Adatfeldolgozás

ELTE

Adatfeldolgozási alapok

Adatfeldolgozás: ebben a tárgyban szekvenciális fájlok feldolgozása.

Bemenet: InpSzekvFile(BeElemTípus)

Kimenet: OutSzekvFile(KiElemTípus)

Általában kell egy függvény:

f: BeElemTípus → KiElemTípus

Előfordulhat, hogy több bemeneti, illetve kimeneti fájl is van.

ELTE

Adatfeldolgozási típusfeladatok

Másolás

$$\text{BeElemTípus}^* \rightarrow \text{KiElemTípus}^*$$

$$\exists f\colon \text{BeElemTípus} \rightarrow \text{KiElemTípus}$$

Alesetei:

- Adatfelvitel (konverzió karakteresről)
- Listázás (konverzió karakteresre)
- Fájl-másolás
- Struktúraváltás (sorrend változtatás, szétbontás, összevonás)
- Elhagyás (horizontális projekció)
- Bővítés

ELTE

Adatfeldolgozási típusfeladatok

Szűrés

BeElemTípus* → KiElemTípus*

∃*T*: BeElemTípus → Logikai

vagy

∃*Tk*,*Tv*: BeElemTípus → Logikai

(csoport kezdete-e, csoport vége-e?)

Alesetei:

- Elemek szűrése (kiválogatás)
- Elemcsoportok szűrése

ELTE

Adatfeldolgozási típusfeladatok

Általánosított másolás

$$\text{BeElemTípus}^* \rightarrow \text{KiElemTípus}^*$$

$$\exists f\colon \text{BeElemTípus}^* \rightarrow \text{KiElemTípus}^*$$

Alesetei:

- Csoportba foglalás (N elem → 1 elem)
- Csoportbontás (1 elem → N elem)
- Összegfokozat (N elem → N+1 elem)

ELTE

Adatfeldolgozási típusfeladatok

Összeválogatás

$$(\text{BeElemTípus}^*)^2 \rightarrow \text{KiElemTípus}^*$$

$$\exists f\colon \text{BeElemTípus}^2 \rightarrow \text{KiElemTípus}$$

Alesetei:

- Keverés
- Egymásutánírás
- Párosítás
- Összefésülés
- Időszerűsítés

ELTE

Adatfeldolgozási típusfeladatok

Szétszedés

$$\text{BeElemTípus}^* \rightarrow (\text{KiElemTípus}^*)^2$$

$$\exists f\colon \text{BeElemTípus} \rightarrow \text{KiElemTípus}^2$$

Alesetei:

- Szétválogatás (tulajdonság alapján)
- Darabolás (sok darabra)

ELTE

Adatfeldolgozási típusfeladatok

Az alábbiak szekvenciális fájlokra nem túl hatékonyak, az „igazi” megoldáshoz más ábrázolásra van szükség.

Rendezés

Keresés

ELTE

Szekvenciális fájlok feldolgozása

A feldolgozás általános sémája:

```
Nyit(f,Inputra); Nyit(g,Outputra)
Ciklus amíg nem Vége?(f)
  Olvas(f,adat)
  újadat:=függvény(adat)
  Ír(g,újadat)
Ciklus vége
Zár(f); Zár(g)
```

Struktúra szerinti feldolgozás

Típusfinomítás: az algoritmus „felül-ről-lefelé” elvének adatokra alkalmazott párja, vagyis az adatok dekomponálása megengedett struktúrák felhasználásával.

ELTE

Struktúra szerinti feldolgozás

Megengedett struktúrák:

- Direktszorzat-képzés (pl. rekord)
- Unió (pl. alternatív rekord)
- Sokaságképzés (pl. halmaz; tömb, file stb.)

Struktúra szerinti feldolgozás elve

- Direktszorzat-képzés – szekvencia
- Unió – elágazás
- Sokaságképzés – ciklus (vagy rekurzió)

ELTE

Struktúra szerinti feldolgozás

Példa: egy személynyilvántartás...

SzemélyiÁllomány = **File**(SzemélyiAdat)

```
Nyit(f)
Ciklus amíg nem vége?(f)
  Olvas(f,R)
  Feldolgoz(R)
Ciklus vége
Zár(f)
```

Megj.: A `Feldolgoz(R)` tartalmazza a kiírást is.

