Gyakorlat
a szöveges fájlok bevezetéséhez

(téma: szöveges fájlok a feladatmegoldásban)

0. Bevezetésként

Az elkövetkezőekben a fájlok feladathoz kapcsolását –az egyszerűség kedvévért– kizárólag kódolási többletnek tekintjük.

A szöveges fájlok jellemzője, hogy

- **karakter**ekből állnak,
- amelyeket sokszor **sorvégjel**ekkel határolt (CrLf =10- és 13-kódú jelpárral) sorokká egyesítünk,
- így akármilyen **egyszerűbb szövegszerkesztő**vel is szerkeszthetjük, ill. olvashatjuk; az „akármilyen"-ek közé tartozik (és kézenfekvő választásnak látszik) maga a Pascal programszerkesztője, annál is inkább, mert így a konzolalkalmazások és az ablakos alkalmazások karakterkód különbözősége kevesebb gondot fog okozni (a kódolási különbség az ékezetes betűk terén áll fenn),
- **vagy olvasni** lehet belőle, **vagy írni** lehet egy időben!

1. Specifikációban

Röviden elintézhető, mert e miatt nincs min változtatni. A feladat leírása mit sem változik. Egyszerűen az input esetleg az output nem a konzol lesz, hanem fájl vagy fájlok. Ebből persze következik, hogy ha van is az inputtal szemben elvárás (miért is ne lenne), akkor a beolvasást

1. vagy nem terheljük ellenőrzéssel,
2. vagy ellenőrzünk, de ha nem teljesül a feltétel, akkor mást nem tehetünk, mint hogy hibaüzenet kíséretében azon nyomban megállunk.

2. Algoritmusban

Semmi extra: néhány magától értetődő funkciójú utasítás, amelyet most nem részletezünk, mondván: kódolási kérdésként kezeljük csupán a fájlkezelést.

3. A Pascal kódolásban

3.1. Pascal tudnivalók – mindenek előtt

A fájlok az eddigi adatoktól egy dologban alapvetően eltérnek. Abból következőleg, hogy a fájlok a program futásától függetlenül léteznek (már vagy majd) a háttértáron többlet „szertartással" kell hozzájuk fordulni: **meg** kell őket **nyitni**. Amiatt, hogy a kezelés minél kevesebb idővel járjon, a fájlok egy része mindig a memóriában (egy ún. fájlpufferben) tartózkodik. Ez része az ún. **fájlváltozó**nak, amely a programban szimbolizálja magát a fájlt. Hogy a fájl részben és időlegesen a memóriában tartózkodik, ennek is van következménye:

ha a fájlra már nincs szükség, vagy a fájlt már elkészítettük, akkor **le** kell **zárni**. (Ekkor a fájlpuffer kiürül, és már nem foglalja fölöslegesen a helyet a memóriában, kiíráskor pedig ekkor készül el a határtáron a fájl végleges bejegyzése.)

Az alábbi táblázatokban összefoglaljuk azokat a Pascal utasításokat, -csoportokat, amelyek a fájlokkal kapcsolatban szokás szerint használandók.

Jó tudni, hogy (sok programozási nyelv, így a Pascal szerint is) a konzol is szöveges fájl, amely egy speciális periférián leledzik/létesül. Így, amit már kitapasztalt a billentyűről történő beolvasásról, illetőleg a képernyőre történő kiírásról, az érvényes a háttértáron létező szöveges fájlok esetében is. (L. alább a `Read`, `Write` utasításokat.)

