איך כותבים את זה ב- פסקל / C#

טיפוסי משתנים:

	Pascal - פסקל	C#
מספרי שלם	**integer** (בתחום ±32,767)	**int** (בתחום ±2,147,483,648)
מספרי ארוך	**longint** (בתחום ±2,147,483,648)	**long**
מספר ממשי	**real**	**double** (float)
תו	**char**	**char**
מחרוזת	**string**	**string**

קבועים :	**const** שם-הקבוע = ערך ;	**const** <טיפוס-נתונים> <שם-הקבוע> = ערך-הקבוע ;
דוגמא :	**const** N = 10; X = 2.34;	**const int** N = 10; **const double** X = 2.34;

הגדרת משתנים:	**var** a, b : **integer**; x, y : **real**;	**int** a, b; **double** x, y;

מבנה התכנית:

Pascal - פסקל	C#

program שם-התכנית;
const
רשימת קבועים;
var
הגדרת המשתנים;
begin
הוראה;
הוראה;
end. ← שים ♥ : מסתיים בנקודה
	public class שם-המחלקה
	{
	public static void Main(**string**[] args)
	{
	הגדרת המשתנים;
	
	הוראה;
	הוראה;
	}
	}

תיעוד והערות:

הערת שורה	אין. השתמש ב- { }	// הערת שורה
הערת קטע	{ הערה } (* הערה *)	/* הערה */

- בפסקל אין חשיבות לגודל אות.
  ב- C# יש חשיבות לגודל אות (case sensitive)

קלט/פלט:

	Pascal - פסקל	C#
פלט מחרוזת	write ('מחרוזת'); writeln ('מחרוזת');	Console.Write ("מחרוזת"); Console.WriteLine ("מחרוזת");
הדפסת תוכן משתנה	write (שם-משתנה); write (שם-משתנה, שם-משתנה); writeln (שם-משתנה); writeln (שם-משתנה, שם-משתנה);	Console.Write (שם-משתנה); Console.Write (שם-משתנה + " " + שם-משתנה); Console.WriteLine (שם-משתנה); Console.WriteLine (שם-משתנה + " " + שם-משתנה);
הדפסת משתנים ומחרוזות	write (שם-משתנה, ' מחרוזת '); write (' מחרוזת ', שם-משתנה, ' מחרוזת '); writeln (שם-משתנה, ' מחרוזת '); writeln (' מחרוזת ', שם-משתנה, ' מחרוזת ');	Console.Write (" מחרוזת " + שם-משתנה); Console.Write (" מחרוזת " +שם-משתנה+ " מחרוזת"); Console.WriteLine (" מחרוזת " + שם-משתנה); Console.WriteLine ("מחרוזת"+שם-משתנה+" מחרוזת"); בתוך המחרוזת ניתן להדפיס תווים מיוחדים: \n - שורה חדשה, \t - קפיצת tab , \" - סימן הגרשיים, \\ - סימן ה- \
	var הגדרת משתנים: a : **integer**; x : **real**; tav : **char**; str : **srting**;	
קלט מספר שלם	הוראות קלט בגוף התכנית: read (שם-משתנה, שם-משתנה); read (a, x, tav, str); readln (שם-משתנה, שם-משתנה); readln (a, x, tav); readln (str);	**int** a = **int**.Parse(Console.ReadLine());
קלט מספר ממשי		**double** x = **double**.Parse(Console.ReadLine());
קלט מספר תו		**char** tav = (**char**)Console.Read();
קלט מחרוזת		**string** str = Console.ReadLine();

דגשים	• ניתן לקלוט יותר ממשתנה אחד בהוראת הקלט. • ניתן לקלוט למשתנים מסוגים שונים באותה הוראה הקלט. • לפני כל הוראת read/readln יש לרשום הוראת write/writeln שתתאר את הקלט. write ('הקש מספר שלם → '); read (a);	• בכל הוראת קלט ניתן לקלוט בדיוק משתנה אחד. • חובה לשלב משפט הסבר לפני הוראות הקלט: Console.Write("type a number → "); **int** a = **int**.Parse(Console.ReadLine());

השמה:

	a := 5;	a = 5;
אתחול בהגדרה	לא קיים (כי המשתנים מוגדרים מחוץ לתכנית ב- var). כדי לאתחל משתנה נציב לתוכו את הערך הרצוי: a := 0;	**int** a = 0; אתחול בהגדרה

פעולות חישוביות:

