פרק 3 : ביצוע מותנה

יעדים

הבנת ביטויים בוליאניים, מושג התנאי, הצורך בביצוע מותנה, מבנה הבקרה , if ותפקידם בהקשר הכללי של משימה חישובית ומימושה; העמקת ההבנה של עבודה עם משתנים.

תכנים

1. אמת ושקר ב C .
2. ביטויי יחסים => , =< , > , < , != , == .
3. ביטויים לוגיים not ! , or || , and && וטבלאות האמת שלהם .
4. ביטויים בוליאניים פשוטים, מורכבים וסדר הפעולות הבוליאני .
5. ביצוע מותנה if , else , if else , switch
6. ביטויים טרנריים ternary operation ? x : b ; a = (b > c)
7. ביצוע תקינות קלט, מסננת קלט פשוטה הכוללת תנאי בלבד

סיכום שעות ההוראה: 14 שעות עיוניות ו- 6 שעות התנסותיות. סה"כ 20 שעות.

הערה : החומר המופיע בהמשך בין הסעיפים 1 עד 4 מוזכר גם בפרק 2 .

3.1 אמת ושקר – TRUE and FALSE וייצוגם במחשב

בתוכניות מחשב יש משפטים שהם ביטויים בוליאניים שתפקידם לבדוק האם תנאי או תנאים מסוימים מתקיימים. לדוגמה רוצים לדעת האם x>y (האם הערך שבמשתנה x גדול מהערך שבמשתנה y ?). תוצאת הביטוי יכולה להיות או FALSE (שקר) או TRUE (אמת) . הערך FALSE נרשם כ 0 והערך TRUE נרשם כ 1 .

כל מספר חיובי או שלילי השונה מ 0 הוא TRUE , רק המספר 0 הוא FALSE .

בהמשך נראה כיצד תוצאה של ביטוי לוגי קובעת מה יתבצע בהמשך התוכנית.

3.2 אופרטורים בוליאניים / אופרטורים להשוואה

אופרטורים בוליאניים הם אופרטורים שהתוצאה שלהם היא TRUE או FALSE (1 או 0). בדוגמאות שבטבלה נניח ש : a=10 b=20 .

האופרטור	משמעות	דוגמה
= =	האם שני צידי הביטוי שווים ?	a==b התוצאה FALSE (0)
=!	האם שני צידי הביטוי שונים (לא שווים). ?	a!=b התוצאה TRUE (1)
>	האם צד שמאל של הביטוי גדול מצד ימין שלו ?	a>b התוצאה FALSE (0)

<	האם צד שמאל של הביטוי קטן מצידו הימני	a<b התוצאה TRUE (1)
>=	האם צד שמאל של הביטוי גדול או שווה לצד ימין שלו ?	a>=b התוצאה FALSE (0)
<=	האם צד שמאל של הביטוי קטן או שווה לצידו הימני	a<=b התוצאה TRUE (1)

טבלה 1 : אופרטורים של השוואה

3.3 פעולות לוגיות

בפעולות לוגיות יש לזכור את הכלל : **כל ערך חיובי או שלילי השונה מ 0 הוא TRUE ורק 0 הוא FALSE** .

ישנן 3 פעולות לוגיות בסיסיות . NOT OR AND. בדוגמאות הבאות נניח ש a=25, b=0 :

3.4 ביטויים בוליאניים פשוטים, מורכבים וסדר הפעולות הבוליאני

פעולת NOT בעזרת הסימן !

הפעולה NOT הופכת את המצב מ TRUE ל FALSE ולהפך. דוגמאות :

!a → FALSE

כי הערך שיש ב a איננו 0 ולכן הוא TRUE ואחרי היפוך על ידי NOT יהיה בו FALSE שמבוטא על ידי 0 .

!b → TRUE

כי הערך שיש ב b הוא 0 שהוא FALSE ואחרי NOT יהיה בו TRUE המבוטא על ידי 1 .

פעולת OR (2 קווים ישרים)

הפעולה הלוגית OR אומרת שמספיק שאחד הביטויים יהיה TRUE כדי שהתוצאה תהיה TRUE.

ניתן גם לומר שכדי שהתוצאה תהיה FALSE צריך שכל הביטויים יהיו FALSE . טבלת האמת של פעולת OR נראית כך :

ביטוי ראשון	ביטוי שני	תוצאת פעולת OR
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	TRUE

טבלה 2 : טבלת אמת של פעולת OR

שוב נניח ש a=25, b=0 :

a || b → TRUE

(!a) || b → FALSE

!(a||b) → FALSE

פעולת AND מסומנת על ידי && (2 אמפרסנטים)

הפעולה הלוגית AND אומרת שחובה שכל הביטויים יהיה TRUE כדי שהתוצאה תהיה TRUE.

בדרך נוספת , ניתן גם לומר שמספיק שאחד הביטויים הוא FALSE כדי שתוצאת פעולת ה AND תהיה FALSE .

ביטוי ראשון	ביטוי שני	תוצאת פעולת AND
FALSE	FALSE	FALSE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	TRUE

טבלה 3 : טבלת אמת של פעולת AND

שוב נניח ש a=25, b=0 :

a || b → TRUE

(!a) && b → FALSE

a && (!b) → TRUE

(a || b) && a → TRUE

(a || b) && b → FALSE

2.9 אופרטורים העובדים על ביטים - Bitwise Operators

הפעולות הלוגיות בסעיף הקודם עבדו על ערך המשתנה כולו. 6 הפעולות הבאות עובדות על הביטים שבאופרנדים .

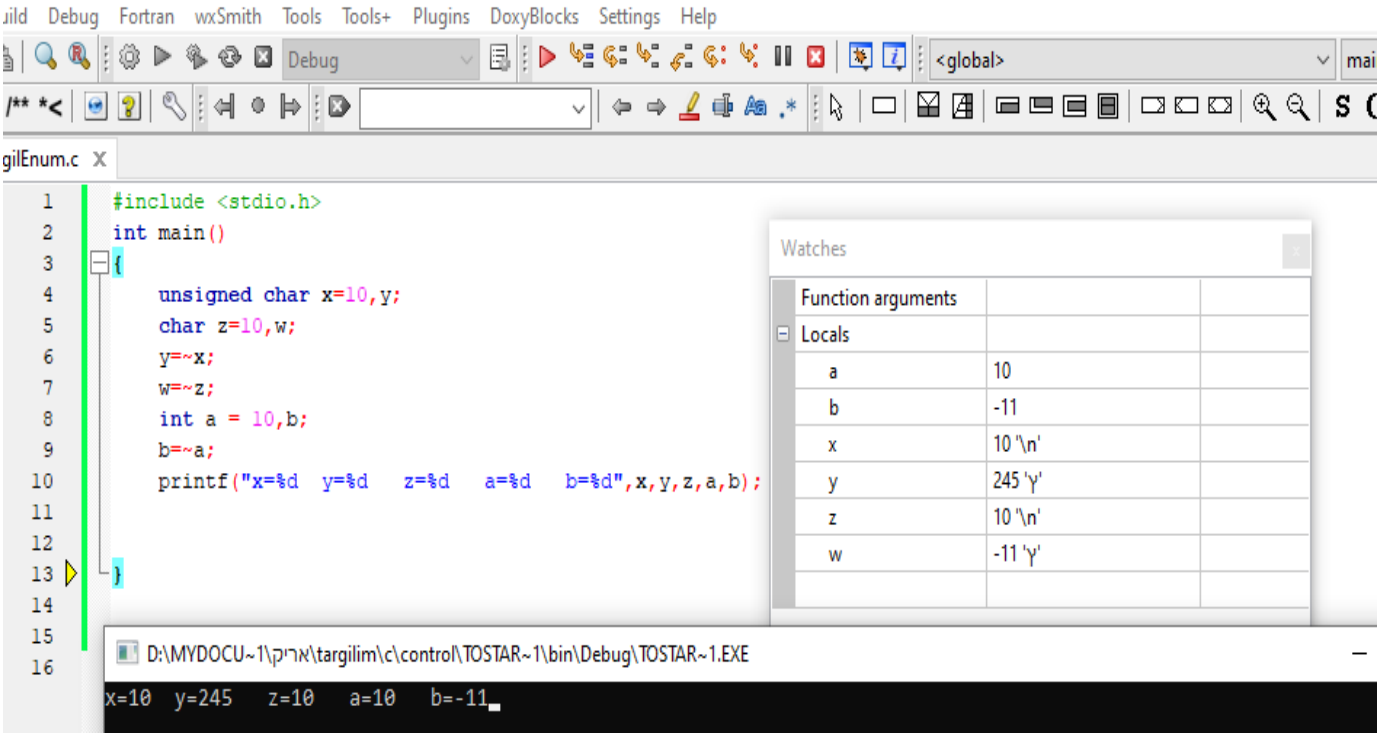
הסמל	הפעולה
~	היפוך הביטים
&	AND בין הביטים של שני האופרנדים
|	OR בין הביטים של שני האופרנדים
^	XOR בין הביטים של שני האופרנדים
>>	הזזה ימנית
<<	הזזה שמאלית

טבלה 4 : פעולות העובדות על ביטים

פעולת היפוך ביטים מסומנת עם התו ~ (טילדה)

פעולת ~ (הסימן נקרא טילדה) מבצעת היפוך של הביטים של האופרנד . כל ביט שהוא אפס מתהפך ל 1 וכל ביט של 1 מתהפך ל 0 . למעשה עושים כאן משלים ל 1 .

