F2. Adatállományok kezelése

F2.1 Feladatok szöveges állományok kezelésére

Írjunk programot, amely rákérdez az áruk számára és beolvassa az arufajta típusú rekord adatait és kiírja *aruk.txt* szöveges állományba. Olvassuk vissza az adatokat, számítsuk ki az áruk össz értékét, keressük meg a legkevesebb darabszámú, és a legdrágább árut! (*ARUK\aruk*)

```
program aruk;
type
       arufajta = record
          db: integer;
          ara: real;
           azonosito: string[10];
       end;
var
    f:text;
    aru,aru_min,aru_max:arufajta;
    OsszErtek:real;
    i, n:integer;
begin
  OsszErtek:=0;
  aru_min.db := maxint;
  aru_max.ara := 0;
  writeln('Az áru adatainak állományba való írása');
  assign(f,'aruk.txt');
  rewrite(f);
  write('Az áruk száma: ');
  readln(n);
  for i:= 1 to n do
  begin
    write('Áru azonosítója: '); readln(aru.azonosito);
    write('     darabszáma: '); readln(aru.db);
    write('           ára: '); readln(aru.ara);
    writeln(f, aru.db:4,aru.ara:12:2, aru.azonosito:11);
  end;
  close(f);
  writeln;
  writeln('Az áru adatainak állományból való olvasása');
  assign(f,'aruk.txt');
  {$I-}   { az  I/O hibák figyelésének kikapcsolása }
  reset(f);
  if ioresult<>0 then  { hibakezelés }
  begin
    writeln(' állomány nem érhető el!');     readln;
    halt(1);
  end;
```

```
  {$I+}   { az  I/O hibák figyelésének visszakapcsolása }

  while not eof(f) do
  begin
   readln(f,aru.db,aru.ara,aru.azonosito);
   writeln(aru.azonosito:10,'  ',aru.db:5,'  ',aru.ara:6:1);
   OsszErtek:= OsszErtek+aru.ara*aru.db;
   if aru.ara>aru_max.ara then aru_max:=aru;
   if aru.db<aru_min.db then aru_min:=aru;
  end;
  close(f);
  writeln;
  writeln('Az aru ossz erteke: ',Osszertek:6:2);
  writeln('Aru: ', aru_min.azonosito,' legkevesebb darab: ',aru_min.db);
  writeln('Aru: ', aru_max.azonosito,' legdragabb: ',aru_max.ara:6:2);
  readln;
end.
```

A program futásának eredménye:

```
Az áru adatainak állományba való írása
Az áruk száma: 4
Áru azonosítója: asztal
     darabszáma: 4
            ára: 1200
Áru azonosítója: szek
     darabszáma: 12
            ára: 500
Áru azonosítója: szekreny
     darabszáma: 2
            ára: 2500
Áru azonosítója: fotel
     darabszáma: 1
            ára: 1500

Az áru adatainak állományból való olvasása
     asztal      4  1200.0
       szek     12   500.0
   szekreny      2  2500.0
      fotel      1  1500.0

Az áru össz értéke: 17300.00
Áru:       fotel legkevesebb darab: 1
Áru:    szekreny legdrágább: 2500.00
```

Írjunk programot, amely beolvassa a *vonat.txt* állományt. Számítsuk ki a vonat átlagsebességét, a keressük meg a legdrágább jegyet, számláljuk meg vonatok és a hálókocsik számát! (*VONAT\vonatpr*)

A *vonat.txt* állomány tartalma:

```
 Seb. Ár  hálók. azonosító
  56 150 1 szemely
  80 200 1 sebes
 100 220 0 gyors
 120 400 0 express
  70 120 1 szemely
```

```
program vonatpr;
type
       vonat = record
          sebesseg: integer;
          ar: real;
          halokocsi:integer;
          azonosito: string[10];
       end;
var
    f:text;
    v:vonat;
    vonatok_szama: integer;
    atlag_seb, max_ar: real;
    halo_db:integer;
begin
  halo_db:=0; max_ar:=0; atlag_seb:=0;
  assign(f,'vonat.txt');
  reset(f);
  writeln('  sebesség  jegy ára hálókocsi  azonosító');
  while not eof(f) do
  begin
   readln(f, v.sebesseg,v.ar,v.halokocsi,v.azonosito);
   writeln(v.sebesseg:10,v.ar:10:1,v.halokocsi:4,v.azonosito:15);
   atlag_seb:= atlag_seb+v.sebesseg;
   vonatok_szama:=vonatok_szama+1;
   if v.ar> max_ar then max_ar:=v.ar;
   if v.halokocsi = 1 then halo_db:=halo_db+1;
  end;
  atlag_seb:=atlag_seb/vonatok_szama;
  close(f);
  writeln;
  writeln('Átlagsebesség   : ',atlag_seb:6:2);
  writeln('Hálókocsik száma: ',halo_db);
  writeln('Legdrágább jegy : ',max_ar:6:2);
  writeln('Vonatok száma   : ',vonatok_szama);
  readln;
end.
```

A program futásának eredménye:

```
  sebesség  jegy ára hálókocsi  azonosító
        56     150.0   1        szemely
        80     200.0   1        sebes
       100     220.0   0          gyors
       120     400.0   0        express
        70     120.0   1        szemely

Átlagsebesség   :  85.20
Hálókocsik száma: 3
Legdrágább jegy : 400.00
Vonatok száma   : 5
```

