Gregorics Tibor **1. beadandó/0.feladat** 2011. december 28.
EHACODE.ELTE
gt@inf.elte.hu
0.csoport

Feladat

Egy osztályba n diák jár, akik m darab tantárgyat tanulnak. Ismerjük a félév végi osztályzataikat. Igaz-e, hogy minden diáknak van legalább két ötöse?

Megoldási terv

$A = (napló: \mathbb{N}^{n \times m}, l: \mathbb{L})$
$Ef = (\ napló = napló'\ )$
$Uf = (\ Ef \wedge (l = \forall i \in [1..n] : ötösdb(i) \geq 2\ )\ ) = (\ Ef \wedge (\ l = \forall \overset{n}{\underset{i=1}{\textbf{search}}} (ötösdb(i) \geq 2)\ )\ )$

ahol $ötösdb(i) : [1..n] \rightarrow \mathbb{N}$ és $ötösdb(i) = \sum_{\substack{j=1 \\ napló[i,j]=5}}^{m} 1$

opt. lin ker		
m..n	~	*1..n*
β(*i*)	~	*ötösdb(i)* ≥ *2*

számlálás		
m..n	~	*1..m*
β(*i*)	~	*napló*[*i,j*]=5
c	~	*db*
i	~	*j*

l,i:=true,1	
l* ∧*i*≤*n	
	l:= ötösdb(i)≥2
	i:= i+1

db:= ötösdb(i)	
db:=0	
j= 1..m	
	***napló*[*i,j*]=5**
	db:=db+1 / ***SKIP***

Implementáció

Adattípusok megvalósítása

A tervben szereplő mátrixot `vector<vector<int>>`-ként deklaráljuk. Mivel a vektor *0*-tól indexelődik, azért a tervbeli ciklusok indextartományai a *0..n–1* és a *0..m–1* intervallumra módosulnak, ahol a *n*-re `t.size()` alakban, *m*-re pedig `t[i].size()` alakban hivatkozhatunk.

A megvalósításban a diákok és a tantárgyak neveit is tároljuk egy-egy külön tömbben.

Bemenő adatok formája

Az adatokat be lehet olvasni egy szöveges állományból vagy meg lehet adni billentyűzetről. Ha a programot parancssorból indítjuk úgy, hogy paraméterként megadjuk a bemenő adatokat tartalmazó szöveges állomány nevét, akkor innen olvassa be a program az adatokat. Ha nem adunk meg a parancssorban állomány nevet vagy nem parancssorból indul a program, akkor az először megkérdezi az adatbevitel módját, majd a szöveges állományból való olvasást választva bekéri az állomány nevét. A billentyűzetről vezérelt adatbevitelt a program párbeszéd-üzemmódban irányítja, és azt megfelelő adat-ellenőrzésekkel vizsgálja.

