# Pásztor Attila

# Algoritmizálás és programozás tankönyv az emeltszintű érettségihez

5. A TURBO PASCAL 7.0 FEJLESZTŐI KÖRNYEZET	45
5.1 TELEPÍTÉS	45
5.2. GYORSBILLENTYŰK:	49
5.3. A TP 7.0 menürendszere	
File (Fáil)	
Edit (Szerkesztés)	
Search (Keresés)	
Run (Futtatás)	53
Compile (Fordítás)	
Debug (Tesztelés)	
Tools (Eszközök)	
Options (Lehetőségek)	
Window (Ablak)	
Help (Súgó)	
Helyi menü	
5.4. AZ ELSŐ PASCAL PROGRAM	59
5.5. Egyszerű programok	60
5.5.1. feladat: Előjel szöveges kiírása	
5.5.2. feladat: Számok kiírása 0-tól a megadott határig	
5.5.3. feladat: Benne van-e a szövegben a megadott karakter	
5.5.4. feladat: Beolvasott számok legnagyobbika	
5.5.5. feladat: Átlagszámítás	
5.5.6. feladat: Számkitaláló program	69
5.6. Összefoglalás	71
5.7. FELADATOK	71

# 5. A Turbo Pascal 7.0 fejlesztői környezet

## 5.1. Telepítés

A hivatalos telepítőkészlet 4 db 1,44 MB-os floppy-t tartalmaz. Az elsőt kell behelyezni a floppy meghajtóba, és az install.exe programot kell elindítani. A telepítés lépéseit (és a használat módját) Windows XP Professional HU operációs rendszeren mutatom be.



A telepítés elkezdődik, megjelenik a következő képernyő:



A telepítésre használt floppy meghajtó megadása:







5. A Turbo Pascal 7.0 fejlesztői környezet

C:\TP\BIN       Fájl     Szerkesztés       Nézet     Kedvencek       Eszk       O Vissza       Yissza	özök Súgó 🏱 Mappák 🕼 🏂 🍞 🗙 🌱	) <b></b> .			A program a C:\TP\BIN mappában lévő TURBO
Cim 🗁 C:\TP\BIN			💌 🄁 Ugrás		
Mappák ×	Név -	Méret Típus	Módosítva 🔺		parancsikonra kattintya
Asztal	GREP2M5G.EXE	4 KB Alkalmazás	1993.03.03. 7:01		P
Dokumentumok	f GREP2M5G.PAS	2 KB Delphi Source File	1993.03.03. 7:01		indítható (ha nem változ-
🖃 星 Sajátgép	GREP.COM	7 KB MS-DOS alkalmazás	1993.03.03. 7:01		
🗉 뷇 3,5"-es hajlékonylemez (A:)	MAKE.EXE	69 KB Alkalmazás	1993.03.03. 7:01		tatta mag a talapítási útvo
E 🥪 VAIO C (C:)	PRNFLTR.EXE	7 KB Alkalmazás	1993.03.03. 7:01		talla meg a lefepilesi ulvo-
E Cocumentation	SP PRNFLIR.PA5	13 KB Delphi Source Hile	1993.03.03. 7:01		
Documents and Settings		107 KB Alkalmazas	1993.03.03		nalat).
Drivers	TENC EVE	57 KB Alkalmazár	1993.0		
H D FPC	S THELP CES	1 KB Microsoft Office Or	1:59		
Oktatoprogram	THELP.COM	11 KB MS-DOS alk	03.03.7:01		
	TPC.CFG	1 KB Miga	2006.12.24.21:59		
BGI	TPC.EXE	74 KB	1993.03.03. 7:01		
Din Bin	TPUMOVER.EXE	almazás	1993.03.03. 7:01		
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	TPX TPX	KB Parancsikon MS-DO	. 1993.03.03.7:01		
EXAMPLES	TPX.EXE	464 KB Alkalmazás	1993.03.03. 7:01		
SOURCE	TPX.ICO	1 KB Ikon	1993.03.03. 7:01		
C UNITS	TPX.TP	4 KB TP fájl	2006.12.24. 21:59		
🗉 🛅 Utilities	TE IUREO	1 KB Parancsikon MS-DO	. 1993.03.03. 7:01		
Image: Marchain Ma		395 KB Alkalmazas	1993.03.03. 7:01		
E C WINDOWS		I KB IKON	2006 12 24 21/50		
E C Worklocal		ARS VR TPH fáil	1993 03 03 7:01		
H VAIO D (D:)		48 KB TPI fáil	1993.03.03. 7:01		
H      MemoryStick (F:)	1992 02 02 7:01 Márst: E4E báit	EAE háit			Természetesen készíthető
pipes. Parancsikon MD-DOD programmoz Modosicva.	1990/03/03/ 7/01 Maret: 343 bajt	jono bajc 🛛 🕃 .	bajargep ///		
					narancsikon a Windows
					purunesikon a windows
and the second					asztalon is a szokott mó-
s 差 🕹					don Ügyelni arra hogy
					don. Ogychin arra, hogy
TUDDO					az indítási mappa a
TURBO					
				-	C:\IP\BIN legven
					C. (II (DII) ICR (CII)

A program indítását követően az alábbi képernyőt láthatjuk:

💽 Turbo Pa	scal 7.0									
File	Edit	<mark>- S</mark> earc	ch <mark>R</mark> un	Compile	Debug	Tools	Options	Window	Help	
<b>[</b> ]]				—— N	Ionameøø	. PAS —				-1-[\$]
										<u>^</u>
	4.4									
E1 Hol	n E2	Sauo	E3 Open	01++E0	Compile	EQ Ma	ko 01+≠E	10 Local	MODU	

Sajnos a mai korszerű (gyors) gépeken a program már a képernyőt kezelő legegyszerűbb eljárás hívása közben is (**ClrScr**) 0-val osztási hiba miatt "elszállna", ezért a telepítés után keresd fel a következő weboldalt:

http://www.borland.hu/products/downloads/tpascal.html

Töltsd le a "Runtime error 200" hiba javítását és a magyar nyelvű súgót! A telepítéshez kövesd a tömörített állományokban lévő útmutatókat!

