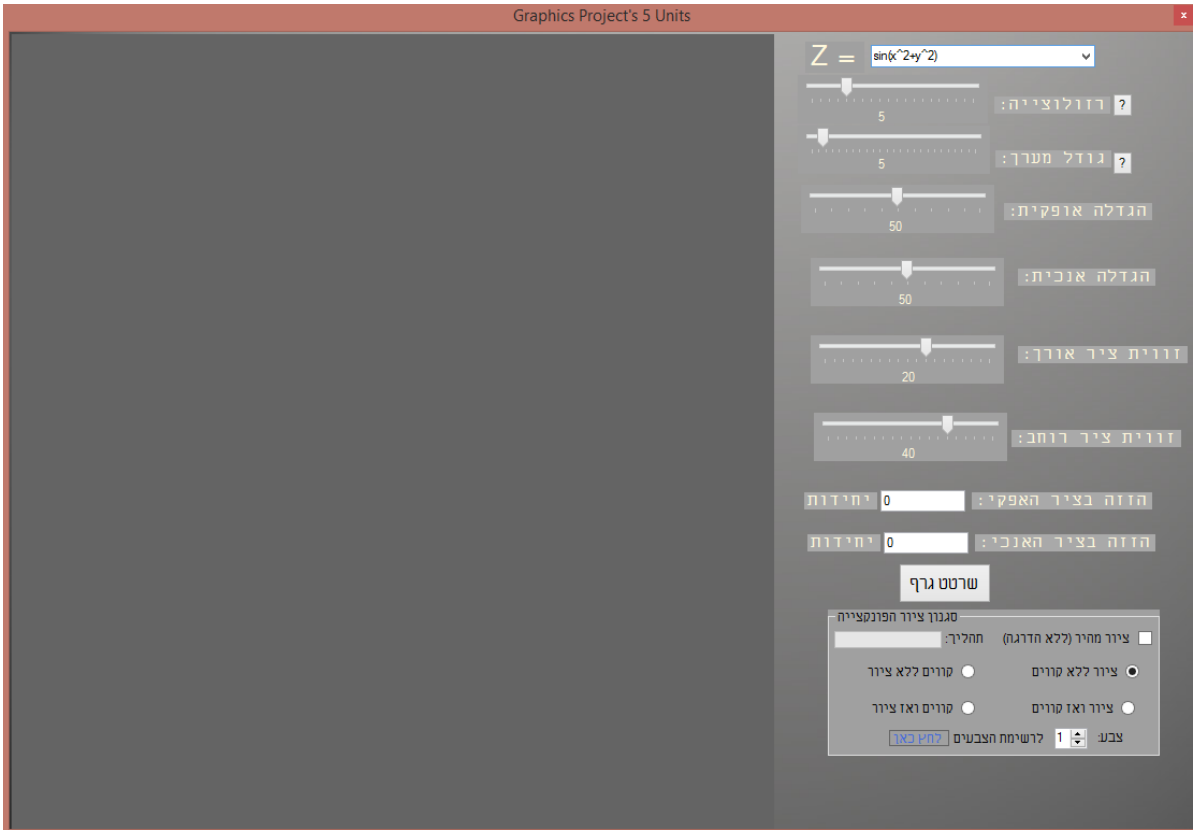
פרויקט בגרות במדעי המחשב 5 יח"ל
בית הספר זיו ומרקס, ירושלים

שם הפרויקט: ציור גרפים תלת ממדיים
מגיש: ינאי אלטר
ת"ז: 211377775
בהנחיית המורה: מאיר פנחס



הדר ב' תשע"ו

תוכן עניינים:

מבוא למשתמש

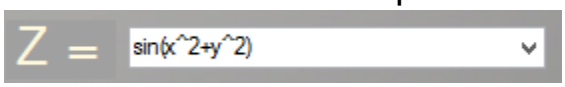
גרפיקה תלת-ממדית היא תחום גרפיקה ממוחשבת, משתמשים בו במחשבים כדי ליצור חיקוי של מראה תלת ממדי. גרפיקה תלת ממדית עוזרת במתמטיקה, מדעי המחשב, רפואה, עיצוב ועוד. בפרויקט תוכלו ליצור פונקציות תלת ממדיות למשל

Z = sin(x)+sin(y), וכך לראות מחזוריות תלת ממדית של פונקציות, הדמיה של גרפים ועוד.

