Prirodno-matematički fakultet
Odsjek za matematiku
Sarajevo, 19.09.2009. godine

Zbirka zadataka iz predmeta

Uvod u programiranje i
Objektno orjentirano i generičko programiranje

Doc. dr. Željko Jurić
Ass. mr. Esmir Pilav

Sadržaj

Naredbe za ispis, ucitavanje podataka i naredbe grananja

1.1

Napišite program koji traži da sa tastature unesemo podatke o dva vremenska trenutka, u satima, minutama i sekundama, a koji računa i ispisuje vrijeme koje je proteklo između ta dva vremenska trenutka, također u satima, minutama i sekundama. Pretpostaviti da je drugi vremenski trenutak uvijek nakon prvog. Na primjer, dijalog nakon pokretanja programa može da izgleda poput sljedećeg:

```
Unesi prvo vrijeme (h min s): 14 15 7
Unesi drugo vrijeme (h min s): 16 9 34
Između ova dva trenutka proteklo je 1h 54min 27s.
```

Uputa: Jedan način rješavanja mogao bi biti sljedeći: pretvorite oba unesena vremena u broj sekundi. Na primer, imamo 14h 15min 7s = 51307s i 16h 9min 34s = 58174s. Oduzmite broj sekundi da dobijete broj sekundi između ovih trenutaka (6867s u zadanom primjeru). Konačno, razložite dobijeni broj sekundi ponovo na sate, minute i sekunde (6867s = 1h 54min 27s)

1.2 Napišite program koji će na ekranu ispisati vaše lične podatke u sljedećem obliku:

```
Vaše ime i prezime
------------------                    ime podvući crticama
                                      ostaviti jedan prazan red
Datum rođenja
Smjer studija
Grad
```

Program napišite u C++ stilu, koristeći biblioteku "`iostream`" i objekat izlaznog toka "`cout`".

1.2 Napišite program koji traži da se sa tastature unese brzina broda u čvorovima koja se zadaje isključivo kao cijeli broj (obavezno koristiti promjenljivu tipa "int"), a zatim izračunava i ispisuje brzinu broda u km/h kao decimalan broj. Koristite činjenicu da je čvor morska milja na sat, a da je jedna morska milja 1852 m (ovaj podatak obavezno definirati u programu kao konstantu). Na primjer, ukoliko se kao brzina broda unese broj 20, program treba da ispišerezultat 38.87689 jer je 20 čvorova = 38.87689 km/h.

1.3 Napišite program koji traži da se sa tastature unese neki podatak, a koji u zavisnosti od toga kakav je uneseni podatak ispisuje jedan od sljedećih pet komentara:

```
Uneseni podatak je prirodan broj.
Uneseni podatak je cijeli broj, ali nije prirodan.
Uneseni podatak je realan broj, ali nije cijeli.
Uneseni podatak nije broj.
```

Obavezno testirajte sve navedene slučajeve. Uputa: Prvo probajte unijeti podatak u realnu promjenljivu, a zatim testirajte ispravnost ulaznog toka. Ukoliko tok nije ispravan, podatak nije broj. Cijelost broja ćete testirati ispitivanjem da li se odsjecanjem decimala (tj. konverzijom u cjelobrojnu vrijenost) zadržava ista vrijednost.

1.4 Dvije firme nude usluge priključenja na Internet, pri čemu prva firma traži fiksnu pretplatu od 10 KM plus 50 feninga po svakom potrošenom satu, dok druga firma ne traži fiksnu pretplatu, ali traži 80 feninga po svakom potrošenom satu. Napišite program koji od korisnika traži da unese željeni broj sati a nakon toga da mu preporuči koja je firma isplatnija za njegove potrebe. Na primjer, za 15 sati isplatnija je druga firma ($10 + 15 \cdot 0.5 = 17.5 > 15 \cdot 0.8 = 12$), dok je za 40 sati isplatnija prva firma ($10 + 40 \cdot 0.5 = 30 < 40 \cdot 0.8 = 32$).

1.5 Napišite program koji traži da se unesu koeficijenti a, b i c kvadratne jednačine $a x2 + b x + c = 0$ a koji zatim računa i ispisuje njena rješenja. Program napraviti tako da se pri svakom unosu koeficijenta uvijek čitaju "svježi" podaci, bez obzira što je pri unosu prvog koeficijenta korisnik eventualno odmah unio tri podatka (uputa: koristite "`cin.ignore`"). Predvidite i mogućnost postojanja kompleksnih rješenja, koje ćete ispisivati kao uređene parove realnih brojeva. Program treba da predvidi i sve specijalne slučajeve (za $a = 0$, jednačina se svodi na linearnu, tako da imamo samo jedno rješenje, ukoliko je ujedno $b \neq 0$; za $a = b = 0$ i $c \neq 0$ nema rješenja, dok je za $a = b = c = 0$ jednačina identički zadovoljena za svaku vrijednost x).

1.6 Napišite program koji traži da se sa tastature unesu tri realna broja, a koji zatim ispisuje da li ta tri broja mogu biti stranice nekog pravouglog trougla. Napomena: ne zna se koji od tri unesenabroja predstavljaju katete, a koji hipotenuzu, tako da program treba da ponudi potvrdan odgovor kako na trojku brojeva 3, 4, 5, tako i na trojku brojeva 5, 4, 3 ili 3, 5, 4. Obavezno testirajte program i na ulaznim podacima 0.3, 0.4 i 0.5!

1.7 Napisati program koji ćeza unesena dva prirodna broja `M` i `N` ispisati njihov omjer. Na primjer, za `M=20` i `N=2` treba ispisati `M:N=5:1`, a za `M=20` i `N=50` bilo bi `M:N=1:2.5`.

1.8 Funkcija `f` definisana je po segmentima sa:

$$f(x) = \begin{cases} 0 & x \le 0 \\ x^2 & 0 < x \le 3 \\ 1 & 3 < x \le 5 \\ x^2\text{-}6x+9 & x > 5 \end{cases}$$

Napisati program koji će za datu vrijednost promjenljive `x` izračunati `f(x)`. Program testirati za slijedeće primjere: $f(-3) = 0$, $f(2) = 4$, $f(3) = 9$, $f(5) = 1$, $f(6) = 9$.

1.9 Stranice `a`, `b` i `c` čine trougao ako vrijedi

$$\frac{(a+b+c)}{2} > \max(a,b,c),$$

gdje je "`max`" maksimum od a, b i c. Napisati program za izračunavanje površine trougla uz provjeru valjanosti ulaznih podataka.

Uputa: koristite Heronov obrazac

$$P = \sqrt{s(s-a)(s-b)(s-c)}, \quad s = \frac{(a+b+c)}{2}.$$

Na primjer, ukoliko se unese `a = 6`, `b = 4` i `c = 3`, rezultati bi trebali da budu `P = 5.33268`.

1.10 Funkcija `f(x)` zadana je tablično:

x	< 0	0	3	5	> 2
f(x)	nije definisana	0	5	2	nije definisana

Napisati program koji će za zadani `x`, (`0 ≤ x ≤ 2`) izračunati `f(x)` uz pretpostavku linearne interpolacije izmedju datih parova tačaka, odnosno, ispisati "`f(x)` nije definisana" za `x<0` ili `x>2`.

1.11 Napisati program koji traži da se sa tastature unese matrica formata 2 x 2, a koji zatim računa inverznu matricu unesene matrice pretpostavljajući da je ona regularna.

Na primjer, ukoliko se unese matrica:

```
1      3
2.1  -0.5
```

rezultat bi trebao da bude matrica

```
0.0735294     0.441176
0.308824      -0.147059.
```

1.12 Napisati program koji traži da se sa tastature unesu dva realna broja *a* i *b*, i koji računa i ispisuje vrijednost dvojnog razlomka

$$\frac{a+b}{1+\dfrac{a^2+b^2}{a^2-b^2}}$$

Na primjer, ukoliko se unese $a = 3.7$ i $b = 2.61$ rezultat treba da bude 1.58508. Posebno testirati šta se događa ukoliko se za *a* i *b* unesu jednaki brojevi. Obrazložiti zaključak.

1.13 Napisati program koji traži da se unesu koeficijenti sistema jednačina $a_{11}x + a_{12}y + c_1 = 0$, $a_{21}x + a_{22}y + c_2 = 0$ i koji računa i ispisuje njegovo rješenja podrazumijevajući da rješenje postoji i da je jedinstveno (bez testiranja ispravnosti unesenih podataka). Na primjer, ukoliko se unese $a_{11} = 2$, $a_{12} = -3$, $c_1 = 4$, $a_{21} = -1.5$, $a_{22} = 3.4$, $c_2 = -2$, rješenja treba da budu $x_1 = -3.30435$, $x2 = -0.869565$. Posebno testirati šta se dešava ukoliko se unesu takve vrijednosti koeficijenata za koje ne postoje realna rješenja, na primjer $a_{11} = 2$, $a_{12} = -3$, $c_1 = 4$, $a_{21} = -4$, $a_{22} = 6$, $c_2 = -2$,. Obrazložiti zaključak.

1.14 Napisati program koji traži da se sa tastature unesu dvije stranice trougla *a* i *b*, kao i ugao γ između njih *u stepenima*, a nakon toga izračunava i ispisuje dužinu treće

stranice *c*. Na primjer, ukoliko se unese $a = 6$, $b = 4$ i $\gamma = 30^o$, rezultat treba da bude c = 3.22967. Posebno testirati slučaj pravouglog trougla $a = 3$, $b = 4$ i $\gamma = 90^o$ za koji se dobija $c = 5$. Za vrijednost broja π koristiti relaciju $\pi = 4 \cdot \text{arctg } 1$. Uputa: koristiti kosinusnu teoremu.

1.15 Napisati program koji traži da se sa tastature unesu dva prirodna broja *x* i *n*, a zatim ispisuje koliko je 1/*x* na *n* decimala. Na primjer, ukoliko se unese *x = 7* i *n = 8*, ispis treba da bude 0.14285714.

1.16 Napisati program koji traži da se sa tastature unesu dva realna broja x i y a zatim treba da ispiše maksimalan od njih. Na primjer, ukoliko unesemo brojeve $x = -7$ i $y = 2$ ispis treba da bude *2*.

1.17 Napisati program koji traži da se unese broj atoma vodonika *(H)-h*, broj atoma kiseonika *(O)-o*, broj atoma sumpora *(S)-s*, a zatim ispisuje koliko se molekula sumporne kiseline (H_2SO_4) može dobiti.

1.18 Horizontalne i vertikalne šahovske table označene su brojevima od 1 do 8. Unose se četiri broja *a*, *b*, *c* i *d*. Napisati program koji koji određuje da li su iste boje polja *(a, b)* i *(c, d)*

1.19 Napisati program koji traži da se sa tastature unese realan broj, zatim ispisuje na ekran da li je taj broj negativan, nula ili pozitiva.

1.20 Napisati program koji traži da se unesu četiri realna broja, a koji zatim na ekran ispisuje najveći među njima.

1.21 Napišite program koji traži da se sa tastature unesu pravougle koordinate *x, y* jedne tačke, a koji zatim na ekran ispisuje njene polarne koordinate tj. ugao (u stepenima) i argument kompleksnog broja *x+iy*.

1.22 Napišite program koji traži da se sa tastature unese iznos ugla u radijanima, a koji zatim ispisuje vrijednost tog istog ugla ispisanog u stepenima, minutama i sekundama (ignorirajte decimalni dio koji može preostati u sekundama). Računajte da *1* radijan ima *180/π* stepeni. Vrijednost π možete računati po formuli *π = 4 arctg 1*. Program treba da proizvede dijalog poput sljedećeg:

```
Unesite ugao u radijanima: 2.43
2.43 radijana iznosi 139 stepeni, 13 minuta i 43 sekundi
```

Za unos podataka i ispis rezultata koristite objekte "`cin`" i "`cout`" iz biblioteke "`iostream`".

1.23 Napišite program koji izračunava koliko je keramičkih pločica potrebno za popločavanje bazena čije se dimenzije u metrima unose sa tastature. Dimenzije pločica u centimetrima se također unose sa tastature. Program nakon pokretanja treba da na ekranu proizvede dijalog poput sljedećeg (naravno, brojevi koje korisnik zadaje odabrani su proizvoljno):

```
Unesi dimenzije bazena (axbxc) u metrima: 5 15 3
Unesi dimenzije pločice (axb) u centimetrima: 10 10
```

Za popločavanje bazena dimenzija 5x15x3 m sa pločicama dimenzija 10x10 cm potrebno je 19500 pločica.

Prećutno pretpostavite da su dimenzije zadane tako da je popločavanje uvijek izvodivo sa cijelim brojem pločica. Za unos podataka i ispis rezultata koristite objekte "`cin`" i "`cout`" iz biblioteke "`iostream`".

Petlje

2.1 Napišite program koji traži da se sa tastature unese prirodan broj n i n realnih brojeva, a koji računa i ispisuje na ekran zbir tih brojeva.

2.1 Napišite program koji traži da se sa tastature unesu cijeli brojevi n i m, a koji zatim iscrtava na ekranu pravougaonik sastavljen od zvjezdica čije su dužine stranica respektivno n i m. Na primjer, za n = 15 i m = 5, ispis na ekranu treba da izgleda kao

```
***************
*             *
*             *
*             *
***************
```

2.2 Napišite program koji traži da se sa tastature unese cijeli broj n, a zatim iscrtava na ekranujednakostranični trougao sastavljen od zvjezdica čija je osnovica horizontalna a vrh usmjeren nagore. Na primjer, ukoliko se unese n = 4, ispis na ekranu treba da izgleda kao

```
   *
  ***
 *****
*******
```

2.3 Napišite program koji traži da se sa tastature unese 6 brojeva, a koji zatim ispisuje da li su svi uneseni brojevi pozitivni i da li među njima ima neparnih brojeva. Za realizaciju programa ne koristiti nizove. Program testirajte na slijedećim karakterističnim primjerima:

```
Primjer 1: 2 6 10 18 8 6
Primjer 2: 4 -12 10 18 -18 10
Primjer 3: 3 5 4 19 7 12
Primjer 4: -3 9 15 -7 13 11
```

2.4 Napišite program koji za cijeli broj unesen sa tastature ispisuje sve njegove proste faktore razdvojene razmacima, pri čemu se svaki prosti faktor javlja onoliko puta koliko učestvuje u tom broju. Na primer, ukoliko se unese broj 290472, program treba da ispiše 2 2 2 3 7 7 13 19, jer je $290472 = 23 \cdot 3 \cdot 72 \cdot 13 \cdot 19$.

2.5 Napišite program koji traži da se unese realan broj x i prirodan broj n, a zatim računa i ispisuje vrijednost sume

$$S = \sum_{k=1}^{n} \frac{(-1)^k}{x \cdot (x+k)}$$

Na primjer, za $x = 2$ i $n = 5$ program treba da izbaci rezultat –0.261905. U programu nije dozvoljeno koristiti funkciju "`pow`".

2.6 Aritmetički niz je niz vrijednosti u kojima je razlika između susjednih vrijednosti konstantna

```
a,  a+d,  a+2d,  ...,  a+(n-1)d.
```

Zbir prvih n članova se računa po formuli:

```
S=n(2a+(n-1)d)/2.
```

Napisati program koji će za dato `a` i `d` napisati prvih `n` članova niza i izračunati njihov zbir koristeći prethodnu formulu.

2.7 Napisati program koji će N puta pročitati vrijednosti paralelno vezanih otpora i izračunati ukupan otpor. Koristiti formulu:

$$\frac{1}{R} = \sum_{i=1}^{N} \frac{1}{R_i}$$

Napisati program koji će ispisati sve cijele brojeve između `n` i `m` koji se mogu napisati ka zbir kvadrata dva nenegativne cijela broja `x` i `y`. Na proimjer, za `n=80` i `m=100` program treba da ispiše sljedeće brojeve: 80, 81, 82, 85, 89, 90, 97, 98, 100.

2.8 Napraviti program koji traži da se sa tastature unese cijeli broj n, a zatim iscrtava na ekranu ispunjeni jednakostranični trougao sastavljen od zvjezdica čija je osnovica horizontalna a vrh usmjeren nagore. Na primjer, ukoliko se unese n = 6, ispis na ekranu treba da izgleda kao

```
* * * * * * * * * * *
  * * * * * * * * *
    * * * * * * *
      * * * * *
        * * *
          *
```

2.9 Napisati program koji će ispisati tablicu ASCII kodova znakova s kodom 32 do 126, ali tako da u svakom redu bude ispisano po pet znakova i njihovih kodova.

2.10 Napraviti program koji zahtijeva da se sa tastature unose realni brojevi sve dok se ne unese broj 0. Nakon toga, program treba da ispiše aritmetičku i geometrijsku sredinu unesenih brojeva, ne računajući unesenu nulu. Na primjer, ukoliko se unesu brojevi 3, 8, 5.4, 2.13, 7 i 0, aritmetička sredina treba da bude 5.106, a geometrijska sredina 4.54168.

2.11 Napisati program koji traži da se unesu dva prirodna broja m i n, a koji nakon toga ispisuje najveći zajednički djelilac brojeva Fm i Fn tj. NZD(Fm, Fn) i FNZD(m,n) pri čemu je Fn n-ti Fibonačijev broj. Uvjerite se u istinitost tvrdnje koja kaže da je NZD(Fm, Fn) = FNZD(m,n). Na primjer, ukoliko se unesu brojevi m = 5 i n = 10, kako je F5 = 5 i F10 = 55 to je NZD(F5, F10) = 5, sa druge strane kako je NZD(5, 10) = 5, to iz F5 = 5 vidimo da je NZD(F5, F10) = FNZD(5, 10).

2.12 Napraviti program koji traži da se sa tastature unese cijeli broj *n*, a zatim iscrtava na ekranu ispunjeni jednakostranični trougao sastavljen od slova čija je osnovica horizontalna a vrh usmjeren nagore. Na primjer, ukoliko se unese n = 5, ispis na ekranu treba da izgleda kao

```
    A
   BAB
  CBABC
 DCBABCD
EDCBABCDE
```

213 Napraviti program koji traži da se sa tastature unese cijeli brojevi *m* i *n*, a zatim iscrtava na ekranu ispunjeni paralelogram slovima visine *m* i širine *n*. Na primjer, ukoliko se unese *m = 5* i *n=7,* ispis na ekranu treba da izgleda kao

```
   ABCDCBA
  BCDEDCB
 CDEFEDC
DEFGFED
EFGHGFE
```

2.13 Napisati program koji traži od korisnika unos rečenice, a koji zatim ispisuje istu rečenicu šifriranu Cezarovom šifrom. Cezarova šifra spada u najstarije poznate sisteme šifrovanja, prema kojoj se svaki znak izvorne rečenice zamijenjuje sa znakom koji se po abecedi nalazi 3 znaka ispred (uz izuzetak posljednja tri znaka abecede, koji se mijenjaju sa prva tri znaka abecede), odnosno zamjena se vrši prema sljedećoj tablici:

Izvorno	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Šifrovano	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Znaci koji nisu slova ostaju neizmijenjeni. Na primjer, ukoliko izvorna rečenica glasi "DANAS JE ZADNJI DAN LJETA", šifrovana rečenica će glasiti "GDQDV MH CDGQML GDQ OMHWD". Radi jednostavnosti, sve očitane znakove pretvarajte u velika slova. Uputa: ASCII šifre svih znakova u opsegu od 'A' do 'W' treba povećati za 3, a ASCII šifre znakova u opsegu od 'X' do 'Z' treba umanjiti za 23.

2.14 Za neki broj kažemo da je *palindroman* ukoliko je jednak broju koji se dobije čitanjem njegovih cifara u obrnutom poretku. Na primjer, broj 6574756 je palindroman broj. Napisati program koji traži da se sa tastature unese cijeli broj *n*, a zatim na ekran ispisuje da li je broj *n* palindroman ili nije. Za realizaciju programa ne koristiti nizove.

2.15 Napisati program koji traži da se sa tastature unese realan broj *x*. Ukoliko je zaista unesen broj, i ukoliko je $x \geq 0$, ispisati njegov kvadratni korijen, a u suprotnom ispisati poruku o greški i tražiti ponovan unos sve dok unos ne bude korektan. Obavezno testirati ispravnost programa za slučaj kada se više puta zaredom unesu neispravni podaci.

2.16 Iz matematičke analize je poznato da je za dovoljno veliko *n* vrijednost sume

$$S = \sum_{k=0}^{n} \frac{x^k}{k!}$$

približno jednaka vrijednosti funkcije $F = e^x$, s obzirom da za $n \to \infty$ vrijednost sume S konvergira ka F (s obzirom da je S Taylorov razvoj funkcije F, koji konvergira za svaku vrijednost *x*). Napisati program koji za zadanu vrijednost *x* određuje minimalnu vrijednost *n* takvu da se S i F poklapaju na barem prvih 5 decimala (tj. da je $|S - F| < 10^{-5}$). Program treba da ispiše nađenu vrijednost *n*, kao i odgovarajuće vrijednosti S i F (da bi se uvjerili da se one zaista poklapaju barem na prvih 5 decimala). Pri računanju sume S nije dozvoljeno posebno računati faktorijel niti koristiti funkciju "`pow`" za računanje stepena x^k, već treba iskoristiti ovisnost koja postoji između *k*-tog i *k*+1-vog člana sume.

2.17 Napisati program koji traži da se sa tastature unese neki binarni broj, a koji nakon toga ispisuje isti broj pretvoren u dekadni brojni sistem. Na primjer, ukoliko se unese broj 1101011, rezultat treba da bude 107, jer je $(1101011)_2 = (107)_{10}$. Program treba da prijavi grešku ukoliko ulazni podatak sadrži cifru koja nije 0 ili 1. Uputa: čitajte znakove unesenog podatka znak po znak, sve dok se ne dostigne kraj reda, ili znak koji nije cifra 0 ili 1. Svaki put kada očitate novu cifru, tekuću vrijednost broja pomnožite sa 2 i na nju dodajte vrijednost očitane cifre. Ova ideja se zasniva na Hornerovoj shemi, odnosno činjenici da je

$$b_N \cdot 2^N + b_{N-1} \cdot 2^{N-1} + b_{N-2} \cdot 2^{N-2} + \ldots + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 =$$
$$= 2 \cdot (2 \cdot (\ldots 2 \cdot (2 \cdot (2 \cdot b_N + b_{N-1}) + b_{N-2}) + \ldots b_2) + b_1) + b_0$$

2.18 Napišite program koji će prvo tražiti od korisnika da unese realan broj x, i stepen polinoma N, a zatim koeficijente polinoma $a_N, a_{N-1}, a_{N-2}, \ldots, a_2, a_1$ i a_0 (počev od koeficijenta uz najveći stepen). Nakon toga, program treba da izračuna i ispiše vrijednost polinoma

$$P(x) = a_N x^N + a_{N-1} x^{N-1} + a_{N-2} x^{N-2} + \ldots + a_2 x^2 + a_1 x + a_0$$

U programu ne treba koristiti nizove, s obzirom da se vrijednost polinoma može lako izračunati u "hodu" (tj. bez potrebe za pamćenjem koeficijenata). Najboljim rješenjem će se smatrati rješenje koje ne koristi ni funkciju za stepenovanje, s obzirom da se polinom P(x) može veoma jednostavno izračunati bez stepenovanja pomoću Hornerove sheme:

$$P(x) = ((\ldots((a_N x + a_{N-1})x + a_{N-2})x + \ldots + a_2)x + a_1)x + a_0$$

2.19 Napišite program koji ce prvo tražiti od korisnika da unese prirodan broj N, a zatim N realnih brojeva $a_1, a_2, a_3, \ldots, a_N$. Nakon toga, program treba da izračuna i ispiše vrijednost izraza

$$\frac{1}{a_1} + \frac{1}{a_1 + a_2} + \frac{1}{a_1 + a_2 + a_3} + \ldots + \frac{1}{a_1 + a_2 + a_3 + \ldots + a_N}$$

2.20 Napisati program koji od korisnika traži da unese realan broj x, cijeli broj N, kao i N cijelih brojeva $a_1, a_2, ..., a_N$. Nakon toga, program treba da izračuna i ispiše vrijednost izraza

$$\frac{a_1}{x} - \frac{a_2}{x^2} - \frac{a_3}{x^3} - ... \pm \frac{a_N}{x^N}$$

Pri čemu je posljednji znak "+" za neparno N, a "−" za parno N. U programu NE SMIJETE koristiti funkciju "`pow`", niti biblioteku "`cmath`" uopšte.

2.20 Napisati program koji će tražiti da se unese prirodan broj N, a koji zatim treba da ispiše njegovu faktorizaciju. Na primjer, ako je $N = 1222456$ program treba da ispiše $\{2,3\}, \{41,1\}, \{3727,1\}$ jer je $1222456 = 2^3 \cdot 42 \cdot 3727$, a za $N = 207368$ program treba da ispiše $\{2,3\}, \{7,2\}, \{23,2\}$ jer je $207368 = 2^3 \cdot 7^2 \cdot 23^2$.

2.20 Za neki broj kažemo da je savršen ukoliko je jednak sumi svih svojih djelilaca. Na primjer, 28 je savršen broj: njegovi djelioci su 1, 2, 4, 7 i 14, a 1 + 2 + 4 + 7 + 14 = 28. Napisati program koji traži da se sa tastature unese cijeli brojevi a i b, a koji zatim ispisuje sve savršene brojeve u opsegu od a do b. Kao provjeru ispravnosti programa možete koristiti činjenicu da su jedini savršeni brojevi u opsegu od 1 do 100 brojevi 6 i 28.

2.21 Napisati program koji traži da se unese prirodan broj n, a zatim računa i ispisuje vrijednost izraza (verižnog razlomka)

$$Y = \cfrac{1}{1+\cfrac{1}{2+\cfrac{1}{3+\cfrac{1}{...+\cfrac{1}{n}}}}}$$

Na primjer, za $n = 3$ treba da se dobije rezultat 0.7, jer je

$$Y|_{n=3} = \cfrac{1}{1+\cfrac{1}{2+\cfrac{1}{3}}} = \cfrac{1}{1+\cfrac{3}{7}} = \frac{7}{10} = 0.7$$

2.22 Za neki broj kažemo da je *palindroman* ukoliko je jednak broju koji se dobije čitanjem njegovih cifara u obrnutom poretku. Na primjer, broj 6574756 je palindroman broj. Napisati program koji traži da se sa tastature unese cijeli broj *n*, a zatim na ekran ispisuje da li je broj *n* palindroman ili nije. Za realizaciju programa ne koristiti nizove.

2.23 Napisati program koji traži da se sa tastature unesu dva prirodna broja *a* i *b*, i koji ispisuje broj brojeva u rasponu od *a* i *b* (uključujući i *a* i *b*) koji su djeljivi sumom svojih cifara (takav je, na primjer, broj 351, jer je djeljiv sa 3 + 5 + 1 = 9). Kao neke karakteristične vrijednosti za testiranje mogu vam poslužiti sljedeći rezultati:

a	10	100	351	352	10000	100000
b	50	1000	351	353	20000	200000

Rezultat	14	181	1	0	1417	11167

2.24 Napišite program koji će prvo tražiti od korisnika da unese dva prirodna broja “a” i “b”, a koji zatim ispisuje tablicu kvadrata i kubova svih prirodnih brojeva “N” u opsegu od “a” do “b” uključivo. Tablica bi trebala da izgleda poput tablice na sljedećoj slici, koja prikazuje izgled tablice za vrijednosti 8 i 11 respektivno za “a” i “b”:

```
+-----+-------+---------+
| N   | N^2   | N^3     |
+-----+-------+---------+
| 8   | 64    | 512     |
| 9   | 81    | 729     |
| 10  | 100   | 1000    |
| 11  | 121   | 1331    |
+-----+-------+---------+
```

Za formatiranje ispisa koristite manipulator “setw”. Radi ispravnog formatiranja tablice, pretpostavite da “a” i “b” neće biti veći od 100.

2.25 Napišite program koji za prirodan broj n unesen sa tastature ispisuje na ekran formatiranu tablicu množenja za sve brojeve od 1 do n uključivo. Na primjer, ukoliko se unese n = 5, ispis na ekran bi trebao izgledati poput sljedećeg:

```
+-----+-----+-----+-----+-----+
|   1 |   2 |   3 |   4 |   5 |
+-----+-----+-----+-----+-----+
|   2 |   4 |   6 |   8 |  10 |
+-----+-----+-----+-----+-----+
|   3 |   6 |   9 |  12 |  15 |
+-----+-----+-----+-----+-----+
|   4 |   8 |  12 |  16 |  20 |
+-----+-----+-----+-----+-----+
|   5 |  10 |  15 |  20 |  25 |
+-----+-----+-----+-----+-----+
```

Pretpostavite da je n takav da će čitava tablica stati na ekran i da niti jedan produkt neće imati više od 3 cifre. Za ispis koristite objekat izlaznog toka “cout”, a za prilagođavanje širine ispisa manipulator “setw”.

2.26 Položaj zamišljenog robota koji može da se kreće kroz koordinatni sistem sa cjelobrojnim koordinatama opisuje se pomoću tri promjenljive “x”, “y” i “orjentacija”. Promjenljive “x” i “y” su tipa “int” i one čuvaju x odnosno y koordinatu pozicije na kojoj se robot trenutno nalazi. Promjenljiva “orjentacija” je tipa “Pravci”, koji predstavlja pobrojani tip definiran kao

```
enum Pravci {Sjever, Istok, Jug, Zapad};
```

Ova promjenljiva sadrži informaciju o pravcu u kojem robot trenutno gleda. Potrebno je napraviti program koji korisniku nudi sljedeće opcije za upravljanje robotom: L – Nalijevo; D – Nadesno; I – Idi; K – Kraj. Opcije L odnosno D obrću robota nalijevo odnosno nadesno za 90^o. Opcija I treba da pita korisnika za broj koraka, a nakon toga pomjera robota zadani broj koraka u pravcu u kojem robot trenutno gleda. Opcija K završava program. Sve druge opcije su ilegalne, i trebaju dovesti do prijave greške i

ponovnog izbora opcije. Na početku rada, robot se nalazi na poziciji (0,0) i gleda na sjever. Dijalog između programa i korisnika mogao bi izgledati poput sljedećeg:

```
Robot se nalazi na poziciji (0,0) i gleda na sjever.
Unesi komandu (L – Nalijevo, D – Nadesno, I – Idi, K – Kraj): D
Robot se nalazi na poziciji (0,0) i gleda na istok.
Unesi komandu (L – Nalijevo, D – Nadesno, I – Idi, K – Kraj): I
Unesi broj koraka: 5
Robot se nalazi na poziciji (5,0) i gleda na istok.
Unesi komandu (L – Nalijevo, D – Nadesno, I – Idi, K – Kraj): S
Pogrešna komanda!
Unesi komandu (L – Nalijevo, D – Nadesno, I – Idi, K – Kraj): L
Robot se nalazi na poziciji (5,0) i gleda na sjever.
Unesi komandu (L – Nalijevo, D – Nadesno, I – Idi, K – Kraj): I
Unesi broj koraka: 3
Robot se nalazi na poziciji (5,3) i gleda na sjever.
Unesi komandu (L – Nalijevo, D – Nadesno, I – Idi, K – Kraj): L
Robot se nalazi na poziciji (5,3) i gleda na zapad.
Unesi komandu (L – Nalijevo, D – Nadesno, I – Idi, K – Kraj): KK
Pogrešna komanda!
Unesi komandu (L – Nalijevo, D – Nadesno, I – Idi, K – Kraj): K
Dovidjenja!
```

Nizovi, vektori i red

3.1 Napišite program koji traži da se prvo unese prirodan broj “n”, a nakon toga elementi vektora “a” koji ima “n” cjelobrojnih elemenata, za koje ćemo prećutno pretpostaviti da su svi prirodni brojevi, tj. veći od nule (ovu pretpostavku ne treba provjeravati u programu). Zatim, program treba da kreira dva nova vektora “b” i “c”, i da u vektor “b” prepiše sve proste brojeve iz vektora “a” (tj. one djeljive samo sa 1 i sa samim sobom), a u vektor “c” sve složene brojeve iz vektora “a” (tj. one koje imaju više od dva djelioca). Konačno, program treba da u jednom redu ispiše elemente vektora “b”, a u drugom redu elemente vektora “c”. Brojevi trebaju biti međusobno razdvojeni zarezom, pri čemu iza posljednjeg broja u svakom redu ne treba da bude zarez. Na primjer, ukoliko se u vektor “a” unese slijed brojeva 3, 4, 2, 5, 9, 4, 10, 1, 15, 13, 8 i 2, ispis na ekranu treba da bude

```
3,2,5,7,13,2
4,9,4,10,15
```

s obzirom da se broj 1 ne smatra ni prostim ni složenim. Posebno testirajte slučajeve kada su svi uneseni brojevi parni ili kada su svi uneseni brojevi neparni.

3.2 Prepravite prethodni tako da umjesto tipa “vector” koristi tip “deque” (tj. da koristi dek umjesto vektora), i uvjerite se da sve i dalje radi isto. Zatim, zamijenite poziv funkcije “push back” sa pozivom funkcije “push front” i uporedite razliku.

3.3 Napisati program koji traži da se unese 10 brojeva sa tastature, a zatim ispisuje u jednom redu brojeve koji su parni, a u drugom redu brojeve koji su neparni. Brojevi trebaju biti međusobno razdvojeni zarezom. Iza posljednjeg broja u svakom redu ne treba zarez. Na primjer, ukoliko se unese niz brojeva 3, 4, 2, 5, 9, 4, 10, 15, 8 i 2, ispis na ekranu treba da bude

```
4,2,4,10,8,2
3,5,9,15
```

Posebno testirati slučajeve kada su svi uneseni brojevi parni ili kada su svi uneseni brojevi neparni.

3.3 Napisati program koji traži da se sa tastature unese prirodni broj n, a koji zatim na ekran ispisuje vrijednost broja n prikazanog u binarnom zapisu. Na primjer, ako unesemo broj 1339871, na ekranu se treba ispisati 10100011100011101111 (Napomena: brojevi koji staju u promjenljive tipa "int" nikada neće imati više od 32 binarne cifre).

3.4 Napisati program koji traži da se sa tastature unese niz xi od n različitih prirodnih brojeva a koji ispisuje elemente tog niza tako da vrijedi sljedeće uređenje: x1 < x2 > x3 < x4 > x5 <... Izračunati zbirove S1 i S2 gdje je:

a) $S_1 = x_1 + x_3 + x_5 + \ldots$
b) $S_2 = x_2 + x_4 + x_6 + \ldots$

Na primjer, ako unesemo sljedećih sedam brojeva 2, 9, 12, 5, 6, 3, 4, program treba da date brojeve ispiše u sljedećem redosljedu: 2, 12, 3, 9, 4, 6, 5 (ili na neki drugi način tako da vrijedi traženo uređenje), pri čemu je S1 = 2 + 3 + 4 + 5 i S2 = 12 + 9 + 6.

3.5 Napisati program koji traži da se sa tastature unese *n* cijelih brojeva (pri čemu se *n* također prethodno zadaje sa tastature), i koji nakon toga iz niza odstranjuje one brojeve koji se ponavljaju više od jedanput. Program treba takođe da ispiše i broj elemenata koji se ponavljaju. Na primjer, ukoliko se sa tastature unese niz od 15 brojeva: 7, 10, 4, 2, 4, 4, 5, 6, 7, 3, 9, 1, 8, 6 i 7 program treba da ispiše brojeve 10, 2, 5, 3, 9, 1 i 8, kao i da ispiše da se 3 broja ponavljaju (to su brojevi 7, 4 i 6).

3.6 Napisati program koji traži da se unese niz od n prirodnih brojeva a_i, a zatim računa i ispisuje vrijednost izraza (verižnog razlomka)

$$Y = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{\ldots + \cfrac{1}{a_n}}}}}$$

3.7 Napisite program koja traži da se unese niz od *n* cijelih brojeva, a program treba da iz niza *odstrani* sve elemente niza koji su potpuni kvadrati (odnosno elemente koji se mogu napisati kao kvadrat nekog cijelog broja). Program treba da vrati kao rezultat broj elemenata niza nakon obavljenog odstranjivanja i da ispiše elemete niza nakon odstranjivanja potpunih kvadrata.

3.8 Članovi Hammingovog niza su cijeli brojevi generisani prema slijedćim pravilima:

a) 1 je član Hammingovog niza
b) Ako je x član Hammingovog niza, tada su i 2x, 3x i 5x također članovi Hammingovog niza.

Napisati program koji traži da se unese prirodan broj $n < 255$, a zatim ispisati članove Hammingovog niza u intervalu od *1* do *n*. Na primjer, za $n=15$ članovi Hammingovog niza su *:1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15*.

3.9 Napisati program koji traži da se sa tastature unesu prirodni brojevi B1 i B2 koji se nalaze između 2 i 16 (uključiva) i broj n u bazi B1 a program treba da ispiše taj broj u bazi B2. Na primjer, za $B_1=2$ i $n=11000011111$ i $B_2=13$ program treba da ispiše 937 jer je $(11000011111)_2=(937)_{13}$

Nizovi stringova i pokazivači

6.1 Napišite funkciju sa dva parametra, od kojih je prvi tipa "string", a drugi cjelobrojnog tipa. Prvi parametar predstavlja neku rečenicu, a drugi parametar redni broj riječi unutar te rečenice. Funkcija treba da izdvoji tu riječ iz rečenice, i da vrati kao rezultat tako izdvojenu riječ. Na primjer, ukoliko je kao prvi parametar zadan tekst " Na vrh brda vrba mrda" a kao drugi parametar broj 4, funkcija treba kao rezultat da vrati string "vrba". Ovdje pod pojmom riječ podrazumijevamo bilo koji slijed uzastopnih znakova koji nisu razmaci, a ispred kojeg se eventualno nalazi razmak (ili ništa), i iza kojeg eventualno slijedi razmak (ili ništa). Tako se, na primjer, u tekstu "Kiša pada.Trava raste" slijed znakova "pada.Trava" tretira kao jedna riječ (druga po redu), jer iza tačke nema razmaka (ovakav tretman vrijedi i u tekst procesorima poput Microsoft Word-a). Obratite pažnju da riječi mogu biti razdvojene sa više uzastopnih razmaka, kao i da na početku i kraju teksta može, ali i ne mora biti razmaka. Ukoliko je drugi parametar manji od 1 ili veći od broja riječi u rečenici, funkcija treba baciti izuzetak. Napisanu funkciju demonstrirajte u testnom programu u kojem se za rečenicu unesenu sa tastature i prirodan broj n ispisuje n-ta riječ te rečenice (pozivom napisane funkcije). U testnom programu obavezno predvidite hvatanje izuzetaka koji mogu biti bačeni iz funkcije.

6.2 Napišite program koji od korisnika traži da sa tastature unese rečenicu, a koji će zatim ispisati unesenu rečenicu bez prve riječi te rečenice. Unesena rečenica se smješta u klasični niz znakova (dakle, ne u promjenljivu tipa "`string`"). Za realizaciju zadatka koristiti isključivo pokazivačku aritmetiku. Nije dozvoljena upotreba funkcija iz biblioteka "cstring" niti "string", kao ni upotreba indeksiranja (uključujući i njegovu trivijalnu simulaciju koja podrazumijeva pisanje "`*(a+n)`" umjesto "`a[n]`").

6.3 Napišite funkciju sa jednim parametrom n, koja dinamički alocira niz od n cijelih brojeva, popunjava ga sa prvih n stepena broja 2 (stepeni dvojke su redom 1, 2, 4, 8, 16 itd.) i vraća kao rezultat pokazivač na prvi element tako alociranog niza. U slučaju da je $n \leq 0$, funkcija treba da baci tekst "Broj elemenata mora biti pozitivan" kao izuzetak, a u slučaju da alokacija ne uspije, funkcija treba da baci tekst "Alokacija nije uspjela" kao izuzetak. Napisanu funkciju iskoristite u testnom programu u kojem se sa tastature unosi broj n, zatim poziva napisana funkcija za kreiranje traženog niza, koji se potom ispisuje na ekran, i na kraju se vrši oslobađanje prostora koji je zauzeo niz. Pored toga, u glavnom programu treba predvidjeti hvatanje svih izuzetaka koji bi funkcija eventualno mogla baciti.

6.4 Napišite generičku funkciju čiji je parametar vektor elemenata proizvoljnog tipa. Funkcija prvo treba da dinamički kreira niz čiji su elementi istog tipa kao i zadani vektor i koji sadrži isti broj elemenata kao i zadani vektor. Funkcija zatim treba prepisati sve elemente vektora u kreirani niz u obrnutom poretku (tj. prvi element vektora treba da postane posljednji element niza itd.) i vratiti pokazivač na prvi element tako kreiranog niza kao rezultat. Napišite i mali testni program u kojem ćete demonstrirati kako se upotrebljava napisana funkcija na vektoru realnih brojeva čiji se elementi unose sa tastature.

6.5 Napišite generičku funkciju koja kao svoj parametar prima dvodimenzionalnu strukturu (recimo matricu, ali pri čemu broj elemenata u svakom redu ne mora nužno biti isti) predstavljenu kao vektor vektōrâ čiji su elementi proizvoljnog tipa. Funkcija prvo treba da dinamički alocira prostor za dvodimenzionalnu strukturu identičnog oblika kao i parametar, zatim da u nju prepiše elemente dvodimenzionalne strukture predstavljene parametrom i, konačno, da kao rezultat vrati dvojni pokazivač preko kojeg se može izvršiti pristup elementima ove strukture. U slučaju da dođe do problema sa alokacijom memorije, funkcija treba baciti izuzetak. Pri tome, ni u kom slučaju ne smije doći do curenja memorije. Napisanu funkciju testirati u testnom programu koji sa tastature unosi elemente matrice formata 3 × 3 organizirane kao vektor vektōrâ, a nakon toga poziva napisanu funkciju sa ciljem kreiranja odgovarajuće dinamičke matrice i, konačno, ispisuje elemente tako kreirane dinamičke matrice na ekran i oslobađa zauzetu memoriju. U testnom programu predvidjeti i eventualno hvatanje bačenih izuzetaka.

6.7 Napisati program prvo traži da se sa tastature unese rečenica, a koji zatim ispisuje posljednju riječ te rečenice. Dobro obratiti pažnju na to da rečenica može imati samo jednu riječ, kao i da se nakon posljednje riječi u rečenici mogu nalaziti razmaci (koje ne treba ispisivati).

6.8 Napisati program koji prvo traži da se sa tastature unese rečenica, a zatim ispisuje svaku riječ te rečenice u posebnom redu, kao i informaciju koliko ta rečenica ima slova (ne računajući interpunkcijske znakove) i riječi. Obratiti pažnju da riječi rečenice mogu biti razdvojene sa više od jednog razmaka, kao i da se na samom početku i na samom kraju rečenice mogu također nalaziti razmaci.

6.9 Napisati program koji će za rečenicu unesenu sa tastature ispisati da li je palindrom ili nije. Pod palindromima smatramo riječi ili rečenice koje se isto čitaju sa obe strane (npr. “kapak”). Prilikom ispitivanja treba ignorirati eventualne razmake, interpunkcijske znake i razliku između velikih i malih slova, tako da rečenica “Ana voli Milovana” treba da bude prepoznata kao palindrom, iako bukvalno pročitana sa suprotnog kraja glaso “anavoliM ilov Ana”. Još neki od poznatih palindromnih rečenica su “Udovica baci vodu”, “E, sine, ženi se”, “I jogurt ujutru goji”, “I Dara za mast sama zaradi”, “Jovi limaru gumu gura Milivoj”, itd. Historijski najpoznatiji palindrom je latinska rečenica "Sator Arepo tenet opera rotas" za koju se u srednjem vijeku smatralo da ima magična svojstva.

6.10 Napisati funkciju “`Sastavi`“ sa tri parametra “`A`”, “`B`“ i “`C`“. Sva tri parametra predstavljaju nul-terminirane nizove znakova (klasične stringove). Funkcija treba da u parametar “`C`“ smjesti string koji se dobija nadovezivanjem stringa “`A`“ na string “`B`”. Sami stringovi “`A`“ i “`B`“ treba da ostanu nepromijenjeni. Za pisanje ove funkcije nije

dozvoljeno koristiti funkcije iz biblioteke “`cstring`”. Napisati i kratki testni program u kojem ćete demonstrirati ovu funkciju.

6.11 Napisati program prvo traži da se sa tastature unese rečenica, a koji zatim ispisuje posljednju riječ te rečenice. Dobro obratiti pažnju na to da rečenica može imati samo jednu riječ, kao i da se nakon posljednje riječi u rečenici mogu nalaziti razmaci (koje ne treba ispisivati).

6.12 Napisati program koji prvo traži da se sa tastature unese rečenica, a zatim ispisuje svaku riječ te rečenice u posebnom redu, kao i informaciju koliko ta rečenica ima slova (ne računajući interpunkcijske znakove) i riječi. Obratiti pažnju da riječi rečenice mogu biti razdvojene sa više od jednog razmaka, kao i da se na samom početku i na samom kraju rečenice mogu također nalaziti razmaci.

6.13 Napisati program koji će za rečenicu unesenu sa tastature ispisati da li je palindrom ili nije. Pod palindromima smatramo riječi ili rečenice koje se isto čitaju sa obe strane (npr. “kapak”). Prilikom ispitivanja treba ignorirati eventualne razmake, interpunkcijske znake i razliku između velikih i malih slova, tako da rečenica “Ana voli Milovana” treba da bude prepoznata kao palindrom, iako bukvalno pročitana sa suprotnog kraja glaso “anavoliM ilov Ana”. Još neki od poznatih palindromnih rečenica su “Udovica baci vodu”, “E, sine, ženi se”, “I jogurt ujutru goji”, “I Dara za mast sama zaradi”, “Jovi limaru gumu gura Milivoj”, itd. Historijski najpoznatiji palindrom je latinska rečenica "Sator Arepo tenet opera rotas" za koju se u srednjem vijeku smatralo da ima magična svojstva.

6.14 Napisati funkciju “`Sastavi`“ sa tri parametra “`A`”, “`B`“ i “`C`“. Sva tri parametra predstavljaju nul-terminirane nizove znakova (klasične stringove). Funkcija treba da u parametar “`C`“ smjesti string koji se dobija nadovezivanjem stringa “`A`“ na string “`B`”. Sami stringovi “`A`“ i “`B`“ treba da ostanu nepromijenjeni. Za pisanje ove funkcije nije dozvoljeno koristiti funkcije iz biblioteke “`cstring`”. Napisati i kratki testni program u kojem ćete demonstrirati ovu funkciju.

6.15 Napisati generičku (šablonsku) funkciju “`UnosBroja`“ sa tri parametra. Funkcija treba da omogući pouzdano unošenje brojeva u program, uz potpunu kontrolu grešaka pri unosu. Prvi parametar predstavlja tekst koji se ispisuje korisniku kao obavijest da treba unijeti broj, drugi parametar predstavlja tekst koji se ispisuje korisniku kao upozorenje u slučaju da unos nije ispravan, dok treći parametar predstavlja promjenljivu u koju će se smjestiti uneseni broj. Na primjer, funkcija se može pozvati na sljedeći način:

```
UnosBroja("Unesi prvi broj: ", "Neispravan unos!\n", PrviBroj);
```

Funkcija treba da traži unos od korisnika sve dok unos ne bude ispravan. Napisanu funkciju demonstrirati u testnom programu koji od korisnika traži da unese tri koeficijenta kvadratne jednačine (pozivom napisane funkcije), a zatim računa i ispisuje njena rješenja.

6.16 Koristeći tip podataka “`string`“ definiran u istoimenoj standardnoj biblioteci, napisati program koji traži da se sa tastature unesu dvije rečenice, a zatim na ekranu ispisuje koliko svaka od rečenica sadrži znakova, koja od ove dvije rečenice dolazi prije po abecednom poretku, kao i rečenicu koja se sastoji od ove dvije rečenice sastavljene zajedno. Također,

ispisati prvu od ove dvije rečenice u obrnutom poretku (od kraja ka početku). Pri tome, za tu svrhu koristiti operacije koje su podržane za tip podataka "`string`".

