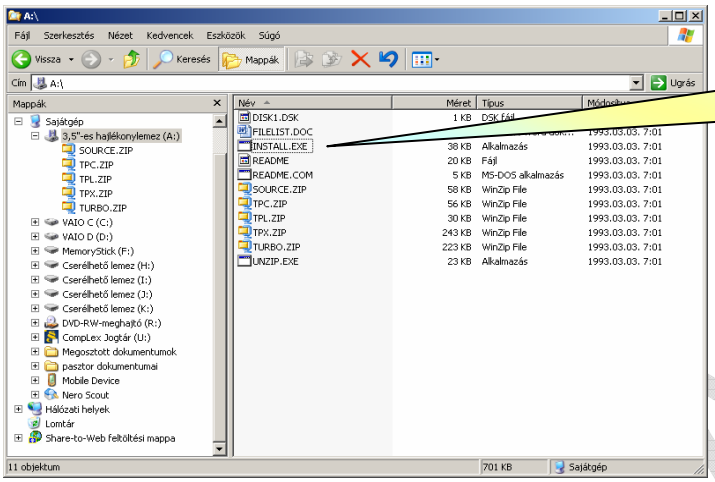
Pásztor Attila

Algoritmizálás és programozás tankönyv az emeltszintű érettségihez

5. A Turbo Pascal 7.0 fejlesztői környezet

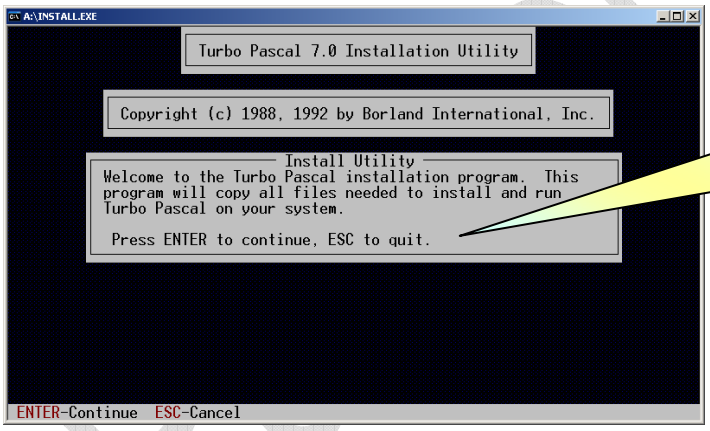
5.1. Telepítés

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.



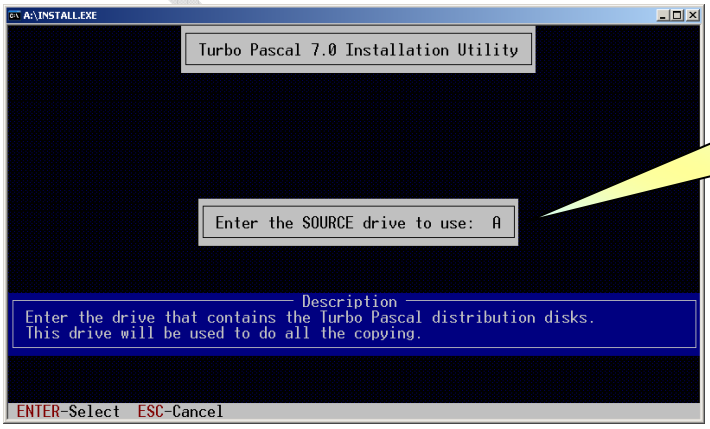
Az INSTALL.EXE programot kell elindítani a telepítés megkezdéséhez.

A telepítés elkezdődik, megjelenik a következő képernyő:

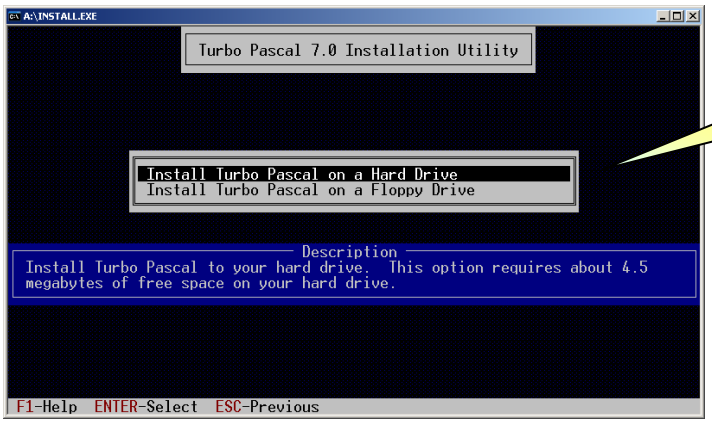


A telepítés az ENTER billentyű megnyomására indul el. ESC-et nyomva megszakítható.

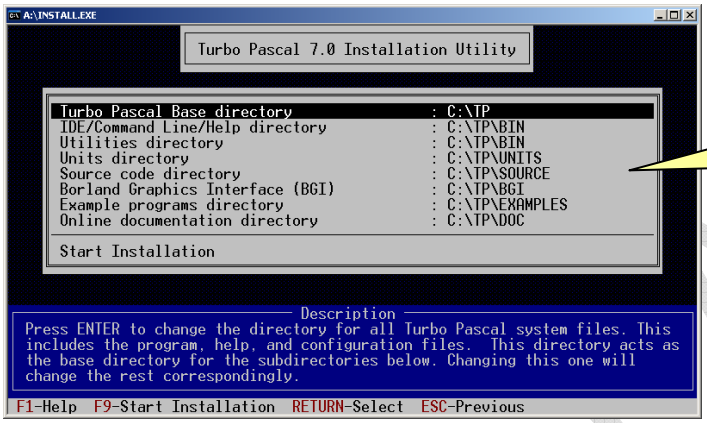
A telepítésre használt floppy meghajtó megadása:



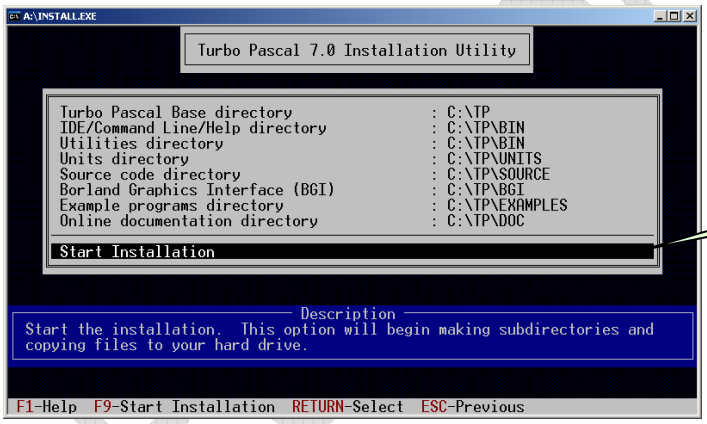
Ha nem az A meghajtót használod, akkor adj meg másik meghajtót!



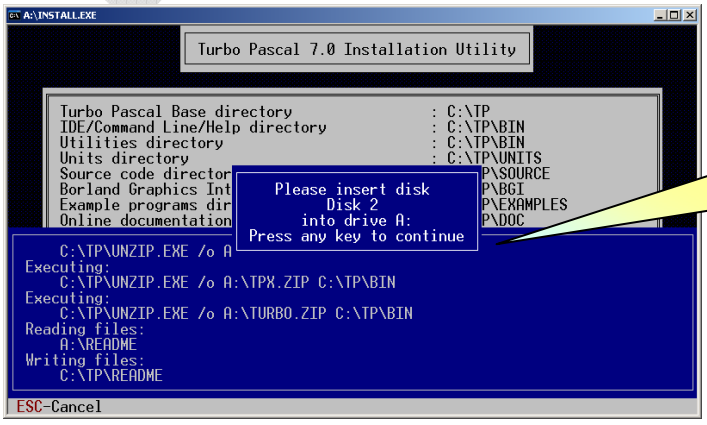
A program a merevlemezre fog települni.



Az egyes összetevők telepítési mappája is megváltoztatható.

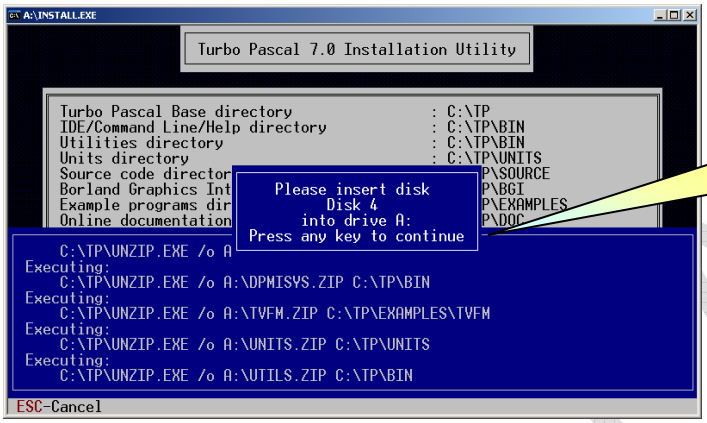


A tényleges telepítés a START kiválasztásával indul.

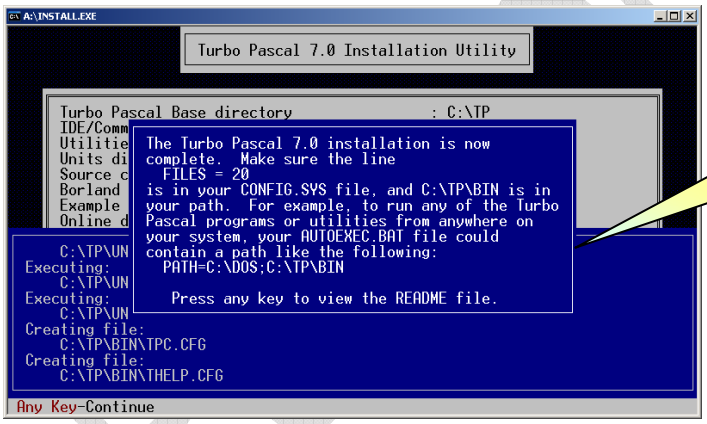


A program kéri a második lemezt. A cserét követően meg kell nyomni egy billentyűt!