מדריך למשתמש

בפרויקט תוכלו לגשת למבחר של מאפיינים:

1) **<u>בחירת הפונקציה:</u>** תוכלו לבחור איזה פונקציה המחשב יצייר מהצורה f(x,y), כלומר מהצורה z. הכוונה היא שהפונקציה תקבל 2 ערכים – x,y.
דוגמא לפונקציה שאפשר לרשום:



2) **<u>רזולוציה</u>** – תוכלו לבחור רזולוציה בין 20-1.

3) **<u>גודל מערך:</u>**
גודל המערך הוא טווח הx,y. אם גודל המערך הוא 5 יוצבו בפונקציה איקסים וואים בין מינוס 5 ל-5. משפיע על מחזוריות הפונקציה. (ככל שגודל המערך יותר גדול כך הטווח יותר גדול ונראה יותר מהגרף).

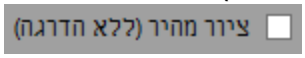
4) **<u>הגדלה אופקית:</u>**
הגדלה אופקית היא למעשה מתיחת הפונקציה לרוחב (זום לרוחב), ככל שנגדיל את הגודל האופקי הפונקציה תמתח כלפי רוחב המסך יותר ויותר.

5) **<u>הגדלה אנכית:</u>**
הגדלה אנכית היא מתיחת הפונקציה לאורך (זום לאורך), ככל שנגדיל את הגודל האנכי הפונקציה תמתח כלפי אורך המסך יותר ויותר.

6) **<u>זווית ציר אורך:</u>**
הזווית שבה ציר האורך נמצא ביחס לזווית ציר הרוחב.

7) **<u>זווית ציר רוחב:</u>**
הזווית שבה ציר הרוחב נמצא ביחס לציר העומק.

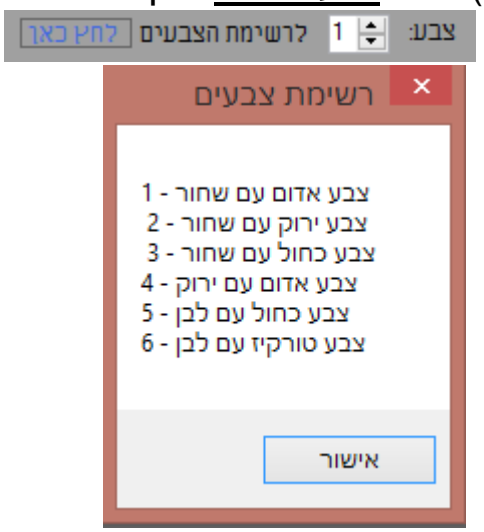
8) **<u>ציור מהיר:</u>** ציור הפונקציה בדרך מהירה יותר, היתרונות הם שהפונקציה תצויר על המסך בזמן נמוך הרבה יותר מבלי לבחור בדרך המהירה, החסרונות הם שלא נראה את תהליך הציור (נוכל לראות רק את התוצר הסופי).



9) **<u>אופן הציור:</u>** הכוונה בקווים היא בקווי עזר הצבועים בשחור ומסמנים את המרווח בין הפיקסלים. ככל שהרזולוציה גבוהה הפרש הקווים יהיה קטן

יותר.

10) **<u>צבע הציור:</u>** ניתן לבחור צבע מ1-6:



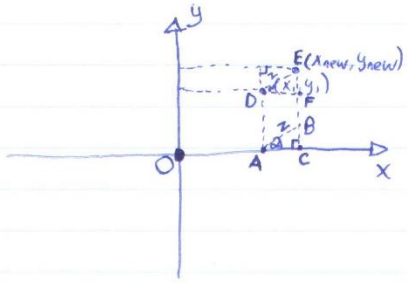
11) **<u>ציור הפונקציה:</u>**

ציור הפונקציה יתבצע בצד שמאל (הצד הכהה של הפרויקט) אחרי לחיצה על כפתור "שרטט גרף". תוכלו לראות בחלון התהליך את השלב בו הציור נמצא.

