

פעולות אריתמטיות משפטי תנאי



קרון כליף

ביחידה זו נלמד:

פעולות אריתמטיות □

■ ביטויים חשבוניים

■ אופרטור ++

■ ביטויים מקוצרים

המרות בין טיפוסים (casting) □

□ ביטויים לוגיים

□ משפטי תנאי

□ משפט switch

□ משפט if מקוצר

פעולות חשבוניות

ניתן לחשב ביטויים בעזרת הפעולות הבאות:

■ חיבור

■ חיסור

■ כפל

■ חילוק

■ שימו לב: חלוקה ששני מרכיביה שלמים מחזירה את תוצאת השלם של החלוקה

$$2 = 5 / 13$$

$$2.6 = 5 / 13.0$$

■ אסור לחלק ב-0!

■ מחזירה את תוצאת השארית של החלוקה

$$3 = 5 \% 13$$

■ המחלק והמחולק חייבים להיות שלמים

$$13 \% 5.2 \text{ לא יתקמפל}$$

שימוש בפעולות חשבוניות

שימוש ב- 2% כדי לבדוק האם מספר הוא זוגי

1 = $23\%2$ ■

0 = $24\%2$ ■

1 = $25\%2$ ■

0 = $26\%2$ ■

שימוש ב- 10% כדי לקבל את הספרה הימנית של מספר

8 = $8\%10$ ■

5 = $95\%10$ ■

7 = $357\%10$ ■

שימוש ב- 10/ כדי לקצץ את הספרה הימנית של מספר

0 = $8/10$ ■

9 = $95/10$ ■

35 = $357/10$ ■

- יש לקלוט מהמשתמש מספר המייצג שעה
- למשל 1245 מייצג את השעה 12 ו- 45 דקות
- יש להציג הודעה ברורה מהי השעה
- לקבלת הדקות:
 - $45 = 1245 \% 100$
- לקבלת השעות:
 - $12 = 1245 / 100$

קדימויות הפעולות

- כמו בחשבון, קודם מבצעים חישוב שנמצא בסוגריים
- אח"כ כפל, חילוק ומודולו
- ולבסוף חיבור וחסור

- במידה ויש כמה פעולות באותה רמת עדיפות החישוב יבוצע משמאל לימין
- דוגמאות:

$$1 + (4 + 5) * (2/3) \% 5 = 1$$

$$1 + (4 + 5) * (3/2) \% 5 = 5$$

$$1 + 4 + 5 * 3/2 \% 5 = 7$$

ביטוי חשבוני - שימוש

ניתן להשתמש בביטוי החשבוני ישירות בתוכנית, או
לאחר שמירתו בתוך משתנה

```
#include <stdio.h>
```

```
void main()
```

```
{
```

```
    int n1, n2, sum;
```

```
    printf("Please enter 2 numbers: ");
```

```
    scanf("%d %d", &n1, &n2);
```

```
    printf("The sum is %d", n1+n2);
```

שימוש ישיר בביטוי

```
    sum = n1 + n2;
```

```
    printf("The sum is %d", sum);
```

שימוש בביטוי לאחר
שמירתו במשתנה

```
}
```

הגדלה/הקטנה ב- 1

□ כדי להגדיל/להקטין משתנה ב- 1:

$x = x - 1;$

$x = x + 1;$

□ יש לנו סינטקס מיוחד עבור מקרה זה:

■ ++ עבור הגדלת משתנה ב- 1: $x++$ או $++x$

■ -- עבור הקטנת משתנה ב- 1: $x--$ או $--x$

□ כלומר:

$x = x + 1; \quad \equiv \quad x++; \quad \equiv \quad ++x;$

$x = x - 1; \quad \equiv \quad x--; \quad \equiv \quad --x;$

הגדלה/הקטנה ב- 1 (2)

אבל כאשר משתמשים באופרטור ++ לצורך השמה למשתנה אחר יש הבדלים בין תוצאות הביטויים:

■ $y = x + 1;$ ← ערכו של x לא משתנה בעקבות פקודה זו, y משתנה
■ $++y = x$
■ $y = ++x$ ← ערכו של x גדל ב-1 בעקבות פקודה זו, y משתנה

אם האופרטור ++ מופיע מימין למשתנה (למשל $x++$) אז לתוך y נכנס הערך המקורי של x ורק אז ערכו של x גדל:

```
int x = 4;  
int y = x++; //□ y=4, x=5
```

אם האופרטור ++ מופיע משמאל למשתנה (למשל $++x$) אז קודם ערכו של x גדל ואז ערכו החדש נכנס לתוך y :

```
int x = 4;  
int y = ++x; //□ y=5, x=5
```

ביטויים מקוצרים

□ כדי להוסיף ערך כלשהו למשתנה ניתן להשתמש בכתיב מקוצר:

$$;y = y+3; \quad \equiv \quad y += 3$$

□ ניתן להשתמש בכתיבה מקוצרת זו עבור כל האופרטורים שלמדנו (+ - * / %)
■ דוגמא:

$$x *= y+1; \quad \left\{ \begin{array}{l} \underline{x} = x*(y+1); \\ x = \underline{x}*y+1; \end{array} \right.$$

ביטויים מקוצרים - דוגמא

```
void main()
{
    int  n1, n2, n3, n4;

    n1 = 2;
    n2 = ++n1;
    n2 *= n1--;    // □ n2 = n2*n1--
    n3 = --n1 * n2--;
    n3 = (n1 * n2)--; // □ doesn't compile, since -- requires l-value
    n3 = n4 = n1 - 1;
    n4 = n1++ + n2++;
    n3 = (n1++) + (n2++);
}
```

int: n1	3	1000
int: n2	10	1004
int: n3	11	1008
int: n4	9	1012

האופרטור ++ מימין תמיד פועל
בסוף,
ללאפילו אם הוא בסוגריים

המרות בין טיפוסים (casting)

ייתכן ונרצה לחשב ביטוי המכיל משתנים מטיפוסים שונים, ולכן צריך לדעת איך להתייחס לפעולות על טיפוסים שונים

קיימת היררכיה של המרות כך שמשתנים מטיפוסים בעלי עדיפות נמוכה יותר בביטוי מומרים לטיפוס בעל העדיפות הגבוהה ביותר:

1. double
2. float
3. long
4. int
5. char

ההיררכיה היא לפי גודל
הטיפוס, כך שלא נאבד מידע

המרות בין טיפוסים (casting) (2)

ניתן לכתוב ביטוי המכיל טיפוסים שונים, למשל:

```
double res, d1=5.2;
```

```
int n1=4;
```

```
res = n1 + d1;
```

בדוגמא זו המחשב ממיר את המשתנה $n1$ ל-
`double`
ואז מבוצעות פעולות החיבור וההשמה

נשים לב כי $n1$ בזיכרון אינו הופך ל-`double`, שכן
ההמרה נעשית באופן זמני בלבד לצורך הערכת הביטוי

המרות בין טיפוסים – דוגמא 1

```
void main()
{
    int n1=5, n2=8;
    double d0, d1=7;    // □ d1 is actually 7.0

    d0 = d1 + n1 / n2;
}
```

□ ראשית מחושב הביטוי $n1/n2$. מאחר ושני מרכיביו הם `int` לא מתבצעת המרה. תוצאת ביטוי זה היא 0.

□ הביטוי המחושב כעת הוא חיבור בין `double` ל-`int` ולכן ה-`int` מומר ל-`double` (0.0) והתוצאה היא 7.0

המרות בין טיפוסים – דוגמא 2

```
void main()
{
    int n1=5;
    double d0, d1=7, d2=8.0;

    d0 = d1 + n1 / d2;
}
```

- ראשית מחושב הביטוי $n1/d2$. הפעם $n1$ מומר ל- `double` וחלוקת שני ה- `double` מחזירה 0.625.
- הביטוי המחושב כעת הוא חיבור בין `double` ל- `double` ולכן אין המרות נוספות והתוצאה היא 7.625

המרות בין טיפוסים – דוגמא 3

חוק ההמרות: נחשב את הביטוי בצד ימין ע"י המרת כל המשתנים לטיפוס הגדול ביותר, ולבסוף נבצע המרת התוצאה לטיפוס שמשמאל

```
void main()
{
    int n0;
    double d1=9, d2=2.0;

    n0 = d1 / d2 + 5;
}
```

- ראשית מחושב הביטוי $d1/d2$ שתוצאתו 4.5
- הביטוי הבא המחושב הוא חיבור בין double ל- int שתוצאתו היא 9.5
- ערך הביטוי מושם לתוך int ולכן יש המרה מ- double ל- int והתוצאה תהייה 9

המרות בין טיפוסים – דוגמא 4

```
void main()
{
    int x;
    char ch = 'a';
    x = ch+3;
}
```

מתבצעת המרה מ- char ל- int ולכן תוצאת הביטוי $a' (97) + 3 \square 100$ של ASCII היא ערכו ה- \square

המרות בין טיפוסים – דוגמא 5

```
void main()
{
    char ch = 'a';
    int n1=5, n2=7;
    double d1=2.5;

    n1 = ch++ + 2;
    ch = n1+2;
    n1 = (n1-'a'+3) * d1;
    n2 *= d1;
}
```

// □ $n1=99$, $ch='b'$
// □ $ch='e'$
// □ $n1=(99-97+3)*2.5 = 12$
// □ $n2 = 7*2.5 = 17$

המרה מכוונת

□ זוהי המרה מכוונת בין הטיפוסים השונים:

$(type) <expression>$

■ דוגמאות:

□ 5 □ $(2.5+3)(int)$ □

□ 5.0 □ $(3+2)(double)$ □

□ לאופרטור casting יש את הקדימות הגבוהה ביותר

לאחר סוגריים

■ $x/y(double)$ עושים המרה של x ל- $double$ ורק אז

מבצעים את החילוק

המרה מכוונת - דוגמא

```
void main()
{
    int  n1=5, n2=6;
    double d1;

    d1 = n1/n2;           // d1=0
    d1= (double)n1/n2;    // d1= 0.833333
    d1= n1/(double)n2;    // d1= 0.833333
    d1= (double)(n1/n2);  // d1= 0.0
    n1 = (int)(n2/4.23);  // n1=1
}
```

ההמרה המכוונת אינה הכרחית בדוגמא זו כי בכל מקרה תהייה המרה ל-int, מאחר וזהו הטיפוס שמשמאל, אבל יעלם ה-warning

ביטויים לוגיים

□ ביטוי לוגי הינו דרך לבטא יחס בין 2 ביטויים

■ דוגמאות ליחסים בין ביטויים: גדול, קטן שוויון

■ תוצאת הביטוי היא True/False

□ ניתן להכניס ערכו של ביטוי לוגי לתוך משתנה:

■ אם הביטוי הלוגי החזיר True יוכנס 1 לתוך המשתנה

■ אם הביטוי הלוגי החזיר False יוכנס 0 לתוך המשתנה

אופרטורי יחס ושוויון

```
int x=3, y=5, z=5, res;
```

```
// res is 1           ;res = x < y
```

```
// res is 1           ;res = x <= y
```

■ ובאופן דומה פועלים האופרטורים > ו- >=

```
// res is 1           ;res = x != y
```

```
// res is 0           ;res = y != z
```

```
// res is 0           ;res = x == y
```

■ בניגוד לרשימה מופיעה...
// res is y

נשים לב שבבדיקת שוויון יש פעמיים את
הסימן שווה (=), לעומת השמה (=)

אופרטורי יחס ושוויון (2)

□ מה תוצאת הביטוי $4 > 9 < x$?

- מאחר ויש יותר מאופרטור אחד, נעריך את הביטוי משמאל לימין
- תוצאת הביטוי הראשון תהיה או 0 או 1 (תלוי בערכו של x)
- כעת יוערך הביטוי השני, שתוצאתו תמיד תהייה 1 (כי 0 או 1 שניהם תמיד קטנים מ- 9)

□ מה תוצאת הביטוי $4 > 1 < x$?

