2 Változók használata

- **integer**, **string** típusú változók
- változók beolvasása billentyűzetről - **readln**
- **byte, real, char**, és **boolean** típusú változók

2.1 Integer, string típusú változók

Ha a programunk futtatása alatt tárolni szeretnénk valamilyen adatokat egy bizonyos ideig, akkor változókra lesz szükségünk. Mielőtt azonban bármilyen változót használni szeretnénk, azt a program elején fel kell sorolnunk (deklarálnunk kell) és meg kell határoznunk hogy minek a tárolására fogjuk használni az adott változót – egész számnak vagy szövegnek. Ez szerint két adattípussal ismerkedünk most meg.

Az **integer** típusú változókban egész számok tárolhatunk, tehát az ilyen típusú változóink értéke egész szám lehet. Az ilyen változó –32768-tól 32767-ig vehet fel értékeket.

A **string** típusú változó szöveg tárolására alkalmas, tehát az ilyen típusú változó értéke bármilyen szöveg lehet, melynek azonban a hossza nem lehet több mint 255 karakter.

Amint már említettük, ha változókat akarunk a programunkba használni, azokat a program elején a **var** kulcsszó után fel kell sorolnunk, ahol meg kell adnunk a változóink típusát is. Pl.:

```
program Pelda04;
uses crt;
var i:integer;
    s:string;
begin
clrscr;

readln;
end.
```

Ebben a programban két változót deklaráltunk. Az egyik változó neve: i, melynek értéke egész szám lehet (mivel **integer** típusú), a másik változó neve: s, amelynek az értéke szöveg lehet (mivel **string** típusú). A változók neve, ami az esetünkben az i és az s, bármilyen betű vagy szó lehet, de minden változónév csak betűre kezdődhet és csak az angol ábécé betűit és számokat tartalmazhat. Ha több ugyanolyan típusú változót szeretnénk használni, akkor azokat felsorolással, vesszővel elválasztva adhatjuk meg:

```
program Pelda05a;
uses crt;
var a1,a2,i:integer;
begin
clrscr;

readln;
end.
```

Ebben a programban 3 változót használtunk: a1, a2 és i. Mindhárom változónk **integer** típusú. Természetesen ha nekünk úgy jobban tetszik, a változók típusát akár külön-külön is megadhatjuk, vagy némely hasonló nevűt együtt csoportosíthatunk, az eredmény ugyanaz lesz:

```
program Pelda05b;
uses crt;
var a1,a2:integer;
    i:integer;
begin
clrscr;

readln;
end.
```

A fenti program az a1 és a2 típusát egyszerre, az i változó típusát külön adtuk meg, de ez a programunkon semmit sem változtat, hisz itt is ugyanazok a változóink vannak és mindhármuk típusa **integer**.

Miután a programunk elején a **var** szócska után deklaráltuk a változókat, elkezdhetjük ezeket használni a programunkban. A változónak valamilyen értéket a := (olvassuk így: **legyen egyenlő**) segítségével adhatunk. Például az i változónak így adhatunk értéket (így rakhatjuk bele a 25-ös számot):

```
i:=25;
```

Szavakkal: i legyen egyenlő 25-tel. Ha szöveges változónak (**string**) szeretnénk valamilyen értéket adni, akkor hasonlóan mint a **write** és **writeln** parancsoknál aposztrófokat kell használnunk:

```
s:='alma';
```

Mivel ez egy parancs, amivel a változónak értéket adunk, ezért az értékadás után természetesen pontosvesszőt kell tennünk.

Feladat: Mi a különbség a következő két utasítás között?

```
a:='z';                    a:=z;
```

Megoldás: Látjuk, hogy a különbség csak az aposztrófokban van. Mit jelent ez valójában?

- Az első utasításnál az **a** változóba beleraktuk a z betűt. Itt csak egy váltózónk van, mégpedig az **a**, amely egy **string** típusú változó (mivel egy z betűt raktunk bele).
- A második utasításnál nincs aposztróf. Ez annyit jelent, hogy az **a** változóba beleraktuk a **z** változó értékét (tartalmát). Ha például a parancs kiadása előtt **z** változóban a 25-ös szám volt, akkor az a:=z; paranccsal az **a** változóba is beleraktunk a 25-ös számot. Ebben az esetben tehát bármi is volt eredetileg az **a** változóban, a parancs végrehajtása után az **a**-ban is a 25-ös szám lesz. Amint láthatjuk, itt két ugyanolyan típusú változónk van, mégpedig az **a** és a **z** változó.

