Programozás Alapjai 5. ZH

30. feladatsor

Szoftverfejlesztés Tanszék

2023, Ősz

Feladat Töltsd le a bíróról a `minta.zip` állományt, majd tömörítsd ki! A `feladat.c` fájlban megtalálod a feladatok megoldás-kezdeményeit. Bővítsd ezt az alább olvasható feladatok alapján! Lehetőség szerint ellenőrizd megoldásod, majd töltsd fel a `feladat.c` fájlt a bíróra!

Kiértékelés A bíró lefordítja a programot, majd lefuttatja azt a feladat pontszámának megfelelő számú tesztesettel. Egy teszteset egy bemenet-kimenet pár, amely a megfelelő feladathoz készült. A teszteset akkor helyes, ha az adott bemenethez tartozó kimenet **minden egyes karaktere** megegyezik az előre eltárolt referencia kimenettel. *További feltételek: a program futása nem tarthat tovább 5 másodpercnél, egyszerre nem fogyaszthat többet 16 MiB memóriánál és nem történhet futási hiba (pl. illetéktelen memória hozzáférés).*

Ellenőrzés Feltöltés előtt érdemes ellenőrizni a megoldásod.

1. **Fordítás** Ellenőrizd, hogy a programod lefordul-e! A bíró a `gcc -O2 -static -o feladat feladat.c` paranccsal fordít, érdemes ezt használni. A `-Wall` kapcsoló is hasznos lehet.
2. **Példa tesztesetek** Ellenőrizd, hogy a programod helyesen működik-e! A `minta.zip` tartalmaz a bíró által futtatott tesztesetek közül feladatonként egyet-egyet. Az első feladat teszteléséhez másold a programod mellé az `ex1.be` fájlt `be.txt` néven, futtasd le a programod, majd az így kapott `ki.txt` tartalmát hasonlítsd össze az `ex1.ki` fájlban található referencia kimenettel.
3. **Extra tesztesetek** Ellenőrizd a programod működését további példák segítségével! Néhány további teszteset is elérhető, de ezek csupán ellenőrzésre használhatóak, a bíró nem futtatja őket. Ezek használatához futtasd a programod a `-t` vagy `-test` kapcsolóval, például a `./feladat -test` paranccsal. Csak az első feladat teszteléséhez futtasd a programod a `./feladat -t 1` paranccsal.

1. feladat (5 pont)
Írj egy `teglalap` nevű struktúrát, ami az `a` és `b` nevű `float` típusú mezőiben egy téglalap két oldalát tárolja. Írj egy `compare` nevű függvényt, ami két ilyen téglalapot kap paraméterül, és a nagyobb területűvel tér vissza. Amennyiben a két téglalap területe egyforma, akkor a függvény az első paraméterben kapott téglalapot adja vissza. A téglalap területe: ab.

```
teglalap compare(teglalap t1, teglalap t2);
```

2. feladat (5 pont)
Egy túraútvonalon több ellenőrzőpont van, amiknek egyenként ismert a magasságkülönbsége a kiindulóponthoz képest. Ezek a pontok az útvonalat szakaszokra bontják. A feladatunk meghatározni az útvonal szintemelkedését, ami úgy számolható, hogy a felfelé vezető szakaszok két végpontjának magasságkülönbségeit összeadjuk. A lefelé vezető szakaszokkal nem kell foglalkozni. A függvény első paramétere egy tömb, amiben az ellenőrzőpontok magasságait kapjuk, olyan sorrendben, ahogy követik egymást. A második paraméter egy egész szám, ami a tömb méretét adja meg. A kiindulópont magassága 0. A függvény visszatérési értéke az útvonal szintemelkedése legyen.

```
int szintemelkedes(int magassagok[], int pontok);
```