## 5.2. Gyorsbillentyűk:

A teljesség igé	nye nélkül felsorolom a leggyakrabban használt gyorsbillentyűket:
F1	Súgó kezdőoldalának indítása.
F2	Mentés.
F3	Fájl megnyitás a szerkesztő ablakban.
F9	Készítés.
ALT+F9	Fordítás.
CTRL+F9	Fordítás és futtatás.
ALT+F10	Helyi menü (megegyezik az egér jobb gombjának megnyomásával).
ALT+X	Kilép a programból.
F8	A következő utasítást elvégzi.
F7	A következő utasítást úgy végzi el, hogy az eljárásokba belelép.
F4	A kurzorig fut a program.
CTRL+F2	A futó programot megállítja.
ALT+F5	Vált a kimeneti ablak és a fejlesztői ablak között.
CTRL+F4	Kifejezést kiértékel, illetve változó értékét állítja.
CTRL+F7	A változót felveszi a megtekintendők közé.
CTRL+F5	Ablak átméretezés, mozgatás.
F5	Az aktív ablak teljes méretű lesz.
F6	A következő ablakot teszi aktívvá (sorszáma a címsorán jobbra látható).
SHIFT+F6	Az előző ablakot teszi aktívvá.
ALT+F3	Bezárja az aktív ablakot.
ALT+0	Az ablakok listája.
ALT+szám	A számnak megfelelő ablakot teszi aktívvá.

# 5.3. A TP 7.0 menürendszere

## File (Fájl)

A		
File Ec	lit s	Search
New Open Save Save as	5	F3 F2
Save al	l dir.	
Print Printer DOS she	setu ell	up
Exit		Alt+X

A *New* –val új, üres szerkesztő ablakot nyithatunk, melyből új fájlba lehet menteni a programot.

Az *Open*-nel korábban elkészített programot lehet betölteni, a mappák között is lehet navigálni (a mappa neve után látható egy fordított perjel:  $\$ 

#### A fájl megnyitás ablaka:

<b>[</b> ]	• Open a File ——	]	
Hame * PAS	Û I T	<u> Înen</u>	
iles			
GREP2MSG.PAS PRNFLTR.PAS		Replace	
		Cancel	A
			V
••••••••••••••••••••••••••••••••••••••		петр	•
GREP2MSG.PAS 133	5 Mar 3, 19	93 7:01am	

Programot menteni a **Save**, illetve a **Save as** menüponttal lehet a szokásos módon. Ügyeljünk arra, hogy az állományok neve megfeleljen a DOS állománynévre vonatkozó szabályainak (legfeljebb nyolc karakter, magyar ékezetes betű ne legyen benne, ...).

Az alapértelmezett könyvtár beállítása a *Change Dir* segítségével adható meg. Minden alkalommal győződjünk meg arról, hogy a megfelelő helyre mutat:

[•]	= Change Directory	
Directory	ame I <sup>n</sup> lll	r
	U	01/
Directory Directory	ree	
C:\ TP BIN		Chdir
		Revert
		Help

A program a *Print* menüponttal kinyomtatható. A *DOS shell*-lel egy parancssor nyitható, melyből exit-tel lehet visszalépni, bár ennek a Windowsos felület alatt nincs sok jelentősége. A TP programból az *Exit*-tel lehet kilépni.

## Edit (Szerkesztés)

Edit \$	earch	Run	C
Undo	Alt+	BkSp	
Cut	Shift	+Del	1
Copy Paste Clear	Ctrl Shift Ctrl	+Ins +Ins +Del	
<mark>S</mark> how clipboard			

A szokásos szövegszerkesztési funkciók végezhetők el vele. Érdemes megjegyezni a *Cut* (kivágás), a *Copy* (másolás) és a *Paste* (beillesztés) gyorsbillentyűit. A vágólap megtekinthető és tisztítható.

### Search (Keresés)



A *Find* menüponttal lehet keresni a programszövegben.

A kereséshez kiválasztható lehetőségek:

Case sensitive – megkülönbözteti a kis- és nagybetűket Whole words only – csak teljes szóra keres Regular espression – reguláris kifejezést keres

Global – teljes szövegben keres Selected text – a kijelölt szövegben keres

Forward – a további szövegben keres Backward – a megelőző szövegben keres

A keresett szöveg a "Text to find" mezőbe írandó!

A keresés ablaka a következő:

[•] Fir	nd
lext to find WriteLn_	Û <mark>.</mark> Ţ
Options [] Case sensitive [] Whole words only [] Regular expression	Direction (*) Forward ( ) Backward
Scope (•) Global ( ) Selected text	Origin (•) From cursor ( ) Entire scope
OK	Cancel Help

A *Replace* menüponttal szöveget lehet cserélni. Egyetlen mezőben különbözik az előzőtől: a "New text" mezőben adható meg, hogy a keresett szöveget mire kell lecserélni.

Íme a csere ablaka:

[•] Repla	ice
ext to find <u>WriteLn</u>	Ů <mark>.</mark> Ţ
New text Write	Û <mark>.</mark> Ţ
Options [ ] Case sensitive [ ] Whole words only [ ] Regular expression [X] Prompt on replace	Direction (•) Forward ( ) Backward
Scope (•) Global ( ) Selected text OK Change all	Origin (•) From cursor ( ) Entire scope Cancel Help

## Run (Futtatás)



Run	A program futtatása (szükség esetén fordítással).
Step over	Utasítás végrehajtása.
Trace into	Utasítás végrehajtása (belép az eljárás törzsébe).
Go to cursor	A kurzor helyéig futtat.

## Compile (Fordítás)



*Compile Destination Memory* Destination Disk A forráskódot fordítja, de nem futtatja (ellenőrzéshez hasznos). A fordítás helye a memória (rákattintva változik). A fordítás helye a diszk (rákattintva változik).

Figyelem!

Érettségin, versenyen a forrás és a lefordított változat beadását is kérik, ezért a **Destination Disk** lehetőséget kell kiválasztani (rákattintva a sorra)! Ha nem változtatunk az alapbeállításokon, akkor a futtatható állomány a forrásállomány mentési helyének megfelelő mappában keletkezik programnév.exe névvel.

