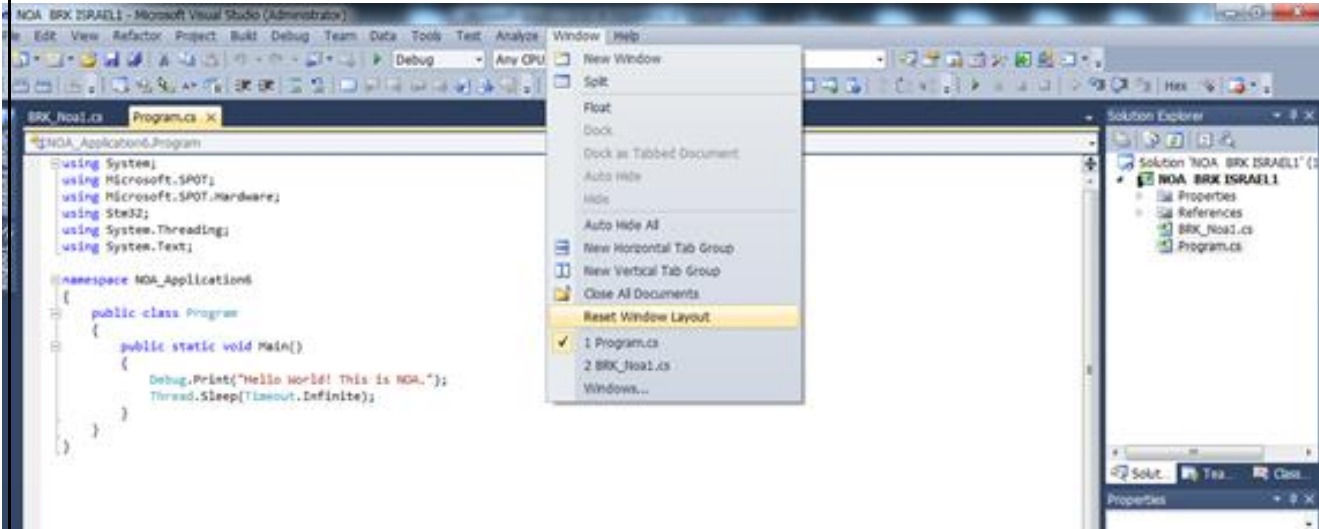
View כמתואר באיור הבא:



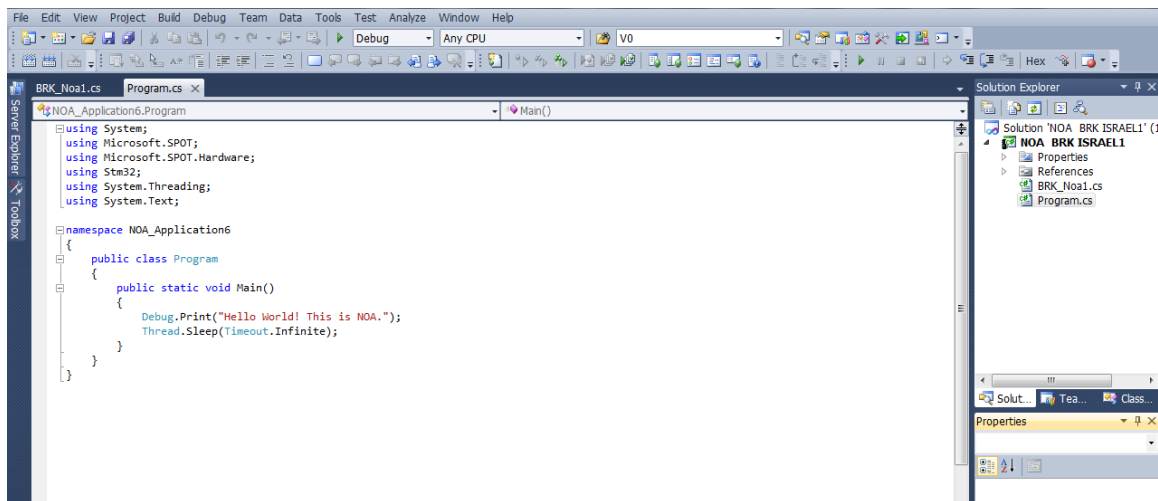
אם מאיזו שהיא סיבה החלונות נמחקו.

ניתן לשחזר את המצב הבסיסי ע"י לחיצה על WINDOW ולאחר מכן על RESET WINDOW

LAYOUT

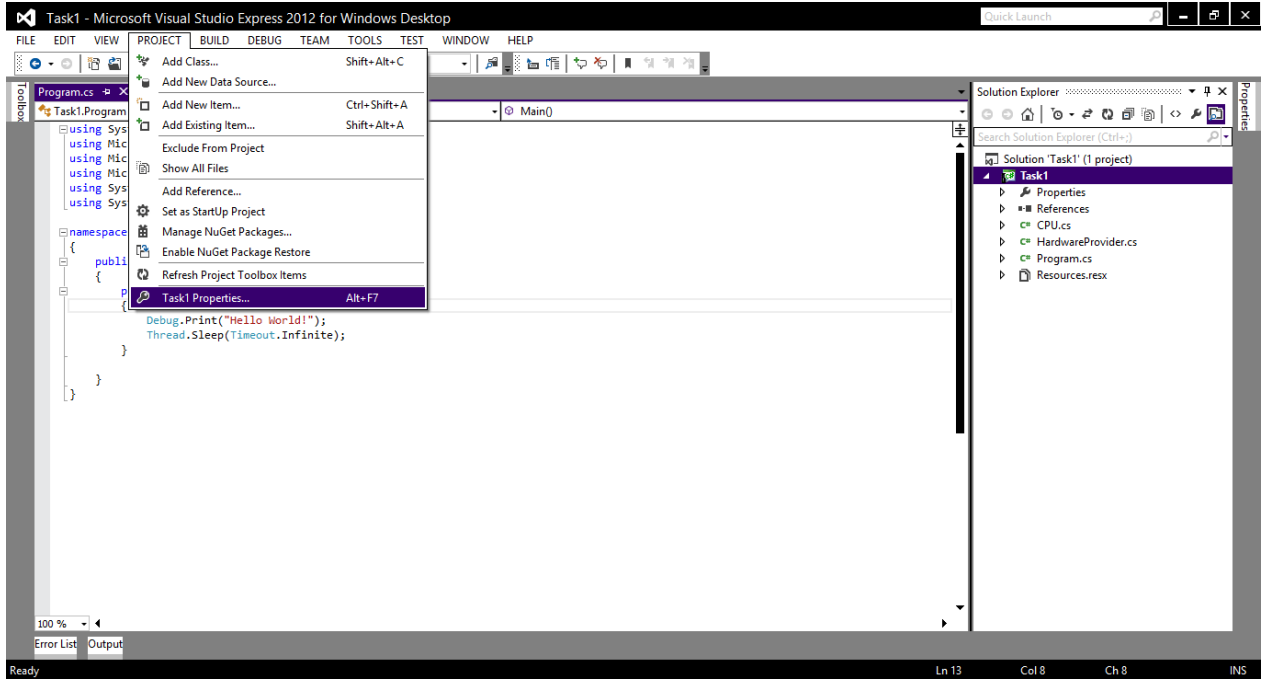


15. אחרי הלחיצה הכפולה עם העכבר על Program.cs תפתח החלונית כמתואר באיור הבא:

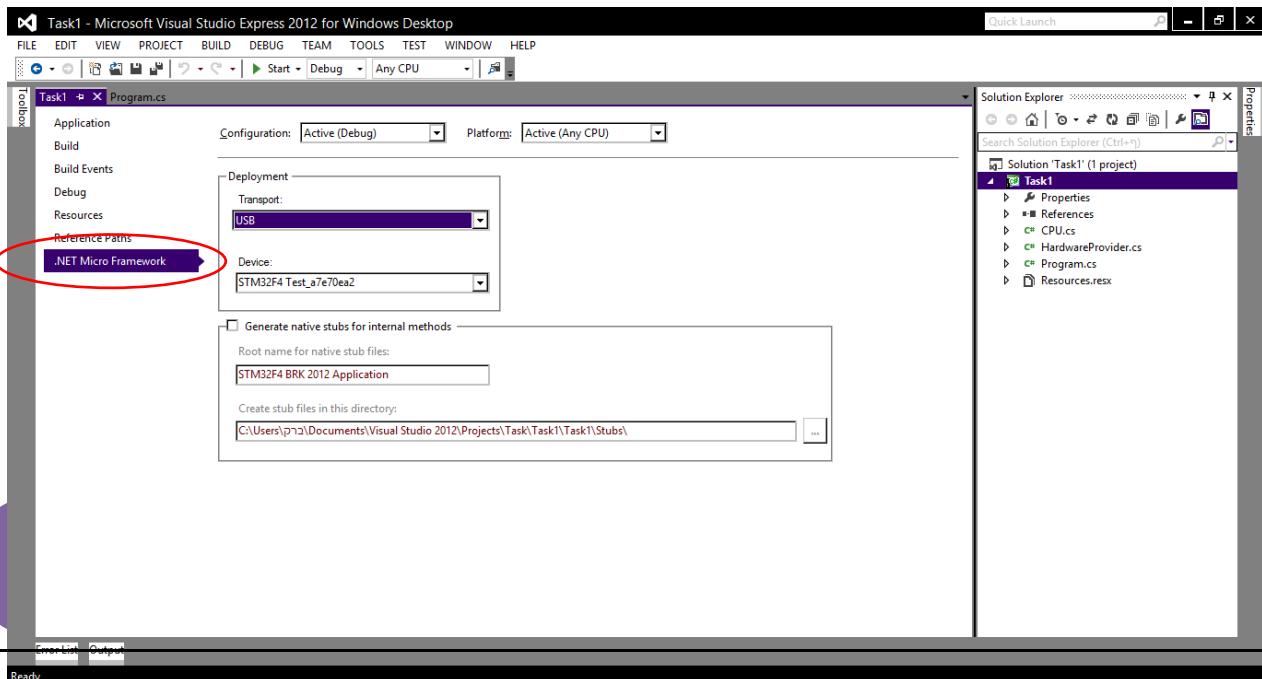


ספר קורס תכנות מיקרו-בקר באמצעות שפת C#

חברת BRK – חינוך טכנולוגי מתקדם



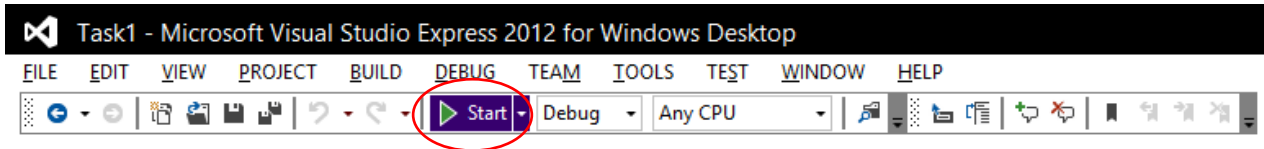
16. יפתח החלון של מאפייו הפרויקט



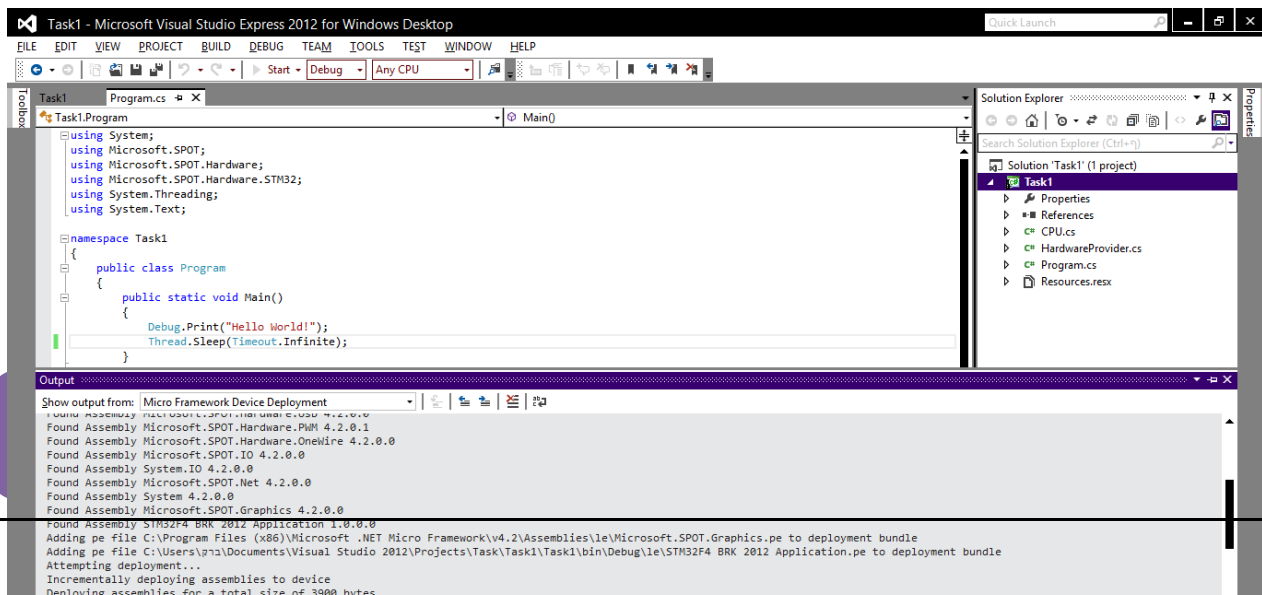
17. וודאו שבלשונית Net MicroFramework. ההגדרות של Deployment הם כמתואר באיור.