ELTE

Struktúra szerinti feldolgozás

Példa: egy személynyilvántartás...

Finomítása:

SzemélyiAdat=**Rekord**(név: Szöveg
szsz: SzemélyiSzám
email: EMailCím
osztály: OsztályNév)

```
Feldolgoz(R):
  Ki: R.név
  Ki: R.szsz
  Ki: R.EMailCím
  Ki: R.Osztálynév
Eljárás vége.
```

Struktúra szerinti feldolgozás

Példa: Kiválogatás tétel

Bemenet: elemek sorozata

Elem: T tulajdonságú ∪ Nem T tulajdonságú

```
Nyit(f,Inputra); Nyit(g,Outputra)
Ciklus amíg nem Vége?(f)
  Olvas(f,adat)
  Ha T(adat) akkor Ír(g,adat)
Ciklus vége
Zár(f); Zár(g)
```

ELTE

Struktúra szerinti feldolgozás

Példa: Kiválogatás tétel

Kimenet: T tulajdonságú elemek sorozata

Bemenet: T tulajdonságú elemekre végződő részsorozatok sorozata

```
VanMég:=igaz
Ciklus amíg VanMég
  Ciklus
    Olvas(f,adat)
  amíg nem T(adat)
  Ciklus vége
  Ír(g,adat)
  Ha vége?(f) akkor VanMég:=hamis
Ciklus vége
```

ELTE

Struktúra szerinti feldolgozás

Példa: Kiválogatás tétel

Kimenet szerinti feldolgozás:

Sorozaton belül unió →

Cikluson belül elágazás

Bemenet szerinti feldolgozás:

Sorozaton belül sorozat →

Cikluson belül ciklus

Kérdés: Melyik megoldást válasszuk?

ELTE

Struktúra szerinti feldolgozás

Problémák:

- nem egyértelmű a típusfinomítás;
- a program szerkezete függ a típusfinomítástól;
- akár az input, akár az output alapján végezhető.

Melyiket válasszuk?

- a legtöbb információt hordozó finomítást…
- a leghatékonyabban megvalósítható adatszerkezetet…

ELTE

Struktúra megfeleltetés

Példa: Kiválogatás tétel

Bemenet: BeElemek sorozata

Kimenet: KiElemek sorozata

BeElem: T tulajdonságú ∪ Nem T tulajdonságú

KiElem: T tulajdonságú ∪ *üres (fiktív elem)*

A két struktúra részenként megfeleltethető egymásnak, a feldolgozás a megfeleltetés szerinti lehet.

ELTE

Struktúra megfeleltetés példa

Egy egyetem személyi adatnyilvántartásának kilistázása.

1. Megoldás: [*hierarchikus szerkezet*]

Az egyetem karokból áll.

A karok tanszékcsoportokból állnak (*egyes karokon intézetekből*).

A tanszékcsoportok tanszékekből állnak.

A tanszékeken dolgozók vannak.

A dolgozók személyes adatait ismerjük.

ELTE

Struktúra megfeleltetés példa

1. Megoldás: [*hierarchikus szerkezet*]

```
TEgyetem=Rekord(név:Szöveg,
                kAdat:File(TKar))
TKar=Rekord(név:Szöveg,
            cAdat:File(TTCs))
TTCs=Rekord(név:Szöveg,
            tAdat:File(TTSz))
TTSz=Rekord(név:Szöveg,
            dAdat:File(TDolgozó))
TDolgozó=Rekord(név:Szöveg,
     beoszt:TBeosztás,fizetés:Egész)
TBeosztás=(Tanseg,Adj,Doc,Egytan)
```

ELTE

Struktúra megfeleltetés példa

Az egyetem karokból áll.

```
TEgyetem=Rekord(név:Szöveg,
          kAdat:File(TKar))

Listáz(e):
  Nyit(g); SorÍr(g,e.név)
  Nyit(e.kAdat)
  Ciklus amíg nem Vége?(e.kAdat)
    Olvas(e.kAdat,k)
    KariLista(k,g)
  Ciklus vége
  Zár(e.kAdat); Zár(g)
Eljárás vége.
```