6.17 Nazovimo neku riječ "korektnom" ukoliko se u njoj ne pojavljuje skupina uzastopnih suglasnika duža od 2 slova. Na primjer, riječ "badlekuvje" je korektna, a riječ "balgftekrihvnu" nije korektna u smislu date definicije, jer sadrži 2 grupe suglasnika sa više od 2 uzastopna suglasnika ("lgft" i "hvn"). Napišite funkciju koja prihvata string kao parametar, a koja kao rezultat vraća logičku vrijednost "true" ili "false", ovisno od toga da li string koji joj je proslijeđen kao parametar predstavlja korektnu riječ ili ne (u smislu gore date definicije). Napisanu funkciju demonstrirajte u testnom programu koji za riječ unesenu sa tastature ispisuje da li je korektna ili nije.

6.18 U biblioteci "algorithm" nalazi se generička funkcija "equal". Ova funkcija vraća kao rezultat logičku logičku vrijednost "true" ukoliko je blok elemenata između pokazivača p1 i p2 identičan po sadržaju bloku elemenata na koji pokazuje pokazivač p3, a u suprotnom vraća logičku vrijednost "false". Napišite sami generičku funkciju "JednakiBlokovi" koja radi posve istu stvar kao i funkcija "equal". Napisanu funkciju demonstrirajte u testnom programu koji na nekom primjeru demonstrira da napisana funkcija radi isto kao i funkcija "equal".

6.18 U biblioteci "algorithm" nalazi se generička funkcija "find if". Ova funkcija vraća kao rezultat pokazivač na prvi element u bloku između pokazivača p1 i p2 za koje funkcija f vraća kao rezultat "true" kad joj se proslijedi kao argument (ili p2 ukoliko takav element ne postoji), pri čemu je sintaksa poziva ove funkcije "find if(p1, p2, f)". Napišite sami generičku funkciju "Nadji" koja prima potpuno iste parametre i obavlja istu funkciju kao i funkcija "find if". Napisanu funkciju demonstrirajte u testnom programu koji na nekom primjeru demonstrira da napisana funkcija radi isto kao i funkcija "find if".

6.19 U biblioteci "algorithm" nalazi se generička funkcija "replace if". Ova funkcija zamjenjuje sve elemente između pokazivača p1 i p2 za koje funkcija f vraća kao rezultat "true" kad joj se proslijede kao argument, sa elementima sa vrijednošću v, pri čemu je sintaksa poziva ove funkcije "replace if(p1, p2, f, v)". Napišite sami generičku funkciju "Zamijeni" koja prima potpuno iste parametre i obavlja istu funkciju kao i funkcija "replace if". Napisanu funkciju demonstrirajte u testnom programu koji na nekom primjeru demonstrira da napisana funkcija radi isto kao i funkcija "replace if".

6.20 U biblioteci "algorithm" nalazi se generička funkcija "count if". Ova funkcija vraća kao rezultat broj elemenata u bloku između pokazivača p1 i p2 za koje funkcija f vraća kao rezultat "true" kad joj se proslijede kao argument, pri čemu je sintaksa poziva ove funkcije "count if(p1, p2, f)". Napišite sami generičku funkciju "Prebroji" koja prima potpuno iste parametre i obavlja istu funkciju kao i funkcija "count if". Napisanu funkciju demonstrirajte u testnom programu koji na nekom primjeru demonstrira da napisana funkcija radi isto kao i funkcija "count if".

6.21 U biblioteci "algorithm" nalazi se generička funkcija "remove copy if". Ova funkcija kopira blok elemenata između pokazivača p1 i p2 na lokaciju određenu pokazivačem p3, uz izbacivanje elemenata za koji funkcija f vraća kao rezultat "true" kad joj se proslijede kao argument (i vraća kao rezultat pokazivač koji pokazuje tačno iza

posljednjeg elementa odredišnog bloka), pri čemu je sintaksa poziva ove funkcije "remove copy if(p1, p2, p3, f)". Napišite sami generičku funkciju "Ukloni" koja prima potpuno iste parametre i obavlja potpuno istu funkciju kao i funkcija "remove copy if".

6.22 U biblioteci "algorithm" nalazi se generička funkcija "equal". Ova funkcija vraća kao rezultat logičku logičku vrijednost "true" ukoliko je blok elemenata između pokazivača p1 i p2 identičan po sadržaju bloku elemenata na koji pokazuje pokazivač p3, a u suprotnom vraća logičku vrijednost "false". Napišite sami generičku funkciju "RazlicitiBlokovi" koja radi suprotnu stvar u odnosu na funkciju "equal", odnosno vraća "true" ukoliko se blokovi razlikuju, a "false" ukoliko su identički (pri tome, za realizaciju ne smijete koristiti funkciju "equal"). Napisanu funkciju demonstrirajte u testnom programu koji na nekom primjeru demonstrira da napisana funkcija radi ono što se od nje traži-

6.23 Napišite generičku funkciju "BrojZajednickih" sa četiri parametra "p1", "p2", "p3" i "p4" koji su "u duhu" parametara funkcija iz biblioteke "algorithm". Parametri "p1" i "p2" omeđuju jedan blok podataka (tj. "p1" pokazuje na početak bloka a "p2" tačno iza kraja bloka), dok "p3" i "p4" omeđuju drugi blok podataka. Elementi oba bloka su proizvoljnog ali istog tipa. Funkcija treba da kao rezultat vrati broj elemenata koji se javljaju kao zajednički elementi i u jednom i u drugom bloku. Na primjer, neka su date sljedeće deklaracije:

```
int a[8] = {3, 7, 2, 3, 1, 5, 5, 2};
int b[10] = {4, 6, 7, 8, 1, 3, 1, 6, 4, 7};
```

Tada sljedeća naredba

```
cout << BrojZajednickih(a, a + 8, b, b + 10);
```

treba da ispiše broj 3, jer se tri elementa (3, 7 i 1) pojavljuju u oba niza. Funkciju bi u cijelosti trebalo napisati korištenjem pokazivačke aritmetike, tj. bez upotrebe indeksiranja. Napišite i kratki testni program u kojem ćete demonstrirati napisanu funkciju.

6.24 Napišite program koji će napraviti vektor realnih brojeva popunjen kvadratima prvih 100 prirodnih brojeva. Nakon toga, bez upotrebe petlji na mjesto svakog broja u vektoru treba upisati ostatak pri dijeljenju tog broja sa brojem 42, pripremiti vektor za binarno pretraživanje i binarnim pretraživanjem za broj unesen sa tastature ustanoviti da li se nalazi u nizu ili ne.

6.25 Pretpostavimo da želimo sortirati niz kompleksnih brojeva tako da kompleksni broj z1 dolazi ispred kompleksnog broja z2 ako i samo ako je imaginarni dio od z1 manji od imaginarnog dijela od z2, ili ako su imaginarni dijelovi od z1 i z2 jednaki, ali je realni dio od z1 manji od realnog dijela od z2. Definirajte odgovarajuću funkciju kriterija, i mali testni program u kojem ćete pokazati kako biste deklarirali niz kompleksnih brojeva, napunili ga vrijednostima unesenim sa tastature i sortirali ga u traženi poredak korištenjem funkcije "sort" iz biblioteke "algorithm".

6.26 Napišite program koji će od korisnika tražiti da unosi kompleksne brojeve, završno sa kompleksnim brojem 0. Unesene brojeve treba stavljati u vektor kompleksnih brojeva

koji će se kasnije sortirati po apsolutnoj vrijednosti (modulu) u rastući poredak, koristeći ugrađenu funkciju “sort” iz biblioteke “algorithm” (uz prethodno definiranu funkciju kriterija). Nakon toga, sortirani niz treba kopirati u dva dinamički alocirana niza od kojih će jedan sadržavati realne dijelove, a drugi kompleksne dijelove brojeva iz vektora. Na kraju je potrebno ispisati realni dio onog kompleksnog broja koji ima najmanji modul.

6.27 Napišite program u kojem ćete sa tastature unositi niz rečenica, sve dok se kao rečenica ne unese prazna rečenica (tj. samo pritisne ENTER bez ikakvog prethodnog unosa). Unesene rečenice treba smještati u vektor čiji su elementi dinamički stringovi (tj. tipa “string”). Nakon toga, uneseni vektor treba sortirati u rastući abecedni poredak, ignorirajući razliku između velikih i malih slova i ispisati ga na ekran. Na kraju, program za unesenu rečenicu sa tastature treba ispisati da li se ona nalazi ili ne nalazi u prethodno unesenom spisku rečenica korištenjem postupka binarnog pretraživanja.

6.28 Napišite generičku funkciju koja služi za dinamičku alokaciju matrica čiji su elementi proizvoljnog tipa. Funkcija ima 4 parametra, od kojih se četvrti može izostaviti. Drugi i treći parametar su brojredova i broj kolona matrice respektivno, dok je četvrti parametar inicijalna vrijednost kojom sepopunjavaju elementi matrice. Funkcija treba da izvrši dinamičku alokaciju prostora za pamćenje elemenata matrice, zatim da popuni elemente matrice zadanom inicijalnom vrijednošću i, konačno, dau prvi parametar funkcije smjesti dvojni pokazivač preko kojeg se može pristupiti elementima matrice.U slučaju da se četvrti parametar izostavi, za inicijalizaciju elemenata matrice treba koristiti odgovarajuću podrazumijevanu vrijednost za tip elemenata matrice. U slučaju da dinamička alokacijane uspije, u prvi parametar funkcije treba smjestiti 0 (tj. nul-pokazivač). Ova funkcija ne vraćanikakvu vrijednost, niti baca izuzetke (tj. svi izuzeci koji bi se mogli pojaviti trebaju biti uhvaćeni iobrađeni unutar same funkcije). Vodite računa da u slučaju neuspješne alokacije ni u kom slučaju ne dođe do curenja memorije. Napišite i mali isječak glavnog programa (ne treba sve) u kojem ćete demonstrirati kako biste pozvali ovu funkciju za kreiranje matrice realnih brojeva formata 10 × 10 i testirali da li je alokacija matrice uspjela ili ne.

Funkcije (bez parametara)

8.1 Napišite funkciju “UnesiBroj” bez parametara, koja od korisnika očekuje da unese neki realni broj. Ukoliko je zaista unesen realan broj, funkcija treba da ga vrati kao rezultat iz funkcije. Ukoliko nije unesen broj, funkcija treba ispisati “Neispravan unos. Molimo, pokušajte ponovo.” i ponovo tražiti broj, sve dok se zaista ne unese ispravan broj (prema tome, funkcija ne završava dok se ne unese ispravan broj). Napisanu funkciju iskoristite u programu koji traži da se unesu koeficijenti a, b i c linearne jednačine ax + b = c, a koji zatim nalazi i ispisuje njeno rješenje, pod uvjetom da ono postoji, ili odgovarajuću poruku u suprotnom.

8.2 Napišite funkciju koja ima jedan cjelobrojni parametar. Funkcija treba da vrati kao rezultat logičku vrijednost “true” ukoliko je broj simetričan, odnosno ukoliko se isto čita sa obje strane (na primjer, broj 13431 je simetričan). U suprotnom, funkcija vraća kao rezultat logičku vrijednost “false”.

Napisanu funkciju trebate demonstrirati u kratkom testnom programu na brojevima koji se unose sa tastature. Program za svaki uneseni broj treba da ispise da li je simetričan ili nije. Programtreba da završi rad kada se kao broj unese 0.

Funkcije (prenos po vrijednosti)

4.1 Napišite funkciju "`stepen`" sa dva parametra `x` i `n` koja računa i vraća kao rezultat x^n bez korištenja funkcije "`pow`" pri čemu su `x` i `n` cijeli brojevi proizvoljnog znaka (posebno obratite pažnju da n može biti i negativan). Drugim riječima, napišite funkciju koja radi slično kao funkcija "`pow`" iz biblioteke "`cmath`" samo za cijele brojeve, bez upotrebe ijedne funkcije iz biblioteke "`cmath`". Na primjer, ukoliko se izvrše slijedeće naredbe

```
cout << stepen (2, 5) << endl;
cout << stepen (10, -3)
cout << stepen (4) << endl;
```

na ekranu treba da budu ispisani brojevi 32 (25), 0.001 (10–3) i 16 (42). Također napišite i kratki glavni program (funkciju "`main`") u kojoj ćete demonstrirati napisanu funkciju na brojevima koji se unose sa tastature.

4.2 Još je Heronu prije 2000 godina bio poznat sljedeći postupak (algoritam) za računanje kvadratnog korijena proizvoljnog broja x: formira se niz brojeva a0, a1, a2 itd. po sljedećem pravilu:

$a_0 = 1$; $a_{k+1} = (a_k + x / a_k) / 2$ za $k > 0$

Ovaj niz konvergira vrlo brzo ka korijenu iz `x`. U praksi je dovoljno izračunati samo nekoliko elemenata ovog niza, jer se vrlo brzo elementi počinju praktično ponavljati (sa onolikom tačnošću koliku dopušta realni tip podataka). Tada postupak možemo obustaviti, i posljednju izračunatu vrijednost proglasiti traženim korijenom. Napišite funkciju "`korijen`" koja računa korijen svog argumenta Heronovim postupkom. Za realizaciju funkcije ne koristiti nizove. Napisanu funkciju testirajte u glavnom programu koji za argument unesen sa tastature ispisuje vrijednosti korijena korištenjem funkcije "`korijen`" kao i korištenjem ugrađene funkcije "`sqrt`" (naravno, rezultati treba da budu isti).

4.3 Napišite funkciju koja kao parametar prima vektor realnih brojeva. Funkcija treba da ispita da li elementi vektora čine slijed koji se periodično ponavlja ili ne. Na primjer, za vektor čiji su elementi 5, 9, 7, 2, 5, 9, 7, 2, 5, 9, 7 uočavamo da njegovi elementi čine slijed koji se periodično ponavlja sa dužinom perioda 4. Ukoliko elementi vektora čine periodičan slijed, funkcija treba da vrati kao rezultat dužinu perioda, a u suprotnom, funkcija treba da vrati nulu kao rezultat.

Napisanu funkciju demonstrirajte u glavnom programu u kojem ćete unositi elemente sa tastature u neki vektor sve dok se sa tastature ne unese nula, koja označava kraj unosa (tu nulu ne treba smjestiti u vektor). Nakon završetka unosa, program poziva napisanu funkciju sa ciljem da utvrdi da li se elementi periodično ponavljaju ili ne, nakon čega ispisuje odgovarajući komentar na ekranu (informaciju o dužini perioda, ili da elementi ne čine periodičan slijed). Napomenimo da nije unaprijed poznato koliko će korisnik unijeti elemenata prije nego što unese nulu kao oznaku završetka unosa.

4.4 Napišite funkciju "IzbaciNule" koja ima kao parametar neki prirodan broj, a koja daje kao rezultat isti taj broj, samo iz kojeg su izbačene sve cifre koje su nule. Na primjer, ukoliko se kao parametar funkciji proslijedi broj 35020040, rezultat funkcije treba da bude 3524. Ukoliko broj ne sadrži niti jednu nulu kao svoju cifru, onda je on sam ujedno i rezultat funkcije. Napisanu funkciju trebate demonstrirati u kratkom testnom

programu na brojevima koji se unose sa tastature. Program treba da završi rad kada se kao broj unese 0.

4.5 Napišite funkciju "Podbroj" sa dva parametra "N" i "Parnost", od kojih je prvi cjelobrojnog a drugi logičkog tipa. Ukoliko parametar "Parnost" ima vrijednost "true", funkcija treba da kao rezultat vrati broj koji se sastoji samo od parnih cifara broja "N" (uzetih u istom poretku), a ukoliko parametar "Parnost" ima vrijednost "false", funkcija treba da kao rezultat vrati broj koji se sastoji samo od neparnih cifara broja "N". Na primjer, nakon izvrsenja naredbi

```
cout << Podbroj(3427816, true) << endl;
cout << Podbroj(3427816, false) << endl;
```

treba da se ispišu brojevi 4286 i 371.

4.6 Napisanu funkciju trebate demonstrirati na kratkom testnom programu koji će za broj unesen sa tastature ispisati brojeve sastavljene od njegovih parnih odnosno neparnih cifara. Proces treba da se ponavlja sve dok korisnik ne unese nulu.

4.7 Neka je n prirodan broj, i neka je n1 proizvod cifara broja n, n2 proizvod cifara broja n1, n3 proizvod cifara broja n2 itd. Najmanji broj k za koji vrijedi da je nk jednocifren broj u teoriji brojeva se naziva multiplikativna otpornost broja n. Na primjer, multiplikativna otpornost broja 6788 iznosi 6, jer je:

$n_1 = 6 * 7 * 8 * 8 = 2688$
$n_2 = 2 * 6 * 8 * 8 = 768$
$n_3 = 7 * 6 * 8 = 336$
$n_4 = 3 * 3 * 6 = 54$
$n_5 = 5 * 4 = 20$
$n_6 = 2 * 0 = 0$

Napišite funkciju "MultOtpornost" koja kao parametar prima prirodan broj n a vraća kao rezultat njegovu multiplikativnu otpornost.

Napisanu funkciju trebate demonstrirati na kratkom testnom programu koji će ispisati sve brojeve u opsegu od a do b uključivo koji imaju zadanu multiplikativnu otpornost k pri cemu se vrijednosti a, b i k zadaju putem tastature. Na primjer, ukoliko se za a, b i k unesu vrijednosti 9985, 10002 i 3, treba da se ispišu brojevi 9987, 9988, 9989, 9992, 9993, 9994, 9995, 9998 i 9999, jer ti brojevi u zadanom intervalu imaju multiplikativnu otpornost 3.

Napomena: Za sve razumne vrijednosti n, multiplikativna otpornost ima uglavnom jednocifrenu vrijednost, pa o tome vodite računa kada budete testirali program. Brojevi koji imaju multiplikativnu otpornost veću od 9 izrazito su rijetki!

4.8 Napišite funkcije "`SviElementiSuJednaki`" i "`SviElementiSuRazliciti`". I jedna i druga funkcija primaju kao parametar jedan vektor realnih brojeva, a vraćaju kao rezultat logičku vrijednost (tj. vrijednost tipa "bool"). Funkcija "`SviElementiSuJednaki`" vraća vrijednost "true" ako i samo ako su svi elementi vektora međusobno identični, a u suprotnom vraća "false". Funkcija "`SviElementiSuRazliciti`" vraća vrijednost "true" ako i samo ako su svi

elementi vektora međusobno različiti (tj. ukoliko nikoja dva elementa nisu međusobno jednaka), a u suprotnom vraća "false". Napisane funkcije testirajte u glavnom programu ("main" funkciji) u kojem se prvo sa tastature unosi prirodan broj n i n elemenata vektora, a nakon toga pozivaju napisane funkcije sa ciljem da se utvrdi da li su svi elementi vektora jednaki odnosno različiti (naravno, rezultate obavljene analize treba prikazati na ekranu u vidu odgovarajućih poruka).

4.9* Napišite funkciju koja kao parametar prima vektor cijelih brojeva, a koja vraća logičku vrijednost "tačno" ili "netačno", u ovisnosti da li u vektoru ima parnih brojeva ili ne. Napisanu funkciju demonstrirajte u testnom programu u kojem se sa tastature unosi prirodan broj n a zatim n cijelih brojeva, koji se unose u vektor. Nakon unosa, program treba pozvati napisanu funkciju sa ciljem da utvrdi ima li među unesenim brojevima parnih brojeva ili ne i ispisati odgovarajuću poruku, ovisno od rezultata funkcije.

4.10* Napišite funkciju koja kao parametar prima vektor cijelih brojeva, a koja vraća logičku vrijednost "tačno" ili "netačno", u ovisnosti da li u vektoru ima potpunih kvadrata (tj. brojeva koji se mogu napisati kao kvadrati nekog drugog prirodnog broja) ili ne. Napisanu funkciju demonstrirajte u testnom programu u kojem se sa tastature unosi prirodan broj n a zatim n cijelih brojeva, koji se unose u vektor. Nakon unosa, program treba pozvati napisanu funkciju sa ciljem da utvrdi ima li među unesenim brojevima potpunih kvadrata ili ne i ispisati odgovarajuću poruku, ovisno od rezultata funkcije.

4.11* Napišite funkciju koja kao parametar prima vektor cijelih brojeva, a koja vraća logičku vrijednost "tačno" ili "netačno", u ovisnosti da li u vektoru ima trocifrenih brojeva ili ne. Napisanu funkciju demonstrirajte u testnom programu u kojem se sa tastature unosi prirodan broj n a zatim n cijelih brojeva, koji se unose u vektor. Nakon unosa, program treba pozvati napisanu funkciju sa ciljem da utvrdi ima li među unesenim brojevima trocifrenih brojeva ili ne i ispisati odgovarajuću poruku, ovisno od rezultata funkcije.

4.12* Napišite funkciju koja kao parametar prima vektor cijelih brojeva, a koja vraća logičku vrijednost "tačno" ili "netačno", u ovisnosti da li u vektoru ima brojeva koji su stepeni broja 2 ili ne (ukoliko želite, možete koristiti funkciju "log" iz bibbioteke "cmath", ali to nije neophodno). Napisanu funkciju demonstrirajte u testnom programu u kojem se sa tastature unosi prirodan broj n a zatim n cijelih brojeva, koji se unose u vektor. Nakon unosa, program treba pozvati napisanu funkciju sa ciljem da utvrdi ima li među unesenim brojevima stepena dvojke ili ne i ispisati odgovarajuću poruku, ovisno od rezultata funkcije.

4.13* Napišite funkciju koja kao parametar prima jedan prirodan broj, a kao rezultat vraća vektor čiji su elementi svi djelioci broja zadanog kao parametar. Napisanu funkciju demonstrirajte u glavnom programu u kojem se sa tastature unosi prirodan broj n i koji nakon toga poziva napisanu funkciju sa ciljem generiranja vektora čiji su elementi djelioci broja n, čije elemente na kraju treba ispisati na ekran (međusobno razdvojene razmakom).

4.14* Napišite funkciju koja kao parametar prima jedan prirodan broj, a kao rezultat vraća vektor čiji su elementi svi prirodni brojevi koji su potpuni kvadrati (tj. koji se mogu napisati kao kvadrat nekog drugog prirodnog broja), a koji su manji od broja zadanog kao parametar. Napisanu funkciju demonstrirajte u glavnom programu u kojem se sa tastature unosi prirodan broj n i koji nakon toga poziva napisanu funkciju sa ciljem generiranja vektora čiji su elementi svi potpuni kvadrati manji od n, čije elemente na kraju treba ispisati na ekran (međusobno razdvojene razmakom).

4.15* Napišite funkciju koja kao parametar prima jedan prirodan broj, a kao rezultat vraća vektor čiji su elementi svi stepeni broja 2, a koji su manji od broja zadanog kao parametar. Napisanu funkciju demonstrirajte u glavnom programu u kojem se sa tastature unosi prirodan broj n i koji nakon toga poziva napisanu funkciju sa ciljem generiranja vektora čiji su elementi svi stepeni dvojke manji od n, čije elemente na kraju treba ispisati na ekran (međusobno razdvojene razmakom).

4.16* Napišite funkciju koja kao parametar prima jedan prirodan broj, a kao rezultat vraća vektor čiji su elementi svi trocifreni brojevi koji su djeljivi sa brojem koji je zadan kao parametar. Napisanu funkciju demonstrirajte u glavnom programu u kojem se sa tastature unosi prirodan broj n i koji nakon toga poziva napisanu funkciju sa ciljem generiranja vektora čiji su elementi svi trocifreni brojevi djeljivi sa n, čije elemente na kraju treba ispisati na ekran (međusobno razdvojene razmakom).

4.17* Napišite funkciju koja ima jedan cjelobrojni parametar (nazovimo ga “n”), koja kao rezulat vraća vektor koji se sastoji od svih prostih faktora broja “n”, pri čemu se svaki faktor pojavljuje onoliko puta koliko se puta on pojavljuje u faktorizaciji broja na proste faktore. Napišite i mali testni program u kojem ćete demonstrirati kako se napisana funkcija može iskoristiti za ispis prostih faktora broja koji se unosi sa tastature.

4.18* Napišite funkciju koja ima dva parametra, od kojih je prvi niz realnih brojeva, a drugi broj elemenata u nizu. Funkcija treba da kreira vektor realnih brojeva koji sadrži isti broj elemenata kao i zadani niz, da prepiše sve elemente niza u kreirani vektor u obrnutom poretku (tj. prvi element niza treba da postane posljednji element vektora, drugi element niza treba da postane pretposljednji element vektora, itd.), i da vrati tako popunjeni vektor kao rezultat. Napisanu funkciju demonstrirajte u glavnom programu koji prvo sa tastature unosi elemente niza kapaciteta 10 elemenata. Nakon toga, program poziva napisanu funkciju sa ciljem generiranja vektora čiji su elementi u obrnutom poretku u odnosu na elemente unesenog niza. Na kraju, program ispisuje elemente generiranog vektora na ekran (međusobno razdvojene razmakom).

4.19* Napišite funkciju koja ima dva parametra, od kojih je prvi niz cijelih brojeva, a drugi broj elemenata u nizu. Funkcija treba da kreira vektor cijelih brojeva koji sadrži iste elemente kao i zadani niz, ali tako da prvo idu parni, a zatim neparni brojevi (međusobni poredak parnih odnosno neparnih brojeva treba da bude isti kao i u izvornom nizu), i da vrati tako popunjeni vektor kao rezultat. Na primjer, ukoliko je niz sadržavao redom elemente 3, 5, 2, 7, 6, 4, 1, 8, 5, 9, 4, 3 i 4, vraćeni vektor treba da sadrži redom elemente 2, 6, 4, 8, 4, 4, 3, 5, 7, 1, 5, 9 i 3. Napisanu funkciju demonstrirajte u glavnom programu koji prvo sa tastature unosi elemente niza kapaciteta 10 elemenata. Nakon toga, program poziva napisanu funkciju sa ciljem generiranja vektora čiji su elementi u poretku kako je gore opisano u odnosu na

elemente unesenog niza. Na kraju, program ispisuje elemente generiranog vektora na ekran (međusobno razdvojene razmakom).

4.20* Napišite funkciju koja kao parametar prima vektor cijelih brojeva, a koja kao rezultat vraća novi vektor čiji su elementi broj cifara odgovarajućih elemenata vektora koji je zadan kao parametar. Na primjer, ukoliko se funkciji proslijedi vektor čiji su elementi 32, 459, 72, 8, 24, 771 i 13, funkcija treba da kao rezultat vrati vektor čiji su elementi 2, 3, 2, 1, 2, 3 i 2. Napisanu funkciju demonstrirajte u kratkom testnom programu u kojem se sa tastature prvo unosi prirodan broj n, a zatim n elemenata vektora. Program tada poziva napisanu funkciju sa ciljem da kreira novi vektor koji sadrži brojeve cifara unesenih elemenata i ispisuje elemente novokreiranog vektora na ekran (međusobno razdvojene razmakom).

4.21* Napišite funkciju koja kao parametar prima vektor cijelih brojeva, a koja kao rezultat vraća novi vektor čiji su elementi sume cifara odgovarajućih elemenata vektora koji je zadan kao parametar. Na primjer, ukoliko se funkciji proslijedi vektor čiji su elementi 32, 459, 72, 8, 24, 771 i 13, funkcija treba da kao rezultat vrati vektor čiji su elementi 5, 18, 9, 8, 6, 15 i 4. Napisanu funkciju demonstrirajte u kratkom testnom programu u kojem se sa tastature prvo unosi prirodan broj n, a zatim n elemenata vektora. Program tada poziva napisanu funkciju sa ciljem da kreira novi vektor koji sadrži sume cifara unesenih elemenata i ispisuje elemente novokreiranog vektora na ekran (međusobno razdvojene razmakom).

4.22* Napišite funkciju sa dva parametra, pri čemu je prvi parametar neki vektor cijelih brojeva, a drugi parametar cijeli broj. Funkcija treba da vrati kao rezultat novi vektor čiji su elementi oni elementi vektora zadanog kao prvi parametar čija je suma cifara veća od broja zadanog drugim parametrom. Napišite i mali testni program u kojem ćete demonstrirati kako se upotrebljava napisana funkcija.

4.23# Napišite funkciju "`OdstraniDuplikate`" i "`BrojDuplikata`" koja kao parametar primaju vektor cijelih brojeva "`v`". Funkcija "`OdstraniDuplikate`" treba da kao rezultat vrati novi vektor u koji će biti prepisani svi elementi vektora "v" izostavljajući duplikate (tj. brojeve koji se ponavljaju). Funkcija "`BrojDuplikata`" treba da vrati kao rezultat broj brojeva koji seponavljaju. Na primjer, ukoliko vektor "v" ima elemente 7, 10, 4, 2, 4, 4, 5, 6, 7, 3, 9, 1, 8, 6 i 7, funkcija "`OdstraniDuplikate`" treba da vrati vektor sa elementima 7, 10, 4, 2, 5, 6, 3, 9, 1 i 8, dok funkcija "`BrojDuplikata`" treba da vrati broj 3, jer se 3 broja ponavljaju (to su brojevi 7, 4 i 6). Napisane funkcije testirajte u programu koji traži da se prvo unese prirodan broj n, a nakon toga elementi nekog vektora koji ima n cjelobrojnih elemenata. Potom program treba da ispiše elemente vektora iz kojeg su odstranjeni brojevi koji se ponavljaju, kao i broj elemenata koji se ponavljaju.

4.24# Napišite funkcije "`ProstiBrojevi`", "`ProstiBrojeviBlizanci`", "`ProstiFaktori`" i "`Djelioci`". Sve četiri funkcije imaju jedan cjelobrojni parametar (nazovimo ga "n"). Funkcija "`ProstiBrojevi`" treba da kao rezultat vrati vektor koji se sastoji od prvih "n" prostih brojeva. Funkcija "`ProstiBrojeviBlizanci`" treba da vrati vektor koji ima "2n" elemenata, a koji se sastoji od prvih "n" parova tzv. prostih brojeva blizanaca, koji predstavljaju proste brojeve koji se međusobno razlikuju za 2 (npr. 3 i 5, 11 i 13, 29 i 31, itd.). Funkcija "`ProstiFaktori`" treba da kao rezultat vrati vektor koji se sastoji od svih prostih faktora broja "n", pri čemu se svaki faktor pojavljuje onoliko puta

koliko se puta on pojavljuje u faktorizaciji broja na proste faktore. Funkcija "`Djelioci`" treba da kao rezultat vrati vektor koji se sastoji od svih djelilaca broja "n". Napisane funkcije testirajte u malom testnom programu u kojem se sa tastature unosi prirodan broj n, a zatim pozivaju napisane funkcije sa ciljem generiranja odgovarajućih vektora koji se na kraju ispisuju na ekran (uz odgovarajuće poruke koje opisuju šta ti vektori predstavljaju).

4.25 Napisati funkciju "`Trougao`" sa dva parametra "`Visina`" i "`Znak`" koja iscrtava na ekranu jednakokraki trougao sa osnovicom okrenutom nadolje čija unutrašnjost *nije ispunjena*. Visina trougla (u znakovima) određena je prvim parametrom, a znak koji se koristi za iscrtavanje određen je drugim parametrom. Na primjer, ako funkciju pozovemo sa

```
Trougao (5, '#');
```

na ekranu treba da dobijemo sljedeći prikaz:

```
    #
   # #
  #   #
 #     #
#########
```

Predvidjeti i mogućnost da se drugi parametar može izostaviti. U tom slučaju kao znak za iscrtavanje treba koristiti zvjezdicu. Obavezno napisati i kratki testni program ("`main`" funkciju) u kojem ćete demonstrirati napisanu funkciju.

4.26 Napisati program koji treba da ponudi korisniku sljedeće tri opcije:

a) Rješavanje linearne jednačine
b) Rješavanje kvadratne jednačine
c) Kraj programa

Ukoliko korisnik unese opciju a), program treba da pita korisnika za vrijednosti koeficijenata a i b i da izračuna i ispiše rješenje jednačine $a\,x + b = 0$. Ukoliko korisnik unese opciju b), program treba da pita korisnika za vrijednosti koeficijenata a, b i c i da izračuna i ispiše rješenja jednačine $a\,x^2 + b\,x + c = 0$. Ukoliko korisnik unese opciju c), program treba da završi sa radom. Ukoliko korisnik unese bilo šta drugo, treba ispisati poruku da je unos pogrešan. Nakon što se obavi proračun u opcijama a) i b), korisniku treba ponovo ponuditi izbor, pri čemu se postupak ponavlja sve dok korisnik ne zatraži kraj. Program treba realizirati isključivo korištenjem modularnog pristupa, tj. cjeline za ispis menija, rješavanje linearne i kvadratne jednačine treba realizirati kao posebne module (funkcije) koji će se pozivati iz glavne funkcije. Pri tome, glavnu funkciju treba napisati *ispred* preostale tri funkcije. Sve tri pomoćne funkcije ne zahtijevaju nikakve parametre, i ne vraćaju nikakav rezultat, osim funkcije za ispis menija, koja treba da kao rezultat vrati broj koji odgovara rednom broju opcije koju je korisnik izabrao (1, 2 ili 3).

4.27 Napisati funkciju "`rimski`" koja ima jedan parametar "`n`" koji treba biti prirodan broj manji od 4000 i koja ispisuje odgovarajući broj u rimskoj notaciji. Na primjer, ukoliko se pozove

```
rimski(1429);
```

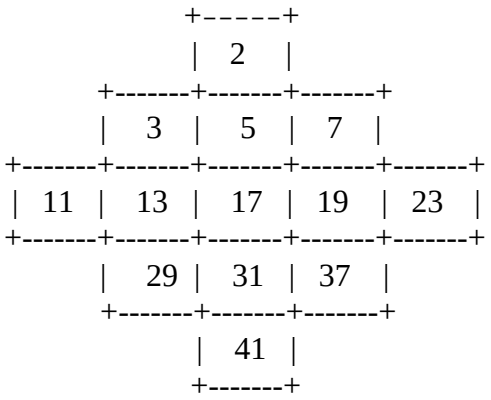
na ekranu treba da se ispiše MCDXXIX. Napisati i kratki glavni program (funkciju "`main`") u kojem ćete demonstrirati napisanu funkciju. Glavni program treba tražiti od korisnika unos broja sa tastature, ispisati njegov rimski ekvivalent (ukoliko je broj u propisanom opsegu) i ponavljati taj postupak sve dok korisnik ne unese nulu (što označava kraj rada).

4.28 Napišite funkciju "*AnalizaBroja*" koja treba da ima 4 parametra "*N*", "*BrojCifara*", "*BrojParnihCifara*" i "*RasponCifara*". Funkcija treba da u parametru "*N*" pronađe redom ukupan broj cifara, broj parnih cifara i raspon cifara odnosno razliku između najveće i najmanje cifre po vrijednosti, i da smjesti pronađene vrijednosti respektivno u parametre "*BrojCifara*", "*BrojParnihCifara*" i "*RasponCifara*". Na primjer, naredba

```
Cifre(37212646, A, B, C);
```

treba da u promjenljive "`A`", "`B`" i "`B`" (pod uvjetom da su propisno deklarirane) smjesti brojeve 8, 5 i 6, jer broj 37212646 ima 8 cifara, od kojih je 5 parnih cifara, dok su najveća i najmanj cifra 1 i 7, tako da njihov raspon iznosi 6. Funkcija ne smije koristiti nikakve pomoćne nizove ili vektore (samo individualne promjenljive). Napišite i kratki testni program u kojem ćete demonstrirati napisanu funkciju.

4.29 Napisati program koji traži da se unese prirodan broj n <10, a koji zatim printa proste brojeve, kao na primjer za *n=3:*



Program treba napisati na modularan način, korištenjem funkcija.

4.30 Napisati funkciju koja računa vrijednost funkcije $f(x, n)$ definirane sljedećim izrazom, pri čemu su x i n realni argumenti:

$$f(x, n) = \sum_{i=0}^{n} \frac{x^i}{\sqrt{x(x+i) - (-1)^i i^2}}$$

U slučaju da funkcija nije definirana, ona treba da baci izuzetak. To se može desiti u tri slučaja: ako n nije prirodan broj ili nula (tada suma nema smisla), zatim ukoliko se pod korijenom pojavi negativan broj, i konačno, ukoliko nazivnik dobije vrijednost nula. Za svaki od ova tri slučaja treba baciti različite tipove izuzetaka. Funkciju treba testirati u glavnom programu koji u beskonačnoj petlji traži da se sa tastature unose vrijednosti argumenata x i n, a koji potom ispisuje vrijednost funkcije ili odgovarajuću poruku o greški (na osnovu bačenog izuzetka). Pri testiranju obavezno pronaći takve vrijednosti argumenata za koje će se desiti svaki od tri moguća izuzetka (nađene vrijednosti pribilježiti u svesku).

4.31 Napišite funkciju koja kao parametar prima cijeli broj n, a koja kreira i vraća kao rezultat sve cijele brojeve iz opsega od 1 do n uključivo koji su djeljivi sa sumom svojih cifara. Napisanu funkciju demonstrirajte u testnom programu u kojem se sa tastature traži cijeli broj n, a koji zatim generira kreira traženi vektor pozivom napisane funkcije i, na kraju, ispisuje elemente tako kreiranog vektora na ekran, međusobno razdvojene zarezima (iza posljednjeg elementa ne treba ispisivati zarez).

4.32 Napišite funkciju "`IzbrisiPodstring`" sa dva parametra tipa "`string`". Funkcija treba da kreira novi string u koji će prepisati sadržaj prvog stringa iz kojeg su izbačena sva eventualna pojavljivanja stringa koji je zadan drugim parametrom kao njegovog podstringa (ukoliko takvih pojavljivanja nema, string se prepisuje neizmijenjen), i da vrati tako kreirani string kao rezultat. Na primjer, nakon poziva funkcije

```
s = IzbrisiPodstring("abcxyzslkgxyzalj", "xyz");
```

u stringu "`s`" (uz pretpostavku da je propisno deklariran) trebao bi se nalaziti niz znakova "`abcslkgalj`". Funkciju testirajte na primjerima stringova koji se unose sa tastature (program treba da traži unos glavnog stringa i podstringa koji iz njega treba ukloniti).

4.33 Na predavanjima je obrađeno nekoliko korisnih funkcija iz biblioteke "algorithm". Među njima su i funkcije "max element", "count" i "find if" i "replace copy if". Napišite vlastite verzije ovih funkcije nazvane "Najveci", "Prebroji", "NadjiUvjetno" i "KopirajUzZamjenuUvjetno", koje rade potpuno istu stvar kao i odgovarajuće bibliotečke funkcije. Funkcije treba realizirati isključivo korištenjem pokazivačke aritmetike. Ove funkcije testirajte u glavnom programu koji će za vektor brojeva unesenih sa tastature uraditi sljedeće:

- Ispisati koliko puta se u vektoru pojavljuje najveći element niza;
- Ispisati koji je prvi element u vektoru koji je stepen trojke (tj. broj poput 1, 3, 9, 27, 81 itd.) i na kojoj se poziciji nalazi (ukoliko takvog elementa nema, treba ispisati odgovarajuću poruku koja govori o tome);
- Iskopirati sve elemente vektora u drugi vektor uz zamjenu onih elemenata koji su potpuni kvadrati (tj. brojevi poput 1, 4, 9, 16, 25, itd.) nulama, nakon čega treba ispisati sve elemente tako formiranog vektora.

U glavnom programu nije dozvoljeno koristiti petlje (osim za unos i ispis elemenata niza), već sve manipulacije treba ostvariti isključivo pozivima napisanih funkcija:

Napomena: Kao test da li Vam napisane funkcije rade ispravno, Vaš glavni program treba identično raditi ukoliko pozive funkcija "Najveci", "Prebroji", "NadjiUvjetno" i "KopirajUzZamjenuUvjetno" zamijenite pozivima odgovarajućih bibliotečkih funkcij

4.34 Napišite program koji će Vas uvjeriti koliko je bolje koristiti bibliotečku funkciju "sort" od ručnog sortiranja. U programu ćete prvo definirati tri funkcije "GenerirajVektor", "SortirajRucno" i "SortirajEfikasno". Funkcija "GenerirajVektor" treba da ima jedan cjelobrojni parametar "n" i ona vraća kao rezultat vektor od "n" elemenata popunjen slučajnim cijelim brojevima (za generiranje slučajnih brojeva koristite funkciju "`rand`" bez parametara iz biblioteke "`cstdlib`" koja kao rezultat vraća slučajan cijeli broj u opsegu od 0 do neke velike vrijednosti nazvane "`RAND_MAX`" a koja je ovisna od

implementacije). Funkcija "`SortirajRucno`" ima jedan parametar "`v`" koji je vektor cijelih brojeva, a koja vrši sortiranje vektora "`v`" u rastući poredak koristeći neki od jednostavnih postupaka za sortiranje koji su Vam poznati (npr. BubbleSort, SelectionSort, itd.). Funkcija "`SortirajEfikasno`" obavlja isti zadatak, ali koristeći poziv funkcije "`sort`" iz biblioteke "`algorithm`". Napisane funkcije ćete iskoristiti u testnom programu koji prvo definira dvije cjelobrojne konstante "`n1`" i "`n2`" (čije ćete tačne vrijednosti odrediti kasnije) a koji zatim kreira dva vektora "`v1`" i "`v2`" sa respektivno "`n1`" i "`n2`" elemenata. Nakon toga, prvi vektor treba sortirati pozivom funkcije "`SortirajRucno`", a drugi pozivom funkcije "`SortirajEfikasno`". Testiranje započnite od malih vrijednosti "`n1`" i "`n2`", a zatim povećavajte ove vrijednosti sve dok trajanje svakog od sortiranja bude trajalo približno 10 sekundi. Uporedite ove vrijednosti i sami izvucite zaključak.

Napomena: Kada budete testirali program za male vrijednosti "`n1`" i "`n2`", ispišite elemente vektora nakon sortiranja da se uvjerite da sortiranje zaista radi kako treba. Kasnije, kada budete povećavali vrijednosti "`n1`" i "`n2`" izbacite ispisivanje, jer će ove vrijednosti na kraju biti zaista velike. Recimo, na računaru na kojem radi predmetni nastavnik, ove vrijednosti iznose 10500 i 13800000 respektivno.

4.35 Napišite funkciju koja prima vektor od n realnih brojeva a1, a2, ... an kao parametar, i koja računa i vraća kao rezultat vrijednost izraza

$$\sqrt{a_1 + \sqrt{a_2 + \sqrt{... + \sqrt{a_n}}}}$$

Vodite računa da u jeziku C++ indeksi vektora idu od 0, a ne od 1. U slučaju da su elementi takvi da rezultat nije realan broj, funkcija treba baciti izuzetak. Napišite i kratki testni program u kojem ćete demonstrirati napisanu funkciju na sekvenci od n brojeva koji se unose sa tastature (prethodno se n također unosi sa tastature). Obavezno predvidite i hvatanje eventualno bačenih izuzetaka iz funkcije.

4.36 Napišite funkciju sa jednim parametrom n koji je prirodan broj, a koja kreira i vraća kao rezultat kvadratnu matricu formata n × n organiziranu kao vektor vektôrā. Elementi ove matrice su slijed prirodnih brojeva 1, 2, 3, ... n2 razmješteni u redoslijedu po spirali, u smjeru kazaljke na satu, počev od gornjeg lijevog ugla. Na primjer, za n = 5, matrica treba da ima sljedeći oblik:

```
1  2  3  4  5
16 17 18 19 6
15 24 25 20 7
14 23 22 21 8
13 12 11 10 9
```

Napisanu funkciju demonstrirajte u testnom programu koji za uneseni broj n sa tastature generira i ispisuje elemente ovako formirane matrice.

4.37 Napišite funkciju koja ima kao parametar neki cijeli broj n, a koja vraća kao rezultat string koji predstavlja zapis tog istog broja u binarnom brojnom sistemu. Na primjer, ukoliko se kao parametar zada broj –137419, funkcija treba da kao rezultat vrati string "–10001100011001011", jer binarni zapis broja –137419 glasi –10001100011001011. Napisanu funkciju demonstrirajte u testnom programu koji za broj unesen sa tastature računa i ispisuje njegov binarni ekvivalent.

4.38 Deklarirajte pobrojane tipove “Dani” i “Mjeseci”. Promjenljive tipa “Dani” mogu uzimati samo vrijednosti “Ponedjeljak”, “Utorak”, “Srijeda”, “Cetvrtak”, “Petak”, “Subota” i “Nedjelja”, dok promjenljive tipa “Mjeseci” mogu uzimati samo vrijednosti “Januar”, “Februar”, “Mart”, “April”, “Maj”, “Juni”, “Juli”, “August”, “Septembar”, “Oktobar”, “Novembar” i “Decembar”. Zatim napišite funkciju “StampajKalendar” koja ne vraća nikakav rezultat (tj. čiji je povratni tip “void”), a koja štampa kalendar za zadani mjesec. Ova funkcija zahtijeva tri parametra. Prvi parametar “mjesec” je tipa “Mjeseci”, drugi parametar “pocetni dan” je tipa “Dani”, dok je treći parametar “prestupna” tipa “bool”. Parametar “mjesec” određuje mjesec (informacija o mjesecu će se koristiti za određivanje broja dana u mjesecu), parametar “pocetni dan” određuje dan u sedmici kojim započinje taj mjesec, dok parametar “prestupna” određuje da li je godina prestupna ili ne (vrijednost ovog parametra bitna je samo ukoliko parametar “mjesec” ima vrijednost “Februar”). Pored toga, parametar “prestupna” treba da ima podrazumijevanu vrijednost “false”, tako da se u većini slučajeva ne mora navoditi. Na primjer, ukoliko želimo odštampati kalendar za februar koji počinje srijedom u prestupnoj godini, to možemo uraditi sljedećim pozivom:

```
StampajKalendar(Februar, Srijeda, true);
```

Taj poziv treba da na ekranu proizvede ispis poput sljedećeg:

```
P  U  S  Č  P  S  N
      1  2  3  4  5
6  7  8  9  10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29
```

Slično, da odštampamo kalendar za oktobar koji počinje subotom, možemo koristiti poziv poput sljedećeg:

```
StampajKalendar(Oktobar, Subota);
```

Napisanu funkciju iskoristite u glavnom programu (funkciji “main”) koja ispisuje kalendar za čitavu 2009. godinu. Ne morate koristiti nikakvu posebnu logiku za određivanje početnih dana pojedinih mjeseci u 2009. godini, već te informacije prosto prepišite iz kalendara i ugradite u program.

4.39 Napišite generičku funkciju “Izdvajanje” sa 4 parametara: “A”, “N”, “Prvi” i “Drugi”. “A” je niz elemenata za koje se pretpostavlja da se mogu međusobno porediti, i za koji ćemo pretpostaviti da sadrži barem 2 različita elementa. “N” je broj elemenata u tom nizu. Funkcija treba da pronađe prva 2 elementa po veličini, i da smjesti pronađene vrijednosti u parametre “Prvi” i “Drugi”. Na primjer, neka je dat slijedeći niz

```
int Niz[10] = {6, 7, 4, 1, 6, 5, 3, 8, 5, 9};
```

Tada poziv funkcije

```
Izdvajanje(Niz, 10, P, Q);
```

treba da u promjenljive “P” i “Q” (naravno, uz pretpostavku da su propisno deklarisane) smjesti redom 1 i 3, jer su dvije najmanje vrijednosti po veličini u ovom nizu upravo brojevi 1

i 3. Napišite i kratki testni program ("main" funkciju) u kojem ćete demonstrirati napisanu funkciju.

4.40 Napišite generičku funkciju "Analiza" sa 4 parametra "A", "N", "Bmin" i "Bmax". Prvi parametar je niz elemenata za koje se pretpostavlja da se mogu međusobno porediti, a drugi parametar je broj elemenata u tom nizu. Funkcija treba da pretraži zadani niz, i da smjesti u parametre "Bmin" i "Bmax" broj elemenata niza koji su jednaki najmanjem, odnosno najvećem elementu niza. Na primjer, neka je dat slijedeći niz:

```
double Niz[10] = {2.5, 6, 1.25, 6, 6, -2.19, -1, 0, 3, 1};
```

Nakon poziva funkcije

```
Analiza(Niz, 10, N1, N2);
```

promjenljive "N1" i "N2" (pod uvjetom da su propisno deklarisane) treba da prime vrijednosti 1 i 3, jer 1 element je jednak najmanjoj vrijednosti u nizu (koja iznosi –2.19), a 3 elementa sujednaka najvećoj vrijednosti u nizu (koja iznosi 6).

