- Alapok
- Karakterláncok
- Tömbök
- Szöveges állományok
- Típusos állományok
- Halmazok
- Memóriakezelés
- Dinamikus adatszerkezetek
- Típusnélküli állományok
- Rendszerközeli programozás
- Overlay-technika
- Grafika
- Assembly rutinok
- Objektum orientált programozás
- Turbo Access
- Turbo Sort
- Egérkezelés
- Memóriabővítések

Angster Erzsébet
Kertész László

Turbo Pascal 6.0 feladatgyűjtemény

I.

Oktatócsomag lemezmelléklettel

Angster Erzsébet – Kertész László

Turbo Pascal 6.0 feladatgyűjtemény

I.

Második, javított kiadás

Felelős kiadó: Angster Erzsébet, Kertész László

Készült a Kertészeti és Élelmiszeripari Egyetem Nyomdájában, 1993-ban.

Felelős vezető: Wilpert Gábor

Megrendelés száma: 93/040

ISBN 963 400 535 70
ISBN 963 400 536 5

Előszó

Turbo Pascal 6.0 feladatgyűjteményünk első kötetét tartja kezében a Kedves Olvasó. A könyvet azoknak ajánljuk, akik most kezdenek ismerkedni a Turbo Pascal nyelvvel, vagy alapismereteiket szándékoznak bővíteni. A második kötet feladatait azok számára állítottuk össze, akik összetettebb, a gép és a nyelv által adott lehetőségeket mélyebben kihasználó programok készítéséhez várnak segítséget.

A feladatgyűjteményt sokéves oktatói és gyakorló programozói tapasztalatunk alapján készítettük. A két kötetben igyekeztünk teljeskörű áttekintést adni a nyelv lehetőségeiről, leggyakoribb felhasználási területeiről. Bemutatunk jónéhány, a programozásban gyakran használatos módszert, struktúrált programtervezési fogást.

A kötetben feldolgozott anyagot kilenc fejezetre osztottuk – témakörök szerint. Minden fejezet könnyebb feladatokkal kezdődik, majd fokozatosan egyre összetettebb, nehezebb problémákat oldunk meg. A feladatok megfogalmazása után szerepelnek a forráskódok, melyeket minden esetben teszteltünk. A programokat igyekeztünk úgy írni, hogy azok olvashatóak, könnyen érthetőek legyenek (tiszta, struktúrált szerkezet, beszédes azonosítók, sok megjegyzés).

A programok a Turbo Pascal 5.5, 6.0, 7.0 és a Borland Pascal 7.0 verziója alatt is futtathatók – ezalól 5.5 esetén kivétel a Memóriakezelés fejezet néhány programja, valamint azok, amelyek a $X direktívát használják.

A forráskódokat lemezen mellékeljük, így hosszadalmas gépelés nélkül is követheti a Kedves Olvasó a programok működését.

Kérjük, a könyvvel kapcsolatos észrevételeit, tapasztalatait írja meg az alábbi címre:

Angster–Kertész

Budapest, Pf. 701/234

1399

a szerzők

ALAPOK

KARAKTERLÁNCOK

TÖMBÖK

ALAPOK

1. feladat

Olvassuk be képernyőről egy piskótatorta méreteit – átmérőjét és magasságát -, valamint a rá teendő krém vastagságát cm–ben. Számoljuk ki, mennyi krémre van szükség a torta bevonásához, ha 5%–os ráhagyással dolgozunk (gyerekek is vannak a családban...) !

{ A kapcsos zárójelben lévő szövegek megjegyzések, azokat a fordító figyelmen kívül hagyja (kivétel a fordítási direktíva, melyet közvetlenül a nyitó kapcsos zárójel után álló '$' karakter vezet be). }

```
  { Programfej. A program neve: Alap1 ; célszerű a katalógusban is ilyen néven tárolni: }
Program Alap1 ;

  { Változók deklarációja. A programban használt összes változót deklarálni kell, és meg kell adni
    típusukat! A változók fix memóriahelyet foglalnak, értékük a program futása során változhat: }
Var
    { Valós típusú változók: }
  Atmero,
  NagyAtmero,
  Magassag,
  Kremvastagsag,
  OldalKrem,
  TetoKrem,
  OsszKrem,
  Nyalakodnivalo : Real ;

  { Programblokk, a program utasításai (FŐPROGRAM): }
Begin
    { A torta átmérőjének beolvasása képernyőről. Szöveg kiírása a képernyőre a kurzor helyétől
      kezdődően. A kurzor a szöveg utáni pozícióra kerül: }
  Write('A torta átmérôje: ') ;
    { Adatbevitel a képernyőről. Az Atmero változó tartalma ezután a beütött érték lesz. Az
      adatbevitel az <ENTER> leütésére fejeződik be, a kurzor a következő sor elejére kerül. Ha
      a bevitt adat nem felel meg a változó típusának, a program futási hibával leáll: }
  ReadLn(Atmero) ;
    { A torta magasságának beolvasása képernyőről: }
  Write('A torta magassága: ') ;
  ReadLn(Magassag) ;
    { A krém vastagságának beolvasása képernyőről: }
  Write('A krém vastagsága: ') ;
  ReadLn(Kremvastagsag) ;

    { Részadatok kiszámítása. Átmérő, a krémet is beleszámítva: }
  NagyAtmero := Atmero + 2 * Kremvastagsag ;
```

```
    { Az oldalkrém térfogata = (Torta alapterülete a krémet is beszámítva – Torta alapterülete
      krém nélkül) * Torta magassága: }
  OldalKrem := (Sqr(NagyAtmero / 2) - Sqr(Atmero / 2)) * Pi *
               Magassag ;

    { A tetőkrém térfogata = Nagy alapterület * Torta magassága: }
  TetoKrem := Sqr(NagyAtmero / 2) * Pi * Kremvastagsag ;

    { A teljes krémmennyiség kiszámítása, majd a nyalakodnivaló hozzáadása: }
  OsszKrem := OldalKrem + TetoKrem ;
  Nyalakodnivalo := OsszKrem * 0.05 ;
  OsszKrem := OsszKrem + Nyalakodnivalo ;


    { Az eredmény kiírása. Az eljárás vesszővel elválasztott paraméterei egymás után kerülnek
      a képernyőre sorfolytonosan (az OsszKrem változó 8 karakternyi helyet fog elfoglalni, amiből
      egy karakter a tizedespont, kettő pedig a tizedesjegyek száma): }
  WriteLn('A tortához szükséges krém mennyisége: ',OsszKrem:8:2,
          ' cm3, vagyis ',OsszKrem / 1000:5:2,' liter') ;
  WriteLn('Ebbôl a gyerekeké: ',Nyalakodnivalo / 1000:4:2,
          ' liter') ;


    { Várunk az <ENTER> leütésére, hogy ne szaladjon el a képernyő (a program lefutása után
      a keretrendszer automatikusan visszakapja a vezérlést). Ha a programot nem a keretrend-
      szerből futtatjuk, erre nincs szükség! }
  ReadLn ;
End.
```

2. feladat

Olvassuk be egy háromszög három oldalát cm–ben (egy oldal maximum 255 cm lehet)! Amennyiben szerkeszthető e három adatból háromszög, számítsuk ki annak területét!

```
Program Alap2 ;

Var
    { Egész típusú változók, melyek értékei 0 és 255 közé eshetnek: }
  A,
  B,
  C        : Byte ;
  Terulet,
  S        : Real ;

    { Logikai változó, mely az igaz (True), vagy a hamis (False) értéket veheti fel: }
  Szerkesztheto : Boolean ;
```

```
  { FŐPROGRAM: }
Begin
    { A háromszög oldalainak beolvasása: }
  Write('A oldal: ') ;
  ReadLn(A) ;
  Write('B oldal: ') ;
  ReadLn(B) ;
  Write('C oldal: ') ;
  ReadLn(C) ;

    { Szerkeszthető háromszög? }
  Szerkesztheto := (A + B > C) And (A + C > B) And (B + C > A);
  If Szerkesztheto
  Then
      { Igen: }
    Begin
        { A háromszög félkerülete: }
      S := (A + B + C) / 2 ;
        { Terület kiszámítása (Heron képlet): }
      Terulet := Sqrt(S * (S - A) * (S - B) * (S - C)) ;
      Write('A háromszög területe: ',Terulet:5:1,' cm2') ;
        { A '2' jel kódja 253. Mivel ilyen billentyű nincs a klaviatúrán, ezt úgy írhatjuk be,
          hogy az <ALT> billentyű nyomva tartása mellett a numerikus billentyűzeten beütjük
          a jel kódját. }
    End
  Else
      { Nem: }
    Write('Ebbôl a három adatból nem szerkeszthetô háromszög!');
  ReadLn ;
End.
```

3. feladat

Olvassunk be egy karaktert! Írjuk ki az ASCII kódját, a következő és az előző karaktert az ASCII táblában; valamint azt, hogy a beolvasott karakter nagybetű–e, vagy nem!

```
Program Alap3 ;

Var
    { Karakter típusú változó, értékei az ASCII táblában felsorolt karakterek lehetnek: }
  Kar : Char ;
```

```
  { FŐPROGRAM: }
Begin
    { A karakter beolvasása: }
  Write('Kérek egy karaktert: ') ;
  ReadLn(Kar) ;

    { A kért információk kiírása: }
  WriteLn('A(z) ',Kar,' ASCII kódja : ',Ord(Kar)) ;
  WriteLn('Elôzô karakter     : ',Pred(Kar)) ;
  WriteLn('Következô karakter : ',Succ(Kar)) ;

  If (Kar >= 'A') And (Kar <= 'Z')
  Then
      { A karakter 'A' és 'Z' között van: }
    Write('Nagybetû')
  Else
      { Egyébként: }
    Write('Nem nagybetû') ;
  ReadLn ;
End.
```

4. feladat

Olvassunk be egy fizetést! A fizetés nagyságától függően írjuk ki, hogy az alacsony, átlagos, illetve magas!

```
Program Alap4 ;

  { A konstansként definiált értékeket a fordító a program kódjába teszi, ezért azok a program
    futása során nem változtathatók. }
Const
  Fizeteshatar1 = 15000 ;
  Fizeteshatar2 = 40000 ;

Var
    { 4 bájton tárolt egész típusú változó. A legnagyobb érték, amit a változó felvehet, a
      MaxLongInt előre definiált konstans (lehet negatív is): }
  Fizetes : LongInt ;

  { FŐPROGRAM: }
Begin
    { A fizetés beolvasása: }
  Write('Kérek egy fizetést: ') ;
  ReadLn(Fizetes) ;
```

```
  If Fizetes >= 0
  Then
    Begin
      If Fizetes < FizetesHatar1
      Then
        Write('Alacsony')
      Else
        If Fizetes < FizetesHatar2
        Then
          Write('Átlagos')
        Else
          Write('Magas') ;
      WriteLn(' fizetés') ;
    End
  Else
    WriteLn('Ilyen azért (még) nincsen!') ;
  ReadLn ;
End.
```

5. feladat

Olvassunk be egy karaktert, majd írjuk ki, hogy az nagybetű, kisbetű, szám, speciális karakter vagy egyéb!

```
Program Alap5 ;

Var
  Karakter : Char ;

  { FŐPROGRAM: }
Begin
    { A karakter beolvasása: }
  Write('Kérek egy karaktert: ') ;
  ReadLn(Karakter) ;
    { Válogatás a tevékenységek között a karakter értékétől függően: }
  Case Karakter Of
    'A'..'Z'   : Write('Nagybetû') ;
    'a'..'z'   : Write('Kisbetû') ;
    '0'..'9'   : Write('Szám') ;
    ' '..'/',
    ':'..'@',
    '['..#96   : Write('Speciális karakter') ;
    Else         Write('Egyéb') ;
  End ;
  ReadLn ;
End.
```

6. feladat

Olvassunk be pozitív egész számokat 0 végjelig! Írjuk ki a számok számát, a számok átlagát, a beolvasott legkisebb és legnagyobb számot!

```
Program Alap6 ;

Const
  Vegjel = 0 ;
    { A típussal rendelkező konstans változóként viselkedik, a megadott érték a változó kezdeti
    értéke lesz: }
  SzamokSzama    : Byte = 0 ;
  SzamokOsszege : LongInt = 0 ;

Var
    { 2 bájton tárolt egész típusú változók, értékeik 0 és $FFFF közé eshetnek: }
  Szam,
  MinSzam,
  MaxSzam : Word ;

  { FŐPROGRAM: }
Begin
    { Első szám beolvasása, kezdeti értékek megadása: }
  WriteLn('Üssön be számokat ''',Vegjel,''' végjelig: ') ;
  ReadLn(Szam) ;

  MinSzam := Szam ;
  MaxSzam := Szam ;

    { A ciklust mindaddig végrehajtjuk, amíg nem nullát ütnek: }
  While Szam <> Vegjel Do
      { A szám nem nulla, tehát feldolgozható: }
    Begin
        { Gyűjtjük a számok darabszámát és összegét: }
      Inc(SzamokSzama) ;
      Inc(SzamokOsszege,Szam) ;

        { Ha a szám az eddigi minimumnál kisebb, legyen az a minimum: }
      If Szam < MinSzam Then
        MinSzam := Szam ;
        { Ha a szám az eddigi maximumnál nagyobb, legyen az a maximum: }
      If Szam > MaxSzam Then
        MaxSzam := Szam ;
```

```
        { Következő szám beolvasása: }
      ReadLn(Szam) ;
    End ;

    { Ha a beolvasott számok száma nulla, nem lehet a kiértékelést elvégezni: }
  If SzamokSzama = 0 Then
    Begin
      Write('Egy számot sem írtak be!') ;
      ReadLn ;
        { A program futásának azonnali leállítása: }
      Halt ;
    End ;

    { Az eredmény kiírása.
    Ha a kiírandó kifejezés után egy szám áll kettősponttal elválasztva, akkor a kifejezés jobbra
    igazítva olyan hosszon íródik ki. Ha a kifejezés ennél hosszabb, akkor a teljes érték kiíródik
    csonkulás nélkül. Ha a kifejezés numerikus, újabb kettőspont után a tizedesek száma is
    megadható: }
  WriteLn('A számok száma : ',SzamokSzama:3) ;
  WriteLn('A számok átlaga: ',SzamokOsszege / SzamokSzama:5:1) ;
  WriteLn('A legkisebb  beolvasott szám: ',MinSzam:4) ;
  WriteLn('A legnagyobb beolvasott szám: ',MaxSzam:4) ;
  ReadLn ;
End.
```

7. feladat

Olvassunk be pozitív egész számokat $FFFF végjelig, és állapítsuk meg a számsorozat rendezettségét – egyenlő, növekvő, csökkenő vagy rendezetlen! A növekvő, illetve csökkenő rendezettségnél az egyenlő számokat is megengedjük. Ha a sorozatról menetközben kiderül, hogy rendezetlen, fejezzük be a bevitelt! (A billentyűzetről hexadecimális számot is be lehet vinni.)

```
Program Alap7 ;

Const
  Vegjel = $FFFF ;

Var
    { Felsorolt típusú változó – a lehetséges értékeket soroljuk fel. Az értékek azonosítóként
    viselkednek, azoknak egyedieknek kell lenniük: }
  Rendezettseg : (Egyenlo,Novekvo,Csokkeno,Rendezetlen) ;
  Elozo,
  Aktualis     : Word ;
```

```
  { FŐPROGRAM: }
Begin
    { Az első szám beolvasása. A rendezettség kezdetben egyenlő: }
  WriteLn('Kérem a számokat, megnézem, rendezett-e a sorozat:') ;
  ReadLn(Aktualis) ;
  Elozo := Aktualis ;
  Rendezettseg := Egyenlo ;
    { Amíg nem végjelet ütnek, és rendezett a sorozat: }
  While (Aktualis <> Vegjel) And (Rendezettseg <> Rendezetlen) Do
    Begin
      Case Rendezettseg Of
          { Eddig egyenlőek voltak a beolvasott számok: }
        Egyenlo  : Begin
                     If Elozo < Aktualis Then
                         { A sorozat növekvővé vált: }
                       Rendezettseg := Novekvo ;
                     If Elozo > Aktualis Then
                         { A sorozat csökkenővé vált: }
                       Rendezettseg := Csokkeno ;
                   End ;
          { Eddig növekvő volt a sorozat: }
        Novekvo  : If Elozo > Aktualis Then
                         { A sorozat rendezettsége megszűnt: }
                     Rendezettseg := Rendezetlen ;
          { Eddig csökkenő volt a sorozat: }
        Csokkeno : If Elozo < Aktualis Then
                         { A sorozat rendezettsége megszűnt: }
                     Rendezettseg := Rendezetlen ;
      End ;
      If Rendezettseg <> Rendezetlen Then
          { A rendezettség megmaradt, a következő szám beolvasása: }
        Begin
          Elozo := Aktualis ;
          ReadLn(Aktualis) ;
        End ;
    End ;

    { A rendezettség kiírása: }
  Case Rendezettseg Of
    Egyenlo     : Write('Egyenlô') ;
    Novekvo     : Write('Növekvô') ;
    Csokkeno    : Write('Csökkenô') ;
    Rendezetlen : Write('Rendezetlen') ;
  End ;
  ReadLn ;
End.
```

8. feladat

Írjuk ki az 1 köbméternél kisebb térfogatú, 10 cm–enként növekvő sugarú gömbök térfogatait!

```
Program Alap8 ;

Var
  Sugar    : Byte ;
  Terfogat : Real ;

  { FŐPROGRAM: }
Begin
    { Kezdőértékek: }
  Sugar := 0 ;
  Terfogat := 0 ;

    { Térfogat ciklikus kiírása, számítása: }
  Repeat
        { A sugár és a hozzátartozó térfogat kiírása: }
      WriteLn('Sugár= ',Sugar:3,' cm    Térfogat= ',
              Terfogat:8:4,' m3') ;

        { Következő sugár és a hozzátartozó térfogat számítása: }
      Inc(Sugar,10) ;
      Terfogat := 4 / 3 * Sugar * Sugar * Sugar * Pi / 1E6 ;
    { Kilépés, ha a térfogat az 1 köbmétert eléri, vagy annál nagyobb: }
  Until Terfogat >= 1 ;

  ReadLn ;
End.
```

9. feladat

Olvassunk be a képernyő 12. sorának 40. oszlopától kezdve egy számot. A számot csak akkor fogadjuk el, ha teljesülnek a következő feltételek:

- **a szám vagy kisebb, mint 1000, vagy nagyobb, mint 3000;**
- **a szám egész, páros, és héttel nem osztható.**

```
Program Alap9 ;

  { A program alaputasításait a System egység tartalmazza, mely automatikusan programunkhoz
    szerkesztődik. Amennyiben más egység(ek)ben deklarált azonosítókat akarunk használni, a
    megfelelő egység(ek)et hozzá kell szerkeszteni programunkhoz úgy, hogy az(oka)t a Uses
    fenntartott szó után megadjuk. A ClrScr, ClrEol, GotoXY eljárások a Crt egységben vannak: }
Uses
  Crt ;

Var
  Valos    : Real ;
    { 2 bájton tárolt egész típusú változó. A legnagyobb érték, amit a változó felvehet, a MaxInt
      előre definiált konstans (lehet negatív is): }
  Egesz    : Integer ;
  JoSzam,
  JoEgesz : Boolean ;

  { FŐPROGRAM: }
Begin
    { Képernyő törlése: }
  ClrScr ;
    { Pozícionálás a 12. sor 24. karakterhelyére (a képernyőnek általában 25 sora és 80 oszlopa
      van, (1,1) a bal felső sarok): }
  GotoXY(24,12) ;
    { Szöveg kiírása a kurzor pozíciójától: }
  Write('Kérem a számot:') ;
  GotoXY(10,14) ;
  Write('< 1000 vagy > 3000 páros egész, nem osztható 7-el!') ;

    { A szám ciklikus olvasása, kiértékelése: }
  Repeat
        { Pozícionálás a 12. sor 40. pozíciójára: }
      GotoXY(40,12) ;
        { Sor törlése a kurzor helyétől végig, a kurzor marad: }
      ClrEol ;
        { Szám beolvasása a 12. sor 40. pozíciójától: }
      ReadLn(Valos) ;
        { JoSzam True lesz, ha a szám egész: }
      JoSzam := Frac(Valos) = 0.0 ;
        { A valós szám egész részét betesszük egy egész változóba: }
      Egesz := Trunc(Valos) ;
        { JoEgesz True lesz, ha a szám kisebb, mint 1000 vagy nagyobb, mint 3000, nem
          páratlan, és héttel nem osztható: }
      JoEgesz := ((Egesz < 1000) Or (Egesz > 3000)) And
                 Not Odd(Egesz) And (Egesz Mod 7 <> 0) ;
```

```
      JoSzam := JoSzam And JoEgesz ;
    { Kilépés, ha jó számot ütöttek be: }
  Until JoSzam ;

  GotoXY(30,24) ;
  Write('Brávó!') ;

  ReadLn ;
End.
```

10. feladat

Írjuk ki az angol ABC összes nagybetűjét növekvő, majd csökkenő sorrendben!

```
Program Alap10 ;

Var
  C : Char ;

  { FŐPROGRAM: }
Begin
    { Betűk kiírása növekvő sorrendben: }
  For C := 'A' To 'Z' Do
    Write(C) ;
  WriteLn ;
    { Betűk kiírása csökkenő sorrendben: }
  For C := 'Z' DownTo 'A' Do
    Write(C) ;
  WriteLn ;
    { Várakozás <ENTER> leütéséig: }
  ReadLn ;
End.
```

11. feladat

Olvassunk be egy dátumot: év, hó, nap. Írjuk ki, hogy ez a dátum az év hányadik napja!

```
Program Alap11 ;

  { Érték– és indexhatár ellenőrzést bekapcsoló fordítási direktíva: }
{$R+}
```

```
Var
    { Az intervallum típusú változónak általában csak dokumentációs jelentősége van, de ha
      bekapcsoltuk az érték– és indexhatár ellenőrzést ($R+), akkor ezek a változók csak a
      megadott intervallumba eső értékeket vehetik fel. }
  AktHo  : 1..12 ;
  Ho     : 1..11 ;
  AktNap : 1..31 ;

  AktEv,
  Nap    : Word ;

  { FŐPROGRAM: }
Begin
    { Adatok bekérése ellenőrzés nélkül: }
  Write('Év  = ') ;
  ReadLn(AktEv) ;
  Write('Hó  = ') ;
  ReadLn(AktHo) ;
  Write('Nap = ') ;
  ReadLn(AktNap) ;

    { Az aktuális hónapot megelőző hónapok napjainak összeadása: }
  Nap := 0 ;

    { Ha a FOR ciklus végértéke (TO esetén) kisebb, mint a kezdőérték, a ciklus egyszer sem
      kerül végrehajtásra: }
  For Ho := 1 To AktHo - 1 Do
    Case Ho Of
        { 30 napos hónapok: }
      4,6,9,11   : Inc(Nap,30) ;
        { Február: }
      2          : If (AktEv Mod 4 = 0) And
                   (AktEv Mod 100 <> 0) Or (AktEv Mod 400 = 0)
                   Then
                         { Szökőév: }
                     Inc(Nap,29)
                   Else
                     Inc(Nap,28)
        { 31 napos hónapok: }
      Else       Inc(Nap,31) ;
    End ;
    { Aktuális hónap napjának hozzáadása: }
  Inc(Nap,Aktnap) ;
  WriteLn('Ez a dátum az év ',Nap,'. napja') ;
  ReadLn ;
End.
```

12. feladat

Rajzoljunk adott méretű 'X'–eket a képernyő közepére! A méreteket a rajzolás előtt olvassuk be, méret = 0 –ra fejeződjék be a program!

```
Program Alap12 ;

Uses
  Crt ;

Const
    { Ilyen karakterekből rajzoljuk ki az 'X'–et: }
  Kar = '@' ;

Var
  X,
  Y,
  Meret,
  I      : Byte ;

  { FŐPROGRAM: }
Begin
  ClrScr ;
    { Egy karakterláncban a közvetlenül egymást követő két aposztróf egy aposztrófot jelent: }
  Write('Az ''X'' mérete: ') ;
    { Az első méret beolvasása: }
  ReadLn(Meret) ;
    { Amíg nem nullát írnak be méretre: }
  While Meret <> 0 Do
    Begin
      If Meret > 25
      Then
          { A méret nagyobb, mint a képernyő sorainak száma: }
        Write('Nem fér ki a képernyőre!')
      Else
          { X kirajzolása: }
        Begin
          X := (80 - Meret) Div 2 + 1 ;
          Y := (25 - Meret) Div 2 + 1 ;
          For I := 0 To Meret - 1 Do
            Begin
              GotoXY(X + I,Y + I) ;
              Write(Kar) ;
              GotoXY(X + Meret - 1 - I,Y + I) ;
              Write(Kar) ;
            End ;
```

```
        End ;
      ReadLn ;
        { Következő méret beolvasása: }
      ClrScr ;
      Write('Az ''X'' mérete: ') ;
      ReadLn(Meret) ;
    End ;
End.
```

13. feladat

Nyomogassuk a billentyűket! A program minden egyes lenyomott billentyű kódját – ha két kódja van, mindkettőt – írja ki a képernyőre! Ha a lenyomott billentyűhöz a képernyőn megjeleníthető karakter tartozik, azt is írjuk ki! Az <ESC> gomb lenyomására a program fejeződjék be!

```
Program Alap13 ;

Uses
  Crt ;

Const
    { Egy karaktert az ASCII kódjával is megadhatunk, ha az ASCII kód elé egy '#' jelet teszünk.
    A leggyakoribb vezérlőkaraktereknek szabvány nevük van (a Turbo Pascal nem deklarálja
    konstansként őket): }
  NUL = #0 ;        { Null – semmi }
  BEL = #7 ;        { Bell – csengő }
  BS  = #8 ;        { BackSpace – visszatörlés }
  LF  = #10 ;       { Line Feed – soremelés }
  CR  = #13 ;       { Carriage Return – kocsivissza }
  ESC = #27 ;       { Escape – menekülés, kilépés }
  SPC = #$20 ;      { Space – szóköz (kódját hexadecimálisan adtuk meg) }
Var
  Ch : Char ;

  { FŐPROGRAM: }
Begin
    { Fejléc kiírása: }
  ClrScr ;
  WriteLn('Karakter  1. kód   2. kód') ;
```

```
    { Olvasás a billentyűzetről. Ha van a billentyűzet pufferében karakter, akkor egyet kivesz
      abból. Ha a puffer üres, akkor a program vár egy billentyű leütésére. Mivel több, mint 256
      különböző billentyűkombináció létezik, bizonyos billentyűk leütésekor két kód kerül a
      pufferbe. Kettős billentyűkód esetén az első kód #0. }
  Ch := ReadKey ;

    { Amíg nem <ESC> billentyűt ütnek: }
  While Ch <> ESC Do
    Begin

        { A képernyőn megjeleníthető billentyűt ütöttek? Bizonyos karaktereknek vezérlő
          szerepük van, a képernyőn nem jelennek meg karakter formájában. }
      If Not (Ch In [NUL,BEL,BS,LF,CR])
      Then

          { Megjeleníthető, megjelenítjük: }
        Write(Ch:4)
      Else

          { Vezérlőkarakter esetén szóközöket írunk ki: }
        Write(SPC:4) ;

        { A karakter első kódjának kiírása: }
      Write(Ord(Ch):10) ;


        { Kettős kódot adó billentyűt ütöttek le: }
      If (Ch = NUL) And KeyPressed Then
        Begin

            { Beolvassuk a második kódot: }
          Ch := ReadKey ;

            { A karakter második kódjának kiírása: }
          Write(Ord(Ch):8) ;
        End ;
      WriteLn ;


        { Következő billentyű olvasása: }
      Ch := ReadKey ;
    End ;
End.
```

14. feladat

Tegyünk a képernyő közepére egy jelet, majd a fel, le, balra, jobbra nyilak segítségével mozgassuk azt a képernyőn! A jelet a képernyőről ne engedjük kimenni, s az kezdetben húzzon maga után vonalat! Az <INS> gomb hatására, ha eddig volt vonalhúzás, ne legyen, ha nem volt, legyen! <ENTER>–re a program fejeződjön be!

```
Program Alap14 ;

Uses
  Crt ;

Const
    { A kontroll (harminckettőnél kisebb kódú, vezérlő) karakterek úgy adhatók meg, hogy
    közvetlenül az illető kontroll karaktert azonosító nagybetű elé egy '^' jelet írunk: }
  Jel   = ^B ;
  Ures = ' ' ;

Var
  Ch        : Char ;
  X,
  Y         : Byte ;
  VonalHuz : Boolean ;

  { FŐPROGRAM: }
Begin
    { Kezdeti értékek beállítása: a jel a 12. sor 40. oszlopába kerül, vonalhúzás legyen: }
  ClrScr ;
  X := 40 ;
  Y := 12 ;
  VonalHuz := True ;
  GotoXY(X,Y) ;
  Write(Jel) ;
    { Visszapozícionálás a jel alá, hogy ott villogjon a kurzor: }
  GotoXY(X,Y) ;

    { Olvasás a billentyűzetről: }
  Ch := ReadKey ;
    { <ENTER>–re a program befejeződik: }
  While Ch <> #13 Do
    Begin
        { A kurzorvezérlő és az Ins billentyűk kettős kóddal rendelkeznek, csak azokkal
        foglalkozunk: }
      If Ch = #0 Then
        Begin
            { Beolvassuk a kettős billentyű második kódját: }
          Ch := ReadKey ;
            { Ha nincs vonalhúzás, előző jel törlése: }
          If Not VonalHuz Then
            Begin
              GotoXY(X,Y) ;
              Write(Ures) ;
            End ;
```

```
                { A jobb alsó sarokba nem engedhetjük a jelet, mert akkor görgetés (scroll)
                  történne, vagyis a képernyő összes sora eggyel feljebb kerülne! }
            Case Ch Of
                  { Balra nyíl: }
              #75 : If X > 1 Then
                      Dec(X) ;
                  { Jobbra nyíl: }
              #77 : If (Y < 25) And (X < 80) Or (X < 79) Then
                      Inc(X) ;
                  { Fel nyíl: }
              #72 : If Y > 1 Then
                      Dec(Y) ;
                  { Le nyíl: }
              #80 : If (X < 80) And (Y < 25) Or (Y < 24) Then
                      Inc(Y) ;
                  { Ins billentyű: }
              #82 : VonalHuz := Not VonalHuz ;
            End ;
                { Jel kiírása az új pozícióra: }
            GotoXY(X,Y) ;
            Write(Jel) ;
            GotoXY(X,Y) ;
          End ;

          { Következő billentyű olvasása: }
        Ch := ReadKey ;
      End ;
End.
```

15. feladat

Olvassunk be 0 és 20 között egy számot! Menüből választhatóan írjuk ki 0–tól eddig a számig n négyzetét, gyökét, 2–nek n. hatványát!

```
Program Alap15 ;

Uses
  Crt ;

Var
  Valasz : Char ;
  Meddig : Byte ;
```

```
  { A Menukep eljárás kiírja a menü szövegét: }
Procedure Menukep ;
  Begin
    ClrScr ;
    GotoXY(30,10) ;
    Write('n².............1') ;        { A ² karakter kódja 253 }
    GotoXY(30,11) ;
    Write('√n.............2') ;        { A √ karakter kódja 251 }
    GotoXY(30,12) ;
    Write('2ⁿ.............3') ;        { Az ⁿ karakter kódja 252 }
    GotoXY(30,13) ;
    Write('Vége...........4') ;
  End ;

  { A Negyzetszamok eljárás a számok négyzeteit írja ki. N az eljárás lokális változója, csak az
    eljáráson belül lehet rá hivatkozni. Az eljárás hívásával életre kel, majd befejeztével megszűnik
    létezni: }
Procedure Negyzetszamok ;
  Var
    N : Byte ;
  Begin
    ClrScr ;
    WriteLn('  n         n²') ;
    WriteLn('-------------') ;
    For N := 0 To Meddig Do
      WriteLn(N:3,Sqr(N):10) ;
    ReadLn ;
  End ;

  { A Gyokok eljárás a számok gyökeit írja ki: }
Procedure Gyokok ;
  Var
   N : Byte ;
  Begin
    ClrScr ;
    WriteLn('  n         √n') ;
    WriteLn('-------------') ;
    For N := 0 To Meddig Do
      WriteLn(N:3,Sqrt(N):10:1) ;
    ReadLn ;
  End ;

  { A Kettohatvanyai eljárás 2 hatványait írja ki: }
Procedure Kettohatvanyai ;
  Var
    N : Byte ;
  Begin
    ClrScr ;
    WriteLn('  n         2ⁿ') ;
    WriteLn('-------------') ;
```

```
    For N := 0 To Meddig Do
      WriteLn(N:3,Exp(Ln(2) * N):10:1) ;
    ReadLn ;
  End ;

  { FŐPROGRAM: }
Begin
    { Egy 0 és 20 közötti szám beolvasása: }
  Repeat
      ClrScr ;
      Write('Meddig [0..20]: ') ;
      ReadLn(Meddig) ;
  Until (Meddig <= 20) ;

    { Az egyes funkciókat akárhányszor lehet kérni: }
  Repeat
        { Menüszöveg kiírása: }
      Menukep ;

        { Választókarakter beolvasása: }
      Valasz := ReadKey ;
        { A választástól függően a funkciók végrehajtása: }
      Case Valasz Of
         '1' : Negyzetszamok ;
         '2' : Gyokok ;
         '3' : Kettohatvanyai ;
      End ;

    { Ha a 4–es funkciót választották, kilépés: }
  Until Valasz = '4' ;
End.
```

16. feladat

Olvassunk be két egész számot, A–t és B–t! Írjuk ki a két szám közti abszolút eltérést; írjuk ki a kettő közé eső páros számokat; állapítsuk meg a két szám sorrendiségét (csökkenő, egyforma vagy növekvő); írjuk ki a két szám közül a kisebbet! Ha rajzolható a képernyőre egy A*B méretű tömör téglalap, akkor rajzoljuk meg!

```
Program Alap16 ;

Uses
  Crt ;
```

```
Type
  Sorrendiseg = (Csokkeno,Egyforma,Novekvo) ;

Var
  A,B : Integer ;

  { Az A_Olvas eljárás beolvas a képernyőről az A változóba egy számot: }
Procedure A_Olvas ;
  Begin
    GotoXY(20,1) ;
    Write('A= ') ;
    ReadLn(A) ;
  End ;

  { A B_Olvas eljárás beolvas a képernyőről a B változóba egy számot: }
Procedure B_Olvas ;
  Begin
    GotoXY(50,1) ;
    Write('B= ') ;
    ReadLn(B) ;
  End ;

  { A ElteresKiir eljárás kiírja a paraméterként kapott két szám abszolút eltérését: }
Procedure ElteresKiir(Szam1, Szam2: Integer) ;
  Begin
    WriteLn('A két szám közti eltérés: ',Abs(Szam1 - Szam2)) ;
  End ;

  { A PárosKiir eljárás kiírja a paraméterként kapott két szám közötti összes páros számot. Tol,
    Ig és I az eljárás lokális változói. Tol és Ig ezenkívül paraméterek, vagyis az eljárás hívásakor
    aktuális értéket kapnak, az eljárás aktuálisan ezekkel az induló értékekkel hajtódik végre. }
Procedure ParosKiir(Tol, Ig: Integer) ;
  Var
    I : Integer ;
  Begin
    Write(Tol,' és ',Ig,' közé esô páros számok: ') ;
    If Tol <= Ig
    Then
        { A páros számok növekednek.
          Itt a szintaxis nem követelné meg a Begin–End használatát, de ha elhagynánk, a
          következő Else nem a Tol <= Ig feltételhez, hanem a Not Odd(I) –hez tartozna! }
      Begin
        For I := Tol To Ig Do
          If Not Odd(I) Then
            Write(I:3) ;
      End
```

```
    Else
        { A páros számok csökkennek: }
      For I := Tol DownTo Ig Do
        If Not Odd(I) Then
          Write(I:3) ;
    WriteLn ;
  End ;

  { A Sorrend függvény N és M értékétől függően felveszi a sorrendiség egyik értékét : }
Function Sorrend(N, M: Integer) : Sorrendiseg ;
  Begin
    If N = M
    Then
      Sorrend := Egyforma
    Else
      If N < M
      Then
        Sorrend := Novekvo
      Else
        Sorrend := Csokkeno ;
  End ;

  { A Minimum függvény a paraméterként kapott két egész szám közül kiválasztja a kisebbet, ez
    lesz a függvény értéke: }
Function Minimum(X, Y: Integer): Integer ;
  Begin
    If X < Y
    Then
      Minimum := X
    Else
      Minimum := Y ;
  End ;

  { A Joteglalap függvény True értékű, ha a két paraméter által meghatározott téglalap ráfér a
    képernyőre: }
Function Joteglalap(Szelesseg, Magassag: Integer): Boolean ;
  Begin
    Joteglalap := (Szelesseg <= 80) And (Magassag <= 25) And
                  Not ((Szelesseg = 80) And (Magassag = 25)) ;
  End ;

  { A Teglarajz eljárás egy tömör téglalapot rajzol a képernyő közepére. A téglalap két oldala
    az eljárás két paramétere: }
Procedure Teglarajz(Szelesseg, Magassag: Integer) ;
  Var
    Oszlop,
    Sor,
    KezdX,
    KezdY : Byte ;
```

```
  Begin
    ClrScr ;
    KezdX := (80 - Szelesseg) Div 2 + 1 ;
    KezdY := (25 - Magassag) Div 2 + 1 ;
    For Sor := KezdY To KezdY + Magassag - 1 Do
      Begin
        GotoXY(KezdX,Sor) ;
        For Oszlop := 1 To Szelesseg Do
          Write(#219) ;
      End ;
  End ;

  { FŐPROGRAM: }
Begin
  ClrScr ;
    { Adatok beolvasása: }
  A_Olvas ;
  B_Olvas ;

    { A két szám közti eltérés kiírása: }
  ElteresKiir(A,B) ;

    { Páros számok kiírása: }
  ParosKiir(A,B) ;

    { A két szám egymáshoz való viszonyának kiírása: }
  Case Sorrend(A,B) Of
    Csokkeno : WriteLn('Csökkenô sorrend') ;
    Egyforma : WriteLn('Egyformák') ;
    Novekvo  : WriteLn('Növekvô sorrend') ;
  End ;

    { A kisebb érték kiírása: }
  WriteLn('A kisebb szám: ',Minimum(A,B)) ;

    { Téglalap kirajzolása, ha lehet: }
  If Joteglalap(A,B)
  Then
    Begin
      WriteLn('Rajzolható téglalap') ;
      ReadLn ;
      Teglarajz(A,B) ;
    End
  Else
    WriteLn('Nem rajzolható téglalap') ;

  ReadLn ;
End.
```

17. feladat

Olvassunk be sorra egész számokat (a szám lehet negatív is), és írjuk ki azok kettes számrendszerbeli alakját!

```
Program Alap17 ;

Var
  Szam : Integer ;

  { Az eljárás kiírja a paraméterként kapott szám bitképét, azaz kettes számrendszerbeli alakját: }
Procedure BitKep(Bitek: Integer) ;
  Var
    Bit : Word ;
  Begin
      { Bit változó (balról számított) első bitje 1-es, a többi (tizenöt) 0: }
    Bit := $8000 ;
      { Bitek paraméter összes bitjét egyenként levizsgáljuk: }
    While Bit <> 0 Do
      Begin
        If Bitek And Bit = 0
        Then
            { A bit értéke nulla: }
          Write('0')
        Else
            { A bit értéke egy: }
          Write('1') ;
          { Bit változó minden bitjét eggyel jobbra toljuk, a legfelső (bal szélső) bit helyére
            nulla kerül): }
        Bit := Bit Shr 1 ;
          { A bájtokat elválasztjuk egymástól: }
        If Bit = $80 Then
          Write(' ') ;
      End ;
    WriteLn ;
  End ;

  { FŐPROGRAM: }
Begin
    { Szám beolvasása (ne feledjük, hogy hexadecimális érték is bevihető, ha $ jellel indítjuk a
      hexadecimális számjegyeket!): }
  Write('A binárisan kiírandó szám: ') ;
  ReadLn(Szam) ;
```

```
    { Amíg nem nullát írnak be: }
  While Szam <> 0 Do
    Begin
        { A szám kiírása kettes számrendszerben: }
      BitKep(Szam) ;
        { Következő szám beolvasása: }
      Write('A binárisan kiírandó szám: ') ;
      ReadLn(Szam) ;
    End ;
End.
```

18. feladat

Írjunk programot, mely a képernyő mindig nyolccal arrébb lévő pozíciójáról (szükség esetén új sorból) olvas be számokat! A bevitt számot csak akkor fogadjuk el, ha az ténylegesen szám, egész, és az előző számtól való eltérés 20% –nál nem nagyobb!

```
Program Alap18 ;

Uses
  Crt ;

Var
  Sor,
  Oszlop       : Byte ;
  Szam,
  ElozoSzam    : Real ;
  Egesz,
  Nagyelteres : Boolean ;

  { A Hibahang eljárás hangjelzést ad: }
Procedure Hibahang ;
  Begin
    Write(Chr(7)) ;
  End ;

  { A Szamolv eljárásnak három paramétere van: az első kettő megadja a képernyőpozíciót,
  ahonnan beolvasunk, a harmadik paraméter egy változó, melybe a beolvasott szám kerül.
  Változó paraméter esetén az eljárás a változó címét kapja meg, míg értékparaméter esetén (X
  és Y) a változó vagy kifejezés értéke kerül átadásra. Változó paraméter esetén az eljárás az
  illető változó értékét megváltoztathatja, míg érték paraméter esetén csak felhasználja a kapott
  értéket. }
Procedure Szamolv(X, Y: Byte ; Var Valos: Real) ;
  Var
    Ok : Boolean ;
  Begin
```

```
    { Kikapcsoljuk a beviteli és kiviteli (B/K, Input/Output) műveletek ellenőrzését, hogy a
      program olvasásnál ne álljon le B/K futási hibával (pl. ha olyan karaktert olvasunk be,
      melyet nem lehet számnak értelmezni). Ilyenkor az IOResult függvénnyel feltétlenül le kell
      kérdeznünk a művelet eredményességét, különben hiba esetén a további B/K műveletek
      nem hajtódnak végre: }
    {$I-}
    Repeat
        GotoXY(X,Y) ;
        ClrEol ;
        ReadLn(Valos) ;
            { Az IOResult függvény egy hibakódot ad vissza (ha 0, sikerült a beolvasás). A
              függvény hívása után a hibakód törlődik, így azt még egyszer nem lehet lekérdezni.
              Ezért a hibakódot egy változóba mentjük el: }
        Ok := IOResult = 0 ;
            { Ha hiba történt a bevitel során: }
        If Not Ok Then
          Hibahang ;
      { Kilépés, ha nincs hiba: }
    Until Ok ;
    {$I+}
  End ;

  { FŐPROGRAM: }
Begin
  ClrScr ;
  WriteLn('Kérem a számokat: ') ;
  Sor := 3 ;
  Oszlop := 3 ;

    { Első szám beolvasása. Az előző szám kezdetben legyen ugyanaz: }
  Szamolv(Oszlop,Sor,Szam) ;
  ElozoSzam := Szam ;
    { Amíg nem nullát írnak be: }
  While Szam <> 0 Do
    Begin
      Egesz := Int(Szam) = Szam ;
      NagyElteres := Abs(Szam - ElozoSzam) > Abs(ElozoSzam) * 0.2;
      If Egesz And Not Nagyelteres
      Then
          { A szám egész és az eltérés nem nagyobb 20%-nál: }
        Begin
            { Oszlop növelése, szükség esetén sorváltás: }
          Inc(Oszlop,8) ;
          If Oszlop > 75 Then
            Begin
              Inc(Sor) ;
```

```
                    { Ha betelt a lap, új lapot kezdünk. Az utolsó sorból nem olvasunk, mert
                      a ReadLn miatt görgetés történne: }
                  If Sor > 24 Then
                    Begin
                      ClrScr ;
                      Sor := 1 ;
                    End ;
                  Oszlop := 3 ;
                End ;
              { Előző szám megjegyzése: }
            ElozoSzam := Szam ;
          End
        Else
            { A szám nem egész, vagy az eltérés nagyobb, mint 20%: }
          Hibahang ;

          { Következő szám beolvasása: }
        Szamolv(Oszlop,Sor,Szam) ;
      End ;
End.
```

19. feladat

A képernyő alapszíne legyen kék! A képernyő közepén egy kis fehér keretű világosszürke téglalapban feketével írjunk ki két véletlen számot, az egyik 0 és 20, a másik 5 és 15 között legyen! Kérjük be a két szám szorzatát! Ha jó a válasz, írjuk ki zölddel, hogy 'Jó', egyébként pirossal, hogy 'Rossz'! A fentieket ismételjük meg tízszer, a végén írjuk ki a jó válaszok számát!

```
Program Alap19 ;

Uses
  Crt ;

Var
  I             : Byte ;
  A,
  B,
  Eredmeny,
  Jovalasz      : Word ;
  Jo            : Boolean ;
  EredetiSzinek : Byte ;
```

```
{ FŐPROGRAM: }
Begin

  { A program el fogja állítani a képernyő színeit. Megjegyezzük a program indításakor
    aktuális színeket, hogy a végén visszaállíthassuk azokat. A Crt egység TextAttr változója
    tartalmazza az aktuális kiíró attribútumokat (jellemzőket): }
  EredetiSzinek := TextAttr ;

  { Háttérszín és karakterszín beállítása. Az ezután kiírt karakterek fehérek lesznek kék
    háttérrel. Az eljárások TextAttr megfelelő bitjeit állítják be (a Crt egység konstansait
    használjuk): }
  TextBackground(Blue) ;
  TextColor(White) ;

  { Aktuális ablak törlése az aktuális háttérszínnel (a program indításakor az aktuális ablak
    az egész képernyő): }
  ClrScr ;

  { Keret készítése. Pozícionálás a képernyő 11–17. sora 30. oszlopára, és a félgrafikus
    karakterek kiírása: }
  GotoXY(30,11) ; Write('┌──────────┐') ;
  GotoXY(30,12) ; Write('│          │') ;
  GotoXY(30,13) ; Write('│          │') ;
  GotoXY(30,14) ; Write('│          │') ;
  GotoXY(30,15) ; Write('│          │') ;
  GotoXY(30,16) ; Write('│          │') ;
  GotoXY(30,17) ; Write('└──────────┘') ;

  { Az új ablak kereten belülre állítása. (31,12) az ablak bal felső sarka, (50,16) az ablak jobb
    alsó sarka. Az új ablak mérete: 20x5: }
  Window(31,12,50,16) ;

  { Az ablakba írás színeinek beállítása (világosszürke háttér, fekete karakterek), ablaktörlés: }
  TextBackground(LightGray) ;
  TextColor(Black) ;
  ClrScr ;

  { Szövegek kiírása, a pozícionálások ablakrelatívak: }
  GotoXY( 4,2) ; Write('A= ') ;
  GotoXY(14,2) ; Write('B= ') ;
  GotoXY( 8,4) ; Write('A*B= ') ;

  { 10–szer két véletlen szám kiírása, azok szorzatának bekérése. A jó válaszokat számoljuk: }
  Jovalasz := 0 ;

  { Véletlenszám generátor inicializálása: }
  Randomize ;
  For I := 1 To 10 Do
    Begin

      { A véletlenszámokat feketével írjuk: }
      TextColor(Black) ;
```

```
      { Egy 0 és 20 közötti véletlen szám létrehozása, kiírása: }
    A := Random(21) ;
    GotoXY(6,2) ;
    Write(A:2) ;

      { Egy 5 és 15 közötti véletlen szám létrehozása, kiírása: }
    B := Random(11) + 5 ;
    GotoXY(16,2) ;
    Write(B:2) ;

      { A szorzat bekérése, az előző eredményt előbb letöröljük: }
    GotoXY(13,4) ;
    ClrEol ;
    ReadLn(Eredmeny) ;

      { Ha jó a beütött eredmény, Jo változó értéke True lesz: }
    Jo := Eredmeny = A * B ;

      { A 'Jo' vagy a 'Rossz' szöveget az 5. sor 10. oszlopába írjuk: }
    GotoXY(10,5) ;
    If Jo
    Then
      Begin
        TextColor(Green) ;
        Write('Jó    ') ;
        Inc(Jovalasz) ;
      End
    Else
      Begin
        TextColor(Red) ;
        Write('Rossz') ;
      End ;
  End ;
  { Teljes ablak visszaállítása, jó válaszok számának kiírása a kis ablak alá: }
Window(1,1,80,25) ;
TextColor(White) ;
GotoXY(30,18) ;
Write('A jó válaszok száma:',Jovalasz:2) ;


  { Várunk az <ESC> billentyű leütésére: }
Repeat
Until ReadKey = #27 ;
  { A program futása előtti képernyő színeinek visszaállítása, képernyőtörlés: }
TextAttr := EredetiSzinek ;
ClrScr ;
End.
```

20. feladat

Írjunk programot, mely kiírja a két bekért szám közötti számok szorzatát és összegét!

```
Program Alap20 ;

  { Mivel a produktum értéke várhatóan nagyon nagy lesz, Extended típust kell használnunk. Ezt
    a típust csak a numerikus társprocesszor tudja kezelni, így az N+ direktívával annak használatát
    engedélyezzük. Hogy 80x87–es társprocesszor nélkül is fusson a program, az esetlegesen
    szükséges emulációt szintén engedélyezzük az E+ direktívával. A verem ellenőrzését bekapcsol-
    juk a rekurzív hívások miatt (túl sok rekurzió esetén a verem betelhet): }
{$N+,E+,S+}

Var
  Mettol,
  Meddig : Byte ;

  { A függvény a szorzatot úgy számítja ki, hogy önmagát hívja (rekurzió) a kapott paraméter
    csökkentett értékével. }
Function Produktum(N : Byte) : Extended ;
  Begin
    If N > Mettol
    Then
      Produktum := N * Produktum(N - 1)
    Else
      Produktum := N ;
  End ;

  { A függvény az összeget úgy számítja ki, hogy önmagát hívja (rekurzió) a kapott paraméter
    csökkentett értékével. }
Function Szumma(N : Byte) : LongInt ;
  Begin
    If N > Mettol
    Then
      Szumma := N + Szumma(N - 1)
    Else
      Szumma := N ;
  End ;

  { FŐPROGRAM: }
Begin
  WriteLn('Számok szorzata és összege') ;
  Write('Mettôl : ') ;
  ReadLn(Mettol) ;
  Write('Meddig : ') ;
  ReadLn(Meddig) ;
  WriteLn ;
```

```
  WriteLn('Produktum: ',Produktum(Meddig):25:0) ;
  WriteLn('Szumma   : ',Szumma(Meddig):25) ;
  ReadLn ;
End.
```

21. feladat

Írjunk programot, mely egytől n–ig a következő sorozat értékét kiszámítja, majd kiírja:

1 * (2 + 3 * (4 + 5 * (... n) ...))

