Zadaci za samostalno vježbanje 10.

Svi zadaci dati ovdje su takvi da se mogu uraditi korištenjem isključivo gradiva prvih deset predavanja i ranije stečenog predznanja na predmetu "Osnove računarstva". Tarabom (#) su označeni zadaci koji su u prethodnim generacijama bili zadaci za zadaću. S obzirom da upotrebljivi zadaci sa klasama zahtijevaju koncepte koji se obrađuju na predavanjima koja slijede iza Predavanja 10, za sada je broj zadataka za samostalni rad još uvijek relativno mali.

1. Proširite klasu "`Sat`" iz Zadatka 4 sa Laboratorijskih vježbi 9 sa tri konstruktora. Konstruktor bez parametara kreira objekat tipa "`Sat`" inicijaliziran na vrijeme "00:00:00". Konstruktor sa jednim parametrom prima kao parametar broj sekundi i kreira objekat tipa "`Sat`" koji čuva vrijeme koje odgovara zadanom broju sekundi razloženom na sate, minute i sekunde (npr. ako se kao parametar zada broj 5000, vrijeme treba inicijalizirati na "01:23:20". Treba dozvoliti da se ovaj konstruktor koristi za automatsku pretvorbu cjelobrojnih vrijednosti u objekte tipa "`Sat`". Konstruktor sa tri parametra kreira objekat tipa "`Sat`" koji čuva zadani broj stepeni, minuta i sekundi. Prepravite i prateći testni program sa ciljem da testirate i napisane konstruktore i uvedenu automatsku pretvorbu tipova.

2. Proširite klasu "`Sat`" iz Zadatka 3 sa Laboratorijskih vježbi 9 na isti način kao u prethodnom zadatku.

3. Proširite klasu "`Robot`" iz Zadatka 1 iz Zadataka za samostalno vježbanje 9 sa tri konstruktora. Konstruktor bez parametara kreira robota koji se nalazi u koordinatnom početku i gleda u pozitivnom smjeru x-ose. Konstruktor sa jednim parametrom kreira robota koji se nalazi u koordinatnom početku i gleda u pravcu koji je određen parametrom. Pri tome treba zabraniti da se ovaj konstruktor koristi za automatsku konverziju vrijednosti tipa "`Pravci`" u objekte tipa "`Robot`". Konstruktor sa tri parametra kreira robota na zadanoj poziciji i koji gleda u zadanom pravcu, pri čemu se zadavanje vrši putem parametara. Prepravite i prateći testni program sa ciljem da testirate i napisane konstruktore.

4. Proširite klasu "`Robot`" iz Zadatka 2 iz Zadataka za samostalno vježbanje 9 na analogan način kao i u prethodnom zadatku. Pri tome, treba onemogućiti da se konstruktor sa jednim parametrom koristi za automatsku konverziju realnih brojeva u objekte tipa "`Robot`".

5. Proširite klasu "`Tacka`" iz Zadatka 3 iz Zadataka za samostalno vježbanje 9 sa dva konstruktora. Konstruktor bez parametara kreira tačku koja se nalazi u koordinatnom početku. Konstruktor sa tri parametra kreira tačku na zadanoj poziciji. Ukoliko treći parametar ima vrijednost "`false`", tada prva dva parametra predstavljaju Dekartove koordinate tačke, a ukoliko treći parametar ima vrijednost "`true`", tada prva dva parametra predstavljaju polarne koordinate tačke. Treći parametar treba da ima podrazumijevanu vrijednost "`false`", tako da se ovaj konstruktor može koristiti i sa dva parametra (pri čemu on tada kreira tačku na osnovu Dekartovih koordinata). Pored toga, prepravite i funkcije "`Rotiraj`" tako da nije potrebno imati dvije verzije ove funkcije, nego da verzija funkcije koja prima dva parametra prosto podrazumijeva da je centar rotacije koordinatni početak u slučaju da se drugi parametar izostavi. Obavezno prepravite i prateći testni program sa ciljem da testirate i napisane konstruktore.

