Feladat

Madarak életének kutatásával foglalkozó szakemberek n különböző településen m különböző madárfaj előfordulását tanulmányozzák. Egy adott időszakban megszámolták, hogy az egyes településen egy madárfajnak hány egyedével találkoztak. Volt-e olyan település, ahol mindegyik madárfaj előfordult?

Megoldási terv

$$A = \{madarak \in \mathbb{N}^{n \times m}, l \in \mathbb{L}\}$$

$$Ef = (madarak = madarak' \wedge n, m \in \mathbb{N})$$

$$Uf = \big(Ef \wedge (l = \exists i \in [1, n]: MindenElofordul(i))\big)$$

ahol $MindenElofordul(i): [1..n] \to \mathbb{L}$ és

$$MindenElofordul(i) = \forall j = [1..m]: madarak[i][j] > 0$$

pesszimista lineáris keresés
$m..n \sim 1..n$ $\beta(i) \sim MindenElofordul(i)$

optimista lineáris keresés
$m..n \sim 1..m$ $\beta(i) \sim madarak[i][j] > 0$

$l, i := hamis, 1$	
$\dashv l \wedge i \leq n$	
	$l := MindenElofordul(i)$
	$i := i + 1$

$l := MindenElofordul(i)$	
$l, j := igaz, 1$	
$l \wedge i \leq m$	
	$l := madarak[i][j] > 0$
	$j := j + 1$

Implementáció

Adattípusok meghatátozása

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 megvalósulásban a városok neveit egy külön tömbben tároljuk.

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, de hibás input esetén a program ReadException hibával kilép. Az első sor a városok, és a madárfajok számát tartalmazza, szóközzel elválasztva. Ezután a városok nevei soronként. Ezután a madarak egyedeinek számai következnek, minden sorban egy-egy város madarai, az egyes egyedek számával szóközökkel elválasztva. Minden sor végén (az utolsó sor végén is) sorvége jel legyen.

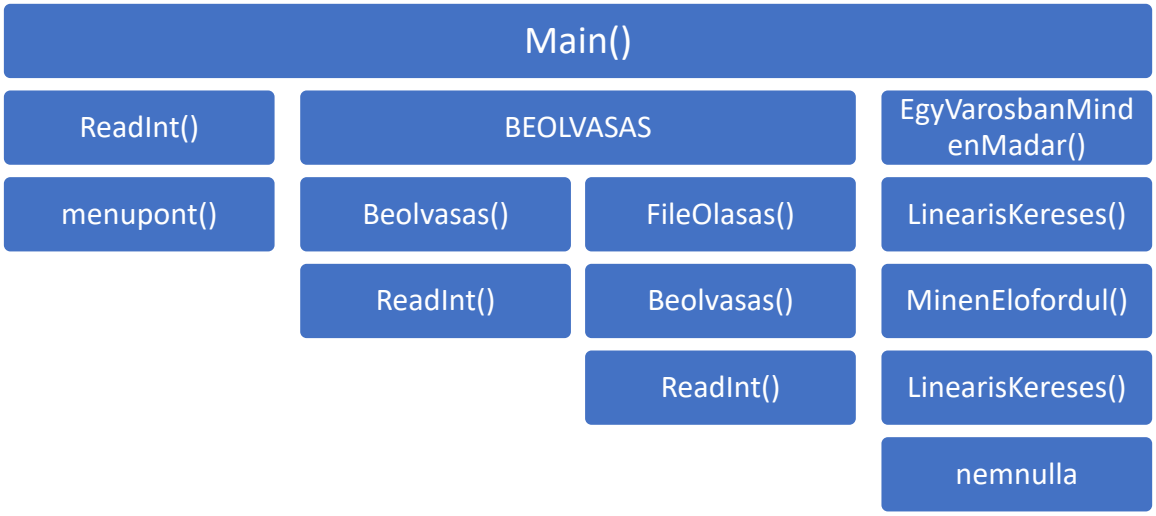
Példa:

```
3 5
Az elso varos neve
A masodik varos neve
A harmadik varos neve
0 1 0 0 2
1 1 1 2 1
3 0 0 0 4
```