Lássunk most egy rövid kis programot, melyben két változót fogunk használni: **a** és **b**. Mindkét változónk integer típusú lesz. A programban az **a** változóba berakjuk a 8-as számot, a **b** változóba pedig a 6-ost. Végül kiíratjuk a képernyőre a két változó összegét:

```
program Pelda06;
uses crt;
var a,b:integer;
begin
clrscr;
 a:=8;
 b:=6;
 writeln('A ket valtozo osszege: ',a+b);
readln;
end.
```

Programunk a következőt fogja kiírni a képernyőre:

```
A ket valtozo osszege: 14
_
```

Láthattuk, hogy a kiíratásnál az a+b nincs aposztrófok között. Ez így helyes, hiszen nem azt a három karaktert akartuk kiíratni a képernyőre, hogy: **a+b**, hanem a két változó értékeinek – vagyis a 8-nak és a 6-nak – az összegét, tehát a 14-et. Jegyezzük meg, hogy a változók értékeinek kiíratásánál soha ne rakjunk aposztrófokat, hiszen a változók értékeit akarjuk kiíratni (tehát azt amit tartalmaznak).

Feladat: Van három **integer** típusú változónk. Állapítsuk meg, mit csinál a következő programrész az a és b változók értékeivel:

```
x:=a;
a:=b;
b:=x;
```

Megoldás: A programrész felcseréli **a** és **b** változók értékeit. Vegyünk egy példát. Legyen a programrész lefutása előtt az **a=5** és a **b=8**. Ekkor fenti a programrész a következőt fogja tenni:

1. Az első sor az **a** változóban levő értéket berakja az **x** változóba. Tehát az első sor lefutása után az **a**, **b** értéke változatlan, az **x** értéke pedig ugyanaz, mint az **a** változó értéke (tehát **a=5**, **b=8**, **x=5**).
2. A második sorban a **b** változó értékét raktuk bele az **a** változóba (tehát **a=8**, **b=8**, **x=5**). Így az **a** változóban levő eredeti értékünk elveszett, de mivel az első sorban ezt beraktuk az **x** változóba, ezért ott még megvan.
3. A harmadik sorban az **x** változóból raktuk át (vissza) az értékét a **b** változóba (tehát **a=8**, **b=5**, **x=5**).

Így valójában az **a** és **b** változó értékét felcseréltük. Az **x** segédváltozóra azért volt szükségünk, hogy ebben megőrizzük az **a** értékét mielőtt az **a-**ba beleraktuk volna a **b** értékét. Végül ebből az **x** változóból raktuk vissza a megőrzött értéket a **b** változóba.

Végül lássunk egy példát, melynél ugyanabba a változóba teszünk előbb 25-öt, majd 17-et. Ellenőrzésképpen mindig amikor a k változónak új értéket adunk, kiíratjuk a változó értékét a képernyőre. Megfigyelhetjük, hogy a **k** változó értéke mindig csak az utoljára megadott szám (ez előtte belerakott szám elveszik).

```
program Pelda07;
uses crt;
var k:integer;
begin
clrscr;
 k:=25;
 writeln('A k valtozo erteke: ',k);
 k:=17;
 writeln('A k valtozo erteke: ',k);
readln;
end.
```

*Megjegyzés: egy változónak amíg nem adunk a programban semmilyen értéket, addig az értéke **integer** típusú változó esetén: 0, **string** típusú változó esetén pedig üres szó. Ez azonban csak a pascal nyelvben igaz, más programozási nyelvekben a változó kezdeti értéke nincs meghatározva, ami annyit jelent hogy értéke kezdetben bármi lehet.*

2.2 Változók beolvasása billentyűzetről - readln

Eddig a változó értékét mindig előre megadtuk a programban. Gyakran előfordulhat azonban, hogy a változó értékét a program felhasználójától szeretnénk megtudni. A **readln** parancs segítségével a program futása alatt a felhasználó billentyűzeten keresztül adhatja meg az a változó értékét.