A számítást két egymást hívó függvény segítségével végezzük el!

```
Program Alap21 ;

{$N+,E+,S+}

Var
  Meddig : Byte ;

  { Egyik függvény hívja Masik–at, és viszont. Ez csak úgy oldható meg, hogy Masik fejét előre
    hozzuk, hogy a rá való hivatkozáskor már ismert legyen. Erre a Forward direktíva szolgál. A
    függvény blokkját később, a deklarációs részben bárhol leírhatjuk. }
Function Masik(X : Extended) : Extended ; Forward ;

  { Egyik függvény deklarálása: }
Function Egyik(X : Extended) : Extended ;
  Begin
    If X < Meddig
    Then
      Egyik := X * Masik(X + 1)
    Else
      Egyik := Meddig ;
  End ;

  { A Forward direktívával ellátott rutin blokkjának leírása. Az esetleges paramétereket nem
    szükséges ismét leírni: }
Function Masik ;
  Begin
    If X < Meddig
    Then
      Masik := X + Egyik(X + 1)
    Else
      Masik := Meddig ;
  End ;
```

```
  { FŐPROGRAM: }
Begin
  Write('Meddig számoljunk? ') ;
  ReadLn(Meddig) ;
  WriteLn('Az érték: ',Egyik(1):3:0) ;
  ReadLn ;
End.
```

KARAKTERLÁNCOK

1. feladat

Olvassunk be egy karakterláncot, majd

- **írjuk ki a hosszát;**
- **írjuk ki fordítva;**
- **számoljuk meg, hány szóköz van benne;**
- **az első 'A' betűtől kezdve írjuk ki 10 karakterét;**
- **az első 'jaj' részláncot cseréljük ki 'hajaj'–ra, és innen kezdve írjuk ki az egész karakterláncot;**
- **tegyük a végéhez, hogy ' Hello';**
- **végül vegyük ki belőle a szóközöket!**

```
Program Karlanc1 ;

Var
    { Karakterlánc típusú változó, értéke egy maximum 80 karakter hosszú karakterlánc lehet.
    Beolvasásnál és egyéb karakterlánc műveleteknél, ha ezt a hosszt túllépjük, az eredmény
    csonkulva kerül a vátozóba: }
  Karlanc : String[80] ;

  I,
  Szamol,
  Poz      : Byte ;

  { FŐPROGRAM: }
Begin
    { Karakterlánc beolvasása: }
  Write('A karakterlánc: ') ;
  ReadLn(Karlanc) ;

    { A karakterlánc aktuális hosszának kiírása: }
  WriteLn('A karakterlánc hossza: ',Length(Karlanc)) ;

    { A karakterlánc kiírása fordítva: }
  Write('A karakterlánc fordítva: ') ;
  For I := Length(Karlanc) Downto 1 Do
    Write(Karlanc[I]) ;
  WriteLn ;

    { Szóközök megszámolása a karakterláncban: }
  Szamol := 0 ;
  For I := 1 To Length(Karlanc) Do
    If Karlanc[I] = ' ' Then
      Inc(Szamol) ;
  WriteLn('Szóközök száma: ',Szamol) ;
```

```
    { Az első 'A' betűtől kezdve 10 karakter kiírása. Az első 'A' betű indexe a karakterláncban: }
  Poz := Pos('A',Karlanc) ;
  If Poz <> 0
  Then
    Begin
      Write('Az elsô ''A''-tól 10 karakter: ') ;
        { Ha Poz–tól nincs 10 karakter a láncban, a részlánc rövidebb lesz: }
      WriteLn(Copy(KarLanc,Poz,10)) ;
    End
  Else
    WriteLn('Nincs benne ''A''') ;

    { Ha van a karakterláncban 'jaj' részlánc, kicseréljük 'hajaj'–ra: }
  Poz := Pos('jaj',Karlanc) ;
  If Poz <> 0
  Then
    Begin
      Write('''hajaj''-tól a karakterlánc: ') ;
        { Poz. helytől 'ha' beszúrása a karakterláncba: }
      Insert('ha',Karlanc,Poz) ;
        { Poz. helytől a lánc kiírása: }
      WriteLn(Copy(Karlanc,Poz,Length(Karlanc))) ;
    End
  Else
    WriteLn('Nincs benne ''jaj''') ;

    { A ' Hello' hozzáírása Karlanc végéhez: }
  Karlanc := Karlanc + ' Hello' ;
  WriteLn('A karakterlánc a '' Hello'' hozzáfûzése után: ',
          KarLanc) ;

    { Szóközök kivétele a karakterláncból: }
  Poz := Pos(' ',Karlanc) ;
    { Amíg van szóköz a láncban: }
  While Poz <> 0 Do
    Begin
        { Poz. karakter törlése a láncból: }
      Delete(Karlanc,Poz,1) ;
      Poz := Pos(' ',Karlanc) ;
    End ;
  WriteLn('A karakterlánc a szóközök kivétele után: ',Karlanc) ;

  ReadLn ;
End.
```

2. feladat

Olvassunk be egy szöveget, valamint azt a két karakterláncot, amit és amire ki akarunk cserélni a szövegben! Végezzük el a cserét, az eredményt írjuk ki a képernyőre!

```
Program Karlanc2 ;

Var
  Szoveg,
  Mit,
  Mire       : String[127] ;

    { Ha nem adjuk meg a maximális hosszt, akkor az 255 lesz: }
  UjSzoveg   : String ;

  N,
  MitHossz,
  MireHossz : Byte ;

  { FŐPROGRAM: }
Begin
    { Szövegek bekérése. A beolvasáshoz használt pufferbe maximálisan 127 karakter fér, ezért
      annál hosszabb karakterláncot alaphelyzetben nem lehet beolvasni. }
  Write('A szöveg: ') ;
  ReadLn(Szoveg) ;
  Write('Mit cseréljek?  ') ;
  ReadLn(Mit) ;
  Write('Mire cseréljem? ') ;
  ReadLn(Mire) ;
  UjSzoveg := '' ;
  MitHossz := Length(Mit) ;
  MireHossz := Length(Mire) ;
    { Első cserélendő szöveg kikeresése: }
  N := Pos(Mit,Szoveg) ;
    { Amíg van cserélendő szöveg: }
  While N <> 0 Do
    Begin
        { Új szöveg építése, Mit helyett a Mire szöveggel. Ha az új szöveg hosszabb lenne,
          mint 255 karakter, csonkul: }
      UjSzoveg := UjSzoveg + Copy(Szoveg,1,N - 1) + Mire ;
        { Szöveg elejének törlése (hogy ne találjuk meg ugyanazt mégegyszer): }
      Delete(Szoveg,1,N + MitHossz - 1) ;
        { Következő cserélendő szöveg kikeresése: }
      N := Pos(Mit,Szoveg) ;
    End ;
```

```
  Szoveg := UjSzoveg + Szoveg ;

    { A szöveg csere után: }
  WriteLn('Az új szöveg: ',Szoveg) ;
  ReadLn ;
End.
```

3. feladat

Egy beolvasott karakterláncban keressük meg az első '<<<' és '>>>' közé zárt részláncot!

```
Program Karlanc3 ;

Const
  KezdoJelsor = '<<<' ;
  VegzoJelsor = '>>>' ;

Var
  S,
  Reszlanc : String[127] ;
  Reszkezd,
  Reszveg  : Byte ;

  { Ha kiderül, hogy nincs ilyen részlánc, ide kerül a vezérlés: }
Procedure Nincs ;
  Begin
    Write('Nincs ilyen részlánc a beolvasott karakterláncban') ;
    ReadLn ;
    Halt ;
  End ;

  { FŐPROGRAM }
Begin
  Write('Kérem a karakterláncot: ') ;
  ReadLn(S) ;
    { Első '<<<' kikeresése: }
  Reszkezd := Pos(KezdoJelsor,S) ;
  If Reszkezd = 0 Then
      { Ha nincs ilyen jelsor, üzenet kíséretében program vége: }
    Nincs ;
    { A részlánc a jelsorozat után kezdődik: }
  Inc(Reszkezd,Length(KezdoJelsor)) ;
  Reszlanc := Copy(S,Reszkezd,Length(S)) ;
```

```
    { Első '>>>' kikeresése a részláncból: }
  Reszveg := Pos(VegzoJelsor,Reszlanc) ;

      { Ha nincs ilyen jelsor, üzenet kíséretében program vége: }
  If Reszveg = 0 Then
    Nincs ;


    { A jelsorok közé zárt részlánc tehát: }
  Reszlanc := Copy(Reszlanc,1,Reszveg - 1) ;
  Write('A keresett részlánc: ',Reszlanc) ;
  ReadLn ;
End.
```

4. feladat

Olvassunk be szöveget egy karakterláncba! A szövegben lévő első számot növeljük meg 5–el! Az eredményt 2 tizedesjeggyel írjuk vissza a szövegbe!

```
Program Karlanc4 ;

Var
  Szoveg,
  SzamStr    : String ;
  Szam       : Real ;
  Kod        : Integer ;
  VanSzam    : Boolean ;
  SzamKezd,
  SzamHossz  : Byte ;

  { FŐPROGRAM: }
Begin
    { Szöveg bekérése – üres karakterláncot nem fogadunk el: }
  Repeat
      Write('Kérem a szöveget: ') ;
      ReadLn(Szoveg) ;
  Until Szoveg <> '' ;


    { Keressük az első szám előfordulásának kezdetét: }
  SzamKezd := 0 ;
  Repeat
      Inc(SzamKezd) ;

      { A Val eljárás a szöveget számmá alakítja: }
      Val(Copy(Szoveg,SzamKezd,Length(Szoveg)),Szam,Kod) ;
```

```
        { Ha Kod = 0, akkor a SzamKezd pozíciótól az egész szöveg szám ; ha Kod = 1, akkor
          a SzamKezd pozíciótól nincs szám ; ha Kod > 1, akkor a SzamKezd pozíciótól
          valameddig szám áll. A Val eljárás a szóközt 0–vá alakítja, a kezdő szóközöket
          lenyeli: }
      VanSzam := (Szoveg[SzamKezd] <> ' ') And (Kod <> 1) ;

    { Kilépés, ha van szám a szövegben vagy nincs több lehetőség: }
  Until VanSzam Or (SzamKezd = Length(Szoveg)) ;

    { Vigyázat! Nagy szám esetén az algoritmus nem működik helyesen. A Real típus csak 12
      jegy pontosságú, és ha a szám a Real értéktartományába nem esik bele, akkor azt a Val
      eljárás 0–vá alakítja. }
  If VanSzam
  Then

      { Van szám a szövegben: }
    Begin
      If Kod = 0
      Then

          { SzamKezd–től végig szám áll: }
        SzamHossz := Length(Szoveg) - (SzamKezd - 1)
      Else

          { Kod–1 karaktert sikerült átalakítani: }
        SzamHossz := Kod - 1 ;

        { A szám karakterlánc formája: }
      SzamStr := Copy(Szoveg,SzamKezd,SzamHossz) ;

        { SzamStr–t számmá alakítjuk, hozzáadunk 5–öt, majd visszaalakítjuk: }
      Val(SzamStr,Szam,Kod) ;
      Szam := Szam + 5 ;

        { SzamStr minimálisan 4 hosszú lesz: }
      Str(Szam:4:2,SzamStr) ;

        { Szoveg–ből kitöröljük az eredeti számot, s az újat beszúrjuk: }
      Delete(Szoveg,SzamKezd,SzamHossz) ;
      Insert(SzamStr,Szoveg,SzamKezd) ;
      Write('A szöveg a szám megnövelése után: ') ;
      WriteLn(Szoveg) ;
    End
  Else
    WriteLn('Nincs szám a szövegben') ;
  ReadLn ;
End.
```

5. feladat

Olvassunk be egy dátumot ÉÉ–HH–NN alakban! A dátumot ismételten kérjük, ha az nem helyes (formai hiba, hibás év, hó vagy nap)! Elég az első hibát felderíteni. Bontsuk szét a dátumot 3 részre: Év, Hó, Nap, mindhárom érték numerikus legyen!

```
Program Karlanc5 ;

  { A logikai kifejezéseknél a kiértékelést csak addig kell végezni, amíg egyértelműen ki nem derül,
    hogy az eredmény igaz vagy hamis: }
{$B-}

Uses
  Crt ;

Label
  BeolvasVege ;

Var
  Puffer : Array[1..10] Of Char ;
  Datum  : String[8] ;
  Ev,
  Ho,
  Nap,
  Kod    : Integer ;
  Hibas  : Boolean ;

  { Hibaszöveg kiírása a 24. sor közepére. <ENTER> lenyomása után a szöveg törlődik, a
    program folytatódik: }
Procedure Hiba(Hibaszoveg: String) ;
  Begin
    Hibas := True ;
    GotoXY((80 - Length(Hibaszoveg)) Div 2 + 1,24) ;
    Write(Hibaszoveg) ;
    Repeat
    Until ReadKey = #13 ;
    GotoXY(1,24) ;
    ClrEol ;
  End ;

  { FŐPROGRAM: }
Begin
  ClrScr ;
  GotoXY(25,12) ;
  Write('Dátum (ÉÉ-HH-NN): ') ;
    { A billentyűzet átmeneti pufferének átállítása. Ezután maximálisan 8 karaktert lehet beütni.
      A puffer utolsó két karaktere mindig a CR+LF, ezért a méretet kettővel nagyobbra kell
      venni: }
  SetTextBuf(Input,Puffer,10) ;

    { A dátum ismételt beolvasása, amíg jó dátumot nem ütnek be. A különböző hibákról a
      program üzenetet ad. }
  Repeat
        { Dátum beolvasása: }
      GotoXY(43,12) ;
```

```
    ClrEol ;
    ReadLn(Datum) ;

      { Feltételezzük, hogy jó a dátum. Ha a vizsgálatok közben kiderül, hogy mégsem, Hibas
        változót True–ra állítjuk: }
    Hibas := False ;

    If (Length(Datum) <> 8) Or (Datum[3] <> '-') Or
       (Datum[6] <> '-') Then

        { A dátum nem 8 hosszú, vagy a 3. és 6. karakterek nem '-' elválasztójelek: }
      Begin
        Hiba('Formai hiba!') ;
        Goto BeolvasVege ;
      End ;

      { Az év vizsgálata. Ehhez a dátum első két karakterét számmá alakítjuk: }
    Val(Copy(Datum,1,2),Ev,Kod) ;
    If (Kod <> 0) Or (Ev < 0) Or (Ev > 99) Then

        { Az év nem 0 és 99 közötti szám: }
      Begin
        Hiba('Hibás évszám!') ;
        Goto BeolvasVege ;
      End ;
    Ev := 1900 + Ev ;

      { A hónap vizsgálata. Ehhez a dátum középső két karakterét számmá alakítjuk: }
    Val(Copy(Datum,4,2),Ho,Kod) ;
    If (Kod <> 0) Or (Ho < 1) Or (Ho > 12) Then

        { A hó nem 1 és 12 közötti szám : }
      Begin
        Hiba('Hibás hónap!') ;
        Goto BeolvasVege ;
      End ;

      { A nap vizsgálata. Ehhez a dátum utolsó két karakterét számmá alakítjuk: }
    Val(Copy(Datum,7,2),Nap,Kod) ;
    If (Kod <> 0) Or (Nap < 1) Or (Nap > 31) Then

        { A nap nem 1 és 31 közötti szám: }
      Begin
        Hiba('Hibás nap!') ;
        Goto BeolvasVege ;
      End ;

BeolvasVege:

  { Kilépés, ha jó dátumot ütöttek: }
Until Not Hibas ;
```

```
    { Az eredmény kiírása: }
  GotoXY(25,20) ;
  Write('Év: ',Ev,'  Hó: ',Ho:2,'  Nap: ',Nap:2) ;
  ReadLn ;
End.
```

6. feladat

Olvassunk be egy karakterláncot! A karakterláncban a kapcsos zárójelek közé eső részlánc megjegyzésnek, míg az aposztrófok közé eső részlánc szövegkonstansnak számít. A karakterláncban szereplő megjegyzéseket vegyük ki zárójelekkel együtt, a szövegkonstansokat csupa nagybetűre változtassuk, míg a fennmaradó részt húzzuk szét, azaz minden két karakter közé szúrjunk be egy szóközt! Írjuk ki a karakterláncban szereplő összes megjegyzést, valamint az átalakított karakterláncot!

```
Program Karlanc6 ;

Const
  Megjegyzes_nyit = '{' ;
  Megjegyzes_zar  = '}' ;
  Aposztrof       = '''' ;
  Szokoz          = ' ' ;

Var
  Lanc,
  UjLanc,
  Megjegyzes : String ;
  I          : Byte ;

  { FŐPROGRAM: }
Begin
    { A karakterlánc bekérése, kezdeti értékek beállítása: }
  Write('A karakterlánc: ') ;
  ReadLn(Lanc) ;
  UjLanc := '' ;
  I := 1 ;

    { Amíg I létező karaktert címez: }
  While I <= Length(Lanc) Do
    Begin
      Case Lanc[I] Of
          { Megjegyzés következik: }
        Megjegyzes_nyit : Begin
                            Megjegyzes := '' ;
                            Inc(I) ;
```

```
                                { Megjegyzés zárásáig illetve a szöveg végéig minden
                                  karakter a megjegyzéshez tartozik: }
                                While (Lanc[I] <> Megjegyzes_zar) And
                                      (I <= Length(Lanc)) Do
                                  Begin
                                    Megjegyzes := Megjegyzes +
                                                  Lanc[I] ;
                                    Inc(I) ;
                                  End ;
                                { Megjegyzést UjLanc–hoz nem adtuk hozzá. A meg-
                                  jegyzés kiírása: }
                                WriteLn('Megjegyzés: ',Megjegyzes) ;
                                Inc(I) ;
                              End ;
        { Szövegkonstans következik: }
      Aposztrof         : Begin
                                { UjLanc–hoz hozzáadjuk a szövegkonstans nagybetűs
                                  változatát, aposztrófokkal együtt: }
                                Repeat
                                    UjLanc := UjLanc +
                                              Upcase(Lanc[I]) ;
                                    Inc(I) ;
                                { Amíg a szövegkonstans vagy a lánc végéhez nem
                                  értünk: }
                                Until (Lanc[I] = Aposztrof) Or
                                      (I > Length(Lanc)) ;
                                UjLanc := UjLanc + Aposztrof ;
                                Inc(I) ;
                              End ;
        { Szöveg következik: }
      Else                  Begin
                                { UjLanc–hoz hozzáadjuk a lánc többi karakterét,
                                  széthúzva: }
                                Repeat
                                    UjLanc := UjLanc + Lanc[I] +
                                              Szokoz ;
                                    Inc(I) ;
                                { Amíg nem érünk megjegyzéshez, szövegkonstanshoz
                                  vagy a lánc végéhez: }
                                Until (Lanc[I] = Megjegyzes_nyit) Or
                                      (Lanc[I] = Aposztrof) Or
                                      (I > Length(Lanc)) ;
                              End ;
    End ;
  End ;
 WriteLn('Az átalakított karakterlánc: ',UjLanc) ;
 ReadLn ;
End.
```

7. feladat

Írjunk karakterlánc függvényeket és eljárásokat!

```
Program Karlanc7 ;

  { A paraméterek és a függvények típusa csak típusazonosító lehet, ezért az ott használandó
    típusokat előre deklarálnunk kell: }
Type
  St6  = String[6] ;
  St30 = String[30] ;

Var
  Lanc     : String ;
  MaiDatum : St6 ;

  { A függvény visszaadja a lánc nagybetűs alakját: }
Function Nagy(S: String): String ;
  Var
    I : Byte ;
  Begin
    For I := 1 To Length(S) Do
      S[I] := UpCase(S[I]) ;
    Nagy := S ;
  End ;

  { A függvény visszaadja a lánc kisbetűs alakját: }
Function Kis(S: String): String ;
  Var
    I : Byte ;
  Begin
    For I := 1 To Length(S) Do
      If S[I] In ['A'..'Z'] Then
          { Az I. karakternek típus–rádefiniálással adunk értéket: }
        S[I] := Char(Ord(S[I]) + 32) ;
    Kis := S ;
  End ;

  { A függvény visszaad egy Hossz hosszúságú, Jel–ből álló vonalat: }
Function Vonal(Jel: Char ; Hossz: Byte): String ;
  Var
    I : Byte ;
  Begin
    For I := 1 To Hossz Do
      Vonal[I] := Jel ;
```

```
    { A karakterlánc elemein végzett műveletek nem állítják a karakterlánc aktuális hosszát,
      ezért azt külön be kell állítani: }
    Vonal[0] := Chr(Hossz) ;
  End ;

  { A függvény a kapott karakterláncot kiegészíti szóközökkel adott hosszúságban (balra igazítás).
    A Hossz utáni karaktereket levágja: }
Function Bal(S: String ; Hossz: Byte): String ;
  Var
    I : Byte ;
  Begin
    S := Copy(S,1,Hossz) ;
    For I := Length(S) + 1 To Hossz Do
      S := S + ' ' ;
    Bal := S ;
  End ;

  { Az eljárás megfordítja a paraméterként kapott karakterláncot: }
Procedure Fordit(Var S: String) ;
  Var
    Ss : String ;
    I  : Byte ;
  Begin
    Ss := '' ;
    For I := Length(S) Downto 1 Do
      Ss := Ss + S[I] ;
    S := Ss ;
  End ;

  { Az eljárás kitörli S-ből az összes Kar karaktert: }
Procedure TorolKar(Var S: String ; Kar: Char) ;
  Var
    N : Byte ;
  Begin
    N := Pos(Kar,S) ;
    While N <> 0 Do
      Begin
        Delete(S,N,1) ;
        N := Pos(Kar,S) ;
      End ;
  End ;

  { Az eljárás a karakterláncban lévő összes '*' karaktert '&&'-re változtatja: }
Procedure Atalakit(Var S: String) ;
  Var
    CsillagPoz : Byte ;
  Begin
    CsillagPoz := Pos('*',S) ;
```

```
    While CsillagPoz <> 0 Do
      Begin
        Delete(S,CsillagPoz,1) ;
        Insert('&&',S,CsillagPoz) ;
        CsillagPoz := Pos('*',S) ;
      End ;
  End ;

  { A függvény az NNHHÉÉ alakú dátumot ÉÉHHNN alakra hozza: }
Function Datumford(Datum: St6): St6 ;
  Begin
    Datumford := Copy(Datum,5,2) + Copy(Datum,3,2) +
                 Copy(Datum,1,2) ;
  End ;

  { A függvény előállít egy véletlen, de maximum 30 hosszú, véletlen karakterekből (nagybetű,
    szám) álló karakterláncot: }
Function Veletlenlanc: St30 ;
  Var
    Hossz,
    I       : Byte ;
    Kar     : Char ;
  Begin
    Randomize ;
    Hossz := Random(31) ;
    Veletlenlanc[0] := Char(Hossz) ;
    For I := 1 To Hossz Do
      Begin
        Repeat
          Kar := Char(Random(Ord('Z') - Ord('0')+1) + Ord('0')) ;
        Until Kar In ['0'..'9','A'..'Z'] ;
        Veletlenlanc[I] := Kar ;
      End ;
  End ;

  { FŐPROGRAM: }
Begin
    { Karakterlánc bekérése: }
  Write('Lánc: ') ;
  ReadLn(Lanc) ;

    { Vonalhúzás '/' karakterekkel: }
  WriteLn(Vonal('/',79)) ;
    { Balra igazított nagybetűs és kisbetűs karakterlánc kiírása: }
  WriteLn('A lánc nagy- és kisbetûs formája, balra igazítva: ') ;
  WriteLn(Bal(Nagy(Lanc),30),Bal(Kis(Lanc),30)) ;
```

```
    { Karakterlánc megfordítása és kiírása: }
  Fordit(Lanc) ;
  WriteLn('A lánc fordítva: ',Lanc) ;

    { Karakterlánc visszafordítása: }
  Fordit(Lanc) ;

    { Szóközök törlése a karakterláncból, a lánc kiírása: }
  Torolkar(Lanc,' ') ;
  WriteLn('A lánc szóközök nélkül: ',Lanc) ;

    { Vonalhúzás '/' karakterekkel: }
  WriteLn(Vonal('/',79)) ;

    { Mai dátum fordított kiírása: }
  Write('Dátum: ') ;
  ReadLn(MaiDatum) ;
  MaiDatum := Datumford(Maidatum) ;
  WriteLn('A dátum fordítva: ',MaiDatum) ;

    { Véletlen karakterlánc előállítása és kiírása: }
  Lanc := Veletlenlanc ;
  WriteLn('Ime egy véletlenlánc: ',Lanc,' (',Length(Lanc),
          ' hosszú)') ;

  ReadLn ;
End.
```

TÖMBÖK

1. feladat

Olvassunk be számokat adott végjelig egy maximum 100 elemű tömbbe!

- **Írjuk ki a számokat a beolvasás sorrendjében, majd fordítva;**
- **Írjuk ki az elemek közül a legkisebbet és a legnagyobbat, tömbindexükkel együtt;**
- **Olvassunk be egy számot, és keressük meg azt a tömbben!**

```
Program Tomb1 ;

Const
  MaxElemszam = 100 ;
  VegJel      = 0 ;

Var
    { Tömb típusú változó. „Tomb"–nek MaxElemszam eleme van, 1–től MaxElemszam–ig
    minden indexhez tartozik egy–egy Integer típusú elem. Az elemekre a tömb indexeivel
    hivatkozhatunk: }
  Tomb      : Array [1..MaxElemszam] Of Integer ;
  Szam,
  MinSzam,
  MaxSzam   : Integer ;
  Ind,
  Elemszam,
  MinInd,
  MaxInd    : Word ;

  { FŐPROGRAM: }
Begin
    { Számok beolvasása: }
  WriteLn('Kérem a számokat: ') ;
  Elemszam := 0 ;
    { Az első szám beolvasása: }
  ReadLn(Szam) ;
    { Amíg nem végjelet ütöttek és van hely a tömbben: }
  While (Szam <> VegJel) And (Elemszam < MaxElemszam) Do
    Begin
        { A következő tömbelem a szám lesz: }
      Inc(Elemszam) ;
      Tomb[Elemszam] := Szam ;
        { A következő szám beolvasása: }
      ReadLn(Szam) ;
    End ;
```

```
  { Ha nincs szám a tömbben, leállás: }
If Elemszam = 0 Then
  Begin
    Write('Nincs beolvasott szám') ;
    ReadLn ;
    Halt ;
  End ;

  { Számok kiírása: }
Write('A számok: ') ;
For Ind := 1 To Elemszam Do
  Write(Tomb[Ind]:8) ;
WriteLn ;

  { Számok kiírása fordítva: }
Write('Fordítva: ') ;
For Ind := Elemszam Downto 1 Do
  Write(Tomb[Ind]:8) ;
WriteLn ;

  { A minimális és a maximális elem kiválasztása: }
MinSzam := MaxInt ;
MaxSzam := -MaxInt ;
MinInd := 0 ;
MaxInd := 0 ;
For Ind := 1 To Elemszam Do
  Begin
      { Ha az aktuális elem kisebb, mint az eddigi legkisebb, akkor az lesz a legkisebb,
      indexét megjegyezzük: }
    If Tomb[Ind] < MinSzam Then
      Begin
        MinSzam := Tomb[Ind] ;
        MinInd := Ind ;
      End ;
      { Ha az akuális elem nagyobb, mint az eddigi legnagyobb, akkor az lesz a legnagyobb,
      indexét megjegyezzük: }
    If Tomb[Ind] > MaxSzam Then
      Begin
        MaxSzam := Tomb[Ind] ;
        MaxInd := Ind ;
      End ;
  End ;
WriteLn('A beolvasott számok száma: ',Elemszam) ;
WriteLn('A legkisebb  elem: ',MinSzam,', indexe: ',MinInd) ;
WriteLn('A legnagyobb elem: ',MaxSzam,', indexe: ',MaxInd) ;

  { Elem keresése a tömbben: }
Write('A keresendô szám: ') ;
ReadLn(Szam) ;
Ind := 1 ;
```

```
    { Amíg nincs meg az elem és nincs vége a tömbnek: }
  While (Szam <> Tomb[Ind]) And (Ind < Elemszam) Do
    Inc(Ind) ;
  If Szam = Tomb[Ind]
  Then
    WriteLn('Megvan, indexe: ',Ind)
  Else
    WriteLn('Nincs meg') ;
  ReadLn ;
End.
```

2. feladat

Olvassunk be a billentyűzetről egy szöveget úgy, hogy az a képernyőn ne jelenjen meg. <ESC>–re fejeződjön be a bevitel! Ekkor írjuk ki a teljes szöveget úgy, hogy ahol <ENTER>–t nyomtunk, ott új bekezdés kezdődjék!

```
Program Tomb2 ;

Uses
  Crt ;

Const
  ESC   = #27 ;
  Enter = #13 ;

Var
    { 2000 karakterből álló tömb: }
  Szoveg  : Array[1..80 * 25] Of Char ;
  Ch      : Char ;
  Karszam,
  I       : Word ;

  { FŐPROGRAM: }
Begin
    { Karakterek beolvasása és gyűjtése a Szoveg karaktertömbben. Csak a megjeleníthető
      karaktereket és az <ENTER>–t gyűjtjük: }
  ClrScr ;
  WriteLn('Irjon be egy szöveget ''vakon''!') ;
  Karszam := 0 ;
    { Első billentyű olvasása: }
  Ch := ReadKey ;
```

```
    { <ESC>–re a karakterek bevitele befejeződik: }
  While Ch <> ESC Do
    Begin
        { A vezérlő billentyűket nem fogadjuk el, kivéve az <ENTER>–t: }
      If Not (Ch In [#0..#31] - [Enter]) Then
        Begin
          Inc(Karszam) ;
          Szoveg[Karszam] := Ch ;
        End ;
        { Kettős kódú billentyűt sem fogadunk el, de a kód párját ki kell olvasnunk a
        pufferből: }
      If Ch = #0 Then
        Ch := ReadKey ;
        { Következő billentyű olvasása: }
      Ch := ReadKey ;
    End ;

    { Az összegyűjtött karakterek kiírása (paragrafussal kezdünk): }
    Write('    ') ;
    For I := 1 To Karszam Do
      If Szoveg[I] = Enter
      Then
          { Új paragrafus kezdődik: }
        Begin
          WriteLn ;
          WriteLn ;
          Write('    ') ;
        End
      Else
        Write(Szoveg[I]) ;

      { Várakozás az <ESC> billentyű lenyomására: }
    Repeat
    Until ReadKey = ESC ;
End.
```

3. feladat

Írjunk zongorát utánzó programot! Egy alaphangtól kezdve definiáljuk félhangonként a hanghoz tartozó billentyűket! A programmal ezen a „zongorán" lehessen játszani!

```
Program Tomb3 ;

Uses
  Crt ;
```

```
Const
    { Normál „a" hang (a1): }
  AlsoFrek     = 440 ;
    { Álljon a zongora 13 hangból, a következő „a" hangig: }
  HangokSzama = 13 ;
    { Ezeket a billentyűket lehet majd használni: }
  Bill         : Array[1..HangokSzama] Of Char =
    ('A','W','S','E','D','F','T','G','Y','H','U','J','K') ;

Var
  Ch           : Char ;

  { A függvény kiszámítja a Billentyu–höz tartozó frekvenciaértéket: }
Function Frekv(Billentyu: Char): Word ;
  Var
    I        : Byte ;
    F,
    Felhang : Real ;
  Begin
      { Az egyenletesen temperált hangsor egy oktávja 12 részre osztott; a szomszédos hangok
      hányadosai egyenlőek; egy hang oktávjának frekvenciája az illető hang frekvenciájának
      kétszerese: }
    Felhang := Exp(Ln(2) / 12) ;  { Tizenkettedik gyök 2 }
    I := 1 ;
    F := AlsoFrek ;
      { Amíg nincs meg a billentyű és van mit keresni, növeljük a frekvenciát: }
    While (Billentyu <> Bill[I]) And (I < HangokSzama) Do
      Begin
        Inc(I) ;
        F := F * FelHang ;
      End ;
      { Ha nincs ilyen billentyű: }
    If Billentyu <> Bill[I] Then
      F := 0 ;
    Frekv := Round(F) ;
  End ;

  { FŐPROGRAM: }
Begin
  WriteLn('Zongorázzon!') ;
    { Első billentyű bekérése: }
  Ch := UpCase(ReadKey) ;
```

```
    { Amíg nem <ESC>–et ütnek: }
  While Ch <> #27 Do
    Begin
        { Hang megszólaltatása: }
      Sound(Frekv(Ch)) ;
        { Várakozás a következő billentyű leütésére: }
      Ch := UpCase(ReadKey) ;
        { Hang kikapcsolása: }
      NoSound ;
        { Késleltetés a hangok tagolása miatt: }
      Delay(50) ;
    End ;
End.
```

4. feladat

A tárgyhó 7 és 12–e közötti számlák alapján kimutatást kell készíteni a napi forgalmakról. A napokat és a hozzá tartozó összegeket egymás után, rendezetlenül olvassuk be nap = 0 –ig! Írjuk ki a gyűjtött összegeket nap szerint rendezetten!

```
Program Tomb4 ;

  { Érték– és indexhatár ellenőrzés bekapcsolása: }
{$R+}

Const
  ElsoNap   = 7 ;
  UtolsoNap = 12 ;

Type
  Napok = ElsoNap..UtolsoNap ;

Var
  Nap, I : Byte ;
  Osszeg : Word ;
  Gyujto : Array[Napok] Of Word ;

  { Nap beolvasása: }
Procedure NapOlv ;
  Begin
    Repeat
        Write('Nap (',ElsoNap,'-',UtolsoNap,',Vége=0) : ') ;
        ReadLn(Nap) ;
    Until Nap In [ElsoNap..UtolsoNap,0] ;
  End ;
```

```
  { Összeg beolvasása: }
Procedure OsszegOlv ;
  Begin
    Write('Összeg : ') ;
    ReadLn(Osszeg) ;
  End ;

  { FŐPROGRAM: }
Begin
    { Tömb feltöltése nullákkal: }
  For I := ElsoNap To UtolsoNap Do
    Gyujto[I] := 0 ;

    { Első nap beolvasása: }
  NapOlv ;
    { Amíg nem nullát ütnek: }
  While Nap <> 0 Do
    Begin
        { Összeg beolvasása: }
      OsszegOlv ;
      WriteLn ;
        { Összeg gyűjtése: }
      Inc(Gyujto[Nap],Osszeg) ;
        { Következő nap beolvasása: }
      NapOlv ;
    End ;

    { Forgalom kiírása (előtte két soremelés): }
  WriteLn(#10#10,'N A P I   F O R G A L M A K') ;
  For I := ElsoNap to UtolsoNap Do
    WriteLn(I:3,'. nap összesen: ',Gyujto[I]:4) ;
  ReadLn ;
End.
```

5. feladat

Számoljuk meg, melyik betűből hány van egy beolvasott karakterláncban! A kis és nagybetűket együtt számoljuk! Az eredményt jelenítsük meg először számokkal, majd grafikonnal!

```
Program Tomb5 ;

  { Érték– és indexhatár ellenőrzés bekapcsolása: }
{$R+}
```

```
Uses
  Crt ;

Const
  AlsoSor = 25 ;

Type
  Betuk = 'A'..'Z' ;

Var
  Lanc        : String ;
    { BetukSzama változónak 26 eleme van, minden nagybetűhöz tartozik egy bájt. A tömböt a
      nagybetűkkel indexeljük: }
  BetukSzama : Array[Betuk] Of Byte ;
  Betu        : Char ;
  I,
  X,
  Magassag    : Byte ;

  { FŐPROGRAM: }
Begin
  ClrScr ;
  Write('Karakterlánc: ') ;
  ReadLn(Lanc) ;

    { A betűk számát tartalmazó tömb nullázása: }
  For Betu := 'A' To 'Z' Do
    BetukSzama[Betu] := 0 ;

    { Ha a lánc adott karaktere betű, megnöveljük az ahhoz a betűhöz tartozó tömbelemet: }
  For I := 1 To Length(Lanc) Do
    Begin
      Betu := Upcase(Lanc[I]) ;
      If Betu In ['A'..'Z'] Then
        Inc(BetukSzama[Betu]) ;
    End ;

    { A betűk számát tartalmazó tömb kiírása: }
  For Betu := 'A' To 'Z' Do
    Write(Betu,':',BetukSzama[Betu]:3,'    ') ;
  ReadLn ;

    { A betűk számához tartozó grafikon elkészítése: }
  ClrScr ;
```

```
  For Betu := 'A' To 'Z' Do
    Begin
        { Az 'A' betű a 27. oszlopba kerül, 'B' a 28.-ba, stb. }
      X := 27 + Ord(Betu) - Ord('A') ;
      GotoXY(X,AlsoSor) ;
      Write(Betu) ;
        { A betűk felett álló oszlop nem futhat ki a képernyőről: }
      If BetukSzama[Betu] <= 23
      Then
        Magassag := BetukSzama[Betu]
      Else
        Magassag := 23 ;
        { A betű feletti oszlop elkészítése: }
      For I := 1 To Magassag Do
        Begin
          GotoXY(X,AlsoSor - I + 1) ;
          Write(#219) ;
        End ;
    End ;

    { Várakozás <ESC> vagy <ENTER> leütésére: }
  Repeat
  Until ReadKey In [#13, #27] ;
  WriteLn ;
End.
```

6. feladat

Olvassunk be egymás után színeket (Fehér, Sárga, Piros, Kék,...)! Gyűjtsük, hogy melyik színből hány szerepelt, majd készítsünk erről százalékos kimutatást is!

```
Program Tomb6 ;

  { A karakterlánc típusú formális és aktuális változó paraméterek típusának nem kell egyezniük: }
{$V-}

Uses
  Crt ;

Type
    { Színek felsorolása. Ezekkel fogjuk a tömböt indexelni: }
  Szinek = (Feher, Sarga, Narancs, Piros,
            Lila, Kek, Zold, Fekete) ;
```

```
Const
    { A színekhez tartozó karakterláncok (a színek kiírható alakjai): }
  SzinSt : Array[Szinek] Of String[8] =
             ('Feher','Sarga','Narancs','Piros',
              'Lila','Kek','Zold','Fekete') ;

Var
  Szin   : Szinek ;
  Gyujto : Array[Szinek] Of Word ;
  Osszes : Word ;
  St     : String[8] ;

  { Az eljárás egy lánc első betűjét nagyra, a többit kicsire alakítja: }
Procedure Atalakit(Var S : String) ;
  Var
    I : Byte ;
  Begin
    S[1] := UpCase(S[1]) ;
    For I := 2 To Length(S) Do
      If S[I] In ['A'..'Z'] Then
        S[I] := Chr(Ord(S[I]) + (Ord('a') - Ord('A'))) ;
  End ;

  { FŐPROGRAM: }
Begin
    { A tömb nullázása: }
  For Szin := Feher To Fekete Do
    Gyujto[Szin] := 0 ;

    { Első szín beolvasása: }
  ClrScr ;
  WriteLn('Kérem a színeket: ') ;
  ReadLn(St) ;
    { Amíg színként nem üres karakterláncot ütnek: }
  While St <> '' Do
    Begin
        { Szin változó beállítása a beolvasott karakterlánc alapján: }
      Atalakit(St) ;
      Szin := Feher ;
      While (Szin < Fekete) And (St <> SzinSt[Szin]) Do
        Inc(Szin) ;
        { Ha van ilyen szín, akkor a színhez tartozó gyűjtő növelése: }
      If St = SzinSt[Szin] Then
        Inc(Gyujto[Szin]) ;
        { Következő szín beolvasása: }
      ReadLn(St) ;
    End ;
```

```
    { A beolvasott színek száma összesen: }
  Osszes := 0 ;
  For Szin := Feher To Fekete Do
    Osszes := Osszes + Gyujto[Szin] ;

    { Kimutatás elkészítése: }
  WriteLn('KIMUTATÁS A BEOLVASOTT SZINEKRÖL') ;
  WriteLn(' Szín    Db          %') ;
  WriteLn ;
  For Szin := Feher To Fekete Do
    Begin
      Write(SzinSt[Szin]) ;
      GotoXY(10,WhereY) ;
      Write(Gyujto[Szin]:2) ;
      GotoXY(20,WhereY) ;
      If Osszes <> 0 Then
        Write(Gyujto[Szin] * 100.0 / Osszes:5:1) ;
      WriteLn ;
    End ;
  ReadLn ;
End.
```

7. feladat

Olvassunk be Integer típusú számokat, és írjuk ki azok hexadecimális ábrázolását!

```
Program Tomb7 ;

Type
  St8 = String[8] ;

Var
  DecSzam : Integer ;

  { A függvény egy 0 és 15 közötti decimális értéket hexadecimális jellé alakít: }
Function Hex(B: Byte) : Char ;
  Const
    HexJegy : Array [0..15] Of Char = '0123456789ABCDEF' ;
  Begin
    Hex := HexJegy[B] ;
  End ;
```

```
  { A függvény egy tetszőleges Integer típusú egész számot hexadecimális jelsorozattá alakít: }
Function HexForma(Dec: Integer) : St8 ;
  Var
    S : String[8] ;
    L : LongInt ;
    I : Byte ;
  Begin
      { A negatív számok belső ábrázolása kettes komplemensben történik. Két bájton a
        legkisebb negatív szám a $8000, a legnagyobb pozitív szám pedig a $7FFF: }
    L := Dec ;
    If L < 0 Then
      L := L + $10000 ;

      { A hexadecimális jelsorozat előállítása: }
    S := '' ;
    Repeat
        S := Hex(L Mod 16) + S ;
        L := L Div 16 ;
    Until L = 0 ;

      { Ha a hexadecimális jelsorozat hossza rövidebb, mint 4 karakter, a bevezető nullákat
        elé tesszük: }
    For I := Length(S) To 3 Do
      S := '0' + S ;
    HexForma := '$' + S ;
  End ;

  { Szám beolvasása, mely értéke az Integer típus értékkészletébe esik: }
Procedure Be(Var Szam: Integer) ;
  Var
    L : LongInt ;
  Begin
    Repeat
      ReadLn(L) ;
    Until (L >= -MaxInt - 1) And (L <= MaxInt) ;
    Szam := L ;
  End ;

  { FőPROGRAM: }
Begin
  WriteLn('Kérek Integer számokat,',
          ' megadom hexadecimális alakjukat: ') ;
  Be(DecSzam) ;
    { Amíg nem nullát ütnek: }
  While DecSzam <> 0 Do
    Begin
      WriteLn(HexForma(DecSzam)) ;
      WriteLn ;
```

```
      Be(DecSzam) ;
    End ;
End.
```

8. feladat

Modellezzük a verem működését! A verembe egyesével betehetünk elemeket, de mindig csak a verem tetejére, valamint a veremből egyesével kivehetünk elemeket, de mindig csak a verem tetején lévőt. Minden egyes tranzakció után jelenítsük meg a verem tartalmát!

A verem (stack) egyfajta LIFO (Last In First Out) szerkezet, vagyis amelyből mindig csak a legutoljára bevitt adatot vehetjük ki.

```
Program Tomb8 ;

Uses
  Crt ;

Const
  VeremMeret = 15 ;

Type
  ElemTip = String[5] ;

Var
  Verem      : Array[1..VeremMeret] Of ElemTip ;
  Elem       : ElemTip ;
  Mutato,
  I          : Word ;
  Ures,
  Tele       : Boolean ;
  Tranzakcio : Char ;

  { Ha nincs még tele a verem, az eljárás beolvas egy elemet, és azt a verem tetejére teszi: }
Procedure Betesz ;
  Begin
      { Ha a verem betelt, kilépés az eljárásból: }
    If Tele Then
      Exit ;
      { Elem beolvasása és a verembe tétele: }
    GotoXY(50,14) ;
    Write('A beteendô elem: ') ;
    ReadLn(Elem) ;
    Verem[Mutato] := Elem ;
    GotoXY(12,25 - Mutato) ;
    Write(Elem:5) ;
```

```
    Inc(Mutato) ;
    Tele := Mutato > VeremMeret ;
    Ures := False ;
  End ;

  { Ha nem üres a verem, az eljárás kivesz egy elemet a verem tetejéről: }
Procedure Kivesz ;
  Begin
      { Ha a verem üres, kilépés az eljárásból: }
    If Ures Then
      Exit ;
      { Elem kivétele a veremből: }
    Dec(Mutato) ;
    GotoXY(12,25 - Mutato) ;
    Write('':5) ;
    Ures := Mutato = 1 ;
    Tele := False ;
  End ;

  { FŐPROGRAM: }
Begin
    { Verem körvonalának megrajzolása: }
  ClrScr ;
    { Verem oldalai: }
  For Mutato := 1 To VeremMeret Do
    Begin
      GotoXY(10,25 - Mutato) ;
      Write(#179,#179:8) ;
    End ;
    { Verem alja: }
  GotoXY(10,25) ;
  Write(#192) ;
  For I := 1 To 7 Do
    Write(#196) ;
  Write(#217) ;

    { Kezdeti értékek beállítása: }
  Mutato := 1 ;
  Tele := False ;
  Ures := True ;
  GotoXY(12,1) ;
  Write('Üres') ;

    { Első tranzakció bekérése: }
  GotoXY(50,12) ;
  Write('B(e)/K(i)/V(ége)') ;
  Tranzakcio := Upcase(ReadKey) ;
```

```
    { Amíg még szeretnénk a verembe betenni vagy onnan kivenni: }
  While Tranzakcio <> 'V' Do
    Begin
      Case Tranzakcio Of
        'B' : Betesz ;
        'K' : Kivesz ;
      End ;
        { Információs sorok törlése, 'Tele' vagy 'Üres' kiírása: }
      GotoXY(50,14) ;
      ClrEol ;
      GotoXY(12,1) ;
      ClrEol ;
      If Tele Then
        Write('Tele') ;
      If Ures Then
        Write('Üres') ;
        { Következő tranzakció bekérése: }
      GotoXY(50,12) ;
      Write('B(e)/K(i)/V(ége)') ;
      Tranzakcio := Upcase(ReadKey) ;
    End ;
  ClrScr ;
End.
```

9. feladat

Modellezzük a sor működését! A sorhoz egyesével adhatunk elemeket, de mindig csak a sor végéhez, valamint a sorból egyesével elvehetünk elemeket, de mindig csak a sor elejéről. Minden egyes tranzakció után jelenítsük meg a sor tartalmát!

A sor (queue) egyfajta FIFO (First In First Out) szerkezet, vagyis amelyből mindig csak a legelőször bevitt adatot vehetjük ki.

```
Program Tomb9 ;

Uses
  Crt ;

Const
  Max = 15 ;
Type
  IndexTip = 0..Max ;
  ElemTip  = String[4] ;
```

```
Var
    { Az állandó tömbmásolás elkerülése végett a sort ciklikusan tároljuk a tömbben, és mindig
    mejegyezzük a sor első és utolsó elemét, valamint a sorban álló elemek számát: }
  Sor          : Array[1..Max] Of ElemTip ;
  Elem         : ElemTip ;
  Sorban,
  ELso,
  Utolso       : IndexTip ;
  Ures,
  Tele         : Boolean ;
  Tranzakcio : Char ;

  { Az eljárás megnöveli a sor indexét eggyel: }
Procedure Kovetkezo(Var Index: IndexTip) ;
  Begin
    If Index = Max
    Then
      Index := 1
    Else
      Inc(Index) ;
  End ;

  { Az eljárás a teljes sort kiírja: }
Procedure SorKiir ;
  Var
    I,
    Index : IndexTip ;
  Begin
    GotoXY(5,13) ;
    ClrEol ;
    Index := Elso ;
    For I := 1 To Sorban Do
      Begin
        Write(Sor[Index]:5) ;
        Kovetkezo(Index) ;
      End ;
  End ;

  { Ha nincs még tele a sor, az eljárás beolvas egy elemet, és azt a sor végére teszi: }
Procedure Sorba ;
  Begin
      { Ha tele a sor, kilépés az eljárásból: }
    If Tele Then
      Exit ;
      { Elem beolvasása és a sor végére tétele: }
    GotoXY(30,22) ;
    Write('A beteendô elem: ') ;
    GotoXY(50,22) ;
    ReadLn(Elem) ;
```

```
    Kovetkezo(Utolso) ;
    Sor[Utolso] := Elem ;
    GotoXY(5 + 5 * Sorban,13) ;
    Write(Elem:5) ;
    Inc(Sorban) ;
    Tele := Sorban = Max ;
    Ures := False ;
  End ;

  { Ha nem üres a sor, az eljárás kivesz egy elemet a sor elejéről: }
Procedure Sorbol ;
  Begin
      { Ha üres a sor, kilépés az eljárásból: }
    If Ures Then
      Exit ;
      { Elem kivétele a sorból: }
    Kovetkezo(Elso) ;
    Dec(Sorban) ;
      { Az egész sort ki kell írni, mert a sor elejéről vettünk ki: }
    SorKiir ;
    Ures := Sorban = 0 ;
    Tele := False ;
  End ;

Procedure VonalHuz(X,Y, Hossz: Byte) ;
  Var
    I : Byte ;
  Begin
    GotoXY(X,Y) ;
    For I := 1 To Hossz Do
      Write(#196) ;
  End ;

  { FŐPROGRAM: }
Begin
  ClrScr ;
    { Sor keretének megrajzolása: }
  VonalHuz(5,12,5 * Max) ;
  VonalHuz(5,14,5 * Max) ;

    { Kezdeti értékek beállítása: }
  Sorban := 0 ;
  Elso := 1 ;
  Utolso := 0 ;
  Tele := False ;
  Ures := True ;
  GotoXY(5,10) ;
  Write('Üres') ;
```

```
    { Első tranzakció bekérése: }
  GotoXY(30,20) ;
  Write('B(e)/K(i)/V(ége)') ;
  Tranzakcio := Upcase(ReadKey) ;
    { Amíg még szeretnénk a sorhoz hozzátenni, vagy onnan kivenni: }
  While Tranzakcio <> 'V' Do
    Begin
      Case Tranzakcio Of
        'B' : Sorba ;
        'K' : Sorbol ;
      End ;
        { Információs sorok törlése, 'Tele' vagy 'Üres' kiírása: }
      GotoXY(30,22) ;
      ClrEol ;
      GotoXY(5,10) ;
      ClrEol ;
      If Tele Then
        Write('Tele') ;
      If Ures Then
        Write('Üres') ;
        { Következő tranzakció bekérése: }
      GotoXY(30,20) ;
      Write('B(e)/K(i)/V(ége)') ;
      Tranzakcio := Upcase(ReadKey) ;
    End ;
  ClrScr ;
End.
```

10. feladat

Írjuk ki két adott szám között az összes prímszámot!

```
Program Tomb10 ;

  { Bekapcsoljuk a numerikus kooprocesszort. Ha fordításkor nincsen a gépben, emuláljuk. }
{$N+}
{$IFNDEF CPU87}
  {$E+}
{$ENDIF}
Uses
  Crt ;

Var
  Primek        : Array[1..10000] Of Longint ;
```

```
  PrimDarab,
  PrimHatar,
  M             : Word ;
  N,
  PrimNegyzet,
  Mettol,
  Meddig        : Longint ;
  Sor           : Byte ;

Label
  Tovabb ;

  { FŐPROGRAM: }
Begin
    { A két számhatár beolvasása: }
  ClrScr ;
  WriteLn('Prímszámok kiírása') ;
  Sor := WhereY ;
  Write('Mettôl: ') ;
  ReadLn(Mettol) ;
  GotoXY(20,Sor) ;
  Write('Meddig: ') ;
  ReadLn (Meddig) ;
    { Elegendő egy szám prím osztóit keresni, azok közül is csak azokat, melyek kisebbek a szám
      gyökénél. Ha nincs ilyen, a szám prím.
    Az első prímszám a 2: }
  Primek[1] := 2 ;
    { A talált prímek száma kettőtől: }
  PrimDarab := 1 ;
    { Az oszthatóságot az ilyen indexű prímszámig kell elvégezni: }
  PrimHatar := 1 ;
    { Az utolsó utáni vizsgálandó prím négyzete (a program gyorsasága így nagyban megnő,
      nem kell minden kérdéses számból gyököt vonni a határ megállapításához: }
  PrimNegyzet := 9 ;

    { A kért határig minden N–ről megállapítjuk, hogy prímszám–e: }
  For N := 3 To Meddig Do
    Begin
      If N >= PrimNegyzet Then
        Begin
          Inc(PrimHatar) ;
          PrimNegyzet := Sqr(Primek[PrimHatar + 1]) ;
        End ;
      For M := 1 To PrimHatar Do
        If N Mod Primek[M] = 0 Then
          Goto Tovabb ;
```

```
        { A talált prímszámot betesszük a Primek tömbbe: }
      Inc(PrimDarab) ;
      Primek[PrimDarab] := N ;
    Tovabb:
    End ;

    { A Mettol számig nem kell kiírni a prímeket: }
  N := 1 ;
  While Primek[N] < Mettol Do
    Inc(N) ;

    { A kért intervallumba eső prímszámok kiírása: }
  WriteLn('A prímszámok :') ;
  For M := N To PrimDarab Do
    Write(Primek[M]:8) ;
  WriteLn ;
  WriteLn(Mettol,' és ',Meddig,' között ',Primdarab - N + 1,
          ' prímszám van !') ;
  ReadLn ;
End.
```

