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

Feladat

Egy nemzetközi cég egy szöveges fájlban tartja számon, hogy mely országok mely városaiban van üzlete.

A szöveges fájl minden sora egy országot reprezentál. A sor elején az ország neve (egy szó) szerepel, majd városnév-szám párok. A városnév lehet több szavas is, a szám az adott város lakosságát jelenti (1000 főben).

A fájlban ezek és csak ezek az adatok találhatók meg. Lehet hogy szerepel a fájlban olyan ország, amihez nincs város, és lehet, hogy a fájl üres. Feltehetjük, hogy a szöveges állomány helyesen van kitöltve.

Válaszoljuk meg az alábbi kérdéseket:

- Listázzuk ki országokra lebontva, hogy hány városában van jelen a vállalat
- A cég potenciálisan hány emberhez jut el világszerte? Azaz mennyi az érintett városok összlakosságszáma?
- Melyik országban tudja a cég a legtöbb embert megszólítani?

Csak egyszer menjünk végig a fájlon.

Specifikáció

Definiáljuk a *fájl*ban tárolt felsorolandó adatok típusát ekképpen:

$$Data \coloneqq record(orszag\colon String, varosok\colon (String \times \mathbb{N})^*)$$

Tehát olyan *rekord*ok, melyeknek egyik *mező*je az adott ország neve, a másik mezője pedig egy szöveg-szám párosokból álló lista. Ezek lesznek a városok: a szöveg a város neve, a szám pedig a lélekszáma.

$$A = (x\colon SeqInFile(Data), lista\colon (String \times \mathbb{N})^*, osszlakok\colon \mathbb{N}, maxnev\colon String)$$

$$ef = (x = x' \wedge |x| > 0)$$

A második feltételre a *maximumkiválasztás* tétel miatt van szükség.

$$uf = \Bigg(lista = \bigoplus_{e \in x'} (e.orszag, |e.varosok|) \wedge s = \sum_{e \in x'} osszlakossag(e) \wedge osszlakok = s \cdot 1000 \wedge maxorszag = \underset{e \in x'}{MAX_2}\ osszlakossag(e) \wedge maxnev = maxorszag.orszag \Bigg)$$

ahol:

$osszlakossag\colon Data \to \mathbb{N}$, úgy hogy:

$$osszlakossag(e) \coloneqq \sum_{v \in e.varosok} v_2$$

Tehát úgy kapjuk egy adott e ország összlakosságát, ha a városainak listáján végigmenve, azoknak rendre a második mezőjét (azaz a lélekszámát) összegezzük.

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

Programozás, 10. csoport
2. feladatcsoport, 0. feladat
2014. május 2.

Visszavezetés és algoritmus

Felsoroló:

(nevezetes) fájlfelsoroló, mely $Data$ típusú elemeket képes felsorolni

E	~	$Data$
$enor(e)$	~	$SeqInFile(Data)$
t	~	x

Külső tételek:

összegzés (listázás)

s	~	$lista$
H	~	$(String \times \mathbb{N})^*$
$f(e)$	~	$< (e.orszag, \lvert e.varosok \rvert) >$
$+,0$	~	$\oplus, <>$

maximumkiválasztás

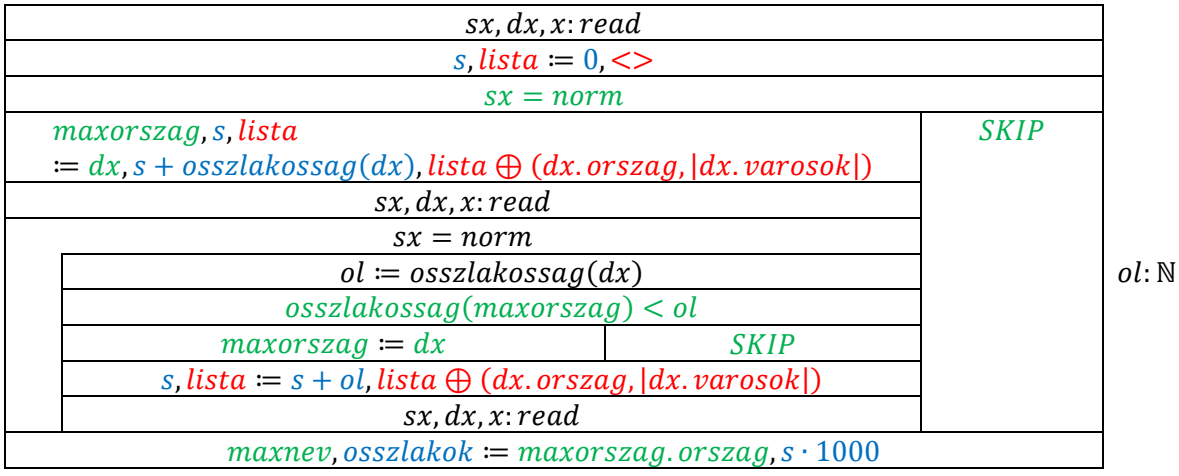
$elem$	~	$maxorszag$
max elhagyva		
H	~	$\mathbb{N}$
$f(e)$	~	$összlakosság(e)$