12) **<u>אפשרויות לאחר הציור:</u>**

לאחר הציור תוכלו לגלגל עם העכבר כלפי מעלה או מטה וכך הגרף יגדל או יקטן בהתאמה.

נוסחת המרה



נתון:

$Z,\ y_1,\ x_1,\ \alpha$

y_1, x_1 – שיעורי הנקודה שברצונו להמיר.

α – זווית בין Z.

(שמתקבל מהבדלת x_1, y_1 בפונקציה שמשתמש מקליד)

(טריגונומטריה ב- ΔABC) $\cos\alpha = \frac{AC}{Z} \mid \cdot Z$

$\Updownarrow$

(1) $Z \cdot \cos\alpha = AC$

(הצבת טענה (1) + חוקי הצמצום) $xNew = x_1 + AC = \boxed{x_1 + Z \cdot \cos\alpha}$

נתבונן ב ΔDEF:

(טריגונומטריה ב ΔDEF) $\sin\alpha = \frac{EF}{Z} \mid \cdot Z$

$\Updownarrow$

$Z \cdot \sin\alpha = EF$

$ynew = EF + y_1 = \boxed{y_1 + \sin\alpha \cdot Z}$

כדי להמיר את הנקודות בגרף לתלת ממדיות, השתמשתי בנוסחה:

$$newX = \ x + \cos(degrees) * z$$

$$newY = y + \sin(degrees) * z$$

הוכחת הנוסחה (נמצאת למעלה):

כאשר newY וnewX הם הנקודות החדשות שמתקבלות לאחר ההמרה של x,y לנקודות תלת ממדיות לפי זווית הציר z (degrees) וz עצמו (אורך העומק). מכיוון שברצוני להחליף בין ציר הy לציר הz כדי שהגרף יראה בצורה תלת ממדית, נחליף כל y בנוסחאות בz, וכל z בy. נקבל:

$$newX = \ x + \cos(degrees) * y$$

$$newY = z + \sin(degrees) * y$$

מכיוון שהוכחת הנוסחה נעשית לפי הצירים הרגילים של מתמטיקה, ולא כמו הצירים המוגדרים במחשב (במחשב יש רק ערכים חיוביים בצירים), הציור המתקבל יופיע בצד ימין למטה במסך של המחשב. לכן נצטרך "לשפץ" את הנוסחה על מנת שתציג את הציור באמצע המסך במחשב. לשם כך נוסיף לכל נוסחה הכפלה של כולה ב0.5.

למה דווקא 0.5?

מכיוון שהפונקציה תופיע בסוף המסך בצד ימין אם נכפיל ב0.5 את האיקסים של הפונקציה הם יעברו לאמצע המסך. אם נכפיל את הyים של הפונקציה ב0.5 הם גם יעברו מסוף המסך לאמצע. לכן נעשה זאת:

$$newX = (\ x + \cos(degrees) * y\ ) * 0.5$$

$$newY = (z + \sin(degrees) * y) * 0.5$$

גודל המערך, כפי שנאמר במדריך למשתמש, אחראי על הטווחים של x,y. כמובן שגודל המערך לא יכול להיות שלילי, ולכן נוצר מצב שתחום הפונקציה הוא רק בין x חיוביים. לכן "נשפץ" את הנוסחה עוד קצת, כדי שהיא תכלול ערכי x שליליים וכך נוכל לראות את כל הפונקציה ולא רק את ציור לפי x חיוביים. נסמן את טווח הx בoldXYZ.getLength(0) ואת טווח הy ב-oldXYZ.getLength(1):

$$newX = (\ (x - oldXYZ.getLength(0)/2) + \cos(degrees) * (y - oldXYZ.getLength(1)/2\ ) * 0.5$$

$$newY = (z + \sin(degrees) * (y - oldXYZ.getLength(1)/2)) * 0.5$$

מה שעשינו הוא בעצם להוריד לכל *x* חצי מטווח האיקסים ולכל *y* חצי מטווח הy. נניח שיש לנו פונקציה עם טווח *x* של 10 (0 עד 10).