## Debug (Tesztelés)

Debug Tools Opti	ions Window
Breakpoints Call stack Register Watch Output	Ctrl+F3
User screen	Alt+F5
Evaluate/modify. Add watch Add break <mark>p</mark> oint	Ctrl+F4 Ctrl+F7

Breakpoints	Töréspontok megtekintése és törlése
-	Hatására a forráskód egy sora kijelölődik.
	Futtatáskor újabb parancsig itt megáll a futás.
Add breakpoint	Töréspontot lehet elhelyezni (törölni).
Watch	A futás megszakadásakor változók/kifejezések értéke tekinthető meg.
Add watch	Változó/kifejezés hozzáadása a watch ablakhoz.
Output	A program kimeneti ablakát nyitja meg (kicsiben).
User screen	A program kimeneti ablaka teljes méretben.

## Tools (Eszközök)

Tools	Options	Window	Не
Messag Go to Go to	pes next previous	Alt+F8 Alt+F7	
Grep	ę	Shift+F2	

Itt jelennek meg a külső programok, amelyeket a TP 7.0 környezetben meghívhatunk. Az alap funkciókhoz nem kell használni.

## **Options (Lehetőségek)**

Options Window	He
Compiler Memory sizes Linker Debugger Directories Tools	
Environment •	
Open Save TURBO.TP Save as	

A program egyes tulajdonságainak beállítására szolgál, amelyek elmenthetők, és indításkor automatikusan betölthetők.

Open	Elmentett konfiguráció betöltése (alapértelmezetten automatikus).
Save	Elmenti a beállításokat.
Save as	Megadható helyre menti el a beállítások.
	Érdemes az indítási mappába menteni a beállításokat.

### Compiler (A fordító beállításai)

[•]———— Compile	er Options ———————
Code generation	[V] Word align data
[] Overlays allowed	[] 286 instructions
Runtime errors [] Range checking [X] Stack checking [X] I/O checking [] Overflow checking Debugging	Syntax options [X] Strict var-strings [] Complete boolean eval [X] Extended syntax [] Typed @ operator [] Open parameters
[X] Local symbols	[] 8087/80287 [X] Emulation
	K Cancel Help

Az alapbeállítások megtartása mellett válasszuk a következőket:Range checkingTömbök túlcímzésének ellenőrzésére jó például.Overflow checkingTúlcsordulást vizsgál.

 $\mathbf{\mathbf{N}}$ 

## Memory sizes... (Memória méretek)

[•] Memory Sizes	;
<mark>S</mark> tack size	16384
ow heap limit	0
¦igh heap limit	655360
OK Cancel	Help

Alap esetben nem igényel módosítást.

#### Directories (Könyvtárak)

<b>[</b> [•]	= Directories			<b>b</b>
EXE & TPU directory			Ů <mark>.</mark> Ţ	
Include directories			Ů <b>.</b> Ţ	
Unit directories	C:\TP\UNITS		Ů <mark>IJ</mark> Ţ	
<pre>bject directories</pre>			Ů <mark>IJ</mark> Ţ	
	ОК	Cancel	Help	

Az egyes fordítási egységek helye adható meg. Ellenőrizzük, hogy a "Unit Directories" a megfelelő helyre mutat. A többi megadása nem kötelező, s ekkor a forrással megegyező mappát fog használni.

#### Environment (Környezet)

Options Window Hel
Compiler Memory sizes Linker Debugger Directories Tools
Environment
Preferences Editor Mouse Startup Colors

A környezeti beállítások itt végezhetők el. Nem szükséges semmit módosítani, a festéktakarékos nyomtatás miatt én átalakítottam a színbeállításokat.

[=[•]	Colors		1
Call stack Compiler Desktop Dialogs Editor Help Menus Messages Output Register Syntax	Item Frame passive Frame active Frame icons Scroll bar page Scroll bar icons Normal text Selected text Error message Breakpoint Source position	Foreground ackground text Text Text Text Text Text Text Text Text	
Environment/Profer	<b>OK</b> Cances meniinont	el Help	$\sqrt{0}$
[*]		<u> </u>	
Screen sizes (•) 25 lines ( ) 43/50 lines	Source tracki (•) New wind s () Current	ng Iow window	
Auto save []Editor file []Environmen []Desktop	Options es [] Auto tra t [X] Close on [] Change d	ck source go to source ir on open	
Desktop file ( ) Current di (•) Config fil	rectory e directory		
	OK Cancel	Help	

Environment/Colors/Editor menüpont (Editor színkészletének testreszabása)

A 43/50 soros képméret választással jobban áttekinthető a forráskód.



Az ablakok kezelése a szokásos lehetőségekkel: mozaikos, lépcsőzetes elrendezés, méretezés, mozgatás, nagyítás, aktiválás, bezárás.

### Help (Súgó)

tions Window Help	
ContentsIndexShift+F1Topic searchCtrl+F1Previous topicAlt+F1Using helpFiles	
Compiler directives Reserved words Standard units Turbo Pascal Language Error messages	
About	

A súgó menüpontban az ismertetőmből hiányzó részletek is megtalálhatók, akár magyar nyelven is.

Egy utasításon állva a CTRL+F1 gyorsbillentyűvel azonnal segítséget kap.

## Helyi menü

<mark>Cut</mark>	Shift+Del
Copy	Ctrl+Ins
Paste	Shift+Ins
Clear	Ctrl+Del
Open <mark>f</mark> ile at curs	or
Topic <mark>s</mark> earch	Ctrl+F1
Toggle breakpoint	Ctrl+F8
Go to cursor	F4
Evaluate/modify	. Ctrl+F4
Add watch	Ctrl+F7
<mark>O</mark> ptions	

A szerkesztő ablakban az egér jobb gombjával hívható elő, és a legfontosabb menüpontokat tartalmazza.

Kezdjünk hozzá az első programhoz!

## 5.4. Az első pascal program

Indítsuk el a Turbo Pascal 7.0 programot, és a kurzort a megjelenő szerkesztő ablakban letéve gépeljük be a következő sorokat (a nyomtató festékhasználatának csökkentésére az alapbeállítástól eltérő színkészletet használok)!



Néhány tanács:

A sorok végén ENTER-t ütünk (ez nem szövegszerkesztési feladat)!