11. feladat

Állítsunk elő egy 500 elemű tömböt! A tömbelemek kulcsai –9999 és 9999 közötti véletlen egészek legyenek! Rendezzük a tömböt kulcs szerint növekvően különböző, ismert módszerekkel! Minden esetben állapítsuk meg a rendezés idejét!

```
Program Tomb11 ;

  { A verem méretét nagyobbra kell venni, hogy a nagyméretű tömb, – a rendezési eljárások érték
  szerinti paramétere – elférjen: }
{$M 50000,0,655360}

  { Kiterjesztett szintaktika esetén hívhatjuk a függvényeket eljárásként is, ilyenkor a függvény
  értéke érdektelen (pl.: ReadKey;): }
{$X+}

Uses
  Crt,
  Dos ;

Const
  Elemszam = 500 ;
  MinKulcs = -9999 ;
  MaxKulcs =  9999 ;
```

```
Type
    { Ilyen rekordok tömbjében tároljuk az adatokat: }
  ElemTip = Record
              Kulcs : Integer ;
              Egyeb : String[20] ;
            End ;
  TombTip = Array [0..Elemszam] Of ElemTip ;

Var
  Tomb      : TombTip ;
  I         : Word ;
  IdoKezd,
  IdoVeg    : LongInt ;

  { A függvény a számítógép óráját átszámítja századmásodpercekre: }
Function SzazadMasodPerc : LongInt ;
    { Az Egy állandó azért kell, hogy a kifejezés eredménytípusa LongInt legyen, egyébként az
    eredmény csonkul: }
  Const
    Egy : LongInt = 1 ;
  Var
    Ora,
    Perc,
    MPerc,
    SzMPerc : Word ;
  Begin
    GetTime(Ora, Perc, MPerc, SzMPerc) ;
    SzazadMasodPerc := Egy * 100 * (60 * (60 * Ora + Perc) +
                       MPerc) + SzMPerc ;
  End ;

  { Az eljárás megjegyzi a kezdési időt századmásodpercekben: }
Procedure StopperBe ;
  Begin
    IdoKezd := SzazadMasodPerc ;
  End ;

  { Az eljárás megjegyzi a befejezési időt századmásodpercekben: }
Procedure StopperKi ;
  Begin
    IdoVeg := SzazadMasodPerc ;
  End ;

  { A függvény kiszámítja a két időpont között eltelt másodperceket: }
Function IdoMp : Real ;
  Begin
    IdoMp := (IdoVeg - IdoKezd) / 100 ;
  End ;
```

```
  { Az eljárás az eddig leütött karaktereket kiolvassa a billentyűzet pufferéből, majd vár az
  <ENTER> leütésére: }
Procedure Varj ;
  Begin
    Write('Tovább <ENTER>') ;
    While KeyPressed Do
      ReadKey ;
    Repeat
    Until ReadKey = #13 ;
    WriteLn ;
  End ;

  { Véletlen tömb generálása: }
Procedure Tomb_General ;
  Var
    I : Word ;
  Begin
    Randomize ;
      { Bizonyos rendezések felhasználják a 0. elemet, mint ütközőt: }
    Tomb[0].Kulcs := 0 ;
    Tomb[0].Egyeb := '' ;
    For I := 1 To Elemszam Do
      Begin
          { A kulcs az adott határok közötti véletlen szám, az Egyeb mezőt pedig kitöltjük
          annak szöveges változatával: }
        Tomb[I].Kulcs := Random(MaxKulcs - MinKulcs + 1) +
                         MinKulcs ;
        Str(Tomb[I].Kulcs,Tomb[I].Egyeb) ;
        Tomb[I].Egyeb := 'Elem ' + Tomb[I].Egyeb ;
      End ;
  End ;

  { A tömb kulcsainak kiírása: }
Procedure Kulcs_Kiir(Var T: TombTip) ;
  Var
    I : Word ;
  Begin
    For I := 1 To Elemszam Do
      Begin
        If WhereY > 23 Then
          Begin
            Varj ;
            ClrScr ;
          End ;
        Write(T[I].Kulcs:5) ;
      End ;
    WriteLn ;
  End ;
```

```
  { Két változó értékének felcserélése: }
Procedure Csere(Var Elem1, Elem2: ElemTip) ;
  Var
    Ment : ElemTip ;
  Begin
    Ment := Elem1 ;
    Elem1 := Elem2 ;
    Elem2 := Ment ;
  End ;
```

{ Rendezés a legkisebb elem kiválasztásával:

A módszer lényege, hogy az I. helyre (I = 1, 2...) sorban kiválasztjuk az I..Elemszam helyen lévők közül a legkisebbet. }

```
Procedure Rendez_Minvalaszt(T: TombTip) ;
  Var
    I,
    J,
    Min : Word ;
  Begin
    StopperBe ;
    For I := 1 To Elemszam - 1 Do
      Begin
        Min := I ;
        For J := I + 1 To Elemszam Do
          If T[J].Kulcs < T[Min].Kulcs Then
            Min := J ;
        Csere(T[I],T[Min]) ;
      End ;
    StopperKi ;
  End ;
```

{ Rendezés a szomszédos elemek cseréjével (buborékos rendezés):

A módszer lényege, hogy egyesével végignézzük a szomszédos elemeket, és ha előbb van a nagyobb, megcseréljük azokat. Így biztosan a legnagyobb helyre kerül a legnagyobb elem („a buborék felszáll"). Ugyanezt sorban végrehajtjuk az utolsó előtti elemig stb., egészen az elsőig. Ha közben kiderül, hogy a tömb rendezett, az eljárást befejezzük. }

```
Procedure Rendez_Szomszedcsere(T: TombTip) ;
  Var
    I,
    J  : Word ;
    Ok : Booleam ;
  Begin
    StopperBe ;
    I := Elemszam - 1 ;
    Ok := False ;
    While (I >= 1) And Not Ok Do
      Begin
        Ok := True ;
```

```
        For J := 1 To I Do
          If T[J].Kulcs > T[J + 1].Kulcs Then
            Begin
              Csere(T[J],T[J + 1]) ;
              Ok := False ;
            End ;
        Dec(I) ;
      End ;
    StopperKi ;
  End ;
```

{ Rendezés beszúrásos módszerrel:

A módszer hasonlít a kártyák kézben való rendezéséhez. Mindig vesszük a következő elemet, s azt a megfelelő helyre beszúrjuk úgy, hogy a többi, már rendezett elemet eggyel feljebb toljuk. }

```
Procedure Rendez_Beszur(T: TombTip) ;
  Var
    Ment : ElemTip ;
    I,
    J    : Word ;
  Begin
    StopperBe ;
```

{ A tömb nulladik eleme a rendezésben ütköző szerepet játszik: }

```
    T[0].Kulcs := -MaxInt ;
    For I := 2 To ElemSzam Do
      Begin
```

{ A tömb az I–1. elemig rendezett. Az I. elemet (ha nincs a helyén) a helyére tesszük, s a többi elemet felcsúsztatjuk: }

```
        If T[I].Kulcs < T[I - 1].Kulcs Then
          Begin
            Ment := T[I] ;
            J := I ;
            Repeat
                Dec(J) ;
                T[J + 1] := T[J] ;
            Until T[J - 1].Kulcs <= Ment.Kulcs ;
            T[J] := Ment ;
          End ;
      End ;
    StopperKi ;
  End ;
```

{ Rendezés beszúrásos módszerrel, lépésenkénti finomítással (Donald L. Shell, 1959):

Vegyük az első elemtől kezdve a tömb minden n. elemét (lépésköz = n), és rendezzük ezen elemeket. Majd vegyük a második elemtől kezdve a tömb minden n. elemét, s azokat is rendezzük. Az utolsó rendezendő sorozat az n–1. elemtől indul (így a tömb összes elemét érintettük). Ezután csökkentsük a lépésközt, és végezzük el így is a rendezést. Ezt addig folytassuk, amíg a lépésköz = 1, ekkor a tömb már biztosan rendezett lesz. }

```
Procedure Rendez_Shell(T: TombTip) ;
  Var
    Ment      : ElemTip ;
    LepesKoz,
    Kezd,
    I,
    J         : Word ;
  Begin
    StopperBe ;
    LepesKoz := Elemszam ;
    T[0].Kulcs := -MaxInt ;
    Repeat
        LepesKoz := LepesKoz Div 3 + 1 ;
        For Kezd := 1 To LepesKoz Do
          Begin
            I := Kezd + LepesKoz ;
            While I <= Elemszam Do
              Begin
                If T[I].Kulcs < T[I - LepesKoz].Kulcs Then
                  Begin
                    J := I ;
                    Ment := T[I] ;
                    Repeat
                        Dec(J,LepesKoz) ;
                        T[J + LepesKoz] := T[J] ;
                    Until (J <= Kezd) Or
                          (T[J - LepesKoz].Kulcs <= Ment.Kulcs) ;
                    T[J] := Ment ;
                  End ;
                Inc(I,LepesKoz) ;
              End ;
          End ;
    Until LepesKoz = 1 ;
    StopperKi ;
  End ;
```

{ Gyorsrendezés (quicksort, C.A.R. Hoare, 1962):

Tekintsük a tömb középső elemét (felezzük a tömböt). Balról keressük meg azt az első elemet, mely ennél nem kisebb, jobbról azt az első elemet, mely ennél nem nagyobb. Cseréljük ki a két elemet, s folytassuk a cserélgetést egészen addig, amíg a baloldalon a középső elemnél (mely természetesen cserélődhet menet közben) csupa nem nagyobb, jobboldalon pedig csupa nem

kisebb elem áll. Rekurzív hívással most rendezzük a tömb alsó és felső felét, majd ezeknek alsó és felső felét, és így tovább. }

```
Procedure Rendez_Quick(Var T: TombTip) ;

  Procedure Quick(Bal,Jobb: Word) ;
    Var
      I,
      J             : Word ;
      KozepsoKulcs : Integer ;
    Begin
      I := Bal ;
      J := Jobb ;
      KozepsoKulcs := T[(I + J) Div 2].Kulcs ;
      While I <= J Do
        Begin
          While T[I].Kulcs < KozepsoKulcs Do
            Inc(I) ;
          While T[J].Kulcs > KozepsoKulcs Do
            Dec(J) ;
          If I <= J Then
            Begin
              Csere(T[I],T[J]) ;
              Inc(I) ;
              Dec(J) ;
            End ;
        End ;
      If Bal < J Then
        Quick(Bal,J) ;
      If I < Jobb Then
        Quick(I,Jobb) ;
    End ;

  Begin
    StopperBe ;
    Quick(1,Elemszam) ;
    StopperKi ;
  End ;

  { FŐPROGRAM: }
Begin
  ClrScr ;
    { Tömb feltöltése: }
  Tomb_General ;

  WriteLn('Rendezés minimális elem kiválasztásával:') ;
  Rendez_Minvalaszt(Tomb) ;
  WriteLn('  rendezési idô: ',IdoMp:5:2) ;

  WriteLn('Rendezés szomszédos elemek cseréjével:') ;
  Rendez_Szomszedcsere(Tomb) ;
  WriteLn('  rendezési idô: ',IdoMp:5:2) ;
```

```
  WriteLn('Rendezés beszúrással:') ;
  Rendez_Beszur(Tomb) ;
  WriteLn('  rendezési idô: ',IdoMp:5:2) ;

  WriteLn('Rendezés Shell módszerével:') ;
  Rendez_Shell(Tomb) ;
  WriteLn('  rendezési idô: ',IdoMp:5:2) ;

  WriteLn('Gyorsrendezés:') ;
  Rendez_Quick(Tomb) ;
  WriteLn('  rendezési idô: ',IdoMp:5:2) ;

  WriteLn('A tömb kulcsainak kiírása következik: ') ;
  Varj ;
  Kulcs_Kiir(Tomb) ;
  Varj ;
End.
```

Megjegyzés a Tomb11 programhoz:

A különböző rendezési eljárások tesztelése céljából a program az eredeti tömböt mindig meghagyja rendezetlenül. Csak az eljárás értékparamétereként megadott T tömb lesz rendezett, ami természetesen az eljárás befejezésekor elvész. Az utolsó rendezés esetén azonban a paraméter változó, így az eredeti tömb rendezett lesz. Három tesztfutás alkalmával különböző elemszámmal 386 SX mikroprocesszorral (16 MHz) a következő eredmények születtek (az értékek másodpercben):

	Minimum	Szomszédos	Beszúrásos	Shell	Gyors
500 elem	1.15	5.16	1.54	0.22	0.11
1000 elem	4.56	21.09	6.10	0.49	0.28
2000 elem	18.18	83.82	24.16	1.05	0.66

12. feladat

Állítsunk elő egy 500 elemű rendezett tömböt. A tömbelemek kulcsai –9999 és 9999 közötti véletlen egészek legyenek. Ezután olvassunk be egy kulcsot, s azt keressük meg a tömbben lineáris illetve bináris keresési módszerrel!

```
Program Tomb12 ;

{$X+}
```

```
Uses
  Crt,
  Dos ;

Const
  Elemszam =  500 ;
  MinKulcs = -9999 ;
  MaxKulcs =  9999 ;

Type
    { Ilyen rekordok tömbjében tároljuk az adatokat: }
  ElemTip    = Record
                 Kulcs : Integer ;
                 Egyeb : String[20] ;
               End ;
  TombTip    = Array [0..Elemszam] Of ElemTip ;

Var
  Tomb       : TombTip ;
  I          : Word ;
  Elso,
  Utolso,
  Kozepso    : Word ;
  Keresendo  : Integer ;
  Van        : Boolean ;
  IdoKezd,
  IdoVeg     : LongInt ;

  { A függvény a számítógép óráját átszámítja századmásodpercre: }
Function SzazadMasodPerc : LongInt ;
  Var
    Ora,
    Perc,
    MPerc,
    SzMPerc : Word ;
  Begin
    GetTime(Ora, Perc, MPerc, SzMPerc) ;
    SzazadMasodPerc := LongInt(1) * 100 * (60 * (60 * Ora +
                       Perc) + MPerc) + SzMPerc ;
  End ;

Procedure StopperBe ;
  Begin
    IdoKezd := SzazadMasodPerc ;
  End ;

Procedure StopperKi ;
  Begin
    IdoVeg := SzazadMasodPerc ;
  End ;
```

```
  { A függvény a két, századmásodpercekben megadott időpont között eltelt időt adja vissza
  másodpercekben: }
Function IdoMp : Real ;
  Begin
    IdoMp := (IdoVeg - IdoKezd) / 100 ;
  End ;

  { A függvény előállít adott két szám között egy véletlen számot: }
Function Veletlen(N, M: Integer): Integer ;
  Begin
    Veletlen := Random(M - N + 1) + N ;
  End ;

  { Az eljárás előállít egy rendezett tömböt: }
Procedure Tomb_General ;
  Var
    Lepes : Real ;
    I     : Word ;
  Begin
    Tomb[0].Kulcs := 0 ;
    Tomb[0].Egyeb := '' ;
      { Véletlenszám generátor inicializálása véletlen értékkel: }
    Randomize ;
      { A legnagyobb és a legkisebb kulcs közötti különbséget felosztjuk egyenletes intervallu-
      mokra: }
    Lepes := (MaxKulcs - MinKulcs) / Elemszam ;
      { Az első véletlen számot az első intervallumból vesszük: }
    Tomb[1].Kulcs := Veletlen(MinKulcs,Trunc(Minkulcs + Lepes)) ;
      { Az I. véletlen elem az előző elem és az I. intervallum vége között legyen: }
    For I := 2 To Elemszam Do
      Tomb[I].Kulcs := Veletlen(Tomb[I - 1].Kulcs,
                                Trunc(MinKulcs + Lepes * I));
      { Az Egyeb mezőt is kitöltjük, csak a formaság kedvéért: }
    For I := 1 To Elemszam Do
      Begin
        Str(Tomb[I].Kulcs,Tomb[I].Egyeb) ;
        Tomb[I].Egyeb := 'Elem ' + Tomb[I].Egyeb ;
      End ;
  End ;

  { A tömb kulcsainak kiírása : }
Procedure Kulcs_Kiir ;
  Var
    I : Word ;
  Begin
    ClrScr ;
```

```
    For I := 1 To Elemszam Do
      Begin
        If WhereY > 23 Then
          Begin
            WriteLn ;
            Write('Tovább <ENTER>') ;
            ReadKey ;
            ClrScr ;
          End ;
        Write(Tomb[I].Kulcs:5) ;
      End ;
    WriteLn ;
  End ;

  { FŐPROGRAM: }
Begin
  ClrScr ;
    { A rendezett tömb előállítása, majd kulcsainak kiírása: }
  Tomb_General ;
  Kulcs_Kiir ;

    { A keresendő kulcs bekérése: }
  Write('Kérem a keresendő kulcsot: ') ;
  ReadLn(Keresendo) ;

    { Lineáris keresés – az első kulcstól kezdve sorban végignézzük a kulcsokat. Ha megvan,
      vagy a tömb végéhez értünk, abbahagyjuk a keresést. Ezt a keresést általában csak
      rendezetlen, vagy kis elemszámú tömbökre érdemes használni: }
  WriteLn('Lineáris keresés: ') ;
    { Stopper indítása: }
  StopperBe ;
  I := 1 ;
    { Amíg van elem és nincs meg a keresett: }
  While (I < Elemszam) And (Keresendo <> Tomb[I].Kulcs) Do
    Inc(I) ;
  Van := Keresendo = Tomb[I].Kulcs ;
    { Stopper leállítása: }
  StopperKi ;
  If Van
  Then
      { Megvan a keresett elem: }
    With Tomb[I] Do
      WriteLn('Az ',I,'.',' Kulcs= ',Kulcs,'  Egyéb= ',Egyeb)
  Else
      { Nincs meg: }
    WriteLn('Nincs') ;
  WriteLn('Keresési idô: ',IdoMp:6:4,#13#10) ;
```

```
    { Bináris keresés – csak rendezett tömb esetén alkalmazható. A keresett kulcsot a tömb
      közepén lévő kulcshoz hasonlítjuk, s a keresést ettől függő en a tömb alsó vagy felső felében
      folytatjuk. A felezést addig folytatjuk, míg meg nincs az elem, vagy nem lehet tovább felezni: }
  WriteLn('Bináris keresés: ') ;

    { Stopper indítása: }
  StopperBe ;
  Elso := 1 ;
  Utolso := Elemszam ;
  Van := False ;
  Repeat

        { Középső elem kiválasztása: }
      Kozepso := (Elso + Utolso) Div 2 ;
      If Keresendo = Tomb[Kozepso].Kulcs
      Then

          { A keresett érték éppen a középső: }
        Van := True
      Else
        If Keresendo < Tomb[Kozepso].Kulcs
        Then

            { A keresett érték az alsó félben van: }
          Utolso := Kozepso - 1
        Else

            { A keresett érték a felső félben van: }
          Elso := Kozepso + 1 ;

    { Kilépés, ha megvan az érték vagy a vizsgált résztömb mérete nulla: }
  Until Van Or (Elso > Utolso) ;


    { Stopper leállítása: }
  StopperKi ;
  If Van
  Then
    With Tomb[I] Do
      WriteLn('Az ',I,'.',' Kulcs= ',Kulcs,'  Egyéb= ',Egyeb)
  Else
    WriteLn('Nincs') ;
  WriteLn('Keresési idô: ',IdoMp:6:4) ;
  Write('<ENTER>') ;
  ReadLn ;
End.
```

13. feladat

Írjunk programot, mely egy nevekből álló tömböt karbantart: a tömb név szerint rendezett, melybe új neveket szúrhatunk be, adott neveket kereshetünk meg illetve törölhetünk ki! Ezeket a funkciókat menüből lehessen kérni, a tömböt minden funkció után listázzuk ki!

```
Program Tomb13 ;

Uses
  Crt ;

Const
  MaxNev = 20 ;

Type
  St30 = String[30] ;
  St80 = String[80] ;

Var
  Nev          : St30 ;
  NevTomb      : Array [1..MaxNev] Of St30 ;
  NevekSzama,
  Index        : Word ;
  MenuKar      : Char ;

  { A név beolvasása: }
Procedure Nevolvas ;
  Begin
    GotoXY(40,22) ;
    ClrEol ;
    ReadLn(Nev) ;
  End ;

  { Hiba esetén információs szöveg kiírása, várakozás egy másodpercig, majd a szöveg törlése: }
Procedure HibaIr(Szoveg: St80) ;
  Begin
    GotoXY(35,24) ;
    Write(Szoveg) ;
    Delay(1000) ;
    GotoXY(35,24) ;
    ClrEol ;
  End ;
```

```
  { Név keresése a névtömbben. Ha megvan, a függvény értéke True lesz. Poz a keresett név
    pozíciója. Ha nincs meg a név, ezen a pozíción lenne a helye a rendezett tömbben: }
Function Van(Var Poz: Word): Boolean ;
  Var
    Stop : Boolean ;
  Begin
    Poz := 1 ;
    Stop := False ;
    While (Poz <= NevekSzama) And Not Stop Do
      If (Nev > NevTomb[Poz])
      Then
        Inc(Poz)
      Else
        Stop := True ;
    If Stop
    Then
      Van := Nev = NevTomb[Poz]
    Else
      Van := False ;
  End ;

  { Név beszúrása névtömbbe a Poz. helyre: }
Procedure Beszur(Poz: Word) ;
  Var
    I : Word ;
  Begin
      { A Poz. helytől feltoljuk a neveket, az új név a Poz. helyre kerül: }
    For I := NevekSzama DownTo Poz Do
      NevTomb[I + 1] := NevTomb[I] ;
    Inc(NevekSzama) ;
    NevTomb[Poz] := Nev ;
  End ;

  { A Poz. hely törlése a névtömbből: }
Procedure Torol(Poz: Word) ;
  Var
    I : Word ;
  Begin
    Dec(NevekSzama) ;
      { A Poz. elem törlése, a felette álló elemek eltolása lefelé: }
    For I := Poz To NevekSzama Do
      NevTomb[I] := NevTomb[I + 1] ;
  End ;

  { A névtömb listázása – képernyőtörlés nélkül: }
Procedure Lista ;
  Var
    I : Word ;
```

```
  Begin
    GotoXY(1,1) ;
    For I := 1 To NevekSzama Do
      WriteLn(NevTomb[I],'':30 - Length(NevTomb[I])) ;
    For I := NevekSzama + 1 To MaxNev Do
      WriteLn('':30) ;
  End ;

  { FŐPROGRAM: }
Begin
  NevekSzama := 0 ;
    { A képernyő szövegeinek kiírása: }
  ClrSCr ;
  GotoXY(35,12) ; Write('Beszúr.............B') ;
  GotoXY(35,14) ; Write('Töröl..............T') ;
  GotoXY(35,16) ; Write('Vége...............V') ;

  GotoXY(35,22) ; Write('Név:') ;
  Repeat
        { Választás a menüből: }
      GotoXY(55,16) ;
      MenuKar := Upcase(ReadKey) ;
      Case MenuKar Of
          { Beszúrás: }
        'B' : If NevekSzama = MaxNev
              Then
                    { Nincs már hely a tömbben: }
                HibaIr('Nincs több hely')
              Else
                Begin
                  Nevolvas ;
                  If Van(Index)
                  Then
                    HibaIr('Már van')
                  Else
                    Begin
                      Beszur(Index) ;
                      Lista ;
                    End ;
                End ;
          { Törlés: }
        'T' : If NevekSzama = 0
              Then
                    { A tömb üres: }
                HibaIr('Nincs egy név sem')
              Else
                Begin
                  Nevolvas ;
```

```
                    If Van(Index)
                    Then
                      Begin
                        Torol(Index) ;
                        Lista ;
                      End
                    Else
                      HibaIr('Nincs ilyen név') ;
                  End ;
      End ;
    { Kilépés a 'V' billentyű leütésére: }
  Until MenuKar = 'V' ;
  ClrScr ;
End.
```

14. feladat

Írjunk beviteli rutint, mely adott képernyőpozícióról, adott hosszúságban beolvas egy karakterláncot! Paraméterben adjuk meg továbbá azon karakterek halmazát, melyek a karakterlánc bevitelekor engedélyezettek, valamint azokét, melyekkel a bevitel befejeződhet! Az eljárás adja vissza azt a karaktert, mellyel a bevitel ténylegesen befejeződött! A beviteli mező szerkeszthető legyen, vagyis használhassuk a kurzorvezérlő, törlő billentyű ket! Az <INS> gombbal a beszúró illetve felülíró mód között lehessen váltani! Definiáljunk három mezőt a képernyőn, és a fenti bevivő rutinnal vigyük be az adatokat úgy, hogy a mezők között a fel, le nyíllal, valamint az <ENTER>–rel lehessen váltani! <ESC>–re fejeződjék be a bevitel!

```
Program Tomb14 ;

Uses
  Crt ;

Const
  Beszuro_Mod : Boolean = True ;

  BS        = #8 ;
  Enter     = #13 ;
  ESC       = #27 ;
  SPC       = ' ' ;

    { A következő billentyűk kettős kóddal rendelkeznek. A BillKod eljárás azonban a kettős
      kódokat egy trükkel szimplává változtatja. Természetesen ekkor e magas kódok eredeti
      jelentése (pl.: î,û,...) megszűnik, azokat tovább nem használhatjuk. }
  UpKey     = #200 ;
  DownKey   = #208 ;
  LeftKey   = #203 ;
  RightKey  = #205 ;
```

```
  HomeKey   = #199 ;
  EndKey    = #207 ;
  InsKey    = #210 ;
  DelKey    = #211 ;

Type
  Karakterhalmaz = Set Of Char ;
```

{ Egy beviteli mező jellemzői: kezdési pozíciója a képernyőn, hossza, tartalma, és a bevihető karakterek halmaza: }

```
  Mezotip         = Record
                      X,Y, Hossz     : Byte ;
                      Tartalom       : String ;
                      Engedelyezett  : Karakterhalmaz ;
                    End ;

Const
```

{ Három mező definiálása a képernyőn: }

```
  Mezo : Array[1..3] Of Mezotip =
         ((X: 20 ; Y: 12 ; Hossz: 20 ; Tartalom: '' ;
           Engedelyezett: ['a'..'z','A'..'Z']),
          (X: 20 ; Y: 14 ; Hossz: 6 ; Tartalom: '' ;
           Engedelyezett: ['0'..'9']),
          (X: 28 ; Y: 14 ; Hossz: 1 ; Tartalom: '' ;
           Engedelyezett: ['$','£','%'])) ;

Var
  MezoInd      : Byte ;
  BefKar       : Char ;
```

{ A billentyűzet puffer ürítése: }

```
Procedure PufferUrit ;
  Var
    Ch : Char ;
  Begin
```

{ Amíg van várakozó karakter a billentyűzet pufferben: }

```
    While KeyPressed Do
```

{ Karakter kiolvasása: }

```
      Ch := Readkey ;
  End ;
```

{ Az eljárás a kettős kódú billentyűk két kódjából egyet készít úgy, hogy a 128–nál kisebb második kódhoz hozzáad 128–at: }

```
Function BillKod : Char ;
  Var
    Ch : char ;
  Begin
    Ch := ReadKey ;
```

```
    If (Ch = #0) And KeyPressed Then
      Begin
        Ch := ReadKey ;
        If Ord(Ch) < 128 Then
          Inc(Ch,128) ;
      End ;
    BillKod := Ch ;
  End ;
```

{ Hangjelzés hiba esetén: }

```
Procedure Hibahang ;
  Begin
    Sound(440) ;
    Delay(40) ;
    Nosound ;
  End ;
```

{ Az eljárás segítségével szerkeszthető az S karakterlánc, mely a képernyő (X,Y) pozíciójától látható, és maximum Hossz hosszúságú. A mezőbe a leütött karakterek közül csak a Bevihetok kerülhetnek. Ha a Befejezok közül ütünk le egy befejező karaktert, akkor a bevitel befejeződik. A rutin a billentyűzetről olvas, karbantartja az S karakterláncot, és a képernyőt folyamatosan aktualizálja. }

```
Procedure Bevitel(Var S                    : String ;
                      X,Y,Hossz            : Byte ;
                      Bevihetok,Befejezok: Karakterhalmaz ;
                  Var Befejezo             : Char) ;
  Var
    Poz : Byte ;
    Kar : Char ;
  Begin
    GotoXY(X,Y) ;
    Write(S,'':Hossz - Length(S)) ;
    Poz := 0 ;
    Repeat
        GotoXY(X + Poz,Y) ;
        PufferUrit ;
        Kar := BillKod ;
        If Not (Kar In Befejezok) Then
          Case Kar Of
              { Nem vezérlő karakter: }
            #32..#186 : If (Poz < Hossz) And (Kar In Bevihetok)
                          { Nem a mező vége, és bevihető a karakter: }
                        Then
                          Begin
                            Inc(Poz) ;
                            If Beszuro_Mod
                            Then
                              Begin
                                If Length(S) = Hossz Then
                                  Delete(S,Hossz,1) ;
```

```
                        Insert(Kar,S,Poz) ;
                        Write(Copy(S,Poz,Hossz)) ;
                      End
                    Else
                      Begin
                        If Poz > Length(S)
                        Then
                          S := S + Kar
                        Else
                          S[Poz] := Kar ;
                        Write(Kar) ;
                      End ;
                  End
                Else
                  Hibahang ;
    { Balra: }
^S,
LeftKey    : If Poz > 0
             Then
               Dec(Poz)
             Else
               Hibahang ;
    { Jobbra: }
^D,
RightKey   : If Poz < Length(S)
             Then
               Inc(Poz)
             Else
               Hibahang ;
    { Mező elejére: }
^A,
HomeKey    : Poz := 0 ;
    { Mező végére: }
^F,
EndKey     : Poz := Length(S) ;
    { Karakter törlése: }
^G,
DelKey     : If Poz < Length(S)
             Then
               Begin
                 Delete(S,Poz + 1,1) ;
                 Write(Copy(S,Poz + 1,Hossz),' ') ;
               End
             Else
               Hibahang ;
```

```
                { Előző karakter törlése: }
            BS          : If Poz > 0
                          Then
                            Begin
                              Delete(S,Poz,1) ;
                              GotoXY(X + Poz - 1,Y) ;
                              Write(Copy(S,Poz,Hossz),' ') ;
                              Dec(Poz) ;
                            End
                          Else
                            Hibahang ;
                { Törlés a kurzortól a mező végéig: }
            ^Y          : Begin
                            Write('':Hossz-Poz) ;
                            Delete(S,Poz + 1,Hossz) ;
                          End ;
            InsKey      : Beszuro_Mod := Not Beszuro_Mod ;
          End ;
    Until Kar In Befejezok ;
    Befejezo := Kar ;
  End ;

  { FŐPROGRAM: }
Begin
    { A teljes képernyő legyen kék: }
  TextBackground(Blue) ;
  ClrScr ;
    { A szerkesztési mezők világosszürkék, a karakterek feketék: }
  TextBackground(Lightgray) ;
  TextColor(Black) ;
    { A mezőkhöz tartozó szerkesztési „ablakok" megjelenítése: }
  For MezoInd := 1 To 3 Do
    With Mezo[MezoInd] Do
      Begin
        GotoXY(X,Y) ;
        Write(SPC:Hossz) ;
      End ;

    { Mezők szerkesztése. Az első szerkesztendő mező az 1., a következő szerkesztendő mező
      pedig a mező szerkesztését befejező karaktertől függ. <ESC>-re befejeződik a mezők
      szerkesztése. }
  MezoInd := 1 ;
  Repeat
        { Az aktuális mező szerkesztése: }
      With Mezo[MezoInd] Do
        Bevitel(Tartalom,X,Y,Hossz,Engedelyezett,
                [Enter,UpKey,DownKey,ESC],BefKar) ;
```

```
        { BefKar–tól függően a következő szerkesztendő mező: }
      Case BefKar Of
          { A mező feletti, ha az nem a legfelső: }
        UpKey    : If MezoInd > 1 Then
                     Dec(MezoInd) ;
          { A mező alatti, legalsó esetén az első: }
        Enter,
        DownKey : If MezoInd < 3
                  Then
                    Inc(MezoInd)
                  Else
                    MezoInd := 1 ;
      End ;
  Until BefKar = ESC ;
  TextBackground(Black) ;
  TextColor(LightGray) ;
  ClrScr ;
End.
```

15. feladat

Olvassunk be pozitív, maximum 5 jegyű egész számokat 0 végjelig! Állapítsuk meg, hány 1,2,3,4 vagy 5 jegyű szám volt köztük, és azon belül hány végződött 0–ra, 1–re ... 9–re!

```
Program Tomb15 ;

Uses
  Crt ;

Var
  Db          : Array [1..5] Of Array [0..9] Of Byte ;
  Szam        : LongInt ;
  Puffer      : Array [1..7] Of Byte ;

  { Az eljárás beolvas egy maximum 5 jegyű pozitív számot: }
Procedure SzamOlvas ;
  Var
    OK      : Boolean ;
    SzamSt  : String[5] ;
    Kod     : Integer ;
  Begin
    WriteLn ;
    Repeat
        GotoXY(3,WhereY - 1) ;
        ClrEol ;
        ReadLn(SzamSt) ;
```

```
        Val(SzamSt,Szam,Kod) ;
        OK := (Kod = 0) And (Szam >= 0) ;
        If Not OK Then
          Write(#7) ;
    Until OK ;
  End ;
```

{ Az eljárás meghatározza a szám végződését és jegyeinek számát, s azok alapján növeli a táblázat megfelelő elemét: }

```
Procedure SzamFeldolgoz ;
  Var
    Sz : LongInt ;
    Vegzodes     : 0..9 ;
    Jegyekszama : 0..5 ;
  Begin
    Vegzodes := Szam Mod 10 ;
    Sz := Szam ;
    Jegyekszama := 0 ;
    Repeat
        Sz := Sz Div 10 ;
        Inc(Jegyekszama) ;
    Until Sz = 0 ;

    Inc(Db[Jegyekszama,Vegzodes]) ;
  End ;
```

{ Az eljárás az (X,Y+1) képernyőpozíciótól lefelé függőlegesen kiír egy szöveget: }

```
Procedure FuggIr(X, Y: Byte ; Szoveg: String) ;
  Var
    I : Byte ;
  Begin
    For I := 1 To Length(Szoveg) Do
      Begin
        GotoXY(X,Y + I) ;
        Write(Szoveg[I]) ;
      End ;
  End ;

Procedure TablazatKeszit ;
  Var
    Vegzodes     : 0..9 ;
    Jegyekszama : 0..5 ;
  Begin
      { A táblázat szövegeinek kiírása: }
    TextColor(Yellow) ;
    GotoXY(48,5) ;
    Write('V é g z ô d é s') ;
    GotoXY(40,6) ;
    For Vegzodes := 0 To 9 Do
      Write(Vegzodes:3) ;
```

```
    FuggIr(33,6,'Jegyek száma') ;
    For Jegyekszama := 1 To 5 Do
      Begin
        GotoXY(38,6 + 2 * Jegyekszama) ;
        Write(Jegyekszama) ;
      End ;

      { A gyűjtött adatok kiírása: }
    TextColor(Black) ;
    For Jegyekszama := 1 To 5 Do
      Begin
        GotoXY(40,6 + 2 * Jegyekszama) ;
        For Vegzodes := 0 To 9 Do
          Write(Db[Jegyekszama,Vegzodes]:3) ;
      End ;
  End ;
  { FŐPROGRAM: }
Begin
  TextBackground(LightGray) ;
  TextColor(Blue) ;
  ClrScr ;

    { A tömb nullázása: }
  FillChar(Db,SizeOf(Db),#0) ;

    { Adatok gyűjtése: }
  WriteLn('Kérem a számokat (0 = vége):') ;
  SetTextBuf(Input,Puffer) ;
  SzamOlvas ;
    { Amíg nem nullát ütnek be: }
  While Szam <> 0 Do
    Begin
      SzamFeldolgoz ;
      SzamOlvas ;
    End ;

  TablazatKeszit ;
  ReadLn ;
  TextBackground(Black) ;
  TextColor(LightGray) ;
  ClrScr ;
End.
```

16. feladat

Egy fesztiválra különböző országokból érkeznek diákok, akik 1970 és 1980 között születtek. Az országok neveit előre nem tudjuk. Írjuk be sorban a fesztiválra érkezettek nevét, születési évét, és az ország nevét, ahonnan érkeztek! Írjuk fel az adatokat egy állományba (ha volt már, akkor bővítsük azt)! Készítsünk kimutatást országonként és azon belül születési évenként a résztvevők létszámáról! A gyűjtött értékeket táblázat formájában jelenítsük meg a képernyőn (az országok ABC sorrendben legyenek)! Írjuk ki, hogy hányan vettek részt a fesztiválon összesen országonként és születési évenként, valamint a teljes létszámot!

```
Program Tomb16 ;

Uses
  Crt ;

Const
  MaxOrszag = 20 ;
  Tol       = 1970 ;
  Ig        = 1980 ;

Type
  St15          = String[15] ;
    { Ilyen rekordokban tároljuk az adatokat: }
  ResztvevoTip = Record
                   Nev       : String[25] ;
                   SzulEv    : Word ;
                   OrszagNev : St15 ;
                 End ;

Var
  Letszam     : Array[1..MaxOrszag,Tol..Ig] Of Word ;
  OrszagNevek : Array[1..MaxOrszag] Of St15 ;
  OrszagSzam  : Byte ;
  Resztvevok  : File Of ResztvevoTip ;
  Resztvevo   : ResztvevoTip ;

  { Új résztvevők adatainak felvitele: }
Procedure Ujresztvevok ;
  Var
    Ev : LongInt ;
    Y  : Byte ;
  Begin
      { Állomány megnyitása: }
    {$I-}
    Reset(Resztvevok) ;
    {$I+}
```

```
    { Ha sikerült, pozícionálás az állomány végére, egyébként új állományt hozunk létre: }
  If IOResult = 0
  Then
    Seek(Resztvevok,FileSize(Resztvevok))
  Else
    Rewrite(Resztvevok) ;

  ClrScr ;
  With Resztvevo Do
    Begin
        { Első résztvevő nevének bekérése: }
      Write('Név                    : ') ;
      ReadLn(Nev) ;
        { Amíg névként nem üres karakterláncot ütnek: }
      While Nev <> '' Do
        Begin
            { A résztvevő többi adatának bekérése: }
          {$I-}
          Write('Szül.év(',Tol,'-',Ig,'): ') ;
          Y := WhereY ;
          Repeat
              GotoXY(20,Y) ;
              ClrEol ;
              ReadLn(Ev) ;
            { Amíg jó születési évet nem ütnek be: }
          Until (IOResult = 0) And (Ev >= Tol) And (Ev <= Ig) ;
          {$I+}
          SzulEv := Ev ;
          Write('Országnév            : ') ;
          ReadLn(OrszagNev) ;
            { Egy résztvevő adatainak felvitele az állományba: }
          Write(Resztvevok,Resztvevo) ;
            { Következő résztvevő nevének bekérése: }
          WriteLn ;
          Write('Név                  : ') ;
          ReadLn(Nev) ;
        End ;
    End ;
      { Állomány zárása: }
  Close(Resztvevok) ;
End ;

{ A függvény megállapítja, hogy van–e már egy adott országnév a rendezett Orszagnevek
  tömbben. Ha van, megadja annak helyét – ha nincs, megadja azt a pozíciót, ahova be kellene
  azt szúrni. }
Function Letezik(Onev: St15 ; Var Pozicio: Byte): Boolean ;
  Begin
    Pozicio := 1 ;
```

```
      { Boolean evaluation (feltételek teljes kiértékelése) fordítási direktíva kikapcsolása – ha
        Pozicio = Orszagszam + 1, akkor a logikai kifejezés második felét nem szabad
        kiértékelni: }
    {$B-}
    While (Pozicio <= OrszagSzam) And
          (Onev > OrszagNevek[Pozicio]) Do
      Inc(Pozicio) ;
    Letezik := (Pozicio <= OrszagSzam) And
               (Onev = OrszagNevek[Pozicio]) ;
    {$B+}
  End ;

  { Az állomány végigolvasása elölről, adatok gyűjtése: }
Procedure Gyujtes ;
  Var
    I,
    N : Byte ;
    J : Word ;
  Begin
      { Állomány megnyitása: }
    {$I-}
    Reset(Resztvevok) ;
    {$I+}
      { Ha nem sikerült, kilépés az eljárásból: }
    If IOResult <> 0 Then
      Exit ;
      { Tömb feltöltése nullákkal: }
    FillChar(Letszam, SizeOf(Letszam),#0) ;
    OrszagSzam := 0 ;
      { Amíg nincs vége az állománynak: }
    While Not Eof(Resztvevok) Do
      With Resztvevo Do
        Begin
            { Egy résztvevő adatainak beolvasása az állományból: }
          Read(Resztvevok,Resztvevo) ;
          If Letezik(OrszagNev,N)
          Then
              { Már van ilyen országnév, növeljük a megfelelő tömbelemet: }
            Inc(Letszam[N,SzulEv])
          Else
              { Nincs, előbb beszúrjuk azt az OrszagNevek tömbbe, valamint a Letszam
                kétdimenziós tömbbe is beszúrunk egy sort az új ország számára: }
            Begin
              If (OrszagSzam < MaxOrszag) Then
                Begin
                  For I := OrszagSzam DownTo N Do
                    Begin
                      OrszagNevek[I + 1] := OrszagNevek[I] ;
```

```
                    Letszam[I + 1] := Letszam[I] ;
                  End ;
                OrszagNevek[N] := OrszagNev ;
                    { Az új sor feltöltése nullákkal: }
                For J := Tol To Ig Do
                  Letszam[N,J] := 0 ;
                Inc(OrszagSzam) ;
                Inc(Letszam[N,SzulEv]) ;
              End ;
          End ;
      End ;
    { Állomány zárása: }
  Close(Resztvevok) ;
End ;

{ Táblázat megjelenítése a képernyőn: }
Procedure Kimutatas ;
  Var
    I          : Byte ;
    J,
    Osszes,
    VegOsszes : Word ;
  Begin
    ClrScr ;
    GotoXY(18,3) ;
    For J := Tol To Ig Do
      Write(J:5) ;
    Write('    Össz') ;
    For I := 1 To OrszagSzam Do
      Begin
        GotoXY(2,I + 4) ;
        Write(OrszagNevek[I]) ;
        Osszes := 0 ;
        GotoXY(17,WhereY) ;
        For J := Tol To Ig Do
          Begin
            Write(Letszam[I,J]:5) ;
            Inc(Osszes,Letszam[I,J]) ;
          End ;
        Write(Osszes:8) ;
      End ;
    GotoXY(18,5 + OrszagSzam) ;
    For I := 1 To 63 Do
      Write('-') ;
    VegOsszes := 0 ;
    GotoXY(17,6 + OrszagSzam) ;
    For J := Tol To Ig Do
      Begin
        Osszes := 0 ;
```

```
        For I := 1 To OrszagSzam Do
          Inc(Osszes,Letszam[I,J]) ;
        Write(Osszes:5) ;
        Inc(VegOsszes,Osszes) ;
      End ;
    Write(VegOsszes:8) ;
    ReadLn ;
  End ;

  { FŐPROGRAM: }
Begin
  Assign(Resztvevok,'Resztv.Dat') ;
  Ujresztvevok ;
  Gyujtes ;
  Kimutatas ;
End.
```

17. feladat

Egy országhatáron az országot elhagyók statisztikai adatait (állampolgárság, kor, nem, tartózkodási hetek száma) évenként számítógépen rögzítik. Minden adatbevitel után készítsük el a következő statisztikai kimutatásokat:

- **hány angol, német, stb. állampolgárnő lépte át a határt, s azok átlagban hány hetet tartózkodtak itt?**
- **a határt átlépők korcsoport szerinti megoszlása összesen és százalékosan (Korcsorortok: 0–9 év ; 10–19 év ; stb. ; 90–99).**
- **hány 40 és 49 év közötti holland férfi lépte át a határt?**

```
Program Tomb17 ;

Uses
  Crt ;

Const
  Maxhet = 8 ;

Type
  St10              = String[10] ;
  Allampolgarsagok  = (Angol, Belga, Francia, Holland,
                       Magyar, Nemet, Olasz, Osztrak) ;
  Korcsoportok      = 0..9 ;
  Hetek             = 0..Maxhet ;
  Nemek             = (Ferfi,No) ;
  StatTip           = Array[Allampolgarsagok,Korcsoportok,
                            Nemek,Hetek] Of Word ;
```

```
Const
  AllampolgarsagokSt : Array [Allampolgarsagok] Of St10 =
    ('Angol','Belga','Francia','Holland',
     'Magyar','Nemet','Olasz','Osztrak') ;

Var
  Stat            : StatTip ;
  StatMent        : File Of StatTip ;
  Nem             : Nemek ;
  Kor,
  TartHet         : Byte ;
  Allampolgarsag : Allampolgarsagok ;
  Vege            : Boolean ;

  { Az eljárás információt ad a bevihető adatokról: }
Procedure Informacio ;
  Var
    A : Allampolgarsagok ;
  Begin
    Write('Állampolgárságok: ') ;
    For A := Angol To Osztrak Do
      Write(AllampolgarsagokSt[A],' ') ;
    WriteLn ;
    WriteLn('Kor: 0-99') ;
    WriteLn('Nemek: No, Ferfi') ;
    WriteLn('Tartozkodási hét: 0-',MaxHet) ;
    WriteLn ;
  End ;

  { A függvény a karakterlánc első betűjét nagyra, a többit pedig kicsire alakítja: }
Function NagyKis(S: String) : String ;
  Var
    I : Byte ;
  Begin
    S[1] := Upcase(S[1]) ;
    For I := 2 To Length(S) Do
      If S[I] In ['A'..'Z'] Then
        S[I] := Chr(Ord(S[I]) + 32) ;
    NagyKis := S ;
  End ;

  { A függvény visszaad egy számot a megadott két határérték között: }
Function Szam(Tol, Ig: LongInt) : LongInt ;
  Var
    X,
    Y : Byte ;
    L : LongInt ;
  Begin
    X := WhereX ;
```

```
    Y := WhereY ;
    {$I-}
    Repeat
        GotoXY(X,Y) ;
        ClrEol ;
        ReadLn(L) ;
    Until (IOResult = 0) And (L >= Tol) And (L <= Ig) ;
    {$I+}
    Szam := L ;
  End ;
```

{ A függvény visszad egy Nemek típusú értéket a beolvasott karakterlánctól függően:}

```
Function NemBe : Nemek ;
  Var
    X,
    Y  : Byte ;
    St : St10 ;
  Begin
    X := WhereX ;
    Y := WhereY ;
    Repeat
        GotoXY(X,Y) ;
        ClrEol ;
        ReadLn(St) ;
        St := NagyKis(St) ;
    Until (St = 'Ferfi') Or (St = 'No') ;
    If St = 'Ferfi'
    Then
      NemBe := Ferfi
    Else
      NemBe := No ;
  End;
```

{ A függvény visszad egy Allampolgarsagok típusú értéket a beolvasott karakterlánctól függően. Az üres karakterlánc a bevitel végét fogja jelenteni:}

```
Function AllamPolgBe : Allampolgarsagok ;
  Var
    X,
    Y  : Byte ;
    St : St10 ;
    A  : Allampolgarsagok ;
  Begin
    X := WhereX ;
    Y := WhereY ;
    Repeat
        GotoXY(X,Y) ;
        ClrEol ;
        ReadLn(St) ;
        St := NagyKis(St) ;
        A := Angol ;
```

```
      While (St <> AllampolgarsagokSt[A]) And (A < Osztrak) Do
        Inc(A) ;
      Vege := St = '' ;
    Until (St = AllampolgarsagokSt[A]) Or Vege;
    AllampolgBe := A ;
  End ;
```

{ A függvény meghatározza a kornak megfelelő korcsoportot. Korcsoportok: 0–9 év; 10–19 év; stb. 90–99 év: }

```
Function Korcsoport(Kor: Byte): Korcsoportok ;
  Begin
    Korcsoport := Kor Div 10 ;
  End ;
```

{ 1. statisztikai kimutatás: hány angol, német, stb. állampolgárnő lépte át a határt, s azok átlagban hány hetet tartózkodtak itt: }

```
Procedure Statisztika1 ;
  Var
    OsszPolg,
    OsszHet         : Word ;
    Allampolgarsag : Allampolgarsagok ;
    Korcsoport      : Korcsoportok ;
    Atlag           : Real ;
  Begin
    WriteLn('A különbozô nôi állampolgárok átlag tartózkodási '+
            'ideje: ') ;
    For Allampolgarsag := Angol To Osztrak Do
      Begin
        OsszPolg := 0 ;
        OsszHet := 0 ;
        For Korcsoport := 0 To 9 Do
          For TartHet := 0 to Maxhet Do
            Begin
              Inc(OsszPolg,
                  Stat[Allampolgarsag,Korcsoport,No,TartHet]) ;
              Inc(OsszHet,
                  Stat[Allampolgarsag,Korcsoport,No,TartHet] *
                  TartHet) ;
            End ;
        Write(AllampolgarsagokSt[Allampolgarsag]) ;
        GotoXY(10,WhereY) ;
        Write(': ') ;
        If OsszPolg > 0
        Then
          Atlag := OsszHet / OsszPolg
        Else
          Atlag := 0 ;
        WriteLn(Atlag:3:1,'  hét') ;
      End ;
    WriteLn ;
  End ;
```

```
  { 2. statisztikai kimutatás: a határt átlépők korcsoport szerinti megoszlása összesen és
    százalékosan: }
Procedure Statisztika2 ;
  Var
    Gyujto      : Array[Korcsoportok] Of Word ;
    Korcsoport  : Korcsoportok ;
    Ossz        : Word ;
    Szazal      : Real ;
  Begin
    Ossz := 0 ;
    For Korcsoport := 0 To 9 Do
      Begin
        Gyujto[Korcsoport] := 0 ;
        For Allampolgarsag := Angol To Osztrak Do
          For Nem := Ferfi To No Do
            For TartHet := 0 To Maxhet Do
              Inc(Gyujto[Korcsoport],
                  Stat[Allampolgarsag,Korcsoport,Nem,TartHet]) ;
        Inc(Ossz,Gyujto[Korcsoport]) ;
      End ;
    WriteLn('Korcsoport szerinti százalékos megoszlás:') ;
    For Korcsoport := 0 To 9 Do
      Begin
        If Ossz > 0
        Then
          Szazal := Gyujto[Korcsoport] * 100 / Ossz
        Else
          Szazal := 0 ;
        WriteLn(10 * Korcsoport:2,'-',10 * Korcsoport + 9:2,
                ': ',Szazal:6:2,' %') ;
      End ;
    WriteLn ;
  End ;

  { 3. statisztikai kimutatás: hány 40 és 49 év közötti holland férfi lépte át a határt: }
Procedure Statisztika3 ;
  Var
    Allampolgarsag : Allampolgarsagok ;
    Nem        : Nemek ;
    TartHet    : Hetek ;
    Osszes     : Word ;
  Begin
    Osszes := 0 ;
    For TartHet := 0 To Maxhet Do
      Inc(Osszes,Stat[Holland,4,Ferfi,TartHet]) ;
    WriteLn('A határt átlépô 40-49 éves holland férfiak',
            ' száma: ',Osszes) ;
  End ;
```

```
  { FŐPROGRAM: }
Begin
  ClrScr ;
    { Statisztikai adatok beolvasása: Ha nincs még adat, nullázzuk a tömböt: }
  Assign(StatMent,'Stat.Dat') ;
  {$I-}
  Reset(StatMent) ;
  {$I+}
  If IOResult = 0
  Then
    Begin
      Read(StatMent,Stat) ;
      Close(StatMent) ;
    End
  Else
    FillChar(Stat,SizeOf(Stat),#0) ;

    { Adatok (állampolgárság, kor, nem, tartózkodási idő) gyűjtése a statisztikai tömbbe: }
  Informacio ;
  Window(10,10,70,25) ;
  Write('Állampolgárság    : ') ;
  Allampolgarsag := AllampolgBe ;
  Vege := False ;

    { Amíg az állampolgárságnál nem üres karakterláncot ütnek be, nincs vége a bevitelnek: }
  While Not Vege Do
    Begin
      Write('Kor               : ') ;
      Kor := Szam(0,99) ;
      Write('Nem               : ') ;
      Nem := NemBe ;
      Write('Tartózkodási idô : ') ;
      TartHet := Szam(0,MaxHet) ;

        { A statisztikai tömb megfelelő elemének növelése: }
      Inc(Stat[Allampolgarsag,Korcsoport(Kor),Nem,TartHet]) ;
      WriteLn ;

      ClrScr ;
      Write('Állampolgárság    : ') ;
      Allampolgarsag := AllampolgBe ;
    End ;

    { Statisztikai tömb elmentése: }
  Rewrite(StatMent) ;
  Write(StatMent,Stat) ;
  Close(StatMent) ;
```

```
    { Statisztikák készítése: }
  Window(1,1,80,25) ;
  ClrScr ;
  Statisztika1 ;
  Statisztika2 ;
  Statisztika3 ;
  ReadLn ;
End.
```

SZÖVEGES ÁLLOMÁNYOK

1. feladat

Olvassunk be karakterláncokat (sorokat), és írjuk ki azokat egy 'Szoveg.Txt' szöveges állományba! Sor = '*' jelezze a bevitel végét, ekkor olvassuk vissza a szöveget!

```
Program Szoveg1 ;

Var
    { Szöveges állomány típusú változó. Helyfoglalása 256 bájt, melyet a rendszer kezel, és a
      hozzárendelt fizikai állomány kezeléséhez szükséges. Állományjellemzőket és az átmeneti
      puffert tartalmazza: }
  Szoveg : Text ;
  Sor    : String ;

  { FŐPROGRAM: }
Begin
    { Szoveg logikai állomány hozzárendelése a 'Szoveg.Txt' fizikai állományhoz (az aktuális
      alkönyvtár tartalomjegyzékében ilyen néven fog szerepelni): }
  Assign(Szoveg,'Szoveg.Txt') ;
    { A Szoveg–hez előzőleg hozzárendelt fizikai állomány megnyitása írásra. Ha már volt ilyen
      nevű állomány az aktuális alkönyvtárban, akkor az törlődik: }
  Rewrite(Szoveg) ;

    { Sorok bevitele, felírásuk a szöveges állományba: }
  WriteLn('Kérem a sorokat: ') ;
  ReadLn(Sor) ;
    { Amíg nem a '*'–ot ütik: }
  While Sor <> '*' Do
    Begin
        { A beolvasott sor felírása a szöveges állományba: }
      WriteLn(Szoveg,Sor) ;
      ReadLn(Sor) ;
    End ;
    { A szöveges állomány lezárása: }
  Close(Szoveg) ;

  WriteLn('———— SZOVEG.TXT állomány: ————') ;
    { A szöveges állomány megnyitása olvasásra: }
  Reset(Szoveg) ;
    { A szöveg végigolvasása soronként, a sorok kiírása képernyőre: }
  While Not Eof(Szoveg) Do
    Begin
      ReadLn(Szoveg,Sor) ;
```

```
      WriteLn(Sor) ;
    End ;
  Close(Szoveg) ;
  WriteLn('———————— VÉGE ————————————') ;
  ReadLn ;
End.
```