דוגמה : התוכנית הבאה הופכת את הביטים של המשתנים



איור 1 : פעולת היפוך ביטים

נסביר מדוע y קיבל את הערך 245.

x= 10 → 00001010 Binary

y = ~x → 11110101 Binary

הערך של y יהיה 245 כי הוא לא מסומן.

נסביר מדוע הערך של w הוא -11 :

z = 10 → 00001010

w = ~z → 11110101

היות והמספרים מסומנים וקיבלנו בביט ה MSB 1 זה אומר שהמספר שלילי. אם נבדוק מהו המספר השלילי על ידי הפיכה בעזרת המשלים ל 2 נראה שזה -11 .

פעולת AND על ביטים - הסימן &

האופרטור AND על ביטים מסומן עם אמפרסנד יחיד : & והוא מבצע פעולת AND בין הביטים התואמים בין שני האופרנדים להבדיל מפעולת AND עם && שמבצעת פעולת AND בין ערכם של האופרנדים (אם האופרנד איננו 0 אז הוא TRUE) ולא בין ביט אחד לביט שני.

טבלת האמת של פעולת AND נראית כך :

ביט 1	ביט 2	תוצאת ה AND בין 2 הביטים
0	0	0
0	1	0
1	0	0
1	1	1

טבלה 5 : פעולת AND על ביטים

לדוגמה :

```
  11001000
&
  10101101
----------
התוצאה :  10001000
```

פעולת OR על ביטים מסומנת בתו | (קו אנכי)

פעולת OR בין ביטים מסומנת עם קו אנכי אחד להבדיל מ || המבצעת פעולת OR בין הערך של האופרנד הראשון לשני. הפעולה | מבצעת OR בין כל 2 ביטים מתאימים בשני האופרנדים.

טבלת האמת של שער ה OR בין 2 ביטים :

ביט 1	ביט 2	תוצאת ה OR בין 2 הביטים
0	0	0
0	1	1
1	0	1
1	1	1

טבלה 6 : פעולת OR על ביטים

לדוגמה :

```
  11001000
|
  10101101
----------
התוצאה :  11101101
```

פעולת XOR על ביטים מסומנת עם התו ^

פעולת OR בין ביטים מסומנת ^ (עם חץ כלפי מעלה) ומבצעת פעולת XOR בין כל 2 ביטים מתאימים בשני האופרנדים.

טבלת האמת של שער ה XOR בין 2 ביטים :

ביט 1	ביט 2	תוצאת ה XOR בין 2 הביטים
0	0	0
0	1	1
1	0	1
1	1	0

טבלה 7 : פעולת XOR על ביטים

לדוגמה :

```
 11001000
^
 10101101
----------
 01100101
```

התוצאה : 01100101

הזזות ימינה או שמאלה

ניתן לבצע הזזה ימינה או שמאלה של כל אחד מהביטים של משתנה. כל ביט זז פעם אחת ימינה או שמאלה . בהזזה ימנית יש חשיבות האם המשתנה הוא מסומן או לא מסומן. בהזזה ימנית שומר ביט ה MSB שמציין האם המספר חיובי או שלילי על הסימן שלו . בהזזה שמאלית ביט ה MSB נופל ומימין נכנס 0 לביט ה LSB כפי שנראה בדוגמאות הבאות.

הזזה ימנית >>

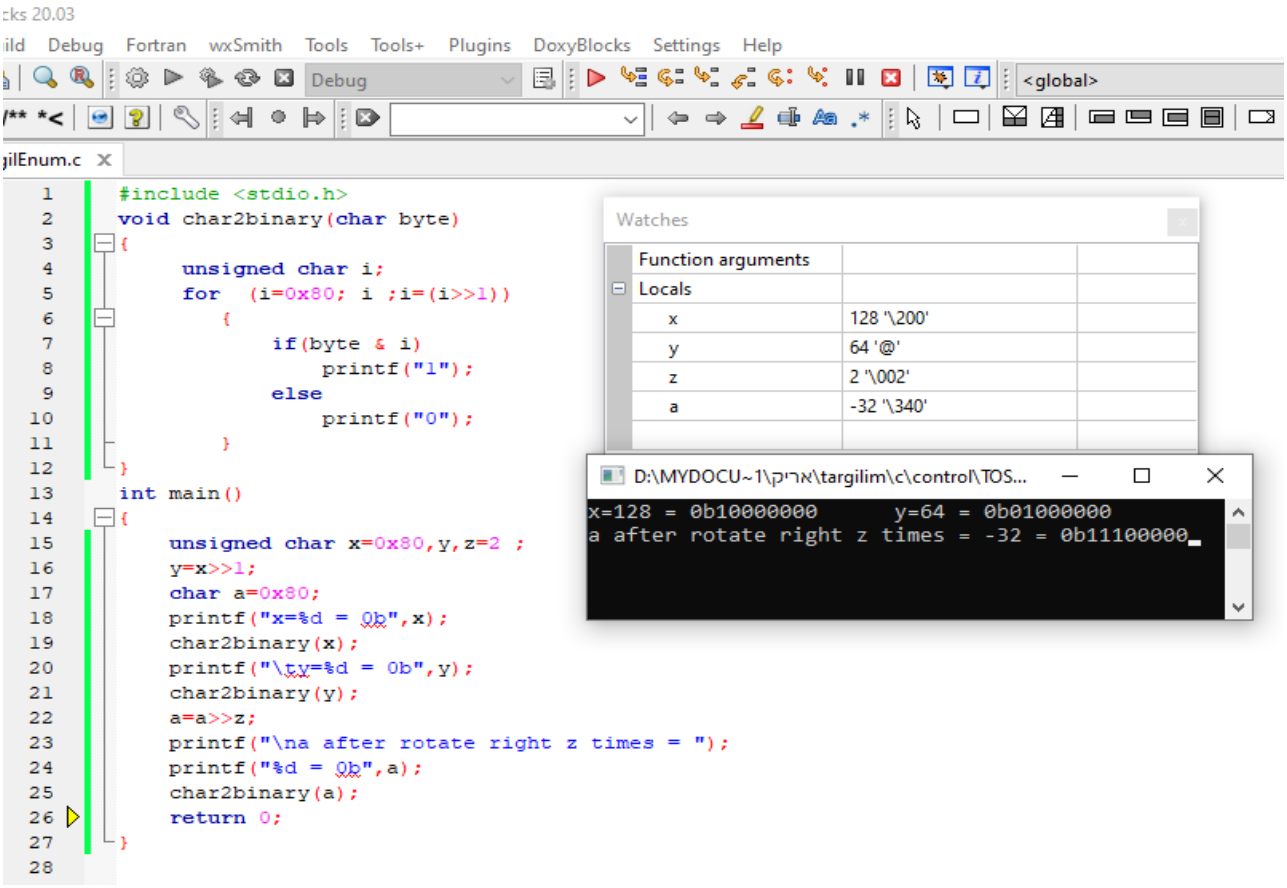
הזזה ימנית מזיזה ימינה את הביטים של האופרנד תוך **שמירה על ביט הסימן** שלו.

התחביר (syntax) של הפקודה :

משתנה >> כמות הפעמים ;

הפקודה אומרת להזיז ימינה את הביטים של המשתנה. **כמות הפעמים** היא כמה פעמים להזיז והיא יכולה להיות מספר (1 , 2 ,...) או הערך של משתנה כלשהו או ערך של ביטוי וכו' .

