Szekvenciális fájlok kezelése

Gregorics Tibor

gt@inf.elte.hu

http://people.inf.elte.hu/gt/oep

Szekvenciális inputfájl

- A szekvenciális inputfájl (infile(E)) elnevezést olyan gyűjteményre használjuk, amelynek elemeit (amelyek mind egy adott E halmazhoz tartoznak) sorban egymás után szeretnénk kinyerni.
- Egy ilyen gyűjteményre olyan sorozatként tekintünk (a reprezentációja többnyire tényleg egy sorozat), amelyre egyetlen műveletet engedünk csak meg: a sorozat első elemének „kiolvasását” (leszakítását).
- Az olvasást az st, e, x : read szimbólummal jelöljük (ami egy értékadásra utal: st, e, x:=read(x)), ahol x:infile(E), e:E, st:Status = {abnorm, norm}, és amely az alábbi tevékenységet fedi:

if x = <> **then** st := abnorm

else st, e, x := norm, x_1, <x_2 , … , $x_{|x|}$>

Szekvenciális outputfájl

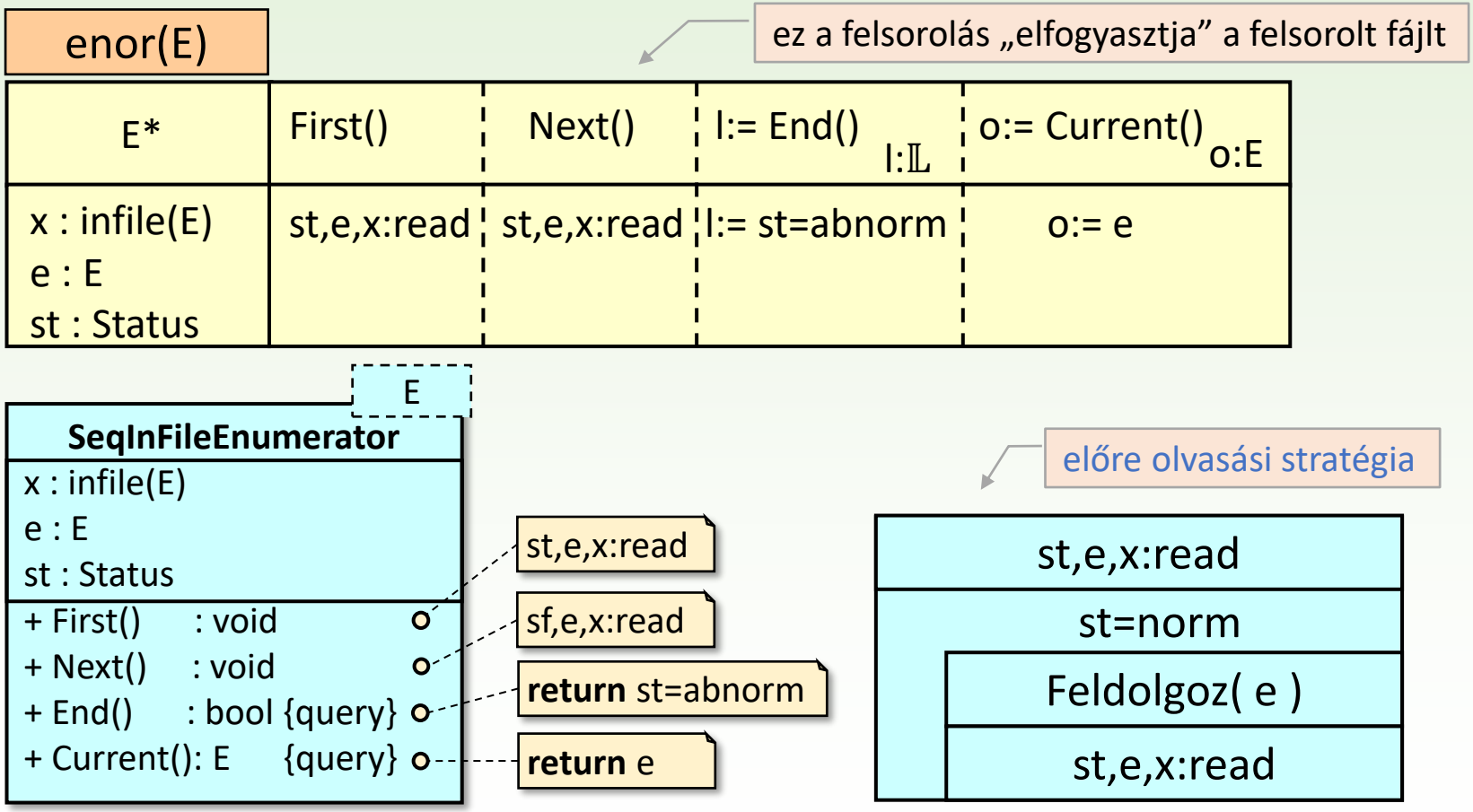
- A szekvenciális outputfájl (outfile(E)) elnevezést olyan gyűjteményre használjuk, amelybe sorban egymás után helyezhetünk el egy adott E halmazhoz tartozó elemeket.
- Egy ilyen gyűjteményre olyan sorozatként tekintünk (a reprezentációja többnyire tényleg egy sorozat), amelyre két műveletet vezetünk be:
  1. kezdetben az üres sorozat létrehozását
  2. a sorozat végéhez történő hozzáírást (hozzáfűzést)
- Az inicializálást az x := <> értékadás végzi, ahol x:outfile(E), és <> az üres sorozat.
- Az írást az x : write(e) szimbólummal jelöljük (ami az x := write(x, e) értékadásra utal), ahol x:outfile(E), e:E, és amely az alábbi tevékenységet fedi: $x := x \oplus \langle e \rangle$