A programsorok behúzását szóközökkel, vagy tabulátorokkal végezhetjük.

Az utasításokat pontosvessző választja el egymástól.

A kisbetű/nagybetű használata csak az olvashatóságot segíti.

Futtasuk le a programot (CTRL+F9)!



Szinte hihetetlen, de elkészült az első pascal program (szinte minden programozás tankönyv ezzel a példával kezdi, én sem akartam a szokásostól eltérni).

Mentsük el a forráskódot!

*File/Save as*, majd adjuk meg a nevet: *elso* (a pas kiterjesztést nem szükséges megadni). Katttintsunk az *OK*-ra, vagy üssünk *ENTER*-t!

<pre>[•] ave file as elso_</pre>	Save File As —— Ŭ <mark>J</mark> T	OK	
Files GREP2MSG.PAS PRNFLTR.PAS \			5
		Cancel Help	
C:\TP\BIN\*.PAS GREP2MSG.PAS 133	5 Mar 3, 19	993 7:01am	

Ügyeljünk arra, hogy a program készítése közben gyakran mentsünk (az első mentést követően már elegendő az F2 gyorsbillentyűt használni)!

## 5.5. Egyszerű programok

Ebben a részben néhány egyszerű, de komplett programot készítünk.

### 5.5.1. feladat: Előjel szöveges kiírása

Készítsünk programot, amelyik beolvas egy egész számot, és szövegesen kiírja, hogy a beolvasott szám negatív, nulla, vagy pozitív. Az algoritmust már elkészítettük a sokágú elágazások tárgyalásánál, csupán a beolvasással kell kiegészíteni.

ALGORITMUS	TURBO PASCAL KÓD
Program előjel:	Program elojel;
	uses crt;
Változó i:egész	Var i:Integer;
	Begin
Be: i	Write('Adj meg egy egész számot: ');
	ReadLn(i);
Ha i>O akkor Ki: 'Pozitív.'	If i>0 then WriteLn('Pozitív.')
különben Ha i=0 akkor Ki: 'Nulla.'	else If i=0 then WriteLn('Nulla.')
különben Ki: 'Negatív'	else WriteLn('Negatív.');
Program vége.	End.

A példából látszik, hogy az algoritmust sorról-sorra átkódoltuk, csupán a beszédesebb beolvasásnál alkalmaztunk a programban eltérő viselkedést.

Gépeljük be a kódot a Turbo Pascal szerkesztő ablakába, és mentsük el elojel.pas néven! Egészítsük ki a kódot a képernyőtörléssel (**ClrScr**), és a végén egy gombnyomásra várással: **Repeat until KeyPressed;** (így nem kell a kimeneti ablakra átváltani).

A program szövege a következő képen látható:



Ellenőrizzük a program működését mindhárom lehetséges esetre (CTRL+F9 a futtatás gyorsbillentyűje)!

Mivel egy számra egyszerre csak az egyik feltétel teljesülhet, ezért a sokágú elágazást megvalósító szerkezet most egyszerűsíthető az alábbi módon (elojel1.pas):



Figyelem! A Windows XP DOS ablakában futtatott TP esetén a < relációjel beírása az **ALTGR+M** billentyűkkel lehetséges (a felirattal ellentétben).

Van még egy "érdekes" jel, a szorzás jele, a \*. Ezt az ALTGR+P billentyűkkel lehet bevinni.

### 5.5.2. feladat: Számok kiírása 0-tól a megadott határig

Készítsünk programot, amely beolvas egy egész számot, és kiírja a természetes számokat (0-tól kezdve) növekvő sorrendben a megadott számig.

ALGORITMUS	TURBO PASCAL KÓD
Program szamok:	Program szamok;
	uses crt;
Változó i,n:egész	Var i,n:Integer;
	Begin
	Write('Adj meg egy egész számot: ');
Be: n	ReadLn(n);
	WriteLn('Természetes számok ',n,'-ig');
Ciklus i:=1-től n-ig	For i:=0 to n do
Ki: i	WriteLn(i);
Ciklus vége	
Program vége.	End.

A program kódja a szerkesztőablakban (a szokásos kényelmi kiegészítésekkel):



A futás eredménye, ha n=5-öt adunk meg:

```
Adj meg egy egész számot! 5
Természetes számok 5-ig:
0
1
2
3
4
5
Nyomj egy billentyűt!
```

Szándékosan szúrtam be a képet eltérő színekkel takarékosságból. Ezután mindig ezt a színösszeállítást használom a képernyőkép beillesztésére.

A példából az is látható, hogyan lehet egyetlen WriteLn utasítással több kifejezést kiíratni.

### 5.5.3. feladat: Benne van-e a szövegben a megadott karakter

Készítsünk programot, amelyik beolvas egy szöveget, majd egy karaktert, és kiírja, hogy a megadott karakter benne van-e a szövegben. Feltételezhetjük, hogy a szöveg nem hosszabb 255 karakternél.

ALGORITMUS	TURBO PASCAL KÓD
Program szoveg:	Program szoveg;
	uses crt;
Változó c:karater	Var c:Char;
s:szöveg	s:String;
	Begin
Be: s,c	WriteLn('Adj meg egy szöveget: ');
	ReadLn(s);
	<pre>Write('Adj meg egy karaktert: ');</pre>
	ReadLn(c);
Ha BenneVan(c,s)	If $Pos(c, s) > 0$
akkor Ki: 'Benne van.'	then WriteLn('Benne van.')
különben Ki: 'Nincs benne.'	<pre>else WriteLn('Nincs benne.);</pre>
Elágazás vége	End.
Program vége.	

A program kódja a szerkesztőablakban (a szokásos kényelmi kiegészítésekkel):



A **Pos(substr,str)** függvény megvizsgálja, hogy a **substr** szöveg hányadik karaktertől található meg az **str** szövegben. 0-t ad vissza, ha nincs benne.

Ellenőrizzük a programot igaz és hamis esetre is.

Vegyük észre, hogy a Pos függvény sajnos különbözőnek tekinti a kis-, és nagybetűket.

#### Módosítás:

A program kezelje azonosan a kis-, és nagybetűket!

