Programozási módszertan zárthelyi

Lőrentey Károly
lorentey@valerie.inf.elte.hu

2000. október 31.

Előszó

Ez a pár oldalas leírás a másodéves programozó matematikusok által a *programozási módszertan 2.* tárgyból 2000. október 31-én megírt írásbeli zárthelyi feladatainak megoldását tartalmazza. A hallgatóknak másfél óra állt rendelkezésükre a két feladat papíron történő kidolgozásához. Mindkét feladat 30-30 pontot ért, tehát a ZH-n összesen 60 pontot lehetett elérni.

A megoldás során feltesszük, hogy az olvasó tisztában van az ELTÉ-n oktatott, relációalgebrára épülő programozási modellel [1], valamint ismeri az alapvető programozási tételeket [2].

A megoldás közreadásának célja a tanyanyag megértésének megkönnyítésén túl a LaTeX szedőrendszer[4], valamint a struktogramok rajzolására szolgáló `stuki` segédcsomag[5] használatának bemutatása. A dokumentum LaTeX forrása elérhető a

`http://www.inf.elte.hu/~lorentey/modszertan2/2000-10-31.tar.gz`

címen.

Az érdeklődők további gyakorlófeladatokat találhatnak a gyakorlat hivatalos feladatgyűjteményében[3].

Kellemes tanulást kíván

a gyakvezér.

1. Első feladat

> A büfé szakos hallgatóknak n féle tárgyat oktatnak. Az egyes tárgyakat felvett hallgatók törzsszámait az f_i fájlokban tárolják ($i \in [1..n]$). A hallgatói nyilvántartást egy (*törzsszám*, *név*) párokból álló h szekvenciális fájl tárolja. Adjuk meg azon hallgatók névsorát, akik k-nál kevesebb, vagy m-nél több tárgyat vettek fel! (A fájlok törzszám szerint szigorúan monoton növekvően rendezettek.)

Néhány megjegyzés:

1. Előfordulhat, hogy egy hallgató egyáltalán nem vett fel tárgyat. Ekkor a hallgató törzsszáma egyetlen f_i fájlban sem szerepel, de a nyilvántartásban benne van. Az ilyen hallgatókat is figyelembe kell venni, és ha $k > 0$, akkor ki kell őket írni a kimeneti fájlba!

2. Természetesen a kimeneti fájlnak nem kell alfabetikusan növekvő sorrendben lennie. (A *névsor* itt csak egyszerűen nevek sorozatát jelenti.)

3. A k és m számok viszonyáról semmit sem tudunk. Ha véletlenül $k < m$, akkor az összes hallgató nevét ki kell írnunk, így a feladat érdektelenné válik. Ez persze nem baj, ezt az esetet sem szabad kizárni.

A feladat állapottere a következő:

$$A = \underset{f_1}{F} \times \cdots \times \underset{f_n}{F} \times \underset{h}{H} \times \underset{k}{\mathbb{N}_0} \times \underset{m}{\mathbb{N}_0} \times \underset{x}{X}$$

$$F = \mathrm{seq}(\mathbb{N}) \qquad I_F(f) : \forall i, j \in [1..\mathrm{dom}\, f] : i < j \to f_i < f_j$$

$$H = \mathrm{seq}(tsz : \mathbb{N}, nev : \textit{Név}) \qquad I_H(h) : \forall i, j \in [1..\mathrm{dom}\, h] : i < j \to h_i.tsz < h_j.tsz$$

$$X = \mathrm{seq}(\textit{Név})$$

(A *Név* típust nem definiáljuk közelebbről, de feltesszük, hogy a típusértékhalmazának elemein értelmezett az egyenlőségvizsgálat.)

Ezenkívül az előfeltételben még szerepel, hogy a hallgatói nyilvántartás teljes, azaz

$$\bigcup_{i=1}^{n} \{f_i\} \subseteq \{h.tsz\}.$$

Térjünk át egy új állapottérre, amelyben az f_i és a h fájlokat egyetlen g fájlba vonjuk össze, mely egy hallgató törzsszáma és neve mellett tartalmazza a hallgató által felvett tárgyak számát is:

$$A' = \underset{g}{G} \times \underset{k}{\mathbb{N}_0} \times \underset{m}{\mathbb{N}_0} \times \underset{x}{X}$$

$$G = \mathrm{seq}(tsz : \mathbb{N}, nev : \textit{Név}, db : \mathbb{N}_0) \qquad I_G(g) : \forall i, j \in [1..\mathrm{dom}\, g] : i < j \to g_i.tsz < g_j.tsz$$

