

Zbirka riješenih ispitnih zadataka

iz programiranja jezikom

C++

Autor:
Marjan Sikora

SPLIT, 2015.

Ovo je reorganizirana i proširena zbirka zadataka. Rad na zbirci je još u tijeku, te nije do kraja provjerena i ispravljena. Molim studente koji uoče greške ili nelogičnosti da mi ih dojavе na mail: sikora@fesb.hr

Marjan Sikora

U Splitu, 30.1.2015.

Ova zbirka sadrži riješene zadatke sa kolokvija i ispita kolegija Objektno orijentirano programiranje, koji se predaje na FESB-u, smjerovi 111, 112, 114, 910, 920, 940 i 120. U ovom kolegiju se uči programiranje u jeziku C++, tako da zbirka može poslužiti i svima drugima koji uče programirati tim jezikom.

Zbirka je podijeljena u dva dijela. U prvom dijelu su zadatci, a u drugom rješenja zadataka. Zadaci i rješenja zadataka sadrže one sa kolokvija, te one sa ispita. Redoslijed zadataka prati redoslijed kojim se oni pojavljuju na kolokvijima i ispitima, a slaže se i sa redoslijedom gradiva koje se uči na ovom kolegiju. Zbirka je predviđena kao popratno sredstvo uz skriptu prof. Ive Mateljana "C++", koja je udžbenik za ovaj kolegij.

Onima koji koriste ovu zbirku preporučam zadatke rješavati na računalu, te što više koristiti izvršenje koda liniju po liniju i praćenje varijabli. Ukoliko zadatak podrazumijeva izradu samo dijela koda, jedne funkcije ili klase, preporučam da se potrudе i naprave cijeli kod, zajedno sa `main` funkcijom kako bi mogli u potpunosti ispitati i shvatiti rad koda. Takav kod neka studenti pokrenu u razvojnoj okolini, te po mogućnosti izvrše liniju po liniju, uz praćenje varijabli.

Zadaci u zbirci testirani su u razvojnoj okolini Microsoft Visual Studio 2013, ali su pisani u standardnom jeziku C++, pa ih možete pokrenuti i sa drugim razvojnim okolinama, odnosno kompajlerima. Ukoliko u zbirci naidete na kakovu grešku, molim vas da na email sikora@fesb.hr pošaljete poruku sa opisom o čemu je riječ, kako bih to mogao ispraviti. Na taj način ćete pomoći budućim generacijama studenata.

Također zahvaljujem kolegama Anni Šušnjari, Filipu Maretiću, Lovri Šušnjari, Anti Radi i Josipu Vukoviću na pomoći prilikom izrade ove zbirke zadataka.

Marjan Sikora

U Splitu, 7.10.2014.

SADRŽAJ:

Zbirka riješenih ispitnih zadataka	1
iz programiranja jezikom	1
C++.....	1
SADRŽAJ:	3
1. ZADACI.....	4
1.1 Operatori, algoritamska struktura programa.....	4
1.2 Prijenos argumenata u funkciju	6
1.3 Stringovi	11
1.4 Klase i objekti.....	12
1.5 Datoteke	21
1.6 Rekurzija	25
1.7 Nasljeđivanje	25
1.8 Generičko programiranje.....	29
2. RJEŠENJA	36

1. ZADACI

1.1 Algoritamska struktura programa i operatori

1) Koji tip imaju sljedeći izrazi:

```
char c; int i; float f;  
a) c + 8.3  
b) int(f) + c  
c) 3 * f  
d) float(i) * 5.1
```

2) Napišite koji ispis generira program:

```
#include <iostream>  
int main() {  
    int a = 2, b = 2;  
    std::cout << "The value of c is: " << ( (a>b)? a:b );  
    return 0;  
}
```

3) Što će biti ispisano po izvršenju sljedećeg programa?

```
#include <iostream>  
#define LI i  
#define IMATE "Imam "  
#define KUNU " dolara!\n"  
  
int main() {  
    int i = 100;  
    std::cout << IMATE << LI << KUNU;  
    return 0;  
}
```

4) Napišite što će biti ispisano nakon izvršenja ovog programa:

```
#include <iostream>  
using namespace std;  
int y = 0;  
int main() {  
    if(y) {  
        int y = 5;  
        cout << "y = " << y << endl;  
    } else {  
        y++;  
        cout << "y = " << y-- << endl;  
    }  
    cout << "y = " << y;  
  
    return 0;  
}
```

5) Napišite što će biti ispisano nakon izvršenja sljedećeg koda:

```
#include <iostream>
using namespace std;

int main() {
    int y = 0;

    cout << y++ << endl;
    if(y) {
        int y = 5;
        cout << y << endl;
    }
    cout << y << endl;

    return 0;
}
```

6) Napišite koji ispis generira sljedeći program:

```
#include <iostream>
using namespace std;

int main() {
    int a = 3, b = 1, c;

    while( c = (a-b) ) {
        cout << "Vrijednost od c iznosi: " << c << endl;
        b++;
    }

    return 0;
}
```

7) Petlju:

```
for(i=0, k=0; i<10; k++) {
    c = k*i; i++;
}
```

zapišite pomoću ekvivalentne `while` petlje.

8) Zadana je `while` petlja kojom se računa suma niza od `N` brojeva:

```
int a[N];
int i = 0, sum = 0;
while(i < N) {
    sum += a[i];
    i++;
}
```

Napišite ekvivalentni program u kojem se umjesto `while`-petlje koristi `for`-petlja.

1.2 Prijenos argumenata u funkciju

9) Napišite koji ispis generira sljedeći program:

```
#include <iostream>
using namespace std;

void Inc(int& x)          { x += 1; }
void Reset(int x, int& y) { y = x; }

int main() {
    int a=1, b=1;

    cout << "a + b = " << a + b << endl;
    Inc(b);
    cout << a << " " << b << endl;
    Reset(b, a);
    cout << a << " " << b << endl;
    return 0;
}
```

10) Napišite koji ispis daje sljedeći program:

```
#include <iostream>
using namespace std;

int  GetPlus(int x)          { return x+1;}
void Reset(int& y)           { y = 4; }
void Combine(int x, int& y) { y = x+1; }

int main() {
    int x=1, y=1;
    x = GetPlus(y);  cout << x << " " << y << endl;
    Reset(y);        cout << x << " " << y << endl;
    Combine(y, x);   cout << x << " " << y << endl;
    return 0;
}
```

11) Napišite koji ispis generira sljedeći program:

```
#include <iostream>
using namespace std;

int FunctionByValue(int i) { return ++i; }
int FunctionByRef(int& ir) { return ++ir; }
int FunctionByAddr(int* ia) { return ++(*ia); }

int main() {
    int i = 0;          cout << i << endl;

    FunctionByRef(i);    cout << i << endl;
    FunctionByValue(i);  cout << i << endl;
    FunctionByAddr(&i);  cout << i << endl;

    return 0;
}
```

```
}
```

12) Napišite koji ispis generira sljedeći program:

```
#include <iostream>
using namespace std;

int funcA(int x) {
    x += 7;
    return x;
}

void funcB(int& y) {
    y = 42;
}

void funcC(int z, int& w) {
    z = 0;
    w = 0;
}

int main() {
    int r = 13, s = 11;
    cout << r + s << " r + s" << endl;
    r = funcA(s);
    cout << r << " " << s << endl;
    funcB(s);
    cout << r << " " << s << endl;
    funcC(r, s);
    cout << r << " " << s << endl;

    return 0;
}
```

13) Napišite funkciju:

```
double PovrsinaZidova(double x, double z, double z)
```

kojoj su argumenti x, y i z dimenzije pravokutne sobe. Funkcija kao rezultat daje ukupnu površinu svih zidova.

Također napišite preopterećenu funkciju istog imena, koja površinu zidova neće vraćati sa return, već prijenosom argumenta po reference.

Napišite main funkciju sa primjerom poziva obje funkcije.

14) Napišite funkciju imena KvadratKub, koja računa kvadrat i kub broja. Prototip funkcije neka je:

```
void KvadratKub( double x, double& rKvadrat, double& rKub );
```

Funkcija prima broj koji treba potencirati putem varijable x, a kvadrat i kub vraća putem dvije reference rKvadrat i rKub. Napišite i glavni program u kojem ćete pomoću ove funkcije izračunati kvadrat i kub broja 3.0

15) Napišite funkciju:

```
void RadijaniUStupnjeve(double& kut)
```

Funkcija prima jedan argument: referencu varijable `kut`, koja sadrži vrijednost kuta u radijanima, a nakon završetka funkcije sadrži vrijednost kuta u stupnjevima. U glavnom programu sa tipkovnice učitajte jedan kut u radijanima, te pomoću funkcije ispišite njegovu vrijednost u stupnjevima.

16) Napišite funkciju `FormirajPismo` čiji je prototip:

```
void FormirajPismo(string ime, string magazin, int godina, int mjesec,
string& pismo);

// POST: formirano pismo se vraća pomoću stringa pismo
// Argumenti: ime - ime korisnika kojem se šalje pismo
//             magazin - ime časopisa kojem prestaje pretplata
//             godina - u kojoj prestaje preplata
//             mjesec - u kojem prestaje preplata
```

Primjerice, nakon poziva funkcije s argumentima:

```
FormirajPismo("Jure", "Nacional", 2003, 6, p):
```

u varijabli `p` treba biti pohranjen sljedeći tekst:

```
Dragi Jure, upozoravamo Vas da vasa preplata na magazin Nacional
prestaje vrijediti u mjesecu: Lipanj, 2003. Molimo, obnovite
preplatu!
```

17) Napišite funkciju:

```
void PovecajSmanjiZaDelta(int delta, int& x, int& y)
```

Koja prima tri argumenta: `delta` je cjelobrojna vrijednost za koju se povećava vrijednost varijable `x` i istovremeno smanjuje vrijednost varijable `y`.

18) Napišite što će biti ispisano nakon izvršenja sljedećeg koda koji koristi funkciju

`PovecajSmanjiZaDelta` iz prethodnog zadatka:

a)

```
int x=33, y=22;
for (int delta =0; delta <5; delta++) {
    PovecajSmanjiZaDelta (delta, x, y);
}
std::cout << "x=" << x<< endl<< "y="<< y<< endl;
```

b)

```
int x=33, y=22, delta = 3;
PovecajSmanjiZaDelta (++delta, x, y);
PovecajSmanjiZaDelta (delta++, y, x);
std::cout << "x=" << x<< endl<< "y="<< y<< endl;
```


19) Zadana je funkcija OpsegKrug:

```
void OpsegKrug(float r, float* O) {  
    *O = 2*r*3.14;  
}
```

Preradite funkciju tako da umjesto putem pokazivača argument `o` prebacuje putem reference.

20) Napišite funkciju prototipa:

```
abc(int& a, int& b)
```

Tu funkciju koristite na način kako je opisano u funkciji `main()`. Kod koji vrši funkcija `abc()` mora rezultirati prikazanim ispisom.

```
void abc(int& a, int& b) {  
    // napišite tijelo ove funkcije  
}  
  
int main() {  
    // ne mijenjajte funkciju main()  
  
    int a = 50, b = 12;  
    cout << "a = " << a << " " << "b = " << b << endl;  
    abc(a, b);  
    cout << "a = " << a << " " << "b = " << b << endl;  
  
    return 0;  
}
```

Ispis nakon izvršenja mora biti:

```
a = 50  b = 12  
a = 200 b = 36
```

21) Napišite dvije verzije funkcije:

```
double StrToDouble(char* s);  
double StrToDouble(string s);
```

pomoću kojeg se iz stringa `s`, koji sadrži niz numeričkih znakova, dobije ekvivalentna numerička realna vrijednost. Prvi oblik funkcije kao parametar prima pokazivač na niz znakova (ASCIIZ), a drugi oblik prima objekt tipa standardne klase `string`.