6. Proširite klasu "`Tacka`" iz Zadatka 4 iz Zadataka za samostalno vježbanje 9 na analogan način kao u prethodnom zadatku.

7#. Definirajte i implementirajte klasu "`Datum`" koja omogućava čuvanje datumskih podataka. Klasa treba da ima sljedeći interfejs:

```
enum Mjeseci {Januar = 1, Februar, Mart, April, Maj, Juni, Juli, August, Septembar,
  Oktobar, Novembar, Decembar};
enum Dani {Ponedjeljak = 1, Utorak, Srijeda, Cetvrtak, Petak, Subota, Nedjelja);
Datum(int dan, Mjeseci mjesec, int godina);
Datum(int dan, int mjesec, int godina);
void Postavi(int dan, Mjeseci mjesec, int godina);
void Postavi(int dan, int mjesec, int godina);
int DajDan() const;
Mjeseci DajMjesec() const;
const char *DajImeMjeseca() const;
```

```
int DajBrojDanaUMjesecu() const;
int DajGodinu() const;
bool DaLiJePrestupna() const;
int DajDanUGodini() const;
Dani DajDanUSedmici() const;
const char *DajImeDanaUSedmici() const;
void IdiNaSljedeciDan();
void IdiNaPrethodniDan();
void PomjeriZa(int broj_dana);
void Unesi();
void Ispisi() const;
friend int BrojDanaIzmedju(const Datum &d1, const Datum &d2);
```

Konstruktor klase kreira datum u skladu sa informacijama koje se prosljeđuju kroz parametre (uz obaveznu kontrolu ispravnosti datuma) i baca izuzetak u slučaju da su proslijeđeni pogrešni podaci. Podržane su dva načina kako se može zadati mjesec: kao vrijednost tipa "`Mjeseci`" koji je lokalno definiran u interfejsu klase, ili kao cijeli broj u opsegu od $1-12$. Metode "`Postavi`" načelno obavljaju istu funkciju kao i konstruktori. Metode "`DajDan`", "`DajMjesec`" i "`DajGodinu`" služe za očitavanje informacija o danu, mjesecu i godini koje su pohranjene u datumu. Metoda "`DajImeMjeseca`" vraća ime mjeseca koji se čuva u datumu (preciznije, umjesto čitavog imena, vraća se samo pokazivač na prvi znak imena). Metoda "`DajBrojDanaUMjesecu`" vraća ukupan broj dana u mjesecu koji se čuva u datumu. Metoda "`DaLiJePrestupna`" vraća informaciju o tome da li je godina koja se čuva u datumu prestupna ili ne. Metoda "`DajDanUGodini`" vraća kao rezultat redni broj dana koji odgovara posmatranom datumu unutar godine koja se čuva u datumu (npr. datum 3/7/2002. je 164-ti dan u 2002. godini, tako da ova metoda za taj datum treba vratiti vrijednost 164). Metoda "`DanUSedmici`" vraća dan u sedmici koji odgovara podacima zapamćenim u datumu (rezultat treba da bude tipa "`Dani`" koji je lokalno definiran u interfejsu klase), dok metoda "`DajImeDanaUSedmici`" vraća ime dana u sedmici koji odgovara datumu, slično kao metoda "`DajImeMjeseca`". Metode "`IdiNaSljedeciDan`" i "`IdiNaPrethodniDan`" pomjeraju datum za jedan dan unaprijed odnosno unazad, dok metoda "`PomjeriZa`" pomjera datum za broj dana naveden parametrom. Pomjeranje se vrši unaprijed ukoliko je parametar pozitivan, a unazad ukoliko je parametar negativan. Metoda "`Unesi`" traži da se sa tastature unesu podaci o datumu u obliku *dan*/*mjesec*/*godina* (npr. 3/7/2002) nakon čega se vrši postavljanje datuma u skladu sa unesenim podacima. Ukoliko su uneseni podaci neispravni u bilo kojem smislu (što uključuje ne samo besmislene datume nego i slučajeve da nije unesena kosa crta ili da su unesena slova umjesto brojeva, itd.) treba baciti izuzetak (ne zaboravite prije bacanja izuzetka vratiti ulazni tok u ispravno stanje ukoliko je priroda greške bila takva da dovede ulazni tok u neispravno stanje). Metoda "`Ispisi`" ispisuje datum na sljedeći način: prvo se ispisuje redni broj dana, zatim tačka, zatim puno ime mjeseca (riječima) iza koje slijedi razmak, zatim redni broj godine iza kojeg takođe slijedi tačka i, konačno, ime odgovarajućeg dana u sedmici u zagradama. Na primjer, ukoliko je u objektu zapamćen datum 3/7/2002. poziv ove metode treba da na ekranu proizvede ispis "`3. Juli 2002. (Srijeda)`". Konačno, prijateljska funkcija "`BrojDanaIzmedju`" kao rezultat treba da vrati broj dana koji je protekao između dva datuma koji se zadaju kao parametri. Sve metode implementirajte izvan klase, osim trivijalnih metoda čija implementacija može stati u jedan red ekrana. Obavezno napišite i mali testni program u kojem će se testirati *svi zahtijevani elementi* ove klase.