Írjunk menüvezérelt programot, amely winchester azonosítóját és kapacitását tárolja rekord típusú adatstruktúrában. Szöveges állományba írja, illetve visszaolvassa, megjeleníti az állomány tartalmát és megkeresi a legnagyobb kapacitású winchestert! (*HDD\menupr*)

```
program menupr;     {Text típusú állomány}
type
    hdd = record
             azonosito: string[10];
             kapacitas: integer;
          end;
    bolt = array[1..20] of hdd;
    fh = text;
var
   Trigon:bolt;
   van:boolean;
   db, m, mhiba:integer;
   mstr : string;
   Maxkapacitas:integer;
   fx:fh;

procedure hddIr(var f:fh);
var h:hdd;
    i,n:integer;
begin

     write('Hard diszkek száma : ');readln(n);
     assign(f,'hdd.TXT');rewrite(f);
     for i:=1 to n do
         begin
              writeln;
              write('Azonosító: ');readln(h.azonosito);
              write('Kapacitás : ');readln(h.kapacitas);
              writeln(f,h.kapacitas:4, #32, h.azonosito);
         end;
     close(f);
end;
```

```
procedure hddOlvas(var f:fh;var x:bolt;var n:integer);
begin
   assign(f,'hdd.TXT');
   reset(f);
   n:=0;
   writeln('  Azonosító   Kapacitás');
   writeln('------------------------');
  while not eof(f) do
   begin
    n:=n+1;
    readln(f,x[n].kapacitas, x[n].azonosito);
    writeln(x[n].azonosito:10, x[n].kapacitas:10);
   end;
  close(f);
end;
function maxkap(x:bolt; n:integer):integer;
var i,ns:integer;

begin
  ns:=x[1].kapacitas;
    for i:=2 to n do
     if ns<x[i].kapacitas then ns:=x[i].kapacitas;
 maxkap:=ns;
end;

begin
    van := false;
    repeat
      writeln('1. Adatok megadása');
      writeln('2. Adatok beolvasása');
      writeln('3. Max kapacitás keresése');
      writeln('4. Kilépés');
      writeln;
      write('Menü: '); readln(mstr);
      val(mstr, m, mhiba);
      case m of
       1: begin
            writeln('Adatok adatállományba írása');
            hddir(fx);
            van:=false;
          end;
       2: begin
            writeln('Adatok beolvasasa fájlból');
            hddOlvas(fx,Trigon,db);
            van:=true;
          end;
       3: begin
           if van then
           begin
              Maxkapacitas:=maxkap(Trigon,db);
              writeln('A legnagyobb kapacitás: ',Maxkapacitas);
           end
           else
             writeln('Kérem az adatbázist beolvasni!');
          end;
```

```
      end;
      writeln;
      writeln('<Enter> ');
      readln;

    until m=4;
end.
```

A program futásának eredménye:

```
1. Adatok megadása
2. Adatok beolvasása
3. Max kapacitás keresése
4. Kilépés

Menü: 2
Adatok beolvasasa fájlból
  Azonosító   Kapacitás
------------------------
         WD         40
     MAXTOR        250
        IBM         80
    FUJITSU         32

<Enter>

1. Adatok megadása
2. Adatok beolvasása
3. Max kapacitás keresése
4. Kilépés

Menü: 3
A legnagyobb kapacitás: 250

<Enter>

1. Adatok megadása

2. Adatok beolvasása

3. Max kapacitás keresése

4. Kilépés

Menü: 4
```

Írjunk programot, amely a mondat.txt szöveges állományból mondatokat és szétválogatja szöveges állományba írva a kijelentő, a kérdő és a felkiáltó mondatokat, valamint meg is jeleníti a képernyőn! (*VALOGAT\valogat*)

```
program valogat;
uses graph;
var
   ki, fe, ke: integer;
   procedure olvas(var kij,felk,kerd:integer);
   var
      f  : text;
      ki_file, fel_file, kerd_file :text;
      i  : integer;
      ki_volt, felk_volt, kerd_volt: integer;
      sor: string;
      ch : char;
   begin
      assign(f,'mondat.txt');
      reset(f);
      writeln('    db  mondat ');
      ki_volt:=0;  felk_volt:=0; kerd_volt:=0;
      kij:=0; felk:=0; kerd:=0;
      i:=0;

      while not eof(f) do
      begin
          while not eoln(f) do
          begin
              i:=i+1;
              read(f,ch);
              sor[i]:= ch;
              case ch  of
              '.'  : begin kij:=kij+i; sor[0]:= chr(i);
                           writeln(kij:6,'  ', sor);
                           if ki_volt = 0 then
                            begin
                              assign(ki_file,'kimondat.txt');
                              rewrite(ki_file); ki_volt := 1;
                            end
                            else
                              begin
                               append(ki_file);
                              end;
                           writeln(ki_file,sor);
                           close(ki_file);
                           i:=0;
                     end;
              '!'  : begin felk:=felk+i;sor[0]:= chr(i);
                           writeln(felk:6,'  ', sor);
                           if felk_volt = 0 then
                            begin
                              assign(fel_file,'felmondat.txt');
```

```
                              rewrite(fel_file); felk_volt := 1;
                            end
                            else
                              begin
                               append(fel_file);
                              end;
                           writeln(fel_file,sor);
                           close(fel_file);
                           i:=0;
                    end;
            '?'  : begin kerd:=kerd+i;sor[0]:= chr(i);
                           writeln(kerd:6,'  ', sor);
                           if kerd_volt = 0 then
                            begin
                              assign(kerd_file,'kermondat.txt');
                              rewrite(kerd_file); kerd_volt := 1;
                            end
                            else
                              begin
                               append(kerd_file);
                              end;
                           writeln(kerd_file,sor);
                           close(kerd_file);
                           i:=0;
                    end;
              end;
          end;
          readln(f);
      end;
      close(f);
  end;
  procedure kiir(ki, fe, ke: integer);
  var
      ki_file, fel_file, kerd_file :text;
      sor : string;
  begin
    if ki > 0 then begin
                    writeln;
                    writeln('Kijelentő mondat:');
                    assign(ki_file,'kimondat.txt');
                    reset(ki_file);
                    while( not eof(ki_file)) do
                    begin
                     readln(ki_file,sor);
                     writeln(sor);
                    end;
                    close(ki_file);
                  end;
    if fe > 0 then begin
                    writeln;
                    writeln('Felkiáltó mondat:');
                    assign(fel_file,'felmondat.txt');
                    reset(fel_file);
```

```
                    while( not eof(fel_file)) do
                    begin
                     readln(fel_file,sor);
                     writeln(sor);
                    end;
                    close(fel_file);
                  end;
    if ke > 0 then begin
                    writeln;
                    writeln('Kérdő mondat');
                    assign(kerd_file,'kermondat.txt');
                    reset(kerd_file);
                    while( not eof(kerd_file)) do
                    begin
                     readln(kerd_file,sor);
                     writeln(sor);
                    end;
                    close(kerd_file);
                  end;
  end;

begin
  olvas(ki, fe, ke);
  kiir(ki, fe, ke);
  readln;
end.
```

