Szita Balázs
SZBRAGI.ELTE, CVDW69
szitab@ik.elte.hu

Programozás, 10. csoport
1. feladatcsoport, 0. feladat
2014. április 2.

Feladat

Egy csokigyár közvélemény-kutatást tartott arról, mennyire népszerű egy bizonyos terméke.
Az alanyoktól megkérdezték a korukat, valamint azt, hogy egy ötös skálán hányasra értékelik a kérdéses terméket.

Igaz-e, hogy nincs olyan korosztály, amelyik átlagban 4-esnél rosszabbra értékelte volna?

Azt, hogy mi számít „korosztálynak”, a vállalat maga dönti el, egy megfelelően megválasztott skála alapján.

Specifikáció

Tegyük fel, hogy korosztályokra lebontva rendelkezésre állnak az értékelések. Azt kell megállapítanunk, igaz-e, hogy nincs olyan korosztály, amelyikhez tartozó átlag 4-esnél rosszabb, másképp fogalmazva *igaz-e, hogy minden korosztály, olyan hogy az átlag nem rosszabb 4-esnél*. Ez tehát egy *optimista lineáris keresés*.

A bemenő adatokat fogjuk fel tehát úgy, hogy adott egy n elemű tömb, ahol n a korosztályok száma és a tömb i. eleme tartalmazza az i. korosztályhoz tartozó összes értékelést – mivel nem tudjuk egy korosztályhoz hány értékelés tartozik, ezért ezt absztrakt szinten értékelésekből álló sorozatokként adhatjuk meg.

$$A = (o\colon ([1..5]^*)^n, l\colon \mathbb{L})$$

$$ef = (o = o')$$

$$uf = \left(ef \wedge l = \overset{n}{\underset{i=1}{\forall SEARCH}} \left(joatlag(o[i]) \right) \right)$$

ahol a $joatlag$ egy olyan függvény lesz, ami akkor ad igazat, ha a paraméterül kapott sorozat értékeinek átlaga legalább 4. Külön foglalkozni kell azzal az esettel, ha egy korosztályhoz nem tartozik értékelés. Ekkor a sorozat üres, és nullával osztanánk az átlag megállapítása közben. Kiköthetnénk előfeltételben, hogy minden sorozat legalább egy elemű, ehelyett inkább azt az értelmezést, azt a megoldást választottam, hogy az ilyen üres sorozatok feleljenek meg a feltételnek, hiszen végső soron nem tartalmaznak 4-esnél rosszabb értékelést, következésképpen az átlaguk se lehet „rossz”.

Ez alapján:

$$joatlag\colon [1..5]^* \to \mathbb{L} \text{ és}$$

$$joatlag(s) = \left(|s| = 0 \vee \frac{\sum_{i=1}^{|s|} s_i}{|s|} \geqq 4 \right)$$

Ezt akár így is leírhattam volna, ekvivalens az előzővel:

$$joatlag(s) = \left(|s| \neq 0 \to \frac{\sum_{i=1}^{|s|} s_i}{|s|} \geqq 4 \right)$$

A $joatlag$ kiszámításához jól láthatóan szükség lesz a sorozat elemeinek *összegzésé*re, ezt egy külön alprogramban fogjuk megírni, amit a $joatlag$ számításakor függvényszerűen meg is fogunk tudni hívni.

Tehát a $joatlag$ végső definíciója:

$$joatlag(s) = \left(|s| = 0 \vee \frac{osszeg(s)}{|s|} \geqq 4 \right)$$

ahol:

$$osszeg: [1..5]^* \to \mathbb{N}$$

és

$$osszeg(s) = \sum_{i=1}^{|s|} s_i$$

Ez alapján specifikálhatom a belső tételt is ($ossz \coloneqq osszeg(s)$ kiszámítása):

$$A = (s: [1..5]^*, ossz: \mathbb{N})$$

$$ef = (s = s')$$

$$uf = \left(ef \wedge ossz = \sum_{i=1}^{|s|} s_i \right)$$

Visszavezetés és algoritmus

Külső tétel:

optimista lineáris keresés

$[m..n]$ ~ $[1..n]$
$\beta(i)$ ~ $joatlag(o[i])$
ind elhagyva

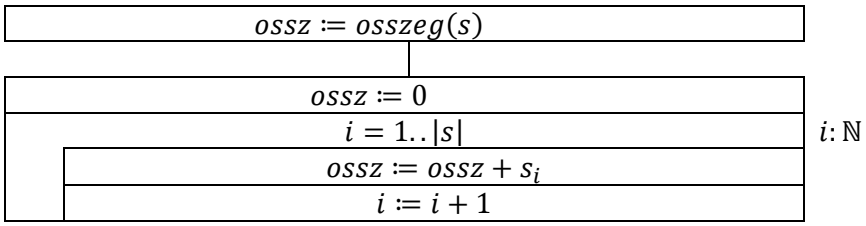
$i, l \coloneqq 1, \uparrow$	$i: \mathbb{N}$
$l \wedge i \leqq n$	
$l \coloneqq joatlag(o[i])$	
$i \coloneqq i + 1$	

Szita Balázs
SZBRAGI.ELTE, CVDW69
szitab@ik.elte.hu

Belső tétel:

összegzés

$[m..n]$ ~ $[1..|s|]$
$f(i)$ ~ s_i
s ~ $ossz$



Implementáció

A programot `C++` nyelven valósítottuk meg.

Adattípusok

A feladat megfogalmazásakor definiált „listák tömbjét” egy „kesztyűmátrixszal”, azaz egy olyan `std::vector`ral fogjuk reprezentálni, aminek elemei maguk is `vector`ok.

Az adatokat az *osztalyzatok* változóba olvassuk be, melynek típusa tehát `vector<vector<int> >`. Az osztályzatok 1 és 5 közötti egész számok lehetnek, ezt a feltételt a beolvasás során ellenőrizzük.

A külső vektor elemei a korosztályokat jelentik, ezeket is be kell tehát majd olvasnunk. A belső vektorok elemszáma pedig különböző, a beolvasás során a soron következő értékelést a `vector` `push_back()` metódusával szúrjuk a megfelelő korosztályt jelképező `vector` végére.

Az *osztalyzatok* beolvasása előtt még meghatározunk egy *korosztalyok* nevű segédtömböt is.

A struktogrambeli n nem más, mint az *osztalyzatok* mérete, azaz az `osztalyzatok.size()`, az egyes $|s|$ sorozathossz pedig az i-edik sorozatra az `osztalyzatok[i].size()` kifejezés értékével fogalmazható meg.

A tömbindexelést minden esetben 0-tól végezzük.

Adatok beolvasása

Először a korosztályokat kérjük be, ebből felépítünk egy korosztály-vektort, majd ennek adatai alapján fel tudjuk építeni az értékelések „mátrixát” is.

Korosztályok:

A korosztályok bekérésénél az aktuális korosztály felső határát kérjük be. A korosztályokat növekvő sorrendben kérjük be. A felhasználó tetszőlegesen sok korosztályt megadhat, a beolvasás folyamatát hibásan megadott értékkel szakíthatja meg.

Szita Balázs
SZBRAGI.ELTE, CVDW69
szitab@ik.elte.hu

A korosztályokat úgy építjük fel, hogy minden potenciális életkor (nem negatív számok) beleférjen egy korosztályba. Ezért ha a felhasználó egy ilyen „korosztály felső határ” számot sem ad meg, akkor egy darab korosztályt definiálunk, amibe minden lehetséges életkor bele fog tartozni. Általánosan, ha n darab korosztály-határt olvasunk be, akkor azzal $n + 1$ darab korosztályt határoztunk meg, ahogy a mellékelt példa mutatja:

input:

```
5 10 25
```

(természetes számok növekvő sorozata, elemei egy vagy több WS[1] karakterrel elválasztva)

korosztályok:

- 0-tól 5-ig
- 6-tól 10-ig
- 11-től 25-ig
- 26-tól

Az inputot akár fájlból, akár billentyűzetről is megadhatjuk. Fájlból való bekérés esetén először megadjuk az adatokat a fenti formában tartalmazó fájl nevét, ezután a program megkísérli megnyitni a fájlt és feldolgozni tartalmát. Hiba esetén azt jelzi, majd kilép.

Hibalehetőségek:

- Nem létező vagy nem megnyitható fájl
- A korosztály-határok nem *szigorúan monoton növekvő* sorrendben vannak

Ezekben az esetekben a fájlt hibásnak tekintjük.

Az is előfordulhat, hogy a fájlban nem értelmezhető formátumú adatok vannak, pl. `string`ek. Ekkor úgy járunk el, hogy az addig esetlegesen értelmesen beolvasott adatokat megtartjuk, az első hibás adatot és az őt követő (akár értelmes) adatokat pedig figyelmen kívül hagyjuk.