Fájlkezeléses feladatok

- A gyakorlatban sokszor találkozhatunk olyan feladatokkal, amelyekben szekvenciális inputfájl (esetleg több fájl) alapján kell elállítani szekvenciális outputfájlt (esetleg fájlokat):
  - másolás
  - kiválogatás
  - szétválogatás
  - összefuttatás
- Ezeknél a feladatoknál a szekvenciális inputfájlt elemenként dolgozzuk fel, azaz az inputfájl elemeit annak olvasó műveletére támaszkodva felsoroljuk, és a megoldást az összegzés algoritmus mintára vezetjük vissza.

Szekvenciális inputfájl felsorolója

E-beli értékeket tartalmazó szekvenciális inputfájl elemeinek felsorolása



Összegzés fájlkezeléshez

Általános összegzés

A = (t:enor(E), s:H)

Ef = ($t = t_0$)

Uf = ($s = \Sigma_{e \in t_0} f(e)$)

f :E → H
+:H×H→H
0∈H neutr. elem

Fájlfeldolgozás

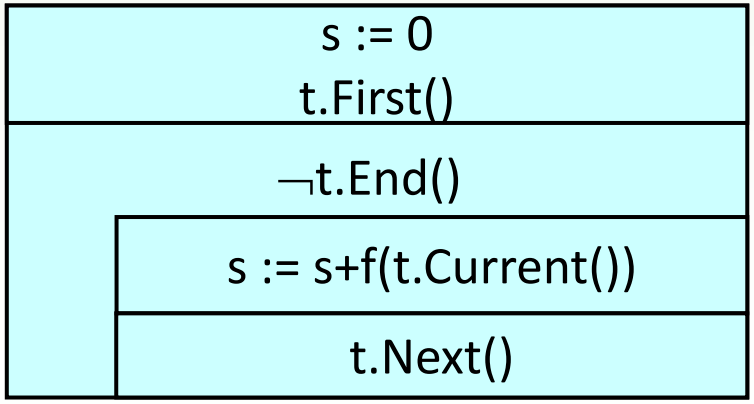
A = (x:infile(E), y:outfile(F))

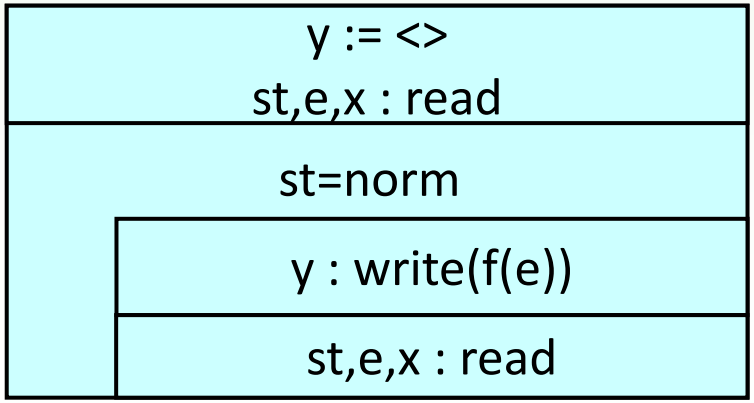
Ef = ($x = x_0$)

Uf = ($y = \bigoplus_{e \in x_0} f(e)$)

f :E → F*
⊕:F*×F*→F*
<>∈F* neutr. elem

Összegzés:
t:enor(E) ~ x:infile(E) felsorolása a st,e,x : read művelettel
H,+,0 ~ F*, ⊕, <>





C# nyelvi elemek

- Szöveges állomány hátterű szekvenciális inputfájlból egy egyedi TextFileStream típusú objektum segítségével olvashatjuk be az adatokat. Ehhez szükség van a TextFileStream.dll és a using TextFile hivatkozásra.
- Az st,e,x:read műveletet az e típusától függően eltérő utasítások helyettesítik, amelyek igaz értéket adnak vissza, ha st=norm
  - karakterenkénti olvasásnál: x.ReadChar(out char e)
  - lexikális egységenkénti olvasásnál: x.ReadInt(out int e)
    x.ReadDouble(out double e)
    x.ReadString(out string e)
    x.ReadLine(out string e)

- Szöveges állomány hátterű szekvenciális outputfájlba egy szabványos StreamWriter objektum segítségével írhatunk adatokat.
  Ehhez szükség van a using System.IO hivatkozásra.
- Az y:write(e) műveletet az y.Write(e) utasítás helyettesíti, de teljes sorok kiírásához használhatjuk az y.WriteLine(e) utasítást is.

1. rész
Karakterenkénti olvasás

Gregorics Tibor

gt@inf.elte.hu

http://people.inf.elte.hu/gt/oep

1.Feladat

Alakítsunk át egy ékezeteket tartalmazó szöveget (szöveges állományt) ékezet nélkülire (az eredmény egy másik szöveges állományba kerüljön)!