Uputa: u nedostatku bolje ideje, metodu "`PomjeriZa`" možete implementirati pozivajući u petlji metode "`Sljedeci`" odnosno "`Prethodni`" (koje su mnogo jednostavnije za implementaciju), mada je takvo rješenje dosta neefikasno. Što se tiče računanja dana u sedmici koji odgovara zadanom datumu, najbolje je da prvo izračunate broj dana koji je protekao između nekog po volji izabranog referentnog datuma za koji znate na koji je dan u sedmici pao i posmatranog datuma (što nije posve baš trivijalno uraditi na efikasan način, naročito zbog prestupnih godina), a zatim izračunate ostatak dijeljenja tog broja dana sa 7. Na osnovu tog ostatka i poznate činjenice na koji je dan pao referentni datum, vrlo lako se zaključuje na koji dan pada posmatrani datum.

8#. Za potrebe neke aplikacije za dvodimenezionalnu kompjutersku grafiku, potrebno je razviti klasu "`Tacka3D`" koja predstavlja tačku u prostoru, sa sljedećim interfejsom:

```
Tacka3D();
Tacka3D(double x, double y, double z);
void Postavi(double x, double y, double z);
void PostaviSferno(double rho, double phi, double theta);
double DajX() const;
double DajY() const;
```

```
double DajRho() const;
double DajPhi() const;
double DajTheta() const;
void PostaviX(double x);
void PostaviY(double y);
void PostaviZ(double z);
bool DaLiJeKoordinatniPocetak() const;
void Transliraj(double delta_x, double delta_y, double delta_z);
friend bool DaLiSuIdenticne(const Tacka3D &t1, const Tacka3D &t2);
friend double Rastojanje(const Tacka3D &t1, const Tacka3D &t2);
```