ELTE

Struktúra megfeleltetés példa

A karok tanszékcsoportokból állnak.

```
TKar=Rekord(név:Szöveg,
            cAdat:File(TTCs))

Karilista(k,g):
  SorÍr(g,k.név)
  Nyit(k.cAdat)
  Ciklus amíg nem Vége?(k.cAdat)
    Olvas(k.cAdat,tc)
    TanszékcsoportiLista(tc,g)
  Ciklus vége
  Zár(k.cAdat)
Eljárás vége.
```

ELTE

Struktúra megfeleltetés példa

A tanszékcsoportok tanszékekből állnak.

```
TTCs=Rekord(név:Szöveg,
            tAdat:File(TTSz))

Tanszékcsoportilista(tc,g):
  SorÍr(g,tc.név)
  Nyit(tc.tAdat)
  Ciklus amíg nem Vége?(tc.tAdat)
    Olvas(tc.tAdat,t)
    TanszékiLista(t,g)
  Ciklus vége
  Zár(tc.tAdat)
Eljárás vége.
```

ELTE

Struktúra megfeleltetés példa

A tanszékek dolgozókból állnak.

```
TTSz=Rekord(név:Szöveg,
                dAdat:File(TDolgozó))

Tanszékilista(t,g):
  SorÍr(g,t.név)
  Nyit(t.dAdat)
  Ciklus amíg nem Vége?(t.dAdat)
    Olvas(t.dAdat,d)
    DolgozóiiLista(d,g)
  Ciklus vége
  Zár(t.dAdat)
Eljárás vége.
```

Struktúra megfeleltetés példa

A dolgozó nevét, beosztását és fizetését ismerjük.

```
TDolgozó=Rekord(név:Szöveg,
  beoszt: TBeosztás,fizetés: Egész)

Dologozóilista(d,g):
  SorÍr(g,d.név)
  SorÍr(g,d.beoszt)
  SorÍr(g,d.fizetés)
Eljárás vége.
```

Struktúra megfeleltetés példa

ELTE

Problémafelvetés?

- A struktúra megfeleltetés mindig el-végezhető?
  - Bemeneti és kimeneti struktúra között?
  - Két be- vagy kimeneti struktúra között?
- Ha nem végezhető el, akkor mit vá-lasszunk?

ELTE

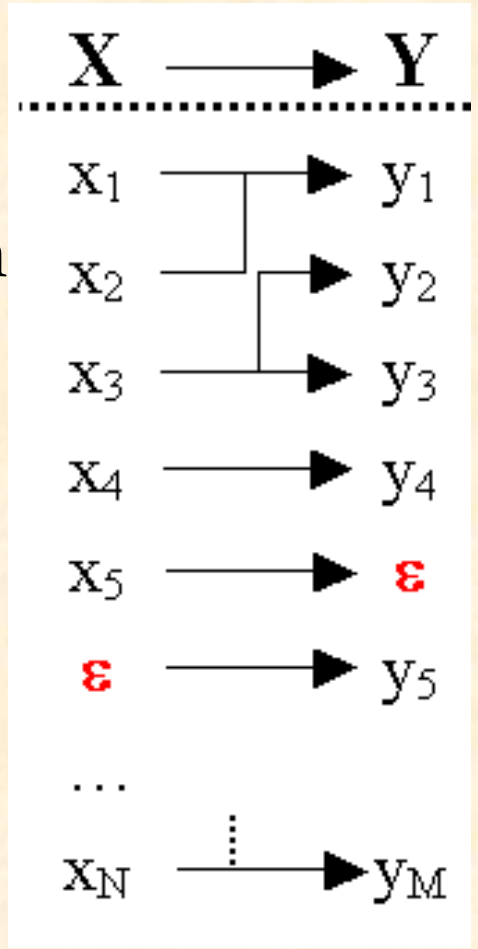
A struktúramegfelelte-tés konfliktusai

Tagolási konfliktus

A lényeg: a bemeneti és a kimeneti fájl elemei nem egy az egyben feleltethe-tők meg egymásnak.

Az elemek sorrendje azonban azonos.

Általános megoldás: virtuális típus bevezetése.



ELTE

A struktúramegfelelte-tés konfliktusai

Összefonódási konfliktus

A lényeg: Legalább 2 be-meneti sorozat elemei összekeveredtek, de ön-magukon belül jó sorrend-ben vannak.

Általános megoldás: szétválogatás.