A program írjon ki felülről a második sorban középre egy tájékoztató szöveget! A billentyű megnyomására várakozás előtt a legalsó sorban középen írjon ki figyelmeztetést!

#### Megoldási javaslatok:

Az UpCase(c) függvény a c karakteres változót nagybetűre konvertálja (az angol abc szerint). Egy szöveg (string) karakterenként nagybetűsítő ciklus segítségével. A szöveg hoszszát a Length(s) függvénnyel lehet megtudni.

A képernyő bal felső sarkának koordinátája az (1,1), a **GotoXY(oszlop,sor)** mozgatja a kurzort a megadott helyre.

A képernyő közepére úgy lehet pozícionálni, hogy a lehetséges karakterszámból levonjuk a kiírandó szöveg hosszát, majd a kapott értéket elosztjuk 2-vel, hiszen ennyivel kell bentebb kezdeni a kiírást. Közben nem szabad elfelejteni, hogy a koordináták egész számok, és egész számok körében az egész osztás értelmezett:

A DIV B A osztva B-vel egész osztás hányadosaA MOD B A osztva B-vel egész osztás maradéka

A módosított pascal kód:

🖬 Turbo Pascal 7.0	
File Edit Search Run Compile Debug Tools (	Options Window Help
SZOVEG1.PAS —	1-l <b>t</b> ]-j
program szoveg;	•
USES CTI; Const titetainen?Venelten el%fondellés vissenfleter	
tonst tistring- Karakter elolurdulas vizsgalata s	(* TP tipusos konstans *)
Var cichar:	(* ir tipusus konstans *)
var C.onar, s:Stripa:	(* a beoluasandó szöuga *)
i:Integer:	(* ciklusuáltozó *)
Regin	
ClrScr:	(* képernvő törlés *)
GotoXY((80-Length(t1)) DIV 2.2): WriteLn(t1):	(* üdvözlő szöveg 2. sor*)
<pre>GotoXY(1,4); WriteLn('Adj meg egy szöveget!');</pre>	(* s szöveg beolvasása *)
ReadLn(s);	
<pre>For i:=1 to Length(s) do s[i]:=UpCase(s[i]);</pre>	(* s nagybetűsítése *)
Write('Adj meg egy karaktert: ');	(* c karakter beolvasása *)
ReadLn(c);	
c:=UpUase(c);	(* c_nagybetusitese *)
If Pos(c,s)>0 then WriteLn( Benne van. )	(* vizsgalat *)
else writeLn( Nincs Denne. ); CataVU((00 Langth(+2)) DTU 2 25), Unita(+2).	(v - fuf V- out v)
Denost until KouProceed	(* zaro uzenet *)
Fod	
F1 Help F2 Save F3 Open Alt+F9 Compile F9 Make	e Alt+F10 Local menu

Ellenőrizzük ezt is mindkét lehetséges kimenetelre!

Gondoljuk át, hogy miként lehetne megoldani azt, hogy a magyar abc-re is helyesen működjön a program?

Hogyan lehetne kezelni a dupla mássalhangzókat?

### 5.5.4. feladat: Beolvasott számok legnagyobbika

Készítsünk programot, amely először beolvas egy pozitív egész számot (ellenőrzi), majd beolvas ennyi darab valós számot, és kiírja közülük a legnagyobbat (eltárolás nélkül)! Megoldási javaslat:

Az első szám beolvasásakor azt tekintjük a legnagyobbnak, a további beolvasások után megvizsgáljuk, hogy a most beolvasott szám nagyobb-e az eddig eltároltnál: ha nagyobb, akkor ezt tároljuk el legnagyobbként.

ALGORITMUS	TURBO PASCAL KÓD				
Program legnagy:	Program legnagy;				
	uses crt;				
Változó i,db,sorsz:egész	<pre>Var i,db,sorsz:Integer;</pre>				
x,max:valós	x,max:Real;				
	Begin				
	Repeat				
Be: db [db>0]	Write('Adj meg egy pozitív számot: ');				
	ReadLn(db);				
	until (db>0);				
Ciklus i:=1-től db-ig	For i:=1 to db do				
	Begin				
Be:x	Write(i,'. szám: ');				
	ReadLn(x);				
Ha i=l akkor max:=x : sorsz:=i	If i=1 then				
	Begin				
	max:=x;				
	sorsz:=1;				
	End;				
Ha X>max akkor max:=x : sorsz:=1	li x>max then				
	Begin				
	maxx,				
	501521, End.				
Ciklus váco	End.				
Ki. may	WriteLn(/A legnagyobb: / max):				
Ki: sorsz	WriteLn('A(z) '.i.' szám volt '):				
Program vége	End.				
110910					

A pascal kód (43/50 soros üzemmódra váltva):

🛤 Turbo Pascal 7.0	
File Edit Search Run Compile Debug	Tools Options Window Help
LEGNAGY.P	IAS1-[‡]
program legnagy;  llsos_crt:	
Const t1:string='Számok közül melvik a leg	nagyobh':
t2:string='Nvomi meg egy gombot!':	(* TP tipusos konstans *)
Var i,db:Integer;	(* ciklusváltozó, darabszám *)
sorsz:Integer;	(* ennyiedik volt a legnagyobb *)
x,max:Real;	(* beolvasandó, legnagyobb *)
Begin	(* kénernuő törlég *)
GotoXV((80-Length(t1)) DTV 2 2): Writeln	(t1) (* üduözlő szöuga 2 sor*)
Repeat	(11), (" uuvozio szoveg z. soi")
Write('Adi meg egy pozitív egész számo	(t: '): (* darabszám beolvasása *)
ReadLn(db);	
until (db>0);	
For 1:=1 to db do	(* ciklusban olvas, kiertekel *)
Begin White(i, ', optime, '))	
$\frac{m}{Readlen(x)}$	(* beoluasás tájékoztatással *)
If i=1 then	(* az első beolvasott egyben max *)
Begin	
max:=x;	
sorsz:=i;	
End;	
Bogin	(* na a most beolvasott nagyobb *)
max:=v:	(* ez resz a regnagyobb */
sorsz:=i:	
End;	
End;	
<pre>WriteLn('A legnagyobb: ',max);</pre>	(* c karakter beolvasása *)
WriteLn( H(Z) ,1, . SZAM VOIT. ); CotoVU((80_Longth(+2)) DTV 2 25); Write(	+2 (x záró üzenet x)
Repeat until KeuPressed:	(* gombnuomásra vár *)
End.	
L <sub>*</sub>	
F1 Help F2 Save F3 Open Alt+F9 Compile	F9 Make Alt+F10 Local menu

Futtatás során tapasztaljuk, hogy a valós számokat a pascal normálalakban írja ki, például az 50-et így: **5.00000000E+01**, azaz 5-ször 10 az első hatványon. Az 5 a mantissza, az 1 a karakterisztika.

