Pázmány Péter Katolikus Egyetem
Információs Technológiai Kar

Bevezetés a programozásba I

4. gyakorlat

PLanG: Szekvenciális fájlkezelés

© 2011.10.04. Giachetta Roberto
groberto@inf.elte.hu
http://people.inf.elte.hu/groberto

Szekvenciális fájlkezelés
Fájlok használata

- A programok meghatározó része használ fájlokat az adatok kezelésére, hiszen nem mindig közvetlenül adjuk meg az adatokat (megeshet, hogy több adatforrást is kell használnunk)
- A fájloknál hasonló a működés, mint a szokásos bemenet és kimenet esetében:
  - több típusú adat is előfordulhat, tetszőleges sorrendben, csak figyelnünk kell arra, hogy a megfelelőt olvassuk ki
  - ha az adatokat sorban olvassuk be, akkor *szekvenciális fájlról* beszélünk, mi csak ilyen fájlokkal foglalkozunk
  - tudjuk, mikor van vége az állománynak, ugyanis minden fájl végén egy *fájl vége jel* (end of file: EOF), és le tudjuk kérdezni, hogy ezt sikerült-e beolvasni

Szekvenciális fájlkezelés
Fájlok használata

- Valamikor a bemenő adatokat olvassuk be a fájlból, valamikor a kimenő adatokat írjuk ki, ennek megfelelően megkülönböztetünk:
  - *bemeneti fájl*t: csak beolvasásra szolgál
  - *kimeneti fájl*t: csak adatkiírásra szolgál
  - *be- és kimeneti fájl*t: egyszerre írhatunk benne és olvashatunk belőle (PLanG-ban nincs ilyen)
- A fájlok ugyanúgy típussal rendelkező változók a programban
  - vannak nyelvek, ahol egy fájl csak egy típusú adatokat tartalmazhat, és vannak olyanok, ahol a fájlban bármi lehet
  - de a fájlműveletek külön utasításokat igényelnek, nem teljesen úgy dolgozunk velük, mint a többi változóval

Szekvenciális fájlkezelés
Logikai és fizikai fájl

- Amikor deklarálunk egy fájl típusú változót, akkor létrehozunk egy *logikai fájlnevet*, ez lesz a változó neve
- A logikai fájlnévből önmagában nem derül ki, melyik az a tényleges fájl, amit kezelni szeretnénk, ezért a változóhoz egy *fizikai fájlnevet* rendelünk, ez a tényleges fájl neve, ami a lemezünkön van, pl.: **`/home/groberto/adatok.txt`**
- A logikai fájlnevet a program során összepárosítjuk a fizikaival, csak ettől a ponttól használhatjuk a fájlt
  - egy logikai fájlnévhez több fizikai név is tartozhat a program során, de egyszerre csak egy
  - egy fizikai név többször is előfordulhat a program során, de egyszerre csak egy változóhoz tartozhat

Szekvenciális fájlkezelés
PLanG fájlok

- A PLanG környezet nem kezel tényleges fájlokat, csak szimulálja a működésüket
  - nincsenek fizikai fájlok, hanem a fájlablakok a bemeneti, illetve kimeneti ablakok mellett jelennek meg a fájlok
  - bemeneti fájl esetén ugyanúgy kell az adatokat beírni, mintha a bemenetre írnánk őket
  - ha kimeneti fájlt is használunk, az eredmények a kimeneti fájl ablakában jelennek meg
  - de azért itt is kellenek fizikai fájlnevek, amelyek megjelennek ablakcímként
- PLanG-ban két fájltípust különböztetünk meg: **`BEFÁJL`**, **`KIFÁJL`**
  - pl.: **`f : BEFÁJL`**

Szekvenciális fájlkezelés
PLanG fájlok

- Egy fájlban bármilyen típusú adatot tárolhatunk, egyben többfélét is, bármilyen sorrendben, csak mindig a megfelelő típusra kell hivatkoznunk (ahogy megszokhattuk)
- A fizikai fájlnevet a **`MEGNYIT`** paranccsal párosítjuk
  - a fizikai fájlnak nem kell megadnunk az elérését, hiszen nem tényleges fájlt fog hozzárendelni a környezet
  - pl.: **`MEGNYIT f : "adatok.txt"`**

```
** az f logikai névhez az adatok.txt fájl
** (ablak) fog tartozni
```

- A megnyitott fizikai fájlokat be is kell zárni, mert addig más program nem férhet hozzá
  - a **`LEZÁR`** paranccsal zárjuk be, pl.: **`LEZÁR f`**