X&Y	⟶	Z
x_1		z_1
x_2		z_2
y_1		z_3
x_3		z_4
y_2	⟶	z_5
x_4		z_6
x_5		z_7
…		…
y_N		z_M

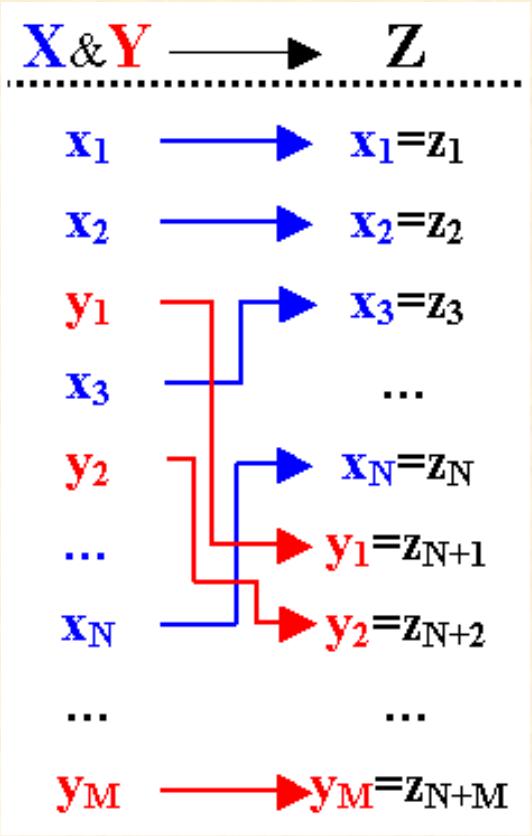
ELTE

A struktúramegfelelte-tés konfliktusai

Összefonódási konfliktus (speciális)

A lényeg: A két kevere-dő sorozat elemeit egy-más mögé kell elhelyez-ni.

Általános megoldás: szétválogatás+hozzáírás.

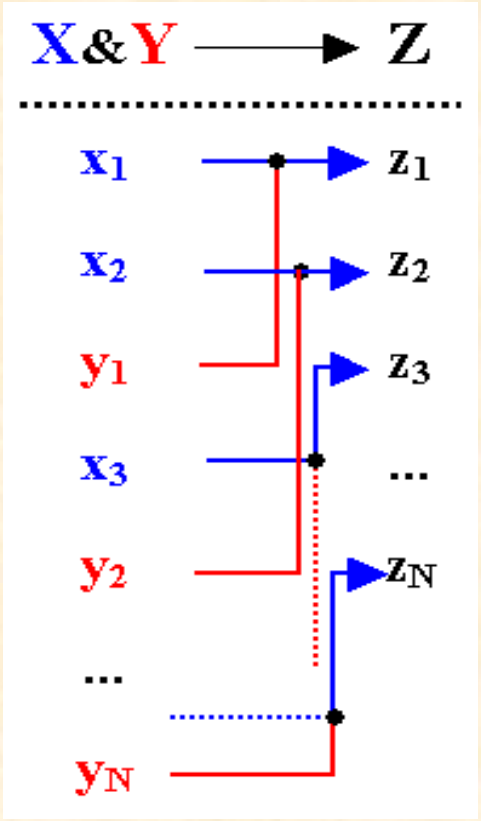


A struktúramegfelelte-tés konfliktusai

Összefonódási konfliktus (speciális)

A lényeg: A két kevere-dő sorozat elemeit páro-sítani kell egymással.

Általános megoldás: szétválogatás+ párosítás/másolás.



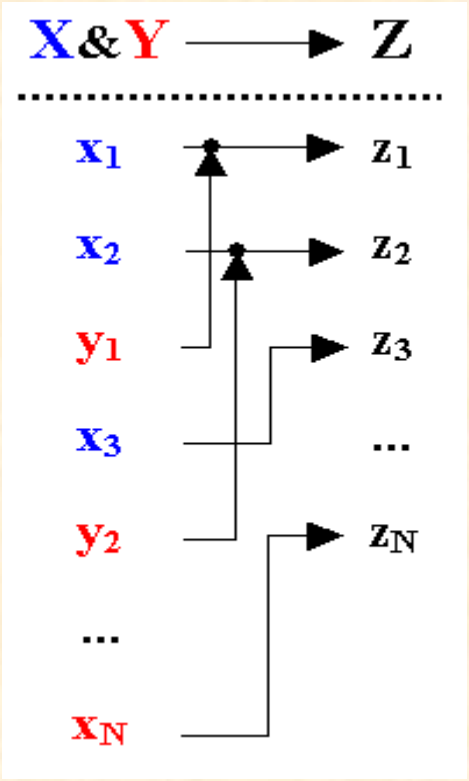
ELTE

A struktúramegfelelte-tés konfliktusai

Összefonódási konfliktus (speciális)

A lényeg: A két kevere-dő sorozat bizonyos ele-meit párosítani kell egy-mással, másokat le kell másolni.