	Pascal - פסקל	**C#**
	var a, b, c : **integer**; x, y, z : **real**;	משתנה שלא אותחל מכיל "זבל" **int** a, b, c; **double** x, y, z;
חיבור	c := a + b;	c = a + b;
חיסור	c := a – b;	c = a – b;
כפל	c := a * b;	c = a * b;
חילוק:	שלם / שלם ← ממשי ממשי / ממשי ← ממשי שלם / ממשי ← ממשי ממשי / שלם ← ממשי	שלם / שלם ← שלם ממשי / ממשי ← ממשי שלם / שלם(**double**) ← ממשי
חילוק בממשיים • המרה מפורשת ב- C# • המרה מרומזת בפסקל	z := x / y; z := a / b;	z = x / y; z = (**double**)a / b;
חילוק בשלמים (מנה)	c := a **div** b;	c = a / b;
שארית	c := a **mod** b;	c = a % b;
דגשים: • המרה	פעולת החילוק / מבצעת המרה אוטומטית לממשיים.	ניתן להמיר משלם לממשי ולהיפך בהמרה מפורשת (casting). **double** x = a / 3.0; **double** y = (**double**)a / b;
• חילוק	האופרטור / מחזיר תמיד ערך ממשי. div ו- mod אינם מוגדרים לממשיים.	

קיצורים:

הגדלה עצמית	inc (a); { a := a + 1; }	a ++ ;
הקטנה עצמית	dec (a); { a := a - 1; }	a -- ;

הוראות השפה:

	Pascal - פסקל		C#	
סוגי יחס	שווה =	שונה <>	שווה ==	שונה =!
	גדול מ- >	קטן או שווה =<	גדול מ- >	קטן או שווה =<
	קטן מ- <	גדול או שווה =>	קטן מ- <	גדול או שווה =>

בלוק הוראות:

	begin
	בלוק הוראות
	end;
		{
		בלוק הוראות
		}

אין לשים נקודה-פסיק ; אחרי המילה begin

שים ♥ : מסתיים בנקודה-פסיק ;

אם ...
(הוראה יחידה)
	if פסוק-לוגי **then**
	ביצוע;
	
	if a > 5 **then**
	b := b + 1;
		if (פסוק-לוגי)
		ביצוע;
		
		if (a > 5)
		b = b + 1;

אם ...
(בלוק הוראות)
	if פסוק-לוגי **then**
	begin
	הוראה;
	הוראה;
	end;
	
	if a = 3 **then**
	begin
	c := b * 2;
	writeln (c , ' התוצאה :');
	end;
		if (פסוק-לוגי)
		{
		הוראה;
		הוראה;
		}
		
		if (a = = 3)
		{
		c = b * 2;
		Console.WriteLine ("result: " + c);
		}

אם... אחרת ...
(הוראה יחידה)
	if פסוק-לוגי **then**
	ביצוע1
	else
	ביצוע2;
	
	if a <> 5 **then**
	b := b + 1
	else
	b := b – 1;
		if (פסוק-לוגי)
		ביצוע1;
		else
		ביצוע2;
		
		if (a != 5)
		b = b + 1;
		else
		b = b – 1;

שים ♥ : אין לשים ; לפני else

אם... אחרת ...
(בלוק הוראות)
	if פסוק-לוגי **then**
	begin
	הוראה1;
	הוראה2;
	end
	else
	begin
	הוראה3;
	הוראה4;
	end;
	
	if a > 3 **then**
	begin
	write ('הקש מספר → ');
	read (b);
	end
	else
	begin
	b := a;
	c := c + 1
	end;
		if (פסוק-לוגי)
		{
		הוראה1;
		הוראה2;
		}
		else
		{
		הוראה3;
		הוראה4;
		}
		
		if (a > 3)
		{
		Console.Write ("הקש מספר → ");
		b = **int**.Parse(Console.ReadLine());
		}
		else
		{
		b = a;
		c = c + 1;
		}

ברירת החלטה
• המשתנה הנבדק הוא מטיפוס סדור: מספר שלם או תו
• בפסקל: מתבצעת השורה המתאימה בלבד.
• ב-C#: מבצע עד ה- break
	case משתנה **of**
	ערך-1 : ביצוע1;
	ערך-2 : ביצוע2;
	ערך-3 : **begin**
	הוראה; { ביצוע3 }
	הוראה;
	end;
	ערך-4 : ביצוע4;
	else ביצוע5;
	end;
	
	תחום ערכים - מערך .. עד ערך: 5..8
	אחד מקבוצת נערכים: 1,5,7,9,12..16
		switch (משתנה)
		{
		case ערך-1 : ביצוע1;
		break;
		case ערך-2 : ביצוע2;
		break;
		case ערך-3 : הוראה; // ביצוע3
		הוראה;
		break;
		case ערך-4 : ביצוע4;
		break;
		default : ביצוע5;
		break;
		}
		רק ערכים בדידים

