Zadaća 4.

Ova zadaća nosi ukupno 5 poena, pri čemu svi zadaci nose po 1 poen. Prva dva zadatka se mogu uraditi na osnovu gradiva sa prvih 13 predavanja i pretpostavljenog predznanja iz predmeta "Osnove računarstva", dok posljednja tri zadatka zahtijevaju i poznavanje Predavanja 14. Rok za predaju ove zadaće je nedjelja, 18. VI 2023. (do kraja dana).

1. U raznim oblastima nauke i tehnike često se javlja potreba rada sa promjenljivim veličinama koje se mijenjaju po sinusoidalnom zakonu oblika $A \sin(\omega t + \varphi)$, pri čemu se parametri A, ω i φ nazivaju redom *amplituda*, frekvencija i faza. Tipičan primjer su, recimo, struje i naponi u kolima naizmjenične struje. Interesantna osobina veličina ovog tipa je da zbir odnosno razlika dvije veličine ovog oblika sa istom frekvencijom ponovo predstavlja veličinu ovog oblika s istom frekvencijom, tj. vrijedi $A_1 \sin(\omega t + \varphi_1) \pm A_2 \sin(\omega t + \varphi_2) = A_3 \sin(\omega t + \varphi_3)$, pri čemu vrijednosti A_3 i φ_3 zavise na neki način od veličina A_1, A_2, φ_1 i φ_2. Ova zavisnost može se izvesti uz malu pomoć trigonometrije, ali može i još jednostavnije koristeći kompleksne brojeve i tzv. *fazorski prikaz sinusoidalnih veličina*, iz kojeg direktno slijedi da je $A_3 e^{i\varphi_3} = A_1 e^{i\varphi_1} \pm A_2 e^{i\varphi_2}$.

   Vaš zadatak je da napravite klasu nazvanu "`Sinusoida`" koja će služiti za podršku radu sa sinusoidalnim veličinama. Pored privatnih atributa koji čuvaju vrijednosti amplitude, frekvencije i faze (sve ugaone veličine su u radijanima), klasa treba sadržavati sljedeće elemente:

   - Konstruktor sa tri parametra, koji kreira objekat tipa "`Sinusoida`" na osnovu parametara koji predstavljaju vrijednosti amplitude, frekvencije i faze respektivno. Dozvoljavaju se bilo kakve vrijednosti za amplitudu, frekvenciju i fazu, ali pri kreiranju objekta objekat uvijek treba svesti na ekvivalentni oblik u kojem su amplituda i frekvencija nenegativne, a faza u opsegu od –π do π (tzv. normirani oblik). Na primjer, negativna vrijednost amplitude može se svesti na pozitivnu na osnovu činjenice da vrijedi $-A \sin(\omega t + \varphi) = A \sin(\omega t + \varphi \pm \pi)$, dok se negativne frekvencije možemo osloboditi na osnovu činjenice da vrijedi $A \sin(-\omega t + \varphi) = A \sin(\omega t - \varphi \pm \pi)$.
   - Trivijalne pristupne metode "`DajAmplitudu`", "`DajFrekvenciju`" i "`DajFazu`" koje daju kao rezultat vrijednosti amplitude, frekvencije odnosno faze.
   - Pristupnu metodu "`DajParametre`" koja daje kao rezultat uređenu trojku (tj. objekat tipa "`tuple`") koja sadrži "upakovane" sve parametre koje opisuju sinusoidu u obliku (A, ω, φ).
   - Metode "`PostaviAmplitudu`", "`PostaviFrekvenciju`" i "`PostaviFazu`" pomoću kojih se može izvršiti naknadna izmjenu amplitude, frekvencije i faze. Dozvoljeno je zadati bilo kakve vrijednosti, ali nakon bilo koje izmjene objekat uvijek mora ostati normiran. Recimo, ukoliko se zada negativna vrijednost amplitude, objekat nakon obavljene izmjene treba svesti na ekvivalentni objekat u kojem je amplituda nenegativna. Sve tri metode vraćaju kao rezultat referencu na izmijenjeni objekat, da se omogući kaskadno pozivanje.
   - Metodu "`PostaviParametre`" koja omogućava istovremenu naknadnu izmjenu amplitude, frekvencije i faze. Kao parametar metode zadaje se uređena trojka oblika (A, ω, φ). Slično kao i prethodne metode, nakon poziva ove metode objekat također mora biti normiran, a metoda vraća kao rezultat referencu na izmijenjeni objekat.
   - Preklopljene binarne operatore "`+`" i "`-`" koji primijenjeni na dva operanda koji su sinusoidalne veličine daju kao rezultat novu sinusoidalnu veličinu koja predstavlja zbir odnosno razliku sinusoidalnih veličina koje su zadane operandima. Pri tome, obje sinusoidalne veličine moraju imati istu frekvenciju, inače treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Razlicite frekvencije"). Rezultati moraju biti normirani. Za realizaciju Vam mogu biti od velike koristi tipovi i funkcije iz biblioteke "`complex`" (mada niste obavezni da ih koristite).
   - Preklopljeni unarni operator "`-`" koji primijenjen na sinusoidalnu veličinu daje kao rezultat negiranu sinusoidalnu veličinu. Formalno, negirana sinusoidalna veličina dobija se prostim negiranjem amplitude, ali će nakon normiranja (koje obavezno treba provesti) ona ponovo postati pozitivna (uz odgovarauću promjenu faze).
   - Preklopljene binarne operatore "`*`" i "`/`" koji omogućavaju množenje sinusoidalne veličine realnim brojem (kao i ekvivalentnu operaciju množenja realnog broja sinusoidalnom veličinom), odnosno dijeljenje sinusoidalne veličine realnim brojem. Ovi operatori kreiraju nove sinusoidalne veličine dobijene kao rezultat izvršene operacije. Pri tome, sinusoidalna veličina se množi odnosno dijeli sa realnim brojem tako što se prosto amplituda pomnoži odnosno podijeli zadanim brojem, dok frekvencija i faza ostaju iste. Rezultat svakako mora biti normiran.

- Preklopljene binarne operatore "`+=`", "`-=`", "`*=`" i "`/=`", koji omogućavaju da se kad god izrazi poput "`X = X + Y`", "`X = X - Y`", "`X = X * Y`" i "`X = X / Y`" imaju smisla, isti izrazi mogu skraćeno pisati kao "`X += Y`", "`X -= Y`", "`X *= Y`" i "`X /= Y`".
- Preklopljeni operator "`[]`" koji omogućava da se amplitudi, frekvenciji i fazi neke sinusoidalne veličine (nazovimo je "`s`") može pristupiti pomoću konstrukcija poput "`s["A"]`", "`s["omega"]`" i "`s["phi"]`". Tekstovi koji se mogu zadati između navodnika mogu biti jedino "A" za amplitudu, "omega" ili "w" za frekvenciju, te "phi" ili "fi" za fazu. U svim ostalim slučajevima, treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Neispravan naziv parametra".
- Preklopljeni operator "`()`" koji daje kao rezultat vrijednost sinusoidalne veličine u trenutku *t*, pri čemu se vrijednost *t* zadaje kao parametar. Uvođenje ovog operatora omogućiće da se objekti tipa "`Sinusoida`" ponašaju kao funkcije sa jednim argumentom *t* (što oni, logički gledano, zapravo i jesu).

Obavezno napišite i mali testni program u kojem ćete testirati sve elemente napisane klase.

2. Za potrebe neke meteorološke stanice neophodno je vrsiti čestu registraciju količine padavina. Za tu svrhu meteorološka stanica koristi računarski program u kojem je definirana i implementirana klasa nazvana "`Padavine`". Ova klasa omogućava čuvanje podataka o količini padavina za izvjesni vremenski period u vektoru cijelih brojeva, kojem se pristupa preko odgovarajućeg privatnog atributa (količina padavina se zadaje u centimetrima, zaokruženo na cijeli broj). Interfejs klase sadrži sljedeće elemente:

   - Konstruktor sa jednim parametrom koji predstavlja maksimalno dozvoljenu količinu padavina koja se može registrirati (oprez: ovo nije maksimalan broj količina padavina koje se mogu registrirati, nego maksimalan iznos količine padavina koji se smije zadati pri jednoj registraciji). Ovaj parametar mora biti pozitivan, u suprotnom treba baciti izuzetak tipa "`range_error`" uz odgovarajući prateći tekst "Ilegalna maksimalna kolicina". Potrebno je zabraniti da se ovaj konstruktor koristi za automatsku konverziju cijelih brojeva u objekte tipa "`Padavine`", s obzirom da bi takva konverzija bila besmislena.
   - Metodu "`RegistrirajPadavine`" koja vrši registraciju nove količine padavina, pri čemu se količina padavina koja se registrira prenosi kao parametar metode. U slučaju je količina padavina manja od nule ili veća od maksimalne dozvoljene količine padavina, metoda treba da baci izuzetak tipa "`range_error`" uz prateći tekst "Ilegalna kolicina padavina".
   - Metodu "`DajBrojRegistriranihPadavina`" koja daje broj registriranih količina padavina.
   - Metodu "`BrisiSve`" koja briše sve unesene količine padavina.
   - Metode "`DajMinimalnuKolicinuPadavina`" i "`DajMaksimalnuKolicinuPadavina`" koje vraćaju kao rezultat minimalnu i maksimalnu količinu padavina (u slučaju da nema registriranih količina padavina, obje metode treba da bace izuzetak tipa "`range_error`" uz prateći tekst "Nema registriranih padavina"). Za realizaciju nije dozvoljeno koristiti petlje, nego isključivo odgovarajuće funkcije iz biblioteke "`algorithm`".
   - Metodu "`DajBrojDanaSaPadavinamaVecimOd`" koja vraća kao rezultat broj dana u kojima je količina padavina bila veća od vrijednosti koja se zadaje kao parametar (u slučaju da nema registriranih količina padavina, metode treba da bace izuzetak tipa "`range_error`" uz prateći tekst "Nema registriranih padavina"). Za realizaciju nije dozvoljeno koristiti petlje, niti lambda funkcije ili pomoćne imenovane funkcije kriterija, nego isključivo samo odgovarajuće funkcije i/ili funktore iz biblioteka "`algorithm`" i "`functional`". Napomena: Ovdje ćete morati koristiti veznike, a na njihovu upotrebu treba se navići. Prvo probajte riješiti problem uz pomoć lambda funkcija. Kada Vam funkcija proradi, probajte istu funkcionalnost postići pomoću jednostavnijih ali zastarjelih veznika "`bind1st`" ili "`bind2st`". (koji su, usput, u verziji C++17 potpuno odbačeni), kao međukorak da bolje steknete osjećaj šta se zaista dešava. Kada Vam i to proradi, zamijenite zastarjele veznike sa boljim i univerzalnijim veznikom "`bind`" (zastarjele verzije veznika nemojte ostavljati u završnoj verziji).
   - Metodu "`Ispisi`" koja ispisuje sve unesene (registrirane) količine padavina sortirane u opadajućem poretku (tj. najveća količina padavina se ispisuje prva), pri čemu se svaka količina padavina ispisuje u posebnom redu. Pri tome je neophodno koristiti funkcije i/ili funktore iz biblioteka "`algorithm`" i "`functional`" (upotreba lambda funkcija ili pomoćnih imenovanih funkcija nije dozvoljena). Ova funkcija obavezno treba biti inspektor funkcija, tj. treba biti deklarirana sa modifikatorom "**`const`**"!

