Nevek feladat

Egy iskola több évre visszamenőlegesen vezetett kivonatos beiratkozási listájának sorai véletlenül összekeveredtek. A *nevek.txt* állomány ennek az összekeveredett listának egy részletét tartalmazza. Az állomány egyes soraiban sorrendben a következő adatok szerepelnek: a beiratkozás éve, annak az osztálynak a betűjele, amelybe a tanuló beiratkozott, a tanuló neve, vezetéknév keresztnév sorrendben. Ha valaki két keresztnevet használ, mindkét keresztneve fel van tüntetve. Az egyes adatokat pontosan egy szóköz választja el egymástól. A nevek ékezetmentes írással, kisbetűkkel íródtak. Az állomány legfeljebb 150 soros, a keresztnevek illetve a vezetéknevek legfeljebb 15 karakterből állnak.

Pl.:
2007 a horvath bela
2007 a kallai anasztazia nora

Készítsen programot, amely

1. beolvassa a *nevek.txt* fájl tartalmát, majd a beolvasott adatok alapján:

2. megadja, hányan használnak két keresztnevet;

3. megadja a leghosszabb nevű tanuló nevét;

4. megadja annak a tanulónak a nevét, ha van ilyen, akinek mindkét keresztneve azonos betűvel kezdődik;

5. megjeleníti, milyen évszámokat tartalmaz a lista, továbbá azt, hogy az egyes évszámokhoz hány beiratkozott tanuló tartozik;

6. összesítést készít arról, hogy az egyes években a különböző betűjelű osztályokhoz hány beiratkozott tanuló tartozik;

7. rendezett kimeneti listát készít az adatokból évek szerint növekvően rendezve *uj_nev.txt* néven!

Megjegyzések az **1.** feladathoz:

Csak számokat tartalmazó szöveges állományban a számok szóközökkel történő elválasztása a beolvasási procedúra során nem okoz gondot, ugyanis nem szoktunk egy számot számjegyeinek írása közben szóközzel megszakítani. A szóköz tehát mindenképpen a szám végét (vagy a következő kezdetét) jelenti. Erre a kiolvasó eljárások fel is vannak készítve. Szóközzel készült ezres tagolással számokat csak akkor szoktunk megjeleníteni, ha már nem akarunk velük további számításokat végezni, stringként tárolva azokat.

Ugyanakkor szavak szóközzel elválasztva már nem olvashatók be szavanként (pl. külön vezetéknév és külön keresztnév), mivel a szóköz ugyanolyan legális karakter, mint bármely más betű a szavakban, így a szóköz – a kiolvasó eljárás szempontjából – nem szakaszolja a szöveg stream-et, csak a szöveg látványát. Az egész szöveg így valójában egyetlen "szó".

A *nevek.txt* fájl tartalmának beolvasására két különböző megoldási stratégia is kínálkozik:

a.)
Belenyúlunk a txt fájlba: Word-el megnyitjuk, *szerkesztés/cseré*vel lecseréljük a szóközöket paragrafus jelekre (^p), majd mentjük MS-DOS szövegként. (ilyenkor kiderülhet pl., hogy néhol dupla szóközök vannak gépelési hiba miatt.) Ezután soronként olvassuk az állományt ***readln*** eljárással, mely sorvégjelig olvas. Az egyes sorok helyes olvasását az adott sor adattípusának megfelelő típusú változó alkalmazásával kényszerítjük ki. Minden ötödik kiolvasást azonban ellenőrizni kell: vajon új évszámot vagy második keresztnevet olvastunk-e. Az általános módszer ilyenkor az, hogy string típusú változóba olvasunk be (biztos, ami biztos), majd megkíséreljük ***val*** eljárással érvényes számmá konvertálni a stringet. Ha ez sikerül, számot olvastunk, ellenkező esetben második keresztnevet.

b.)
Első megoldásunk, bár működhet, kerülendő! Ugyanis a programozó legtöbbször nem magányosan, hanem csapatmunkában dolgozik, így programjainak bemeneti struktúráját általában nem ő szabályozza, hanem objektív külső tényezők! Ha pl. rendszeresen kap állományokat feldolgozásra, az ilyen jellegű ”kézimunka” jelentősen rontja a hatékonyságot. Éppen ezt a valós szituációt kívánják modellezni az emeltszintű érettségin, amikor *programjainkat a megadott mintaállománytól különböző állományokkal tesztelik az értékelés során!* Meg kell tehát próbálnunk a karakter stream-ként, soronként beolvasott állományt egy szóközig olvasó eljárással ”szavakra” tördelni. A ”szavakat” később, ha szükséges, típus konverziónak vethetjük alá:

```
eljárás tördel(sor);
    h = hossz(sor); i = 1; j = 1;
    ciklus amíg sor≠''
        szó='';
        ciklus amíg (j≤h és első(sor) ≠' ')
            szó=szó+első(sor);
            sor = elsőnélküli(sor)
            j = j+1;
        ciklus vége
        szavak(i)=szó;
        i = i+1;
        j = j+1;
        sor = elsőnélküli(sor)
    ciklus vége
eljárás vége
```

Így megírt eljárásunk érzékeny az egyébként szabálytalanul halmozott felesleges szóközökre, az úgynevezett whitespace-ekre. Kis **módosítással** hibatűrővé tehetjük eljárásunkat:

```
eljárás tördel(sor)2;
    h = hossz(sor); i = 1; j = 1;
    ciklus amíg sor≠''
        szó='';
        ciklus amíg (j≤h és első(sor) ≠ ' ')
            szó=szó+első(sor);
            sor = elsőnélküli(sor)
            j = j+1;
        ciklus vége
        ha  szó≠''  akkor
        szavak(i)=szó;
        i = i+1;
        j = j+1;
        sor = elsőnélküli(sor)
        elágazás vége;
    ciklus vége
eljárás vége
```

További finomítást végzünk az eljáráson. A kódsorokat elemezve látjuk, hogy a ***szavak(i)*** tömb globális változó, melyet nem szabványos módon hívunk meg az eljárás belsejéből. (a ***h***, ***i***, ***j***, ***szó*** változók nyilván lokálisak) Erre ugyan a legtöbb magas szintű nyelv lehetőséget teremt, de kerülendő. A változókhoz az eljárás paraméterlistáján (interface-én) keresztül biztonságos a hozzáférés. Az sem derül ki, hogy hány szót tudunk, vagy akarunk beolvasni egy sorból, hiszen nem ismerjük a ***szavak(i)*** tömb dimenzióját, ezért jó lenne, ha tudnánk szabályozni a beolvasni kívánt szavak számát is.

```
eljárás tördel(sor, szavak, n)3;
    h = hossz(sor); i = 1; j = 1;
    ciklus amíg sor≠''
        szó='';
        ciklus amíg (j≤h és első(sor) ≠ ' ')
            szó=szó+első(sor);
            sor = elsőnélküli(sor)
            j = j+1;
        ciklus vége
        ha  szó≠''  és  i≤n akkor
        szavak(i)=szó;
        i = i+1;
        j = j+1;
        sor = elsőnélküli(sor)
        elágazás vége;
    ciklus vége
eljárás vége
```