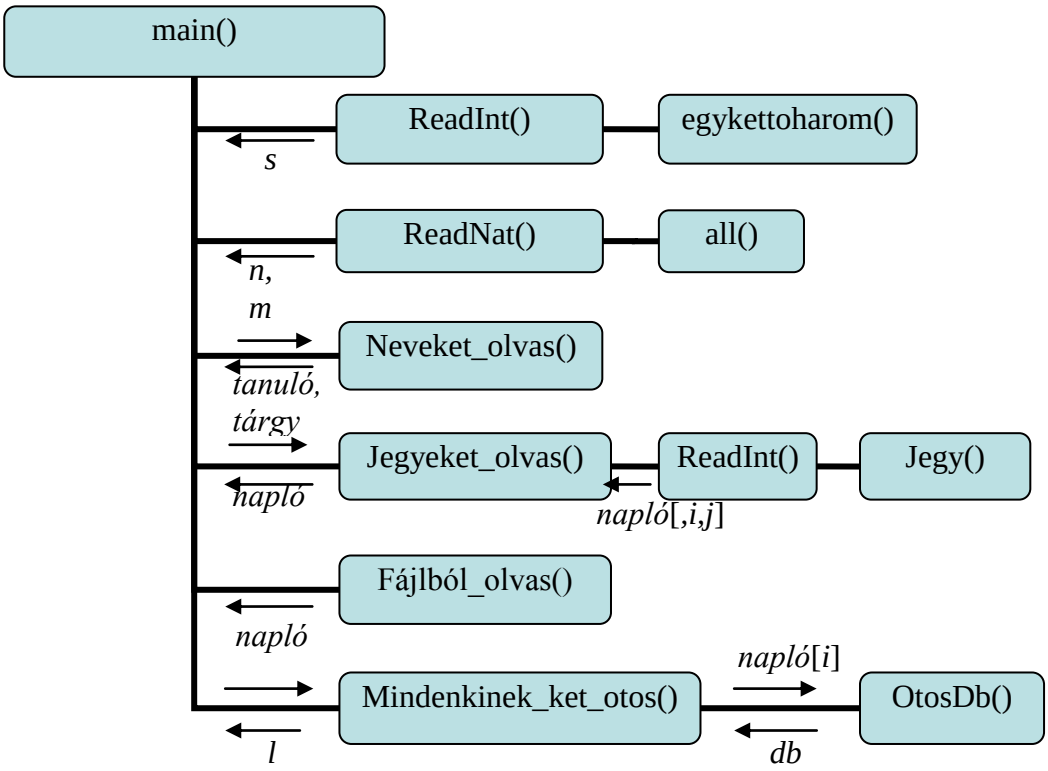
A szöveges állomány formája kötött, arról feltesszük, hogy helyesen van kitöltve, ezért ezt külön nem ellenőrizzük. Az első sor a tanulók és a tantárgyak számát tartalmazza, szóközökkel vagy tabulátor jelekkel elválasztva. Ezt követően olvashatók a tanulók nevei soronként, majd a tantárgyak nevei ugyancsak soronként. Végül az osztályzatok következnek a tanulók sorrendjében úgy, hogy minden tanuló jegyei egy sorban szóközökkel vagy tabulátor jelekkel legyenek elválasztva a tantárgyak megadott sorrendjében. Minden sor végén (az utolsó sor végén is) sorvége jel legyen.

Példa:

```
2 3
Kerek Berci
Nagy János
Matek
Föci
Orosz
4   3   2
5   4   4
```

Program váz

A program több állományból áll. A *read* csomagban (`read.h, read.cpp`) a `ReadInt()`, `ReadNat()`, és az `all()` függvényeket találjuk (ezek az egész számok billentyűzetről való beolvasását támogatják) , az összes többi függvény a `naplo.cpp` állományban van.



A `ReadInt()` segítségével azt a felhasználói döntést olvassuk be, hogy fájlból vagy billentyűzetről történjen-e az adatok bevitele. A `ReadNat()` a billentyűzetről való beolvasás esetén a napló méreteinek megadására szolgál. Mindkettő bemenete két sztring (címke, hibaüzenet) és egy ellenőrző függvény.

A `Neveket_olvas()` kétszer kerül felhasználásra: mind a tanulók, mind a tantárgyak neveinek beolvasásánál. Bemenete az *n* illetve az *m*. A `Jegyeket_olvas()` az osztályzási naplót tölti fel. Bemenete az „üres” osztályzási napló, valamint a tanulók és tantárgyak névsora, kimenete a feltöltött osztályzási napló. Egy jegy beolvasásához a `ReadInt()`-et használja. A `Fajlbol_olvas()` egy szöveges állományból olvassa be mind a tanulók és tantárgyak számát, nevét, mind a jegyeket.

Tesztelési terv

Tesztesetek a feladat specifikációja alapján (fekete doboz tesztelés)

Érvényes tesztesetek:

A. Külső programozási tétel (intervallum és lineáris keresés):
   1. Üres napló esetei:
      (t10.txt: nincs tanuló, sem tantárgy – válasz: igaz)
      (t11.txt: nincs tanuló – válasz: igaz)
   2. Egy tanuló, egy tantárgy esete. (t2.txt: 5 – válasz: hamis)
   3. Több tanuló, több tantárgy: csak az első tanulónak nincs két ötöse.
      (t4.txt: 3×2 [5, 3, 5, 5, 5, 5] – válasz: hamis)
   4. Több tanuló, több tantárgy: csak az utolsó tanulónak nincs két ötöse.
      (t5.txt: 3×2 [5, 5, 5, 5, 1, 5] – válasz: hamis)
   5. Több tanuló, több tantárgy: mindenkinek van legalább két ötöse.
      (t3.txt: 2×3 [5, 3, 5, 5, 5, 5] – válasz: igaz)