22) Napišite funkciju:

```
string DoubleToString(double num);
```

pomoću kojeg se iz broja `num` dobije formatirani `string`.

23) Napišite funkciju `ReverseString()`:

```
string ReverseString(string &Str);
```

Funkcija vraća string u kojem su znakovi u obrnutom redoslijedu u odnosu na ulazni string `Str`.

24) Napišite funkciju za ispis rimskih brojeva kojoj je specifikacija:

```
string RimskiBroj(int n)

// PRE: argument funkcije je cijeli broj n: 0 <= n <= 10
// POST: vraća string koji sadrži rimski zapis broja n
```

primjerice `cout << RimskiBroj(7) << endl;` dat će ispis VII. Napišite program u kojem se testira ova funkcija.

25) Napišite funkciju `PromptRange` kojoj je prototip:

```
int PromptRange(string poruka, int low, int high)
// pre: low <= high
// post: dobavlja vrijednost iz intervala [low.. high]
```

Funkcija prvo ispisuje poruku, a zatim korisnik unosi vrijednost. Ako je ta vrijednost iz intervala `[low.. high]` funkcija završava i vraća unesenu vrijednost. Ako ta vrijednost nije iz intervala `[low.. high]` ponovo se ispisuje poruka i čeka unos. Ako se primjerice funkcija pozove sa sljedećim argumentima:

```
int dan = PromptRange("Unesi broj dana u mjesecu", 1, 31);
cout << "Unesena vrijednost je:" << dan << endl;
```

treba se dobiti sljedeća interakcija s korisnikom:

```
Unesi broj dana u mjesecu (vrijednost izmedju 1 i 31): 44
Unesi broj dana u mjesecu (vrijednost izmedju 1 i 31): 4
Unesena vrijednost je: 4
```

26) Napišite funkciju kojoj je specifikacija:

```
string ZamijeniSlova(const string str)
/* Argument str je tipa klase string.
   Funkcija vraća string koji je jednak ulaznom argumentu,
   ali se mala slova pretvaraju u velika slova i obrnuto. */
```

Primjerice, nakon programskog odsječka:

```
#include <string>
...
String s = "Hello World!";
cout << ZamijeniSlova(s);
```

treba biti ispisano: `HELLO WORLD!`

27) Napišite funkciju `StringSpace`, kojoj je argument objekt tipa `string`, a vraća broj razmaka u njemu. Prototip funkcije je:

```
int StringSpace(string x);
```

1.3 Stringovi

28) Napišite program u kojem:

- a) korisnik unosi niz znakova u objekt tipa `string`
- b) određuje da li u unesenom stringu postoji string `"kraj"`
- c) ako postoji, program se prekida, a ako ne postoji ispisuje se uneseni `string` i ponavlja korak a)

29) Napišite funkciju kojoj je specifikacija:

```
string ZamijeniRazmakePodvlakom(const string str)
/* Argument str je tipa klase string.
 * Funkcija vraća string koji je jednak ulaznom argumentu ali se
 * umjesto jednog ili više uzastopnih znaka razmaka zapisuje znak
 * podvlake. */
```

Primjerice, nakon programskog odsječka:

```
#include <string>
...
string s = " Hello World .";
cout << ZamijeniRazmakePodvlakom(s);
```

treba biti ispisano:

```
_Hello_World_.
```

30) Napišite koji ispis na ekranu nastaje nakon izvršenja segmenta programa:

```
string word;
word = "predmet";
int num = 3;

cout << "1) " << num / 2 << "+" << num % 2 << endl;
cout << "2) " << 9 << "9" << 6 + num << endl;
cout << "3) " << word.substr(word.find("re"),3) << "ov" << endl;
```

31) Napišite kakav će biti ispis slijedećeg dijela koda:

```
#define PRVI "pljesak"
...
string word = "Prvi pljesak!";
cout << 31 % 4 * 2 << endl;
cout << word.substr( word.find( "Prvi" ), word.length()/2 ) << endl;
```

32) Napišite koji ispis na ekranu nastaje nakon izvršenja segmenta programa:

```
string word;
word = "omiljeni predmet";
int num = 11;

cout << num/3 << "+" << num % 7 << endl;
cout << 11 << "111" << '1' << '1' << endl;
cout << word.substr(word.find("pr"),4) << "ivo" << endl;
```

33) Napišite koji ispis na ekranu nastaje nakon izvršenja segmenta programa:

```
string word1 = "predmet";
string word2 = "redar";

cout << word1.substr(word1.find("re"),3)
    << " "
    << word2.substr(word2.find("re"),3) << endl;
```

34) Napišite tri verzije funkcije ToString:

```
string ToString(int x);
string ToString(double x);
string ToString(bool b);
```

Funkcija vraća string koji predstavlja literarni zapis cjelobrojne vrijednosti, realne vrijednosti i logičke vrijednosti.

35) Napišite funkciju koja decimalni broj pretvara u string po hrvatskom standardu. Takav string mora imati decimalni zarez umjesto decimalne točke. Specifikacija funkcije je:

```
string HrvBroj(float x)
// PRE: argument funkcije je decimalni broj n
// POST: vraća string koji sadrži znamenke broja n i decimalni zarez
//       (decimalnu točku pretvori u zarez)
```

primjerice `cout << HrvBroj(7.8) << endl;` dat će ispis 7,9

1.4 Klase i objekti

36) Napišite klasu imena `Registracija`, koja predstavlja registraciju automobila u BiH. Klasa neka ima dvije javne varijable tipa `int`, imena `reg1` i `reg2`, koje sadrže dva troznamenasta broja koji čine registraciju.

Npr: ako je `reg1=143`, a `reg2=186`, onda je registracija “143-186”.

Klasa neka ima default konstruktor koji i `reg1` i `reg2` postavlja na vrijednost 100. U glavnom programu deklarirajte objekt imena `r` klase `Registracija`, čija vrijednost neka bude “534-234”. Ispišite registraciju pohranjenu u objektu `r`, ispravno formatiranu.

37) Promijenite klasu `Registracija` tako da `reg1` i `reg2` budu privatne varijable. Napravite dvije funkcije:

- funkciju `Postavi` koja postavlja vrijednost registracije. Prototip neka je:

```
void Postavi( int temp1, int temp2);
```

 Funkcija `Postavi` neka članske varijable `reg1` i `reg2` postavi na vrijednosti `temp1` i `temp2`, ukoliko su `temp1` i `temp2` pozitivni troznamenkasti brojevi. Inače neka ne mijenja `reg1` i `reg2`.
- funkciju `Ispisi` koja neka ispisuje registraciju na ekran (formatirano, sa povlakom između `reg1` i `reg2`). Prototip neka je:

```
void Ispisi();
```

Promijenite glavnu funkciju iz prošlog zadatka, u skladu s promjenama javnosti varijabli i novim funkcijama.

38) Napisati specifikaciju i implementaciju klase `kocka` (deklarirati i definirati klasu koja opisuje kocku). Svi podatkovni članovi klase trebaju biti deklarirani kao javni članovi (`private`). Objekti klase `kocka` trebaju imati integriranu funkcionalnost proračuna volumena i oplošja kocke (funkcijske članove za proračun volumena i oplošja kocke).

39) Implementirajte klasu koja predstavlja adresu, prema sljedećoj specifikaciji:

```
konstruktor:
    inicira podatke o adresi: ulica, grad, broj, drzava
pristupnici:
    get_ulica   vraća string koji sadrži podatak o ulici
    get_grad    vraća string koji sadrži podatak o gradu
    get_broj    vraća cijeli broj koji sadrži postanski broj
    get_drzava  vraća string koji sadrži podatak o državi
mutatori:
    set_ulica   postavlja string koji sadrži podatak o ulici
    set_grad    postavlja string koji sadrži podatak o gradu
    set_broj    postavlja cijeli broj koji sadrži postanski broj
    set_drzava  postavlja string koji sadrži podatak o državi
ispis:
    Definirajte operator >> za čitanje adrese s ulaznog toka
```

40) Deklarirana je klasa `Razlomak`, kojom se definira objekt razlomka određen cjelobrojnim brojnikom i nazivnikom:

```
class Razlomak {
    int m_brojnik;
    int m_nazivnik;
public:
    Razlomak(): m_brojnik(0), m_nazivnik(1) {}
    Razlomak(int brojnik, int nazivnik = 1)
        { m_brojnik = brojnik; m_nazivnik = nazivnik; }

    // pristupnici
    int Brojnik() const ; /*definiraj*/
    int Nazivnik() const { return m_nazivnik; }
```

```

        // mutatori
        void Brojnik(int br); /*definiraj*/
        void Nazivnik(int naz) { m_nazivnik = naz; }
};

// operacije s razlomcima

Razlomak operator* (const Razlomak& r, const Razlomak& s) {
    int a = r.Brojnik();
    int b = r.Nazivnik();
    int c = s.Brojnik();
    int d = s.Nazivnik();
    return Razlomak(a*c, b*d);
}

Razlomak operator + (const Razlomak& r, const Razlomak& s);
Razlomak operator - (const Razlomak& r, const Razlomak& s);

```

Koristeći ovu deklaraciju klase Razlomak,

- Definiraj funkcije Brojnik, kojima se postavlja i dobavlja vrijednost brojnika.
- Definiraj operator+ i operator- za klasu Razlomak.
- Napišite program u kojem su definirani razlomci:

```

Razlomak r1(3,4); Razlomak r2(7,3);
Razlomak zbroj, razlika;

```

a program računa i ispisuje zbroj i razliku razlomaka r1 i r2. Operacije zbrajanja i oduzimanja treba izvršiti pomoću prethodno definiranih operatora + i -.

41) Za klasu Razlomak iz prethodnog zadatka definiraj operatore za ispis i unos razlomka u obliku a/b:

```

ostream& operator << (ostream& out, const Razlomak& s);
istream& operator >> (istream& in, Razlomak& r);

```

tako da se programom:

```

int main() {
    Razlomak r, s;

    cout << "Unesi razlomak u obliku a/b: " << flush;
    cin >> r;
    cout << "Unesi razlomak u obliku a/b: " << flush;
    cin >> s;

    Razlomak suma = r + s;
    Razlomak produkt = r * s;

    cout << r << " + " << s << " = " << suma << endl;
    cout << r << " * " << s << " = " << produkt << endl;

    return 0;
}

```

dobije ispis:

Unesi razlomak u obliku a/b: 3/4
 Unesi razlomak u obliku a/b: 7/3
 $3/4 + 7/3 = 37/12$
 $3/4 * 7/3 = 21/12$

42) Zadana je klasa Student :

```
class Student {
public:
    string m_broj;        // maticni broj
    string m_ime;         // ime
    string m_prezime;     // prezime
    Student() { }
    Student(string& mb, string& ime, string& pr)
        : m_broj(mb), m_ime(ime), m_prezime(pr), { }
};
```

Napišite implementaciju sljedećih operatora:

```
ostream& operator << (ostream &out, const Student& s)
{...}

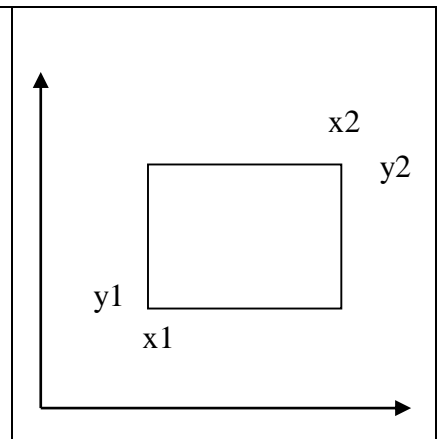
Student& operator = (const Student& d)
{...}
```

43) Deklarirana je klasa Pravokutnik

```
class Pravokutnik {
public:
    Pravokutnik ();
    Pravokutnik& operator= (const Pravokutnik& p);

    int GetSirina();
    int GetVisina();

public:
    int x1, y1, x2 , y2;
};
```



Koristeći ovu deklaraciju:

- napišite konstruktor, kojim se ujedno inicijaliziraju članovi: $x1=y1=0$ i $x2=y2=1$.
- definiraj operator= za klasu Pravokutnik.
- definiraj funkcije GetSirina() i GetVisina(), koje vraćaju širinu i visinu pravokutnika.

