## Feladat

Madarak életének kutatásával foglalkozó szakemberek különböző településen különböző madárfaj előfordulását tanulmányozzák. Egy adott időszakban megszámolták, hogy az egyes településen egy madárfajnak hány egyedével találkoztak. Melyik az a madárfaj, amelyik a leggyakrabban (azaz a legtöbb településen) fordult elő?

## Bemenő adatok

A program billentyűzetről vagy fájlból is tud bemeneti adatokat fogadni. Kimenetét a konzolra () írja, melyet fájlba irányíthatunk vagy megtekinthetünk a képernyőn.

A bemeneti fájlt paraméterként kell megadni.
Ha nincs paraméter, a program automatikusan billentyűzetről kezd olvasni.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |
|  |  |  |  |  |  |

## Bemenő adatfájlok elvárt formája

A szöveges állomány formátumáról feltételezzük, hogy helyes.

### Például (bemenet/pelda.txt)

## Specifikáció

→Feltételes maximumkeresésbe ágyazott számlálás.

Azért nem az egyszerűbb maximumkiválasztás tételt választottam, mert ha minden madár előfordulása minden településen, akkor nem szeretnék megoldást visszaadni. Inkább kiírnám, hogy nincs elég madár.

### Feltételes maximumkeresés

|  |
| --- |
|  |
|  |
|  |  |
|  |  |  |
|  |  |  |
|  |  |

A nem megengedett értékadást kitranszformáljuk:

### Számlálás

|  |
| --- |
|  |
|  |  |
|  |
|  |
|  |  |
|  |  |
|  |

# Implementáció

## Adattípusok megvalósítása

Kódoláskor a vektort -ként deklaráljuk, amelynek méretei és alakban érhetők el.

A és a vektorokban tároljuk a neveket. (Lehetne tömbben is.)

### Programozási trükkök / szükségletek

C++-ban a vektorokat -tól indexeljük, ezért a tervbeli ciklus nem az , hanem a jobb oldalon nyílt intervallumot futja be.

A hármas értékadást a következőképpen oldottam meg. Először bevezettem egy nevű osztályt:

Majd a feltételes maximumkeresés végén létrehoztam ennek egy objektumát és feltöltöttem a visszatérési értékekkel:

Végül visszatértem vele:

## Függvények kapcsolódási szerkezete

Az előző beadandómhoz képest most sokkal egyszerűbb függvényszerkezettel dolgoztam:

## Tesztelési terv

### A feladat specifikációjára épülő (fekete doboz) tesztesetek:

1. Feltételes maximumkeresés, Számlálás, és Intervallumaik:
	1. Üres fájl (ures.txt)
	2. Egy-egy érték (egyertek.txt)
	3. Minden 0 (minden0.txt)
	4. Utolsó a max (utolsomax.txt)
	5. Első a max (elsomax.txt)
	6. Első 0 (elso0.txt)
	7. Utolsó 0 (utolso0.txt)
2. Különleges értékek esetei (Billentyűzetről tesztelendő):
	1. Üres sztring, mint érték (Figyelmen kívül hagyja, tovább vár a bemenetre, mintha misem történt volna.)
	2. Negatív számok (Hibaüzenetet ír ki.)
	3. Betűt tartalmazó értékek esete (Figyelmen kívül hagyja a betűt.)
	4. Speciális karakterek esete (Hibaüzenetet ír ki)

### A megoldó programra épülő (fehér doboz) tesztesetek:

1. Hibás vagy nem létező állománynév megadása. (Hasznos hibaüzenetet ír ki.)
2. Paraméterként megadott fájlnév. (Az elvárt módon működik.)
3. Ismételt futtatás kipróbálása. (Az elvárt módon működik.)
4. Olyan állomány olvasása, ahol az utolsó sort nem zárja sorvége jel. (pelda.txt)
5. Stressz-teszt, 1.000.000 értékkel. (stresszteszt.txt), (!stressz-teszt.bat)

### A és az függvények tesztelése

1. Minden 0 (minden0.txt)
2. Első 0 (elso0.txt)
3. Utolsó 0 (utolso0.txt)
4. Első a max (elsomax.txt)
5. Utolsó a max (utolsomax.txt)
6. Ezen kívül még lehetne tesztelni például a darabszámok növelésszámára (0, 1, több)

### Minden teszteset (kivéve stressz teszt) lefuttatása egyszerre: !teszt.bat

Még paraméter tesztet is raktam bele. (Nem létező fájl.123)

### Stressz teszt

Előszeretettel generálok hatalmas tesztfájlokat és csodálkozom rá a számítógépek többmagos processzorainak számítási sebességére újra és újra. Ha majd CUDA programozással foglalkozzunk, még gyorsabb lesz minden: 20 billió helyett 1000 billió számítás/másodperc.

Az elemű, soros, oszlopos méretű stressz-tesztesetet a mellékelt BEDTACI.ZIP/bemenet/Stressz teszt.xlsx Excel fájl létrehozásával és txt-ként mentésével készítettem. (A tabulátorokat lecseréltem szóközre, ami 100% CPU kihasználtság mellett is eltartott 2 percig.) A használt formula:

Annak reményében bonyolítottam két véletlenszerű szám különbségére az egyszerűbb, formulát, mert így nagyobb valószínűséggel lesz az eredmény, ami csökkenti a fájlméretet és segíti a tömörítést. ( helyett lett, tömörítve mindössze .) Az így kapott egymillió szám átlaga , míg az egyszerűbb formulával kapottaké volt. (Miért nem ? A lefelé kerekítés miatt.)

Majd átneveztem néhány száz madarat Magyarországon honos fajokra (Wikipédiáról másoltam a neveket, annyi módosítással, hogy a kis kezdőbetűket átírtam nagyra)

Legutolsó lépésként pedig népszerűsítettem a verebeket: 2 kivétellel minden településre raktam belőlük, hogy biztos nyerjenek. Az eredeti nyertes "Madár704" lett volna, 938 településsel.