Tekintsük át röviden a lebegőpontos és fixpontos kijelzés lehetőségeit:

📧 Turbo Pa	scal 7.0									
File	Edit	<mark>S</mark> earch	Run	Compile	Debug	Tools	Options	₩indow	Help	
<b> _</b> [ • ]─					VALOS.P	AS ——				1=[1]
progra	m <sub>_</sub> valo	s;								
Uses	rt <u>;</u> j	4004567								
Const	a:Real	=123456/	89,12	3456789;						
	b:Real	=-987654	321.9	87654321;						
Begin										
UIr8	cr;	1								
Writ	eLn∏L	ebegopon	tos a	lakok:`);			<pre>call = a</pre>			
Writ	eLn	1:[,a];		{ pozi	tiv for	mazas n	elkul }			
Writ	eLn	2:[, <b>b</b> ];		l nega	itiv for	mazas n	elkul}			
Writ	eLn	3:[,a:1]	i	tul į	rovidh	oșsz -	a tenyleg	ies hossz	abb les	z }
Writ	eLn	4:, <b>a</b> :14	Į;	l az a	prazola	sipont	ossagnal	rovidebb	hossz	}
Writ	eLn	5: , <b>a</b> :20	9:	l az a	ibrazola	s ipont	ossagnal	nagyobb	hossz }	
Writ	eLnijF	ixpontos	alak	ok:[];						
Writ	eLnţ	6:, <b>a</b> :22	:8/;	f pozi	tiv, ab	razolas	i pontoss	agnal ho	sszabb	} II
Writ	eLn	7:,, <b>b</b> :22	:87;	{ nega	itiv, ab	razolas	1 pontoss	agnal ho	sszabb	}
Writ	erut	8:,, <b>a</b> :Z:	21:	i tul	rovia n	osszak	1			
Writ	eLn	9:,a:12	:0);	( csak	egesze	k, abra	zolasnal	nagyobb	hossz }	
Writ	eLn( 1	U: ,D:12	:0);	E CSAK	egesze	к (-),	abrazolas	nai nagy	opp nos	SZ }
кере	at unt	11 KeyPr	essea	;						
End.										
	15.00									
	-13:20 n <b>E2</b>	Saus E2	- Onon	01++E0	Panaila	EQ M-		10		
гт пет	p FZ	save FJ	open	HI(+F9	compile	r⊅ Ma	Ke HIT+F	To Local	menu	

A kimeneti ablak képe:

```
1: 1.2345678912E+08

2:-9.8765432199E+08

3: 1.2E+08

4: 1.2345679E+08

5: 1.2345678912E+08

Fixpontos alakok:

6: 123456789.12000000

7: -987654321.99000000

8:123456789.12

9: 123456789

10: -987654322
```

Következtetések lebegőpontos alakra:

A formázás kivitelezése: WriteLn(kifejezés:jegyek száma);

- Mantisszája az ábrázolásnak megfelelő pontosságú (egy egész jeggyel), ha nem adunk meg formázó utasítást. A karakterisztika két egész számjegyből áll. Pozitív szám esetén kihagyja az előjel helyét a számjegy bal oldalán. Így a szám szélessége: előjel + egész + tizedespont + 10 tizedes + E + 2 jegyű karakterisztika, összesen: 17 karakter. (1,2)
- Ha túl rövid hosszt adunk meg, akkor a mantissza 1 tizedes jegyig fog látszani. (3)
- Ha a hossz rövidebb az ábrázolási pontosságból fakadónál, akkor a mantisszát kerekíti a szükséges mértékben. (4)
- Ha a hossz túl nagy, akkor a számjegyek bal oldala szóközzel feltöltődik (5)

Következtetések a fixpontos alakra:

A formázás kivitelezése: WriteLn(kifejezés:összes jegyek száma:tizedes jegyek száma);

- A hosszba az előjel is beleértendő.(7)
- Ha a megadott hossz túl nagy, akkor a tizedes jegyek 0-val feltöltődnek a megadott tizedes jegy számig, az egészek pedig balról szóközzel töltődnek fel. (6,7).
- Túl rövid hossz megadása esetén minden szükséges egész számjegy látszik, de csak a formázásnál megadott tizedes jegy jelenik meg. (8)
- A tizedes jegyek a megadott pontosságra kerekítődnek. (9)
- Az összes jegyek számába az előjel beletartozik. (10)

## 5.5.5. feladat: Átlagszámítás

Az algoritmizálás (tervezés) fejezetben készítettünk egy átlagszámító programot, amelyben eljárást és függvényt is használtunk. Utolsó egyszerű példaként kódoljuk most Turbo Pascal nyelven!

Íme az algoritmus:

```
Program:
    Változó: N: egész
                               [íqy adom meg a sorozatot]
             A(1..N: valós)
    Sorozatbeolvasás (N, A)
    Átlagkiírás(N,A)
Program vége.
Eljárás Sorozatbeolvasás (változó DB:egész,
                     változó Sorozat(1..N): valós):
    DB:=0
    Ciklus amíg DB<1
         Ki: 'Adja meg a sorozat elemszámát:'
         Be: DB
    Ciklus vége
    Ciklus i:=1-től DB-ig
         Be: Sorozat[i]
    Ciklus vége
Eljárás vége.
Függvény Átlag(konstans DB:egész,
              konstans Sorozat(1..N:valós)): valós
Változó összeg: valós
    összeg:=0
    Ciklus i:=1-től DB-ig
         összeg:=összeg+Sorozat(i)
    Ciklus vége
    Átlag=összeg/DB
Függvény vége.
Eljárás Átlagkiírás(konstans: DB: egész, Sorozat(1..DB): valós):
    Ki: 'A sorozat elemeinek átlaga:', Átlag(DB,A)
Eljárás vége.
```

A kódolás előtt tudnunk kell, hogy mindaz amit használni akarunk, annak a használat előtt deklarálva kell lennie, és ez vonatkozik a függvényekre és az eljárásokra is. Ezért az egy-

szerűség kedvéért a program szövegében előre helyezzük mindazt (eljárás, függvény), amit később használni fogunk.