4.41 Napišite generičku funkciju "UkloniNadprosjecne" koja ima jedan parametar nazvan "v". Ovaj parametar predstavlja vektor elemenata koji pripadaju konceptu uporedivih elemenata (tj. elemenata za koje su definirane operacije poređenja, tako da se mogu porediti po veličini). Funkcija treba da odstrani sve elemente iz zadanog vektora čija je vrijednost veća od aritmetičke sredine svih brojeva. Na primjer, neka vektor "a" realnih brojeva sadrži redom vrijednosti 7.43, 13.5, 7.11, 2.3, 10.3, –5.12, 6.12, 4.39, 5.01, 6.3 i 0. Nakon poziva funkcije

```
UkloniNadprosjecne(a);
```

elementi vektora "a" treba da postanu 2.3, –5.12, 4.39, 5.01 i 0, s obzirom da je aritmetička sredina svih elemenata približno 5.213. Funkcija "UkloniNadprosjecne" ne treba da koristi nikakav pomoćni vektor osim samog parametra "v". Također, za realizaciju ove funkcije ne smijete koristiti nikakve koncepte koji nisu obrađivani na predavanjima (npr. funkciju "erase", iteratore, itd.). Napišite i kratki testni program u kojem ćete demonstrirati napisanu funkciju nasekvenci realnih brojeva koji se unose sa tastature.

4.42 Za neki element an nekog niza brojeva kaže se da je lokalni maksimum tog niza brojeva ukoliko je veći od oba svoja susjeda, tj. ukoliko je $a_n > a_{n-1}$ i $a_n > a_{n+1}$. Napišite generičku funkciju koja prima dva parametra, od kojih je prvi parametar niz elemenata proizvoljnog tipa, dok je drugi parametar broj elemenata u tom nizu. Funkcija treba da kao rezultat vrati broj lokalnih maksimuma u tom nizu. Napišite i mali testni program u kojem ćete demonstrirati kako se napisana funkcija može primijeniti na jednom fiksno zadanom nizu od 10 cijelih brojeva.

4.43 Napišite generičku funkciju koja prima dva parametra, od kojih je prvi parametar niz elemenata proizvoljnog tipa, dok je drugi parametar broj elemenata u tom nizu. Funkcija treba da ispita da li je niz periodičan ili ne. Na primjer, niz čiji su elementi 5, 9, 7, 2, 5, 9, 7, 2, 5, 9, 7 je periodičan, sa dužinom perioda 4. Ukoliko niz jeste periodičan, funkcija treba da vrati kao rezultat dužinu njegovog perioda, a ukoliko nije

periodičan, funkcija treba da vrati nulu. Napišite i mali testni program u kojem ćete demonstrirati kako se napisana funkcija može primijeniti na nizu cijelih brojeva koji se unose sa tastature.

4.44 Napišite generičku funkciju “DaLiJeRastuci” koja prima dva parametra nazvana “niz” i “n”. “niz” je niz elemenata proizvoljnog tipa, za koje se pretpostavlja da se mogu međusobno porediti, dok je “n” broj elemenata u tom nizu. Funkcija treba da vrati logičku vrijednost “true” kao rezultat ukoliko je niz monotono rastući, a logičku vrijednost “false” u suprotnom. Napisanu funkciju demonstrirajte u kratkom testnom programu u kojem ćete unijeti niz realnih brojeva sa tastature (broj elemenata se također zadaje sa tastature, i neće biti veći od 100), a koji će zatim ispisati da li je niz monotono rastuci ili nije.

4.45 Napišite generičku funkciju “DaLiJeSimetrican” koja kao parametar prima vektor elemenata prouzvoljnog tipa. Funkcija treba da vrati logičku vrijednost “true” kao rezultat ukoliko je vektor simetričan, tj. ukoliko mu se elementi isto čitaju sa obe strane. U suprotnom, funkcija treba da vrati logičku vrijednost “false” kao rezultat. Na primjer, vektor čiji elementi redom glase 3, 5, 8, 4, 5, 8, 6, 8, 5, 4, 8, 5, i 3 predstavlja primjer simetričnog vektora. Napisanu funkciju demonstrirajte u kratkom testnom programu u kojem ćete unijeti slijed brojeva sa tastature (broj elemenata se također zadaje sa tastature), a koji će zatim ispisati da li je uneseni slijed brojeva simetričan ili nije.

4.46 Napišite generičku funkciju koja kao parametar prihvata vektor elemenata proizvoljnog tipa, a koja kao rezultat vraća broj koji se najviše puta pojavio unutar tog niza. Na primjer, ukoliko se u vektoru nalaze brojevi 3, 8, 5, 4, 2, 3, 8, 5, 1, 8, 2, 7, 8 i 2, funkcija treba da kao rezultat vrati broj 8, jer se taj broj javlja najviše puta u nizu. Ukoliko postoji više brojeva sa najvećim brojem pojava, funkcija treba da vrati kao rezultat prvi od njih. Testirajte funkciju na nizu brojeva koji se unose sa tastature.

4.47 Napišite generičku funkciju sa četiri parametra. Prva dva parametra su pokazivači koji omeđuju slijed elemenata (tj. prvi pokazivač pokazuje na prvi element slijeda, dok drugi pokazivač pokazuje iza posljednjeg elementa slijeda), za koje se pretpostavlja da se mogu međusobno porediti. Funkcija treba da pronađe najmanji i najveći element u slijedu, i da ih smjesti respektivno u treći i četvrti parametar funkcije. Napišite i mali testni program u kojem ćete demonstrirati kako se upotrebljava napisana funkcija na nizu realnih brojeva.

4.48 Napišite generičku funkciju sa dva parametra, koji su pokazivači koji omeđuju slijed cjelobrojnih elemenata (tj. prvi pokazivač pokazuje na prvi element slijeda, dok drugi pokazivač pokazuje iza posljednjeg elementa slijeda). Funkcija treba da kao rezultat vrati broj trocifrenih brojeva u zadanom slijedu. Napišite i mali testni program u kojem ćete demonstrirati kako se upotrebljava napisana funkcija na nizu cijelih brojeva.

4.49 Napišite generičku funkciju sa dva parametra, koji su pokazivači koji omeđuju slijed cjelobrojnih elemenata (tj. prvi pokazivač pokazuje na prvi element slijeda, dok drugi pokazivač pokazuje iza posljednjeg elementa slijeda). Funkcija treba da kao rezultat vrati broj parnih brojeva u zadanom slijedu. Napišite i mali testni program u kojem ćete demonstrirati kako se upotrebljava napisana funkcija na nizu cijelih brojeva.

4.50

a) Napišite funkciju koja kao parametar prima jedan prirodan broj, a kao rezultat vraća vektor čiji su elementi svi djelioci broja zadanog kao parametar.
b) Napišite funkciju koja prima kao parametar vektor cijelih brojeva, a koja zatim transformira elemente primljenog vektora po zakonu vi = (vi 2 + 1) mod 13, gdje vi predstavlja i-ti element vektora, dok "mod" označava operaciju "ostatak pri dijeljenju sa". Funkcija ne vraća nikakav rezultat, već samo modificira vektor koji joj je prenesen kao parametar.
c) Napišite funkciju koja kao prvi parametar prima vektor cijelih brojeva, a koja u drugi i treći parametar redom smješta indeks (redni broj) najmanjeg i najvećeg elementa vektora. Za realizaciju ove funkcije nije dozvoljeno koristiti funkcije iz biblioteke "algorithm", niti je dozvoljeno sortirati niz.
d) Napišite funkciju koja kao parametar prima vektor cijelih brojeva, a koja vraća logičku vrijednost "tačno" ili "netačno", u ovisnosti da li u vektoru ima potpunih kvadrata (tj. brojeva koji se mogu napisati kao kvadrati nekog drugog prirodnog broja) ili ne.
e) Napišite generičku funkciju koja kao parametre prima dva pokazivača na proizvoljni ali isti tip, a koja ispisuje sve elemente bloka koji je omeđen sa ta dva pokazivača. Pri tome se pretpostavlja da prvi pokazivač pokazuje na prvi element bloka, a drugi pokazivač tačno iza posljednjeg elementa bloka.
f) Napišite glavni program koji prvo poziva funkciju pod a) da generira vektor svih djelilaca broja unesenog sa tastature, a zatim ga transformira pozivom funkcije pod b). Nakon toga, program treba uz pomoć poziva funkcije pod c) da odredi poziciju najmanjeg i najvećeg elementa u vektoru, i da na njihovo mjesto upiše nulu. Konačno, program treba da uz pomoć poziva funkcije pod e) ispiše sve elemente vektora nakon obavljenih manipulacija.

4.51

a) Napišite funkciju koja kao parametar prima jedan prirodan broj, a kao rezultat vraća vektor čiji su elementi svi prirodni brojevi koji su potpuni kvadrati (tj. koji se mogu napisati kao kvadrat nekog drugog prirodnog broja), a koji su manji od broja zadanog kao parametar.
b) Napišite funkciju koja prima kao parametar vektor cijelih brojeva, a koja zatim transformira elemente primljenog vektora po zakonu vi = (4 vi + 7) mod 19, gdje vi predstavlja i-ti element vektora, dok "mod" označava operaciju "ostatak pri dijeljenju sa". Funkcija ne vraća nikakav rezultat, već samo modificira vektor koji joj je prenesen kao parametar.
c) Napišite funkciju koja kao prvi parametar prima vektor cijelih brojeva, a koja u drugi i treći parametar redom smješta najmanji i najveći element vektora. Za realizaciju ove funkcije nije dozvoljeno koristiti funkcije iz biblioteke "algorithm", niti je dozvoljeno sortirati niz.
d) Napišite funkciju koja kao parametar prima vektor cijelih brojeva, a koja vraća logičku vrijednost "tačno" ili "netačno", u ovisnosti da li u vektoru ima parnih brojeva ili ne.
e) Napišite generičku funkciju koja kao parametre prima dva pokazivača na proizvoljni ali isti tip, a koja ispisuje sve elemente bloka koji je omeđen sa ta dva pokazivača. Pri tome se pretpostavlja da prvi pokazivač pokazuje na prvi element bloka, a drugi pokazivač tačno iza posljednjeg elementa bloka.
f) Napišite glavni program koji prvo poziva funkciju pod a) da generira vektor svih potpunih kvadrata manjih od broja unesenog sa tastature, a zatim ga transformira pozivom funkcije pod b). Nakon toga, program treba uz pomoć poziva funkcije pod c)

da odredi i ispiše najmanji i najveći element u vektoru. Konačno, program treba da uz pomoć poziva funkcije pod e)ispiše sve elemente vektora nakon obavljenih manipulacija.

4.52

a) Napišite funkciju koja kao parametar prima jedan prirodan broj, a kao rezultat vraća vektor čiji su elementi svi stepeni broja 2, a koji su manji od broja zadanog kao parametar.
b) Napišite funkciju koja prima kao parametar vektor cijelih brojeva, a koja zatim transformira elemente primljenog vektora po zakonu vi = (3 vi 2 + 2) mod 17, gdje vi predstavlja i-ti element vektora, dok "mod" označava operaciju "ostatak pri dijeljenju sa". Funkcija ne vraća nikakav rezultat, već samo modificira vektor koji joj je prenesen kao parametar.
c) Napišite funkciju koja kao prvi parametar prima vektor cijelih brojeva, a koja u drugi i treći parametar redom smješta najmanji i najveći element vektora. Za realizaciju ove funkcije nije dozvoljeno koristiti funkcije iz biblioteke "algorithm", niti je dozvoljeno sortirati niz.
d) Napišite funkciju koja kao parametar prima vektor cijelih brojeva, a koja vraća logičku vrijednost "tačno" ili "netačno", u ovisnosti da li u vektoru ima trocifrenih brojeva ili ne.
e) Napišite generičku funkciju koja kao parametre prima dva pokazivača na proizvoljni ali isti tip, a koja ispisuje sve elemente bloka koji je omeđen sa ta dva pokazivača. Pri tome se pretpostavlja da prvi pokazivač pokazuje na prvi element bloka, a drugi pokazivač tačno iza posljednjeg elementa bloka.
f) Napišite glavni program koji prvo poziva funkciju pod a) da generira vektor svih potpunih kvadrata manjih od broja unesenog sa tastature, a zatim ga transformira pozivom funkcije pod b). Nakon toga, program treba uz pomoć poziva funkcije pod c) da odredi najmanji i najveći element u vektoru, i da ispiše njihovu razliku. Konačno, program treba da uz pomoć poziva funkcije pod e) ispiše sve elemente vektora nakon obavljenih manipulacija.

4.53

a) Napišite funkciju koja kao parametar prima jedan prirodan broj, a kao rezultat vraća vektor čiji su elementi svi trocifreni brojevi koji su djeljivi sa brojem koji je zadan kao parametar.
b) Napišite funkciju koja prima kao parametar vektor cijelih brojeva, a koja zatim transformira elemente primljenog vektora po zakonu vi = (6 vi + 5) mod 23, gdje vi predstavlja i-ti element vektora, dok "mod" označava operaciju "ostatak pri dijeljenju sa". Funkcija ne vraća nikakav rezultat, već samo modificira vektor koji joj je prenesen kao parametar.
c) Napišite funkciju koja kao prvi parametar prima vektor cijelih brojeva, a koja u drugi i treći parametar redom smješta indeks (redni broj) najmanjeg i najvećeg elementa vektora. Za realizaciju ove funkcije nije dozvoljeno koristiti funkcije iz biblioteke "algorithm", niti je dozvoljeno sortirati niz.
d) Napišite funkciju koja kao parametar prima vektor cijelih brojeva, a koja vraća logičku vrijednost "tačno" ili "netačno", u ovisnosti da li u vektoru ima brojeva koji su stepeni broja 2 ili ne (ukoliko želite, možete koristiti funkciju "log" iz bibbioteke "cmath", ali to nije neophodno).
e) Napišite generičku funkciju koja kao parametre prima dva pokazivača na proizvoljni ali isti tip, a koja ispisuje sve elemente bloka koji je omeđen sa ta dva pokazivača. Pri

tome se pretpostavlja da prvi pokazivač pokazuje na prvi element bloka, a drugi pokazivač tačno iza posljednjeg elementa bloka.

f) Napišite glavni program koji prvo poziva funkciju pod a) da generira vektor svih djelilaca broja unesenog sa tastature, a zatim ga transformira pozivom funkcije pod b). Nakon toga, program treba uz pomoć poziva funkcije pod c) da odredi poziciju najmanjeg i najvećeg elementa u vektoru, i da ispiše te elemente na ekran (na osnovu određene pozicije). Konačno, program treba da uz pomoć poziva funkcije pod e) ispiše sve elemente vektora nakon obavljenih manipulacija.

4.54 Napišite generičku funkciju sa 3 parametra x, f i n. x je vrijednost nekog nedefiniranog tipa T, f je funkcija koja prima parametar tipa T i vraća rezultat tipa T, dok je n cijeli broj. Funkcija treba da kao rezultat vrati vrijednost f (f (f (...(f(x))...)) gdje se funkcija f uzastopno primjenjuje n puta, odnosno vrijednost koja se dobije kada se na argument x funkcija f primijeni n puta. Napišite i mali testni program u kojem ćete demonstrirati napisanu funkciju prosljeđujući joj kao parametar neku funkciju napisanu po volji.

4.55 Poznato je da se izvod neke funkcije f u tački x može približno izračunati pomoću formule

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

ukoliko je h dovoljno mala vrijednost (tačna vrijednost izvoda dobija se za $h \to 0$). Napišite funkciju "Izvod" sa 3 parametra "f", "x" i "h" koja računa približnu vrijednost izvoda funkcije "f" u tački "x". Pri tome je parametar "f" funkcija koja prima realni argument i daje realan rezultat, dok su "x" i "h" realne vrijednosti. Pri tome, parametar "h" se može izostaviti, i tada se uzima podrazumijevana vrijednost h = 10–5. Napišite i mali testni program u kojem ćete demonstrirati kako se napisana funkcija "Izvod" može upotrijebiti za računanje približne vrijednost izvoda funkcije sin u tački x = 0.

4.56 Napišite generičku funkciju koja u bloku elemenata između pokazivača "p1" i "p2" pronalazi element za koji funkcija "f" daje najmanju moguću vrijednost i kao rezultat vraća pokazivač na taj element (ukoliko takvih elemenata ima više, funkcija treba da vrati pokazivač na posljednji takav element). Parametri funkcije su "p1", "p2" i "f" (takva funkcija ne postoji u biblioteci "algorithm", ali je potpuno "u duhu" funkcija iz ove biblioteke). Funkcija bi trebala biti zasnovana na potpunoj dedukciji tipova, tako da se umjesto pokazivača mogu koristiti i iteratori (ukoliko ne znate izvesti potpunu dedukciju, koristite parcijalnu dedukciju, samo ćete time izgubiti 1 poen). Napišite i mali isječak programa u kojem ćete demonstrirati kako biste ovu funkciju primijenili da iz nekog fiksnog niza od 6 realnih brojeva ispišete onaj element čiji je kosinus najmanji.

Funkcije (prenos poreferenci)

7.1 Napišite funkciju "`Cifre`" koja treba da ima 3 parametra "n", "c min" i "c max". Funkcija treba da pronađe najmanju i najveću cifru u parametru "n" (koji predstavlja neki prirodan broj), i da smjesti pronađene cifre u parametre "c min" i "c max". Na primjer, naredba

```
Cifre(37232645, a, b);
```

treba da u promjenljive "a" i "b" (pod uvjetom da su propisno deklarirane) smjesti brojeve 2 i 7, jer su upravo to najmanja i najveća cifra u broju 37232645. Funkcija ne smije koristiti nikakve pomoćne nizove ili vektore (samo individualne promjenljive). Napišite i kratki testni program ukojem ćete demonstrirati napisanu funkciju.

7.2 Napišite funkciju "IzvrniString" sa jednim parametrom tipa "string". Funkcija treba da ispremješta elemente stringa tako da prvi znak postane posljednji, a posljednji prvi. Na primjer, ukoliko se u glavnom programu izvrši sekvenca naredbi

```
string s("Ovo je neki tekst...");
Izvrni(a);
cout << s;
```

ispis na ekranu treba da bude "`...tsket iken ej ovO`". Funkcija treba da bude realizirana tako da vrši premještanje "u mjestu", odnosno da ne koristi nikakav pomoćni string osim stringa koji je prenesen kao parametar (uputa: razmijenite prvi znak sa posljednjim, drugi sa pretposljednjim, itd.). Napišite i kratki glavni program u kojoj ćete demonstrirati napisanu funkciju na rečenici koja se unosi sa tastature. Oprez: funkcija izvrni ne smije sama po sebi ništa da ispisuje na ekran. Njeno dejstvo treba samo da bude izmjena poretka znakova u stringu, koje će kasnije na ekran ispisati neko drugi (npr. "main" funkcija).

7.3 Napišite generičku funkciju "UnosBroja" sa tri parametra. Funkcija treba da omogući pouzdano unošenje brojeva u program, uz potpunu kontrolu grešaka pri unosu. Prvi i drugi parametar su tipa "string". Pri tome, prvi parametar predstavlja tekst koji se ispisuje korisniku kao obavijest da treba unijeti broj (prompt), dok drugi parametar predstavlja tekst koji se ispisuje korisniku kao upozorenje u slučaju da unos nije ispravan. Treći parametar je referenca na proizvoljni numerički tip, a predstavlja promjenljivu u koju će se smjestiti uneseni broj. Na primjer, funkcija se može pozvati na sljedeći način:

```
UnosBroja("Unesi prvi broj: ", "Neispravan unos!\n", prvi_broj);
```

Funkcija treba da traži unos od korisnika sve dok unos ne bude ispravan. Napisanu funkciju demonstrirajte u testnom programu koji od korisnika traži da unese realni broj x i cijeli broj n, a zatim računa i ispisuje vrijednost stepena xn.

7.4 Napišite generičku funkciju "Presjek" koja prima dva parametra "v1" i "v2". Ovi parametri su vektori proizvoljnog ali istog tipa elemenata. Funkcija treba da kao rezultat vrati novi vektor koji se sastoji od elemenata koji se javljaju i u vektoru "v1" i u vektoru "v2" (drugim riječima, treba formirati presjek skupova čiji su elementi

pohranjeni u vektorima "v1" i "v2"). Pored toga, u vektoru koji je vraćen kao rezultat iz funkcije svi elementi treba da budu različiti (odnosno, elemente koji se ponavljaju ne treba prepisivati više puta). Poredak brojeva pohranjenih u rezultatu nije bitan. Napisanu funkciju testirajte u testnom programu prvo na dva vektora realnih brojeva, a zatim na dva vektora čiji su elementi stringovi (elementi se unose sa tastature).

7.5 Realizirati program za rješavanje kvadratne jednačine $a\ x^2 + b\ x + c = 0$ ali na modularan način, tako što će se tri cjeline (unos podataka, nalaženje rješenja i ispis rezultata) realizirati kao zasebne funkcije koje će se pozivati iz glavne funkcije. Pri tome program *ne smije da koristi globalne promjenljive*. Umjesto njih, sva komunikacija između funkcija treba da bude ostvarena putem prenosa parametara. Program treba da bude *zaštićen od unosa neispravnih podataka*, odnosno ukoliko program zatraži broj, a korisnik unese nešto drugo, program treba da prijavi grešku i zatraži od korisnika novi unos. Također, zabraniti korisniku da se kao koeficijent *a* unese nula. Za potrebe unosa koeficijenata uz provjeru korektnosti ulaza napraviti posebnu funkciju.

7.6 Napisati dvije preopterećene verzije funkcije "`pretvorba`" koje pretvaraju radijane u stepene. Prva verzija prihvata dva realna parametra "`alfa`" i "`stepeni`", dok druga verzija prihvata četiri parametra nazvana "`alfa`", "`stepeni`", "`minute`" i "`sekunde`", od kojih je prvi realan a ostali cjelobrojni. U oba slučaja prvi parametar predstavlja neki ugao *u radijanima*. Prva verzija funkcije treba da pretvori tu vrijednost u stepene (kao realan broj) i *smjesti* pretvorenu vrijednost u drugi parametar. Druga verzija funkcije treba da pretvori vrijednost ugla u *stepene*, *minute* i *sekunde* i da smjesti pretvorenu vrijednost redom u drugi, treći i četvrti parametar. Na primjer, ukoliko se izvrše naredbe

```
pretvorba(1.326, s);
pretvorba(1.326, x, y, z);
```

u promjenljive "`s`", "`x`", "`y`" i "`z`" (pod pretpostavkom da su ispravno deklarirane) treba da se smjeste brojevi 75.9742, 75, 58 i 27, jer je 1.326 rad = 75.9742° = 75° 58′ 27″. Napisati i kratki glavni program u kojoj ćete demonstrirati napisanu funkciju.

7.7 Napisati funkciju "`Statistika`" sa četiri parametra "`A`", "`N`","`Br1`" i "`Br2`". Prvi parametar je niz realnih brojeva, a drugi parametar predstavlja broj elemenata u nizu. Funkcija treba da u parametre "`Br1`" i "`Br2`" respektivno smjesti broj elemenata niza "`A`" koji su manji odnosno koji su veći od aritmetičke sredine svih elemenata niza. Na primjer, neka je dat slijedeći niz:

```
double Niz[10] = {2.8, 4.3, 7.6, 3.9, 5, 1.8, 6, 3, 7.2, 1.4};
```

Nakon poziva funkcije

```
Statistika (Niz, 10, P, Q);
```

promjenljive "`P`" i "`Q`" (uz pretpostavku da su propisno deklarirane) treba da prime vrijednosti 5 i 4 (Objašnjenje: aritmetička sredina svih elemenata niza je 4.3; elementi niza 2.8, 3.9, 1.8, 3 i 1.4 su manji od 4.3, a elementi niza 7.6, 5, 6 i 7.2 su veći od 4.3). Napisati i kratki testni program u kojem ćete demonstrirati napisanu funkciju na nizu

brojeva koji se unose sa tastature. Obavezno testirati program i na slučaju kada su svi uneseni brojevi jednaki!

7.8 Napisati funkciju "`faktori`" koja ima dva parametra "`a`" i "`n`". Prvi parametar je niz cijelih brojeva, a drugi parametar je cijeli broj. Funkcija treba da napuni niz "`a`" sa prostim faktorima broja "`n`" i da pored toga *vrati kao rezultat* koliko je bilo prostih faktora. Napisati i kratki glavni program u kojem ćete demonstrirati napisanu funkciju.

7.9 Napisati funkciju "`izvrni`" sa dva parametra. Prvi parametar je niz cijelih brojeva, a drugi je broj elemenata u nizu. Funkcija treba da *ispremješta* elemente niza tako da prvi element postane posljednji, a posljednji prvi. Na primjer, ukoliko se u glavnom programu izvrši sekvenca naredbi

```
int a[5] = {3, 5, 6, 1, 2};
izvrni(a, 5);
for(int i = 0; i < 5; i++) cout << a[i] << " ";
```

ispis na ekranu treba da bude "2 1 6 5 3". Funkcija treba da bude realizirana tako da ne koristi *nikakav pomoćni niz* (tj. ne smije se koristiti nikakav drugi niz osim niza prenesenog kao parametar). Napisati i kratki glavni program u kojoj ćete demonstrirati napisanu funkciju na nizu brojeva koji se unose sa tastature. Oprez: funkcija izvrni ne smije sama po sebi ništa da ispisuje na ekran. Njeno dejstvo treba samo da bude *izmjena poretka elemenata u nizu*, koje će kasnije na ekran ispisati neko drugi (npr. "`main`" funkcija).

7.10 Napisati funkciju "`NZD`" sa dva parametra "`Niz`", "`N`". Parametri "`Niz`" je niz prirodnih brojeva, a "`N`" predstavlja broj elemenata u tom nizu. Funkcija treba da kao rezultat vrati najveći zajednički djelitelj brojeva iz niza"`Niz`". Na primjer:

```
int A[5] = {3, 21, 9, 15, 12};
```

Tada sljedeća naredba

```
cout << NZD(A, 5);
```

treba da ispiše broj 3, jer je 3 najveći zajednički djelitelj datih brojeva. Napisati i kratki testni program u kojem ćete demonstrirati napisanu funkciju.

7.11 Napisati generičku funkciju "`Srednja`" sa četiri parametara: "`A`", "`N`", "`Min`" i "`Max`". "`A`" je niz elemenata proizvoljnog tipa, za koji ćemo pretpostaviti da sadrži *barem dva različita elementa*, a "`N`" je broj elemenata u tom nizu. Funkcija treba da pronađe minimalan i maksimalan element u nizu, i da smjesti pronađene elemente u parametre "`Min`" i "`Max`", a funkcija kao rezultat treba da vrati aritmetičku sredinu elemenata niza "`A`" bez minimalnog i maksimalnog elementa . Na primjer, neka je dat sljedeći niz

```
int Niz[10] = {6, 7, 4, 1, 6, 5, 3, 8, 5, 9};
```

Tada poziv funkcije

```
cout << Srednja(Niz, 10, Min, Max);
```

treba da ispiše `5.5` i u promjenljive "`Min`" i "`Max`" (naravno, uz pretpostavku da su propisno deklarisane) smjesti redom vrijednosti 1 i 9, jer su jer su minimalna i

maksimalna vrijednost u ovom nizu upravo brojevi 1 i 9. Napisati i kratki testni program u kojem ćete demonstrirati napisanu funkciju.

7.12 Napisati funkciju "`Eksponent`" sa dva parametra "`N`", "`p` ". Prvi parametar je prirodan broj ,a drugi parametar je neki prost broj. Funkcija treba da *vrati maksimalan* prirodan broj `e` tako da p^e dijeli `N`. Ako broj N nije djeljiv sa p rezultat treba da bude 0. Na primjer,

Sljedeća naredba

```
cout << Eksponent(10, 2);
```

ispisuje broj 1, jer je 1 maksimalan stepen dvojke sa kojim je djeljiv broj 10. Napisati i kratki testni program u kojem ćete demonstrirati napisanu funkciju tako što ćete za dati prirodan broj n odrediti njegovu faktorizaciju na proste brojeve.

7.13 Godine 1742. C. Goldbach je, eksperimentišući sa brojevima, naslutio da se svaki parni broj veći od 2 može napisati kao zbir dva prosta broja i to često na više različitih načina (recimo, imamo 24 = 5 + 19 = 7 + 17 = 11 + 13). Mada je Goldbach ovu pretpostavku provjerio na velikom broju primjera, on nije uspio dokazati da ona zaista uvijek vrijedi, te je uputio pismo L. Euleru sa molbom da proba dokazati ili opovrgnuti tu hipotezu. Euleru to nije pošlo za rukom, ali interesantno je da do današnjeg dana (dakle, skoro 300 godina) niko nije uspio u tome (bez obzira na brojne pokušaje), tako da je ova pretpostavka, poznata kao Goldbachova hipoteza, do danas ostala kao jedna od najpoznatijih nedokazanih hipoteza u teoriji brojeva. Ova hipoteza je do danas testirana grubom silom za sve brojeve manje od $4 \cdot 1014$ i pritom nije ni jednom zakazala, tako da se vjeruje da je ona tačna (mada do danas nije pružen dokaz da ona zaista vrijedi za apsolutno sve parne brojeve veće od 2).

Vaš zadatak je da napravite funkciju "Goldbach" koja ima tri parametra "n", "p" i "q". Funkcija za zadani prirodni broj n, koji predstavlja prvi parametar, treba da nađe dva prosta broja p i q takva da je n = p + q (ukoliko oni postoje), i da nađene vrijednosti p i q smjesti redom u parametre "p" i "q". U slučaju da se takva rastava ne pronađe, funkcija treba baciti izuzetak (ovo će se jedino desiti za $n \leq 2$ ili za neparno n, s obzirom da je Goldbachova hipoteza sigurno tačna za sve parne brojeve veće od 2 koji mogu stati u cjelobrojni tip podataka u jeziku C++). U slučaju da postoji više rastava oblika n = p + q treba uzeti takvu rastavu kod koje p ima najmanju a q najveću moguću vrijednost (tako za n = 24 treba uzeti p = 5 i q = 19). Pri tome, funkcija "Goldbach" ne treba da poziva nikakve druge funkcije. Napisanu funkciju testirajte u testnom programu koji za broj unesen sa tastature ispisuje rastavu na dva prosta sabirka ili informaciju da takva rastava ne postoji.

7.14 Napišite generičku funkciju koja ima tri parametra "v", "min" i "max". Parametar "v" je vektor proizvoljnog tipa elemenata za koje se pretpostavlja da se mogu porediti, dok su "min" i "max" istog tipa kao i tip elemenata vektora. Funkcija treba da iz vektora "v" odstrani sve elemente koji nisu u opsegu od "min" do "max" uključivo. Funkcija ne vraća nikakvu vrijednost, već samo utiče na elemente parametra "v" (koji pri tome, naravno, može promijeniti svoju veličinu). Na primjer, ako prije poziva funkcije vektor "v" sadrži redom elemente 3, 8, 5, 6, 1, 4, 9, 7, 2, 2, 6, 4, 9, 1, 4, 8, 3, 6 i 5, i ako parametri "min" i "max" imaju respektivno vrijednosti 3 i 7, nakon poziva funkcije vektor "v" treba da sadrži redom elemente 3, 5, 6, 4, 7, 6, 4, 4, 3, 6 i 5.

Napisanu funkciju demonstrirajte u testnom programu koji će za vektor realnih brojeva čiji se elementi unose sa tastature odstraniti iz vektora sve elemente koji nisu u zadanom opsegu koji se također zadaje putem tastature i ispisati elemente vektora nakon obavljenog odstranjivanja.

7.15 Napišite dvije preklopljene verzije funkcije "Pretvorba" koje pretvaraju radijane u stepene. Prva verzija prihvata dva realna parametra "ugao" i "stepeni", dok druga verzija prihvata četiri parametra nazvana "ugao", "stepeni", "minute" i "sekunde", od kojih je prvi realan a ostali cjelobrojni. U oba slučaja prvi parametar predstavlja neki ugao u radijanima. Prva verzija funkcije treba da pretvori tu vrijednost u stepene (kao realan broj) i smjesti pretvorenu vrijednost u drugi parametar. Druga verzija funkcije treba da pretvori vrijednost ugla u stepene, minute i sekunde i da smjesti pretvorenu vrijednost redom u drugi, treći i četvrti parametar. Na primjer, ukoliko se izvrše naredbe

```
Pretvorba(1.326, s);
Pretvorba(1.326, x, y, z);
```

u promjenljive "`s`", "`x`", "`y`" i "`z`" (pod pretpostavkom da su ispravno deklarirane) treba da se smjeste brojevi 75.9742, 75, 58 i 27, jer je 1.326 rad = 75.9742◦ = 75◦ 58′ 27″. Napišite i kratku glavnu funkciju ("`main`") u kojoj ćete demonstrirati napisanu funkciju.

7.16 Napišite funkciju sa tri parametra, koja broji koliko broj predstavljen prvim parametrom ima parnih a koliko neparnih djelilaca, i smješta rezultate brojanja respektivno u drugi i treći parametar funkcije. Funkciju upotrijebite u testnom programu koji ispisuje rezultate analize za broj unesen sa tastature, nakon čega traži unos novog broja i tako unedogled, sve dok se kao broj za analizu ne unese negativan broj.

7.17 Napišite funkciju sa tri parametra koja računa zbir svih parnih i svih neparnih cifara cijelog broja koji joj je naveden kao prvi parametar, i smješta rezultate u drugi i treći parametar respektivno. Napišite i mali testni program u kojem ćete demonstrirati kako se upotrebljava napisana funkcija.

7.18 Napišite funkciju sa tri parametra koja računa zbir svih cifara koje su veće odnosno koje nisu veće od 5 u cijelom broju koji joj je naveden kao prvi parametar, i smješta rezultate u drugi i treći parametar respektivno. Napišite i mali testni program u kojem ćete demonstrirati kako se upotrebljava napisana funkcija.

7.19 Napišite funkciju "Sredine" koja treba da ima 3 parametra "N", "AS" i "HS". Funkcija treba da izračuna aritmetičku i harmonijsku sredinu svih cifara u parametru "N" koji je prirodan broj, i da smjesti izračunate vrijednosti u parametre "AS" i "HS" respektivno (podsjetimo se da se harmonijska sredina za n brojeva definira kao recipročna vrijednost aritmetičke sredine njihovih recipročnih vrijednosti). Napišite i mali testni program u kojem ćete demonstrirati kako se upotrebljava napisana funkcija.

7.20 Napišite funkciju "UnosDatuma" sa 3 parametra nazvana redom "Dan", "Mjesec" i "Godina". Funkcija treba da zahtijeva od korisnika da unese sa tastature dan, mjesec i godinu kao tri cijela broja i da ih smjesti u odgovarajuće parametre funkcije. U slučaju da uneseni podaci ne predstavljaju ispravan datum u 21. vijeku (bilo da su uneseni dan ili mjesec besmisleni, bilo da godina nije u opsegu od 2000 do 2099), funkcija treba da

kao rezultat vrati logičku vrijednost “false” (vrijednosti smještene u parametre funkcije tada nisu bitne). U suprotnom, tj. ukoliko je datum korektan, funkcija kao rezultat vraća logičku vrijednost “true”. Napisanu funkciju trebate demonstrirati pomoću kratkog testnog programa koji u petlji poziva ovu funkciju, sve dok korisnik ne unese ispravan datum, a koji zatim ispisuje na ekran unesene vrijednosti za dan, mjesec i godinu u obliku DAN.MJESEC.GODINA.

Napomena: Ne zaboravite na prestupne godine, u kojima februar ima 29 dana! Kako ste ograniceni samo na 21. vijek, mozete uzeti da su prestupne godine one i samo one godine čiji je redni broj djeljiv sa 4 (za duzi vremenski period pravila za odredjivanje koje su godine prestupne po gregorijanskom kalendaru su složenije).

7.21 Napišite funkciju sa tri parametra, od kojih je prvi parametar vektor cijelih brojeva. Funkcija treba da prebroji koliko u tom vektoru ima brojeva sa parnim odnosno neparnim brojem cifara, i da rezultat tog brojanja smjesti respektivno u drugi i treći parametar funkcije. U slučaju da je neki element vektora negativan, funkcija treba baciti izuzetak koji saopštava da je funkcija namijenjena za rad samo sa pozitivnim brojevima. Napišite i kratki testni program u kojem ćete testirati napisanu funkciju na skupini brojeva koji se unose sa tastature. Pri tome je potrebno predvidjeti hvatanje eventualno bačenog izuzetka iz funkcije.

7.22 Napišite funkciju koja ima četiri parametra, pri čemu su prva dva parametra niz realnih brojeva i broj elemenata u tom nizu. Funkcija treba da u treći i četvrti parametar funkcije smjesti respektivno broj pozitivnih i broj negativnih elemenata u nizu. Pored toga, funkcija treba da vrati kao rezultat logičku vrijednost “true” ako i samo ako je makar jedan element niza jednak nuli (inače vraća logičku vrijednost “false”). Napišite i mali testni program u kojem ćete demonstrirati kako biste pozvali ovu funkciju i ispisali rezultate koje ona donosi na primjeru nekog niza sa fiksno zadanim elementima.

7.23 Napišite funkciju koja ima četiri parametra, pri čemu su prva dva parametra niz cijelih brojeva ibroj elemenata u tom nizu. Funkcija treba da u treći i četvrti parametar funkcije smjesti respektivno broj parnih i broj neparnih elemenata u nizu. Pored toga, funkcija treba da vrati kao rezultat logičku vrijednost “true” ako i samo ako je makar jedan element niza negativan (inače vraća logičku vrijednost “false”). Napišite i mali testni program u kojem ćete demonstrirati kako biste pozvali ovu funkciju i ispisali rezultate koje ona donosi na primjeru nekog niza sa fiksno zadanim elementima.

7.24 Napišite funkciju koja ima četiri parametra, pri čemu su prva dva parametra niz realnih brojeva i broj elemenata u tom nizu. Funkcija treba da u treći i četvrti parametar funkcije smjesti respektivno broj cjelobrojnih elemenata i broj elemenata koji nisu cijeli u nizu. Pored toga, funkcija treba da vrati kao rezultat logičku vrijednost “true” ako i samo ako je makar jedan element niza negativan broj (inače vraća logičku vrijednost “false”). Napišite i mali testni program u kojem ćete demonstrirati kako biste pozvali ovu funkciju i ispisali rezultate koje ona donosi na primjeru nekog niza sa fiksno zadanim elementima.

7.25 Napišite funkciju koja ima četiri parametra, pri čemu su prva dva parametra niz cijelih brojeva i broj elemenata u tom nizu. Funkcija treba da u treći i četvrti parametar funkcije smjesti respektivno broj jednocifrenih i broj dvocifrenih elemenata u nizu.

Pored toga, funkcija treba da vrati kao rezultat logičku vrijednost "true" ako i samo ako je makar jedan element niza jednak nuli (inače vraća logičku vrijednost "false"). Napišite i mali testni program u kojem ćete demonstrirati kako biste pozvali ovu funkciju i ispisali rezultate koje ona donosi na primjeru nekog niza sa fiksno zadanim elementima.

7.26 Napišite funkciju koja kao parametar prima neki string, a koja transformira taj string u novi string kod kojeg su ASCII šifre svih znakova uvećane za 1 u odnosu na izvorni string (sama funkcija ne vraća nikakav rezultat). Na primjer, ukoliko se funkciji proslijedi string "s" čiji je sadržaj "proba", sadržaj stringa nakon poziva funkcije treba da bude "qspcb". Napišite i mali testni program u kojem ćete demonstrirati napisanu funkciju.

7.27 Napišite funkciju koja kao parametar prima neki string, a koja transformira taj string u novi string kod kojeg su ASCII šifre svih znakova umanjene za 1 u odnosu na izvorni string (sama funkcija ne vraća nikakav rezultat). Na primjer, ukoliko se funkciji proslijedi string "s" čiji je sadržaj "proba", sadržaj stringa nakon poziva funkcije treba da bude "oqna@". Napišite i mali testni program u kojem ćete demonstrirati napisanu funkciju.

7.28 Napišite funkciju "AnalizaStringa" sa četiri parametra. Prvi parametar je neki dinamički string (tj. objekat tipa "string"). Funkcija treba da utvrdi koliko u tom stringu ima znakova koji su velika slova, zatim znakova koji su mala slova, kao i ostalih znakova (tj. znakova koji nisu slova), i da smjesti rezultate analize redom u drugi, treći i četvrti parametar respektivno. Napišite i mali testni program u kojem ćete demonstrirati napisanu funkciju.

7.29 Napišite funkciju "`AnalizaStringa`" sa četiri parametra. Prvi parametar je neki dinamički string (tj. objekat tipa "`string`"). Funkcija treba da utvrdi koliko u tom stringu ima znakova koji su slova (bilo velika bilo mala), zatim znakova koji su cifre, kao i ostalih znakova (tj. znakova koji nisu niti slova niti cifre), i da smjesti rezultate analize redom u drugi, treći i četvrti parametar respektivno. Napišite i mali testni program u kojem ćete demonstrirati kako bi se mogla upotrijebiti napisana funkcija.

7.30 Napišite funkciju "`Sume`" sa 4 parametra. Prvi parametar treba da bude niz cijelih brojeva, a drugi parametar treba da predstavlja broj elemenata u tom nizu. Funkcija treba da izračuna posebno sumu svih parnih i svih neparnih brojeva u nizu, i da smjesti dobivene rezultate redom u treći i četvrti parametar funkcije. Pored toga funkcija treba da vrati kao rezultat razliku između broja parnih i neparnih brojeva u nizu. Na primjer, ukoliko je data deklaracija

```
int a[10] = {3, 6, 4, 7, 2, 8, 9, 1, 2, 0};
```

tada poziv funkcije

```
z = Sume(a, 10, x, y);
```

treba da u promjenljive "`x`", "`y`" i "`z`" (pretpostavljajući da su one propisno deklarisane) smjesti brojeve 22, 20 i 2 (jer je 6 + 4 + 2 + 8 + 2 + 0 = 22, 3 + 7 + 9 + 1 = 20, a razlika između broja parnih i neparnih brojeva je 2). Napišite i kratki glavni program (funkciju "`main`") u kojem ćete upotrebiti ovu funkciju.

7.31 Napišite funkciju “Statistika” sa 4 parametra “A”, “N”, “Br1” i “Br2”. Prvi parametar je niz realnih brojeva, a drugi parametar predstavlja broj elemenata u nizu. Funkcija treba da u parametre “Br1” i “Br2” respektivno smjesti broj elemenata niza A koji su manji odnosno koji su veći od aritmetičke sredine svih elemenata niza. Na primjer, neka je dat slijedeći niz:

```
double Niz[10] = {2.8, 4.3, 7.6, 3.9, 5, 1.8, 6, 3, 7.2, 1.4};
```

Nakon poziva funkcije

```
Statistika(Niz, 10, P, Q);
```

promjenljive “P” i “Q” (uz pretpostavku da su propisno deklarirane) treba da prime vrijednosti 5 i 4 (Objašnjenje: aritmetička sredina svih elemenata niza je 4.3; elementi niza 2.8, 3.9, 1.8, 3 i 1.4 su manji od 4.3, a elementi niza 7.6, 5, 6 i 7.2 su veći od 4.3). Napišite i kratki testni program u kojem ćete demonstrirati napisanu funkciju na nizu brojeva koji se unose sa tastature. Obavezno testirajte program i na slučaju kada su svi uneseni brojevi jednaki!

7.32 Napišite generičku funkciju “BrojZajednickih” sa četiri parametra “niz1”, “n1”, “niz2” i “n2”. Parametri “niz1“ i “niz2“ su klasični nizovi (ne vektori) proizvoljnog ali istog tipa elemenata (tj. tip elemenata u oba niza je isti), a “n1“ i “n2“ predstavljaju brojeve elemenata u tim nizovima respektivno. Funkcija treba da kao rezultat vrati broj elemenata koji se javljaju kao zajednički elementi i u jednom i u drugom nizu. Na primjer, neka su date sljedeće deklaracije:

```
int a[8] = {3, 7, 2, 3, 1, 5, 5, 2};
int b[10] = {4, 6, 7, 8, 1, 3, 1, 6, 4, 7};
```

Tada sljedeća naredba

```
cout << BrojZajednickih(a, 8, b, 10);
```

treba da ispiše broj 3, jer se tri elementa (3, 7 i 1) pojavljuju u oba niza. Napišite i kratki testni program u kojem ćete demonstrirati napisanu funkciju.

Rekurzija

9.1 Napisati sljedeće rekurzivne funkcije:

a) Funkciju "`SumaKubova`" koja računa zbir kubova svih brojeva od 1 do `N` gdje je `N` prirodan broj prosljeđen kao parametar;

b) Funkciju "`SumaCifara`" koja računa zbir cifara prirodnog broja `N` prosljeđenog kao parametar;

c) Funkciju "`Stepen`" sa dva parametra "`X`" i "`N`" od kojih je prvi realni broj a drugi cijeli broj, a koja računa vrijednost stepena X^N (voditi računa da "`N`" može biti i negativan);

d) Funkciju "`NZD`" sa dva cjelobrojna parametra "`A`" i "`B`" koja računa najveći zajednički djelilac brojeva `A` i `B`;

Sve funkcije treba da budu napisane u čisto rekurzivnom stilu, bez ikakve upotrebe petlji. Napisane funkcije testirati u kratkom testnom programu ("`main`" funkciji) u kojem će se svaka od napisanih funkcija pozvati nad nekoliko karakterističnih primjera. Sve četiri funkcije testirati u *istom* testnom programu!

9.2 Napisati rekurzivnu funkciju "`NaopakiIspis`" bez parametara koja će nakon pozivanja tražiti od korisnika da unosi brojeve sa tastature sve dok korisnik unese nulu. Nakon unosa nule, svi uneseni brojevi (osim nule koja je služila kao graničnik) treba da se ispišu na ekran u obrnutom poretku, počev od posljednjeg unesenog broja ka prvom. Funkcija treba da bude napisana čisto rekurzivno, bez upotrebe petlji i bez upotrebe niza za čuvanje unesenih brojeva (oni se trebaju pamtiti u instancama lokalnih promjenljivih koje se stvaraju prilikom rekurzivnih poziva). Napisanu funkciju testirati u kratkom testnom programu koji će samo pozvati napisanu funkciju.

9.3 Prepraviti program za rješavanje problema Hanojskih kula rađen na vježbama tako da se ispred svakog poteza ispisuje redni broj poteza. Drugim riječima, ispis treba da izgleda poput sljedećeg:

1. potez: Prebaci disk sa štapa 1 na štap 3
2. potez: Prebaci disk sa štapa 1 na štap 2
3. potez: Prebaci disk sa štapa 3 na štap 2

itd. Pri tome, program ne smije da koristi niti globalne niti statičke promjenljive. Pored toga, funkcija "`Hanoi`" koja obavlja glavninu zadatka treba da se iz glavnog programa poziva samo sa jednim stvarnim argumentom – brojem kula. Uputa: razmotrite mogućnost da funkcija "`Hanoi`" vraća vrijednost (koja će biti iskorištena samo u nekim slučajevima).

9.4 Napisati funkcije "`FibIter`" i "`FibRek`" i "`FibRekBrza`" koje primaju jedan cjelobrojni parametar "`N`", a koje računaju `N`-ti Fibonačijev broj respektivno iterativnim postupkom, klasičnim rekurzivnim postupkom, i rekurzivnim postupkom uz provedenu tehniku ubrzavanja (memoizaciju). Uporediti brzine izvršavanja sve tri funkcije za `N = 30`, `N = 35` i `N = 40`. Zaključke zapisati u svesku. Također

eksperimentalno provjeriti koliko puta funkcije "`FibRek`" i "`FibRekBrza`" pozovu same sebe u ova tri slučaja i zabilježiti rezultate. (Uputa: Definirajte neku globalnu promjenljivu koja se uvećava za 1 unutar tijela funkcije "`FibRek`" odnosno "`FibRekBrza`". Prije poziva funkcije postavite ovu promjenljivu na nulu, a nakon poziva funkcije ispišite njenu vrijednost).