הערה: אם לאחר שבחרתם USB בשדה ה Transport אין את ה STM32F4 ברשימת ההתקנים שבשדה Device, וודאו שחיבורה USB מחובר לכרטיס פיתוח ולמחשב (על הכרטיס דולקים 2 לדים אדומים).

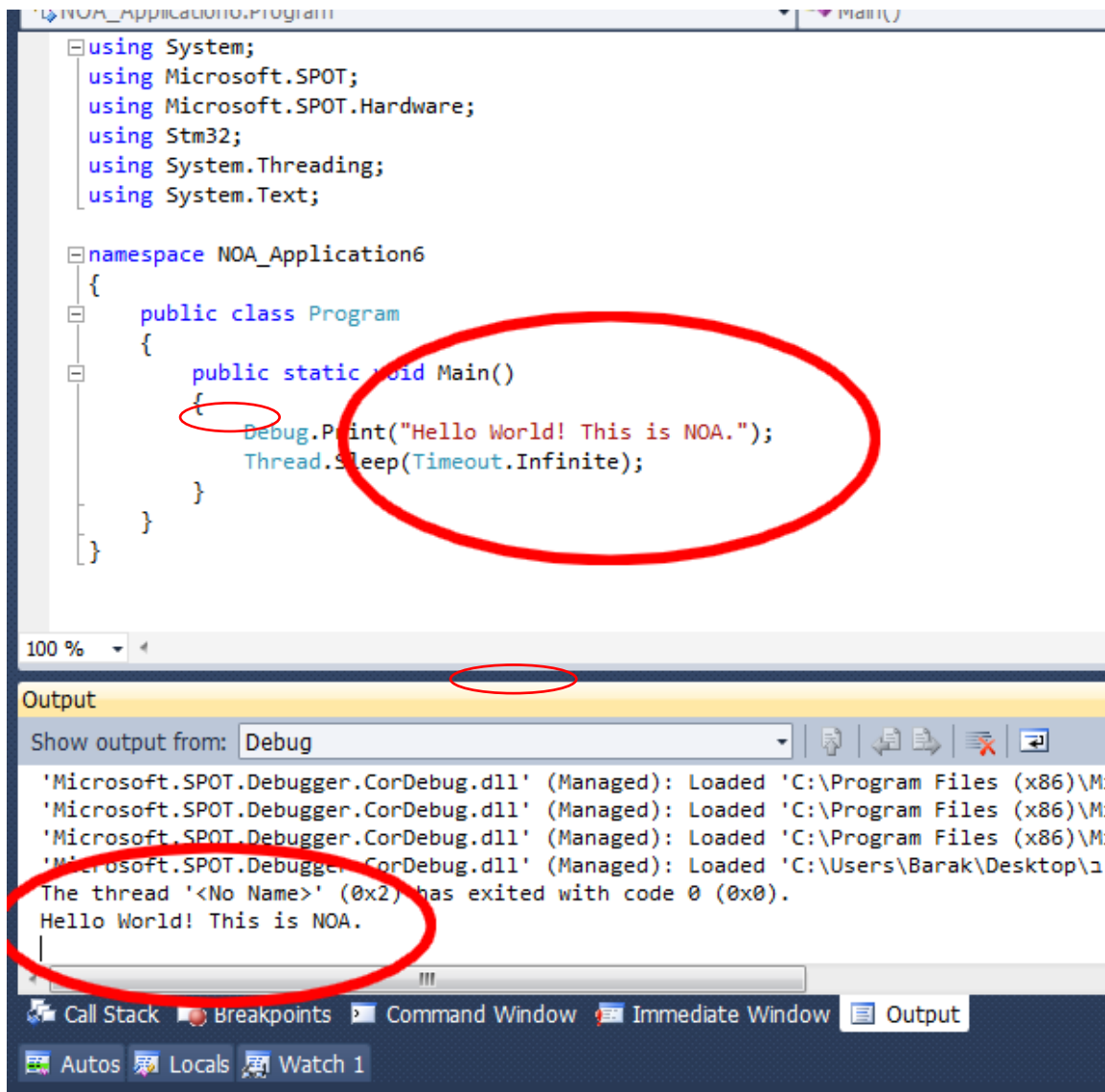
18. נרץ את הפרויקט ונצרוב אותו לבקר ע"י לחיצה על Start בשורת הפקודות



19. בחלקו התחתון של החלון תפתח חלונית Output המהווה פלט של המערכת



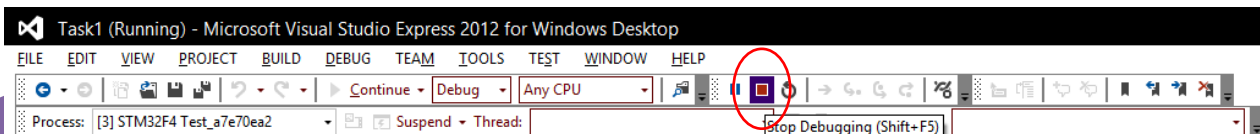
הערה: במידה והחלונית Output אינה מופיעה, ניתן להציגה מתוך תפריט ה View או ע"י הקשה בו זמנית במקלדת על הלחצנים Alt והסיפרה 2.



.20

בסיום התהליך, בחלונית Output יופיע הכיתוב Hello world ! This is NOA כפי שהקוד הראשוני בתוכנית דרש.

21. בסיום ההרצה, בכדי לערוך שינויים בתוכנית, יש לעצור את ההרצה ע"י הקשה על Stop Debbing בשורת הפקודות, כמתואר באיור



1-9

חברת BRK – חינוך טכנולוגי מתקדם

22. ניתן לצרוב את הבקר גם ע"י הלחיצה על לחצן F5.

23. ניתן גם לצרוב את הבקר מבלי אפשרות ה debugging (חיפוש ותיקון שגיאות בזמן הרצת התוכנית). צריבה זאת מהירה יותר וניתן לבצע אותה ע"י הקשה על שני הלחצנים יחד: Ctrl + F5.