2. feladat

Olvassunk be karakterenként egy adott könyvtárban lévő Pascal programot, és írjuk ki a képernyőre lassítva!

```
Program Szoveg2 ;

{$X+}

Uses
  Crt ;

Var
   PasProg     : Text ;
   AllSpec,
   Utvonal     : String ;
   PasProgNev  : String[12] ;
   Kar         : Char ;

  { FŐPROGRAM: }
Begin
    { A könyvtár és a program nevének beolvasása, hozzárendelés: }
  ClrScr ;
  Write('Könyvtár neve: ') ;
  ReadLn(Utvonal) ;
  Write('A Pascal program neve (kiterjesztés nélkül): ') ;
  ReadLn(PasProgNev) ;
  AllSpec := Utvonal + '\' + PasProgNev + '.Pas' ;
  Assign(PasProg,AllSpec) ;
    { A PasProg állomány megnyitása olvasásra. A beviteli/kiviteli műveletek ellenőrzését
      kikapcsoljuk, különben futási hibával leáll a program, ha nincs meg az állomány. Ilyenkor
      az IOResult függvénnyel le kell kérdeznünk a művelet sikerességét: }
  {$I-}
  Reset(PasProg) ;
  {$I+}
  If IOResult <> 0
  Then
    WriteLn(Utvonal,'-ban nincs ',PasProgNev,' program!')
```

```
  Else
    Begin
        { PasProg beolvasása karakterenként, kiírása a képernyőre lassítva, amíg nincs vége
        az állománynak.}
      While Not Eof(PasProg) Do
        Begin
          Read(PasProg,Kar) ;

            { Ha leütöttek egy billenyűt, addig nem folytatjuk, amíg még egyet le nem
            ütnek: }
          If KeyPressed Then
            Begin
              ReadKey ;
              ReadKey ;
            End ;

          Write(Kar) ;
          Delay(10) ;
        End ;
      Close(PasProg) ;
    End ;
  ReadLn ;
End.
```

3. feladat

Olvassunk be karakterláncokat addig, amíg '*' karaktert nem ütnek be! Vigyük fel ezeket a karakterláncokat az aktuális könyvtár 'Szoveg.Txt' állományába! Ha az állomány már létezett, akkor írjuk ki az eddigi tartalmát, majd bővítsük azt!

```
Program Szoveg3 ;

Var
  Szoveg : Text ;
  Lanc   : String ;

  { FŐPROGRAM: }
Begin
    { Szoveg hozzárendelése a 'Szoveg.Txt' állományhoz: }
  Assign(Szoveg,'Szoveg.Txt') ;
  {$I-}
  Reset(Szoveg) ;
  {$I+}
```

```
  If IOResult = 0
  Then
       { 'Szoveg.Txt' már létezik az aktuális könyvtárban: }
    Begin
         { A már eddig is létezett 'Szoveg.Txt' állomány kiírása a képernyőre: }
      WriteLn('A SZOVEG.TXT ÁLLOMÁNY TARTALMA:') ;
      While Not Eof(Szoveg) Do
        Begin
          ReadLn(Szoveg,Lanc) ;
          WriteLn(Lanc) ;
        End ;
         { Állomány megnyitása bővítésre: }
      Append(Szoveg) ;
    End
  Else
       { Nem létezik: }
    Begin
      WriteLn('SZOVEG.TXT EDDIG MÉG NEM VOLT') ;
         { Az állomány létrehozása: }
      Rewrite(Szoveg) ;
    End ;

    { Az állomány bővítése: }
  WriteLn('Irja be a szöveget:  (Kilépés = <*>)') ;
  ReadLn(Lanc) ;
  While Lanc <> '*' Do
    Begin
      WriteLn(Szoveg,Lanc) ;
      ReadLn(Lanc) ;
    End ;

  Close(Szoveg) ;
End.
```

4. feladat

Olvassunk be neveket és azokhoz tartozó életkorokat! Irányítsuk át az Output–ot, vagyis a képernyőt egy 'Kepernyo.Id' szöveges állományba, s a beolvasott adatokat sorszámmal ellátva írjuk folyamatosan a „képernyőre" soronként! Végül az Output–ot rendeljük újra a képernyőhöz, és olvassuk vissza az elmentett képernyőt!

```
Program Szoveg4 ;

Uses
  Crt ;
```

```
Var
  KepernyoMent : Text ;
  Sor          : String ;
  Nev          : String[20] ;
  I,
  Kor          : Word ;

  { FŐPROGRAM: }
Begin
    { Az Output átirányítása után a Write nem a képernyőre fog írni, így előre kiírjuk az
    információs szöveget: }
  ClrScr ;
  WriteLn('Irjon be folyamatosan neveket és korokat ' +
          'Név = ''''-ig:') ;
  WriteLn('Név                    Kor') ;

    { A szabványos Output szöveges állományhoz a rendszer a 'CON' perifériát rendeli, s
    megnyitja azt írásra. Ha használjuk a Crt egységet, akkor annak inicializáló része Output–ot
    a gyorsabb (az egységben definiált) 'CRT' perifériához rendeli. Az Output újabb átirányí-
    tásával a rendszert felülbíráljuk: }
  Close(Output) ;
  Assign(Output,'Kepernyo.Id') ;
  Rewrite(Output) ;

    { Az adatok beolvasása, és kiírása a „képernyőre" (az olvasások elé nem tudunk információs
    szöveget írni, a kurzort viszont pozícionálhatjuk): }
  I := 0 ;
  ReadLn(Nev) ;
    { Amíg nem üres karakterláncot ütnek: }
  While Nev <> '' Do
    Begin
      Inc(I) ;
      GotoXY(22,WhereY - 1) ;
      ReadLn(Kor) ;
        { Kiírás az átirányított Output állományba: }
      WriteLn(I:3,': ',Nev,'':20 - Length(Nev),',',Kor:4,' éves');
      ReadLn(Nev) ;
    End ;
    { Az állomány lezárása: }
  Close(Output) ;

    { Ha a képernyőre ('CRT') a továbbiakban írni szeretnénk, az Output–ot vissza kell
    irányítani: }
  AssignCrt(Output) ;
  Rewrite(Output) ;
    { Az elmentett képernyő tartalmának visszaolvasása és kiírása: }
  WriteLn('Ime az elmentett képernyô:') ;
```

```
  Assign(KepernyoMent,'Kepernyo.Id') ;
  Reset(KepernyoMent) ;
  While Not Eof(KepernyoMent) Do
    Begin
      ReadLn(KepernyoMent,Sor) ;
      WriteLn(Sor) ;
    End ;
  Close(KepernyoMent) ;
  ReadLn ;
End.
```

5. feladat

Listázzunk ki egy szöveges állományt kérés szerint képernyőre vagy nyomtatóra!

```
Program Szoveg5 ;

Uses
  Crt ;

Var
  Szoveg,
  Hova        : Text ;
  SzovegNeve,
  Sor         : String ;
  Valasz      : Char ;

  { FŐPROGRAM: }
Begin
  Write('A listázandó állomány neve ') ;
  WriteLn('(lehet teljes állományspecifikációt is adni): ') ;
  ReadLn(SzovegNeve) ;
    { VIGYÁZAT! SzovegNeve = '' esetén az állományváltozóhoz a 'CON' perifériát rendeli az
      eljárás. Így ^Z leütéséig a begépelt szöveg sorait listázzuk. }
  Assign(Szoveg,SzovegNeve) ;
  {$I-}
  Reset(Szoveg) ;
  {$I+}
    { Ha nincs meg az állomány, álljon le a program: }
  If IOResult <> 0 Then
    Begin
      WriteLn('Az állomány nem található') ;
      ReadLn ;
      Halt ;
    End ;
```

```
    { Hova listázzunk? }
  WriteLn('K(épernyôre)/N(yomtatóra)') ;
  Repeat
    Valasz := Upcase(ReadKey) ;
  Until Valasz In ['K','N'] ;

    { A Hova állományba fogunk írni, azt a választól függően hozzárendeljük a képernyőhöz
    ('CON') vagy a nyomtatóhoz ('PRN'): }
  Case Valasz Of
    'K' : Assign(Hova,'CON') ;
    'N' : Assign(Hova,'PRN') ;
  End ;
  Rewrite(Hova) ;


    { Szöveg listázása: }
  WriteLn(Hova) ;
  While Not Eof(Szoveg) Do
    Begin
      ReadLn(Szoveg,Sor) ;
      WriteLn(Hova,Sor) ;
    End ;
  Close(Hova) ;
  If Valasz = 'K' Then
    ReadLn ;
End.
```

6. feladat

Nyomtató vezérlése Turbo Pascal–ból: Nyomtató inicializálása, különböző írásmódok (proporcionális, levélminőség, dőltbetű...), grafika kipróbálása. (A mintában szereplő nyomtatóvezérlő kódok Epson FX kompatibilis nyomtatókra érvényesek.)

```
Program Szoveg6 ;

Uses
  Printer ;

Const
  ESC   = #27 ;
  NUL   = #0 ;
  LF    = #10 ;
  FF    = #12 ;


    { Nyomtatókódok (EPSON FX): }
  PrIni         = ESC + '@' ;
  LevelMin      = ESC + 'x1' ;
  LevelMinKi    = ESC + 'x0' ;
```

```
  Dolt         = ESC + '4' ;
  Doltki       = ESC + '5' ;
  Prop         = ESC + 'p1' ;
  PropKi       = ESC + 'p0' ;
  Alahuz       = ESC + '-1' ;
  AlahuzKi     = ESC + '-0' ;
  Kiemelt      = ESC + 'E' ;
  KiemeltKi    = ESC + 'F' ;
  DuplaSzel    = #14 ;
  DuplaSzelKi  = #20 ;
  Suru         = #15 ;
  SuruKi       = #18 ;
  Grafika      = ESC + '*' + NUL ;
  Sortav0      = ESC + '3' + #24 ;

Var
  Sor,
  Pont       : Word ;
  Szelesseg  : Word ;

  { FŐPROGRAM: }
Begin
    { Nyomtató inicializálása (mintha ki– majd bekapcsolnánk a nyomtatót): }
  Write(Lst,PrIni) ;

    { Különböző írásmódok kipróbálása: }
  WriteLn(Lst,'Normál írásmód') ;
  WriteLn(Lst,Prop,'Proporcionális irásmód') ;
  WriteLn(Lst,PropKi,'Nem proporcionális írásmód, ' +
          'a betûk egyforma szélesek') ;
  WriteLn(Lst,LevelMin,'Levélminôségû írásmód',LevelMinKi) ;
  WriteLn(Lst,DuplaSzel,'Dupla szélességû írásmód',DuplaSzelKi) ;
  WriteLn(Lst,Alahuz,'Aláhúzott szöveg',AlahuzKi) ;
  WriteLn(Lst,Suru,'Sûrû írás',SuruKi) ;
  WriteLn(Lst,Kiemelt,'Kiemelt írás',KiemeltKi) ;

    { Grafika nyomtatása 24/216 inch sortávolsággal: }
  Write(Lst,Sortav0) ;
  For Sor := 1 To 3 Do
    Begin
      Szelesseg := 300 Div Sor ;
      Write(Lst,Grafika + Chr(Szelesseg Mod 256) +
            Chr(Szelesseg Div 256)) ;
      For Pont := 1 To Szelesseg Do
        Write(Lst,Chr($FF - Pont)) ;
      Write(Lst,LF) ;
    End ;
```

```
    { Szöveg lapdobással: }
  Write(Lst,'Próba vége',FF) ;
End.
```

7. feladat

Listázzunk ki egy szöveges állományt kérés szerint képernyőre vagy nyomtatóra! A listázás lapozással történjék mindkét esetben, cím és lapszám legyen minden oldal tetején!

```
Program Szoveg7 ;

Uses
  Crt,
  Printer ;

Var
  Szoveg      : Text ;
  SzovegNeve,
  Sor         : String ;
  Sorszam,
  Oldalszam   : Byte ;
  Valasz      : Char ;
  ELso        : Boolean ;

    { Eljárás típusú változó, értéke egy eljárás azonosítója lehet. Itt paraméterek nélkül adjuk
    meg, tehát csak olyan eljárást vehet fel, melynek nincs paramétere. Futáskor az eljárás címét
    tartalmazza. Ha a változóra úgy hivatkozunk, mintha egy eljárás lenne, akkor végrehajtódik
    az az eljárás, melynek azonosítóját, mint értéket felvette.}
  Kiiras      : Procedure ;

  { Azon eljárásokat, melyek azonosítóját eljárás típusú változó kapja, távoli hívással kell
  fordítani: }
{$F+}

  { Az eljárás egy sort kiír a képernyőre. Ha betelt a lap, akkor vár egy billentyű leütésére.
  Minden képernyő tetejére címet és lapszámot ír. }
Procedure Kepernyore ;
  Begin
    If Sorszam > 22 Then
      Begin
        TextColor(Cyan) ;
        If Elso
        Then
          Elso := False
```

```
      Else
        Begin
          GotoXY(1,25) ;
          Write('Tovább = <ENTER>') ;
          ReadLn ;
        End ;
      Inc(Oldalszam) ;
      ClrScr ;
      GotoXY(30,1) ;
      Write('LISTA KÉPERNYÖRE') ;
      GotoXY(70,1) ;
      Write(Oldalszam:3,'. oldal') ;
      GotoXY(30,2) ;
      Write('————————') ;
      Sorszam := 3 ;
    End ;

  Inc(Sorszam) ;
  GotoXY(1,Sorszam) ;
  TextColor(LightGray) ;
  WriteLn(Sor) ;
End ;
```

{ Az eljárás egy sort kiír a nyomtatóra, ha betelt a lap, lapot dob. Minden lap tetejére cím és lapszám kerül. }

```
Procedure Nyomtatora ;
  Begin
    If Sorszam > 50 Then
      Begin
        If Elso
        Then
          Begin
                { Nyomtató inicializálása }
            Write(Lst,#27 + '@') ;
            Elso := False ;
          End
        Else
              { Lapdobás }
          Write(Lst,#12) ;
        Inc(Oldalszam) ;
        WriteLn(Lst,'LISTA NYOMTATORA':40,'':20,Oldalszam:3,
                '. oldal') ;
        WriteLn(Lst,'————————':40) ;
        WriteLn(Lst) ;
        Sorszam := 0 ;
      End ;

    Inc(Sorszam) ;
    WriteLn(Lst,Sor) ;
  End ;
```

```
  { Hívási mód visszaállítása: }
{$F-}

  { FŐPROGRAM: }
Begin
  WriteLn ;
  Write('A listázandó állomány neve: ') ;
  ReadLn(SzovegNeve) ;
  Assign(Szoveg,SzovegNeve) ;
  {$I-}
  Reset(Szoveg) ;
  {$I+}
  If IOResult <> 0 Then
    Begin
      WriteLn('Az állomány nem található') ;
      ReadLn ;
      Halt ;
    End ;

  Write('K(épernyô)/N(yomtató)') ;
  Repeat
    Valasz := Upcase(ReadKey) ;
  Until Valasz In ['K','N'] ;
  Case Valasz Of
    'K' : Kiiras := Kepernyore ;
    'N' : Kiiras := Nyomtatora ;
  End ;

  Elso := True ;
  Sorszam := 100 ;
  Oldalszam := 0 ;

    { Szöveg listázása: }
  While Not Eof(Szoveg) Do
    Begin
      ReadLn(Szoveg,Sor) ;
      Kiiras ;
    End ;
  WriteLn ;
  TextColor(Cyan) ;
  Write('Vége a listának') ;
  ReadLn ;
  TextColor(LightGray) ;
  ClrScr ;
End.
```

8. feladat

Nyomtassuk ki parancssor–paraméterrel megadott Pascal programunkat úgy, hogy a fenntartott szavak kiemelten, a megjegyzések dőltbetűvel jelenjenek meg. Ha a program elején nem definiáljuk a 'Nyomtato' feltételes szimbólumot, akkor a kiírás képernyőre történjen – ekkor nincsenek kiemelt és dőlt betűk –, és lassítsuk a kiírást!

```
Program Szoveg8 ;

  { Nyomtato feltételes szimbólum definiálása. A program bizonyos részein ezen szimbólum
    jelenléte meghatározza a fordítást. Elég a $ jelet kivenni a DEFINE szó elől, s a program a
    képernyőre ír: }
{$DEFINE Nyomtato}

Uses
  Crt

    { Ha a Nyomtato szimbólum definiált, a Printer egység is hozzászerkesztendő a program-
      hoz: }
  {$IFDEF Nyomtato}
    ,Printer
  {$ENDIF} ;

Const
  FszavakSzama     = 48 ;
  Betuk            = ['A'..'Z','a'..'z','_'] ;
  Szamok           = ['0'..'9'] ;
  Megjegyzes_nyit  = '{' ;
  Megjegyzes_zar   = '}' ;
  Aposztrof        = '''' ;

  {$IFDEF Nyomtato}
      { Ha Nyomtato definiált, a vezérlőkódok (EPSON FX): }
    PrInit     = #27 + '@' ;
    Dolt       = #27 + '4' ;
    Doltki     = #27 + '5' ;
    Kiemelt    = #27 + 'E' ;
    KiemeltKi  = #27 + 'F' ;
    LapDob     = #12 ;
  {$ELSE}
      { Ha Nyomtato nem definiált, a vezerlőkódokat letiltjuk: }
    PrInit     = '' ;
    Dolt       = '' ;
    Doltki     = '' ;
    Kiemelt    = '' ;
    KiemeltKi  = '' ;
    LapDob     = '' ;
  {$ENDIF}
```

```
Type
  St15        = String[15] ;

Const
```

{ A Turbo Pascal 6.0 verzióban a következő szavak foglaltak: }

```
  Fenntartott : Array[1..FszavakSzama] Of St15 =
    ('AND','ASM','ARRAY','BEGIN','CASE','CONST','CONSTRUCTOR',
     'DESTRUCTOR','DIV','DO','DOWNTO','ELSE','END','FILE','FOR',
     'FUNCTION','GOTO','IF','IMPLEMENTATION','IN','INLINE',
     'INTERFACE','LABEL','MOD','NIL','NOT','OBJECT','OF','OR',
     'PACKED','PROCEDURE','PROGRAM','RECORD','REPEAT','SET','SHL',
     'SHR','STRING','THEN','TO','TYPE','UNIT','UNTIL','USES',
     'VAR','WHILE','WITH','XOR') ;
```

{ Figyelem! Néhány, az 5.5–ös verzióban foglalt szó a 6.0–ban szabványos direktíva (pl. ABSOLUTE, EXTERNAL ...). Ezek tehát átdefiniálhatóak, ami azonban nem ajánlatos. }

```
Var
  F        : 1..FszavakSzama ;
  PasProg  : Text ;
  ProgNev  : String ;
  Vege     : Boolean ;

  {$IFNDEF Nyomtato}
    Lst    : Text ;
  {$ENDIF}

  I        : Byte ;
  Sor      : String ;
  N        : Word ;
  Ch       : Char ;
```

{ A karaktereket előre kell látnunk, ezért ha még nincs vége az állománynak, beolvassuk a következőt: }

```
Procedure Olvas(Var Ch: Char) ;
  Begin
    Vege := Eof(PasProg) ;
    If Not Vege Then
      Read(PasProg,Ch) ;
  End;
```

{ A függvény a kapott szót nagybetűs formában adja vissza: }

```
Function Nagy(S : St15) : St15 ;
  Var
    I : Byte ;
  Begin
    For I := 1 to Length(S) Do
      S[I] := Upcase(S[I]) ;
    Nagy := S ;
  End ;
```

```
  { A függvény a kapott szót úgy adja vissza, hogy az nagybetűvel kezdődik, a többi pedig csupa
  kisbetű: }
Function NagyKicsi(S : String) : String ;
  Var
    I : Byte ;
  Begin
    S[1] := Upcase(S[1]) ;
    For I := 2 to Length(S) Do
      If S[I] In ['A'..'Z'] Then
        S[I] := Chr(Ord(S[I]) + 32) ;
    NagyKicsi := S ;
  End ;

  { Az eljárás összeállítja a megjegyzést, és azt kinyomtatja: }
Procedure Megjegyzes_Kiir ;
  Var
    Megjegyzes : String ;
  Begin
    Megjegyzes := '' ;
    Olvas(Ch) ;
      { Amíg a karakter nem a záró kapcsos zárójel, és nincs vége a programnak: }
    While (Ch <> Megjegyzes_zar) And Not Vege Do
      Begin
        Megjegyzes := Megjegyzes + Ch ;
        Olvas(Ch) ;
      End ;

    Write(Lst,Dolt,Megjegyzes_Nyit,Megjegyzes,Megjegyzes_Zar,
          DoltKi) ;
    Olvas(Ch) ;
  End ;

  { Az eljárás összeállítja a karakterláncot, és azt kinyomtatja: }
Procedure Karlanc_Kiir ;
  Var
    Karlanc : String ;
  Begin
    Karlanc := '' ;
    Olvas(Ch) ;
      { Amíg a karakter nem aposztróf, és nincs vége a programnak: }
    While (Ch <> Aposztrof) And Not Vege Do
      Begin
        Karlanc := Karlanc + Ch ;
        Olvas(Ch) ;
      End ;

    Write(Lst,Aposztrof,Karlanc,Aposztrof) ;
    Olvas(Ch) ;
  End ;
```

```
  { Az eljárás összeállítja az azonosítót, és ha az fenntartott szó, akkor csupa nagybetűs formában
    írja ki, egyébként pedig úgy, hogy a kezdőbetű nagy, a többi kicsi: }
Procedure Azon_Kiir ;
  Var
    Azon,
    Fenn  : String ;
  Begin
    Azon := '' ;
    Repeat
        Azon := Azon + Ch ;
        Olvas(Ch) ;
      { Amíg a karakter az azonosítóhoz tartozik, és nincs vége a programnak: }
    Until Not (Ch In Betuk + Szamok) Or Vege ;

      { Azonosító keresése a fenntartott szavak közt: }
    N := 1 ;
    Fenn := Nagy(Azon) ;
    While (Fenn <> Fenntartott[N]) And (N < FszavakSzama) Do
      Inc(N) ;
    If Fenn = Fenntartott[N]
    Then
        { Az azonosító fenntartott szó: }
      Write(Lst,Kiemelt,Fenn,KiemeltKi)
    Else
        { Nem fenntartott: }
      Write(Lst,NagyKicsi(Azon)) ;
  End ;

  { Az eljárás az első speciális karaktertől kezdve kinyomtatja a karaktereket addig, amíg valami
    mást nem talál: }
Procedure Spec_Kiir ;
  Begin
    Repeat
        Write(Lst,Ch) ;
        Olvas(Ch) ;
      { Amíg a karakter nem egy azonosító, megjegyzés vagy karakterlánc kezdete, és nincs
        vége a programnak: }
    Until (Ch In Betuk + [Megjegyzes_nyit] + [Aposztrof]) Or
          Vege ;
  End ;

  { FŐPROGRAM: }
Begin
    { A program a parancssor–paraméterrel megadott programot nyomtatja: }
  ProgNev := ParamStr(1) ;
  If ProgNev = '' Then
    Begin
      WriteLn('Használat: SZOVEG8 <forrásállomány[.PAS]>') ;
```

```
    ReadLn ;
    Halt ;
  End ;

  { A kiterjesztés .PAS, ha azt nem adják meg: }
If Nagy(Copy(ProgNev,Length(ProgNev) - 3,4)) <> '.PAS' Then
  ProgNev := ProgNev + '.PAS' ;

  { A program, mint szöveges állomány megnyitása: }
Assign(PasProg,ProgNev) ;
{$I-}
Reset(PasProg) ;
{$I+}
If IOResult <> 0 Then
  Begin
    WriteLn('A(z) ',ProgNev,' állomány nem található') ;
    ReadLn ;
    Halt ;
  End ;

  { Ha a Nyomtato szimbólum a program elején nem volt definiálva, akkor az Lst logikai
    állományt hozzárendeljük a képernyőhöz: }
{$IFNDEF Nyomtato}
  Assign(Lst,'CON') ;
  Rewrite(Lst) ;
{$ENDIF}

  { Program kinyomtatása. A szeiektálás megjegyzés, karakterlánc, azonosító (azon belül
    fenntartott vagy egyéb), és speciális karaktersor szerint történik: }
Write(Lst,PrInit) ;
Olvas(Ch) ;
While Not Vege Do
  Begin
    Case Ch Of
        { Megjegyzés következik: }
      Megjegyzes_nyit       : Megjegyzes_Kiir ;
        { Karakterlánc következik: }
      Aposztrof             : Karlanc_Kiir ;
        { Azonosító következik: }
      'A'..'Z','a'..'z','_' : Azon_Kiir ;
        { Speciális karakterek következnek: }
      Else                    Spec_Kiir ;
    End ;
      { Ha képernyőre írunk, lassítjuk a kiírást: }
    {$IFNDEF Nyomtato}
      Delay(50) ;
    {$ENDIF}
  End ;
```

```
  Close(PasProg) ;
  Write(Lst,LapDob) ;
End.
```

9. feladat

Írjunk programot, mely a paraméterként megkapott szöveges állományt illetve állományokat ('*' és '?' karakterek megengedettek) sűrített írásmóddal kinyomtatja úgy, hogy minden állomány elején szerepel aláhúzva, nagybetűvel az állomány neve, valamint a nyomtatási dátum!

```
Program Szoveg9 ;

Uses
  Dos,
  Printer ;

Type
  AllSpec = String[40] ;

Var
  DirInfo   : SearchRec ;
  Ev,
  Ho,
  Nap,
  HetNapja : Word ;

  { Az eljárás kinyomtatja a paraméterben megadott állományt, majd lapot dob: }
Procedure Nyomtat(SzovegNev: AllSpec) ;
  Var
    Szoveg : Text ;
    Ch     : Char ;
  Begin
    Assign(Szoveg,SzovegNev) ;
    Reset(Szoveg) ;
    While Not Eof(Szoveg) Do
      Begin
        Read(Szoveg,Ch) ;
        Write(Lst,Ch)
      End ;
    Close(Szoveg) ;
    Write(Lst,#12) ;
  End ;
```

```
  { FŐPROGRAM (nyomtatókódok EPSON FX-hez): }
Begin
    { Aktuális dátum bekérése: }
  GetDate(Ev,Ho,Nap,HetNapja) ;
    { Első nyomtatandó állomány lekérése: }
  FindFirst(ParamStr(1),Archive,DirInfo) ;
  If DosError <> 0 Then
    Begin
      Write('Nincs nyomtatandó állomány') ;
      ReadLn ;
      Halt ;
    End ;

    { Nyomtató inicializálása, sűrű írásmód bekapcsolása: }
  WriteLn(Lst,#27'@'#15) ;
  While DosError = 0 Do
    Begin
        { Az állomány neve és a dátum kiírása aláhúzva, dupla szélességgel: }
      Write(Lst,#27'-1'#14) ;
      WriteLn(Lst,Dirinfo.Name,'    Nyomtatva: ',
              Ev,'.',Ho,'.',Nap,'.') ;
      WriteLn(Lst,#27'-0'#20) ;
        { A szöveg kinyomtatása: }
      Nyomtat(DirInfo.Name) ;
        { Következő nyomtatandó állomány: }
      FindNext(DirInfo) ;
    End ;

    { Sűrű írás vége: }
  WriteLn(Lst,#18) ;
End.
```

10. feladat

Írjunk programot, mely a parancssor-paraméterekben megadott állományokat kinyomtatja a nyomtatón. Ha egy állomány-specifikáció előtt '@' karakter áll, akkor ez azt jelenti, hogy az illető állományokban a nyomtatandó állományok listája található. Ekkor minden sor egy újabb állomány-specifikáció, melyben szerepelhet '*' és '?' is, amit ismét megelőzhet '@' karakter. Ha az állománylistában a sor elején ';' karakter áll, tekintsük a sort megjegyzés-sornak (ugorjuk át).

```
Program Szoveg10 ;

{$V-,I-}
```

```
  { A rekurzív eljárás miatt a verem méretét a lehető legnagyobbra állítjuk: }
{$M 65520,0,655360}

Uses
  Crt,
  Dos,
  Printer ;
Const
  IOhiba      = $08 ;
  Papirhiany  = $20 ;
  OK          = $10 ;

Var
  I      : Byte ;
  Valasz : Char ;

  { Nyomtató állapotának lekérdezése megszakítás hívásával: }
Function NyomtatoAllapot : Byte ;
  Var
    Regiszterek : Registers ;
  Begin
    With Regiszterek Do
      Begin
        AH := 2 ;          { Nyomtatóállapot lekérdezése }
        DX := 0 ;          { Első nyomtató }

          { A $17-es megszakítás (BIOS printer rutin) hívása. Híváskor a CPU regiszterek
            felveszik a Regiszterek változó megfelelő mezőinek értékét, majd visszatéréskor a
            regiszterek tartalmát a változóban kapjuk vissza: }
        Intr($17,Regiszterek) ;

          { A megszakítás az AH regiszterben adja vissza a nyomtató állapotát, melyből
            számunkra a 0. (Time out), 3. (I/O error) ,4. (Printer selected) , és 5. (Out of paper)
            bitek érdekesek): }
        NyomtatoAllapot := AH And $39 ;
      End
  End ;

    { Egy karakter kinyomtatása megszakítással: }
  Procedure KarNyomtat(C : Char) ;
    Var
      Regiszterek : Registers ;
    Begin
      If NyomtatoAllapot = OK Then
        With Regiszterek Do
          Begin
            AH := 0 ;          { Karakter kinyomtatása }
            AL := Ord(C) ;     { Kinyomtatandó karakter kódja }
            DX := 0 ;          { Első nyomtató }
            Intr($17,Regiszterek) ;
          End ;
    End ;
```

```
  { Szöveg kinyomtatása az állományból – ha az létezik. A képernyőn megjelenítjük az állomány
    nevét és a megfelelő információt, előtte Eltolas db. szóközzel (struktúráltan írjuk ki): }
Procedure AllomanyNyomtatas(AllomanyNev : String ;
                            Eltolas: Byte);
  Var
    SzovegAll : Text ;
    Karakter,
    C         : Char ;
    I,
    X         : Byte ;

  Begin
    AllomanyNev := FExpand(AllomanyNev) ;
    Assign(SzovegAll,AllomanyNev) ;
    Reset(SzovegAll) ;
    If IOResult <> 0
    Then
      Begin
        GotoXY(Eltolas,WhereY) ;
        WriteLn(AllomanyNev,' nem található') ;
      End
    Else
      Begin
          { Az éppen nyomtatott állomány nevének struktúrált kiírása: }
        GotoXY(Eltolas,WhereY) ;
        WriteLn(AllomanyNev,' a pufferben') ;
          { Az állomány nyomtatása, végén lapdobás: }
        While Not Eof(SzovegAll) Do
          Begin
            Read(SzovegAll,Karakter) ;
            KarNyomtat(Karakter)
          End ;
        KarNyomtat(^L) ;
        Close(SzovegAll) ;
      End ;
  End ;

  { Állomány(–ok ,ha az állománynévben '*' vagy '?' karakter volt) nyomtatása: }
Procedure AllomanyokNyomtatasa(AllomanyNevek : String ;
                               Eltolas : Byte) ;
  Var
    DirInfo    : SearchRec ;
    Utvonal    : PathStr ;
    Nev        : NameStr ;
    Kit        : ExtStr ;
  Begin
    AllomanyNevek := FExpand(AllomanyNevek) ;
```

```
    { Az állomány specifikációját szét kell bontani, mert a FindFirst és FindNext eljárások
      az útvonalat nem adják vissza: }
    FSplit(AllomanyNevek,Utvonal,Nev,Kit) ;

    FindFirst(AllomanyNevek,AnyFile,DirInfo) ;
    If DosError <> 0
    Then
        { Nincs ilyen állomány egy darab sem: }
      Begin
        GotoXY(Eltolas,WhereY) ;
        WriteLn(AllomanyNevek,' nem található')
      End
    Else
        { Amíg van ilyen bejegyzés: }
      While DosError = 0 Do
        Begin
            { A nyomtatható állomány tartalmának nyomtatása: }
          If DirInfo.Attr And
             (Directory Or VolumeId Or SysFile) = 0 Then
            AllomanyNyomtatas(Utvonal + DirInfo.Name,Eltolas) ;
          FindNext(DirInfo) ;
        End ;
  End ;

  { Nyomtatás állománylistából: }
Procedure ListaAllomanyFeld(ListaAllomanyNev : String ;
                            Eltolas : Byte) ;
  Var
    ListaAllomany : Text ;
    ListaSor      : String ;
    DirInfo       : SearchRec ;
    Utvonal       : PathStr ;
    Nev           : NameStr ;
    Kit           : ExtStr ;
  Begin
      { Ha nem teljes az állománynév, kiegészítjük azt: }
    ListaAllomanyNev := FExpand(ListaAllomanyNev) ;

      { Az állomány teljes nevét szét kell bontani, mert a FindFirst és FindNext eljárások az
        útvonalat nem adják vissza: }
    FSplit(ListaAllomanyNev,Utvonal,Nev,Kit) ;

      { Az első listaállomány megkeresése: }
    FindFirst(ListaAllomanyNev,AnyFile,DirInfo) ;
```

```
    If DosError <> 0
    Then
        { Nincs ilyen bejegyzés – hibaüzenet: }
      Begin
        GotoXY(Eltolas,WhereY) ;
        WriteLn(ListaAllomanyNev,' nem található') ;
      End
    Else
        { Amíg van bejegyzés: }
      While DosError = 0 Do
        Begin
            { Az állománylistát tartalmazó állomány nevének struktúrált kiírása: }
          GotoXY(Eltolas,WhereY) ;
          WriteLn(Utvonal,DirInfo.Name,' feldolgozása:') ;
          Assign(ListaAllomany,Utvonal + DirInfo.Name) ;
          Reset(ListaAllomany) ;

            { Amíg vannak sorok az állománylistát tartalmazó állományban: }
          While Not Eof(ListaAllomany) Do
            Begin
              ReadLn(ListaAllomany,ListaSor) ;
              If Copy(ListaSor,1,1) <> ';' Then
                  { Ha nem megjegyzés az aktuális sor az állománylistát tartalmazó
                    állományban: }
                If Copy(ListaSor,1,1) = '@'
                Then
                    { Ha újabb állománylistát tartalmazó állománynév szerepel, az
                      eljárás rekurzív hívása: }
                  ListaAllomanyFeld(Copy(ListaSor,2,
                                    Length(ListaSor)),
                                    Eltolas + 3)
                Else
                    { A sorban lévő állomány tartalmának nyomtatása: }
                  AllomanyokNyomtatasa(ListaSor,Eltolas + 3) ;
            End ;
            { Következő listaállomány megkeresése: }
          FindNext(DirInfo) ;
        End ;
  End ;

  { FŐPROGRAM: }
Begin
  ClrScr ;
```

```
    { Ha nincs parancssor–paraméter – használati utasítás kiírása: }
  If ParamCount < 1 Then
    Begin
      WriteLn('Használat: SZOVEG10 [@]Állománynév ',
              '[[@]Állománynév ...]') ;
      ReadLn ;
      Halt ;
    End ;

    { A programot addig nem folytatjuk, amíg a nyomtató nem képes fogadni a karaktereket: }
  While NyomtatoAllapot <> OK  Do
    Begin
      Case NyomtatoAllapot Of
        IOHiba+Papirhiany : WriteLn('Kifogyott a papír!') ;
        IOhiba            : WriteLn('Kapcsolja be a nyomtatót!') ;
        Else                WriteLn('Nyomtató hiba.') ;
      End ;
      Write(#7'Folytassam? (I/N)') ;
      Repeat
        Valasz := Upcase(ReadKey) ;
      Until Valasz In ['I','N'] ;
      WriteLn ;
      If Valasz = 'N' Then
        Halt ;
    End ;

  I := 1 ;
    { Amíg van parancssor–paraméter: }
  While I <= ParamCount Do
    Begin
      If Copy(ParamStr(I),1,1) = '@'
      Then
          { Állománylistából történik a nyomtatás: }
        ListaAllomanyFeld(Copy(ParamStr(I),2,
                          Length(ParamStr(I))),1)
      Else
          { Az állományokat nyomtatjuk: }
        AllomanyokNyomtatasa(FExpand(ParamStr(I)),1) ;
        { Parancssor–paraméter mutató növelése: }
      Inc(I) ;
    End ;
End.
```

11. feladat

Írjunk programot, mely a parancssor–paraméterben megadott, vagy a beolvasott karakterláncot végigkeresi az aktuális alkönyvtár összes Pascal programjában, és kiírja azokat a programneveket, melyekben megtalálta! (A kis és nagybetűket ne különböztessük meg!)

```
Program Szoveg11 ;

Uses
  Dos ;

Type
  St12 = String[12] ;

Var
  KeresendoLanc : String ;
  DirInfo       : SearchRec ;

  { A karaterlánc nagybetűs formáját visszaadó függvény: }
Function Nagy(S : String) : String ;
  Var
    I : Byte ;
  Begin
    For I := 1 to Length(S) Do
      S[I] := Upcase(S[I]) ;
    Nagy := S ;
  End ;

  { Az eljárás egy adott szöveges állományban keresi a láncot, s ha van benne, kiírja az állomány
  nevét: }
Procedure Kereses(ProgNev : St12) ;
  Var
    PasProg  : Text ;
    Sor      : String ;
    VanBenne : Boolean ;
  Begin
    Assign(PasProg,ProgNev) ;
    Reset(PasProg) ;
    VanBenne := False ;
    While Not Eof(Pasprog) And Not VanBenne Do
      Begin
        ReadLn(PasProg,Sor) ;
        VanBenne := Pos(KeresendoLanc,Nagy(Sor)) > 0 ;
      End ;
    Close(PasProg) ;
    If VanBenne Then
      WriteLn(ProgNev) ;
  End ;
```

```
  { FŐPROGRAM: }
Begin
    { Ha nincs keresendő lánc, bekérés: }
  If ParamCount = 0
  Then
    Begin
      Write('A keresendô lánc: ') ;
      ReadLn(KeresendoLanc) ;
    End
  Else
    KeresendoLanc := ParamStr(1) ;
  KeresendoLanc := Nagy(KeresendoLanc) ;

    { Aktuális alkönyvtár végigolvasása, lánc keresése minden egyes Pascal programban: }
  WriteLn('A ''',KeresendoLanc,
          ''' a következô programokban található:') ;
  FindFirst('*.Pas',ReadOnly + Hidden + Archive,DirInfo) ;
  While DosError = 0 do
    Begin
      Kereses(DirInfo.Name) ;
      FindNext(DirInfo) ;
    End ;

  Write('<ENTER>') ;
  ReadLn ;
End.
```

TÍPUSOS ÁLLOMÁNYOK

1. feladat

Olvassunk be egész számokat 0 végjelig, s tároljuk el azokat lemezen az aktuális alkönyvtárban!

```
Program Tipall1 ;

Var
    { Típusos állomány típusú változó. Helyfoglalása 128 bájt, melyet a rendszer kezel, és a
    tényleges adatforgalomhoz szükséges állományjellemzőket tartalmazza: }
  Egeszek : File Of Integer ;
  Egesz   : Integer ;

  { FŐPROGRAM: }
Begin
    { Az Egeszek logikai állomány hozzárendelése az 'Egeszek.Dat' fizikai állományhoz (ez lesz
    az állomány neve a lemezen): }
  Assign(Egeszek,'Egeszek.Dat') ;
    { Az állomány megnyitása. Ha volt eddig 'Egeszek.Dat' állomány az aktuális alkönyvtárban,
    az törlődik: }
  Rewrite(Egeszek) ;

  WriteLn('Kérem az egész számokat: ') ;
    { Első egész beolvasása: }
  ReadLn(Egesz) ;
    { Amíg nem nullát ütnek: }
  While Egesz <> 0 Do
    Begin
        { Egesz felírása az Egeszek állományba. Írás után az állománymutató a következő
        pozícióra lép: }
      Write(Egeszek,Egesz) ;
        { Következő egész beolvasása: }
      ReadLn(Egesz) ;
    End ;

    { Az állomány lezárása: }
  Close(Egeszek) ;
End.
```

2. feladat

Az előzőleg létrehozott állomány egészeit írjuk ki képernyőre, és számítsuk ki azok átlagát!

```
Program Tipall2 ;

Var
  Egeszek : File Of Integer ;
  Egesz   : Integer ;
  Osszeg,
  Darab   : LongInt ;

  { FŐPROGRAM: }
Begin
    { Az állomány megnyitása. Ha nincs 'Egeszek.Dat' állomány az aktuális alkönyvtárban,
    futási hiba áll elő: }
  Assign(Egeszek,'Egeszek.Dat') ;
  Reset(Egeszek) ;

  Osszeg := 0 ;
  Darab := 0 ;
  WriteLn('Az ''Egeszek.Dat'' állományban található egész' +
          ' számok: ') ;

    { Amíg nincs vége az állománynak, vagyis az állománymutató létező elemre mutat: }
  While Not Eof(Egeszek) Do
    Begin
        { Elem olvasása az állományból. Olvasás után az állománymutató a következő
        pozícióra lép: }
      Read(Egeszek,Egesz) ;
        { Az egész kiírása a képernyőre: }
      WriteLn(Egesz) ;
        { Összeg és darabszám növelése: }
      Inc(Osszeg,Egesz) ;
      Inc(Darab) ;
    End ;

    { Állomány lezárása: }
  Close(Egeszek) ;
    { Ha nem volt üres az állomány, átlag kiírása: }
  If Darab > 0
  Then
    WriteLn('A számok átlaga: ',Osszeg / Darab:8:2)
  Else
    WriteLn('Az állomány üres') ;

  ReadLn ;
End.
```

3. feladat

Olvassunk be egész számokat 0 végjelig, s bővítsük azokkal az esetlegesen már meglévő 'Egeszek.Dat' állományt az aktuális alkönyvtárban! Ha nincs ilyen állomány, hozzuk létre azt!

```
Program Tipall3 ;

Var
  Egeszek : File Of Integer ;
  Egesz   : Integer ;

  { FŐPROGRAM: }
Begin
    { Az állomány nyitásának idejére kikapcsoljuk a beviteli és kiviteli műveletek ellenőrzését.
    Így akkor sem áll le a program futási hibával, ha nincs 'Egeszek.Dat' állomány. Ekkor az
    IOResult függvénnyel le kell kérdeznünk a művelet eredményességét. }
  Assign(Egeszek,'Egeszek.Dat') ;
  {$I-}
  Reset(Egeszek) ;
  {$I+}
    { Ha van már ilyen állomány, pozícionálunk az állomány végére, egyébként létrehozzuk azt: }
  If IOResult = 0
  Then
    Seek(Egeszek,FileSize(Egeszek))
  Else
    Rewrite(Egeszek) ;

    { Egészek beolvasása, s hozzáírásuk az állományhoz: }
  WriteLn('Kérem az egész számokat:') ;
  ReadLn(Egesz) ;
    { Amíg nem nullát ütnek be, az egészek kiírása az állományba: }
  While Egesz <> 0 Do
    Begin
      Write(Egeszek,Egesz) ;
      ReadLn(Egesz) ;
    End ;

    { Az állomány lezárása: }
  Close(Egeszek) ;
End.
```

4. feladat

Olvassuk az előzőleg létrehozott, egészekből álló állományt visszafelé, és írjuk ki az egészeket a képernyőre!

```
Program Tipall4 ;

Var
  Egeszek  : File Of Integer ;
  Egesz,
  I        : Integer ;

  { FŐPROGRAM: }
Begin
    { Állomány megnyitása: }
  Assign(Egeszek,'Egeszek.Dat') ;
  Reset(Egeszek) ;

    { Olvasás az állomány utolsó elemétől visszafelé (az elemek 0–tól vannak sorszámozva, az
    utolsó elem az állomány mérete–1 sorszámú: }
  WriteLn('Az egészek visszafelé:') ;
  For I := FileSize(Egeszek) - 1 Downto 0 Do
    Begin
        { Pozícionálás az I. elemre: }
      Seek(Egeszek,I) ;
        { Elem beolvasása, kiírása képernyőre: }
      Read(Egeszek,Egesz) ;
      WriteLn(Egesz) ;
    End ;

    { Állomány zárása: }
  Close(Egeszek) ;
  ReadLn ;
End.
```

5. feladat

Olvassunk be születési dátumokat (Hó, Nap), és gyűjtsük, hogy melyik napon hány ember született! Az adatbevitel végén listázzuk ki adott dátumtól adott dátumig a különböző napokon született emberek számát!

```
Program Tipall5 ;

{$X+}

Uses
  Crt ;

Const
  AktEv = 1991 ;
  Napok : Array [1..12] Of 1..31 =
            (31,28,31,30,31,30,31,31,30,31,30,31) ;

Var
  AktEvStr    : String[4] ;
  Szuletesek : File Of Word ;
  Ho,
  Nap         : Byte ;
  Szulszam,
  Tol,
  Ig,
  I           : Word ;

  { A függvény megállapítja egy adott évről, hogy az szökőév–e: }
Function SzokoEv(Ev:Word) : Boolean ;
  Begin
    SzokoEv := (Ev Mod 4 = 0) And (Ev Mod 100 <> 0) Or
               (Ev Mod 400 = 0)
  End ;

  { A függvény kiszámítja a hónap és nap alapján, hogy az a dátum az év hányadik napjára esik: }
Function Evnapja (Ho, Nap: Byte) : Word ;
  Var
    NapSorsz,
    I         : Word ;
  Begin
    NapSorsz := 0 ;
    For I := 1 To Ho - 1 Do
      NapSorsz := NapSorsz + Napok[I] ;
    EvNapja := NapSorsz + Nap ;
  End ;

  { Az eljárás az év adott sorszámú napjából kiszámítja a hónapot és a napot: }
Procedure Datum(Evnap: Word ; Var Ho, Nap: Byte) ;
  Var
    Napokszama : Word ;
  Begin
    Ho := 1 ;
    Napokszama := 0 ;
```

```
    While Evnap > Napokszama + Napok[Ho] Do
      Begin
        Inc(Napokszama,Napok[Ho]) ;
        Inc(Ho) ;
      End ;
    Nap := Evnap - Napokszama ;
  End ;
```

{ A függvény visszaad egy megadott intervallumba eső számot, melyet a képernyő egy adott pozíciójáról olvas be: }

```
Function Szam(X,Y: Byte; Tol,Ig: LongInt) : LongInt ;
  Var
    L : LongInt ;
  Begin
    {$I-}
    Repeat
        GotoXY(X,Y) ;
        ClrEol ;
        ReadLn(L) ;
    Until (IOResult = 0) And (L >= Tol) And (L <= Ig) ;
    {$I+}
    Szam := L ;
  End ;
```

{ FŐPROGRAM: }

```
Begin
```

{ Szökőév esetén február napjainak a száma nem 28, hanem 29: }

```
  Inc(Napok[2],Ord(SzokoEv(AktEv))) ;
```

{ Az állomány megnyitása. Ha még nem volt ilyen állomány, akkor létrehozzuk, és elemeit feltöltjük nullákkal. A nulladik elemet (mivel nulladik nap nincs) később nem használjuk. }

```
  Str(AktEv,AktEvStr) ;
  Assign(Szuletesek,AktEvStr + '.Dat') ;
  {$I-}
  Reset(Szuletesek) ;
  {$I+}
  If IOResult <> 0 Then
    Begin
      Rewrite(Szuletesek) ;
      Szulszam := 0 ;
      For I := 0 To EvNapja(12,31) Do
        Write(Szuletesek,Szulszam) ;
    End ;
```

{ Születésnapok felvitele. Csak jó hónapot és napot engedünk bevinni: }

```
  ClrScr ;
  WriteLn('Születésnapok felvitele:') ;
  WriteLn('————————————') ;
```

```
  { Első hónap beolvasása: }
Write('Hó  : ') ;
Ho := Szam(WhereX,WhereY,0,12) ;

  { Amíg a hónapnál nem 0–át ütnek be: }
While Ho <> 0 Do
  Begin

      { A hónaphoz tartozó nap beolvasása: }
    Write('Nap : ') ;
    Nap := Szam(WhereX,WhereY,1,Napok[Ho]) ;

      { Ho és Nap értéke alapján az elem állománypozíciójának kiszámítása, pozícionálás: }
    Seek(Szuletesek,Evnapja(Ho,Nap)) ;

      { A dátumhoz tartozó eddigi születések számának beolvasása az állományból, és a
        megnövelt érték visszaírása az állományba: }
    Read(Szuletesek,Szulszam) ;
    Inc(Szulszam) ;

      { Viszaírásnál újra kell pozícionálnunk, mert minden Read és Write eljárás alkalmával
        az állomány pozíciója eggyel nő: }
    Seek(Szuletesek,Evnapja(Ho,Nap)) ;
    Write(Szuletesek,Szulszam) ;

      { Következő hónap beolvasása: }
    WriteLn ;
    If WhereY >= 23 Then
      ClrScr ;
    Write('Hó  : ') ;
    Ho := Szam(WhereX,WhereY,0,12) ;
  End ;


  { Az állomány kilistázása dátumtól dátumig: }
ClrScr ;
WriteLn('Listázás adott dátumtól adott dátumig') ;
WriteLn('-------------------------') ;
WriteLn('Kezdô dátum: ') ;
Write('Hó  : ') ;
Ho := Szam(WhereX,WhereY,1,12) ;
Write('Nap : ') ;
Nap := Szam(WhereX,WhereY,1,Napok[Ho]) ;
Tol := Evnapja(Ho,Nap) ;

WriteLn('Utolsó dátum: ') ;
Write('Hó  : ') ;
Ho := Szam(WhereX,WhereY,1,12) ;
Write('Nap : ') ;
Nap := Szam(WhereX,WhereY,1,Napok[Ho]) ;
Ig := Evnapja(Ho,Nap) ;
```

```
    { Pozícionálás a kezdő dátumhoz tartozó elemre, a kért dátumokhoz tartozó elemek
      beolvasása, képernyőre írása: }
  Seek(Szuletesek,Tol) ;
  For I := Tol To Ig Do
    Begin
      Read(Szuletesek,Szulszam) ;
      Datum(I,Ho,Nap) ;
      WriteLn(Ho:2,'. hó ',Nap:2,'. Összesen: ',Szulszam) ;
        { Képernyő alján várunk: }
      If WhereY > 23 Then
        Begin
          ReadKey ;
          ClrScr ;
        End ;
    End ;
  ReadKey ;
End.
```

6. feladat

Olvassunk be dolgozókhoz tartozó adatokat (személyi kód, név, fizetés) addig, amíg a személyi kódnál <ENTER>–t nem ütnek le! Vigyük fel az adatokat az aktuális lemezegység aktuális könyvtárában lévő 'Dolgozok.Dat' állományba! Az adatfelvitel végén listázzuk ki az állományt, majd írjuk ki az állományban lévő dolgozók számát, valamint az állomány méretét bájtokban!

```
Program Tipall6 ;

Uses
  Crt ;

Type
    { Ilyen rekordokban tároljuk egy dolgozó adatait: }
  DolgozoTip = Record
                 Szemkod   : String[11] ;
                 Nev       : String[25] ;
                 Fizetes   : Real ;
               End ;

Var
    { Az állomány elemei rekordok lesznek: }
  Dolgozok : File Of DolgozoTip ;

    { Az állomány egyes elemeit ide fogjuk beolvasni, innen fogjuk kiírni azokat: }
  Dolgozo  : DolgozoTip ;
```

```
  { FŐPROGRAM: }
Begin
    { Állomány bővítése illetve létrehozása: }
  Assign(Dolgozok,'Dolgozok.Dat') ;
  {$I-}
  Reset(Dolgozok) ;
  {$I+}
  If IOResult = 0
  Then
    Seek(Dolgozok,FileSize(Dolgozok))
  Else
    Rewrite(Dolgozok) ;

    { Képernyő elkészítése: }
  ClrScr ;
  GotoXY(20,10) ; Write('Szem. kód  :') ;
  GotoXY(20,12) ; Write('Név        :') ;
  GotoXY(20,14) ; Write('Fizetés    :') ;

    { Első dolgozó személyi kódjának beolvasása: }
  GotoXY(34,10) ;
  ReadLn(Dolgozo.SzemKod) ;
    { Amíg nem csak <ENTER>-t ütnek: }
  While Dolgozo.SzemKod <> '' Do
    Begin
        { A dolgozó többi adatának beolvasása: }
      GotoXY(34,12) ;
      ReadLn(Dolgozo.Nev) ;
      GotoXY(34,14) ;
      ReadLn(Dolgozo.Fizetes) ;
        { A dolgozó adatainak felírása az állományba (a dolgozó adatait csak egyszerre tudjuk
          felírni, hiszen a típusos állomány írási/olvasási egysége az elem: }
      Write(Dolgozok,Dolgozo) ;
        { Előző dolgozó adatainak törlése a képernyőről: }
      GotoXY(34,10) ; ClrEol ;
      GotoXY(34,12) ; ClrEol ;
      GotoXY(34,14) ; ClrEol ;
        { Következő dolgozó személyi kódjának beolvasása: }
      GotoXY(34,10) ;
      ReadLn(Dolgozo.SzemKod) ;
    End ;

    { Állomány lezárása: }
  Close(Dolgozok) ;

    { Rekordok visszaolvasása (egy rekord = egy dolgozó adatai): }
  ClrScr ;
```

```
    { Az állomány megnyitása: }
  Reset(Dolgozok) ;

    { Amíg van dolgozó az állományban: }
  While Not Eof(Dolgozok) Do
    Begin

        { A következő rekord beolvasása: }
      Read(Dolgozok,Dolgozo) ;

        { A rekord mezőinek kiírása képernyőre (a mezőhivatkozások a Dolgozo rekordra
        fognak vonatkozni): }
      With Dolgozo Do
        Begin
          GotoXY( 2,WhereY) ; Write(SzemKod) ;
          GotoXY(15,WhereY) ; Write(Nev) ;
          GotoXY(50,WhereY) ; WriteLn(Fizetes:8:0) ;
        End ;
    End ;

    { Adatok kiírása, állomány lezárása: }
  WriteLn ;
  WriteLn('A dolgozók száma: ',Filesize(Dolgozok)) ;
  WriteLn('Az állomány helyfoglalása a lemezen: ',
          FileSize(Dolgozok) * SizeOf(DolgozoTip),' bájt') ;
  Close(Dolgozok) ;
  ReadLn ;
End.
```

7. feladat

A raktárból történő szállításokat számítógépen rögzítik. Egy szállításról a következő adatokat tartják nyilván:

Árukód	**: 3 karakter**
Irányítószám	**: 4 karakter**
Cím	**: 30 karakter**
Szállított mennyiség	**: egész**

Vigyük fel az adatokat az aktuális lemezegység '\Adatok' alkönyvtárába (amennyiben nem létezik, hozzuk létre azt). Az adatállomány neve 'Szallit.Dat' legyen, és ha már van ilyen állomány, akkor azt bővítsük (az adatbevitel akkor fejeződjék be, ha az árukódnál csak <ENTER>–t ütöttek)!

