Tavalyi ZH

Sorok gyakorisága

Oldd meg az alábbi feladatot. Az elkészült `.c` és `.h` fájlokat (és csak ezeket a fájlokat) egyetlen **zip** fájlba csomagolva töltsd fel!

Alapfeladat (25 pont)

Készíts el egy programot szabványos C eszközökkel, amely az inputon szereplő sorok gyakoriságát számolja meg. A program minden különböző sort egyetlenegyszer ír ki, elé írva, hogy hányszor fordult elő az a sor az inputban. Azaz, ha az input például:

```
abcdef
xyz
abcdefghi
xyz
xyzwu
abcdef
xyz
```

...akkor az output:

```
2 abcdef
3 xyz
1 abcdefghi
1 xyzwu
```

A program parancssori paraméterként kapjon egy fájlt, amelyet megnyitva beolvassa a benne szereplő sorokat. Feltételezhetjük, hogy a sorok mindegyike rövidebb 1023 karakternél, és az egész input elfér a memóriában. Ha a fájl megnyitása sikertelen, a program írjon ki hibaüzenetet, majd fejezze be a futást.

Hozz létre egy struktúrát, amelyben egy sort és előfordulásai számát tárolod! Minden új sort tárolj egy dinamikus tömb elemeként, a struktúra felhasználásával. A tömb kezdőmérete legyen 4, majd ha megtelt a tömb, duplázzuk meg az aktuális méretet.

Modularizálás (4 pont)

Bontsd fordítási egységekre a programot! A `main` függvény szerepeljen másik fordítási egységben, mint a többi függvény. A `main`-től különböző függvényeket tartalmazó fordítási egységhez készíts header fájlt is. Ne felejts el header guardot használni!

Beolvasás több fájlból (5 pont)

A program legyen képes több fájlt feldolgozni parancssori paraméterekként. Ebben az esetben minden fájlban külön-külön számoljuk a sorokat. Ha valamely fájl megnyitása sikertelen, a program írjon ki hibaüzenetet, majd a következő fájllal folytassuk a végrehajtást.

Beolvasás a standard inputról (4 pont)

Ha a programnak nincs parancssori paramétere, a program legyen képes a standard inputról beolvasni sorokat (EOF-ig).

(Tipp: az EOF esemény Unix rendszeren a Ctrl+D, Windowson a Ctrl+Z paranccsal váltható ki.)

A kimenet rendezése (7 pont)

A sorok jelenjenek meg a kimeneten gyakoriságuk szerint csökkenő sorrendben!

```
3 xyz
2 abcdef
1 abcdefghi
1 xyzwu
```

Kis- és nagybetűk (5 pont)

Tekintsük a(z angol) kisbetűket és nagybetűket egyformának. Az outputon az elsőnek megjelenő formátum jelenjen meg, azaz, ha az input:

```
abcdEf
xyZ
aBcdefghi
xYZ
xyzwu
abcdef
Xyz
```

akkor az output:

```
3 xyZ
2 abcdEf
1 aBcdefghi
1 xyzwu
```

Elvárások a programmal szemben

- A nem forduló kód automatikusan 0 pontot ér. (Természetesen ez csak a legutoljára feltöltött megoldásra vonatkozik.)
- Ne használj globális változókat!
- Logikusan tagold a megoldást. A megoldás részeit külön függvényekben valósítsd meg.
- Ügyelj, hogy ne legyen a programban memóriaszivárgás!
- Kerüld a nem definiált viselkedést okozó utasításokat!

Tanácsok

- Ne feledkezzünk meg a dinamikus memóriafoglalás sikerességének ellenőrzéséről, és a foglalt memória felszabadításáról! Teszteld a programot `valgrind`dal, hogy felderítsd az esetleges memóriaszivárgást.
- Amennyiben a beolvasáshoz az `fgets` függvényt használod, ne feledd, hogy ha a sor rövidebb, mint a puffer maximális hossza, a függvény beolvassa a sortörést is, ami a kiírásnál hibás eredményhez vezethet.

Megoldások

- **best.zip (50 pont)**
- **worst.zip (3 pont)**