A = (x:infile($\mathbb{K}$) , y:outfile($\mathbb{K}$))

Ef = ($x = x_0$)

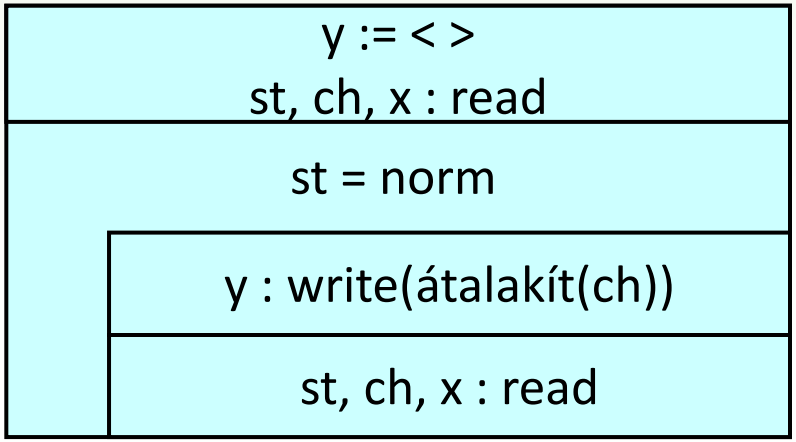
Uf = (y = $\bigoplus_{ch \in x_0}$ <átalakít(ch)>)

ahol átalakít : $\mathbb{K} \to \mathbb{K}$ és átalakít(ch) = …

másolás-átalakítás

Összegzés:

t:enor(E)	~	x:infile($\mathbb{K}$) st,ch,x : read
f(e)	~	<átalakít(ch)>
H,+,0	~	$\mathbb{K}^*$, $\oplus$, <>

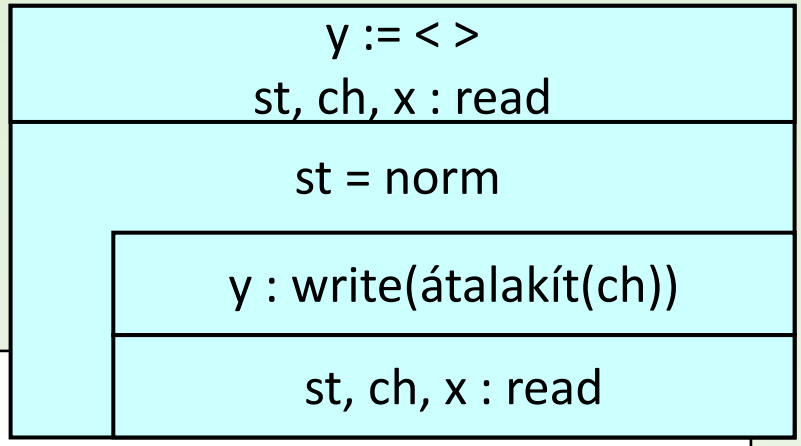


Szürkedoboz tesztelés

- Az összegzés teszteléséhez vizsgálni kell
  - a felsorolót
    - felsorolás hossza szerint: 0, 1, 2, illetve több elem felsorolása
    - felsorolás eleje, vége szerint: két eltérő betű felsorolásával már ellenőrizhető
  - a felsoroló skálázása most kevésbé fontos, hiszen az outputfájlt hossza meg fog egyezni az inputfájléval
- Ezeken kívül ellenőrizni kell a konverziót.

felsoroló szerint	hosszra:	x = <>, <a>, <ab>, <…>	→	y = <>, <a>, <ab>, <…>
	elejére:	x = <ab>, <áé>	→	y = <ab>, <ae>
	végére:	x = <ab>, <áé>	→	y = <ab>, <ae>
átalakítás szerint	ék-es mgh:	x = <ááíöőúüű>	→	y = <aeioouuu>
	ék-tlen mgh:	x = <aeioouuu>	→	y = <aeioouuu>
	msh:	x = <bsmnz>	→	y = <bsmnz>

C# program



Mire való ez a using?

TextFileReader.dll
using TextFile

```
static void Main()
{
  TextFileReader reader = new ("input.txt");
  using StreamWriter writer = new StreamWriter("output.txt");

  while(reader.ReadChar(out char ch))
  {
    writer.WriteChar(Transform(ch));
  }
}
```

using System.IO

```
StreamWriter writer = new ("output.txt");
...
writer.WriteChar(Transform(ch));
...
writer.Close();
```

Close() nélkül benn ragadhatnak a kiírandó elemek a pufferben

```
using ( StreamWriter writer = new ("output.txt") )
{
   ...
   writer.WriteChar(Transform(ch));
   ...
}
```

using hatókörének végén rögtön megszűnik a writer objektum, és ekkor lefut a Close()

Karakterek átalakítása

```
static char Transform(char ch)
{
   char new_ch;
   switch (ch) {
      case 'á'    :                          new_ch = 'a'; break;
      case 'é'    :                          new_ch = 'e'; break;
      case 'í'    :                          new_ch = 'i'; break;
      case 'ó'    : case 'ö' : case 'õ' :    new_ch = 'o'; break;
      case 'ú'    : case 'ü' : case 'û' :    new_ch = 'u'; break;
      case 'Á'    :                          new_ch = 'A'; break;
      case 'É'    :                          new_ch = 'E'; break;
      case 'Í'    :                          new_ch = 'I'; break;
      case 'Ó'    : case 'Ö' : case 'Õ' :    new_ch = 'O'; break;
      case 'Ú'    : case 'Ü' : case 'Û' :    new_ch = 'U'; break;
      default     :                          new_ch = ch;  break;
   }
   return new_ch;
}
```

2. rész
Rekordonkénti olvasás

Gregorics Tibor

gt@inf.elte.hu

http://people.inf.elte.hu/gt/oep

2.Feladat (algoritmus minták szimultán alkalmazása)

Egy sportoló adott időközönként feljegyezte a pulzusszámát, és ezeket egy szekvenciális inputfájlban időpont-pulzusszám (sztring-természetes szám) párok sorozataként rögzítette. Gyűjtsük ki a legalább 100-as pulzusszámú méréseket, adjuk meg a mérések közt talált legmagasabb pulzusszámot, és mondjuk meg, hogy volt-e a pulzusszám 60-nál kisebb.