A program futásának eredménye:

```
    db  mondat
    12  Esik az eső.
    12  Szép az idő?
     7  Nagyon!
    25  De szép ez a kert!
    20  Rendben.
    40  Holnap vasárnap van.
    27  Voltál sétálni?

Kijelentő mondat:
Esik az eső.
Rendben.
Holnap vasárnap van.

Felkiáltó mondat:
Nagyon!
De szép ez a kert!

Kérdő mondat
Szép az idő?
Voltál sétálni?
```

Írjunk programot, amely beolvas két szöveges állományt és megfejti az üzenetet! (*KODOLO\kodol1*)

Használja az alábbi típusdefiníciót és deklarációt:

```
type
            Ukod = string[80];
            Skod = array[1..20] of integer;

var
        fu, fk :text;
        titok, Kod: Ukod;
        Sk   : Skod;
        db   : integerl;
```

Olvassunk be két TEXT típusú KOD.TXT és UZENET.TXT állományt. Az UZENET.TXT állomány egysoros, max 80 karakteres szöveget tartalmazhat. A KOD.TXT állomány egysoros, maximálisan 20 egész számot tartalmazhat.

Az *fk* a KOD.TXT állomány fájlváltozója. Olvassuk be az egysoros karaktersorozatot a *Kod* sztringbe.
Az *fu* az UZENET.TXT állomány fájlváltozója. Olvassuk be az egysorban lévő egész számokat a sorvégét figyelve számonként az *Sk* tömbbe, számlálva a beolvasott számok darabszámát. Egyébként az *Sk* tömb utolsó eleme a tömb tényleges darabszámát tartalmazza.

Az *Sk* tömb páratlan indexű elemének tartalma kódolja ki a *Kod* sztringból a *titok* sztringbe a keresett szöveget. Pl: titok[1] := Kod[Sk[1]]; titok[2]:= Kod[Sk [3]]; stb.

KOD.TXT állomány tartalma: `A(LklS6CuAoPt-fgZklHrt`
UZENET.TXT állomány tartalma:

```
12  7  1  23   6   3   8   1  10   4   3   8   14   0   17   2  20   17
```

Kódolja ki az üzenetet és írja ki a képernyőre.

```
program kodol1;
type
   Ukod = string[80];
   Skod = array[1..20] of integer;
   fx = text;

var
  fk, fu: text;
  titok,Kod : Ukod;
  Sk    : Skod;
  i,j,k,db : integer;
```

```
begin
  assign(fk,'KOD.TXT');
  reset(fk);
  readln(fk,Kod);
  close(fk);
  writeln(Kod);
  assign(fu,'UZENET.TXT');
  reset(fu);
  i:=0;
  while not eoln(fu) do
  begin
    i:=i+1;
    read(fu,Sk[i]);
    write(Sk[i]:3);
  end;
  readln(fu);
  writeln;
  close(fu);
  db := sk[i];
  k:=0;
  for j:=1 to db do
  begin
    if odd(j) then
                begin
                   k:=k+1;
                   titok[k]:= Kod[Sk[j]];
                end;
  end;
  titok[0]:= char(db);
  writeln('titok: ',titok);
  readln;
end.
```

A program futásának eredménye:

```
12  7  1 23  6  3  8  1 10  4  3  8 14  0 17  2 20 17
titok: PASCAL-ZH
```

Írjunk programot, amely beolvassa a gyumolcs.txt állományt és ábécé szerint rendezi, majd kiírja! Használjunk alprogramokat a feladat megvalósítására! (*GYUM\gyumolv*)

```
program gyumolv;
 type
       s=string[12];
       tomb= array[1..100] of s;
 var
       db : integer;
       gy : tomb;
```

```
procedure olvas(var x:tomb; var n: integer);
var
   i,j: integer;
   sz : s;
   f  : text;
   c  : char;
begin
  assign(f,'gyumolcs.txt');
  reset(f);
  j:=0;
  n:=0;
  while not eof(f) do
  begin
      read(f,c);
      if c<> ' ' then begin
                         j:= j+1;
                         sz[j]:=c;
                      end
                 else begin
                         sz[0]:= chr(j);
                         j:=0;
                         n:=n+1;
                         x[n]:=sz;
                      end;
   end;
end;
procedure kiir(x: tomb; n:integer);
var
   i:integer;
 begin
   for i:=1 to n do
    writeln(x[i]);
 end;
procedure rendez(var x:tomb; n:integer);
var
  i,j:integer;
  sz: s;
begin
  for i:=1 to n-1 do
   for j:=i+1 to n do
   begin
     if x[i] > x[j] then begin
                           sz:= x[i]; x[i]:=x[j]; x[j]:=sz;
                         end;
   end;
end;
begin
 olvas(gy,db);
 writeln('A gyumolcs.txt tartalma'); kiir(gy,db);
 rendez(gy,db); writeln; writeln('Rendezett adatok');
 kiir(gy,db);
 readln;
end.
```

A program futásának eredménye:

```
A gyumolcs.txt tartalma
alma
korte
szilva
barack
szolo
dinnye
malna
dio

Rendezett adatok
alma
barack
dinnye
dio
korte
malna
szilva
szolo
```

Írjunk programot, amely numerikus adatokat tartalmazó szöveges állományt olvassa be és egy valós tömbbe tárolja. Számítsuk ki az adatok összegét és átlagát. Az adatok száma nem több 50-nél. A feladat megoldásához használjunk alprogramokat! (*NUM\numolv*)

```
program numolv;
const N=50;
  type
    tomb=array[1..N] of real;

  var
    x,y1:tomb;
    db,n1:integer;
    ossz,szorz:real;
  procedure olvas(var y:tomb; var n:integer);
    var i:integer;
        f:text;
        szam:real;
    begin
      assign(f,'numadat.txt');
      reset(f);
      n:=0;
      i:=1;
      while not eof(f) do
      begin
        readln(f,szam);
        writeln('x[',i:2,']= ',szam:5:2);
        y[i]:=szam;  n:=n+1;
        i:=i+1;
      end;
```

```
      close(f);
    end;
  function osszeg(y:tomb; n:integer):real;
    var szum:real; i:integer;
      begin
        szum:=0;
        for i:=1 to n do
        begin
          szum:=szum+y[i];
        end;
        osszeg:=szum;
      end;
  function szorzat(y:tomb; n:integer):real;
    var szor:real; i:integer;
      begin
        szor:=1;
        for i:=1 to n do
        begin
          szor:=szor*y[i];
        end;
      szorzat:=szor;
    end;
begin
  olvas(y1,n1);
  ossz:=osszeg(y1,n1);
  writeln('Az elemek osszege: ',ossz:5:2);
  szorz:=szorzat(y1,n1);
  write('Az elemek szorzata: ',szorz:5:2);
  readln;
end.
```