2 צעדים ראשונים בתכנות בשפת c#

בפרק זה אנו נדבר על הפקודות הבסיסיות בשפת C#. כפי שניתן יהיה לירות, מרבית הפקודות דומות לשפת C הרגילה והמוכרת. ראשית, יש לפתוח פרויקט לעבודה עם הבקר שלנו. ניתן לראות את ההסבר המפורט על כך בנספח "פתיחת פרויקט חדש" המצורף בסוף דפי הסבר אלה.

2.1 טיפוסים משתנים

בשפת C# (בסביבת פיתוח Visual studio 2012) קיימים טיפוסים (סוגים) שונים מהם ניתן ליצור משתנים. כל סוג או הטיפוס של כל משתנה צריך להתאים למידע אותו נרצה לאכסן בו. כפי שישנן משאיות המיועדות לאכסון והובלה של מטענים שונים: נזל דליק, מים, מזון בקירור, מזון בהקפאה, שתיה קלה בבקבוקים ופחיות, אבנים במחצבה וכו', כך גם קיימים טיפוסים שונים עבור המשתנים. את תמציתם ניתן לראות בטבלאות הבאות:

2.1.1.1 שלמים

טיפוס	טווח ערכים	גודל
sbyte	-128 – 127	שלם, 8 סיביות עם סימן
byte	0 – 255	שלם, 8 סיביות ללא סימן
char	U+0000 to U+ffff	תו Unicode של 16 סיביות
short	-32,768 – 32,767	שלם, 16 סיביות עם סימן
ushort	0 – 65,535	שלם, 8 סיביות ללא סימן
int	-2,147,483,648 – 2,147,483,647	שלם, 32 סיביות עם סימן
uint	0 – 4,294,967,295	שלם, 32 סיביות ללא סימן
long	-9,223,372,036,854,775,808 – 9,223,372,036,854,775,807	שלם, 64 סיביות עם סימן
ulong	0 – 18,446,744,073,709,551,615	שלם, 64 סיביות ללא סימן

ספר קורס תכנות מיקרו-בקר באמצעות שפת C#

2.1.1.2

עם נקודה עשרונית

טיפוס	טווח ערכים	גודל	דיוק
float	$\pm 1.5e-45 - \pm 3.4e38$	32 סיביות	7 סיביות
double	$\pm 5.0e-324 - \pm 1.7e308$	64 סיביות	15-16 סיביות
decimal	$(-7.9 \times 10^{28} - 7.9 \times 10^{28}) / (10^{0-28})$	128 סיביות	28-29 סיביות

2.1.1.3 בוליאני

טיפוס	טווח ערכים	גודל
bool	false – true	8 סיביות

2.1.1.4 מחרוזת

טיפוס	טווח ערכים	גודל
string	מחרוזת של תווים	16 סיביות * מס' תווים במחרוזת + 20 * 8 סיביות

איך נדע באיזה טיפוס להשתמש?

בכדי לקבוע איזה טיפוס מתאים למשתנה, עלינו לדעת מה נרצה לאחסן בו. במידה ונרצה לשמור את שם המשתמש, נצטרך משתנה שיכול לשמור בתוכו טקסט. עבור מספר תעודת הזהות או מספר חשבון בנק נצטרך משתנה שיכול לאגור בתוכו מספרים שלמים וכן המסכורת החודשית של איש צוות תכנס למשתנה שיכול לאגור בתוכו מספרים לא שלמים, בעלי נקודה עשרונית. לפעמים, נפעיל פונקציות (על כך נרחיב בהמשך) המחזירות לנו את התוצאה בתוך משתנה מטיפוס מסויים ולכן עלינו לשמור אותו בסוג זה של טיפוס דווקא. כך למשל אם נרצה לקרוא את מצב הלחצן הממוקם ע"ג ערכת הפיתוח של הבקר, הפונקציה שאותה נפעיל בשביל זה תחזיר לנו משתנה מטיפוס bool ובתוכו: false – במידה והלחצן לא לחוץ ו true המידה והלחצן לחוץ. ניתן לסכם את האמור בטבלה הבאה:

ספר קורס תכנות מיקרו-בקר באמצעות שפת C#

חברת BRK – חינוך טכנולוגי מתקדם

דוגמאות	סוג המידע הנשמר	טיפוס המשתנה
שם פרטי ושם משפחה, הודעה איתה נרצה להציג למשתמש על מסך המחשב או תצוגת גביש נוזלי	טקסט	String
מספר תעודת זהות, מס' שניות שיש להמתין עד לביצוע פעולה מסויימת, מס' תלמידים בבית ספר	מספרים שלמים	int
המתח האנלוגי המתקבל במבוא של הבקר, תוצאות חישובים כדוגמת חילוק.	מספרים עם נקודה עשרונית	double
מצב של מבוא ספרתי של הבקר או מוצא ספרתי שלו, תוצאה של תנאי לוגי,	כן / לא	bool

דוגמא:

נגדיר משתנה מטיפוס `bool` (קיצור של Boolean) שיכול להכיל רק ערכים לוגיים: "1" לוגי = `true` או "0" לוגי = `false`.

לשם כך בין הסוגריים המסולסלים {} של הפונקציה הראשית `Main()` המתבצעת עם הרצת התוכנית, נכתוב את השורה הבאה:

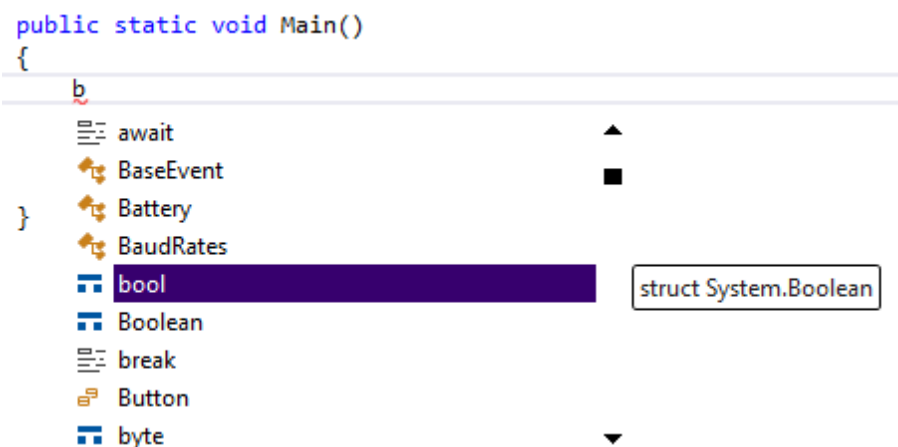
```
bool condition = false;
```



שימו לב שסביבת הפיתוח בתכנות בשפת C# עוזרת למתכנת לבחור את הפקודה הרצויה תוך שמירה על כתיבתה הנכון. עם התכלת ההקלדה של המילה `bool` תופיע הרשימה של הפקודות האפשרויות עם המתחילות באותיות שהוקלדו.

```
public static void Main()
{
    b

```



- await
- BaseEvent
- Battery
- BaudRates
- bool** (struct System.Boolean)
- Boolean
- break
- Button
- byte

ניתן לבחור את הפקודה הרצויה מתוך הרשימה ע"י מקשי החצים של המקלדת, או להקליק עליה עם העכבר. במידה והקלדתם מספיק אותיות והפקודה הרצויה כבר נבחרה עם השורה בצבע כחול, ניתן להקיש על המקש `Enter` או `Tab` שבמקלדת וסביבת הפיתוח תשלם עבורכם את הקלדת האותיות החסרות.