A = (x:infile(Mérés) , y:outfile(Mérés), max:$\mathbb{N}$, volt:$\mathbb{L}$)

Mérés = rec(idő:$\mathbb{S}$, pulzus:$\mathbb{N}$)

$Ef = (x = x_0 \wedge |x|>0)$

$Uf = (y = \bigoplus_{\substack{e \in x_0 \\ e.pulzus \geq 100}} <e> \wedge (max, .) = MAX_{e \in x_0}\ e.pulzus \wedge$

$\wedge\ volt = SEARCH_{e \in x_0}\ e.pulzus<60)$

$|x_0|>0$ hiányában felt. max. ker. kellene:
$(l,max, .) = MAX_{e \in x_0}\ e.pulzus$
igaz

<u>Összegzés</u>:
t:enor(E) ~ x:infile(Mérés)
st,e,x : read
f(e) ~ <e> ha e.pulzus≥100
H,+,0 ~ Mérés*, ⊕, <>

<u>Maximum kiválasztás</u>:
t:enor(E) ~ x:infile(Mérés)
st,e,x : read
f(e) ~ e.pulzus

<u>Lineáris keresés</u>:
t:enor(E) ~ x:infile(Mérés)
st,e,x : read
felt(e) ~ e.pulzus<60

Szimultán feldolgozás közös felsorolón

A részfeladatokat megoldó programokat össze kell vonni egy közös ciklusba, hiszen mind ugyanarra a meg nem ismételhető felsorolásra épülnek. Ehhez az összevonandó ciklusoknak szinkronban kell lenniük.

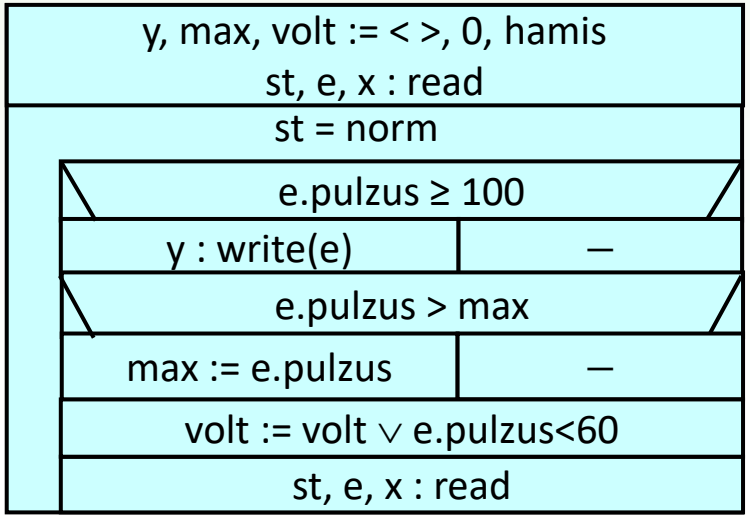
Szinkron működő mintákat használjunk

- max. kiv. helyett felt. max. ker., vagy összegzés, mivel itt a vizsgált értékeknél van kisebb

f(e) ~ e.pulzus
H,+,0 ~ $\mathbb{N}$, $\dot{+}$, 0
ahol a$\dot{+}$b = a ha a≥b, b kül

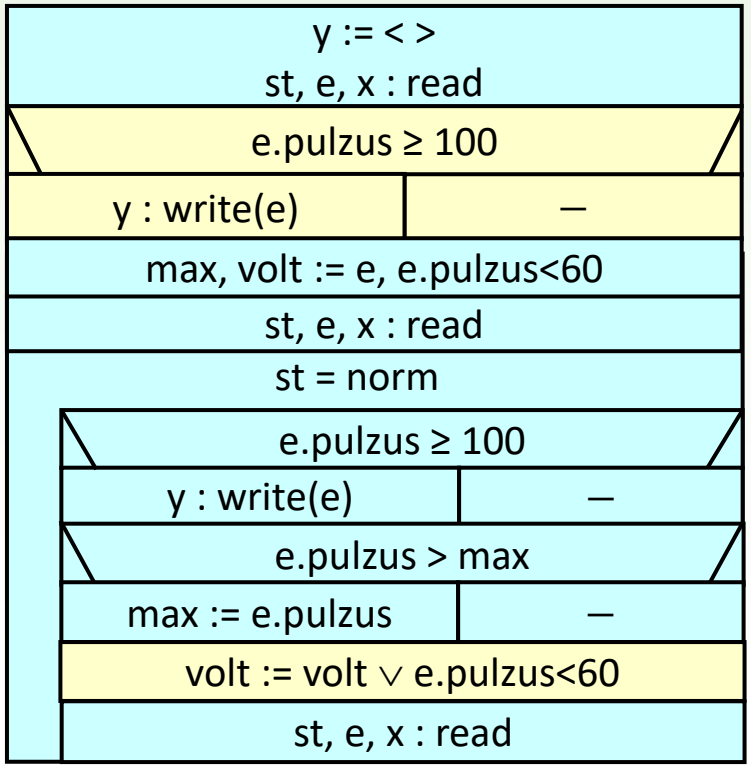
- lineáris keresés (eldöntés) helyett összegzés

f(e) ~ e.pulzus < 60
H,+,0 ~ $\mathbb{L}$, $\vee$, hamis



Programátalakítással szinkronizálunk

- a felsorolás első elemét mindhárom részprogram a ciklus előtt dolgozza fel,
- a lineáris keresés ne teminálјon, ha volt találat