6. Több tanuló, több tantárgy: senkinek nincs két ötöse.
   (t6.txt: 3×2 [5, 3, 5, 3, 1, 5] – válasz: hamis)

B. Belső programozási tétel (intervallum és számlálás):
1. Üres napló esetei:
   (t10.txt: nincs tanuló, sem tantárgy – válasz: igaz)
   (t12.txt: nincs tantárgy – válasz: hamis)
2. Egy tanuló, egy tantárgy esete.
   (t2.txt: 5 – válasz: hamis)
3. Egy tanulónak két ötöse van: az első és utolsó
   (t13.txt: 1×4 [5, 3, 4, 5] – válasz: igaz)
4. Egy tanulónak nincs ötöse
   (t8.txt: 1×4 [1, 3, 4, 3] – válasz: hamis)
5. Egy tanulónak egyetlen ötöse van
   (t9.txt: 1×4 [1, 5, 4, 3] – válasz: hamis)
6. Egy tanulónak két ötöse van
   (t13.txt: 1×4 [5, 3, 4, 5] – válasz: igaz)
7. Egy tanulónak sok ötöse van
   (t14.txt: 1×4 [5, 5, 5, 3] – válasz: igaz)

Érvénytelen tesztesetek:
1. Nem megengedett értékek:
   Nem *1* és *5*- közé eső osztályzatok (nem okoz problémát)
   Nem számok az osztályzatok (ilyen eseteket eleve kizártunk, nem ellenőrizzük)

Tesztesetek a megoldó kód alapján (fehér doboz tesztelés)

A beolvasást végző függvények tesztelése:
1. Menü választás tesztelése (1, 2, 3, más)
2. Beolvasás mindhárom módozatának tesztelése.
3. Parancssorból indítás fájlnévvel és anélkül.
4. Nem létező fájlnév megadása.
5. Hibás adatok a billentyűzetről (negatív tanuló vagy tantárgy szám, hibás osztályzat).

A `Mindekinek_ket_otos()` függvény tesztelése:
1. igen válasz esete több tanulónál (ciklus feltétel tesztje):
   (t7.txt: 3×2 [5, 5, 5, 5, 5, 5] – válasz: igaz)
   (t11.txt: 0×3 [] – válasz: igaz)
2. nemleges válasz, amely az első tanulónál jelentkezik (kilépés az első):
   (t6.txt: 3×2 [5, 3, 5, 3, 1, 5] – válasz: hamis)
3. nemleges válasz, amely az utolsó tanulónál jelentkezik (kilépés több menet):
   (t5.txt: 3×2 [5, 5, 5, 5, 1, 5] – válasz: hamis)
4. elágazás feltétel tesztelése (kisebb, épp egyenlő, nagyobb, mint 2 esetei):
   (t5.txt: 3×2 [5, 5, 5, 5, 1, 5] – válasz: hamis)

A `OtosDb()` függvény tesztelése:
1. olyan adatsorra, amikor egyszer sem lép a ciklusba a vezérlés;
   (t12.txt: 2×0 [] – válasz: 0)
2. olyan adatsorra, amikor egyszer sem növeljük a darabszámot;

(t8.txt: 1×4 [1, 3, 4, 3] – válasz: 0)

3. olyan adatsorra, amikor egyszer növeljük a darabszámot;
(t9.txt: 1×4 [1, 5, 4, 3] – válasz: 1)

4. olyan adatsorra, amikor többször növeljük a darabszámot;
(t13.txt: 1×4 [5, 3, 4, 5] – válasz: 2)

Alternatív megoldási terv (rekurzív függvénnyel)

$$Uf = (Ef \wedge (l = \forall i \in [1..n] : jótanuló(i))) = (Ef \wedge (l = \forall \overset{n}{\underset{i=1}{\textbf{search}}} (jótanuló(i))))$$

ahol $jótanuló(i) = \exists j \in [1..m] : ötösdb\_i(j) = 2$

ahol $ötösdb\_i : [0..m] \rightarrow \mathbb{N}$

$ötösdb\_i(0) = 0$

$$ötösdb\_i(j) = ötösdb\_i(j\text{-}1) + \begin{cases} 1 & ha\ napló[i, j] = 5 \\ 0 & ha\ napló[i, j] \neq 5 \end{cases} \quad (j \geq 1)$$