Olvasandó (=input) fájl	
Feltétel:	Létezik
Műveletkategória	**Pascal utasítás(-csoport)**
Deklaráció:	`Var beF:Text;`[1]`//fájlváltozó inputfájl esetén`

Megnyitás:	`Assign(beF,fN);//összekapcsolja a fájl-`
	`  //változót a háttértár fN nevű fájljával`
	`Reset(beF);//olvasásra nyitja, ha létezik az`
	`  //fN nevű fájl`

Olvasás:	`Read(beF,v1,…);//a fájl aktuális sorából`
	`  //beolvas a v1,… változókba`
	
	`Readln(beF,v1,…);//a fájl aktuális sorából`
	`  //beolvas a v1,… változókba, majd a köv.`
	`  //sorra áll`

Fájlvég-ellenőrzés:	`Eof(beF) //a függvény értéke True, ha már az`
	`  //utolsó karakterést is beolvastuk,`
	`  //különben False`

Lezárás:	`Close(beF);//lezárja fájlt`

Írandó (=output) fájl	
Feltétel:	Nem létezik
Műveletkategória	**Pascal utasítás(-csoport)**
Deklaráció:	`Var kiF:Text;//fájlváltozó outputfájl esetén`

Megnyitás:	`Assign(kiF,fN);//összekapcsolja a fájlvál-`
	`  //tozót a háttértár fN nevű`
	`  //fájljával`
	`Rewrite(kiF);//(újra)írásra nyitja az fN ne-`
	`  //vű fájlt`

Irás:	`Write(kiF,e1,…);//a fájl aktuális sorához`
	`  //írja az e1,… értékeket`
	
	`Writeln(kiF,e1,…);//a fájl aktuális sorához`
	`  //írja az e1,… értékeket, majd lezárja`
	`  //CrLf-fel`

Lezárás:	`Close(kiF);//lezárja fájlt`

[1] `Text = File of Char`

Mint látható: vannak a használatnak feltételei. Felvetődik a kérdés, miként lehet ellenőrizni, hogy egy olvasásra megnyitandó fájl valójában létezik-e, vagy az éppen generálandó fájl még nincs. (Mert ez esetben felülíródna.)

Az ellenőrzés pont úgy történik, mint ahogy a billentyűről történő olvasáskor történt, vagyis az `IOResult` függvénnyel, amely az utolsó I/O művelet sikeres vagy sikertelen voltát árulja el. Ha sikeres volt, akkor értéke 0, különben valami –hibára utaló kódú– nem 0 egész szám.

Nézzük azt a függvénypárt, amely alkalmas arra, hogy megnyissa a paraméterül adott nevű fájlt, és a sikeres vagy sikertelen voltát egy logikai értékkel visszaadja!

```
Function OlvasasraNyit(Var f:Text; Const fN:String):Boolean;
Begin
  Assign(f,fN);
  {$i-}
  Reset(f);
  {$i+}
  OlvasasraNyit:=IOResult=0
End;{OlvasasraNyit}
```

```
Function IrasraNyit(Var f:Text; Const fN:String):Boolean;
Begin
  If OlvasasraNyit(f,fN) then
  Begin//hiba van, hiszen olvasásra megnyitható
    Close(f);
    IrasraNyit:=False
  End
    else
  Begin//nem létezik a fájl, eddig OK
    {$i-}
    Rewrite(f);
    {$i+}
    IrasraNyit:=IOResult=0
  End;
End;{IrasraNyit}
```

A *pirossal és dőlten* szedett „kommentsorok” kellenek ahhoz, hogy (csak) a `Reset`-nél / `Rewrite`-nál elkövetett esetleges hibát az `IOResult` érzékelhesse.

3.2. Tipikus példák

3.2.1. Számok fájlban, darabszám nélkül

Számsorozat beolvasása tömbbe fájlból. Állapodjunk meg abban, hogy a számok különkülön sorban találhatók, az elemszámot azonban nem tartalmazza.

