17 Állományok kezelése

- szöveges állomány
- típusos állomány
- típus nélküli állomány

17.1 Szöveges állomány

A programból elmenthetünk ill. beolvashatunk adatokat állományokból. A Pascal-ban három fajta állománytípust különböztethetünk meg: szöveges állomány, típusos állomány és típus nélküli állomány.

Az első programunkban egy szöveges állományt fogunk kiíratni a képernyőre. Ehhez előbb hozzunk létre egy ilyen állományt. Szöveges állományt megírhatjuk pl. a windows jegyzettömbjében. Írjunk tehát a jegyzettömbben pár sort, majd mentsük el a szöveget abba a könyvtárba, ahova a Pascal programjainkat is szoktuk menteni. A szöveget "szoveg.txt" néven mentsük el. Ezen előkészületek után indítsuk el a FreePascal-t. Az alábbi program segítségével kiírathatjuk az elmentett állományt a képernyőre:

```
program Pelda53;
uses crt;
var f:text;
    s:string;
begin
clrscr;
assign(f,'szoveg.txt');
{$I-}
reset(f);
{$I+}
if IOResult<>0 then begin
                    writeln('Hiba: nincs meg a file.');
                    readln;
                    halt;
                    end;
while not eof(f) do begin
                    readln(f,s);
                    writeln(s);
                    readln;
                    end;
close(f);
end.
```

Ha szöveges állományt használunk, azt a változók deklarálásánál **text** típusként kell deklarálnunk.

Az **assign(f,'szoveg.txt');** parancs hozzárendeli az **f** állományhoz a merevlemezen található **szoveg.txt** állományt.

Ezek után a **reset(f);** parancs megnyitja az állományt olvasásra. A **{$I-}** és **{$I+}** utasítások kikapcsolják/bekapcsolják a pascal hibaellenőrzését, tehát azt, hogy ha a file megnyitásánál hiba keletkezik, akkor azt ne a pascal értékelje ki, hanem mi. A megnyitás után az **IOResult** változó értéke **0**, ha sikerült megnyitni az állományt, különben a hiba kódját tartalmazza. Hiba esetén a **halt;** parancsnál befejeződik a programunk futása.

Ha sikerült megnyitni az állományt, akkor egy ciklus segítségével kiolvasunk belőle egy sort a **readln(f,s);** paranccsal, majd ezt kiírjuk a képernyőre. A ciklus ezt a két utasítást addig ismétli, amíg nem értünk az állomány végére - **not eof(f)**. (eof = end of file).

Végül a **close(f)** paranccsal bezárjuk az állományt.

Mindig, amikor a programunkban állományt használunk előbb az **assign** paranccsal hozzárendeljük a változót a merevlemezen található állományhoz. Utána a **reset** paranccsal megnyithatjuk az állományt olvasásra vagy a **rewrite** paranccsal felülírásra. Szöveges állományból olvashatunk a **read, readln**

parancsokkal, írhatunk bele a **write, writeln** parancsokkal. Típusos állományoknál a **readln** és **writeln** parancsokat nem használhatjuk. Miután befejeztük a munkánkat az állománnyal, ne felejtsük el bezárni a fájlt a **close** paranccsal.

A következő program azt mutatja be, hogyan írhatunk szöveges állományba. Miután ezzel a programmal beírtunk valamit az állományba, megnyithatjuk azt olvasásra az előző programmal vagy akár bármilyen szövegszerkesztővel (pl. windows jegyzettömb).

```
program Pelda54;
uses crt;
var f:text;
    s:string;
begin
clrcsr;
assign(f,'szoveg.txt');
rewrite(f);
s:='elso sor';
writeln(f,s);
writeln(f,'masodik sor');
write(f,'harmadik ');
writeln(f,'sor');
close(f);
readln;
end.
```

A **rewrite** parancs törli az állomány tartalmát ha már létezik, majd megnyitja írásra. Ha még nem létezik ez az állomány a merevlemezen, akkor létrehozza azt.

17.2 Típusos állomány

A típusos állományba bármilyen típusú változókat (akár általunk létrehozott rekordokat) menthetünk el. Ezt a fajta állományt ha megnyitnánk szövegszerkesztővel, nem egy egyszerű szöveget látnánk, hanem mindenféle jeleket is (akárcsak pl. egy futtatható állomány vagy kép megnyitásakor).