Szekvenciális fájlkezelés
PLanG fájlok

- Lehetséges műveletek az olvasás és kiírás (a típustól függően csak egyik alkalmazható)
- Beolvasást változóba végezhetjük
  - pl.:
```
BE f : a
** ekkor a fájl aktuális eleme a változóba
** kerül, a típusa lehet bármi, amit be
** szeretnénk olvasni
```
- Kiírásnál változót, értéket, vagy tetszőleges szöveg eredményű kifejezést is megadhatunk
  - pl.: `KI f : a," : ",b`
  - bármit kiírhatunk, amit a kimeneti ablakra

Szekvenciális fájlkezelés
Példa

Feladat: Olvassunk be 10 pozitív egész számot fájlból, és adjuk meg a legnagyobb számot a kimenetre.
- tudjuk, hogy a bemeneti fájlban van legalább 10 adat, ezért dolgozhatunk számláló ciklussal
- a megoldáshoz a maximumkeresés tételét alkalmazzuk

Specifikáció:
- bemenet: számok (*t*) sorozata, ami fájlban van (*f*)
- kimenet: a számok maximuma (*max*)

Szekvenciális fájlkezelés
Példa

Megoldás:
```
PROGRAM fajl_maximuma
   VÁLTOZÓK:
      f : BEFÁJL, ** bemenő adatok
      i, max, t : EGÉSZ
   MEGNYIT f : "szamok.dat" ** társítás
   i := 0
   max := 0
   CIKLUS AMÍG (i < 10)
      BE f : t ** beolvasunk egy számot
      HA (t > max) AKKOR
         max := t
      HA_VÉGE
```


Szekvenciális fájlkezelés
Példa

Megoldás:
```
      i := i + 1
   CIKLUS_VÉGE
   LEZÁR f ** nincs már szükségünk a fájlra
   KI: "A számok maximuma: ", max
PROGRAM_VÉGE
```


Szekvenciális fájlkezelés
Feldolgozás végjelig

- Persze a legtöbb esetben nem lehetünk biztosak, mennyi adat van a fájlban, ezért a fájl vége jelig kell olvasnunk
- A fájl végét a PLanG-ban a **`VÉGE`** utasítással tudjuk lekérdezni
  - ez egy logikai eredményt ad, és igaz, ha eljutottunk az EOF jelig, pl.: **`VÉGE f`**
  - tehát a beolvasó utasítás nem csak az adatokat, hanem az EOF jelet is beolvassa, ezért figyelnünk kell a feldolgozás közben, hogy mit olvastunk be, adatot, vagy a vége jelet, újabb elágazás kell a ciklusba
  - ha a vége jelet olvastuk be, akkor be kell fejeznünk a feldolgozást

Szekvenciális fájlkezelés
Feldolgozás végjelig

Feladat: Határozzuk meg, hány sornyi szöveg van egy fájlban.
- bármennyi szám lehet a fájlban, ezért a fájl vége jelig kell feldolgoznunk, sőt, előfordulhat, hogy egyáltalán nincsenek adatok a fájlban, ekkor nem kell semmit sem csinálnunk
- a sorok számát megállapíthatjuk az alapján, hányszor sikerül sorvége jelet karakterként beolvasnunk, ekkor karakterenként olvassuk a fájlt
- a számlálás tételével megnézzük, sorvége jelet (SV) sikerült-e beolvasni, ha igen, növeljük a számlálót
- a ciklus addig tart, amíg vége nincs a fájlnak, és mindig meg kell néznünk, hogy a beolvasott karakter nem az EOF-e, tehát két elágazásunk lesz egymásba ágyazva

Szekvenciális fájlkezelés
Példa

Specifikáció:

- bemenet: karaktereket (*k*) tartalmazó fájl (*f*)
- kimenet: sorok száma (*c*)

Megoldás:

```
PROGRAM sorok_szama
  VÁLTOZÓK:
    f : BEFÁJL,
    k : KARAKTER,
    c : EGÉSZ
  MEGNYIT f : "szoveg.txt"
  c := 0 ** számláló nullázás
```


Szekvenciális fájlkezelés
Példa

Megoldás:

```
  CIKLUS
    BE f : k
    HA (NEM VÉGE f) AKKOR
      ** újabb elágazás, mit olvastunk be
      HA (k = SV) AKKOR
      ** számláló elágazása
        c := c + 1
      HA_VÉGE
    HA_VÉGE
  AMÍG (NEM VÉGE f)
  LEZÁR f
  KI: "Sorok száma: ", c
PROGRAM_VÉGE
```