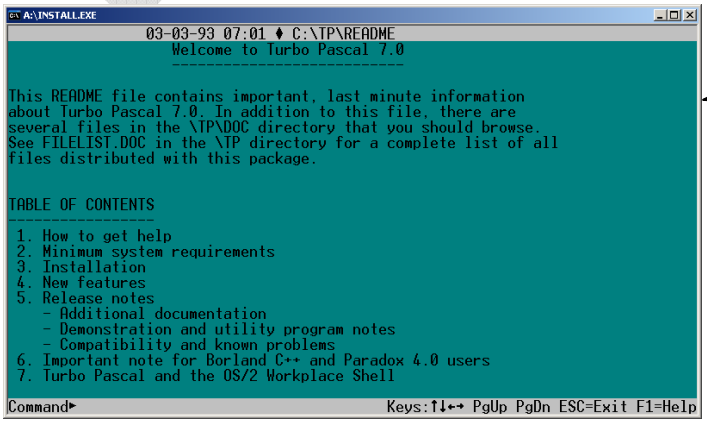
A program kéri a harmadik lemezt. A cserét követően meg kell nyomni egy billentyűt!



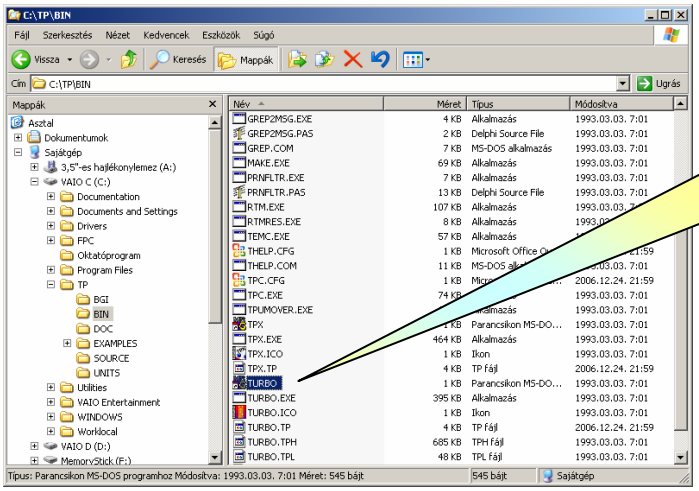
A program kéri a negyedik lemezt. A cserét követően meg kell nyomni egy billentyűt!



A telepítés befejeződött, a tájékoztató állomány megtekintéséhez nyomni kell egy billentyűt!



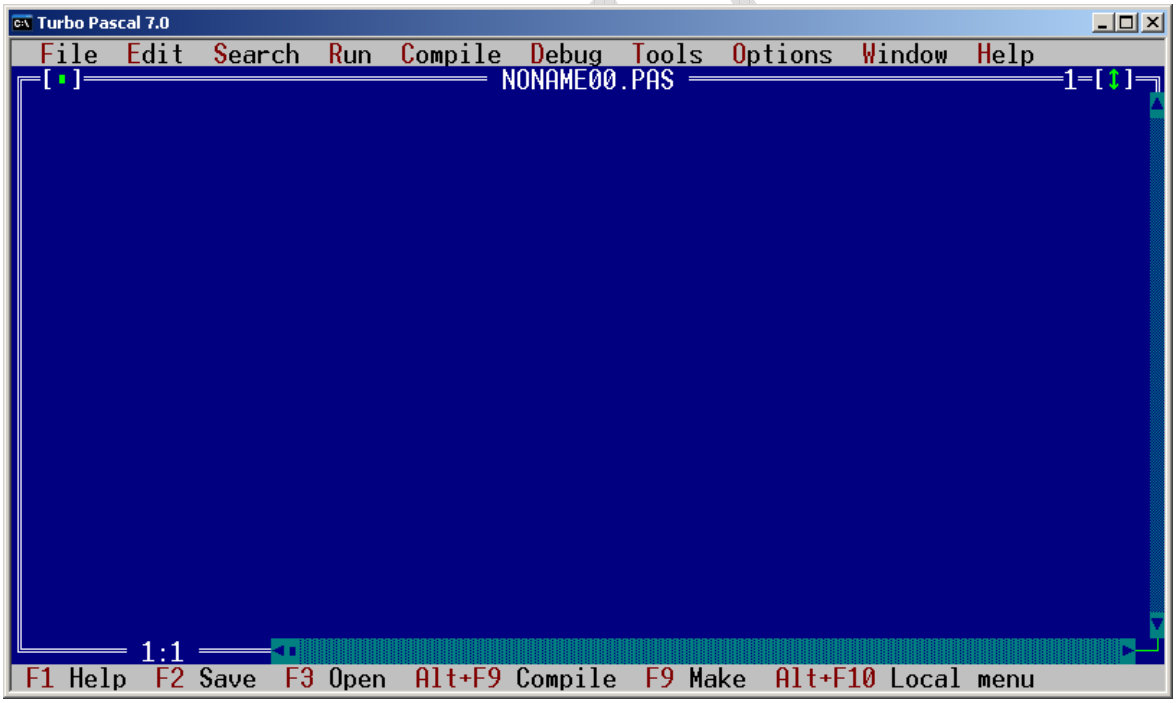
A tájékoztatóból az ESC billentyű megnyomásával lehet kilépni.



A program a C:\TP\BIN mappában lévő TURBO parancsikonra kattintva indítható (ha nem változtatta meg a telepítési útvonalat).

Természetesen készíthető parancsikon a Windows asztalon is a szokott módon. Ügyelni arra, hogy az indítási mappa a C:\TP\BIN legyen!

A program indítását követően az alábbi képernyőt láthatjuk:



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!

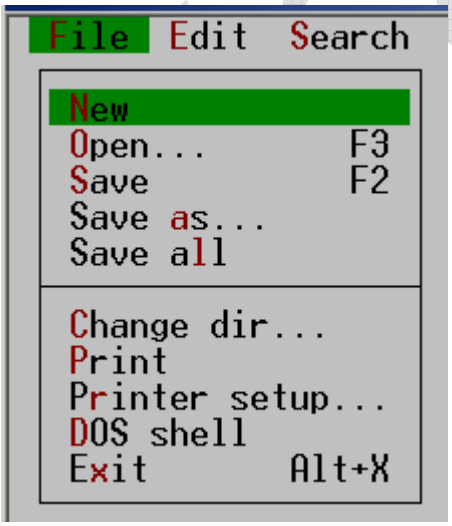
5.2. Gyorsbillentyűk:

A teljesség igénye 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.

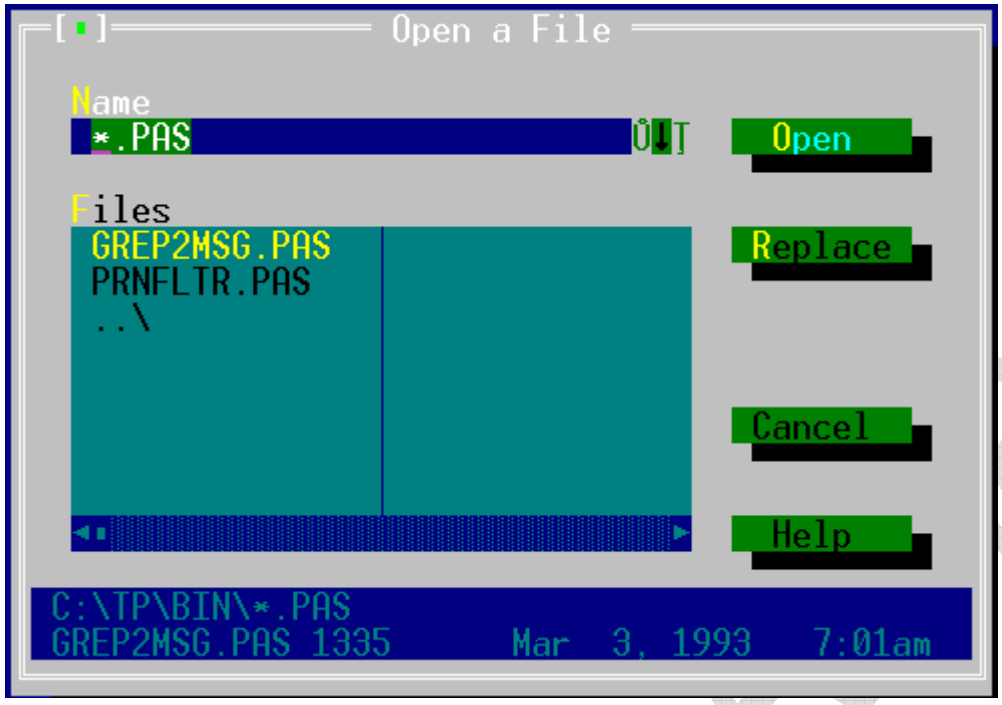
5.3. A TP 7.0 menürendszere

File (Fájl)

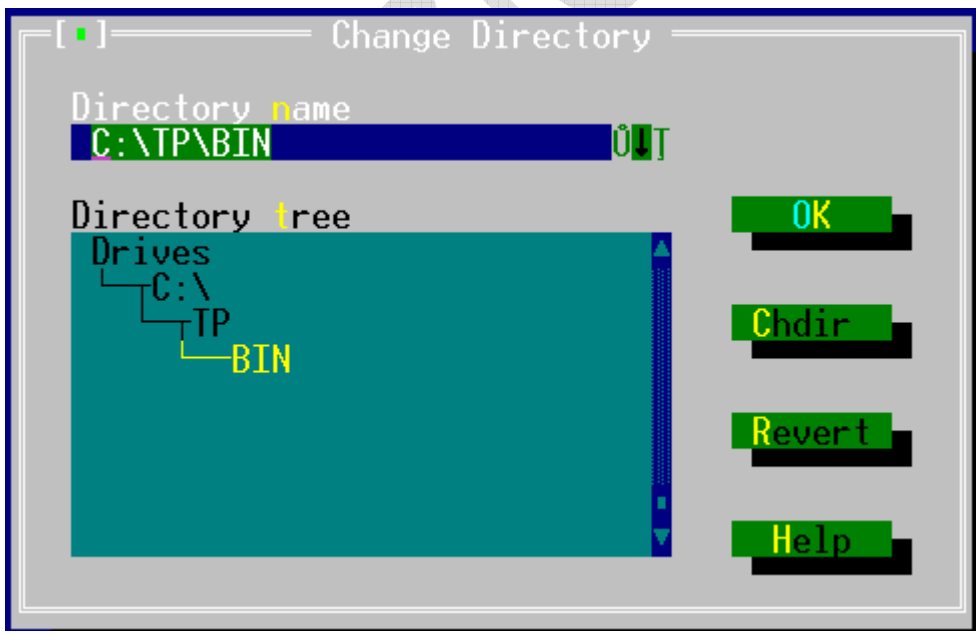


A ***New*** –val új, üres szerkesztő ablakot nyithatunk, melyből új fájlba lehet menteni a programot.
Az ***Open***-nel korábban elkészített programot lehet betölteni, a mappák között is lehet navigálni (a mappa neve után látható egy fordított perjel: \