44) Napišite implementaciju operatora >> klasu Pravokutnik iz prethodnog zadatka, prema sljedećoj deklaraciji:

```
istream& operator >> (istream& s, Pravokutnik& c1);
```

Operator `>>` unosi koordinate pravokutnika `x1`, `y1`, `x2` i `y2` u objekt klase `Pravokutnik`.

45) Deklarirana je klasa `PointR` kojom se bilježi položaj točke u cilindričnom koordinatnom sustavu (pomoću udaljenosti od ishodišta `m_r` i kuta `m_fi` – u radijanima):

```
class PointR {
public:
    PointR();
    void SetXYPosition(float x, float y);
    void GetXYPosition(float& x, float& y);
    float GetAngle() const {return m_fi;}
    float GetRadius() const {return m_r;}
private:
    float m_r;
    float m_fi;
};
```

Koristeći ovu deklaraciju

- Definirajte konstruktor, kojim se ujedno inicijalizira članove `m_r` (radijus) i `m_fi` (kut) na vrijednost 0.
- Definirajte kopirni konstruktor

Definirajte funkcije `SetXYPosition` i `GetXYPosition`, kojima se postavlja i dobavlja pozicija točke u Kartezijusovom koordinatnom sustavu iz varijabli `x` i `y`.

46) Deklarirana je klasa `PointR` kojom se bilježi položaj točke u cilindričnom koordinatnom sustavu (pomoću udaljenosti od ishodišta `m_r` i kuta `m_fi` – u radijanima):

```
class PointR {
public:
    PointR();
    void SetXYPosition(float x, float y);
    void GetXYPosition(float& x, float& y);
    float GetAngle() const {return m_fi;}
    float GetRadius() const {return m_r;}
private:
    float m_r;
    float m_fi;
};
```

- Definirajte operator `=` za klasu `PointR`.
- Definirajte operator `<<` za klasu `ostream` i objekt klase `PointR`. Njegovo je djelovanje da se ispiše udaljenost od ishodišta i kut u stupnjevima, a deklaracija funkcije je:

```
ostream& operator << (ostream& out, const PointR& p);
```

47) Zadana je klasa `CUcenik`:

```
class CUcenik {
    string m_ime;           // ime i prezime
    int m_ocjena;           // ocjena
```



```

public:
    CUcenik();
    int GetOcjena();
    string GetIme();
    void SetOcjena(int ocjena);
    void SetIme(string ime);
};

```

Napišite:

- konstruktor, koji ime postavlja na prazan string, a ocjenu na 0
- implementaciju operatora >> koji sa toka in čita najprije string m_ime, a potom cijeli broj m_ocjena:

```

istream& operator >> (istream& in, CUcenik& u)
{...}

```

48) Zadana je klasa `Complex`, pomoću koje se manipulira kompleksnim brojem;

```

class Complex {
public:

    double real;
    double imag;
    Complex(): real(0), imag(0) {};
    Complex(const Complex&);

    double Mag() const;
    double Phase() const;

};

```

Definirajte kopirni konstruktor klase, koji služi za deklariranje kompleksnog broja pomoću postojećeg objekta tipa `Complex`:

```

Complex a;
Complex b(a);

```

Definirajte funkcije:

```

double Complex::Mag() const
    // vraća magnitudu polarnog prikaza kompleksnog broja.
double Complex::Phase() const
    // vraća fazu polarnog prikaza kompleksnog broja.

```

49) Koristeći deklaraciju klase `Complex` iz prethodnog zadatka:

- Definirajte operator `=` za klasu `Complex`.
- Definirajte operator `<<` za klasu `ostream` i objekt klase `Complex`.

```

ostream& operator << (ostream& out, const Complex& x);

```

Njegovo je djelovanje da se ispiše kompleksni broj u obliku:

(realni dio) \pm j (imaginarni dio)

50) Zadana je klasa:

```
class Prekidac {
    bool m_iskljucen;
public:
    Prekidac();
    void Ukljuci();
    void Iskljuci();
    bool Ukljucen();
};
```

a) Definirajte konstruktor klase, kojim se stanje `m_iskljucen` prekidača postavlja u `true`.

b) Definirajte funkcije:

```
void Ukljuci();    - postavlja prekidač u stanje da nije isključen
void Iskljuci();   - postavlja prekidač u stanje da je isključen
bool Ukljucen();   - vraća true ako je prekidač uključen, inače vraća false
```

51) Koristeći deklaraciju klase `Prekidac` iz prethodnog zadatka:

a) Definiraj `operator==` za klasu `Prekidac`.

b) Definiraj kopirni konstruktor klase `Prekidac`

c) Definirajte `operator <<` za klasu `ostream` i objekt klase `Prekidac`. Njegovo je djelovanje da se ispiše `ukljucen` ili `iskljucen` ovisno o tome dali je prekidač uključen ili isključen.

```
ostream& operator << (ostream& out, const Prekidac& s);
```

52) Napišite klasu `Loto` na način da se niz brojeva alocira dinamički. Alokaciju izvršite u konstruktoru. Definirajte i pripadni destruktor klase, u kojem se oslobađa alocirana memorija. Koristite sljedeću shemu klase:

```
class Loto {
    int *m_br;
    int m_rezervni_broj;
public:
    Loto() {
        /*Alociraj niz m_br*/
        PostaviBrojeve();
    }
    ~Loto() { /*Dealociraj niz m_br*/ }

    void PostaviBrojeve();           // postavi brojeve na
                                    // nasumične vrijednosti
    int *DobaviNizBrojeva() const;   // vrati pokazivač na niz
    int DobaviRezervniBroj() const;   // vrati rezervni broj
    // const iza funkcije znači da ove dvije funkcije ne mijenjaju
    // članske varijable klase
};
```

- 53)** Napišite definiciju za operator `<<` kojim se ispisuje sadržaj objekta klase `Loto` u obliku liste 6 brojeva odvojenih zarezom, a kraju se u zagradama ispisuje rezervni broj.
Primjer ispisa:

```
22, 37, 2, 3, 6, 12 (4)
```

Deklaracija operatora je:

```
ostream& operator << (ostream& s, const Loto& L)
```

Primijetite da su članske funkcije klase `Loto` privatne.

- 54)** Deklarirana je klasa `SimpleCircle`:

```
class SimpleCircle {
public:
    SimpleCircle();
    void SetRadius(int);
    int GetRadius();

private:
    int m_radius;
};
```

Koristeći ovu deklaraciju:

- napišite konstruktor, kojim se ujedno inicijalizira član `m_radius` na vrijednost 5
- Definiraj operator `=` za klasu `SimpleCircle`
- Definiraj kopirni konstruktor klase `SimpleCircle`
- Definiraj funkcije `SetRadius` i `GetRadius` kojima se postavlja i dobavlja vrijednost članske varijable `m_radius`

- 55)** Deklarirana je klasa `Point2D` kojom se opisuje položaj točke u 2D prostoru (x,y):

```
class Point2D {
public:
    Point2D();
    void SetX(double x);
    void SetY(double y);
    double GetX();
    double GetY();
protected:
    double m_x, m_y;
};
```

Koristeći ovu deklaraciju:

- napišite konstruktor, kojim se ujedno inicijalizira položaj točke u `x=0` i `y=0`
- Definiraj funkcije `Set()` i `Get()`
- Definiraj kopirni konstruktor klase `Point2D`

56) Koristeći klasu `Point2D` i pravila nasljeđivanja definirajte klasu `Point3D`, pomoću koje se opisuje položaj točke u trodimenzionalnom prostoru (x,y,z).

Definirajte:

- a) potrebne varijable (osim onih koje se nasljeđuju)
- b) potrebne `public` metode kojima se mijenja i očitava položaj točke.
- c) člansku funkciju `Distance()`, koja vraća udaljenost točke od ishodišta.
- d) kopirni konstruktor klase `Point3D`

57) Deklarirana je klasa `Circle`, kojom se bilježi radijus kružnice:

```
class Circle {
public:
    Circle() : m_radius(0) {}
    void SetRadius(int r) {m_radius = r;};
    int GetRadius() {return m_radius;};

    Circle& operator++();           //prefiks
    Circle operator++(int unused);  //postfiks

protected:
    int m_radius;
};
bool operator == (Circle &c1, Circle & c2);
```

Koristeći ovu deklaraciju:

- a) Napišite implementaciju operatora `++` za klasu `Circle`
- b) Napišite implementaciju operatora `==`

58) Napišite implementaciju operatora `>>` i `<<` za klasu `Circle`, prema sljedećoj deklaraciji:

```
ostream & operator << (ostream& s, const Circle& c1);
istream & operator >> (istream& s, Circle& c1);
```

Ispisuje se (ili unosi) radijus kruga.

59) Deklarirana je klasa `HorLine` kojom se bilježi obojana horizontalna linija:

```
class HorLine {
public:
    HorLine() : m_HorLine (0), m_color (0){}

    void SetHorLine (double d);
    double GetHorLine();
    void SetColor (COLORREF c);
    COLORREF GetColor();

    HorLine& operator++();           // prefiks
    HorLine operator++(int unused);  // postfiks

    HorLine& operator=(const HorLine& d);
```

```

        protected:
            double m_HorLine;           // duljina linije
            COLORREF m_color;           // boja linije
    };

```

Koristeći ovu deklaraciju:

- Napišite implementaciju za prefiks i postfiks `operator++` za klasu `HorLine` (uvećava duljinu linije za 1.0).
- Napišite implementaciju za `operator =`

60) Napravite slijedeće:

- Za klasu `HorLine` iz prethodnog zadatka napišite funkcije `SetColor` i `GetColor`, te `SetHorLine` i `GetHorLine`
- Napišite klasu `PosHorLine`, koja predstavlja pozicioniranu horizontalnu liniju. Za definiciju kase koristite nasljeđivanje od klase `HorLine`. Klasa `PosHorLine` neka ima definiranu poziciju i to kao dvije privatne varijable `m_x` i `m_y`, tipa `double`
- Nadopunite konstruktor naslijeđene klase tako da postavlja obje koordinate na nulu.

61) Zadana je klasa `Tocka`:

```

class Tocka {
    protected:
        int m_x;
        int m_y;
    public:
        Tocka();
        int GetX() const {return m_x;}
        int GetY() const {return m_y;}
        void SetX(int x) {m_x = x;}
        void SetY(int x) {m_y = y;}
};

```

- Definirajte `operator==`

```
bool operator== (const Tocka& a, const Tocka& b);
```

- Definirajte `operator<<`

```
ostream& operator<< (ostream& out, const Tocka& t);
```

1.5 Dinamička alokacija memorije

62) Klasa `Registracija` zadana je na slijedeći način:

```

class Registracija {
    private:
        int reg1;
        int reg2;
    public:
        Registracija();

```

```

        void Postavi(int temp1, int temp2);
        void Ispisi();
};

```

Napišite program kojim se radi sljedeće:

- a) dinamički alocira niz od deset objekata klase `Registracija`
- b) postavlja prvu registraciju na vrijednost 123-456
- c) ispisuje prvu registraciju
- b) dealocira niz

1.6 Datoteke

63) Napišite program kojim se iz tekstualne datoteke imena "data.txt" očitavaju znakovi po sistemu "znak po znak". Program neka na ekranu ispisuje samo one znakove kojima započinje svaki redak teksta .