Program váz

A program több állományból áll. A *read* csomag (*Read.h, Read.cpp*) tartalmazza a *ReadInt* függvényt, amely egy input streamet felhasználva egész számot olvas be, majd egy feltételnek megfelelteti, és hiba esetén újrakérdez, vagy hibát dob. Ehhez hozzátartoznak az *all* és *nat* függvények, amelyek átadhatók feltételként, és bármely, illetve a természetes számokat fogadják el. Továbbá a csomag tartalmaz egy int->string konvertáló *toString* függvényt. A *madar* csomag (*Madar.h, Madar.cpp*) tartalmazza a program logikai felépítését, és működtető függvényeit. A *tetelek* csomag (*Tetelek.h, Tetelek.cpp*) tartalmazza a programban felhasznált programozási tétel interpretációját, ami ebben az esetben csak az optimista/pesszimista lineáris keresés és eldöntés.

A ReadInt() segítségével azt a felhasználói döntést olvassuk be, hogy fájlból vagy billentyűzetről történjen-e az adatok bevitele. Ezután a választásnak megfelelő módon a Beolvasas() vagy FileOlasas() függvények segítségével beolvassuk az adatokat.



Tesztelési terv

Tesztelések a feladat specifikáció alapján (fekete doboz tesztelés)

Érvénytelen tesztesetek: a program bármilyen tesztesetet képes kezelni vagy újra kérdezéssel, vagy a hiba jelzésével input típusától (console, file) függően.

Érvényes tesztesetek: lásd alább

Tesztelések tételek alapján (szürke dobozos tesztelés)

A. Külső tétel (pesszimista lineáris keresés)
   I. Intervallum hossza
      1. A városok száma nulla
         a. t0_0_0.txt: 0 város, 0 madár – válasz: Nincs ilyen város
         b. t0_0_3.txt: 0 áros, több madár – válasz: Nincs ilyen város
      2. Egy város, több madár
         a. t1_1_3.txt: 1 város, több madár, nincs ilyen város – válasz: Nincs ilyen város
         b. t1_1_4.txt: 1 város, több madár, van ilyen város – válasz: Van ilyen város
   II. Intervallum eleje
      1. Több város, több madár, az első városban van minden madár, máshol nincs:
         a. t2_5_5.txt: – válasz: Van ilyen város
   III. Intervallum vége
      1. Több város, több madár, az utolsó városban van minden madár, máshol nincs:
         a. t3_5_5.txt: – válasz: Van ilyen város
   IV. Tételre jellemző esetek:
      1. Sehol nincs meg minden madár:
         a. t4_10_10.txt: – válasz: Nincs ilyen város
      2. Egy városban van minden madár:
         a. t4_10_10_2.txt: – válasz: Van ilyen város
      3. Minden városban van minden madár:
         a. t4_10_10_3.txt: – válasz: Van ilyen város

B. Belső tétel (optimista lineáris keresés)
   I. Intervallum hossza
      1.A madarak száma nulla
         a. t5_3_0.txt: több város, 0 madár – válasz: Van ilyen város
      2.Több város, egy madár
         a. t5_3_1.txt: több város, 1 madár, nincs ilyen város – válasz: Nincs ilyen város
         b. t5_4_1.txt: több város, 1 madár, van ilyen város – válasz: Van ilyen város

Tesztelések a megoldó kód alapján (fehér dobozos tesztelés)

1. Menü választás tesztelése (0,1,2)
2. Beolvasás mindhárom módjának tesztelése
3. Parancssorból indítás filenévvel és anélkül
4. Nem létező file megadása
5. Hibás adatok billentyűzetről (negatív mátrix méret, negatív számú madár)

A tesztesetek automatikus futtatása

A programhoz tartozik tesztelési rendszer, amely a fent látható teszteseteket futtatja. Ez a tesztesen elérhető a *main.ccp* fő programrésze előtt a *TESTRUN* makró definiálásával (*#define TESTRUN*). Ebben a módban a program újrafordításánál a preprocesszor teljesen figyelmen kívül hagyja a *main.cpp* -ben szereplő *Main()* függvényt, és a *testcases.hpp* tartalmát fogja a helyére illeszteni, amely tartalmazza a fent lévő teszteseteket a *catch.h* rendszer implamentációjával. A program lefordítása után a fenti tesztesetek automatikusan lefutnak, így tesztele a program logikai helyességét.