Figyeljük meg az alábbi programdarabot, amelyben felhasználjuk az `OlvasasraNyit` függvényt!

```
…
Const
  MaxN=100;
Type
  TSorozat=Array [1..MaxN] of Integer;
Var
  t:TSorozat;
  N:Integer;
...
Procedure TombFeltoltesFajlbol(Var t:TSorozat; Var N:Integer);
  Var
    f:Text;
    fN:String;
    siker:Boolean;
```

```
Begin
  //fájlmegnyitás:
  Repeat
    Write('Kérem a sorozatot tartalmazó fájl nevét:');
    Readln(fN);
    siker:=OlvasasraNyit(f,fN);
    If not siker then Writeln('Nem létező fájl, kérem újra!');
  Until siker;
  //adatbeolvasás:
  N:=0;
  While not Eof(f) and (N<MaxN) do
  Begin
    Inc(N); Readln(f,t[N]);
  End;{While}
  If not Eof(f) then//maradt még beolvasatlan a fájlban
  Begin
    Writeln('Túl sok adat van a fájlban. A program megáll.');
    Halt(1);
  End;{If}
  //fájllezárás:
  Close(f);
End;{TombFeltoltesFajlbol}
```

3.2.2. *Számok fájlban darabszámmal kezdve*

A fájl most eggyel több sort tartalmaz. Az első sorban a számok (azaz most a további sorok) száma legyen. A további sorok csak egy-egy számot tartalmaz.

Gondolja meg, miben tér lel az előzőtől!

3.2.3. *Számmátrix fájlban méretekkel kezdve*

Logikus ábrázolása egy számokból álló mátrixnak fájlban a következő:

1. sor: `N M` – N×M-es lesz a mátrix
2. sor: $x_{1\,1}$ $x_{1\,2}$ … $x_{1\,M}$ – a mátrix `1.` sora

…

`i+1`. sor: $x_{i\,1}$ $x_{i\,2}$ … $x_{i\,M}$ – a mátrix `i.` sora

…

`N+1`. sor: $x_{N\,1}$ $x_{N\,2}$ … $x_{N\,M}$ – a mátrix `N.` sora

Feltétel, hogy az egyes mátrixelemek között legalább egy szóköz legyen (s ne más karakter).

Az `X` mátrixot az f fájlból feltöltő programdarab –legegyszerűbb esetben– az alábbi lehet:

```
…
For i:=1 to N do
Begin
  For j:=1 to M do
  Begin
    Read(f,X[i,j]);
  End;{For j}
  Readln(f);//a sorvége jel megevése
End;{For i}
…
```

Vegyük észre, hogy hol `Read`, hol `Readln`-t használtunk, alapos okkal!

3.2.4. *Számmátrix fájlba írása méretekkel kezdve*

Az előbbi fájlszerkezetnek megfelelő fájl létrehozását végzi az alábbi kóddarab (a fájlt most a `kiF` fájlváltozó jelöli):

```
…
//fájlmegnyitás:
Repeat
  Write('Kérem az outputfájl nevét (és útvonalát)!');
  Readln(fN);
Until IrasraNyit(kiF,fN);
//mátrix kiírása a fájlba:
Writeln(kiF,N,' ',M);//az elválasztó szóközzel együtt kiírás
For i:=1 to N do
Begin
  For j:=1 to M do
  Begin
    Write(kiF,X[i,j],' ');//az elem és a követő szóköz kiírása
  End;{For j}
  Writeln(kiF);//a sorvége jel megevése
End;{For i}
//a fájl lezárása, e nélkül a fájlnak nem maradna nyoma:
Close(kiF);
…
```

4. Házi feladatok

1. Készítsen egy „torzó fájlt”, az AltRutinok.inc mintájára, amely tartalmazza a fentebb (3.1.-ben) részletezett fájlnyitás szertartást elvégző rutinpárt! Legyen a neve a tartalmára utaló, pl.: FajlNyitasok.inc !
2. A múlt órán feldolgozott ’madaras’ feladat megoldása úgy, hogy az adatokat fájl tartalmazza. Az eredmények keletkezhetnek a képernyőn. (Persze, ha szeretné, akkor keletkezhet egy output fájlban **is**!) Alkalmazza az 1. feladatban körvonalazott inklúd-fájlt!