Konstruktor bez parametara kreira tačku u koordinatnom početku, dok konstruktor sa tri parametra kreira tačku na osnovu zadanih Dekartovih koordinata x, y i z. Potpuno istu stvar radi i metoda "`Postavi`" (osim što je namijenjena za naknadnu promjenu pozicije tačke). Metoda "`PostaviSferno`" radi sličnu stvar, samo na osnovu sfernih koordinata ρ, φ i θ, pri čemu je veza između Dekartovih i sfernih koordinata izražena pomoću formula $x = \rho \cos\varphi \sin\theta$, $y = \rho \sin\varphi \sin\theta$, $z = \rho \cos\theta$ odnosno $\rho = \sqrt{x^2 + y^2 + z^2}$, $\varphi = \mathrm{atan2}(y, x)$, $\theta = \arccos(y/\rho)$ ("atan2" je četverokvadrantni arkus tangens, funkcija srodna funkciji "arkus tangens", a podržana je u jezicima C/C++ baš pod tim imenom). Metode "`DajX`", "`DajY`", "`DajZ`", "`DajRho`", "`DajPhi`" i "`DajTheta`" vraćaju kao rezultat odgovarajuće Dekartove odnosno sferne koordinate tačke, dok metode "`PostaviX`", "`PostaviY`" i "`PostaviZ`" omogućavaju promjenu x, y odnosno z koordinate tačke (bez promjene preostale dvije koordinate). Metoda "`DaLiJeKoordinatniPocetak`" vraća logičku vrijednost "tačno" ili "netačno" u ovisnosti da li tačka na koju je metoda primijenjena predstavlja koordinatni početak ili ne. Metoda "`Transliraj`" pomjera tačku za iznos Δx u smjeru x-ose, Δy u smjeru y-ose i Δz u smjeru z-ose, pri čemu se vrijednosti Δx, Δy i Δz navode kao parametri. Konačno, klasa podržava i dvije prijateljske funkcije. Funkcija "`DaLiSuIdenticne`" vraća logičku vrijednost "tačno" ili "netačno" u ovisnosti da li su tačke koje su joj proslijeđene kao parametri identične ili ne, dok funkcija "`Rastojanje`" vraća kao rezultat rastojanje između tačaka koje su joj proslijeđene kao parametri.

Implementirajte klasu sa navedenim osobinama, pri čemu ćete uzeti da su atributi klase Dekartove koordinate tačke x, y i z. Napisanu klasu demonstrirajte u testnom programu koji traži da se tastature unese prirodan broj n, koji zatim treba dinamički alocirati niz od n tačaka (tj. objekata tipa "`Tacka3D`") koje treba inicijalizirati na osnovu podataka koji se unose sa tastature (podaci o svakoj tački se unose posebno). Nakon okončanja unosa, program treba translirati sve unesene tačke u skladu sa podacima koji se unose sa tastature, te ispisati vrijednosti x, y, z, ρ, φ i θ za sve unesene tačke nakon obavljenih transformacija. Na kraju, program treba ispitati da li među unesenim tačkama ima jednakih tačaka (rezultat ispitivanja treba prikazati na ekranu), te pronaći par tačaka koje se nalaze na najmanjem međusobnom rastojanju i ispisati koje su to tačke.

9#. Pri razvoju neke hipotetičke aplikacije koja koristi klasu "`Tacka3D`" razvijenu u prethodnom zadatku, uočeno je da će neki vitalni algoritmi u aplikaciji mnogo brže raditi ukoliko se koordinate tačke interno čuvaju u sfernom koordinatnom sistemu, te da je pogodnije kao atribute klase čuvati njene polarne koordinate ρ, φ i θ. Implementirajte ponovo klasu "`Tacka3D`" iz prethodnog zadatka koristeći ovako izmijenjen dizajn, ali tako da nova klasa zadrži isti interfejs, odnosno da korisnici klase (bar sa aspekta njene upotrebe) ne primijete da je njena implementacija izmijenjena. Posebno, testni program koji je razvijen u prethodnom zadatku treba da radi ispravno bez ikakvih izmjena sa novonapisanom verzijom klase "`Tacka3D`".

Napomena: Ukoliko prilikom rješavanja Zadatka 3. pametno napišete pojedine metode klase, nećete ih uopće morati mijenjati u novoj verziji klase (ideja je pisati metode tako da se što više oslanjaju na druge napisane metode). U suprotnom, možda ćete morati iznova napisati sve metode, što nije baš relaksirajuće.