שים ♥: אין לשים ; לפני else

קשרים לוגיים

וגם	**and**	**&&**
או	**or**	||
לא	**not**	**!**
	חובה לעטוף כל תנאי לוגי פשוט בסוגריים: **if** (a > 3) **and** ((b = 5) **or** (c <> 0)) **then** …	**if** (a > 3 **&&** (b = = 5 || c != 0)) …

לולאות:

לולאת for
(לולאה עולה)
	for אינדקס := ערך-תחילי **to** ערך-סיום **do**
	begin
	הוראה;
	הוראה;
	end;
	
	for I := 1 **to** 5 **do**
	begin
	write ('מספר → ');
	read (a);
	sum := sum + a;
	end;
		for (קידום-אינדקס ; ערך-סיום <= אינדקס ; ערך-תחילי = אינדקס)
		{
		הוראה;
		הוראה;
		}
		
		for (**int** i = 1 ; i <= 5 ; i++)
		{
		Console.Write ("a number → ");
		a = **int**.Parse(ReadLine());
		sum = sum + a;
		}

לולאת for
(לולאה יורדת)
	for אינדקס := ערך-תחילי **downto** ערך-סיום **do**
	begin
	הוראה;
	הוראה;
	end;
	
	for i := 5 **downto** 1 **do**
	begin
	write ('מספר → ');
	read (a);
	sum := sum + a;
	end;
		for (הקטנת-אינדקס ; ערך-סיום >= אינדקס ; ערך-תחילי = אינדקס)
		{
		הוראה;
		הוראה;
		}
		
		for (i = 5 ; i >= 1 ; i --)
		{
		Console.Write ("a number → ");
		a = **int**.Parse(ReadLine());
		sum = sum + a;
		}

לולאת while	**while** ביטוי-לוגי **do**
	begin
	הוראה;
	הוראה;
	end;
	
	sum := 0;
	write ('מספר → '); read (a);
	while a <> 999 **do**
	begin
	sum := sum + a;
	write ('מספר → '); read (a);
	end;
	writeln (sum, ' :סכום המספרים');
		while (ביטוי-לוגי)
		{
		הוראה;
		הוראה;
		}
		
		int sum = 0;
		Console.Write ("a number → ");
		int a = **int**.Parse(ReadLine());
		while (a != 999)
		{
		sum = sum + a;
		Console.Write ("a number → ");
		a = **int**.Parse(ReadLine());
		}
		Console.WriteLine ("sum = " + sum);

פונקציות מתמטיות:

	Pascal - פסקל			C#		
	מוחזר	מתקבל		מוחזר	מתקבל	
טיפוס הערך	שלם ממשי	שלם ממשי	abs (x)	שלם ממשי	שלם ממשי	Math.Abs(x)
ערך מוחלט $\|x\|$	שלם ממשי	שלם ממשי	sqrt (x)	ממשי	שלם ממשי	Math.Sqrt (x)
שורש ריבועי $\sqrt{x}$	ממשי	שלם ממשי	sqr (x)	ממשי	שלם ממשי	Math.Pow (x, 2)
חזקת 2 x^2	שלם ממשי	שלם ממשי	sqr (x)	ממשי	שלם ממשי	Math.Pow (x, y)
חזקה x^y	לא קיים (חישוב בלולאה)			ממשי	שלם ממשי	Math.Pow (x, y)
עיגול לשלם הקרוב (4 ← 3.67)	שלם	ממשי	round (x)	שלם	ממשי	Math.Round (x)
עיגול החלק העשרוני:				ממשי	ממשי	Math.Round (x,2)
החלק השלם (3 ← 3.67)	שלם	ממשי	trunc (x)	שלם	ממשי	(**int**) x

דגשים: במשפט הזימון לפונקציה חובה לציין מה לעשות עם הערך המוחזר:

- השמה: x = שם-הפונקציה (פרמטרים)
- שילוב בתנאי לוגי: if שם-הפונקציה (פרמטרים) > 25 ...
- הדפסת הערך המוחזר: הצג כפלט את שם-הפונקציה (פרמטרים)
- פרמטר לפונקציה אחרת: y = שם-פונקציה1 (שם-הפונקציה2 (פרמטרים), ערך)

מספר אקראי

בתוך גוף בתכנית:	randomize;	יצירת עצם: Random rnd = **new** Random();
מספר שלם בתחום 0 עד n (לא כולל) $0 \le x < n$	x := random (n);	**int** x = rnd.Next (n);
מספר שלם בתחום 1 עד n (כולל) $1 \le x \le n$	x := random (n) + 1;	**int** x = rnd.Next (n) + 1;
מספר שלם בתחום a עד b (כולל) a b	n := b - a + 1; x := random (n) + a;	**int** x = rnd.Next (a, b); $a \le x < b$