Például szeretnénk egy olyan programot készíteni, amelyet ha elindítunk, akkor a számítógép előbb megkérdezi a nevünket, majd miután megadtuk (pl. beírtuk hogy Micimackó) üdvözöl bennünket (kiírja, hogy Szia Micimackó!). Mivel a nevünk valamilyen szó, ezért a név tárolására egy **string** típusú változót fogunk használni. Programunk a következőképpen néz ki:

```
program Pelda08;
uses crt;
var nev:string;
begin
clrscr;
 write('Kerlek add meg a neved: ');
 readln(nev);
 writeln('Szia ',nev,'!');
readln;
end.
```

Miután a programot elindítottuk, lefut a **write** parancs, ami kiír egy mondatot a képernyőre és a kurzor a kiírt szöveg mögött marad. Majd lefut a **readln** parancs és ahol állt a kurzor, ott kéri a nev változó értékének a megadását. Tehát a következő jelenik meg a képernyőn:

```
Kerlek add meg a neved: _
```

A **readln** parancsnál a program addig vár, amíg nem írunk be valamilyen nevet, majd nem nyomjuk meg az ENTER billentyűt. Ekkor a kurzor a következő sor elejére ugrik, majd folytatódik a program futása.

A következő parancs a programban a **writeln**, amely kiírja az üdvözlő mondatot, majd a nev változó értékét (amit előtte beírtunk a **readln** parancsnál) és végül kirak a képernyőre egy felkiáltójelet, majd a kurzort a következő sor elejére teszi. Most ez olvasható a képernyőnkön:

```
Kerlek add meg a neved: Micimacko
Szia Micimacko!
_
```

A program használhatóságának szempontjából fontos, hogy mindig mielőtt beolvasunk valamilyen változót, írjuk ki a képernyőre, hogy minek a megadását várjuk a felhasználótól. Ezt legcélszerűbb a **write** paranccsal kiírni, mivel ekkor a felhasználó rögtön a kiírt szöveg mellett adhatja meg a változó értéket. Ha itt **write** helyett **writeln**-t használnánk, akkor a szöveg kiírása után a kurzor új sorba ugorna, így a beolvasásnál a felhasználó a következő sor elejére írná be a nevét. Logikailag természetesen ez is ugyanolyan jó, de az előzőnek szebb formája van.

Hasonlóan, mint a **write**, **writeln** parancsoknál, a **readln** parancsnak is van egy **read** verziója. Ennek azonban elsősorban a külső állományokból való beolvasásnál van jelentősége. A különbséget a kettő között egy rövid példán szemléltetjük:

```
program Pelda09;
uses crt;
var a,b,c:integer;
begin
clrscr;
 write('Irj be 3 szamot helykozzel elvalasztva: ');
 read(a,b);
 read(c);
 writeln('A beolvasott szamok: ',a,' ',b,' ',c);
 write('Irj be masik 3 szamot helykozzel elvalasztva: ');
 readln(a,b);
 read(c);      { mivel az elozo readln parancs uj sorra ugrott, ezert itt ez
                 a read parancs keri billentyuzetrol a c ertek megadasat }
  writeln('A beolvasott szamok: ',a,' ',b,' ',c);
readln;
end.
```

Ha lefuttatjuk a programot, a következő történik:

```
Irj be 3 szamot helykozzel elvalasztva: 3 5 7
A beolvasott szamok: 3 5 7
Irj be masik 3 szamot helykozzel elvalasztva: 2 4 6
8
A beolvasott szamok: 2 4 8

_
```

Amint láttuk a példánkban az első **read(a,b)** beolvassa a **3**, **5** számokat, de mivel ezeket **read**-el olvastattuk be a legközelebbi beolvasás innen folytatódhat. Ezért a következő **read(c)** parancsnál nem kell megadnunk billentyűzeten keresztül semmit, itt automatikusan beolvassa a **7**-es számot.

Miután ismét megadtunk három számot, a **readln(a,b)** parancs beolvassa ezek közül a **2**, **4**-es számokat, majd mivel ezt a **readln** (read line) utasítással végeztettük el, a legközelebbi beolvasás a következő sorban fog folytatódni. Ezért a **read(c)** már a következő (új) sor elején várja a **c** változó értékének billentyűzeten keresztüli megadását. Itt miután beírtuk a 8-as számot, az bekerül a **c** változóba.

Mivel a mi programjainkban általában egyszerre csak egy változó értékét fogjuk bekérni, ezért szinte mindig a **readln** parancsot fogjuk használni a billentyűzetről való beolvasáshoz.

Fenti példánkban észrevehettük még a kapcsos zárójelek (**{** ... **}**) használatát. Ezek segítségével programunkba bármilyen megjegyzéseket tehetünk, így később (akár hónapok, évek múlva) is könnyen olvasható, érthető lesz a program. Természetesen a program fordításakor és futtatásakor a FreePascal a kapcsos zárójelek közötti szöveget figyelmen kívül hagyja.