Billentyűzetről való bekérés esetén, amíg megfelelő adatot olvasunk, addig azt folyamatosan beszúrja a tömbbe, amennyiben hibázunk, választhatunk, hogy újra megkíséreljük beolvasni az aktuális értéket, vagy véget vetünk a beolvasásnak.

A megvalósítás szintjén mindkét módot egyazon függvény, a `beolv_korosztalyok()` kezeli, paramétereiként meg kell adni a beolvasandó tömböt, a beolvasás forrását (fájl vagy konzol), illetve hogy a beolvasás során lehetőséget kívánunk-e adni a javításra. Jellemzően a konzolos beolvasás során igen, a fájlos beolvasás esetén nem.

A függvény hívása fájlos beolvasás esetén:

```
beolv_korosztalyok(korosztályok tömbje, ifstream objektum, false);
```

Billentyűs beolvasásra:

```
beolv_korosztalyok(korosztályok tömbje, cin, true);
```

[1] WS: *whitespace*; a szóköz, tabulátorjel és újsor jel gyűjtőneve

A függvény egy logikai értékkel tér vissza, aminek értéke akkor igaz, ha a beolvasás sikeres volt és a korosztályok vektorában immár valami a feladat invariánsainak megfelelő adathalmaz található, és hamis, ha a beolvasás nem volt sikeres.

Az értékelések (osztályzatok) beolvasása:

Ezt is megadhatjuk fájlból vagy billentyűzetről. A beolvasás mechanizmusa a korosztályokéhoz hasonló, azaz fájl esetén meg kell adni a fájlnevet, ellenőrizzük a fájl meglétét és megnyithatóságát, majd feldolgozzuk azt. Billentyűs beolvasáskor pedig egyesével kell megadnunk az adatokat és minden hibázáskor dönthetünk, hogy javítunk vagy kilépünk.

Az adatokat (kor, értékelés) párokként kell megadni, ahol a kor egy természetes szám, az értékelés pedig egy 1 és 5 közé eső érték. Ha mindkét megadott érték megfelel ezen feltételeknek, az osztályzat hozzáadódik a korosztályának megfelelő lista végéhez.

Az adatokat tartalmazó fájl megfelelő formátumára egy példa:

```
3 5
10 5
99 4
3 2
6 3
26 2
```

(természetes szám–[1..5]-beli értékekből álló párok; a számok között egy vagy több WS karakterrel)

Az első szám az értékelő életkorát, a második pedig az osztályzatot jelenti. Láthatjuk, nem kötelező életkor szerinti rendezett sorban megadni az értékeket, valamint értelemszerűen egy életkor illetve egy osztályzat többször is szerepelhet.

A fenti korosztályokra és az iménti osztályzatokra vonatkozó példa alapján az alábbi osztályzat-adatbázis állna elő:

[0..5] → <5,2>
[6..10] → <5,3>
[11..25] → <>
[26..+ ∞)→ <4,2>

Hasonlóan a korosztályok bekéréséhez, az implementáció során egy függvénybe összevontuk a billentyűs és fájlos bekérést, a megvalósított függvény neve `beolv_osztalyzatok`. Paraméterül várja az osztályzatok tömbjét, ahova be szeretnénk olvasni, a korosztályok tömbjét, ami meghatározza, hogy melyik életkor melyik korosztályhoz tartozzon, valamint az adatforrást és a bekérés ismételhetőségét meghatározó bool értéket. Hasonlóan a fentihez, a beolvasás sikerességét jelölő logikai értékkel tér vissza a hívásának helyére, a beolvasást magát pedig mellékhatásként végzi el.

A függvény hívása fájlos beolvasás esetére:

```
beolv_osztalyzatok (osztályzatok tömbje, korosztályok tömbje, ifstream
objektum, false);
```

Billentyűs beolvasásra:

```
beolv_osztalyzatok (osztályzatok tömbje, korosztályok tömbje, cin, true);
```

Mindkét beolvasó függvény a tényleges beolvasás előtt törli a vektorok eddigi tartalmát!