כאשר *x=0*: *x-oldXYZ.getLength(0)/2 = 0-10/2 = -5*. אם נמשיך עד *x=10*

נקבל טווח איקסים של -5 עד 5, שזה בדיוק מה שרצינו, שהנוסחה תקלוט גם ערכים שלילים.
עכשיו התקבלה נוסחה שתוכל כבר לצייר לנו פונקציה תלת ממדית, אך לא התייחסנו עוד לרזולוציה. לצורך העניין ניקח פונקציה עם טווח *x* של 10 וטווח *y* של 10. הפונקציה תצויר מ-5 עד 5 לפי הנוסחה החדשה אך מה עם הערכים באמצע? כמו 4.5 למשל, הוא גם מ-5 עד 5 אך הוא לא ערך שלם. לשם כך נצטרך "לשפץ" עוד קצת את הנוסחה ולהוסיף רזולוציה – מרווח בין פיקסלים, כלומר, כמה קפיצות יהיו בטווח. כרגע יש קפיצה אחת, הפונקציה הנ"ל תצויר על המסך לפי הנקודות:
$(-5,y_0),(-4,y_1),(-3,y_2)\ldots(5,y_{10})$.
ולא לפי הנקודות:
$(-5,y_0),(-4.5,y_1),(-4,y_2)\ldots(5,y_{10})$ למרות שגם הנקודה $4.5,y_1$ היא חלק מהטווח. נוסיף את הרזולוציה לנוסחה:

$$newX = \left(\frac{x - oldXYZ.\frac{getLength(0)}{2}}{res} + \cos(degrees) * \left(\frac{y - oldXYZ.\frac{getLength(1)}{2}}{res}\right) \right) * 0.5$$

$$newY = \left(z + \sin(degrees) * \left(\left(y - oldXYZ.\frac{getLength(1)}{2}\right) / res \right) \right) * 0.5$$

כאשר res הוא משתנה הרזולוציה שהמשתמש יבחר. אם res = 10, הקפיצה של ערכי x,y תהיה ב1/10, וכך הלאה. כך יהיו יותר נקודות של הפונקציה מצוירות על המסך וזה יבוא לידי ביטוי בכך שנוכל לראות תמונה יותר חדה וברורה של הפונקציה. כאן סיימנו עם הנוסחה.

5 אלגוריתמים מרכזיים

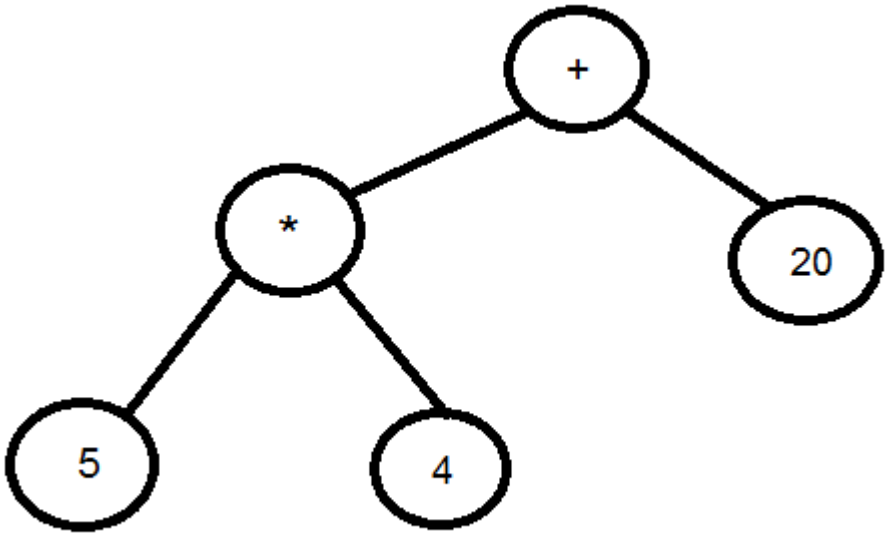
<u>האלגוריתם: בניית עץ בינארי לפי ביטוי</u>