דוגמה לתוכנית שמזיזה ימינה עבור מספר לא מסומן ועבור מספר מסומן



איור 2 : הזזה ימנית עבור מספר לא מסומן ועבור מספר מסומן

ניתן לומר שהזזה ימנית מחלקת את הערך של המשתנה ב 2 .

הפונקציה ( )char2binary מקבלת מספר מטיפוס תווי char ומדפיסה את הערך הבינארי שלו למסך .

הפונקציה מקבלת משתנה מטיפוס תווי הנכנס למשתנה byte .

בפונקציה מפעילים לולאת for . הכניסה ללולאה היא i=0x80 . התנאי של הלולאה הוא כל עוד i הוא TRUE (התנאי i אומר i==TRUE) . הלולאה מתבצעת 8 פעמים כי בסיום כל איטרציה (מעבר של התוכנה על שורת ההוראות של הלולאה) מזיזים את ערכו של i פעם אחת ימינה.

בהתחלה הפונקציה עושה פעולת AND של ביטים בין byte לבין הערך של i הראשוני שהוא : 0x80 = 0b10000000 .

למעשה בודקים האם בביט ה MSB של byte יש 0 או 1 ומדפיסים למסך. מסובבים בלולאת for של 8 פעמים את הערך של i (עד שערכו אחרי 8 פעמים של הזזה הוא 0). בפעם הבאה i=0x40 = 01000000 ובודקים את ערכו של הביט שמימין לביט ה MSB ומדפיסים את ערכו – 0 או 1 – וכך הלאה.

הזזה שמאלית <<

הזזה שמאלית מזיזה שמאלה את הביטים של האופרנד .

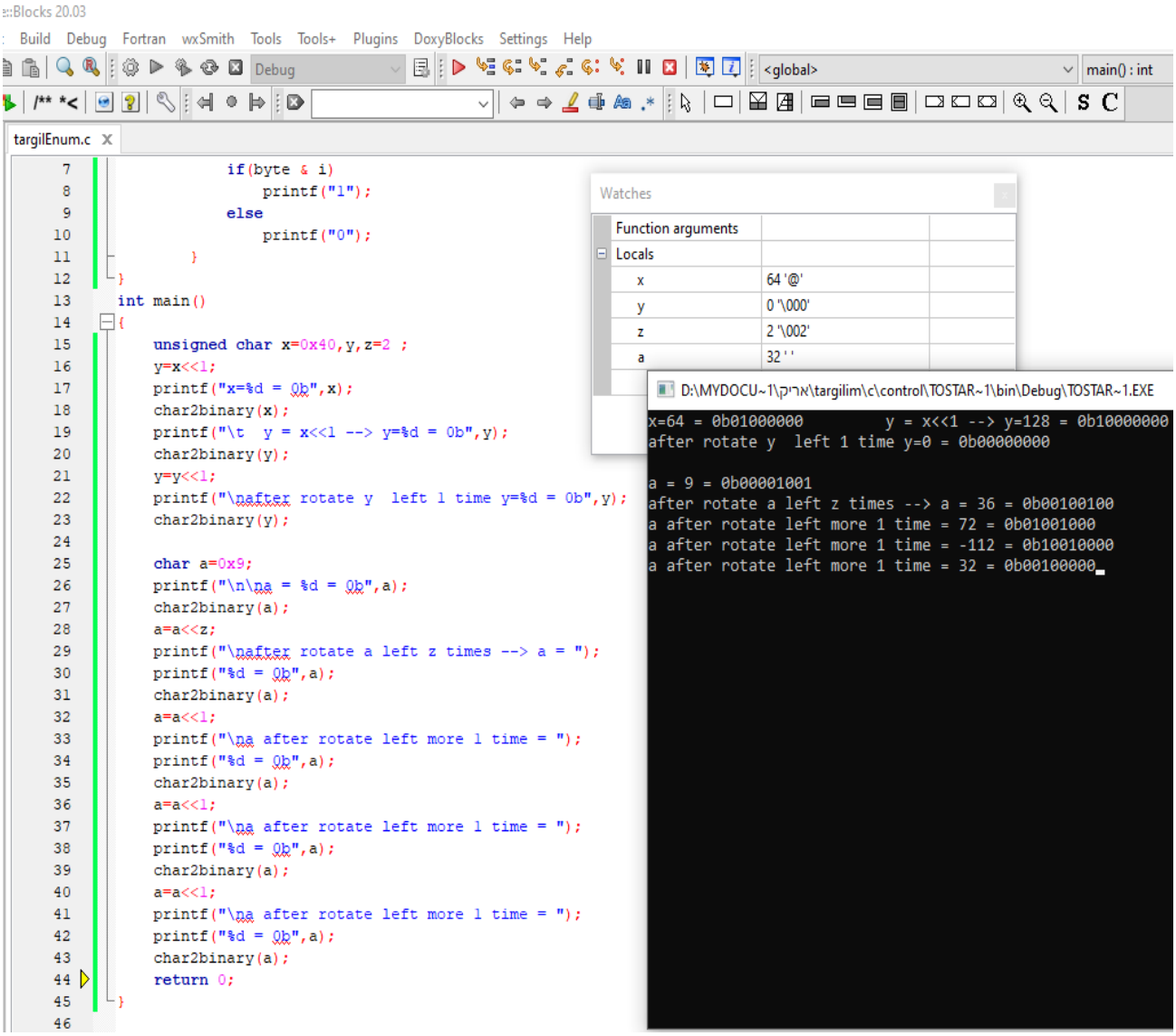
הזזה ימנית מזיזה שמאלה את הביטים של האופרנד ללא **שמירה על ביט הסימן** שלו.

התחביר (syntax) של הפקודה :

משתנה >> כמות הפעמים ;

הפקודה אומרת להזיז שמאלה את הביטים של המשתנה. **כמות הפעמים** היא כמה פעמים להזיז והיא יכולה להיות מספר (1 , 2 ,...) או הערך של משתנה כלשהו או ערך של ביטוי וכו' .

האיור הבא מתאר הזזה שמאלית על מספר לא מסומן ועל מספר מסומן.



איור 3 : הזזה שמאלית של מספר לא מסומן ושל מספר מסומן.

באיור רואים שהמספר זז שמאלה וגדל פי 2 . במספר מסומן כאשר מזיזים שמאלה ויש 1 ב MSB המספר הופך לשלילי.

כדאי לשים לב ששתי שורות לפני הסוף המספר הוא שלילי ואחרי הזזה נוספת הוא שוב חיובי.

קידום והפחתה ב 1 - INCREMENT/DECREMENT

האופרטור ++ מקדם ב 1 את ערכו של משתנה או ביטוי והאופרטור – מפחית 1 מערכו של משתנה או ביטוי.

דוגמאות :

```
short x=10 ,y;
y=x++;
```

ב משפט y=x++; מתבצעות הפעולות הבאות : **א.** y=x כלומר y=10 **ב.** ואחר כך מתבצע x = x+1 ו x יהיה 11.