Az A és A' állapotterek közötti kapcsolatot formálisan az alábbi módon írhatjuk le:

$$\{h\} = \{(g_i.tsz, g_i.nev) \mid i \in [1..\mathrm{dom}\, g]\} \tag{1}$$

$$\forall i \in [1..\mathrm{dom}\, g] : g_i.db = \left|\left\{ j \in [1..n] \mid g_i.tsz \in \{f_j\} \right\}\right| \tag{2}$$

Az (1) képlet írja le, hogy mi kerül g elemeinek első két komponensébe, a (2) pedig a harmadik komponensek értékeit adja meg. Egyszerűen fogalmazva a g

sorozatot a h-ból úgy kaphatjuk meg, hogy az egyes hallgatók adatait kiegészítjük az általa felvett tárgyak számával.

Az új állapottéren már egyszerűen specifikálhatjuk a feladatot:

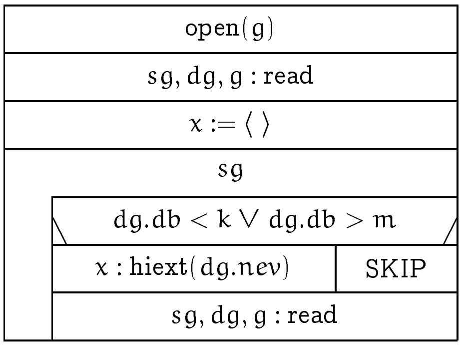
$$B = \underset{g'}{G} \times \underset{k'}{\mathbb{N}_0} \times \underset{m'}{\mathbb{N}_0}$$

$$Q : g = g' \wedge k = k' \wedge m = m'$$

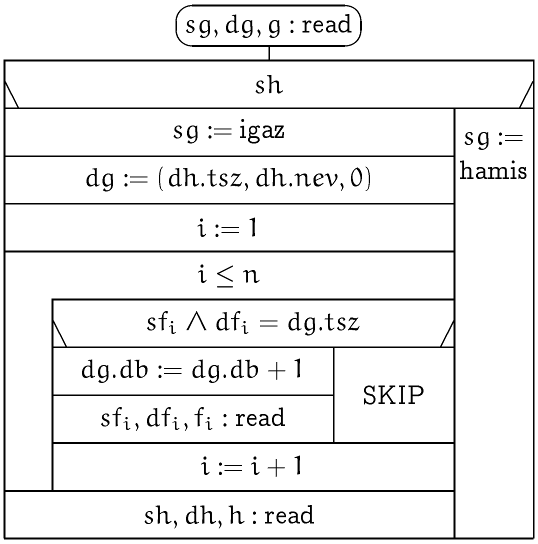
$$R : k = k' \wedge m = m' \wedge x = \underset{i=1}{\overset{\mathrm{dom}\, g'}{\mathrm{con}}} \Phi(i), \quad \text{ahol}$$

$$\Phi : [1..\mathrm{dom}\, g'] \longrightarrow X \qquad \Phi(i) = \begin{cases} \langle g_i.nev \rangle, & \text{ha } g_i.db < k \vee g_i.db > m \\ \langle\, \rangle, & \text{különben.} \end{cases}$$

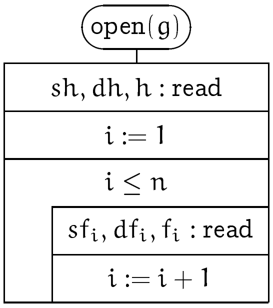
Belátható, hogy a $\mathrm{con}_{i=1}^{\mathrm{dom}\, g'} \Phi(i)$ függvény elemenként feldolgozható (például alkalmazható [2] 45. oldalán bizonyított tétel). Ezek szerint a feladat ezen az állapottéren visszavezethető az egyváltozós egyértékű elemenkénti feldolgozás programozási tételére. A kapott program:



A g fájl read műveletében a $dg.tsz$, $dg.nev$ értékeket h következő elemével inicializáljuk, majd pedig a $dg.db$ értékben megszámoljuk azon f_i fájlokat, amelyben szerepel a $dg.tsz$ érték. (Számlálás tétele.) A megtalált értékeket rögtön el is hagyjuk az f_i fájlokból. A művelet végén a h fájlból is elhagyjuk a feldolgozott elemet.



A g fájl open műveletében egyszerűen minden fájlból előreolvasunk egy elemet.



2. Második feladat

Egy szövegfájlban a sorokat sorvége jel (EOF) vagy fájlvége határolja. A sorokon belül a szavakat elválasztójelek ($\mathtt{ELV} \subseteq \mathit{Char} \setminus \{\mathtt{EOL}\}$) választják el egymástól. A fájlban a bekezdéseket üres (csak elválasztójeleket tartalmazó) sorok választják el egymástól.