Višedimenzionalni nizovi

5.1 Napravite funkciju “TablicaMnozenja” koja prima dva cjelobrojna parametra “m” i “n” koja kreira matricu formata m × n organiziranu kao vektor vektôrā sa cjelobrojnim elementima, popunjava je “tablicom množenja” pri čemu množenici u redovima tablice idu od 1 do m a množitelji u kolonama tablice od 1 do n, i na kraju, vraća tako kreiranu matricu kao rezultat. Na primjer, za m = 3 i n = 5 kreirana matrica treba da ima elemente kako je prikazano ispod:

```
1  2  3  4  5
2  4  6  8  10
3  6  9 12 15
```

Napisanu funkciju demonstrirajte u testnom programu u kojem se sa tastature unose brojevi m i n, zatim kreira matrica formata m × n koja predstavlja tablicu množenja kako je opisano gore, i na kraju ispisuje kreiranu matricu uredno poravnatu na način sličan kako je prikazano gore. Za ispis svakog elementa matrice zauzmite po 5 mjesta (pomoću manipulatora “setw”).

5.2 Napišite funkciju “PascalovTrougao” (ili “PascalovTrokut”) koja ima jedan cjelobrojni parametar “n”. Ova funkcija treba kreirati strukturu “grbave matrice” sa “n” redova u kojoj prvi red ima jedan element, drugi red dva elementa, treći red tri elementa, itd. koju nakon kreiranja treba popuniti elementima Pascalovog trougla (trokuta) sa “n” redova. Funkcija treba da kao rezultat vrati upravo tako kreiranu “grbavu matricu”. Funkciju testirajte u programu koji za unesenu vrijednost “n” sa tastature ispisuje Pascalov trougao (trokut) sa “n” redova. Na primjer, za unesenu vrijednost 7 ispis na ekranu treba izgledati poput sljedećeg:

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
```

Pri tome, elemente Pascalovog trougla nije dozvoljeno računati preko binomnih koeficijenata, već treba koristiti činjenicu da je svaki element Pascalovog trouglajednak zbiru elemenata koji se nalaze tačno iznad njega, i njegovog susjeda sa lijeve strane.

5.3 Napišite program koji traži od korisnika da unese spisak riječi (broj riječi se prethodno unosi sa tastature), a zatim ispisuje na ekran prvu i poslednju riječ iz spiska po abecednom poretku, kao i popis svih unesenih riječi, ali bez ispisivanja duplikata (tj. bez ispisivanja riječi koje su se već jednom ispisale). Program realizirajte korištenjem vektora stringova, odnosno vektora čiji su elementi tipa “string”. Pri tome, nije dozvoljeno sortirati sadržaj vektora.

5.4 Pomoću naredbe “**`typedef`**“ definirajte novi tip “`Mat3x3`“ koji predstavlja matricu realnih brojeva formata 3×3 a zatim napišite funkciju “`Determinanta`“ sa jednim parametrom “`A`“ tipa “`Mat3x3`“ koja kao rezultat vraća determinantu ove matrice. Napisanu funkciju testirajte u programu koji će za tri para tačaka (x_1, y_1), (x_2, y_2) i

(x_3, y_3) čije se koordinate unose sa tastature ispituje da li leže na istom pravcu. Podsjetimo se da tri tačke leže na istom pravcu ukoliko vrijedi relacija

$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = 0$$

5.5 Napišite funkciju "`MnoziMatrice`" sa 7 parametara "`mat1`", "`mat2`", "`mat3`", "`m1`", "`N1`", "`m2`" i "`n2`". Prva tri parametra predstavljaju matrice realnih brojeva maksimalnih dimenzija 20×20, dok su "`m1`", "`n1`", "`m2`" i "`n2`" stvarne dimenzije matrica "`mat1`" i "`mat2`". Ukoliko matrice "`mat1`" i "`mat2`" nisu saglasne za množenje, funkcija treba samo da kao rezultat vrati logičku vrijednost "**`false`**". Međutim, ukoliko matrice "`mat1`" i "`mat2`" jesu saglasne za množenje, funkcija treba da izračuna produkt matrica "`mat1`" i "`mat2`" i da rezultat množenja smjesti u matricu "`mat3`". Pored toga, funkcija kao rezultat treba da vrati "**`true`**" (kao indikaciju uspješno obavljenog množenja). Napisanu funkciju testirajte u programu u kojem će korisnik sa tastature unijeti dvije matrice, nakon čega će biti izračunat i ispisan njihov produkt (ukoliko postoji), odnosno informacija da matrice nisu saglasne za množenje (dozvoljeno je koristiti funkcije "`UnesiMatricu`" i "`IspisiMatricu`" rađene na auditornim vježbama).

5.6 Napišite funkcije "`TragMatrice`", "`DaLiJeGornjaTrougaona`" i "`Transpozicija`". Sve tri funkcije su generičke, i primaju dva parametra, "`A`" i "`N`". Parametar "`A`" predstavlja kvadratnu matricu proizvoljnih dimenzija i proizvoljnog tipa elemenata, dok je "`N`" stvarna dimenzija matrice (broj redova i kolona su jednaki, s obzirom da je matrica kvadratna). Ukoliko je broj redova ili kolona matrice zadane kao stvarni parametar eventualno veći od "`N`", funkcija treba da ignorira sve redove i kolone sa indeksima većim od "`N`" (tj. da se ponaša kao da su dimenzije matrice zaista $N \times N$). Funkcija "`TragMatrice`" treba da kao rezultat vrati trag matrice (tj. sumu elemenata na glavnoj dijagonali). Funkcija "`DaLiJeGornjaTrougaona`" treba da kao rezultat vrati logičku vrijednost "**`true`**" ukoliko je matrica gornja trougaona (tj. ako ima sve nule ispod glavne dijagonale), a "**`false`**" ako nije. Konačno, funkcija "`Transpozicija`" treba da izmijeni elemente matrice tako da ona postane transponirana u odnosu na izvornu matricu (tj. treba razmijeniti elemente na pozicijama (i,j) i (j,i)). Napisane funkcije testirajte u glavnom programu na primjeru matrice čije dimenzije i elemente unosi korisnik putem tastature.

5.6 Napisati program koji traži od korisnika da unese niz riječi (broj riječi se prethodno unosi sa tastature), a zatim ispisuje na ekran prvu i poslednju riječ iz spiska po abecednom poretku, kao i popis svih unesenih riječi, ali bez ispisivanja duplikata (tj. bez ispisivanja riječi koje su se već jednom ispisale). Program realizirati korištenjem klasičnih dvodimenzionalnih nizova znakova

5.7 Napišite generičku funkciju koja kao svoj parametar prima dvodimenzionalnu strukturu (recimo matricu, ali pri čemu broj elemenata u svakom redu ne mora nužno biti isti) predstavljenu kao vektor vektōrâ čiji su elementi proizvoljnog tipa. Funkcija prvo treba da dinamički alocira prostor za dvodimenzionalnu strukturu identičnog oblika kao i parametar, zatim da u nju prepiše elemente dvodimenzionalne strukture predstavljene parametrom i, konačno, da kao rezultat vrati dvojni pokazivač preko kojeg se može izvršiti pristup elementima ove strukture. U slučaju da dođe do

problema sa alokacijom memorije, funkcija treba baciti izuzetak. Pri tome, ni u kom slučaju ne smije doći do curenja memorije. Napisanu funkciju testirati u testnom programu koji sa tastature unosi elemente matrice formata 3 × 3 organizirane kao vektor vektōrâ, a nakon toga poziva napisanu funkciju sa ciljem kreiranja odgovarajuće dinamičke matrice i, konačno, ispisuje elemente tako kreirane dinamičke matrice na ekran i oslobađa zauzetu memoriju. U testnom programu predvidjeti i eventualno hvatanje bačenih izuzetaka.

5.8 Neki element matrice naziva se njeno sedlo, ukoliko je on ujedno najveći element u redu u kojem se nalazi i najmanji element u koloni u kojoj se nalazi, ili obrnuto (tj. najmanji element u redu u kojem se nalazi i najveći element koloni u kojoj se nalazi). Pojam sedla matrice od velikog je značaja u teoriji igara. Nemaju sve matrice sedlo, a neke ih mogu imati i više. Napišite funkciju koja kao parametar prima matricu organiziranu kao vektor vektôrā, a koja kao rezultat vraća koliko sedala posjeduje ta matrica (odnosno 0 ukoliko matrica ne posjeduje niti jedno sedlo). Ukoliko preneseni parametar nema strukturu matrice (tj. ukoliko svi redovi proslijeđenog vektora vektôrā nemaju isti broj elemenata), funkcija treba baciti izuzetak. Napisanu funkciju testirajte u glavnom programu na primjeru matrice čije se dimenzije i elementi unose sa tastature.

5.9 Napišite generičku funkciju sa 4 parametra “niz1”, “n1”, “niz2” i “n2”. Parametri “niz1” i “niz2” su nizovi proizvoljnog ali istog tipa elemenata, za koje ćemo pretpostavljati da je definirana operacija sabiranja (što uključuje sve brojčane tipove, ali i recimo tip “string”), dok su “n1” i “n2” brojevi elemenata u tim nizovima respektivno. Funkcija treba da dinamički alocira dvodimenzionalni niz sa “n1” redova i “n2” kolona istog tipa elemenata kao i u nizovima “niz1” i “niz2”, zatim da popuni elemente tako kreiranog dvodimenzionalnog niza tako da element u i–tom redu i j-toj koloni bude jednak zbiru i–tog elementa niza “niz1” i j-tog elementa niza “niz2” i na kraju, da kao rezultat vrati dvojni pokazivač preko kojeg de može pristupiti tako kreiranom dvodimenzionalnom nizu. U slučaju da alokacija ne uspije, funkcija treba baciti izuzetak “Alokacija nije uspjela”, pri čemu ni u kom slučaju ne smije doći do curenja memorije. Napisanu funkciju demonstrirajte u malom testnom programu koji će primijeniti ovu funkciju nad dva niza čiji su elementi fiksno zadani stringovi (po vlastitom izboru) od kojih prvi niz ima 6 a drugi 4 elementa, a koji će zatim ispisati na ekran elemente kreiranog dvodimenzionalnog niza u formi tablice (da ne biste imali problema sa ispisom, neka zadani stringovi budu kratki). Na kraju, program treba prije završetka da oslobodi memoriju zauzetu kreiranim dvodimenzionalnim nizom (bez obzira što će se to svakako desiti automatski po završetku programa).

5.10 Napišite program koji traži da se sa tastature unese prirodan broj n, a nakon toga elementi kvadratne matrice formata n × n (matricu treba organizirati kao vektor vektôrā). Program nakon toga treba da ispiše redni broj kolone sa najvećom sumom elemenata, redni broj reda sa najmanjom sumom elemenata, kao i sumu elemenata na dijagonali.

5.11 U matematici, ako imamo dva vektora ai, i = 1..n i bi, i = 1..m, njihov Kroneckerov (ili tenzorski) produkt je matrica cij, i = 1..n, j = 1..m čiji su elementi dati pravilom cij= ai bj, i = 1..n, j = 1..m. Napišite generičku funkciju koja prima kao parametre dva vektora a i b čiji su elementi proizvoljnog ali istog tipa, i vraća kao rezultat njihov Kroneckerov produkt u vidu vektora vektôrâ. Vodite računa da u jeziku C++ indeksi

vektora idu od 0, a ne od 1! Napišite i mali testni program u kojem ćete demonstrirati kako se upotrebljava napisana funkcija na primjeru vektora čiji su elementi cijeli brojevi.

Klase

10.1 Klasa Razlomak

Jedan od nedostataka jezika opće namjene kao sto je C++ u odnosu na specijalističke matematski orjentirane jezike je nepostojanje podrške za egzaktni rad sa racionalnim brojevima (razlomcima) kakav postoji npr. u programskom paketu Mathematica. Srećom, koncept klasa u jeziku C++ omogućava korisniku da sam definira nove tipove podataka koji zadovoljavaju njegove potrebe. Vaš zadatak je da definirate novi tip podataka (klasu) nazvan "Razlomak" koji omogućava tačan rad sa razlomcima (umjesto njihovog približnog predstavljanja preko decimalnih brojeva). Klasa "Razlomak" treba da sadrzi sljedeće elemente:

a) Privatne atribute koji cuvaju vrijednost brojnika i nazivnika razlomka;

b) Konstruktor sa dva parametra, koji respektivno predstavljaju vrijednost brojnika i nazivnika razlomka. Ovaj konstruktor treba automatski da izvrsi skraćivanjeeventualnih zajedničkih faktora u brojniku i nazivniku, tako da ce deklaracija poput

```
Razlomak r(10, 15);
```

dovesti do toga da ce brojnik razlomka "r" biti 2, a nazivnik 3. Također, konstruktor treba da obezbijedi da u slučaju da je razlomak negativan, negativni znak bude zapamćen u brojniku, a ne u nazivniku. Drugim riječima, nakon deklaracije

```
Razlomak r(2, -3);
```

vrijednost brojnika treba da bude –2, a nazivnika 3. Naravno, nakon deklaracije poput

```
Razlomak r(-2, -3);
```

vrijednosti brojnika i nazivnika treba da budu 2 i 3, jer se negativni znak krati. U slučaju da se za vrijednost nazivnika zada nula, konstruktor treba da baci izuzetak;

c) Konstruktor sa jednim parametrom, koji postavlja vrijednost brojnika na vrijednost zadanu parametrom, a vrijednost nazivnika na jedinicu. Ovaj konstruktor treba da omoguci pretvorbu cjelobrojnih vrijednosti u vrijednosti tipa "Razlomak";

d) Konstruktor bez parametara, koji postavlja vrijednost brojnika na nulu, a vrijednost nazivnika na jedinicu, tj. koji kreira nul-razlomak;

e) Metode kojima je moguće saznati vrijednost brojnika, odnosno nazivnika razlomka;

f) Preklopljene binarne operatore "+", "-", "*" i "/" koji treba da omoguće četiri osnovne operacije sa razlomcima. Rezultati svake od operacija uvijek treba da budu do kraja skraćeni (sto ćete najlakše postići tako što ćete brojnik i nazivnik željenog rezultata prosto proslijediti konstruktoru, koji će obaviti neophodno skraćivanje);

g) Preklopljene unarne operatore "+" i "–". Unarni operator "–" treba da izvrće znak razlomku, a unarni operator "+" ne treba da radi ništa;

h) Preklopljene operatore “+=”,“–=”,“*=” i “/=”, pri čemu izrazi “R1 += R2”, “R1 –= R2”, “R1 *= R2” i “R1 /= R2” treba da respektivno predstavljaju skraćene zapise za izraze “R1 = R1 + R2”, “R1 = R1 – R2”, “R1 = R1 * R2” i “R1 = R1 / R2”;
i) Preklopljene unarne operatore “++” i “—”, pri čemu izrazi “R++” i “R--” efektivno djeluju kao izrazi “R += 1” i “R -= 1”. Pri tome treba podržati kako prefiksnu, tako i postfiksnu verziju ovih operatora, pri čemu postfiksne verzije vraćaju kao rezultat vrijednost razlomka prije obavljene izmjene;
j) Preklopljen operator “<<” koji treba da podrži ispis objekata tipa “Razlomak” na ekran. Ukoliko je p brojnik a q nazivnik razlomka koji se ispisuje, ispis treba da izgleda kao p/q, osim ukoliko je nazivnik jednak jedinici, kada se ispisuje samo vrijednost brojnika (kao cijeli broj);
k) Preklopljen operator “>>” koji treba da podrži unos objekata tipa “Razlomak” sa tastature. Potrebno je omogućiti da se objekti tipa “Razlomak” unose u formi p/q, gdje su p i q cijeli brojevi, ali treba omogućiti da se unese i obični cijeli broj (u tom slučaju se podrazumijeva da je nazivnik jedinica);
l) Preklopljene relacione operatore “<”,“>”,“<=”,“>=”,“==” i “!=” koje ispituju odnose izmedju dva razlomka. Poput svih relacionih operatora, ovi operatori vraćaju kao rezultat “true” u slučaju da je relacija zadovoljena, a “false” ukoliko nije;
m) Preklopljeni operator konverzije tipa “Razlomak” u tip “double” koji omogućava upotrebu promjenljivih tipa “Razlomak” u kontekstima gdje se očekuju realni brojevi, tj. promjenljive tipa “double”. Ovaj operator treba da vrati kao rezultat približnu decimalnu vrijednost koja se dobija dijeljenjem brojnika sa nazivnikom.

Sve metode implementirajte izvan deklaracije klase, osim trivijalnih metoda koje trebate implementirati direktno unutar deklaracije klase. Sve metode koje su inspektori, obavezno deklarirajte kao takve. Dalje je potrebno da napišete testni program koji ce odraditi sljedeće:

a) Demonstrirati rad svih napisanih metoda i operatora nad nekoliko fiksno zadanih razlomaka (po vlastitom izboru);
b) Tražiti od korisnika da unese broj elemenata niza, a zatim dinamički alocirati niz elemenata tipa “Razlomak”;
c) Učitati sa tastature elemente alociranog niza, sortirati ih u rastući poredak, i ispisati sortirani niz razlomaka na ekran i u tekstualnu datoteku “RAZLOMCI.TXT”.

10.2 Klasa Skup

U raznim oblastima matematike često se javlja potreba za radom sa skupovima. Vaš zadatak je da razvijete generičku klasu “Skup” koja predstavlja skup čiji elementi mogu biti proizvoljnog, ali istog tipa (odnosno svi elementi skupa moraju biti istog tipa, tako da skupovi heterogenog sastava nisu podržani), za koje ćemo pretpostaviti da je definiran poredak pomoću operatora “<”.
Klasa treba da ima sljedeće elemente:

a) Privatne atribute koji redom predstavljaju broj elemenata u skupu, kapacitet skupa (tj. maksimalni broj elemenata koji se mogu smjestiti u skup) i pokazivač na dinamički alocirani niz (preciznije, na njegov prvi element) koji će čuvati elemente skupa;
b) Konstruktor sa jednim parametrom, koji predstavlja kapacitet skupa, i koji vrši dinamičku alokaciju prostora za čuvanje elemenata skupa. Inicijalni broj

elemenata kreiranog skupa je 0 (prazan skup). Pri tome je potrebno zabraniti da se ovaj konstruktor koristi za automatsku konverziju cjelobrojnog tipa u tip “Skup”;

c) Destruktor, koji oslobadja memoriju zauzetu konstruktorom;
d) Konstruktor kopije koji ce osigurati da se objekti tipa “Skup” mogu bezbjedno prenositi kao parametri po vrijednosti i vraćati kao rezultati iz funkcija;
e) Preklopljeni operator dodjele, koji garantira sigurno dodjeljivanje jednog skupa drugom skupu, bez problema koji mogu biti uzrokovani kreiranjem plitkih kopija;
f) Metodu sa jednim parametrom, koja dodaje u skup element predstavljen parametrom. U slučaju da se takav element već nalazi u skupu, unos elementa treba ignorirati (ne zaboravimo da skupovi po matematskoj definiciji ne mogu sadržavati jednake elemente). U slucaju da je dostignut kapacitet skupa, metoda treba da baci izuzetak;
g) Metodu bez parametara, koja vraća kao rezultat broj elemenata skupa;
h) Metodu sa jednim parametrom, koja vraća kao rezultat “true” ukoliko se element predstavljen parametrom nalazi u skupu, a “false” ukoliko se ne nalazi;
i) Preklopljen unarni operator “!”, koji primijenjen na skup daje “true” ukoliko je skup prazan, a “false” ukoliko nije;
j) Preklopljen operator “<<” koji treba da podrži ispis skupova na ekran. Skupovi se ispisuju kao slijed svojih elemenata unutar vitičastih zagrada, međusobno razdvojenih zarezima. Pri ispisu, elementi treba da budu sortirani u rastući poredak (redoslijed elemenata je kod skupova svakako nebitan). Za sortiranje možete koristiti funkiju “sort” (bez funkcije kriterija, jer je pretpostavljeno da su elementi skupa takvi da je za njih definiran poredak uz pomoć operatora “<”);
k) Preklopljene binarne operatore “+”, “*” i “-” koji redom nalaze uniju, presjek i razliku skupova koji su navedeni kao operandi. Uputa: Ranije napisane metode za dodavanje novog elementa u skup i ispitivanje da li element pripada skupu mogu Vam biti od velike koristi pri pisanju operatorskih funkcija za ove operatore. Dalje, kapacitet skupa koji ćete vratiti kao rezultat unije “A + B” možete definirati kao zbir broja elemenata skupova “A” i “B”, bez obzira sto će broj elemenata unije “A + B” možda biti manji, ukoliko skupovi “A” i “B” sadrže zajedničke elemente (ionako će prilikom dodjele tipa “C = A + B” rezultat biti kopiran u skup “C”, a pri ispisu tipa “cout << A + B” zbir “A + B” se svakako tretira kao privremeni objekat). Slično, za kapacitet presjeka “A * B” i razlike “A - B” možete uzeti broj elemenata skupa “A”, jer “A * B” i “A – B” svakako ne mogu imati više elemenata nego što ima skup “A”;
l) Prekopljene binarne operatore “+=”,“*=” i “–=” pri cemu izrazi poput “A += B”, “A *= B” i “A –= B” predstavljaju respektivno skraćene zapise za izraze “A = A + B”, “A = A * B” i “A = A – B”;
m) Preklopljene binarne relacione operatore “<=”,“>=”,“==”,“!=”,“<” i “>” pri čemu relacije “A <= B”, “A >= B”, “A == B”, “A != B”, “A < B” i “A > B” predstavljaju respektivno odnose “A je podskup od B”, “A je nadskup od B”, “A i B su jednaki”, “A i B su različiti”, “A je pravi podskup od B”, “A je pravi nadskup od B”. Uputa: Prvo definirajte operator “<=”. Dalje, vrijedi “A >= B” ako je “B <= A”. Vrijedi “A == B” ako je “A >= B” i “A <= B”. Vrijedi “A != B” ako nije “A == B”. Vrijedi “A < B” ako je “A <= B” i “A != B”. Vrijedi “A > B” ako je “A >= B” i “A != B”. Dakle, sve relacione operatore možete definirati polazeći samo od operatora “<=”.

Sve metode implementirajte izvan deklaracije klase, osim trivijalnih metoda koje trebate implementirati direktno unutar deklaracije klase. Obavezno napišite i testni

program u kojem ćete demonstrirati sve elemente napisane generičke klase uzimajući stringove (tj. objekte tipa “string”) za elemente skupova.

10.3 Struktura Vrijeme

Definirajte strukturu “Vrijeme” koja sadrži tri polja “sati”, “minute” i “sekunde”, i koja predstavlja tekuće vrijeme u toku dana, izraženu u satima, minutama i sekundama. Zatim definirajte funkciju “IspisiVrijeme” koja ispisuje vrijeme proslijeđeno kao parametar u funkciju u obliku hh:mm:ss (tj. sati, minute i sekunde se ispisuju kao dvocifreni brojevi, eventualno sa vodećim nulama), kao i funkciju “SaberiVrijeme” koja prima dva vremena kao parametre, i vraća kao rezultat treće vrijeme koje nastaje sabiranjem vremena koja su proslijeđena kao parametri (npr. sabiranjem 3h 34min 52sek i 4h 42min 20sek treba da se dobije vrijeme 8h 17min 12sek). Konačno, napravite mali testni program u kojem ćete testirati napisane funkcije.

10.4 Struktura Čvor

Definirajte strukturu “Cvor” koja sadrži polje “element” tipa “double” i polje “veza” koje je tipa pokazivač na “Cvor”. Iskoristite ovu strukturu u programu koji čita slijed realnih brojeva sa tastature sve dok se ne unese 0, uvezuje te elemente u povezanu listu čvorova, i na kraju prolazi kroz listu čvorova sa ciljem da prebroji i ispiše koliko ima elemenata u listi koji su veći od aritmetičke sredine svih unesenih elemenata (potrebna su zapravo dva prolaza: prvi za računanje aritmetičke sredine, i drugi za brojanje elemenata sa traženim svojstvom).

10.4 Struktura Čvor

Definirajte strukturu “Cvor” koja sadrži polje “lokacija” tipa “string”, zatim polja “cijena” i “kvadratura” tipa “int” i polje “veza” koje je tipa pokazivač na “Cvor”. Iskoristite ovu strukturu u programu koji čita slijed podataka o nekretninama sa tastature, koji se sastoje od lokacije (adrese), cijene i kvadrature za svaku nekretninu, sve dok se kao podatak o lokaciji ne unese prazan string (tj. dok se ne pritisne samo ENTER bez ikakvog prethodnog unosa). Program treba da uveže te podatke u povezanu listu čvorova, ali koji nakon unosa svakog podatka umeće čvor na takvo mjesto da lista u svakom trenutku bude sortirana po cijenama u rastućem poretku kada se lista posmatra u redoslijedu kako idu veze između čvorova (tj. nekretnina sa manjom cijenom treba da dođe ispred nekretnine sa većom cijenom). U slučaju da više nekretnina ima istu cijenu, njih treba međusobno poredati u opadajući poredak po kvadraturi (tj. od dvije nekretnine sa istom cijenom, nekretnina veće kvadrature treba da dođe ispred nekretnine manje kvadrature). Na kraju, program treba da ispiše podatke o svim nekretninama u listi, da se uvjerimo da je lista zaista sortirana u skladu sa traženim specifikacijama.

10.4 Struktura Čvor

Definirajte strukturu “Cvor” koja sadrži polje “ime i prezime” tipa “string”, zatim polje “prosjek” tipa “double” i polje “veza” koje je tipa pokazivač na “Cvor”. Iskoristite ovu strukturu u programu koji čita slijed podataka o studentima sa tastature, koji se sastoje od imena i prezimena i prosjeka za svakog studenta, sve dok se kao podatak o imenu studenta ne unese prazan string (tj. dok se ne pritisne samo ENTER bez ikakvog prethodnog unosa). Program treba da uveže te podatke u povezanu listu čvorova, ali koji nakon unosa svake od rečenica umeće čvor na takvo mjesto da lista u svakom trenutku bude sortirana po prosjeku u opadajućem redoslijedu kada se

posmatra u redoslijedu kako idu veze između čvorova (tj. student sa većim prosjekom treba da dođe ispred studenta sa manjim prosjekom). U slučaju da više studenata ima isti prosjek, njih treba međusobno poredati po abecedi (preciznije, po poretku ASCII kodova). Na kraju, program treba da ispiše podatke o svim studentima u listi, da se uvjerimo da je lista zaista sortirana u skladu sa traženim specifikacijama.

10.4 Struktura Čvor (Josip)

Godine 67. nove ere na području današnje Palestine desio se jedan historijski događaj koji danas služi kao inspiracija za brojne matematičke i programerske zadatke. Naime, te godine, Rimljani su zauzeli jevrejski grad Jotapata, kojim je upravljao Josip Flavije (Josephus Flavius). Josip je, bježeći sa još 40 ljudi, upao u zasjedu. Rimljani su ih pozvali da se predaju, što je Josip htio da prihvati, ali njegovi saborci mu to nisu dozvolili. Josip je tada predložio da se svi međusobno poubijaju, da ne bi živi pali u ruke Rimljanima. Po tradiciji, redoslijed pogubljenja u takvim situacijama se određivao na sljedeći način. Svi ljudi stanu u krug, nakon čega bi svaki treći čovjek po redu bio odstranjen iz kruga i bio ubijen (ili, općenito, svaki m-ti čovjek, pri čemu se broj m slučajno odabirao u svakoj sličnoj situaciji). Posljednji čovjek koji ostane trebao bi da ubije sam sebe. Interesantno je da je posljednji koji je ostao bio upravo Josip, koji se na kraju predao Rimljanima (baš kao što je i želio), umjesto da ubije sam sebe. Do danas je ostalo nepoznato da li je u pitanju čista slučajnost, ili je Josip nekako znao da treba da stane na 31-vo mjesto (ukoliko je znao, postavlja se pitanje kako, s obzirom da do današnjeg dana eksplicitna formula koja daje sigurno mjesto u krugu u funkciji od ukupnog broja ljudi n i razmaka m između ljudi koji se odstranjuju do danas nije poznata).

Vaš zadatak je da napravite program koji će za uneseni broj ljudi n i razmak odstranjivanja m na ekranu ispisati redoslijed odstranjivanja ljudi iz kruga (da ne budemo posve brutalni i kažemo redoslije pogubljenja). Pri tome, program treba biti zasnovan na povezanim listama čvorova, koje se često primjenjuju upravo za rješavanje problema srodnih opisanom problemu. Konkretno, u programu ćete prvo definirati čvornu strukturu “Osoba” koja sadrži polje “redni_broj” tipa “int” i polje “sljedeci” koje je po tipu pokazivač na strukturu tipa “Osoba”. Polje “redni_broj” sadržavaće redni broj osobe u krugu, dok će polje “sljedeci” pokazivati na sljedeću osobu u krugu. Zatim ćete kreirati povezanu listu osoba (tj. čvorova tipa “Osoba”) čiji će redni brojevi biti postavljeni redom na vrijednosti od 1 do n, pri čemu će svaka osoba pokazivati na osobu sa narednim rednim brojem, osim posljednje osobe koja će pokazivati ponovo na prvu osobu, čime se zapravo kreira krug osoba (takve povezane liste u kojima posljednji čvor u listi pokazuje nazad na prvi čvor nazivaju se kružne ili cirkularne liste). Nakon što ste formirali traženu listu, prosto ćete se kretati kroz listu, pri čemu ćete nakon svakih m napravljenih koraka ispisati redni broj osobe na kojoj se trenutno nalazite i odstraniti tu osobu iz liste. Odstranjivanje ćete izvesti tako što ćete pokazivač “sljedeci” osobe koja prethodi osobi na kojoj se trenutno nalazite preusmjeriti tako da ne pokazuje više na osobu na kojoj se trenutno nalazite nego na osobu koja slijedi iza nje (čime je efektivno odstranjujete iz kruga) i preći na sljedeću osobu. Nakon toga, ćete pomoću operatora “delete” izbrisati čvor koji odgovara odstranjenoj osobi, da ne zauzima više memoriju (možete uzeti da ovaj korak odgovara “pogubljenju”). Postupak ćete ponavljati dok se ne eliminira svih n osoba. Za potrebe testiranja, redoslijed odstranjivanja za slučaj n = 41 i m = 3 (kao u opisanom historijskom događaju) glasi:

3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 1, 5, 10, 14, 19, 23, 28, 32, 37, 41, 7, 13, 20, 26, 34, 40, 8, 17, 29, 38, 11, 25, 2, 22, 4, 35, 16, 31

10.5 Struktura Skup

Napisati strukturu "`Skup`" koja predstavlja skup realnih brojeva Struktura treba da ima slijedeće parametre: maksimalnu veličinu skupa, stvarnu veličinu skupa i pokazivač na niz tipa double u kojem se čuvaju elementi skupa. Napomenimo da svi elementi u skupu su različiti tj. ako se dodaje neki broj koji je već u skupu, skup ostaje isti.

Potrebno je implementirati slijedeće funkcije koje rade sa strukturom `Skup`:

```
Skup KreirajSkup(int maksimalnom_velicina);
Skup Unija(Skup s1, Skup s2);
Skup Razlika(Skup s1, Skup s2);
Skup Presjek(Skup s1, Skup s2);
void Ispisi(Skup s) const;
void DodajElement(Skup &s, double element);
```

Funkcija `KreirajSkup` treba da kreira objekat tipa `Skup` sa datom maksimalnom veličinom, da postavi stvarnu veličinu na 0, te da alocira dinamički niz sa datom maksimalnom veličinom, te ako dođe problema koda alokacije memorije da se baci izuzetak. Funkcije `Unija`, `Razliku` i `Presjek` trebaju za data dva skupa da izračunaju njihovu uniju, razliku te presjek dva data skupa, te da odgovarajuće vrijednosti vrate kao rezultat. Funkcija `Ispisi` treba da ispise dati skup u obliku $\{x_1, x_2, \ldots, x_n\}$. Funkcija `DodajElement` treba datom skupu da doda element pod uslovom da je stvarna veličina skupa manja od maksimalne veličine i da stvarnu veličinu poveća za jedan. Ako nema mjesta potrebno je izvršiti realokaciju novog prostora za elemente skupa kreiranjem novog niza koji je dva puta veći od prethodne veličine, te kopiranjem odgovarajućih elemenata iz starog niza u novi niz i brisanjem starog niza. Nakon ovog se u novi niz dodaje element koji se želi umetnuti.

Potrebno je napraviti testni program u kojem će se demonstrirati upotreba datih funkcija za manipulaciju sa objektima tipa `Skup`

10.6 Struktura Čvor

Definirajte strukturu "`Cvor`" koja sadrži polje "`lokacija`" tipa "`string`", zatim polja "`cijena`" i "`kvadratura`" tipa "**`int`**" i polje "`veza`" koje je tipa pokazivač na "`Cvor`". Iskoristite ovu strukturu u programu koji čita slijed podataka o nekretninama sa tastature, koji se sastoje od lokacije (adrese), cijene i kvadrature za svaku nekretninu, sve dok se kao podatak o lokaciji ne unese prazan string (tj. dok se ne pritisne samo ENTER bez ikakvog prethodnog unosa). Program treba da uveže te podatke u povezanu listu čvorova, ali koji nakon unosa svakog podatka umeće čvor na takvo mjesto da lista u svakom trenutku bude sortirana po cijenama u rastućem poretku kada se lista posmatra u redoslijedu kako idu veze između čvorova (tj. nekretnina sa manjom cijenom treba da dođe ispred nekretnine sa većom cijenom). U slučaju da više nekretnina ima istu cijenu, njih treba međusobno poredati u opadajući poredak po kvadraturi (tj. od dvije nekretnine sa istom cijenom, nekretnina veće kvadrature treba da dođe ispred nekretnine manje kvadrature). Na kraju, program treba da ispiše

podatke o svim nekretninama u listi, da se uvjerimo da je lista zaista sortirana u skladu sa traženim specifikacijama.

10.6 Struktura Matrica

Dopunite program "matrica_struct.cpp" priložen uz Predavanje 7. sa dvije nove funkcije "ProduktMatrica" i "StepenMatrica". Funkcija "ProduktMatrica" prima dvije matrice kao parametre (matrice su definirane kao odgovarajuće strukture) i vraća njihov produkt kao rezultat, odnosno baca izuzetak ukoliko matrice nisu saglasne za množenje. Funkcija "StepenMatrica" prima matricu kao jedan parametar, kao drugi parametar prima prirodan broj n, i kao rezultat vraća matricu dignutu na n-ti stepen (u matričnom smislu) odnosno baca izuzetak ukoliko matrica nije kvadratna, s obzirom da je stepen matrice definiran samo za kvadratne matrice (pri realizaciji ove funkcije možete koristiti već napisanu funkciju "ProduktMatrica"). Također, proširite funkciju "IspisiMatricu" dodatnim parametrom "treba_brisati" tipa "bool". Ukoliko ovaj parametar ima vrijednost "true", funkcija treba da oslobodi prostor zauzet matricom koja joj je proslijeđena kao parametar, u suprotnom ne treba da radi ništa. Ovim se omogućava da možemo zadavati pozive poput

```
IspisiMatricu(ZbirMatrica(a, b), 7, true);
```

tako da se oslobađanje memorije koju je zauzela pomoćna matrica koja predstavlja zbir matrica može obaviti bez korištenja pomoćne promjenljive (kao u programu "matrica_struct.cpp"). Definirajte da parametar "treba_brisati" ima podrazumijevanu vrijednost "false", tako da ga ne treba navoditi ukoliko nam brisanje ne treba. Napisane funkcije testirajte u glavnom programu na primjeru matrica čije dimenzije i elemente unosi korisnik putem tastature. U glavnom programu predvidite hvatanje svih izuzetaka koji bi eventualno mogli nastupiti. Također, dobro pazite da nigdje ne dođe do curenja memorije (ni pod kakvim okolnostima). I na kraju, što je od svega važnije, barem ovaj zadatak zaista uradite sami, a nemojte uzeti odnekud njegovo rješenje i prepisati ga (naravno, samostalno biste trebali uraditi i sve ostale zadatke, ali je izuzetno važno da baš ovaj zadatak ne prepišete ni po koju cijenu).

matrica_struct.cpp

```
#include <iostream>
#include <iomanip>

using namespace std;

template <typename TipElemenata>
struct Matrica {
  int broj_redova, broj_kolona;
  TipElemenata **elementi;
};

template <typename TipElemenata>
void UnistiMatricu(Matrica<TipElemenata> mat) {
  if(mat.elementi == 0) return;
  for(int i = 0; i < mat.broj_redova; i++) delete[]
     mat.elementi[i];
```

```
  delete[] mat.elementi;
}

template <typename TipElemenata>
Matrica<TipElemenata>   StvoriMatricu(int   broj_redova,   int
broj_kolona) {
  Matrica<TipElemenata> mat;
  mat.broj_redova = broj_redova; mat.broj_kolona = broj_kolona;
  mat.elementi = new TipElemenata*[broj_redova];
  for(int i = 0; i < broj_redova; i++) mat.elementi[i] = 0;
  try {
    for(int i = 0; i < broj_redova; i++)
      mat.elementi[i] = new TipElemenata[broj_kolona];
  }
  catch(...) {
    UnistiMatricu(mat);
    throw;
  }
  return mat;
}

template <typename TipElemenata>
void UnesiMatricu(char ime_matrice, Matrica<TipElemenata> &mat)
{
  for(int i = 0; i < mat.broj_redova; i++)
    for(int j = 0; j < mat.broj_kolona; j++) {
 cout << ime_matrice << "(" << i + 1 << "," << j + 1 << ") = ";
      cin >> mat.elementi[i][j];
    }
}

template <typename TipElemenata>
void   IspisiMatricu(const   Matrica<TipElemenata>   &mat,   int
sirina_ispisa) {
  for(int i = 0; i < mat.broj_redova; i++) {
    for(int j = 0; j < mat.broj_kolona; j++)
      cout << setw(sirina_ispisa) << mat.elementi[i][j];
    cout << endl;
  }
}

template <typename TipElemenata>
Matrica<TipElemenata>  ZbirMatrica(const  Matrica<TipElemenata>
&m1,
  const Matrica<TipElemenata> &m2) {
    if(m1.broj_redova != m2.broj_redova
      || m1.broj_kolona != m2.broj_kolona)
        throw "Matrice nemaju jednake dimenzije!\n";
  Matrica<TipElemenata>
m3(StvoriMatricu<TipElemenata>(m1.broj_redova,
    m1.broj_kolona));
  for(int i = 0; i < m1.broj_redova; i++)
    for(int j = 0; j < m1.broj_kolona; j++)
      m3.elementi[i][j]       =       m1.elementi[i][j]       +
m2.elementi[i][j];
  return m3;
```

```
}

int main() {
  Matrica<double> a = {0, 0, 0}, b = {0, 0, 0}, c = {0, 0, 0};
  int m, n;
  cout << "Unesi broj redova i kolona za matrice:\n";
  cin >> m >> n;
  try {
    a = StvoriMatricu<double>(m, n);
    b = StvoriMatricu<double>(m, n);
    cout << "Unesi matricu A:\n";
    UnesiMatricu('A', a);
    cout << "Unesi matricu B:\n";
    UnesiMatricu('B', b);
    cout << "Zbir ove dvije matrice je:\n";
    IspisiMatricu(c = ZbirMatrica(a, b), 7);
  }
  catch(...) {
    cout << "Nema dovoljno memorije!\n";
  }
  UnistiMatricu(a); UnistiMatricu(b); UnistiMatricu(c);
  return 0;
}
```

10.7 Klasa Sat

Definirajte i implementirajte klasu "Sat" koja predstavlja digitalni sat. Klasa treba da ima sljedeći interfejs:

```
Sat();
Sat(int sati, int minute, int sekunde);
void Postavi(int sati, int minute, int sekunde);
void Sljedeći();
void Prethodni();
void PomjeriZa(int pomak);
int DajSate() const;
int DajMinute() const;
int DajSekunde() const;
void Ispisi() const;
```

Konstruktur bez parametara inicijalizira sat na nulu, dok konstruktor sa tri parametra omogućava postavljanje sata na zadani iznos sati, minuta i sekundi. Ovaj konstruktor treba da baci izuzetak u slučaju da se zadaju neispravni parametri. Metoda "Postavi" obablja isti zadatak kao i konstruktor, a služi za naknadno postavljanje vremena. Metoda "Sljedeci" treba da poveća vrijeme zapamćeno u satu za 1 sekundu (npr. ukoliko je tekuće vrijeme "12:48:59", nakon poziva ove metode vrijeme treba da postane "12:49:00"). Slično, metoda "Prethodni" treba da smanji vrijeme zapamćeno u satu za 1 sekundu, dok metoda "PomjeriZa" predstavlja generalizaciju prethodne dvije metode tako što vrši pomak tekućeg vremena za broj sekundi koji je zadan parametrom "pomak" (pomjeranje je unazad ukoliko je vrijednost parametra negativna). Metoda "Ispisi" treba da ispiše stanje sata u obliku "hh : mm : ss". Metode "DajSate", "DajMinute" i "DajSekunde" vraćaju trenutni broj sati, minuta i sekundi u tekućem vremenu. Kao atribute klase uzmite trenutni broj sati, minuta i sekundi, koje ćete definirati kao privatne atribute. Obavezno napišite i testni program u kojem će se upotrebiti svi elementi interfejsa napisane klase. Uputa: Metode "Sljedeci" i

"Prethodni" mogu se realizirati pozivom metode "PomjeriZa", s obzirom da su one njen specijalni slučaj.

10.8 **Klasa Sat**

Proširite klasu "Sat" iz Zadatka 3. sa Tutoriala 8. 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.

10.8 **Klasa Sat**

Definirajte i implementirajte klasu "Sat" koja ima potpuno isti interfejs i potpuno isto ponašanje kao klasa iz prethodnog zadatka, samo čija se interna struktura umjesto tri atributa koja čuvaju trenutni broj sati, minuta i sekundi sastoji samo od jednog atributa, koji čuva ukupan broj sekundi (npr. umjesto informacije "3 sata, 20 minuta, 15 sekundi" čuva se samo informacija koja kaže 12015 sekundi). Mada će ovo tražiti izmjenu implementacije svih (ili skoro svih) metoda klase (konstruktori i metoda "Postavi" će se sasvim neznatno izmijeniti, metode "Sljedeci", "Prethodni" i "PomjeriZa" će se bitno pojednostaviti, dok će se metode "DajSate", "DajMinute", "DajSekunde" i možda metoda "Ispisi" zakomplicirati), pokažite da će testni program iz prethodnog zadatka bez ikakve prepravke raditi sa ovako modificiranom klasom.

10.9 **Klasa Vektor**

Definirajte i implementirajte klasu "Vektor" u skladu sa deklaracijom i implementacijom prikazanom pred kraj Predavanja 8, s tim što ćete dodati još dva konstruktora, kao i tri nove metode "Postavi x", "Postavi y" i "Postavi z". Konstruktor bez parametara treba da inicijalizira sve koordinate vektora na nule, a konstruktor sa tri parametra treba da inicijalizira sve atribute vektora na vrijednosti date parametrima. Metode "Postavi x", "Postavi y" i "Postavi z" treba da omoguće neovisnu izmjenu pojedinačnih koordinata vektora. Napišite i mali testni program u kojem ćete demonstrirati sve elemente razvijene klase.

10.10 **Klasa Vektor**

Izmijenite implementaciju klase razvijene u prethodnom zadatku tako da se za čuvanje koordinata vektora umjesto tri privatna atributa "x", "y" i "z" koji su tipa realnih brojeva koristi jedan privatni atribut "koordinate" koji je tipa niza od tri realna elementa. Izmjenu treba izvesti tako da zaglavlja svih metoda unutar interfejsa klase ostanu neizmijenjena. Demonstrirajte da će testni program napisan u prethodnom zadatku raditi bez ikakvih izmjena sa ovako izmijenjenom klasom.

10.11 **Klasa Tim**

Definirajte i implementirajte klasu "Tim" koja predstavlja jedan tim u fudbalskom prvenstvu, sa privatnim atributima "ime", "broj odigranih", "broj pobjeda", "broj nerijesenih", "broj poraza", "broj datih", "broj primljenih" i "broj poena" koji sadrže

redom naziv tima (do 20 znakova), broj odigranih utakmica, broj pobjeda, broj neriješenih utakmica, broj poraza, ukupan broj datih i primljenih golova, kao i broj poena za razmatrani tim. Atribut "ime" treba izvesti kao klasični niz znakova. Klasa treba da ima sljedeći interfejs:

```
Tim(const char ime[]);
void ObradiUtakmicu(int broj datih, int broj primljenih);
const char *ImeTima() const;
int BrojPoena() const;
int GolRazlika() const;
void IspisiPodatke() const;
```

Konstruktor treba da postavi ime tima na vrijednost zadanu parametrom, a sve ostale atribute klase na nulu. Metoda "ObradiUtakmicu" treba da na osnovu rezultata utakmice koji joj se prenosi kao parametar (u vidu broja datih i primljenih golova sa posmatrane utakmice) ažurira ne samo atribute koje broje golove, nego i atribute koji broje odigrane utakmice, broj pobjeda, poraza i neriješenih utakmica, kao i broj bodova. Pri tome se za pobjedu dodjeljuju 3 boda, za neriješen rezultat 1 bod, a za poraz ništa. Metode "ImeTima", "BrojPoena" i "GolRazlika" treba da vrate respektivno ime tima (preciznije, pokazivač na prvi znak imena), broj poena kao i gol razliku (tj. razliku između ukupnog broja datih i primljenih golova) za posmatrani tim (ove metode implementirati unutar deklaracije klase). Konačno, metoda "IspisiPodatke" treba da ispiše na ekran sve podatke o timu u jednom redu, i to sljedećim redom: ime tima, broj utakmica, broj pobjeda, broj neriješenih utakmica, broj poraza, broj datih golova, broj primljenih golova i broj poena. Pri ispisu, za ime tima zauzmite prostor od 20 znakova, u kojem ime treba da bude ispisano poravnato ulijevo, a za sve brojne podatke prostor od 4 znaka na ekranu, pri čemu ispis svakog brojčanog podatka treba biti poravnat udesno unutar predviđenog prostora. Napišite i kratki testni program u kojem ćete demonstrirati napisanu klasu.

10.12 Klasa Tim

Napisati klasu "`Tim`" koja predstavlja jedan tim u fudbalskom prvenstvu. Klasa "`Tim`" treba posjedovati privatne atribute nazvane "`ime`", "`broj_odigranih`", "`broj_pobjeda`", "`broj_nerijesenih`", "`broj_poraza`", "`broj_datih`", "`broj_primljenih`" i "`broj_poena`" koji sadrže redom naziv tima (do 20 znakova), broj odigranih utakmica, broj pobjeda, broj neriješenih utakmica, broj poraza, ukupan broj datih i primljenih golova, kao i broj poena za razmatrani tim. Klasa treba da ima sljedeći interfejs:

```
Tim(const char ime[]);
void ObradiUtakmicu(int broj_datih, int broj_primljenih);
const char *ImeTima() const;
int BrojPoena() const;
int GolRazlika() const;
void IspisiPodatke() const;
```

Konstruktor treba da postavi ime tima na vrijednost zadanu parametrom, a sve ostale atribute klase na nulu. Metoda "`ObradiUtakmicu`" treba da na osnovu rezultata utakmice koji joj se prenosi kao parametar (u vidu broja datih i primljenih golova sa posmatrane utakmice) ažurira ne samo atribute koje broje golove, nego i atribute koji broje odigrane utakmice, broj pobjeda, poraza i neriješenih utakmica, kao i broj bodova. Pri tome se za pobjedu dodjeljuju 3 boda, za neriješen rezultat 1 bod, a za

poraz ništa. Metode "ImeTima", "BrojPoena" i "GolRazlika" treba da vrate respektivno ime tima (preciznije, pokazivač na prvi znak imena), broj poena kao i gol razliku (tj. razliku između ukupnog broja datih i primljenih golova) za posmatrani tim (ove metode implementirati unutar deklaracije klase). Konačno, metoda "IspisiPodatke" treba da ispiše na ekran sve podatke o timu u jednom redu, i to sljedećim redom: ime tima, broj utakmica, broj pobjeda, broj neriješenih utakmica, broj poraza, broj datih golova, broj primljenih golova i broj poena. Za ime tima zauzeti prostor od 20 znakova, a za sve brojne podatke prostor od 4 znaka na ekranu. Napisati i kratki testni program u kojem ćete demonstrirati napisanu klasu.

10.13 Klasa Liga

Napisati klasu "Liga" koja se oslanja na klasu "Tim" napisanu u prethodnom zadatku. Klasa treba da ima privatne atribute "broj_timova" i "max_br_timova" koji čuvaju redom broj timova odnosno maksimalni dozvoljeni broj timova u ligi (atribut "max_br_timova" treba da bude konstantni atribut), kao i privatni atribut "timovi" koji će služiti za pristup dinamički alociranom nizu od "max_br_timova" elemenata, pri čemu je svaki element niza pokazivač na objekat tipa "Tim". Interfejs klase treba da izgleda ovako:

```
explicit Liga(int velicina_lige);
~Liga();
void DodajNoviTim(const char ime_tima[]);
void RegistrirajUtakmicu(const char tim1[], const char tim2[],
  int rezultat_1, int rezultat_2);
void IspisiTabelu();
```

Konstruktor treba da izvrši dinamičku alokaciju memorije za prihvatanje onoliko timova koliko je navedeno parametrom, dok destruktor treba da izvrši oslobađanje svih resursa koje je klasa "Liga" alocirala tokom svog rada. Metoda "DodajNoviTim" kreira tim sa navedenim imenom i upisuje ga na prvo slobodno mjesto u ligu (pri tome se naravno broj timova u ligi povećava za jedinicu). Metoda ne smije da dozvoli upis više timova od maksimalno dozvoljenog broja timova. U metodi "RegistrirajUtakmicu" prva dva parametra predstavljaju imena timova koji su odigrali utakmicu, dok su treći i četvrti parametar broj golova koji su dali prvi i drugi tim respektivno. Ova metoda treba da ažurira rezultate u tabeli za oba tima, odnosno da baci izuzetak ukoliko timovi sa navedenim imenima ne postoje u tabeli. Konačno, metoda "IspisiTabelu" treba da ispiše tabelu lige sortiranu u opadajućem poretku po broju bodova. Ukoliko dva tima imaju isti broj poena, tada u tabeli prvo dolazi tim sa većom gol razlikom. Sortiranje vršite pozivom funkcije "sort", uz pogodno definiranu funkciju kriterija, koju možete izvesti kao privatnu statičku funkciju članicu klase. Ispis treba vršiti pozivom metode "IspisiPodatke" iz klase "Tim", tako da bi tabela trebala da ima izgled poput sljedećeg:

```
Čelik          18  11   5   1  34  10  38
Jedinstvo      18   9   4   5  33  20  31
Željezničar    18   9   4   5  25  19  31
Velež          18   8   6   4  23  24  30
Sarajevo       18   8   5   5  32  16  29
```

itd. Obavezno treba napisati i testni program u kojem ćete demonstrirati sve elemente razvijenih klasa ("Tim" i "Liga"), na manjem broju fiksnih timova (npr. liga od 6

elemenata) i rezultatima utakmica koji se unose sa tastature. Predvidjeti i hvatanje eventualno bačenih izuzetaka.

10.14 **Klasa Sa**t

Napisati klasu "`Sat`" koja predstavlja digitalni sat. Klasa treba da ima sljedeći interfejs:

```
Sat();
void Postavi(int sati, int minute, int sekunde);
void Sljedeći();
void Ispisi() const;
```

Konstruktor treba da kreira sat inicijaliziran na "00:00:00", dok metoda "`Postavi`" treba da služi za namještanje tekućeg vremena. Metoda treba da baci izuzetak u slučaju da se zadaju neispravni parametri. Metoda "`Sljedeci`" treba da poveća vrijeme zapamćeno u satu za 1 sekundu (npr. ukoliko je tekuće vrijeme "12:48:59", nakon poziva ove metode vrijeme treba da postane "12:49:00"). Metoda "`Ispisi`" treba da ispiše stanje sata u obliku "*hh* : *mm* : *ss*". Sve eventualno neophodne atribute klase neophodno je definirati kao privatne atribute. Obavezno napisati i testni program u kojem će se upotrebiti napisana klasa.

10.15 **Klasa Robot**

Neka je data deklaracija

```
enum Pravci {Sjever, Istok, Jug, Zapad};
```

Definirati klasu "`Robot`" koja predstavlja zamišljenog robota koji može da se kreće kroz koordinatni sistem sa cjelobrojnim koordinatama i koji može da gleda u jednom od četiri pravca definirana u pobrojanom tipu "`Pravci`". Klasa treba da ima sljedeći interfejs:

```
Robot(int x, int y, Pravci p);
void PomjeriSe(int korak);
void Nalijevo();
void Nadesno();
int Pozicija_x() const;
int Pozicija_y() const;
Pravci Orjentacija() const;
void Ispisi() const;
```

Konstruktor treba da kreira robot na zadanoj poziciji koji gleda u zadanom pravcu. Metoda "`PomjeriSe`" pomjera robota zadani broj koraka u pravcu u kojem robot trenutno gleda. Metode "`Nalijevo`" i "`Nadesno`" obrću robota za 90^o na lijevo odnosno na desno. Metode "`Pozicija_x`", "`Pozicija_y`" i "`Orjentacija`" treba da vrate tekuću poziciju odnosno orjentaciju robota. Metoda "`Ispisi`" treba da ispiše podatke o poziciji i orjentaciji robota na način kao u sljedećem primjeru:

Robot se nalazi na poziciji (3,–5) i gleda na istok.

Napisanu klasu upotrebiti u programu u kojem će korisnik moći da bira neku od dozvoljenih operacija sa robotom, a nakon svake izvršene operacije program treba da prikaže tekuću poziciju i orjentaciju (kratke metode dozvoljeno je implementirati odmah unutar deklaracije klase).

10.15 **Klasa Robot**

Neka je data deklaracija

```
enum Pravci {Sjever, Istok, Jug, Zapad};
```

Definirajte klasu "Robot" koja predstavlja zamišljenog robota koji može da se kreće kroz koordinatni sistem sa cjelobrojnim koordinatama i koji može da gleda u jednom od četiri pravca definirana u pobrojanom tipu "Pravci". Klasa treba da ima sljedeći interfejs:

```
void Postavi(int x, int y, Pravci p);
void PomjeriSe(int korak);
void OkreniSeNalijevo();
void OkreniSeNadesno();
int DajPozicijuX() const;
int DajPozicijuY() const;
Pravci DajOrjentaciju() const;
void Ispisi() const;
```

Metoda "Postavi" postavlja robot na zadanu poziciju, i usmjerava ga u zadani pravac. Metoda "PomjeriSe" pomjera robota zadani broj koraka u pravcu u kojem robot trenutno gleda. Metode "OkreniSeNalijevo" i "OkreniSeNadesno" obrću robota za 90o na lijevo odnosno na desno. Metode "DajPozicijuX", "DajPozicijuY" i "DajOrjentaciju" treba da vrate tekuću poziciju odnosno orjentaciju robota. Metoda "Ispisi" treba da ispiše podatke o poziciji i orjentaciji robota na način kao u sljedećem primjeru:

Robot se nalazi na poziciji (3,–5) i gleda na istok.

Napisanu klasu upotrijebite u programu u kojem će korisnik moći da bira neku od dozvoljenih operacija sa robotom, a nakon svake izvršene operacije program treba da prikaže tekuću poziciju i orjentaciju (kratke metode dozvoljeno je implementirati odmah unutar deklaracije klase).

10.15 **Klasa Robot**

Definirajte klasu "Robot" koja predstavlja zamišljenog robota koji može da se kreće kroz koordinatni sistem sa realnim koordinatama i koji može da gleda u proizvoljnom pravcu. Klasa treba da ima sljedeći interfejs:

```
void Postavi(double x, double y, double ugao);
void IdiNaprijed(double pomak);
void IdiNazad(double pomak);
void OkreniSeNaijevo(double ugao);
void OkreniSeNadesno(double ugao);
double DajPozicijuX() const;
double DajPozicijuY() const;
double DajOrjentaciju() const;
void Ispisi() const;
```

Metoda "Postavi" postavlja robot na zadanu poziciju, i usmjerava ga da gleda pod zadanim uglom u odnosu na pozitivni smjer x ose. Pri tome se ugao zadaje u stepenima (to vrijedi i za sve ostale metode u kojima se spominju uglovi). Metoda "IdiNaprijed" pomjera robota za zadani iznos dužine u smjeru u kojem robot trenutno

gleda, dok metoda "IdiNazad" pomjera robota za zadani iznos dužine u smjeru suprotnom od smjera u kojem robot trenutno gleda. Metode "OkreniSeNalijevo" i "OkreniSeNadesno" obrću robota za zadani iznos ugla (u stepenima) nalijevo odnosno nadesno (tj. u smjeru suprotnom od kazaljke na satu, odnosno u smjeru kazaljke na satu). Metode "DajPozicijuX", "DajPozicijuY" i "DajOrjentaciju" treba da vrate tekuću poziciju odnosno ugao pod kojim gleda robot. Metoda "Ispisi" treba da ispiše podatke o poziciji i orjentaciji robota na način kao u sljedećem primjeru:

Robot se nalazi na poziciji (3.42,–5.173) i gleda pod uglom 42.5 stepeni u odnosu na x osu.

Napisanu klasu upotrijebite u programu u kojem će korisnik moći da bira neku od dozvoljenih operacija sa robotom, a nakon svake izvršene operacije program treba da prikaže tekuću poziciju i orjentaciju (kratke metode dozvoljeno je implementirati odmah unutar deklaracije klase).

10.16 Klasa Datum

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

```
enum Mjeseci {Januar, Februar, Mart, April, Maj, Juni,
Juli, August,
Septembar, Oktobar, Novembar, Decembar};
enum Dani {Ponedjeljak, Utorak, Srijeda, Cetvrtak, Petak,
Subota,
Nedjelja);
Datum(int dan, Mjeseci mjesec, int godina);
void Postavi(int dan, Mjeseci mjesec, int godina);
int Dan() const;
Mjeseci Mjesec() const;
char *ImeMjeseca() const;
int Godina() const;
bool DaLiJePrestupna() const;
int Sedmica() const;
int BrojDanaOdPocetkaGodine() const;
Dani DanUSedmici() const;
char *ImeDanaUSedmici() const;
void Sljedeci();
void Prethodni();
void PomjeriZa(int broj dana);
void IspisiKratko() const;
void IspisiKompletno() const;
```

Konstruktor klase kreira datum i u skladu sa informacijama koje se prosljeđuju kroz parametre. Konstrukto rši kontrolu ispravnosti datuma, i baca izuzetak u slučaju da su proslijeđenipogrešni podaci. Metoda "`Postavi`" načelno obavlja istu funkciju kao i konstruktor sa tri parametra. Metode "`Dan`", "`Mjesec`" i "`Godina`" služe za očitavanje informacija o danu, mjesecu i godini koje su pohranjene u datumu. Metoda "`ImeMjeseca`" vraća ime mjeseca koji se čuva u datumu (preciznije, umjesto

čitavog imena, vraća se samo pokazivač na prvi znak imena). Metoda "`DaLiJePrestupna`" vraća informaciju o tome da li je pohranjena godina 3 prestupna ili ne. Metoda "`Sedmica`" vraća informaciju o tome u kojoj sedmici unutar godine pada zapamćeni datum (u opsegu od 1 do 52). Metoda "`BrojDanaOdPocetkaGodine`" vraća kao rezultat broj dana koji je protekao od početka godine zapamćene u datumu do dana zapamćenog u datumu (npr. ukoliko je u datumu zapamćen 3. 7. 2002. godine, metoda vraća broj dana koji je protekao od 1. 1. 2002. do 3. 7. 2002.). 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 "`ImeDanaUSedmici`" vraća ime dana u sedmici koji odgovara datumu, slično kao metoda "`ImeMjeseca`". Metode "`Sljedeci`" i "`Prethodni`" 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 "`IspisiKratko`" ispisuje datum kao dan, mjesec i godinu prikazane kao brojeve razdvojene tačkom i jednim razmakom, npr. "`5. 12. 2006`". Konačno, metoda "`IspisiKompletno`" 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 također slijedi tačka i, konačno, ime odgovarajućeg dana u sedmici u zagradama. Na primjer, ukoliko je u datumu zapamćen datum 3. 7. 2002. na ekranu treba da se ispiše "`3.Juli 2002.(Srijeda)`". 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.

10.17 **Klasa Datum**

Definirati i implementirati klasu "`Datum`" koja omogućava rad sa datumskim podacima. Klasa treba da sadrži sljedeće elemente:

a) Privatne atribute koji čuvaju informaciju o danu, mjesecu i godini, pri čemu se informacija o mjesecu čuva kao cijeli broj u opsegu od 1 do 12;
b) Konstruktor sa tri parametra koja postavlja atribute za dan, mjesec i godinu na osnovu informacija koje se prosljeđuju kroz parametre. U slučaju neispravnih informacija, konstruktor treba da baci izuzetak;
c.) Metode bez parametara koje služe za očitavanje informacija o danu, mjesecu i godini (kojima se ne može direktno pristupiti, s obzirom da predstavljaju privatne atribute);
d.) Metodu bez parametara koja vraća kao rezultat naziv dana u sedmici koji odgovara datumu (npr. ukoliko je u datumu zapamćen 24. 5. 2004. godine, metoda treba da vrati tekst "Ponedjeljak", preciznije pokazivač na prvi znak teksta);
e.) Preklopljene operatore "`++`" i "`--`", koji pomjeraju datum za jedan dan unaprijed odnosno unazad (potrebno je podržati i prefiksne i postfiksne verzije ovog operatora);
f.) Preklopljene operatore "`+`" i "`-`", pri čemu je dozvoljeno je sabrati primjerak klase "`Datum`" sa pozitivnim cijelim brojem, odnosno oduzeti cijeli broj od primjerka klase "`Datum`", gdje se kao rezultat dobija novi primjerak klase "`Datum`" u kojem je datum pomjeren unaprijed odnosno unazad za broj dana iskazan drugim parametrom (najjednostavnija, ali ujedno i dosta neefikasna ideja je primijeniti operator "`++`" odnosno "`--`" u petlji neophodan broj puta).

g.) Preklopljeni operator "–" koji omogućava oduzimanje dva primjera klase "`Datum`", pri čemu se kao rezultat dobija broj dana između dva datuma.

h.) Preklopljene relacione operatore "==", "`!=`","`<`","`>`","`<=`" i "`>=`" koji daju rezultat poređenja dva datuma (u hronološkom poretku, tj. manji je onaj datum koji dolazi prije po kalendaru).

i.) Preklopljene operatore "+=" pri čemu "`d += n`" odnosno "`d –= n`" predstavlja prosto kraći zapis izraza "`d = d + n`" odnosno "`d = d – n`".

j.) Preklopljeni operator "<<" za ispis datuma na ekran, pri čemu se prvo ispisuje redni broj dana, zatim tačka, zatim puno ime mjeseca (riječima) iza koje slijedi razmak, i konačno redni broj godine iza kojeg takođe slijedi tačka (Na primjer, ukoliko je u datumu zapamćen datum 3. 7. 2002. na ekranu treba da se ispiše "3. Juli 2002.").

k.) Preklopljeni operator ">>" za unos datuma sa tastature. Datum treba da se unosi u obliku *dan*/*mjesec*/*godina*, pri čemu su *dan*, *mjesec* i *godina* cijeli brojevi. U slučaju da unos nije ispravan (što uključuje i slučaj brojeva izvan dozvoljenog opsega), ulazni tok treba postaviti u neispravno stanje.

Sve metode implementirati izvan klase, osim trivijalnih metoda čija implementacija može stati u jedan red ekrana. Obavezno napisati i mali testni program u kojem će se testirati svi zahtijevani elementi ove klase.

10.18 Klasa Polinom

Definirati i implementirati klasu "`Polinom`", koja omogućava rad sa polinomima. Klasa treba da ima konstruktor sa dva parametra koji predstavljaju redom stepen polinoma, kao i ime nezavisne varijable u polinomu, koje obavezno predstavlja *samo jedan znak*, a ne string (ovo će se koristiti samo pri ispisu polinoma). Koeficijenti polinoma treba da se čuvaju u dinamički alociranom nizu realnih brojeva. Zbog toga je neophodno da klasa ima definirane i destruktor, konstruktor kopije i prekopljeni operator dodjele (inače neće biti moguće definirati operatorske funkcije koje će raditi ispravno). Klasa bi trebala da ima preklopljeni operator "`[]`" kojim se omogućava pristup koeficijentima polinoma. Indeksi koeficijenata polinoma se kreću od nule (slobodni član) do stepena polinoma, a u slučaju indeksa izvan opsega treba baciti izuzetak. Za računanje vrijednosti polinoma koristi se preklopljeni operator "`()`" kojem se prosljeđuje vrijednost argumenta za koji treba izračunati vrijednost polinoma. Treba podržati i binarne operatore "+", "–", "*", "/" i "%" za sabiranje, oduzimanje, množenje, djeljenje i nalaženje ostatka pri dijeljenju dva polinoma, njihove bliske srodnike "+=", "–=", "*=","/=" i "%=", kao i unarni operator "–" koji kao rezultat daje polinom sa izvrnutim predznacima svih koeficijenata. Binarne operacije nad polinomima su podržane samo nad dva polinoma koji zavise od iste promjenljive (u suprotnom, treba baciti izuzetak). Dalje, treba podržati i binarne relacione operatore "==" i "`!=`" koji testiraju jednakost, odnosno nejednakost dva polinoma. Konačno, treba podržati i operator za ispis "<<" koji ispisuje polinom na ekran, koristeći ime nezavisne varijable koje je bilo zadato u konstruktoru. Na primjer, mogući izgled ispisanog polinoma mogao bi izgledati ovako:

```
3 + 2 x^2 - 4 x^3 - x^5 + 6 x^6
```

uz pretpostavku da su koeficijenti polinoma bili redom 3, 0, 2, –4, 0, –1 i 6. Obavezno napisati i mali testni program u kojem će se testirati svi zahtijevani elementi ove klase.

10.18 Klasa Polinom

Definirajte i implementirajte klasu “Polinom”, koja omogućava rad sa polinomima. Klasa treba da ima konstruktor sa jednim parametrom koji predstavlja stepen polinoma. Ovaj konstruktor ne smije se koristiti za automatsku pretvorbu cijelih brojeva u objekte tipa “Polinom”. Koeficijenti polinoma treba da se čuvaju u vektoru realnih brojeva (tj. atributu tipa “vector<double>”). Zahvaljujući ovoj činjenici, destruktor, konstruktor kopije i preklopljeni operator dodjele u klasi “Polinom” neće biti potrebni (za oslobađanje memorije kao i za ispravno kopiranje i međusobno dodjeljivanje objekata tipa “Polinom” pobrinuće se destruktor, konstruktor kopije i preklopljeni operator dodjele implementirani unutar tipa “vector<double>”). Klasa bi trebala da ima preklopljeni operator “[]” kojim se omogućava pristup koeficijentima polinoma. Indeksi koeficijenata polinoma se kreću od nule (slobodni član) do stepena polinoma, a u slučaju indeksa izvan opsega treba baciti izuzetak. Za računanje vrijednosti polinoma koristi se preklopljeni operator “()” kojem se prosljeđuje vrijednost argumenta za koji treba izračunati vrijednost polinoma. Treba podržati i binarne operatore “+”, “–”, “*”, “/” i “%” za sabiranje, oduzimanje, množenje, djeljenje i nalaženje ostatka pri dijeljenju dva polinoma, njihove bliske srodnike “+=”, “–=”, “*=”,“/=” i “%=”, kao i unarni operator “–” koji kao rezultat daje polinom sa izvrnutim predznacima svih koeficijenata. Dalje, treba podržati i binarne relacione operatore “==” i “!=” koji testiraju jednakost, odnosno nejednakost dva polinoma. Konačno, treba podržati i operator za ispis “<<” koji ispisuje polinom na ekran, koristeći “x” kao ime nezavisne varijable. Na primjer, mogući izgled ispisanog polinoma mogao bi izgledati ovako:

```
3 + 2 x^2 - 4 x^3 - x^5 + 6 x^6
```

uz pretpostavku da su koeficijenti polinoma bili redom 3, 0, 2, –4, 0, –1 i 6. Obavezno napišite i mali testni program u kojem će se testirati svi elementi ove klase.

10.18 **Klasa Polinom**

Definirajte klasu “Polinom”, koja omogućava rad sa polinoma sa jednom nezavisnom varijablom (koju ćemo, bez umanjenja općenitosti, zvati x). Klasa treba da sadrži sljedeće elemente:

a) Privatne atribute koji predstavljaju stepen polinoma i pokazivač na dinamički alocirani niz koji čuva koordinate polinoma.
b) Konstruktor sa jednim parametrom, koji predstavlja deklarirani stepen polinoma (tj. najveći dozvoljeni eksponent uz neki od članova polinoma). Ovaj konstruktor treba da izvrši dinamičku alokaciju prostora za koeficijente polinoma, koji trebaju da se automatski inicijaliziraju na nulu. Pri tome treba zabraniti da se ovaj konstruktor koristi za automatsku konverziju tipova.
c) Konstruktor sa dva parametra, od kojih prvi predstavlja deklarirani stepen polinoma, a drugi niz realnih brojeva. Ovaj konstruktor treba da izvrši dinamičku alokaciju prostora za koeficijente, i izvrši inicijalizaciju svih koeficijenata vrijednostima iz navedenog niza (za koji pretpostavljamo da sadrži dovoljno elemenata za neophodne inicijalizacije, barem za 1 više od deklariranog stepena polinoma).
d) Destruktor, koji oslobađa svu memoriju koju je neki primjerak klase “Polinom” zauzeo tokom svog postojanja.
e) Konstruktor kopije i preklopljeni operator dodjele, koji obezbjeđuju sigurno kopiranje i međusobno dodjeljivanje primjeraka klase “Polinom”.
f) Metodu bez parametara, koja vraća kao rezultat deklarirani stepen polinoma.

g) Metodu bez parametara, koja vraća stvarni stepen polinoma, koji može biti i manji od deklariranog u slučaju da su koeficijenti uz članove polinoma sa najvećim stepenima jednaki nuli. Na primjer, stvarni stepen polinoma 0 x4 + 0 x3 + 3 x2 + 5 x + 6 je 2 a ne 4. Stvarni stepen polinoma je zapravo najveći stepen među svim članovima koji imaju nenulte koeficijente.

h) Preklopljeni operator indeksiranja "[]" koji omogućava pristup koeficijentima polinoma, pri čemu "p[i]" predstavlja koeficijent uz član sa i-tim stepenom (tako da je "p[0]" slobodni član). Ovaj operator treba definirati tako da radi korektno i sa konstantnim i sa nekonstantnim objektima, pri čemu u slučaju nekonstantnih objekata treba dozvoliti mogućnost promjene koeficijenata. U slučaju da je indeks izvan opsega, treba baciti izuzetak.

i) Preklopljeni funkcijski operator "()" koji omogućava računanje polinoma za vrijednost argumenta koji se prosljeđuje kao parametar, tako da je "p(x)" vrijednost polinoma "p" u tački "x".

j) Metodu sa jednim parametrom, koja vrši promjenu deklariranog stepena polinoma na novu vrijednost. Prilikom promjene, svi koeficijenti koji postoje i u novom polinomu treba da zadrže iste vrijednosti kakve su bile i prije promjene, a eventualne novododane koeficijente treba inicijalizirati na nulu.

k) Metodu bez parametara koja vrši redukciju deklariranog stepena polinoma na njegov stvarni stepen, tj. nakon poziva ove metode deklarirani i stvarni stepen polinoma trebaju biti jednaki (uputa: iskoristite prethodnu metodu).

l) Preklopljeni operator "!" koji vraća logičku vrijednost "true" ukoliko je polinom nul-polinom (tj. polinom čiji su svi koeficijenti nule), inače vraća logičku vrijednost "false".

m) Preklopljeni operator "+" za sabiranje polinoma. Dozvoljeno je sabrati dva polinoma, kao i sabrati polinom sa brojem odnosno broj sa polinomom (pri tome se broj sabira samo sa slobodnim članom polinoma, a ostali koeficijenti ostaju netaknuti). Podrazumijeva se da oba polinoma zavise od iste nezavisne varijable.

n) Preklopljeni operator "+=" koji će obezbijediti da izrazi "X += Y" i "X = X + Y" uvijek imaju isto značenje, kad god ovaj drugi izraz ima smisla.

o) Preklopljeni operator "++" koji povećava slobodni član polinoma za jedinicu. Potrebno je podržati kako prefiksnu, tako i postfiksnu verziju ovog operatora.

p) Preklopljene relacione operatore "==" i "!=" koji daju logičku vrijednost "true" ako i samo ako su polinomi jednaki odnosno različiti.

q) Preklopljeni operator za ispis "<<" koji ispisuje polinom na ekran, koristeći "x" kao ime nezavisne varijable. Na primjer, uz pretpostavku da su koeficijenti polinoma bili redom 3, 0, –2, 4, 0, –1 i 6, mogući izgled ispisanog polinoma mogao bi izgledati ovako:

```
3 - 2 x^2 + 4 x^3 - x^5 + 6 x^6
```

Sve metode implementirajte izvan deklaracije klase, osim trivijalnih metoda koje trebate implementirati direktno unutar deklaracije klase. Sve metode koje su inspektori, obavezno deklarirajte kao takve. Pored toga, potrebno je da napišete i testni program u kojem ćete demonstrirati sve elemente napisane klase.

10.19 **Klasa Student**

Napisati i implementirati baznu klasu "`Student`", zatim iz nje izvesti naslijeđene klase "`Student_VI_stepena`", "`Student_VII_stepena`", "`Student_postdiplomac`", te

konačno formirati i implementirati klasu "StudentskaSluzba". Bazna klasa "Student" treba da ima sljedeće elemente:

a) Privatne atribute "broj_indeksa", "ime", "prezime", "broj_ocjena" i "prosjecna_ocjena". Atribut "broj_ocjena" sadrži broj ispita koje je student položio, dok su ostali atributi jasni iz naziva. Atributi "ime" i "prezime" trebaju biti tipa "string".
b) Metodu "VratiIndeks" koja vraća broj indeksa studenta.
c) Konstruktor sa 3 parametra koji postavlja atribute "ime", "prezime" i "broj_indeksa" na vrijednosti koje se zadaju preko parametara, a atribute "prosjecna_ocjena" i "broj_ocjena" na 5 i 0 respektivno.
d) Metodu "upisi_ocjenu" koja ima kao parametar ocjenu koju je dobio student na posljednjem ispitu i koja ažurira atribute "broj_ocjena" i "prosjecna_ocjena" u skladu sa novom ocjenom.
e) Virtualnu metodu "IspisiPodatke" bez parametara koja ispisuje podatke o studentu u slijedećem obliku (prosjek se ispisuje na 2 decimale).

Student <ime> <prezime> sa brojem indeksa <indeks> ima prosjek <prosjek>.

Izvedene klase "Student_VI_stepena" i "Student_VII_stepena" razlikuju se samo u metodi "IspisiPodatke" koja ispisuje "Student VI stepena" odnosno "Student VII stepena" umjesto samo "Student". Izvedena klasa "Student_postdiplomac" ima i dodatni atribut "GodinaDiplomiranja" kao i dodatni parametar u konstruktoru koji omogućava postavljanje ovog atributa. Metoda "IspisiPodatke" za ovu klasu ispisuje podatke o studentu u slijedećem obliku:

Student postdiplomskog studija <ime> <prezime> sa brojem indeksa <indeks>, diplomirao godine <godina>, ima prosjek <prosjek>.

Klasa "StudentskaSluzba" treba da ima slijedeće elemente:

a) Privatne atribute "broj_studenata", "MaxBrStudenata" i "studenti". Atributi "broj_studenata" i "MaxBrStudenata" imaju jasna značenja (pored toga, atribut "MaxBrStudenata" treba biti konstantan), dok je atribut "studenti" pokazivač na prvi element dinamičkog niza čiji su elementi pokazivači na objekte tipa "Student").
b) Konstruktor sa jednim parametrom koji predstavlja kapacitet studentske službe (tj. maksimalni broj studenata koji se mogu upisati) i koji vrši neophodnu alokaciju memorije, kao i odgovarajući destruktor. Kopiranje primjeraka klase kao i njihovo međusobno dodjeljivanje treba zabraniti.
c) Metodu "UpisiStudenta", koja upisuje novog studenta. Metoda ima 4 parametra: broj indeksa, ime, prezime i godina diplomiranja. Za studente koji se upisuju na dodiplomski studij, umjesto godine diplomiranja upisuje se stepen studija (6 ili 7).
d) Metodu "UpisiOcjenu" sa 2 parametra: broj indeksa studenta čija se ocjena upisuje, kao i sama ocjena;
e) Metodu "IspisiSpisak" koja ispisuje spisak svih studenata sa pripadnim podacima (pozivom metode "IspisiPodatke" za svakog studenta).

Obavezno napisati i testni program u kojem ćete testirati sve elemente navedenih klasa. Kao test ispravnosti klase može da posluži slijedeća sekvenca naredbi:

```
StudentskaSluzba pmf(1000);
pmf.UpisiStudenta(11,"Ivo","Ivić",6);
pmf.UpisiStudenta(12,"Meho","Mehić",1998);
pmf.UpisiStudenta(13,"Vaso","Vasić",7);
pmf.UpisiStudenta(14,"Ibro","Ibrić",6);
```

```
pmf.UpisiStudenta(15,"Marko","Marković",2001);
pmf.UpisiOcjenu(11,8);
pmf.UpisiOcjenu(11,7);
pmf.UpisiOcjenu(11,9);
pmf.UpisiOcjenu(12,7);
pmf.UpisiOcjenu(12,8);
pmf.UpisiOcjenu(13,6);
pmf.UpisiOcjenu(13,10);
pmf.UpisiOcjenu(14,7);
pmf.UpisiOcjenu(14,8);
pmf.UpisiOcjenu(14,8);
pmf.UpisiOcjenu(15,9);
pmf.IspisiSpisak();
```

Ova sekvenca naredbi trebala bi proizvesti sljedeći ispis na ekranu.

```
Student VI stepena Ivo Ivić sa brojem indeksa 11 ima prosjek 8.
Student postdiplomskog studija Meho Mehić sa brojem indeksa 12,
diplomirao godine 1998, ima prosjek 7.5.
Student VII stepena Vaso Vasić sa brojem indeksa 13 ima prosjek 8.
Student VI stepena Ibro Ibrić sa brojem indeksa 14 ima prosjek 7.66.
Student postdiplomskog studija Marko Marković sa brojem indeksa 15,
diplomirao godine 2001, ima prosjek 9.
```

Najbolje bi bilo da testni program manipulira sa podacima koji se unose sa tastature.

10.20 Klas Temperatura

Za potrebe neke meteorološke stanice neophodno je vršiti čestu registraciju spoljašnje temperature. Za tu svrhu meteorološka stanica koristi računarski program u kojem je definirana i implementirana klasa nazvana “Temperature”. Ova klasa omogućava čuvanje podataka o temperaturi za izvjesni vremenski period u dinamički alociranom nizu realnih brojeva kojem se pristupa preko nekog od internih atributa klase. Poznato je da klasa sadrži sljedeće elemente:

a) Konstruktor sa jednim parametrom, koji vrši dinamičku alokaciju memorije, pri čemu parametar predstavlja maksimalan broj temperatura koje se mogu registrirati (pri tome je potrebno spriječiti da se taj konstruktor koristi za automatsku konverziju cjelobrojnih podataka u objekte tipa “Temperature”).
b) Destruktor, koji oslobađa memoriju zauzetu konstruktorom.
c) Konstruktor kopije koji će osigurati da se objekti tipa “Temperature” mogu bezbjedno prenositi kao parametri po vrijednosti, koristiti za inicijalizaciju drugih objekata istog tipa i vraćati kao rezultati iz funkcija.
d) d) Preklopljeni operator dodjele, koji garantira sigurno dodjeljivanje jednog objekta tipa “Temperature” drugom objektu istog tipa, bez problema koji mogu biti uzrokovani kreiranjem plitkih kopija;
e) Metodu koja vrši registraciju nove temperature, pri čemu se temperatura koja se registrira prenosi kao parametar metode (u slučaju da se dostigne maksimalan broj temperatura koje se mogu registrirati, treba baciti izuzetak);
f) Metodu koja briše sve unesene temperature;
g) Metode koje vraćaju kao rezultat prosječnu, minimalnu i maksimalnu temperaturu;
h) Metodu koja ispisuje sve unesene (registrirane) temperature sortirane u opadajućem poretku

i) (tj. najveća temperatura se ispisuje prva), pri čemu se svaka temperatura ispisuje u posebnom redu.

Implementirajte klasu sa navedenim svojstvima. Sve neophodne atribute obavezno izvesti kaoprivatne članove klase, a sve metode implementirati izvan klase, osim metoda čija implementacija zahtijeva jednu ili dvije naredbe. Obavezno napišite i mali testni program u kojem će se testirati sve navedene metode.

10.21 **Klasa Vektor**

Za potrebe nekog računarskog programa za naučno-tehničke proračune koristi se klasa "Vektor". Ova klasa omogućava čuvanje podataka o vektorima u ravni ili prostoru. Poznato je da klasa sadrži slijedeće elemente:

a) Privatne atribute koji čuvaju informaciju o tri koordinate vektora (sve tri koordinate mogu biti realni brojevi).
b) Konstruktor bez parametara koji postavlja sve tri koordinate na nulu, konstruktor sa dva parametra koji postavlja prve dvije koordinate na vrijednosti zadane parametrima a treću koordinatu na nulu, i konstruktor sa tri parametra koji postavlja sve tri koordinate na vrijednostizadaneparametrima.
c) Metodu sa tri parametra koja obavlja istu funkciju kao i konstruktor sa tri parametra.
d) Metode bez parametara koje služe za očitavanje informacija o koordinatama (kojima se ne može direktno pristupiti, s obzirom da predstavljaju privatne atribute).
e) Metodu bez parametara koja vraća dužinu vektora (koja se računa kao kvadratni korijen iz sume kvadrata sve tri koordinate).
f) Metodu bez parametara koja vraća kao rezultat logičku vrijednost "true" ukoliko je vektor ravanski vektor (tj. ukoliko je treća koordinata jednaka nuli), a logičku vrijednost "false" ukoliko nije.
g) Preklopljen operator "<<" za ispis vektora na ekran u obliku "{x,y,z}".
h) Preklopljene operatore "+" i "–", koji vrše sabiranje odnosno oduzimanje dva vektora.
i) Preklopljene operatore "+=" i "–=" pri čemu izrazi "d += n" i "d –= n" predstavljaju prosto kraći zapis izraza "d = d + n" i "d = d + n".
j) Preklopljen operator "*" za množenje. Dozvoljeno je pomnožiti vektor sa brojem, odnosno broj sa vektorom, a također je dozvoljeno pomnožiti dva vektora, pri čemu se kao rezultat dobija broj (skalarni produkt).
k) Preklopljen operator "%" koji računa vektorski produkt dva vektora. Vektorski produkt vektora {x1, y1, z1} i {x2, y2, z2} dat je izrazom {y1z2 – y2z1, z1x2– x1z2, x1y2– x2y1}.
l) Preklopljen operator "|" koji računa ugao između dva vektora. Podsjetimo se da je ugao ϕ između dva vektora v1 i v2 vezan s njima relacijom v1.v2 = |v1| |v2| cos ϕ, pri čemu "." označava skalarni produkt.

Definirajte i implementirajte klasu sa navedenim svojstvima. Sve metode implementirati izvan klase, osim trivijalnih metoda čija implementacija može stati u jedan red ekrana. Obavezno napisati i mali testni program u kojem će se testirati svi zahtijevani elementi ove klase.

10.21 **Klasa VektorNd**

Za potrebe nekog računarskog programa za naučno-tehničke proračune koristi se klasa "VektorNd". Primjerci ove klase predstavljaju n-dimenzionalne vektore, odnosno uređene ntorke realnih brojeva. Poznato je da klasa sadrži sljedeće elemente:

a) Privatne atribute koji predstavljaju dimenziju vektora (tj. broj koordinata vektora) i pokazivač na dinamički alocirani niz koji čuva koordinate vektora.
b) Konstruktor sa dva parametra, od kojih prvi predstavlja dimenziju vektora, a drugi inicijalnu vrijednost na koju treba inicijalizirati sve koordinate vektora. Ovaj konstruktor treba da izvrši dinamičku alokaciju prostora za koordinate, i izvrši inicijalizaciju svih koordinata navedenom vrijednošću. Drugi parametar konstruktora treba da ima podrazumijevanu vrijednost 0, tako da se u slučaju njegovog izostavljanja sve koordinate vektora inicijaliziraju na nulu (odnosno, ovaj konstruktor se može koristiti i sa samo jednim parametrom). Pri tome treba zabraniti da se ovaj konstruktor koristi za automatsku konverziju tipova.
c) Konstruktor sa dva parametra, od kojih prvi predstavlja dimenziju vektora, a drugi niz realnih brojeva. Ovaj konstruktor treba da izvrši dinamičku alokaciju prostora za koordinate, i izvrši inicijalizaciju svih koordinata vrijednostima iz navedenog niza.
d) Destruktor, koji oslobađa svu memoriju koju je neki primjerak klase "VektorNd" zauzeo tokom svog postojanja. e) Konstruktor kopije i preklopljeni operator dodjele, koji obezbjeđuju sigurno kopiranje i međusobno dodjeljivanje primjeraka klase "VektorNd".
e) Metodu bez parametara, koja vraća kao rezultat dimenziju vektora.
f) Metodu sa jednim parametrom, koja daje vrijednost koordinate vektora čiji je redni broj zadan parametrom. Redni broj koordinate mora biti u opsegu od 1 do N gdje je N dimenzija vektora. U suprotnom, metoda treba da baci izuzetak.
g) Metodu sa dva parametra, koja postavlja zadanu koordinatu vektora na zadanu vrijednost. Prvi parametar je redni broj koordinate, a drugi parametar vrijednost koordinate. Redni broj koordinate mora biti u opsegu od 1 do N gdje je N dimenzija vektora. U suprotnom, metoda treba da baci izuzetak.
h) Metodu bez parametara, koja vraća dužinu vektora (koja se računa kao kvadratni korijen iz sume kvadrata svih koordinata).
i) Prijateljsku funkciju sa jednim parametrom, koja vraća dužinu vektora koji je zadan parametrom (odnosno, koja obavlja isti zadatak kao i prethodna metoda, samo uz korištenje drugačije sintakse). k) Metodu bez parametara koja daje rang vektora, odnosno broj koordinata vektora koje su različite od nule.
j) Metodu sa jednim parametrom, koja vrši promjenu dimenzionalnosti vektora na novu vrijednost. Prilikom redimenzioniranja, sve koordinate koje postoje i u novom vektoru treba da zadrže iste vrijednosti kakve su bile i prije redimenzioniranja, a eventualne novododane koordinate treba inicijalizirati na nulu.
k) Preklopljeni operator "<<" za ispis vektora na ekran. Vektor treba ispisati kao slijed koordinata razdvojenih zarezima unutar vitičastih zagrada (npr. "{3,–1,2,0,5}").
l) Preklopljene operatore "+" i "–", koji vrše sabiranje odnosno oduzimanje dva vektora.
m) Vektori se sabiraju ili oduzimaju tako da se saberu odnosno oduzmu odgovarajuće koordinate oba vektora. Pri tome, oba vektora moraju imati istu dimenziju. U suprotnom, treba baciti izuzetak.
n) Preklopljen operator "*" za množenje. Dozvoljeno je pomnožiti vektor sa brojem, odnosno broj sa vektorom, a također je dozvoljeno pomnožiti dva vektora, pri čemu se kao rezultat dobija broj (skalarni produkt). Pri množenju vektora brojem, sve koordinate vektora se množe tim brojem. Skalarni produkt dva vektora se računa tako da se saberu svi produkti odgovarajućih koordinata jednog i drugog vektora. Pri tome, oba vektora moraju imati istu dimenziju. U suprotnom, treba baciti izuzetak.
o) Preklopljene operatore "+=", "–=" i "*=", koji treba da obezbijede da izrazi "X += Y", "X -= Y" i "X *= Y" imaju isto značenje kao i izrazi "X = X + Y", "X = X – Y" i "X = X * Y" kad god ovi izrazi imaju smisla.

p) Preklopljeni unarni operator "++" koji povećava sve koordinate vektora za jedinicu. Potrebno je podržati kako prefiksnu, tako i postfiksnu verziju ovog operatora. r) Preklopljeni unarni operator "!" koji daje logičku vrijednost "true" ako i samo ako je razmatrani vektor nul-vektor (tj. ukoliko su mu sve koordinate jednake nuli).

q) Preklopljene relacione operatore "==" i "!=" koji daju logičku vrijednost "true" ako i samo ako su vektori jednaki odnosno različiti. Vektori su jednaki ako i samo ako su iste dimenzije i ako su im sve odgovarajuće koordinate međusobno jednake. t) Prijateljsku funkciju sa dva parametra koja daje kao rezultat ugao između vektora koji su zadani parametrima. Podsjetimo se da je ugao ϕ između dva vektora v1 i v2 vezan s njima relacijom $v1 \cdot v2 = |v1|\,|v2| \cos\phi$, pri čemu "·" označava skalarni produkt.

Definirajte i implementirajte klasu sa navedenim svojstvima. Sve metode implementirajte izvan klase, osim trivijalnih metoda čija implementacija može stati u jedan red ekrana. Napišite i mali testni program u kojem ćete demonstrirati sve elemente napisane klase.

10.22 Klasa Lik

Definirajte apstraktnu baznu klasu "Lik", koja predstavlja temelj za hijerarhijsku izgradnju porodice klasa koje predstavljaju geometrijske likove. Klasa treba da sadrži privatni atribut "naziv" koji sadrži naziv lika, zatim čisto virtualne metode "IspisiAtribute", "Obim" i "Povrsina", kao i metodu "IspisiPodatke" koja ispisuje podatke o liku pozivajući virtualne metode "IspisiAtribute", "Obim" i "Povrsina". Zatim, iz klase "Lik" naslijedite tri klase "Krug", "Pravougaonik" i "Trougao". U sve tri klase dodajte atribute koji su neophodni za specifikaciju primjeraka te klase, odgovarajući konstruktor, kao i konkretne metode "IspisiAtribute", "Obim" i "Povrsina" koje ispisuju atribute konkretnog lika, odnosno vraćaju kao rezultat obim odnosno površinu konkretnog lika. Sa tako definiranim klasama napravite program koji traži da se sa tastature unese broj likova, i kreira polimorfni niz pokazivača na likove. Dalje, korisnik treba da unosi sa tastature podatke o svakom liku u sljedećem formatu:

```
Kr – Krug radijusa r (npr. K5)
Pa,b – Pravougaonik sa stranicama a i b (npr. P3,2)
Ta,b,c – Trougao sa stranicama a, b i c (npr. T3,4,5)
```

Nakon svakog unesenog podatka, program treba dinamički kreirati odgovarajući objekat, i smjestiti pokazivač na njega u niz. Nakon unosa svih podataka, program treba ispisati podatke o svim unesenim likovima, koristeći poziv metode "IspisiPodatke".

10.23 Tekstualne datoteke

Uz pomoć nekog tekstualnog editora (recimo NotePad-a) kreirajte tekstualnu datoteku "PODACI.TXT" koja sadži podatke o studentima. Datoteka je organizirana na sljedeći način. U prvom redu nalazi se ime i prezime studenta, u drugom redu njegov broj indeksa, a u trećem njegove ocjene. Ocjene su međusobno razdvojene zarezima. Dalje se podaci ponavljaju za svakog od studenata. Slijedi primjer moguće datoteke "PODACI.TXT":

```
Paja Patak
1234
7,8,6,10,8,9,8,7,9,8
Miki Maus
3412
9,7,8,6,8,9,7,7,10
Duško Dugouško
4321
8,10,9,9,10,6,7,7,9,8,10,7,7
```

Zatim napravite program koji iščitava sadržaj ove tekstualne datoteke i kreira drugu tekstualnu datoteku "IZVJESTAJ.TXT" koja sadrži izvještaj o studentima u sljedećem obliku:

```
Student Duško Dugouško, sa indeksom 4321, ima prosjek 8.23
Student Paja Patak, sa indeksom 1234, ima prosjek 8
Student Miki Maus, sa indeksom 3412, ima prosjek 7.89
```

Spisak treba biti sortiran po prosjeku, kao što je gore prikazano. Pretpostavite da ulazna datoteka sadrži samo ispravne podatke, u ispravnom formatu.

10.24 **Tekstualne datotek**e

Proširite klasu "Liga" sa Tutoriala 10. metodom "Sacuvaj" koja sprema cijelo stanje lige u binarnu datoteku čije je ime zadano kao parametar, kao i konstruktorom koji obnavlja stanje lige iz binarne datoteke čije je ime zadano kao parametar. Ukoliko to smatrate potrebnim, dozvoljeno je izvršiti i neke dopune u klasi "Tim" (ali bez mijenjanja njenog interfejsa). Nakon izvršene modifikacije, izmijenite i testni program tako da po završetku rada obavezno snima stanje lige u binarnu datoteku "LIGA.DAT". Na početku rada programa, ukoliko datoteka "LIGA.DAT" postoji, program treba da obnovi sadržaj lige iz ove datoteke, tako da program prosto nastavlja raditi sa istom ligom i istim rezultatima sa kojima je radio prilikom prethodnog pokretanja. U slučaju da datoteka "LIGA.DAT" ne postoji, program treba da kreira novu ligu (na primjer, na osnovu podataka o ligi koji se unose sa tastature).

10.24 **Klasa Pritisci**

Za potrebe neke meteorološke stanice neophodno je vrsiti čestu registraciju atmosferskog pritiska. Za tu svrhu meteorološka stanica koristi računarski program u kojem je definirana i implementirana klasa nazvana "Pritisci". Ova klasa omogućva čuvanje podataka o atmosferskim pritiscima za izvjesni vremenski period u dinamički alociranom nizu realnih brojeva, kojem se pristupa preko odgovarajućeg privatnog atributa. Pored ovog atributa (i eventualno drugih privatnih atributa neophodnih za normalno funkcioniranje klase), poznato je da klasa sadrzi sljedeće elemente:

a) Konstruktor sa tri parametra, koji redom predstavljaju maksimalan broj pritisaka koje se mogu registrirati, minimalni dozvoljeni pritisak i maksimalno dozvoljeni pritisak. Ovaj konstruktor je odgovoran za dinamičku alokaciju memorije.
b) Destruktor, koji oslobadja resurse koji su zauzeti od strane ove klase.
c) Konstruktor kopije, koji obezbjedjuje siguran prenos objekata tipa "Pritisci" kao parametara po vrijednosti u funkcije i njihovo vraćanje kao rezultata iz funkcije, kao i preoklopljeni operator dodjele "=" koji obezbjedjuje sigurnu dodjelu jednog objekta tipa "Pritisci" drugom.
d) Metodu koja vrši registraciju novog pritiska, pri čemu se pritisak koja se registrira prenosi kao parametar metode. U slučju da se dostigne maksimalan broj pritisaka koje se mogu registrirati, ili u slučaju da pritisak nije u dozvoljenom opsegu, metoda treba da baci izuzetak.
e) Metodu koja daje broj registriranih pritisaka.
f) Metodu koja briše sve unesene podatke.
g) Metode koje vracaju kao rezultat prosječni, minimalni i maksimalni pritisak (u slučaju da nema registriranih pritisaka, sve tri metode treba da bace izuzetak). Za realizaciju je potrebno koristiti odgovarajuće funkcije iz biblioteke "algorithm".
h) Metode koje vraćaju kao rezultat broj dana u kojima je pritisak bilo veći odnosno manji od vrijednosti koja se zadaje kao parameta (u slučaju da nema registriranih

pritisaka, metode treba da bace izuzetak). Za realizaciju je potrebno koristiti odgovarajuće funkcije iz biblioteke "algorithm".

i) Metodu koja ispisuje sve unesene (registrirane) pritiske sortirane u opadajućem poretku (tj. najveći pritisak se ispisuje prvi), pri čemu se svaki pritisak ispisuje u posebnom redu. Pri tome je neophodno koristiti funkcije i/ili funktore iz biblioteke "algorithm".
j) Preklopljene operatore "+" i "-" koji djeluju na sljedeći način: Ukoliko je "X" objekat tipa "Pritisci", a "Y" realan broj, tada je "X + Y" novi objekat tipa "Pritisci" u kojem su svi registrirani pritisci povećani za iznos "Y". Na sličan način se interpretira i izraz "X - Y" u slučaju da je "X" objekat tipa "Pritisci", a "Y" realan broj. U slučaju kada su i "X" i "Y" objekti tipa "Pritisci", tada je izraz "X - Y" novi objekat tipa "Pritisci" koji sadrži razlike odgovarajućih pritisaka iz objekata "X" i "Y". U ovom posljednjem slučaju se podrazumijeva da "X" i "Y" sadrže isti broj registriranih pritisaka (u suprotnom, treba baciti izuzetak). U svim ostalim slučajevima, značenje izraza "X + Y" odnosno "X - Y" nije definirano.
k) Preklopljene operatore "+=" i "-=" čiji je cilj da značenje izraza oblika "X += Y" odnosno "X –= Y" uvijek bude identično značenju izraza "X = X + Y" i "X = X - Y".
l) Preklopljeni operator "++" koji povećava sve registrirane pritiske za jedinicu. Potrebno je podržati kako prefiksnu, tako i postfiksnu verziju ovog operatora.
m) Preklopljene relacione operatore "==" i "!=" koje ispituju da li su dva objekta tipa "Pritisci" jednaka ili nisu. Dva objekta ovog tipa smatraju se jednakim ukoliko sadrže isti broj registriranih pritisaka, i ukoliko su svi registrirani pritisci u oba objekta jednaki.

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

10.26 **Klas Vreća**

Definirajte i implementirajte generičku klasu "Vreca" koja služi za smještanje kolekcije elemenata jednakog ali proizvoljnog tipa, pri čemu poredak elemenata u kolekciji nije bitan, i nije moguće imati dva identična elementa unutar kolekcije (slično matematskom pojmu skupa). Kolekcija treba da sadrži sljedeće elemente:

a. Privatne atribute koji redom predstavljaju broj elemenata u vreći, kapacitet vreće (tj. maksimalni broj elemenata koji se mogu smjestiti u vreću), te pokazivač na dinamički alocirani niz (preciznije, na njegov prvi element) koji ce čuvati elemente vreće.
b. Konstruktor sa jednim parametrom tipa cijeli broj, koji predstavlja kapacitet vreće, i koji vrši dinamičku alokaciju prostora za čuvanje elementa vreće. Inicijalno vreća treba da bude prazna. Pri tome je potrebno zabraniti da se ovaj konstruktor koristi za automatsku konverziju cjelobrojnog tipa u tip "Vreca".
c. Destruktor, koji oslobađa memoriju zauzetu konstruktorom.
d. Konstruktor kopije koji će osigurati da se objekti tipa "Vreca" mogu bezbjedno prenositi kao parametri po vrijednosti, koristiti za inicijalizaciju drugih objekata istog tipa i vraćati kao rezultati iz funkcija.
e. Preklopljeni operator dodjele, koji garantira sigurno dodjeljivanje jednog objekta tipa "Vreca" drugom objektu istog tipa, bez problema koji mogu biti uzrokovani kreiranjem plitkih kopija.

f. Metodu sa jednim parametrom, koja dodaje u vreću element predstavljen parametrom. U slučaju da se takav element već nalazi u vreći, unos elementa treba ignorirati. U slučaju da je dostignut kapacitet vreće, metoda treba da baci izuzetak.
g. Metodu bez parametara, koja vraća kao rezultat broj elemenata u vreći.
h. Metodu sa jednim parametrom, koja vraća kao rezultat "true" ukoliko se element predstavljen parametrom nalazi u vreći, a "false" ukoliko se ne nalazi.
i. Metodu sa jednim parametrom koja uklanja element predstavljen parametrom iz vreće. U slučaju da se takav element ne nalazi u vreći, metoda treba da baci izuzetak.
j. Metode bez parametara koja vraća kao rezultat neki od elemenata iz vreće po slučajnom izboru, osim u slučaju kada je vreća prazna (u tom slučaju, treba baciti izuzetak). Upute za realizaciju ove metode date su u napomeni ispod.

Sve metode implementirajte izvan deklaracije klase, osim trivijalnih metoda koje treba implementirati direktno unutar deklaracije klase. Napisane metode testirajte u testnom programu koji unosi niz riječi sa tastature, trpa ih u vreću, a zatim ispisuje na ekran sve riječi smještene u vreću (koristeći samo metode koje objekti tipa "Vreca" podržavaju).

NAPOMENA: Za realizaciju slučajnog izbora potrebno je koristiti funkciju "rand" bez parametara iz biblioteke "cstlib". Ova funkcija vraća slučajni broj u opsegu od 0 do "RAND MAX" gdje je "RAND MAX" neka velika konstanta definirana u istoj biblioteci, tako da ćevrijednost izraza "rand() % N" biti slučajna vrijednost u opsegu od 0 do N–1. Međutim, tako generirani "slučajni" brojevi imaće iste vrijednosti kad god se program pokrene iznova. Ovaj problem se može riješiti tako da se na početku rada generator slučajnih brojeva "zasije" nekom nepredvidljivom vrijednošću, recimo uzetom iz sistemskog sata. To se može učiniti pomoću konstrukcije "srand(time(0))". Najbolje mjesto gdje bi se ova inicijalizacija mogla ubaciti je upravo konstruktor klase "Vreca". Napomenimo da funkcija "srand" (također iz biblioteke "cstdlib") služi za "zasijavanje" generatora, dok funkcija "time" očitava sistemsko vrijeme. Nažalost, funkcija "time" zahtijeva uključivanje biblioteke "ctime" u program.

10.26 Klas Vreća

Definirajte i implementirajte klasu "Trougao" (ili "Trokut") koja omogućava čuvanje podataka o trouglovima (trokutima). Klasa treba da ima sljedeći interfejs (svugdje gdje piše "Trougao" možete pisati "Trokut", u skladu sa Vašim jezičkim preferencijama):

```
Trougao(double a = 0);
Trougao(double a, double b);
Trougao(double a, double b, double c);
void Postavi(double a);
void Postavi(double a, double b);
void Postavi(double a, double b, double c);
static bool TestLegalnosti(double a, double b, double c);
double DajA() const;
double DajB() const;
double DajC() const;
double DajAlfa() const;
double DajBeta() const;
double DajGama() 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);
```

Konstruktor sa tri parametra kreira trougao čije su dužine stranica određene parametrima. Konstruktor sa dva parametra kreira pravougli trougao pri čemu parametri predstavljaju dužine kateta, dok konstruktor sa jednim parametrom kreira jednakostranični trougao pri čemu su dužine svih stranica jednake vrijednosti parametra. Ovaj parametar ima podrazumijevanu vrijednost 0, tako da se ovaj konstruktor može koristiti i kao konstruktor bez parametara, pri čemu se tada kreira trougao čije su sve stranice dužine 0 (tj. koji se degenerira na tačku). Metode "Postavi" (u tri varijante) načelno obavljaju isti zadatak kao i odgovarajući konstruktori, a omogućavaju naknadnu izmjenu podataka o trouglu. Svi konstruktori kao i metode "Postavi" trebaju baciti izuzetak ukoliko nije moguće kreirati trougao sa zadanim parametrima (pri tome, treba dozvoliti degenerirane slučajeve u kojima se trougao reducira na duž, recimo trougao sa stranicama dužine 1, 2 i 3, ili dužine 2, 2 i 0). Statička metoda "TestLegalnosti" 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, metode "DajAlfa", "DajBeta" i "DajGama" vraćaju kao rezultat dužine odgovarajućih uglova trougla u radijanima (alfa je ugao naspram stranice a, itd.), dok metode "DajObim" i "DajPovrsinu" 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 nenegativan, inače treba baciti izuzetak. 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 dinamički alocirati niz od n trouglova (tj. objekata tipa "Trougao") koje treba inicijalizirati na osnovu podataka koji se unose sa tastature (podaci o svakom trouglu se unose posebno). 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. Na kraju, program treba pronaći sve parove podudarnih i sličnih trouglova i ispisati koji su to trouglovi (ili obavijest da takvih parova nema).

10.27 **Klasa Željeznice**

Za potrebe rekonstrukcije i modernizacije Željeznica Bosne i Hercegovine (planirane za prvu polovinu 2047. godine), potrebno je napraviti program koji ce vrsiti automatsku najavu polazaka preko ozvučenja na željezničkoj stanici. Sistem treba da najavljuje sve vozove koji odlaze iz stanice, kao i eventualna kašnjenja u polascima. Za tu svrhu je predvidjeno da program sadrzi klase nazvane "Polazak", koja vodi evidenciju o jednom polasku, i "RedVoznje", koja sadrži informacije o svim polascima u toku jednog dana. Klasa "Polazak" treba da sadrži sljedeće elemente:

a) Privatne atribute koji redom predstavljaju naziv odredišne stanice, broj voza, broj perona (cijeli brojevi), vremenu polaska (sati i minute), informaciju da li se radi o lokalnom ili brzom vozu (false = lokalni, true = brzi), kao i informaciju o eventualnom kašnjenju (u minutama). Pretpostavite da naziv odredišne stanice ne sadrži više od 20 znakova.
b) Konstruktor sa parametrima koji inicijaliziraju sve atribute klase na vrijednosti zadane parametrima konstruktora, osim atributa za eventualno kašnjenje koji se automatski inicijalizira na nulu. Konstruktor treba da baci izuzetak ukoliko bilo koji od parametara ima besmislene vrijednosti (uključujući i predugačak naziv odredisne stanice).
c) Metodu sa jednim parametrom koja postavlja informaciju o eventualnom kašnjenju na vrijednost zadanu parametrom.
d) Metodu kojom je moguće saznati da li odgovarajuci polazak kasni ili ne. Metoda treba da vrati logičku vrijednost "true" u slucaju kašnjenja, a "false" u suprotnom slučaju (kako kod nas stvari stoje, u realnosti ce ova metoda uvijek vracati "true").
e) Metode kojima je moguce saznati očekivano vrijeme polaska (sati i minute) kada se uračuna iznos kašnjenja u odnosu na predvidjeno vrijeme polaska.
f) Preklopljen operator "<<" koji treba da podrži ispis objekata tipa "Polazak" na ekran. U slučaju da se radi o polasku bez kasnjenja, ispis bi trebao da izgleda poput sljedećeg:

```
Lokalni voz broj 3423, odredište Zenica, polazi sa perona 2 u
15:40. Putnicima i voznom osoblju želimo ugodno putovanje.
```

U stvarnom sistemu, ovaj tekst bi se trebao proslijediti sklopu za sintezu govora koji bi emitirao govornu informaciju preko ozvučenja, ali za potrebe ove zadaće zadovoljićemo se prostim ispisom na ekran.

U slučaju da se radi o polasku koji kasni, ispis bi trebao da izgleda poput sljedećeg:

```
Brzi voz broj 358, odrediste Ploče, sa predviđenim vremenom
polaska u 6:20, kasni oko 35 minuta. Izvinjavamo se putnicima
zbog eventualnih neugodnosti.
```

g) Pretpostavlja se da će ovakva poruka razbjesniti putnike, koji znaju da je poruku generirao računar, kojem je potpuno svejedno što voz koji čekate kasni.
h) Preklopljene relacione operatore "<" i ">" koji ispituju koji polazak nastupa ranije, odnosno kasnije. Operator "<" vraća logičku vrijednost "true" ukoliko polazak sa lijeve strane nastupa prije polaska sa desne strane, u suprotnom vraća "false". Analogno vrijedi za operator ">". Prilikom upoređivanja treba uzeti u obzir i očekivano vrijeme kašnjenja, a ne samo planirano vrijeme polaska.

Klasa "RedVoznje" treba da sadrži sljedeće elemente:

a) Privatne atribute koji predstavljaju broj registriranih polazaka u toku dana, maksimalni broj polazaka koji se mogu registrirati, te pokazivač na dinamicki alocirani niz (preciznije, na njegov prvi element) koji će čuvati informacije o svim registriranim polascima. Da biste izbjegli kasnije probleme, niz realizirajte kao niz pokazivača na polaske.
b) Konstruktor sa jednim parametrom tipa cijeli broj, koji predstavlja maksimalni broj polazaka koji se mogu registrirati u redu vožnje, i koji vrši alokaciju prostora za

čuvanje podataka o polascima. Pri tome je potrebno zabraniti da se ovaj konstruktor koristi za automatsku konverziju cjelobrojnog tipa u tip "RedVoznje".

c) Destruktor, koji oslobađa memoriju koju su primjerci ove klase zauzeli tokom svog rada.
d) Konstruktor kopije i preklopljeni operator dodjele koji će osigurati da se objekti tipa "RedVoznje" mogu bezbjedno prenositi kao parametri po vrijednosti, koristiti za inicijalizaciju drugih objekata istog tipa, vraćati kao rezultati iz funkcija i međusobno dodjeljivati.
e) Metodu sa jednim parametrom, koja registrira novi polazak. Parametar metode predstavlja pokazivač na polazak koji želimo registrirati. U slučaju da je dostignut maksimalan broj polazaka koji se mogu registrirati, metoda treba da baci izuzetak.
f) Metodu koja ispisuje na ekran kompletan spisak svih polazaka, počev od zadanog vremena do kraja dana (vrijeme se navodi preko dva parametra, za sate i minute respektivno), sortiranu po očekivanim vremenima polazaka (dakle, uključujući i kašnjenja). Za poređenje dva polaska koristite operatore "<" ili ">" koji su definirani u klasi "Polazak", a za ispis informacija o polasku koristite operator "<<".

Implementirajte klase "Polazak" i "RedVoznje" sa navedenim svojstvima. Sve metode implementirajte izvan tijela klase, osim metoda čija implementacija zahtijeva jednu ili dvije naredbe. Obavezno napišite i mali glavni program, u kojem ćete demonstrirati rad svih napisanih metoda.

10.28 **Klasa Student**

Za potrebe nekog fakulteta neophodno je vršiti evidenciju o polaznicima (studentima) i njihovom uspjehu. Za tu svrhu fakultet koristi računarski program u kojem je definirane i implementirane apstraktna bazna klasa nazvana "Student", dvije klase "StudentBachelor" i "StudentMaster" izvedene iz bazne klase "Student", kao i klasa "Fakultet" koja predstavlja kolekciju objekata izvedenih iz klase "Student". Bazna klasa "Student" treba da ima sljedeće elemente:

a) Privatne atribute koji čuvaju podatke o imenu i prezimenu studenta (po maksimalno 20 znakova), broju indeksa (cijeli broj), broju položenih ispita (cijeli broj) i prosječnoj ocjeni (realan broj), i nikave druge atribute.
b) Konstruktor čiji su parametri ime i prezime studenta, kao i broj indeksa studenta, koji odgovarajuće atribute inicijalizira na vrijednosti zadane parametrima, dok se broj položenih ispita i prosječna ocjena inicijaliziraju respektivno na 0 i 5. U slučaju da su imena preduga, treba baciti izuzetak.
c) Trivijalne metode koje vraćaju vrijednosti svih odgovarajućih atributa.
d) Metodu sa jednim parametrom, koja služi za registraciju novog ispita, a koja povećava broj položenih ispita za jedinicu i ažurira prosječnu ocjenu u skladu sa ocjenom iz novopoloženog ispita, koja se zadaje kao parametar (u slučaju neispravne ocjene, treba baciti izuzetak).
e) Čistu virtualnu metodu "IspisiPodatke" bez parametara, koja ovu baznu klasu čini apstraktnom.

Izvedena klasa "StudentBachelor" razlikuje se od bazne klase "Student" samo po tome što sadrži konkretnu implementaciju virtualne metode "IspisiPodatke", koja ispisuje podatke o studentu u obliku

Student bachelor studija <***ime***> <***prezime***>***, sa brojem indeksa*** <***indeks***>***, ima prosjek*** <***prosjek***>***.***

Izvedena klasa "StudentMaster" razlikuje se od bazne klase "Student" po tome što sadrži dodatni atribut koji čuva informaciju kada je student završio prvi stepen studija (bachelor studij), kao i dodatni parametar u konstruktoru koji omogućava postavljanje ovog atributa. Metoda "IspisiPodatke", u ovoj klasi je implementirana tako da ispisuje podatke o studentu u sljedećem obliku:

Student master studija <ime> <prezime> sa brojem indeksa <indeks>, završio bachelor studij godine <godina>, ima prosjek <prosjek>.

Klasa "Fakultet" treba da ima sljedeće elemente:

a) Privatne atribute koji predstavljaju broj upisanih studenata, maksimalno mogući broj studenata (ovaj atribut treba da bude konstantan), kao i pokazivač na dinamički alocirani niz pokazivača na objekte negog od tipova izvedenih iz tipa "Student", koji sadrže podatke u upisanim studentima.
b) Konstruktor sa jednim parametrom, koji vrši dinamičku alokaciju memorije, pri čemu parametar predstavlja maksimalan broj studenata koji se mogu upisati (i koji se ne smije koristiti za potrebe automatske konverzije cjelobrojnih vrijednosti u objekte tipa "Fakultet"), kao i odgovarajući destruktor koji treba da izvrši oslobađanje svih resursa koje su objekti tipa "Fakultet" alocirali tokom svog rada;
c) Dvije metode istog imena koje služe za upis novog studenta, od kojih jedna ima tri, a druga četiri parametra. Metoda sa tri parametra upisuje studenta bachelor studija, pri čemu su parametri broj indeksa, ime i prezime. Metoda sa četiri parametra upisuje studenta master studija, pri čemu četvrti dodatni parametar predstavlja godinu kada je student završio bachelor studij. U slučaju da se dostigne maksimalan broj studenata koji se mogu upisati, treba baciti izuzetak.
d) Metodu koja briše sve upisane studente.
e) Metodu koja vrši registraciju nove ocjene. Metoda kao parametre zahtijeva broj indeksa studenta kao i ocjenu koju je student dobio. Ova metoda treba da ažurira prosječnu ocjenu studenta na osnovu novounesene ocjene. U slučaju da student sa zadanim brojem indeksa ne postoji, treba baciti izuzetak.
f) Metodu koja poništava sve podatke o unesenim ocjenama za studenta sa brojem indeksa koji se zadaje kao parametar (tj. broj položenih ispita se vraća na nulu, a prosjek na 5). U slučaju da student sa zadanim brojem indeksa ne postoji, treba baciti izuzetak.
g) Metodu koja ispisuje podatke o studentu sa brojem indeksa koji se zadaje kao parametar. U slučaju da student sa zadanim brojem indeksa ne postoji, treba baciti izuzetak.
h) Metodu koja ispisuje imena i prezimena svih studenata čija je prosječna ocjena veća od vrijednosti koja se zadaje kao parametar, ili informaciju da takvih studenata nema.
i) Metodu koja ispisuje imena i prezimena svih studenata koji su položili više ispita od broja koji se zadaje kao parametar, ili informaciju da takvih studenata nema.
j) Metodu koja ispisuje spisak svih studenata sa pripadnim podacima (pozivom metode "IspisiPodakte" za svakog studenta), sortiran u opadajućem poretku po prosječnoj ocjeni (tj. student sa najvećom prosječnom ocjenom se ispisuje prvi). Kopiranje i međusobno dodjeljivanje primjeraka klase "Fakultet" treba zabraniti.

Napisane klase iskoristite u programu koji će korisniku nuditi izbor od nekoliko mogućih radnji, kao što su upis novog studenta, upis ocjene za postojećeg studenta, ispis studenata koji zadovoljavaju određene kriterije (prosjek veći od zadanog ili broj položenih ispita veći od zadanog), te ispis podataka o svim studentima sortiran po prosjeku. Pored navedenih opcija,

program treba da posjeduje opcije za čitanje podataka o studentima ili ocjenama iz teksualne datoteke. Tekstualna datoteka za opis podataka o studentima zove se "STUDENTI.TXT", a u svakom redu sadrži podatke za po jednog studenta, i to redom ime, prezime, broj indeksa i godinu završetka bachelor studija (ili 0 ako se radi o studentu bachelor studija). Na primjer:

```
Meho Mehić 11 0
Ibro Ibrić 12 1998
Vaso Vasić 13 0
Ivo Ivić 14 0
Marko Marković 15 2001
```

Datoteka sa podacima o ocjenama zove se "OCJENE.TXT", a prosto u svakom redu sadrži prvo broj indeksa studenta čija se ocjena upisuje, a zatim samu ocjenu. Na primjer:

```
11 8
11 7
11 9
12 7
12 8
13 6
13 19
14 7
14 8
14 8
15 9
```

Nakon unosa podataka iz ovih datoteka, kompletan sortirani spisak studenata trebao bi izgledati ovako:

Student master studija Marko Marković, sa brojem indeksa 15,
završio bachelor studij godine 2001, ima prosjek 9.
Student bachelor studija Meho Mehić, sa brojem indeksa 11,
ima prosjek 8.
Student bachelor studija Vaso Vasić, sa brojem indeksa 13,
ima prosjek 8.
Student bachelor studija Ivo Ivić, sa brojem indeksa 14,
ima prosjek 7.66.
Student master studija Ibro Ibrić sa brojem indeksa 12,
završio bachelor studij godine 1998, ima prosjek 7.5.

Program bi trebao da presreće sve izuzetke i druge probleme koji bi se eventualno mogli pojaviti, i da ispisuje odgovarajuće poruke upozorenja korisniku.

10.29 Prepravite program "ucenici.cpp" priložen uz Predavanje 7, tako da bude spriječen unos bilo kakvih neispravnih podataka (neispravne ocjene, neispravnog datuma, itd.). U slučaju da na bilo kojem mjestu korisnik unese neispravan podatak, potrebno je ispisati prikladno upozorenje i zatražiti ponovni unos istog podatka.

10.30 Prepravite program "ucenici_pok.cpp" priložen uz Predavanje 7, tako da bude spriječen unos bilo kakvih neispravnih podataka (neispravne ocjene, neispravnog

datuma, itd.). U slučaju da na bilo kojem mjestu korisnik unese neispravan podatak, potrebno je ispisati prikladno upozorenje i zatražiti ponovni unos istog podatka.

10.31 Definirajte i implementirajte klasu “Tacka” koja predstavlja tačku u ravni. Klasa bi trebala sadržavati sljedeći interfejs:

```
void Postavi(double x, double y);
void PostaviPolarno(double ro, double theta);
double DajX() const;
double DajY() const;
double DajRo() const;
double DajTheta() const;
void Transliraj(double delta_x, double delta_y);
void Rotiraj(double alpha);
void Rotiraj(double alpha, const Tacka &centar);
friend double Rastojanje(const Tacka &t1, const Tacka &t2);
```

Metoda “Postavi” postavlja informacije o položaju tačke na osnovu Dekartovih koordinata x i y koje se zadaju kao parametri, dok metoda “PostaviPolarno” radi sličnu stvar samo na osnovu polarnih koordinata ρ i θ. Metode “DajX”, “DajY”, “DajRo” i “DajTheta” vraćaju kao rezultat odgovarajuće Dekartove odnosno polarne koordinate tačke. Metoda “Transliraj” pomjera tačku 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” sa jednim parametrom rotira tačku za ugao α oko koordinatnog početka u smjeru suprotnom od kazaljke na satu, pri čemu se α zadaje kao parametar. Predviđena je i metoda “Rotiraj” sa dva parametra, koja radi slično kao i prethodna, samo što umjesto oko koordinatnog početka vrši rotaciju oko tačke koja se zadaje kao drugi parametar funkcije. Konačno, prijateljska funkcija “Rastojanje” vraća kao rezultat rastojanje između tačaka koje se zadaju kao parametri funkcije. Implementaciju klase zasnujte na privatnim atributima koji čuvaju Dekartove koordinate tačke. Napišite i kratki testni program u kojem ćete demonstrirati ispravan rad svih elemenata napisane klase. Posebno testirajte da li klasa ispravno radi sa tačkama u sva četiri kvadranta i tačkama koje leže na nekoj od koordinatnih osa (pri nepažljivoj implementaciji ovdje često nastaju greške).

Napomena: Funkcija “atan2” iz biblioteke “cmath” Vam može biti od velike koristi. Ako ne znate šta radi ova funkcija, informacije su lako dostupne na internetu (wikipedia recimo).

10.31 Proširite klasu “Tacka” iz Zadatka 3. iz Zadataka za samostalno vježbanje 8. 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). Prepravite i prateći testni program sa ciljem da testirate i napisane konstruktore.

10.32 Izmijenite implementaciju klase razvijene u prethodnom zadatku tako da se u klasi umjesto Dekartovih koordinata čuvaju polarne koordinate tačke. Uvjerite se da će

nakon obavljene izmjene implementacije testni program koji koristi napisanu klasu i dalje raditi posve identično, bez ikakvih izmjena.

10.33 Razmotrite kakve bi se izmjene mogle učiniti da klase razvijene u prethodna dva zadatka opisuju tačku u prostoru umjesto tačaka u ravni. Recimo, mogle bi se podržati cilindrične i sferne koordinate (najveći problem bi predstavljala rotacija, s obzirom da se tačka u prostoru ne može rotirati oko druge tačke, nego samo oko pravca).

10.34 Definirajte i implementirajte klasu "Vektor3d" u skladu sa deklaracijom i implementacijom prikazanom pred kraj Predavanja 8, s tim što ćete dodati još tri nove metode "PostaviX", "PostaviY" i "PostaviZ" koje treba da omoguće neovisnu izmjenu pojedinačnih koordinata vektora. Napišite i mali testni program u kojem ćete demonstrirati sve elemente razvijene klase.

10.35 Izmijenite implementaciju klase razvijene u prethodnom zadatku tako da se za čuvanje koordinata vektora umjesto tri privatna atributa "x", "y" i "z" koji su tipa realnih brojeva koristi jedan privatni atribut "koordinate" koji je tipa niza od tri realna elementa. Izmjenu treba izvesti tako da zaglavlja svih metoda unutar interfejsa klase ostanu neizmijenjena. Demonstrirajte da će testni program napisan u prethodnom zadatku raditi bez ikakvih izmjena sa ovako izmijenjenom klasom.

10.36 Proširite klasu "Vektor3d" iz Zadatka 1. sa Tutoriala 8. konstruktorom sa tri parametra koji obavlja istu funkciju kao i metoda "Postavi" sa tri parametra, kao i konstruktorom bez parametara koji kreira trodimenzionalni vektor čije su sve koordinate postavljene na 0. Prepravite i prateći testni program u skladu sa obavljenom izmjenom.

10.37 Proširite klasu "Vektor3d" iz Zadatka 2. sa Tutoriala 8. na isti način kao u prethodnom zadatku.

10.38 **Klasa Kvaternion**

U raznim oblastima matematike (posebno u algebri) i fizike (posebno u kvantnoj mehanici) veliku primjenu imaju tzv. kvaternioni, koji se mogu posmatrati kao poopćenja kompleksnih brojeva. Slično kao što se kompleksan broj z može posmatrati kao uređeni par realnih brojeva x i y, odnosno z = (x, y), tako se i kvaternion q može posmatrati kao uređena četvorka realnih brojeva x, y, z i t, odnosno q = (x, y, z, t). Sa kvaternionima su definirane sve četiri računske operacije, kao i za realne i kompleksne brojeve. Neka su q1 = (x1, y1, z1, t1) i q2 = (x2, y2, z2, t2) dva kvaterniona. Njihov zbir, razlika, proizvod i količnik definirani su izrazima

$q_1 + q_2 = (x_1 + x_2, y_1 + y_2, z_1 + z_2, t_1 + t_2)$
$q_1 - q_2 = (x_1 - x_2, y_1 - y_2, z_1 - z_2, t_1 - t_2)$
$q_1 \cdot q_2 = (x_1x_2 - y_1y_2 - z_1z_2 - t_1t_2, x_1y_2 + y_1z_2 + z_1t_2 - t_1z_2, x_1z_2 - y_1t_2 + z_1x_2 + t_1y_2, x_1t_2 + y_1z_2 - z_1y_2 + t_1x_2)$

$$q_1 / q_2 = \left(\frac{x_1x_2 + y_1y_2 + z_1z_2 + t_1t_2}{x_2^2 + y_2^2 + z_2^2 + t_2^2}, \frac{-x_1y_2 + y_1x_2 - z_1t_2 + t_1z_2}{x_2^2 + y_2^2 + z_2^2 + t_2^2}, \frac{-x_1z_2 + y_1t_2 + z_1x_2 + t_1y_2}{x_2^2 + y_2^2 + z_2^2 + t_2^2}, \frac{-x_1t_2 - y_1z_2 + z_1y_2 + t_1x_2}{x_2^2 + y_2^2 + z_2^2 + t_2^2} \right)$$

Dva kvaterniona q1 i q2 su jednaki ako i samo ako je x1= x2, y1= y2, z1= z2 i t1= t2. Slično kompleksnim brojevima, kvaternioni se ne mogu porediti po veličini.

Apsolutna vrijednost kvaterniona q = (x, y, z, t) definirana je kao |q| = (x2 + y2 + z2 + t2)1/2. Suprotni kvaternion –q dat je kao –q = (–x, –y, –z, –t). Kvaternioni specijalnog oblika (x, 0, 0, 0) mogu se poistovjetiti sa realnim brojem x, dok se kvaternioni specijalnog oblika (x, y, 0, 0) mogu poistovjetiti sa kompleksnim brojem (x, y), odnosno x + y i. Uvedemo li oznake j = (0, 0, 1, 0) i k = (0, 0, 0, 1), svaki kvaterinion q = (x, y, z, t) može se napisati u obliku x + y i + z j + t k, što poopštava algebarski način zapisivanja kompleksnih brojeva. Za "imaginarne jedinice" i, j i k vrijede pravila i2 = j2 = k2 = –1, kao i pravila i j = –j i = k, j k = –k j = i, k i = –i k = j, iz čega vidimo da su kvaternioni, na neki način, "zbir" skalara i trodimenzionalnog vektora.

Definirajte klasu za rad sa kvaternionima. Klasa bi trebala da sadrži tri konstruktora: konstruktor bez parametara, koji inicijalizira sve četiri komponente kvaterniona na nulu, zatim konstruktor sa jednim parametrom, koji inicijalizira prvu komponentu kvaterniona na zadani parametar, a preostale tri komponente na nulu, i konstruktor sa četiri parametra, koji inicijalizira sve četiri komponente kvaterniona na zadane vrijednosti. Konstruktor sa jednim parametrom treba da se koristi i za automatsku konverziju realnih brojeva u kvaternione. Dalje, klasa treba podržavati operatore "+", "–", "*" i "/ " za obavljanje četiri računske operacije, zatim kombinovane operatore "+=", "–=", "*=" i "/=" sa uobičajenim značenjem, relacione operatore "==" i "!=", unarni operator negacije "–", kao i operatore "<<" i ">>" za ispis kvaterniona na ekran i unos kvaterniona sa tastature. Kvaternione treba ispisivati kao uređene četvorke (x, y, z, t), bez obzira da li su im neke komponente nule ili nisu, dok unos kvaterniona sa tastature treba podržati bilo kao uređenu četvorku, bilo kao običan realan broj, koji se potom interpretira kao kvaternion. Također, treba podržati i prijateljsku funkciju "abs", koja računa apsolutnu vrijednost kvaterniona.

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

10.38 **Klasa Ugao i Kontenjerska klasa Uglovi**

Razvijte klasu koja omogućava rad sa uglovima izraženim u stepenima, minutama i sekundama, kao i kontejnersku klasu koja čuva podatke o kolekciji uglova (ovakva klasa bi, na primjer, mogla služiti za čuvanje podataka o uglovima nekog mnogougla). Interfejs klase koja omogućava rad sa uglovima treba da sadrži sljedeće elemente:

a) Konstruktor sa tri cjelobrojna parametra, koji redom predstavljaju stepene, minute i sekunde. Ovaj konstruktor treba da inicijalizira ugao na zadani iznos stepeni, minuta i sekundi. Treba dozvoliti mogućnost i da vrijednost parametara kojim se zadaju stepeni, minute i sekunde mogu biti i izvan tipičnog opsega, ali tada automatski treba preračunavati preljev (npr. 130 sekundi treba automatski tretirati kao 2 minute i 10 sekundi, 750 stepeni treba tretirati kao 30 stepeni jer 360 stepeni predstavlja pun krug, itd.). Stoga, na primjer, ugao zadan kao 130 stepeni, 90 minuta i 150 sekundi treba tretirati kao 131 stepen, 32 minuta i 30 sekundi. S druge strane, ukoliko je makar jedan od parametara negativan, konstruktor treba da baci izuzetak.

b) Metodu sa tri parametra koja obavlja načelno istu funkciju kao i konstruktor, a omogućava naknadno postavljanje ugla.

c) Metodu sa tri parametra koja očitava informacije o stepenima, minutama i sekundama koje tvore ugao i smješta ove informacije u odgovarajuće parametre metode.

d) Metodu bez parametara koja daje iznos ugla samo u stepenima (kao realni broj). Na primjer, ova metoda primijenjena na ugao od 40 stepeni i 30 minuta vraća 40.5 kao rezultat.
e) Preklopljeni unarni operator "–" koji primijenjen na neki ugao daje njegovu dopunu do punog kruga od 360 stepeni. Na primjer, ukoliko ugao "a" sadrži 110 stepeni, 20 minuta i 45 sekundi, ugao "–a" treba da sadrži 249 stepeni, 39 minuta i 15 sekundi.
f) Preklopljene binarne operatore "+" i "–" koji sabiraju odnosno oduzimaju dva ugla. Uputa: ukoliko ste konstruktor realizirali kako treba, realizacija ovih operatora trebala bi biti trivijalna.
g) Preklopljeni binarni operator "*" koji množi ugao sa cijelim brojem. Trebalo bi biti podržano kako množenje ugla sa brojem, tako i množenje broja sa uglom.
h) Preklopljene binarne operatore "+=", "–=" i "*=" koji osiguravaju da izrazi "X += Y", "X – = Y" i "X *= Y" uvijek imaju isto značenje kao i izrazi "X = X + Y", "X = X – Y" i "X = X * Y" kadgod ovi izrazi imaju smisla.
i) Preklopljen unarni operator "++" koji povećava ugao na koji je primijenjen za jednu ugaonu sekundu. Potrebno je podržati i prefiksnu i sufiksnu verziju ovog operatora.
j) Preklopljene relacione operatore "<",">", "==","!=","<=" i ">=" koji upoređuju dva ugla. k) Preklopljeni operator "<<" za ispis ugla na izlazni tok podataka koji vrši ispis u obliku <stepeni>d <minute>m <sekunde>s, na primjer 23d 15m 42s, kao i preklopljeni operator ">>" za čitanje ugla iz ulaznog toka podataka, u istom obliku u kojem se i vrši ispis.

Kontejnerska klasa koja čuva kolekciju uglova treba da sadrži pokazivač na prvi element dinamički alociranog niza pokazivača koji pokazuju na dinamički alocirane objekte koji sadrže stvarne podatke o pohranjenim uglovima. Ova klasa još sadrži informaciju o broju pohranjenih uglova, kao i informaciju o kapacitetu, odnosno maksimalnom broju uglova koji se mogu pohraniti (ova informacija se čuva u konstantnom atributu). Inače, svi atributi klase moraju biti u privatnoj sekciji klase. Interfejs klase treba da sadrži sljedeće elemente:
a) Konstruktor sa jednim cjelobrojnim parametrom, koji predstavlja maksimalni broj uglova koji se mogu pohraniti. Konstruktor treba da inicijalizira odgovarajuće atribute i da izvrši neophodnu alokaciju memorije. Pri tome je potrebno zabraniti da se ovaj konstruktor koristi za automatsku konverziju cjelobrojnog tipa u tip klase.
b) Destruktor, koji oslobađa svu memoriju koja je zauzeta tokom života primjeraka klase.
c) Konstruktor kopije, koji obezbjeđuje da se primjerci ove klase mogu bezbjedno kopirati, prenositi po vrijednosti u funkcije i vraćati kao rezultati iz funkcija.
d) Preklopljeni operator dodjele, koji omogućava bezbjedno međusobno dodjeljivanje primjeraka ove klase. Dodjelu treba podržati samo u slučaju da izvorni i odredišni objekat imaju isti kapacitet, a u suprotnom treba baciti izuzetak.
e) Metodu sa tri parametra, koja upisuje novi ugao u kolekciju. Parametri su broj stepeni, minuta i sekundi. Metoda treba da baci izuzetak u slučaju da je kolekcija popunjena.
f) Metodu koja briše sve pohranjene uglove.
g) Metodu koja ispisuje sve pohranjene uglove na ekran (svaki ugao u novom redu), koristeći operator za ispis definiran u klasi koja opisuje ugao.

h) Metode koje vraćaju najveći i najmanji ugao pohranjen u kolekciji.
i) Metode koje vraćaju ukupan broj uglova u kolekciji, kao i broj oštrih uglova (tj. uglova manjih od 90 stepeni).
j) Preklopljen operator indeksiranja "[]" koji omogućava da se pristupi i-tom pohranjenom uglu, gdje je indeks i naveden kao parametar u uglastim zagradama. U slučaju da je indeks izvan opsega, treba baciti izuzetak. Ovaj operator bi trebao omogućiti da se vraćeni ugao koristi sa lijeve strane znaka dodjeljivanja kad god tako nešto ima smisla.

Sve metode implementirajte izvan deklaracije klase, osim trivijalnih metoda koje trebate implementirati direktno unutar deklaracije klase. Sve metode koje su inspektori, obavezno deklarirajte kao takve. Obavezno napišite i mali testni program u kojem ćete testirati sve implementirane elemente napisanih klasa.

10.39 **Klasa Let i Kontenjerska klasa Letovi**

Potrebno je napraviti program koji vrši najavu letova na aerodromskom displeju. Program treba najavljivati sve letove u toku dana, kao i eventualna kašnjenja u polijetanjima. Za tu svrhu, u programu je potrebno razviti dvije klase nazvane "Let" i "Letovi". Klasa "Let" vodi evidenciju o jednom letu, dok klasa "Letovi" vodi evidenciju o svim letovima u toku dana. Klasa "Let" treba sadržavati sljedeće elemente:

a) Privatne atribute koji redom predstavljaju naziv odredišta, oznaku leta (npr. "JFK 327"), broj izlazne kapije (cijeli broj), vrijeme polijetanja (sati i minute), trajanje leta u minutama, kao i informacija o eventualnom kašnjenju (također u minutama). Naziv odredišta i oznaka leta čuvaju se kao nizovi znakova (ne kao dinamički stringovi), sa maksimalnim dužinama od 30 i 10 znakova respektivno. Ove maksimalne dužine treba definirati kao statičke konstantne atribute.
b) Konstruktor sa parametrima koji inicijalizira sve atribute klase na vrijednosti zadane parametrima konstruktora, osim atributa za eventualno kašnjenje koji se automatski inicijalizira na nulu. Konstruktor treba da baci izuzetak ukoliko bilo koji od parametara ima besmislene vrijednosti, uključujući i situaciju kada se za naziv odredišta ili za oznaku leta daju predugački nizovi znakova.
c) Metodu sa jednim parametrom koja postavlja informaciju o eventualnom kašnjenju na vrijednost zadanu parametrom.
d) Metodu kojom je moguće saznati da li odgovarajući let kasni ili ne. Metoda treba da vrati logičku vrijednost "`true`" u slučaju kašnjenja, a logičku vrijednost "`false`" u suprotnom slučaju.
e) Metodu koja vraća kao rezultat trajanje leta.
f) Metodu kojom je moguće saznati očekivano vrijeme polijetanja kada se uračuna iznos kašnjenja u odnosu na predviđeno vrijeme polijetanja (ovdje Vam može biti od koristi preklopljeni operator "+", koji ćete definirati kasnije). Metoda treba da smjesti sate i minute očekivanog vremena polijetanja u dva cjelobrojna parametra koji joj se prosljeđuju.
g) Metodu kojom je moguće saznati očekivano vrijeme slijetanja. Vrijede iste napomene kao i za prethodnu metodu..
h) Preklopljeni operator "<<" koji treba da podrži ispis objekata tipa "Let" na ekran. U slučaju da se radi o polijetanju bez kašnjenja, ispis bi trebao da izgleda poput sljedećeg:

JFK 327 New York 12:50 19:30 5

Ovi podaci predstavljaju redom oznaku leta, naziv odredišta, vrijeme polijetanja, očekivano vrijeme slijetanja i broj izlazne kapije. Predvidite širinu od 10 mjesta za ispis šifre leta, 30 mjesta za naizv odredišta, po 10 mjesta za vrijeme polijetanja i slijetanja, odnosno 8 mjesta za ispis broja izlazne kapije.
U slučaju da se radi o letu koji kasni, ispis bi trebao da izgleda poput sljedećeg:

IST 932 Istanbul 15:50 (Planirano 15:30, Kasni 20 min)

Oznaku leta, naziv odredišta i vrijeme polijetanja formatirajte kao i u prethodnom slučaju, a dopunske informacije pišite u produžetku iza vremena polijetanja.

i) Preklopljeni operator "+" koji omogućava da se objekat tipa "Let" sabere sa cijelim brojem, pri čemu se kao rezultat dobija novi objekat tipa "Let", u kojem je vrijeme polijetanja pomjereno unaprijed za iznos određen drugim sabirkom (računato u minutama), kao i preklopljeni operator "+=" koji obezbjeđuje da izraz oblika "X += Y" uvijek ima isto značenje kao i izraz "X = X + Y".

j) Preklopljeni operator "!" koji obavlja potpuno istu funkciju kao i metoda koja testira da li let kasni ili ne.

k) Prekopljeni operator "++" koji pomjera vrijeme polaska zapamćeno u objektu tipa "Let" za jedan sat unaprijed. Potrebno je podržati i prefiksnu i postfiksnu verziju ovog operatora.

l) Preklopljene relacione operatore "<" i ">" koji ispituju koji let nastupa ranije, odnosno kasnije. Operator "<" vraća logičku vrijednost "true" ukoliko polijetanje leta sa lijeve strane nastupa prije polijetanje leta sa desne strane, a u suprotnom vraća logičku vrijednost "false". Analogno vrijedi za operator ">". Prilikom upoređivanja treba uzeti u obzir i očekivano vrijeme kašnjenja, a ne samo planirano vrijeme polijetanja.

Klasa "Letovi" treba da sadrži sljedeće elemente:

a) Privatne atribute koji predstavljaju broj registriranih letova u toku dana, maksimalni broj letova koji se mogu registrirati (ovaj atribut mora biti konstantan), te pokazivač na dinamički alocirani niz (preciznije, na njegov prvi element) koji će čuvati informacije o svim registriranim letovima. Da biste izbjegli kasnije probleme, niz realizirajte kao niz pokazivača na letove.

b) Konstruktor sa jednim parametrom tipa cijeli broj, koji predstavlja maksimalni broj letova koji se mogu registrirati, i koji vrši alokaciju prostora za čuvanje podataka o letovima. Pri tome je potrebno zabraniti da se ovaj konstruktor koristi za automatsku konverziju cjelobrojnog tipa u tip "Letovi".

c) Destruktor, koji oslobađa svu memoriju koja je zauzeta tokom života objekta.

d) Konstruktor kopije koji omogućava bezbjedno kopiranje objekata tipa "Letovi", kao i preklopljeni operator dodjele koji omogućava bezbjednu međusobnu dodjelu objekata tipa "Letovi". Pri tome, dodjelu treba podržati samo ukoliko su izvorišni i odredišni objekat istih kapaciteta. U suprotnom treba baciti izuzetak.

e) Metodu sa jednim parametrom, koja registrira novi let. Parametar metode predstavlja pokazivač na let koji želimo registrirati. U slučaju da je dostignut maksimalan broj letovakoji se mogu registrirati, metoda treba da baci izuzetak.

f) Metodu koja daje kao rezultat ukupan broj letova, kao i broj letova koji kasne.

g) Metode koje daju kao rezultat prvi i posljednji let (tj. odgovarajući objekat tipa "Let") u toku dana (uključujući i eventualna kašnjenja).
h) Metodu koja daje prosječno trajanje svih pohranjenih letova.
i) Metodu koja ispisuje kompletan spisak svih letova, počev od zadanog vremena do kraja dana, sortiranu po očekivanim vremenima polazaka. U spisak treba dodati i prikladno zaglavlje, tako da bi ispis mogao izgledati poput sljedećeg:

Let Odredište Polazak Dolazak Kapija

JFK 327 New York 12:50 19:30 5
IST 932 Istanbul 15:50 (Planirano 15:30, Kasni 20 min)

Sve metode implementirajte izvan deklaracije klase, osim trivijalnih metoda koje trebate implementirati direktno unutar deklaracije klase. Sve metode koje su inspektori, obavezno deklarirajte kao takve.

10.40 Klasa Rječnik

Definirajte i implementirajte klasu "Rjecnik" koja čuva kolekciju riječi koje se mogu naknadno obrađivati (recimo, za potrebe nekog rječnika). Ova klasa čuva podatke u dinamički alociranom nizu objekata tipa "string", kojem se pristupa preko odgovarajućeg privatnog atributa. Pored ovog atributa (i eventualno drugih privatnih atributa neophodnih za normalno funkcioniranje klase), poznato je da klasa sadrži sljedeće elemente:

a) Konstruktor sa jednim parametrom, koji predstavlja maksimalan broj riječi koje se mogu registrirati. Ovaj konstruktor je odgovoran za dinamičku alokaciju memorije, i ne smije se koristiti za automatsku konverziju numeričkih podataka u objekte tipa "Rjecnik".
b) Destruktor, koji oslobadja resurse koji su zauzeti od strane ove klase.
c) Konstruktor kopije, koji obezbjedjuje siguran prenos objekata tipa "Rjecnik" kao parametara po vrijednosti u funkcije i njihovo vraćanje kao rezultata iz funkcije, kao i preoklopljeni operator dodjele "=" koji obezbjedjuje sigurnu dodjelu jednog objekta tipa "Rjecnik" drugom.
d) Metodu koja vrši upis nove riječi, pri čemu se riječ koja se registrira prenosi kao parametar metode. U slučju da se dostigne maksimalan broj podataka koje se mogu registrirati metoda treba da baci izuzetak. Izuzetak također treba baciti i u slučaju da string koji se zadaje kao parametar ne predstavlja riječ, nego recimo čitavu rečenicu, ili nešto slično. Kao kriterij koristite uvjet da riječ ne smije sadržavati razmake (tj. ukoliko u parametru ima razmaka, parametar je nelegalan).
e) Metodu koja daje broj registriranih riječi.
f) Metodu koja briše sve unesene riječi.
g) Metodu koja vraća kao rezultat broj riječi koje počinju slovom zadanim kao parametar (u slučaju da nema registriranih riječi, metoda treba da baci izuzetak);
h) Metodu koja ispisuje sve unesene (registrirane) riječi sortirane po abecedi, pri čemu se svaka riječ ispisuje u posebnom redu;
i) Preklopljeni operator "+" koji djeluje na sljedeći način: Ukoliko je "X" objekat tipa "Rjecnik", a "Y" objekat tipa "string", tada je "X + Y" novi objekat tipa "Rjecnik" u kojem je na svaku registriranu riječ nadovezan sadržaj stringa "Y". U svim ostalim slučajevima, značenje izraza "X + Y" nije definirano.
j) Preklopljeni operator "+=" čiji je cilj da značenje izraza "X += Y" uvijek bude identično značenju izraza "X = X + Y".

k) Preklopljeni operator “++” koji na sve registrirane riječi dodaje jednu zvjezdicu na kraj. Potrebno je podržati kako prefiksnu, tako i postfiksnu verziju ovog operatora.
l) Preklopljene relacione operatore “==” i “!=” koje ispituju da li su dva objekta tipa “Rjecnik” jednaka ili nisu. Dva objekta ovog tipa smatraju se jednakim ukoliko sadrže isti broj registriranih riječi, i ukoliko su sve registrirane riječi u jednom objektu identične odgovarajućim riječima u drugom objektu.

Sve metode implementirajte izvan deklaracije klase, osim trivijalnih metoda koje trebate implementirati direktno unutar deklaracije klase. Sve metode koje su inspektori, obavezno deklarirajte kao takve. Obavezno napišite i kratki testni program u kojem ćete demonstrirati sve napisane elemente ove klase.

10.39 **KlasaStudenti Kontenjerska klasa Odsjek**

Za potrebe računarske evidencije podataka o studentima nekog odsjeka, potrebno je razviti klase “Student” koja vodi evidenciju podataka o jednom studentu, i “Odsjek” koja vodi evidenciju o svim studentima jednog odsjeka. Klasa “Student” treba da sadrži sljedeće elemente:

a) Privatne atribute koji redom predstavljaju ime i prezime studenta (radi jednostavnosti, ime i prezime ce se čuvati u istom atributu), broju indeksa, i ocjene studenta iz pojedinih predmeta. S obzirom da nam neće biti potrebne ikakve ozbiljnije manipulacije sa imenima i ocjenama, ove atribute realizirajte kao obične nizove znakova i cijelih brojeva, a ne kao objekte tipa “string” i “vector”. Broj indeksa treba da bude konstantni atribut. Maksimalni broj znakova u imenu i prezimenu, kao i maksimalni broj ocjena deklarirajte kao statičke konstantne atribute inicijalizirane na 50 odnosno 30 respektivno, koje ćete koristiti svugdje umjesto fiksnih konstanti 50 i 30, čime olakšavate eventualne kasnije izmjene klase.
b) Konstruktor sa dva parametra, koji predstavljaju ime i prezime studenta, i broj indeksa. Konstruktor treba da sve ocjene studenta inicijalizira na 5. Takodjer, konstruktor treba da baci izuzetak u slučaju da je zadano ime (sa prezimenom) studenta duže od predviđenog broja znakova.
c) Metode koje redom vraćaju broj indeksa studenta, kao i ime i prezime studenta (preciznije, pokazivac na prvi znak imena i prezimena).
d) Metodu sa dva parametra, koja upisuje ocjenu studentu. Parametri su redni broj predmeta, kao i ocjena iz tog predmeta. Metoda treba da baci izuzetke u slučaju da je ocjena izvan opsega od 5 do 10, ili u slučaju da redni broj predmeta nije u opsegu od 1 do maksimalnog broja predmeta.
e) Metodu sa jednim parametrom koja vraća kao rezultat ocjenu iz predmeta ciji je redni broj
f) zadan kao parametar. Metoda treba da baci izuzetak u slučaju da redni broj predmeta nije u opsegu od 1 do maksimalnog broja predmeta.
g) Metodu koja vraća broj položenih ispita (tj. broj ocjena koje su različite od 5). Po
h) mogućnosti, za realizaciju koristite odgovarajuće funkcije iz biblioteke “algorithm”.
i) Metodu koja vraca prosječnu ocjenu studenta. Nepoloženi ispiti (tj. ispiti čija je trenutna ocjena 5) ne računaju se u prosjek. U slučaju da student nije položio niti jedan ispit, kao prosjek treba vratiti 5.

j) Preklopljeni operator “<<” koji treba da podrži ispis objekata tipa “Student” na ekran. Pri tome se prilikom ispisa treba da ispiše ime i prezime studenta, broj indeksa i prosjek (stil ispisa uredite po sopstvenoj volji).
k) Preklopljene relacione operatore “==” i “!=” koji ispituju da li su dva objekta tipa “Student” jednaka ili nisu. Smatrajte da su dva objekta tipa “Student” jednaka ukoliko im je isto ime i prezime i broj indeksa.
l) Preklopljene relacione operatore “<” i “>” koji ispituju koji je student bolji. Smatrajte da je bolji onaj student koji ima veći prosjek, a ako su im prosjeci jednaki, onda je bolji onaj koji je položio više ispita.

Klasa “Odsjek” treba da sadrži sljedeće elemente:

a) Privatne atribute koji predstavljaju broj studenata na odsjeku, maksimalni broj studenata koji odsjek moze primiti (ovaj atribut treba biti konstantan), te pokazivač na dinamički alocirani niz (preciznije, na njegov prvi element) preko kojeg će se pristupati podacima o studentima na odsjeku.
b) Konstruktor sa jednim parametrom tipa cijeli broj, koji predstavlja maksimalni broj
a) studenata koji se mogu upisati na odsjek. Konstruktor treba da inicijalizira odgovarajuće atribute i da izvrši neophodnu alokaciju dinamičkog niza. Pri tome je potrebno zabraniti da se ovaj konstruktor koristi za automatsku konverziju cjelobrojnog tipa u tip “Odsjek”.
b) Destruktor, koji oslobađa svu memoriju koja je zauzeta tokom života primjeraka klase.
c) Konstruktor kopije, koji obezbjeđuje da se objekti tipa “Odsjek” mogu bezbjedno kopirati, prenositi po vrijednosti u funkcije i vraćati kao rezultati iz funkcija. S druge strane, međusobno usobno dodjeljivanje objekata tipa “Odsjek” treba zabraniti.
d) Metodu sa dva parametra, koja upisuje novog studenta na odsjek. Parametri metode su ime i prezime studenta, i broj indeksa. Metoda treba da baci izuzetak u slučaju da već postoji student sa istim brojem indeksa, ili u slučaju da je odsjek popunjen.
e) Metodu koja ispisuje na ekran listu svih studenata, sortiranu po prosjeku u opadajući
f) poredak. Za ispis studenata, koristite operator “<<” definiran za klasu “Student”. Za
g) sortiranje, po mogućnosti koristite funkciju “sort” iz biblioteke “algorithm”.
h) Preklopljen operator indeksiranja “[]” koji omogućava da se studentima u klasi “Odsjek” pristupa po njihovom broju indeksa. Na primjer, ukoliko je “informatika” neki objekat klase “Odsjek”, tada “informatika[1234]” treba da predstavlja studenta sa indeksom 1234. Operator treba da vrati referencu na odgovarajućeg studenta, tako da budu omogućene konstrukcije poput “informatika[1234]. UpisiOcjenu(11, 7)” (u slučaju da ne vratimo referencu, upis ocjene ce biti izvršen nad kopijom objekta vraćenom kao rezultat operatorske funkcije, a ne nad samim objektom). U ovom primjeru je pretpostavljeno da se metoda za upis ocjene zove “UpisiOcjenu”. U slučaju da nema studenta sa brojem indeksa koji je zadan kao indeks u uglastim zagradama, operatorska funkcija treba da baci izuzetak.

Sve metode implementirajte izvan deklaracije klase, osim trivijalnih metoda koje trebate implementirati direktno unutar deklaracije klase. Sve metode koje su inspektori, obavezno deklarirajte kao takve. Napišite i mali testni program u kojem ćete demonstrirati sve elemente napisane klase "Odsjek".

10.40 **KlasaStudenti Kontenjerska klasa Odsjek**

Za potrebe vođenja evidencije o knjigama u nekoj biblioteci, u evidencijskom programu koriste se klase nazvane "Knjiga" i "Biblioteka". Vaš zadatak je da definirate i implementirate te klase. Klasa "Knjiga" opisuje podatke o jednoj knjizi, a sadrži sljedeće elemente:

a) Privatni evidencijski broj knjige (cijeli broj).
b) Privatne podatke o naslovu knjige, imenu pisca i žanr (svi ovi podaci su tipa "string").
c) Privatni podatak o godini izdanja knjige (cijeli broj).
d) Privatni podatak o članskom broju čitaoca kod koga se trenutno nalazi knjiga, ili 0 ukoliko knjiga trenutno nije na čitanju (tj. ukoliko je trenutno raspoloživa).
e) Privatni podatak koliko dugo dana je knjiga na čitanju (ukoliko knjiga trenutno nije na čitanju, vrijednost ovog podatka je nebitna).
f) Konstruktor, koji inicijalizira podatke o evidencijskom broju, naslovu knjige, imenu pisca, žanru i godini izdanja na vrijednosti zadane parametrima. Pri tome se podaci o čitaocu inicijaliziraju tako da signaliziraju da je knjiga slobodna.
g) Trivijalne pristupne metode, kojim se može saznati vrijednost svih privatnih podataka.
h) Metodu koja knjigu proglašava "zaduženom". Parametar metode je članski broj čitaoca koji zadužuje knjigu. Pri zaduženju, informacija o tome koliko dugo je knjiga na čitanju postavlja se na nulu.
i) Metodu bez parametara koja "razdužuje" knjigu.
j) Preklopljeni unarni operator "!" koji vraća logičku vrijednost "true" ukoliko je knjiga zadužena, a "false" ukoliko nije.
k) Preklopljeni unarni operator "++" koji povećava evidenciju o tome koliko je dugo knjiga na čitanju za jedinicu, ukoliko je knjiga zadužena. Ukoliko knjiga nije zadužena, ovaj operator ne radi ništa. Potrebno je podržati i prefiksnu i postfiksnu verziju ovog operatora.
l) Preklopljene relacione operatore "==" i "!=" koji testiraju da li su dvije knjige iste ili nisu. Smatra se da su dvije knjige iste ukoliko im se slažu naslov, ime pisca, žanr i godina izdanja.
m) Preklopljeni operator ispisa "<<" koji ispisuje na ekran podatke evidencijskom broju, naslovu, imenu pisca, žanru i godini izdanja knjige. Stil ispisa odredite po želji.

Klasa "Biblioteka" opisuje kolekciju knjiga, a sadrži sljedeće elemente:

a) Privatne informacije o broju knjiga u biblioteci, kao i o maksimalnom broju knjiga koje biblioteka može evidentirati. Pri tome, podatak o maksimalnom broju knjiga koje biblioteka može evidentirati treba biti izveden kao konstantni atribut.
b) Podatke o evidentiranim knjigama, izvedene kao dinamički niz pokazivača na dinamički alocirane objekte tipa "Knjiga", kojima se pristupa preko privatnog dvojnog pokazivača.

c) Konstruktor koji vrši dinamičku alokaciju memorije, pri čemu parametar predstavlja maksimalan broj knjiga koje biblioteka može evidentirati, kao i odgovarajući destruktor. Pri tome, treba zabraniti da se ovaj konstruktor koristi za automatsku konverziju cijelih brojeva u objekte tipa "Biblioteka". Pored toga, kopiranje i međusobno dodjeljivanje objekata tipa "Biblioteka" treba zabraniti.

d) Metodu koja evidentira novu knjigu u biblioteku. Metoda prima kao parametar pokazivač na knjigu koju treba evidentirati.

e) Metodu koja "zadužuje" knjigu, pri čemu se kao parametri zadaju evidencijski broj knjige, i članski broj čitaoca koji zadužuje knjigu. Ukoliko knjiga sa zadanim evidencijskim brojem ne postoji, treba baciti izuzetak.

f) Metodu koja "razdužuje" knjigu, pri čemu se kao parametri zadaje evidencijski broj knjige koja se razdužuje. Ukoliko knjiga sa zadanim evidencijskim brojem ne postoji, treba baciti

g) izuzetak.

h) Metodu koja vraća logičku vrijednost "true" ili "false", ovusno da li je knjiga čiji je evidencijski broj zadan kao parametar trenutno zadužena ili ne. Ukoliko knjiga sa zadanim evidencijskim brojem ne postoji, treba baciti izuzetak.

i) Metodu koja vraća članski broj čitaoca kod kojeg se nalazi knjiga čiji je evidencijski broj zadan kao parametar (odnosno nulu ukoliko knjiga nije zadužena). Ukoliko knjiga sa zadanim evidencijskim brojem ne postoji, treba baciti izuzetak.

j) Medode koje ispisuje podatke o svim slobodnim knjigama, svim zaduženim knjigama, i knjigama koje su na čitanju duže od n dana, pri čemu se n zadaje kao parametar.

k) Metodu koja sortira sve knjige u takav poredak da se knjige koje se su duže vremena zadužene nađu ispred onih koje su kraće vremena zadužene. Ukoliko dvije knjige imaju isto trajanje zaduženja, knjiga čiji naziv dolazi prije po abecedi treba da bude ispred knjige koja dolazi kasnije po abecedi.

l) Preklopljeni operator "[]" koji vraća referencu na knjigu čiji se evidencijski broj navodi unutar uglastih zagrada. Ukoliko knjiga sa zadanim evidencijskim brojem ne postoji, treba baciti izuzetak.

m) Preklopljeni unarni operator "++" koji povećava evidenciju o tome koliko je dugo knjiga na čitanju za jedinicu za sve zadužene knjige. Potrebno je podržati samo prefiksnu verziju ovog operatora.

Sve metode traženih klasa implementirajte izvan klase, osim trivijalnih metoda čija implementacija zahtijeva maksimalno 2–3 kratke naredbe, koje možete implementirati unutar klase. Također, sve metode koje su po prirodi inspektori, obavezno moraju biti deklarirane kao takve. Napišite i kratki testni program, u kojem ćete kreirati tri knjige, evidentirati ih u biblioteku, zadužiti jednu od njih, i ispisati podatke o slobodnim knjigama.

10.50 **Klasa Vodostaji**

Za praćenje dnevnog vodostaja neke rijeke u toku određenog vremenskog perioda hidrometeorološka stanica koristi računarski program u kojem je definirana i implementirana klasa nazvana "Vodostaji". Ova klasa omogućava čuvanje podataka o vodostaju posmatrane rijeke za u nekom od internih atributa klase koji predstavlja dinamički alocirani niz cijelih brojeva (vodostaj se izražava u centimetrima, zaokruženo na cijeli broj). Pored ovog atributa, poznato je da klasa sadrži sljedeće elemente:

a. Konstruktor sa jednim parametrom, koji vrši dinamičku alokaciju memorije i inicijalizaciju svih ostalih atributa, pri čemu parametar predstavlja maksimalan broj podataka o vodostaju koji se mogu registrirati (pri tome je potrebno spriječiti da se ovaj konstruktor koristi za automatsku konverziju cjelobrojnih podataka u objekte tipa "Vodostaji").
b. Destruktor koji oslobađa memoriju koju je alocirao konstruktor.
c. Metodu koja vrši registraciju novog podatka o vodostaju, pri čemu se podatak koja se registrira prenosi kao parametar metode (u slučaju da se dostigne maksimalan broj podataka koje se mogu registrirati, treba baciti izuzetak).
d. Metodu koja vraća kao rezultat prosječni vodostaj tokom posmatranog perioda.
e. Metodu koja briše sve unesene podatke.
f. Metode koje vraćaju broj dana u kojima je registriran vodostaj veći od vrijednosti koja je zadata kao parametar, kao i broj dana u kojima je registriran vodostaj manji od vrijednosti koja je zadata kao parametar (podrazumijeva se da svaki registrirani vodostaj odgovara jednom danu, tj. da se vodostaji registriraju jedanput dnevno).
g. Metodu koja ispisuje sve registrirane podatke sortirane u rastućem poretku (tj. najmanji vodostaj se ispisuje prvi), pri čemu se svaki podatak ispisuje u posebnom redu. Kopiranje i međusobno dodjeljivanje objekata tipa "Vodostaji" treba zabraniti.

Implementirajte klasu sa navedenim svojstvima. Sve neophodne atribute treba obavezno izvesti kao privatne članove klase, a sve metode implementirati izvan klase, osim metoda čija implementacija zahtijeva jednu ili dvije naredbe. Obavezno napišite i mali testni program u kojem će se testirati sve navedene metode.

10.51 **Klasa Student i klasa Fakultet**

Za potrebe nekog fakulteta neophodno je vršiti evidenciju o polaznicima (studentima) i njihovom uspjehu. Za tu svrhu fakultet koristi računarski program u kojem je definirana i implementirana klasa nazvana "Student", kao i klasa nazvana "Fakultet". Klasa "Student" sadrži sljedeće elemente:

a) Privatne atribute koje čuvaju podatke o imenu i prezimenu studenta (po maksimalno 20 znakova), broju indeksa (cijeli broj), broju položenih ispita (cijeli broj) i prosječnoj ocjeni (realan broj), i nikave druge atribute.
b) Konstruktor čiji su parametri ime i prezime studenta, kao i broj indeksa studenta, koji odgovarajuće atribute inicijalizira na vrijednosti zadane parametrima, dok se broj položenih ispita i prosječna ocjena inicijaliziraju respektivno na 0 i 5.
c) Trivijalne metode koje vraćaju vrijednosti svih odgovarajućih atributa.
d) Metodu sa jednim parametrom, koja služi za registraciju novog ispita, a koja povećava broj položenih ispita za jedinicu i ažurira prosječnu ocjenu u skladu sa ocjenom iz novopoloženog ispita, koja se zadaje kao parametar (u slučaju neispravne ocjene, treba baciti izuzetak).
e) Metodu koja ispisuje podatke (ime, prezime, broj položenih ispita, prosječna ocjena) o studentu.

Klasa "Fakultet" sadrži sljedeće elemente:

a) Privatne atribute koji predstavljaju broj upisanih studenata, maksimalno mogući broj studenata, kao i pokazivač na dinamički alocirani niz pokazivača na objekte tipa "Student", koji sadrže podatke u upisanim studentima.
b) Konstruktor sa jednim parametrom, koji vrši dinamičku alokaciju memorije, pri čemu parametar predstavlja maksimalan broj studenata koji se mogu upisati (i koji se ne smije

koristiti za potrebe automatske konverzije cjelobrojnih vrijednosti u objekte tipa "Student"), kao i odgovarajući destruktor.

c) Konstruktor kopije koji će osigurati da se objekti tipa "Fakultet" mogu bezbjedno prenositi kao parametri po vrijednosti, koristiti za inicijalizaciju drugih objekata istog tipa i vraćati kao rezultati iz funkcija.
d) Preklopljeni operator dodjele, koji garantira sigurno dodjeljivanje jednog objekta tipa "Fakultet" drugom objektu istog tipa.
e) Metodu koja vrši upis novog studenta, pri čemu se kao parametri metode prenose broj indeksa studenta, kao i njegovo ime i prezime studenta (u slučaju da se dostigne maksimalan broj studenata koji se mogu upisati, treba baciti izuzetak).
f) Metodu koja briše sve upisane studente.
g) Metodu koja vrši registraciju nove ocjene. Metoda kao parametre zahtijeva broj indeksa studenta kao i ocjenu koju je student dobio. Ova metoda treba da ažurira prosječnu ocjenu studenta na osnovu novounesene ocjene.
h) Metodu koja ispisuje podatke (ime, prezime, broj položenih ispita, prosječna ocjena) o studentu sa brojem indeksa koji se zadaje kao parametar.
i) Metodu koja ispisuje imena i prezimena svih studenata čija je prosječna ocjena veća od vrijednosti koja se zadaje kao parametar.
j) Metodu koja ispisuje imena i prezimena svih studenata koji su položili više ispita od broja koji se zadaje kao parametar.
k) Metodu koja ispisuje sve upisane studente sortirane u opadajućem poretku po prosječnoj ocjeni (tj. student sa najvećom prosječnom ocjenom se ispisuje prvi), pri čemu se podaci o svakom studentu ispisuju u posebnom redu.

Implementirajte klase "Student" i "Fakultet" sa navedenim svojstvima. Sve metode implementirajte izvan tijela klase, osim metoda čija implementacija zahtijeva jednu ili dvije naredbe. Obavezno napišite i mali testni program u kojem će se testirati sve navedene metode.

10.52 **Klasa Zahtjev**

Za potrebe obrade zahtjeva koji dolaze na studentsku službu od strane studenata, neki računarski program definira i implementira klase nazvane "Zahtjev" i "Zahtjevi". Klasa "Zahtjev" opisuje jedan zahtjev, dok klasa "Zahtjevi" predstavlja kolekciju zahtjeva, odnosno objekata tipa "Zahtjev".

Klasa "Zahtjev" kao atribute sadrži ime i prezime podnosioca zahtjeva, broj indeksa, tekst zahtjeva, kao i atribute koji predstavljaju dan, mjesec i godinu kada je zahtjev podnesen. Pored ovih atributa, klasa "Zahtjev" sadrži još samo konstruktor, koji omogućava propisnu inicijalizaciju svih atributa klase na vrijednosti zadane parametrima. Pri tome, konstruktor testora da li je zadan legalan datum (ukoliko nije, potrebno je baciti izuzetak). Klasa "Zahtjevi" predstavlja kolekciju zahtjeva, izvedenu kao dinamički niz pokazivača na objekte tipa "Zahtjev". Ova klasa sadrži konstruktor sa jednim parametrom, koji vrši odgovarajuću dinamičku alokaciju memorije, pri čemu parametar predstavlja maksimalan broj zahtjeva koji se mogu pohraniti. Pored toga, klasa sadrži i odgovarajući destruktor, koji oslobađa memoriju koju je objekat tipa "Zahtjevi" zauzeo tokom svog života. Konstruktor kopije i preklopljeni operator dodjele nisu predviđeni, ali je kopiranje i međusobno dodjeljivanjeprimjeraka klase "Zahtjevi" potrebno zabraniti.

Klasa “Zahtjevi” posjeduje dvije metode. Prva metoda vrši kreiranje i dodavanje novog zahtjeva, i ona ima iste parametre kao i konstruktor klase “Zahtjev”. Ova metoda kreira novi zahtjev (u skladu sa navedenim parametrima) i smješta ga u kolekciju. Druga metoda nema parametara, i ona vrši ispis podataka o prvom primljenom zahtjevu na ekran (stil ispisa odaberite sami). Pored toga, ova metoda vrši brisanje obrađenog zahtjeva iz kolekcije, tako da će sljedeći poziv ove metode ispisati sljedeći zahtjev po redu, i tako dalje, sve dok se ne istroše svi zahtjevi. U slučaju da se ova metoda pozove kada je kolekcija prazna, metoda treba da baci izuzetak. Pored toga, klasa “Zahtjevi” posjeduje i metodu bez parametara koja daje kao rezultat logičku vrijednost “true” ako i samo ako objekat nad kojim je primijenjena predstavlja praznu kolekciju, a u suprotnom daje kao rezultat logičku vrijednost “false”.

10.52 **Klase Zaposlenik i Fakultet**

Napisane klase testirajte u testnom programu koji kreira nekoliko zahtjeva prema podacima unesenim sa tastature, a zatim ispisuje sve unijete zahtjeve u redoslijedu pristizanja. Radi učestalih nedolazaka na posao i izbjegavanja radnih obaveza, rukovodsto Elektrotehničkog fakulteta u Sarajevu odlučilo je da uvede novi sistem evidentiranja dolaska i odlaska zaposlenika na posao. Pošto je novi sistem registriranja veoma zahtjevan, zabrinutoj administrativnoj službi (čitaj: teti Nihadi) je potreban računarski program koji će voditi evidenciju o broju sati koji su pojedini zaposlenici proveli na poslu u toku radne sedmice, i na osnovu toga obračunati njihovu sedmičnu platu i naknadu za topli obrok.

Fakultet radi od ponedjeljka do petka. Normalno radno vrijeme je od 8:00 do 16:00, ali se pojedinim zaposlenicima, u zavisnosti od složenosti posla koji obavljaju (recimo, nastavnom osoblju) može dopustiti manje sedmično opterećenje, bez umanjenja plate za tu sedmicu. Stoga, za svakog zaposlenika postoje individualni podaci o njegovoj nominalnoj sedmičnoj plati kao i minimalnom sedmičnom opterećenju koje zaposlenik mora provesti na poslu da bi zaslužio puni iznos plate za tu sedmicu (ovo opterećenje ne smije biti veće od Zakonom predviđenog sedmičnog opterećenja od 40 sati). Zaposlenicima koji provedu manji broj sati u toku jedne sedmice od minimalnog sedmičnog opterećenja, odbija se za svaki puni sat ispod minimalne norme iznos jedne satnice, koja se dobija kao iznos sedmične plate podijeljen sa tipičnim sedmičnim opterećenjem od 40 sati. Pored toga, svi zaposlenici imaju pravo na dnevnu naknadu za topli obrok, koja je fiksna za sve zaposlenike (tj. ne zavisi od njihove plate). Zaposlenik stiče pravo na dnevnu naknadu za topli obrok za one dane u toku kojih je na poslu proveo veći broj sati od predviđene kvote, koja je također fiksna za sve zaposlenike. Na kraju, predviđena je i mogućnost dodatnog plaćanja prekovremenog rada. Ukoliko neki zaposlenik provede u toku jednog dana više od 8 sati na poslu, za svaki sat prekovremenog rada u toku tog dana isplaćuje mu se naknada u iznosu od 1.5 satnice za svaki sat prekovremenog rada. Treba napomenuti da se fakultet zaključava u 20:00, tako da do tog vremena svi zaposlenici moraju napustiti fakultet.

Za opisanu svrhu potrebno je razviti program u koji će se zasnivati na dvije klase nazvane “Zaposlenik” i “Fakultet”. Klasa “Zaposlenik” čuva podatke o jednom zaposleniku, dok je “Fakultet” kontejnerska klasa koja čuva podatke o kolekciji objekata tipa “Zaposlenik”. Ona treba da sadrži sljedeće elemente:

a) Privatne atribute koji predstavljaju broj zaposlenika na fakultetu, maksimalan broj zaposlenika koji fakultet može zaposliti, dnevnu naknadu za topli obrok i minimalnu satnicu neophodnu za ostvarivanje prava na topli obrok (s obzirom da su ovi podaci isti za sve zaposlenike, njih nije neophodno čuvati za svakog zaposlenika posebno), kao i

atribut koji služi za pristup dinamički alociranom nizu koji čuva pokazivače na objekte tipa “Zaposlenik”.

b) Konstruktor sa tri parametra, koji predstavljaju maksimalnan broj zaposlenika na fakultetu, dnevnu naknadu za topli obroj i minimalnu satnicu neophodnu za ostvarivanje prava na topli obrok. Ovaj konstruktor treba inicijalizirati odgovarajuće atribute i izvršiti neophodnu alokaciju prostora za čuvanje podataka o zaposlenicima. Broj zaposlenika se inicijalizira na 0, s obzirom da će se informacije o zaposlenicima dodavati naknadno.

c) Destruktor, koji oslobađa sve resurse koji je objekat tipa “Fakultet” zauzeo tokom svog postojanja.

d) Konstruktor kopije i preklopljeni operator dodjele koji omogućavaju sigurno kopiranje i međusobno dodjeljivanje objekata tipa “Fakultet” (što se tiče preklopljenog operatora dodjele, najbolje je prvo uništiti sadržaj odredišnog objekta a zatim iskopirati izvorni u odredišni objekat, s obzirom da bi svako drugo rješenje koje bi eventualno iskoristilo već postojeći prostor u odredišnom objektu bilo dosta komplicirano).

e) Metodu koja kreira i evidentira novog zaposlenika. Parametri ove metode su evidencijski platni broj zaposlenika (taj broj jednoznačno određuje zaposlenika), njegovo ime i prezime, iznos njegove sedmične plate, kao i minimalno sedmično opterećenje koje zaposlenik treba provesti na poslu da bi zaradio puni iznos plate (ovi podaci se mogu razlikovati od jednog do drugog zaposlenika). Metoda treba da baci izuzetak ukoliko već postoji zaposlenik sa zadanim platnim brojem, ili ukoliko je dostignut maksimalni broj zaposlenika koji fakultet može primiti.

f) Metodu koja vrši evidenciju dolaska zaposlenika na posao. Parametri metode su platni broj zaposlenika, kao i sat i minuta kada je zaposlenik došao na posao. Metoda treba da baci izuzetak ukoliko zaposlenik sa zadanim platnim brojem ne postoji, ukoliko vrijeme dolaska nije u intervalu od 8:00 do 16:00, ili ukoliko je njegov dolazak za taj dan već evidentiran (novo evidentiranje je moguće tek nakon evidencije odlaska).

g) Metodu koja vrši evidenciju odlaska zaposlenika sa posla. Parametri su isti kao kod prethodne metode. Metoda treba da baci izuzetak ukoliko zaposlenik sa zadanim platnim brojem ne postoji, ukoliko vrijeme odlaska nije u intervalu od 8:00 do 20:00, ukoliko je vrijeme odlaska manje od vremena dolaska za taj dan, ili ukoliko je odlazak zaposlenika za taj dan već evidentiran (novo odjavljivanje je moguće tek nakon nove evidencije dolaska).

h) Metodu koja vrši ispis podataka o zaposleniku. Parametar metode je platni broj zaposlenika, a metoda treba da ispiše ime zaposlenika, platni broj, broj normalno provedenih radnih sati (od početka evidentiranja), broj prekovremenih radnih sati (također od početka evidentiranja), platu za normalne radne sate (u skladu sa trenutnim stanjem, koje će biti kompletno tek na kraju sedmice), dodatak za prekovremeni rad, naknadu za topli obrok i ukupan iznos za isplatu (sve u skladu sa trenutnim stanjem).

i) Metodu koja vrši ispis platne liste za sve zaposlenike na fakultetu (ova metoda bi se tipično trebala pozivati na kraju svake radne sedmice, jer će tek tada svi podaci biti kompletirani). Metoda treba za svakog zaposlenika ispisati iste podatke kao u predhodnoj metodi.

j) Metodu koja vrši zaključivanje radne sedmice. Nakon poziva ove metode, sve informacije o broju provedenih radnih sati za sve zaposlenike vraćaju se na nulu, čime je sistem spreman za novo evidentiranje u toku sljedeće sedmice.

U gore navedenim specifikacijama nije ništa rečeno o tome šta treba sadržavati klasa “Zaposlenik”. Kako je ona samo pomoćna klasa neophodna za funkcioniranje klase “Fakultet” (koja se na nju oslanja), nju možete dizajnirati kako god želite, pod uvjetom da to obezbijedi ispravno funkcioniranje klase “Fakultet” u skladu sa gore postavljenim zahtjevima.

Sve metode implementirajte izvan tijela klase, osim metoda čija implementacija zahtijeva jednu ili dvije naredbe. Obavezno napišite i mali testni program u kojem će se testirati sve navedene metode. Najbolje bi bilo da program manipulira sa podacima koji se unose sa tastature.

10.52 **Sinusoida**

U raznim oblastima nauke i tehnike često se javlja potreba rada sa promjenljivim veličinama koje se mijenjaju po sinusoidalnom zakonu oblika A sin (ω t + ϕ), 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 sa istom frekvencijom, tj. vrijedi A1 sin (ω t + ϕ1) ± A2 sin (ω t + ϕ2) = A3 sin (ω t + ϕ3), pri čemu vrijednosti A3 i ϕ3 zavise na neki način od veličina A1, A2, ϕ1 i ϕ2. Ova zavisnost može se izvesti uz malu pomoć trigonometrije, ali može i još jednostavnije koristeći tzv. fazorski prikaz sinusoidalnih veličina, iz kojeg direktno slijedi da je A3 e iϕ3 = A1 e iϕ1 ± A2 e iϕ2. 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:

a) 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 ukoliko se faza poveća ili umanji za iznos π.
b) Trivijalne pristupne metode koje vraćaju vrijednosti amplitude, frekvencije odnosno faze.
c) Metode koje omogućavaju naknadnu 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.
d) Preklopljene 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 se pretpostavlja da obje sinusoidalne veličine imaju istu frekvenciju (u suprotnom treba baciti izuzetak). Rezultati moraju biti normirani. Za realizaciju Vam mogu biti od velike koristi tipovi i funkcije iz biblioteke "complex" " (mada niste obavezni da ih koristite).
e) Preklopljene 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.
f) Preklopljene operatore "+=", "–=", "*=" i "/=" koji omogućavaju da se kad god izrazi poput "X = X + Y", "X = X – Y", "X = X * Y" i "X = X / Y" isti izrazi mogu skraćeno pisati kao "X += Y", "X –= Y", "X *= Y" i "X /= Y".
g) 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.

10.52 **Klasa Perpleksni**

Svima koji se bave tehničkim naukama dobro su poznati komplesni brojevi i njihova svojstva, koji se mogu formalno posmatrati kao uređeni parovi realnih brojeva oblika (a, b) gdje su a i b realni brojevi. Međutim, iako su kompleksni brojevi najpoznatije i najprimjenljivije tvorevine takve vrste, one nisu i jedini korisni matematički objekti koji se modeliraju kao parovi realnih brojeva. Tako su u modernoj fizici, a posebno u specijalnoj teoriji relativnosti (uključujući i špekulativne teorije koje predviđaju postojanje i opisuju svojstva čestica koje se kreću nadsvjetlosnom brzinom) veliku primjenu našli bliski srodnici komplesnih brojeva poznati kao perpleksni brojevi (susreću se i nazivi hiperbolički brojevi, kontrakompleksni brojevi i još neki drugi). Oni posjeduju prividno bizarno svojstvo da im apsolutna vrijednost može biti i negativna. Intuitivno, perpleksni brojevi su brojevi oblika x + y h, gdje su x i y realni brojevi, dok je h tzv. hiperbolička (kontraimaginarna, halucinogena) jedinica, koja je blizak srodnik imaginarne jedinice i, ali za nju vrijedi h2 = 1 (dok je i2 = –1, kao što znamo). Pri tome je bitno naglasiti da iako je h2 = 1, vrijedi h ≠ 1 i h ≠ –1 (postojanje ovakvih bizarnih objekata može se lako opravdati pomoću linearne algebre). Interesantno je da se h od 1 i –1 razlikuje i po bizarnom svojstvu | h | = –1. Formalno, perpleksni brojevi se mogu također predstaviti kao parovi realnih brojeva, koje ćemo označiti sa (x | y) da bismo ih razlikovali od kompleksnih brojeva (x, y), i za koje vrijede sljedeća pravila računanja:

$(x_1 | y_1) \pm (x_2 | y_2) = (x_1 \pm x_2 | y_1 \pm y_2)$

$(x_1 | y_1) \cdot (x_2 | y_2) = (x_1 x_2 + y_1 y_2 | x_1 y_2 + x_2 y_1)$

$$(x_1 | y_1) / (x_2 | y_2) = \left(\frac{x_1 x_2 - y_1 y_2}{x_2^{\,2} - x_2^{\,2}}, \frac{y_1 x_2 - x_1 y_2}{x_2^{\,2} - x_1^{\,2}} \right)$$

Može se primijetiti upadljiva sličnost sa računom sa kompleksnim brojevima, uz male izmjene u predznacima (koje su posljedica činjenice da je h2 jednako 1 a ne –1). Uz ovakav formalizam, imamo h = (0 | 1). Broj x se naziva realni, a broj y hiperbolni (kontraimaginarni, halucinogeni) dio perpleksnog broja (x | y). Negacija perpleksnog broja (x | y) data je kao (–x | –y), dok je njegova konjugacija data kao (x | –y). Apsolutna vrijednost perpleksnog broja (x | y) data je kao

$$| (x | y) | = \sqrt{x^2 - y^2} \;\; za \;\; x^2 \geq y^2 \,, \; | (x | y) | = -\sqrt{y^2 - x^2} \;\; za \;\; x^2 < y^2$$

Perpleksni broj (x | 0) može se poistovijetiti sa realnim brojem x. Dva perpleksna broja (x1| y1) i (x2 | y2) jednaki su ako i samo ako je x1 = x2 i y1 = y2. Poredak perpleksnih brojeva se ne definira. Definirajte klasu za rad sa perpleksnim brojevima. Klasa bi trebala da sadrži konstruktor koji kreira perpleksni broj na osnovu zadanog realnog i hiperbolnog dijela koji se zadaju kao parametri. Oba parametra trebaju imati podrazumijevanu vrijednost 0, tako da će se ovaj konstruktor moći koristiti i kao konstruktor bez parametara odnosno sa jednim parametrom. Klasa treba podržavati dvije pristupne metode bez parametara, kojima se pristupa realnom i hiperbolnom dijelu perpleksnog broja. Dalje, treba podržavati operatore “+”, “–”, “*” i “/” za obavljanje četiri osnovne računske operacije, zatim kombinovane operatore “+=”, “–=”, “*=” i “/=” sa uobičajenim značenjem, relacione operatore “==” i “!=”, unarni operator negacije “–”, te unarni operator “++” koji povećava realni dio perpleksnog

broja za jedinicu, dok hiperbolni dio ostaje isti (potrebno je podržati i prefiksnu i postfiksnu verziju ovog operatora). Također je potrebno podržati i operatore "<<" i ">>" za ispis perpleksnih brojeva na izlazni tok i njihov unos sa ulaznog toka. Perpleksne brojeve treba ispisivati kao parove oblika (x | y) bez obzira da li su im neke komponente nule ili nisu, dok unos perpleksnih brojeva sa ulaznog toka treba podržati bilo kao par oblika (x | y), bilo kao običan realan broj, koji se potom interpretira kao perpleksni broj. Konačno, treba implementirati i četiri obične funkcije (ne članice) koje kao rezultat daju respektivno apsolutnu vrijednost, konjugaciju, realni i hiperbolni dio perpleksnog broja (neka Vas ne zbunjuje što postoje i metode koje daju realni i hiperbolni dio perpleksnog broja – ovako se omogućava veća sintaksna šarolikost u primjeni).

Rješenje:

```
class Perpleksni {
      double re, hip;
      public:
      Perpleksni(double realni = 0, double hiperbolni = 0)
      re(realni), hip(hiperbolni) {}
      double DajRealni() const { return re; }
      double DajHiperbolni() const { return hip; }
      friend Perpleksni operator +(const Perpleksni &p1, const
      Perpleksni &p2) {return Perpleksni(p1.re + p2.re, p1.hip +
      p2.hip);
      }
      friend Perpleksni operator -(const Perpleksni &p1, const
      Perpleksni &p2) {return Perpleksni(p1.re - p2.re, p1.hip -
      p2.hip);
      }
      friend Perpleksni operator *(const Perpleksni &p1, const
      Perpleksni &p2) {return Perpleksni(p1.re * p2.re + p1.hip *
      p2.hip,p1.re * p2.hip + p2.re * p1.hip);
      }
      friend Perpleksni operator /(const Perpleksni &p1, const
      Perpleksni &p2) { double nazivnik(p2.re * p2.re - p2.hip *
      p2.hip); return Perpleksni((p1.re * p2.re + p1.hip * p2.hip) /
      nazivnik,(p2.re * p1.hip - p1.re * p2.hip) / nazivnik);
      }
      Perpleksni &operator +=(const Perpleksni &p) { return *this =
      *this + p; }
      Perpleksni &operator -=(const Perpleksni &p) { return *this =
      *this - p; }
      Perpleksni &operator *=(const Perpleksni &p) { return *this =
      *this * p; }
      Perpleksni &operator /=(const Perpleksni &p) { return *this =
      *this / p; }
      friend bool operator ==(const Perpleksni &p1, const Perpleksni
      &p2) {
      return p1.re == p2.re && p1.hip == p2.hip;
      }
      friend bool operator !=(const Perpleksni &p1, const Perpleksni
      &p2) {
      return !(p1 == p2);
      }
```

```
    Perpleksni operator -() { return Perpleksni(-re, -hip); }
    Perpleksni &operator ++() { re++; return *this; }
    Perpleksni operator ++(int) { Perpleksni p(*this); re++; return
    p; }
    friend ostream &operator <<(ostream &cout, const Perpleksni
    &p) {return cout << "(" << p.re << "|" << p.hip << ")";
    }
    friend istream &operator >>(istream &cin, Perpleksni &p);
    friend double ApsolutnaVrijednost(const Perpleksni &p);
    friend Perpleksni Konjugirani(const Perpleksni &p) {
    return Perpleksni(p.re, -p.hip);
    }
    friend double RealniDio(const Perpleksni &p) { return p.re; }
    friend double HiperbolniDio(const Perpleksni &p) { return
    p.hip; }
};

double ApsolutnaVrijednost(const Perpleksni &p) {
    double norma(p.re * p.re - p.hip * p.hip);
    if(norma >= 0) return sqrt(norma);
    else return -sqrt(-norma);
}

istream &operator >>(istream &cin, Perpleksni &p) {
    if(cin.peek() != '(') {
    cin >> p.re; p.hip = 0;
    }
    else {
    char znak;
    cin >> znak;
    if(znak != '(') cin.setstate(ios::failbit);
    cin >> p.re >> znak;
    if(znak != '|') cin.setstate(ios::failbit);
    cin >> p.hip >> znak;
    if(znak != ')') cin.setstate(ios::failbit);
    }
    return cin;
}
```

10.52 **Klasa Telefonski imenik**

Uprava neke radne organizacije je odlučila da napravi interni telefonski imenik svojih uposlenika. Podaci o jednom uposleniku čuvaju se u strukturi koja kao svoje atribute sadrži ime uposlenika zajedno sa prezimenom (tipa niza od max. 20 znakova), naziv odjeljenja (tipa niza od max. 10 znakova) i broj telefona (cjelobrojnog tipa). Potrebno je razviti kontejnersku klasu koja će čuvati kolekciju podataka o uposlenicima i njihovim telefonskim brojevima. Ovi podaci će se alocirati dinamički, a pristupaće im se preko dinamički alociranog niza pokazivača koji pokazuju na ove podatke. Interfejs klase treba sadržavati sljedeće elemente:

a) Konstruktor sa jednim parametrom koji vrši neophodnu dinamičku alokaciju, pri čemu parametar predstavlja maksimalni broj uposlenika koji se mogu evidentirati. Pri tome se ovaj konstruktor ne smije koristiti za automatsku konverziju cjelobrojnih podataka u tip ove klase. b) Destruktor, koji oslobađa sve resurse koje su primjerci ove klase zauzeli tokom svog života.

b) Konstruktor kopije i preklopljeni operator dodjele koji omogućavaju da se primjerci ove klase mogu bezbjedno kopirati i međusobno dodjeljivati.
c) Metodu koja dodaje u kolekciju podatke o novom uposleniku, pri čemu su parametri metodepodaci o uposleniku (ime i prezime, odjeljenje i broj telefona). U slučaju da su podaci o imenu i prezimenu ili odjeljenju predugi, ili ukoliko se dostigne maksimalni broj uposlenika koji se mogu evidentirati, treba baciti izuzetak.
d) Metodu koja ispisuju podatke o uposleniku čije se ime i prezime zadaje kao parametar i metodu koja ispisuje podatke o uposleniku čiji se broj telefona zadaje kao parametar.
e) Metodu koja ispisuje telefonski imenik za sve uposlenike koji se nalaze u odjeljenju koje se zadaje kao parametar, zatim metodu koja ispisuje telefonski imenik za sve uposlenike čije ime počinje slovom koje se zadaje kao parametar, te metodu koja ispisuje čitav telefonski imenik.
f) Metodu koje sortira imenik po abecednom poretku uposlenika (koristiti funkciju "sort").
g) Preklopljeni operator "[]" koji vraća referencu na broj telefona uposlenika čije se ime zadaje unutar uglastih zagrada (referenca se vraća sa ciljem da se omogući izmjena broja). U slučaju da se ovaj operator primijeni na konstantni objekat, umjesto reference treba vratiti kopiju broja.
h) Metodu koja snima čitav telefonski imenik u binarnu datoteku čije se ime zadaje kao parametar, te konstruktor sa jednim parametrom koji prilikom kreiranja imenika obnavlja njegov sadržaj iz binarne datoteke čije se ime zadaje kao parametar konstruktora.

Rješenje:

Napomena: U prikazanom rješenju, struktura "`Uposlenik`" umjesto da se deklarira izvan klase na globalnom nivou, deklarirana je lokalno u privatnom dijelu klase, tako da joj samo metode klase "`Imenik`" mogu pristupiti, odnosno ostatak programa "ne zna" za postojanje ove strukture. Mada se to nije očekivalo od studenta da uradi, ovdje je prikazano čisto kao ilustracija da se takve tehnike često koriste.