64) U memoriji su podaci spremljeni u nizu deklariranom s:

```
float A[1000];
```

Napišite programski odsječak u kojem prvo otvara binarna datoteka imena "brojevi.bin", zatim se podaci iz niza `A[]` spremaju u tu datoteku u binarnom obliku i na kraju se zatvara datoteka. Koristite klasu `ofstream`.

65) U memoriji su podaci spremljeni u nizu objekata klase `Pravokutnik` deklariranom s:

```
Pravokutnik Pr[1000];
```

Napišite programski odsječak u kojem prvo otvara binarna datoteka imena "pr.bin", zatim se podaci iz niza `Pr[]` spremaju u tu datoteku u binarnom obliku i na kraju se zatvara datoteka. Koristite klasu `ofstream`.

66) U memoriji su objekti klase `Complex` iz jednog od prethodnih zadataka spremljeni u nizu deklariranom s:

```
Complex A[1000];
```

Napišite programski odsječak u kojem se:

- a) otvara tekstualna datoteka imena "kbrojevi.txt"
- b) zatim se podaci iz niza `A[]` spremaju u tu datoteku, tako da u svakoj liniji teksta budu dva broja - prvi predstavlja realni, a drugi predstavlja imaginarni dio kompleksnog broja
- c) na kraju se zatvara datoteka

67) Otvorite datoteku imena "test.dat". Očitajte prva 3 znaka iz te datoteke. Ako ti znakovi su redom: 'a', 'b' i 'x', ispišite poruku da je prepoznat tip datoteke, inače ispišite da je to nepoznat tip datoteke. Na kraju zatvorite datoteku.

68) Napišite program kojim se analizira sadržaj neke tekstualne datoteke. Program izvještava:

- a) koliko je znakova
- b) koliko je riječi
- c) koliko je redaka teksta

zapisano u toj datoteci.

69) Napisati program koji sa tipkovnice čita 20 imena u niz varijabli tipa string. Program potom imena zapisuje u tekstualnu datoteku imena „20imena.txt.“. Imena se u datoteku zapisuju na način da se svakoj liniji nalazi uneseno ime, zatim razmak pa broj slova koje sadrži uneseno ime.

Primjer:

<i>korisnikov unos</i>	<i>sadržaj datoteke</i>
Branimir	Branimir 8
Ivan	Ivan 4
Goran	Goran 5
...	...

70) Podaci o studentima su zapisani u datoteci "student.txt" u obliku teksta koji sadrži podatke o matičnom broju, imenu i prezimenu studenta, primjerice:

```
110497339001 Ivo Banac
120397347001 Jure Topic
030497431001 Miljenko Vukas
011297432001 Ivan Tadin
```

Podatke za pojedinog studenta možemo prikazati klasom Student :

```
class Student {
public:
    string m_broj;    // maticni broj
    string m_ime;     // ime
    string m_prezime; // prezime
    Student() { }
    Student(string& mb, string& ime, string& pr)
        : m_broj(mb), m_ime(ime), m_prezime(pr), { };
};
```

Napišite funkciju CitajStudDatoteku kojom se učitavaju podaci o studentima iz toka input u strukturu koja je vektor objekata tipa Student. Prototip funkcije CitajStudDatoteku je:

```
void CitajStudDatoteku (ifstream& input, vector<Student>& spisak)
// PRED: datoteka otvorena za čitanje pomoću toka input
// POST: pročitani svi podaci u vektor spisak
```

Pazi: iz svake linije datoteke treba učitati tri stringa.

71) Napišite program imena `ncopy.c` kojim se sadržaj jedne tekstualne datoteke kopira u drugu datoteku, ali tako da se ispred svake linije zapiše redni broj linije. Ime izvorne i odredišne datoteke zadaje korisnik u komandnoj liniji.

Primjerice: Ako korisnik otkuca na komandnoj liniji:

```
ncopy dat1.txt dat2.txt
```

tada treba formirati datoteku `dat2.txt` koja će imati sadržaj od `dat1.txt` s označenim rednim brojem linije teksta.

72) Napišite program kojim se sadržaj jedne `.cpp` datoteke kopira u drugu datoteku `.cpp` datoteku, ali tako da se odstrane sve prazne linije i linije koje započinju znakom komentara `//` ispred kojeg može biti proizvoljan broj bijelih mjesta.

Primjerice, ako je u izvorna datoteka ima sadržaj:

```
if(p) node.next = p;
    // komentar

if(q) node.next = q;
```

tada u odredišnoj datoteci mora biti ispisano:

```
if(p) node.next = p;
if(q) node.next = q;
```

73) Napišite program kojim se s komandne linije dobavlja ime tekstualne datoteke. Očitavajte sve znakove iz te datoteke, te ih na ekranu ih ispišite bez razmaka i tako da u jednoj liniji ne bude više od 40 znakova. Na kraju zatvorite datoteku.

```
int main(int argc, char** argv)    // returns 1 on error
```

74) Deklarirana je klasa `PointR` kojom se bilježi položaj točke u cilindričnom koordinatnom sustavu (pomoću udaljenosti od ishodišta `m_r` i kuta `m_fi` – u radijanima):

```
class PointR {
public:
    PointR();
    void SetXYPosition(float x, float y);
    void GetXYPosition(float& x, float& y);
    float GetAngle() const {return m_fi;}
    float GetRadius() const {return m_r;}
private:
    float m_r;
    float m_fi;
};
```


U tekstualnoj datoteci imena „tocke.txt“ su upisane koordinate točaka u Kartezijevom koordinatnom sustavu – u svakom retku koordinata x i koordinata y razdvojene znakom razmaka. Napišite program kojim se iz te datoteke učitavaju koordinate u objekt tipa `PointR` i na ekranu se ispisuju za svaku točku udaljenost od ishodišta i kut u stupnjevima.

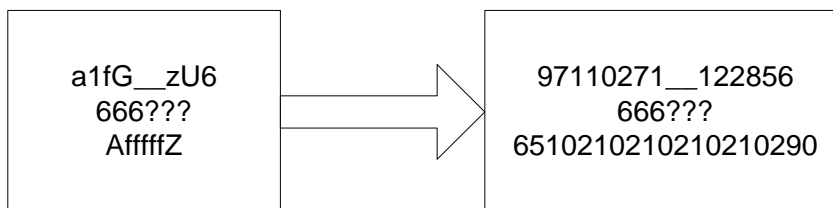
75) Napisati program kojim se sadržaj postojeće tekstualne datoteke kopira u novu tekstualnu datoteku na slijedeći način:

- sva slova (a-z i A-Z) iz postojeće datoteke se trebaju u novoj datoteci zapisati svojom ASCII vrijednosti
- svi ostali znakovi se ne mijenjaju

Primjer:

ORIGINALNA DATOTEKA

NOVA KREIRANA DATOTEKA



1.7 Rekurzija

76) Napišite funkciju slijedećeg prototipa, kojim se rekurzivno generiraju binomni koeficijenti Pascalovog trokuta:

```
int P(int n , int k);
```

Binomni koeficijenti su definirani rekurzivno:

$$\begin{aligned}
 P(n, 0) &= 1 \text{ i } P(n, n) = 1 & , \text{ za } n \geq 0 \\
 P(n, k) &= P(n-1, k) + P(n-1, k-1) & , \text{ za } n > k > 0
 \end{aligned}$$

Dolje je prikazan manji dio Pascalovog trokuta. Svaki element trokuta je izračunat zbrajanjem dva elementa iznad njega.

```

      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
  
```

1.8 Nasljeđivanje

77) Deklarirana je klasa `Circle`, kojom se bilježi radijus kružnice:

```

class Circle {
public:
    Circle() : m_radius(0) {}
    void SetRadius(int r) {m_radius = r;};
    int GetRadius() {return m_radius;};

protected:
    int m_radius;
};

```

Napišite klasu `ColoredCircle` koja pored radijusa kružnice sadrži podatak o boji kružnice `m_color` tipa `int`. U definiranju klase `ColoredCircle` obvezno koristite mehanizam nasljeđivanja, tako da klasa `ColoredCircle` nasljeđuje klasu `Circle`. Za klasu `ColoredCircle` definirajte operator `=`.

78) Deklarirana je klasa `PozicioniraniString`:

```

class PozicioniraniString {
public:
    PozicioniraniString() : m_tekst(""), m_x(1), m_y(1) {}

    void SetTekst(string&);
    string& GetTekst();
    void SetX(int);
    int GetX();
    void SetY(int);
    int GetY();

private:
    int m_x, m_y;
    string m_tekst;
};

```

koja sadrži jedan `string` i njegove koordinate na ekranu, te klasa `Tocka`:

```

class Tocka {
protected:
    int m_x;
    int m_y;
public:
    Tocka();
    int GetX() const {return m_x;}
    int GetY() const {return m_y;}
    void SetX(int x) {m_x = x;}
    void SetY(int x) {m_y = y;}
};

```

Koristeći definiciju klase `Tocka`, definirajte klasu `PozicioniraniString` koristeći mehanizam nasljeđivanja od klase `Tocka`.

```

class PozicioniraniString : public Tocka {
public:
    // odredite potrebne javne članove
private:
    // odredite privatne članove

```

```
};
```

Definirajte sve članske funkcije klase i potrebne podatke.

79) Deklarirane su klase `Tocka` i `Krug`:

```
class Tocka
{
    int m_x, m_y;
public:
    Tocka() { m_x= 0; m_y=0;};
    void SetX(int);
    int GetX();
    void SetY(int);
    int GetY();
};
```

```
class Krug
{
    int m_radius, m_x, m_y;
public:
    Krug ();
    void SetRadius(int);
    int GetRadius();
    void SetX(int);
    int GetX();
    void SetY(int);
    int GetY();
};
```

Definirajte klasu `Krug1` koristeći mehanizam nasljeđivanja od klase `Tocka`, ali tako da na ima istu funkcionalnost kao klasa `Krug`.

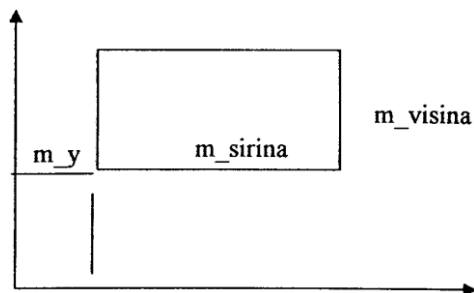
```
class Krug1 : public Tocka {
public:
    // odredite potrebne javne članove
private:
    // odredite privatne članove
};
```

80) Deklarirane su klase `Pravokutnik` i `Tocka`:

```
class Pravokutnik {
public:
    Pravokutnik ();

    void SetSirina(int);
    int GetSirina();
    void SetVisina(int);
    int GetVisina();
    int Povrsina();

    void SetX(int x) {m_x = x;};
    void SetY(int y) {m_y = y;};
    int GetX() {return m_x; };
    int GetY() {return m_y; };
private:
    int m_sirina;
    int m_visina;
    int m_x, m_y;
};
```



```
class Tocka {
public:
    Tocka (int x = 0, int y = 0): m_x(x), m_y(y) {};
    void SetX(int x) { m_x = x; };
    void SetY(int y) { m_y = y; };
};
```

```

        int GetX() { return m_x; };
        int GetY() { return m_y; };
protected:
        int m_x, m_y;
};

```

- Napišite klasu `Pravokutnik` tako da ona za opisivanje položaja pravokutnika koristi klasu `Točka` (Promišljaj: `Pravokutnik` "has a" `Točka`).
- Napišite klasu `Pravokutnik` tako da ona za opisivanje položaja pravokutnika nasljeđuje klasu `Točka` (Promišljaj: `Pravokutnik` "is a" `Točka`).