Általános megoldás: szétválogatás+ párosítás/másolás.



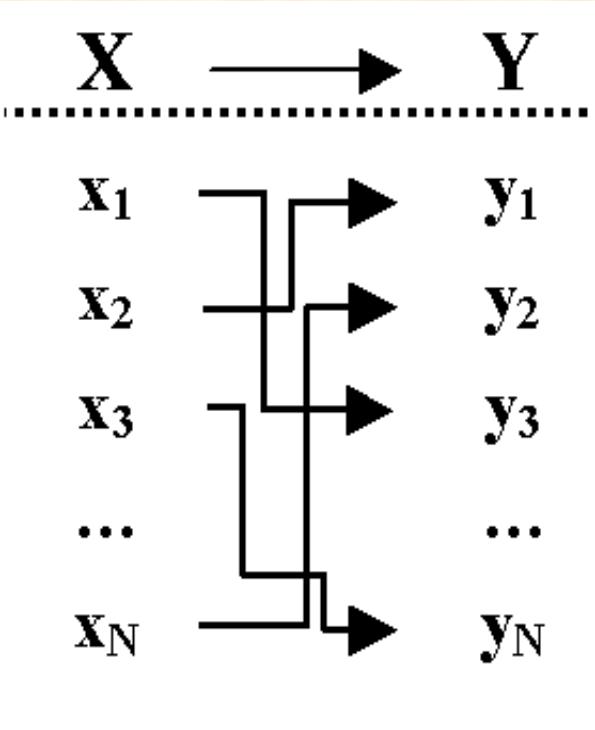
ELTE

A struktúramegfelelte-tés konfliktusai

Rendezetlenségi konfliktus

A lényeg: A bemenet és a kimenet sorrend-je nem felel meg egy-másnak.

Általános megoldás: rendezés.



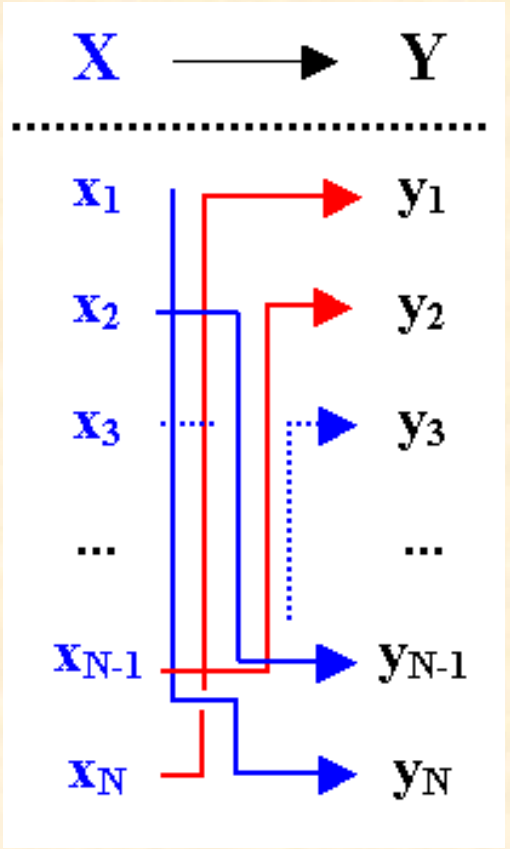
ELTE

A struktúramegfelelte-tés konfliktusai

Rendezetlenségi konfliktus (speciális)

A lényeg: A bemenet és a kimenet sorrend-je ellenkezője egymásnak.

Általános megoldás: verem használata.