- תוצאת ביטוי זה אינה ידועה כי לא ידוע ערכו של x

אופרטורים לוגיים

לפעמים נרצה שערכו של ביטוי יורכב מתוצאתם של כמה ביטויים

■ למשל, כדי לדעת האם $4 < x < 9$ צריך לבדוק האם $x < 9$ וגם $x > 4$

וגם **&&**: יחזיר 1 אם שני הביטויים המרכיבים אותו החזירו 0, true, אחרת

או **||**: יחזיר 1 אם לפחות אחד משני הביטויים המרכיבים אותו החזירו true, false, אחרת

```
int res;  
res = 4 < 7 && 7 < 9;    // res is 1  
res = 4 < 7 && 7 > 9;    // res is 0  
res = 4 < 7 || 7 < 9;    // res is 1  
res = 4 < 7 || 9 < 7;    // res is 1  
res = 7 < 4 || 9 < 7;    // res is 0
```

דוגמאות:

אופרטורים לוגיים (2)

כאשר נרצה לקבל את שלילתו של ביטוי מסוים נשתמש באופרטור !

דוגמא:

```
int x=4, y=5, res;
```

```
res = x>y;    // ☐ res is 0
```

```
res = !(x>y); // ☐ res is 1
```

אופרטורים לוגיים (3)

□ אופרטור בינארי: עובד על 2 ביטויים, יוצר ביטוי לוגי מורכב

■ &&

■ ||

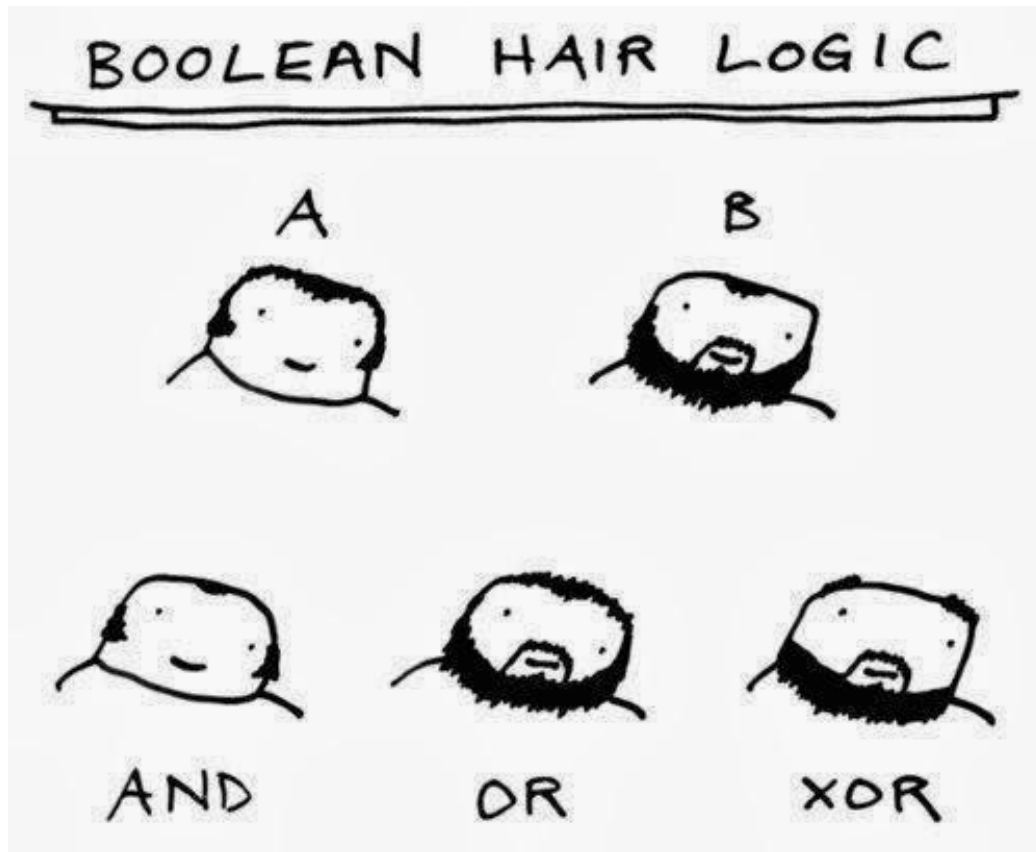
□ אופרטור אונארי: עובד על ביטוי אחד

■ !

□ סיכום תוצאות הפעלת האופרטורים על ביטויים לוגיים:

a	b	a&&b	a b	a!
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

סיכום אופרטורים לוגיים



https://lh4.googleusercontent.com/-Wkph_Ly5ZGM/UumXqMSYkDI/AAAAAAAAAE10/_QjUO_diBxg/it_jokes_boolean.jpg

קדימויות אופרטורים - דוגמא

1 □ (1 || (1 &&

0 □ ((1 || 1) &

1 □ (1 || 1 &

1 □ (1 && 1 |' ^`

((0 < 4) == 1)&& (5 > 2) ┐

(0 == (4 < 1))&& (5 > 2) ┐

קדימויות האופרטורים
!
= < < = > >
= ! = =
&&

אופטימיזציות בעת חישוב ביטוי לוגי המכיל אופרטור בינארי (short-circuit evaluation)

- כאשר מחשבים ביטוי המכיל && ותוצאת הביטוי הראשון היא false הקומפיילר לא יטרח לבדוק גם את השני
- כאשר מחשבים ביטוי המכיל || ותוצאת הביטוי הראשון היא true הקומפיילר לא יטרח לבדוק גם את השני
- כדי להכריח את הקומפיילר לבדוק את כל הביטויים, ללא קשר לתוצאת נשתמש ב- & או ב- | עבור "וגם" או "או" בהתאמה