81) Deklarirana je klasa `Point2D` kojom se opisuje položaj točke u 2D prostoru (x,y):

```

class Point2D {
public:
    Point2D();
    void SetX(double x);
    void SetY(double y);
    double GetX();
    double GetY();
protected:
    double m_x, m_y;
};

```

Koristeći klasu i pravila nasljeđivanja definirajte klasu `Point3D`, pomoću koje se opisuje položaj točke u trodimenzionalnom prostoru (x,y,z).

Definirajte:

- potrebne varijable (osim onih koje se nasljeđuju)
- potrebne `public` metode kojima se mijenja i očitava položaj točke.
- konstruktor, kojim se ujedno inicijalizira položaj točke u $x=0$ i $y=0$, $z=0$.
- operator `==`
- operator `>>`

82) Zadane su klase `CUcenik` i `CKolokvij`:

```

class CUcenik {
    string m_ime;           // ime i prezime
    int m_ocjena;           // ocjena
public:
    CUcenik();
    int GetOcjena();
    string GetIme();
    void SetOcjena(int ocjena);
    void SetIme(string ime);
};

class CKolokvij {
public:
    int m_ocjena[3];
}

```

Koristeći nasljeđivanje klase `CUcenik`, definirana je klasa `CStudent`, kojom se pored ocjene studenta bilježi i rezultat na tri kolokvija.

```

class CStudent : public CUcenik {
    CKolokvij m_kolokvij;
public:
    CStudent();
    void SetOcjenaKolokvija(int brojKolokvija, int ocjena);
    int GetOcjenaKolokvija(int brojKolokvija);
};

```

- a) Napišite konstruktor koji neka ime postavlja na prazan string, a konačnu ocjenu studenta i sve ocjene kolokvija postavlja na nulu
- b) Definirajte operator <<:

```

ostream& operator << (ostream& out, const CStudent& s)
{...}

```

Kojim se redom ispisuje:

- ime
- sve ocjene na kolokvijima
- konačna ocjena

1.9 Generičko programiranje

- 83)** Napišite generički oblik klase `Circle` (tako da tip radijusa može biti bilo koji numerički tip):

```

template <class T> class Circle {...};

```

- 84)** Standardna klasa `vector<T>` služi kao spremnik za objekte tipa `T`.

- a) Napišite funkciju sljedećeg prototipa:

```

template <class T> int PronadjiElement(const vector<T> V, T x)

```

kojom se u spremniku `v` traži element `x`. Funkcija neka vraća indeks na kojem je pronađen element `x`, ili `-1` ako nije pronađen.

- b) Napišite funkciju sljedećeg prototipa:

```

template <class T> vector<T> SortirajVektor (const vector<T> V)

```

koja vraća sortirani vektor od vektora `v`, koji je argument funkcije.

- 85)** Napišite generičku funkciju:

```

typedef list<double> LIST;

void Pomnozi_elemente_liste( LIST& L, double multiplikator )

```

kojom se svi elementi liste `L` množe s vrijednošću parametra `multiplikator`.

86) Napišite generičku funkciju, kojom se iz jednog vektora `V` formiraju dva vektora: `Vshort` treba sadržavati stringove kraće od deset znakova, a vektor `Vlong` stringove dulje od deset znakova.

```
typedef vector<string> Vec;

void Razdijeli_vektor( Vec& V, Vec& Vshort, Vec& Vlong )
```

87) Napišite funkciju, kojom se iz jednog vektora `V` formiraju dva vektora: `V1` treba sadržavati neparne, a vektor `V2` parne elemente vektora:

```
typedef vector<int> Vec;

void Razdijeli_vektor( Vec& V, Vec& V1, Vec& V2)
```

88) Napišite funkciju kojom se sortira niz stringova:

```
void SortStrVector(vector<string>& strVec)
```

89) Zadana je klasa `Student`:

```
class Student {
    unsigned m_broj;    // maticni broj
    string m_prezime;   // ime
    string m_ime;       // ime
public:
    Student();
    void Set(unsigned broj, const string& prezime, const string& ime);
    void Get(unsigned& broj, string& prezime, string& ime) const;

    friend bool SpisakSortiran(const vector<Student>& spisak);
};
```

Napišite funkciju `SpisakSortiran()` kojom se određuje da li je spisak studenata leksikografski sortiran. Prototip funkcije je:

```
bool SpisakSortiran(const vector<Student>& spisak);
// PRED: spisak sadrži podatke o studentima
// POST: funkcija vraća true, ako je spisak leksikografski sortiran,
//       a ako nije vraća false
```

90) Pretpostavite da se podaci o grupi studenata nalaze u vektoru objekata klase `Student` iz jednog od prethodnih zadataka:

```
vector<Student> spisak;
```

Treba napisati funkciju `MatBroj()` kojom se dobavlja matični broj studenta kojem znamo prezime. Prototip funkcije je:

```
string MatBroj(const vector<Student>& spisak, const string& prezime);

// PRED: spisak sadrži podatke o studentima
// argumenti: spisak - sadrži podatke o studentima
//             prezime - prezime studenta kojem se traži matični broj
// POST: funkcija vraća matični broj studenta, ukoliko se prezime
//       nalazi u vektoru spisak, a ako ga nema vraća "00000"
```

91) Napišite dio programa u kojem se koristi niz stringova `strVec` deklariran s:

```
vector<string> strVec;
```

Program treba podržavati sljedeće radnje:

- Korisnik unosi više stringova u niz `strVec` (sve dok nije unesen string „kraj“)
- Zatim se stringovi ispišu obrnutim redom od reda kojim su uneseni (bez da se ispisuje string „kraj“)

92) Lista sadrži niz znakova. Napišite funkciju `IsSorted` koja vraća `true` ako su elementi liste sortirani uzlazno, inače vraća `false`.

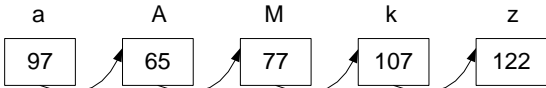
```
bool IsSorted(const list<char>& L)
// return true ako je lista znakova sortirana
```

93) Napisati definicije funkcije koja vraća listu elemenata tipa `int`. Kao argument funkcija prima jednu varijablu tipa `string` i zatim svaki znak stringa kopira u listu na način da u listu kopira njihove ASCII vrijednosti. Deklaracija funkcije je:

```
list<int> KopirajStringLista(string var);
```

Primjer:

KopirajStringLista("aAMkz");



vraća:

94) Napišite program kojim se s tipkovnice unosi niz brojeva. Unos završava kada se otkuca nula. Brojeve treba unositi u listu tako da se pozitivni brojevi ubacuju na početak liste, a negativni brojevi na kraj liste. Na kraju treba ispisati sumu pozitivnih i sumu negativnih brojeva. Kostur programa je:

```
list<double> L; // lista u kojoj pamtimo unesene brojeve

double x;      // broj koji se unosi
double sumapozitivnih, sumanegativnih;

// Ponavljač:
// 1. Dobavi broj x s tipkovnice
// 2. Ako je broj x jednak nuli
```

```
//          prekini unos
//      Ako je broj x pozitivan
//          dodaj njegovu vrijednost na početak liste L.
//      Ako je broj x negativan
//          dodaj njegovu vrijednost na kraj liste L.
// Sumiraj odvojeno sve pozitivne i sve negativne brojeve pa
// ispiši vrijednosti obe sume
```

95) Napišite funkciju čiji prototip glasi:

```
void SortListToVector(const list<double>& L, vector<double>& V)

// pre: lista L sadrži realne brojeve
// post: elementi liste se također nalaze u vektoru V sortirani
```

kojom se brojevi iz liste `L` prebacuju u vektor `V`, koji nakon izvršenja funkcije mora biti sortiran.

96) Deklaracijom klase `StringStog` izvršena je specifikacija za kontejner objekata tipa `string`. Objektima se pristupa po principu LIFO.

```
#include <vector>
#include <string>

using namespace std;

class StringStog {
public:
    void push(const string& str); // postavlja string str na vrh stoga
    void pop (string& str );    // skida string s vrha stoga u referencu str
    bool empty()const {return m_stog.empty();} // true ako je stog prazan
    int size()const {return m_stog.size();}    // vraća broj stringova na stogu

private:
    vector<string> m_stog;
};
```

- a) Implementirajte funkcije `push` i `pop`.
- b) Napišite operator `==`

97) Napišite program za testiranje klase `StringStog` iz prethodnog zadatka. Program treba obaviti sljedeće radnje:

- a) Izvijestiti korisnika da unese proizvoljan broj stringova, zaključno s stringom "kraj"
- b) Napisati petlju u kojoj se dobavljaju stringovi u kontejner tipa `StringStog`. Petlja se prekida kada korisnik unese string "kraj".
- c) Po završenom unosu:
 - Provjeri da li je stog prazan
 - Ako je stog prazan, ispisati poruku: "prazan"
 - inače, ispisati koliko je uneseno stringova i sadržaj stoga.

98) Napišite prijateljsku funkciju klase `StringStog` prototipa:

```
void SortStringStog(StringStog &s);
```

pomoću koje se može leksički sortirati sadržaj stoga `s` tako da na vrhu stoga bude string koji je posljednji po leksičkom redosljedu.

Napomena: deklaracijom prijateljske funkcije omogućuje se pristup privatnom članu `m_stog` koji je tipa vektor stringova i s kojim se može manipulirati kao s običnim nizovima (pristup elementima je pomoću cjelobrojnog indeksa).

99) Deklaracijom klase `RedStringova` izvršena je specifikacija za kontejner objekata tipa `string`. Objektima se pristupa po principu FIFO. Implementirajte funkcije `get`, `put` i `remove`.

```
#include <vector>
#include <string>

class RedStringova {
    vector<string> m_red;

public:
    void put(const string& str); // postavlja string str na kraj reda
    void remove();              // skida string s pocetka reda
    string get();                // vraca string s pocetka reda,
                                // ako je red prazan,
                                // vraca prazni string

    bool empty()const {return m_red.empty();} // true ako je red prazan
    int size()const {return m_red.size();}    // vraća broj stringova
};
```

100) Napišite program za testiranje klase `RedStringova` iz prethodnog zadatka. U program treba zapisati sljedeće radnje:

- Izvestiti korisnika da unese proizvoljan broj stringova, zaključno s stringom "kraj"
- Napisati petlju u kojoj se dobavljaju stringovi u kontejner tipa `RedStringova`
- Po završenom unosu provjeriti da li je `RedStringova` prazan.