Adott egy n darab szót tartalmazó vektor. Adjuk meg azon bekezdések sorszámait, melyek az n darab szó mindegyikét tartalmazzák, a vektorban megadott sorrendben! (A bekezdések sorszámozása 1-től indul, a vektor egyes szavakat többször is tartalmazhat. A szavaknak nem feltétlenül kell egymás mellett lenniük. Szavak összehasonlítása megengedett művelet.)

Néhány megjegyzés:

1. Legyen $\mathit{Betű} := \mathit{Char} \setminus \mathtt{ELV} \setminus \{\mathtt{EOL}\}$. Ekkor a $\mathit{Betű}$, ELV, $\{\mathtt{EOL}\}$ halmazok páronként diszjunktak.

2. Legyen például a v vektor a következő:

   $$v = \langle \mathtt{foo}, \mathtt{bar}, \mathtt{foo}, \mathtt{foo}, \mathtt{bar} \rangle,$$

   és legyen a szövegfájl egy bekezdése

   $\mathtt{bar\ baz\ bar}\ \boxed{\mathtt{foo}}\ \mathtt{baz\ foo\ foo\ baz}\ \boxed{\mathtt{bar}}\ \mathtt{bar}\ \boxed{\mathtt{foo}}$
   $\mathtt{bar\ baz\ bar}\ \boxed{\mathtt{foo}}\ \mathtt{foo\ baz}\ \boxed{\mathtt{bar}}\ \mathtt{foo\ foo\ bar\ baz.}$

   Ekkor a bekezdés a v vektor összes szavát tartalmazza, a vektorban megadott sorrendben. Az illeszkedő szavakat bekereteztük.

3. Egy bekezdés ellenőrzése a következő módon történhet: először megkeressük a vektor első szavának első előfordulását a bekezdésben. Ha nem találtunk ilyen szót, akkor a bekezdés rossz. Ellenkező esetben a keresést a második szó első előfordulásával folytatjuk, de nem a bekezdés elejétől, hanem az előző találat utáni első szótól kezdve. Ha ezt a módszert folytatva mind az n szót megtaláltuk, akkor a bekezdés illeszkedik a vektorra.

4. Természetesen a fájl elején és végén lévő esetleges üres sorok nem kezdenek új bekezdést. Minden bekezdés tartalmaz legalább egy szót.

A feladat állapottere a következő:

$$A = \underset{t}{T} \times \underset{v}{V} \times \underset{s}{X}$$

$$T = \mathrm{seq}(\mathit{Char}) \qquad \mathit{Char} = \mathit{Betű} \cup \mathtt{ELV} \cup \{\mathtt{EOL}\}$$

$$V = \mathrm{vect}([1..n], \mathit{Szó}) \qquad \mathit{Szó} = \mathrm{seq}^{+}(\mathit{Betű})$$

$$X = \mathrm{seq}(\mathbb{N}).$$

(A $\mathrm{seq}^+(\mathit{Betű})$ a betűkből képezett nemüres sorozatok halmazát jelöli.)

Ezen az állapottéren nem tűnik egyszerűnek a feladat megoldása, ezért térjünk át egy új állapottérre:

$$A = \underset{f}{F} \times \underset{v}{V} \times \underset{s}{X}$$

$$F = \mathrm{seq}(ssz : \mathbb{N}, jo : \mathbb{L}) \qquad I_F(f) = \forall i \in [1..\mathrm{dom}\, f] \colon f_i.ssz = i.$$

Az f fájl i-edik elemének jo értéke igaz, ha az i. bekezdés illeszkedik a vektorra.

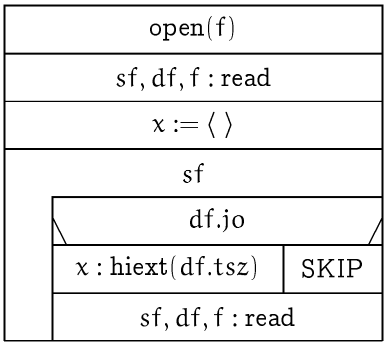
Ezen az állapottéren a feladat egy egyszerű elemenkénti feldolgozásra vezethető vissza[1]:

$$B = \underset{f'}{F} \times \underset{v'}{V}$$

$$Q : f = f' \wedge v = v'$$

$$R : v = v' \wedge x = \overset{\mathrm{dom}\, f'}{\underset{i=1}{\mathrm{con}}}\ \Phi(i), \quad \text{ahol}$$

$$\Phi : [1..\mathrm{dom}\, f'] \longrightarrow X \qquad \Phi(i) = \begin{cases} \langle f_i.ssz \rangle, & \text{ha } f_i.jo \\ \langle\,\rangle, & \text{ha } \neg f_i.jo. \end{cases}$$