Az állományba csak jó adatokat engedjünk felvinni: árukód pontosan 3 karakter és az első karakter nagybetű; irányítószam 4 karakter, csupa számból áll, és budapesti irányítószám esetén a középső két szám a kerület.

Az adatbevitel végén listázzuk ki az állományt, és írjuk ki a szállítások számát!

```
Program Tipall7 ;

Uses
  Crt ;

Type
  SzallitasTip = Record
                   Arukod     : String[ 3] ;
                   Irszam     : String[ 4] ;
                   Cim        : String[30] ;
                   Mennyiseg  : LongInt ;
                 End ;

Var
  Szallitasok : File Of SzallitasTip ;
  Szallitas   : SzallitasTip ;
  AktDir      : String ;
  Puffer      : Array[1..32] Of Char ;
  I           : Byte ;

  { Beviteli mezők törlése a képernyőről: }
Procedure Mezo_Torles ;
  Begin
    GotoXY(35,10) ; ClrEol ;
    GotoXY(35,12) ; ClrEol ;
    GotoXY(35,14) ; ClrEol ;
    GotoXY(35,16) ; ClrEol ;
  End ;

  { Olvasás a Szallitas rekord Arukod mezőjébe. Az eljárásból akkor térünk vissza, ha jó árukódot
    vagy üres karakterláncot ütöttek be: }
Procedure Arukod_Beolvas ;
  Begin
      { A bemeneti puffert átállítjuk Puffer változóra, így Arukod változóba maximum 3
        karaktert lehet beírni: }
    SetTextBuf(Input,Puffer,5) ;
    With Szallitas Do
      Repeat
          GotoXY(35,10) ;
          ClrEol ;
          ReadLn(Arukod) ;
      Until (Length(Arukod) = 3) And (Arukod[1] In ['A'..'Z']) Or
            (Arukod = '') ;
  End ;

  { Olvasás a Szallitas rekord Irszam mezőjébe. Az eljárásból csak akkor térünk vissza, ha jó
    irányítószámot ütöttek be: }
Procedure Irszam_Beolvas ;
  Var
    JoIrszam : Boolean ;
```

```
    Szam      : Word ;
    Kod       : Integer ;
  Begin
    SetTextBuf(Input,Puffer,6) ;
    With Szallitas Do
      Repeat
          GotoXY(35,12) ;
          ClrEol ;
          ReadLn(Irszam) ;
          JoIrszam := (Irszam >= '1000') And (Irszam <= '9999') ;
          If JoIrszam And (Irszam[1] = '1') Then
            Begin
              Val(Copy(Irszam,2,2),Szam,Kod) ;
              JoIrszam := Szam In [1..22] ;
            End ;
      Until JoIrszam ;
  End ;

  { A Szallitas rekord Cim mezőjének beolvasása: }
Procedure Cim_Beolvas ;
  Begin
    SetTextBuf(Input,Puffer,32) ;
    GotoXY(35,14) ;
    ReadLn(Szallitas.Cim) ;
  End ;

  { A Szallitas rekord Mennyiseg mezőjének beolvasása. Az eljárásból csak akkor térünk vissza,
    ha nemnegatív egész számot ütöttek be: }
Procedure Mennyiseg_Beolvas ;
  Begin
    SetTextBuf(Input,Puffer,8) ;
    {$I-}
    Repeat
        GotoXY(35,16) ;
        ClrEol ;
        ReadLn(Szallitas.Mennyiseg) ;
    Until (IOResult = 0) And (Szallitas.Mennyiseg >= 0) ;
    {$I+}
  End ;

  { FŐPROGRAM: }
Begin
    { Megjegyezzük az aktuális lemezegység aktuális alkönyvtárát, hogy a program végén ide
      visszakapcsolhassunk: }
  GetDir(0,AktDir) ;
  {$I-}
    { Kapcsolás a '\Adatok' alkönyvtárba: }
  ChDir('\Adatok') ;
```

```
If IOResult <> 0 Then
  Begin
      { Nem létezik az alkönyvtár, létrehozzuk: }
    MkDir('\Adatok') ;
    If IOResult <> 0 Then
        { Nem sikerült a létrehozás – lemezhiba: }
      Begin
        WriteLn('Az alkönyvtárat nem sikerült létrehozni') ;
        ReadLn ;
        Halt ;
      End ;
      { Átkapcsolás a most létrehozott alkönyvtárba: }
    ChDir('\Adatok') ;
  End ;
  { Állomány megnyitása. Ha sikerült, pozícionálás az állomány végére, egyébként az
  állomány létrehozása: }
Assign(Szallitasok,'Szallit.Dat') ;
Reset(Szallitasok) ;
{$I+}
If IOResult = 0
Then
  Seek(Szallitasok,Filesize(Szallitasok))
Else
  Rewrite(Szallitasok) ;

  { Beviteli képernyő elkészítése: }
ClrScr ;
GotoXY(20,10) ; Write('Árukód        :') ;
GotoXY(20,12) ; Write('Irányítószám :') ;
GotoXY(20,14) ; Write('Cím           :') ;
GotoXY(20,16) ; Write('Mennyiség     :') ;
  { Rekordok hozzáadása az állományhoz: }
Arukod_Beolvas ;
While Szallitas.Arukod <> '' Do
  Begin
    Irszam_Beolvas ;
    Cim_Beolvas ;
    Mennyiseg_Beolvas ;
      { A szállítás rekord felírása lemezre: }
    Write(Szallitasok,Szallitas) ;
    Mezo_Torles ;
    Arukod_Beolvas ;
  End ;
Close(Szallitasok) ;

  { Rekordok visszaolvasása: }
ClrScr ;
WriteLn('Árukód','Irszám':8,'Cím':10,'Mennyiség':38) ;
```

```
  For I := 1 To 62 Do
    Write('-') ;
  WriteLn ;
  Reset(Szallitasok) ;
  While Not Eof(Szallitasok) Do
    Begin
      Read(Szallitasok,Szallitas) ;
      With Szallitas Do
        Begin
          GotoXY( 2,WhereY) ; Write(Arukod) ;
          GotoXY(10,WhereY) ; Write(Irszam) ;
          GotoXY(20,WhereY) ; Write(Cim) ;
          GotoXY(55,WhereY) ; WriteLn(Mennyiseg:6) ;
        End ;
    End ;
  WriteLn ;
  WriteLn('Összesen: ',FileSize(Szallitasok),' szállítás') ;
  Close(Szallitasok) ;

    { Visszakapcsolás az eredeti alkönyvtárba: }
  ChDir(AktDir) ;
  ReadLn ;
End.
```

8. feladat

Az előző, Tipall7 programmal létrehozott állomány azon szállításait, melyek budapesti címre történnek (irányítószám első karaktere 1–es), vigyük fel egy 'Szallit.Bp' állományba, a vidékre történő szállításokat pedig nyomtassuk ki! (Az adatállományok a '\Adatok' alkönyvtárban vannak.)

```
Program Tipall8 ;

Uses
  Printer ;

Type
  SzallitasTip = Record
                   Arukod    : String[ 3] ;
                   Irszam    : String[ 4] ;
                   Cim       : String[30] ;
                   Mennyiseg : LongInt ;
                 End ;

Var
  Szallitasok,
  Bp_Szallitasok : File Of SzallitasTip ;
```

```
  Szallitas       : SzallitasTip ;
  I               : Byte ;

  { A függvény a kapott karakterláncot adott hosszúságúra alakítja: ha hosszabb, csonkítja – ha
    rövidebb, szóközökkel egészíti ki: }
Function Bal(S: String ; Hossz: Byte): String ;
  Var
    I : Byte ;
  Begin
    S := Copy(S,1,Hossz) ;
    For I := 1 To Hossz - Length(S) Do
      S := S + ' ' ;
    Bal := S ;
  End ;

  { FŐPROGRAM: }
Begin
    { A szállításokat tartalmazó állomány megnyitása: }
  Assign(Szallitasok,'\Adatok\Szallit.Dat') ;
  {$I-}
  Reset(Szallitasok) ;
  {$I+}
  If IOResult <> 0 Then
    Begin
      WriteLn('Nincs szállítás állomány!') ;
      ReadLn ;
      Halt ;
    End ;
    { Új állomány létrehozása a budapesti szállítások számára: }
  Assign(Bp_Szallitasok,'\Adatok\Szallit.Bp') ;
  Rewrite(Bp_Szallitasok) ;

    { Fejléc írása a nyomtatóra: }
  WriteLn(Lst,
          '           V I D É K I   S Z Á L L I T Á S O K ') ;
  WriteLn(Lst) ;
  WriteLn(Lst,'Árukód  Irszám','Cím':10,'Mennyiség':38) ;
  For I := 1 To 62 Do
    Write(Lst,'-') ;
  WriteLn(Lst) ;

    { A szállításokat tartalmazó állomány végigolvasása, rekordok feldolgozása: }
  While Not Eof(Szallitasok) Do
    Begin
      Read(Szallitasok,Szallitas) ;
```

```
      If Szallitas.Irszam[1] = '1'
      Then
           { Budapesti a szállítás, felírás az új állományba: }
        Write(Bp_Szallitasok,Szallitas)
      Else
           { A nem budapestit kinyomtatjuk: }
        With Szallitas Do
          WriteLn(Lst,' ',Bal(Arukod,8),Bal(Irszam,10),
                  Bal(Cim,35),Mennyiseg:6) ;
   End ;

    { Állományok lezárása: }
  Close(Szallitasok) ;
  Close(Bp_Szallitasok) ;
End.
```

9. feladat

Az előző, Tipall8 programmal létrehozott '\Adatok\Szallit.Bp' állomány csak budapesti szállításokat tartalmaz. Gyűjtsük össze, hogy összesen mennyi szállítás történik az egyes kerületekbe!

```
Program Tipall9 ;

Type
  SzallitasTip = Record
                   Arukod    : String[ 3] ;
                   Irszam    : String[ 4] ;
                   Cim       : String[30] ;
                   Mennyiseg : LongInt ;
                 End ;

Var
  Szallitasok : File Of SzallitasTip ;
  Szallitas   : SzallitasTip ;
  Gyujto      : Array [1..22] Of LongInt ;
  Kerulet     : 1..22 ;
  Kod         : Integer ;

  { FŐPROGRAM: }
Begin
    { Állomány megnyitása, ha nincs, program leállítása: }
  Assign(Szallitasok,'\Adatok\Szallit.Bp') ;
  {$I-}
  Reset(Szallitasok) ;
```

```
  {$I+}
  If IOResult <> 0 Then
    Begin
      WriteLn('Az állomány nem található') ;
      ReadLn ;
      Halt ;
    End ;
    { Az egyes kerületekhez tartozó gyűjtők nullázása: }
  For Kerulet := 1 To 22 Do
    Gyujto[Kerulet] := 0 ;
    { Amíg van szállítás: }
  While Not Eof(Szallitasok) Do
    Begin
        { Szállítás beolvasása: }
      Read(Szallitasok,Szallitas) ;
        { Az irányítószámban található kerület alapján a kerülethez tartozó gyűjtő növelése a
        szállított mennyiséggel (az átalakítás eredményességét itt nem vizsgáljuk, hiszen csak
        jó adatokat vihettünk fel): }
      Val(Copy(Szallitas.Irszam,2,2),Kerulet,Kod) ;
      Gyujto[Kerulet] := Gyujto[Kerulet] + Szallitas.Mennyiseg ;
    End ;
    { Állomány lezárása, a gyűjtött mennyiségek kiírása: }
  Close(Szallitasok) ;
  For Kerulet := 1 To 22 Do
    WriteLn(Kerulet:3,'. kerület összesen: ',Gyujto[Kerulet]:8) ;
  ReadLn ;
End.
```

10. feladat

Az aktuális lemezegység '\Adatok' alkönyvtárában 'Aruk.Dat' néven szeretnénk nyilvántartani a raktáron lévő áruk adatait:

Árukód	**: 3 karakter**
Árunév	**: 15 karakter**
Egységár	**: valós**

Írjunk programot, mely lehetővé teszi ezen állomány karbantartását! Az állományban az árukód egyedi mező, a rekordot az árukód azonosítja.

Karbantartó műveletek:

- **Új áru beszúrása az állományba. (Ha van már ilyen áru az állományban, adjunk hibaüzenetet!)**
- **Adott áru adatainak módosítása. (Ha nincs az állományban a megadott áru, adjunk hibaüzenetet!)**

- **Adott áru törlése az állományból. (Ha nincs az állományban a megadott áru, adjunk hibaüzenetet!)**

Új rekordot mindig az állomány végéhez fűzzünk! Törlés esetén nem ésszerű a rekordot minden esetben fizikailag törölni (az állományt sűríteni), ez rengeteg lemezműveletet igé nyelne. Vegyünk fel minden rekordban egy plusz mezőt, mely jelzi, hogy a rekord érvényes–e, vagy törölt. Így a rekordot minden esetben csak logikailag töröljük. Keresésnél, listázásnál az így „törölt" rekordokat természetesen nem vesszük figyelembe.

A karbantartás végén sűrítsük az állományt, vagyis szabaduljunk meg a törölt rekordoktól!

```
Program Tipall10 ;

Uses
  Crt ;

Type
  St3        = String[3] ;
  St80       = String[80] ;
  Karhalmaz  = Set Of Char ;
  AruTip     = Record
                 Ervenyes : Boolean ;
                 Kod      : St3 ;
                 Nev      : String[15] ;
                 Egysegar : Real ;
               End ;

Var
  Aruk     : File Of AruTip ;
  Aru      : AruTip ;
  Arukod   : St3 ;
  Puffer   : Array[1..17] Of Char ;
  Menukar,
  Valasz   : Char ;
  Pozicio : LongInt ;
Procedure Mezo_Szoveg ;
  Begin
    ClrScr ;
    GotoXY(25,10) ; Write('Árukód   : ') ;
    GotoXY(25,12) ; Write('Árunév   : ') ;
    GotoXY(25,14) ; Write('Egységár : ') ;
  End ;

Procedure Kod_Beolvas ;
  Begin
      { A billentyűzetpuffer méretét a kód maximális hosszára állítjuk: }
    SetTextBuf(Input,Puffer,SizeOf(AruKod)+1) ;
    Repeat
        GotoXY(36,10) ;
        ClrEol ;
```

```
        ReadLn(Arukod) ;
    Until (Length(Arukod) = SizeOf(Arukod)-1) And
          (Arukod[1] In ['A'..'Z']) ;
  End ;

Procedure Egyeb_Beolvas ;
  Begin
    SetTextBuf(Input,Puffer,SizeOf(Aru.Nev)+1) ;
    GotoXY(36,12) ;
    ClrEol ;
    ReadLn(Aru.Nev) ;
    SetTextBuf(Input,Puffer,8) ;
    {$I-}
    Repeat
        GotoXY(36,14) ;
        ClrEol ;
        ReadLn(Aru.Egysegar) ;
    Until IOResult = 0 ;
    {$I+}
  End ;

Procedure Egyeb_Kiir ;
  Begin
    GotoXY(36,12) ; Write(Aru.Nev) ;
    GotoXY(36,14) ; Write(Aru.Egysegar:9:2) ;
  End ;
```

{ A függvény megállapítja, hogy van–e a megadott kódú áru az állományban, és ha van, megadja annak pozícióját: }

```
Function Van(Kod : St3 ; Var Poz: LongInt): Boolean ;
  Var
    Megvan : Boolean ;
  Begin
      { A keresés az állomány elejéről indul: }
    Seek(Aruk,0) ;
    Megvan := False ;
    While Not Eof(Aruk) And Not Megvan Do
      Begin
        Read(Aruk,Aru) ;
        Megvan := (Aru.Kod = Kod) And Aru.Ervenyes ;
      End ;
    Poz := FilePos(Aruk) ;
    If Megvan Then
      Dec(Poz) ;
    Van := Megvan ;
  End ;
```

```
  { Az eljárás kiírja az üzenet szövegét, s válaszra vár. A válasz csak a Valaszok halmazban
    megadott lehet, a tényleges választ az eljárás visszadja: }
Procedure Uzenet(Szoveg: St80 ; Valaszok: Karhalmaz ;
                 Var Valasz: Char) ;
  Begin
    GotoXY(25,24) ; Write(Szoveg) ;
    Repeat
      Valasz := Upcase(ReadKey) ;
    Until Valasz In Valaszok ;
    GotoXY(30,24) ;
    ClrEol ;
  End ;

  { Az eljárás laponként kilistázza az áruk adatait: }
Procedure Lista ;
  Var
    I : LongInt ;

  Procedure Lapfej ;
    Begin
      ClrScr ;
      WriteLn('Árukód     Árunév                          Egységár') ;
      WriteLn('-----------------------------------------------') ;
    End ;

  Begin
    Lapfej ;
    Seek(Aruk,0) ;
    For I := 1 To FileSize(Aruk) Do
      Begin
        If WhereY = 23 Then
          Begin
            Write('<ENTER>') ;
            Repeat
            Until ReadKey = #13 ;
            LapFej ;
          End ;
        Read(Aruk,Aru) ;
        With Aru Do
          If Ervenyes Then
            Begin
              GotoXY( 2,WhereY) ; Write(Kod) ;
              GotoXY(10,WhereY) ; Write(Nev) ;
              GotoXY(34,WhereY) ; WriteLn(Egysegar:9:2) ;
            End ;
      End ;
    Write('<ENTER>') ;
    Repeat
    Until ReadKey = #13 ;
  End ;
```

```
  { Az eljárás az Aruk állomány érvényes rekordjait átírja az UjAruk állományba. Törli az Aruk
    állományt, majd az UjAruk állományt átnevezi az eredetileg Aruk–hoz tartozó névre. (Adatbiz-
    tonság: az eredeti állományt csak akkor töröljük ki, ha az új már biztosan megvan.) }
Procedure Surit ;
  Var
    UjAruk : File Of AruTip ;
  Begin
    Assign(UjAruk,'\Adatok\UjAruk.Dat') ;
    Rewrite(UjAruk) ;
    Seek(Aruk,0) ;
    While Not Eof(Aruk) Do
      Begin
        Read(Aruk,Aru) ;
        If Aru.Ervenyes Then
          Write(UjAruk,Aru) ;
      End ;
    Close(Aruk) ;
    Close(UjAruk) ;
    Erase(Aruk) ;
    Rename(UjAruk,'\Adatok\Aruk.Dat') ;
  End ;

  { FŐPROGRAM: }
Begin
    { Az árukat nyilvántartó állomány megnyitása – ha még nem volt, létrehozása: }
  Assign(Aruk,'\Adatok\Aruk.Dat') ;
  {$I-}
  Reset(Aruk) ;
  {$I+}
  If IOResult <> 0 Then
    Rewrite(Aruk) ;

    { Főciklus: }
  Repeat
      ClrScr ;
      GotoXY(18, 6) ;
      Write('Å R U K    K A R B A N T A R T Å S A') ;
      GotoXY(25,10) ; Write('Felvitel.....F') ;
      GotoXY(25,12) ; Write('Törlés.....T') ;
      GotoXY(25,14) ; Write('Módosítás.....M') ;
      GotoXY(25,16) ; Write('Lista.....L') ;
      GotoXY(25,18) ; Write('Vége.....V') ;

      Menukar := Upcase(ReadKey) ;
      Case Menukar Of
        'F' : Begin
                Mezo_Szoveg ;
                Kod_Beolvas ;
```

```
                If Van(Arukod,Pozicio)
                Then
                  Uzenet('Már van ilyen áru <ESC>',[#27],Valasz)
                Else
                  Begin
                    Aru.Ervenyes := True ;
                    Aru.Kod := Arukod ;
                    Egyeb_Beolvas ;
                    Seek(Aruk,FileSize(Aruk)) ;
                    Write(Aruk,Aru) ;
                  End ;
              End ;
        'T' : Begin
                Mezo_Szoveg ;
                Kod_Beolvas ;
                If Van(Arukod,Pozicio)
                Then
                  Begin
                    Egyeb_Kiir ;
                    Uzenet('Töröljem? (I/N)',['I','N'],Valasz);
                    If Valasz = 'I' Then
                      Begin
                        Aru.Ervenyes := False ;
                        Seek(Aruk,Pozicio) ;
                        Write(Aruk,Aru) ;
                      End ;
                  End
                Else
                  Uzenet('Nincs ilyen áru <ESC>',[#27],Valasz);
              End ;
        'M' : Begin
                Mezo_Szoveg ;
                Kod_Beolvas ;
                If Van(Arukod,Pozicio)
                Then
                  Begin
                    Egyeb_Kiir ;
                    Uzenet('Módosítsam? (I/N)',['I','N'],Valasz) ;
                    If Valasz = 'I' Then
                      Begin
                        Egyeb_Beolvas ;
                        Seek(Aruk,Pozicio) ;
                        Write(Aruk,Aru) ;
                      End
                  End
                Else
                  Uzenet('Nincs ilyen áru <ESC>',[#27],Valasz);
              End ;
        'L' : Lista ;
      End ;
  Until Menukar = 'V' ;
```

```
  ClrScr ;
  Surit ;
End.
```

11. feladat

A Tipall8 programmal létrehozott '\Adatok\Szallit.Bp' állományt rendezzük árukód, és azon belül kerület szerint növekvően! A rendezett állomány neve 'RSzall.Bp' legyen ugyanabban az alkönyvtárban! A rendezést úgy végezzük el, hogy a rekordok rendezési kulcsait (a rendezésben résztvevő adatait) és állománypozícióit beírjuk egy tömbbe, majd a tömb rendezése után a rendezett állományt a tömb elemeinek sorrendje szerint építjük fel. (Feltételezzük, hogy az állományban nincs 5000–nél több rekord.)

```
Program Tipall11 ;

Type
  SzallitasTip = Record
                   Arukod     : String[ 3] ;
                   Irszam     : String[ 4] ;
                   Cim        : String[30] ;
                   Mennyiseg  : LongInt ;
                 End ;
  ElemTip      = Record
                   Kulcs      : String[5] ;
                   Rekordszam : LongInt ;
                 End ;

Var
  Szallitasok,
  Rendezett     : File Of SzallitasTip ;
  Szallitas     : SzallitasTip ;
  AllomanyMeret,
  I             : Word ;
  Tomb          : Array [1..5000] Of ElemTip ;
```

{ A kulcsokból és rekordszámokból álló tömböt gyorsrendezéssel fogjuk rendezni: }

```
Procedure Quick_Rendez ;

  Procedure Csere(Var Elem1, Elem2: ElemTip) ;
    Var
      Ment : ElemTip ;
    Begin
      Ment := Elem1 ;
      Elem1 := Elem2 ;
      Elem2 := Ment ;
    End ;
```

```
  Procedure Quick(Bal,Jobb: Word) ;
    Var
      I,
      J             : Word ;
      KozepsoKulcs : String[5] ;
    Begin
      I := Bal ;
      J := Jobb ;
      KozepsoKulcs := Tomb[(I + J) Div 2].Kulcs ;
      While I <= J Do
        Begin
          While Tomb[I].Kulcs < KozepsoKulcs Do
            Inc(I) ;
          While Tomb[J].Kulcs > KozepsoKulcs Do
            Dec(J) ;
          If I <= J Then
            Begin
              Csere(Tomb[I],Tomb[J]) ;
              Inc(I) ;
              Dec(J) ;
            End ;
        End ;
      If Bal < J Then
        Quick(Bal,J) ;
      If I < Jobb Then
        Quick(I,Jobb) ;
    End ;

  Begin
    Quick(1,AllomanyMeret) ;
  End ;

  { FŐPROGRAM: }
Begin
    { Az állomány megnyitása: }
  Assign(Szallitasok,'\Adatok\Szallit.Bp') ;
  {$I-}
  Reset(Szallitasok) ;
  {$I+}
  If IOResult <> 0 Then
    Begin
      WriteLn('Nincsenek budapesti szállítások!') ;
      ReadLn ;
      Halt ;
    End ;

    { Az állomány végigolvasása, a rekord rendezési kulcsának összeállítása, a tömb felépítése: }
  AllomanyMeret := FileSize(Szallitasok) ;
```

```
  For I := 1 To AllomanyMeret Do
    Begin
      Read(Szallitasok,Szallitas) ;
      Tomb[I].Kulcs := Szallitas.Arukod +
                       Copy(Szallitas.Irszam,2,2) ;
      Tomb[I].Rekordszam := I - 1 ;
    End ;

    { A tömb rendezése kulcs szerint növekvő sorrendbe: }
  Quick_Rendez ;

    { A rendezett állomány felírása egy új állományba a tömb alapján: }
  Assign(Rendezett,'\Adatok\RSzall.Bp') ;
  Rewrite(Rendezett) ;
  For I := 1 To AllomanyMeret Do
    Begin
        { A rendezettség szerint I. kulcshoz tartozó rekord beolvasása: }
      Seek(Szallitasok,Tomb[I].Rekordszam) ;
      Read(Szallitasok,Szallitas) ;
        { A rekord felírása az új állományba, szekvenciálisan: }
      Write(Rendezett,Szallitas) ;
    End ;
  Close(Szallitasok) ;
  Close(Rendezett) ;

    { A rendezett állomány kilistázása: }
  WriteLn('Árukód   Kerület  Mennyiség') ;
  Reset(Rendezett) ;
  For I := 1 To AllomanyMeret Do
    Begin
      Read(Rendezett,Szallitas) ;
      With Szallitas Do
        WriteLn(Arukod:4,Copy(Irszam,2,2):10,Mennyiseg:12) ;
    End ;
  Close(Rendezett) ;
  Write('<ENTER>') ;
  ReadLn ;
End.
```

12. feladat

Egy csoportváltásos (kontrollváltásos) feladat:
Az előző, Tipall11 programmal létrehozott 'RSzall.Bp' állomány rekordképe:

Árukód	**: 3 karakter**
Irányítószám	**: 4 karakter**

Cím : 30 karakter
Szállított mennyiség : egész

Az állomány csak budapesti szállításokat tartalmaz (irányítószám 1. karaktere 1–es), árunként és azon belül kerületenként rendezett. Az árukódhoz tartozó név és egységár egy külön állományban található, mely állományt a Tipall10 programmal hoztuk létre. Az állomány neve 'Aruk.Dat', rekordképe:

Érvényes : logikai
Árukód : 3 karakter
Árunév : 15 karakter
Egységár : valós

Mindkét állomány a '\Adatok' alkönyvtárban található. Készítsünk kimutatást, mely megmutatja, hogy árunként és azon belül kerületenként milyen mennyiségben és értékben történt szállítás!

A lista formátuma:

```
                    S Z Á L L I T Á S O K                99. oldal

-------------------------------------------------------------------
Árukód: ...  Áru neve: ..............  Egységár: 999999.99 Ft

         Kerület       Mennyiség          Érték
           --            ------          -------.--
            1.           999999         9999999.99
           ...
           22.
                         ------          ----------
 Áru összesen:           999999         9999999.99

-------------------------------------------------------------------
Árukód: ...  Áru neve: ..............  Egységár: 999999.99 Ft

         Kerület       Mennyiség          Érték
             .
             .
    .
    .
    .

-------------------------------------------------------------------
      ÖSSZESEN:          9999999        99999999.99
```

Kontrollmezők: árukód és kerület (irányítószám 2. és 3. karaktere). Csoportok (mezők) váltásakor a gyűjtött értékeket ki kell írni. Az ilyen típusú feladatot összegfokozatos feladatnak is szokás nevezni. A feladatot M. A. Jackson módszerével oldjuk meg.

A kimutatást ne közvetlenül a nyomtatóra készítsük, hanem egy lemezes állományba. Így az egyszer elkészített kimutatást később korlátlan példányszámban kinyomtathatjuk (pl. DOS Print paranccsal), és a kimutatások különböző állapotai is megjegyezhetők!

```
Program Tipall12 ;

Const
  Maxsorszam        = 50 ;
    { EPSON FX kódok: }
  DoltBe            = #27 + '4' ;
  DoltKi            = #27 + '5' ;
  KiemeltBe         = #27 + 'E' ;
  KiemeltKi         = #27 + 'F' ;

Type
  St3           = String[3] ;
  SzallitasTip  = Record
                    Arukod : String[3] ;
                    Irszam : String[4] ;
                    Cim    : String[30] ;
                    Menny  : LongInt ;
                  End ;
  AruTip        = Record
                    Ervenyes : Boolean ;
                    Kod      : St3 ;
                    Nev      : String[15] ;
                    Egysegar : Real ;
                  End ;

Var
  MaiDatum        : String[6] ;
  Szallitasok     : File Of SzallitasTip ;
  Szallitas       : SzallitasTip ;
  Szallitas_Vege  : Boolean ;
  Aruk            : File Of AruTip ;
  Aru             : AruTip ;
  Lista           : Text ;
  Kerulet,
  AktKerulet      : String[2] ;
  AktArukod       : String[3] ;
  AruPoz,
  OsszMenny,
  Aru_OsszMenny,
  Ker_OsszMenny   : LongInt ;
  OsszErtek       : Real ;
  Sorszamlalo,
  Lapszamlalo     : Byte ;
```

{ Olvasás a Szallitas állományból. A mezőváltás figyelése miatt mindig előre kell olvasni a szállításokat. Emiatt az állomány vége jelét késleltetni kell: }

```
Procedure Szallitas_Olvas ;
  Begin
    Szallitas_Vege := Eof(Szallitasok) ;
```

```
    If Not Szallitas_Vege Then
      Begin
        Read(Szallitasok,Szallitas) ;
        Kerulet := Copy(Szallitas.Irszam,2,2) ;
      End ;
  End ;
```

{ Adott kódú áru megkeresése az Aruk állományban. A lista készítésénél szükségünk lesz az áru nevére és egységárára: }

```
Function Van(Kod : St3): Boolean ;
  Var
    Megvan : Boolean ;
  Begin
    Seek(Aruk,0) ;
    Megvan := False ;
    While Not Eof(Aruk) And Not Megvan Do
      Begin
        Read(Aruk,Aru) ;
        Megvan := (Aru.Kod = Kod) And Aru.Ervenyes ;
      End ;
    Van := Megvan ;
  End ;
```

{ Minden lap tetejére fejléc kerül: }

```
Procedure Lapfej ;
  Begin
    If Lapszamlalo > 0 Then
      Write(Lista,#12) ;
    Inc(Lapszamlalo) ;
    WriteLn(Lista,KiemeltBe,'S Z Á L L I T Á S O K':40,
            '':15,Lapszamlalo,'. oldal',KiemeltKi) ;
    WriteLn(Lista) ;
    Sorszamlalo := 3 ;
  End ;
```

{ Új áru listázásának kezdésekor kiírjuk az áru jellemzőit, előkészítjük az ahhoz az áruhoz tartozó listázást: }

```
Procedure Aru_Fej ;
  Var
    I : Byte ;
  Begin
    If Sorszamlalo > Maxsorszam - 6 Then
      Lapfej ;
    Write(Lista, KiemeltBe) ;
    For I := 1 To 63 Do
      Write(Lista,'-') ;
    WriteLn(Lista) ;
    WriteLn(Lista,' Árukód: ',AktArukod,'  Áru neve: ',Aru.Nev,
            '':15 - Length(Aru.Nev),'  Egységár:',
            Aru.Egysegar:10:2,' Ft') ;
```

```
    WriteLn(Lista) ;
    Write(Lista,KiemeltKi,DoltBe) ;
    WriteLn(Lista,'          Kerület       Mennyiség',
            '          Érték') ;
    Write(Lista,DoltKi) ;
    WriteLn(Lista,'             --           ------',
            '          -------.--') ;
    Inc(Sorszamlalo,5) ;
  End ;

  { Kerületváltáskor az előző kerülethez tartozó adatokat összegezni kell: }
Procedure Kerulet_Osszesen ;
  Var
    Ertek : Real ;
  Begin
    If Sorszamlalo > Maxsorszam Then
      Lapfej ;
    Ertek := Ker_OsszMenny * Aru.Egysegar ;
    WriteLn(Lista,AktKerulet:14,'.',Ker_OsszMenny:15,Ertek:20:2) ;
    Inc(Sorszamlalo) ;
  End ;

  { Áruváltáskor az előző áruhoz tartozó adatokat összegezni kell: }
Procedure Aru_Osszesen ;
  Var
    Ertek : Real ;
  Begin
    Ertek := Aru_OsszMenny * Aru.Egysegar ;
    OsszErtek := OsszErtek + Ertek ;
    WriteLn(Lista,'':24,'------          ----------') ;
    WriteLn(Lista,'Áru összesen:':15,Aru_OsszMenny:15,Ertek:20:2);
    WriteLn(Lista) ;
    Inc(Sorszamlalo,3) ;
  End ;

  { A lista végén kiírandó sorok elkészítése, és kiírása: }
Procedure Osszesen ;
  Var
    I     : Byte ;
  Begin
    For I := 1 To 63 Do
      Write(Lista,'-') ;
    WriteLn(Lista) ;
    WriteLn(Lista,'ÖSSZESEN:':15,OsszMenny:15,OsszErtek:20:2) ;
  End ;
```

```
  { FŐPROGRAM: }
Begin
    { Előkészítő tevékenységek. A rendezett szállításokat és az árukat tartalmazó állományok
    megnyitása: }
  Assign(Szallitasok,'\Adatok\RSzall.Bp') ;
  Assign(Aruk,'\Adatok\Aruk.Dat') ;
  {$I-}
  Reset(Szallitasok) ;
  {$I+}
  If IOResult <> 0 Then
    Begin
      WriteLn('Nincs szállítás állomány!') ;
      ReadLn ;
      Halt ;
    End ;
  {$I-}
  Reset(Aruk) ;
  {$I+}
  If IOResult <> 0 Then
    Begin
      WriteLn('Nincs áru állomány!') ;
      ReadLn ;
      Halt ;
    End ;

    { A listát egy lemezes állományhoz rendeljük, mely neve a nyomtatás dátumától függ: }
  Write('Mai dátum [ÉÉHHNN] : ') ;
  ReadLn(MaiDatum) ;
  Assign(Lista,MaiDatum+'.Lst') ;
  Rewrite(Lista) ;
  Lapszamlalo := 0 ;
  Lapfej ;
  OsszMenny := 0 ;
  OsszErtek := 0 ;

    { Kísérlet az első szállítás rekord beolvasására: }
  Szallitas_Olvas ;
    { A ciklus addig tart, ameddig van még rekord: }
  While Not Szallitas_Vege Do
    Begin
        { Egy árucsoporthoz tartozó előkészítő tevékenységek: Árukód megjegyzése: }
      AktArukod := Szallitas.Arukod ;
        { Az áru megkeresése az állományban (ha nincs, egységára 0): }
      If Not Van(AktArukod) Then
        Begin
          Aru.Nev := '' ;
          Aru.Egysegar := 0 ;
        End ;
      Aru_Fej ;
```

```
      Aru_OsszMenny := 0 ;

         { A ciklus addig tart, míg ugyanaz az árukód. A külső ciklusok feltételei mindig
         öröklődnek: }
      While Not Szallitas_Vege And
            (Szallitas.Arukod = AktArukod) Do
        Begin
            { Egy kerületcsoporthoz tartozó előkészítések: Kerület megjegyzése: }
          AktKerulet := Kerulet ;
          Ker_OsszMenny := 0 ;

            { A ciklus addig tart, amíg ugyanaz az árukód, és ugyanaz a kerület: }
          While Not Szallitas_Vege And
                (Szallitas.Arukod = AktArukod) And
                (Kerulet = AktKerulet) Do
            Begin
              Inc(Ker_OsszMenny,Szallitas.Menny) ;
                { Kísérlet a következő szállítás rekord olvasására: }
              Szallitas_Olvas ;
            End ;

            { Kerület végéhez tartozó tevékenységek: }
          Inc(Aru_OsszMenny,Ker_OsszMenny) ;
          Kerulet_Osszesen ;
        End ;

         { Áru végéhez tartozó tevékenységek: }
      Inc(OsszMenny,Aru_OsszMenny) ;
      Aru_Osszesen ;
    End ;

    { A lista végén, egyszer végrehajtandó tevékenység: }
  Osszesen ;
  Close(Lista) ;
End.
```

13. feladat

Adott két állomány, melyek két különböző raktárból történő budapesti szállításokat tartalmaznak. Az állományok a '\Adatok' alkönytárban találhatók 'RSzall1.Bp' illetve 'RSzall2.Bp' néven. Ezeket a Tipall11 programmal hozhatjuk létre 'RSzall.Bp' mintájára. Mindkét állomány rendezett árukód és azon belül kerület (irányítószám 2. és 3. karaktere) szerint. Rekordképük a következő:

Árukód	**:**	**3 karakter**

Irányítószám : 4 karakter
Cím : 30 karakter
Szállított mennyiség : egész

Készítsünk a két rendezett állományból egy állományt '\Adatok\Szall.Bp' néven! A feladatot a rekordok összeválogatásával, M. A. Jackson módszerével oldjuk meg!

```
Program Tipall13 ;

Uses
  Crt ;

Const
  MaxKulcs = 'zzzzz' ;

Type
  SzallitasTip = Record
                   Arukod    : String[ 3] ;
                   Irszam    : String[ 4] ;
                   Cim       : String[30] ;
                   Mennyiseg : LongInt ;
                 End ;
  KulcsTip     = String[5] ;

Var
  Szallitasok1     : File Of SzallitasTip ;
  Szallitas1       : SzallitasTip ;
  Szallitasok2     : File Of SzallitasTip ;
  Szallitas2       : SzallitasTip ;
  Szallitasok      : File Of SzallitasTip ;
  Szallitas1_Vege,
  Szallitas2_Vege : Boolean ;
  Kulcs1,
  Kulcs2,
  AktKulcs         : String[5] ;

  { Ha hiányzik valamelyik szállítás állomány, ide kerül a vezérlés: }
Procedure Allj(N: Byte) ;
  Begin
    WriteLn('Hiányzik a(z) ',N,'. szállítás állomány!') ;
    ReadLn ;
    Halt ;
  End ;

  { Olvasás a Szallitas1 állományból. Az előolvasás miatt az állomány vége feltétel beálltát késleltetni kell. Ha az állománynak vége van, akkor a kulcsot olyan nagyra kell állítanunk, hogy a Szallitas2 állomány maradék rekordjai is feldolgozásra kerüljenek: }
Procedure Szallitas1_Olvas ;
  Begin
    Szallitas1_Vege := Eof(Szallitasok1) ;
```

```
    If Not Szallitas1_Vege
    Then
      Begin
        Read(Szallitasok1,Szallitas1) ;
          { Rendezési kulcs összeállítása: }
        Kulcs1 := Szallitas1.Arukod + Copy(Szallitas1.Irszam,2,2);
      End
    Else
      Kulcs1 := MaxKulcs ;
  End ;

  { Olvasás a Szallitas2 állományból: }
Procedure Szallitas2_Olvas ;
  Begin
    Szallitas2_Vege := Eof(Szallitasok2) ;
    If Not Szallitas2_Vege
    Then
      Begin
        Read(Szallitasok2,Szallitas2) ;
        Kulcs2 := Szallitas2.Arukod + Copy(Szallitas2.Irszam,2,2);
      End
    Else
      Kulcs2 := MaxKulcs ;
  End ;

  { A függvény két kulcs közül a kisebbiket adja vissza: }
Function Minimum(K1, K2 : KulcsTip) : KulcsTip ;
  Begin
    If K1 < K2
    Then
      Minimum := K1
    Else
      Minimum := K2 ;
  End ;

  { FŐPROGRAM: }
Begin
    { Állományok megnyitása. Ha valamelyik nincs, a program leállítása: }
  Assign(Szallitasok1,'\Adatok\RSzall1.Bp') ;
  {$I-}
  Reset(Szallitasok1) ;
  {$I+}
  If IOResult <> 0 Then
    Allj(1) ;
  Assign(Szallitasok2,'\Adatok\RSzall2.Bp') ;
  {$I-}
  Reset(Szallitasok2) ;
  {$I+}
```

```
If IOResult <> 0 Then
  Allj(2) ;
  { Új állomány létrehozása: }
Assign(Szallitasok,'\Adatok\Szall.Bp') ;
Rewrite(Szallitasok) ;

  { Mindkét állományból olvasunk egy rekordot: }
Szallitas1_Olvas ;
Szallitas2_Olvas ;
AktKulcs := Minimum(Kulcs1,Kulcs2) ;
  { A ciklusban mindig a soronkövetkező legkisebb kulcsot dolgozzuk fel: az ahhoz tartozó
    rekordot mindkét állományból felírjuk az új állományba. Helyére egy következő rekordot
    olvasunk abból az állományból, amelyikből a felírás történt. A feldolgozás addig tart, amíg
    van feldolgozandó kulcs: }
While AktKulcs < MaxKulcs Do
  Begin
      { Mivel a Case szerkezetben nem lehet feltételeket megadni, itt Else If szerkezettel
        szimuláljuk azt: }
    If Kulcs1 < Kulcs2 Then
        { (Kulcs1 = AktKulcs) és (Kulcs2 <> AktKulcs), vagyis Kulcs1 a feldolgozandó
          kulcs, a másik állományban nincsen párja: }
      Begin
        Write(Szallitasok,Szallitas1) ;
        Szallitas1_Olvas ;
      End
    Else If Kulcs2 < Kulcs1 Then
        { (Kulcs1 <> AktKulcs) és (Kulcs2 = AktKulcs), vagyis csak a Szallitasok2–ben
          van az aktuális feldolgozandó kulcsból: }
      Begin
        Write(Szallitasok,Szallitas2) ;
        Szallitas2_Olvas ;
      End
    Else If Kulcs1 = Kulcs2 Then
        { (Kulcs1 = AktKulcs) és (Kulcs2 = AktKulcs), vagyis mindkét állományban van
          az aktuális kulcsból, azokat egyszerre dolgozzuk fel: }
      Begin
        Write(Szallitasok,Szallitas1) ;
        Write(Szallitasok,Szallitas2) ;
        Szallitas1_Olvas ;
        Szallitas2_Olvas ;
      End ;
      { A következő feldolgozandó kulcs meghatározása: }
    AktKulcs := Minimum(Kulcs1,Kulcs2) ;
  End ;

  { Állományok lezárása: }
Close(Szallitasok1) ;
```

```
  Close(Szallitasok2) ;
  Close(Szallitasok) ;
End.
```

14. feladat

Készítsünk egy állomány–karbantartó programot kulcsállomány segítségével. A rekordok kulcsait egy külön rendezett kulcsállományban tároljuk, melyet a program futásának kezdetekor betöltünk a memóriába, majd a karbantartás végeztével kiírjuk azt lemezre. A keresést így mindig a rendezett kulcstömbben, memóriában végezzük, és csak a kulcshoz tartozó rekordszám alapján kell lemezhez fordulnunk, ha a rekord egyéb adataira van szükségünk.

```
Program Tipall14 ;

Uses
  Crt ;

Const
  MaxKulcs = 1000 ;

Type
  St20         = String[20] ;
  KulcsTip     = String[15] ;
  KulcsRekTip = Record
                  Kulcs : KulcsTip ;
                  Reksz : LongInt ;
                End ;
  TorzsRekTip = Record
                  Torolt : Boolean ;
                  Kulcs  : KulcsTip ;
                  Egyeb  : String[200] ;
                End ;

Var
  TorzsAll      : File Of TorzsRekTip ;
  TorzsRek      : TorzsRekTip ;
  KulcsTomb     : Array [1..MaxKulcs] Of KulcsRekTip ;
  KulcsAll      : File Of KulcsRekTip ;
  KulcsokSzama : LongInt ;
  Ch            : Char ;

  { A kulcsállomány beolvasása a kulcstömbbe: }
Procedure KulcsAll_Beolvas ;
  Var
    I : LongInt ;
```

```
  Begin
    KulcsokSzama := FileSize(KulcsAll) ;
    For I := 1 To KulcsokSzama Do
      Read(KulcsAll,KulcsTomb[I]) ;
  End ;
```

{ A KulcsokSzama darab kulcs és a hozzátartozó rekordszám kiírása a kulcstömbből a kulcsállományba: }

```
Procedure KulcsTomb_Kiir ;
  Var
    I : LongInt ;
  Begin
    Rewrite(KulcsAll) ;
    For I := 1 To KulcsokSzama Do
      Write(KulcsAll,KulcsTomb[I]) ;
    Close(KulcsAll) ;
  End ;
```

{ A függvény megkeres egy kulcsot a kulcstömbben. Ha van, megadja annak pozícióját, ha nincs, megadja a pozíciót, ahová be kellene szúrni: }

```
Function Van(Kulcs: KulcsTip ; Var Poz: LongInt): Boolean ;
  Var
    Tovabb : Boolean ;
  Begin
    Poz := 1 ;
    Tovabb := True ;
    Van := False ;
      { Amíg nem értünk a tömb végéhez, és még nem találtuk meg a kulcs helyét: }
    While (Poz <= KulcsokSzama) And Tovabb Do
      Begin
        Tovabb := Kulcs > KulcsTomb[Poz].Kulcs ;
        If Tovabb
        Then
          Inc(Poz)
        Else
          Van := Kulcs = KulcsTomb[Poz].Kulcs ;
      End ;
  End ;
```

{ Az eljárás a törzsrekord kulcsát beszúrja a kulcstömbbe a Poz. helyre. A kulcshoz tartozó rekordszám Reksz lesz: }

```
Procedure Kulcs_Beszur(Poz, Reksz: LongInt) ;
  Var
    I : LongInt ;
  Begin
      { A Poz. helytől kezdve az elemek feltolása: }
    For I := KulcsokSzama Downto Poz Do
      KulcsTomb[I + 1] := KulcsTomb[I] ;
```

```
      { Új kulcs és a hozzá tartozó rekordszám beszúrása a kulcstömbbe: }
    KulcsTomb[Poz].Kulcs := TorzsRek.Kulcs ;
    KulcsTomb[Poz].Reksz := Reksz ;

      { Kulcsok száma eggyel növekszik: }
    Inc(KulcsokSzama) ;
  End ;

  { A kulcstömb felépítése a törzsállományból elveszett kulcsállomány esetén: }
Procedure KulcsTomb_Felepit ;
  Var
    Poz,
    TorzsPoz : LongInt ;
  Begin
    KulcsokSzama := 0 ;

      { Törzsrekordok olvasása, az érvényes rekordok kulcsainak beszúrása a kulcstömbbe: }
    For TorzsPoz := 0 To FileSize(TorzsAll) - 1 Do
      Begin
        Read(TorzsAll,TorzsRek) ;
        If Not TorzsRek.Torolt And
           Not Van(TorzsRek.Kulcs,Poz) Then
          Kulcs_Beszur(Poz,TorzsPoz) ;
      End ;
  End ;

  { Törzsállomány sűrítése, vagyis átírása egy új állományba a logikailag törölt rekordok
    kihagyásával. A rekordokat kulcs szerint rendezve írjuk ki. Párhuzamosan elkészítjük az új
    kulcstömböt is: }
Procedure Surit ;
  Var
    UjTorzsAll : File Of TorzsRekTip ;
    I          : LongInt ;
  Begin
    Reset(TorzsAll) ;
    Assign(UjTorzsAll,'UjTorzs.Dat') ;
    Rewrite(UjTorzsAll) ;

      { A kulcstömb alapján beolvassuk a törzsállományt, és felépítjük az újat. Így csak az
        érvényes rekordok kerülnek felvitelre, rendezetten. A kulcstömbben tárolt rekordpozíció-
        kat aktualizáljuk: }
    For I := 1 To KulcsokSzama Do
      Begin
        Seek(TorzsAll,KulcsTomb[I].Reksz) ;
        Read(TorzsAll,TorzsRek) ;
        Write(UjTorzsAll,TorzsRek) ;
        KulcsTomb[I].Reksz := I-1 ;
      End ;
    Close(TorzsAll) ;
    Close(UjTorzsAll) ;

      { Eredeti törzsállomány törlése, az új átnevezése a régi névre: }
    Erase(TorzsAll) ;
```

```
    Rename(UjTorzsAll,'Torzs.Dat') ;
  End ;

  { A rekord kulcsának beolvasása a billentyűzetről: }
Procedure Kulcs_Beolvas ;
  Begin
    GotoXY(30,20) ; Write('Kulcs = ') ;
    ReadLn(TorzsRek.Kulcs) ;
  End ;

  { A rekord egyéb adatainak beolvasása a billentyűzetről: }
Procedure Egyeb_Beolvas ;
  Begin
    GotoXY(30,21) ; Write('Egyéb = ') ;
    ReadLn(TorzsRek.Egyeb) ;
  End ;

  { Hibaüzenet kiírása, <ENTER>-re tovább megy a program: }
Procedure HibaUzenet(HibaSt: St20) ;
  Begin
    GotoXY(25,25) ;
    Write(HibaSt + ' <ENTER>') ;
    Repeat
    Until ReadKey = #13 ;
  End ;

  { Új adatrekord felvitele az állományba: }
Procedure Felvitel ;
  Var
    Poz,
    TorzsPoz : LongInt ;
  Begin
      { Ha már nincs több kulcs számára hely a kulcstömbben, hibaüzenetet adunk, és kilépünk
      az eljárásból: }
    If KulcsokSzama = MaxKulcs Then
      Begin
        HibaUzenet('Betelt!') ;
        Exit ;
      End ;
      { Törzsrekord adatainak beolvasása a billentyűzetről: }
    Kulcs_Beolvas ;
      { Kulcs keresése a kulcstömbben: }
    If Van(TorzsRek.Kulcs,Poz)
    Then
        { Van már ilyen kulcs a kulcstömbben, hibaüzenetet adunk: }
      HibaUzenet('Már van ilyen kulcs')
```

```
    Else
        { Még nem volt eddig ilyen kulcs. Az új kulcsot a poz. helyre kell beszúrni: }
      Begin
          { Beolvassuk a többi adatot: }
        Egyeb_Beolvas ;
          { Az új rekordot a törzsállomány utolsó helyére írjuk: }
        TorzsPoz := FileSize(TorzsAll) ;
        Seek(TorzsAll,TorzsPoz) ;
        TorzsRek.Torolt := False ;
        Write(TorzsAll,TorzsRek) ;
          { A kulcsot a rendezett kulcstömb Poz. helyére beszúrjuk. Poz értékét a Van
          függvény határozta meg: }
        Kulcs_Beszur(Poz,TorzsPoz) ;
      End ;
      { Az állomány bővítése után célszerű az állományt lezárni és újra megnyitni. Így egy
      esetleges áramkimaradás vagy futási hiba esetén sem sérül a törzsállomány. }
    Close(TorzsAll) ;
    Reset(TorzsAll) ;
  End ;

  { Adatrekord törlése az állományból: }
Procedure Torles ;
  Var
    Poz,
    I    : LongInt ;
  Begin
    Kulcs_Beolvas ;
      { Kulcs keresése a kulcstömbben: }
    If Van(TorzsRek.Kulcs,Poz)
    Then
        { Van ilyen kulcs a kulcstömbben, a poz. helyen: }
      Begin
          { A kulcshoz tartozó rekordszám alapján beolvassuk a törzsrekordot: }
        Seek(TorzsAll,KulcsTomb[Poz].Reksz) ;
        Read(TorzsAll,TorzsRek) ;
          { A rekordot töröltre állítjuk, majd visszaírjuk az állományba: }
        TorzsRek.Torolt := True ;
        Seek(TorzsAll,KulcsTomb[Poz].Reksz) ;
        Write(TorzsAll,TorzsRek) ;
          { A kulcsot kitöröljük a kulcstömbből: }
        For I := Poz + 1 To KulcsokSzama Do
          KulcsTomb[I - 1] := KulcsTomb[I] ;
          { Kulcsok száma eggyel csökken: }
        Dec(KulcsokSzama) ;
      End
```

```
    Else
        { Nincs ilyen kulcs a kulcstömbben: }
      HibaUzenet('Nincs ilyen kulcs') ;
  End ;

  { Adatrekord módosítása: }
Procedure Modositas ;
  Var
    Poz,
    I    : LongInt ;
  Begin
    Kulcs_Beolvas ;
      { Kulcs keresése a kulcstömbben: }
    If Van(TorzsRek.Kulcs,Poz)
    Then
        { Van ilyen kulcs a kulcstömbben, a poz. helyen: }
      Begin
          { A kulcshoz tartozó törzsrekord beolvasása: }
        Seek(TorzsAll,KulcsTomb[Poz].Reksz) ;
        Read(TorzsAll,TorzsRek) ;
          { A módosítandó adatok beolvasása, a rekord visszaírása a törzsállományba
          ugyanarra a pozícióra: }
        Egyeb_Beolvas ;
        Seek(TorzsAll,KulcsTomb[Poz].Reksz) ;
        Write(TorzsAll,TorzsRek) ;
      End
    Else
        { Nincs ilyen kulcs a kulcstömbben: }
      HibaUzenet('Nincs ilyen kulcs') ;
  End ;

  { A törzsállomány kilistázása rendezetten, a kulcstömb alapján: }
Procedure Lista ;
  Var
    I : LongInt ;
  Begin
    ClrScr ;
    WriteLn('Kulcs                Egyéb') ;
    WriteLn('-----------------------') ;
    For I := 1 To KulcsokSzama Do
      Begin
          { Törzsrekord olvasása a kulcstömbben található rekordszám alapján: }
        Seek(TorzsAll,KulcsTomb[I].Reksz) ;
        Read(TorzsAll,TorzsRek) ;
        With TorzsRek Do
          Begin
            GotoXY( 1,WhereY) ; Write(Kulcs) ;
```

```
            GotoXY(20,WhereY) ; WriteLn(Egyeb) ;
          End ;
      End ;
    Write('<ENTER>') ;
    ReadLn ;
  End ;

  { FŐPROGRAM: }
Begin
    { A törzsállomány és a kulcsállomány hozzárendelése a fizikai állományokhoz: }
  Assign(TorzsAll,'Torzs.Dat') ;
  Assign(KulcsAll,'Kulcs.Dat') ;
    { Kísérlet a törzsállomány megnyitására: }
  {$I-}
  Reset(TorzsAll) ;
  {$I+}
  If IOResult = 0
  Then
      { A törzsállomány létezik: }
    Begin
        { Kísérlet a törzsállományhoz tartozó kulcsállomány megnyitására: }
      {$I-}
      Reset(KulcsAll) ;
      {$I+}
      If IOResult = 0
      Then
          { Sikerült, beolvassuk: }
        Begin
          KulcsAll_Beolvas ;
          Close(KulcsAll) ;
        End
      Else
          { A kulcsállomány megsérült, a kulcsokat a törzsállományból olvassuk be: }
        KulcsTomb_Felepit ;
    End
  Else
      { Létrehozunk egy üres törzsállományt, kulcsok száma 0: }
    Begin
      Rewrite(TorzsAll) ;
      KulcsokSzama := 0 ;
    End ;

    { A törzsállomány, és vele párhuzamosan a kulcstömb karbantartása: }
  Repeat
      ClrScr ;
      GotoXY(30,10) ; Write ('Felvitel.....F') ;
      GotoXY(30,11) ; Write ('Módosítás....M') ;
```

```
      GotoXY(30,12) ; Write ('Törlés.......T') ;
      GotoXY(30,13) ; Write ('Lista........L') ;
      GotoXY(30,14) ; Write ('Vége.........V') ;
      Repeat
        Ch := Upcase(ReadKey) ;
      Until Ch In ['F','M','T','L','V'] ;
      Case Ch Of
        'F' : Felvitel ;
        'M' : Modositas ;
        'T' : Torles ;
        'L' : Lista ;
      End ;
  Until Ch = 'V' ;

    { Ha a logikailag törölt rekordok száma nagyobb, mint 10, sűrítjük az állományt, majd a
    kulcstömböt lemezre írjuk: }
  If FileSize(TorzsAll) - KulcsokSzama > 10
  Then
    Surit
  Else
    Close(TorzsAll) ;

  KulcsTomb_Kiir ;
End.
```

15. feladat

Egy vállalat a tulajdonában lévő kamionokról nyilvántartást vezet. Egy kamion adatai a következők: rendszám, típus, megtett kilométer, állapot (foglalt, szabad, vagy javítandó). A kamion állapotától függően az állomány más és más egyéb adatot tartalmaz. Ha a kamion foglalt, akkor a célállomás, a vezető neve, valamint a visszatérési dátum szerepel a nyilvántartásban. Ha szabad, akkor a kamiont az jellemzi, hogy melyik garázsban található, és mi az esetleges terv vele. Ha viszont az autó hibás, fontos tudni, hogy mi a hibája, és mikorra javítják meg.