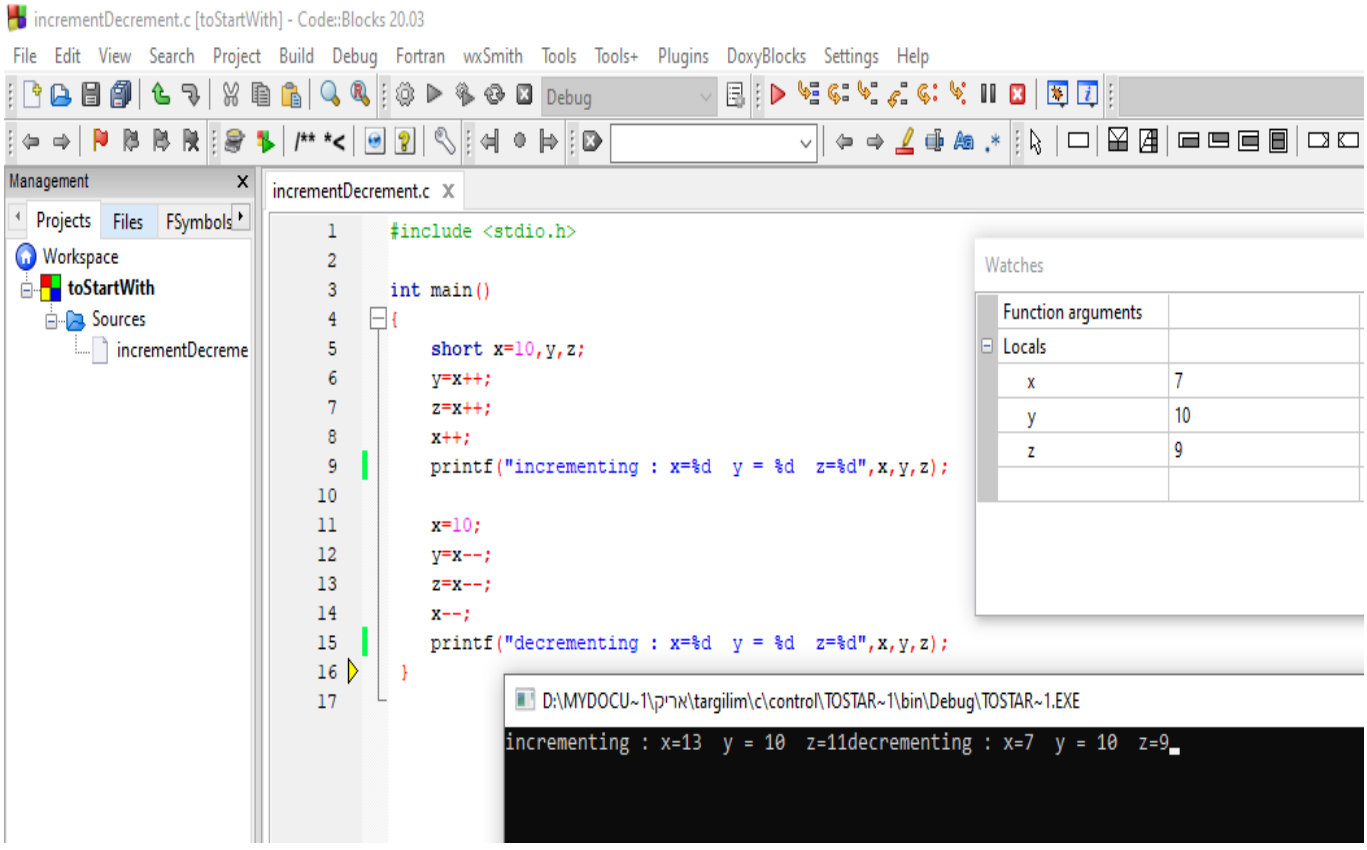
מצב כזה נקרא **קידום מאוחר** ובסיום המשפט : x=11 y=10 . בדוגמה הבאה מתבצע **קידום מוקדם**:

```
short x=10 ,y;
y=++x;
```

במקרה של קידום מוקדם בו ה ++ לפני x מתבצע : **א.** x=x+1 = 10+1=11 **ב.** y=x . כאן y=11 x=11 .

דבר דומה קורה עבור חיסור מאוחר y=x--; ועבור חיסור מוקדם y=--x; .

נראה דוגמה :



איור 4 : דוגמה לפעולות קידום ואיחור

שורות 6 עד 9 מבצעות הגדלה ב 1 ואילו שורות 12 עד 14 מבצעות הפחתה ב 1 .

טבלה המסכמת את האופרטורים השונים:

כפי שהסברנו , אופרטור הוא פעולה חשבונית או לוגית. הפעולה מתבצעת על אופרנדים שהם משתנים או קבועים. לדוגמה : c=a+10; . ה + הוא האופרטור ו a ו 10 הם האופרנדים .

הטבלה הבאה מסכמת את סוגי האופרטורים שהזכרנו :

אופרטור מתמטי	תיאור	אופרטור להשוואה	תיאור (התשובה היא 0 או 1 – TRUE / FALSE)
+	חיבור	= =	האם שווה (**שני** סימני שווה)
-	חיסור	! =	לא שווה
*	כפל	>	גדול מ-
/	חילוק	<	קטן מ-
%	שארית (מודולו)	> =	גדול או שווה ל-
		< =	קטן או שווה ל-
אופרטור לוגי	**תיאור : עבודה על בתים או מילים (התשובה 0 או 1)**	**אופרטורים העובדים על ביטים**	**תיאור**
!	היפוך	**!**	היפוך הביט
&&	וגם (and)	&	AND בין הביטים של שני האופרנדים
||	או (or)	|	OR בין הביטים של שני האופרנדים
		^	XOR בין הביטים של שני האופרנדים
++	קידום ב 1	>>	הזזה ימנית
- -	הפחתה ב 1	<<	הזזה שמאלית

טבלה 8 : אופרטורים מתמטיים לוגיים והשוואה.

קדימות – עדיפות – PRIORITY

כאשר מבצעים פעולות חשבוניות או לוגיות , קיימת עדיפות של סדר פעולות. העדיפות נתונה בטבלה הבאה. רואים שפעולות חשבוניות בעדיפות גבוהה מלוגיות. לאופרטורים - + * בצורה האונרית (פעולה על אופרנד אחד כמו x++) יש עדיפות על השימוש בהם בצורה הבינארית (פעולות על שני אופרנדים).

סדר ביצוע	אופרטורים (פעולות חשבוניות/לוגיות/השוואה)
משמאל לימין	( ) [ ] . -> n++ n--
מימין לשמאל	++ n -- n ! ~ sizeof (type) + - & *
משמאל לימין	* / %
משמאל לימין	+ -
משמאל לימין	<< >>
משמאל לימין	< <= > >=
משמאל לימין	== !=
משמאל לימין	&
משמאל לימין	^
משמאל לימין	|
משמאל לימין	&&
משמאל לימין	||
מימין לשמאל	? :
מימין לשמאל	= += *= <<= >>= /= %=
משמאל לימין	,

טבלה 9 : סדר עדיפות בפעולות חשבוניות ולוגיות.

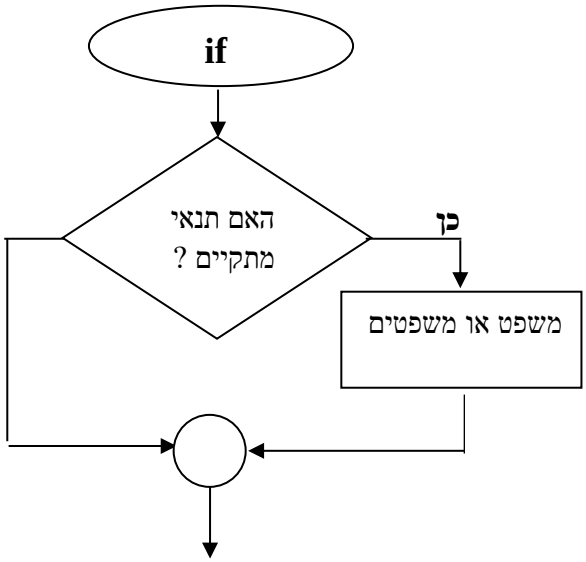
3.5. ביצוע מותנה if , else , if else , switch

משפטי תנאי

משפט תנאי הוא משפט שבו יש תנאי. אם התנאי מתקיים מבצעים משפט/משפטים מסוימים. משפט תנאי מחזיר או TRUE (נחשב כ 1) או FALSE (נחשב כ 0).

תבנית הפקודה if :

במשפט if בודקים האם תנאי מתקיים. אם כן מבצעים משפט / משפטים. אם התנאי לא מתקיים לא מבצעים שום דבר.



איור 5 : תרשים זרימה של תנאי if

התחביר :

```
if  ( תנאי )
  משפט ;
```

אם התנאי מתקיים עוברים למשפט הבא. אם התנאי לא מתקיים עוברים לשורה אחרי המשפט.

אם יש מספר פקודות לביצוע תוחמים אותן בעזרת סוגריים מסולסלים:

```
if ( תנאי )
  {
     משפט 1 ;
     משפט 2 ;
     ……..
  }
```

לדוגמה:

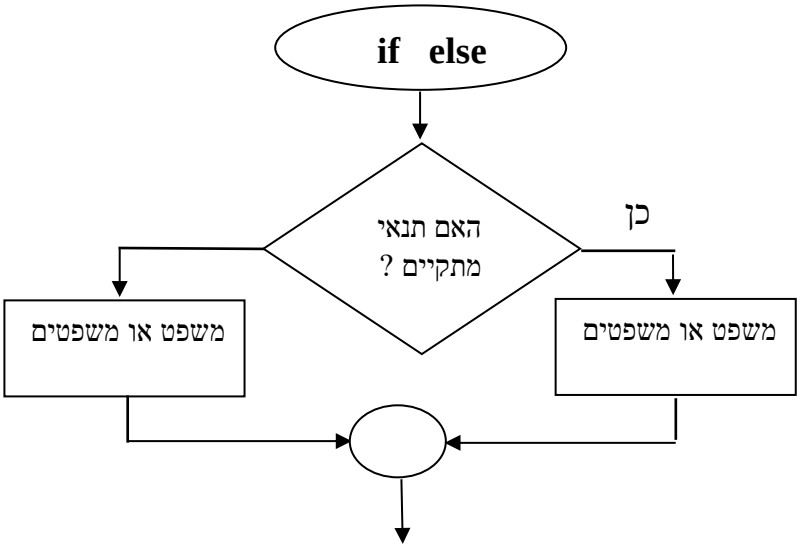
```
if ( a>b)

   printf("a is bigger than b");
```

התנאי בדק האם a>b . אם כן מדפיסים ש a גדול מ b . אם b יותר גדול או שווה ל a ממשיכים הלאה בתוכנית.

if else

משפט if else הוא משפט שבו בודקים האם תנאי מתקיים. אם כן מבצעים משפט/משפטים. אם התנאי לא מתקיים מבצעים משפט/משפטים אחרים.