A fájl megnyitás ablaka:

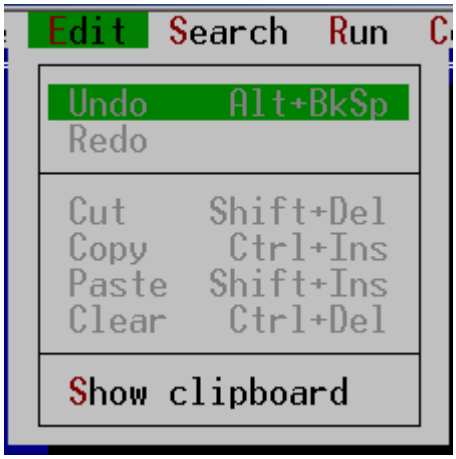


Programot menteni a **Save**, illetve a **Save as** menüponttal lehet a szokásos módon. Ügyeljünk arra, hogy az állományok neve megfeleljen a DOS állománynévre vonatkozó szabályainak (legfeljebb nyolc karakter, magyar ékezetes betű ne legyen benne, …).
Az alapértelmezett könyvtár beállítása a ***Change Dir*** segítségével adható meg. Minden alkalommal győződjünk meg arról, hogy a megfelelő helyre mutat:



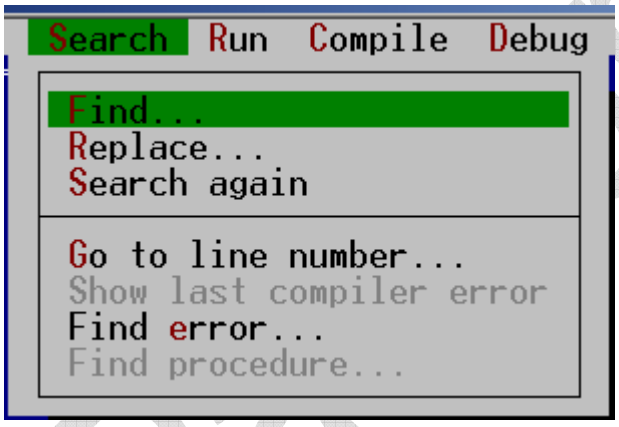
A program a ***Print*** menüponttal kinyomtatható. A ***DOS shell***-lel egy parancssor nyitható, melyből exit-tel lehet visszalépni, bár ennek a Windowsos felület alatt nincs sok jelentősége. A TP programból az ***Exit***-tel lehet kilépni.

Edit (Szerkesztés)



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:

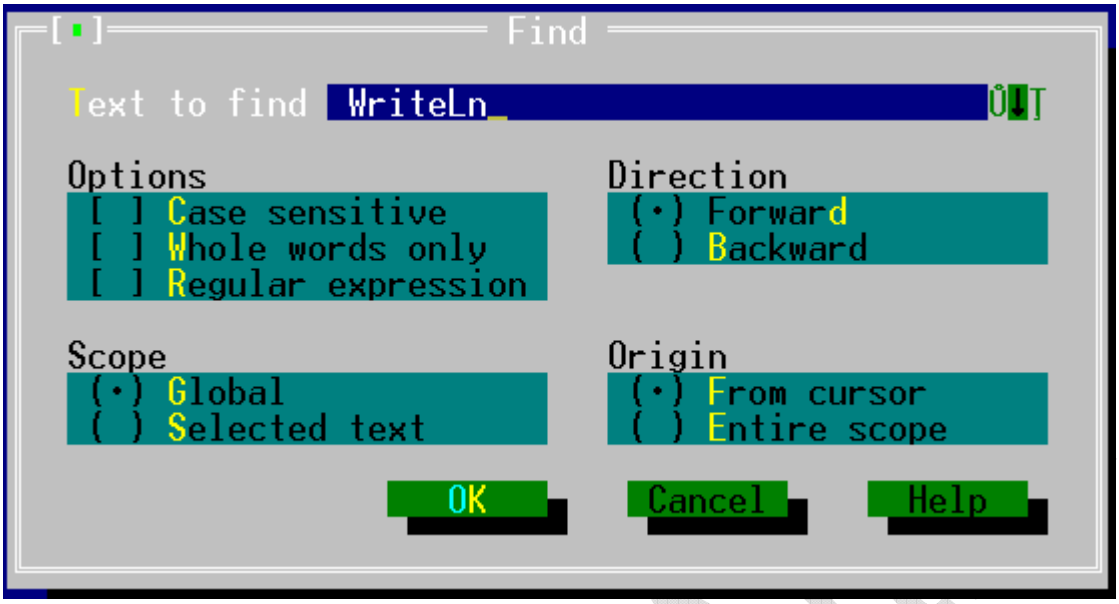
Case sensitive – megkülönbözteti a kis- és nagybetűket
Whole words only – csak teljes szóra keres
Regular espression – reguláris kifejezést keres

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

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

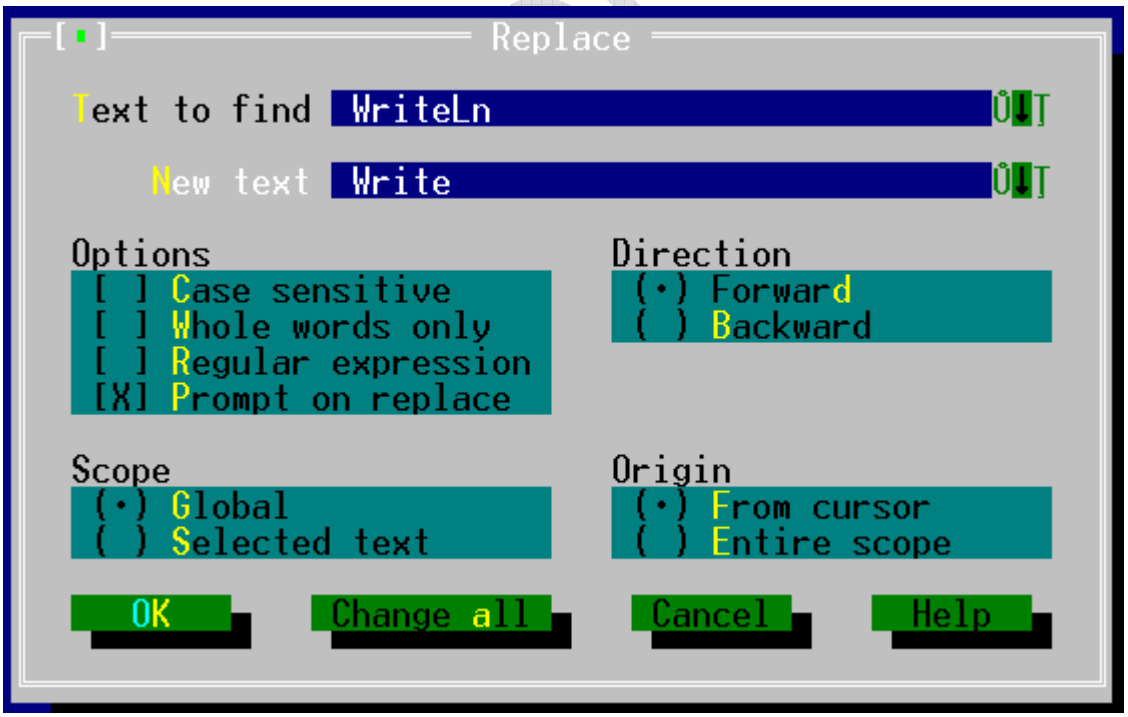
A keresett szöveg a „Text to find” mezőbe írandó!

A keresés ablaka a következő:

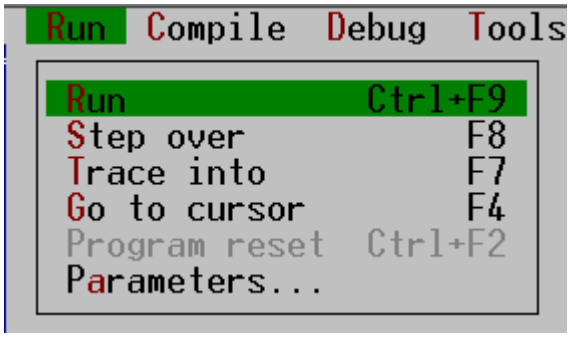


A ***Replace*** menüponttal szöveget lehet cserélni. Egyetlen mezőben különbözik az előzőtől: a „New text” mezőben adható meg, hogy a keresett szöveget mire kell lecserélni.

Íme a csere ablaka:

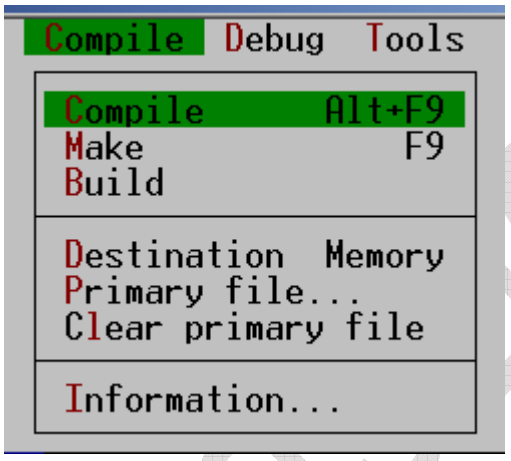


Run (Futtatás)



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)

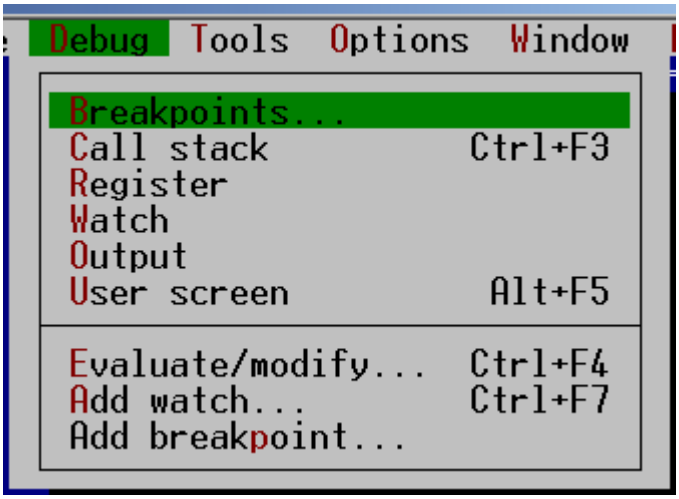


Compile	A forráskódot fordítja, de nem futtatja (ellenőrzéshez hasznos).
Destination Memory	A fordítás helye a memória (rákattintva változik).
Destination Disk	A fordítás helye a diszk (rákattintva változik).