Az állapottér-transzformáció feloldásához vezessünk be egy közbülső állapotteret, melyben a bekezdéseket szavakra bontjuk fel:

[1] Az F típusba éppen azért vettük fel a bekezdés sorszámát, hogy a kiszámítandó függvény elemenként feldolgozható legyen. A sorszám feleslegesnek látszik, hiszen egy számlálóban is számolhatjuk, hogy hány elemet olvastunk eddig be f-ből. Így tehát az $F = \mathrm{seq}\,\mathbb{L}$ fájl is elég lenne. Ha azonban egy külön változóban számoljuk a beolvasott értékeket, akkor x értékét kiszámoló függvény nem lenne elemenként feldolgozható, így csak a rekurzív függvény kiszámításának tételét használhatnánk, melyhez némileg bonyolultabb specifikáció tartozik.

$$A'' = \underset{g}{G} \times \underset{v}{V} \times \underset{x}{X}$$
$$G = \text{seq}(ssz : \mathbb{N}, \text{szó} : \textit{Szó})$$
$$I_G(g) : \forall i, j \in [1..\text{dom}\, g]: i < j \to g_i.ssz \leq g_j.ssz \ \wedge \ g \neq \langle\ \rangle \to g_1.ssz = 1.$$

A g fájl i-edik elemének második komponense a szövegfájl i-edik szavát tartalmazza, az első komponens pedig a szót tartalmazó bekezdés sorszámát adja meg.

Az A és A'' állapotterek közötti kapcsolat egy részét formálisan a *szekvenciális megfelelő* [2, 78. oldal] fogalmának segítségével az alábbi módon írhatjuk fel:

$$\text{seq}(g \mid \textit{Betű}) = \text{seq}(t \mid \textit{Betű}). \tag{3}$$

(Azaz ha a t fájlból elhagyjuk az elválasztó- és sorvégejeleket, akkor a g fájl szavait felépítő betűk sorozatát kapjuk meg.)

Most valósítsuk meg az f fájlra vonatkozó műveleteket! A read művelet feladata tulajdonképpen az, hogy egy bekezdés szavainak sorozatáról eldöntse, hogy illeszkedik-e a v vektorra. Ezt a 3. megjegyzésben leírt módszerrel kényelmesen el lehet dönteni. A módszert formálisan egy rekurzív függvény segítségével lehet leírni, mely egy bekezdés szavainak sorozatáról eldönti, hogy mekkora a v vektor leghosszabb kezdőszelete, ami illeszkedik a bekezdésre:

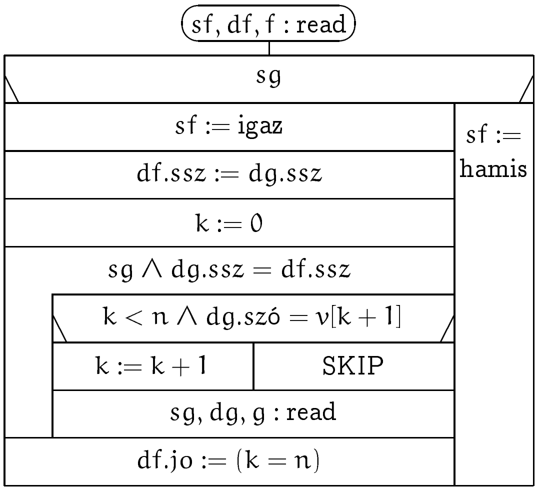
$$\Psi : \text{seq}(\textit{Szó}) \longrightarrow \mathbb{N}_0$$
$$\Psi(\langle\ \rangle) = 0$$
$$\Psi(\text{hiext}(g, s)) = \begin{cases} \Psi(g) + 1, & \text{ha } \Psi(g) < n \wedge s = v[\Psi(g) + 1] \\ \Psi(g), & \text{különben.} \end{cases}$$

A definíció első sora azt írja le, hogy az üres bekezdés[2] a vektor egyetlen elemét sem tartalmazza, azaz az egyetlen szót sem tartalmazó bekezdés a vektornak csak a nulla hosszúságú kezdőszeletére illeszkedik.