A következő program megnyit egy állományt (ha létezik), beolvassa belőle az adatokat egy tömbbe majd a program végén elmenti a tömbből az adatokat az állományba.

```
program Pelda55;
var f:file of string;
    a:array [1..1000] of string;
    i,n:integer; { n - a tombben levo elemek szamat jeloli }
begin
n:=0;
{beolvasas allomanybol}
assign(f,'nevsor.dat');
{$I-}
reset(f);
{$I+}
if ioresult=0 then begin
                   while not eof(f) do begin
                                       n:=n+1;
                                       read(f,a[n]);
                                       end;
                   close(f);
                   end;
{ ... itt dolgozunk a tömbbel, megvaltoztatjuk, stb ... }
{kiiras allomanyba}
rewrite(f);
for i:=1 to n do write(f,a[i]);
close(f);
end.
```

17.3 Típus nélküli állomány

Míg egy típusos állományba csak ugyanolyan típusú változókat írhattunk, addig a típus nélküli állományba tetszés szerint egymás után beírhatunk bármilyen típusú változókat, vagy akár a memória egy tetszőleges részét. Amire azonban ügyelnünk kell, hogy az adatokat ugyanolyan sorrendben olvassuk ki, ahogy beírtuk azokat.

A következő program létrehoz egy ilyen típus nélküli állományt, majd beleír egy string típust, ezután három integer típust, végül még egy real típust is. A program második része megnyitja az elmentett állományt olvasásra, kiolvassa az adatokat és kiírja a képernyőre.

```
program Pelda56;
var f:file;
    nev:string;
    ev,honap,nap:integer;
    r:real;
begin
 write('Kerlek add meg a neved: ');
 readln(nev);
 write('Szuletesi datumod - ev (pl. 1985): ');
 readln(ev);
 write('Szutetesi datumod - honap (1..12): ');
 readln(honap);
 write('Szuletesi datumod - nap (1..31): ');
 readln(nap);
 write('Kerlek adj meg egy tetszoleges tizedes szamot: ');
 readln(r);
 {mentes tipus nelkuli fajlba}
 assign(f,'adatok.dat');
 {$I-}
 rewrite(f,1);
 {$I+}
 if ioresult<>0 then begin
                     writeln('Hiba a fajl megnyitasanal!');
                     halt;
                     end;
 blockwrite(f,nev,sizeof(nev));
 blockwrite(f,ev,sizeof(ev));
 blockwrite(f,honap,sizeof(honap));
 blockwrite(f,nap,sizeof(nap));
 blockwrite(f,r,sizeof(r));
 close(f);
 {beolvasas a fajlbol}
 {$I-}
 reset(f,1);
 {$I+}
 if ioresult<>0 then begin
                     writeln('Hiba a fajl megnyitasanal!');
                     halt;
                     end;
 blockread(f,nev,sizeof(nev));
 blockread(f,ev,sizeof(ev));
 blockread(f,honap,sizeof(honap));
 blockread(f,nap,sizeof(nap));
 blockread(f,r,sizeof(r));
 close(f);
 {adatok kiirasa a kepernyore}
 writeln('Fajlbol visszaolvasott adatok:');
 writeln('Nev: ',nev);
 writeln('Szul.dat.: ',ev,'.',honap,'.',nap,'.');
 writeln('Tizedes szam: ',r);
end.
```

A típus nélküli állomány deklarálásánál a **file** kulcsszót kell használnunk.

Az állományt a **reset** ill. **rewrite** paranccsal nyithatjuk meg olvasásra ill. írásra. Itt a **reset** ill. **rewrite** parancsok második paramétereként (ezt az előző fájltípusoknál nem adtuk meg) meg kell adnunk egy blokk méretét. Ezt célszerű 1 bájtra adni, ahogy a mintapéldánkban is tettük.

Ezek után a típus nélküli állományból a **blockread** paranccsal olvashatunk, írni pedig a **blockwrite** paranccsal írhatunk. Ezeknek a parancsoknak az első paramétere maga a fájl, a második paraméter a fájlba beírandó vagy kiolvasandó változó neve (ill. mutató egy memóriacímre), a harmadik paraméter pedig egy szám, amely megadja, hogy mennyi blokkot akarunk olvasni ill. írni a fájlba (a második paraméterben megadott memóriacímtől kezdődően). Mi a példaprogramban a harmadik paraméter megadásánál a **sizeof** függvény segítségével megállapítottuk, hogy mennyi bájton van tárolva az adott változó a memóriában és ennyi blokkot olvastunk ill. írtunk a fájlba (mivel a mi esetünkben 1 block = 1 bájt). **seek**(file,pozíció) direkt file-ban a megadott sorszámú rekordra áll (file elején 0). **filepos**(file) függvény, megadja, hogy a direkt file hányadik rekordjánál tartunk.

Végül a fájlt a **close** paranccsal be kell zárnunk, hasonlóan mint a többi fájltípusnál.