איור 6 : תרשים זרימה של if else

התחביר :

```
if (תנאי)
    משפט ;
else
    משפט ;
```

דוגמה:

```
if ( a>b)
    printf("a is bigger than b")
else
        printf("b is bigger or equal to a");
```

אם ישנן מספר פקודות לביצוע :

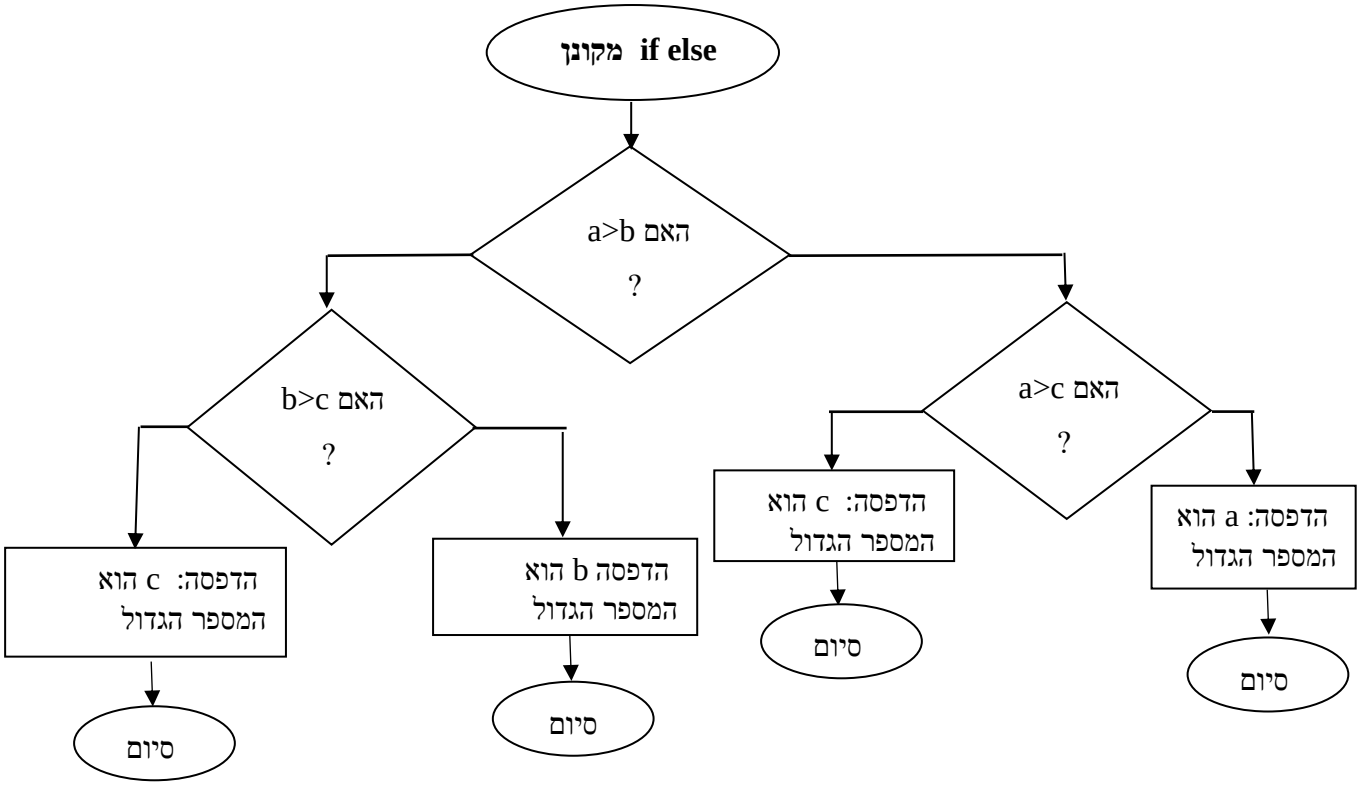
```
if <תנאי>
    {
       משפט 1 ;
        משפט 2 ;
        ……..
    }
  else
    משפט ;
```

if-else מקונן

המילה מקונן באה מהמילה קן (של ציפורים). זהו משפט אם מספר if ומספר else .

דוגמה : נניח שיש לנו 3 מספרים לא שווים הנמצאים במשתנים a, b ו c . נמצא מיהו המספר הגדול משלושתם.

הערה : בנקודות החלטה: ימינה-כן, שמאלה-לא.



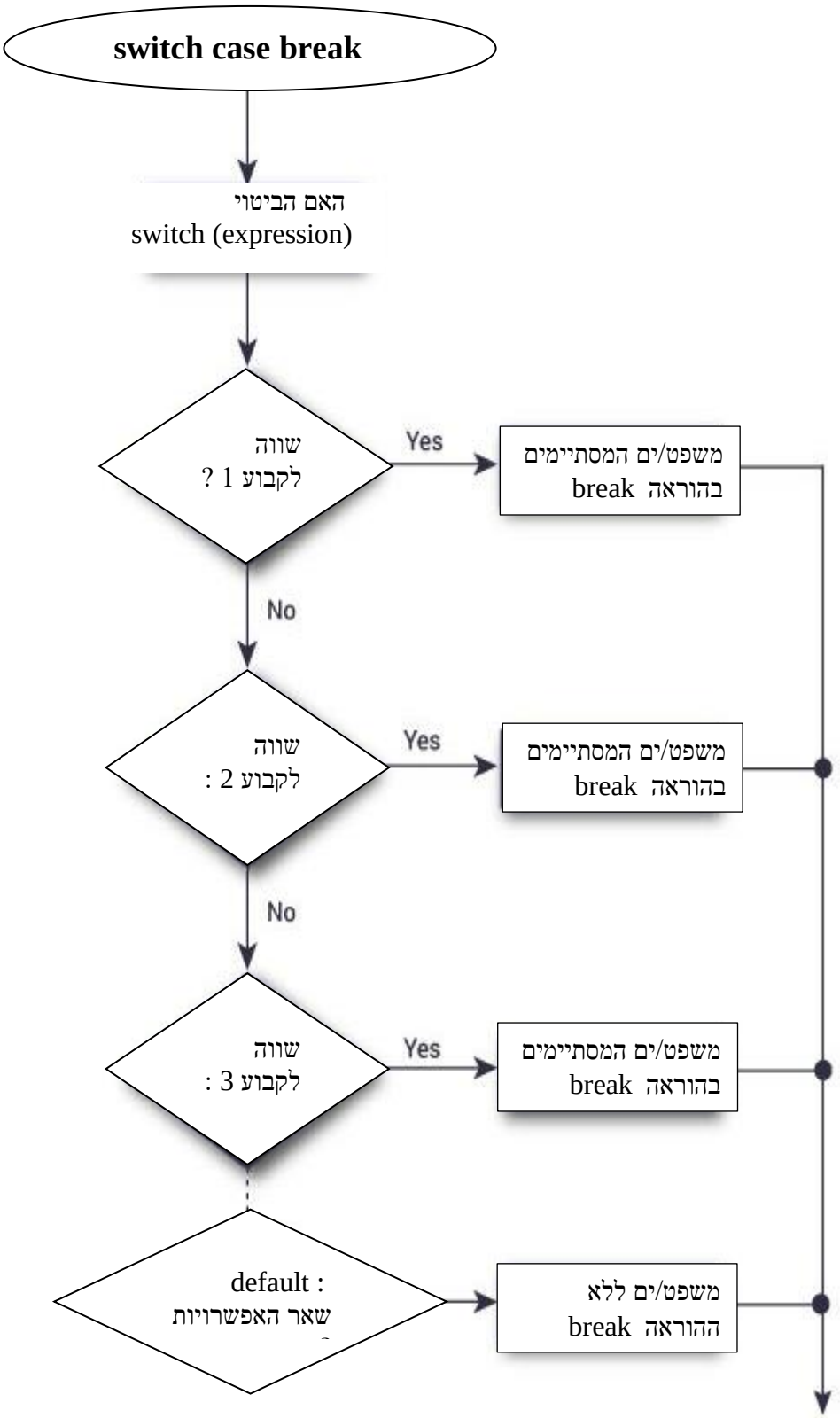
איור 7 : דוגמה לתרשים זרימה של if else מקונן של מציאת מספר גדול מבין 3 מספרים.