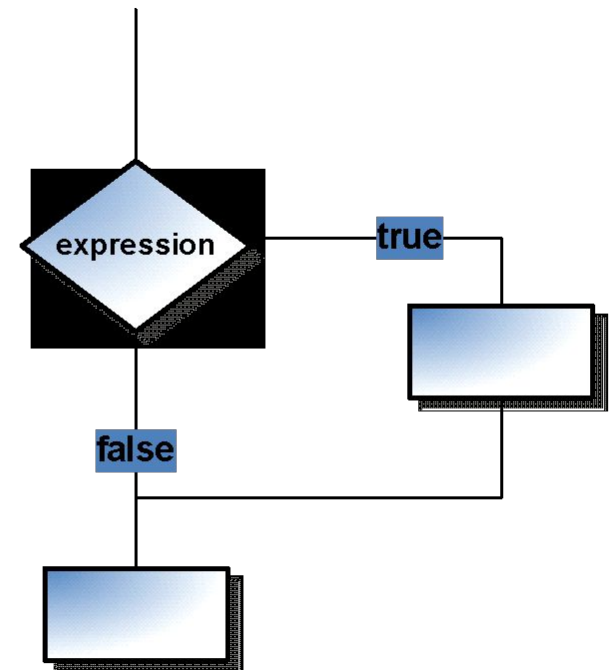
משפטי התנאי if

משפט תנאי נועד לבצע פעולה רק אם תנאי מסוים מתקיים

■ דוגמא: קבל מספר והדפס הודעה רק אם הוא חיובי

```
void main()
{
    int num;
    printf("Please enter a number: ");
    scanf("%d", &num);

    if (num > 0)
    {
        printf("Your number is positive");
    }
}
```



משפט תנאי if - תחביר

```
if (expression)
{
    statement;
    statement1;
}
statement2;
...
}
```

נשים לב שאין ; בסוף שורת ה-if, שכן לא מבוצעת שום פקודה בשורה זו, אלא רק מתבצעת בדיקה

נשים לב שהקו בין ה- { למכנס סגור אחרי פנימה.
סגנון כתיבה הנקרא אינדנטציה ויש להקפיד עליו.

□ ה-expression יהיה ביטוי לוגי כלשהו (גם ביטוי מורכב)

□ גוף ה-if יכיל אוסף פקודות אותן נרצה לבצע

- אוסף הפקודות יהיה עטוף ב- {}
- במידה ויש פקודה אחת בלבד אין חובה לעטוף אותה ב- {} למרות שמומלץ תמיד כן לעשות זאת

משפט תנאי if - דוגמא

קבל 3 מספרים שונים מהמשתמש, והדפס הודעה רק אם המספר הראשון שהוכנס הוא המקסימלי

```
void main()
{
    int num1, num2, num3;
    printf("Please enter 3 numbers: ");
    scanf("%d %d %d", &num1, &num2, &num3);

    if (num1 > num2 && num1 > num3)
    {
        printf("%d is the maximum", num1);
    }
}
```

שימוש בביטוי לוגי מורכב

אבל הינו רוצים לתת גם הודעה לגבי שאר
המקרים...

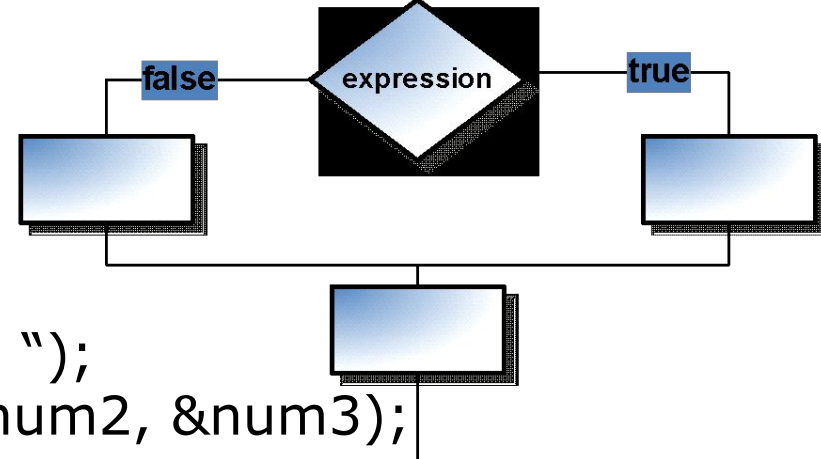
משפט תנאי if-else

```
#include <stdio.h>
```

```
void main()  
{
```

```
    int num1, num2, num3;  
    printf("Please enter 3 numbers: ");  
    scanf("%d %d %d", &num1, &num2, &num3);
```

```
    if (num1 > num2 && num1 > num3)  
    {  
        printf("%d is the maximum", num1);  
    }  
    else  
    {  
        printf("The maximum is %d or %d", num2, num3);  
    }  
}
```

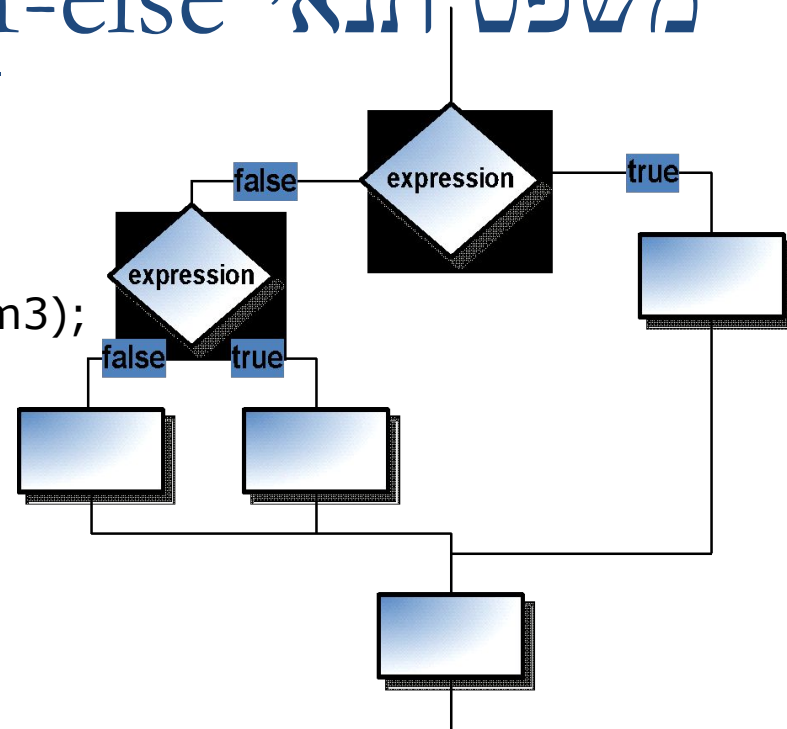


חלק ה- else מתבצע רק
אם הביטוי ב- if היה
false

משפט תנאי if-else

```
void main()
{
    int num1, num2, num3;
    printf("Please enter 3 numbers: ");
    scanf("%d %d %d", &num1, &num2, &num3);

    if (num1 > num2 && num1 > num3)
    {
        printf("%d is the maximum", num1);
    }
    else
    {
        if (num2 > num1 && num2 > num3)
        {
            printf("%d is the maximum", num2);
        }
        else
        {
            printf("%d is the maximum", num3);
        }
    }
}
```



נשים לב לאינדנטציה!!

