Elektrotehnički fakultet
Banja Luka

Programski jezici 2

– 28.06.2023. –

1. **(30)** Potrebno je implementirati različite vrste Lego kockica koje imaju sljedeće zajedniče osobine: površina, obim i boja. Lego kockice mogu da budu "građene" i "rastavljanje", kao i da imaju mogućnost rotacije. Potrebno je implementirati tri vrste Lego kockica: pravougaona, kvadratna i cilindrična. Pravougaona i kvadratna kockica imaju osobine "građenje" i "rastavljanje". Dodatno, pravougaona kockica može i da se rotira. Pravougaona kockica ima dodatne atribute dužinu i širinu, kvadratna samo dužinu, a cilindrična kockica ima poluprečnik. Potrebno je implementirati simulaciju postavljanja kockica na ploču, čije dimenzije se prosljeđuju kao argumenti komandne linije. Broj redova i kolona ne smije biti veći od 15 i manji od 3.

   Simulacija postavljanja kockica počinje tako što se kreira po 20 kockica svakog od oblika i sve kockice postave u jednu zajedničku listu i izmiješaju, a zatim se redom postavljaju na matricu. Dužine generisati kao slučajan cijeli broj vrijednosti između 2 i 7. Prilikom postavljanja na matricu se upisuju slova K za kvadratnu, P za provaougaonu i C za cilindričnu kockicu, u odgovarajućem obliku po dimenzijama same kockice. Prilikom postavljanja C, smatrati da se cilindrični oblik prikazuje kao kvadrat čija je dužina stranica jednaka poluprečniku. Voditi računa o tome da dimenzije kocke prate dimenzije matrice - ukoliko je npr. u pitanju kvadrat dimenzija 4x4, on "zauzima" toliko prostora na matrici - četiri kolone i pruža se kroz četiri reda. Zbog pojednostavljenja implementacije, pratiti dužinu matrice (broj kolona u jednom redu), a kada se dođe do kraja prvog reda, sljedeći element postavljati u prvom sljedećem slobodnom redu po istom principu. Ukoliko prilikom postavljanja pravougaonika nema dovoljno mjesta u dužini, pravougaonik rotirati, pa pokušati postaviti na taj način. Prilikom svakog postavljanja na konzoli ispisivati šta se postavlja i koliki su površina i obim te kockice. Na kraju simulacije prikazati izgled matrice. U slučaju da za neke kockice nije bilo mjesta na matrici, jer je bila popunjena, prikazati na kraju simulacije i broj preostalih kockica, kao i procenat uspješno postavljenih kockica.

2. **(20)** Napisati simulaciju takmičenja niti. Broj niti koj će biti se zadaje putem argumenata komandne linije i veći je od nula. Pored tog broja kao argument komandna linije se unosi još jedan broj koji mora biti veći od nula i naziv tekstualne datoteke koja mora biti kreirana ako ne postoji. Nakon toga se kreira n niti. Niti čitaju iz tekstualne datoteke trenutno upisani broj (ukoliko broj ne postoji inicijalni broj je nula), te u 30% slučajeva umanjuju vrijednost broja za jedan, a u 70% slučajeva uvećavaju vrijednost broja za jedan. Nakon toga, nit novu vrijednost upisuje u datoteku. Program se izvršava sve do trenutka dok broj koji je učitan iz datoteke nije jednak broju koji je proslijeđen kao argument komandne linije.

3. **(20)** Napisati interfejs koji ima metode *Double getValue(), String getName() i Status getStatus()*. Status je enumeracija sa vrijednostima: *NEW, PROCESSING, DONE.* Napraviti 3 klase koje implementiraju ovaj interfejs. Nazive klasa odrediti proizvoljno. Napraviti tri niza objekata ovih klasa sa po 10 objekata. Statuse postaviti naizmjenično. Napisati generičku metodu koja u proslijeđenih *n* nizova grupiše podatke na način da

ostanu samo jedinstveni *name* atributi, sa sumiranim *value* vrijednostima. Metoda vraća jedan niz, a prima argument Status po kojem filtrira podatke iz ulaznih nizova. Iskoristiti ovu metodu da sumirate tri niza u jedan. Primjer:

A = [{name:'pj2', value: 100, status: NEW}, {name:'java', value: 20, status: PROCESSING},...]
B = [{name:'ispit', value: 100, status: DONE}, {name: 'pj2', value: 200, status: NEW},...]
REZ = [{name: 'pj2', value: 300}, {name: 'java', value: 20}, {name: 'ispit', value: 100}, …]

Vrijeme za rad: 180 minuta