összegzés (lélekszám)

H	~	$\mathbb{N}$
$f(e)$	~	$összlakosság(e)$

Mivel a felsorolón csak *egyszer* mehetünk végig, és az is tiltva van, hogy *elmentsük* a felsorolt adatokat egy segédtömbbe, ezért a feladat három tételét kénytelenek vagyunk „*párhuzamosan*”, egy ciklusban számolni.

Amúgy a feladathoz tartozik még két *összetett függvény kiszámítása* konstrukció is, hiszen a végső lélekszámot úgy kapjuk, hogy s-t megszorozzuk 1000-rel, a legnagyobb lélekszámú ország nevét pedig úgy, hogy lekérjük $maxorszag$ nevét.



- feketével jelöltem a *felsorolás* műveleteit, és az *FHV*-t,
- pirossal a *listázós összegzés tételt*,
- kékkel az *összlakosságszámos összegzés* tételt (és a hozzá tartozó *összetett függvény*t),
- zölddel a *maximumkiválasztás*t (és a hozzá tartozó *összetett függvény*t).

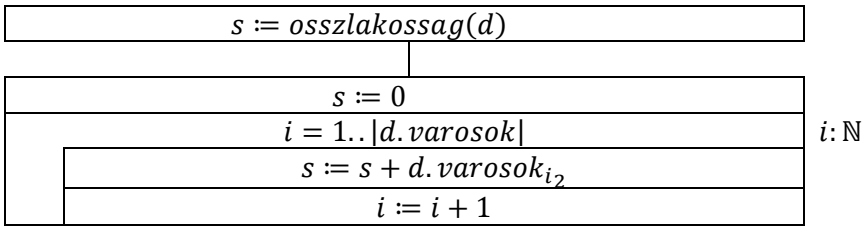
Jegyezzük meg, hogy mivel a két számlálás ciklusának „intervalluma” (teljes felsorolás) eltér a maximumkiválasztásétól (2. elemtől kezdődően a felsorolás), ezért az utóbbihoz igazodtunk és hasonlóan a maximumkiválasztáshoz, a két összegzés első lépését is kidelegáltuk a ciklus elé.

Szóval ezek miatt ilyen bitang bonyolult ez a stuki ☺

Belső tétel:

összegzés + sorozat felsorolása

H	~	$\mathbb{N}$	E	~	$String \times \mathbb{N}$
$f(e)$	~	e_2	$enor(E)$	~	$(String \times \mathbb{N})^*$
			t	~	$(d.varosok, i)$



Tehát minden i-re az i. város második mezőjét, a lakosságszámot összegezzük.

Implementáció

A programot `C++` nyelven valósítottuk meg.

Adattípusok

A feladat megfogalmazásakor megadott *lista* nevű sorozat szerepét a konzolra kiírás (azaz a `cout`) veszi át.

A városok listája egy `vector`, aminek az elemtípusa egy rekord (`struct`). Ezt a rekordot `Varos`nak nevezzük és két adattagja van: egy `nev` nevű string és egy `lakok` nevű egész szám.

Magát a felsorolót egy osztállyal valósítottuk meg, a négy művelet kódja értelemszerű, pontosan úgy működik, mint ahogy azt a nevezetes fájlfelsorolónál tanultuk, ezért itt külön nem részletezem. Ez azért lehet így, mert a $Data$-hoz, sőt a $Varos$hoz is megírtuk a *beolvasó operátor*t (és egyébként a *kiíró operátor*t is).

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

Adatok formátuma és beolvasásuk

A fájlt soronként dolgozzuk fel, a memóriában a követelményeknek megfelelően mindig egy sornyi adatot tárolunk csak.