Szürke doboz tesztelés

- Vizsgálni kell
  - a felsorolót
    - felsorolás hossza: 0, 1, 2, több hosszú csupa 100 feletti mérés
    - felsorolás eleje: 100 feletti ill. 100 alatti mérés elől
    - felsorolás vége: 100 feletti ill. 100 alatti mérés hátul
  - az eredeti algoritmus minták szerint
    - összegzés: skálázás most sem érdekes
    - maximum kiválasztás: első, középső, vagy utolsó elem a maximum, több azonos maximum van
    - lineáris keresés: nincs keresett mérés, a keresett mérés az első, középső, vagy utolsó elem
  - a kiválogatás és a keresés feltételében: kisebb, nagyobb, egyenlő

Szekvenciális inputfile

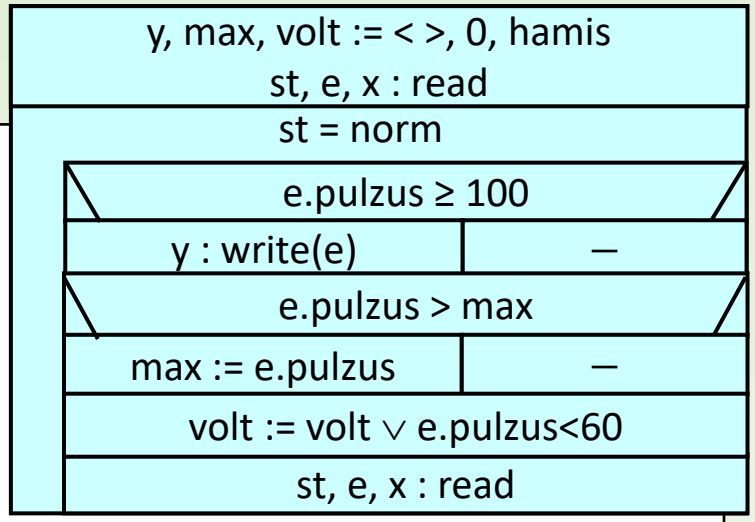
```
class InFile
{
  public struct Measurement
  {
    public string time;
    public int pulse;
  }

  private readonly TextFile.TextFileReader reader;

  public InFile(string fname)
  {
    reader = new TextFile.TextFileReader(fname);
  }

  public bool Read(out Measurement dx)
  {
    reader.ReadString(out time);
    return reader.ReadInt(out pulse);
  }
}
```

közös névtérben
a class Program -mal

a `struct` érték típus
lehetne `class` is, de akkor new kell a létrehozáshoz

st, e, x : read megvalósítása
az x objektum metódusaként

Kódolás




```
class Program
{
  static void Main()
  {
    try
    {
      InFile x = new ("input.txt");
      Console.WriteLine("Selected measurements:");

      int max = 0;
      bool low = false;
      while (x.Read(out InpFile.Measurement e))
      {
        if (e.pulse >= 100) Console.WriteLine($"time: {e.time}, pulse: {e.pulse}");
        if (e.pulse > max) max = e.pulse;
        low = low || e.pulse < 60;
      }
      string ans = low ? "" : "not ";
      Console.WriteLine($"maxima: {max}, there is {ans}a low pulse");
    }
    catch (System.IO.FileNotFoundException)
    {
      Console.WriteLine("Could not open the textfile");
    }
  }
}
```

st, e, x : read az x objektum metódusa

3.Feladat (algoritmus minták egymás után egy felsorolón)

Egy sportoló adott időközönként feljegyezte a pulzusszámát, és ezeket egy szekvenciális inputfájlban időpont-pulzusszám (sztring-természetes szám) párok sorozataként rögzítette. A fájl időpont szerint növekedően rendezett. Hányszor mért 100-nál alacsonyabb pulzusszámot azt megelőzően, hogy először elérte a pulzusa a legalább 100-at, és mekkora volt a legalacsonyabb pulzusszáma ezt követően? Legalább egyszer biztosan mért 100-as pulzusszámot, és azt követően is voltak még mérések.

a felsorolás ezen feltétel fennállásáig tart

A = (x:infile(Mérés) , c:$\mathbb{N}$, min:$\mathbb{N}$) Mérés=rec(idő:$\mathbb{S}$, pulzus:$\mathbb{N}$)

Ef = ($x = x_0 \wedge x \nearrow_{\text{idő}} \wedge \exists i \in [1..|x|-1] : x_i.\text{pulzus} \geq 100$)

folytatja x felsorolását

Uf = ((c, (st',e',x')) = $\sum_{e \in x_0}^{e.\text{pulzus}<100} 1 \wedge \text{min} = \text{MIN}_{e \in x'}\ (e.\text{pulzus})$)

lehet (. , . , x')-ot írni:

összegzés másodlagos outputja:
st, e, x változók aktuális értékei
st'=norm és e'.pulzus≥100