- Preklopljeni operator "`[]`" koji omogućava da se direktno pročita *i*-ta registrirana količina padavina (numeracija ide od jedinice). Ukoliko je indeks izvan dozvoljenog opsega, treba baciti izuzetak tipa "`range_error`" uz prteći tekst "Neispravan indeks". Pri tome, ovaj operator se *ne može koristiti* za izmjenu podataka, odnosno ne može se koristiti sa lijeve strane znaka jednakosti.
- Preklopljeni unarni operator "`++`" koji povećava sve registrirane količine padavina za jedinicu (pri tome se za jedinicu povećava i informacija o maksimalno dozvoljenoj količini padavina). Potrebno je podržati kako prefiksnu, tako i postfiksnu verziju ovog operatora. Za realizaciju nije dozvoljeno koristiti petlje, niti lambda funkcije ili pomoćne imenovane funkcije, nego isključivo odgovarajuće funkcije i/ili funktore iz biblioteka "`algorithm`" i "`functional`". Vrijedi ista napomena kao u metodi "`DajBrojDanaSaPadavinamaVecimOd`".
- Preklopljene binarne operatore "`+`" i "`-`" koji djeluju na sljedeći način: Ukoliko je "`X`" objekat tipa "`Padavine`", a "`Y`" cijeli broj, tada je "`X + Y`" novi objekat tipa "`Padavine`" u kojem su sve registrirane količine padavina povećane za iznos "`Y`" (u novodobijenom objektu treba povećati i informaciju o maksimalno dozvoljenoj količini padavina). Izraz "`Y + X`" treba da ima isto značenje kao i izraz "`X + Y`". Izraz "`X - Y`" u slučaju da je "`X`" objekat tipa "`Padavine`", a "`Y`" cijeli broj interpretira se analogno (uz odgovarajuće smanjenje informacije o maksimalno dozvoljenoj količini padavina), dok je tada "`Y - X`" objekat tipa "`Padavine`" u kojem su sve registrirane količine padavina oduzete od vrijednosti "`Y`", dok je maksimalna dozvoljena količina padavina upravo "`Y`". Pri tome, ukoliko se kao rezultat sabiranja ili oduzimanja dobije da neka od količina padavina postane negativna, treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Nekorektan rezultat operacije". U slučaju kada su i "`X`" i "`Y`" objekti tipa "`Padavine`", tada je izraz "`X - Y`" novi objekat tipa "`Padavine`" koji sadrži *razlike* odgovarajućih količina padavina iz objekata "`X`" i "`Y`". U ovom posljednjem slučaju se podrazumijeva da "`X`" i "`Y`" sadrže isti broj registriranih količina padavina, kao i da su registrirane padavine u objektu "`X`" veće ili jednake od odgovarajućih registriranih padavina u objektu "`Y`" (u suprotnom, treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Nesaglasni operandi"). Maksimalna dozvoljena količina padavina u novokreiranom objektu treba biti kakva je bila u objektu "`X`". U svim ostalim slučajevima, značenje izraza "`X+Y`" odnosno "`X-Y`" nije definirano. Za realizaciju ovih operatora nije dozvoljeno koristiti petlje, niti lambda funkcije ili pomoćne imenovane funkcije, nego isključivo odgovarajuće funkcije i/ili funktore iz biblioteka "`algorithm`" i "`functional`". Vrijedi ista napomena kao u metodi "`DajBrojDanaSaPadavinamaVecimOd`".
- Preklopljene binarne operatore "`+=`" i "`-=`" čiji je cilj da značenje izraza oblika "`X += Y`" odnosno "`X -= Y`" bude identično značenju izraza "`X = X + Y`" i "`X = X - Y`" kad god oni imaju smisla.
- Preklopljeni unarni operator "`-`" koji daje kao rezultat novi objekat tipa "`Padavine`" u kojem su sve količine padavina oduzete od maksimalno dozvoljene količine padavina. Za realizaciju nije dozvoljeno koristiti petlje, niti lambda funkcije ili pomoćne imenovane funkcije, nego isključivo odgovarajuće funkcije i/ili funktore iz biblioteka "`algorithm`" i "`functional`". Vrijedi ista napomena kao u metodi "`DajBrojDanaSaPadavinamaVecimOd`".
- Preklopljene binarne relacione operatore "`==`" i "`!=`" koje ispituju da li su dva objekta tipa "`Padavine`" jednaka ili nisu. Dva objekta ovog tipa smatraju se jednakim ukoliko sadrže isti broj registriranih količina padavina, i ukoliko su sve odgovarajuće registrirane količine padavina oba objekta jednake. Za realizaciju ovih operatora nije dozvoljeno koristiti petlje, nego isključivo odgovarajuće funkcije i/ili funktore iz biblioteka "`algorithm`" i "`functional`".