A `Data` és `Varos` rekordok beolvasó operátorai végzik a beolvasás érdekesebbik felét:

```
istream& operator>>(istream& is, Data& d);
```

A paraméterül átadott `istream` maga a bemeneti fájl. Ebből megkísérel egy *sor*t olvasni, ha nem sikerül, akkor visszatér az `istream`mel (aminek immáron a `failbit`je bebillent igazra, hiszen nem sikerült az olvasás). Amennyiben sikerül, akkor azt egy `stringstream`mel feldolgozza, mégpedig úgy, hogy előbb kiolvassa az országnevet, majd ameddig tud, addig kiolvassa a városokat és feltölti vele a városok listáját. Minden hívásakor ki kell üríteni a városok `vector`át, különben az eddigi országok városait is megjegyezné.

```
istream& operator>>(istream& is, Varos& d)
```

A városok beolvasója pedig úgy működik, hogy megkeresi egy *lineáris keresés*sel az első számként *parse*-olható szót, és egészen eddig (*összegzés*) összefűzi amit talált, ez alkotja majd a város nevét. A linker futtatásának végére az `istream`ről lekerült minden az aktuális várossal kapcsolatban, a „current” pedig tartalmazni fogja az első számot, azaz a lélekszámot. Még elhagyjuk az utolsó szóközt a város nevéből, majd visszatérünk az `istream`mel

A bemenet formátuma *kötött*, és helyességét NEM ellenőrizzük. A program ugyanakkor MŰKÖDIK üres fájlra is, ekkor a maximumkiválasztás tétel nem fut le, hiszen sérül az EF-e. Ezen kívül értelmetlenül kitöltött fájlokra se „fagy ki” (lásd tesztesetek).

A bemenet egy sora ezt a formátumot követi:

országnév (1 szó) városnév$_1$ (valahány szó) lakosok$_1$ … városnév$_n$ lakosok$_n$

Példa:

```
Japan Szapporo 1900 Fukuoka 1400 Oszaka 2600
USA New York 8300 Seattle 600
Romania
Mexiko Ciudad de Mexico 8800 Guadalajara 1450 Monterrey 1300 Tijuana 1300
```

A kimenet

A listázás a feldolgozás során (annak ciklusában), a két összetett függvényes tételre adott válasz pedig a teljes feldolgozás után kerül kiírásra. A formátum így alakul:

```
listázás (országnév városszám párok)
összlakosságszám
max. lakosságszámú ország neve
```

A bemenetnél megadott példára:

```
Japan 3
USA 2
Romania 0
Mexiko 4
```

```
27650000
Mexiko
```

A projekt felépítése

A következő modulokat használjuk:

- *main* – a `main.cpp` forrásfájl, a program belépési pontja (`main` függvény, benne a három külső tétellel), valamint a belső tétel implementációja található itt.
- *enor* – `enor.h` fejlécállomány valamint `enor.cpp` forrásállomány. Előbbi adja meg a fájlfelsoroló típusdefinícióját, illetve a felsorolandó rekord típusát, míg utóbbi a műveletek definícióit.
- *iostream* – külső könyvtár, mely a *konzolos kommunikáció* eszközeit teszi elérhetővé.
- *fstream* – külső könyvtár, mely a *fájlkezelés*ben nélkülözhetetlen, a beolvasáshoz használjuk
- *sstream* – külső könyvtár, amelynek segítségével a bemenet egy sorát tudjuk könnyen kezelni és feldolgozni
- *cstdlib* – a program megírásához szükséges volt `atoi()` függvény lelőhelye
- *vector* – külső könyvtár, az `std::vector` típushoz

Tesztelés

Az alábbiakban megadok néhány érvényes és érvénytelen tesztesetet

- Nem létező fájl [**faf.txt**] – úgy kezeli, mintha üres lenne
- Üres fájl [**f0.txt**] – erdemény: 0 (a listázás üres string, a maximum értelmetlen, ez a 0 az összlakosságszám eredménye)
- Hibás formátumú fájl, lakosságszámok nélkül [**f2.txt**] – nullának érzékeli ezeket, illetve nem tudja meghatározni a városnevek határát, azt gondolja, minden országhoz egy város tartozik
- Hibás formátumú fájl, üres sorral [**f3.txt**] – az üres sort is megpróbálja feldolgozni, de nem sikerül neki, a városok listája üres lesz, az ország neve pedig jobb híján a memóriában maradt előző sorhoz tartozó országnév. Amúgy a többi sort jól olvassa be
- Hibás formátumú fájl, egy ország több sorba kerül [**f4.txt**] – a külön sorokat külön országként kezeli
- Hibás fájl, negatív lakosságszámmal [**f5.txt**] – logikusan működik

Egy elem:

- nem létezik hozzá város [**f1.txt**] – Ő a max, 0 város, 0 a lakosság
- egy város létezik hozzá [**f6.txt**] – Ő a max, 1 város, annak lakossága a lakosság
- több város létezik hozzá [**f7.txt**] – Ő a max, n város, az összlakosságszámuk a lakosság

Több elem:

- van köztük 0 városos [**f.txt**] – lásd a dokumentáció példája
- az első a legnagyobb lakosságú [**f8.txt**] – azt is írja ki
- nem az első a legnagyobb lakosságú [**f.txt**] – lásd a dokumentáció példája