11

AWK
tömbök, függvények

zh

- felépítése
  - három feladat (6-6-8)
- anyagrész: minden, ami a félév során elhangzott, de főleg az első zh óta elhangzott anyag (reguláris kifejezése, awk)
- amit használhatsz (ne bízd el magad, ezek zh-n való lapozgatása nem lesz elég, ahhoz hogy megold a feladatokat): Griechisch Erika gyakorlati jegyzete

Vezérlési szerkezetek

- if (FELTÉTEL) {
  UTASÍTÁS
  } else if (FELTÉTEL) {
  UTASÍTÁS
  } else {
  UTASÍTÁS
  }
- while (FELTÉTEL) {}
- do {} while (FELTÉTEL)

Vezérlési szerkezetek

- for (KIF1; KIF2; KIF3;) {}
- for (INDEX in NÉV) {}
  - nem sorrendben járja be a tömböt!
  - csak ha nem számít a sorrend, pl: halmaz
- break, continue
- exit
  - (ha nem az END-ben van, akkor az END szekció még lefut!!)

for példa

- az esetek nagy részében az alábbi két használat gyakori:

```
for (i=1;i<=NF;i++){
   print($i)
}
for (tombindex in tomb){
    print(tomb[tombindex])
}
```

Feladatok

1) A szamok.csv-ből azokat a sorokat írjuk ki és számoljuk meg, ahol az elemek összege 700000-nél nagyobb
2) Írjuk ki azokat a sorokat a bocsanat.txt -ből amelyekben megtalálható a „nem” szó (reguláris kifejezés használatával)
3) Írjuk ki az összes szót a bocsanat.txt -ből
4) Írjuk ki az 10 karakternél hosszabb szavakat a bocsanat.txt -ből

Tömb - Asszociatív tömb

- Dinamikus indexelés, szöveges indexekkel:
  - nem kell előre megadni a méretet
  - tetszőleges szöveges indexszel használható
  - értékek típusában sem kell egyeznie (tömbön belül lehet szöveg és szám is)
  - ha olyan indexre hivatkozunk, ami nem létezik, akkor 0 vagy ”” lesz az eredmény
- NÉV[INDEX] = ÉRTÉK
- delete NÉV[INDEX] vagy delete NÉV

Tömb - feladat

- Számoljuk ki bocsanat.txt-ben lévő szavak előfordulását, majd írjuk ki
  - figyeljünk oda arra, hogy a különböző írásjelek miatt egy szó ne szerepeljen többször → reguláris kifejezés segítségével adjuk meg az elválasztó jeleket
  - üres szó előfordulása nem érdekel → töröljük a tömbünkből

Beépített szöveges függvények

- getline valtozo; – beolvas a STDIN-ről
- tolower(str), toupper(str) – kisbetűs/nagybetűs változata az str-nek
- length(str) – szöveg hossza
- index(str, substr) – a substr-et keresi a str-ben és visszatér a pozíciójával
- split(str, array, delim) – a delim mentén feldarabolja str-t és az array tömbbe tárolja el a darabokat
- substr(str, start_index) – a str start_index pozíciójától kezdődő részt adja vissza
- substr(str, start_index, len) – a str start_index pozíciójától kezdődve len hosszú részt ad vissza

Feladatok

- Egészítsük ki az előző feladatot az alábbiakkal: hagyja figyelmen kívül a kis/nagy betűket
- Írjuk ki azokat a sorokat amelyekben megtalálható a „nem” szó (reguláris kifejezés használata nélkül) az index függvény segítségével.
- A split.txt-ből határozzuk meg azokat a sorokat, amelyeknél 2014. 10. 21-nél újabbak

Beépített numerikus függvények

- int(x) - egészre vágás (simán elhagyjuk a tizedes jegyeket)
- sqrt(x) - négyzetgyök
- exp(x) - (e ^ x)
- log(x) - természetes alapú logaritmus
- sin(x), cos(x) - sin, cos radiánon értelmezve
- rand() - 0 és 1 közötti véletlen számot ad

  Az intervallum [0,1). Tehát ha pl.: indexet akarunk vele dobni, akkor hozzá kell adnunk 1-et!