Figyelem!

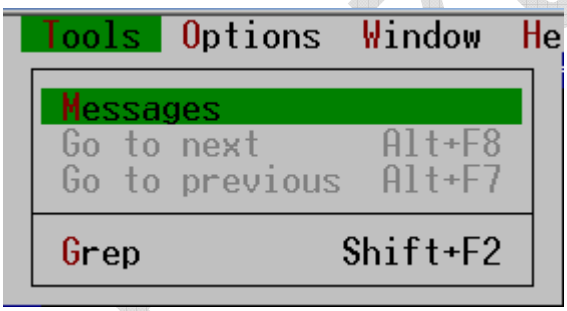
Érettségin, versenyen a forrás és a lefordított változat beadását is kérik, ezért a **Destination Disk** lehetőséget kell kiválasztani (rákattintva a sorra)! Ha nem változtatunk az alapbeállításokon, akkor a futtatható állomány a forrásállomány mentési helyének megfelelő mappában keletkezik programnév.exe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

Debug (Tesztelés)



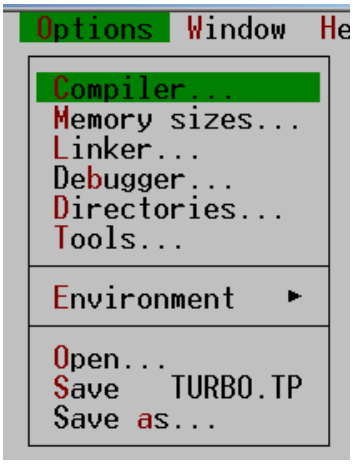
Breakpoints	Töréspontok megtekintése és törlése . Hatására a forráskód egy sora kijelölődik. Futtatáskor újabb parancsig itt megáll a futás..
Add breakpoint…	Töréspontot lehet elhelyezni (törölni).
Watch	A futás megszakadásakor változók/kifejezések értéke tekinthető meg.
Add watch…	Változó/kifejezés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

Tools (Eszközök)



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

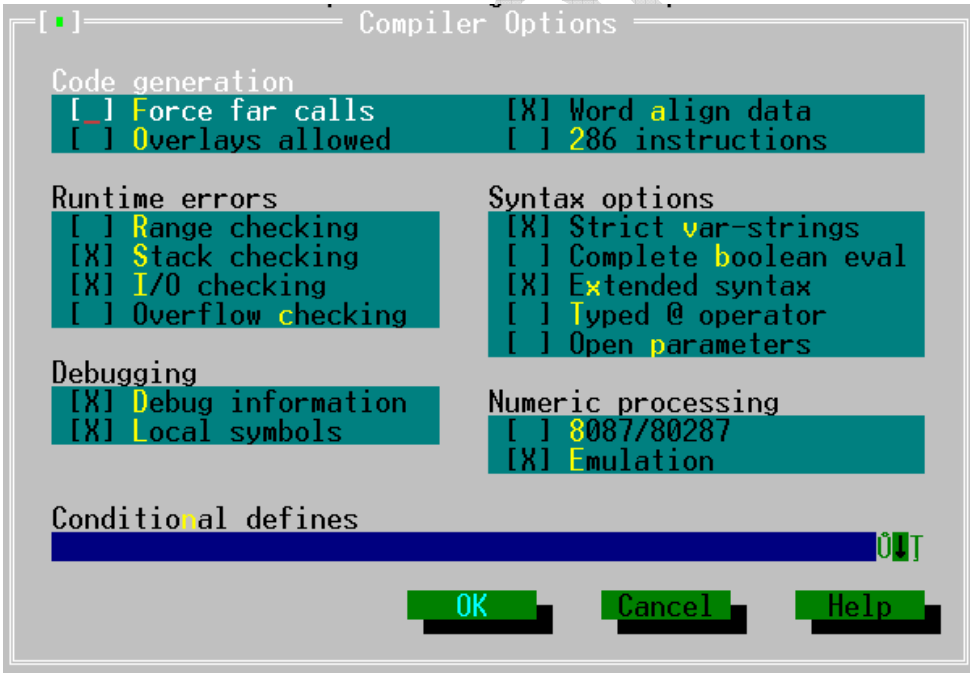
Options (Lehetőségek)



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.

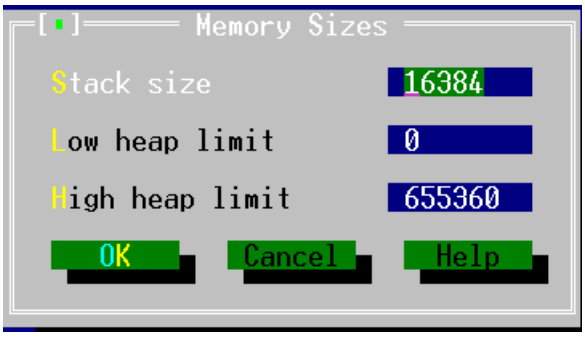
Compiler (A fordító beállításai)



Az alapbeállítások megtartása mellett válasszuk a következőket:

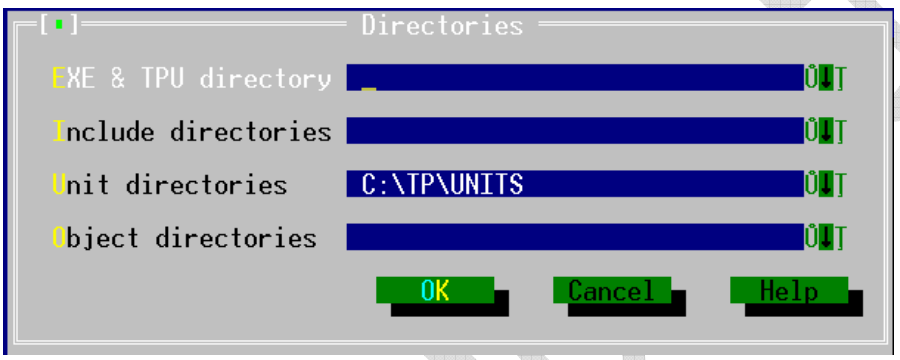
Range checking Tömbök túlcímzésének ellenőrzésére jó például.
Overflow checking Túlcsordulást vizsgál.

Memory sizes… (Memória méretek)



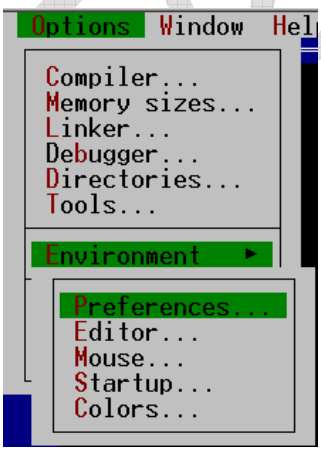
Alap esetben nem igényel módosítást.

Directories (Könyvtárak)



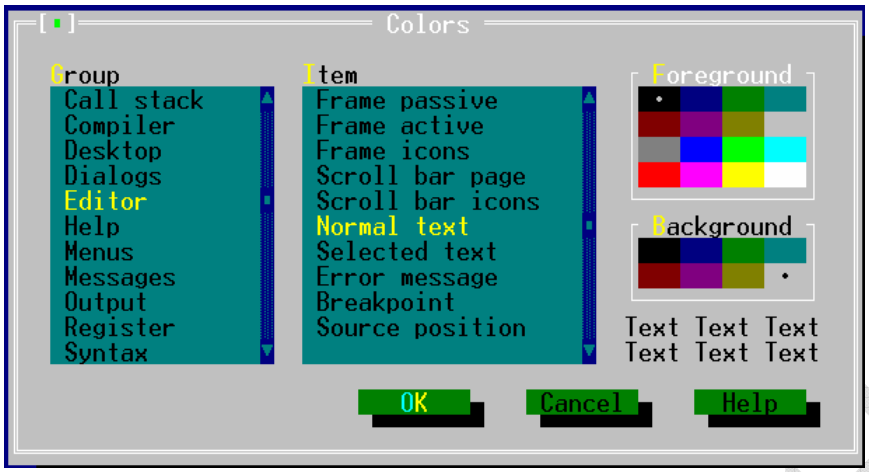
Az egyes fordítási egységek helye adható meg. Ellenőrizzük, hogy a „Unit Directories” a megfelelő helyre mutat. A többi megadása nem kötelező, s ekkor a forrással megegyező mappát fog használni.

Environment (Környezet)

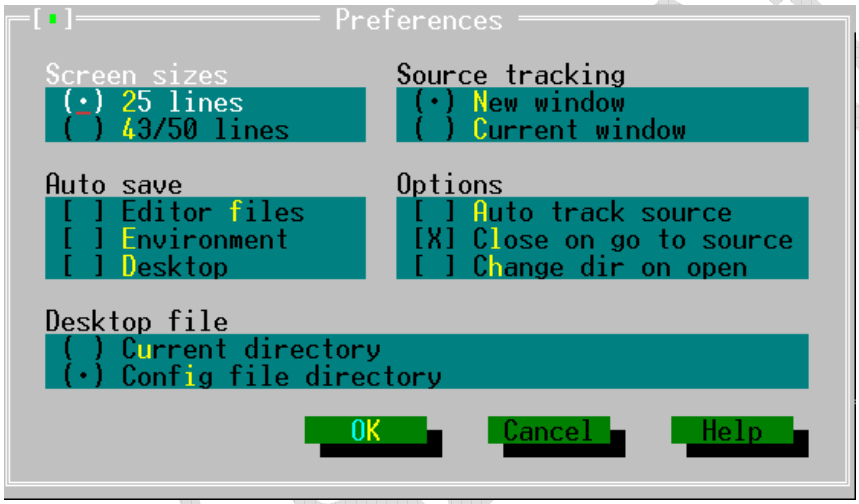


A környezeti beállítások itt végezhetők el. Nem szükséges semmit módosítani, a festékta-karékos nyomtatás miatt én átalakítottam a színbeállításokat.