Implementirajte klasu sa navedenim svojstvima. Sve neophodne atribute treba obavezno izvesti kao privatne članove klase, a sve metode implementirajte izvan klase, osim metoda čija je implementacija dovoljno kratka, u smislu da zahtijeva recimo jednu ili dvije naredbe. Sve metode koje su po prirodi inspektori obavezno treba deklarirati kao takve. Obavezno napišite i mali testni program u kojem će se testirati sve elemente napisane klase.

3. Neka je *vozilo* objekat koji je, između ostalog, karakteriziran svojom težinom. *Automobil* je specijalna vrsta vozila, koje može primiti određeni manji broj putnika od kojih svaki ima svoju težinu. *Kamion* je specijalna vrsta vozila koje se može nakrcati teretom određene težine. *Autobus* je sličan automobilu, ali je predviđen za veći broj putnika.

   Cilj ovog zadatka je da razvijete surogatsku klasu "`Vozilo`" koja predstavlja polimorfni omotač za proizvoljnu vrstu vozila. Međutim, za tu svrhu, prvo je potrebno razviti hijerarhiju klasa koje opisuju pojedine vrste vozila. Bazna klasa "`ApstraktnoVozilo`" predstavlja apstraktnu baznu klasu

koja sadrži ono što je zajedničko za sve razmatrane vrste vozila. Ona posjeduje konstruktor kojim se zadaje težina vozila (tipa cijeli broj), te metode nazvane "`DajTezinu`", "`DajUkupnuTezinu`", "`DajKopiju`" i "`IspisiPodatke`". Prva metoda daje vlastitu težinu vozila.. Druga metoda je u baznoj klasi apstraktna i predviđena je da daje ukupnu težinu vozila u koju je uračunata i težina svega što se u vozilu nalazi. Treća metoda je također apstraktna, a predviđena je da kreira kopiju objekta nad kojim je pozvana i da vrati kao rezultat adresu kreirane kopije. Ova metoda će služiti za potrebe polimorfnog kopiranja. Konačno, apstraktna je i metoda "`IspisiPodatke`", koja je namijenjena za ispis podataka o tipu vozila, te njegovoj vlastitoj i ukupnoj težini. Naravno, sve apstraktne metode moraju biti izvedene tako da ukoliko se svim opisanim objektima pristupa preko pokazivača ili reference na baznu klasu "`ApstraktnoVozilo`", uvijek treba da bude pozvana ispravna verzija metode, koja će uzeti u obzir specifičnosti objekta.

Klasa "`Automobil`" nasljeđuje se iz klase "`ApstraktnoVozilo`". Njen konstruktor ima dodatni parametar, koji predstavlja vektor težina putnika (sve težine su cijeli brojevi). Zbog činjenice da postoji automatska konverzija inicijalizacionih listi u vektore, biće moguće konstrukcije tipa "`Automobil a(700, {80, 90, 60})`". Također, ova klasa sadrži i konkretne realizacije metoda "`DajUkupnuTezinu`", "`DajKopiju`" i "`IspisiPodatke`". Ispis treba da izgleda poput sljedećeg:

```
Vrsta vozila: Automobil
Vlastita tezina: 700 kg
Tezine putnika: 80 kg, 90 kg, 60 kg
Ukupna tezina: 930 kg
```

Klasa "`Kamion`" se također nasljeđuje iz klase "`ApstraktnoVozilo`", a njen konstruktor posjeduje dodatni cjelobrojni parametar koji predstavlja težinu tereta. Naravno, ova klasa sadrži konkretne realizacije svih apstraktnih metoda. Ispis za ovu klasu treba da izgleda poput sljedećeg:

```
Vrsta vozila: Kamion
Vlastita tezina: 1600 kg
Tezina tereta: 850 kg
Ukupna tezina: 2450 kg
```

