DIGITÁLIS KULTÚRA

14.
TURBO PASCAL PROGRAMOZÁS ALAPJAI

oktatási segédanyag és feladatgyűjtemény

Összeállította: Kolman Krisztián

Tartalomjegyzék:

1 A pascal program szerkezete

- a pascal program alapszerkezete
- kiíratás képernyőre - **write, writeln**

1.1 A pascal program alapszerkezete

A Pascal programozási nyelvben minden programnak a következő szerkezete van:

```
program program_neve;
var változók deklarálása;
begin

   parancsok, amit a program végrehajtson;

end.
```

Az első sorban a **program** kulcsszó után megadjuk a programunk nevét. Ide bármilyen nevet választhatunk, de a program neve nem tartalmazhat szóközt és néhány speciális karaktert. A sor (parancs) végét pontosvesszővel (**;**) fejezzük be. Ezt az első sort nem kötelező megadnunk.

A második sorban a **var** kulcsszó *(variables = változók)* után felsoroljuk, hogy a programunkban milyen változókat fogunk használni. Az egyszerű statikus változót úgy képzelhetjük el, mint valamilyen tárolóhelyet a számítógép memóriájában, amelyben egyszerre csak egy számot vagy szöveget tárolhatunk. Ha például van benne már valamilyen szám és rakunk bele egy másikat, akkor az előző szám elveszik, mindig csak az utoljára belerakott érték marad benne.

A **var** parancsnál meg kell adnunk azt is, hogy az egyes változókban (tárolóhelyekben) milyet típusú adatot fogunk tárolni - egész számot (integer), szöveget (string), vagy más típust. A programunkban több változót is használhatunk, pl. kettő változót számok tárolására és egy változót szöveg tárolására. Itt is a sor végét pontosvesszővel fejezzük be. Amennyiben a programban nem használunk változót (bár ez nagyon ritkán fordul elő), ezt a sort is kihagyhatjuk.

A következő sortól kezdődik maga a program – azok az egymás után következő utasítások, melyeket a számítógép végrehajt a program futtatásakor. Ezeket az utasításokat a **begin** és az **end** kulcsszavak *(begin = kezdete, end = vége)* közé kell írnunk. Fontos, hogy minden parancs után a sort pontosvesszővel (**;**) fejezzük be. A programunk végét ponttal (**.**) zárjuk.

Célszerű egy úgynevezett **crt** unit használata. Például képernyőtörlésre. **(clrscr)** Ennek használata ajánlott az összes programban! (Aztán például a **delay**(2000) – a program 2000 milliszekundumig (2 másodpercig) várakozik.)

A programozás során egy fekete, 80 karakter széles és 35 karakter magas felületen dolgozunk!

Lássunk most egy egyszerű példát:

```
program ElsoProgram;
uses crt;
begin
clrscr;
 write('Hello!');
readln;
end.
```

Ez a program, csupán annyit csinál, hogy kiírja a képernyőre a Hello! mondatot. Ha programunkat begépeltük az editorba (FreePascal-ba), mentsük el, majd a **CTRL** + **F9** -el (vagy a főmenüből **run - run** paranccsal) fordítsuk le és futtassuk. Ekkor a program kiírja a *Hello!* mondatot a képernyőre, majd rögtön visszatér az editorba (ezért nem látjuk a kiírt mondatot). A FreePascal egy másik szöveges képernyővel dolgozik a program futtatása alatt, mint amiben mi írjuk magát a programot (ezért most nem látjuk a kiírt *Hello!* üzenetet). Erre a képernyőre az **ALT** + **F5** billentyűzetkombinációval válthatunk át. Vissza az editorba az **ALT** + **F5** újabb megnyomásával juthatunk.

*Megjegyzés: Ha a FreePascal DOS alatt futó verzióját használjuk, akkor a billentyűzetet a **CTRL** + **ALT** + **F1** billentyűzetkombinációval állíthatjuk át angol nyelvűre.*

1.2 Kiírás a képernyőre - write, writeln parancsok

Ha valamit ki szeretnénk írni a képernyőre, azt amint az előző példában is láthattuk, a **write** és **writeln** *(write = ír, write line = sort ír)* parancsokkal tehetjük meg.

A **write** parancs kiírja a megadott szöveget a képernyőre. A kurzor közvetlenül a kiírt szöveg után marad, így a következő kiírásnál rögtön ez után íródik ki a további szöveg. Például:

```
program Pelda01a;
uses crt;
begin
clrscr;
 write('Hat fele:');
 write(3);
readln;
end.
```

Ez a program a következő szöveget fogja kiírni a képernyőre:

```
Hat fele:3_
```

Az első **write** parancs kiírja a *Hat fele:* szöveget. A kurzor közvetlenül a szöveg mögött marad. A második **write** kiírja a *3* számot. A kurzor a 3-as mögé kerül. Megfigyelhettük, hogy ha szöveget akarunk kiíratni, akkor azt mindig aposztrófok (') közé kell raknunk. Az aposztrófok közé írt szöveg egy az egyben kiíródik a képernyőre. Ha számot vagy valamilyen változó értékét írjuk ki, akkor elhagyjuk az aposztrófokat. Egy **write** paranccsal kiírhatunk szöveget és számot ill. változót is, ekkor az egyes elemeket (szöveg, szám vagy változó) vesszővel választjuk el egymástól. Az előző programunk így is kinézhetett volna, a végeredmény ugyanaz:

```
program Pelda01b;
uses crt;
begin
clrscr;
 write('Hat fele:',3);
readln;
end.
```