משפט תנאי if-else – בלי סוגריים

```
void main()
{
    int num1, num2, num3;
    printf("Please enter 3 numbers: ");
    scanf("%d %d %d", &num1, &num2, &num3);

    if (num1 > num2 && num1 > num3)
        printf("%d is the maximum", num1);
    else
        if (num2 > num1 && num2 > num3)
            printf("%d is the maximum", num2);
        else
            printf("%d is the maximum", num3);
}
```

ניתן לכתוב קוד זה בלי סוגריים מאחר ומבחינת
הקומפילר משפט if+else נחשב לפקודה אחת

משפט תנאי if-else-else

```
#include <stdio.h>
```

```
void main()
```

```
{  
    int num1, num2, num3;  
    printf("Please enter 3 numbers: ");  
    scanf("%d %d %d", &num1, &num2, &num3);  
  
    if (num1 > num2 && num1 > num3)  
        printf("%d is the maximum", num1);  
    else if (num2 > num1 && num2 > num3)  
        printf("%d is the maximum", num2);  
    else  
        printf("%d is the maximum", num3);  
}
```

דוגמא זו והדוגמא הקודמת
זהות,
אכל טעמים סגנון בתיבה זה

הצגת הפלט פעם אחת בסוף התוכנית

תמיד נעדיף שפלט התוכנית יהיה במקום מרוכז
בסוף,
במקום לשכפל את הודעת ההדפסה כמה פעמים.

```
void main()
{
    int num1, num2, num3, max;

    printf("Please enter 3 numbers: ");
    scanf("%d %d %d", &num1, &num2, &num3);

    if (num1 > num2 && num1 > num3)
        max = num1;
    else if (num2 > num1 && num2 > num3)
        max = num2;
    else
        max = num3;

    printf("%d is the maximum", max);
}
```

משפט תנאי – דוגמא 1

תוכנית הקולטת תו מהמשתמש ומדפיסה האם זהו אות, ספרה או משהו אחר.

נכתוב אלגוריתם לפתרון הבעיה:

בתוכנית זו אנו מתבססים
על העובדה שהאותיות
בטבלת ה-ASCII נמצאות
ברצף!

- קלוט תו

- אם התו בין 'a' ל- 'z' או אם התו בין 'A' ל- 'Z' הדפס שזהו אות

- אם התו בין '0' ל- '9' הדפס שזהו ספרה

- אחרת הדפס שזהו סימן

main

'a' <= ch <= 'z'
||
'A' <= ch <= 'Z'

'0' <= ch <= '9'

all other

משפט תנאי – דוגמא 1

```
void main()
{
    char letter;
    printf("Please enter a letter: ");
    scanf("%c", &letter);

    if ( (letter >= 'a' && letter <= 'z') || (letter >= 'A' && letter <= 'Z') )
    {
        printf("A Letter\n");
    }
    else if (letter >= '0' && letter <= '9')
    {
        printf("Digit\n");
    }
    else // any symbol
    {
        printf("Not a letter nor a digit\n");
    }
}
```

מאחר והתנאים זרים (רק אחד מהם יקרה)
אנחנו משתמשים ב- if-else.
ניתן היה להשתמש רק ב- if אבל אז הקוד
פחות

היה (לא הכרחי כמובן שכל אחד מהם יקרה)

משפט תנאי – דוגמא 1

```
void main()
{
    char letter;
    printf("Please enter a letter: ");
    scanf("%c", &letter);

    if ( (letter >= 'a' && letter <= 'z') || (letter >= 'A' && letter <= 'Z') )
    {
        printf("A Letter\n");
    }

    if (letter >= '0' && letter <= '9')
    {
        printf("Digit\n");
    }

    if (!(letter >= 'a' && letter <= 'z') && // any symbol
        !(letter >= 'A' && letter <= 'Z') &&
        !(letter >= '0' && letter <= '9') )
    {
        printf("Not a letter nor a digit\n");
    }
}
```


משפט תנאי – דוגמא 2

□ תוכנית הקולטת ציון מהמשתמש ומדפיסה הודעה מתאימה.

□ נכתוב אלגוריתם לפתרון הבעיה:

■ קלוט ציון

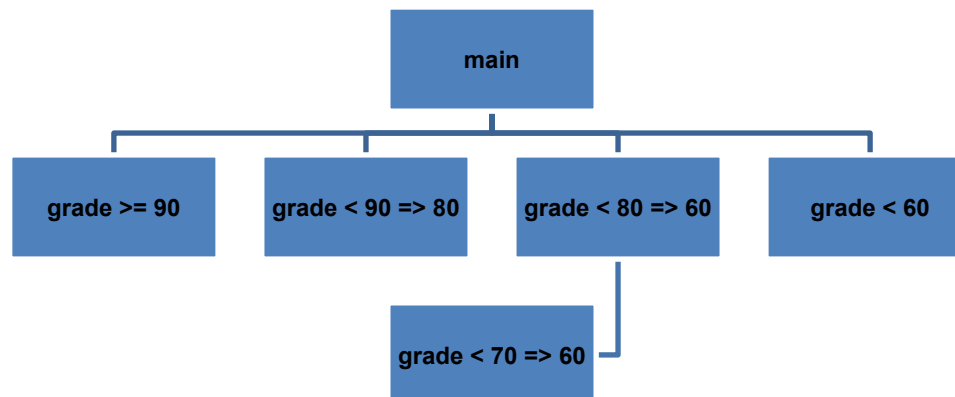
■ אם הציון מעל 90 הדפס Excellent

■ אם הציון בין 80 ל- 90 הדפס Very Good

■ אם הציון בין 60 ל- 80 הדפס Passed

□ אם הציון בין 60 ל- 70 הדפס גם You should work harder

■ אם הציון מתחת ל- 60 הדפס Failed



משפט תנאי – דוגמא 2

```
#include <stdio.h>
```

```
void main()  
{
```

```
    int grade;
```

```
    printf("Please enter a grade: ");
```

```
    scanf("%d", &grade);
```

```
    if (grade >= 90)
```

```
        printf("Excellent!\n");
```

```
    else if (grade < 90 && grade >= 80)
```

```
        printf("Very Good!\n");
```

```
    else if (grade < 80 && grade >= 60)
```

```
    {
```

```
        printf("Passed.\n");
```

```
        if (grade >= 60 && grade < 70)
```