הקריאה לשיטה מתבצעת אחרי שיטת פתיחת הסוגריים. השיטה מקבלת כפרמטר ביטוי חשבוני (למשל "3+1*2^5) ומחזירה עץ בינארי לפי קריטריונים מסוימים:

כדי שנוכל לחשב ביטוי מסוים חשבוני, נבנה עץ בינארי לפי הסדר עדיפויות הזה:

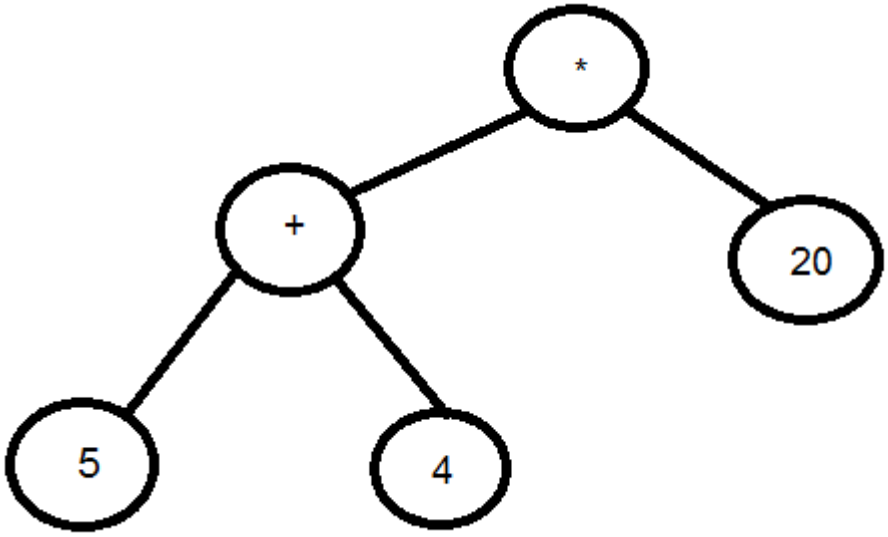
1) חיבור וחיסור בשורש כל עץ קודמים לכל.
2) אם אין פעולת חיבור או חיסור בביטוי זה, כפל וחילוק קודמים בשורש כל עץ.
3) אם אין פעולת כפל או חילוק וגם אין פעולת חיבור או חיסור בביטוי, חזקה תקדם לכל.

דוגמא לעץ בינארי כזה המייצג את הביטוי: 5*4+20



העץ עונה על הקריטריונים.

דוגמא לעץ בינארי שאינו מקיים את הקריטריונים:



אמנם תת העץ שלו 5+4 כן מקיים את הקריטריונים, אך העץ המתאר את הביטוי הכללי אינו מקיים את הקריטריונים שכן '+' אמור להיות האבא של '*' ולא ההפך.

<u>האלגוריתם (רקורסיבי):</u>

0.0 עבור על כל איבר בביטוי str שהתקבל כפרמטר

0.0.1 לכל איבר בביטוי

0.0.1.1 בדוק האם הוא '+' או '-'

0.0.1.2 אם הוא '+' או '-' בנה עץ tr חדש ששורשו הוא האיבר

הגדר את הבן השמאלי של tr :

BuildTreeOfParser(str.Substring(0,i-1))

הגדר את הבן הימני של tr:

BuildTreeOfParser(str.Substring(I,str.length-i))

0.0.2 לכל איבר בביטוי

0.0.2.1 בדוק האם הוא '*' או '/'.