Mivel itt mindkét példánkba a 3-ast csak kiírtuk, nem számoltunk vele (és nem egy változó értékét írattuk ki), ezért természetesen ebben az esetben a 3-as számot kiírhatjuk úgy is, mint karaktert, tehát a következő képen:

```
program Pelda01c;
uses crt;
begin
clrscr;
 write('Hat fele:3');
readln;
end.
```

Az alábbi két példában jól láthatjuk a különbséget a szöveg, ill. szám kiírása között:

```
program Pelda02a;
uses crt;
begin
clrscr;
 write('2+8');
readln;
end.


program Pelda02b;
uses crt;
begin
clrscr;
 write(2+8);
readln;
end.
```

Az első példa pontosan azt írja ki a képernyőre, ami az aposztrófok között van:

```
2+8_
```

A második példában elhagytuk az aposztrófokat, így a program kiszámolja a 2 és 8 számok összegét és ezt írja ki a képernyőre:

```
10_
```

A **writeln** parancs hasonlóan működik, mint a **write** parancs. A különbség csupán annyi, hogy a **writeln** parancs egy egész sort ír ki, tehát miután kiírta a parancsban megadott szöveget vagy számot, a kurzor a következő sor elejére ugrik. Ezért az utána következő kiírásnál már a kiírandó szöveg az új sorba fog kerülni. Lássunk most erre is egy példát:

```
program Pelda03a;
uses crt;
begin
clrscr;
 writeln('8 es 2 osszege:',8+2);
 writeln('8 es 2 kulonbsege:',8-2);
readln;
end.
```

Ez a program a következőt fogja kiírni a képernyőre:

```
8 es 2 osszege:10
8 es 2 kulonbsege:6

_
```

Az első **writeln** parancs kiírja a *8 és 2 összege:* szöveget, majd mivel a következő rész már nincs idézőjelek között, kiszámolja a 8 + 2 eredményét és kiírja a képernyőre a *10*-et. Ezután, mivel mindez a **writeln** paranccsal lett kiíratva, a kurzor a következő sor elejére ugrik.

A második **writeln** parancs hasonlóan kiírja a *8 és 2 különbsége:* szöveget, kiszámolja mennyi 8 – 2 és az eredményt, tehát a *6*-os számot írja ki. Ez után, mivel ezt is **writeln** paranccsal írattuk ki, a kurzor ismét a következő sor elejére ugrik.

Feladat: Írjuk át az előző programot úgy, hogy az eredményt ne írja közvetlenül a szöveg után, tehát a kettőspont után legyen egy üres hely.

Megoldás: Az aposztrófok közé a kettőspont után teszünk egy szóközt. Így ezt a szóközt is egy az egyben kiírja a program a képernyőre és csak a szóköz után fogja kiírni a két szám összegét ill. különbségét.

```
program Pelda03b;
uses crt;
begin
clrscr;
 writeln('8 es 2 osszege: ',8+2);
 writeln('8 es 2 kulonbsege: ',8-2);
readln;
end.
```

A **writeln** parancsot használhatjuk üres sor kihagyására is, ekkor nem kell a parancs után megadnunk semmit. Valójában a parancs annyit fog tenni, hogy nem ír ki semmit a képernyőre és mivel **writeln** parancsról van szó, ezért a kurzort a következő sor elejére viszi. Módosítsuk az előző példánkat úgy, hogy két sor között hagyjunk ki egy üres sort. Ehhez a két **writeln** parancs közé írjunk be egy újabb **writeln** parancsot, melynél azonban ne adjunk meg semmilyen szöveget amit kiírjon – ilyenkor a zárójeleket sem kell kiraknunk. Programunk így néz ki:

```
program Pelda03c;
uses crt;
begin
clrscr;
 writeln('8 es 2 osszege: ',8+2);
 writeln;
 writeln('8 es 2 kulonbsege: ',8-2);
readln;
end.
```

Ha a programunkat most lefuttatjuk, a következőket fogjuk látni a képernyőn:

```
8 es 2 osszege: 10

8 es 2 kulonbsege: 6
_
```

Láthatjuk, hogy az első sor után az üres **writeln** parancsnak köszönhetően a program kihagyott egy üres sort.

Fontos még megjegyezni, hogy a pascal programozási nyelvben a szorzást *-al, az osztást pedig /-al jelöljük. Ha tehát hasonlóan ki szeretnénk íratni 8 és 2 szorzatát, ill. hányadosát, ezeket a 8*2 ill. 8/2 segítségével tehetjük meg. Ezen kívül az egészrészes osztásra használni fogjuk még a **div**, maradék meghatározására pedig a **mod** műveletet.

A **div** segítségével kiszámolhatjuk két szám hányadosának egész részét. Pl. a 11 div 6 értéke 1, mivel 11 osztva 6-tal egyenlő 1,83333...-mal és ennek az egész része (a tizedeseket levágva) 1.

A **mod** segítségével kiszámolhatjuk két szám egész osztásának maradékát. Pl. a **11 mod 6** értéke 5, mivel 11 osztva 6-tal egészekre kiszámolva 1 és a maradék 5 (11 : 6 = 1, maradék 5).