Összegzés:

t:enor(E) ~ x:infile(Mérés), st,e,x:read
amíg: e.pulzus<100

f(e) ~ 1

H, +, 0 ~ $\mathbb{N}$, +, 0

Minimum kiválasztás:

t:enor(E) ~ x:infile(Mérés), st,e,x:read
felsorolás folytatása

f(e) ~ e.pulzus

H, > ~ $\mathbb{N}$, <

Algoritmus

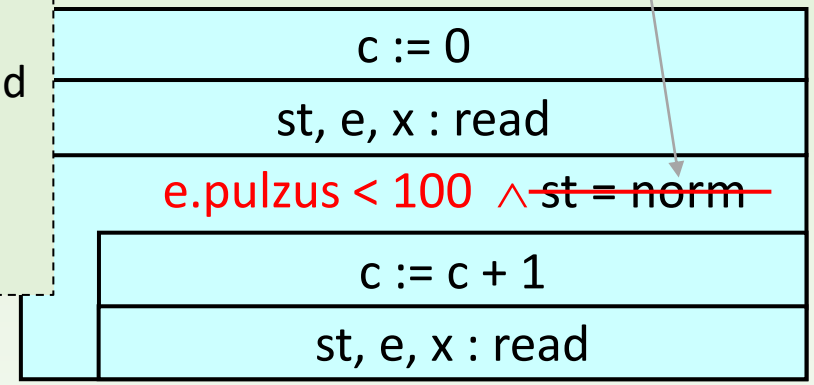
előfeltétel miatt

Összegzés:

t:enor(E) ~ x:infile(Mérés), st,e,x:read
amíg: e.pulzus<100

f(e) ~ 1

H, +, 0 ~ $\mathbb{N}$, +, 0



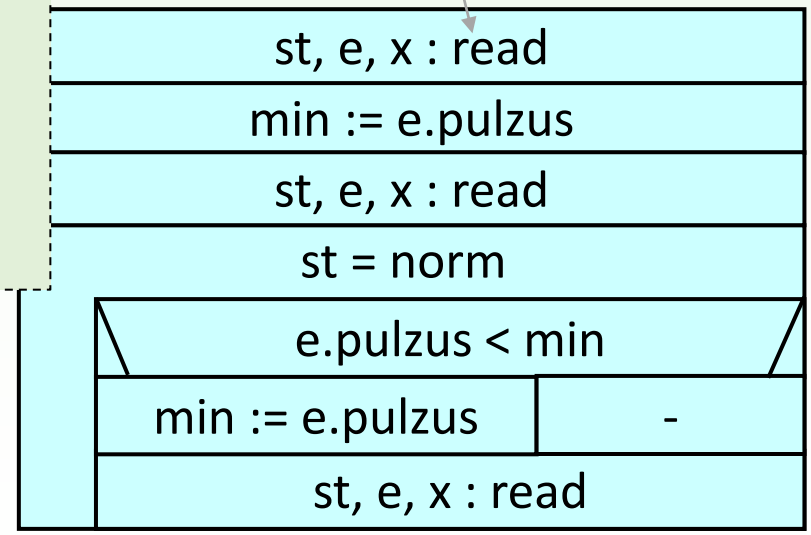
Itt First() helyett Next() kell, de szekvenciális inputfájlnál ez így is, úgy is a read lesz

Minimum kiválasztás:

t:enor(E) ~ x:infile(Mérés), st,e,x:read
felsorolás folytatása

f(e) ~ e.pulzus

H, > ~ $\mathbb{N}$, <



Kitérő

Hogyan módosul a megoldás második része, ha a minimum kiválasztásnak figyelembe kellene vennie az első legalább 100-as pulzusszámot is.

A = (x:infile(Mérés) , c:$\mathbb{N}$, min:$\mathbb{N}$) Mérés=rec(idő:$\mathbb{S}$, pulzus:$\mathbb{N}$)

Ef = ($x = x_0 \wedge x \nearrow_{\text{idő}} \wedge \exists i \in [1..|x|] : x_i.\text{pulzus} \geq 100$)

Uf = ((c, (. , e', x')) = $\sum_{e \in x_0}^{e.\text{pulzus}<100} 1 \wedge \text{min} = \text{MIN}_{e \in <e'> \oplus x'} (e.\text{pulzus})$)

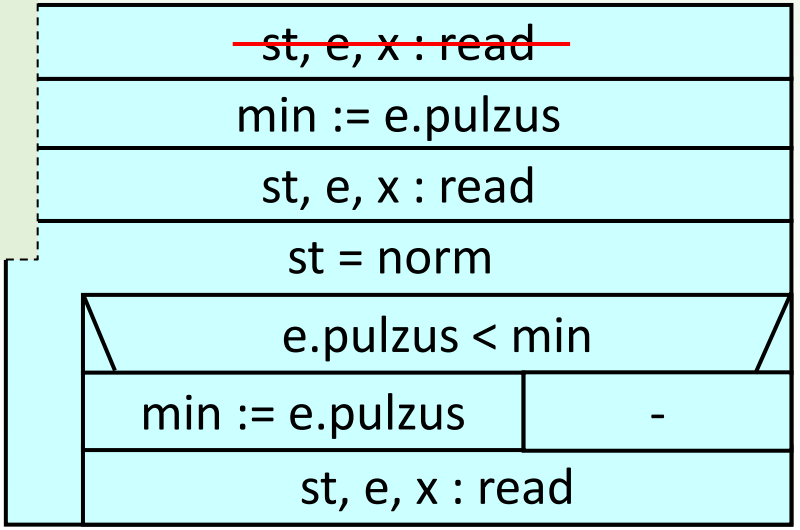
figyelembe vesszük a már kiolvasott e' értéket is