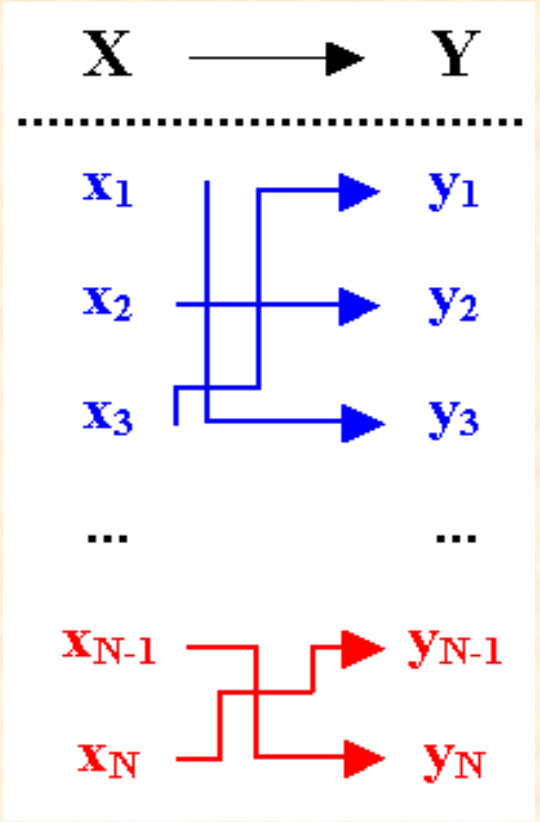
ELTE

A struktúramegfelelte-tés konfliktusai

Rendezetlenségi konfliktus (speciális)

A lényeg: A bemenet csoportokban rossz sorrendben van, de a csoportok sorrendje jó.

Általános megoldás: csoportonkénti rendezés.



ELTE

Tagolási konfliktus

Feloldás:

A be- és kimeneti típusok „összebékítése” virtuális típus(ok) segítségével.

Típusmódosítás:

- Reprezentációmódosítás
- Implementálás
  - *Virtuális elem olvasása (bemeneti oldalon)*
  - *Virtuális elem írása (kimeneti oldalon)*
  - *Sokszor még: nyitás és zárás „szertartása”, a vég érzékelése is.*

ELTE

Tagolási konfliktus:
fájl nyomtatása lapozottan

Példa: Fájl nyomtatása lapozottan.

Megoldás:

```
Bemenet=File(TElem)
Kimenet=File(TLap)
TElem=? ⇔ TLap=Rekord(
                  db: Egész
                  törzs: TSorok)
```

Konfliktus!!! Feloldása: virtuális típus a kimeneti oldalon.

Tagolási konfliktus:
fájl nyomtatása lapozottan

Virtuális típus bevezetése:

```
Bemenet=File(TElem)
Kimenet=File(TLap(TElem))
```

Új típus:

```
TLap=Rekord(db: Egész
            törzs: TSorok)
```

Műveletei:

Lapnyit, Lapraír, Lapzár

ELTE

Tagolási konfliktus:
fájl nyomtatása lapozottan

A megoldás ezek után egy klasszikus adat-feldolgozási séma:

```
Lapozás(f,g):
  Nyit(f); Lapnyit(g)
  Ciklus amíg nem vége?(f)
    Olvas(f,Elem)
    Lapraír(g,Elem)
  Ciklus vége
  Zár(f); Lapzár(g)
Eljárás vége.
```

Természetesen az új típus műveleteit még el kell készíteni!

ELTE

Tagolási konfliktus:
fájl nyomtatása lapozottan

```
Lapnyit(g):
  Nyit(g); Lap.db:=0
Eljárás vége.
Lapraír(g,Elem):
  Ha Lap.db=maxsor
    akkor Ír(g,Lap); Lap.db:=1
    különben Lap.db:=Lap.db+1
  Elágazás vége
  Lap.törzs(Lap.db):=Elem
Eljárás vége.
```

A lap kiegészíthető lenne pl. élőfejjel (ami adott szöveg), élőlábbal (ahol a lapsorszám van), …!

ELTE

Tagolási konfliktus:
fájl nyomtatása lapozottan

Záráskor az utolsó, esetleg nem teljesen kitöltött lapot is ki kell írni!

```
Lapzár(g):
  Ciklus i=Lap.db+1-től maxsor-ig
    Lap.törzs(i):=""
  Ciklus vége
  Ír(g,Lap)
  Zár(g)
Eljárás vége.
```

Megjegyzés: ha a bemenet üres volt, akkor egyetlen üres oldalt írunk a kimenetre! (Ez az eset külön is kezelhető a főprogramban.)