A program futásának eredménye:

```
x[ 1]=  5.00
x[ 2]=  8.00
x[ 3]= 12.00
x[ 4]=  4.00
x[ 5]= 10.00
Az elemek osszege: 39.00
Az elemek szorzata: 19200.00
```

Írjunk programot, amely *tanulo* típusú rekordot olvas a billentyűzetről és a kívánt névvel szöveges állományba tárolja! (*DIAK\FileIr*)

```
program FileIr;
type
    tanulo = record
       neve : string[15];
       kora : integer;
       sulya: real;
     end;
var
     f:text;
```

```
     tan:tanulo;
     i,n:integer;
     fnev: string[12];
 begin
    write('A fájl neve: '); readln(fnev);
    assign(f,fnev);
    {$I-}
    rewrite(f);
    if ioresult <> 0 then
                          begin
                            writeln('Fájl hiba');
                            Halt(1);
                          end;
   {$I+}
   write('tanulók száma: '); readln(n);
   for i:=1 to n do
   begin
     write('Neve: '); readln(tan.neve);
     write('Kora: '); readln(tan.kora);
     write('Súlya: '); readln(tan.sulya);

     writeln(f,tan.kora:4, #32, tan.sulya:6:1, #32, tan.neve);
   end;
   close (f);
end.
```

A program futásának eredménye:

```
A fájl neve: diakok.txt
tanulók száma: 3
Neve: Kiss Lajos
Kora: 18
Súlya: 67

Neve: Nagy Attila
Kora: 21
Súlya: 70

Neve: Toth Imre
Kora: 22
Súlya: 75
```

Írjunk programot, amely *tanulo* típusú rekordot olvas be szöveges állományból, visszaírja az adatokat, és megkeresi a legsúlyosabb diákot! (*DIAK\FileOlv*)

```
program FileOlv;
type
    tanulo = record
       neve : string[15];
       kora : integer;
       sulya: real;
     end;
```

```
 var
     f:text;
     tan, tans:tanulo;
     sulyos:real;
     fnev: string[12];
 begin
    write('A fájl neve: '); readln(fnev);
    assign(f,fnev);
    {$I-}
    reset(f);
    if ioresult <> 0 then
                          begin
                            writeln('A fájl nem elérhető');
                            Halt(1);
                          end;
   {$I+}
   tans.sulya:=0;
   writeln;
   while not eof(f) do
   begin
     readln(f, tan.kora, tan.sulya, tan.neve);
     writeln(tan.neve:15, tan.kora:4, tan.sulya:8:1);
     if tan.sulya > tans.sulya then tans := tan
   end;
   close(f);
   writeln('A legsúlyosabb diak: ',tans.neve,' kora: ',
           tans.kora,'  súlya: ',
   tans.sulya:8:1);
   readln;
end.
```

A program futásának eredménye:

```
A fájl neve: diakok.txt
A fájl neve: diakok.txt

     Kiss Lajos  18    67.0
    Nagy Attila  21    70.0
      Toth Imre  22    75.0
A legsúlyosabb diak:  Toth Imre kora: 22  súlya:     75.0
```

Írjunk programot, amely beolvas egy angol szavat és magyarra fordítja két szöveges állomány beolvasásával. A feladat megoldásához használjuk alprogramokat! (*SZOTAR\szotaroz*)

```
program szotaroz;
vtar
   angszo,mszo: string;
   sorsz: integer;
```

```
 procedure keresa(s :string;var sorszam:integer);
 var
    f  : text;
    i,j: integer;
    ch : char;
    sor:string;
 begin
    assign(f,'angol.txt');
    reset(f);
    i:=0;
    j:=0;
    sorszam := 0;
    repeat
        while not eoln(f) do
        begin
            i:=i+1;
            read(f,ch);
            sor[i]:= ch;
        end;
        readln(f);
        j:=j+1;
        sor[0]:=chr(i);
        i:=0;
        if sor = s then sorszam:=j;
    until eof(f) or (sorszam <> 0);
    close(f);
end;
procedure keresm(var s :string; sorszam:integer);
 var
    f  : text;
    i,j: integer;
    sor:string;
    ch : char;
 begin
    assign(f,'magyar.txt');
    reset(f);
    i:=0;
    j:=0;
    s:='';
    repeat
        while not eoln(f) do
        begin
            i:=i+1;
            read(f,ch);
            sor[i]:= ch;
        end;
        readln(f);
        j:=j+1;
        sor[0]:=chr(i);
        i:=0;
        if sorszam = j then s:=sor;
    until eof(f) or (s <> '');
close(f);
end;
```

```
begin
  write('Kérem az angol szavat:'); readln(angszo);
  keresa(angszo,sorsz);
  if sorsz > 0 then
                  begin
                    keresm(mszo,sorsz);
                    if mszo <> '' then
                                    writeln(angszo,' ',mszo)
                                  else
                                    writeln('Nem találta');
                  end
                 else writeln('Nincs a szótárban');
   readln;
end.
```

A program futásainak eredménye:

```
Kérem az angol szavat:window
window ablak

Kérem az angol szavat:table
table asztal

Kérem az angol szavat:chair
Nincs a szótárban
```

ANGOL.TXT tartalma

```
picture
pen
book
apple
table
window
```

MAGYAR.TXT tartalma

```
kép
toll
könyv
alma
asztal
ablak
```

A szókészlet bővíthető!