Ako je stog prazan:

ispisati poruku: "nije izvršen unos"

inače:

ispisati stringove redom kojim su uneseni

101) Napišite funkciju čiji prototip glasi:

```
void ReverseVector(vector<int>& v)
// pre: vektor v sadrži N cijelih brojeva
// post: elementi od v su u reverznom redosljedu
```

i koja ulaznom vektoru `v` mijenja redosljed elemenata tako da oni budu u reverznom redosljedu u odnosu na početni redosljed. Primjerice:

```
Prije poziva ReverseVector(a)
+-----+-----+-----+-----+
| 61 | 34 | 18 | 99 | 73 |
+-----+-----+-----+-----+
```

```
Poslije poziva ReverseVector(a)
+-----+-----+-----+-----+
| 73 | 99 | 18 | 34 | 61 |
+-----+-----+-----+-----+
```

102) Napišite generičku funkciju:

```
template <class T> void ReverseArray( T a[], int n )
```

kojom se niz `a[]`, koji sadrži `n` objekata tipa `T`, reorganizira tako da se elementi poredaju u reverznom redosljedu (prvi postaje zadnji, drugi postaje predzadnji, itd.).

103) Stanardna klasa `vector<T>` služi kao kontenjer za objekte tipa `T`. U kontenjeru može biti više istih objekata. Napišite funkciju:

```
template <class T>
vector<T> FormirajVektorBezDuplikata(const vector<T> V);
```

koja vraća vektor koji sadrži elemente vektora `V`, ali bez duplikata. (Napomena: možete koristiti STL funkciju `Find()`)

Primjerice, program:

```
vector<int> L;
L.push_back(5); L.push_back(5); L.push_back(2);
L.push_back(1); L.push_back(4); L.push_back(4);

vector<int> V = FormirajVektorBezDuplikata<int>(L);

for(int i=0; i<V.size(); i++)
    cout << V[i] << ", ";
```

ispisuje:

```
5, 2, 1, 4,
```

104) Zadana je klasa `CUcenik`:

```
class CUcenik {
    string m_ime;           // ime i prezime
    int m_ocjena;           // ocjena
public:
    CUcenik();
    int GetOcjena();
    string GetIme();
    void SetOcjena(int ocjena);
    void SetIme(string ime);
};
```

Pretpostavite da se podaci o grupi učenika nalaze u vektoru:

```
vector <CUcenik> SpisakOcjena;
```

Treba napisati funkciju `OcjenaUcenika()` kojom se dobavlja ocjena učenika kojem znamo ime. Prototip funkcije je:

```
int OcjenaUcenika(const vector<CUcenik>& spisak, const string& ime);

// PRED: spisak sadrži podatke o ucenicima
// argumenti: spisak - sadrži vektor podataka o ucenicima
//             ime - ime i prezime studenta kojem se traži ocjena
// POST: funkcija vraća ocjenu studenta, ukoliko se ime nalazi
//       u vektoru spisak, a ako ga nema funkcija vraća 0
```

2. RJEŠENJA

1)

- a) float
- b) int
- c) float
- d) float

2)

The value of c is: 2

3)

Imam 100 dolara!

4)

y = 1
y = 0

5)

0
5
1

6)

Vrijednost od c iznosi: 2
Vrijednost od c iznosi: 1

7)

```
int i=0;
int k=0;
while( i<10 ) {
    c = k*i;
    i++;
    k++;
}
```

8)

```
sum = 0;
for( i = 0 ; i < N ; i++)
    sum += a[i];
```

9)

a + b = 2
1 2
2 2

10)

```
2 1
2 4
5 4
```

11)

```
0
1
1
2
```

12)

```
24 r + s
18 11
18 42
18 0
```

13)

```
#include<iostream>
using namespace std;

double PovrsinaZidova(double x, double y, double z) {
//funkcija za izracunavanje povrsine zidova sobe korištenjem return
    double povrsina = (2 * x*y) + (2 * x*z) + (2 * z*y);
    return povrsina;
}

void PovrsinaZidova(double x, double y, double z, double& povrsina) {
    //funkcija za izracunavanje povrsine zidova sobe korištenjem
    reference
    povrsina = (2 * x*y) + (2 * x*z) + (2 * z*y);
}

int main() {
    double a, b, c, d;

    cout << "unesite dimenzije sobe:" << endl;
    cout << "a:" << endl;
    cin >> a;
    cout << "b:" << endl;
    cin >> b;
    cout << "c:" << endl;
    cin >> c;
    cout << "Povrsina sobe je: " << PovrsinaZidova(a, b, c) <<
endl;
    PovrsinaZidova(a, b, c, d);
    cout << "Povrsina sobe je: " << d << endl;

    return 0;
}
```

14)

```
#include <iostream>
using namespace std;

void KvadratKub(double x, double& rKvadrat, double& rKub) {
// funkcija koja prima tri argumenta
```

```

// i ispisuje na ekran kvadrat i kub

    rKvadrat = x*x;
    rKub = x*x*x;
}

void main() {
    double x = 3.0;
    double kvadrat, kub;
    KvadratKub(x, kvadrat, kub);
//argumenti se salju u funkciju preko reference
    cout << kvadrat << endl;
    cout << kub << endl;
}

```

15)

```

#include <iostream>
using namespace std;

void RadijaniUStupnjeve(double& kut) {
    kut = kut / 3.14 * 180;
}

int main() {
    double fi;

    cout << "Unesite kut u radijanima: " << endl;
    cin >> fi;
    cout << "Kut fi=" << fi << " u stupnjevima iznosi:" << endl;
    RadijanuUStupnjeve(fi);
    cout << fi << endl;

    return 0;
}

```

16)

```

void FormirajPismo(string ime, string magazin, int godina, int
mjesec, string& pismo) {
    string Mjesec[12] = { "Sijecanj", "Veljaca", "Ozujak",
    "Travanj", "Svibanj", "Lipanj", "Srpanj", "Kolovoz",
    "Rujan", "Listopad", "Studeni", "Prosinac" };
    pismo = "Dragi " + ime + ", ";
    pismo = pismo + "upozoravamo Vas da vasa preplata na
magazin " + magazin;
    pismo = pismo + "prestaje vrijediti u mjesecu: " +
Mjesec[mjesec + -1] + ", " + to_string(godina) + ". ";
    pismo = pismo + "Molimo, obnovite preplatu!";
}

```

17)

```

void PovecajSmanjiZaDelta(int delta, int& x, int& y) {
    x = x + delta;
    y = y - delta;
}

```

18)

a)
 x=43
 y=12

b)
 x=33
 y=22

19)

```
void OpsegKrug(a float r, float& O) {  
    O = 2*r*3.14;  
}
```

20)

```
void abc(int& a, int& b) {  
    a = a * 4;  
    b = b * 3;  
}
```

21)

```
double StrToDouble(char* s) {  
    return atof(s);  
}  
  
double StrToDouble(string s) {  
    return atof(s.c_str());  
}
```

22)

```
string DoubleToString(double num) {  
    char* buf = 0;  
    int err;  
  
    // alociramo C string veličine _CVTBUFSIZE  
    buf = (char*)malloc(_CVTBUFSIZE);  
    // vršimo konverziju funkcijom _gcvt_s  
    err = _gcvt_s(buf, _CVTBUFSIZE, num, 5);  
  
    if (err != 0) {  
        cout << "_gcvt_s failed with error code " << err << endl;  
        exit(1);  
    }  
  
    // konvertiramo iz C stringa u C++ string  
    string temp;  
    temp = buf;  
  
    return temp;  
}
```

23)

```

string ReverseString(string& Str) {
    string Rev;

    for( unsigned int i = 0 ; i < Str.length() ; i++) {
        Rev = Rev + Str.at(Str.length() - 1 - i);
    }

    return Rev;
}

```

24)

```

#include <iostream>
#include <string>

using namespace std;

string RimskiBroj(int n) {
    if (n == 0) {
        cout << "Pojam 0 su Rimljanima uveli Arapi!!!";
        return "";
    }

    switch(n) {
        case 1:
            return "I";
        case 2:
            return "II";
        case 3:
            return "III";
        case 4:
            return "IV";
        case 5:
            return "V";
        case 6:
            return "VI";
        case 7:
            return "VII";
        case 8:
            return "VIII";
        case 9:
            return "IX";
        case 10:
            return "X";
    }

    return "";
}

```

25)

```

int PromptRange(string poruka, int low, int high) {
    int i;

    do {
        cout << poruka;
        cin >> i;
    } while ( i < low || i > high );

    return i;
}

```



```
}
```

26)

```
// const je stavljen pred argument,  
// da se ne bi slučajno promijenio ulazni string  
  
string ZamijeniSlova(const string str) {  
  
    unsigned int i;  
    string temp(str);          // radni string  
  
    // petljom prolazimo kroz sve znakove stringa  
    for( i=0 ; i < temp.length() ; i++ ) {  
  
        // ako je slovo malo pretvorimo ga u veliko i obratno  
        if( isupper( temp[i] ) )  
            temp[i] = tolower( temp[i] );  
        else if( islower( temp[i] ) )  
            temp[i] = toupper( temp[i] );  
    }  
  
    return temp;              // vraćamo promijenjeni string  
}
```

27)

```
int StringSpace(string x) {  
    // brojač razmaka  
    int y = 0;  
  
    // petljom prolazimo kroz sve znakove stringa  
    for( unsigned int i = 0 ; i < x.length() ; i++ ) {  
        if( x[i] == ' ' ) {  
            y++;  
        }  
    }  
    return y;  
}
```

28)

```
#include <iostream>  
#include "tstring.h"  
  
using namespace std;  
  
int main() {  
    TString s;  
  
    cin >> s;                      // prvo čitanje stringa  
    while( s.find("kraj") ) {      // petlja i provjera uvjeta  
  
        cout << endl << "Unesen je string: " << s << endl;  
  
        cin >> s;                  // ponovno čitanje stringa  
    }  
    return 0;  
}
```

29)

```
string ZamijeniRazmakePodvlakom(const string& str) {
    string ret;                //string koji se vraca
    for( unsigned int i = 0 ; i < str.size() ; i++ ) {
//prolazimo kroz cijeli string tako da pocnemo od prvog
//znaka i idemo do njegovog kraja .size()
        if( str[i] == ' ' ) {    //provjeravamo prazna mjesta
            ret.push_back('_');
// i ako je uvjet zadovoljen tada prazno mjesto mjenjamo podvlakom
        } else {
//inace stavljamo znak koji je na mjestu i
            ret.push_back( str[i] );
        }
    }
    return ret;
}
```

30)

- 1) 1+1
- 2) 999
- 3) redov

31)

6
Prvi p

32)

3+4
1111111
predivo

33)

red red

34)

```
#include <iostream>
#include <sstream>
#include <string>
using namespace std;

string ToStringa(int x) {
    ostringstream ostr;                //pretvara broj u string
    ostr << x;
    string str0 = ostr.str();
    return str0;
}

string ToStringb(double x) {
    ostringstream ostr;                //pretvara broj u string
    ostr << x;
    string str0 = ostr.str();
    return str0;
}

string ToStringc(bool b) {
```

```

        ostreamstream ostr;                //pretvara bool u string
        ostr << b;
        string str0 = ostr.str();
        return str0;
    }

    int main() {
        int a = 108;
        double b = 109.901;
        bool c = true;
        cout << ToStringa(a) << endl;
        cout << ToStringb(b) << endl;
        cout << ToStringc(c) << endl;
        return 0;
    }

```

35)

```

#include <sstream>
#include <iostream>
#include <string>
using namespace std;

string HrvBroj(float x){
    string str, str1, str2;
    ostreamstream ostr;
    int pos;

    ostr << x;
    str = ostr.str();
    pos = str.find('.');
    str1 = str.substr(0, pos);
    str2 = str.substr(pos+1, str.length());

    return str1 + ',' + str2;
}

int main() {
    float a = float(7.8);
    cout << HrvBroj(a) << endl;
    return 0;
}