Hozzunk létre egy ilyen állományt (ha már volt, bővítsük), majd készítsünk listát, mely felsorolja a szabad kamionok rendszámait, a garázs számát, amelyben találhatók, valamint a tervet!

```
Program Tipall15 ;

Uses
  Crt ;

Type
  Datum      = Record
                 Ev   : Word ;
```

```
                Ho,
                Nap : Byte ;
              End ;
  Allapotok = (Foglalt,Szabad,Javitando) ;
```

{ KamionTip változó rekord. A változó rész a Case kulcsszó utáni mező értékétől függ, hogy milyen további mezőkre hivatkozhatunk a programban. A rekord változó mezőcsoportjai ugyanazon a memóriacímen kezdődnek (egyfajta típus–rádefiniálás). A rekord akkora memóriaterületet foglal, amekkora a nem változó rész, plusz a legnagyobb változó rész együttes mérete. Ebben az esetben a Case kulcsszóhoz nem tartozik külön End. }

```
  KamionTip = Record
                Rendszam  : String[8] ;
                Tipus     : String[10] ;
                MegtettKm : LongInt ;
```

{ Allapot mező pillanatnyi értéke határozza meg, hogy az ugyanazon a címen helyet foglaló további mezőcsoportok közül éppen melyik használható. }

```
                Case Allapot : Allapotok Of
                  Foglalt   : ( CelAllomas : String[5] ;
                                Vezeto     : String[15] ;
                                Visszater  : Datum ) ;
                  Szabad    : ( GarazsSzam : Byte ;
                                Terv       : String[20] ) ;
                  Javitando : ( Hiba       : String[40] ;
                                HatarIdo   : Datum )
              End ;

Var
  Kamionok    : File Of KamionTip ;
  Kamion      : KamionTip ;
  AllapotKar : Char ;
```

{ FŐPROGRAM: }

```
Begin
```

{ Kamion állomány megnyitása. Ha volt már, bővítjük, ha nem, létrehozzuk: }

```
  Assign(Kamionok,'Kamion.Dat') ;
  {$I-}
  Reset(Kamionok) ;
  {$I+}
  If IOResult = 0
  Then
    Seek(Kamionok,FileSize(Kamionok))
  Else
    Rewrite(Kamionok) ;
```

{ Adatok felvitele (adatellenőrzés nem történik): }

```
  ClrScr ;
  WriteLn('K A M I O N O K   ---   A D A T F E L V I T E L') ;
  Write('Rendszém    : ') ;
```

```
ReadLn(Kamion.Rendszam) ;
While Kamion.Rendszam <> '' Do
  With Kamion Do
    Begin
      Write('Típus        : ') ;
      ReadLn(Tipus) ;
      Write('Megtett Km   : ') ;
      ReadLn(MegtettKm) ;

      Write('(F)oglalt/(S)zabad/(J)avítandó') ;
      Repeat
        AllapotKar := Upcase(ReadKey) ;
      Until AllapotKar In ['F','S','J'] ;
      WriteLn ;
      Case AllapotKar Of
        'F' : Begin
                Allapot := Foglalt ;
                Write('Célállomás  : ') ;
                ReadLn(Celallomas);
                Write('Vezetô neve : ') ;
                ReadLn(Vezeto) ;
                Write('Visszatér      '#13#10) ;
                Write('       Év     : ') ;
                ReadLn(Visszater.Ev) ;
                Write('       Hó     : ') ;
                ReadLn(Visszater.Ho) ;
                Write('       Nap    : ') ;
                ReadLn(Visszater.Nap);
              End ;
        'S' : Begin
                Allapot := Szabad ;
                Write('Garázs száma: ') ;
                ReadLn(GarazsSzam);
                Write('Terv         : ') ;
                ReadLn(Terv) ;
              End ;
        'J' : Begin
                Allapot := Javitando ;
                Write('Hiba         : ') ;
                ReadLn(Hiba) ;
                Write('Elkészül       '#13#10) ;
                Write('      Év     : ') ;
                ReadLn(Hatarido.Ev) ;
                Write('      Hó     : ') ;
                ReadLn(Hatarido.Ho) ;
                Write('      Nap    : ') ;
                ReadLn(Hatarido.Nap) ;
              End ;
      End ;
      Write(Kamionok,Kamion) ;
      WriteLn ;
      Write('Rendszám     : ') ;
```

```
        ReadLn(Kamion.Rendszam) ;
      End ;
  Close(Kamionok) ;

    { Lista a szabad kamionokról: }
  ClrScr ;
  WriteLn('   S Z A B A D   K A M I O N O K') ;
  WriteLn('Rendszám    Garázs        Terv') ;
  WriteLn ;
  Reset(Kamionok) ;
  While Not Eof(Kamionok) Do
    Begin
      Read(Kamionok,Kamion) ;
      If Kamion.Allapot = Szabad Then
        With Kamion Do
          WriteLn(Rendszam,'':13 - Length(Rendszam),
                  GarazsSzam:3;'':10,Terv) ;
    End ;
  Close(Kamionok) ;
  WriteLn ;
  WriteLn('   V É G E') ;
  ReadLn ;
End.
```

HALMAZOK

```
      Gyujt(S,Sben) ;
      Kozos := Kozos * Sben ;
      ReadLn(S) ;
    End ;

    { Alaphalmaz kiírása: }
  WriteLn('Az alaphalmaz: ') ;
  For Ch := #0 To #255 Do
    If Ch In AlapHalmaz Then
      Write(Ch:2) ;
  WriteLn ;

    { A közös halmaz elemeinek kiírása: }
  WriteLn('Az alaphalmazhoz tartozó közös karakterek: ') ;
  If LancokSzama > 0 Then
    For Ch := #0 To #255 Do
      If Ch In Alaphalmaz Then
        If Ch In Kozos
        Then
          Write(Ch:2)
        Else
          Write(''':2) ;
  WriteLn ;
  ReadLn ;
End.
```

3. feladat

Írjunk egy beolvasó rutint, mely egy adott koordinátáról olvas be, és csak a paraméterként megadott 0 és 255 közötti egész számokat fogadja el!

```
Program Halmaz3 ;

Uses
  Crt ;

Type
  Szamhalmaz = Set Of Byte ;

Var
  N : Byte ;
```

```
  { Beolvasó rutin: }
Procedure Beolvas(X,Y: Byte ; Var Szam: Byte ;
                  Engedelyezett: Szamhalmaz) ;
  Var
    L   : LongInt ;
    Kod : Integer ;
  Begin
    {$I-,B-}
    Repeat
        GotoXY(X,Y) ;
        ClrEol ;
        ReadLn(L) ;
    Until (IOResult = 0) And (L>=0) And (L<=255) And
          (L In Engedelyezett) ;
    {$I+}
    Szam := L ;
  End ;

  { FŐPROGRAM: }
Begin
  ClrScr ;
  GotoXY(20,10) ;
  Write('Szám (20,22,150..170,199):') ;
  Beolvas(WhereX+1,10,N,[20,22,150..170,199]) ;
End.
```

4. feladat

Erzsi, Laci, Gyuri, Zsófi, Kati és Dániel játszanak. Mindenki mond 1 és 20 között három számot. Az nyer, aki több olyan számot mond, amit senki más nem mondott. Írjuk ki, melyek azok a számok, amelyeket csak egy játékos mondott!

```
Program Halmaz4 ;

Uses
  Crt ;

Const
  JatekosSzam = 6 ;       { Ennyien játszhatnak egyszerre }
  TippSzam    = 3 ;       { Egy játékos ennyit tippelhet }
  MaxTipp     = 20 ;      { Ez a legnagyobb tippelhető szám }
  JatekosNev  : Array [1..JatekosSzam] Of String[6] =
    ('Erzsi ','Laci  ','Gyuri ','Zsófi ','Kati  ','Dániel') ;

Var
  Tippek           : Array [1..JatekosSzam] Of Set Of 1..MaxTipp ;
  Tipp             : LongInt ;
```

```
  Jatekos,
  Tobbi,
  I                 : Byte ;
  Tobbi_mondta,
  Tobbi_nem_mondta  : Set Of 1..MaxTipp ;
  Puffer            : Array[1..5] Of Byte ;
  TippSt            : String[3] ;
  Kod               : Integer;

  { FŐPROGRAM: }
Begin
    { Tippek beolvasása: }
  ClrScr ;
  SetTextBuf(Input,Puffer,5) ;
  WriteLn('Kérem a tippeket: ') ;
  GotoXY(10,2) ;
  For I := 1 To TippSzam Do
    Write(I:5,'.') ;
  For Jatekos := 1 To JatekosSzam Do
    Begin
      Tippek[Jatekos] := [] ;
      GotoXY(1,Jatekos + 2) ;
      WriteLn(JatekosNev[Jatekos],':') ;
      For I := 1 To TippSzam Do
        Begin
          Repeat
              GotoXY(8 + 6 * I,Jatekos + 2) ;
              ClrEol ;
              ReadLn(TippSt) ;
              Val(TippSt,Tipp,Kod) ;
          { Csak akkor fogadjuk el a tippet, ha az egy tippelhető pozitív szám, és még nem
            tippelte ez a játékos: }
          Until (Kod = 0) And (Tipp >= 1) And (Tipp <= MaxTipp)
                And Not (Tipp In Tippek[Jatekos]) ;
          Tippek[Jatekos] := Tippek[Jatekos] + [Tipp] ;
        End ;
    End ;
  WriteLn ;

  WriteLn('A következô számokat csak egy játékos mondta: ') ;
  For Jatekos := 1 To JatekosSzam Do
    Begin
        { Meghatározzuk azon számok halmazát, melyeket legalább egyvalaki mondott,
          leszámítva a szóbanforgó játékost: }
      Tobbi_mondta := [] ;
      For Tobbi := 1 To JatekosSzam Do
        If Tobbi <> Jatekos Then
          Tobbi_mondta := Tobbi_mondta + Tippek[Tobbi] ;
```

```
        { Meghatározzuk azon számok halmazát, melyeket senki sem mondott, leszámítva a
          szóbanforgó játékost : }
      Tobbi_nem_mondta := [1..MaxTipp] - Tobbi_mondta ;

        { Az illető játékos számai közül kiírjuk azokat, melyeket a többiek nem mondták: }
      Write(JatekosNev[Jatekos],': ') ;
      For I := 1 To MaxTipp Do
        If (I In Tippek[Jatekos]) And (I In Tobbi_nem_mondta) Then
          Write(I:3) ;
      WriteLn ;
    End ;

  ReadLn ;
End.
```

5. feladat

Adott a '\Adatok' alkönyvtárban egy 'Szallit.Dat' állomány a következő rekordképpel:

Árukód	**: 3 karakter**
Irányítószám	**: 4 karakter**
Cím	**: 30 karakter**
Szállított mennyiség	**: egész**

(Az állományt a Tipall7.Pas programmal hozhatjuk létre.)

Gyűjtsük ki azokat a budapesti kerületeket, melyekbe történt szállítás!

```
Program Halmaz5 ;

Uses
  Crt ;

Type
  SzallitasTip = Record
                   Arukod    : String[3] ;
                   Irszam    : String[4] ;
                   Cim       : String[30] ;
                   Mennyiseg : LongInt ;
                 End ;

Var
  Szallitasok : File Of SzallitasTip ;
  Szallitas   : SzallitasTip ;
  Kerulet     : 1..22 ;
  Keruletek   : Set Of 1..22 ;
  Kod         : Integer  ;
```

```
  { FŐPROGRAM: }
Begin
    { Állomány megnyitása. Ha nincs, hibaüzenet és a program leállítása: }
  Assign(Szallitasok,'\Adatok\Szallit.Dat') ;
  {$I-}
  Reset(Szallitasok) ;
  {$I+}
  If IOResult <> 0 Then
    Begin
      WriteLn('Nincs állomány') ;
      ReadLn ;
      Halt ;
    End ;

    { Kerületek gyűjtése az állományból. Kezdetben Keruletek legyen az üres halmaz: }
  Keruletek := [] ;
    { Amíg van szállítás: }
  While Not Eof(Szallitasok) Do
    Begin
      Read(Szallitasok,Szallitas) ;
      With Szallitas Do
        If Irszam[1] = '1' Then
            { Budapesti a szállítás: }
          Begin
              { Irányítószámból a kerület meghatározása: }
            Val(Copy(Irszam,2,2),Kerulet,Kod) ;
            If (Kod = 0) And (Kerulet In [1..22]) Then
                { A kerület hozzáadása a Keruletek halmazhoz: }
              Keruletek := Keruletek + [Kerulet] ;
          End ;
    End ;
    { Állomány zárása: }
  Close(Szallitasok) ;

    { A kigyűjtött kerületek kiírása: }
  ClrScr ;
  WriteLn('Budapesti szállítás a következô kerületekbe ' +
          'történt : ') ;
  For Kerulet := 1 To 22 Do
    If Kerulet In Keruletek Then
      Write(Kerulet : 3) ;
  ReadLn ;
End.
```

6. feladat

Tároljunk lemezen különböző karakterkészleteket '1.Chr', '2.Chr', ... néven, karakterhalmaz formájában!

```
Program Halmaz6 ;

Uses
  Crt ;

Type
  Karakterkeszlet = Set Of Char ;

Var
  Keszlet          : Karakterkeszlet ;
  KeszletAllomany  : File Of Karakterkeszlet ;
  KeszletNev       : String[12] ;
  KeszletSt        : String ;
  I,
  N                : Byte ;

  { FŐPROGRAM: }
Begin
  ClrScr ;
    { 1. készlet beolvasása (a készlet karaktereit egy karakterláncba olvassuk be): }
  N := 1 ;
  WriteLn(N:3,'. karakterkészlet betûi, <ENTER> = vége :') ;
  ReadLn(KeszletSt) ;

    { Amíg nem üres készletet adnak meg: }
  While KeszletSt <> '' Do
    Begin
        { N. készlet összeállítása: }
      Keszlet := [] ;
      For I := 1 To Length(KeszletSt) Do
        Keszlet := Keszlet + [KeszletSt[I]] ;
        { N. készlet felírása lemezre: }
      Str(N,KeszletNev) ;
      Assign(KeszletAllomany,KeszletNev + '.Chr') ;
      Rewrite(KeszletAllomany) ;
      Write(KeszletAllomany,Keszlet) ;
      Close(KeszletAllomany) ;
        { Következő készlet beolvasása: }
      Inc(N) ;
      WriteLn(N:3,'. karakterkészlet betûi, <ENTER> = vége :') ;
```

```
      ReadLn(KeszletSt) ;
    End ;
End.
```

7. feladat

Válasszunk a Halmaz6.Pas programmal létrehozott karakterkészletekből! Olvassunk be billentyűzetről karaktereket, de csak a választott karakterkészletből engedjük meg a bevitelt!

```
Program Halmaz7 ;

Uses
  Crt ;

Type
  Karakterkeszlet = Set Of Char ;

Var
  Keszlet          : Karakterkeszlet ;
  KeszletAllomany  : File Of Karakterkeszlet ;
  KeszletSzam      : String[3] ;
  Ch               : Char ;

  { FŐPROGRAM: }
Begin
  ClrScr ;
    { A karakterkészlet kiválasztása: }
  Write('Karakterkészlet száma : ') ;
  ReadLn(KeszletSzam) ;

    { A készlethez tartozó állomány megnyitása: }
  Assign(KeszletAllomany,KeszletSzam + '.Chr') ;
  {$I-}
  Reset(KeszletAllomany) ;
  {$I+}
  If IOResult <> 0 Then
    Begin
      WriteLn('Nincs meg a készlet') ;
      ReadLn ;
      Halt ;
    End ;

    { Karakterkészlet beolvasása: }
  Read(KeszletAllomany,Keszlet) ;
  Close(KeszletAllomany) ;
```

```
    { A választott karakterkészlet kiírása: }
  Write('Elfogadott karakterek: ') ;
  For Ch := #0 To #255 Do
    If Ch In Keszlet Then
      Write(Ch) ;
  WriteLn ;

    { Karakterek bevitele. Csak a készletben lévő karaktereket engedjük bevinni: }
  WriteLn('Kérem a karaktereket (<ENTER> = Vége ): ') ;
  Ch := ReadKey ;
  While Ch <> #13 Do
    Begin
      If Ch In Keszlet
      Then
        Write(Ch)
      Else
        Write(^G) ;
      Ch := ReadKey ;
    End ;
End.
```

8. feladat

Vizsgáljuk meg az aktuális könyvtárban található összes Pascal programot, és írjuk ki azokat a fenntartott szavakat, melyek

- **minden programban szerepelnek**
- **egy programban sem szerepelnek**
- **szerepelnek legalább egy programban!**

```
Program Halmaz8 ;

Uses
  Crt,
  Dos ;

Const
  FszavakSzama      = 48 ;
  Betuk             = ['A'..'Z','a'..'z','_'] ;
  Szamok            = ['0'..'9'] ;
  Megjegyzes_nyit   = '{' ;
  Megjegyzes_zar    = '}' ;
  Aposztrof         = '''' ;

Type
  St15       = String[15] ;
  Szohalmaz = Set Of 1..Fszavakszama ;
```

```
Const
    { Fenntartott szavak a Turbo Pascal 6.0 verzióban: }
  Fenntartott : Array[1..FszavakSzama] Of St15 =
    ('AND','ASM','ARRAY','BEGIN','CASE','CONST','CONSTRUCTOR',
     'DESTRUCTOR','DIV','DO','DOWNTO','ELSE','END','FILE','FOR',
     'FUNCTION','GOTO','IF','IMPLEMENTATION','IN','INLINE',
     'INTERFACE','LABEL','MOD','NIL','NOT','OBJECT','OF','OR',
     'PACKED','PROCEDURE','PROGRAM','RECORD','REPEAT','SET','SHL',
     'SHR','STRING','THEN','TO','TYPE','UNIT','UNTIL','USES',
     'VAR','WHILE','WITH','XOR') ;

Var
  DirInfo : SearchRec ;
  PasProg : Text ;
  Vege    : Boolean ;
    { A halmazok a fenntartott szavak tömbbeli indexeit tartalmazzák: }
  Fszavak,
  Osszes,
  Kozos,
  Nincs   : Szohalmaz ;
  I       : Byte ;

  { A karaktereket előre kell látnunk, így ha még nincs vége az állománynak, beolvassuk a
  következőt }
Procedure Olvas(Var Ch: Char) ;
  Begin
    Vege := Eof(PasProg) ;
    If Not Vege Then
      Read(PasProg,Ch) ;
  End ;

  { A következő eljárás egy szöveg kiírása után vár a megadott karakter leütéséig: }
Procedure Varj(Szoveg: String; Kar: Char) ;
  Begin
    Write(Szoveg) ;
    Repeat
    Until Upcase(ReadKey) = Kar ;
    WriteLn ;
  End ;

  { A függvény a kapott szót nagybetűs formában adja vissza: }
Function Nagy(S : St15) : St15 ;
  Var
    I : Byte ;
  Begin
    For I := 1 to Length(S) Do
      S[I] := Upcase(S[I]) ;
    Nagy := S ;
  End ;
```

```
  { Az eljárás összegyűjti az S paraméterrel megadott Pascal program fenntartott szavait: }
Procedure Gyujtes(S : St15) ;
  Var
    Azon : St15 ;
    N    : Word ;
    Ch   : Char ;
  Begin
    Assign(PasProg,S) ;
    Reset(PasProg) ;
    Fszavak := [] ;
    Olvas(Ch) ;
    While Not Vege Do
      Begin
        Case Ch Of

            { Megjegyzés következik: }
          Megjegyzes_nyit : Begin
                              Repeat
                                Olvas(Ch) ;
                              Until (Ch = Megjegyzes_zar) Or Vege;
                              Olvas(Ch) ;
                            End ;

            { Karakterlánc következik: }
          Aposztrof       : Begin
                              Repeat
                                Olvas(Ch) ;
                              Until (Ch = Aposztrof) Or Vege ;
                              Olvas(Ch) ;
                            End ;

            { Azonosító következik: }
          'A'..'Z',
          'a'..'z',
          '_'             : Begin
                              Azon := '' ;
                              Repeat
                                  Azon := Azon + Ch ;
                                  Olvas(Ch) ;
                              Until Not (Ch In (Betuk + Szamok))
                                    Or Vege ;
                              Azon := Nagy(Azon) ;

                                { Azonosító keresése a deklarált foglalt szavak
                                között, nagybetűs formában: }
                              N := 1 ;
                              While (Azon <> Fenntartott[N]) And
                                    (N < FszavakSzama) Do
                                Inc(N) ;
```

```
                                    { A foglalt szavakat az Fszavak halmazban gyűjtjük: }
                                  If Azon = Fenntartott[N] Then
                                    Fszavak := Fszavak + [N] ;
                                End ;
            { Speciális karakterek következnek: }
          Else                  Repeat
                                  Olvas(Ch) ;
                                Until (Ch In Betuk) Or
                                      (Ch = Megjegyzes_nyit) Or
                                      (Ch = Aposztrof) Or Vege ;
        End ;
      End ;
    Close(PasProg) ;
  End ;

  { FŐPROGRAM: }
Begin
  ClrScr ;
    { A kérdéses halmazok kezdeti értékeinek beállítása: }
  Osszes := [] ;
  Kozos  := [1..FszavakSzama] ;
  Nincs  := [1..FszavakSzama] ;

    { Első Pascal program megkeresése az aktuális könyvtárban: }
  FindFirst('*.Pas',ReadOnly + Hidden + Archive,DirInfo) ;
    { A feldolgozás addig tart, amíg van PAS kiterjesztésű program, legyen az normál, csak olvasható, rejtett vagy archivált: }
  While DosError = 0 do
    Begin
        { A Pascal program foglalt szavainak kigyűjtése: }
      Gyujtes(DirInfo.Name) ;
        { A Pascal program foglalt szavainak kiírása: }
      WriteLn(DirInfo.Name,' fenntartott szavai : ') ;
      For I := 1 To FszavakSzama Do
        If I In Fszavak Then
          Write(Fenntartott[I]:20) ;
      WriteLn ;
      Varj('T(ovább)','T') ;
        { Az összes foglalt szóhoz hozzáadjuk a most kigyűjtött szavakat: }
      Osszes := Osszes + Fszavak ;
        { Az eddig mindegyik Pascal programban szereplő foglalt szavak és a most kigyűjtött szavak közös részét képezzük: }
      Kozos := Kozos * Fszavak ;
        { Az eddig egyik programban sem szereplő szavak és a jelenlegi Pascal programban nem szereplők közös részét képezzük: }
      Nincs := Nincs * ([1..FszavakSzama] - Fszavak) ;
```

```
          { Következő Pascal program: }
        FindNext(DirInfo) ;
      End ;

  WriteLn('A Pascal programokban ezek a fenntartott ',
          'szavak találhatók meg : ') ;
  For I := 1 To FszavakSzama Do
    If I In Osszes Then
      Write(Fenntartott[I]:20) ;
  WriteLn ;
  Varj('<ENTER>',#13) ;

  WriteLn('Az összes Pascal programban megtalálhatók : ') ;
  For I := 1 To FszavakSzama Do
    If I In Kozos Then
      Write(Fenntartott[I]:20) ;
  WriteLn ;
  Varj('<ENTER>',#13) ;

  WriteLn('Egyik Pascal programban sem találhatók meg : ') ;
  For I := 1 To FszavakSzama Do
    If I In Nincs Then
      Write(Fenntartott[I]:20) ;

  WriteLn ;
  Varj('<ENTER>',#13) ;
End.
```

MEMÓRIAKEZELÉS

1. feladat

Írjunk programot, mely egy rekordba beolvas adatokat, majd megjeleníti a rekordot bájtonként!

```
Program Mem1 ;

Var
  Rek     : Record
              B : Byte ;
              W : Word ;
              S : String[3] ;
            End ;

    { RekKep változót „rádefiniáljuk" Rek változóra, így az ugyanazon a memóriacímen
    kezdődik, külön memóriát nem foglal: }
  RekKep : Array [1..SizeOf(Rek)] Of Byte Absolute Rek ;

  I       : Byte ;

  { FŐPROGRAM: }
Begin
    { Rek mezőinek beolvasása: }
  Write('B= ') ; ReadLn(Rek.B) ;
  Write('W= ') ; ReadLn(Rek.W) ;
  Write('S= ') ; ReadLn(Rek.S) ;

    { A Rek által lefoglalt memóriaterület kiírása bájtonként: }
  For I := 1 To SizeOf(Rek) Do
    Write(RekKep[I]:5) ;
  WriteLn ;

  ReadLn ;
End.
```

2. feladat

Jelenítsük meg programunkkal adataink memóriatérképét!

```
Program Mem2 ;

Uses
  Crt ;
```

```
  { A globális változók és a típusos konstansok mind az adatszegmensben vannak: }
Var
  I         : Byte ;
  Adat1     : LongInt ;
  Adat2     : String[4] ;
  ZaroAdat  : Char ;

Const
  Adat3     : Real = 1 ;
  Adat4     : String[13] = 'Megtalálsz?' ;

  { FŐPROGRAM: }
Begin
  ClrScr ;

    { Egy memóriacím mindig két részből tevődik össze: az egyik a szegmenscím, a másik az
      ofszetcím (eltolás), mindkettő egy–egy két bájtos szó. A szegmenscím egy paragrafus
      sorszáma (egy paragrafus = 16 bájt, számozás 0–tól), míg az eltolás azt adja meg, hogy az
      adott memóriacím ettől a paragrafustól számított hányadik bájton van. memóriacím =
      szegmenscím * 16 + ofszetcím. Egy konkrét címet több szegmens:eltolás páros is meghatá-
      rozhat. A program típusos állandóit és globális változóit tartalmazó adatszegmens kezdőcíme
      a program futása során változatlan: }
  WriteLn('Adatszegmens szegmenscíme: ',DSeg) ;

    { Először a típusos konstansok kerülnek tárolásra: }
  WriteLn('Adat3     címe  : ',Seg(Adat3),':',Ofs(Adat3):3) ;
  WriteLn('Adat4     címe  : ',Seg(Adat4),':',Ofs(Adat4):3) ;

    { Aztán a globális változók következnek. Ide kerülnek a program által használt egységek
      globális változói is, pl. a System egység HeapPtr és RandSeed változói, valamint a Crt egység
      LastMode és TextAttr változói: }
  WriteLn('HeapPtr  címe  : ',Seg(HeapPtr),':',Ofs(HeapPtr):3) ;
  WriteLn('RandSeed címe  : ',Seg(RandSeed),':',Ofs(RandSeed):3) ;

  Adat1 := MaxLongInt ;
  Adat2 := 'Vége' ;
  WriteLn('Adat1     címe  : ',Seg(Adat1),':',Ofs(Adat1):3) ;
  WriteLn('Adat2     címe  : ',Seg(Adat2),':',Ofs(Adat2):3) ;

  WriteLn('LastMode címe  : ',Seg(LastMode),':',Ofs(LastMode):3) ;
  WriteLn('TextAttr címe  : ',Seg(TextAttr),':',Ofs(TextAttr):3) ;

    { Az adatok egészen a verem aljáig tartanak: }
  WriteLn('A veremszegmens szegmenscíme: ',SSeg) ;
  WriteLn ;

    { Adatszegmens megjelenítése az elejétől a programban tárolt utolsó adatig először
      bájtokban, majd karakteresen – ekkor a nem megjeleníthető karakterek helyén pont áll: }
  WriteLn('Az adatszegmens eleje bájtokban:') ;
```

```
  For I := 0 To Ofs(ZaroAdat) Do
    Write(Mem[DSeg:I]:4) ;
  WriteLn ;

  WriteLn('Az adatszegmens eleje karakterekben:') ;
  For I := 0 To Ofs(ZaroAdat) Do
    If Mem[DSeg:I] In [32..126,128..254]
    Then
      Write(Chr(Mem[DSeg:I]):4)
    Else
      Write('.':4) ;
  ReadLn ;
End.
```

3. feladat

Írjunk programot, mellyel a NumLock és CapsLock billentyűk állapotát változtathatjuk!

```
Program Mem3 ;

  { A rendszer a váltóbillentyűk állapotát a memória $417. és $418. bájtján tárolja. A $417.
    bájton a jobb Shift, bal Shift, Ctrl, Alt, Scroll Lock, Num Lock, Caps Lock és Ins billentyűk
    mindegyikének 1–1 bit felel meg a felsorolás sorrendjében (0. bit = Jobb shift, stb.). Ha a bit
    értéke 0, a billentyű kikapcsolt, 1 esetén bekapcsolt állapotú. Az alsó öt bitet nem ajánlatos
    megváltoztatni. }
Const
  NumLock    = $20 ;
  CapsLock   = $40 ;

Var

    { A ValtoBill változónak nem a fordító fog helyet foglalni, hanem az a $0:$417 memóriací-
      men lesz. Abszolút memóriacímen lévő változókat csak nagy körültekintéssel használjunk,
      hiszen így bármely rendszerterületet felülírhatunk, ami a rendszer összeomlásához is
      vezethet! }
  ValtoBill : Byte Absolute $0:$417 ;

  { Váltóbillentyű(k) bekapcsolása. Csak a felső hármat engedjük bekapcsolni: }
Procedure BeKapcsol(Bill: Byte) ;
  Begin
    ValtoBill := ValtoBill Or Bill And $E0 ;
  End ;
```

```
  { Váltóbillentyű(k) kikapcsolása. Csak a felső hármat engedjük bekapcsolni: }
Procedure KiKapcsol(Bill: Byte) ;
  Begin
    ValtoBill := ValtoBill And ($FF Xor Bill And $E0) ;
  End ;

  { FŐPROGRAM: }
Begin
  BeKapcsol(CapsLock) ;
  WriteLn('Most a Caps Lock aktív - <ENTER>') ;
  ReadLn ;

  BeKapcsol(NumLock) ;
  WriteLn('Most a Caps Lock és a Num Lock aktív - <ENTER>') ;
  ReadLn ;

  KiKapcsol(CapsLock+NumLock) ;
  WriteLn('Most a Caps Lock és a Num Lock inaktív - <ENTER>') ;
  ReadLn ;
End.
```

4. feladat

Írjunk programot, mellyel a színes, 25*80–as képernyő bizonyos karaktereit a képernyőmemóriába való közvetlen írással villogtatjuk!

```
Program Mem4 ;

Uses
  Crt ;

Type
  Kapcsolo = (Be,Ki,Valt) ;

    { A képernyő 25*80 karakterből áll, és minden karakterhez a képernyőmemóriában két bájt
      tartozik sorfolytonosan, az első a karakter ASCII kódja, a második az attribútuma. Az
      attribútum bájt 0..3 bitje a karakter színét, 4..6 a háttér színét, a 7 bit pedig a villogást
      határozza meg: }
  KepTip    = Array [1..25 * 80,1..2] Of Byte ;

Var

    { A Kep változót rádefiniáljuk a $B800:0 abszolút memóriacímre (nem monokróm monitor
      esetén itt kezdődik a képernyőmemória): }
  Kep : KepTip Absolute $B800:0 ;
  Kar : Char ;
```

```
  { A képernyő adott sorainak teleírása piros alapon fekete karakterekkel: }
Procedure TeleIr(ElsoSor,UtolsoSor: Byte; Kar: Char) ;
  Var
    I : Word ;
  Begin
    For I := (ElsoSor - 1) * 80 + 1 To UtolsoSor * 80 Do
      Begin
        Kep[I,1] := Byte(Kar) ;
          { A karakter háttérszíne piros, karakterszíne fekete legyen, és ne villogjon: }
        Kep[I,2] := 16 * Red + Black ;
      End ;
  End ;

  { A képernyő adott pozíciójától adott hosszon a karakterek villogtatásának be–, kikapcsolása
  illetve váltása: }
Procedure Villogas(Oszlop, Sor:. Byte ; Hossz: Word ;
                   Kapcs: Kapcsolo) ;
  Var
    Kezd,
    I      : Word ;
  Begin
      { A karakterhely kezdő sorszáma: }
    Kezd := (Sor - 1) * 80 + Oszlop ;
    For I := Kezd To Kezd + Hossz - 1 Do
        { Az attribútum bájt 7. bitjének állítása: 1 esetén villog, 0 esetén nem villog: }
      Case Kapcs Of
        Be    : Kep[I,2] := Kep[I,2] Or $80 ;
        Ki    : Kep[I,2] := Kep[I,2] And $7F ;
        Valt : Kep[I,2] := Kep[I,2] Xor $80 ;
      End ;
  End ;

  { FŐPROGRAM: }
Begin
  If LastMode = 7 Then
    Begin
      WriteLn('A képernyô címe $B000:0 !') ;
      ReadLn ;
      Halt ;
    End ;

  ClrScr ;
    { Teleírjuk 'b' betűkkel a képernyőt az 5. sortól a 15. sorig: }
  TeleIr(5,15,'b') ;

  GotoXY(5,24) ;
  Write('1-Balvillog 2-Balnemvillog 3-JobbVillog' +
        ' 4-Jobbnemvillog 5-Vált  Esc-Vége') ;
```

```
  Repeat
      Kar := ReadKey ;
      Case Kar Of
          { A 10. sor bal fele villogni fog: }
        '1' : Villogas(1,10,40,Be) ;
          { A 10. sor bal fele nem fog villogni: }
        '2' : Villogas(1,10,40,Ki) ;
          { A 10. sor jobb fele villogni fog: }
        '3' : Villogas(41,10,40,Be) ;
          { A 10. sor jobb fele nem fog villogni: }
        '4' : Villogas(41,10,40,Ki) ;
          { A 10. sor karakterei közül amelyik villogott, az most nem fog villogni, s amelyik
            nem villogott, most villogni fog: }
        '5' : Villogas(1,10,80,Valt) ;
      End ;
  Until Kar = #27 ;

  ClrScr ;
End.
```

5. feladat

Egy 64 kB nagyságú, bájtokból álló tömböt töltsünk fel véletlen értékekkel, majd számítsuk ki az elemek átlagát!

```
Program Mem5 ;

  { A következő direktívával a program memória–foglalási jellemzőit állítjuk be: a verem számára
    $4000, azaz 16384 bájtot foglalunk le, a heap számára minimálisan $FFFF, azaz 65535 bájtnyi
    memóriára van szükség (ha nincs ennyi, a program nem tölthető be). A heap az egész szabad
    memóriát lefoglalja (maximum 655360 bájtot): }
{$M $4000,$FFFF,655360}

Type
    { A Turbo Pascal–ban ennél nagyobb méretű típus nem adható meg: }
  TombTip = Array[1..$FFFF] Of Byte ;

Var
    { A tömb számára a heap–ben foglalunk helyet, mert az adatszegmensben csak összesen 64
      kB–nyi adatunk lehet, s már a System egység adatai is itt vannak. Tomb egy típusos mutató,
      egy TombTip típusú változóra fog majd mutatni. Az adatszegmensben 4 bájtnyi helyet foglal,
      ide kerül majd a program által a heap–ben lefoglalt tömb címe: }
  Tomb    : ^TombTip ;
  I       : Word ;
```

```
  Osszeg : LongInt ;

  { FŐPROGRAM: }
Begin
    { TombTip típusú változó lefoglalása a heap–ben. Tomb erre a változóra fog mutatni,
      Tomb–re hivatkozás a továbbiakban – Tomb^: }
  New(Tomb) ;

    { Tomb^ feltöltése véletlen értékekkel: }
  WriteLn('Várj, dolgozom!') ;
  For I := 1 To $FFFF Do
    Tomb^[I] := Random(256) ;

    { Tomb^ átlagának kiszámítása: }
  Osszeg := 0 ;
  For I := 1 To $FFFF Do
    Inc(Osszeg,Tomb^[I]) ;
  WriteLn('A tömb átlaga: ',Osszeg/$FFFF:8:2) ;

  ReadLn ;
End.
```

6. feladat

Írjunk programot, mellyel egy karakterláncot elmentünk a heap-be, majd visszahozzuk onnan!

```
Program Mem6 ;

Var
  Karlanc     : String ;
  LancMutato  : Pointer ;
  Hossz       : Byte ;

  { FŐPROGRAM: }
Begin
    { A karakterlánc beolvasása, majd elmentése a heap-be. A mentéshez pontosan annyi bájtot
      folalunk le a heap-ből, amennyire szükség van. LancMutato típus nélküli mutató, a foglalt
      terület elejére mutat: }
  WriteLn('Írjon be egy karakterláncot!') ;
  ReadLn(Karlanc) ;
  Hossz := Length(Karlanc) + 1 ;
  GetMem(LancMutato,Hossz) ;
  Move(Karlanc,LancMutato^,Hossz) ;
```

```
    { A karakterláncot most beolvasással felülírjuk, majd helyébe a mentett értéket visszahozzuk,
      s a heap–ben lefoglalt helyet felszabadítjuk: }
  WriteLn('írjon be egy másik karakterláncot!') ;
  ReadLn(Karlanc) ;

  WriteLn('A karakterláncok:') ;
  WriteLn(Karlanc) ;
  Move(LancMutato^,Karlanc,Hossz) ;
  WriteLn(Karlanc) ;

  FreeMem(LancMutato,Hossz) ;
  ReadLn ;
End.
```

7. feladat

Vegyünk fel egy típussal rendelkező és egy típussal nem redelkező mutatót, melyek mindegyike egy adatszegmensbeli rekordra mutat. A rekord adatait e mutatókon keresztül érjük el!

```
Program Mem7 ;

Type
  SzemelyTip = Record
                 Sorszam : Word ;
                 Nev     : String[20] ;
               End ;

Var
  Szemely      : SzemelyTip ;
    { Mutató típusú változó, mely egy SzemelyTip típusú változóra mutathat. A mutatót a mutatni
      kívánt adatra kell majd irányítani: }
  SzemelyMut1 : ^SzemelyTip ;
    { Mutató típusú változó, melyre nem jellemző, hogy milyen típusú adatra mutat (típusnélküli
      mutató): }
  SzemelyMut2 : Pointer ;

  { FŐPROGRAM: }
Begin
    { Mindkét mutató az adatszegmensben lévő Szemely rekordra mutasson! Egy változó vagy
      szubrutin címét a @ művelettel megkaphatjuk: }
  SzemelyMut1 := @Szemely ;
  SzemelyMut2 := @Szemely ;
```

```
    { Szemely rekord adatainak beolvasása: }
  ReadLn(Szemely.Sorszam) ;
  ReadLn(Szemely.Nev) ;

    { Mivel SzemelyMut1 a Szemely változóra mutat, SzemelyMut^ maga a Szemely rekord. Így
      kiíráskor annak az adatai jelennek meg: }
  WriteLn(SzemelyMut1^.Sorszam) ;
  WriteLn(SzemelyMut1^.Nev) ;

    { Mivel SzemelyMut2 típus nélküli mutató, típusrádefiniálással tudunk a mutatott típusnélküli
      változónak típust adni. A kiírás itt is a beolvasott Szemely rekord adatait jeleníti meg: }
  WriteLn(SzemelyTip(SzemelyMut2^).Sorszam) ;
  WriteLn(SzemelyTip(SzemelyMut2^).Nev) ;
  ReadLn ;
End.
```

8. feladat

Írjunk programot, mely paraméterként megadott számú (maximum 65000) bájtot lefoglal a heap–ből (ha nincs paraméter, nem foglalunk memóriát), majd saját memória–térképének jellemzőit kiírja: a kód–, az adat–, a verem–, a foglalt és a szabad heap–terület méretét, és azok címeit, valamint a teljes memória méretét!

```
Program Mem8 ;

  { Veremméret 4096 bájt legyen, a heap minimálisan 65000, maximálisan 655360 bájtnyi helyet
    foglaljon: }
{$M 4096,65000,655360}

Uses
  Crt ;

Type
  Str4 = String[4] ;
  Str8 = String[8] ;

Const
  TH : LongInt = 16 ;
  B  = ' bájt' ;

Var
    { Típus nélküli mutató. Mutato^ típus nélküli változó lesz: }
  Mutato    : Pointer ;
  Parameter : LongInt ;
  Ok        : Integer ;
```

```
  { FŐPROGRAM: }
Begin
  ClrScr ;
    { Ha van parancssor–paraméter: }
  If ParamCount > 0 Then
    Begin
        { Parancssor–paraméter numerikussá alakítása: }
      Val(ParamStr(1),Parameter,Ok) ;
      If (Ok = 0) And (Parameter >= 0) And (Parameter <= 65000)
      Then
          { Az átalakítás sikeres volt, Parameter darab bájt lefoglalása a heap–ből. Mutato
            a foglalt memória kezdetére fog mutatni: }
        GetMem(Mutato,Parameter)
      Else
          { Az átalakítás sikertelen – hibaüzenet és leállás: }
        Begin
          WriteLn('Hibás paraméter - 0 és 65000 közötti szám' +
                  ' lehet') ;
          ReadLn ;
          Halt ;
        End ;
    End ;

    { Az egyes memóriarészek kiírása. Mivel TH LongInt típusú, a megfelelő szorzatok nem
      fognak túlcsordulni: }
  WriteLn('A program memóriafoglalási jellemzôi:') ;
  WriteLn ;
  WriteLn('Program kezdete (PSP címe)  : ',TH * PrefixSeg) ;
  WriteLn ;
  WriteLn('Kódszegmens címe            : ',TH * Cseg) ;
  WriteLn('Kódszegmens mérete          : ',
          TH * (DSeg - CSeg):6,B) ;
  WriteLn ;
  WriteLn('Adatszegmens címe           : ',TH * DSeg) ;
  WriteLn('Adatszegmens mérete         : ',
          TH * (SSeg - DSeg):6,B) ;
  WriteLn ;
  WriteLn('Veremszegmens címe          : ',TH * SSeg) ;
  WriteLn('Verem elsô szabad bájtja    : ',TH * SSeg + SPtr) ;
  WriteLn('Verem teljes mérete         : ',
          TH * (OvrHeapOrg - SSeg):6,B) ;
  WriteLn('Verem aktuális mérete       : ',
          TH * (OvrHeapOrg - SSeg) - SPtr:6,B) ;
  WriteLn ;
  WriteLn('Overlay puffer címe         : ',TH * OvrHeapOrg) ;
  WriteLn('Overlay puffer mérete       : ',
          TH * (OvrHeapEnd - OvrHeapOrg):6,B) ;
  WriteLn ;
```

```
  WriteLn('Foglalt heap terület        : ',
          (TH * Seg(HeapPtr^) + Ofs(HeapPtr^)) -
          (TH * Seg(HeapOrg^) + Ofs(HeapOrg^)):6,B) ;

  WriteLn('Szabad heap terület         : ',
          (TH * Seg(HeapEnd^) + Ofs(HeapEnd^)) -
          (TH * Seg(HeapPtr^) + Ofs(HeapPtr^)):6,B) ;
  WriteLn ;
  WriteLn('MemAvail függvény értéke    : ',MemAvail) ;

  ReadLn ;
End.
```

9. feladat

Írjunk programot, mely a memória adott területének tartalmát kiírja (DUMP)!

```
Program Mem9 ;

Uses
  Crt ;

Type
  Str2 = String[2] ;
  Str4 = String[4] ;

Var
  S                : String ;
  Ok               : Integer ;
  B,
  Sor              : Byte ;
  C                : Char ;
  KezdoParagrafus,
  VegzoParagrafus : Word ;

  { Bájt / hexadecimális átalakító: }
Function HexaBajt(B : Byte) : Str2 ;
  Const
    HexKar : Array[0..15] Of Char = '0123456789ABCDEF' ;
  Begin
    HexaBajt := HexKar[B And $F0 Shr 4] + HexKar[B And $0F] ;
  End ;
```

```
  { Szó / hexadecimális átalakító: }
Function HexaSzo(W : Word) : Str4 ;
  Begin
    HexaSzo := HexaBajt(Hi(W)) + HexaBajt(Lo(W)) ;
  End ;

  { FŐPROGRAM: }
Begin
    { Főciklus: }
  Repeat
      ClrScr ;
        { Kezdő paragrafus bekérése: }
      Repeat
          GotoXY(1,1) ;
          ClrEol ;
          Write('Kezdô paragrafus : ') ;
          ReadLn(S) ;
          If S <> '' Then
            Val(S,KezdoParagrafus,Ok) ;
        { Kilépés, ha az átalakítás sikeres volt és az érték a címezhető tartományba esik, vagy
          csak <ENTER> –t ütöttek: }
      Until (Ok = 0) And (KezdoParagrafus < Seg(HeapEnd^)) Or
            (S = '') ;

        { Ha nem csak <ENTER>–t ütöttek: }
      If S <> '' Then
        Begin
            { Végző paragrafus bevitele: }
          Repeat
              GotoXY(1,2) ;
              ClrEol ;
              Write('Végzô paragrafus : ') ;
              ReadLn(S) ;
              Val(S,VegzoParagrafus,Ok) ;
            { Kilépés, ha az átalakítás sikeres volt és az érték a címezhető tartományba esik
              és a végzőparagrafus nagyobb, mint a kezdő: }
          Until (Ok = 0) And (VegzoParagrafus <= Seg(HeapEnd^))
                And (VegzoParagrafus > KezdoParagrafus) ;

            { Fejléc kiírása: }
          GotoXY(1,6) ;
          Write('Par.  ' + '00 01 02 03 04 05 06 07 08 ' +
                '09 0A 0B 0C 0D 0E 0F',' 0123456789ABCDEF') ;
          Sor := 0 ;
          Window(1,8,80,25) ;
```

```
        C := #0 ;
            { Amíg a kezdőparagrafus kisebb–egyenlő, mint a végzőparagrafus és nem
              <ESC>–et ütöttek: }
        While (KezdoParagrafus <= VegzoParagrafus) And
              (C <> ^[) Do
          Begin
                { A sor (paragrafus) első bájtjának címe: }
            Write(HexaSzo(KezdoParagrafus),'  ') ;
                { A paragrafus bájtjainak értéke: }
            For B := 0 To 15 Do
              Write(HexaBajt(Mem[KezdoParagrafus:B]),' ') ;
            Write(' ') ;
                { A paragrafus bájtjainak megfelelő karakterek: }
            For B := 0 To 15 Do
              If Chr(Mem[KezdoParagrafus:B]) In [' '..#254]
              Then
                    { Megjeleníthető karakter: }
                Write(Chr(Mem[KezdoParagrafus:B]))
              Else
                    { Nem megjeleníthető karakter: }
                Write('.') ;
            WriteLn ;

                { Következő paragrafus: }
            Inc(KezdoParagrafus) ;
            Inc(Sor) ;
                { Ha 16 sor van a képernyőn, vagy vége az intervallumnak: }
            If (Sor = 17) Or
               (KezdoParagrafus > VegzoParagrafus) Then
              Begin
                    { Várakozás egy billentyű leütésére: }
                C := ReadKey ;
                    { Ablak törlése: }
                ClrScr ;
                Sor := 0 ;
              End ;
          End ;
      End ;

        { Ablak visszaállítása: }
    Window(1,1,80,25) ;
  { Kilépés, ha kezdő paragrafusnak csak <ENTER> –t ütöttek: }
Until S = '' ;
End.
```

