Java függvények és tömbök

Készítette: Goldschmidt Balázs, BME IIT, 2019.

1 Prímek keresése

Írjunk alkalmazást, amely kiírja az első 100 prímszámot!

a) készítsünk függvényt (*osztokSzama*), amely megszámolja, hogy egy paraméterként kapott egész számnak hány darab osztója van (az 1-et és saját magát is beleértve)!
b) az előző függvény felhasználásával készítsünk függvényt (*prim_e*), amely egy paraméterként kapott számról eldönti, hogy prím-e (max. 2 db osztója van, az 1 és saját maga). A visszatérési érték típusát ennek megfelelően válasszuk meg!
c) a főprogramban egy ciklusban vegyük sorra a számokat 2-től, és amelyik prím, írjuk ki. A 100. kiírt szám után a ciklus álljon le!

2 Tömbök feltöltése

Tömbök (nem listák!) használatával írjunk programot, amely beolvas 10 egész számot és kiírja az átlagnál nagyobbakat!

3 Listák feltöltése

Lista (*ArrayList*) használatával írjunk programot, amely beolvas 10 egész számot és kiírja az átlagnál nagyobbakat!

4 Primitív paraméterek módosítása

A feladatban a cél az, hogy kipróbáljuk, a függvények primitív típusú paramétereinek változtatása hogyan hat vissza a hívó változóira.

a) Írjunk függvényt (*ki2*), amely két egész paraméterrel rendelkezik! A függvény írja ki a két paraméter értékét a szabványos kimenetre!
b) A főprogramban hozzunk létre 2 egész változót, inicializáljuk őket és írassuk ki az értéküket a *ki2* függvény segítségével!
c) Írjunk függvényt (*elo2*), amely két egész paraméterrel rendelkezik! A függvény változtassa meg a paraméterek előjelét, és a *ki2* függvény segítségével írja ki őket!
d) A főprogram ezután a két változóval hívja meg sorban az *elo2* és a *ki2* függvényt!

Mit tapasztalunk? Milyen számokat írt ki a *ki2* függvény
(1) az *elo2* meghívása előtt,
(2) az *elo2*-ből hívva és
(3) utána a főprogramból harmadjára is meghívva?

Írjuk a forráskódba komment formájában, mi a magyarázata a kimenetnek, amit látunk!

5 Tömb/lista típusú paraméterek módosítása

A feladatban a cél az, hogy kipróbáljuk, a függvények összetett (tömb, objektum referencia) típusú paramétereinek változtatása hogyan hat vissza a hívó változóira.

a) Írjunk függvényt (*tombKi*), amely egy paraméterül kapott egész tömb tartalmát kiírja a szabványos kimenetre!
b) Írjunk függvényt (*tombPoz*), amely egy paraméterként kapott egész tömb elemein végigmenve a negatív számokat lecseréli az abszolút-értékükre! Pl. az [1,-2,3,-4,0,-3] tartalmú tömbben az új tartalom legyen [1,2,3,4,0,3]! Az eredményt írassuk ki a *tombKi* függvény segítségével!
c) A főprogramban olvassunk be 10 egész számot egy tömbbe, majd hívjuk meg rendre a *tombKi*, *tombPoz* és *tombKi* függvényeket! Mit tapasztalunk? Miért?

d) *Szorgalmi feladat*: az (a)-(c) pontokat ismételjük meg *ArrayList* típus alkalmazásával is!

Mit tapasztalunk? Milyen számokat írt ki a *tombKi* függvény
(1) az *tombPoz* meghívása előtt,
(2) az *tombPoz*-ból hívva és
(3) utána a főprogramból harmadjára is meghívva?

6 Tömbök/listák rendezése

a) Írjunk függvényt, amely a kedvenc rendező-algoritmusunkkal rendezi egy valósokat (double) tartalmazó tömb vagy lista tartalmát! Ha nincs kedvencünk, implementáljuk a kiválasztásos rendezést![1]
b) Írjunk programot, amely beolvas 10 valós számot és (az előző metódust használva) rendezve kiírja az értékeket a szabványos kimenetre!