```
class Imenik {
      struct Uposlenik {
            char ime[20], odjeljenje[10];
            int telefon;
      };
      int kapacitet, broj_uposlenika;
      Uposlenik **uposlenici;
      static void IspisiPodatke(Uposlenik *u);
      // Pomoćna funkcija za     ispis
      static bool KriterijSortiranja(Uposlenik *u1, Uposlenik *u2) {
      return strcmp(u1->ime, u2->ime) < 0;
      }
      public:
      explicit Imenik(int kapacitet) : kapacitet(kapacitet),
      broj_uposlenika(0),
      uposlenici(new Uposlenik*[kapacitet]) {}
      ~Imenik();
      Imenik(const Imenik &im);
      Imenik &operator=(const Imenik &im);
      void DodajUposlenika(const char ime[], const char odjeljenje[],
      int telefon);
      void IspisiUposlenikaSaImenom(const char ime[]) const;
      void IspisiUposlenikaSaTelefonom(int telefon) const;
```

```
      void IspisiImenikZaOdjeljenje(const char odjeljenje[]) const;
      void IspisiImenikNaSlovo(char slovo) const;
      void IspisiCitavImenik() const;
      void SortirajImenik() {
      sort(uposlenici, uposlenici + broj_uposlenika,
      KriterijSortiranja); }
      int &operator[](const char ime[]);
      int operator[](const char ime[]) const;
      void Sacuvaj(const char ime_datoteke[]);
      explicit Imenik(const char ime_datoteke[]);
};

Imenik::~Imenik() {
      for(int i = 0; i < broj_uposlenika; i++) delete uposlenici[i];
      delete[] uposlenici;
}

Imenik::Imenik(const Imenik &im) :
      broj_uposlenika(im.broj_uposlenika),
      kapacitet(im.kapacitet), uposlenici(new
      Uposlenik*[im.kapacitet]) {
      for(int i = 0; i < broj_uposlenika; i++)
            uposlenici[i] = new Uposlenik(*im.uposlenici[i]);
}

Imenik &Imenik::operator =(const Imenik &im) {
      if(&im == this) return *this;
      for(int i = 0; i < broj_uposlenika; i++) delete uposlenici[i];
            delete[] uposlenici;
      broj_uposlenika = im.broj_uposlenika; kapacitet = im.kapacitet;
      uposlenici = new Uposlenik*[kapacitet];
      for(int i = 0; i < broj_uposlenika; i++)
            uposlenici[i] = new Uposlenik(*im.uposlenici[i]);
}

void Imenik::DodajUposlenika(const char ime[], const char
odjeljenje[],int telefon) {
      if(broj_uposlenika == kapacitet) throw "Popunjen kapacitet!";
      Uposlenik *novi(new Uposlenik);
      strcpy(novi->ime, ime); strcpy(novi->odjeljenje, odjeljenje);
      novi->telefon = telefon;
      uposlenici[broj_uposlenika++] = novi;}

void Imenik::IspisiPodatke(Uposlenik *u) {
      cout << "Ime i prezime: " << u->ime << endl << "Odjeljenje: "
      << u->odjeljenje << endl << "Broj telefona: " << u->telefon <<
      endl;
}

void Imenik::IspisiUposlenikaSaImenom(const char ime[]) const {
      int brojac(0);
      for(int i = 0; i < broj_uposlenika; i++)
            if(strcmp(uposlenici[i]->ime, ime) == 0) {
            IspisiPodatke(uposlenici[i]); brojac++;
            }
            if(brojac == 0) throw "Nema uposlenika sa tim imenom!";
```

```
}

void Imenik::IspisiUposlenikaSaTelefonom(int telefon) const {
      int brojac(0);
      for(int i = 0; i < broj_uposlenika; i++)
      if(uposlenici[i]->telefon == telefon) {
            IspisiPodatke(uposlenici[i]); brojac++;
      }
      if(brojac == 0)
            throw "Nema uposlenika sa tim brojem  telefona!";
}

void Imenik::IspisiImenikZaOdjeljenje(const char odjeljenje[]) const
{
      for(int i = 0; i < broj_uposlenika; i++)
      if(strcmp(uposlenici[i]->odjeljenje, odjeljenje) == 0)
            IspisiPodatke(uposlenici[i]);
}

void Imenik::IspisiImenikNaSlovo(char slovo) const {
      for(int i = 0; i < broj_uposlenika; i++)
      if(uposlenici[i]->ime[0] == slovo)
      IspisiPodatke(uposlenici[i]);
}

void Imenik::IspisiCitavImenik() const {
      for(int i = 0; i < broj_uposlenika; i++)
      IspisiPodatke(uposlenici[i]);
}

int &Imenik::operator[](const char ime[]) {
      for(int i = 0; i < broj_uposlenika; i++)
      if(strcmp(uposlenici[i]->ime, ime) == 0) return uposlenici[i]-
      >telefon;
      throw "Nema uposlenika sa tim imenom!";
}

int Imenik::operator[](const char ime[]) const {
      for(int i = 0; i < broj_uposlenika; i++)
      if(strcmp(uposlenici[i]->ime, ime) == 0) return uposlenici[i]-
      >telefon;
      throw "Nema uposlenika sa tim imenom!";}

void Imenik::Sacuvaj(const char ime_datoteke[]) {
      ofstream datoteka(ime_datoteke);
      if(!datoteka) throw "Kreiranje datoteke nije uspjelo!";
      datoteka.write((char*)this, sizeof(*this));
      for(int i = 0; i < broj_uposlenika; i++)
      datoteka.write((char*)uposlenici[i], sizeof(**uposlenici));
      if(!datoteka) throw "Snimanje nije uspjelo!";
}

Imenik::Imenik(const char ime_datoteke[]) {
      ifstream datoteka(ime_datoteke);
      if(!datoteka) throw "Datoteka ne postoji!";
      datoteka.read((char*)this, sizeof(*this));
```

```
        uposlenici = new Uposlenik*[kapacitet];
        for(int i = 0; i < broj_uposlenika; i++) {
        uposlenici[i] = new Uposlenik;
        datoteka.read((char*)uposlenici[i], sizeof(**uposlenici));
        }
        if(!datoteka) throw "Citanje nije uspjelo!";
}
```

10.53 **Klasa Vozilo**

Neka je vozilo objekat koji je, između ostalog, karakteriziran svojom težinom. Putničko vozilo je specijalna vrsta vozila, koje može primiti određeni broj putnika od kojih svaki ima svoju težinu. Teretno vozilo je specijalna vrsta vozila koje se može nakrcati teretom određene težine. Razvijte hijerarhiju klasa koje opisuju ove objekte. Bazna klasa "Vozilo" posjedovaće atribut koji predstavlja težinu vozila (tipa cijeli broj), konstruktor koji inicijalizira ovaj atribut, te dvije metode nazvane "DajTezinu" i "DajUkupnuTezinu". Prva metoda daje vlastitu težinu vozila, dok druga metoda daje ukupnu težinu vozila u koju je uračunata i težina svega što se u vozilu nalazi (u slučaju bazne klase "Vozilo" obje metode će vraćati istu vrijednost). Klasa "PutnickoVozilo" nasljeđuje se iz klase "Vozilo", a posjeduje dodatni atribut koji je tipa vektor cijelih brojeva, koji čuva težine putnika u vozilu. Konstruktor ove klase ima dodatni parametar (vektor težina putnika) i izmijenjenu metodu "DajUkupnuTezinu", koja uračunava težine putnika. Klasa "TeretnoVozilo" se također nasljeđuje iz klase "Vozilo", a posjeduje dodatni cjelobrojni atribut koji predstavlja težinu tereta. Ova klasa također ima dodatni parametar u konstruktoru (težina tereta) i izmijenjenu metodu za računanje ukupne težine. Metoda "DajUkupnuTezinu" mora biti izvedena tako da ukoliko se svim opisanim objektima pristupa preko pokazivača na baznu klasu "Vozilo", uvijek bude pozvana ispravna verzija metode, koja će uzeti u obzir specifičnosti objekta. Napisane klase demonstrirajte u testnom programu koji čita podatke o vozilima iz tekstualna datoteke u vektor, sortira vozila po ukupnoj težini u rastući poredak i na kraju ispisuje težine vozila nakon sortiranja (elementi vektora će biti tipa pokazivači na "Vozilo" da bi se mogao koristiti polimorfizam). Svaki red datoteke sadrži podatke o jednom vozilu, poput "V500" za obično vozilo težine 500, "P500 3 80 60 75" za putničko vozilo težine 500 sa 3 putnika težina 80, 60 i 75 respektivno, te "T500 1200" za teretno vozilo težine 500 natovareno sa teretom težine 1200. Pretpostavite da datoteka sadrži ispravne podatke.

Rješenje:

```
class Vozilo {
      int tezina;
      public:
      Vozilo(int tezina) : tezina(tezina) {}
      int DajTezinu() const { return tezina; }
      virtual int DajUkupnuTezinu() const { return tezina; }
      virtual ~Vozilo() {}
};


class PutnickoVozilo : public Vozilo {
      vector<int> putnici;
      public:
      PutnickoVozilo(int tezina, vector<int> putnici) :
      Vozilo(tezina),
      putnici(putnici) {}
      int DajUkupnuTezinu() const;
```

```
};

int PutnickoVozilo::DajUkupnuTezinu() const {
      int ukupna_tezina(DajTezinu());
      for(int i = 0; i < putnici.size(); i++) ukupna_tezina +=
      putnici[i];
      return ukupna_tezina;
}

class TeretnoVozilo : public Vozilo {
      int teret;
      public:
      TeretnoVozilo(int tezina, int teret) : Vozilo(tezina),
      teret(teret) {}
      int DajUkupnuTezinu() { return DajTezinu() + teret; }
};

bool KriterijSortiranja(Vozilo *v1, Vozilo *v2) {
      return v1->DajUkupnuTezinu() < v2->DajUkupnuTezinu();
}

int main() {
      ifstream datoteka("VOZILA.TXT");
      vector<Vozilo*> vozila;
      for(;;) {
      int tezina, teret, broj_putnika;
      char vrsta;
      datoteka >> vrsta >> tezina;
      if(!datoteka) break;
      if(vrsta == 'V') vozila.push_back(new Vozilo(tezina));
      else if(vrsta == 'P') {
      datoteka >> broj_putnika;
      vector<int> putnici(broj_putnika);
      for(int i = 0; i < broj_putnika; i++) datoteka >> putnici[i];
      vozila.push_back(new PutnickoVozilo(tezina, putnici));
      }
      else {
      datoteka >> teret;
      vozila.push_back(new TeretnoVozilo(tezina, teret));
      }
      }
      sort(vozila.begin(), vozila.end(), KriterijSortiranja);
      for(int i = 0; i < vozila.size(); i++)
      cout << vozila[i]->DajUkupnuTezinu() << endl;
      for(int i = 0; i < vozila.size(); i++) delete vozila[i];
      return 0;
}
```

10.54 **Klasa Dual**

Svima koji se bave tehničkim naukama dobro su poznati komplesni brojevi i njihova svojstva, koji se mogu formalno posmatrati kao uređeni parovi realnih brojeva oblika `(a, b)` gdje su `a` i `b` realni brojevi. Međutim, iako su kompleksni brojevi najpoznatije i najprimjenljivije tvorevine takve vrste, one nisu i jedini korisni matematički objekti koji se modeliraju kao parovi realnih brojeva. U posljednje vrijeme, velika pažnja se posvećuje tzv. dualnim brojevima. Mada su prvobitno nastali kao eksperiment unutar linearne algebre, našli su

izuzetno veliku primjenu upravo u računarstvu i to u situacijama kada treba isprogramirati numeričke algoritme u kojima se javlja potreba za računanjem izvoda funkcija (ovo je odlična tema za završni rad za one koji budu zainteresirani). Ovi neobični brojevi našli su primjenu i u nekim oblastima fizike, recimo za potrebe proučavanja dinamike krutih tijela i za razne perturbacione metode u fizici. Intuitivno, dualni brojevi su brojevi oblika `x + y ε`, gdje su x i y realni brojevi, dok je ε tzv. nilpotenta, donekle bizaran objekat za koji vrijedi `ε2 = 0` mada je `ε ≠ 0` (postojanje ovakvih bizarnih objekata može se lako opravdati pomoću linearne algebre). Formalno, dualni brojevi se mogu također predstaviti kao parovi realnih brojeva, koje ćemo označiti sa `< x ′y >` da bismo ih razlikovali od kompleksnih brojeva `(x, y)`, i za koje vrijede sljedeća pravila računanja:

$$< x_1 ' y_1 > \pm < x_2 ' y_2 > = < x_1 \pm x_2 ' y_1 \pm y_2 >$$
$$< x_1 ' y_1 > \cdot < x_2 ' y_2 > = < x_1 x_2 ' x_1 y_2 + x_2 y_1 >$$
$$< x_1 ' y_1 > / < x_2 ' y_2 > = \left\langle \frac{x_1}{x_2} , \frac{x_2 y_1 - x_1 y_2}{x_2^{\,2}} \right\rangle$$

Može se primijetiti izvjesna sličnost sa računom sa kompleksnim brojevima, uz gubljenje pojedinih članova (koje su posljedica činjenice da je ε2 jednako 0, dok je i2 jednako –1). Uz ovakav formalizam, imamo `ε = <0 ′1>`. Broj x se naziva vidljivi, a broj y skriveni dio dualnog broja <x ’ y >. Negacija dualnog broja `<x ′ y >` data je kao `<−x ′−y >`, dok je njegova konjugacija data kao `<x ′ −y >`. Apsolutna vrijednost dualnog broja `<x ′ y >` data je kao

$$| < x ' y > | = | x |$$

Drugim riječima, skriveni dio dualnog broja ne figurira u njegovoj apsolutnoj vrijednosti. Dualni broj `<x ′ 0 >` može se poistovijetiti sa realnim brojem x. Dva dualna broja `<x1 ′ y1 >` i `<x2 ′ y2 >` jednaki su ako i samo ako je `x1 = x2` i `y1 = y2`. Poredak dualnih brojeva se ne definira. Definirajte klasu za rad sa dualnim brojevima. Klasa bi trebala da sadrži konstruktor koji kreira dualni broj na osnovu zadanog vidljivog i skrivenog dijela koji se zadaju kao parametri. Oba parametra trebaju imati podrazumijevanu vrijednost 0, tako da će se ovaj konstruktor moći koristiti i kao konstruktor bez parametara odnosno sa jednim parametrom. Klasa treba podržavati dvije pristupne metode bez parametara, kojima se pristupa vidljivom i skrivenom dijelu dualnog broja. Dalje, treba podržavati operatore “+”, “–”, “*” i “/” za obavljanje četiri osnovne računske operacije, zatim kombinovane operatore “+=”, “–=”, “*=” i “/=” sa uobičajenim značenjem, relacione operatore “==” i “!=”, unarni operator negacije “–”, te unarni operator “++” koji povećava realni dio dualnog broja za jedinicu, dok skriveni dio ostaje isti (potrebno je podržati i prefiksnu i postfiksnu verziju ovog operatora). Također je potrebno podržati i operatore “<<” i “>>” za ispis dualnih brojeva na izlazni tok i njihov unos sa ulaznog toka. Perpleksne brojeve treba ispisivati kao parove oblika <x ’ y > bez obzira da li su im neke komponente nule ili nisu, dok unos dualnih brojeva sa ulaznog toka treba podržati bilo kao par oblika <x ’ y >, bilo kao običan realan broj, koji se potom interpretira kao dualni broj. Konačno, treba implementirati i četiri obične funkcije (ne članice) koje kao rezultat daju respektivno apsolutnu vrijednost, konjugaciju, vidljivi i skriveni dio dualnog broja (neka Vas ne zbunjuje što postoje i metode koje daju vidljivi i skriveni dio dualnog broja – ovako se omogućava veća sintaksna šarolikost u primjeni).

Rješenje:

```
class Dualni {
```

```
double vid, skr;
public:
Dualni(double vidljivi = 0, double skriveni = 0) :
vid(vidljivi),skr(skriveni) {}
double DajVidljivi() const { return vid; }
double DajSkriveni() const { return skr; }
friend Dualni operator +(const Dualni &d1, const Dualni &d2) {
return Dualni(d1.vid + d2.vid, d1.skr + d2.skr);}
friend Dualni operator -(const Dualni &d1, const Dualni &d2) {
return Dualni(d1.vid - d2.vid, d1.skr - d2.skr);}
friend Dualni operator *(const Dualni &d1, const Dualni &d2) {
return Dualni(d1.vid * d2.vid, d1.vid * d2.skr + d2.vid
*d1.skr);}
friend Dualni operator /(const Dualni &d1, const Dualni &d2) {
return Dualni(d1.vid / d2.vid, (d2.vid * d1.skr
- d1.vid * d2.skr) / (d2.skr * d2.skr));
}
Dualni &operator +=(const Dualni &d)
{ return *this = *this + d; }
Dualni &operator -=(const Dualni &d)
{ return *this = *this - d; }
Dualni &operator *=(const Dualni &d)
{ return *this = *this * d; }
Dualni &operator /=(const Dualni &d)
{ return *this = *this / d; }
friend bool operator ==(const Dualni &d1, const Dualni &d2) {
return d1.vid == d2.vid && d1.skr == d2.skr;}
friend bool operator !=(const Dualni &d1, const Dualni &d2) {
return !(d1 == d2);}
Dualni operator -() { return Dualni(-vid, -skr); }
Dualni &operator ++() { vid++; return *this; }
Dualni operator ++(int) { Dualni d(*this); vid++; return d; }
friend ostream &operator <<(ostream &cout, const Dualni &d) {
return cout << "(" << d.vid << "|" << d.skr << ")";
}
friend istream &operator >>(istream &cin, Dualni &d);
friend double ApsolutnaVrijednost(const Dualni &d) { return
fabs(d.vid); }
friend Dualni Konjugirani(const Dualni &d) { return
Dualni(d.vid, -d.skr); }
friend double VidljiviDio(const Dualni &d) { return d.vid; }
friend double SkriveniDio(const Dualni &d) { return d.skr; }
};
istream &operator >>(istream &cin, Dualni &d) {
     if(cin.peek() != '<') {
     cin >> d.vid; d.skr = 0;
}
else {
char znak;
cin >> znak;
if(znak != '<') cin.setstate(ios::failbit);
     cin >> d.vid >> znak;
if(znak != '\'') cin.setstate(ios::failbit);
     cin >> d.skr >> znak;
```

```
        if(znak != '>') cin.setstate(ios::failbit);
        }
        return cin;
}
```

10.55 **Klasa telefonski imenik**

Neka trgovačka firma je odlučila da napravi interni telefonski imenik svojih klijenata. Podaci o jednom klijentu čuvaju se u strukturi koja kao svoje atribute sadrži ime klijenta zajedno sa prezimenom (tipa niza od max. 20 znakova), grad u kojem se klijent nalazi (tipa niza od max. znakova) i broj telefona (cjelobrojnog tipa). Potrebno je razviti kontejnersku klasu koja će čuvati kolekciju podataka o klijentima i njihovim telefonskim brojevima. Ovi podaci će se alocirati dinamički, a pristupaće im se preko dinamički alociranog niza pokazivača koji pokazuju na ove podatke. Interfejs klase treba sadržavati sljedeće elemente:

a) Konstruktor sa jednim parametrom koji vrši neophodnu dinamičku alokaciju, pri čemu parametar predstavlja maksimalni broj uposlenika koji se mogu evidentirati. Pri tome se ovaj konstruktor ne smije koristiti za automatsku konverziju cjelobrojnih podataka u tip ove klase.
b) Destruktor, koji oslobađa sve resurse koje su primjerci ove klase zauzeli tokom svog života.
c) Konstruktor kopije i preklopljeni operator dodjele koji omogućavaju da se primjerci ove klase mogu bezbjedno kopirati i međusobno dodjeljivati. d) Metodu koja dodaje u kolekciju podatke o novom klijentu, pri čemu su parametri metode podaci o klijentu (ime i prezime, grad i broj telefona). U slučaju da su podaci o imenu i prezimenu ili gradu predugi, ili ukoliko se dostigne maksimalni broj klijenata koji se mogu evidentirati, treba baciti izuzetak.
d) Metodu koja ispisuju podatke o klijentu čije se ime i prezime zadaje kao parametar i metodu koja ispisuje podatke o klijentu čiji se broj telefona zadaje kao parametar.
e) Metodu koja ispisuje telefonski imenik za sve klijente koji žive u gradu koje se zadaje kao
f) parametar, zatim metodu koja ispisuje telefonski imenik za sve klijente čije ime počinje slovom koje se zadaje kao parametar, te metodu koja ispisuje čitav telefonski imenik.
g) Metodu koje sortira telefonski imenik po abecednom poretku klijenata (koristiti funkciju "sort").
h) Preklopljeni operator "[]" koji vraća referencu na broj telefona klijenta čije se ime zadaje unutar uglastih zagrada (referenca se vraća sa ciljem da se omogući izmjena broja). U slučaju da se ovaj operator primijeni na konstantni objekat, umjesto reference treba vratiti kopiju broja.
i) Metodu koja snima čitav telefonski imenik u binarnu datoteku čije se ime zadaje kao parametar, te konstruktor sa jednim parametrom koji prilikom kreiranja imenika obnavlja njegov sadržaj iz binarne datoteke čije se ime zadaje kao parametar konstruktora.

Rješenje:

Napomena: U prikazanom rješenju, struktura "`Klijent`" umjesto da se deklarira izvan klase na globalnom nivou, deklarirana je lokalno u privatnom dijelu klase, tako da joj samo metode klase "`Imenik`" mogu pristupiti, odnosno ostatak programa "ne zna" za postojanje ove strukture. Mada se to nije očekivalo od studenta da uradi, ovdje je prikazano čisto kao ilustracija da se takve tehnike često koriste.

```
class Imenik {

      struct Klijent {
            char ime[20], grad[10];
            int telefon;
      };

      int kapacitet, broj_klijenata;
      Klijent **klijenti;
      static void IspisiPodatke(Klijent *k); // Pomoćna funkcija za ispis
      static bool KriterijSortiranja(Klijent *k1, Klijent *k2)
       {
            return strcmp(k1->ime, k2->ime) < 0;
      }

      public:
      explicit Imenik(int kapacitet) : kapacitet(kapacitet),
      broj_klijenata(0),
      klijenti(new Klijent*[kapacitet]) {}
      ~Imenik();
      Imenik(const Imenik &im);
      Imenik &operator=(const Imenik &im);
      void DodajKlijenta(const char ime[], const char grad[], int
      telefon);
      void IspisiKlijentaSaImenom(const char ime[]) const;
      void IspisiKlijentaSaTelefonom(int telefon) const;
      void IspisiImenikZaGrad(const char grad[]) const;
      void IspisiImenikNaSlovo(char slovo) const;
      void IspisiCitavImenik() const;
      void SortirajImenik()
      {
      sort(klijenti, klijenti + broj_klijenata, KriterijSortiranja);
      }

      int &operator[](const char ime[]);
      int operator[](const char ime[]) const;
      void Sacuvaj(const char ime_datoteke[]);
      explicit Imenik(const char ime_datoteke[]);
};

Imenik::~Imenik() {
      for(int i = 0; i < broj_klijenata; i++) delete klijenti[i];
      delete[] klijenti;
}

Imenik::Imenik(const Imenik &im) :
      broj_klijenata(im.broj_klijenata),
      kapacitet(im.kapacitet), klijenti(new Klijent*[im.kapacitet]) {
      for(int i = 0; i < broj_klijenata; i++)
      klijenti[i] = new Klijent(*im.klijenti[i]);
}

Imenik &Imenik::operator =(const Imenik &im) {
      if(&im == this) return *this;
      for(int i = 0; i < broj_klijenata; i++)
```

```
            delete klijenti[i];
      delete[] klijenti;
      broj_klijenata = im.broj_klijenata; kapacitet = im.kapacitet;
      klijenti = new Klijent*[kapacitet];
      for(int i = 0; i < broj_klijenata; i++)
            klijenti[i] = new Klijent(*im.klijenti[i]);
}

void Imenik::DodajKlijenta(const char ime[], const char grad[], int
telefon) {
      if(broj_klijenata == kapacitet) throw "Popunjen kapacitet!";
      Klijent *novi(new Klijent);
      strcpy(novi->ime, ime); strcpy(novi->grad, grad);
      novi->telefon = telefon;
      klijenti[broj_klijenata++] = novi;
}

void Imenik::IspisiPodatke(Klijent *k) {
      cout << "Ime i prezime: " << k->ime << endl << "Grad: " << k-
      >grad << endl
      << "Broj telefona: " << k->telefon << endl;
}

void Imenik::IspisiKlijentaSaImenom(const char ime[]) const {
      int brojac(0);
      for(int i = 0; i < broj_klijenata; i++)
      if(strcmp(klijenti[i]->ime, ime) == 0) {
      IspisiPodatke(klijenti[i]); brojac++;
      }
      if(brojac == 0) throw "Nema klijenata sa tim imenom!";
}

void Imenik::IspisiKlijentaSaTelefonom(int telefon) const {
      int brojac(0);
      for(int i = 0; i < broj_klijenata; i++)
      if(klijenti[i]->telefon == telefon) {
      IspisiPodatke(klijenti[i]); brojac++;
      }
      if(brojac == 0) throw "Nema klijenata sa tim brojem telefona!";
}

void Imenik::IspisiImenikZaGrad(const char grad[]) const {
      for(int i = 0; i < broj_klijenata; i++)
      if(strcmp(klijenti[i]->grad, grad) == 0)
      IspisiPodatke(klijenti[i]);
}


void Imenik::IspisiImenikNaSlovo(char slovo) const {
      for(int i = 0; i < broj_klijenata; i++)
      if(klijenti[i]->ime[0] == slovo) IspisiPodatke(klijenti[i]);
}

void Imenik::IspisiCitavImenik() const {
      for(int i = 0; i < broj_klijenata; i++)
      IspisiPodatke(klijenti[i]);
```

```
}

int &Imenik::operator[](const char ime[]) {
      for(int i = 0; i < broj_klijenata; i++)
      if(strcmp(klijenti[i]->ime,  ime)  ==  0)  return  klijenti[i]
      >telefon;
      throw "Nema klijenata sa tim imenom!";
}

int Imenik::operator[](const char ime[]) const {
      for(int i = 0; i < broj_klijenata; i++)
      if(strcmp(klijenti[i]->ime, ime) == 0) return klijenti[i]-
      >telefon;
      throw "Nema klijenata sa tim imenom!";
}

void Imenik::Sacuvaj(const char ime_datoteke[]) {
      ofstream datoteka(ime_datoteke);
      if(!datoteka) throw "Kreiranje datoteke nije uspjelo!";
      datoteka.write((char*)this, sizeof(*this));
      for(int i = 0; i < broj_klijenata; i++)
            datoteka.write((char*)klijenti[i], sizeof(**klijenti));
      if(!datoteka) throw "Snimanje nije uspjelo!";
}

Imenik::Imenik(const char ime_datoteke[]) {
      ifstream datoteka(ime_datoteke);
      if(!datoteka) throw "Datoteka ne postoji!";
      datoteka.read((char*)this, sizeof(*this));
      klijenti = new Klijent*[kapacitet];
      for(int i = 0; i < broj_klijenata; i++) {
      klijenti[i] = new Klijent;
      datoteka.read((char*)klijenti[i], sizeof(**klijenti));
      }
      if(!datoteka) throw "Citanje nije uspjelo!";
}
```

10.54 **Klasa Vozilo**

Neka je vozilo objekat koji je, između ostalog, karakteriziran svojom težinom. Automobil je specijalna vrsta vozila, koje može primiti određeni broj putnika od kojih svaki ima svoju težinu. Kamion je specijalna vrsta vozila koje se može nakrcati teretom određene težine. Razvijte hijerarhiju klasa koje opisuju ove objekte. Bazna klasa "Vozilo" posjedovaće atribut koji predstavlja težinu vozila (tipa cijeli broj), konstruktor koji inicijalizira ovaj atribut, te dvije metode nazvane "DajTezinu" i "DajUkupnuTezinu". Prva metoda daje vlastitu težinu vozila, dok druga metoda daje ukupnu težinu vozila u koju je uračunata i težina svega što se u vozilu nalazi (u slučaju bazne klase "Vozilo" obje metode će vraćati istu vrijednost). Klasa "Automobil" nasljeđuje se iz klase "Vozilo", a posjeduje dodatni atribut koji je tipa vektor cijelih brojeva, koji čuva težine putnika u vozilu. Konstruktor ove klase ima dodatni parametar (vektor težina putnika) i izmijenjenu metodu "DajUkupnuTezinu", koja uračunava težine putnika. Klasa "Kamion" se također nasljeđuje iz klase "Vozilo", a posjeduje dodatni cjelobrojni atribut koji predstavlja težinu tereta. Ova klasa također ima dodatni parametar u

konstruktoru (težina tereta) i izmijenjenu metodu za računanje ukupne težine. Metoda "DajUkupnuTezinu" mora biti izvedena tako da ukoliko se svim opisanim objektima pristupa preko pokazivača na baznu klasu "Vozilo", uvijek bude pozvana ispravna verzija metode, koja će uzeti u obzir specifičnosti objekta. Napisane klase demonstrirajte u testnom programu koji čita podatke o vozilima iz tekstualna datoteke u vektor, sortira vozila po ukupnoj težini u rastući poredak i na kraju ispisuje težine vozila nakon sortiranja (elementi vektora će biti tipa pokazivači na "Vozilo" da bi se mogao koristiti polimorfizam). Svaki red datoteke sadrži podatke o jednom vozilu, poput "V500" za obično vozilo težine 500, "A500 3 80 60 75" za automobil težine 500 sa 3 putnika težina 80, 60 i 75 respektivno, te "K500 1200" za kamion težine 500 natovaren sa teretom težine 1200. Pretpostavite da datoteka sadrži ispravne podatke.

Rješenje:

```
class Vozilo {
      int tezina;
      public:
      Vozilo(int tezina) : tezina(tezina) {}
      int DajTezinu() const { return tezina; }
      virtual int DajUkupnuTezinu() const { return tezina; }
      virtual ~Vozilo() {}
};

class Automobil : public Vozilo {
      vector<int> putnici;
      public:
      Automobil(int tezina, vector<int> putnici) : Vozilo(tezina),
      putnici(putnici) {}
      int DajUkupnuTezinu() const;
};

int Automobil::DajUkupnuTezinu() const {
      int ukupna_tezina(DajTezinu());
      for(int i = 0; i < putnici.size(); i++) ukupna_tezina +=
      putnici[i];
      return ukupna_tezina;
}

class Kamion : public Vozilo {
      int teret;
      public:
      Kamion(int tezina, int teret) : Vozilo(tezina), teret(teret) {}
      int DajUkupnuTezinu() { return DajTezinu() + teret; }
      };

bool KriterijSortiranja(Vozilo *v1, Vozilo *v2) {
return v1->DajUkupnuTezinu() < v2->DajUkupnuTezinu();
}

int main() {
      ifstream datoteka("VOZILA.TXT");
      vector<Vozilo*> vozila;
      for(;;) {
      int tezina, teret, broj_putnika;
```

```
        char vrsta;
        datoteka >> vrsta >> tezina;
        if(!datoteka) break;
        if(vrsta == 'V') vozila.push_back(new Vozilo(tezina));
        else if(vrsta == 'A') {
        datoteka >> broj_putnika;
        vector<int> putnici(broj_putnika);
        for(int i = 0; i < broj_putnika; i++) datoteka >> putnici[i];
        vozila.push_back(new Automobil(tezina, putnici));
}
else {
datoteka >> teret;
vozila.push_back(new Kamion(tezina, teret));
}
}
sort(vozila.begin(), vozila.end(), KriterijSortiranja);
for(int i = 0; i < vozila.size(); i++)
cout << vozila[i]->DajUkupnuTezinu() << endl;
for(int i = 0; i < vozila.size(); i++) delete vozila[i];
return 0;
}
```

10.54 **Klase Knjiga, Udžbenik i Biblioteka**

NAPOMENA: U svim klasama koje treba razviti, trivijalne metode koje se mogu implementirati u jednoj ili najviše dvije naredbe možete implementirati direktno unutar deklaracije klase, dok sve ostale metode trebate implementirati izvan deklaracije klase.

Za potrebe vođenja evidencije o knjigama u nekoj biblioteci, u evidencijskom programu koriste se klase nazvane “Knjiga”, “Udzbenik” i “Biblioteka”. Vaš zadatak je da definirate i implementirate te klase i da napišete glavni program koji koristi te klase. Klasa “Knjiga” opisuje podatke o jednoj knjizi, a sadrži sljedeće elemente:

a) Privatni evidencijski broj knjige (cijeli broj).
b) Privatne podatke o naslovu knjige, imenu pisca i žanr (svi ovi podaci su tipa “string”).
c) Privatni podatak o godini izdanja knjige (cijeli broj).
d) Privatni podatak o članskom broju čitaoca kod koga se trenutno nalazi knjiga, ili 0 ukoliko knjiga trenutno nije na čitanju (tj. ukoliko je trenutno raspoloživa).
e) Privatni podatak koliko dugo dana je knjiga na čitanju (ukoliko knjiga trenutno nije na čitanju, vrijednost ovog podatka je nebitna).
f) Konstruktor, koji inicijalizira podatke o evidencijskom broju, naslovu knjige, imenu pisca, žanru i godini izdanja na vrijednosti zadane parametrima. Pri tome se podaci o čitaocu inicijaliziraju tako da signaliziraju da je knjiga slobodna.
g) Trivijalne pristupne metode, kojim se može saznati vrijednost svih privatnih podataka.
h) Metodu koja knjigu proglašava “zaduženom”. Parametar metode je članski broj čitaoca koji zadužuje knjigu. Pri zaduženju, informacija o tome koliko dugo je knjiga na čitanju postavlja se na nulu.
i) Metodu bez parametara koja “razdužuje” knjigu.
j) Preklopljeni unarni operator “!” koji vraća logičku vrijednost “true” ukoliko je knjiga zadužena, a “false” ukoliko nije.
k) Preklopljeni unarni operator “++” koji povećava evidenciju o tome koliko je dugo knjiga na čitanju za jedinicu, ukoliko je knjiga zadužena. Ukoliko knjiga nije zadužena, ovaj operator ne radi ništa. Potrebno je podržati i prefiksnu i postfiksnu verziju ovog operatora.

l) Preklopljene relacione operatore “==” i “!=” koji testiraju da li su dvije knjige iste ili nisu. Smatra se da su dvije knjige iste ukoliko im se slažu naslov, ime pisca, žanr i godina izdanja.
m) Privatnu virtuelnu metodu koja ispisuje na izlazni tok (recimo, ekran) podatke o evidencijskom broju, naslovu, imenu pisca, žanru i godini izdanja knjige. Parametar metode je referenca na objekat toka preko kojeg se vrši ispis (tipa “ostream”). Stil ispisa odredite po želji.
n) Preklopljeni operator ispisa “<<” koji ispisuje na izlazni tok iste podatke kao i prethodna metoda (u principu, ovaj operator samo treba pozvati prethodnu metodu).

Klasa “Udzbenik” je naslijeđena iz klase “Knjiga”. Novi elementi ove klase u odnosu na baznu klasu “Knjiga” su sljedeći:

a) Privatni podatak o tome za koji je predmet udžbenik namijenjen (tipa “string”).
b) Pristupna metoda pomoću koje se može saznati za koji je predmet udžbenik namijenjen.
c) Konstruktor ove klase je proširen da omogući i zadavanje predmeta za koji je udžbenik namijenjen.
d) Privatna metoda za ispis je izmijenjena da predvidi i ispis predmeta za koji je udžbenik namijenjen.

Svi ostali elementi klase “Udzbenik” se prosto preuzimaju iz klase “Knjiga”. Klasa “Biblioteka” opisuje kolekciju knjiga (uključujući i udžbenike), a sadrži sljedeće elemente:
a) Privatne informacije o broju knjiga u biblioteci, kao i o maksimalnom broju knjiga koje biblioteka može evidentirati. Pri tome, podatak o maksimalnom broju knjiga koje biblioteka može evidentirati treba biti izveden kao konstantni atribut.
b) Podatke o evidentiranim knjigama, izvedene kao dinamički niz pokazivača na dinamički alocirane objekte tipa “Knjiga”, kojima se pristupa preko privatnog dvojnog pokazivača.
c) Konstruktor koji vrši dinamičku alokaciju memorije, pri čemu parametar predstavlja maksimalan broj knjiga koje biblioteka može evidentirati, kao i odgovarajući destruktor. Pri tome, treba zabraniti da se ovaj konstruktor koristi za automatsku konverziju cijelih brojeva u objekte tipa “Biblioteka”. Pored toga, kopiranje i međusobno dodjeljivanje objekata tipa “Biblioteka” treba zabraniti.
d) Metodu koja kreira novu knjigu (običnu) i evidentira je u biblioteci. Parametri ove metode su isti kao u konstruktoru klase “Knjiga”.
e) Metodu koja kreira novi udžbenik i evidentira ga u biblioteci. Parametri ove metode su isti kao u konstruktoru klase “Udzbenik”.
f) Metodu koja “zadužuje” knjigu, pri čemu se kao parametri zadaju evidencijski broj knjige i članski
g) broj čitaoca koji zadužuje knjigu. Ukoliko knjiga sa zadanim evidencijskim brojem ne postoji, treba baciti izuzetak.
h) Metodu koja “razdužuje” knjigu, pri čemu se kao parametri zadaje evidencijski broj knjige koja se razdužuje. Ukoliko knjiga sa zadanim evidencijskim brojem ne postoji, treba baciti izuzetak.
i) Metodu koja vraća logičku vrijednost “tačno” ili “netačno”, ovisno da li je knjiga čiji je evidencijski broj zadan kao parametar trenutno zadužena ili ne. Ukoliko knjiga sa zadanim evidencijskim brojem ne postoji, treba baciti izuzetak.

j) Metodu koja vraća članski broj čitaoca kod kojeg se nalazi knjiga čiji je evidencijski broj zadan kao parametar (odnosno nulu ukoliko knjiga nije zadužena). Ukoliko knjiga sa zadanim evidencijskim brojem ne postoji, treba baciti izuzetak.
k) Medode koje ispisuje podatke o svim slobodnim knjigama, svim zaduženim knjigama, i knjigama koje su na čitanju duže od n dana, pri čemu se n zadaje kao parametar.
l) Metodu koja sortira sve knjige u takav poredak da se knjige koje se su duže vremena zadužene nađu ispred onih koje su kraće vremena zadužene. Ukoliko dvije knjige imaju isto trajanje zaduženja, knjiga čiji naziv dolazi prije po abecedi treba da bude ispred knjige koja dolazi kasnije po abecedi.
m) Preklopljeni operator "[]" koji vraća referencu na knjigu čiji se evidencijski broj navodi unutar uglastih zagrada. Ukoliko knjiga sa zadanim evidencijskim brojem ne postoji, treba baciti izuzetak.
n) Preklopljeni unarni operator "++" koji povećava evidenciju o tome koliko je dugo knjiga na čitanju za jedinicu za sve zadužene knjige. Potrebno je podržati samo prefiksnu verziju ovog operatora. Napišite i glavni program, koji će iz tekstualnih datoteka pročitati podatke o knjigama i njihovim zaduženjima i nakon toga ispisati podatke o svim zaduženim i svim slobodnim knjigama. Podaci o knjigama nalaze se u tekstualnoj datoteci "KNJIGE.TXT". Ova datoteka je organizirana tako da se za svaku knjigu podaci o evidencijskom broju, naslovu knjige, imenu pisca, žanru i godini izdanja u datoteci nalaze upravo tim redom, svaki podatak u posebnom redu. Ukoliko je u pitanju udžbenik, ispred evidencijskog broja nalazi se slovo "U", a na kraju se nalazi i podatak o predmetu za koji je udžbenik namijenjen. Na primjer, datoteka "KNJIGE.TXT" može izgledati ovako:

   1234
   Derviš i smrt
   Meša Selimović
   Roman
   1976
   U4312
   Zbirka zadataka iz više matematike
   Momčilo Uščumlić
   Zbirka zadataka
   1983
   Inžinjerska matematika I/II
   ...

   Podaci o zaduženjima nalaze se u datoteci "ZADUZENJA.TXT". Svaki red ove datoteke predstavlja po jednu zaduženu knjigu, a u njemu se nalaze evidencijski broj knjige i članski broj čitaoca koji je zadužio knjigu, razdvojeni zarezom (npr. "4312,344"). Radi jednostavnosti, možete pretpostaviti da obje datoteke sigurno sadrže ispravne podatke.

Rješenje:
Prvo slijedi klasa "Knjiga". Kako su sve metode ove klase kratke, implementirane su uglavnom unutardeklaracije klase, osim operatorske funkcije za prefiksni operator "++" (izvedene kao funkcija članica), operatorske funkcije za operator "==" (izvedene kao prijateljska funkcija) i virtualne funkcije članice "IspisiNaTok" koje su, čisto radi demonstracije, implementirane izvan klase:

```
class Knjiga {
     int ev_broj, god_izdanja, kod_koga_je, koliko_dugo;
```

```
      string naslov, pisac, zanr;
      public:

      Knjiga(int ev_broj, string naslov, string pisac, string zanr,
      int god_izdanja) : ev_broj(ev_broj), naslov(naslov),
      pisac(pisac), zanr(zanr), god_izdanja(god_izdanja),
      kod_koga_je(0) {}

      int DajEvidencijskiBroj() const { return ev_broj; }
      string DajNaslov() const { return naslov; }
      string DajImePisca() const { return pisac; }
      string DajZanr() const { return zanr; }
      int DajGodinuIzdanja() const { return god_izdanja; }
      int DajKodKogaJe() const { return kod_koga_je; }
      int DajKolikoJeDugoNaCitanju() const { return koliko_dugo; }
      void ZaduziKnjigu(int clanski_broj) {
            kod_koga_je = clanski_broj; koliko_dugo = 0;
      }
      void RazduziKnjigu() { kod_koga_je = 0; }
      bool operator !() { return kod_koga_je != 0; }
      Knjiga &operator ++();
      Knjiga operator ++(int) { Knjiga k(*this); ++(*this); return k;
      }
      friend bool operator ==(const Knjiga &k1, const Knjiga &k2);
      friend bool operator !=(const Knjiga &k1, const Knjiga &k2) {
            return !(k1 == k2);
      }
      virtual void IspisiNaTok(ostream &cout) const;
      friend ostream &operator <<(ostream &cout, const Knjiga &k) {
            k.IspisiNaTok(cout); return cout;
      }
};

Knjiga &Knjiga::operator ++() {
      if(kod_koga_je != 0) koliko_dugo++;
      return *this;
}

bool operator ==(const Knjiga &k1, const Knjiga &k2) {
      return k1.naslov == k2.naslov && k1.pisac == k2.pisac
      && k1.zanr == k2.zanr && k1.god_izdanja == k2.god_izdanja;
}

void Knjiga::IspisiNaTok(ostream &cout) const {
      cout << "Evidencijski broj: " << ev_broj << endl << "Naslov: "
      << naslov
      << endl << "Ime pisca: " << pisac << endl << "Žanr: " << zanr
      << endl
      << "Godina izdanja: " << god_izdanja << endl;
}
```

*U klasi "*Udzbenik*" nema se ništa posebno raditi, s obzirom da se radi o neznatno nadograđenoj specijalizaciji klase "*Knjiga*":*

```
class Udzbenik : public Knjiga {
```

```
      string predmet;
      public:
      string DajPredmet() const { return predmet; }
      Udzbenik(int ev_broj, string naslov, string pisac, string zanr,
      int god_izdanja, string predmet) : Knjiga(ev_broj, naslov,
      pisac, zanr, god_izdanja), predmet(predmet) {}
      void IspisiNaTok(ostream &cout) const;
};

void Udzbenik::IspisiNaTok(ostream &cout) const {
      Knjiga::IspisiNaTok(cout);
      cout << "Predmet: " << predmet << endl;
}
```

*U prikazanoj izvedbi klase "*Biblioteka*", implementacije metoda poput "*ZaduziKnjigu*" itd. suizuzetno kratke, s obzirom da se oslanjaju na preklopljeni operator "*[]*" i odgovarajuće metode klase "*Knjiga*". Kasnije će biti demonstrirano kako se ove metode mogu izvesti neovisno od preklopljenog operatora "*[]*" po cijenu da postanu složenije:*

```
class Biblioteka {
      int broj_knjiga;
      const int max_br_knjiga;
      Knjiga **knjige;
      Biblioteka(const Knjiga &b); // Neimplementirano...
      Biblioteka &operator =(const Biblioteka &b); //
      Neimplementirano...
      static bool Kriterij(const Knjiga *k1, const Knjiga *k2);
      public:
      explicit Biblioteka(int kapacitet) : broj_knjiga(0),
      max_br_knjiga(kapacitet), knjige(new Knjiga*[kapacitet]) {}
      void KreirajKnjigu(int ev_broj, string naslov, string pisac,
      string zanr, int god_izdanja);
      void KreirajUdzbenik(int ev_broj, string naslov, string pisac,
      string zanr, int god_izdanja, string predmet);
      void ZaduziKnjigu(int ev_broj, int clanski_broj) {
      (*this)[ev_broj].ZaduziKnjigu(clanski_broj);
      }
      void RazduziKnjigu(int ev_broj) {
      (*this)[ev_broj].RazduziKnjigu(); }
      bool DaLiJeZaduzena(int ev_broj) const { return
      !(*this)[ev_broj]; }
      int KodKogaJeKnjiga(int ev_broj) const {
      return (*this)[ev_broj].DajKodKogaJe();
      }
      void IspisiSlobodne() const;
      void IspisiZauzete() const;
      void IspisiNaCitanjuDuzeOd(int br_dana) const;
      void SortirajKnjige() { sort(knjige, knjige + broj_knjiga,
      Kriterij); }
      Knjiga &operator [](int ev_broj);
      Knjiga operator [](int ev_broj) const;
      Biblioteka &operator ++();
};
```

Konstrukcija "`(*this)[ev broj]`*" u metodama poput "*`ZaduziKnjigu`*" itd. može se zamijeniti ekvivalentnom konstrukcijom "*`operator [](ev broj)`*" u kojoj se direktno poziva operatorska funkcija za operator "*`[]`*", recimo "*`operator [](ev broj).ZaduziKnjigu(clanski broj)`*". Alternativno, ove metode su se mogle napisati bez pozivanja na preklopljeni operator "*`[]`*", na primjer na sljedeći način koji pokazuje alternativnu izvedbu metode "*`ZaduziKnjigu`*" (srodne metode poput "*`RazduziKnjigu`*", "*`DaLiJeZaduzena`*" i "*`KodKogaJeKnjiga`*" mogu se izvesti analogno):*

```
void Biblioteka::ZaduziKnjigu(int ev_broj, int clanski_broj) {
      for(int i = 0; i < broj_knjiga; i++)
      if(knjige[i]->DajEvidencijskiBroj() == ev_broj) {
      knjige[i]->ZaduziKnjigu(clanski_broj);
      return;
      }
      throw "Nema knjige sa traženim evidencijskim brojem!";
}
```

Slijede implementacije ostalih metoda koje nisu implementirane unutar klase:

```
void Biblioteka::KreirajKnjigu(int ev_broj, string naslov, string
pisac, string zanr, int god_izdanja) {
      if(broj_knjiga == max_br_knjiga) throw "Popunjen kapacitet!";
      knjige[broj_knjiga++] = new Knjiga(ev_broj, naslov, pisac,
      zanr, god_izdanja);
}

void Biblioteka::KreirajUdzbenik(int ev_broj, string naslov, string
pisac, string zanr, int god_izdanja, string predmet) {
      if(broj_knjiga == max_br_knjiga) throw "Popunjen kapacitet!";
      knjige[broj_knjiga++] = new Udzbenik(ev_broj, naslov, pisac,
      zanr, god_izdanja, predmet);
}

void Biblioteka::IspisiSlobodne() const {
      for(int i = 0; i < broj_knjiga; i++)
      if(!!*knjige[i]) cout << *knjige[i];
}
```

U prethodnoj metodi upotrijebljena je neobična konstrukcija "`!!*knjige[i]`*". Međutim, ona je posve logična, s obzirom da je "*`*knjige[i]`*" objekat tipa "*`Knjiga`*" za koji je definiran operator "*`!`*", tako da je "*`!*knjige[i]`*" logička vrijednost na koju se ponovo može primijeniti operator "*`!`*", ovaj put u značenju logičke negacije. Bez ove konstrukcije, prethodna metoda se mogla izvesti recimo ovako:*

```
void Biblioteka::IspisiSlobodne() const {
      for(int i = 0; i < broj_knjiga; i++)
      if(knjige[i]->DajKodKogaJe() == 0) cout << *knjige[i];
}
```

Slijedi implementacija preostalih elemenata klase:

```
void Biblioteka::IspisiZauzete() const {
      for(int i = 0; i < broj_knjiga; i++)
      if(!*knjige[i]) cout << *knjige[i];
}
```

```
void Biblioteka::IspisiNaCitanjuDuzeOd(int br_dana) const {
      for(int i = 0; i < broj_knjiga; i++)
      if(knjige[i]->DajKolikoJeDugoNaCitanju() > br_dana) cout <<
      *knjige[i];
}

static bool Kriterij(const Knjiga *k1, const Knjiga *k2) {
      if(k1->DajKolikoJeDugoNaCitanju() != k2
      >DajKolikoJeDugoNaCitanju())
      return k1->DajKolikoJeDugoNaCitanju() > k2-
      >DajKolikoJeDugoNaCitanju();
      return k1->DajNaslov() < k2->DajNaslov();
}

Knjiga &Biblioteka::operator [](int ev_broj) {
      for(int i = 0; i < broj_knjiga; i++)
      if(knjige[i]->DajEvidencijskiBroj() == ev_broj) return
      *knjige[i];
      throw "Nema knjige sa traženim evidencijskim brojem!";
}

Knjiga Biblioteka::operator [](int ev_broj) const {
      for(int i = 0; i < broj_knjiga; i++)
      if(knjige[i]->DajEvidencijskiBroj() == ev_broj) return
      *knjige[i];
      throw "Nema knjige sa traženim evidencijskim brojem!";
}

Biblioteka &Biblioteka::operator ++() {
      for(int i = 0; i < broj_knjiga; i++)
      if(!*knjige[i]) ++(*knjige[i]);
      return *this;
}
```

*Preostaje još glavni program. Postoji mnogo načina da se izvede ono što je traženo od glavnog programa, a ovdje prikazano rješenje je vjerovatno najkraće i logički najelegantnije. S obzirom da nam nije poznato koliko datoteka "*KNJIGE.TXT*" sadrži knjiga, kapacitet biblioteke je postavljen na 1000 knjiga. Bolje rješenje bilo bi da se prvo izvrši jedan prolaz kroz datoteku da se utvrdi koliko ona sadrži knjiga pa da se tek onda kreira biblioteka i u drugom prolazu kroz istu datoteku izvrši stvarno čitanje knjiga. Mada su neki studenti uočili ovu činjenicu, njeno rješenje se ne traži da bi se zadatak prihvatio kao korektan, tako da će u prikazanom rješenju ova činjenica biti zanemarena. Primijetimo da je upotreba funkcije "*getline*" neophodna da bi se pročitala čitava rečenica. Također, treba obratiti pažnju na upotrebu manipulatora "*ws*":*

```
int main() {
      Biblioteka biblioteka(1000);
      ifstream knjige("KNJIGE.TXT");
      for(;;) {
      bool da_li_je_udzbenik(false);
      if(knjige.peek() != 'U') {
      da_li_je_udzbenik = true;
```

```
        knjige.get();
        }
        int ev_broj, god_izdanja;
        string naslov, pisac, zanr, predmet;
        knjige >> ev_broj >> ws;
        if(!knjige) break;
        getline(knjige, naslov); getline(knjige, pisac);
        getline(knjige, zanr);
        knjige >> god_izdanja >> ws;
        if(da_li_je_udzbenik) {
        getline(knjige, predmet);
        biblioteka.KreirajUdzbenik(ev_broj, naslov, pisac, zanr,
        god_izdanja,
        predmet);
        }
        else biblioteka.KreirajKnjigu(ev_broj, naslov, pisac, zanr,
        god_izdanja);
        }
        ifstream zaduzenja("ZADUZENJA.TXT");
        int ev_broj, clanski_broj;
        char znak;
        while(zaduzenja >> ev_broj >> znak >> clanski_broj)
        biblioteka.ZaduziKnjigu(ev_broj, clanski_broj);
        cout << "Spisak zaduženih knjiga:\n";
        biblioteka.IspisiZauzete();
        cout << "Spisak slobodnih knjiga:\n";
        biblioteka.IspisiSlobodne();
        return 0;
}
```