ובשפת C :

```
if (a>b)
  if(a>c)
        printf("a is the biggest number);
  else
        printf("c is the biggest number);
  else if (b>c)
       printf("b is the biggest number);
  else
      printf("c is the biggest number);
```

switch - case - break_

כאשר יש הרבה משפטי if else מקוננים ניתן להשתמש במשפט הצבה מותנה switch case break שמחליף את כל ה else if .



איור 8 : תרשים זרימה switch case break

התחביר :

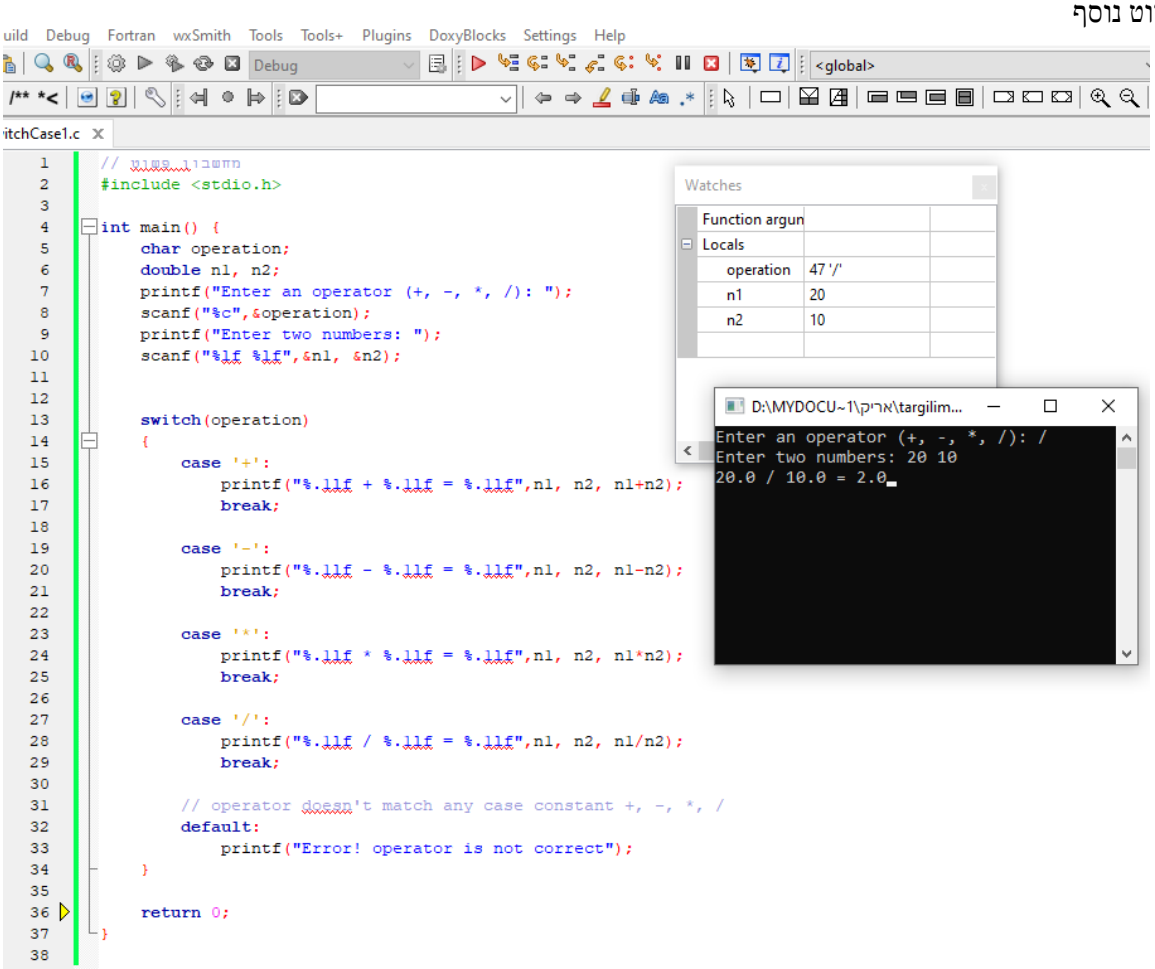
```
switch ( ביטוי )
  {
  case  1 קבוע :
    משפט ;
    משפט ;
    ……..
  break ;
  case  2 קבוע :
    משפט ;
    משפט ;
    ……..
  break ;
  case  3 קבוע :
    משפט ;
    משפט ;
    ……..
  break ;
      ⋮
      ⋮
  default :
   משפט ;
   משפט ;
  }
```

דוגמה 1: התוכנית הבאה היא של מחשבון פשוט המבצע פעולות חיבור חיסור כפל וחילוק. התוכנית מגדירה 2 משתנים a b , מאתחלת אותם , ומדפיסה למסך תפריט שבו המשתמש צריך לבחור איזו פעולה מתמטית הוא רוצה לבצע על a ו b ובהתאמה מבצע את הבחירה ומדפיס את התוצאה למסך.

```
int a=10 , b=20, choice;          // הגדרה של 3 משתנים מטיפוס שלם ואתחול של 2 המשתנים הראשונים
printf("\t*** MENU***\n1.Add\n2.Subtract\n3.Multiply\n4.Divide");    // הדפסת מסך תפריט
printf("\nPlease enter your choice (1 – 4 ): ");        // הדפסה : נא הכנס בחירתך
scanf("%d",&choice);   // קליטה של מספר
switch(choice):  // ? האם choice שווה לקבוע הבא
{                                                  //        ***MENU***
```

```
case 1:    // ? 1 שווה ל choice האם                    //  1. Add
   printf("a+b = %d+%d = %d",a,b,a+b);          //  2.Subtract
break;                                                  //  3.Multiply
case 2: // ? 2 שווה ל choice האם                  //  4. Divide
   printf("a-b = %d -%d = %d",a,b,a-b);          //  Please enter your choice (1 – 4) :
break;                                                  //  a-b = 10 – 20 = -10
case 3:  // ?  3 שווה ל choice האם
   printf("a*b = %d*%d = %d",a,b,a*b);
break;
case 4:   // ?  4 שווה ל choice האם
   printf("a/b = %d/%d = %f",a,b,(float)a/b);   // 10/20 = 0.500
break;
                                  //          casting
default :
   printf("Wrong Choice");
}
```

דוגמה 2 : מחשבון פשוט נוסף



איור 9 : מחשבון נוסף עם קליטת הפעולה כערך תווי

בדוגמה זו קלטנו את הפעולה הרצויה למשתנה operation שהוגדר כמשתנה תווי ולכן במשפט ההשוואה שבוצעה היא עם ערך האסקי הרצוי ולא כמו בדוגמה הקודמת עם הערכים 1 עד 4 . אנחנו ביקשנו פעולת חלוקה שסימנו כ / (אלכסון הפוך) שערך האסקי שלה הוא 47 עשרוני .

6. משפט הצבה מותנה - ביטויים טרנריים ternary operation

תחביר **(תנאי) ? משפט 1 : משפט 2 ;**

אם התנאי מתקיים מתבצע משפט 1 . אם התנאי לא מתקיים מתבצע משפט 2 .

דוגמה :

```
(a>b) ?   printf("a is bigger than b") :   printf ("b is bigger or equal to a")
```

משפט goto

המשפט goto מעביר את התוכנית אל קטע תוכנית אחר שנמצא בתווית label שרשמנו .