A feladat megoldásához **ne használjunk** szabványos könyvtárban elérhető rendező metódust (pl. *ArrayList.sort*, *Collections.sort*, *Arrays.sort* stb.)!

7 Rekurzív függvények

a) Írjunk rekurzív függvényt, amely visszaadja egy paraméterként kapott egész szám faktoriálisát!
b) Módosítsuk az előző függvényt úgy, hogy ha az eredmény kilépne az ábrázolási tartományból, 0-t adunk vissza! A legnagyobb ábrázolható számot *int*-ek esetén az *Integer.MAX_VALUE*, *long* típus esetén a *Long.MAX_VALUE* stb. adja vissza.[2]
c) Írjunk rekurzív függvényt, amely visszaadja az *n.* fibonacci-számot! A függvény paramétere legyen *n*.

[1] A kiválasztásos rendezés alapötlete az, hogy kiválasztjuk a rendezendő tömb legkisebb elemét, és kicseréljük a tömb legelső elemével. Ezzel a tömb első eleme megkapta a végső értékét, és a feladat egyszerűsödött a tömb maradékának rendezésére. Az algoritmust addig ismételjük a maradék tömbön, amíg csak egy elem marad.
[2] Tipp: az *(x*y>M)* kifejezést átalakíthatjuk *(x>M/y)*-ra!

8 Hanoi tornyai

Implementáljuk Java-ban a Hanoi tornyai c. klasszikus feladat megoldását! A feladat, hogy a program kiírja egy N koronggal rendelkező kezdőtorony esetén a korongok átrakásának lépéseit, vagyis hogy az egyes lépésekben melyik oszlopról melyikre kell korongot helyezni.

A feladat már a múlt félévi *Programozás alapjai c.* tárgyban is szerepelt az alábbi leírással:

> *[Három, korongokat tartó rudunk van (A,B és C).] A korongokat át kell tenni az első rúdról a harmadikra, de úgy, hogy 1) egyszerre csak egy korongot mozgathatunk, 2) kisebb korongra nagyobbat nem tehetünk. (A középső oszlop ideiglenes tárolónak használható.) Négy korong esetén ez a lépéssorozat adja a megoldást: A→B, A→C, B→C, A→B, C→A, C→B, A→B, A→C, B→C, B→A, C→A, B→C, A→B, A→C, B→C. A látszólag bonyolult probléma rekurzív megoldása pár soros.*
>
> *Ötlet: Rakjunk félre n-1 korongot... Akkor az alsó korong mozgatható! Ha szeretnénk* honnan*,* hova *pakolni a korongokat a* seged *oszlop használatával, a lépések:*
>
> *Varázsoljunk n-1 korongot a kiindulási (*honnan*) oszlopról a segédoszlopra. Eközben a cél, „hova" oszlop lehet az ideiglenes tároló.*
>
> *Ha ezt megoldottuk, akkor a legalsó korongot csak át kell rakni.*
>
> *És az átrakott legalsó, legnagyobb korongra a félretett n-1 korongot varázsoljuk. Vagyis a segédoszlopról (mert oda tettük őket félre) a céloszlopra (végleges helyükre), közben a kiindulási oszlop (honnan) lehet az ideiglenes tároló.*
>
> *Tehát n-1 korongot varázsolunk, 1-et mozgatunk, végül megint n-1-et varázsolunk. Mit jelent a varázslat? Hogy n-1 korongot helyezünk át; ott viszont ugyanazt kell majd csinálni, mint amit itt kellett. Innen jön a rekurzió.*

Szorgalmi feladat: a kiíratás úgy történjen, hogy mindig kiírjuk, hogy az egyes oszlopokon mely korongok szerepelnek. Ehhez használjunk oszloponként egy-egy tömböt vagy listát, a korongokat jelképezze egy-egy egész szám.

Lehetőség szerint egy korong levétele és felrakása egy-egy külön függvénnyel valósuljon meg:

- a levevő függvény paramétere a torony, visszatérési értéke a levett korong értéke
- a felrakó függvény paramétere a torony és a felrakandó korong értéke