Programozás 2

– 3. zárthelyi dolgozat (C#), pót-ZH –

2022. január 5., 10.00

1. feladat – zárójelek (1 pont)

Adott az `input.txt` állomány, amelynek minden sora egy-egy zárójel szekvenciát tartalmaz.

Vegyük a következő kisebb példát:

```
(())((((()))))
(()()(()()))
))(((()
((()))(())
```

A fenti példában az első két sor szabályosnak mondható, mivel minden nyitó zárójelnek megvan a csukó zárójel párja és a zárójelezés matematikailag helyes.

A 3. sorban levő szekvencia **nem** szabályos, mivel rögtön egy csukó zárójellel kezdődik, hiába szerepel ugyanannnyi nyitó és csukó zárójel.

A 4. sor szekvenciája **sem** szabályos, mivel ott az egyik nyitó zárójelnek nincs párja.

Biztosak lehetünk benne, hogy az input fájl csak nyitó és csukó kerek zárójeleket tartalmaz, amelyek nincsenek elválasztva semmilyen karakterrel sem egymástól.

Dolgozzuk fel az állományt és írassuk ki a konzolra, hogy hány darab szabályos szekvencia van a fájlban.

Futási példa:

```
$ dotnet run
9
```

9 helyett természetesen a helyes eredményt kell kiírni.

Figyelem! A megoldás során **NE** használjon saját *extension* metódust!

2. feladat – zárójelek újratöltve (1 pont)

A feladat ugyanaz, mint az előző feladatban. Az input fájl sem változott.

Most viszont a feladatot *extension* metódus segítségével oldjuk meg! Ehhez a `string` osztályt terjesszük ki egy `IsValid()` nevű metódussal, melyet a következőképpen akarunk használni:

```
Console.WriteLine("(())".IsValid());    // true

Console.WriteLine("))((".IsValid());    // false
```

A feladat megoldása során használja fel ezt az extension metódust!

Futási példa:

```
$ dotnet run
9
```

9 helyett természetesen a helyes eredményt kell kiírni.

3. feladat – átlagéletkor (1 pont)

Adott a `people.csv` állomány, amelynek minden sora egy-egy személy alapadatait tartalmazza a következő módon:

```
first_name;last_name;ID
```

ahol:

- `first_name`: keresztnév
- `last_name`: vezetéknév
- `ID`: egyedi azonosító

Tekintsük az állomány első sorát:

```
Erik;Nicolas;511-25-1688
```

Tegyük fel, hogy az egyedi azonosító közepén (a két '`-`' jel között) az adott személy életkora szerepel. Ebben a példában Erik Nicolas 25 éves.

Írjunk egy programot, ami elemzi az input fájlt, majd kiír róla egy kis statisztikát az alábbi formában:

Példa:

```
$ dotnet run
Legfiatalabb személy:
Erik Cox, 2 éves
---
Legidősebb személy:
Brian Hamming, 96 éves
---
A fájlban szereplő személyek átlagos életkora: 25.78 év.
```

Ha a legfiatalabb / legidősebb személyből több is van, akkor csak annak az adatait írjuk ki, aki először szerepel az állományban!

Az átlagéletkor esetén tizedespont, ill. tizedesvessző is használható. Nem jár pontlevonás, ha az átlagéletkort nem sikerül két tizedesjegy pontossággal kiíratni.

4. feladat – kutyák és gazdáik (1 pont)

Vegyük újra az előző feladatban már látott `people.csv` állományt, amelynek minden sora egy-egy személy alapadatait tartalmazza a következő módon:

```
first_name;last_name;ID
```

ahol:

- `first_name`: keresztnév
- `last_name`: vezetéknév
- `ID`: személyi igazolvány azonosítója

Továbbá adott egy `dogs.csv` állomány is, amely kutyák adatait tartalmazza, soronként egyet, a következő módon:

```
name;breed;age;owner_ID
```

ahol:

- `name`: név
- `breed`: fajta
- `age`: életkor
- `owner_ID`: tulajdonos személyi igazolvány azonosítója, amely megfelel a `people.csv`-ből az utolsó oszlopnak, tehát az `owner_ID` mező *rámutat* a `people.csv` állományban szereplő valamelyik személyre

Egy kutya csak egy személyhez tartozhat, de egy személynek akár több kutyája is lehet.

Hozzunk létre két osztályt: egy `Person` és egy `Dog` osztályt. Mindkettőt külön állományokban helyezzük el. A `Dog` osztály tárolja a fentebb felsorolt attribútumokat (az `owner_ID` mezőt tároljuk sztringként). A `Person` osztály tárolja a kereszt- és vezetéknevet, valamint a személyi igazolvány azonosítóját (sztringként). Továbbá legyen egy `dogs` nevű attribútuma, amely `Dog` objektumok egy listája legyen.

Dolgozzuk fel először a `people.csv` állományt és hozzuk létre a főprogramban a személyek egy listáját. Egyelőre a `dogs` lista minden `Person` objektumban lehet üres. Ezután dolgozzuk fel a `dogs.csv` állományt is, s minden egyes sorhoz hozzunk létre egy `Dog` objektumot, majd a személyek listájában keressük meg az adott kutya tulajdonosát és adjuk hozzá a kutyát a tulajdonos példány `dogs` listájához.

Amikor mindezzel elkészültünk, akkor keressük meg azt a kutyatulajdonost, aki a legtöbb kutyával rendelkezik és írjuk ki a teljes nevét a konzolra. Informatív módon tüntessük fel a kutyáinak a számát is az alábbi módon.

Futási példa:

```
$ dotnet run
Alan Cox has 12 dogs
```

Ez természetesen csak minta, nem pedig a helyes kimenet.