Klasa "`Autobus`" prilično je slična klasi "`Automobil`", tako da se i ona također nasljeđuje iz klase "`ApstraktnoVozilo`". Kako autobus može prevoziti mnogo putnika, ne čuva se informacija o težini svakog od njih, nego se čuva informacija o broju putnika i njihovoj prosječnoj težini (tako da je ukupna težina svih putnika jednaka proizvodu broja putnika i prosječne težine). Konstruktor ove klase ima dva dodatna parametra u odnosu na baznu klasu (broj putnika i prosječna težina jednog putnika), te konkretne realizacije svih apstraktnih metoda. Ispis za ovu klasu treba da izgleda poput sljedećeg:

```
Vrsta vozila: Autobus
Vlastita tezina: 2500 kg
Broj putnika: 30
Prosjecna tezina putnika: 75 kg
Ukupna tezina: 4750 kg
```

Promjenljive tipa "`Vozilo`" moraju biti takve da se u njih može smjestiti bilo automobil, bilo kamion, bilo autobus (tj. sadržaj promjenljive tipa "`Automobil`", "`Kamion`" ili "`Autobus`") odnosno promjenljiva bilo kojeg tipa koji je izveden iz apstraktnog tipa "`ApstraktnoVozilo`" (što uključuje i tipove koji će eventualno biti kreirani u budućnosti). Naravno, sa promjenljivim tipa "`Vozilo`" mogu se raditi sve operacije koje se mogu raditi sa bilo kojom vrstom vozila (takvih operacija ovdje nema mnogo, ali to je samo da zadatak ne bude dugačak), mogu se bezbjedno kopirati, međusobno dodjeljivati, itd.

Napisane klase iskoristite u testnom program koji čita podatke o vozilima iz tekstualne datoteke "`VOZILA.TXT`" u vektor čiji su elementi vozila (tj. koji su tipa "`Vozilo`"), a zatim sortira vozila po ukupnoj težini u rastući poredak (koristeći funkciju "`sort`") i na kraju ispisuje ukupne težine vozila nakon sortiranja (svaka težina u posebnom redu). Svaki red datoteke sadrži podatke o jednom vozilu. Za slučaj automobila, prvi znak u redu je "`A`", nakon čega slijedi vlastita težina automobila, broj putnika i težina svakog od putnika, na primjer "`A500 3 80 60 75`" za automobil težine 500 kg sa 3 putnika težina 80, 60 i 75 kg respektivno. Za slučaj kamiona, prvi znak u redu je "`K`", nakon čega slijedi težina kamiona i težina tereta, na primjer "`K1500 1200`" za kamion težine

1500 kg natovaren sa teretom težine 1200 kg. Konačno, za slučaj autobusa, prvi znak u redu je "`B`", nakon čega slijedi težina autobusa, broj putnika i prosječna težina putnika, na primjer "`B2200 50 80`" za autobus težine 2200 kg sa 50 putnika čija je prosječna težina 80 kg. U slučaju bilo kakvih problema pri čitanju datoteke (što uključuje i slučaj kada datoteka sadrži neispravne podatke) treba ispisati prikladne poruke o greški.

4. Dopunite generičku klasu "`Matrica`" koju ste razvili u Zadatku 6 s Laboratorijske vježbe 12 (a koja se oslanja na klasu razvijenu na Predavanju 11_b) sa četiri metode "`SacuvajUTekstualnuDatoteku`", "`SacuvajUBinarnuDatoteku`", "`ObnoviIzTekstualneDatoteke`" i "`ObnoviIzBinarneDatoteke`", te jednim dodatnim konstruktorom, koji će kasnije biti opisan (ovo je ujedno dobra prilika da dovršite Zadatak 6 s Laboratorijske vježbe 12 ukoliko to već niste uradili prije). Sve ove metode primaju kao parametar naziv datoteke. Metoda "`SacuvajUTekstualnuDatoteku`" snima sadržaj matrice nad kojom je pozvana u tekstualnu datoteku čije je ime zadano. Datoteka treba formatirana tako da se podaci o svakom redu matrice čuvaju u posebnim redovima datoteke, pri čemu su elementi unutar jednog reda međusobno razdvojeni zarezima (iza posljednjeg reda nema zareza). Recimo, za neku matricu formata 3×4 kreirana datoteka može izgledati poput sljedeće:

```
2.5,-3,1.12,4
0,0.25,3.16,42.3
-1.7,2.5,0,5
```