Maxinmum kiválasztás:

t:enor(E) ~ x:infile(Mérés), st,e,x:read
előre olvasás nélkül

f(e) ~ e.pulzus

H, > ~ $\mathbb{N}$, <



Szürke doboz tesztelés

- Tesztelni kell külön-külön a felsoroló első szakaszán működő összegzést, és a hátsó szakaszán működő minimum kiválasztást.
  - Olyan inputfájlok kellenek, ahol a fájl első szakasza lehet 0, 1, 2 hosszú, vagy hosszabb, hátsó szakasza 1, 2 hosszú, vagy hosszabb.
  - Vizsgálni kell, hogy mindkét szakasz mindkét végét figyelembe veszi-e a számítás: az összegzésnél ehhez elég ha a fájl első két mérése 100-nál kisebb pulzusszámú, a minimum kiválasztásnál kell olyan input, amikor a hátsó szakasz első eleme tartalmazza a legkisebb pulzus számot, és olyan is, amikor az utolsó eleme.
  - A minimum kiválasztás tesztelésénél vizsgálni kell azt is, amikor a hátsó szakasz közepén van a legkisebb pulzus számú elem, illetve amikor több egyformán legkisebb pulzus számú elem is van.
- Végül érdemes az e.pulzus<100 feltételt is tesztelni 99, 100, 101 pulzus számokkal.

Kódolás

```
static void Main()
{
  try
  {
    InFile x = new ("input.txt");
    InFile.Measurement e;

    int c = 0;
    while (x.Read(out e) && e.pulse < 100) ++c;

    x.Read(out e);
    int min = e.pulse;
    while (x.Read(out e))
    {
      if (e.pulse < min) min = e.pulse;
    }
    Console.WriteLine($"count: {c}, min: {min}");
  }
  catch(System.IO.FileNotFoundException)
  {
    Console.WriteLine("Could not open the textfile");
  }
}
```

Infile osztályban (ami névtérként is funkcionál) definiált

3. rész
Soronkénti olvasás

Gregorics Tibor

gt@inf.elte.hu

http://people.inf.elte.hu/gt/oep

4.Feladat (egymásba ágyazott algoritmus minták)

Több sportolónak mértük edzés közben a pulzusát, és a mérési adatokat soronként rögzítettük egy szöveges állományban.

Minden sor egy sportoló (több sztringből álló) nevével kezdődik, amelyet a mérési adatok követnek. Egy mérési adat egy időpontból (mm:ss formájú sztring), és egy pulzusszámból (természetes szám) áll. A sor adatait elválasztó jelek (szóközök, tabulátorjelek) választják el egymástól. Feltehetjük, hogy a mérések időpont szerint növekedően rendezettek.

Gyűjtsük ki azokat a sportolókat, akiknek pulzusszáma az első legalább 100-as mérés után 100 alá esett vissza.

```
Gipsz Jakab Elemér          00:01 67 01:15 89   02:55 102 04:03 108
Szer Elek                   00:01 72 00:55 102 03:15 110
Jose Fernando Llano del Colona  00:01 67 02:30 100 04:55 95   06:03 105
```

Elemzés

fizikai elhelyezkedést szem előtt tartó

A = (x : infile(Sor) , y : outfile(Sor)) Sor = $\mathbb{S}$*,
Ef = (x = $x_0 \wedge \forall e \in x$: e olyan, hogy …)
Uf = (…)

bemenetet szem előtt tartó

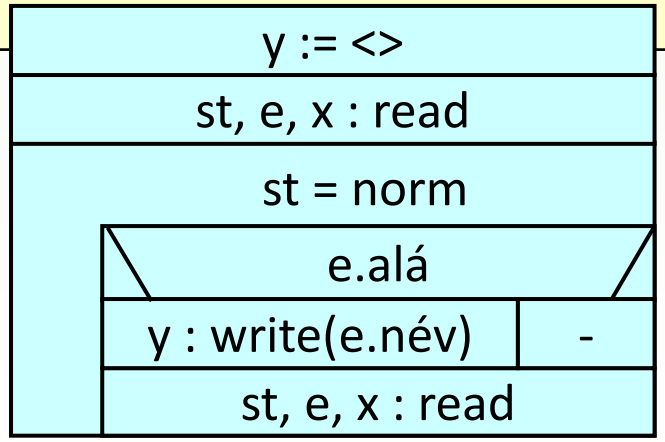
A = (x : infile(Sportoló) , y : outfile($\mathbb{S}$))
Sportoló = rec(név:$\mathbb{S}$, mérések:Mérés*) Mérés=rec(idő:$\mathbb{S}$, pulzus:$\mathbb{N}$)
Ef = (x = $x_0 \wedge \forall e \in x$: e.mérések $\nearrow_{\text{idő}}$)
Uf = (y = $\bigoplus_{e \in x_0}$ f(e)) f = …

kimenetet szem előtt tartó

A = (x : infile(Sportoló) , y : outfile($\mathbb{S}$))
Sportoló = rec(név:$\mathbb{S}$, alá:$\mathbb{L}$)
Ef = (x = x_0)
Uf = (y = $\bigoplus_{e \in x_0}$ <e.név>)
e.alá

Összegzés (kiválogatás):

t:enor(E)	~	x:infile(Sportoló)
e∈t	~	st,e,x :read
f(e)	~	ha e.alá akkor <e.név>
s	~	y
H,+,0	~	$\mathbb{S}$*, $\oplus$, <>