```

36)

```

#include <iostream>
using namespace std;

//klasa registracija koja se sastoji od privatnih clanova i konstruktora
class Registracija {
public:
    int reg1;
    int reg2;
    Registracija();
};

//definiranje konstruktora klase
Registracija::Registracija() {
    reg1 = 100;
}

```

```

        reg2 = 100;
    }

    void main() {
        //stvaranje objekta klase registracije
        Registracija r;
        // javne varijable klase se postavljaju na određene vrijednosti
        r.reg1 = 543;
        r.reg2 = 234;
        // ispis registracije
        cout << r.reg1 << "-" << r.reg2 << endl;
    }

```

37)

```

#include <iostream>
using namespace std;

//klasa registracija sa privatnim varijablama
//konstruktorom i set funkcijom i ispisnom funkcijom
class Registracija {
private:
    int reg1;
    int reg2;
public:
    Registracija();
    void Postavi(int temp1, int temp2); //set funkcija za mjenjanje
    privatnih varijabli
    void Ispisi();
};

Registracija::Registracija() {
    reg1 = 100;
    reg2 = 100;
}

void Registracija::Postavi(int temp1, int temp2) {
    // postavljanje registracijske tablice u intervalu od 100 do 1000
    if (temp1 > 99 && temp1 < 1000 && temp2 > 99 && temp2 < 1000) {
        reg1 = temp1;
        reg2 = temp2;
    }
}

void Registracija::Ispisi() {
    // ne trebaju argumenti jer koristi vlastite varijable
    cout << reg1 << "-" << reg2 << endl;
}

int main() {
    Registracija r; // objekt klase registracija
    r.Postavi(543, 234);
    r.Ispisi();

    return 0;
}

```

38)

```

#include <iostream>

```

```

using namespace std;

class kocka {
public:
    double stranica;
    kocka();
    double VolumenKocke();
    double OplosjeKocke();

};
kocka::kocka() {
    stranica = 0;
}
double kocka::VolumenKocke() {
    double volumen = stranica*stranica*stranica;
    return volumen;
}
double kocka::OplosjeKocke() {
    double oplosje = 6 * stranica*stranica;
    return oplosje;
}

int main() {
    kocka K;
    double y;
    cout << "unesite vrijednost stranice kocke" << endl;
    cin >> y;
    K.stranica = y;
    cout << "volumen kocke je :" << K.VolumenKocke() << endl;
    cout << "oplosje kocke je :" << K.OplosjeKocke() << endl;

    return 0;
}

```

39)

```

class Adresa {
public:
    Adresa();
    string get_ulica();
    string get_broj();
    string get_grad();
    string get_drzava();
    void set_ulica(string temp);
    void set_broj(string temp);
    void set_grad(string temp);
    void set_drzava(string temp);

private:
    string ulica;
    string broj;
    string grad;
    string drzava;
};
Adresa::Adresa() {
    ulica = "";
    grad = "";
    broj = "";
    drzava = "";
}
string Adresa::get_ulica() {

```

```

        return ulica;
    }
    string Adresa::get_grad() {
        return grad;
    }
    string Adresa::get_broj() {
        return broj;
    }
    string Adresa::get_drzava() {
        return drzava;
    }
    void Adresa::set_ulica(string temp) {
        ulica = temp;
    }
    void Adresa::set_grad(string temp) {
        grad = temp;
    }
    void Adresa::set_broj(string temp) {
        broj = temp;
    }
    void Adresa::set_drzava(string temp) {
        drzava = temp;
    }
    istream& operator >> (istream& in, Adresa& r) {
        string a, b, c, d;

        in >> a;
        r.set_ulica(a);
        in >> b;
        r.set_broj( b );
        in >> c;
        r.set_grad( c );
        in >> d;
        r.set_drzava( d );
        return in;
    }
}

```

40)

```

a)
    int Razlomak::Brojnik() const {
        return m_brojnik;
    }

    void Razlomak::Brojnik(int br) {
        m_brojnik = br;
    }

b)
    int nzm(int a, int b) {
        // ova funkcija će nam koristiti za skraćivanje razlomaka
        int t;

        while (b != 0) {
            t = b;
            b = a % b;
            a = t;
        }

        return a;
    }

```

```

Razlomak operator + (Razlomak& r, Razlomak& s) {
    int a = r.Brojniki();
    int b = r.Nazivnik();
    int c = s.Brojniki();
    int d = s.Nazivnik();

    int nazivnik = b*d;
    int brojnik = d*a + b*c;

    // skрати razlomak
    int djelitelj = nzm(brojnik, nazivnik);

    return Razlomak(brojnik/djelitelj, nazivnik/djelitelj);
}

Razlomak operator - (Razlomak& r, Razlomak& s)
{
    int a = r.Brojniki();
    int b = r.Nazivnik();
    int c = s.Brojniki();
    int d = s.Nazivnik();

    int nazivnik = b*d;
    int brojnik = d*a - b*c;

    // skрати razlomak
    int djelitelj = nzm(brojnik, nazivnik);

    return Razlomak(brojnik/djelitelj, nazivnik/djelitelj);
}

```

```

c)
int main(void) {
    Razlomak r1(3,4);
    Razlomak r2(7,3);
    Razlomak zbroj, razlika;

    zbroj = r1 + r2;
    razlika = r1 - r2;

    cout << "Zbroj je: " << zbroj.Brojniki() << "/" <<
    zbroj.Nazivnik() << endl;
    cout << "Razlika je: " << razlika.Brojniki() << "/" <<
    razlika.Nazivnik() << endl;

    return 0;
}

```

41)

```

ostream& operator << (ostream& out, Razlomak& s) {
    out << s.Brojniki() << "/" << s.Nazivnik();

    return out;
}

istream& operator >> (istream& in, Razlomak& r) {
    int a, b;
    char ch;

    if (in >> a >> ch >> b)    // čitam dva int-a

```

```

    {
        r.Brojni(a);
        r.Nazivnik(b);
    }

    return in;
}

```

42)

```

ostream& operator << (ostream& out, const Student& s) {
    out << "Matični broj: " << s.m_broj << endl;
    out << "Ime: " << s.m_ime << endl;
    out << "Prezime: " << s.m_prezime << endl;

    return out;        // vraćamo out zbog nadovezivanja
}

Student& Student::operator = (const Student& d) {
    this->m_broj = d.m_broj;
    this->m_ime = d.m_ime;
    this->m_prezime = d.m_prezime;

    return *this;      // vraćamo *this zbog nadovezivanja
}

```

43)

```

class Pravokutnik {
public:
    Pravokutnik ();
    Pravokutnik& operator= (const Pravokutnik& p);

    int GetSirina();
    int GetVisina();

public:
    int x1, y1, x2, y2;
};

Pravokutnik::Pravokutnik() {
    x1 = y1 = 0;
    x2 = y2 = 1;
}

Pravokutnik& Pravokutnik::operator= (const Pravokutnik& p) {
    x1 = p.x1;
    y1 = p.y1;
    x2 = p.x2;
    y2 = p.y2;

    return *this;      // vraćamo *this zbog nadovezivanja
}

int Pravokutnik::GetSirina() { return x2 - x1; }
int Pravokutnik::GetVisina() { return y2 - y1; }

```


44)

```
istream& operator>> (istream& s, Pravokutnik& c1) {
    int tempX1, tempY1, tempX2, tempY2;

    // ukoliko čitanje ne uspije "s >>" vraća false
    // pa se vrijednosti koordinata ne mijenjaju
    if( s >> tempX1 >> tempY1 >> tempX2 >> tempY2 ) {
        c1.x1 = tempX1;
        c1.y1 = tempY1;
        c1.x2 = tempX2;
        c1.y2 = tempY2;
    }

    return s;
}
```

45)

```
#include<iostream>
#include<cmath>

using namespace std;

class PointR {
public:
    PointR();
    /*~PointR();*/
    void SetXYPosition(float x, float y);
    void GetXYPosition(float& x, float& y);
    float GetAngle() const { return m_fi; }
    float GetRadius() const { return m_r; }
    PointR(const PointR& temp); //kopirni konstruktor

private:
    float m_r;
    float m_fi;
};

PointR::PointR(const PointR& temp) { //kopirni konstruktor
    m_r = temp.m_r;
    m_fi = temp.m_fi;
}

PointR::PointR() { //konstruktor
    m_r = 0.0; m_fi = 0.0;
}

void PointR::GetXYPosition(float& x, float& y) {
    x = m_r*cos(m_fi);
    y = m_r*sin(m_fi);
}

void PointR::SetXYPosition(float x, float y) {
    m_r = sqrt( x*x + y*y);
    m_fi = atan( y/x );
}

int main() {
    float x, y; //ovo je radijus i kut u radijanima
    PointR t;
```

```

        cout << "udaljenost od ishodista: ";
        cin >> x;
        cout << "radijus je: ";
        cin >> y;
        t.SetXYPosition(x, y);
        float x1, y1;          //ovo su tocke kartezijevog koordinatnog
sustava
        t.GetXYPosition(x1, y1);
        cout << x1 << ", " << y1 << endl;

        return 0;
}

```

46)

```

class PointR {

    ...

public:

    ...

    PointR& operator=(const PointR& temp);
};

PointR& PointR::operator=(const PointR& temp) {
    m_r = temp.m_r;
    m_fi = temp.m_fi;

    return *this;
}

ostream& operator << (ostream& out, const PointR& p) {
    out << p.GetRadius() << " " << p.GetAngle();

    return out;
}

```

47)

```

// konstruktor
CUcenik::CUcenik() {
    m_ime = "";
    m_ocjena = 0;
}

// definiranje operatora za upis ocjene i imena u jednom redu
istream& operator >> (istream &in, CUcenik &u) {
    string ime;
    int ocjena;
    if (in >> ime >> ocjena) {
        u.SetIme(ime);
        u.SetOcjena(ocjena);
    }
    return in;
}

```

48)

```

#include <iostream>
using namespace std;
#include <math.h>

```

```

class Complex {
public:

    double real;
    double imag;
    Complex() : real(0), imag(0) {};
    Complex(const Complex&);

    double Mag() const;
    double Phase() const;

};

Complex::Complex(const Complex& temp) {
    real = temp.real;
    imag = temp.imag;
}

// const u deklaraciji funkcije znači da
// funkcija ne mijenja vrijednost članskih varijabli klase
double Complex::Mag() const {
    return sqrt( imag*imag + real*real );
}

double Complex::Phase() const {
    return atan( imag/real );
}

```

49)

```

#include <iostream>
using namespace std;
#include <math.h>

class Complex {
public:

    double real;
    double imag;
    Complex() : real(0), imag(0) {};
    Complex(const Complex &);

    double Mag() const;
    double Phase() const;

    Complex& operator=( const Complex& temp );
};

Complex& Complex::operator =( const Complex& temp ) {
    real = temp.real;
    imag = temp.imag;

    return *this;
}

ostream& operator <<( ostream& out, const Complex& temp ) {
    out << temp.real;
    if( temp.imag >= 0.0 ) {
        out << "+j ";
    } else {

```

```

        out << "-j ";
    }
    cout << abs( temp.imag );

    return out;
}

```

50)

```

Prekidac::Prekidac() {
    m_iskljucen = true;
}
void Prekidac::Ukljuci() {
    m_iskljucen = false;
}
void Prekidac::Iskljuci() {
    m_iskljucen = true;
}
bool Prekidac::Ukljucen() {
    return !m_iskljucen;
}