A kimenet

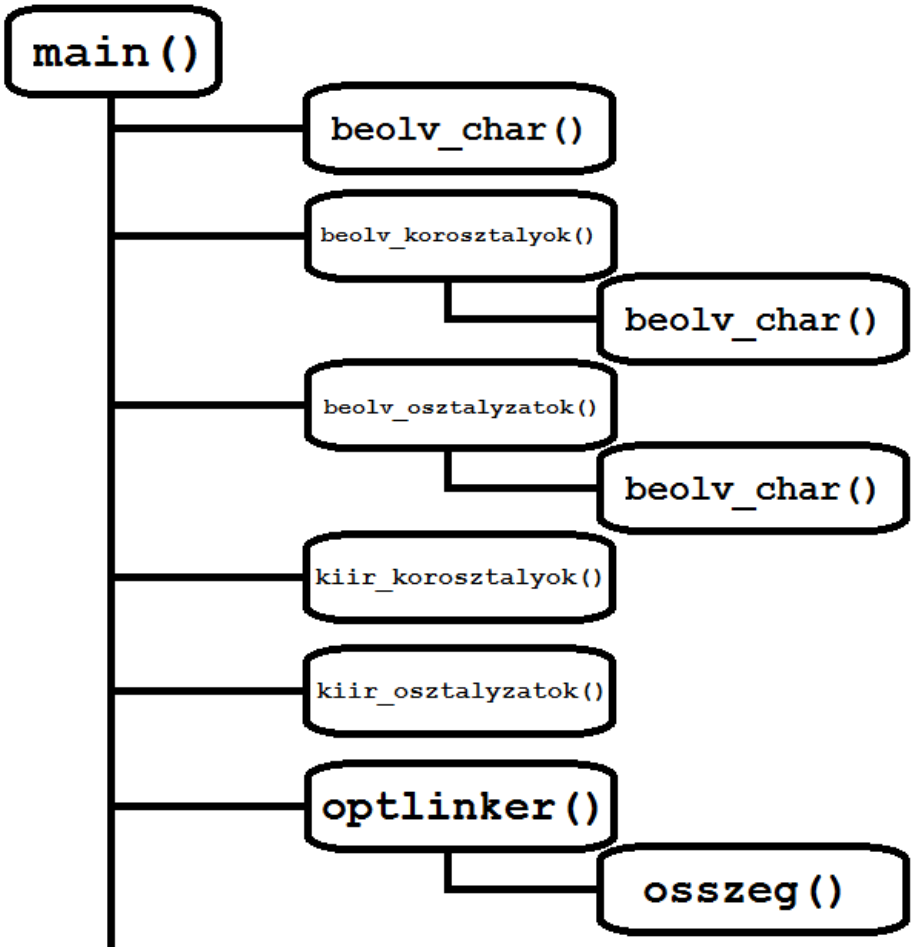
A sikeres beolvasás és feldolgozás után a feladatra adott válasz kerül kiírása a konzolra.

A projekt felépítése

A következő modulokat használjuk:

- *main* - a `main.cpp` forrásfájl, a program belépési pontja (`main` függvény) valamint a két specifikált részfeladat implementációja található itt. A `main` végzi a program vezérlését, a tételek hívását, és az eredmény kiírását
- *beolvasás* - `beolvasas.h` fejlécállomány valamint `beolvasas.cpp` forrásállomány. A fentiekben részletezett beolvasó függvények találhatók meg itt, kiegészítve egy *általános karakterbeolvasó* függvénnyel, melyet több alkalommal is használunk az adatok ismételt bekérésénél, valamint a sikeresen beolvasott adatokat ellenőrzési céllal kiíró műveletek definíciói (`kiir_korosztalyok`, `kiir_osztalyzatok`) is itt kaptak helyet
- *iostream* – külső könyvtár, mely a *konzolos kommunikáció* eszközeit teszi elérhetővé.
- *fstream* – külső könyvtár, mely a *fájlkezelés*ben nélkülözhetetlen, a beolvasáshoz használjuk
- *vector* – külső könyvtár, az std::vector típushoz

Függvényhívások hierarchiája:



Szita Balázs
SZBRAGI.ELTE, CVDW69
szitab@ik.elte.hu

Tesztelés

Az alábbiakban megadok néhány érvényes és érvénytelen tesztesetet

- *felület és adatbekérés* tesztelése billentyűről
  - „*menü*" használatának során olyan értékek beírása, amit nem szabadna → újra bekérés
  - *számként értelmezhetetlen* adatok megadása, oda ahova nem kéne → újra bekérés
  - korosztály-határok megadása *nem növekvő* sorrendben → a hibás adat újra bekérése
  - osztályzatok megadása [1..5] *intervallumon kívül* → a hibás adat újra bekérése