Environment/Colors/Editor menüpont (Editor színkészletének testreszabása)

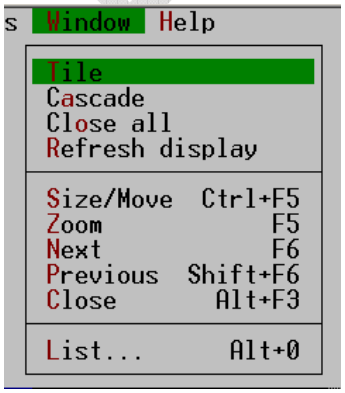


Environment/Preferences menüpont



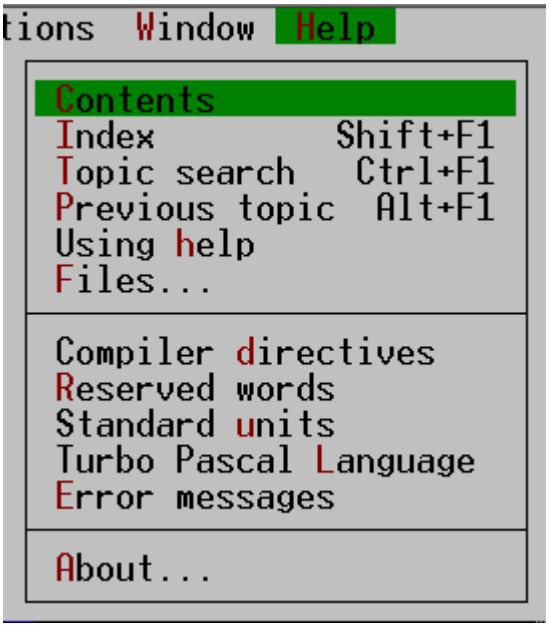
A 43/50 soros képméret választással jobban áttekinthető a forráskód.

Window (Ablak)



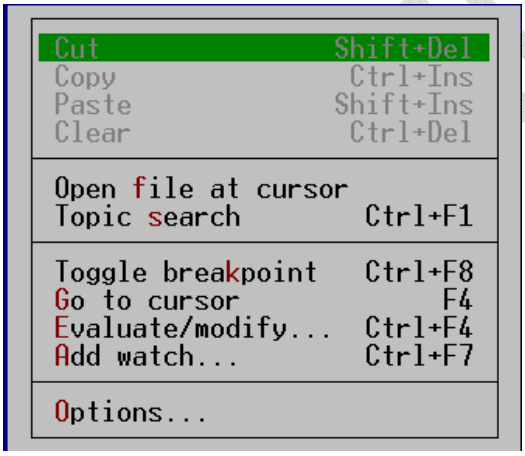
Az ablakok kezelése a szokásos lehetőségekkel: mozaikos, lépcsőzetes elrendezés, méretezés, mozgatás, nagyítás, aktiválás, bezárás.

Help (Súgó)



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

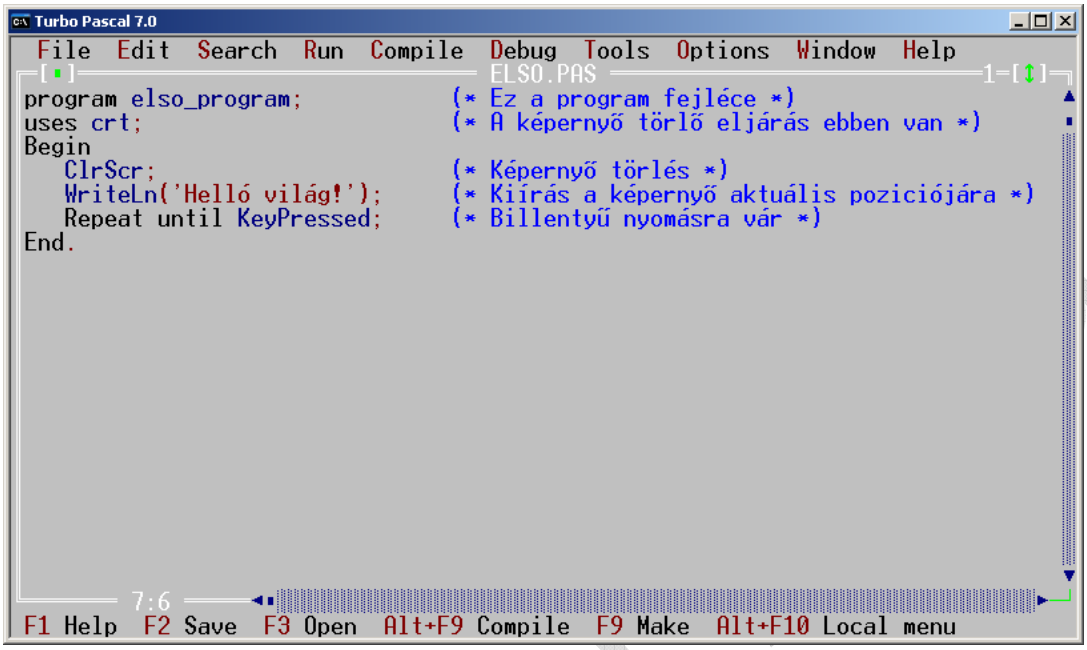


A szerkesztő ablakban az egér jobb gombjával hívható elő, és a legfontosabb menüpontokat tartalmazza.

Kezdjünk hozzá az első programhoz!

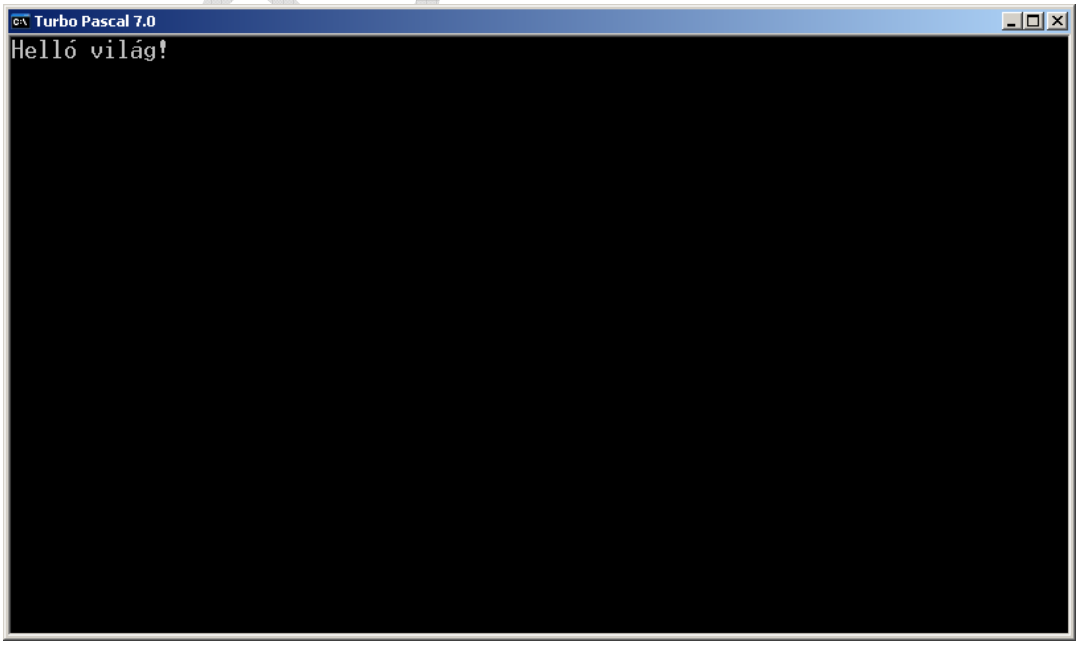
5.4. Az első pascal program

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)!



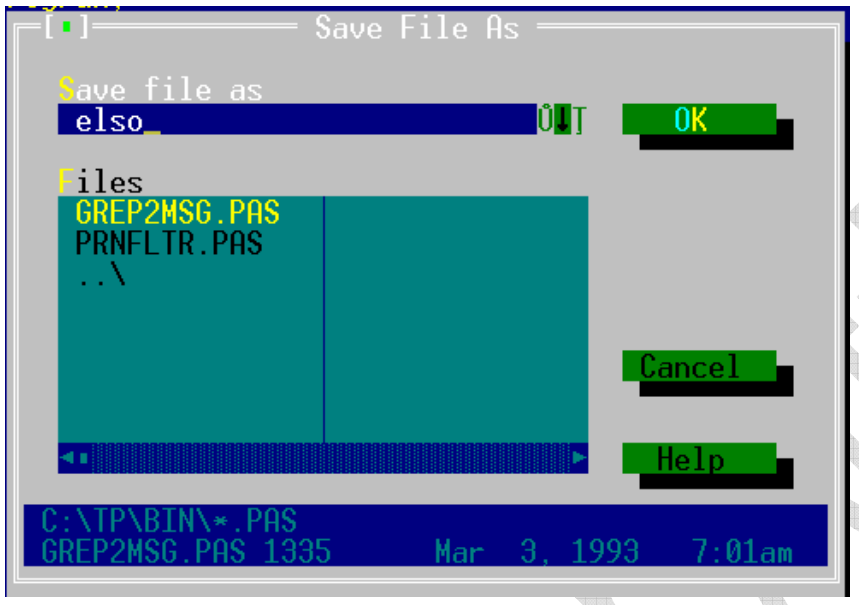
Néhány tanács:
A sorok végén **ENTER**-t ütünk (ez nem szövegszerkesztési feladat)!
A programsorok behúzását szóközökkel, vagy tabulátorokkal végezhetjük.
Az utasításokat pontosvessző választja el egymástól.
A kisbetű/nagybetű használata csak az olvashatóságot segíti.

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!



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.