פונקציות ופרוצדורות	Pascal - פסקל	C#

פרוצדורה	**procedure** procName (**פרמטרים**);
	var משתנים **פנימיים** ;
	begin
	:
	end;
		static void ProcName (**פרמטרים**)
		{
		
		}

	procedure aaa (a, b: **integer**; x : **real**);
	var i : integer
	begin
	:
	end;
		static void Aaa (**int** a, **int** b, **double** x)
		{
		int i;
		:
		}

פונקציה	**function** funcName (פרמטרים):טיפוס-מוחזר;
	var משתנים פנימיים;
	begin
	:
	funcName := ערך להחזרה ;
	end;
		static טיפוס- מוחזר FuncName(פרמטרים)
		{
		:
		return ערך להחזרה;
		}

	function sum (a, b : **integer**; x : real) : **real**;
	var total : **real**;
	begin
	total := a + b + x;
	sum := total;
	end;
		static double Sum (**int** a, **int** b, **double** x)
		{
		double total ;
		total = a + b + x;
		return total;
		}

העברת פרמטרים בפסקל: לפי ערך או לפי כתובת (עם var).

העברת פרמטרים בסיסיים (int, double, char, boolean) ב- Java : **רק** לפי ערך

מערך חד-ממדי	**var** arr : **array** [1..10] **of** טיפוס-נתונים;	טיפוס-נתונים [] arr = **new** טיפוס-נתונים[10];
הגדרה	**var** a : **array** [1..7] **of integer**; x : **array** [1..N] **of real**;	**int** [] a = **new int** [7] ; **double** [] x = **new double** [N];
פנייה לתא	x [3] := 2 * a[1] ;	x[2] = 2 * a[0] ;
	a: 1 2 3 4 5 6 7	a: 0 1 2 3 4 5 6

גודל המערך בפסקל: ערך תחילי עד ערך סיום (יכול להיות כל ערך שלם ובלבד שערך תחילי יהיה קטן מערך סיום).

מספר התא הראשון במערך ב- Java יהיה תמיד 0.

סריקת מערך	**for** i := 1 **to** 7 **do** a[i] := i ;	**for** (**int** i = 0 ; i < a.Length ; i ++) a[i] = i; גודל המערך הוא שם-המערך.Length מספר התא האחרון במערך הוא Length - 1

מערך דו-ממדי (מטריצה)	**var** mat : **array** [1..5,1..7] **of** טיפוס-נתונים;	טיפוס-נתונים [,] arr = **new** טיפוס-נתונים[5,7];
	var mat : **array** [1..6,1..7] **of integer**; x : **array** [1..N,1..M] **of real**;	**int** [,] mat = **new int** [6,7] ; **double** [,] x = **new double** [N,M];
פניה לתא	סכום התא ה"ראשון" והתא ה"אחרון" במערך : mat[3,4] := mat[1,1] + mat[6,7] ;	סכום התא ה"ראשון" והתא ה"אחרון" במערך : mat[3,4] = mat [0,0] + mat [5,6];

סריקת המערך	N = מספר השורות
	M = מספר העמודות
	for i := 1 **to** N **do**
	for j := 1 **to** M **do**
	mat [i,j] := …
		mat.GetLength (0) ⇨ מספר השורות במטריצה
		mat.GetLength (1) ⇨ מספר העמודות במטריצה
		for (**int** i = 0 ; i < mat.GetLength(0) ; i++)
		for (**int** j = 0 ; j < mat.GetLength(1) ; j++)
		mat [i,j] = ….

מחרוזות	**var** str1 : string ; מחרוזת באורך 256 תווים str2 : string [20]; מחרוזת באורך 20 תווים	string str ;
	str1 := 'bla-bla-bla' ;	str = "bla-bla-bla" ; str = **new** string ("bla-bla-bla") ;
		מחרוזת ב- C# היא אובייקט
קלט	readln (str);	string str1 = Console.ReadLine();
אורך	num := length (str);	num = str.Length;
שרשור	str := str1 + str2 + 'aaa' ; str := concat (str1, str2, 'aaa');	str = st1 + str2 + "aaa" ;
השוואה	סימני השוואה : = , <> , <= , < , >= , >	**bool** equal = str1.Equals(str2) ; **int** n = str1.CompareTo (str2) ; str1 > str2 ⇨ n > 0 str1 < str2 ⇨ n < 0 str1 = = str2 ⇨ n = 0
תו i במחרוזת	ch := str [i]; 1 ≤ i ≤ length(str)	ch = str.CharAt (i) ; 0 ≤ i < str.length()
חיפוש תו	place := pos (ch, str) ; (place : **integer**)	**int** place = str.IndexOf (ch) ;
חיפוש תת-מחרוזת	place := pos (subStr, str) ;	**int** place = str.IndexOf (subStr);