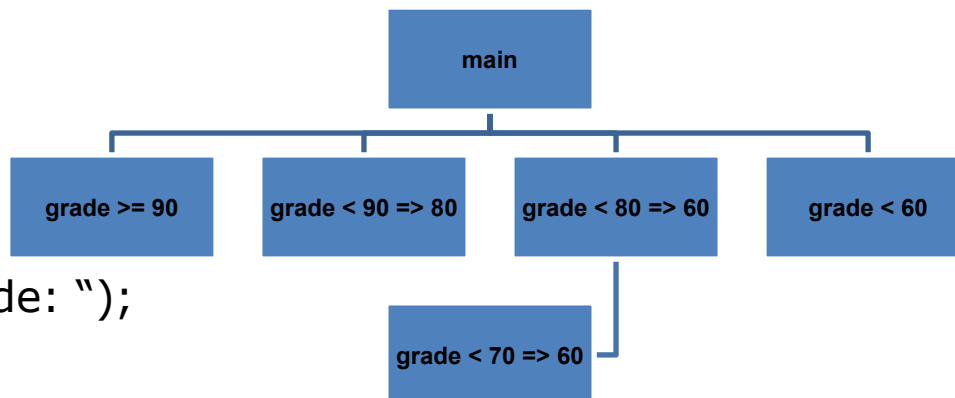
```
            printf("You should work harder!\n");
```

```
    }
```

```
    else // grade < 60
```

```
        printf("Failed!\n");
```

```
}
```



משפט תנאי – דוגמא – עדיף כך..

```
#include <stdio.h>
```

```
void main()  
{
```

```
    int grade;
```

```
    printf("Please enter a grade: ");
```

```
    scanf("%d", &grade);
```

```
    if (grade >= 90)
```

```
        printf("Excellent!\n");
```

```
    else if (grade < 90 && grade >= 80)
```

```
        printf("Very Good!\n");
```

```
    else if (grade < 80 && grade >= 60)
```

```
    {
```

```
        printf("Passed.\n");
```

```
        if (grade >= 60 && grade < 70)
```

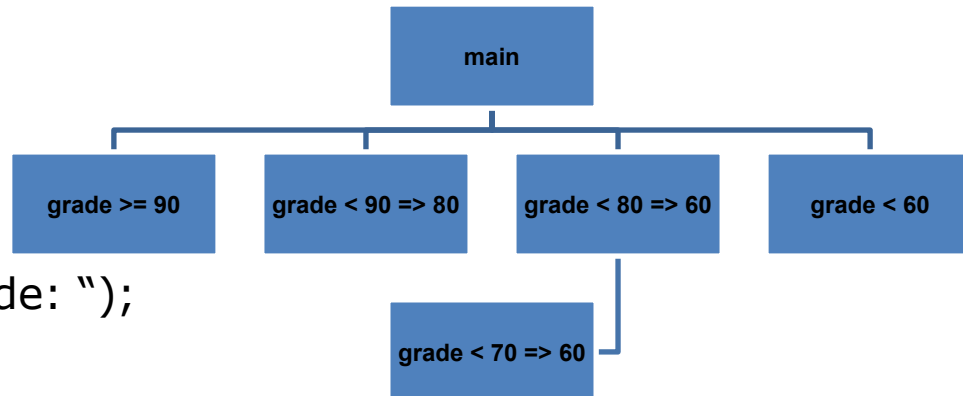
```
            printf("You should work harder!\n");
```

```
    }
```

```
    else // grade < 60
```

```
        printf("Failed!\n");
```

```
}
```



ניתן לוותר על בדיקת תנאים אלו
משום שניתן להסיק אותם מעצם
אי-קיום התנאי הקודם!

מה יקרה אם נוריד את הסוגריים שב- else?

```
#include <stdio.h>
```

```
void main()  
{
```

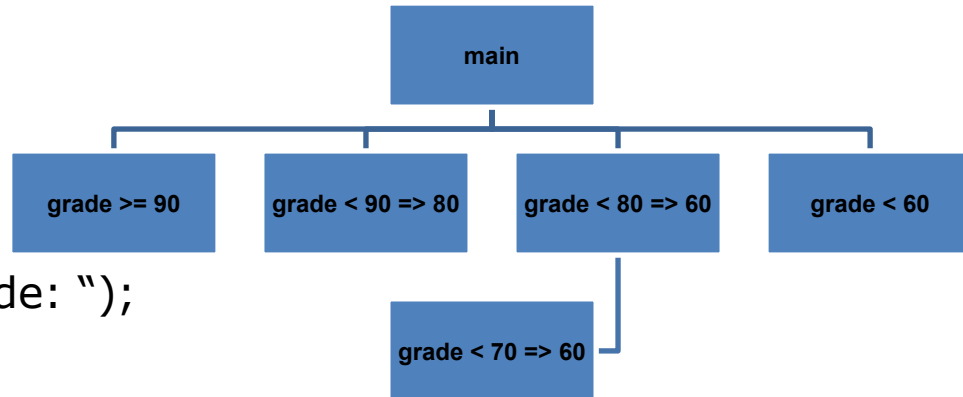
```
    int grade;  
    printf("Please enter a grade: ");  
    scanf("%d", &grade);
```

```
    if (grade >= 90)  
        printf("Excellent!\n");  
    else if (grade >= 80)  
        printf("Very Good!\n");  
    else if (grade >= 60)
```

```
    {  
        printf("Passed.\n");  
        if (grade < 70)  
            printf("You should work harder!\n");
```

```
    }  
    else // grade < 60  
        printf("Failed!\n");
```

```
}
```