ELTE

Tagolási konfliktus:
fájl nyomtatása lapozottan

Virtuális típus bevezetése:

```
Bemenet=File(TLap(TElem))
Kimenet=File(TLap)
```

Új típus:

```
TLap=Rekord(db: Egész
            törzs: TSorok)
```

Műveletei:

Lapnyit, Lapolvas, Lapzár, Lapvége?

ELTE

Tagolási konfliktus:
fájl nyomtatása lapozottan

A megoldás ezek után egy klasszikus adat-feldolgozási séma:

```
Lapozás(f,g):
  Lapnyit(f); Nyit(g)
  Ciklus amíg nem Lapvége?(f)
    Lapolvas(f,Lap)
    Ír(g,Lap)
  Ciklus vége
  Lapzár(f); Zár(g)
Eljárás vége.
```

Természetesen az új típus műveleteit még el kell készíteni!

ELTE

Tagolási konfliktus:
fájl nyomtatása lapozottan

```
Lapolvas(g,Lap):
  i:=0
  Ciklus amíg nem vége?(f) és
                              i<maxsor
    i:=i+1; Olvas(f,Lap.törzs(i))
  Ciklus vége
  Ciklus j=i+1-től maxsor-ig
    Lap.törzs(j):=""
  Ciklus vége
Eljárás vége.
```

Az utolsó lapnál a maradék sorokat is kitöltjük.

ELTE

Tagolási konfliktus:
fájl nyomtatása lapozottan

A `Lapnyit`-nak akkor lehet szerepe, ha pl. oldalszámozás vagy élőfej írás is kell, ekkor itt állítjuk be a kezdősorszámot, az élőfej és élő-láb szöveget, …

A `Lapvége?` és a `Lapzár` nem tartalmaz újdonságot, csak a teljesség kedvéért definiáltuk.

Megjegyzés: Ha a kimenet képernyő vagy nyomtató, akkor az `Ír` eljárást újra kell fogalmazni, mert ezekre csak soronként írhatunk, a képernyőn lapozni kell, …

ELTE

Összefonódási konfliktus

A probléma: Két vagy több sorozat keverése úgy, hogy a részsorozatok elemei a feldolgozásnak megfelelő sorrendben vannak. (A feldolgozáshoz a részsorozatok azonos sorszámú elemei alkotnak „egységet”.)

Feloldás: Szétválogatás, majd „párhuzamos” feldolgozás.

Speciális feladatok:

Szétválogatás és egymásutánírás

Szétválogatás és párosítás

Szétválogatás és időszerűsítés

Összefonódási konfliktus

Probléma: A szétválogatás eredményeképpen két sorozat keletkezik, amit az eredmény előállításához újra végig kell olvasni.

Az igazi megoldásban az olvasások számát minimalizáljuk.

ELTE

Összefonódási konflik-tus: egymásutánírás

Példa:

Egy egyirányú utcában két jelzőlámpánál figyelik az áthaladó autókat: *mikor* haladt át a *lámpá*nál. Az adatok megfigyelési *idő szerinti* sorrendben érkeznek. N db autót figyeltek meg. Csoportosítsuk az adatokat, hogy az *első* lámpánál történt megfigyelések – időbeli sorrendjüket megtartva – előzzék meg a *második* lámpánál megfigyelt adatokat!
(Az utcában előzés nincs.)

ELTE

Összefonódási konflik-tus: egymásutánírás

Ötlet: Az 1-es lámpa adatai azonnal írhatók az eredménybe, a 2-es lámpa adatait egy átmeneti sorban tároljuk, s a végén írjuk az eredménybe. A sor – ha a mérete indokolja – persze lehet háttértáron is.