Írjunk programot, amely beolvas egy mondatot a pontig, kiírja szöveges állományba, majd vissza is olvassa és megszámlálja a magánhangzók számát és az egyéb jeleket! Használjunk alprogramokat a feladat megvalósítására! (*KAR_TEXT\charf1*)

```
program charf1;
type
   fv = file of char;
var
   f      : fv;
   filenev: string;
   c      : char;
   egyeb, maganh: integer;

procedure ir(var f:fv; filenev:string);
var
    ch:char;
begin
    assign(f,filenev);
    rewrite(f);
    writeln('A szöveg végét a . jelzi:');
    ch:='*';
    while ch <> '.' do
    begin
       read(ch);
       write(f,ch);
    end;
    close(f);
end;
procedure olvas(var f:fv; filenev:string;
                var e,mgh:integer);
var
   c:char;
begin
   assign(f,filenev);
   reset(f);
   mgh:=0; e:=0;
   writeln;
   writeln('Fájlból olvas:');
   while not eof(f) do
   begin
      read(f,c);
      write(c);
      if c in ['a','o','i','u','e','A','O','I','U','E']
      then mgh:= mgh+1
      else e:= e+1
   end;
   writeln;
   close(f);
 end;
 begin
    write('Fájl neve: '); readln(filenev);
    ir(f,filenev);
    olvas(f, filenev, egyeb, maganh);
```

```
    writeln; writeln('Statisztika: ');
    writeln('Magánhangzók száma : ',maganh);
    writeln('Egyéb              : ', egyeb);
    readln;  readln;
end.
```

A program futásának eredménye:

```
Fájl neve: adat.txt
A szöveg végét a . jelzi:
ma szep az ido.

Fájlból olvas:
ma szep az ido.

Statisztika:
Magánhangzók száma : 5
Egyéb              : 10
```

F2.2 Típusos állományok kezelése

Írjunk programot, amely beolvas egy mondatot a pontig, kiírja típusos állományba, majd vissza is olvassa és statisztikát készít a magánhangzókról, a mássalhangzókról, az egyéb jelekről és a szóközről! Használjunk alprogramokat a feladat megvalósítására! (*KAR_TIP\charf2*)

```
{R-}
program charf2;
type
   fv = file of char;
var
   f      : fv;
   filenev: string;
   c      : char;
   maganh,massalh,jelek,egyeb: integer;
procedure ir(var f:fv; filenev:string);
var
    ch:char;
begin
    assign(f,filenev);
    rewrite(f);
    writeln('A szöveg végét a . jelzi:');
    ch:='*';
    while ch <> '.' do
    begin
       read(ch);
       write(f,ch);
    end;
    close(f);
end;
procedure olvas(var f:fv; filenev:string;
                var mgh,msh,j,e:integer);
type
   Kisabc    = set of 'a'..'z';
   Nagyabc   = set of 'A'..'Z';
var
   Kabc  : Kisabc;
   Nabc  : Nagyabc;
   Kmgh,Kmsh: Kisabc;
   Nmgh,Nmsh:Nagyabc;
   egyeb_jel : set of char;
   c:char;
begin
   Kabc := ['a'..'z'];
   Nabc := ['A'..'Z'];
   Kmgh := ['a','e','i','u','o'];
   Kmsh := Kabc-Kmgh;
   Nmgh := ['A','E','I','U','O'];
```

```
   Nmsh := Nabc-Nmgh;
   egyeb_jel := ['.',',',':',';'];
   assign(f,filenev);
   reset(f);
   mgh:=0; msh:=0; e:=0; j:=0;
   writeln;
   writeln('Fájlból olvas:');
   read(f,c);
   while c <> '.' do
   begin
      write(c);
      write(c);
      if c in Kmgh+Nmgh
      then maganh:= maganh+1
      else
      if c in Kmsh+Nmsh then msh:=msh+1
      else
      if c in egyeb_jel then j:= j+1
      else e:= e+1;
   end;
   writeln;
   close(f);
 end;
 begin
    write('Fájl neve: '); readln(filenev);
    ir(f,filenev);
    olvas(f, filenev,maganh,massalh,jelek,egyeb);
    writeln; writeln('Statisztika: ');
    writeln('Magánhangzók száma : ', maganh);
    writeln('Mássalhangzók száma: ', massalh);
    writeln('Jelek ;:,.         : ', jelek);
    writeln('Egyéb jelek, szóköz: ', egyeb);
    readln; readln;
end.
```

A program futásának eredménye:

```
Fájl neve: adat.dat
A szöveg végét a . jelzi:
ma szep az ido.

Fájlból olvas:
ma szep az ido.

Statisztika:
Magánhangzók száma : 5
Mássalhangzók száma: 6
Jelek ;:,.         : 1
Egyéb jelek, szóköz: 3
```

Írjunk programot, amely rákérdez a személyek számára, beolvassa a *szemely* típusú rekord adatait, és kiírja típusos állományba. Olvassuk vissza az adatokat és tömbbe tároljuk el. Számláljuk meg az adott kornál nagyobb vagy egyenlő és adott súlynál nagyobb személyek számát, valamint az adott kornál fiatalabbak számát! A feladat megoldásához használjuk alprogramokat! (*TIP_REK**recfile*)

```
program recfile;

type szemely=record
         nev:string[20];
         ev:integer;
         suly:real;
     end;
     tomb=array[1..40] of szemely;
     fsz = file of szemely;
     st12=string[12];
var
    sz  :tomb;
    n,db,kora,h:integer;
    sulya: real;
    ft  : fsz;
    fnev: st12;

procedure diskfileir(var f: fsz; fn:st12);
var i,m :integer;
    s   :szemely;
begin
   assign(f,fn);rewrite(f);
   write('Személyek száma: ');readln(m);
   for i:=1 to m do
   begin
       writeln;
       write('Neve : ');readln(s.nev);
       write('Kora : ');readln(s.ev);
       write('Súlya: ');readln(s.suly);
       write(f,s);
   end;
   close(f);
end;
procedure diskfileolvas(var f: fsz; fn:st12; var t:tomb;
                        var m:integer);
var i:integer;
    s:szemely;
begin
   assign(f,fn);
   writeln;
   writeln('Diszk fájlból olvas ');
   reset(f);
   writeln('Adatlista:');
   m:=0;
```

```
   while not eof(f) do
   begin
       m:=m+1;
       read(f,s);
       t[m]:=s;
       writeln;
       writeln('Neve : ',t[m].nev);
       writeln('Kora : ',t[m].ev);
       writeln('Súlya: ',t[m].suly:3:1);
   end;
   close(f);
end;
procedure keres_suly(t:tomb; m:integer; kor: integer; suly:real;
                     var k:integer);
var i:integer;
begin
   k:=0;
   for i:=1 to m do
     if (t[i].ev >= kor) antd (t[i].suly > suly)
        then k:=k+1;
end;
function keres_kor(t:tomb; m:integer; kor: integer):integer;
var i,k:integer;
begin
   k:=0;
   for i:=1 to m do
     if (t[i].ev <= kor)
        then k:=k+1;
   keres_kor:= k;
end;
begin
   write('Az állomány neve: ');
   readln(fnev);
   diskfileir(ft, fnev);
   diskfileolvas(ft, fnev, sz,n);
   write('A keresett személy kora     : '); readln(kora);
   write('A keresett személy súlya    : '); readln(sulya);
   keres_suly(sz,n,kora,sulya,db);
   writeln('A keresett személyek száma  : ',db);
   writeln;
   write('A keresett személy kora     : '); readln(kora);
   h := keres_kor(sz,n,kora);
   writeln('A keresett személyek száma  : ',h);
   readln;
end.
```