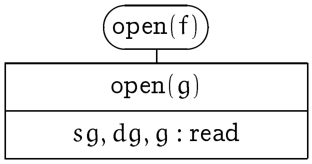
Az induktív stílusú második sor szerint pedig ha egy új szót rakunk egy bekezdés végére, akkor az így kapott bekezdés a vektor a korábbinál eggyel hosszabb kezdőszeletére illeszthető, amennyiben a kiinduló bekezdés még nem illeszkedett a teljes vektorra, és az a szó, amivel meghosszabbítottuk bekezdésünket, megegyezik a vektor következő elemével. Ellenkező esetben a meghosszabbított bekezdés ugyanakkora kezdőszeletre illeszthető, mint az eredeti.

[2] Üres bekezdések a gyakorlatban persze nem fordulnak elő, csak a Ψ definíciójának megkönnyítése érdekében indultunk ki az üres sorozatból.

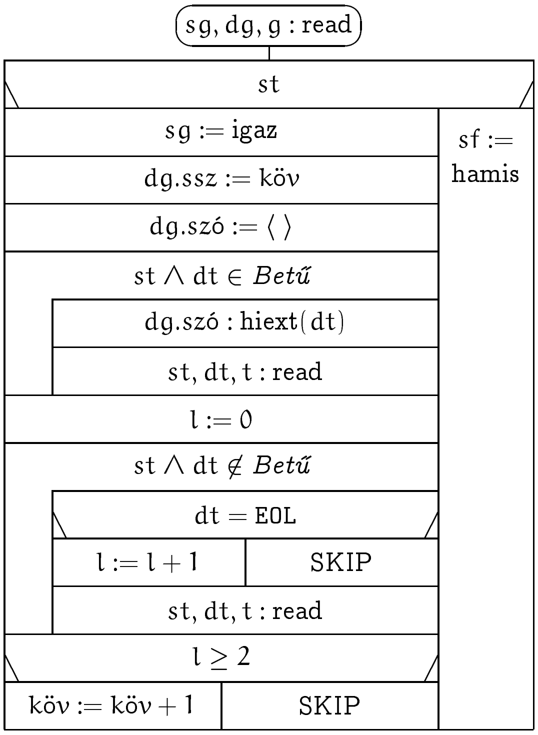
Most már megadhatjuk az f fájl read műveletét, melynek magva a fenti Ψ rekurzív függvény kiszámítása. Egy bekezdés akkor „jó”, ha a vektor n hosszúságú kezdőszeletére (azaz a teljes vektorra) illeszkedik.



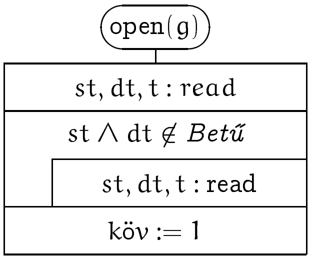
Az open(f) művelet egyszerűen a g megnyitásából és egy elem előreolvasásából áll:



Még hátra van a g fájl műveleteinek megvalósítása. A read művelet előfeltételéül feltesszük, hogy ha a t fájl még nem ürült ki, akkor egy új szó elején állunk, és a köv változó értéke a most következő szó bekezdésének sorszáma. A művelet első részében a következő szó betűit átmásoljuk a dg.szó sorozatba (egyváltozós egyértékű elemenkénti feldolgozás az identitásfüggvénnyel). Miután a szó végére értünk, a következő szó elejéig megszámoljuk (számlálás tétele) a sorvége jeleket. Ha a két szó között egynél több sorvége jel van, akkor ott egy új bekezdés kezdődik, tehát a köv változót eggyel meg kell növelni. A számlálás kényelmes módon helyreállítja az read invariánsát is, tehát két legyet ütünk egy csapásra.



A g fájlra vonatkozó open műveletnek a fő gondja a read művelet előfeltételének beállítása, melyet lineáris kereséssel valósít meg. A 4. megjegyzés alapján a fájl elején tetszőlegesen sok üres sor állhat, a legelső szó bekezdésszáma mindig egy.



Hivatkozások

[1] Fóthi Ákos: *Bevezetés a programozáshoz.* Egyetemi jegyzet (ELTE TTK, Budapest, 1983).

[2] Sike Sándor: *Programozási Módszertan.* Kézirat (`http://www.inf.elte.hu/~sike/prog_mod.zip`).

[3] Gregorics Tibor: *Állapottér-átalakítás*. Kézirat (http://www.inf.elte.hu/~gt/prog/alltrafo.zip).

[4] Wettl Ferenc, Mayer Gyula, Sudár Csaba: *LaTeX kezdőknek és haladóknak*. (Panem kft., Budapest, 1998).

[5] Lőrentey Károly: *Struktogramok készítése LaTeXben*. Kézirat (http://www.inf.elte.hu/~lorentey/stuki/).