A bemenet és a kimenet is forgalom típusú adatok fájlja, és a sorban is ilyeneket tárolunk.

```
Forgalom=Rekord(lámpa: {1,2},
                idő: Egész)
```

ELTE

Összefonódási konflik-tus: egymásutánírás

```
Forgalom:
  Nyit(f); Nyit(g); Üres(S)
  Ciklus amíg nem vége?(f)
    Olvas(f,adat)
    Ha adat.lámpa=1 akkor Ír(g,adat)
      különben Sorba(S,adat)
  Ciklus vége
  Ciklus amíg nem üres?(S)
    Sorból(S,adat); Ír(g,adat)
  Ciklus vége
  Zár(f); Zár(g)
Eljárás vége.
```

ELTE

Összefonódási konflik-tus: párosítás

Példa:

Egy vasútvonal két állomásán egy nap feljegyeztük, az összes egyik irányba áthaladó vonat indulási, illetve érkezési idejét (idő szerinti sorrendben). Adjuk meg az áthaladt vonatok menetidőit a két állomás között! (Nincs előzés.)

ELTE

Összefonódási konflik-tus: párosítás

Ötlet: A korábban érkező adatot egy sorban várakoztatjuk addig, amíg a párja meg nem érkezik.

A bemenet és a sor is állomás típusú adatok fájlja, a kimeneten pedig időtartamokat tárolunk.

```
Állomás=Rekord(áll: {1,2},
               idő: Egész)
```

Tehát az idő típust kétféleképpen értelmezzük: a bemeneten és a sorban időpontot, a kimeneten pedig időtartamot jelent.

ELTE

Összefonódási konflik-tus: párosítás

```
Forgalom:
  Nyit(f); Nyit(g); Üres(S)
  Ciklus amíg nem vége?(f)
    Olvas(f,adat)
    Ha adat.áll=1
        akkor  Sorba(S,adat.idő)
      különben Sorból(S,előző)
               Ír(g,adat.idő-előző)
  Ciklus vége
  Zár(f); Zár(g)
Eljárás vége.
```

ELTE

Összefonódási konflik-tus: időszerűsítés

Példa: Egy számítógéphálózaton az üzenete-ket csomagokra bontva küldik. Minden cso-mag tartalmaz (a csomagsorszám mellett) egy ellenőrző összeget. Ha a vevő valamelyiket hibásnak találja, akkor annak megismétlését kéri a küldőtől. A csomagok jó sorrendben érkeznek (a javítók is egymáshoz képest). A vevő feladata az üzenetek csomagjainak javí-tások utáni helyes sorrendbe állítása.

ELTE

Összefonódási konflik-tus: időszerűsítés

Ötlet: A javítandó adatot és a mögötte jövőket egy sorban várakoztatjuk addig, amíg a párja meg nem érkezik. Ha megjött akkor a párját és a miatta várakozókat kiírhatjuk az eredmény-be.

A bemenet, a kimenet és a sor is üzenet típusú adatok fájlja.

```
Üzenet=Rekord(sorsz: Egész,
              tart: Szöveg,
              ell: TEll)
```

Összefonódási konflik-tus: időszerűsítés

ELTE

```
Forgalom:
  Nyit(f); Nyit(g); Üres(S)
  Ciklus amíg nem vége?(f)
    Olvas(f,adat)
    Ha adat.ell=igaz
        akkor  JóÜzenet(S,adat)
      különben RosszÜzenet(S,adat)
  Ciklus vége
  Zár(f); Zár(g)
Eljárás vége.

RosszÜzenet(S,adat)
 ha adat.sorsz∉S akkor Sorba(S,adat)
  Ismétléskérés(adat.sorsz)
Eljárás vége.
```

Összefonódási konflik-tus: időszerűsítés

```
Jóüzenet(S,adat):
  Ha üres?(S) akkor Ír(g,adat)
  különben
       ha adat.sorsz=első(S).sorsz
           akkor Sorból(S,x);
                  Ír(g,adat);
                 SorÜrítés(S)
           különben Sorba(S,adat)
Eljárás vége.
SorÜrítés(S):
  Ciklus amíg nem üres?(S) és
                    első(S).ell=igaz
    Sorból(S,x); Ír(g,x)
  Ciklus vége
Eljárás vége.
```

ELTE

Rendezetlenségi konfliktus: fordítás

Példa: A bemenet pontosan ellenkező sorrendbe rendezett, mint a kimenet.

```
Megfordít:
  Nyit(f); Nyit(g); Üres(V)
  Ciklus amíg nem vége?(f)
    Olvas(f,adat); Verembe(V,adat)
  Ciklus vége
  Ciklus amíg nem üres?(V)
    Veremből(V.adat); Ír(g,adat)
  Ciklus vége
  Zár(f); Zár(g)
Eljárás vége.
```


Vége