A program futásának eredménye:

```
Az állomány neve: diakok.dat
Személyek száma: 3

Neve : Nagy Attila
Kora : 28
Súlya: 56
```

```
Neve : Kiss Katalin
Kora : 18
Súlya: 49

Neve : Toth Marton
Kora : 32
Súlya: 67

Diszk fájlból olvas
Adatlista:

Neve : Nagy Attila
Kora : 28
Súlya: 56.0

Neve : Kiss Katalin
Kora : 18
Súlya: 49.0

Neve : Toth Marton
Kora : 32
Súlya: 67.0
A keresett személy kora     : 20
A keresett személy súlya    : 50
A keresett személyek száma  : 2

A keresett személy kora     : 20
A keresett személyek száma  : 1
```

Írjunk programot, amely rákérdez az állomány nevére, beolvassa és kijelzi, hogy az állomány hány rekordból áll. Kérdezzünk rá a rekord számára és jelenítsük meg a személy adatait! (*TIP_REK\keresrec*)

```
program KeresRec;
type
    tanulo = record
       neve : string[20];
       kora : integer;
       sulya: real;
     end;
var
     f:file of tanulo;
     tan, tans:tanulo;
     sulyos:real;
     fnev: string[12];
     h : longint;
     rec, i: integer;
 begin
    write('A fájl neve: '); readln(fnev);
    assign(f,fnev);
    {$I-}
    reset(f);
```

```
    if ioresult <> 0 then
                          begin
                            writeln('A fájl nem elérhető');
                            Halt(1);
                          end;
   {$I+}
   h := filesize(f);
   writeln('Össz rekord száma ( 0 - ',h-1,' )');
   for i:= 0 to h-1 do
   begin
     repeat
       write('Rekord száma: ');
       readln(rec);
     until rec<=h-1;
     seek(f,rec);
     tans.sulya:=0;
     read(f,tan);
     writeln(tan.neve:15, tan.kora:4, tan.sulya:8:1);
   end;
   close(f);
   readln;
end.
```

A program futásának eredménye:

```
A fájl neve: diakok.dat
Össz rekord száma ( 0 - 2 )
Rekord száma: 1
   Kiss Katalin  18    49.0
Rekord száma: 0
    Nagy Attila  28    56.0
Rekord száma: 2
    Toth Marton  32    67.0
```

Írjunk programot, amely rákérdez az állomány nevére, és az adatok számára, beolvassa az egész típusú adatokat és típusos állományba írja ki. Olvassuk vissza az adatokat, tároljuk egy egész típusú tömbe, és számláljuk a pozitív, a negatív és zérus elemek számát! A feladat megoldásához használjunk alprogramokat!
(*TIP_NUM**recfile2*)

```
program recfil2;
type
      fv = file of integer;
      tomb = array[1..100] of integer;

var
   f : fv;
   filenev: string;
   x : tomb;
   db, p, z, m : integer;
```

```
procedure ir(var f:fv; filenev:string);
var
    i,k,n: integer;
begin
    assign(f,filenev);
    rewrite(f);
    write('Elemek száma : '); readln(n);
    for i:=1 to n do
    begin
       write('adat: '); readln(k);
       write(f,k);
    end;
    close(f);
end;
procedure olvas(var f:fv; filenev:string; var t: tomb;
                var neg,poz,zero:integer; var n: integer);
var
   k : integer;
begin
    assign(f,filenev);
    reset(f);
    n:= 0;
    poz:=0; neg:=0; zero:=0;
    while not eof(f) do
    begin
        read(f,k);
        n:= n+1;
        t[n]:=k;
        if t[n] < 0 then neg := neg+1;
        if t[n] > 0 then poz := poz+1;
        if t[n] = 0 then zero:= zero+1;
    end;
    writeln;
    close(f);
 end;
 begin
     write('Kérem a fájl nevét: '); readln(filenev);
     ir(f,filenev);
     olvas(f, filenev,x, db, p, z, m);
     writeln('A negatív elemek száma: ',db);
     writeln('A pozitív elemek száma: ',p);
     writeln('A zérus   elemek száma: ',z);
     readln;
end.
```

A program futásának eredménye:

```
Kérem a fájl nevét: szamok.dat
Elemek száma : 5
adat: -2
adat: 12
adat: 0
adat: 5
adat: -3
```

```
A negatív elemek száma: 2
A pozitív elemek száma: 2
A zérus   elemek száma: 1
```

Írjunk programot, amely beolvassa a billentyűzetről és a *par.dat* típusos állományba írja a *partip* típusú rekord adatait. A feladat megoldásához használjuk alprogramokat! (*PAR\par_ir*)

```
(* par_ir.pas *)
program par_ir;
type
      nevtip = string[20];
      parmuttip = ^partip;
      partip = record
           fiu, lany: nevtip;
           fiu_kora : integer;
           kovetkezo: parmuttip;
           elozo : parmuttip;
      end;
      file_tip = file of partip;
var

  f  : file_tip;

procedure file_ir(var f:file_tip);
var
   par :parmuttip;
   fiu : string;
begin
   assign(f,'par.dat');
   rewrite(f);      { irásra nyitja a fájlt }
   new(par);
   repeat
     write('Fiú neve : '); readln(fiu);
     if fiu <> '' then
     begin
        par^.fiu :=fiu;
        write('Kora     : '); readln(par^.fiu_kora);
        write('Lány neve: '); readln(par^.lany);
        writeln;
        write(f,par^);
    end;
  until fiu = '';
  dispose(par);
  close(f);
end;
begin
  file_ir(f);
  readln;
end.
```