- *adatbekérés tesztelése fájlból*
  - korosztályok – *nem létező* fájlnév megadása [**f0.txt**] → hibajelzés, kilépés
  - korosztályok – *üres* fájl [**f1.txt**] → 1 darab korosztály ([0..végtelen]) sikeres beolvasása
  - korosztályok – *string*eket is tartalmazó fájl [**f2.txt**] [**f3.txt**] → az első stringig sikeres beolvasás
  - korosztályok – *negatív* szám [**f6.txt**] → az első ilyenig olvas
  - korosztályok – *nem rendezett* fájl [**f5.txt**] → hibajelzés, kilépés
  - korosztályok – *helyes* fájl [**f4.txt**] → sikeres beolvasás
  - osztályzatok – *nem létező* fájlnév megadása [**f0.txt**] → hibajelzés, kilépés
  - osztályzatok – *üres* fájl [**f1.txt**] → 0 darab értékelés sikeres beolvasása
  - osztályzatok – *string*eket tartalmazó fájl [**f2.txt**] [**f3.txt**] → az első stringig sikeres beolvasás (teljes párokat!)
  - osztályzatok – osztályzatok megadása [1..5] *intervallumon kívül* [**f4.txt**] [**f5.txt**] → az első ilyenig sikeres beolvasás, a hibás osztályzatot helyes életkor esetén se olvassa be
  - osztályzatok – *hibás életkor* [**f3.txt**] [**f6.txt**] → első ilyenig sikeres beolvasás

- adatbekérés, speciális esetek
  - *egy kor*hoz *több osztályzat* is tartozik [k:**f4.txt,** o:**f7.txt**] → mindet beolvassa
  - *egy osztályzat*-érték *több kor*hoz is tartozik [k:**f4.txt,** o:**f7.txt**] → mindet beolvassa
  - *egy elemű intervallum* megadása korosztálynak [**f4.txt**] → helyes, elvárt működés
  - *nem sorban* vannak az osztályzatok [k:**f4.txt,** o:**f7.txt**] → nincs vele gond
  - a konkrét korok az *intervallum szél*én vannak [k:**f4.txt,** o:**f7.txt**] → nincs vele gond

- a *beolvasás ellenőrzése* az ellenőrző kiírással (`kiir_korosztalyok`, `kiir_osztalyzatok` függvények)

- *optimista lineáris keresés* (külső tétel) fekete doboz tesztelése
  - üres osztályzatok tömbje [**f1.txt**] → igazat ad (üres halmaz és minden kvantor esete)
  - olyan eset amire igazat kell adnia [k:**f4.txt,** o:**f8.txt**] → igazat ad
    - béta épp hogy igaz (átlag=4) a példa 5. korosztálya
    - béta azért igaz, mert üres; a példa 3-4. korosztálya
  - olyan esetek amikre hamisat kell adnia
    - béta mindenre hamis [k:**f4.txt,** o:**f10.txt**]
    - béta az első néhányra igaz, de elromlik, aztán megint igaz [k:**f4.txt,** o:**f9.txt**]

- béta csak a végén hamis [k:**f4.txt,** o:**f11.txt**]
- béta csak az elején hamis [k:**f4.txt,** o:**f12.txt**]

- *optimista lineáris keresés* (külső tétel) fehér doboz tesztelése
  - üres osztályzat-lista egy bizonyos korosztályhoz [k:**f4.txt,** o:**f8.txt**] → elfogadja
  - kerekítési hibák kiszűrése [k:**f4.txt,** o:**f8.txt**] → jó

- ö*sszegzés* (belső tétel) fekete doboz tesztelése
  - üres tömb [k:**f4.txt,** o:**f9.txt**] 2. korosztálya → összeg=0
  - 1 elemű tömb [k:**f4.txt,** o:**f9.txt**] 1. korosztálya → összeg=az az egy elem
  - több elemű tömb [k:**f4.txt,** o:**f9.txt**] 5. korosztálya → összeg=stimmel ☺

- *összegzés* (belső tétel) fehér doboz tesztelése
  - itt azt lenne érdemes ellenőrizni, hogy belép-e vagy sem a ciklusba, de azt a fekete doboz tesztesetekkel végül is megnéztük, ezért ehhez az aspektushoz nem tartoznak további markáns tesztesetek…