ניתן להגדיר את המשתנה ללא ערך התחלתי ולבצע השמה מאוחר יותר. כך למשל:

```
bool condition;
...
condition = false;
...
```

השמת מידע לא תואם את סוג המשתנה תגרום ברוב המקרים לשגיאת קומפילציה. כך, למשל, לא ניתן להכניס מספר 2.73 לתוך משתנה וטיפוס `int` או מספר 512 אל תוך משתנה `byte`.

לפעמים עולה צורך לבצע קטע קוד מסוים רק אם מתקיים תנאי מסוים. לשם כתיבת תוכנית מסוג זה נשתמש בפקודת if:

```

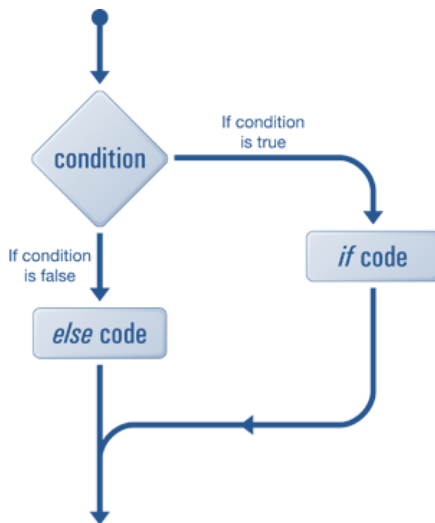
if (a>b)
{
    Debug.Print("The condition is true.");
}
else
{
    Debug.Print("The condition is false.");
}
    
```

מילה תנאי

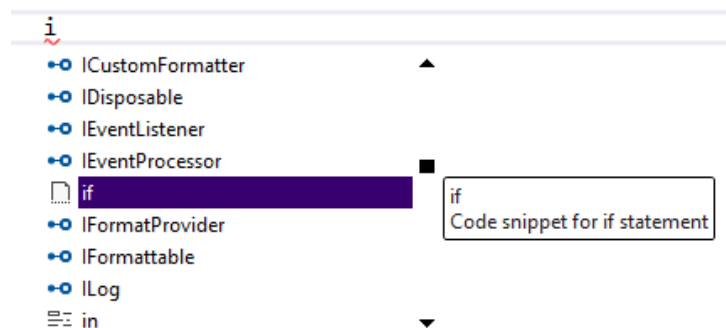
בין הסוגריים במסולסלות רשום הקוד שיתבצע כאשר התנאי הלוגי (בדוגמא זו: a>b) מתקיים

בין הסוגריים במסולסלות שלאחר else, המילה השמורה רשום הקוד שיתבצע כאשר התנאי הלוגי

להבנת אופן זרימת ביצוע התוכנית, ניתן להיעזר בתרשים הבא:



בשפת C# בשונה משפת C קיימים הרבה כלים המקלים על המתכנת את העבודה. כך גם כאן, מספיק להתחיל להקליד את הפקודה הרצויה וסביבת הפיתוח תציע רשימת פקודות התואמות התחלת ההקלדה של המשתמש. עם הקלדת האות i בלבד, תפתח רשימה של כל הפקודות המתחילות האות זו:



ניתן לבחור מתוך הרשימה את פקודת ה if ע"י חצי המקלדת או עם הקליק של העכבר. לאחר שהפקודה הרצויה סומנה, ניתן ללחוץ על הלחצן Tab בכדי שהיא תופיע על המסך. לחיצה מיידית נוספת על ה Tab תביא לפתיחת כל פקודת ה if והדגשת התנאי הלוגי אותו יש למלא ונמצא כ true בברירת מחדל:

```
if (true)
{
}
```

נחליף מילת ה true הגורמת לתנאי להתקיים תמיד, בתנאי לוגי כרצוננו.

התנאי הלוגי המופיע בתוך הסוגריים של if חייב להיות מורכב ממשתנים שהוגדרו, אופרטורים (>, <, = וכד') ומספרים. בכדי שהתנאי a>b יהיה חוקי, עלינו להגדיר לפניכן את המשתנים האלו. נגדיר ונאתחל אותם עם ערכים שונים. ניתן לעשות זאת ע"י הפקודות:

```
int a = 8;
int b = 5;
```

המשתנים a ו b לא חייבים להיות דווקא int, אלא טיפוסים נוספים כדוגמת char, long וכד'.

נציין בנפרד את הטיפוס string שלא קיים בשפת C. הטיפוס הוא למעשה כמין מחרוזת של תווים כפי שזה מוכר לנו בשפת C. ניתן להגדיר ולעבוד בקלות עם משתנים מסוג זה כאשר ישנו צורך לעבד טקסט.

האופרטורים החוקיים בהם ניתן להשתמש הם:

- < קטן מ...
- > גדול מ...
- <= קטן או שווה ל...
- >= גדול או שווה ל...
- == שווה ל...
- != לא שווה ל...
- || או לוגי
- && וגם לוגי
- () סדר פעולות

ישנם עוד מס' אופרטורים חוקיים כגון `is`, אך הדין עליהם יוכל להתקיים בפרקים מתקדמים יותר.

להלן מספר דוגמאות לשימוש בתנאים לוגיים:

```
if (a/2 >= c+10)
```

```
if (a*2 > b)
```

```
if ((a > b) && (a > 20))
```

```
if (++a != b--)
```

חשוב להקפיד שבתוך הסוגריים תמיד יהיה תנאי לוגי ולא פעולה אריתמטית, למשל.

כמוכן, ניתן להגדיר משתנה מטיפוס בוליאני (`bool`) ולהכניס בו ערך `true` או `false` לפני ביצוע התנאי ולהשתמש במשתנה זה לבדו או בצירופו עם משתנים אחרים. האפשרות הזאת יכולה להיות שימושית בעבודה עם תנאים מורכבים וכן העברת נתונים בין הפונקציות השונות בתוכנית.