Milyen eltérések szükségesek az algoritmushoz képest:

A sorozat (tömb) elemszámát már a deklarációnál meg kell mondani, ezért használtuk a MAXN állandót. Emiatt figyelünk arra is, hogy a darabszám ne legyen ennél nagyobb. Az eljárásokat és függvényeket használat előtt deklarálni kell.

```
👞 Turbo Pascal 7.0
                                                                                               - 🗆 ×
                Search
                                                              Options
  File Edit
                                 Compile
                                                                          Window
                                                                                    Help
                           Run
                                             Debug
                                                      Tools
                                                                                               1
Program Atlagpgm;
Uses Crt
Const MAXN=1000;
                                        (* legfeljebb ennyi eleme lehet a sorozatnak *)
        t='Nyomjon meg egy billentyűt
                                             (* később könnyebb megváltoztatni *)
Type ElemTip=Integer;
      SorozatTip=Array[1..MAXN] of ElemTip; (* ez lesz a sorozat típus *)
I:Integer; (* a konkrét feladatban ennyi elem van *)
Var N:Integer
     A:SorozatTip;
                                             (* Beolvasó eljárás *)
Procedure SorozatBeolvasas(Var DB:Integer; Var Sorozat:SorozatTip);
Var i:Integer; (* ciklusváltozó *)
Var i:Integer;
Begin
   DB:=0;
        .e'(DB<1) or (DB>MAXN) do (* tartományba esés ellenőrzése *)
egin (* DB beolvasása *)
Write('Adja meg a sorozat elemszámát (1..',MAXN,'): ');
   While (DB<1) or (DB>MAXN) do
     Begin
        ReadLn(DB);
   End;
For i =1 to DB do
                                             (* elemek beolvasása *)
     Begin
        Write(i,'. elem= ');
ReadLn(Sorozat[i]);
                                             (* tájékoztató üzenet *)
                                             (* beolvasás *)
     End;
End;
                                 (* Átlag számoló függvény*)
Function Atlag(Const DB:Integer; Const Sorozat:SorozatTip): Real;
Var osszeg:Real; (* segédváltozó *)
     i:Integer:
                                             (* ciklusváltozó *)
Begin
                                             (* összeg kezdőértéke *)
   osszeg:=0;
   For i:=1 to DB do osszeg := osszeg + Sorozat[i];
Atlag := osszeg/DB; (* ez lesz a visszaadott érték *)
End:
                                             (* Átlag kiíró eljárás *)
Procedure AtlagKiiras(Const DB:Integer; Const Sorozat:SorozatTip);
Begin
   WriteLn('A sorozat elemeinek átlaga: ',Atlag(N,A));
End;
Begin (* FŐPROGRAM *)
                                  (* képernyő törlés *)
   ClrScr
   SorozatBeolvasas(N,A); (* a sorozat elemszámának és elemeinek beolvasása *)
AtlagKiiras(N,A); (* átlag kiírása – meghívja az átlag függvényt *)
GotoXY((80-Length(t)) DIV 2,25); Write(t); (* záró üzenet *)
   Repeat until KeyPressed;
                                                               (* gombnyomásra vár *)
End.
                      41
F1 Help F2 Save F3 Open Alt+F9 Compile F9 Make Alt+F10 Local menu
```

Ellenőrizzük a program működését lehetséges és meg nem engedett darabszámokra is! Mindig figyeljünk arra, hogy a változóknak legyen kezdeti értéke!

Típusok használatával a program könnyen átírható lesz (egészek helyett például valós számokra).

### 5.5.6. feladat: Számkitaláló program

Készítsünk számkitaláló programot! A gép kiírja, hogy mely számtartományban gondolhatunk egy egész számot, és a gép megpróbálja kitalálni. A program tippjére 3 lehetséges választ adhatunk: eltalálta (e billentyű), a gondolt szám kisebb, mint a tipp (k billentyű), illetve, hogy a gondolt szám nagyobb, mint a tipp (n billentyű). A program algoritmusának elvét már általános iskolában tanultunk: mindig a lehetséges számintervallum közepét fogja a program tippelni. Legyen a számtartomány: 1...128!

Az intervallum felezőpontját a (min+max) div 2-vel határozzuk majd meg. Az div művelet az egész osztás (maradékos osztás) hányadosát jelenti.

Az algoritmus:

```
Program Kitalál:
Változó: min, max: egész [ a legkisebb és legnagyobb szám]
tipp: egész [ a program tippje ]
          tippszam: egész [ tippszámláló ]
     jatszunk:logikai [játszunk-e ismét]
     jo:logikai
                             [ jó-e a tipp ]
  jatszunk:=igaz [ először biztosan játszani fogunk ]
Amíg jatszunk [ jatszunk=igaz is lehetne feltétel ]
min:=1 : max:=128 [a két határ beállítása ]
tippszam:=1 [ kezdőértékek beállítása ]
     Ki: 'Gondolj egy egész számot ',min,' és ',max-1,' között!'
     Ki: 'Ha kigondoltad, akkor nyomj meg egy billentyűt!'
     BillentyűLenyomásraVár
                                    [ tipp: az intervallum közepe ]
     tipp:=(min+max) div 2
     jo:=hamis [ feltételezzük, hogy rossz ]
Amíg nem(jo) [ tippelés ismétlése jo-ig ]
          Ki: tippszam,'. tippem: ',tipp) [ tipp kiírása ]
          Ki: '. A gondolt szám kisebb(k), egyenlő(e),
                 vagy nagyobb(n)?'
          Elágazás MegnyomottGomb szerint [ gomb kódját adja ]
           'e','E' esetén:
                                               [ eqyenlő ]
                             jo:=igaz : Ki: 'Kitaláltam!'
           'k','K' esetén:
                                           [ kisebb a tippnél ]
                             max:=tipp-1; [ a felső határ módosul ]
                             tippszam:=tippszam+1; [ tipp számláló+]
                             tipp:=(min+max) div 2;[ új tipp ]
           'n','N' esetén: [ nagyobb a tippnél ]
                             min:=tipp+1; [ az alsó határ módosul ]
                             tippszam:=tippszam+1; [ tipp számláló+]
                             tipp:=(min+max) div 2;[ új tipp ]
           egyébként: Ki: '*** Helytelen billentyű k/e/n helyett!'
          Elágazás vége
      Ciklus vége
                                                [ tippelések ]
      Ki: 'Akarsz tovább játszani (i/n) ? '[ játék ismétlés ]
      Ha NagyBetű (MegnyomottGomb) <> 'I' akkor jatszunk:=hamis
  Ciklus vége
                                               [ ismétlődő játékok ]
  Ki: 'Köszönöm, hogy játszhattunk!'
  Várakozás(3)
                                               [ 3 mp-et várakozik ]
Program vége.
```