תחביר : goto < תווית > ;

דוגמא 1:

```
goto sof ;

        ⋮

sof:

        ⋮
```

דוגמא 2:

```c
// התוכנית מחשבת ומדפיסה את סכום המספרים החיוביים שהכניס המשתמש ואת הממוצע שלהם
// jump כאשר המשתמש מכניס מספר שלילי התוכנית קופצת לתווית
#include <stdio.h>

int main() {

    const int maxInput = 100;
    int i;
    double number, average, sum = 0.0;

    for (i = 1; i <= maxInput; ++i) {
      printf("%d. Enter a number: ", i);
      scanf("%lf", &number);

      // go to jump if the user enters a negative number
      if (number < 0.0) {
         goto jump;
      }
      sum += number;
    }

jump:
    average = sum / (i - 1);
    printf("Sum = %.2f\n", sum);
    printf("Average = %.2f", average);

    return 0;
}
```

```
1. Enter a number: 10
2. Enter a number: 20
3. Enter a number: 30
4. Enter a number: 40
5. Enter a number: -1
Sum = 100.00
Average = 25.00
```

איור 10 : דוגמה להוראה goto

בדוגמה רואים מימין את המספרים שהמשתמש הקיש . המספר ה 5 המוקש הוא שלילי ואז בשורה 16 בודקים ורואים שהמספר שלילי ושורה 17 מעבירה אותנו לתווית jump שם מתבצע חישוב והדפסה של הסכום והממוצע.

סיבות להימנע משימוש במשפט goto :

השימוש במשפט goto עלול להוביל לקוד שהוא buggy , לא מבני , וקשה לעקוב אחריו . כמו כן, הצהרת goto מאפשרת לעשות "דברים רעים" כגון לקפוץ מחוץ לתחום. עם זאת ההוראה יכולה להיות שימושית כדי לשבור לולאות מקוננות.

האם להשתמש ב goto ??

אם חושבים שההוראה מפשטת את התוכנית אז **אולי** כדאי להשתמש בה. אם כי השימוש בה צריך להיות רק במקרים מיוחדים וניתן ליצור כל תוכנית ב C ללא השימוש בהוראה זו.

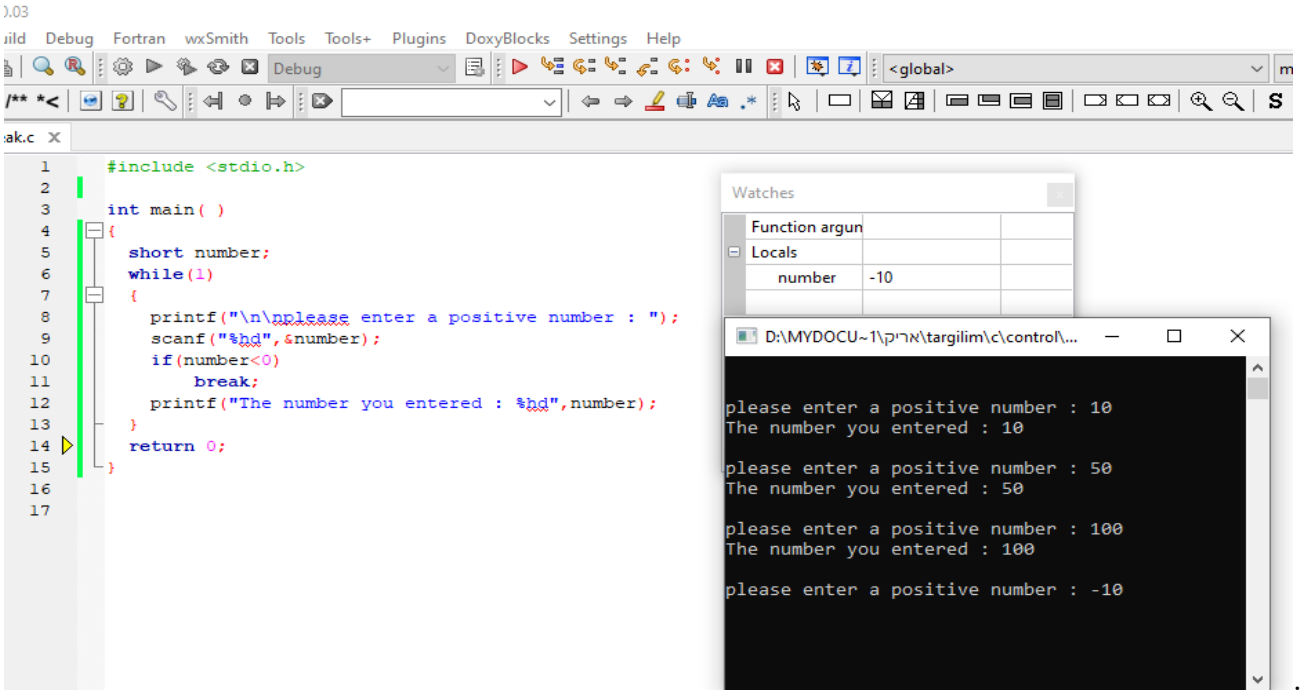
Bjarne Stroustrup היוצר של שפת C++ אמר : " העובדה ש goto יכולה לעשות כל דבר היא בדיוק הסיבה שאנחנו לא משתמשים בה" .....

משפט break

יציאה מתוך משפט הבקרה והפסקת הלולאה שבה נמצאים.

התחביר : break ;

דוגמה : התוכנית הבאה מבקשת מהמשתמש להכניס מספרים ומדפיסה אותם. כשהמשתמש מקיש מספר שלילי התוכנית מסתיימת



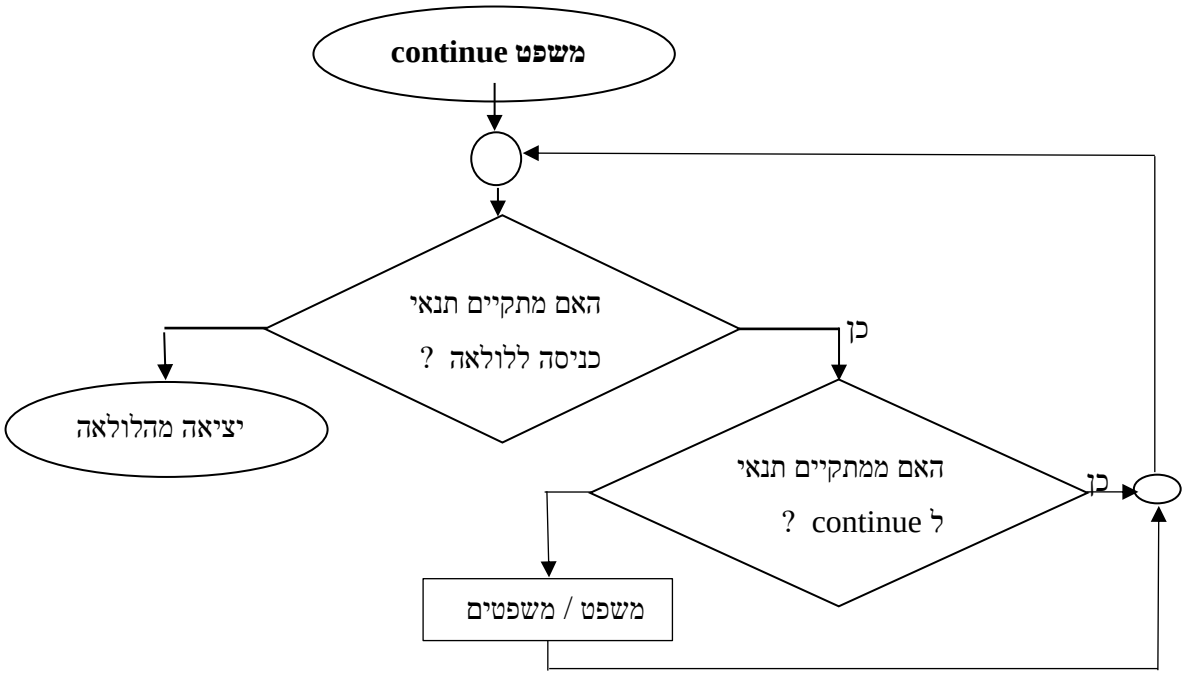
איור 11 : הדפסת המספרים שהמשתמש מכניס עד שהמשתמש מכניס מספר שלילי

משפט continue

הפסקת ביצוע משפטי הלולאה ודילוג לראש הלולאה לביצוע איטרציה נוספת.