A program futásának eredménye:

```
Fiú neve : Laszlo
Kora     : 35
Lány neve: Szilvia

Fiú neve : Zoltan
Kora     : 32
Lány neve: Zsuzsanna

Fiú neve : Ferenc
Kora     : 23
Lány neve: Klara

Fiú neve : Tamas
Kora     : 41
Lány neve: Monika

Fiú neve :
```

Írjunk programot, amely beolvassa a *par.dat* típusos állományt írja és megjeleníti a képernyőn. Olvassunk be egy korhatárt és számláljuk meg, hogy hány fiú esik a korhatár alá! A feladat megoldásához használjuk alprogramokat! (*PAR\par_olv1*)

```
(* par_olv1.pas *)
program par_olv1;
type
    nevtip = string[20];
    parmuttip = ^partip;
    partip = record
        fiu, lany: nevtip;
        fiu_kora : integer;
        kovetkezo: parmuttip;
        elozo : parmuttip;
    end;
    file_tip = file of partip;
var
     f:file_tip;
     db, korhatar:integer;

procedure file_olvas(var f:file_tip; kor:integer; var db:integer);
var
   paros : partip;
begin
   assign(f,'par.dat');
   reset(f); { olvasásra nyitja a file-t }
   {$I-}
   if ioresult <> 0 then
                      begin
                        writeln('A fájl nem létezik!');
                        exit;
                      end;
```

```
   {$I+}
   db :=0;   { számláló változó nullázása }
   writeln;
   writeln('       Fiú neve   Kora    Lány neve');

   while not eof(f) do { fájl vége ? }
   begin
      { egy rekord beolvasása a fájlból a páros recordba }
      read(f,paros);
      writeln(paros.fiu:15,paros.fiu_kora:6,paros.lany:15);
      if( paros.fiu_kora <= kor) then db := db+1;
   end;
   writeln;
   close(f); { file zárása }
end;
begin
   write('Korhatár: '); readln(korhatar);  { kor beolvasása }
   file_olvas(f,korhatar,db);         { olvasás file-ból és számlálás }
   writeln(korhatar:2,' év alatti fiúk száma: ',db);
   readln;
end.
```

A program futásának eredménye:

```
Korhatár: 28

       Fiú neve   Kora    Lány neve
         Laszlo    35        Szilvia
         Zoltan    32      Zsuzsanna
         Ferenc    23          Klara
          Tamas    41         Monika

28 év alatti fiúk száma: 1
```

Írjunk programot, amely beolvassa a *par.dat* típusos állományt írja és megjeleníti a képernyőn az adatokon előre és visszafelé listazva. Olvassunk be egy korhatárt és számláljuk meg, hogy hány fiú esik a korhatár alá! A feladat megoldásához használjuk alprogramokat! (*PAR\par_olv2*)

```
program par_olv2;
type
    nevtip = string[20];
    parmuttip = ^partip;
    partip = record        { pár }
        fiu, lany:nevtip;
        fiu_kora :integer;
        kovetkezo:parmuttip;
        elozo    :parmuttip;
    end;
    file_tip = file of partip;
```

```
var
   f: file_tip;
   db1,db2, korhatar:integer;
   elso_par, utolso_par: parmuttip;
procedure file_olvas(var f:file_tip;
                     var elso, utolso: parmuttip);
var
    uj   :parmuttip; { a párra mutato pointer }
   paros :partip;    { a pár rekordja }
begin
   assign(f,'par.dat');
   reset(f);
   {$I-}
   if ioresult <> 0 then
                      begin
                        writeln('A fájl nem létezik!');
                        exit;
                      end;
   {$I+}
   elso := nil;
   while not eof(f) do
   begin
     new(uj);                          { új rekord helye a memóriában }
     read(f,paros); { egy rekord beolvasása a fájlból a pár recordba }
     uj^.fiu := paros.fiu;                 { az új rekord feltöltése }
     uj^.lany := paros.lany;
     uj^.fiu_kora := paros.fiu_kora;
     if elso = nil then
         begin
            elso:= uj;    { az első rekordra mutat }
            elso^.kovetkezo := nil; elso^.elozo     := nil;
         end
         else
         begin
             utolso^.kovetkezo:=uj; { előre láncolás }
             uj^.elozo := utolso;   { az előző mutató kitöltése }
             uj^.kovetkezo := nil;
         end;
     utolso := uj;
   end;
   close(f);
end;

function keres_elore(kor: integer; elso: parmuttip):integer;
var
   p : parmuttip;
   db: integer;
begin
   db:=0;
   p:=elso;
   writeln;
   writeln('       Fiú neve   Kora    Lány neve');
```

```
   while p <> nil do
   begin
      if( p^.fiu_kora <= kor) then db:=db+1;
      writeln(p^.fiu:15,p^.fiu_kora:6,p^.lany:15);
      p:= p^.kovetkezo;
   end;
   writeln;
   keres_elore:=db;
end;
function keres_hatra(kor: integer; utolso: parmuttip):integer;
var
   p : parmuttip;
   db: integer;
begin
   db:= 0;
   p:=utolso;
   writeln;
   writeln('       Fiú neve   Kora    Lány neve');
   while p <> nil do
   begin
     if(p^.fiu_kora <= kor) then db:=db+1;
     writeln(p^.fiu:15,p^.fiu_kora:6,p^.lany:15);
     p:= p^.elozo;
   end;
   writeln;
   keres_hatra:=db;
end;
begin
   file_olvas(f,elso_par, utolso_par);
   write('Korhatár: '); readln(korhatar);
   writeln;writeln('Elôre keresés a láncban');
   db1:=keres_elore(korhatar,elso_par);
   writeln('    ',korhatar:2,' év alatti fiúk száma: ',db1);
   writeln; writeln('Hátra keresés a láncban');
   db2:=keres_hatra(korhatar,utolso_par);
   writeln('    ',korhatar:2,' év alatti fiúk száma: ',db2);
   readln;
end.
```