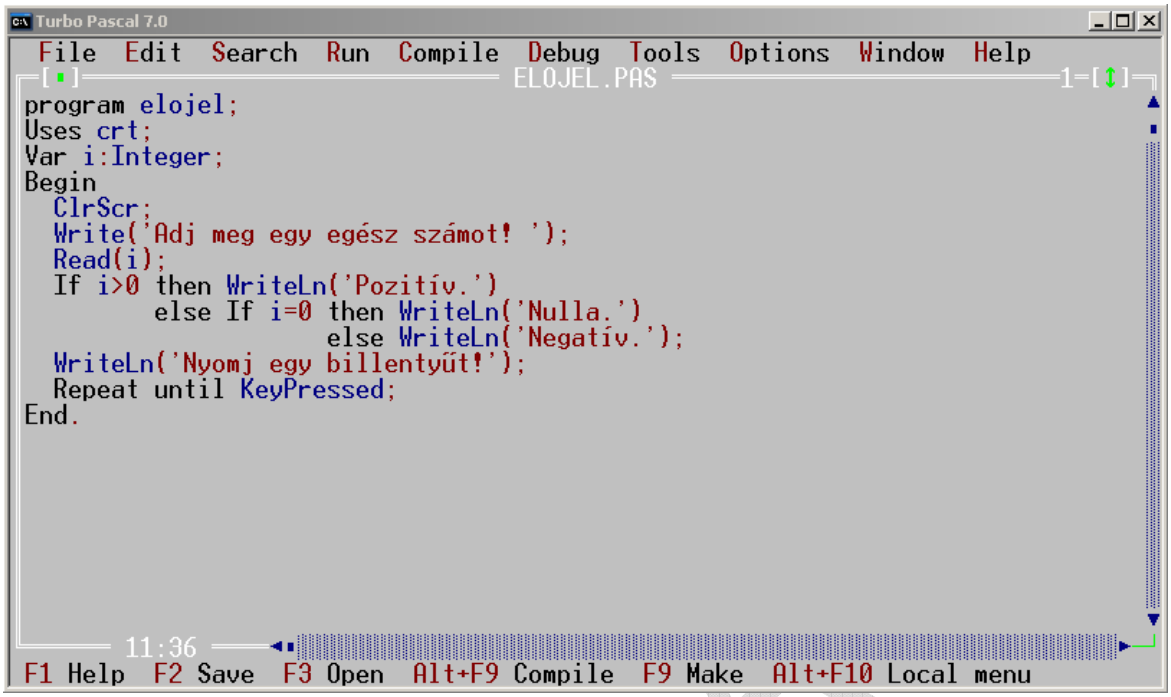
5.5.1. feladat: Előjel szöveges kiírása

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

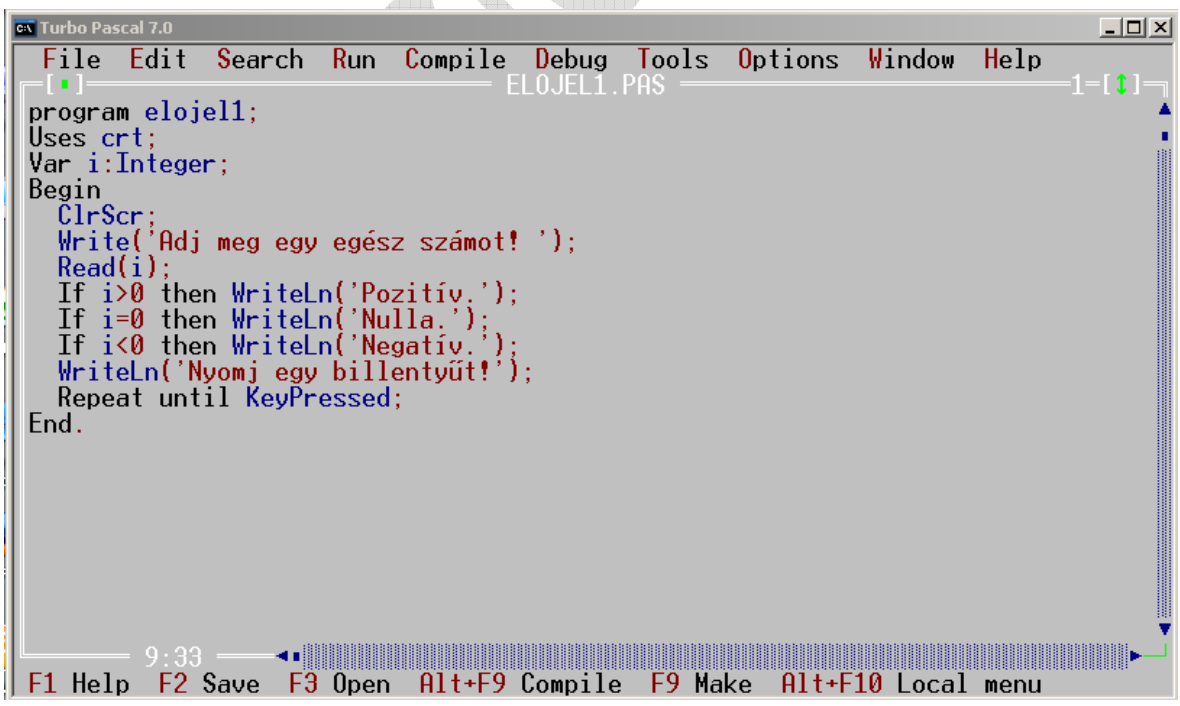
A példából látszik, hogy az algoritmust sorról-sorra átkódoltuk, csupán a beszédesebb beolvasásnál alkalmaztunk a programban eltérő viselkedést.
Gépeljük be a kódot a Turbo Pascal szerkesztő ablakába, és mentsük el elojel.pas néven! Egészítsük ki a kódot a képernyőtörléssel (**ClrScr**), és a végén egy 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.

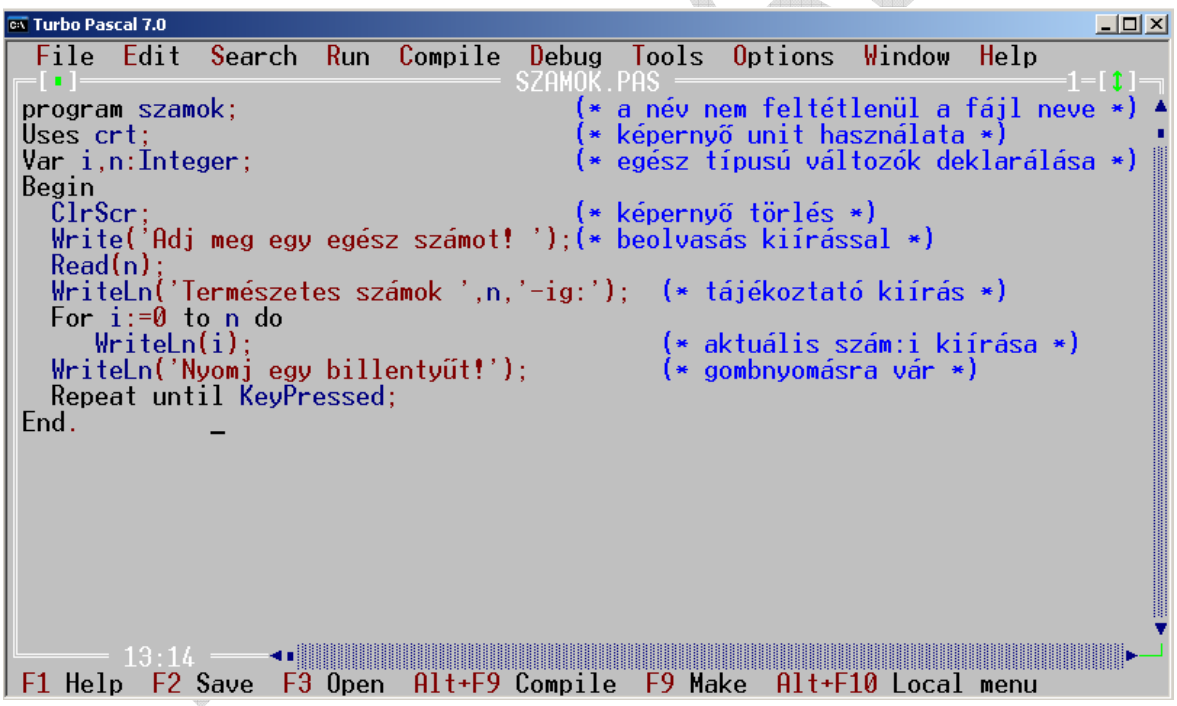
5.5.2. feladat: Számok kiírása 0-tól a megadott határig

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

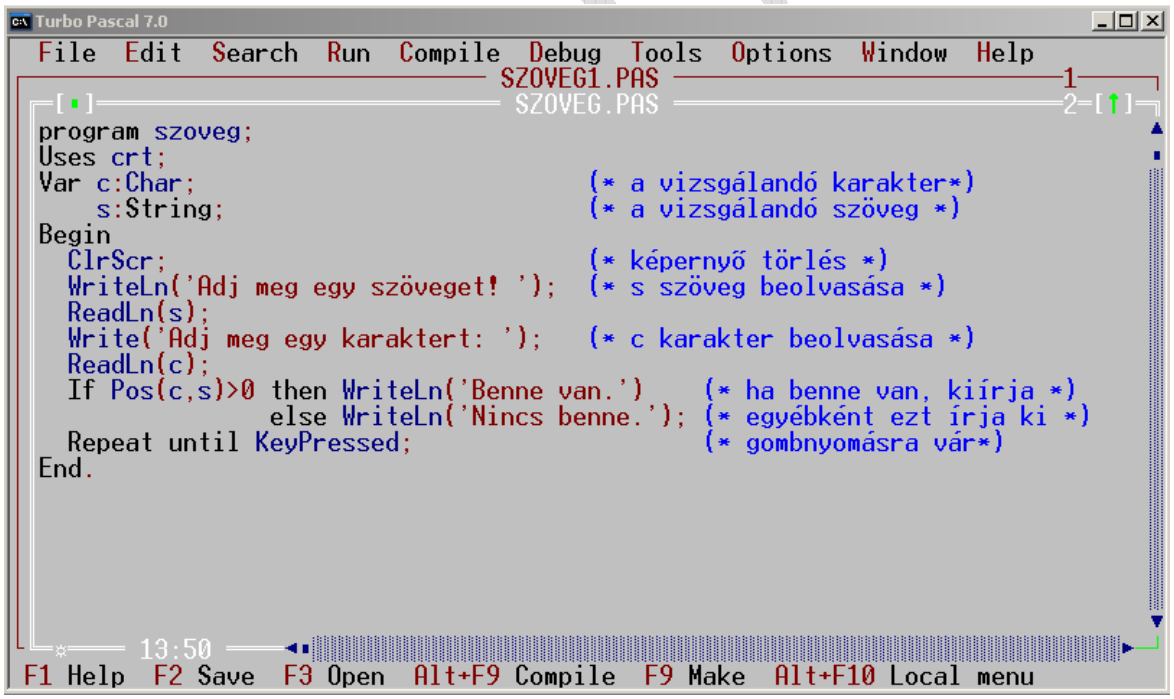
5.5.3. feladat: Benne van-e a szövegben a megadott karakter

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

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.