10#. Date su dvije pomoćne strukture "`Ugao`" i "`Tacka`", koje će se koristiti u daljoj postavci zadatka, definirane kako slijedi:

```
struct Ugao {
  int stepeni, minute, sekunde;
};
struct Tacka {
  double x, y;
};
```

Struktura "Ugao" modelira ugao u ravni izražen u stepenima, minutama i sekundama, dok struktura "Tacka" modelira tačku u ravni predstavljenu Descartesovim koordinatama. Vaš zadatak je da definirate i implementirate klasu "Trougao" koja omogućava čuvanje podataka koji opisuju trougao u ravni. Klasa treba da ima sljedeći interfejs:

```
explicit Trougao(double a);
Trougao(double a, double b);
Trougao(double a, double b, double c);
Trougao(double a, double c, Ugao gamma);
Trougao(Tacka t1, Tacka t2, Tacka t3);
void Postavi(double a);
void Postavi(double a, double b);
void Postavi(double a, double b, double c);
void Postavi(double a, double b, Ugao gamma);
void Postavi(Tacka tA, Tacka tB, Tacka tC);
static bool DaLiJeMoguc(double a, double b, double c);
double DajA() const;
double DajB() const;
double DajC() const;
Ugao DajAlpha() const;
Ugao DajBeta() const;
Ugao DajGamma() const;
double DajObim() const;
double DajPovrsinu() const;
void Skaliraj(double s);
void Ispisi() const;
friend bool DaLiSuPodudarni(const Trougao &t1, const Trougao &t2);
friend bool DaLiSuSlicni(const Trougao &t1, const Trougao &t2);
```

Predviđeno je nekoliko načina kako se trougao može kreirati. Konstruktor sa tri realna parametra kreira trougao čije su dužine stranica određene parametrima. Konstruktor sa dva realna parametra kreira pravougli trougao pri čemu parametri predstavljaju dužine kateta, dok konstruktor sa jednim realnim parametrom kreira jednakostranični trougao pri čemu su dužine svih stranica jednake vrijednosti parametra. Konstruktor sa dva realna parametra i jednim parametrom tipa "Ugao" kreira trougao sa zadanim dužinama dvije stranice, i uglom između njih (ugao je zadan putem odgovarajuće strukture, u stepenima minutama i sekundama). Konstruktor sa tri parametra tipa "Tacka" kreira trougao čija se tjemena nalaze u zadanim tačkama, gdje su t_A, t_B i t_C tjemena koja se nalaze nasuprot stranica a, b i c (odnosno, to su vrhovi uglova α, β i γ). Pri tome se nigdje u objektu ne pamte pozicije tih tačaka, nego one samo služe da se izračunaju dužine stranica. Metode "Postavi" (u pet varijanti) načelno obavljaju isti zadatak kao i odgovarajući konstruktori, a omogućavaju naknadnu izmjenu podataka o trouglu. Svi konstruktori kao i metode "Postavi" trebaju baciti izuzetak tipa "domain error" uz prateći tekst "Ilegalne duzine stranica" ukoliko nije moguće kreirati trougao sa zadanim parametrima (dužine stranica su legalne ukoliko su sve pozitivne i ukoliko je zbir dužina ma koje dvije stranice veći od dužine treće stranice). Degenerirani slučajevi u kojima se trougao reducira na duž (recimo trougao sa stranicama dužine 1, 2 i 3, ili dužine 2, 2 i 0) ili na tačku (recimo, trougao sa dužinama stranica 0, 0 i 0) također *nisu dozvoljeni*. Statička metoda "DaLiJeMoguc" samo testira da li je moguće kreirati trougao sa stranicama koje su zadane kao parametri (bez bacanja izuzetaka) i vraća kao rezultat logičku vrijednost "tačno" ili "netačno" u zavisnosti da li je test uspio ili ne. Metode "DajA", "DajB" i "DajC" vraćaju kao rezultat dužine stranica trougla, dok metode "DajAlpha", "DajBeta" i "DajGamma" vraćaju kao rezultat dužine odgovarajućih uglova trougla (α je ugao naspram stranice a, itd.). Uglovi se vraćaju u stepenima, minutama i sekundama kao struktura tipa "Ugao". Ukoliko broj sekundi koji se dobije nije cijeli broj, treba ga zaokružiti *na najbliži cijeli broj* (a ne samo odsjeći decimale). Metode "DajObim" i "DajPovrsinu" respektivno daju kao rezultat obim odnosno površinu trougla. Metoda "Skaliraj" skalira trougao sa faktorom koji je zadan kao parametar (tj. množi dužine svih stranica sa zadanom vrijednoću parametra). Faktor skaliranja mora biti pozitivan broj, inače treba baciti izuzetak tipa "domain error" uz prateći tekst "Ilegalan faktor skaliranja". Metoda "Ispisi" ispisuje podatke o dužinama stranica, uglovima, obimu i površini trougla (stil ispisa oblikujte po volji). Konačno, prijateljske funkcije "DaLiSuPodudarni" odnosno "DaLiSuSlicni" testiraju da li su dva trougla koja im se prenose kao parametri podudarna odnosno slična i vraćaju kao rezultat logičku vrijednost "tačno" ili "netačno" ovisno od rezultata testiranja. Dva trougla su podudarna ukoliko imaju identične stranice (koje ne moraju biti u istom redoslijedu tako da su, na primjer, podudarni trouglovi sa stranicama 5, 12 i 10 odnosno 10, 5 i 12), a slična ukoliko su im stranice proporcionalne (vrijedi ista primjedba).