מה יודפס אם יוקלד הציון

הקומפיילר אינו עובד לפי
האינדנטציה, ולכן מבחינתו כל
בלוק ההתניות הראשון נגמר פה

כל else משויך ל- if הקרוב ביותר לפניו
שאינו סגור ע"י else אחר או ע"י }

```
Please enter a math expression (without spaces): 8-2
The result is 6
Press any key to continue . . .
```

משפט switch

```
void main()
{
```

```
    char op;
    int num1, num2, res, opOK=1;
    printf("Please enter a math expression (without spaces): ");
    scanf("%d%c%d", &num1, &op, &num2);
```

```
    if (op == '+')
        res = num1+num2;
    else if (op == '-')
        res = num1-num2;
    else if (op == '*')
        res = num1*num2;
    else
        opOK = 0;
```

```
    if (opOK == 1)
        printf("The result is %d\n", res);
    else
        printf("invalid operand\n");
```

```
}
```

כאשר הבדיקה בתנאים היא בדיקת == מול קבוע,
וכאשר התנאים זרים (רק אחד יקרה)
ניתן להחליף מבנה זה במשפט switch

```
switch (op)
{
    case '+':
        res = num1+num2;
        break;
    case '-':
        res = num1-num2;
        break;
    case '*':
        res = num1*num2;
        break;
    default:
        opOK = 0;
        break;
}
```

פקודת ה- break מונעת
גלישה לביצוע ה- case הבא

ה- default יקרה אם אף
אחד מה- case אינו
שווה למשתנה או
לביטוי ב- switch

ניתן גם בלי default ואז
לא יבוצע דבר במקרה
של אי-התאמה

הערך ב- case חייב להיות קבוע (לא משתנה)

— ואם שכתוב ?break

```
#include <stdio.h>
```

```
void main()
```

```
{
```

```
    char op;
```

```
    int num1, num2, res, opOK=1;
```

```
    printf("Please enter a math expression (without spaces): ");
```

```
    scanf("%d%c%d", &num1, &op, &num2);
```

```
    switch (op)
```

```
    {
```

```
    case '+':
```

```
        res = num1+num2;
```

```
        break;
```

```
    case '-':
```

```
        res = num1-num2;
```

```
        //break;
```

```
    case '*':
```

```
        res = num1*num2;
```

```
        break;
```

```
    default:
```

```
        opOK = 0;
```

```
        break;
```

```
    }
```

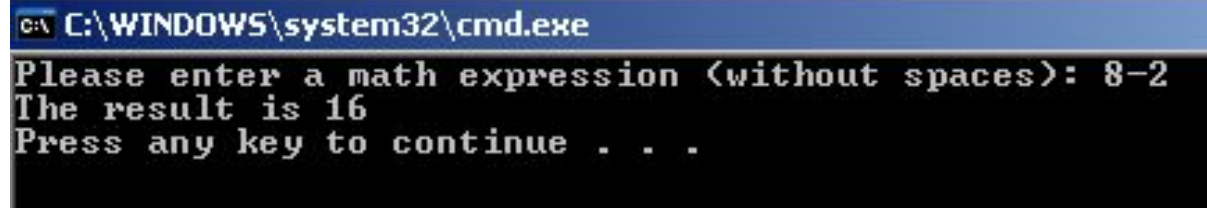
```
    if (opOK == 1)
```

```
        printf("The result is %d\n", res);
```

```
    else
```

```
        printf("invalid operand\n");
```

```
}
```



```
C:\WINDOWS\system32\cmd.exe
```

```
Please enter a math expression (without spaces): 8-2
The result is 16
Press any key to continue . . .
```

```

void main()
{
    char romeDigit;
    int   decimalNum, inputValid = 1;

    printf("Please enter a rome digit: ");
    scanf("%c", &romeDigit);

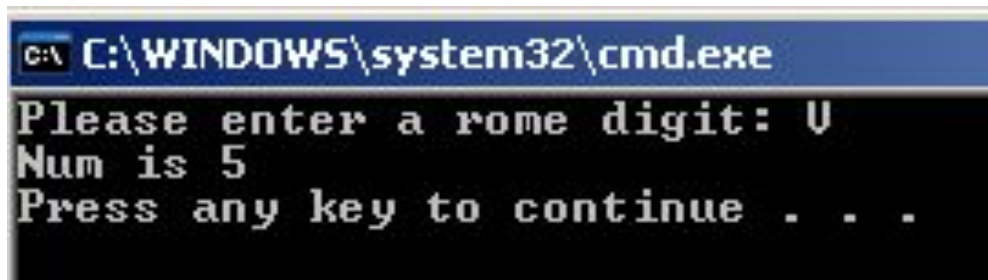
    switch (romeDigit)
    {
        case 'i':
        case 'I':
            decimalNum = 1;
            break;
        case 'v':
        case 'V':
            decimalNum = 5;
            break;
        case 'x':
        case 'X':
            decimalNum = 10;
            break;
        default:
            inputValid = 0;
            break;
    }

    if (inputValid)
        printf("Num is %d\n", decimalNum);
    else
        printf("Invalid input!\n");
}

```

– משפט switch

דוגמת המספרים הרומיים



```

C:\WINDOWS\system32\cmd.exe
Please enter a rome digit: U
Num is 5
Press any key to continue . . .