U slučaju da dođe do bilo kakvih problema pri upisu, treba baciti izuzetak tipa "`logic_error`" uz prateći tekst "Problemi sa upisom u datoteku". Metoda "`SacuvajUBinarnuDatoteku`" obavlja sličnu funkcionalnost, samo što se upis vrši u binarnu datoteku, u koju je potrebno snimiti one podatke iz memorije koji su neophodni da bi se kasnije mogla pouzdano izvršiti rekonstrukcija stanja matrice iz pohranjenih podataka.Pri tome se podrazumijeva da je tip elemenata matrice neki skoro-POD tip podataka (u suprotnom, snimanje u binarnu datoteku najvjerovatnije neće biti uspješno). U slučaju problema pri upisu, vrijedi isto kao kod prethodne metode. Dalje, metode "`ObnoviIzTekstualneDatoteke`" odnosno "`ObnoviIzBinarneDatoteke`" vrše obnavljanje sadržaja matrice na osnovu sačuvanog stanja u tekstualnoj odnosno binarnoj datoteci, pri čemu se prethodni sadržaj matrice uništava. U oba slučaja, ukoliko tražena datoteka ne postoji, treba baciti izuzetak tipa "`logic_error`" uz prateći tekst "Trazena datoteka ne postoji". Pri čitanju iz tekstualne datoteke, ukoliko datoteka sadrži podatke koji nisu u skladu sa tipom elemenata matrice, ukoliko podaci nisu razdvojeni zarezima, ili ukoliko različiti redovi imaju različit broj elementa, treba baciti isti izuzetak, uz prateći tekst "Datoteka sadrzi besmislene podatke" (s obzirom da u tekstualnoj datoteci nije pohranjena nikakva informacija o broju redova i kolona, datoteku ćete morati efektivno iščitati dva puta, prvi put da saznate broj redova i kolona, a drugi put da zaista pročitate vrijednosti elemenata, nakon što su obavljene odgovarajuće dinamičke alokacije). U slučaju bilo kakvih drugih problema pri čitanju, treba također baciti isti izuzetak, uz prateći tekst "Problemi pri citanju datoteke". Za slučaj čitanja iz binarne datoteke, u slučaju bilo kakvih problema (osim nepostojeće datoteke), treba baciti ovaj isti izuzetak.

Konačno, rečeno je da je u klasu "`Matrica`" potrebno dodati i novi konstruktor. Taj konstruktor će imati dva parametra, pri čemu je prvi parametar ime datoteke, a drugi je logička vrijednost koja određuje da li će se konstrukcija objekta vršiti iz tekstualne datoteke (u slučaju kad taj parametar ima vrijednost "`false`") ili iz binarne datoteke (kada parametar ima vrijednost "`true`"). Ovaj konstruktor ponaša se identično kao što se ponašaju metode "`ObnoviIzTekstualneDatoteke`" odnosno "`ObnoviIzBinarneDatoteke`", samo se oslanja na činjenicu da se objekat tek stvara, tako da nema potrebe za oslobađanjem resursa koje je objekat prije toga koristio.

Obavezno napišite i kratki testni program u kojem ćete testirati novododane funkcionalnosti klase vezane za datoteke. Ostale funkcionalnosti klase ne morate testirati, s obzirom da se podrazumijeva da ste ih testirali ranije. Naravno, dozvoljena je upotreba i ostalih funkcionalnosti ukoliko su Vam potrebne za potrebe testiranje onoga što se traži da testirate (recimo, da formirate matrice koje želite snimiti, ili da ispišete sadržaj obnovljene matrice).