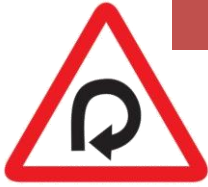
ניתן לראות זאת בדוגמא הבאה:

```
bool condition = false;

if (condition)
{
    Debug.Print("The condition is true.");
}
else
{
    Debug.Print("The condition is false.");
}
```

בנוסף לאפשרות לכתוב תנאי בודד, ניתן גם לשלב את התנאי בתוך תנאי אחר. לדוגמא:

```
if (condition)
{
    Debug.Print("The condition is true.");
    if (a<b)
    {
        Debug.Print("a<b");
    }
    else
    {
        Debug.Print("a>=b");
    }
}
else
{
    Debug.Print("The condition is false.");
}
```



הגדרה: לולאה היא קטע קוד שמתבצע מספר פעמים.
בשפת C# ישנן מספר פקודות בעזרתן ניתן ליצור לולאות מסוגים שונים.

FOR 2.3.1

זוהי פקודה בעזרתה ניתן ליצור לולאה. נשתמש בה ע"פ רוב כאשר מספר החזרות על קטע קוד רצוי ידוע מראש. למשל: בכדי להדפיס את שמכם על המסך 10 פעמים.

תחביר הפקודה הוא:



```
for (int i = 0; i < 10; i++)
{
    Debug.Print("Hello");
}
```

שימו לב כי בתוך הסוגריים העגולים ישנם 3 שדות המופרדים בניהם
עם ; (ולא עם :)

שימו לב כי אין ; בסוף שורת ה for

תנאי התחלה: בשדה זה של הפקודה נרשום את כל התנאים (במידה וקיימים) שמהם אנו רוצים להתחיל את הלולאה – תנאי התחלה. הקוד הרשום בשדה זה יתבצע רק פעם אחת בכניסה אל הלולאה. בדוגמא שלנו הגדרנו משתנה בשם i מסוג int והכנסנו בו ערך 0.

תנאי סיום: בשדה זה של הפקודה עלינו להגדיר את התנאי או התנאים בהם תסתיים ביצוע הלולאה והתוכנית תמשיך לרוץ ולבצע את הפקודות הבאות. בדוגמא שלנו, כל עוד $i < 10$ (תוכנו של המשתנה בשם i קטן מ 10), הלולאה תחזור על עצמה שוב, אך כאשר תנאי זה יופר (למשל, בתוך המשתנה i יהיה מספר

ספר קורס תכנות מיקרו-בקר באמצעות שפת C#

11) הלולאה תסתיים והתוכנית תבצעה את הפקודות הבאות. המחשב עובר לבדיקת תנאי היציאה מהלולאה מיד אחרי ביצוע הקוד בשדה של תנאי ההתחלה וכן בסיום כל לולאה.

בשדה זה של הפקודה, נרשום את הקוד שיתבצע בסיום כל לולאה. בדוגמא שלנו, נוסיף 1 לתוכן של המשתנה בשם i ($i++$). לאחר ביצוע קוד זה, המחשב יבדוק האם התנאי בו יש לבצע את הלולאה שוב (תנאי סיום), עדיין מתקיים.

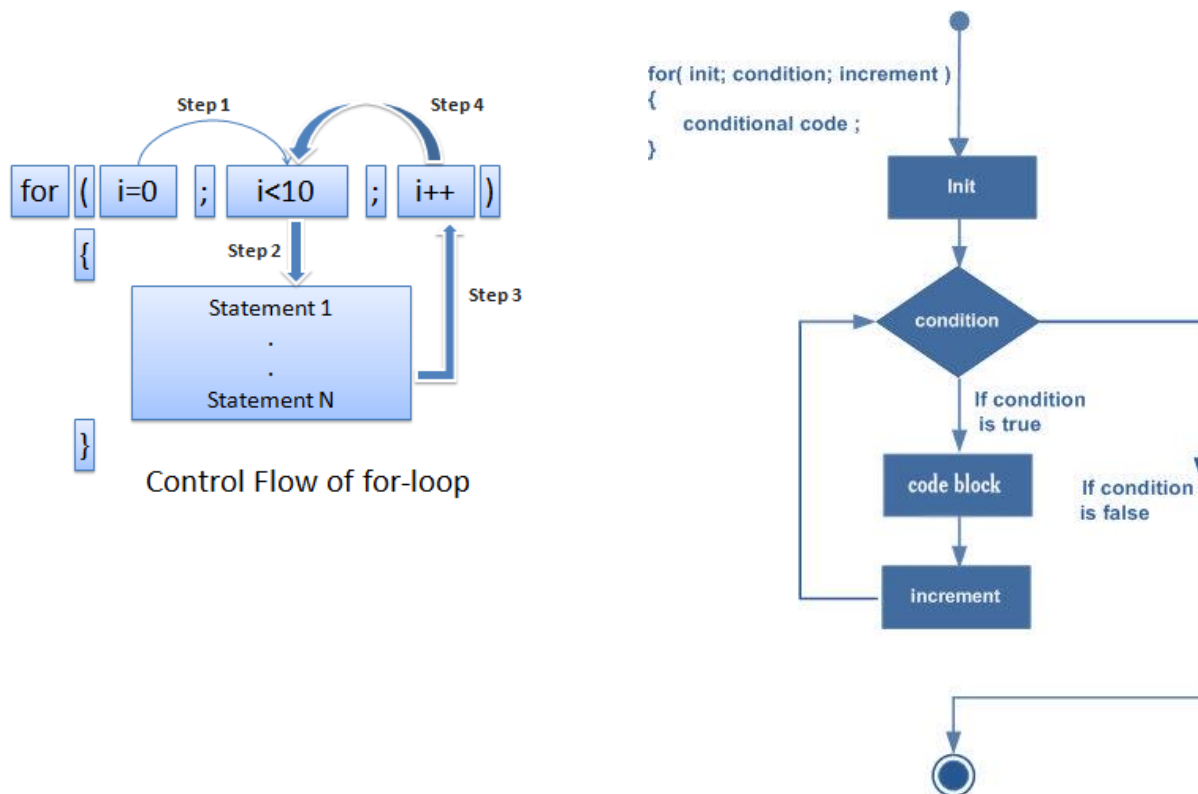
קידום:

במידת הצורך, ניתן להשאיר אחד או יותר משלושת השדות הנ"ל של הלולאה for ריקים.