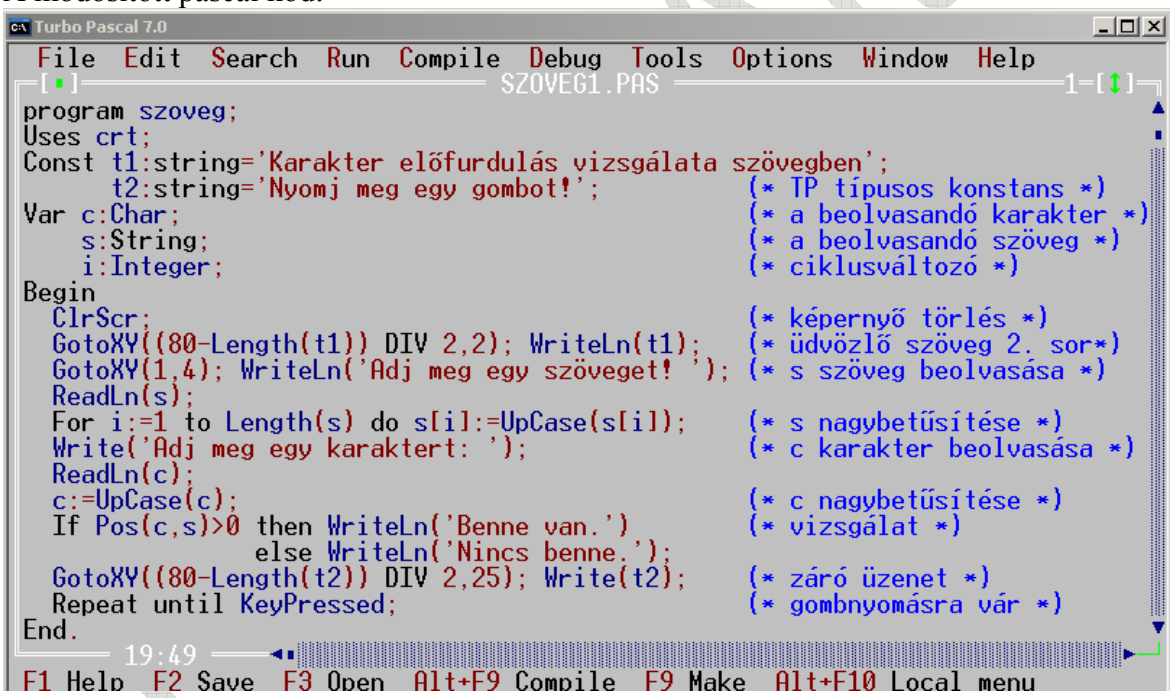
Módosítás:
A program kezelje azonosan a kis-, és nagybetűket!

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 hosszát 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 hányadosa
A MOD B A osztva B-vel egész osztás maradéka

A módosított pascal kód:



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?

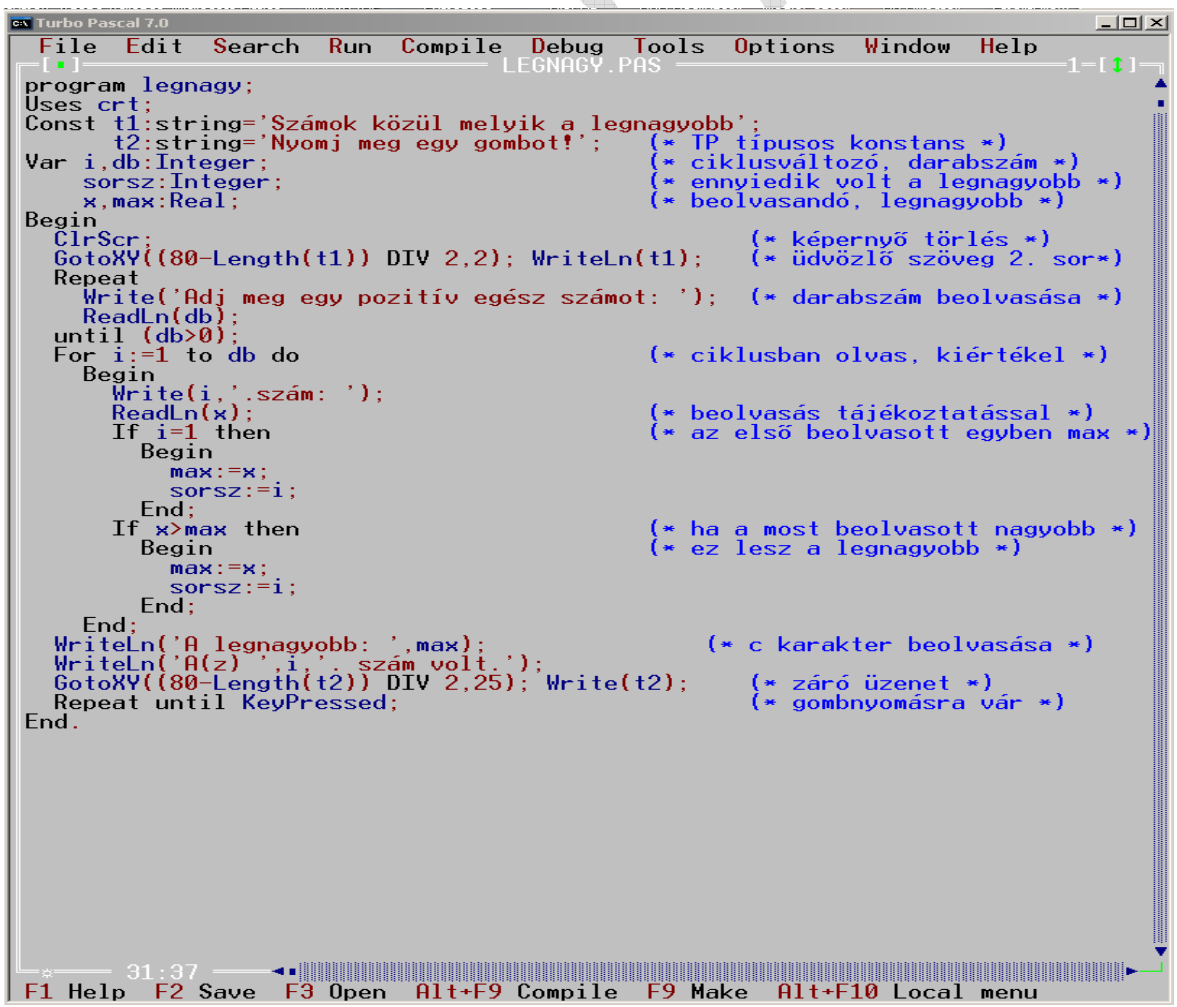
5.5.4. feladat: Beolvasott számok legnagyobbika

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.

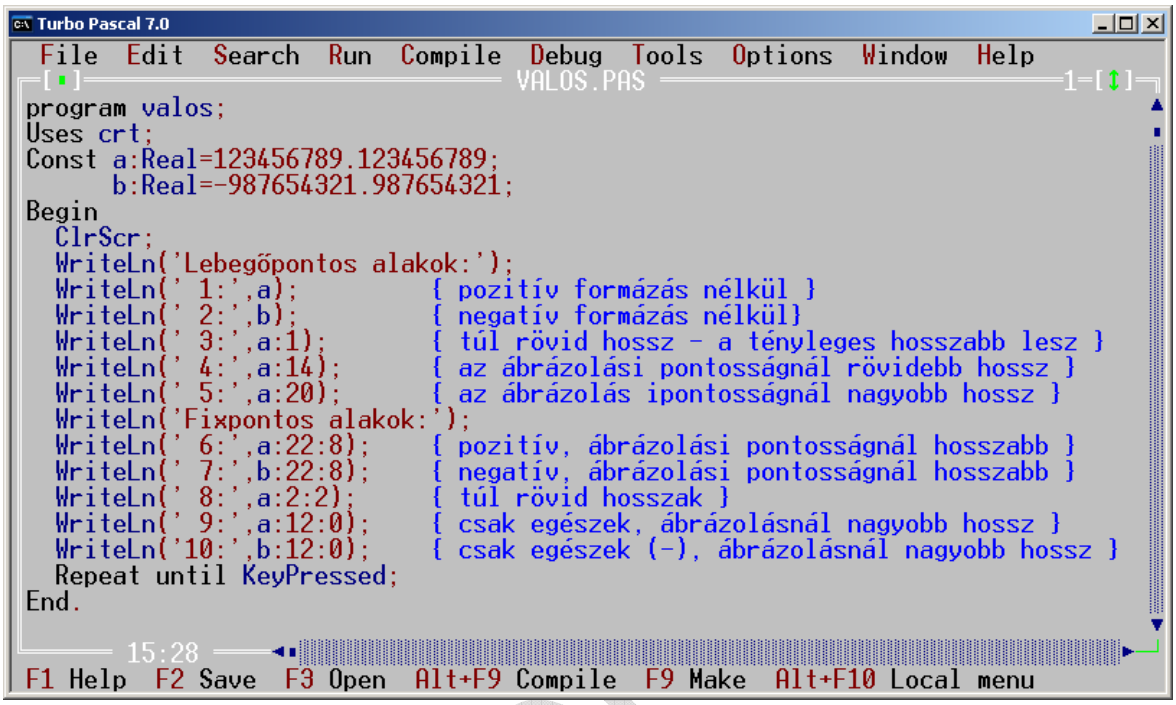
ALGORITMUS	TURBO PASCAL KÓD

Program legnagy:
 	
Változó i,db,sorsz:egész
x,max:valós
 	
Be: db [db>0]
 	
Ciklus i:=1-től db-ig
 	
Be:x
 	
Ha i=1 akkor max:=x : sorsz:=i
 	
Ha x>max akkor max:=x : sorsz:=i
 	
Ciklus vége
Ki: max
Ki: sorsz
Program vége.
	Program legnagy;
	uses crt;
	Var i,db,sorsz:Integer;
	x,max:Real;
	Begin
	Repeat
	Write('Adj meg egy pozitív számot: ');
	ReadLn(db);
	until (db>0);
	For i:=1 to db do
	Begin
	Write(i,'. szám: ');
	ReadLn(x);
	If i=1 then
	Begin
	max:=x;
	sorsz:=i;
	End;
	If x>max then
	Begin
	max:=x;
	sorsz:=i;
	End;
	End;
	WriteLn('A legnagyobb: ',max);
	WriteLn('A(z) ',i,'. szám volt.');
	End.