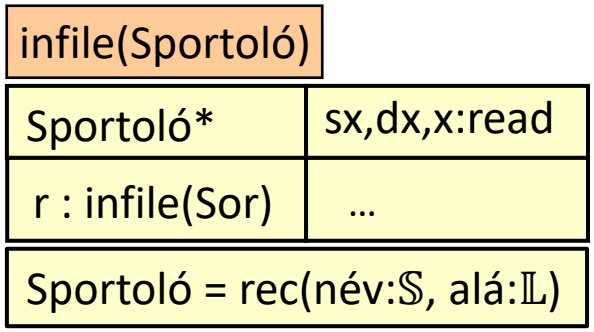


Kódolás

```
static void Main()
{
  try
  {
    InputFile x = new ("input.txt");
    while(x.Read(out InputFile.Athlete e))
    {
      if (e.below)
      {
         Console.WriteLine(e.name);
      }
    }
  }
  catch(System.IO.FileNotFoundException)
  {
    Console.WriteLine("Could not open the textfile");
  }
  catch(InputFile.EmptyRowException)
  {
    Console.WriteLine("Could not open the textfile");
  }
}
```

Felhasználói típusok

```
class InFile
{
  public class  EmptyRowException : Exception { }

  public struct Athlete
  {
    public string name;
    public bool below;
  }

  private readonly TextFileReader reader;

  public InFile(string fname)
  {
    reader = new TextFileReader(fname);
  }

  public bool Read(out Athlete e) { ... }
}
```



Az olvasó (read) művelet

A soron következő sor feldolgozásával egy újabb sportoló adatait állítja elő.

A = (r : infile(Sor), e : Sportoló, st : Status)
Sportoló = rec(név:$\mathbb{S}$, alá:$\mathbb{L}$)
Ef = (r = r')
Uf = ((| r'|> 0 ⟶ st = norm ∧
e = feldolgoz(r'$_1$) ∧
r = < r'$_2$, … , r'$_{|r'|}$>) ∧
(| r'|= 0 ⟶ st = abnorm))

az aktuális sor

amikor nincs több sora az inputfájlnak

```
public bool Read(out Athlete e)
{
  if (reader.ReadLine(out string line) )
  {
    if (line == "") throw new EmptyRowException();
    ... // processing the content of line
    return true;
  }
  else
  {
    e.name = ""; e.below = false;
    return false;
  }
}
```

Egy sor feldolgozása

Szavakra (tokenekre) bontott sorban az első néhány szó alkotja a sportoló nevét, majd az első időpont adattól kezdve jönnek a mérések (időpont-pulzusszám párok). Döntsük el van-e legalább 100-as pulzusszámú, és azt követően 100-nál kisebb pulzusszámú mérés.

Gipsz Jakab Elemér 00:01 67 01:15 89 02:55 102 04:03 108

minden sort tokenekre (szavakra) bontunk

A = (sor : $\mathbb{S}$*, e : Sportoló) Sportoló = rec(név:$\mathbb{S}$, alá:$\mathbb{L}$)

összefűzzük egy névvé az első időpont előtti szavakat

Ef = (sor = sor')

az első időpont

Uf = ((e.név, (név'', sor'')) = $\bigoplus_{\text{név}[3]\neq':' \\ \text{név}\in\text{sor}'}$ <név>

a sor hátralevő része

majd megkeressük az első legalább 100-as pulzusszámú mérést

$\wedge$ (l1, (. , sor''') = SEARCH (pulzus≥100)
(idő, pulzus)∈<név">⊕sor"

a sor hátralevő része

felsorolás folytatása a név'' felhasználásával, de innentől már szópárokat kell olvasni

$\wedge$ l2 = SEARCH (pulzus<100)
(idő, pulzus)∈sor'''

végül eldöntjük, van-e 100-nál kisebb mérés az első legalább 100-as pulzusszám után

$\wedge$ e.alá = (l1 $\wedge$ l2)

)

Egy sor feldolgozása

```
char[] separators = new char[] { ' ', '\t' };
string[] tokens
       = line.Split(separators, StringSplitOptions.RemoveEmptyEntries);
e.name = "";
int i = 0;
foreach (string str in tokens)
{
   if (str[2] == ':') break;
   e.name += str + " ";
   ++i;
}
bool l1 = false;
for (++i ; i < tokens.Length; i += 2)
{
   if (l1 = (int.Parse(tokens[i]) >= 100) ) break;
}

bool l2= false;
for (i += 2; i < tokens.Length; i += 2)
{
   if (l2 = (int.Parse(tokens[i]) < 100) ) break;
}
e.below = l1 && l2;
return true;
```

szavakra való tördelés

szavak felsorolása egyesével

összefűzés feltétel fennállásáig

szavak felsorolása kettesével

lineáris keresés

szavak felsorolása kettesével

lineáris keresés

Szürkedoboz tesztelés vázlata

kiválogatás (feltételes összegzés):

a felsoroló hossza szerint:	0, 1, 2, több feltételnek megfelelő sportoló
a felsorolás eleje/vége:	csak az első és utolsó sportoló felel meg a feltételnek
terhelés:	nem kell

névtagok összefűzése (összegzés):

a felsoroló hossza szerint:	1, 2, több névtagú sportoló
a felsorolás eleje/vége:	előzővel letudva

legalább 100-as mérés keresése (lineáris keresés) :

a felsoroló hossza szerint:	0, 1, 2, több mérés az első legalább 100-as mérés előtt
a felsorolás eleje/vége:	előzővel letudva
eredmény szerint:	99, 100, 101

100 alatti mérés keresése (lineáris keresés):

a felsoroló hossza szerint:	0, 1, 2, több mérés az első legalább 100-as mérés után
a felsorolás eleje/vége:	elején, illetve a végén 100 alatti mérés
eredmény szerint:	99, 100, 101