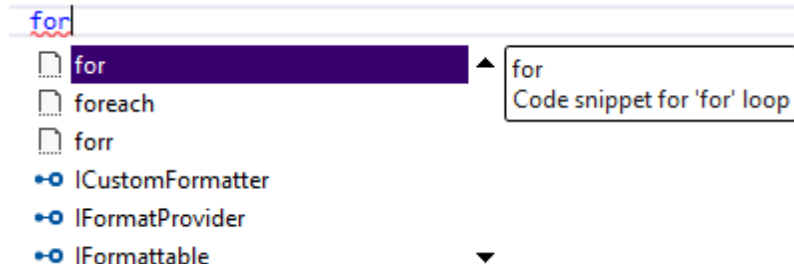
שימו לב: בסוף השורה של for אין ; !

כמו בפקודה if, גם כאן, הקטע קוד שיתבצע בלולאה הוא המוקף בסוגריים מסולסלים {}. גם כאן, במידה ומדובר בפקודה בודדת, לא חייבים לשים סוגרים אלה, אך כמתכון למניעת שגיאות מומלץ לשים סוגריים אלה גם במצב זה.

להסבר וויזואלי של לולאה זאת ניתן להיעזר בתרשימים הבאים:



כפי שהדגמנו זאת עם השימוש ב if, גם כאשר נתחיל להזין את הפקודה for בסביבת הפיתוח visual studio והיא תציעה רשימה של השלמות אפשריות. אם נקליד את הפקודה for עד שהיא תבחר מתוך הרשימה הנפתחת ונלחץ על לחצן ה Tab שבמקלדת, היא תושלם ותופיע על המסך.



בשלב זה, אם נקיש פעם נוספת על הלחצן Tab, סביבת הפיתוח תשלם את כל כתיבת התחביר של for בסיסי הבא:

```
for (int i = 0; i < length; i++)  
{  
  
}
```

מיד לאחר הופעת תבנית ה for ניתן לשנות את שם האינדקס למשל ל j או כל שם אחר, ללחוץ על לחצן ה Tab במקלדת ושם זה יופיע בשלושת המקומות בהם היה רשום i לפניכן. לאחר הלחיצה על מקש ה Tab ניתן לשנות את השדה length (כרגע זה שם משתנה שלא קיים בתוכנית שלנו) למספר כלשהו, למשל 5. נוסף פקודה כלשהיא בלולאה ונקבל:

```
for (int j = 0; j < 5; j++)  
{  
    Debug.Print("It's working well");  
}
```

כפי שהתכן ושמתם לב, בסביבת הפיתוח visual studio קיים קיצור נוסף forr. אם נקליד אותו ונלחץ פעמיים על מקש ה Tab שבמקלדת, נראה שנקבל תבנית של לולאת for בה האינדקס הולך וקטן (בשונה מהתבנית הרגילה בה האינדקס הולך וגדל עם ביצוע הלולאות). תבנית זאת יכולה להיות שימושית במקרים מסוימים וכדאי להכיר אותה.



לפני המעבר ללולאה הבאה, יש לציין כי בשפת C# (בשונה משפת C הרגילה) קיימת לולאה הדומה ל for, אך שונה ממנו במקצת והיא foreach. השימוש בלולאה זאת נעשה בקריאה מתוך אוסף כלשהו כדוגמת מערך וכד'. השימוש בה נוח, אך מאפשר קריאת נתונים בלבד מבלי היכולת לכתוב אותם לתוך האוסף (המערך). המעוניינים, יכולים לתרגל עבודה עם לולאה זאת באופן עצמאי או להיעזר באינטרנט.

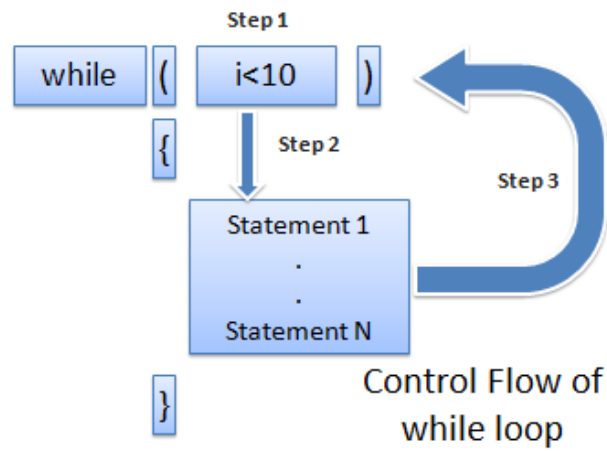
WHILE 2.3.2

בנוסף ללולאת for עליה דיברנו, בשפת C# קיימת גם לולאת while. לולאה זאת, בדומה ל for חוזרת על קטע קוד מסוים מספר פעמים, אלא שנוח יותר להשתמש בה במקרים בהם אין אנו יודעים מראש את מספר החזרות שנצטרך לבצע. למשל, התוכנה יכולה לבקש מהמשתמש להקיש על ספרה כלשהי במקלדת וכל עוד המשתמש מקיש על תווים אחרים, כדוגמת אותיות וסימנים שונים, לחזור על הבקשה להקיש דווקא ספרה.

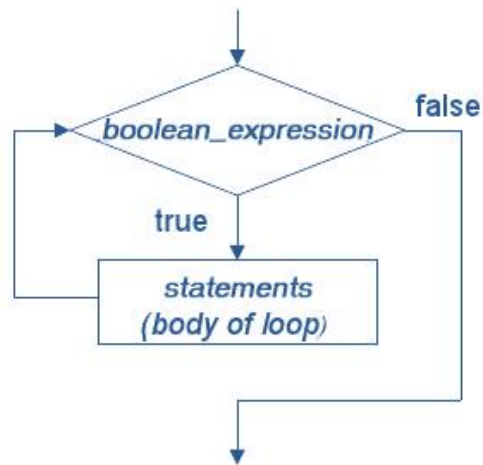
```
int a = 0;
while(a < b)
{
    Debug.Print(a.ToString());
    a++;
}
```

שימו לב שגם במקרה זה אין ; בסוף השורה של while !