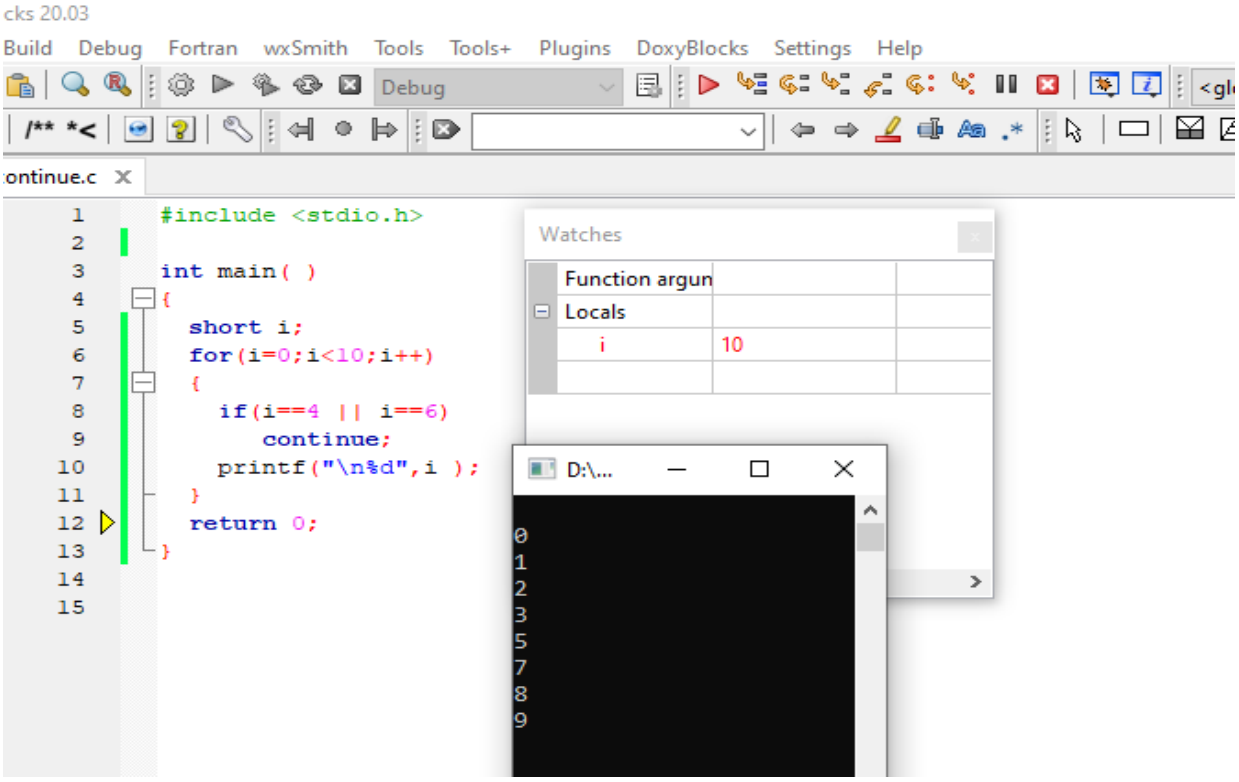
התחביר : continue ;

תרשים הזרימה של המשפט נראה כך:



איור 12 : תרשים זרימה של משפט continue

באיור הבא יש תוכנית המדפיסה את המספרים בין 0 ל 9 ללא המספר 4 וללא המספר 6 .



איור 13 : הדפסת המספרים בין 0 ל 9 ללא המספרים 4 ו 6 .

הערה: בפרק הבא יש חזרה על הוראות break ו continue כולל דוגמאות נוספות.

תרגילים קלט/פלט

1. כתוב תוכנית שתקלוט מהמשתמש את גילו (מספר שלם), ותדפיס בן כמה יהיה בעוד 20 שנה.

2. כתוב תוכנית שתקלוט מספר ותפלוט את ריבועו (לדוגמה : קולטים 5 והפלט של הריבוע הוא 25).

3. כתוב תוכנית שתקלוט 3 מספרים ותפלוט את הממוצע של שלושתם.

4. כתוב תוכנית שתקלוט תו, ותדפיס פירמידה בת 3 קומות של אותו תו. בדוגמה התו שנקלט הוא '+'.

```
+

++

+++
```

5. קלוט אות גדולה מהמשתמש והדפס אותה כאות קטנה(רמז: העזר בטבלת האסקי).

6. קלוט (על ידי getch()) שני מספרים בעלי ספרה אחת ופלוט את מכפלתם.

7. קלוט(על ידי getch()) מספר דו ספרתי, הכנס אותו למשתנה x והדפס אותו על המסך.

8. קלוט שני מספרים בעלי ספרה אחת (על ידי getch()) וסימן של פעולה אריתמטית ('+' , '-','*') ופלוט את התוצאה של התרגיל.

9. קלוט שני מספרים, החסר מהראשון 50, הכפל את השני ב-2, ופלוט את סכום שני המספרים החדשים שנוצרו אם הסכום המתקבל הוא זוגי.

10. כתוב תוכנית שתקבל כקלט שטח בן שלוש ספרות המייצג שטח מעגל, והדפס את הרדיוס שלו.

11. כתוב תוכנית שתקלוט משכורת של אדם, את אחוז המס, ותדפיס את הסכום שירד לו מהמשכורת בהתאם.

12. כתוב תוכנית שקולטת מספר ופולטת את עיגול המספר א. עם משפט תנאי ב. ללא שימוש במשפט תנאי .

13. קלוט מהמשתמש את שער הדולר היומי (גודל המבטא את הקשר בין דולר לש"ח). לאחר מכן קלוט את כמות הדולרים שברצונו להמיר. חשב את הערך בשקלים שעליו לשלם.(התעלם מנושא העמלה).

14. רשום תוכנית שתדפיס את המרחק הנקרא שנת אור. ידוע שמהירות האור היא 300 אלף ק"מ בשנייה ובשנה יש 365 ימים.

15. מה ההבדל בין פונקציות הקלט getch , getche ו getchar ?

16. קלוט 4 מספרים, חשב את הממוצע שלהם והדפס אותו עם 2 ספרות אחרי הנקודה.

17. קלוט מספר עשרוני והדפס את תו האסקי שלו. הסבר מדוע כאשר נקלט מספר כמו 321 מודפס התו 'A' ?

18. הדפס את המספר 1234.5678 עם ספרה אחת אחרי הנקודה ועם 3 ספרות אחרי הנקודה.

19. קלוט 4 מספרים, חשב את הממוצע שלהם והדפס אותו עם 2 ספרות אחרי הנקודה.

20. נתונות לך 2 המחרוזות הבאות : char str1[ ]= "Hello" char str2[ ] = "world". הצג את 2 המחרוזות במסך בצורה הבאה : א. Hello world ב. Hell world ג. HHwor ד. Hello*** (הכוכביות מסמנות רווח)

21. מה עושה התוכנית הבא ומה ההדפסה עבור קלט של radius=2.2 ?

```
#include <stdio.h>
void main( ) {
const float PI=3.14;
float radius,volume;
puts("This program calculates the volume of a ball.\nAll units are in meters");
printf("Please Enter The radius of the ball : ");
scanf("%f",&radius);
volume=4*PI*radius*radius*radius/3;
printf("\nThe volume of the ball is : %f ", volume);  }
```

(תשובה : 445.7962)

22. קלוט מספר ממשי והסב אותו למספר שלם תוך עיגול המספר. לדוגמא: 5.3 ← 5 . 7.6 ← 8 הדפס את המספר הממשי וההפיכה שלו.

23. קלוט מספר עשרוני והצג אותו באוקטלי ובהקסה דצימלי.

24. קלוט תו והצג את ערך האסקי שלו.

25. קלוט מספר עשרוני והדפס את תו האסקי שלו. הסבר מדוע כאשר נקלט מספר כמו 321 מודפס התו 'A' ?

26. כתוב תוכנית המדפיסה כמה בתים תופס כל טיפוס משתנה . (רמז - sizeof).

27. קלוט 2 מספרים שלמים המתארים בהתאמה את האורך והרוחב של מלבן, צלע וגובה של משולש. חשב את שטח המלבן והמשולש.

28. מהם הערכים של x ושל y בסיום קטע התוכנית הבא ? שים לב לעדיפות ! ידוע : x=10 y = 5

```
y=2+x++;

y=++x%x++;

y=--x/(x--*5);
```

29. קלוט מספר עשרוני והצג אותו בצורה בינארית.

30. קלוט 3 מספרים שלמים המייצגים (בהתאמה) את האיבר הראשון של הסדרה החשבונית, ההפרש בין האיברים וכמות האיברים בסדרה. הדפס את כל האיברים ואת סכומם.

31. קלוט 3 מספרים ממשיים המייצגים (בהתאמה) את האיבר הראשון של סדרה הנדסית, היחס בין האיברים וכמות האיברים. הדפס את כל האיברים ואת סכומם.

עבודה נעימה