Szekvenciális fájlkezelés
Előreolvasás

- Jó lenne, ha ezt a további elágazást ki tudnánk küszöbölni, mert csak bonyolítja a program szerkezetét
- Az elágazás csak arra kell, hogy megvizsgálja, nem-e a fájl vége jelet olvastuk be, de a ciklusfeltétel is ugyanez, így elég lenne az utóbbi, csak közvetlenül a beolvasás után kéne helyeznünk, hogy közte ne legyen feldolgozás
- Ötlet: legyen a beolvasás a ciklusmag utolsó utasítása, és eléje tesszük a feldolgozást, így minden beolvasott érték a következő ciklusléptésben kerül feldolgozásra, ha az első elemet a ciklus előtt olvassuk be, akkor ez nem okoz problémát
- Ezt a beolvasási szerkezetet *előreolvasás*nak nevezzük (hiszen egy adott ciklusléptésben a előre olvassuk a következőben feldolgozandó adatot)

Szekvenciális fájlkezelés
Előreolvasás

- Használjuk előtesztelő ciklust, így már az első beolvasás után vizsgálhatjuk a feltételt
  - ha üres a fájl, akkor a ciklus nem kerül lefutásra, hiszen az első feltételvizsgálat hamis értéket fog produkálni
- Szerkezete:

```
BE <fájlnév> : <változó> ** első beolvasás
CIKLUS AMÍG (NEM VÉGE <fájlnév>)
  ** feltételvizsgálat közvetlenül a beolvasás
  ** után
  <változó feldolgozása>
  BE <fájlnév> : <változó>
  ** következő elem beolvasása
CIKLUS_VÉGE
```


Szekvenciális fájlkezelés
Példa

Feladat: Egy egész számokat tartalmazó fájlból írjuk ki a páros számokat egy másik fájlba.

- használjunk előreolvasást
- minden lépésben ellenőrizzük a feltételt (kettővel oszthatóság), és azonnal kiírjuk az eredményt a feldolgozás során, tehát elemenkénti feldolgozást végzünk
- legyenek a fizikai fájlok bemenet.dat és kimenet.dat, míg a logikai fájlok *f_be* és *f_ki*

Specifikáció:

- bemenet: egész számok (*x*) egy fájlban (*f_be*)
- kimenet: a páros számok egy fájlban (*f_ki*)

Szekvenciális fájlkezelés
Példa

Megoldás:

```
PROGRAM paros_be_ki
  VÁLTOZÓK:
    f_be: BEFÁJL, ** bemenet
    f_ki: KIFÁJL, ** kimenet
    x: EGÉSZ

  MEGNYIT f_be: "bemenet.txt" ** társítás
  MEGNYIT f_ki: "kimenet.txt"
  ** előreolvasás:
  BE f_be : x ** első adat
```

Szekvenciális fájlkezelés
Példa

Megoldás:

```
CIKLUS AMÍG (NEM VÉGE f_be)
  ** feltétel vizsgálat
  HA (x MOD 2 = 0) AKKOR
    ** páros, ha kettővel osztható
    KI f_ki : x, SV ** kiíratás
  HA_VÉGE
  BE f_be : x ** következő elem
CIKLUS_VÉGE
LEZÁR f_be ** mindkét fájl lezárása
LEZÁR f_ki
PROGRAM_VÉGE
```


Szekvenciális fájlkezelés
Példa

Feladat: Adott két valós számokat tartalmazó fájl, olvassuk be egyenként a számokat a fájlból, és a két szám összegét, valamint az összes szám összegét írjuk ki a egy kimenő fájlba.

- ismét előreolvasást használunk mindkét fájlban
- párhuzamosan olvasunk mindkét fájlban két valós változóval (elemenkénti feldolgozás), egy harmadik valós változóban számoljuk az összeget (összegzés)

Specifikáció:

- bemenet: valós számok (x, y) két fájlban ($f1$, $f2$)
- kimenet: a számok összegei ($x+y$), valamint az összes szám összege (sum) egy fájlban ($f3$)

Szekvenciális fájlkezelés
Példa

Megoldás:

```
PROGRAM ket_fajl_osszegzese
  VÁLTOZÓK:
    f1, f2: BEFÁJL, ** bemenet
    f3: KIFÁJL, ** kimenet
    x, y, sum: EGÉSZ

  sum := 0
  MEGNYIT f1: "be1.txt" ** társítás
  MEGNYIT f2: "be2.txt" ** társítás
  MEGNYIT f3: "ki.txt"
```


Szekvenciális fájlkezelés
Példa

Megoldás:

```
** előreolvasás mindkét fájlból:
BE f1 : x
BE f2 : y

CIKLUS AMÍG (NEM VÉGE f1) ÉS (NEM VÉGE f2)
  ** feltétel mindkét bemenetre

  ** feldolgozás:
  KI f3 : x + y, SV ** két szám összege
  sum := sum + x + y ** globális összeg
```


Szekvenciális fájlkezelés
Példa

Megoldás:

```
    ** következő adatok beolvasása:
    BE f1 : x
    BE f2 : y
  CIKLUS_VÉGE

  LEZÁR f1
  LEZÁR f2

  KI f3 : "összeg: ", sum, SV ** összeg kiírás
  LEZÁR f3 ** kimenő fájl lezárás
PROGRAM_VÉGE
```