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



Futtatás során tapasztaljuk, hogy a valós számokat a pascal normálalakban írja ki, például az 50-et így: **5.000000000E+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:




```
program valos;
Uses crt;
Const a:Real=123456789.123456789;
      b:Real=-987654321.987654321;
Begin
  ClrScr;
  WriteLn('Lebegőpontos alakok:');
  WriteLn(' 1:',a);          { pozitív formázás nélkül }
  WriteLn(' 2:',b);          { negatív formázás nélkül}
  WriteLn(' 3:',a:1);        { túl rövid hossz - a tényleges hosszabb lesz }
  WriteLn(' 4:',a:14);       { az ábrázolási pontosságnál rövidebb hossz }
  WriteLn(' 5:',a:20);       { az ábrázolás ipontosságnál nagyobb hossz }
  WriteLn('Fixpontos alakok:');
  WriteLn(' 6:',a:22:8);     { pozitív, ábrázolási pontosságnál hosszabb }
  WriteLn(' 7:',b:22:8);     { negatív, ábrázolási pontosságnál hosszabb }
  WriteLn(' 8:',a:2:2);      { túl rövid hosszak }
  WriteLn(' 9:',a:12:0);     { csak egészek, ábrázolásnál nagyobb hossz }
  WriteLn('10:',b:12:0);     { csak egészek (-), ábrázolásnál nagyobb hossz }
  Repeat until KeyPressed;
End.
```

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

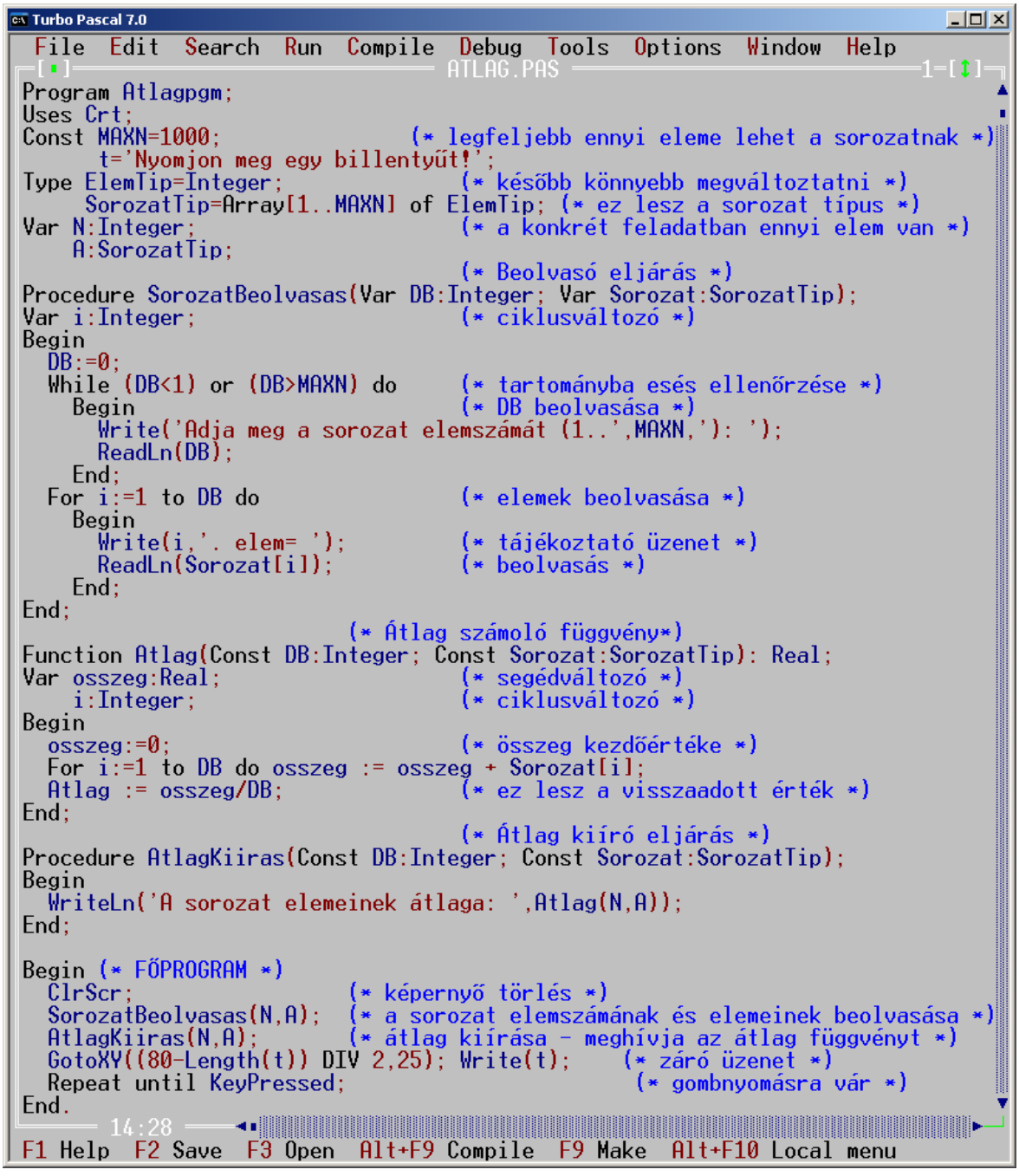
Az algoritmizálás (tervezés) fejezetben készítettünk egy átlagszámító programot, amelyben eljárást és függvényt is használtunk. Utolsó egyszerű példaként kódoljuk most Turbo Pascal nyelven!

Íme az algoritmus:

```
Program:
     Változó: N: egész
              A(1..N: valós)    [így adom meg a sorozatot]
     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.



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

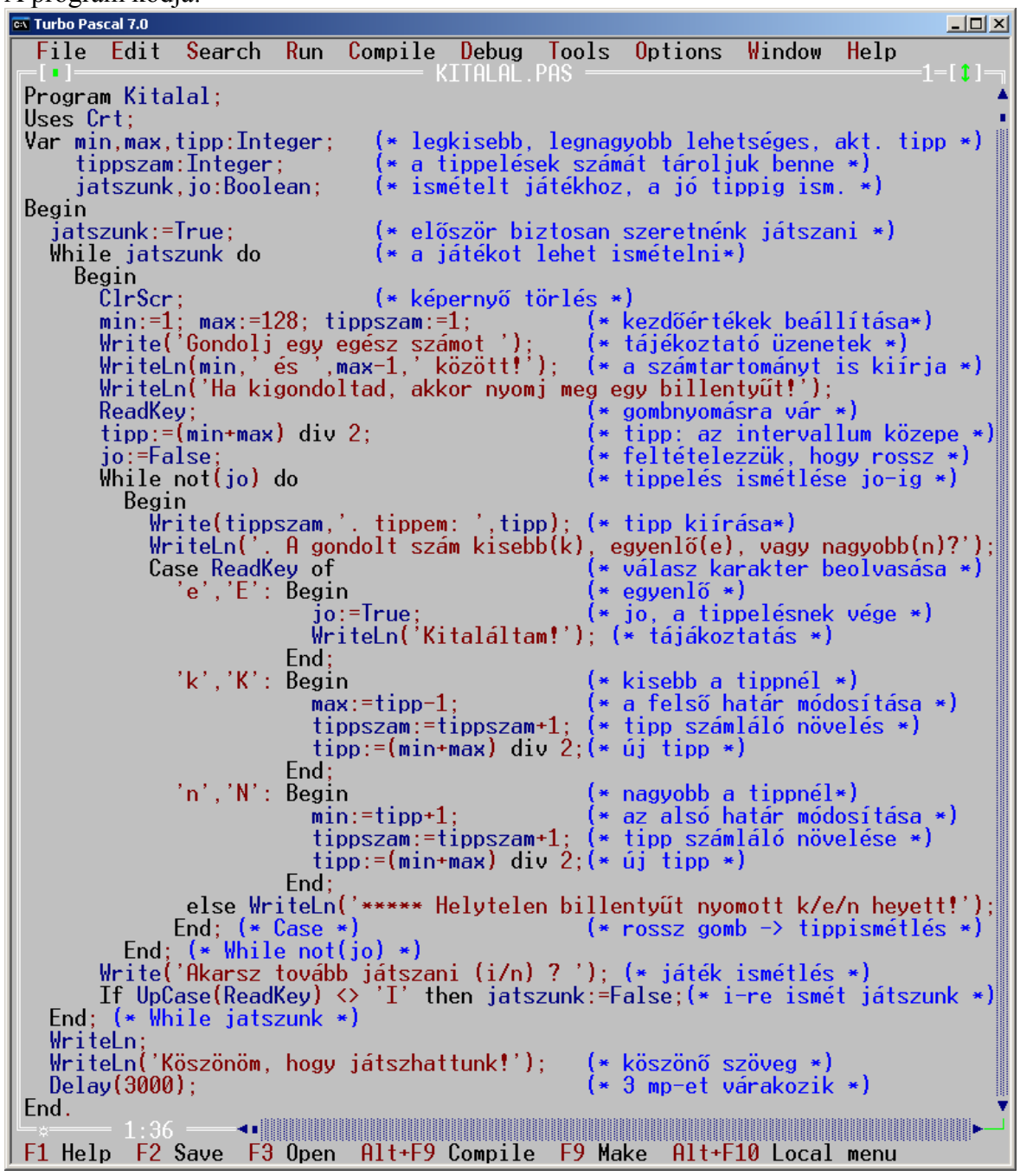
Készítsünk számkitaláló programot! A gép kiírja, hogy mely számtartományban gondolhatunk egy egész számot, és a gép megpróbálja kitalálni. A program tippjére 3 lehetséges választ adhatunk: eltalálta (e billentyű), a gondolt szám kisebb, mint a tipp (k billentyű), illetve, hogy a gondolt szám nagyobb, 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:=(min+max) div 2     [ tipp: az intervallum közepe ]
     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:                 [ egyenlő ]
                         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.

A program kódja:



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!