Feladat:
A következő példaprogram egy típusos állomány kezelésére készült. A program egy bolt árucikkeinek adatait (név, kód, ár) tárolja és kezeli egy állományban. Hozzunk létre egy menüt. A peldaprogramban alkalmazzunk függvényeket, melyeket csak meghívunk a főprogramban.

```
1. Adatbevitel
2. Modositas
3. Torles
4. Listazas
5. Vege
Valassz!
```

Megoldás:

```
program Pelda57;
uses crt;
type TAru = record        {A fajl alaptipusa.}
         kod: string;
         nev: string[15];
         ar: real;
         t: boolean;
      {Ez a mezo jelzi, hogy e rekord torolt-e (logikai törlés).}
       end;

var bolt: file of TAru;
    aru: TAru;
    mkod: string;
    mvalasz: char;

{Megkeres egy adott kodu rekordot az allomanyban.}
function Van(kodja: string): boolean;
var talalt: boolean;
begin
  seek(bolt,0);
  talalt := false;
  while not Eof(bolt) and not talalt do
    begin
      read(bolt, aru);
      if (aru.kod = mkod) and not aru.t then
         talalt := true;
     end;
  van := talalt;
end;
```

```
{Egy rekord felvitele az allomanyba.}
procedure Bevitel;
begin
  ClrScr;
  WriteLn('Kerem a kodot!');
  ReadLn(mkod);
  if not Van(mkod) then
    begin
      Seek(bolt, filesize(bolt));
          {Pozicionalas a fajl vegere.}
      WriteLn('Kerem az aru nevet!');
      ReadLn(aru.nev);
      WriteLn('Kerem az aru arat!');
      ReadLn(aru.ar);
      aru.t := false;
      aru.kod := mkod;
      Write(bolt, aru);
    end
  else
    begin
      WriteLn('Mar van ilyen kod!');
      ReadKey
    end
end;

{Egy rekord modositasa a fajlban.}
procedure Modosit;
begin
  ClrScr;
  WriteLn('Kerem az aru kodjat!');
  ReadLn(mkod);
  if Van(mkod) then
    begin
      Seek(bolt, FilePos(bolt) - 1);
      WriteLn('Kerem az aru nevet!');
      ReadLn(aru.nev);
      WriteLn('Kerem az aru arat!');
      ReadLn(aru.ar);
      aru.t := false;
      aru.kod := mkod;
      Write(bolt, aru);
    end
  else
    begin
      Writeln('Nincs ilyen aru!');
      ReadKey
    end;
end;
```

```
{Egy rekord logikai torlese: a t mezot True ertekure allitja,
 az ilyen rekordokat a program nem letezonek tekinti.
 Fizikai torles kilepeskor.}

procedure Torles;
begin
  ClrScr;
  WriteLn('Kerem az aru kodjat!');
  ReadLn(mkod);
  if Van(mkod) then
    begin
      Seek(bolt, FilePos(bolt) - 1);
      aru.t := true;
      Write(bolt, aru);
    end
  else
    begin
      WriteLn('Nincs ilyen aru!');
      ReadKey
    end
end;

{A fajl tartalmanak kiirasa a kepernyore.}
procedure Lista;
begin
  ClrScr;
  Seek(bolt, 0);
  while not Eof(bolt) do
    begin
      Read(bolt, aru);
      if aru.t = false then
        begin
          Write(aru.kod);
          GotoXy(30, wherey); write(aru.nev);
          GotoXy(60, wherey); writeln(aru.ar:10:0);
        end;
    end;
   ReadKey
end;

{Fizikai torles: azon rekordok atmasolasa egy uj allomanyba, melyek nincsenek
logikailag torolve.
 A regi áalomany torlese, az uj fajl atnevezese a regi nevure.}
procedure Surites;
var ujfile: file of TAru;
begin
  Assign(ujfile, 'ujfile');
  Rewrite(ujfile);
  Seek(bolt, 0);
  while not Eof(bolt) do
    begin
      Read(bolt, aru);
      if aru.t = false then write(ujfile, aru);
     end;
  Close(bolt);
  Erase(bolt);
  Close(ujfile);
  Rename(ujfile, 'bolt');
end;
```

```
{Foprogram, menu.}
begin
  clrscr;
  Assign(bolt, 'bolt');
  {$I-}
  Reset(bolt);
  {$I+}
  if IOResult <> 0 then Rewrite(bolt);
  repeat
    ClrScr;
    WriteLn('1. Adatbevitel');
    WriteLn('2. Modositas');
    WriteLn('3. Torles');
    WriteLn('4. Listazas');
    WriteLn('5. Vege');
    WriteLn('Valassz!');
    repeat mvalasz := readkey until mvalasz in['1'..'5'];
    case mvalasz of
      '1': bevitel;
      '2': modosit;
      '3': torles;
      '4': lista;
      '5': surites;
    end;
  until mvalasz = '5';
end.
```

Feladat: Az a.dat állomány integerekből épül fel, írasd ki a számtani átlagukat! (Ahhoz, hogy ez a program működjön, létre kell hozni az a.dat-ot!)

Megoldás:

```
program Pelda58;
uses crt;
var f:file of integer;
    ossz,x:integer;
begin
  clrscr;
     assign(f,'a.dat');
     reset(f);
     while not eof(f) do
     begin
          read(f,x);
          ossz:=ossz+x;
     end;
     Writeln('Atlag: ',ossz/filesize(f):10:2);
  readln;
end.
```