10. feladat

Az aktuális szöveges képernyő tartalmának kinyomtatása (PrintScreen). A program nem ellenőrzi, hogy van–e nyomtató a rendszerben!

```
Program Mem10 ;

{$X+}

Uses
  Crt,
  Printer ;

Const
  MaxOszlop = 80 ;

Type
    { A képernyőmemóriában minden karakterhelyhez 2 bájt tartozik: a karakter ASCII kódja és
    attribútuma: }
  KarHely = Record
              Kar  : Char ;
              Attr : Byte ;
            End ;
  SorTip  = Array [1..MaxOszlop] Of KarHely ;

Var
  SorMut      : ^SorTip ;      { Mutató a képernyőmemória egy sorára }
  KepKezdCim  : Word ;         { Képernyőmemória szegmenscíme }
  Sor,
  Oszlop,
  I,
  J           : Byte ;

  { FŐPROGRAM: }
Begin
    { Írunk a képernyőre. A képernyő tartalmát majd egy billentyű lenyomása után papíron
    láthatjuk: }
  For I := 1 To 25 Do
    Begin
      GotoXY(1,I) ;
      Write(I:3,'. sor') ;
    End ;
  ReadKey ;
```

```
    { Monokróm monitorvezérlő esetén a képernyőmemória kezdőcíme $B000:$0, színes monitorvezérlő esetén $B800:$0: }
  If Lo(LastMode) = 7
  Then
    KepKezdCim := $B000
  Else
    KepKezdCim := $B800 ;

    { A képernyő sorainak és oszlopainak kiszámítása a WindMax változóból. Windmax tartalmazza az aktuális ablak jobb alsó sarkának koordinátáit; a koordináták 0–tól indulnak, X = alsó bájt; Y = felső bájt: }
  Sor := Hi(WindMax) + 1 ;
  Oszlop := Lo(WindMax) + 1 ;

    { A képernyőmemóriában lévő karakterek kinyomtatása. Ugyanazt a hatást érjük el, mintha lenyomnánk a PrintScreen billenyűt. Vigyázat! Az attribútum bájtot figyelmen kívül hagyjuk, vagyis előfordulhat, hogy a képernyőn nem látszik semmi, mégis történik nyomtatás (ha a karakter és háttérszín megegyezik). A PrintScreen is így működik:}
  For I := 1 To Sor Do
    Begin
        { Az I. sor címének kiszámítása: }
      SorMut := Ptr(KepKezdCim,(I - 1) * Oszlop * 2) ;
        { Az I. sor nyomtatása: }
      For J := 1 To Oszlop Do
        Write(Lst,SorMut^[J].Kar) ;
      WriteLn(Lst) ;
    End ;
End.
```

11. feladat

Készítsünk egy ablakot a képernyőn, írjunk arra rá, majd állítsuk vissza az eredeti képernyőt! (Ablaktechnika)

```
Program Mem11 ;

{$M 16384,0,655360}

Uses
  Crt ;

Type
  St80     = String[80] ;
```

```
  KarHely = Record
              Kar  : Char ;
              Attr : Byte ;
            End ;
  KepTip  = Array[1..25,1..80] Of KarHely ;

Var
  Kep         : ^KepTip ;        { Mutató a képernyőmemóriára }
  AblakCim    : Pointer ;        { Az elmentett ablak címe a heap–ben }
  EX,EY,                         { Előző ablak kurzor helyének koordinátái }
  ETextAttr   : Byte ;           { Előző szöveges attribútum }
  EWindMin,                      { Előző ablak bal felső sarok }
  EWindMax    : Word ;           { Előző ablak jobb alsó sarok }
  I           : Byte ;

  { Az eljárás a képernyő (X,Y) koordinátájától kezdve kiírja az St karakterláncot. Az írás
  közvetlen a képernyőmemóriába történik, a pozícionálás ablakrelatív. A karakterek színét a
  TextAttr változó határozza meg. Az eljárás gyorsabb a Write eljárásnál: }
Procedure MemIras(X, Y: Byte ; St: St80) ;
  Var
    L,
    I : Byte ;
  Begin
      { X és Y abszolút koordinátáinak kiszámolása: Y relatív koordináta + WindMin felső
      bájtja: }
    Y := Y + WindMin Shr 8 ;
      { X relatív koordináta + WindMin alsó bájtja: }
    X := X + WindMin And $FF ;

      { St bemásolása a képernyőmemóriába: }
    L := Length(St) ;
    For I := 1 To L do
      Begin
        Kep^[Y,X - 1 + I].Attr := TextAttr ;
        Kep^[Y,X - 1 + I].Kar := St[I] ;
      End ;
  End ;

  { Adott karakterből adott hosszúságú karakterlánc készítése: }
Function Vonal(Jel: Char ; N: Byte): St80 ;
  Var
    I : Byte ;
  Begin
    For I := 1 to N Do
      Vonal[I] := Jel ;
    Vonal[0] := Char(N) ;
  End ;
```

```
  { A függvény visszaadja azt a mutatót, melynek címe egy adott mutatótól Rel bájttal tér el: }
Function Mutato(P : Pointer ; Rel : LongInt) : Pointer ;
  Var
    L : LongInt ;
  Begin
    L := LongInt(16) * Seg(P^) + Ofs(P^) + Rel ;
    Mutato := Ptr(L Shr 4,L And $0F) ;
  End ;

  { A készítendő ablak bal felső sarka (Xk,Yk); szélessége Szel, magassága Mag. Az utolsó két
    paraméter adja az ablak színeit. Az eljárás először elmenti a heap–be azt a képernyőterületet,
    melyet az ablak letakar, majd az adott színekkel egy üres, keretezett ablakot készít: }
Procedure AblakKeszit(Xk,Yk,Szel,Mag,HatterSzin,KarSzin: Byte) ;
  Var
    Hmutato  : Word ;          { A heap aktuális relatív mutatója }
    Atvitel,                   { Az átvitel egysége }
    I        : Byte ;
    S        : st80 ;
  Begin
      { Az ablak tárolásához elegendő hely lefoglalása a heap–ben, majd az ablak tartalmának
        átmásolása a heap–be, soronként: }
    Atvitel := Szel * 2 ;
    GetMem(AblakCim,Atvitel * Mag) ;
    Hmutato := 0 ;
    For I := Yk To Yk + Mag - 1 Do
      Begin
        Move(Kep^[I,Xk],Mutato(AblakCim,Hmutato)^,Atvitel) ;
        Inc(Hmutato,Atvitel) ;
      End ;

      { Az ablak színeinek beállítása, a keret elkészítése. Monokróm monitor esetén nincsenek
        színek: }
    If LastMode = Mono
    Then
      TextAttr := $07
    Else
      TextAttr := HatterSzin Shl 4 Or KarSzin ;

    Window(Xk,Yk,Xk + Szel - 1,Yk + Mag - 1) ;
    S := #201 + Vonal(#205,Szel - 2) + #187 ;
    MemIras(1,1,S) ;
    S := #186 + Vonal(' ',Szel - 2) + #186 ;
    For I := 2 To Mag - 1 Do
      MemIras(1,I,S) ;
    S := #200 + Vonal(#205,Szel - 2) + #188 ;
    MemIras(1,Mag,S) ;

      { Ablak beállítása a kereten belülre, a kurzor a bal felső sarokba kerül: }
    Window(Xk + 1,Yk + 1,Xk + Szel - 2,Yk + Mag - 2) ;
    GotoXY(1,1) ;
  End ;
```

```
  { Az eljárás visszahozza a heap–ből az elmentett képernyőterületet, melyet az ablak letakart. }
Procedure AblakVissza(Xk, Yk, Szel, Mag: Byte) ;
  Var
    Hmutato   : Word ;          { A heap aktuális relatív mutatója }
    Atvitel,                    { Az átvitel egysége }
    I         : Byte ;
  Begin
      { Az elmentett ablak tartalmának visszamásolása a képernyőmemóriába, soronként: }
    Atvitel := Szel * 2 ;
    Hmutato := 0 ;
    For I := Yk To Yk + Mag - 1 Do
      Begin
        Move(Mutato(AblakCim,Hmutato)^,Kep^[I,Xk],Atvitel) ;
        Inc(Hmutato,Atvitel) ;
      End ;
      { Az elmentett ablak által lefoglalt memóriaterület felszabadítása: }
    FreeMem(AblakCim,Hmutato) ;
  End ;

  { FŐPROGRAM: }
Begin
    { Az aktuális ablak koordinátáinak, színeinek és a kurzor helyének megjegyzése: }
  EWindMin := WindMin ;
  EWindMax := WindMax ;
  EtextAttr := TextAttr ;
  EX := WhereX ;
  EY := WhereY ;

    { Ha monokróm a monitor, akkor a képernyőmemória szegmenscíme $B000, egyébként
      $800–al több: }
  Kep := Ptr($B000 + Ord(LastMode <> Mono) * $800,0) ;

    { Ablak elkészítése, írás az ablakba: }
  AblakKeszit(5,10,40,15,Brown,Black) ;
  For I := 1 to 10 Do
    MemIras(1,I,'  Micsoda ablaktechnika !!!') ;
  MemIras(1,12,'Vigyázat! A kurzor fent van! <ENTER>') ;
  Repeat
  Until ReadKey = #13 ;

    { Ablak eltüntetése, vagyis az eredeti képernyő visszaállítása: }
  AblakVissza(5,10,40,15) ;
    { A program indításakor érvényben lévő ablak koordinátáinak, színeinek és a kurzor helyének
      visszaállítása: }
  WindMin := EWindMin ;
  WindMax := EWindMax ;
  TextAttr := ETextAttr ;
  GotoXY(EX,EY) ;
```

```
  WriteLn('<ENTER>') ;
  Repeat
  Until ReadKey = #13 ;
End.
```

12. feladat

Írjunk programot, mely grafikusan kirajzolja saját memóriatérképét: a kód–, adat–, verem–, foglalt és szabad heap–terület relatív méretét! A program paraméterével megadhatjuk a lefoglalandó dinamikus terület méretét paragrafusokban.

```
Program Mem12 ;

{$M 16384,0,655360}

Uses
  Graph ;

Var
  Meghajto,
  Uzemmod,
  GrafHiba,
  Ok           : Integer ;
  Oszto,
  I            : Word ;
  Mutato       : Pointer ;
  Parameter    : LongInt ;

    { Csupán helyfoglalás miatt, egyébként az adatszegmens túl kicsi lenne: }
  NagyValtozo : Array[1..60000] Of Byte ;
  C           : Char ;

  { Két szegmenscím közötti memóriarésznek megfelelő terület kirajzolása adott mintával és
  szöveggel: }
Procedure MeretRajzolo(AlsoHatar,FelsoHatar,Minta : Word ;
                       Szoveg : String) ;
  Var
    Also,
    Felso  : Word ;
    S      : String ;
    SAlso,
    SFelso : String[10] ;
```

```
  Begin
    If AlsoHatar >= FelsoHatar Then
      Exit ;
      { Az alsó és felső határvonalak függőleges pozíciójának kiszámítása: }
    Also := GetMaxY - (AlsoHatar - CSeg) Div Oszto ;
    Felso := GetMaxY - (FelsoHatar - CSeg) Div Oszto ;
    Str(FelsoHatar,SFelso) ;
    Str(AlsoHAtar,SAlso) ;
    Str((FelsoHatar - AlsoHatar),S) ;
    S := Szoveg + ' ' + S + ' paragrafus (' + SAlso + ' - ' +
         SFelso + ')' ;
      { Négyszög kirajzolása: }
    Rectangle(0,Also,GetMaxX,Felso) ;
      { Ha a négyszögbe belefér a szöveg: }
    If Also - Felso > 15 Then
      Begin
          { Szöveghatároló belső négyszög kirajzolása: }
        Rectangle(GetMaxX Div 2 - TextWidth(S) Div 2 - 10,
                  (Also + Felso) Div 2 - TextHeight(S),
                  GetMaxX Div 2 + TextWidth(S) Div 2 + 10,
                  (Also + Felso) Div 2 + TextHeight(S)) ;
          { Szöveg kiírása: }
        SetTextJustify(CenterText,CenterText) ;
        OutTextXY(GetMaxX Div 2,(Also + Felso) Div 2,S) ;
      End ;

      { Külső négyszög kitöltése: }
    SetFillStyle(Minta,GetColor) ;
    FloodFill(1,Also - 1,GetColor) ;
  End ;

  { FŐPROGRAM: }
Begin
    { Grafikus üzemmód indítása. A grafika is használ heap–et! }
  Meghajto := Detect ;
  InitGraph(Meghajto,Uzemmod,'') ;
  GrafHiba := GraphResult ;
    { Ha hiba történt a grafikus üzemmód indításakor: }
  If GrafHiba <> grOk Then
    Begin
      WriteLn(GraphErrorMsg(GrafHiba)) ;
      ReadLn ;
      Halt ;
    End ;
```

```
    { Az egy képpontnak megfelelő paragrafusszám meghatározásához szükséges osztó kiszámí-
    tása: }
  Oszto := (Seg(HeapEnd^) - CSeg) Div GetMaxY + 1 ;

    { Ha van parancssor–paraméter: }
  If ParamCount > 0 Then
    Begin
        { Parancssor–paraméter numerikussá alakítása: }
      Val(ParamStr(1),Parameter,Ok) ;
      If (Ok = 0) And (Parameter > 0) And
         (Parameter * 16 <= MemAvail)
      Then
          { Parameter darab paragrafus lefoglalása a heap–ből : }
        Begin
          For I := 1 To Parameter Do
            GetMem(Mutato,16) ;
        End
      Else
          { Az átalakítás sikertelen – hibaüzenet és leállás: }
        Begin
          CloseGraph ;
          WriteLn('Hibás paraméter - nincs ennyi heap') ;
          ReadLn ;
          Halt ;
        End ;
    End ;

    { Az egyes memóriarészek kirajzolása: }
  MeretRajzolo(CSeg,DSeg,LineFill,'KOD') ;
  MeretRajzolo(DSeg,SSeg,LtSlashFill,'ADAT') ;
  MeretRajzolo(SSeg,OvrHeapOrg,BkSlashFill,'VEREM') ;
  MeretRajzolo(Seg(HeapOrg^),Seg(HeapPtr^),WideDotFill,
               'FOGLALT HEAP') ;
  MeretRajzolo(Seg(HeapPtr^),Seg(HeapEnd^),XHatchFill,
               'SZABAD HEAP') ;
  ReadLn ;
    { Grafikus üzemmód lezárása: }
  CloseGraph ;
End.
```

13. feladat

Foglaljunk le 10 véletlen méretű blokkot a heap–ből, majd véletlenszerűen szabadítsunk fel közülük néhányat! Minden felszabadítás után adjunk információt a szabad területek listájáról, valamint a heap tetején levő egybefüggő területről!

```
Program Mem13 ;

{$M 16384,0,655360}

Uses
  Crt ;

Type
  MutatoRek         = Record
                        Lo, Hi : Word ;
                      End ;
```

{ FreeList mindig a heap első szabad területére mutat. Ha nincs szabad terület, FreeList értéke HeapPtr. Minden szabad terület (lyuk) eleje egy „szabadrekord", mely a következő lyukra mutat, s tartalmazza a lyuk méretét. Meret Pointer típusú, melyből a BlokkMeret függvény számítja ki a méretet bájtokban. Az utolsó szabadrekord a heap tetejére mutat (HeapPtr): }

```
  SzabadRekordMut = ^SzabadRekord ;
  SzabadRekord    = Record
                      Kovetkezo : SzabadRekordMut ;
                      Meret     : Pointer ;
                    End ;
  FoglaltRekord   = Record
                      Cime   : Pointer ;
                      Merete : Word ;
                    End ;
  Str5            = String[5] ;

Var
  FoglaloTomb     : Array[1..10] Of FoglaltRekord ;
  CiklusValt      : Byte ;
  VeletlenSzam    : Word ;

Function Hexa(Cim : Word) : Str5 ;
  Var
    S : Str5 ;
    I : Word ;
  Const
    H : Array[0..15] Of Char = '0123456789ABCDEF' ;
  Begin
    S := '' ;
    I := 4096 ;
    Repeat
        S := S + H[Cim Div I] ;
        Cim := Cim - I * (Cim Div I) ;
        I := I Div 16 ;
    Until I = 0 ;
    Hexa := '$'+ S ;
  End ;
```

```
  { A függvény a szabadrekordban tárolt Pointer típusú méretből egész értéket állít elő: }
Function BlokkMeret(Meret: Pointer): LongInt ;
  Begin
    BlokkMeret := LongInt(MutatoRek(Meret).Hi) * 16 +
                  MutatoRek(Meret).Lo ;
  End ;

Procedure SzabadListaInfo ;
  Var
    CiklusValt : Byte ;
    LyukSzam   : Word ;
    Aktualis   : SzabadRekordMut ;
  Begin
    Window(1,12,80,24) ;
    ClrScr ;
    WriteLn('A szabad blokkok listája:') ;
    LyukSzam := 0 ;
    Aktualis := FreeList ;
      { Amíg van még szabad terület (lyuk): }
    While Aktualis <> HeapPtr Do
      Begin
          { Kiírjuk a szabad terület címét és méretét: }
        With SzabadRekord(Aktualis^) Do
          Begin
            Write('Kezdôcím: ',
                  Hexa(MutatoRek(Aktualis).Hi),':',
                  Hexa(MutatoRek(Aktualis).Lo)) ;
            Write('  mérete: ',BlokkMeret(Meret)) ;
            WriteLn ;
            Aktualis := Kovetkezo ;
            Inc(LyukSzam) ;
          End ;
      End ;
    WriteLn('Összesen: ',LyukSzam,' ''lyuk''') ;
    WriteLn ;
    WriteLn('A heap tetején levô egybefüggô szabad terület' +
            ' mérete:') ;
    Write(Hexa(Seg(HeapPtr^)),':',Hexa(Ofs(HeapPtr^)),' -tól ') ;
    Write(Hexa(Seg(HeapEnd^)),':',Hexa(Ofs(HeapEnd^)),' -ig ') ;
    WriteLn((Seg(HeapEnd^) * 16 + Ofs(HeapEnd^)) -
            (Seg(HeapPtr^) * 16 + Ofs(HeapPtr^)):5,' bájt') ;

    Window(1,1,80,25) ;
  End ;

  { FŐPROGRAM: }
Begin
  ClrScr ;
```

```
    { 10 véletlen méretű blokkot foglalunk a heap–ből. A blokkok címeit és méreteit megjegyezzük: }
  Randomize ;
  For CiklusValt := 1 To 10 Do
    With FoglaloTomb[CiklusValt] Do
      Begin
        Merete := (Random(10000) + 1) * 2 ;
        If MemAvail < Merete Then
          Merete := 0 ;
        GetMem(Cime,Merete) ;
        GotoXY(1,CiklusValt) ;
        Write(CiklusValt:2,'. blokk - címe: ',
              Hexa(Seg(Cime^)),':',Hexa(Ofs(Cime^)),
              ' - mérete: ',Merete:5,' bájt') ;
      End ;

    { Felszabadítunk valahány területet a lefoglaltak közül. Minden felszabadítás után információt adunk a szabad listáról: }
  For CiklusValt := 1 To 10 Do
    Begin
      VeletlenSzam := Random(10) + 1 ;
      With FoglaloTomb[VeletlenSzam] Do
        If Merete > 0 Then
          Begin
            FreeMem(Cime,Merete) ;
            GotoXY(55,VeletlenSzam) ;
            Write('blokk felszabadítva') ;
            SzabadListaInfo ;
            GotoXY(1,24) ;
            Write('További felszabadítás - <ENTER>') ;
            ReadLn ;
                { Kinullázzuk, hogy legközelebb ne egy már felszabadított területet szabadítsunk fel újra: }
            Merete := 0 ;
          End ;
    End ;

    GotoXY(1,24) ;
    ClrEol ;
    Write('Vége - <ENTER>') ;
    ReadLn ;
End.
```

14. feladat

Foglaljunk le a heap–ből adott méretű blokkokat! Ha már csak kisebb terület szabad a heap–ben, mint a lefoglalandó, a hibát saját függvénnyel detektáljuk, és kezeljük le!

```
Program Mem14 ;

{$X+,M 16384,0,655360}

Uses
  Crt ;

Var
  Mutato       : Pointer ;
  FoglaloMeret : Word ;
  BlokkSzam    : Word ;

  { A HeapError által mutatott heap–hiba lekezelő függvény automatikusan meghívódik, ha a
    New vagy GetMem eljárásnak nem sikerült lefoglalni a kívánt méretű blokkot. Ekkor a
    programozó dönti el, mi legyen a teendő: ha a függvénynek 0 értéket adunk: futási hiba fog
    előállni; 1 esetén a mutató értéke Nil lesz; míg ha SajatHeapHiba 2 értékkel tér vissza, a New
    vagy a GetMem újra kísérletet tesz a helyfoglalásra. A függvény akkor is meghívásra kerül, ha
    nincs a heap–mutató alatt szabad blokk (ekkor Meret=0). Ez nem hiba, csak jelzi, hogy a
    heap–mutató felfelé mozgott. A heap–hiba lekezelő függvényt távoli hívással kell deklarálni: }
Function SajatHeapHiba (Meret: Word): Integer ; Far ;
  Begin
    If Meret = 0 Then
      Exit ;
    GotoXY(1,10) ;
    WriteLn('Hiba a(z) ',BlokkSzam,'. blokk lefoglalása közben');
    WriteLn('A heap-ben nincs ',Meret,
            ' bájt méretû szabad terület,') ;
    WriteLn('a legnagyobb összefüggô terület mérete: ',
             MaxAvail,' bájt') ;
    SajatHeapHiba := 1 ;
  End ;

  { FŐPROGRAM: }
Begin
  ClrScr ;
    { Saját heap–hiba lekezelő függvény installálása: }
  HeapError := @SajatHeapHiba ;

    { Az információkat minden lefoglalás alkalmával ki fogjuk írni: }
  WriteLn('Foglalt blokk száma           : ') ;
  WriteLn('A foglalt heap mérete         : ') ;
  WriteLn('Összes szabad terület         : ') ;
  WriteLn('Legnagyobb összefüggô terület : ') ;

  FoglaloMeret := 10000 ;
  BlokkSzam := 0 ;
```

```
    { Amíg van hely a heap–ben: }
  Repeat
        { Minden foglalás után kiírjuk a foglalt heap nagyságát, valamint az összes és az
          egybefüggő legnagyobb heap–terület nagyságát, s várunk egy billentyű leütésére: }
      GotoXY(35,1) ; Write(BlokkSzam:6) ;
      GotoXY(35,2) ; Write(LongInt(BlokkSzam) * FoglaloMeret:6);
      GotoXY(35,3) ; Write(MemAvail:6) ;
      GotoXY(35,4) ; Write(MaxAvail:6) ;
      ReadKey ;
        { Foglalomeret méretű blokk lefoglalása – ha nincs ekkora egybefüggő terület a
          heap–ben, a SajatHeapHiba függvény meghívódik: }
      Inc(BlokkSzam) ;
      GetMem(Mutato,FoglaloMeret) ;
    { Ha nem sikerült a lefoglalás, Mutato értéke Nil lesz, mivel a heap–hiba lekezelő függvény
      visszatérési értéke 1: }
  Until Mutato = Nil ;
  WriteLn(#10,#13,'<ENTER>') ;

  ReadLn ;
End.
```

DINAMIKUS ADATSZERKEZETEK

1. feladat

Írjunk programot, mely 0 végjelig bekér számokat, egy listára felfűzve a heap–ben tárolja, majd visszaírja azokat!

```
Program Dina1 ;

  { A heap méretét a lehető legnagyobbra vesszük, hogy a dinamikus változók számára legyen
    elegendő hely: }
{$M 16384,0,655360}

Type

    { RekMutato típusú változó egy Rek típusú, heap–beli változóra fog mutatni. Mutatók
      esetében a Pascal nyelv lehetővé teszi, hogy olyan típusra hivatkozzunk, amelyet csak később
      deklarálunk. Ha ez a lehetőség nem lenne, a rekordba nem vehetnénk fel a láncoláshoz
      szükséges mutató típusú mezőt. A hivatkozásnak és a típus deklarálásának ugyanazon a
      programszinten kell szerepelnie: }
  RekMutato = ^Rek ;

    { Ilyen rekordokban tároljuk a beírt számokat: }
  Rek       = Record
                Szam       : LongInt ;

                    { A lánc következő elemére fog mutatni: }
                Kovetkezo : RekMutato ;
              End ;

Var

    { A következő három változó egy–egy futás közben létrehozott Rek típusú változóra fog
      mutatni. A mutató a dinamikus változó címét tartalmazza (4 bájt). Mutatók értékeit nem lehet
      képernyőről beolvasni, illetve oda kiírni: }
  ElsoRek,
  AktualisRek,
  UjRek       : RekMutato ;
  Szam        : LongInt ;

  { FŐPROGRAM: }
Begin

    { A lánc első elemének mutatója. Kezdetben nem mutat sehová: }
  ElsoRek := Nil ;

    { Első szám beolvasása: }
  Write('Kérek egy számot : ') ;
  ReadLn(Szam) ;
```

```
    { Amíg nem nullát írnak be: }
  While Szam <> 0 Do
    Begin
        { Új rekord lefoglalása a heap–ben. A rekord címe UjRek változóba kerül – UjRek
        változó a lefoglalt területre mutat: }
      New(UjRek) ;
        { A beírt szám áttétele az UjRek által mutatott dinamikus rekord Szam mezőjébe: }
      UjRek^.Szam := Szam ;
        { Az új rekord után nem következik további: }
      UjRek^.Kovetkezo := Nil ;

        { Az új rekord hozzáadása a listához: }
      If ElsoRek = Nil
      Then
          { Ez az első lefoglalt rekord: }
        ElsoRek := UjRek
      Else
          { ElsoRek létező címre mutat, tehát van már rekord a heap–ben, az új rekordot a
          lánc végéhez fűzzük: }
        AktualisRek^.Kovetkezo := UjRek ;

        { Ezentúl az új rekord lesz az aktuális: }
      AktualisRek := UjRek ;

        { Következő szám beolvasása: }
      Write('Kérek egy számot : ') ;
      ReadLn(Szam) ;
    End ;

    { Számok kiírása az első rekordtól: }
  AktualisRek := ElsoRek ;
    { Amíg AktualisRek heap–beli rekordra mutat: }
  While AktualisRek <> Nil Do
    Begin
        { Szám kiírása: }
      Write(AktualisRek^.Szam:10) ;
        { Lépés az aktuális utáni rekordra: }
      AktualisRek := AktualisRek^.Kovetkezo ;
    End ;

    { A program végén a heap automatikusan felszabadul: }
  ReadLn ;
End.
```

2. feladat

Írjunk programot, mely 0 végjelig bekér számokat, a heap–ben rendezetten tárol, majd visszaírja azokat! A számok kiírásával folyamatosan szabadítsuk fel azok helyét a heap–ben! Minden foglalás és felszabadítás alkalmával írjuk ki a teljes és az egybefüggő szabad heap–területet!

```
Program Dina2 ;

{$M 16384,0,655360}

Type
  RekMutato = ^Rek ;
    { Ilyen rekordokban tároljuk a beírt számokat: }
  Rek       = Record
                Szam       : LongInt ;
                  { A lánc következő elemére fog mutatni: }
                Kovetkezo : RekMutato ;
              End ;

Var
  ElsoRek,
  ElozoRek,
  AktualisRek,
  UjRek       : RekMutato ;
  Szam        : LongInt ;

  { Információ a heap összes szabad területének, valamint a maximális egybefüggő szabad
  területének a nagyságáról: }
Procedure Info ;
  Begin
    WriteLn('   MemAvail=',MemAvail:6,'   MaxAvail=',MaxAvail:6) ;
  End ;

  { FŐPROGRAM: }
Begin
  WriteLn('Számok beolvasása, folyamatos helyfoglalás') ;
  Info ;
    { A lánc első elemének mutatója. Kezdetben nem mutat sehová: }
  ElsoRek := Nil ;
    { Első szám beolvasása: }
  Write('Kérek egy számot : ') ;
  ReadLn(Szam) ;
```

```
  { Amíg nem nullát írnak be: }
While Szam <> 0 Do
  Begin
      { Új rekord lefoglalása a heap–ben: }
    New(UjRek) ;
    Info ;
      { A beírt szám áttétele az UjRek által mutatott dinamikus rekord Szam mezőjébe: }
    UjRek^.Szam := Szam ;

      { Az új rekord láncbeli helyének kikeresése. A ciklus után az új rekordot az AktualisRek
        által mutatott rekord elé kell majd beszúrni: }
    ElozoRek := Nil ;
    AktualisRek := ElsoRek ;
      { Amíg nem érünk a lánc végére és nincs meg az új rekord helye. A feltételt nem
        szabad teljesen kiértékelni, hiszen ha AktualisRek értéke Nil, akkor nem vizsgálhatjuk
        az általa mutatott rekord Szam mezőjét! }
    {$B-}
    While (AktualisRek <> Nil) And (Szam > AktualisRek^.Szam) Do
      Begin
          { ElozoRek követi AktualisRek–et: }
        ElozoRek := AktualisRek ;
          { AktualisRek a következő rekordra lép: }
        AktualisRek := AktualisRek^.Kovetkezo ;
      End ;
    {$B+}

      { Új rekord beszúrása a rendezett listába, ElozoRek és AtualisRek közé: }
    If ElozoRek = Nil
    Then
        { Az eddigi első rekord elé kell fűzni az újat: }
      ElsoRek := Ujrek
    Else
        { A lánc végéhez, vagy két rekord közé kell fűzni az újat: }
      ElozoRek^.Kovetkezo := UjRek ;
      { Az új rekord után az aktuális következzen: }
    UjRek^.Kovetkezo := AktualisRek ;

      { Következő szám beolvasása: }
    Write('Kérek egy számot : ') ;
    ReadLn(Szam) ;
  End ;

  { Számok kiírása az első rekordtól. Az előző rekordot mindig felszabadítjuk, és minden kiírás
    után információt adunk a heap szabad területeiről: }
WriteLn('Számok kiírása, folyamatos felszabadítás') ;
AktualisRek := ElsoRek ;
```

```
    { Amíg AktualisRek heap–beli rekordra mutat: }
  While AktualisRek <> Nil Do
    Begin
        { Szám kiírása: }
      Write(AktualisRek^.Szam:10) ;
      ElozoRek := AktualisRek ;
        { Lépés az aktuális utáni rekordra: }
      AktualisRek := AktualisRek^.Kovetkezo ;
        { ElozoRek által mutatott heap–terület felszabadítása. A felszabadított terület nagyságát a mutató típusa határozza meg: }
      Dispose(ElozoRek) ;
      Info ;
    End ;
  ReadLn ;
End.
```

3. feladat

Írjunk programot, mely 0 végjelig bekér számokat, azokat a heap–ben rendezetten tárolja, majd visszaírja előre és visszafelé!

```
Program Dina3 ;

{$F+,M 16384,0,655360}

Type
  RekMutato = ^Rek ;
    { Ilyen rekordokban tároljuk a beírt számokat: }
  Rek       = Record
                Szam       : LongInt ;
                  { A lánc előző elemére fog mutatni: }
                Elozo      : RekMutato ;
                  { A lánc következő elemére fog mutatni: }
                Kovetkezo  : RekMutato ;
              End ;
  Eljaras   = Procedure ;

Var
  ElsoRek,
  UtolsoRek,
  AktualisRek : RekMutato ;
  Szam        : LongInt ;
  Merre       : Eljaras ;
```

```
  { Az eljárás listára fűz egy számot úgy, hogy kitölti a már előzőleg lefoglalt, UjRek által mutatott
    rekord megfelelő mezőit. A számok most előre és visszafelé egyaránt fel vannak fűzve: }
Procedure Felfuz(UjRek: RekMutato; Szam: LongInt) ;
  Var
    ElozoRek,
    AktualisRek : RekMutato ;
  Begin
      { Szám helyének keresése a listában. A számot aztán ElozoRek és AktualisRek közé fűzzük
        fel: }
    ElozoRek := Nil ;
    AktualisRek := ElsoRek ;
    {$B-}
    While (AktualisRek <> Nil) And (Szam > AktualisRek^.Szam) Do
      Begin
        ElozoRek := AktualisRek ;
        AktualisRek := AktualisRek^.Kovetkezo ;
      End ;
    {$B+}

    If ElozoRek = Nil
    Then
      ElsoRek := UjRek
    Else
      ElozoRek^.Kovetkezo := UjRek ;
    If AktualisRek = Nil
    Then
      UtolsoRek := UjRek
    Else
      AktualisRek^.Elozo := UjRek ;

    UjRek^.Szam := Szam ;
    UjRek^.Kovetkezo := AktualisRek ;
    UjRek^.Elozo := ElozoRek ;
  End ;

  { Az eljárás a listában egyet lép előre: }
Procedure Elore ;
  Begin
    AktualisRek := AktualisRek^.Kovetkezo ;
  End ;

  { Az eljárás a listában egyet lép hátra: }
Procedure Hatra ;
  Begin
    AktualisRek := AktualisRek^.Elozo ;
  End ;
```

```
  { Az eljárás egy adott listaelemtől listázza a számokat előre vagy visszafelé, melyet a Tovabb
  eljárás típusú paraméterrel adunk meg: }
Procedure Lista(Tol: RekMutato; Tovabb: Eljaras) ;
  Begin
      { Indulás a megadott listaelemtől: }
    AktualisRek := Tol ;
    While AktualisRek <> Nil Do
      Begin
        Write(AktualisRek^.Szam:8) ;
          { Lépés a következő listaelemre. Ha Tovabb = Elore akkor lépés előre, To-
          vabb = Hatra esetén lépés visszafelé: }
        Tovabb ;
      End ;
    WriteLn ;
  End ;

  { FŐPROGRAM: }
Begin
    { A lánc első és utolsó elemének mutatója. Kezdetben nem mutatnak sehová: }
  ElsoRek := Nil ;
  UtolsoRek := Nil ;

    { Első szám beolvasása: }
  Write('Kérek egy számot : ') ;
  ReadLn(Szam) ;
  While Szam <> 0 Do
    Begin
        { A New függvényként is hívható. A New lefoglal egy Rek típusú rekordot a heap–ben,
        a visszatérő függvényérték a lefoglalt terület mutatója. Ezt a mutatót kapja meg
        paraméterként a Felfuz eljárás: }
      Felfuz(New(RekMutato),Szam) ;
        { Következő szám beolvasása: }
      Write('Kérek egy számot : ') ;
      ReadLn(Szam) ;
    End ;
  WriteLn ;

    { Végül kilistázzuk a számokat növekvő, majd csökkenő sorrendben: }
  WriteLn('Lista elôre:') ;
  Lista(ElsoRek,Elore) ;

  WriteLn('Lista visszafelé:') ;
  Lista(UtolsoRek,Hatra) ;

    { A teljes heap felszabadítása – bár ez a program végén automatikusan megtörténik:}
  Release(HeapOrg) ;
```

```
  ReadLn ;
End.
```

4. feladat

Olvassunk be egymás után katalógus– (és állomány–) neveket, amíg a '-' karaktert nem ütik le. Listázzuk ki a megadott állományok neveit ABC szerint rendezetten, dinamikus rekordok felhasználásával! Amennyiben csak az <ENTER>–t ütik le, az aktuális katalógust listázzuk! A katalógusban található katalógusneveket fényes betűkkel írjuk!

```
Program Dina4 ;

{$M 16384,0,655360}
{$V-}

Uses
  Dos,
  Crt ;

Var
  Listazando : PathStr ;

Procedure KatalogusLista(AllSpec: PathStr) ;
  Type
    NevRekMut = ^NevRek ;
    NevRek    = Record
                  Nev         : String[12] ;
                  Attributum  : Word ;
                  Kov         : NevRekMut ;
                End ;

  Var
    HeapTeto      : Pointer ;
    Bejegyzes     : SearchRec ;
    ElsoRek,
    AktualisRek,
    ElozoRek,
    UjRek         : NevRekMut ;
    Sorsz         : Byte ;

    { Az eljárás vár az <ENTER> leütésére: }
  Procedure Varj ;
    Begin
      LowVideo ;
      Write('   Tovább = <ENTER>') ;
      Repeat
      Until ReadKey = ^M ;
```

```
    ClrScr ;
  End ;

Begin
    { Megjegyezzük a heap–mutató állását, hogy az eljárás végén a heap–et idáig fel tudjuk
      szabadítani. HeapTeto a HeapPtr értékét veszi fel: }
  Mark(HeapTeto) ;
  If AllSpec = ''
  Then
      { Az aktuális katalógust listázzuk: }
    AllSpec := '*.*'
  Else
      { A megadott katalógust listázzuk: }
    Begin
        { A megadott specifikációt kiterjesztjük a teljes útvonalra: }
      AllSpec := FExpand(AllSpec) ;
        { Ha katalógusnévről van szó (nincs benne állománynév): }
      If (Pos('.',AllSpec) = 0) And (Pos('*',AllSpec) = 0) And
         (Pos('?',AllSpec) = 0) Then
        Begin
            { Katalógusnév kiegészítése '\*.*'–ra: }
          If Copy(AllSpec,Length(AllSpec),1) <> '\' Then
            AllSpec := AllSpec + '\' ;
          AllSpec := AllSpec + '*.*' ;
        End ;
    End ;

    { Első bejegyzés bekérése: }
  FindFirst(AllSpec,Archive + ReadOnly + Directory,Bejegyzes) ;
    { Választás hibakód szerint: }
  Case DosError Of
      { Nincs hiba, tehát van bejegyzés a katalógusban: }
    0   : Begin
            ElsoRek := Nil ;
              { Amíg van bejegyzés: }
            While DosError = 0 Do
              Begin
                  { Új rekord létrehozása a heap–ben: }
                New(UjRek) ;
                  { Az új dinamikus rekord Nev és Attributum mezőinek feltöltése: }
                With UjRek^ , Bejegyzes Do
                  Begin
                    Nev := Name ;
                    Attributum := Attr ;
                  End ;
```

```
            { Az új rekord leendő helyének kikeresése a láncban: }
          ElozoRek := Nil ;
          AktualisRek := ElsoRek ;
          {$B-}
          While (AktualisRek <> Nil) And
                (Bejegyzes.Name > AktualisRek^.Nev) Do
            Begin
              ElozoRek := AktualisRek ;
              AktualisRek := AktualisRek^.Kov ;
            End ;
          {$B+}
            { Rekord beszúrása a listába: }
          If ElozoRek = Nil
          Then
            ElsoRek := Ujrek
          Else
            ElozoRek^.Kov := UjRek ;
          UjRek^.Kov := AktualisRek ;
            { Következő bejegyzés bekérése: }
          FindNext(Bejegyzes) ;
        End ;
      { Listázás a lánc elejétől: }
    Sorsz := 0 ;
    AktualisRek := ElsoRek ;
      { Amíg van elem a láncban: }
    While AktualisRek <> Nil Do
      Begin
        With AktualisRek^ Do
          Begin
              { Ha katalógus, fényesen jelenik meg, egyébként nem: }
            If Attributum And Directory = Directory
            Then
              HighVideo
            Else
              LowVideo ;
              { Név és kiterjesztés kiírása, lap végén várakozás: }
            Inc(Sorsz) ;
            WriteLn(Nev,'':8 - Length(Nev)) ;
            If Sorsz = 24 Then
              Begin
                Varj ;
                Sorsz := 1 ;
              End ;
          End ;
        AktualisRek := AktualisRek^.Kov ;
      End ;
End ;
```

```
      18  : WriteLn('Nincs ilyen állomány') ;
      3   : WriteLn('A katalógus nem található') ;
      152 : WriteLn('A lemezegység üzemképtelen') ;
    End ;
    Varj ;

      { A heap felszabadítása az előzőleg megjegyzett memóriacímig. HeapPtr felveszi
      HeapTeto értékét: }
    Release(HeapTeto) ;
  End ;

  { FŐPROGRAM: }
Begin
  ClrScr ;
  Write('Listázandó katalógus: ') ;
  ReadLn(Listazando) ;
  While Listazando <> '-' Do
    Begin
      KatalogusLista(Listazando) ;
      ClrScr ;
      Write('Listázandó katalógus: ') ;
      ReadLn(Listazando) ;
    End ;
End.
```

5. feladat

Írjunk programot, mely grafikusan kirajzolja 15, heap–ben lefoglalt rekord láncolatát!

```
Program Dina5 ;

{$M 16384,0,655360}

Uses
  Crt,
  Graph ;

Type
  AdatRekMut = ^AdatRek ;
  AdatRek    = Record
                 Szam      : Word ;
                 Kovetkezo : AdatRekMut ;
                 Pozicio   : Byte ;
               End ;
```

```
Var
  ElsoRek,
  ElozoRek,
  UjRek,
  AktRek    : AdatRekMut ;
  Meghajto,
  Uzemmod,
  GrafHiba : Integer ;
  I         : Byte ;
  Szam      : Word ;

  { A függvény megadja a Poz–adik rekordot ábrázoló téglalap középpontjának X koordinátáját: }
Function XKozep(Poz: Byte) : Word ;
  Begin
    XKozep := Poz * GetMaxX Div 15 - (GetMaxX Div 30) ;
  End ;

  { Üzenetet ír a képernyő aljára, <ENTER> leütésére vár: }
Procedure Uzenet(S : String) ;
  Begin
      { Üzenetsor törlése: }
    SetViewPort(0,GetMaxY - 10,GetMaxX,GetMaxY,ClipOn) ;
    ClearViewPort ;
    SetViewPort(0,0,GetMaxX,GetMaxY,ClipOn) ;
      { Szöveg kiírása: }
    SetTextJustify(LeftText,BottomText) ;
    SetColor(White) ;
    OutTextXY(0,GetMaxY,S) ;
      { Várakozás <ENTER> leütésére: }
    Repeat
    Until ReadKey = ^M ;
  End ;

  { Kiírja az 'Első' szöveget az első rekord fölé: }
Procedure ElsoKiir(Poz : Byte) ;
  Begin
      { Felső sor törlése: }
    SetViewPort(0,0,GetMaxX,9,ClipOn) ;
    ClearViewPort ;
    SetViewPort(0,0,GetMaxX,GetMaxY,ClipOn) ;
      { 'Első' szöveg kiírása – a szöveget úgy igazítjuk, hogy a CP (grafikus kurzor) hozzá
        képest középen és fent legyen: }
    SetColor(White) ;
    SetTextJustify(CenterText,TopText) ;

    OutTextXY(XKozep(Poz),0,'Elsô') ;
  End ;
```

```
  { Kirajzol egy rekordot: }
Procedure RekRajzol(Poz : Byte ; Szam : Word ; Kiemelt : Boolean);
  Var
    S : String ;
  Begin
      { Rekord keretének törlése: }
    SetColor(GetBkColor) ;
    SetLineStyle(SolidLn,0,NormWidth) ;
    Rectangle(XKozep(Poz)-18,10,XKozep(Poz)+18,40) ;
    If Kiemelt
    Then
        { Ez az aktuális – szaggatott keret: }
      SetLineStyle(DottedLn,0,NormWidth)
    Else
        { Nem ez az aktuális – folytonos keret: }
      SetLineStyle(SolidLn,0,NormWidth) ;
      { Keret kirajzolása: }
    SetColor(GetMaxColor) ;
    Rectangle(XKozep(Poz) - 18,10,XKozep(Poz) + 18,40) ;

      { Szám kiírása: }
    Str(Szam,S) ;
    SetTextJustify(CenterText,CenterText) ;
    OutTextXY(XKozep(Poz),25,S) ;
  End ;

  { Kirajzolja a láncolást: }
Procedure LancRajzolas ;
  Var
    Poz,
    KovPoz,
    I       : Byte ;
    Akt     : AdatRekMut ;
  Begin
      { Ha üres a lánc, kilépés: }
    If ElsoRek = Nil Then
      Exit ;

      { Láncolás területének törlése: }
    SetViewPort(0,41,GetMaxX,GetMaxY,ClipOn) ;
    ClearViewPort ;
    SetViewPort(0,0,GetMaxX,GetMaxY,ClipOn) ;
    SetTextJustify(CenterText,TopText) ;
    SetLineStyle(SolidLn,0,ThickWidth) ;
```

```
    Akt := ElsoRek ;
    I := 1 ;
      { Amíg van rekord a láncban: }
    While Akt^.Kovetkezo <> Nil Do
      Begin
        Poz := Akt^.Pozicio ;
        KovPoz := Akt^.Kovetkezo^.Pozicio ;

          { Szín beállítása: }
        If GetMaxColor > 1
        Then
          SetColor(I Mod GetMaxColor)
        Else
          SetColor(1) ;

          { A három vonal kirajzolása: }
        MoveTo(XKozep(Poz) - 5,41) ;
        LineTo(XKozep(Poz) - 5,41 + I * GetMaxY Div 25) ;
        LineTo(XKozep(KovPoz) + 5,41 + I * GetMaxY Div 25) ;
        LineTo(XKozep(KovPoz) + 5,41) ;
          { Nyíl a vonal végére: }
        OutText(#30) ;
        Akt := Akt^.Kovetkezo ;
        Inc(I) ;
      End ;

      { Az utolsó rekord Nil–re mutat: }
    SetColor(GetMaxColor) ;
    With Akt^ Do
      Begin
        MoveTo(XKozep(Pozicio) - 5,41) ;
        LineTo(XKozep(Pozicio) - 5,41 + I * GetMaxY Div 25) ;
      End ;
    OutText(#31) ;
    OutTextXY(GetX,GetY + 15,'NIL') ;
  End ;

  { FŐPROGRAM: }
Begin
    { Grafikus üzemmód indítása: }
  Meghajto := Detect ;
  InitGraph(Meghajto,Uzemmod,'') ;
  GrafHiba := GraphResult ;
    { Ha hiba történt a grafikus üzemmód indításakor: }
  If GrafHiba <> grOk Then
    Begin
      WriteLn(GraphErrorMsg(GrafHiba)) ;
```

```
    ReadLn ;
    Halt ;
  End ;
  { Grafikus üzemmód átállítása a legkisebb felbontásra: }
SetGraphMode(0) ;

Randomize ;
  { A lánc első elemének mutatója. Kezdetben nem mutat sehová: }
ElsoRek := Nil ;
  { Tizenöt rekordot foglalunk le: }
For I := 1 To 15 Do
  Begin
    Uzenet('Uj rekord lefoglalása <ENTER>-re') ;
      { Új rekord lefoglalása: }
    New(UjRek) ;
      { Véletlenszám generálása: }
    Szam := Random(100) ;
    UjRek^.Szam := Szam ;
    UjRek^.Pozicio := I ;
      { Az új rekord kirajzolása: }
    RekRajzol(I,UjRek^.Szam,False) ;

    If ElsoRek <> Nil Then
        { Nem ez az első rekord: }
      Uzenet('Pozició kikeresése <ENTER>-re') ;

      { A láncbeli hely kikeresése az első rekordtól indul: }
    AktRek := ElsoRek ;
    ElozoRek := Nil ;
      { Amíg van rekord, és az abban lévő szám kisebb, mint az új: }
    {$B-}
    While (AktRek <> Nil) And (Szam > AktRek^.Szam) Do
      Begin
          { Aktuális rekord kirajzolása szaggatott vonallal: }
        RekRajzol(AktRek^.Pozicio,AktRek^.Szam,True) ;
        Uzenet('Tovább <ENTER>-re') ;
          { Aktuális rekord kirajzolása folytonos vonallal: }
        RekRajzol(AktRek^.Pozicio,AktRek^.Szam,False) ;
        ElozoRek := AktRek ;
        AktRek := AktRek^.Kovetkezo ;
      End ;
    {$B+}

      { Az új rekord láncolása: }
    UjRek^.Kovetkezo := AktRek ;
```

```
      If ElozoRek = Nil
      Then
           { Az eddigi első elé kell láncolni: }
        Begin
          ElsoRek := Ujrek ;
          Uzenet('Az új rekord lesz az elsô') ;
          ElsoKiir(ElsoRek^.Pozicio) ;
        End
      Else
           { Két rekord közé, vagy a lánc végére kell illeszteni: }
        Begin
          RekRajzol(ElozoRek^.Pozicio,ElozoRek^.Szam,True) ;
          Uzenet('E mögé kell beilleszteni az új rekordot - ' +
                 'tovább <ENTER>') ;
          RekRajzol(ElozoRek^.Pozicio,ElozoRek^.Szam,False) ;
          ElozoRek^.Kovetkezo := UjRek ;
        End ;
        { Összeláncolás kirajzolása színes vonalakal: }
      LancRajzolas ;
    End ;

  Uzenet('Kilépés <ENTER>-re') ;
    { Grafikus üzemmód lezárása: }
  CloseGraph ;
End.
```