A fenti algoritmus kódolását próbáld meg önállóan elkészíteni. A többágú elágazást eddig még nem használtuk, lapozz vissza az átírási szabályokra! Ha elakadnál, itt megtalálod a teljes programot.



🗪 Turbo Pascal 7.0 - 🗆 × File Edit Search Run Compile Debug Window Help Tools **O**ptions 1 KITALAL Program Kitalal; Uses Crt; (\* legkisebb, legnagyobb lehetséges, akt. tipp \*) (\* a tippelések számát tároljuk benne \*) (\* ismételt játékhoz, a jó tippig ism. \*) Var min,max,tipp:Integer; tippszam:Integer; jatszunk,jo:Boolean; Begin atszunk:=True; (\* először biztosan szeretnénk játszani \*) (\* a játékot lehet ismételni\*) While jatszunk do Begin ClrScr; (\* képernyő törlés \*) min:=1; max:=128; tippszam:=1; (\* kezdőértékek beállítása\*)
Write('Gondolj egy egész számot '); (\* tájékoztató üzenetek \*)
WriteLn(min,' és ',max-1,' között!'); (\* a számtartományt is kiírja \*)
WriteLn('Ha kigondoltad, akkor nyomj meg egy billentyűt!'); ReadKey; tipp:=(min+max) div 2; \* gombnyomásra vár (\* tipp: az intervallum közepe \*) (\* feltételezzük, hogy rossz \*) (\* tippelés ismétlése jo-ig \*) jo:=False; While not(jo) do Begin Write(tippszam,'. tippem: ',tipp); (\* tipp kiírása\*) WriteLn('. A gondolt szám kisebb(k), egyenlő(e), vagy nagyobb(n)?'); (\* válasz karakter beolvasása \*) (\* egyenlő \*) Case ReadKey of 'e','E': Begin jo:=True; (\* jo, a tippelésnek vége \*) WriteLn('Kitaláltam!'); (\* tájákoztatás \*) End: 'k','K': Begin (\* kisebb a tippnél \*) (\* a felső határ módosítása \*) max:=tipp-1; tippszam:=tippszam+1; (\* tipp számláló növelés \*) tipp:=(min+max) div 2;(\* új tipp \*) End; 'n','N': Begin (\* nagyobb a tippnél\*) min:=tipp+1; (\* az alsó határ módosítása \*) tippszam:=tippszam+1; (\* tipp számláló növelése \*) tipp:=(min+max) div 2;(\* új tipp \*) End; else WriteLn('\*\*\*\*\* Helytelen billentyűt nyomott k/e/n heyett!') End; (\* Case \*) End; (\* Case \*) End; (\* While not(jo) \*) Write('Akarsz tovább játszani (i/n) ? '); (\* játék ismétlés \*) If UpCase(ReadKey) <> 'I' then jatszunk:=False;(\* i-re ismét j End; (\* While jatszunk \*) (\* rossz gomb -> tippismétlés \*) 'I' then jatszunk:=False;(\* i-re ismét játszunk \*) WriteLn WriteLn('Köszönöm, hogy játszhattunk!'); (\* köszönő szöveg \*) 3 mp-et várakozik \*) Delay(3000); End. 41 F1 Help F2 Save F3 Open Alt+F9 Compile F9 Make Alt+F10 Local menu

Ha kipróbáljuk a programot észrevehetjük, hogy amennyiben az 1-es tippre is nemmel válaszolunk, akkor a 0-t tippeli a program, ami nincs az intervallumban. Ez nem hiba, hiszen a válaszoló hibázott (nem mondott igazat).

Továbbfejlesztési lehetőségek:

- A válaszok eltárolásával ellenőrizhetnénk, hogy a válaszoló igazat mond-e, vagy legalább is következetes.
- Bekérhetnénk a határokat.

# 5.6. Összefoglalás

A fejezet elején a Borland Turbo Pascal 7.0 program telepítését tekintettük át.

Az 5.2. pontban a leggyakrabban használt gyorsbillentyűket ismertettem. Használtuk természetesen nem kötelező, de jelentősen gyorsítja a feladatok megoldását.

Az 5.3. pontban a fejlesztői rendszer menüpontjait tekintettük át, csak a legfontosabb elemekre koncentrálva. Ne feledd, hogy érettségin, vagy versenyen a munka könyvtár mindig legyen helyesen beállítva. Általában a futtatható programot is kérik, ezért a fordító diszk-re dolgozzon!

Ezt követően elkészítettük életünk első TP programját ('Helló világ!'), ami szinte minden programozásról szóló könyv első példája.

Az 5.5. pontban néhány egyszerű feladat algoritmusát, majd kódját írtuk meg.

## 5.7. Feladatok

A feladatok ismertetését most egy mondattal megfogalmazhatom: a korábbi algoritmusokból minél többet valósíts meg TP környezetben!

Ne feledd, hogy nem elég érteni a programozást, önállóan meg is kell tudni csinálni ráadásul az érettségin és a versenyen segédeszköz nélkül!