Szekvenciális fájlkezelés
Példa

Feladat: adjunk meg 10 szót egy fájlban, írjuk ki az ‘a’ betűvel kezdődő szavakat, majd a ‘b’ betűvel kezdődő szavakat a kimenetre

- a szavakat tömbbe tesszük, a tömb egy eleme: `szavak[i]`
- a szöveg első karakterét le tudjuk kérdezni az indexeléssel, mivel már eleve tömbbe tesszük a szavakat, két indexelés lesz egymás mögött, a tömb egy elemének első betűje: `szavak[i][0]`
- összesen 2 ciklus kell, egy ciklus beolvasás, valamint az ‘a’ betűs szavak kiíratása, a második a ‘b’ betűs szavak kiíratása
- használjunk elemenkénti feldolgozást

Szekvenciális fájlkezelés
Példa

Specifikáció:

- bemenet: egy szöveg tömb (*szavak*)
- kimenet: az ‘a’ betűvel kezdődő szavak, valamint a ‘b’ betűvel kezdődő szavak

Megoldás:

```
PROGRAM a_b_betus_szavak
  VÁLTOZÓK:
    bemenet : BEFÁJL, szavak: SZÖVEG[10],
    i: EGÉSZ

  MEGNYIT bemenet : "szavak.txt"
```


Szekvenciális fájlkezelés
Példa

Megoldás:

```
  i := 0 ** első ciklus
  CIKLUS AMÍG (i<10)
    BE bemenet : szavak[i]

    HA (szavak[i][0] = 'a') AKKOR
      KI: szavak[i]
    HA_VÉGE
    i := i + 1
  CIKLUS_VÉGE

  LEZÁR bemenet ** már nincs szükségünk a fájlra
```


Szekvenciális fájlkezelés
Példa

Megoldás:

```
  i := 0 ** második ciklus
  CIKLUS AMÍG (i<10)
    HA (szavak[i][0] = 'b') AKKOR
      KI: szavak[i]
    HA_VÉGE
    i := i + 1
  CIKLUS_VÉGE
PROGRAM_VÉGE
```


Mátrixok
Példák

Feladat: Egy csoportba 10 hallgató jár, akik 5 tárgyat végeznek el a félév során. A hallgatók adatai a „hallgatok.txt” szekvenciális fájlban vannak, minden sor egy hallgatónak az 5 jegye. Adjuk meg a legjobb átlaggal rendelkező hallgatót.

- ehhez el kell tárolnunk az adatokat egy 10 soros, 5 oszlopos mátrixban, azaz összesen 50 elemünk lesz
- az adatokat sorban olvassuk be a fájlból, és tároljuk el a mátrixban egymásba ágyazott ciklusokkal
- egy sorban átlagát kell vennünk a jegyeknek, amihez összegzést használunk, majd ezekből az átlagokból maximumot kell keresnünk, tehát két programozási tételre van szükségünk, amelyek szintén egymásba vannak ágyazva

Mátrixok
Példák

Specifikáció:

- bemenet: 50 egész szám egy szekvenciális fájlban (*f*)
- kimenet: a legjobb átlagú hallgató (*max*) sorszáma (*ind*)

Megoldás:

```
PROGRAM legjobb_hallgato
  VÁLTOZÓK:
    f : BEFÁJL,
    i, j, ind, max : EGÉSZ,
    sum : VALÓS
    hallgatok : EGÉSZ[10][5]
```


Mátrixok
Példák

Megoldás:

```
  MEGNYIT f : "hallgatok.txt"
  i := 0 ** külső ciklus: i index
  CIKLUS AMÍG (i < 10)
    j := 0 ** belső ciklus: j index

    CIKLUS AMÍG (j < 5)
      BE f: hallgatok[i][j]
      j := j + 1
    CIKLUS_VÉGE

    i := i + 1
  CIKLUS_VÉGE
  LEZÁR f ** már nem kell a fájl
```

Mátrixok
Példák

Megoldás:

```
max := 0 ** maximum keresés inicializálás
ind := -1
i := 0 ** külső ciklus: i index

CIKLUS AMÍG (i < 10)
  sum := 0 ** összeget lenullázzuk
  j := 0 ** belső ciklus: j index

  CIKLUS AMÍG (j < 5)
    sum := sum + hallgatok[i][j] ** összegzés
    j := j + 1
  CIKLUS_VÉGE
```


Mátrixok
Példák

Megoldás:

```
    HA (sum / 5 > max) AKKOR ** maximumkeresés
      max := sum / 5
      ind := i
    HA_VÉGE

    i := i + 1
  CIKLUS_VÉGE

  KI: "A legjobb a(z) ", ind+1, ". hallgató."
PROGRAM_VÉGE
```