```

משפט if מקוצר – דוגמא

```
#include <stdio.h>
```

```
void main()  
{
```

```
    int num, absNum;
```

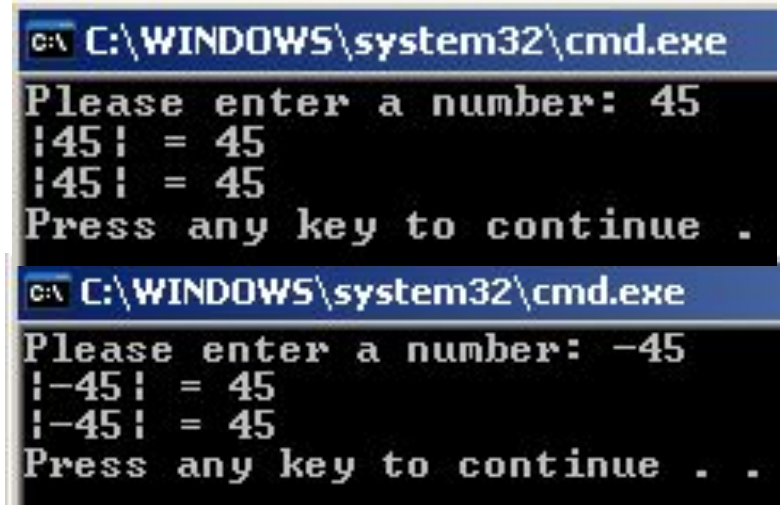
```
    printf("Please enter a number: ");  
    scanf("%d", &num);
```

```
    absNum = num >= 0 ? num : -num;
```

```
    {  
        if (num >= 0)  
            absNum = num;  
        else  
            absNum = -num  
    }
```

```
    printf("|%d| = %d\n", num, absNum);  
    printf("|%d| = %d\n", num, num >= 0 ? num : -num);
```

```
}
```



```
C:\WINDOWS\system32\cmd.exe  
Please enter a number: 45  
|45| = 45  
|45| = 45  
Press any key to continue .  
  
C:\WINDOWS\system32\cmd.exe  
Please enter a number: -45  
|-45| = 45  
|-45| = 45  
Press any key to continue . .
```


משפט if מקוצר

<expression> ? <if true..> : <if false..>

- אם ערך הביטוי הוא true נשתמש בביטוי שמשמאל ל-
":."
- אחרת נשתמש בביטוי שמימין ל- ":"

משפט if מקוצר – דוגמא (2)

```
Please enter your age: 10
You are a child
Press any key to continue .
```

```
Please enter your age: 21
You are NOT a child
Press any key to continue .
```

```
#include <stdio.h>
```

```
void main()
```

```
{
```

```
    int age;
```

```
    printf("Please enter your age: ");
```

```
    scanf("%d", &age);
```

```
    printf("You are%s a child\n", age > 18 ? " NOT" : "");
```

```
}
```

expression

if true

if false

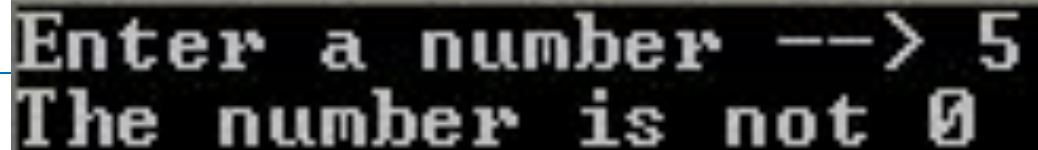
תנאי גם יכול להיות מספר...

```
#include <stdio.h>
```

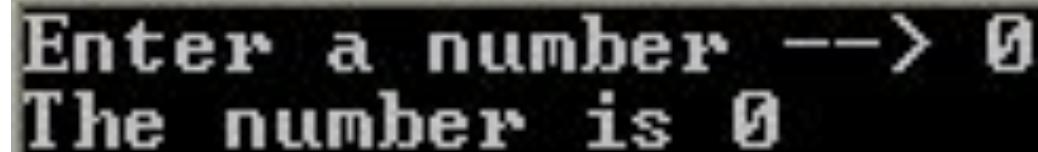
```
void main()
{
    int num;

    printf("Enter a number --> ");
    scanf("%d", &num);

    if (num)
        printf("The number is not 0\n");
    else
        printf("The number is 0\n");
}
```



```
Enter a number --> 5
The number is not 0
```



```
Enter a number --> 0
The number is 0
```

הערך 0 הוא false וכל ערך אחר הוא true

תנאי גם יכול להיות מספר (2) ...

```
#include <stdio.h>
```

```
void main()
```

```
{
```

```
    int num;
```

```
    printf("Enter a number --> ");
```

```
    scanf("%d", &num);
```

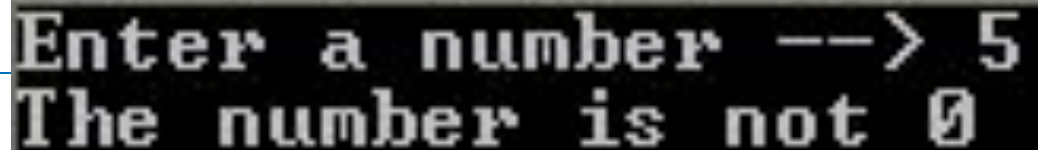
```
    if (!num)
```

```
        printf("The number is 0\n");
```

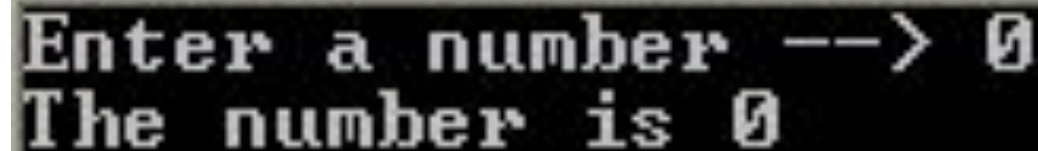
```
    else
```

```
        printf("The number is not 0\n");
```

```
}
```



```
Enter a number --> 5
The number is not 0
```



```
Enter a number --> 0
The number is 0
```

!0 ☐ 1

!1/2/3/.. ☐ 0

דוגמא לשגיאה נפוצה בתנאי לוגי מורכב

□ כדי לבדוק האם הערך של המשתנה X הוא 6 או 7:

■ באופן תקין:

```
if (x == 6 || x == 7)
```

■ ובאופן שגוי:

```
if (x == (6 || 7))
```

הביטוי יחזיר true רק אם ערכו של X יהיה 1!

ביחידה זו למדנו:

פעולות אריתמטיות □

■ ביטויים חשבוניים

■ אופרטור ++

■ ביטויים מקוצרים

המרות בין טיפוסים (casting) □

□ ביטויים לוגיים

□ משפטי תנאי

□ משפט switch

□ משפט if מקוצר