אם הוא אחד מהם, בנה עץ ששורשו הוא האיבר,

הגדר את הבן השמאלי של tr:

BuildTreeOfParser(str.Substring(0,i-1))

הגדר את הבן הימני של tr:

BuildTreeOfParser(str.Substring(I,str.length-i))

0.0.3 לכל איבר בביטוי

0.0.3.1 בדוק האם הוא '^'

אם הוא '^', בנה עץ tr ששורשו הוא האיבר ('^'),

הגדר את הבן השמאלי של tr:

BuildTreeOfParser(str.Substring(0,i-1))

הגדר את הבן הימני של tr:

BuildTreeOfParser(str.Substring(I,str.length-i))

<u>האלגוריתם:</u> פתיחת סוגריים

הקריאה לשיטה מתבצעת ממחלקת main לאחר שהמשתמש שולח את מאפייני הפונקציה. השיטה מקבלת כפרמטר string שהוא מייצג את הפונקציה התלת ממדית, ומקבלת בנוסף 2 מספרים אותם היא תציב במקום x,y בהתאמה. השיטה תחשב את הביטוי (גם אם הוא כולל סוגריים) ותחזיר את התוצאה. התוצאה היא ערך הz בנקודה x,y כמובן.

0.0 חפש בביטוי y, לכל y, הצב במקומו את ערך הy של הפרמטר.

0.1 חפש בביטוי x, לכל x, הצב במקומו את ערך הx של הפרמטר.

1.0 חפש בביטוי מימין לשמאל '('. אם מצאת שמור אותו במשתנה bracketRight

1.1 חפש בביטוי מימין לשמאל ')'. אם מצאת שמור אותו במשתנה bracketLeft והפסק לחפש.

2.0 אם מצאת סוגריים, חשב את מה שבתוכם, ותעבור על כל הביטוי שוב מההתחלה.

2.1 אם לא מצאת סוגריים, בנה עץ בינארי של הביטוי וחשב אותו. החזר את התוצאה.

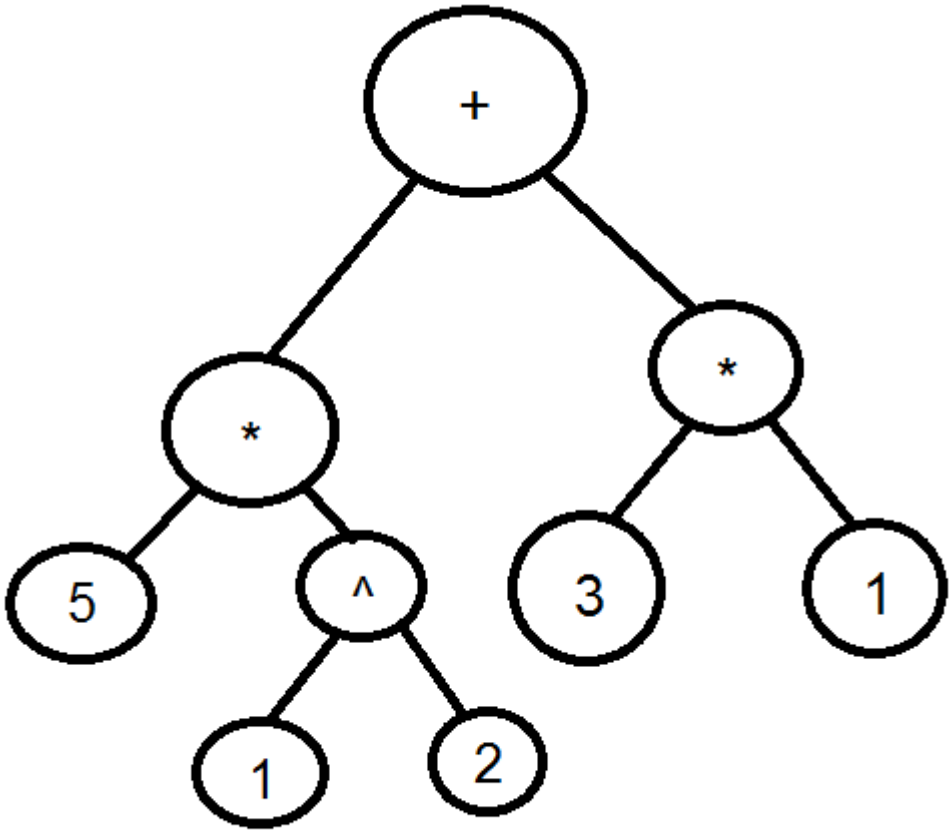
כעת, נדגים על הביטוי "5*x^2+3*y" והפרמטרים: x= 1, y= 1:

שלב ראשון: לכל y להציב "1" במקומו (כערך הפרמטר): "5*x^2+3*1"

שלב שני: לכל x להציב "1" במקומו (כערך הפרמטר): "5*1^2+3*1".