Napisanu klasu demonstrirajte u testnom programu koji traži da se tastature unese prirodan broj n, koji zatim treba kreirati prazan vektor čiji su elementi pametni pokazivači na trouglove (tj. na objekte tipa "Trougao"). Nakon toga, sa tastature treba redom unositi podatke za n trouglova (podaci o svakom trouglu se unose posebno). Za svaki od trouglova, nakon obavljenog unosa treba dinamički kreirati odgovarajući trougao inicijaliziran u skladu sa unesenim podacima i pametni pokazivač na tako kreirani trougao ubaciti u vektor. Ukoliko korisnik zada vrijednosti stranica od kojih se ne može formirati trougao, treba ispisati poruku upozorenja i zatražiti novi unos podataka za isti trougao. Nakon okončanja unosa, program treba sortirati sve unesene trouglove u rastući poredak po površini (tj. troguao sa manjom površinom dolazi prije trougla sa većom površinom) i ispisati podatke o svim trouglovima nakon obavljenog sortiranja. Za sortiranje obavezno koristiti bibliotečku funkciju "sort" uz pogodno definiranu funkciju kriterija kao lambda funkciju. Na kraju, program treba pronaći sve parove podudarnih i sličnih trouglova i ispisati koji su to trouglovi (ili obavijest da takvih parova nema).

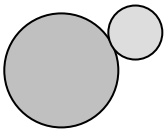
11#. Definirajte i implementirajte klasu "Krug" koja omogućava čuvanje podataka koji opisuju jedan krug u ravni. Krug je opisan sa tri podatka: x i y koordinatom centra i poluprečnikom r, koji mora biti nenegativan broj (dozvoljava se da bude $r = 0$; u tom slučaju, krug se degenerira u tačku). Klasa treba da ima sljedeći interfejs:

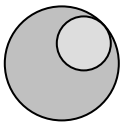
```
Krug(double r = 0);
Krug(double x, double y, double r = 0);
void PostaviPoziciju(double x, double y);
void PostaviPoluprecnik(double r);
double DajX() const;
double DajY() const;
double DajPoluprecnik() const;
double DajObim() const;
double DajPovrsinu() const;
void Ispisi() const;
void Transliraj(double delta_x, double delta_y);
void Rotiraj(double alpha);
void Rotiraj(double alpha, double x_c, double y_c);
friend bool DaLiSuIdenticni(const Krug &k1, const Krug &k2);
friend bool DaLiSuPodudarni(const Krug &k1, const Krug &k2);
friend bool DaLiSuKoncentricni(const Krug &k1, const Krug &k2);
friend bool DaLiSeDodirujuIzvana(const Krug &k1, const Krug &k2);
friend bool DaLiSeDodirujuIznutra(const Krug &k1, const Krug &k2);
friend bool DaLiSePreklapaju(const Krug &k1, const Krug &k2);
friend bool DaLiSeSijeku(const Krug &k1, const Krug &k2);
bool DaLiSadrzi(const Krug &k) const;
friend double RastojanjeCentara(const Krug &k1, const Krug &k2);
```

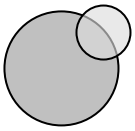
Konstruktor sa jednim parametrom kreira krug zadanog poluprečnika sa centrom u koordinatnom početku, dok konstruktor sa tri parametra kreira krug sa zadanom lokacijom centra i zadanim poluprečnikom. Oba konstruktora imaju podrazumijevanu vrijednost 0 za poluprečnik, tako da se mogu koristiti i bez parametara, odnosno sa dva parametra. Metode "PostaviPoziciju" i "PostaviPoluprecnik" omogućavaju naknadnu izmjenu pozicije kruga (bez izmjene poluprečnika) odnosno poluprečnika (bez izmjene pozicije). Konstruktori kao i metoda "PostaviPoluprecnik" bacaju izuzetak tipa "domain_error" uz prateći tekst "Ilegalan poluprečnik" ukoliko se kao poluprečnik navede negativan broj. Metode "DajX", "DajY", "DajPoluprecnik", "DajObim" i "DajPovrsinu" vraćaju redom x koordinatu centra kruga, y koordinatu centra kruga, poluprečnik, obim i površinu kruga, dok metoda "Ispisi" ispisuje na ekran podatke o krugu u obliku "$\{(x,y),r\}$". Metoda "Transliraj" pomjera krug za iznos Δx u smjeru x-ose i iznos Δy u smjeru y-ose, pri čemu se vrijednosti Δx i Δy navode kao parametri. Metoda "Rotiraj" dolazi u dvije verzije. Prva verzija rotira krug za ugao α koji se zadaje kao parametar (u smjeru suprotnom od kazaljke na satu) oko koordinatnog početka, dok druga verzija omogućava da se zadaju i koordinave tačke (x_c, y_c) oko koje se vrši rotacija. Za one koji nisu dovoljno upućeni u analitičku geometriju, napomenimo da se rotacijom tačke (x, y) oko tačke (x_c, y_c) za ugao α dobija tačka (x', y') gdje su x' i y' dati kao $x' = x_c + (x - x_c)\cos\alpha - (y - y_c)\sin\alpha$ i $y' = y_c + (x - x_c)\sin\alpha + (y - y_c)\cos\alpha$. Ugao α se zadaje *u radijanima*. Naravno, prilikom translacije i rotacije, mijenja se samo pozicija kruga, dok njegov poluprečnik ostaje isti.

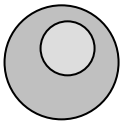
Predviđeno je i nekoliko prijateljskih funkcija koje ispituju međusobne odnose između krugova. Funkcija "DaLiSuIdenticni" vraća logičku vrijednost "tačno" ako i samo ako joj se kao parametri