את התחביר ובקרת הזרימה של הלולאה ניתן לראות באיור הבא:



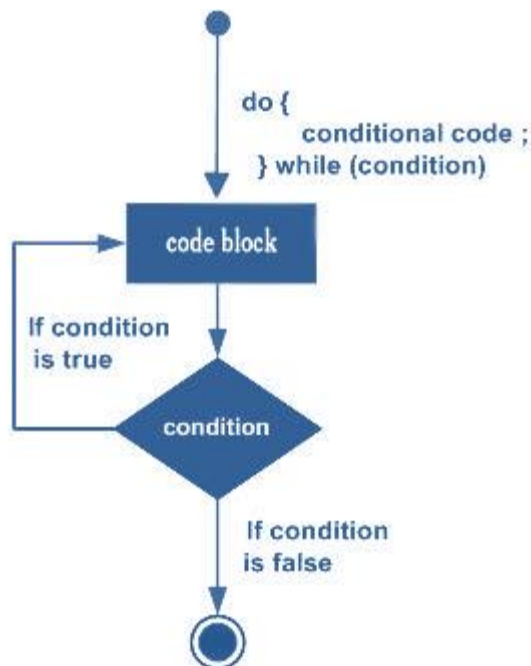
תרשים הזרימה של הלולאה היינו:



שימוש נפוץ נוסף של הלולאה הוא יצירת לולאה אינסופית. ניתן לעשות זאת גם ע"י לולאת for, אך נוח יותר ליישם אותה דווקא עם while. הלולאה האינסופית יכולה להיות שימושית למשל בסיום התוכנית. אין להשאיר את התוכנית לרוץ הלאה מאחר והתוצאה של זה היינה בלתי צפויה, אלא להכניסה ללולאה אינסופית שלא מבצעת כלום.

```
while (true);
```

DO WHILE 2.3.3



נוסף על הלולאה while בה קודם כל אנו בודקים האם התנאי מתקיים ורק לפי התוצאה מחליטים האם לבצע את הקטע קוד, קיימת לולאה דומה נוספת שקודם מבצעת את הקוד פעם אחת ולאחר מכן בודקת את התנאי בו צריך לחזור על הקוד פעם נוספת. לולאה זאת נקראת do while. תחביר הפקודה ותרשים הזרימה שלה נתונים האיור הבא:

גם ע"י לולאה זאת ניתן לממש את הלולאה האינסופית, אלא שהשימוש בה לצורך זה נפוץ פחות:

```
do  
{  
  
} while (true);
```

משפט זה משמש בלולאות לשם מעבר לתחילת הלולאה והמשך ביצוע התוכנית משם. בכדי להמחיש את השימוש במשפט זה נתבונן בדוגמא הבאה המדפיסה בחלונית ה output את המספרים הזוגיים בלבד בין 0 ו 100:

```
for (int i = 0; i < 100; i++)
{
    if (i%2 != 0)
    {
        continue;
    }
    Debug.Print(i.ToString());
}
```

ההוראה continue תתבצע רק כאשר המספר i מתחלק ב 2 עם שארית (כלומר לא מתחלק ב 2 ללא שארית). הוראה זאת תגרום למעבר לראש הלולאה וקידום האינדקס i ב 1 (i++). השורה השולחת נתונים לחלונית ה output לא תתבצע במקרה זה.

משפט זה מבצע סיום הלולאה ויציאה מידיית ממנה ללא כל תנאי. כך בדוגמא הבאה המדפיסה בחלונית ה output את כל המספרים מ 0 עד 100, נעשה שימוש בלולאה אינסופית, כאשר תנאי היציאה ממנה הוא שתוכן המשתנה a יהיה גדול מ 100:

```
int a = 0;
while(true)
{
    if (a > 100)
    {
        break;
    }
}
```

```

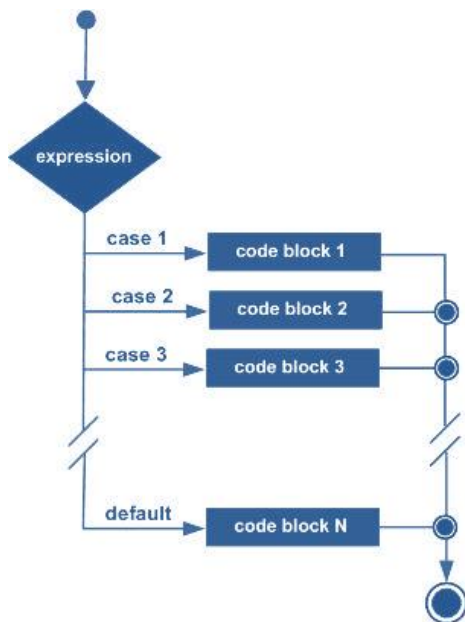
}
Debug.Print(a.ToString());
a++;
}

```

GOTO 2.3.6

בשפת C#, כמו גם בשפות שקדמו לה, ישנו משפט goto המורה לעבור למקום אחר בתוכנית המסומן בתווית ששמה מופיע אחרי ה goto ולהמשיך את ביצוע התוכנית משם. לדוגמא, הפקודה goto drive; תחפש בתוכנה את התווית בשם drive ותמשיך את ביצוע התוכנית ממנה. נציין, כי אחרי שם התווית יש לרשום את התווית : בכדי שהמהדר יתייחס למילה שרשומה לפני ה : כשם התווית. באופן כללי, מומלץ להימנע ככל האפשר מהשימוש במשפט ה goto מאחר והם שוברים את רצף ביצוע התוכנית ויכולים להוות מקור לתקלות.

SWITCH 2.3.7



פקודה זאת, כשמה, מבצעת מיתוג בין האפשרויות הקיימות. למשל, אם קיימים רק מספר מצומצם של אפשרויות ערך משתנה מסוים, כמו קומות בבקר מעלית, נוח להשתמש בפקודה זאת בכדי להגדיר מה על הבקר לעשות בכל מקרה נתון. בכדי למנוע מצבים בלתי צפויים, ניתן ומומלץ (אך לא חובה) להגדיר את מצב ברירת המחדל של הפקודה, כלומר המצב אליו תעבור התוכנה במקרה שבמשתנה המדובר קיים תוכן חורג שלא תואם אף אחד מהמקרים האפשריים. ניתן להיעזר בתרשים זרימה שמשמאל בכדי להבין את אופן ביצוע הפקודה.

```

switch (שם משתנה המכיל את המידע)
{
    case אפשרות 1 של תוכן המשתנה :
        קטע קוד שיתבצע
        break;
    case אפשרות 2 של תוכן המשתנה :
        קטע קוד שיתבצע
        break;
    ...
    default:
        קטע קוד שיתבצע
        break;
}

```

דוגמא:

```

switch (name)
{
    case "Avi":
        Debug.Print("You are menager");
        break;
    case "Itzik":
        Debug.Print("You are super user");
        break;
    case "Kobi":
        Debug.Print("You are registred user");
        break;
    default:
        Debug.Print("You are gest");
        break;
}

```

ניתן גם לשלב case ימים ו/או לקבוע מעברים בניהם:

```
switch (str)
{
    case "1":
    case "small":
        cost += 25;
        break;
    case "2":
    case "medium":
        cost += 25;
        goto case "1";
    case "3":
    case "large":
        cost += 50;
        goto case "1";
    default:
        Debug.Print("Invalid selection.");
        break;
}
```