Feladat: Készítsünk programot, amely billentyűzetről beolvassa a négyzet oldalának a hosszát (aminek egész számot adunk meg), majd a megadott érték alapján kiszámolja a négyzet kerületét és területét.

Megoldás: A **readln** parancs segítségével beolvastatjuk a négyzet oldalának hosszát egy a változóba. Beolvasás előtt természetesen kiírjuk a felhasználónak hogy milyen adatot kérünk. Miután a felhasználó beírta a kért adatot, kiíratjuk a négyzet kerületét a matematikából jól ismert **k = 4.a** képlet segítségével, majd hasonlóan a négyzet területét a **T = a.a** képlet segítségével.

```
program Pelda10a;
uses crt;
var a:integer;
begin
clrscr;
 write('Kerem a negyzet oldalanak hosszat: ');
 readln(a);
 writeln('A negyzet kerulete: ',4*a);
 writeln('A negyzet terulete: ',a*a);
readln;
end.
```

Ha szeretnénk a kerületet és a területet is megőrizni (változókban tárolni), akkor használhatunk ezekre például egy k és egy t változót, melyekbe először kiszámoljuk a kerületet ill. területet, majd utána kiírjuk ezeknek a változóknak az értékét. Programunk ebben az esetben így néz ki:

```
program Pelda10b;
uses crt;
var a,k,t:integer;
begin
clrscr;
 write('Kerem a negyzet oldalanak hosszat: ');
 readln(a);
 k:=4*a;
 t:=a*a;
 writeln('A negyzet kerulete: ',k);
 writeln('A negyzet terulete: ',t);
readln;
end.
```

A fenti példában is láthattuk, hogy egy feladatnak több helyes megoldása is lehet. Próbáljuk meg ezek közül mindig az egyszerűbb, rövidebb, áttekinthetőbb megoldást megkeresni.

Feladat: Készítsünk hasonló programot a téglalap kerületének és területének kiszámítására. Feltételezzük, hogy itt is a téglalap oldalainak hossza egész szám.

Megoldás: A program a négyzet kerületének és területének kiszámításához hasonló lesz, de itt két értéket fogunk beolvasni: a téglalap a és b oldalának a hosszát, majd ezekből számoljuk ki a kerületet és a területet a matematikából jól ismert képletek segítségével. Egy helyes megoldás a feladatra:

```
program Pelda11a;
uses crt;
var a,b:integer;
begin
clrscr;
 writeln('Kerem a teglalap mereteit.');
 write('a = ');
 readln(a);
 write('b = ');
 readln(b);
 writeln('A teglalap kerulete, k = ',2*a+2*b);
 writeln('A teglalap terulete, T = ',a*b);
readln;
end.
```

2.3 byte, real, char, és boolean típusú változók

A **byte** típusú változónál 0-255-ig tudunk számot tárolni

A **real** (valós) típus ábrázolása 6 bájton, lebegőpontosan történik. Használata: x:real; x:8:2
Az értéktartomány:

$S_{min} = 2{,}9*10^{-39}$
$S_{max} = 1{,}7*10^{38}$

A **char** típus egy karakter használatánál hasznos. Egy bájtos típus, tehát $2^8 = 256$ különböző érték , az ASCII kódrendszer 256 elemének a tárolására képes. A karakter típusú változó egy ASCII kódot tartalmaz. Pl. ha a változóhoz tartozó memóriarekesz értéke 65, akkor mivel változónk típusa Char, ezt a rendszer 'A' betűként értelmezi.

A **boolean** egy logikai típusú változó két értéket vehet fel: *igaz* vagy *hamis*. Ábrázolására egy bájton történik (akár egy bit is elég lenne). Ha a bájt értéke 0, akkor a logikai típusúként értelmezett érték *hamis*, nullától eltérő érték esetén pedig *igaz. (true; false)*

Feladat: Kérj be egy szögértéket fokban, írasd ki a szinuszát! (használd a sin fügvényt)

```
program Pelda11b;
uses crt;
var x,r:real;
begin
  clrscr;
   Writeln('Kerem a szoget fokban: ');
   readln(x);
   r:=x*pi/180;
   writeln('A szog szinusza: ', sin(r):8:2);
  readln;
end.
```