prenesu dva *identična* kruga (krugovi na *istoj poziciji* i sa *istim poluprečnikom*), inače vraća logičku vrijednost "netačno". Funkcija "`DaLiSuPodudarni`" samo testira da li su krugovi *podudarni*, odnosno da li imaju *iste poluprečnike* (pri tome im se *pozicije mogu razlikovati*), dok funkcija "`DaLiSuKoncentricni`" testira da li krugovi imaju *zajednički centar*, dok im se *poluprečnici mogu razlikovati*. Dalje, slijede funkcije "`DaLiSeDodirujuIzvana`" odnosno "`DaLiSeDodirujuIznutra`" koje testiraju da li se krugovi *dodiruju*. Pretpostavlja se da se krugovi dodiruju ukoliko im *rubovi* (tj. kružnice kojima su omeđeni) imaju *tačno jednu zajedničku tačku*. Pri tome, dodir može biti *izvana*, kao na Slici 1, ili *iznutra*, kao na Slici 2. (dodir je izvana ukoliko je tim krugovima ta *zajednička tačka ujedno i jedina zajednička tačka*). Navedene funkcije upravo testiraju ta dva tipa dodira. Funkcija "`DaLiSePreklapaju`" testira da li *unutrašnjosti dva kruga* (unutrašnjost kruga čine sve njegove tačke koje *nisu na rubu*) *imaju zajedničkih tačaka*, dok funkcija "`DaLiSeSijeku`" testira da li se *rubovi dva kruga sijeku*, tj. imaju *tačno dvije zajedničke tačke*. Svaka dva kruga koja se sijeku također se i preklapaju, dok obrnuto ne mora vrijediti. Recimo, na Slici 3. imamo dva kruga koja se sijeku (i preklapaju), dok se krugovi na Slici 4. preklapaju, ali se ne sijeku. Konačno, predviđena je i funkcija članica (metoda) "`DaLiSadrzi`" koja testira da li se krug koji se zadaje kao njen parametar u potpunosti sadrži u krugu nad kojim je metoda pozvana (tj. da li je svaka njegova tačka ujedno i tačka kruga nad kojim je metoda pozvana), kao i prijateljska funkcija "`RastojanjeCentara`" koja vraća rastojanje između centara dva kruga koji joj se prenose kao parametri.


Slika 1


Slika 2


Slika 3


Slika 4

Napisanu klasu demonstrirajte u testnom programu koji traži da se tastature unese prirodan broj n, koji zatim treba dinamički alocirati niz od n krugova (tj. objekata tipa "`Krug`") koje treba postaviti na osnovu podataka koji se unose sa tastature (podaci o svakom krugu se unose posebno). Nakon okončanja unosa, program treba prvo translirati a zatim rotirati sve unesene krugove u skladu sa podacima koji se unose sa tastature. Za tu svrhu treba koristiti funkciju "`transform`" iz biblioteke "`algorithm`", pri čemu transformacionu funkciju koja se prosljeđuje funkciji "`transform`" treba izvesti kao lambda funkciju. Nakon obavljenih transformacija, treba ispisati podatke o svim unesenim krugovima (onakvim kakvi se dobiju nakon obavljenih transformacija). Podaci za svaki krug trebaju biti u posebnom redu. Ispis izvršite uz pomoć funkcije "`foreach`" i prikladne lambda funkcije. Potom treba ispisati podatke o krugu koji ima najveću površinu, za šta ćete iskoristiti funkciju "`max_element`" uz definiranje prikladne funkcije kriterija (ponovo kao lambda funkcije). Na kraju, program treba pronaći sve parove krugova koji se presjecaju i ispisati koji su to krugovi (ili ispisati obavijest da takvih krugova nema). To možete uraditi bez korištenja bibliotečkih funkcija, jer nema prikladnih funkcija za tu svrhu.

Napomena 1: Može se činiti da implementacija ove klase zahtijeva dobro poznavanje analitičke geometrije. Naprotiv, poznavanje formule za rastojanje dvije tačke i malo osnovne logike sasvim je dovoljno.

Napomena 2: Mada ovaj zadatak ima dugačak opis, može se uraditi relativno brzo, jer sve tražene funkcije imaju vrlo kratke implementacije (jedan do dva reda koda).