5. Jezik C++ posjeduje veliki broj raznih bibliotečki definirnih kontejnerskih tipova. Međutim, svi ti tipovi elemente čuvaju isključivo u radnoj memoriji. Da ispravite taj nedostatak, implementirajte generičku kontejnersku klasu "`DatotecniKontejner`" koja će elemente umjesto u memoriji čuvati u binarnoj datoteci. Kostur te klase treba izgledati ovako:

```
template <typename TipElemenata>
  class DatotecniKontejner {
    std::fstream tok;
  public:
    DatotecniKontejner(const std::string &ime_datoteke);
    void DodajNoviElement(const TipElemenata &element);
    int DajBrojElemenata();
    TipElemenata DajElement(int pozicija);
    void IzmijeniElement(int pozicija, const TipElemenata &element);
    void Sortiraj(std::function<bool(const TipElemenata &, const TipElemenata &)>
      kriterij = std::less<TipElemenata>());
  };
```

Tip "`TipElemenata`" može biti bilo koji tip koji zadovoljava uvjete da se primjerci tog tipa smiju pouzdano upisivati u binarne datoteke (tj. ma koji *skoro-POD* tip podakata). Jedini atribut klase je "`tok`", koji predstavlja objekat toka putem kojeg se vrši upis u datoteku odnosno čitanje iz datoteke. Konstruktor prima kao parametar ime datoteke i povezuje objekat toka sa datotekom navedenog imena. Ukoliko takva daoteka ne postoji, treba kreirati praznu datoteku navedenog imena. U slučaju bilo kakvih problema prilikom otvaranja ili kreiranja datoteke, treba baciti izuzetak tipa "`logic_error`" uz prateći tekst "Problemi prilikom otvaranja ili kreiranja datoteke".

Metoda "`DodajNoviElement`" dodaje element naveden kao parametar na kraj datoteke. Metoda "`DajBrojElemenata`" vraća kao rezultat ukupan pohranjenih elemenata (tj. broj elemenata u datoteci). Metoda "`DajElement`" vraća kao rezultat element koji se u datoteci nalazi na poziciji koja je zadana putem parametra metode (numeracija počinje od 0, tako da 0 označava prvi element u datoteci, 1 drugi element, itd.). Metoda "`IzmijeniElement`" mijenja element koji se nalazi na zadanoj poziciji u datoteci, pri čemu su parametri pozicija i nova vrijednost elementa. Ova i prethodna metoda trebaju baciti izuzetak tipa "`range_error`" uz prateći tekst "Neispravna pozicija" u slučaju da je pozicija negativna ili veća ili jednaka od ukupnog broja elemenata u datoteci. Također, sve metode trebaju baciti izuzetak tipa "`logic_error`" uz prateći tekst "Probemi prilikom pristupa datoteci" ukoliko dođe do bilo kakvih problema pri upisu u datoteku ili čitanja iz nje.

Predviđena je i funkcija "`Sortiraj`" koja sortira sadržaj kolekcije pohranjene u datoteci (bez njenog prethodnog učitavanja u radnu memoriju). Parametar ove funkcije je funkcija kriterija, koja radi na posve analogan način kao funkcija kriterija u bibliotečkoj funkciji "`sort`", pri čemu se može zadati bilo obična funkcija, bilo lambda funkcija, bilo funkcijski objekat (funktor), tačnije bilo šta što se može pozvati sa dva argumenta koji su tipa "`TipElemenata`" (ili se mogu konvertirati u taj tip) i koja vraća logičku vrijednost (ili nešto što se može interpretirati kao logička vrijednost). Recimo, ukoliko imamo objekat "`dki`" tipa "`DatotecniKontejner<int>`", sortiranje njegovog sadržaja u opadajući poredak možemo izvršiti pozivom

```
dki.Sortiraj([](int x, int y) { return x > y; });
```

ili još jednostavnije, koristeći funkcijske objekte iz biblioteke "`functional`":

```
dki.Sortiraj(std::greater<int>());
```

Ovaj parametar ima podrazumijevanu vrijednost "`std::less<TipElemenata>()`", tako da se može izostaviti ukoliko želimo sortiranje u podrazumijevani rastući poredak.

Obavezno napišite i testni program u kojem ćete demonstrirati sve elemente napisane klase.