שלב שלישי : חיפוש סוגריים, מכיוון שאין סוגריים ממשיכים.

שלב רביעי: בניית עץ בינארי של הביטוי שהתקבל בשלב שני. העץ יראה כך:


+
*
*
5
^
3
1
1
2

<u>האלגוריתם:</u> המרת נקודות y, x והגדרת z.

ההמרה מתבצעת בפונקציה Draw3D. תפקיד ההמרה הוא בעצם להפוך כל x,y בפונקציה שהמשתמש הקליד לx,y חדשים שתלויים גם בz (ציר העומק) ובזווית שלו כמובן.

1.0 לכל טווח y

1.0.1 עבור על כל טווח x

1.0.2 הכנס בz[x,y] (z לפי x ו y) את ערך הפונקציה בנקודה x,y (קריאה לשיטה BracketReader(function,x,y) כאשר function היא הפונקצייה z).

1.0.3 הגדר את newX[x,y] לפי נוסחת ההמרה.

1.0.4 הגדר את newY[x,y] לפי נוסחת ההמרה.

דוגמא: נניח והקלדנו פונקציה Z = sin(x)+cos(y) והטווח הוא 2 (ממינוס 1 עד 1) האלגוריתם יתבצע ככה:

Z[-1,-1]= sin(-1)+cos(-1)

הגדרת newX[-1,-1] לפי הצבת הנתונים בנוסחת ההמרה

הגדרת newY[-1,-1] לפי הצבת הנתונים בנוסחת ההמרה

Z[0,-1] = sin(0)+cos(-1)

הגדרת newX[0,-1] לפי הצבת הנתונים בנוסחת ההמרה

הגדרת newY[0,-1] לפי הצבת הנתונים בנוסחת ההמרה

Z[1,-1] = sin(1)+cos(-1)

הגדרת newX[1,-1] לפי הצבת הנתונים בנוסחת ההמרה

הגדרת newY[1,-1] לפי הצבת הנתונים בנוסחת ההמרה

Z[-1,0] = sin(-1)+cos(0)

הגדרת newX[-1,0] לפי הצבת הנתונים בנוסחת ההמרה

הגדרת newY[-1,0] לפי הצבת הנתונים בנוסחת ההמרה

Z[0,0] = sin(0)+cos(0)

הגדרת newX[0,0] לפי הצבת הנתונים בנוסחת ההמרה

הגדרת newY[0,0] לפי הצבת הנתונים בנוסחת ההמרה

Z[1,0] = sin(1)+cos(0)

הגדרת newX[1,0] לפי הצבת הנתונים בנוסחת ההמרה

הגדרת newY[1,0] לפי הצבת הנתונים בנוסחת ההמרה

Z[-1,1] = sin(-1)+cos(1)

הגדרת newX[-1, 1] לפי הצבת הנתונים בנוסחת ההמרה

הגדרת newY[-1, 1] לפי הצבת הנתונים בנוסחת ההמרה

Z[0,1] = sin(0)+cos(1)

הגדרת newX[0,1] לפי הצבת הנתונים בנוסחת ההמרה

הגדרת newY[0,1] לפי הצבת הנתונים בנוסחת ההמרה

Z[1,1] = sin(1)+cos(1)

הגדרת newX[1, 1] לפי הצבת הנתונים בנוסחת ההמרה

הגדרת newY[1, 1] לפי הצבת הנתונים בנוסחת ההמרה

<u>האלגוריתם:</u> בחירת צבע הפיקסל לפי העומק (Z) וצביעה בין 4 נקודות.