6. feladat

Írjunk programot, mely alapvető adatkezelési funkciókat valósít meg oda–vissza láncolt dinamikus rekordokkal!

```
Program Dina6 ;

{$M 16384,0,655360}

Uses
  Crt ;

Const
  NevHossz = 25 ;
  CimHossz = 40 ;

Type
  AdatRekMut  = ^AdatRek ;
```

```
  AdatRek      = Record
                   Nev         : String[NevHossz] ;
                   Cim         : String[CimHossz] ;
                   Tel         : LongInt ;
                     { Előző illetve következő rekordok mutatói: }
                   Elozo,
                   Kovetkezo : AdatRekMut ;
                 End ;

Const
    { A lánc első rekordjára mutat: }
  ElsoAdat : AdatRekMut = Nil ;

Var
  AktualisAdat,
  ElozoAdat      : AdatRekMut ;
  Nev            : String[NevHossz] ;
  C              : Char ;
  Puffer         : Array[1..80] Of Byte ;

  { Név kikeresése a láncból, a mutatók beállítása: }
Function VanMarIlyenNev : Boolean ;
  Begin
      { Keresés a lánc elejétől: }
    ElozoAdat := ElsoAdat ;
    AktualisAdat := ElsoAdat ;
    {$B-}
      { Amíg nem találtuk meg a nevet, vagy annak helyét: }
    While (AktualisAdat <> Nil) And (Nev > AktualisAdat^.Nev) Do
      Begin
        ElozoAdat := AktualisAdat ;
        AktualisAdat := AktualisAdat^.Kovetkezo ;
      End ;

      { Függvény értékének beállítása: }
    VanMarIlyenNev := (AktualisAdat <> Nil) And
                      (AktualisAdat^.Nev = Nev) ;
    {$B+}
  End ;

  { Üzenet kiírása / törlése: }
Procedure Uzenet(Szoveg : String ; Hang : Boolean) ;
  Begin
    GotoXY(1,25) ;
    ClrEol ;
    Write(Szoveg) ;
```

```
      { Ha kell hangjelzés: }
    If Hang Then
      Begin
        Sound(1000) ;
        Delay(50) ;
        NoSound ;
        Delay(1000) ;
        GotoXY(1,25) ;
        ClrEol ;
      End ;
  End ;

  { Név bekérése: }
Procedure NevBekeres ;
  Begin
    GotoXY(20,10) ;
    Write('Név : ') ;
      { A bemeneti puffer méretének beállítása Nev változó logikai méretére: }
    SetTextBuf(Input,Puffer,NevHossz + 2) ;
    ReadLn(Nev) ;
  End ;

  { Többi adat bekérése: }
Procedure TobbiAdatBekeres(Mutato : AdatRekMut) ;
  Var
    T  : String[10] ;
    Ok : Integer ;
  Begin
    GotoXY(20,12) ;
    Write('Cim : ') ;
      { A bemeneti puffer méretének beállítása Cim változó logikai méretére: }
    SetTextBuf(Input,Puffer,CimHossz + 2) ;
    ReadLn(Mutato^.Cim) ;
      { Bemeneti puffer méretének beállítása úgy, hogy 10 számjegyet lehessen beírni: }
    SetTextBuf(Input,Puffer,12) ;
    Repeat
        GotoXY(20,14) ;
        ClrEol ;
        Write('Tel : ') ;
        ReadLn(T) ;
        Val(T,Mutato^.Tel,Ok) ;
      { Kilépés, ha a konverzió sikeres volt és nem negatív a szám: }
    Until (Ok = 0) And (Mutato^.Tel > 0) ;
  End ;
```

```
  { Új adat felvitele a láncba: }
Procedure AdatFelvitel ;
  Var
    UjAdat : AdatRekMut ;
  Begin
      { Név bekérése: }
    NevBekeres ;
    If VanMarIlyenNev
    Then
        { Van már ilyen név – hibaüzenet, mert a duplikáció tiltott: }
      Uzenet('Van már ilyen név, nem vihetô fel újra',True)
    Else
        { Nincs még ilyen név, beillesztés a láncba. A mutatókat a VanMarIlyenNev függvény
        beállította: }
      Begin
          { Új adatrekord lefoglalása: }
        New(UjAdat) ;
        UjAdat^.Nev := Nev ;
          { A rekord többi mezőjének feltöltése: }
        TobbiAdatBekeres(UjAdat) ;

        If AktualisAdat = ElsoAdat
        Then
            { Az első rekord elé kell beilleszteni az újat: }
          Begin
            If ElsoAdat <> Nil Then
              ElsoAdat^.Elozo := UjAdat ;
            ElsoAdat := UjAdat ;
            UjAdat^.Elozo := Nil ;
          End
        Else
            { Az AktualisAdat által mutatott rekord elé kell beilleszteni az újat: }
          Begin
            If AktualisAdat <> Nil Then
              AktualisAdat^.Elozo := UjAdat ;
            ElozoAdat^.Kovetkezo := Ujadat ;
            UjAdat^.Elozo := ElozoAdat ;
          End ;

        UjAdat^.Kovetkezo := AktualisAdat ;
      End ;
  End ;
```

```
  { Adat törlése a láncból: }
Procedure AdatTorles ;
  Begin
      { Név bekérése: }
    NevBekeres ;
    If VanMarIlyenNev
    Then
        { Van ilyen név, törölhető: }
      Begin
        If AktualisAdat = ElsoAdat
        Then
            { Az első rekordot kell kivenni a láncból: }
          Begin
            ElsoAdat := AktualisAdat^.Kovetkezo ;
            ElsoAdat^.Elozo := Nil ;
          End
        Else
            { Nem az első rekordot kell kivenni a láncból: }
          Begin
            ElozoAdat^.Kovetkezo := AktualisAdat^.Kovetkezo ;
              { Ha nem az utolsó rekordot kell törölni: }
            If AktualisAdat^.Kovetkezo <> Nil Then
              AktualisAdat^.Kovetkezo^.Elozo := ElozoAdat ;
          End ;
            { Adatrekord megszüntetése (helyének felszabadítása): }
        Dispose(AktualisAdat) ;
      End
    Else
        { Nincs ilyen név a láncban: }
      Uzenet('Nincs ilyen név, nem törölhetô',True) ;
  End ;

  { Adat módosítása: }
Procedure AdatModositas ;
  Begin
      { Név bekérése: }
    NevBekeres ;
    If VanMarIlyenNev
    Then
        { Van ilyen név, módosítás a többi adat újbóli bekérésével: }
      TobbiAdatBekeres(AktualisAdat)
    Else
        { Nincs ilyen név a láncban: }
      Uzenet('Nincs ilyen név, nem módosítható',True) ;
  End ;
```

```
  { Listázás: }
Procedure Listazas ;
  Var
    C1,
    C2 : Char ;

    { Adatok megjelenítése: }
  Procedure AdatKiiras ;
    Begin
      With AktualisAdat^ Do
        Begin
          GotoXY(20,10) ;
          ClrEol ;
          Write('Név : ',Nev) ;
          GotoXY(20,12) ;
          ClrEol ;
          Write('Cím : ',Cim) ;
          GotoXY(20,14) ;
          ClrEol ;
          Write('Tel : ',Tel) ;
        End ;
    End ;

  Begin
    AktualisAdat := ElsoAdat ;
    AdatKiiras ;
    Repeat
        Uzenet('Léptetés elôre/hátra a vízszintes' +
               ' kurzorvezérlôkkel / ESC - Kilépés',False) ;
          { Első kód bekérése: }
        C1 := ReadKey ;
          { Ha vezérlőkarakter: }
        If C1 = #0 Then
          Begin
              { Második kód bekérése: }
            C2 := ReadKey ;
            Case C2 Of
                { Balra nyíl: }
              #75 : If AktualisAdat^.Elozo = Nil
                    Then
                        { Nincs előző rekord: }
                      Uzenet('Nincs elôzô adat',True)
                    Else
                        { Léptetés az előző rekordra, kiírás: }
                      Begin
                        AktualisAdat := AktualisAdat^.Elozo ;
                        AdatKiiras ;
                      End ;
```

```
                    { Jobbra nyíl: }
                #77 : If AktualisAdat^.Kovetkezo = Nil
                      Then
                          { Nincs következő rekord: }
                        Uzenet('Nincs következô adat',True)
                      Else
                          { Léptetés a következő rekordra, kiírás: }
                        Begin
                          AktualisAdat :=
                            AktualisAdat^.Kovetkezo ;
                          AdatKiiras ;
                        End ;
            End ;
          End ;
        { Kilépés <ESC> –re: }
    Until C1 = ^[ ;
  End ;

  { FŐPROGRAM: }
Begin
    { Főmenü: }
  Repeat
      ClrScr ;
      Uzenet('F - Felvitel / T - Törlés / M - Módosítás / ' +
             'L - Listázás / ESC - Kilépés',False) ;
        { Karakter bekérése: }
      C := UpCase(ReadKey) ;
      Case C Of
        'F'           : AdatFelvitel ;
        'T','M','L'   : If ElsoAdat = Nil
                        Then
                          Uzenet('Nincs bevitt adat',True)
                        Else
                          Case C Of
                            'T' : AdatTorles ;
                            'M' : AdatModositas ;
                            'L' : Listazas ;
                          End ;
      End ;
    { Kilépés <ESC>–re: }
  Until C = ^[ ;
  ClrScr ;
End.
```

7. feladat

Írjunk programot, mely alapvető karbantartó funkciókat lát el bináris fa felhasználásával! Két egyforma kulcs felvitelét ne engedjük meg!

```
Program Dina7 ;

Uses
  Crt ;

Type
  KulcsTip = String[3] ;
```

{ A bináris fa lényege, hogy egy gyökérből kiindulva minden elem alatt két ág van, egy baloldali és egy jobboldali ág (fejreállított fa). A baloldalon lévő elem mindig kisebb mint a szülő, míg a jobboldali nagyobb, vagy egyenlő, mint a szülő. Így egy n szintű fa elemeinek a száma maximum 2^n, és egy elemet n lépésben biztos megtalálunk.}

```
  FaMutato = ^FaRekord ;
  FaRekord = Record
               Bal,
               Jobb  : FaMutato ;
               Kulcs : KulcsTip ;
               Adat  : String ;
             End ;

Var
  Gyoker  : FaMutato ;
  Kulcs   : KulcsTip ;
  MenuKar : Char ;
```

{ Kulcs beolvasása: }

```
Procedure Kulcs_Beolvas ;
  Begin
    ClrScr ;
    Write('Kulcs = ') ;
    ReadLn(Kulcs) ;
  End ;
```

{ Hibaüzenet kiírása: }

```
Procedure HibaUzenet(HibaSt: String) ;
  Begin
    GotoXY(25,25) ;
    Write(HibaSt) ;
  End ;
```

{ Kulcs keresése a bináris fában. A gyökértől indulunk, és addig megyünk, amíg nincs meg az elem, és még nem értünk a végére. A szülőt az esetleges beszúrás illetve törlés esetére jegyezzük meg: }

```
Function Van(Kulcs: KulcsTip; Var Aktualis, Szulo: FaMutato) :
                                                          Boolean ;
  Var
    OK : Boolean ;
  Begin
    OK := False ;
    Aktualis := Gyoker ;
    Szulo := Nil ;
    While Not OK And (Aktualis <> Nil) Do
      Begin
        If Kulcs = Aktualis^.Kulcs
        Then
          OK := True
        Else
          Begin
            Szulo := Aktualis ;
            If Kulcs < Aktualis^.Kulcs
            Then
              Aktualis := Aktualis^.Bal
            Else
              Aktualis := Aktualis^.Jobb ;
          End ;
      End ;
    Van := OK ;
  End ;
```

{ Új elem beszúrása a bináris fába, duplikációt nem engedünk meg. Akármelyik szinten is van az elem, az őt rendezettség szerint közvetlenül megelőző az ő bal oldali ágának egyik legalsó eleme, míg az őt közvetlenül követő az ő jobb oldali ágának egyik legalsó eleme (ott viszont a bal oldali legalsó). Ezért egy új elem helyének megállapításához mindenképpen le kell mennünk a fa aljáig. Új elem mindig a fa aljára kerül (levél lesz): }

```
Procedure Felvitel ;
  Var
    Aktualis,
    Uj,
    Szulo     : FaMutato ;
  Begin
    Kulcs_Beolvas ;
    If Van(Kulcs,Aktualis,Szulo)
    Then
      HibaUzenet('Már van ilyen kulcs!')
    Else
      Begin
        New(Uj) ;
        Uj^.Kulcs := Kulcs ;
        Uj^.Bal := Nil ;
        Uj^.Jobb := Nil ;
        GotoXY(1,2) ; Write('Egyéb = ') ;
```

```
        ReadLn(Uj^.Adat) ;

            { Ha a keresett elem eddig még nem létezett, akkor Aktualis mindig egy levélre
              mutat. Ha ez a gyökér, akkor a fa még üres. Az új elemet a levél bal vagy jobb
              ágára helyezzük, így az új elem lesz a levél: }
        If Aktualis = Gyoker
        Then
          Gyoker := Uj
        Else
          If Kulcs < Szulo^.Kulcs
          Then
            Szulo^.Bal := Uj
          Else
            Szulo^.Jobb := Uj ;
      End ;
  End ;

  { Elem törlése a bináris fából. Ha nincs ilyen kulcs, hibaüzenettel leállás: }
Procedure Torles ;
  Var
    Aktualis,
    Hely,
    Legnagyobb,
    Szulo     : FaMutato ;
  Begin
    Kulcs_Beolvas ;
    If Not Van(Kulcs,Aktualis,Szulo)
    Then
      HibaUzenet('Nincs ilyen kulcs!')
    Else
      Begin
            { Ha a törlendő elemnek van baloldali ága, akkor a baloldali ág lép a törlendő
              helyére, egyébként a jobboldali. Ha a baloldali ág lép eggyel feljebb, akkor ez az
              ág örökli a jobboldali ágat, azaz a legnagyobb eleméhez – a jobboldali legalsó
              leveléhez – fog kapcsolódni az egész jobboldali ág.}
        If Aktualis^.Bal = Nil
        Then
          Hely := Aktualis^.Jobb
        Else
          Begin
            Hely := Aktualis^.Bal ;
            Legnagyobb := Hely ;
            While Legnagyobb^.Jobb <> Nil Do
              Legnagyobb := Legnagyobb^.Jobb ;
            Legnagyobb^.Jobb := Aktualis^.Jobb ;
          End ;
```

```
            { Ha van szülő, akkor azt összekapcsoljuk a törlendő elem helyébe lépett elemmel,
              egyébként az lesz az új gyökér: }
          If Gyoker = Aktualis
          Then
            Gyoker := Hely
          Else
            If Szulo^.Bal = Aktualis
            Then
              Szulo^.Bal := Hely
            Else
              Szulo^.Jobb := Hely ;
          Dispose(Aktualis) ;
        End ;
    End ;

    { A bináris fa kilistázása rekurzív eljárással: }
Procedure FaLista(Gyoker: FaMutato) ;
    Begin
      If Gyoker <> Nil
      Then
        Begin
          FaLista(Gyoker^.Bal) ;
          With Gyoker^ Do
            WriteLn(Kulcs,'':5-Length(Kulcs),Adat) ;
          FaLista(Gyoker^.Jobb) ;
        End ;
    End ;

Begin
      { Kezdetben a fa üres: }
    Gyoker := Nil ;

    ClrScr ;
    GotoXY(30,10) ; Write ('Felvitel_____F') ;
    GotoXY(30,12) ; Write ('Törlés_______T') ;
    GotoXY(30,14) ; Write ('Vége_________V') ;
    Repeat
        Window(1,1,80,25) ;
        GotoXY(44,14) ;
        Repeat
          MenuKar := Upcase(ReadKey) ;
        Until MenuKar In ['F','T','V'] ;
        Window(30,20,80,25) ;
        Case MenuKar Of
          'F' : Felvitel ;
          'T' : Torles ;
        End ;
        Window(2,2,25,24) ;
        ClrScr ;
```

```
        { Minden tranzakció után kilistázzuk a fa tartalmát a képernyő bal felső sarkában: }
      FaLista(Gyoker) ;
  Until MenuKar = 'V' ;
  Window(1,1,80,25) ;
  ClrScr ;
End.
```

8. feladat

Írjunk programot, mely grafikusan ábrázolja egy a heap–ben tárolt bináris fa felépítését! A felvitelnél duplikációt is megengedünk!

```
Program Dina8 ;

{$X+}

Uses
  Crt,
  Graph ;

Const
  MaxKulcs = 99 ;

Type
  FaMutato = ^FaRekord ;
  FaRekord = Record
               Bal,
               Jobb  : FaMutato ;
               Kulcs : Integer ;
             End ;

Var
  Meghajto,
  Uzemmod,
  GrafHiba  : Integer ;
  XLepes,
  YLepes,
  TeglaSzel,
  TeglaMag  : Integer ;
  Gyoker    : FaMutato ;
  Kar       : Char ;
  St        : String ;

  { Kezdeti értékek számolása: }
Procedure Init ;
  Var
   TeglaSzin : Integer ;
```

```
  Begin
      { Meghatározzuk a kirajzolandó téglalap elemek szélességét, hosszúságát, valamint a
        függőleges lépésközt (a vízszintes lépésköz a fa bejárásánál mindig feleződik): }
    TeglaSzel := TextWidth('00') + 2 ;
    TeglaMag := TextHeight('0') * 2 ;
    YLepes := Trunc(TeglaMag * 1.8) ;
      { Az elemeket ilyen színnel fogjuk rajzolni: }
    Case Meghajto Of
      HercMono : Teglaszin := 1 ;
      Cga      : TeglaSzin := 2 ;
      Else       TeglaSzin := EgaRed ;
    End ;
    SetFillStyle(SolidFill,TeglaSzin) ;
  End ;

  { A grafikus képernyőn lévő szöveg törlése adott pozíciótól, adott hosszúságban: }
Procedure Torol(X,Y,Hossz: Integer) ;
  Var
    Szin,
    I    : Byte ;
    St   : String[80] ;
  Begin
    Szin := GetColor ;
    SetColor(GetBkColor) ;
    St := '' ;
    For I := 1 To Hossz Do
      St := St + #219 ;
    OutTextXY(X,Y,St) ;
    SetColor(Szin) ;
  End ;

  { Az eljárás kirajzol egy adott középpontú tömör téglalapot, s beleírja a paraméterként megadott
    számot: }
Procedure TeglaRajzol(X,Y: Integer; Szam: Integer) ;
  Var
    St      : String[5] ;
  Begin
    Bar(X - TeglaSzel Div 2 + 1, Y - TeglaMag Div 2 + 1,
        X + TeglaSzel Div 2 - 1, Y + TeglaMag Div 2 - 1) ;
    SetTextJustify(CenterText,CenterText) ;
    SetColor(GetBkColor) ;
    Str(Szam,St) ;
    OutTextXY(X,Y,St) ;
  End ;
```

```
  { A téglalap keretének megrajzolása: }
Procedure KeretRajzol(X,Y: Integer) ;
  Begin
    Rectangle(X - TeglaSzel Div 2, Y - TeglaMag Div 2,
              X + TeglaSzel Div 2, Y + TeglaMag Div 2) ;
  End ;

  { Új elem beszúrása a bináris fába, duplikációt megengedünk: }
Procedure Beszur ;
  Var
    Aktualis,
    Uj,
    Szulo     : FaMutato ;
    UjKulcs   : Integer ;
    KulcsSt   : String[5] ;
    Szint,
    XPoz,
    YPoz      : Integer ;
  Begin
    UjKulcs := Random(MaxKulcs) + 1 ;
    Str(UjKulcs,KulcsSt) ;

      { Lefelé megyünk a fán egészen a végéig. Mindig aszerint ágazunk jobbra vagy balra,
        hogy a kulcs kisebb, vagy nagyobb az elágazásban lévő elemnél. Most nem számít, hogy
        találtunk–e ugyanilyen kulcsot, hiszen a duplikáció megengedett. Itt végül a szülő egy
        levél lesz, melyre majd rátesszük az új levelet: }
    Aktualis := Gyoker ;
    Szint := 0 ;
      { Meghatározzuk az aktuális szinthez tartozó vízszintes lépést: }
    XLepes := GetMaxX Div 2 ;
      { Az aktuális elem képernyőpozíciói: }
    XPoz := GetMaxX Div 2 ;
    YPoz := TeglaMag + 10 ;
    While Aktualis <> Nil Do
      Begin
        Inc(Szint) ;
        XLepes := XLepes Shr 1 ;

          { A keresésben aktuális elem megjelölése: }
        SetColor(GetMaxColor) ;
        KeretRajzol(XPoz,YPoz) ;
        SetTextJustify(CenterText,TopText) ;
        OutTextXY(XPoz,YPoz + TeglaMag Div 2 + 2,KulcsSt + '?');

          { Ha mehet tovább a keresés, a megjelölés eltüntetése: }
        SetTextJustify(CenterText,BottomText) ;
        Torol(GetMaxX Div 2,GetMaxY,36) ;
        OutTextXY(GetMaxX Div 2,GetMaxY,
                  'Tovább keresés - <ENTER>');
```

```
      Repeat
      Until ReadKey = #13 ;
      SetColor(GetBkColor) ;
      KeretRajzol(XPoz,YPoz) ;
      SetTextJustify(CenterText,TopText) ;
      Torol(XPoz,YPoz + TeglaMag Div 2 + 2,
            Length(KulcsSt + '?')) ;
        { Ha a kulcs kisebb az aktuálisnál, balra megyünk, egyébként jobbra: }
      Szulo := Aktualis ;
      If UjKulcs < Aktualis^.Kulcs
      Then
        Begin
          Aktualis := Aktualis^.Bal ;
          Dec(XPoz,XLepes) ;
        End
      Else
        Begin
          Aktualis := Aktualis^.Jobb ;
          Inc(XPoz,XLepes) ;
        End ;
      Inc(YPoz,YLepes) ;
    End ;

    { Ha az új elem nem fér rá a képernyőre, kilépés: }
  If (YPoz >= GetMaxY - TeglaMag) Or (XLepes < TeglaSzel) Then
    Begin
      SetTextJustify(CenterText,TopText) ;
      Torol(GetMaxX Div 2,1,25) ;
      SetColor(GetMaxColor) ;
      OutTextXY(GetMaxX Div 2,1,
                KulcsSt + ' nem fért rá a fára!') ;
      Exit ;
    End ;

    { Az új elemnek helyet foglalunk, és rákapcsoljuk a bináris fára. Az új elem biztos levél
    lesz: }
  New(Uj) ;
  Uj^.Kulcs := UjKulcs ;
  Uj^.Bal := Nil ;
  Uj^.Jobb := Nil ;

  TeglaRajzol(XPoz,YPoz,UjKulcs) ;
  SetColor(GetMaxColor) ;
  If Aktualis = Gyoker
  Then
    Gyoker := Uj
  Else
    Begin
      MoveTo(XPoz,YPoz - TeglaMag Div 2) ;
      LineRel(0,- (YLepes - TeglaMag Div 2)) ;
```

```
        If UjKulcs < Szulo^.Kulcs
        Then
          Begin
            Szulo^.Bal := Uj ;
            LineRel((XLepes - TeglaSzel Div 2),0) ;
          End
        Else
          Begin
            Szulo^.Jobb := Uj ;
            LineRel(-(XLepes - TeglaSzel Div 2),0) ;
          End ;
      End ;
  End ;

  { A bináris fa elemei számának meghatározása rekurzióval: }
Function FaMeret(Gyoker: FaMutato) : LongInt ;
  Begin
    If Gyoker = Nil
    Then
      FaMeret := 0
    Else
      FaMeret := 1 + FaMeret(Gyoker^.Bal) + FaMeret(Gyoker^.Jobb);
  End ;

  { FŐPROGRAM: }
Begin
    { Grafikus üzemmód bekapcsolása a legkisebb felbontással: }
  Meghajto := Detect ;
  InitGraph(Meghajto,Uzemmod,'') ;
  GrafHiba := GraphResult ;
  If GrafHiba <> grOk Then
    Begin
      WriteLn(GraphErrorMsg(GrafHiba)) ;
      ReadLn ;
      Halt ;
    End ;
  SetGraphMode(0) ;
  Init ;

    { Kezdetben a fa üres, a beszúrandó elemeket véletlenszámgenerátor segítségével állítjuk
    elő: }
  Gyoker := Nil ;
  Randomize ;
  Repeat
      { Fa méretének kiírása: }
    Str(FaMeret(Gyoker),St) ;
    SetTextJustify(LeftText,TopText) ;
    Torol(1,1,Length(St)) ;
```

```
      SetColor(GetMaxColor) ;
      OutTextXY(1,1,St) ;

      SetTextJustify(CenterText,BottomText) ;
      Torol(GetMaxX Div 2,GetMaxY,36) ;
      OutTextXY(GetMaxX Div 2,GetMaxY,
                'Uj elem - <ENTER>     Kilépés - <ESC>') ;
      Repeat
        Kar := ReadKey ;
      Until Kar In [#13,#27] ;
      If Kar <> #27 Then
        Beszur ;
  Until Kar = #27 ;
  CloseGraph ;
End.
```

TÍPUSNÉLKÜLI ÁLLOMÁNYOK

1. feladat

Írjunk programot, mely egy tömbbe gyűjt értékeket, s a gyűjtött értékeket a program végén lemezre menti. A program következő indításakor a gyűjtés az előzőleg mentett tömbbel induljon!

```
Program Tipnelk1 ;

Uses
  Crt ;

Const
  Max = 10 ;

Var
  Tomb         : Array[1..Max] Of LongInt ;
    { Típus nélküli állomány. Az elemek (rekordok) méretét így nem egy típus, hanem a nyitáskor
    bájtokban megadott érték határozza meg (annak hiányában a rekordméret 128 lesz). A File
    típus helyfoglalása 128 bájt. }
  MentettTomb : File ;
  Index       : Byte ;
  Ertek       : LongInt ;

  { A tömb elemeinek képernyőre írása: }
Procedure TombKiir ;
  Var
    I : Byte ;
  Begin
    GotoXY(1,10) ;
    For I := 1 To Max Do
      Write(Tomb[I]:8) ;
  End ;

  { FŐPROGRAM: }
Begin
    { A MentettTomb logikai állományhoz a lemezen a 'Tomb.Dat' állományt rendeljük: }
  Assign(MentettTomb,'Tomb.Dat') ;
    { Az állomány megnyitása, a rekordhossz 1 bájt: }
  {$I-}
  Reset(MentettTomb,1) ;
  {$I+}
```

```
  If IOResult = 0
  Then
      { Van mentett állomány: }
    Begin
        { Ha az elmentett tömb mérete nem egyezik a programban szereplő tömb méretével,
          a program leáll: }
      If FileSize(MentettTomb) <> SizeOf(Tomb) Then
      Begin
        WriteLn('A tömb mérete nem egyezik a mentettével!') ;
        ReadLn ;
        Halt ;
      End ;
      { Tomb kezdőcímétől kezdve annyi rekord (itt 1 rekord 1 bájt) beolvasása az
        állományból, amekkora a tömb mérete: }
    BlockRead(MentettTomb,Tomb,SizeOf(Tomb)) ;
    Close(MentettTomb) ;
  End
Else
    { Nincs mentett állomány, a tömb összes eleme legyen 0: }
  FillChar(Tomb,SizeOf(Tomb),0) ;

  { A tömb elemeinek gyűjtése, amíg indexként nullát nem ütnek. Közben a tömböt a képernyőn
    aktualizáljuk: }
ClrScr ;
TombKiir ;
GotoXY(30,21) ; Write('Index: ') ;
GotoXY(30,22) ; Write('Érték: ') ;
GotoXY(38,21) ;
ReadLn(Index) ;
While Index <> 0 Do
  Begin
    GotoXY(38,22) ;
    ReadLn(Ertek) ;
    Inc(Tomb[Index],Ertek) ;
    TombKiir ;
    GotoXY(38,22) ; ClrEol ;
    GotoXY(38,21) ; ClrEol ;
    ReadLn(Index) ;
  End ;

  { Tomb elmentése lemezre. Az állományt a tömb méretének megfelelő rekordhosszal nyitjuk
    meg. Az esetlegesen már létező mentett tömb felülíródik: }
Rewrite(MentettTomb,SizeOf(Tomb)) ;
  { Most csak egy rekordot kell felírnunk az állományba Tomb kezdőcímétől kezdve, hiszen
    más a választott rekordhossz: }
BlockWrite(MentettTomb,Tomb,1) ;
Close(MentettTomb) ;
End.
```

2. feladat

Írjunk programot, mely egy adott állományt átmásol egy ugyanolyan nevű, 'BAK' kiterjesztésű állományba!

```
Program Tipnelk2 ;

Var
  Forras,
  Cel             : File ;
  ForrasNeve,
  CelNeve         : String ;
  Puffer          : Pointer ;
  PufferMeret,
  BeolvasottRek,
  KiirtRek        : Word ;

  { Végzetes hiba esetén hibaszöveg kiírása, program leállítása: }
Procedure Hiba(Szoveg: String) ;
  Begin
    WriteLn(Szoveg) ;
    ReadLn ;
    Halt ;
  End ;

  { FŐPROGRAM: }
Begin
    { Forrásállomány nevének beolvasása, célállomány nevének meghatározása: }
  Write('A másolandó állomány neve: ') ;
  ReadLn(ForrasNeve) ;
  CelNeve := ForrasNeve ;
  Delete(CelNeve,Pos('.',CelNeve),255) ;
  CelNeve := CelNeve + '.BAK' ;

      { Állományok nyitása: }
  Assign(Forras,ForrasNeve) ;
  Assign(Cel,CelNeve) ;
  {$I-}
  Reset(Forras,1) ;
  {$I+}
  If IOResult <> 0 Then
    Hiba('Nincs ilyen állomány!') ;
  Rewrite(Cel,1) ;

    { Az állomány másolásához szükséges átmeneti puffer a heap–ben lesz, mégpedig a lehető
      legnagyobb méretű. Ha nincs egyáltalán heap, a programot leállítjuk: }
  PufferMeret := $FFFF ;
```

```
  If MaxAvail < PufferMeret Then
    PufferMeret := MaxAvail ;
  If PufferMeret = 0 Then
    Hiba('Nincs elég memória!') ;
  GetMem(Puffer,PufferMeret) ;

    { Az állomány másolása. Egyszerre maximum egy puffernyi adatot olvasunk be, melyet rögtön
    felírunk a célállományba. Kilépünk a ciklusból, ha nem tudtuk teleolvasni a puffert (nincs
    több adat az állományban), illetve nem tudtuk kiírni az összes beolvasott adatot lemezre
    (betelt a lemez, lemezhiba stb.): }
  Repeat
      BlockRead(Forras,Puffer^,PufferMeret,BeolvasottRek) ;
      BlockWrite(Cel,Puffer^,BeolvasottRek,KiirtRek) ;
  Until (BeolvasottRek < PufferMeret) Or
        (BeolvasottRek <> KiirtRek) ;
  If BeolvasottRek <> KiirtRek Then
    Hiba('Lemez írási hiba!') ;

    { Állományok zárása: }
  Close(Forras) ;
  Close(Cel) ;
End.
```

3. feladat

Írjunk programot, mely egy paraméterben megadott állomány paraméterben megadott mennyiségű bájtját kiírja hexadecimálisan, és alatta karakteresen (a nem megjeleníthető karakterek helyén pont álljon)!

```
Program Tipnelk3 ;

Uses
  Crt ;

Const
  MaxBajtSzam = 20000 ;

Type
  St2 = String[2] ;

Var
  Ismeretlen           : File ;
  Sorszamlalo          : Byte ;
  Kezd,
  SorMutato,
  BajtSzam,
  Beolvasott_Bajtszam : Word ;
```

```
  Kod                 : Integer ;
  Puffer              : Array[1..MaxBajtSzam] Of Byte ;
  Ch                  : Char ;

Function Hex(B: Byte) : St2 ;
  Const
    HexJegy : Array [0..15] Of Char = '0123456789ABCDEF' ;
  Begin
    Hex := HexJegy[B Div 16] + HexJegy[B Mod 16] ;
  End ;

Procedure Utmutatas ;
  Begin
    WriteLn('Használat: TIPNELK3 <állományspec> <Bájtszám (1-',
            MaxBajtSzam,')>') ;
    ReadLn ;
    Halt ;
  End ;

  { FŐPROGRAM: }
Begin
    { Ha a külső paraméterek száma nem kettő, vagy a második nem szám, információ kiírása
      a program használatáról: }
  If ParamCount <> 2 Then
    Utmutatas ;
  Val(ParamStr(2),BajtSzam,Kod) ;
  If Kod <> 0 Then
    Utmutatas ;
  If BajtSzam > MaxBajtSzam Then
    BajtSzam := MaxBajtSzam ;
    { Hozzárendeljük Ismeretlen állományváltozóhoz a paraméterben megadott fizikai nevet: }
  Assign(Ismeretlen,ParamStr(1)) ;
    { Az állományt megnyitjuk, a rekordhossz 1 bájt: }
  {$I-}
  Reset(Ismeretlen,1) ;
  {$I+}
  If IOResult <> 0 Then
    Utmutatas ;

    { Az állomány létezik, beolvassuk Pufferbe a kívánt számú bájtot, illetve ha rövidebb az
      állomány, akkor annyit, amennyi van: }
  BlockRead(Ismeretlen,Puffer,BajtSzam,Beolvasott_BajtSzam) ;

    { Cím kiírása a képernyőre: }
  ClrScr ;
  WriteLn('A ''',ParamStr(1),''' elsô ',Beolvasott_BajtSzam,
          ' bájtja:') ;
  WriteLn ;
```

```
    { Számoljuk a kiírt sorok számát, hogy közben megállíthassuk: }
  Sorszamlalo := 0 ;

    { Puffer–ből Kezd–től írunk ki 26 hexadecimális számot, majd ugyaninnen 26 karaktert
    (ennyi fér el egy sorban): }
  Kezd := 1 ;

    { Amíg nem értünk a puffer végére: }
  While Kezd <= Beolvasott_Bajtszam Do
    Begin

        { Egy sor kiírása hexadecimálisan: }
      SorMutato := 0 ;

        { Amíg nem értünk a sor vagy a puffer végére: }
      While (SorMutato < 26) And
            (Kezd + SorMutato <= Beolvasott_BajtSzam) Do
        Begin
          Write(Hex(Puffer[Kezd + SorMutato]):3) ;
          Inc(SorMutato) ;
        End ;
      WriteLn ;

        { Egy sor kiírása karakteresen: }
      SorMutato := 0 ;

        { Amíg nem értünk a sor vagy a puffer végére: }
      While (SorMutato < 26) And
            (Kezd + SorMutato <= Beolvasott_BajtSzam) Do
        Begin
          Ch := Chr(Puffer[Kezd + SorMutato]) ;
          If Not(Ch In [Chr(0)..Chr(31),Chr(127)])
          Then
            Write(Ch:3)
          Else
            Write('.':3) ;
          Inc(SorMutato) ;
        End ;
      WriteLn ;
      Inc(Sorszamlalo) ;

        { A képernyő aljára értünk (7 * 3 sort írtunk ki), várunk az <ENTER> leütésére: }
      If Sorszamlalo = 7 Then
        Begin
          ReadLn ;
          ClrScr ;
          Sorszamlalo := 0 ;
        End ;

        { Pufferben 26–tal tovább lépünk, a következő kiírás innen történik: }
      Kezd := Kezd + 26 ;
      WriteLn ;
    End ;
  ReadLn ;
End.
```

4. feladat

Írjunk programunkba olyan eljárást, mely indításkor ellenőrzi a programot tartalmazó .EXE állomány méretét, majd ha az nem egyezik a deklarált eredeti mérettel (például vírusos a program), törli a programot tartalmazó állomány tartalmát, majd a programot is!

```
Program Tipnelk4 ;

  { A heap méretét a lehető legnagyobbra állítjuk: }
{$M 16384,0,655360}

Const

    { Ez a típusos állandó tartalmazza az EXE állomány méretét. Az állandónak először 0 értéket
    adunk, a programot lefordítjuk lemezre (Compile/Destination = Disk). Kilépünk az operációs
    rendszerbe a File/DOS Shell menüponton keresztül, és megnézzük az EXE állomány méretét.
    Visszatérünk EXIT paranccsal a keretrendszerbe, és az állandó értékét átírjuk erre a méretre.
    A programot ezután újrafordítjuk. }
  ProgramFajlMeret : LongInt = 0 ;

  BajtokSzama = 10000 ;

  { Ellenőrző eljárás: }
Procedure VirusEllenorzes ;
  Var
    ProgramFajl : File ;
    TombMutato  : Pointer ;
    CiklusValt  : Word ;
  Begin
      { A futó program teljes nevét a 0. parancssor-paraméter tartalmazza. Ha a futtatás a
      keretrendszerből történik, és a lefordított kód még nincs lemezen, akkor a paraméter
      'TURBO.EXE' lesz (amit nem célszerű letörölni...): }
    If Copy(ParamStr(0),Length(ParamStr(0)) - 8,9) = 'TURBO.EXE'
      Then
      Begin
        WriteLn('A programot fordítsa lemezre!') ;
        ReadLn ;
        Halt ;
      End ;
    Assign(ProgramFajl,ParamStr(0)) ;
      { A programot tartalmazó állomány megnyitása egy bájtos rekordmérettel: }
    Reset(ProgramFajl,1) ;
```

```
      { Ha a deklarált és tényleges hossz nem egyezik: }
    If FileSize(ProgramFajl) <> ProgramFajlMeret Then
      Begin
          { BajtokSzama bájt lefoglalása a heap–ből: }
        GetMem(TombMutato,BajtokSzama) ;
          { A terület feltöltése nullákkal: }
        FillChar(TombMutato^,BajtokSzama,#0) ;
          { A lemezen lévő program átírása a heap–ben lévő terület tartalmával, azaz
            nullákkal: }
        For CiklusValt := 1 To FileSize(ProgramFajl) Div
                                BajtokSzama Do
          BlockWrite(ProgramFajl,TombMutato^,BajtokSzama) ;
        BlockWrite(ProgramFajl,TombMutato^,
                   FileSize(ProgramFajl) Mod BajtokSzama) ;
          { A programot tartalmazó állomány lezárása és törlése: }
        Close(ProgramFajl) ;
        Erase(ProgramFajl) ;
          { A lefoglalt terület felszabadítása, üzenet és leállás: }
        FreeMem(TombMutato,BajtokSzama) ;
        WriteLn('A program valószínûleg vírusos volt, ',
                'letörölte önmagát ...') ;
        ReadLn ;
        Halt ;
      End ;
  End ;

  { FŐPROGRAM: }
Begin
  VirusEllenorzes ;
  WriteLn('A program hossza változatlan.') ;

    { Főprogram utasításai... }

  ReadLn ;
End.
```

5. feladat

Írjunk programot, mely állományokat másol egy adott célkatalógusba! A másolandó állományok specifikációinak listáját egy szöveges állományból vegyük! (Az állományok specifikációi nem tartalmazhatnak * és ? karaktereket.) A szöveges állomány és a célkatalógus a program két paramétere legyen!

```
Program Tipnelk5 ;

  { B/K ellenőrzés kikapcsolása: }
{$I-}

Uses
  Crt,
  Dos ;

Var
  ListaFajl     : Text ;
  ForrasFajl    : File ;
  ForrasNev,
  Spec          : PathStr ;
  Katalogus     : DirStr ;
  Nev           : NameStr ;
  Kit           : ExtStr ;
  CelFajl       : File ;
  CelKatalogus  : DirStr ;
  Puffer        : Array[0..32767] Of Byte ;
  Beolvasott,
  Kiirt,
  FajlSzam,
  RegiAttr      : Word ;

  { FŐPROGRAM: }
Begin
  ClrScr ;
    { Ha a parancssor–paraméterek száma nem kettő – használati utasítás kiírása: }
  If ParamCount <> 2 Then
    Begin
      WriteLn('Állományok másolása listaállományból ' +
              'célkatalógusba:') ;
      WriteLn('Használat: TIPNELK3 <listaállspec.> ' +
              '<[d:][katalógus]>') ;
      ReadLn ;
      Halt ;
    End ;

    { Listaállomány megnyitása: }
  Assign(ListaFajl,ParamStr(1)) ;
  Reset(ListaFajl) ;
    { Ha nincs meg a listaállomány – hibaüzenet: }
  If IOResult <> 0 Then
    Begin
      WriteLn(ParamStr(1),' nem található') ;
      ReadLn ;
      Halt ;
    End ;
```

```
  { Aktuális katalógus megjegyzése: }
GetDir(0,Katalogus) ;

  { Megnézzük, van–e célkatalógus: }
CelKatalogus := ParamStr(2) ;
ChDir(CelKatalogus) ;
If IOResult <> 0 Then
  Begin
    WriteLn(CelKatalogus,' katalógus nem található') ;
    ReadLn ;
    Halt ;
  End ;

  { Visszakapcsolás az aktuális katalógusba: }
ChDir(Katalogus) ;

FajlSzam := 0 ;

  { Amíg van állománynév a listaállományban: }
While Not Eof(ListaFajl) Do
  Begin

      { Másolandó állomány nevének beolvasása: }
    ReadLn(ListaFajl,ForrasNev) ;

      { Ha a sor nem üres: }
    If ForrasNev <> '' Then
      Begin

          { Másolandó állomány megnyitása – attribútumának időleges átírása: }
        Assign(ForrasFajl,ForrasNev) ;

          { Az állomány specifikációról a forrásállomány nevének leválasztása: }
        FSplit(FExpand(ForrasNev),Katalogus,Nev,Kit) ;
        GetFAttr(ForrasFajl,RegiAttr) ;
        SetFAttr(ForrasFajl,Archive) ;
        Reset(ForrasFajl,1) ;

          { Válogatás a teendők között a hibakód szerint: }
        Case IOResult Of

            { Minden rendben, másolás: }
          0 : Begin
                If CelKatalogus[Length(CelKatalogus)] <> '\'
                Then
                  CelKatalogus := CelKatalogus + '\' ;
                WriteLn(ForrasNev,' másolása') ;
                Assign(CelFajl,CelKatalogus + Nev + Kit) ;
                Rewrite(CelFajl,1) ;

                  { Blokkonkénti másolás, amíg van mit: }
                Repeat
                  BlockRead(ForrasFajl,Puffer,
                            Sizeof(Puffer),Beolvasott) ;
```

```
                    BlockWrite(CelFajl,Puffer,Beolvasott,Kiirt);
                Until (Beolvasott = 0) Or (Kiirt <> Beolvasott);
                Close(ForrasFajl) ;
                Close(CelFajl) ;

                    { A másolandó állomány attribútumának visszaállítása: }
                SetFAttr(ForrasFajl,RegiAttr) ;
                If Kiirt <> Beolvasott
                Then
                        { Nem sikerült mindent átmásolni: }
                  Begin
                        { Hibaüzenet és a hiányos állomány törlése: }
                    WriteLn('Tele a lemez') ;
                    Erase(CelFajl) ;
                    ReadLn ;
                    Halt ;
                  End
                Else
                        { Másolt állományok számlálása: }
                  Inc(FajlSzam) ;
              End ;
            { Nincs meg az állomány: }
          2 : WriteLn(ForrasNev,' *** nemlétezô állomány') ;
            { Nincs meg a katalógus: }
          3 : WriteLn(ForrasNev,' *** nemlétezô katalógus') ;
            { Tiltott hozzáférés: }
          5 : WriteLn(ForrasNev,' *** hozzáférés tiltott') ;
        End ;
      End ;
  End ;
  { Listaállomány zárása: }
Close(ListaFajl) ;

  { Másolt állományok számának kiírása: }
WriteLn ;
WriteLn(FajlSzam,' állomány átmásolva') ;
Write('  <ENTER>') ;
ReadLn ;
End.
```

6. feladat

Írjunk programot, mely felderíti egy dBase (Clipper) állomány felépítését, megjeleníti az állomány és a mezők jellemzőit, majd az állomány adatrekordjait kilistázza a képernyőre!

```
Program Tipnelk6 ;

{$I-}

Uses
  Crt ;

Type
    { A dBase adatállomány eleje egy 32 bájtos állományleíró rekord: }
  AllomanyLeiro = Record
                    Azonosito    : Byte ;
                    Datum        : Record
                                     Ev, Ho, Nap : Byte ;
                                   End ;
                    RekordSzam   : LongInt ;
                    AdatKezdet   : Word ;
                    RekordMeret  : Word ;
                    NemHasznalt  : Array[1..20] Of Byte ;
                  End ;

    { Minden rekordmezőt egy 32 bájtos mezőleíró rekord definiál: }
  MezoLeiroMut  = ^MezoLeiro ;
  MezoLeiro     = Record
                    MezoNev      : Array[1..10] Of Char ;
                    NemHasznalt1 : Byte ;
                    MezoTipus    : Char ;
                    NemHasznalt2 : Array[1..4] Of Byte ;
                    MezoHossz    : Word ;
                    NemHasznalt3 : Array[1..14] Of Byte ;
                    Kovetkezo    : MezoLeiroMut ;
                  End ;

  RekordTip     = Array[1..65521] Of Char ;

Const
   TablaHossz = 15 ;

Var
  dBaseAllomany : File ;
  AllomanyNev   : String ;
  AllomanyFej   : AllomanyLeiro ;
  Mezo          : MezoLeiro ;
  HeapAllapot,
  ElsoMezo,
  UjMezo,
  AktMezo       : MezoLeiroMut ;
  Rekord        : ^RekordTip ;
  I,
  J,
  MezoSzam,
  TablaKezd     : Word ;
```

```
  C              : Char ;
  AktRek         : LongInt ;
  Beolvasva      : Word ;

Procedure Varj ;
  Begin
    GotoXY(1,25) ;
    Write('Tovább: <ENTER> / Vége: <ESC>') ;
    Repeat
      C := ReadKey ;
    Until C In [^M,^[] ;
    If C = ^[ Then
      Begin
        ClrScr ;
        Halt ;
      End ;
  End ;

  { FŐPROGRAM: }
Begin
  Write('A dBase (Clipper) állomány neve: ') ;
  ReadLn(AllomanyNev) ;
  If AllomanyNev = '' Then
    Halt ;

    { A dBase adatállomány megnyitása és az állományleíró rekord beolvasása: }
  Assign(dBaseAllomany,AllomanyNev) ;
  Reset(dBaseAllomany,1) ;
  If IOResult <> 0 Then
    Begin
      WriteLn('A(z) ',AllomanyNev,' állomány nem található') ;
      ReadLn ;
      Halt ;
    End ;

  BlockRead(dBaseAllomany,AllomanyFej,32,Beolvasva) ;
  If Beolvasva <> 32 Then
    Begin
      WriteLn('Hiba az állomány olvasása közben') ;
      ReadLn ;
      Halt ;
    End ;
  Mark(HeapAllapot) ;
    { Az egyes mezők jellemzőinek beolvasása és felvitele dinamikus rekordok láncába: }
  ElsoMezo := Nil ;
  BlockRead(dBaseAllomany,Mezo,32) ;
    { Az utolsó mezőleíró rekord után $0D áll az állományban: }
  While (IOResult = 0) And (Mezo.MezoNev[1] <> Char($0D)) Do
    Begin
      New(UjMezo) ;
```

```
    UjMezo^ := Mezo ;
    UjMezo^.Kovetkezo := Nil ;
    If ElsoMezo = Nil
    Then
      ElsoMezo := UjMezo
    Else
      AktMezo^.Kovetkezo := UjMezo ;
    AktMezo := UjMezo ;
    BlockRead(dBaseAllomany,Mezo,32) ;
  End ;

  { Az állomány jellemzőinek kiírása: }
ClrScr ;
With AllomanyFej Do
  Begin
    Write('Az állományhoz ') ;
    If Azonosito = $03 Then
      Write('nem ') ;
    WriteLn('tartozik MEMO állomány.') ;
    With Datum Do
      WriteLn('Az utolsó módosítás dátuma : ',
              Ev,'-',Ho,'-',Nap) ;
    WriteLn ;
    WriteLn('Rekordok száma              : ',RekordSzam) ;
    WriteLn('Rekordméret                 : ',RekordMeret,
            ' bájt') ;
  End ;

  { Az egyes mezők jellemzőinek kiírása táblázatos formában: }
WriteLn ;
WriteLn('Nº Mezőnév     Tip Hossz      ' +
        'Nº Mezőnév     Tip Hossz      ' +
        'Nº Mezőnév     Tip Hossz') ;
For I := 1 To 80 Do
  Write('-') ;
MezoSzam := 1 ;
TablaKezd := 0 ;
AktMezo := ElsoMezo ;
While AktMezo <> Nil Do
  Begin
    GotoXY(28 * ((MezoSzam - 1) Div TablaHossz) + 1,
           9 + (MezoSzam - 1) Mod TablaHossz) ;
    With AktMezo^ Do
      Begin
        Write(TablaKezd + MezoSzam:2,' ') ;
        For I := 1 To 10 Do
          Write(MezoNev[I]) ;
        Write('  ',MezoTipus,'  ') ;
        Case MezoTipus Of
          'C' : Write(MezoHossz:5) ;
          'N' : Write(Lo(MezoHossz):2,'/',Hi(MezoHossz):2) ;
```

```
          Else  Write(Lo(MezoHossz):5) ;
        End ;
      End ;
    Inc(MezoSzam) ;
    If MezoSzam > TablaHossz * 3 Then
      Begin
        MezoSzam := 1 ;
        Inc(TablaKezd,TablaHossz * 3) ;
        Varj ;
        Window(1,9,80,25) ;
        ClrScr ;
        Window(1,1,80,25) ;
      End ;
    AktMezo := AktMezo^.Kovetkezo ;
  End ;
If MezoSzam <= TablaHossz * 3 Then
  Varj ;

  { Az adatállomány listázása rekordonként: }
ClrScr ;
If AllomanyFej.RekordSzam = 0 Then
  Begin
    Write('Az adatállomány üres...') ;
    ReadLn ;
    Halt ;
  End ;
If AllomanyFej.RekordMeret > 65521 Then
  Begin
    Write('Az adatrekord mérete a megjelenítéshez túl nagy...');
    ReadLn ;
    Halt ;
  End ;

GetMem(Rekord,AllomanyFej.RekordMeret) ;
Seek(dBaseAllomany,AllomanyFej.AdatKezdet) ;
AktRek := 1 ;
While AktRek <= AllomanyFej.RekordSzam Do
  Begin
      { Adatrekord beolvasása: }
    BlockRead(dBaseAllomany,Rekord^,AllomanyFej.RekordMeret) ;
      { Ha nem törölt a rekord, megjelenítjük: }
    If Rekord^[1] = ' ' Then
      Begin
        AktMezo := ElsoMezo ;
        I := 1 ;
```

```
        While AktMezo <> Nil Do
          Begin
              { A memo típusú mező nem megjeleníthető, így csak a karakteres, logikai,
              dátum és numerikus mezőket írjuk ki: }
            If AktMezo^.MezoTipus In ['C','L','D','N'] Then
              Begin
                Write(AktMezo^.MezoNev,' : ') ;
                  { A mező karaktereinek megjelenítése: }
                For J := 1 To Lo(AktMezo^.MezoHossz) Do
                  Begin
                    If WhereY = 23 Then
                      Begin
                        Varj ;
                        ClrScr ;
                      End ;
                    If WhereX = 80 Then
                      GotoXY(14,WhereY + 1) ;
                    Case AktMezo^.MezoTipus Of
                        { A dátumot ÉÉÉÉ/HH/NN formában jelenítjük meg: }
                      'D' : Begin
                              If J In [5,7] Then
                                Write('/') ;
                              Write(Rekord^[I + J]) ;
                            End ;
                        { A logikai mező .F. érték esetén szóközt is tartalmazhat: }
                      'L' : If Rekord^[I + J] <> 'T'
                            Then
                              Write('F')
                            Else
                              Write('T') ;
                    Else  Write(Rekord^[I + J]) ;
                  End ;
                End ;
              WriteLn ;
            End ;
                { Léptetés a következő mezőt leíró rekordra: }
            AktMezo := AktMezo^.Kovetkezo ;
            Inc(I,J) ;
          End ;
        Varj ;
        ClrScr ;
      End ;
    Inc(AktRek) ;
  End ;

FreeMem(Rekord,AllomanyFej.RekordMeret) ;
Release(HeapAllapot) ;
Close(dBaseAllomany) ;
```

```
  GotoXY(1,25) ;
  Write('Vége - <ENTER>') ;
  ReadLn ;
  ClrScr ;
End.
```

7. feladat

Mentsük el a 80*25–ös képernyő aktuális tartalmát lemezre! Közvetlen képernyő– memóriába való írással csíkozzuk be a képernyőt, majd töltsük vissza lemezről az eredeti képernyő tartalmát!

```
Program Tipnelk7 ;

Uses
  Crt ;

Const
    { Színes 80*25–ös üzemmód esetén a sorok és oszlopok száma:}
  Sor       = 25 ;
  Oszlop    = 80 ;
  Villogas : Boolean = False ;

Type
    { A képernyő minden egyes karakterhelyéhez a képernyőmemóriában két bájt tartozik: a
    karakter ASCII kódja és az attribútum bájt, mely a karakterhely háttérszínét, a karakter
    színét, és az esetleges villogást határozza meg (mint TextAttr): }
  KarHely = Record
              Kar  : Char ;
              Attr : Byte ;
            End ;
    { A képernyő memóriafoglalása: Sor * Oszlop * (2 bájt); a karakterek sorfolytonosan
    helyezkednek el a tárban: }
  KepTip  = Array[1..Sor] Of Array[1..Oszlop] Of KarHely ;

Var
    { A képernyő első bájtjának szegmenscíme: }
  KepKezdSeg    : Word ;
    { Mutató a képernyőmemória elejére: }
  Kep           : ^KepTip ;
    { Állomány a képernyő elmentéséhez: }
  KepFajl       : File ;
  Mono          : Boolean ;
  Hatterszin,
  Karakterszin,
```

```
  Attr,
  I,
  J           : Byte ;

  { A képernyőmemória kiírása típus nélküli állományba: }
Procedure KepKi ;
  Begin
      { Állomány megnyitása. A rekordméret a képernyőmemória egy sora: }
    Rewrite(KepFajl,Oszlop * 2) ;
      { Egyszerre Sor darab rekord (az egész képernyő) kiírása lemezre: }
    BlockWrite(KepFajl,Kep^,Sor) ;
    Close(KepFajl) ;
  End ;

  { A képernyőmemória beolvasása típus nélküli állományból: }
Procedure KepBe ;
  Begin
      { Állomány megnyitása. A rekordméret a képernyőmemória egy sora: }
    {$I-}
    Reset(KepFajl,Oszlop * 2) ;
    {$I+}
    If IOResult = 0 Then
      Begin
          { Sor darab rekord (az egész képernyő) beolvasása az állományból: }
        BlockRead(KepFajl,Kep^,Sor) ;
        Close(KepFajl) ;
      End ;
  End ;

  { FŐPROGRAM: }
Begin
    { 80x25–ös színes, szöveges üzemmód beállítása: }
  TextMode(CO80) ;
    { Képernyőre mutató változó beállítása: }
  Mono := Lo(LastMode) = 7 ;
  If Mono
  Then
      { Monokróm monitor esetén: }
    KepKezdSeg := $B000
  Else
      { Színes monitor esetén: }
    KepKezdSeg := $B800 ;
  Kep := Ptr(KepKezdSeg,0) ;
    { Teleírjuk a képernyőt, majd elmentjük: }
  For I := 1 To 20 Do
    WriteLn('ÆÆÆ   Most itt vagyok, csíkozás után újra látszom!');
```

```
  ReadLn ;
  Assign(KepFajl,'Kep.Dat') ;
  KepKi ;
    { Új, csíkos képernyő készítése közvetlen memóriába való írással. Attr beállítása (0..3 bitek:
    karakterszín; 4..6 bitek: háttérszín; 7. bit: villogás): }
  If Mono
  Then
    Begin
      Hatterszin := 0 ;
      Karakterszin := 7 ;
    End
  Else
    Begin
      Hatterszin := Magenta ;
      Karakterszin := Black ;
    End ;
  Attr := $80 * Ord(Villogas) + $10 * Hatterszin + Karakterszin;

    { A képernyőmemória feltöltése Attr attribútumú '–' karakterekkel: }
  For I := 1 To Sor Do
    For J := 1 To Oszlop Do
      Begin
        Kep^[I,J].Kar := '-' ;
        Kep^[I,J].Attr := Attr ;
      End ;
  ReadLn ;

    { A régi képernyő visszatöltése a lemezről: }
  KepBe ;
  ReadLn ;
End.
```

636 Ft. LEMEZZEL