A program futásának eredménye:

```
Korhatár: 28

Előre keresés a láncban

       Fiú neve   Kora    Lány neve
         Laszlo    35        Szilvia
         Zoltan    32      Zsuzsanna
         Ferenc    23          Klara
          Tamas    41         Monika

    28 év alatti fiúk száma: 1
```

```
Hátra keresés a láncban

       Fiú neve   Kora    Lány neve
          Tamas    41         Monika
         Ferenc    23          Klara
         Zoltan    32       Zsuzsanna
         Laszlo    35         Szilvia

    28 év alatti fiúk száma: 1
```

Írjunk programot, amely beolvassa a *par.dat* típusos állományt írja és megjeleníti a képernyőn. Olvassunk be egy fiú nevet és próbáljuk törölni a láncból, ha nem találjuk írjunk hibajelzést! Az *Enter* zárja a fiú nevének beolvasását! A feladat megoldásához használjuk alprogramokat! (*PAR\par_del*)

```
program par_del;
type
    nevtip = string[20];
    parmuttip = ^partip;
    partip = record        { pár }
        fiu, lany:nevtip;
        fiu_kora :integer;
        kovetkezo:parmuttip;
        elozo    :parmuttip;
    end;
    file_tip = file of partip;
var
   f: file_tip;
   fiu:nevtip;
   db1,db2, korhatar:integer;
   elso_par, utolso_par: parmuttip;

procedure file_olvas(var f:file_tip;
                     var elso, utolso: parmuttip);
var
    uj   :parmuttip; { a párra mutato pointer }
   paros :partip;    { a pár rekordja }
begin
   assign(f,'par.dat');
   reset(f);
   {$I-}
   if ioresult <> 0 then
                      begin
                        writeln('A fájl nem létezik!');
                        exit;
                      end;
   {$I+}
   elso := nil;
```

```
   while not eof(f) do
   begin
     new(uj);                          { új rekord helye a memóriában }
     read(f,paros); { egy rekord beolvasása a fájlból a pár recordba }
     uj^.fiu := paros.fiu;                { az új rekord feltöltése }
     uj^.lany := paros.lany;
     uj^.fiu_kora := paros.fiu_kora;
     if elso = nil then
         begin
            elso:= uj;    { az első rekordra mutat }
            elso^.kovetkezo := nil; elso^.elozo     := nil;
         end
         else

         begin
             utolso^.kovetkezo:=uj; { előre láncolás }
             uj^.elozo := utolso;   { az előző mutató kitöltése }
             uj^.kovetkezo := nil;
         end;
     utolso := uj;
   end;
   close(f);
end;

function keres_elore(kor: integer; elso: parmuttip):integer;
var
   p : parmuttip;
   db: integer;
begin
   db:=0;
   p:=elso;
   writeln;
   writeln('       Fiú neve   Kora    Lány neve');
   while p <> nil do
   begin
      if( p^.fiu_kora <= kor) then db:=db+1;
      writeln(p^.fiu:15,p^.fiu_kora:6,p^.lany:15);
      p:= p^.kovetkezo;
   end;
   writeln;
   keres_elore:=db;
end;
function keres_hatra(kor: integer; utolso: parmuttip):integer;
var
   p : parmuttip;
   db: integer;
begin
   db:= 0;
   p:=utolso;
   writeln;
   writeln('       Fiú neve   Kora    Lány neve');
```

```
   while p <> nil do
   begin
     if(p^.fiu_kora <= kor) then db:=db+1;
     writeln(p^.fiu:15,p^.fiu_kora:6,p^.lany:15);
     p:= p^.elozo;
   end;
   writeln;
   keres_hatra:=db;
end;
procedure kiir(elso:parmuttip);
var
   p : parmuttip;
begin
   p:=elso;
   if p<>nil then
   begin
     writeln;
     writeln('       Fiú neve   Kora    Lány neve');
   end;
   while p <> nil do
   begin
      writeln(p^.fiu:15,p^.fiu_kora:6,p^.lany:15);
      p:= p^.kovetkezo;
   end;
   writeln;
end;
procedure torol(fiunev:nevtip; var elso,utolso:parmuttip);
var
   p : parmuttip;
   van : integer;
begin
   p:=elso;
   van:= 0;
   while p <> nil do
   begin
      if( p^.fiu = fiunev) then
                              begin
                                  van := 1;
                                  if p^.elozo<>nil then
                                     p^.elozo)^.kovetkezo:=p^.kovetkezo
                                  else
                                      elso:=p^.kovetkezo;
                                  if p^.kovetkezo<>nil then
                                       (p^.kovetkezo)^.elozo:=p^.elozo
                                  else utolso:=p^.elozo;
                                dispose(p); break;
                              end;
      p:= p^.kovetkezo;
   end;
   if van = 0 then writeln(fiunev,' nem szerepelt a lácban!');
end;
begin
   file_olvas(f,elso_par, utolso_par);
```

```
   kiir(elso_par);
   writeln('Elem törlése a láncból (Enter a vége)');
   write('Fiú neve :'); readln(fiu);
   while fiu<>'' do
   begin
      torol(fiu,elso_par, utolso_par);
      kiir(elso_par);
      write('Fiú neve (törlésre) :'); readln(fiu);
   end;
end.
```

A program futásának eredménye:

```
       Fiú neve   Kora    Lány neve
         Laszlo    35        Szilvia
         Zoltan    32      Zsuzsanna
         Ferenc    23          Klara
          Tamas    41         Monika

Elem törlése a láncból (Enter a vége)
Fiú neve :Ferenc

       Fiú neve   Kora    Lány neve
         Laszlo    35        Szilvia
         Zoltan    32      Zsuzsanna
          Tamas    41         Monika

Fiú neve :Kalman
Kalman nem szerepelt a lácban!

       Fiú neve   Kora    Lány neve
         Laszlo    35        Szilvia
         Zoltan    32      Zsuzsanna
         Ferenc    23          Klara
          Tamas    41         Monika

Fiú neve (törlésre) :
```