על מנת שהפונקציה תראה תלת ממדית נצטרך ליצור גוונים לפונקציה. אם לפונקציה יהיה אותו גוון (רק שחור) לא נוכל לראות את הצורה התלת ממדית כמו שהיא. לכן נגדיר שככל שהעומק גדול יותר כך הגוון של הנקודה בעומק הזה יהיה יותר קרובה לשחור. נצטרך להתאים את זה לצורת הגוונים RGB.

לשם כך נצטרך למצוא את העומק המקסימלי והעומק המינימלי ונשווה כל פעם את העומק הנוכחי על מנת לקבל גוון בין 0-255. לאחר מכן נצבע בגוון שמצאנו בין 4 נקודות בפונקציה: הנקודה הנוכחית, הנקודה של x+1,y הנקודה של x,y+1 והנקודה של x+1,y+1.

כדי למצוא את הערך בין 0-255 שמהווה העומק הנוכחי ניעזר באלגוריתם הבא:

1) מציאת העומק המקסימלי של הפונקציה שהמשתמש הקליד בטווח שהמשתמש בחר.
2) מציאת העומק המינימלי של הפונקציה שהמשתמש הקליד בטווח שהמשתמש בחר.
3) הגדרת משתנה value שיהווה את הערך בין 0-255. ההגדרה שלו תהיה כזאת:
   value = (z-lowestZ)/(highestZ-lowestZ)*255
   כאשר z הוא העומק הנוכחי, lowestZ העומק המינימלי בפונקציה בין הטווח שהמשתמש בחר, highestZ הוא העומק המקסימלי בפונקציה בין הטווח שהמשתמש בחר.
4) צביעה בגוון שהתקבל בין הנקודות (x,y),(x+1,y),(x,y+1),(x+1,y+1).

<u>האלגוריתם:</u> ציור קו

0.0 לכל y

1.0 לכל x

0.1 הגדר נקודה p1(xNew[y,x],yNew[y,x]) (xNew הכוונה לאחר למערך של הנקודות לאחר ההמרה).

0.2 הגדר נקודה p2(xNew[y,x+1],yNew[y,x+1])

0.3 הגדר נקודה p3(xNew[y+1,x],yNew[y+1,x])

2.0 מתח קו בין p1 לp2

3.0 מתח קו בין p1 לp3

טענות כניסה יציאה

```
//ט .כניסה :הפונקציה מקבלת מחרוזת הכוללת פונקציה
//ט .יציאה :הפונקציה מפעילה לולאה שמחשבת את העומק ונקודות האיקס וואי החדשות לפי נוסחת
המרה מדו-ממד לתלת-ממד והמחרוזת המתקבלת
public void Draw3D(string function)

//הפונקציה מציירת את הגרף לפי הנקודות במערך שהוגדר בפונקציה הקודמת (Draw3D)
public void Fill3D()

//הפונקציה מגדירה מאפיינים התחלתיים
public void button1_Click(object sender, EventArgs e)

//ט .כניסה :הפונקציה מקבלת עץ בינארי
//ט .יציאה :הפונקציה מחשבת את ערך העץ לפי סדר פעולות חשבון ועוברת על העץ משמאל לימין
static double CalculateTreeOfParser(BinTreeNode<string> t)


//   ט .כניסה :הפונקציה מקבלת מחרוזת של ביטוי חשבוני
//   ט .יציאה :הפונקציה מחזירה עץ בינארי המורכב משמאל לימין לפי הביטוי
static BinTreeNode<string> BuildTreeOfParser(String str)

//ט .כניסה :הפונקציה מקבלת ביטוי אלגברי הכולל משתנה איקס ו-ואי ,ושתי מספרים ממשיים
//ט .יציאה :הפונקציה מחזירה את תוצאת הביטוי בהצבת המספרים הממשיים ב איקס ,וואי בהתאמה
public static double BracketReader(String str1, double x, double y)


//  מחלקת חולייה של עץ בינארי מטיפוס גנרי
public class BinTreeNode<T>

//מחלקה המייצגת עצים בינאריים גנריים
public class BinaryTree
```