```

51)

```

Prekidac::Prekidac(const Prekidac& temp) {
    m_iskljucen = temp.m_iskljucen;
}

// ovo je relacijski operator, a ne operator pridjele vrijednosti!
bool operator==(const Prekidac& temp, const Prekidac& temp1) {
    if( temp.m_iskljucen == temp1.m_iskljucen) {
        return true;
    } else {
        return false;
    }
}

// dodati kao friend funkciju u deklaraciju klase!
ostream& operator << (ostream& out, const Prekidac& s) {
    if( !s.m_iskljucen ) {
        out << "ukljucen";
    } else {
        out << "iskljucen";
    }
    return out;
}

// alternativno rješenje - kao članska funkcija klase
ostream& operator << (ostream& out, const Prekidac& s) {
    if( !s.m_iskljucen ) {
        out << "ukljucen";
    } else {
        out << "iskljucen";
    }
    return out;
}

```

52)

```

class Loto {
// privatne varijable

```

```

        int* m_br;                // za dinamičku alokaciju polja brojeva

        int m_rezervni_broj;
public:
        Loto() {
                m_br = new int[6];    // alokacija polja brojeva
                PostaviBrojeve();
        };
        ~Loto() { delete [] m_br; } // dealokacija polja brojeva
        void PostaviBrojeve();
        int *DobaviNizBrojeva() const {return m_br;}
        int DobaviRezervniBroj() const {return m_rezervni_broj;}
};
void Loto::PostaviBrojeve() {        // nasumično postavljanje brojeva
        for(int i=0 ; i<6 ; i++) {
                m_br[i] = rand()%42;
        }
        m_rezervni_broj = rand()%42;
}

```

53)

```

ostream& operator << (ostream& s, const Loto& L) {
        int* A;                // pokazivač pomoću kojeg ćemo pristupat polju

        A = L.DobaviNizBrojeva();
        for( int i = 0 ; i<6 ; i++ ) {
                s << A[i] << ", ";
        }
        s << " (" << L.DobaviRezervniBroj() << ")";

        return s;
}

```

54)

```

SimpleCircle::SimpleCircle() {
        m_radius = 5;
}
SimpleCircle::SimpleCircle(const SimpleCircle& temp) {
        m_radius = temp.m_radius;
}
SimpleCircle& SimpleCircle::operator=(const SimpleCircle& temp) {
        m_radius = temp.m_radius;

        return *this;    // radi nadovezivanja
}
void SimpleCircle::SetRadius(int temp) {
        m_radius = temp;
}
int SimpleCircle::GetRadius() {
        return m_radius;
}

```

55)

```

Point2D::Point2D() {
        m_x = 0;
        m_y = 0;
}

```

```

}

Point2D::Point2D( const Point2D& temp ) {
    m_x = temp.m_x;
    m_y = temp.m_y;
}

void Point2D::SetX( double x ) {
    m_x = x;
}

void Point2D::SetY( double y ) {
    m_y = y;
}

double Point2D::GetX() {
    return m_x;
}

double Point2D::GetY() {
    return m_y;
}

```

56)

```

class Point3D : public Point2D {
public:

    // konstruktori se nasljeđuju i dopunjuju
    Point3D() { m_z = 0; };
    Point3D( const Point3D& temp ) : Point2D( temp ) {
        m_z = temp.m_z;
    };

    // udaljenost se računa na drugi način
    double Distance();

    // pristupne funkcije za z koordinatu
    void SetZ(double z) { m_z = z; };
    double GetZ() {return m_z;};

protected:
    double m_z;
};

double Point3D::Distance() {
    return sqrt( m_x*m_x + m_y*m_y + m_z+m_z );
}

```

57)

```

// prefiks operator (nema argument)
Circle& Circle::operator++() {
    m_radius++;
    return *this;
}

// postfiks operator - argument int unused služi kompajleru
// da raspozna da je ovo postfix, a ne prefix operator
const Circle Circle::operator++(int unused) {

```

```

// služi da privremeno spremimo objekt, tako da nakon inkrementa
// vratimo staro stanje, a ne novo sa inkrementiranim brojačem
    Circle temp(*this);

// inkrementiraj radijus
    ++m_radius;

    return temp;
}

bool operator==(Circle &c1, Circle& c2) {
    return c1.GetRadius() == c2.GetRadius();
}

```

58)

```

istream& operator >>(istream& s, Circle& c1) {
    int temp;          // radna varijabla

// provjera je li čitanje uspjelo
    if (s >> temp) {
        c1.SetRadius(temp);
    }

// vraćam radi nadovezivanja
    return s;
}

ostream& operator << (ostream& s, Circle& c1) {

// ispis i vraćanje radi nadovezivanja u istoj liniji
    return s << "Radius iznosi:" << c1.GetRadius() << endl;
}

```

59)

```

HorLine& HorLine::operator=(const HorLine& obj) {
    m_HorLine = obj.m_HorLine;
    m_color = obj.m_color;

    return *this;
}

HorLine& HorLine::operator++() // prefiks operator
{
    ++m_HorLine;          // inkrementiraj brojač
    return *this;         // vrati njegovu referencu
}

// postfix operator; int unused služi kompajleru da raspozna da je
// ovo postfix, a ne prefix operator
HorLine HorLine::operator++(int unused) {

// služi da privremeno spremimo objekt, tako da nakon inkrementa
// vratimo staro stanje a ne novo sa inkrementiranim brojačem
    HorLine temp(*this);

// inkrementiraj brojač
    ++m_HorLine;

    return temp;
}

```

60)

```
class HorLine {
public:
    HorLine() : m_HorLine (0), m_color (0){}

    void SetHorLine (double d) { m_HorLine = d;};
    double GetHorLine() {return m_HorLine;};

    void SetColor (COLORREF c) { m_color = c;};
    COLORREF GetColor() {return m_color;};

    HorLine & operator++();           //prefiks
    HorLine operator++(int unused);   //postfiks

    HorLine & operator=(const HorLine& d);

private:
    double m_HorLine;                // duljina linije
    COLORREF m_color;                // boja linije
};

HorLine& HorLine::operator=(const HorLine& obj) {
    m_HorLine = obj.m_HorLine;
    m_color = obj.m_color;

    return* this;
}

class PosHorLine : public HorLine {
public:
    PosHorLine() { m_x = 0.0; m_y = 0.0; }

private:
    double m_x, m_y;
};
```

61)

```
bool operator == (const Tocka& a, const Tocka& b) {
    if( a.GetX() != b.GetX() || a.GetY() != b.GetY() ) {
        return false;
    } else {
        return true;
    }
}

ostream& operator << (ostream& out, const Tocka& t) {
    out << t.GetX() << " " << t.GetY();

    return out;
}
```

62)

```
void main() {
    Registracija* pR;           //pokazivac na objekt klase registracija

    pR = new Registracija[10];   //alokacija memorije
    pR[0]->Postavi( 123, 456 ); //postavljane varijabli klase
    pR[0]->Ispisi();            //ispis
}
```



```

        delete [] pR;                                //dealociranje
    }

```

63)

```

#include <iostream>
#include <fstream>
using namespace std;

int main() {
    ifstream input;
    char a;

    input.open("data.txt");        // otvaranje datoteke

    input.get(a);                  // počinje učitavanje
    cout << a;
    while( !input.eof() ) {        // petlja i provjera kraja dat.
        if( a == '\n' ) {
            input.get(a);
            cout << a;              // ispisujem samo prvi znak retka
        } else {
            input.get(a);
        }
    }
    input.close();                 // zatvaranje datoteke

    return 0;
}

```

64)

```

ofstream fout("brojevi.bin", ios::binary);    // otvaranje datoteke
fout.write( (char*)A, 1000 * sizeof(float) ); // binarni ispis
fout.close();                                // zatvaranje datoteke

```

65)

```

ofstream fout("pr.bin", ios::binary);
fout.write( (char*)Pr, 1000 * sizeof(Pravokutnik) );
fout.close();

```

66)

```

ofstream out_file("kbrojevi.txt");            // otvaranje datoteke
for( int i = 0 ; i < 1000 ; i++ ) {          // petlja za ispis

    out_file << A[i].real << " " << A[i].imag << endl;
}
out_file.close();                             // zatvaranje datoteke

```

67)

```

#include <iostream>
#include <fstream>

using namespace std;

int main() {
    ifstream input;

```

```

char a, b, x;

input.open("test.dat");           // otvaranje datoteke

input.get(a);                      // čitanje tri kontrolna znaka
input.get(b);
input.get(x);

// provjera tri kontrolna znaka
if( a == 'a' && b == 'b' && x == 'x' ) {
    cout << "prepoznat tip datoteke!" << endl;
} else {
    cout << "nepoznat tip datoteke!" << endl;
}
input.close();                     // zatvaranje datoteke

return 0;
}

```

68)

```

#include <iostream>
#include <fstream>
using namespace std;

int main() {
    ifstream ulaz;

    ulaz.open("test.txt") ;

    if (!ulaz.good() || !ulaz.is_open())
        return 1;

    int iZnakova = 0;
    int iRijeci = 0;
    int iLinija = 0;

    unsigned char bRijecZapocela = 0;
    char c;

    while (ulaz.get(c)) {
        if ( c != '\n' && c != '\r' ) iZnakova++;

        if ( c == ' ' || c == '\n' || c == '\t' ) {
            // broje se samo linije teksta!
            if ( c == '\n' && bRijecZapocela )
                iLinija++;

            if ( bRijecZapocela == 1 ) {
                bRijecZapocela = 0;
                iRijeci++;
            }
        } else {
            bRijecZapocela = 1;
        }
    }

    //zatvori datoteku
    ulaz.close();

    cout << "Znakova: " << iZnakova << endl;
    cout << "Rijeci: " << (bRijecZapocela ? iRijeci+1 : iRijeci)

```

```

        << endl;
    cout << "Linija: " << (bRijecZapocela ? iLinija+1 : iLinija)
        << endl;

    return 0;
}

```

69)

```

#include<iostream>
#include<string>
#include<fstream>
using namespace std;

int main() {
    ofstream izlaz;
    string s;
    int i;

    izlaz.open("20imena.txt");
    for( i = 0 ; i < 20 ; i++ ) {
        cin >> s;
        izlaz << s << " " << s.length() << endl;
    }
    izlaz.close();

    return 0;
}

```

70)

```

void CitajStudDatoteku (ifstream& input, vector<Student>& spisak) {
    Student s;
    int i;

    for( i = 0 ; !input.eof() ; i++ ) {
        input >> s.m_broj;
        input >> s.m_ime;
        input >> s.m_prezime;

        spisak.push_back(s);
    }
}

```

71)

```

#include <iostream>
#include <fstream>
using namespace std;

int main( int argc, char* argv[] ) {
    if (argc < 3) {
        cout << "Greska... Upotreba: " << argv[0] << " dat1.txt
        dat2.txt" << endl;
        return -1;
    }

    ifstream ulaz;
    ofstream izlaz;
}

```

```

    ulaz.open(argv[1]) ;

    if (!ulaz.good() || !ulaz.is_open()) {
        cout << "Ne moze se otvoriti ulazna datoteka";
        return -1;
    }

    izlaz.open(argv[2]) ;
    if (!izlaz.good() || !izlaz.is_open()) {
        cout << "Ne moze se otvoriti izlazna datoteka";
        return -1;
    }

    int iLineCounter = 1;
    unsigned char bPrintLnNo = 1;

    char c ;
    while (ulaz.get(c)) {
        if (bPrintLnNo) {
            izlaz << iLineCounter << ": ";
            bPrintLnNo = 0;
        }

        izlaz.put(c);

        if (c == '\\n') {
            iLineCounter++;
            bPrintLnNo = 1;
        }
    }

    ulaz.close();
    izlaz.close();

    return 0;
}

```

72)

```

#include <iostream>
#include <string>
#include <fstream>

using namespace std;

int main(void) {
    ifstream ulaz;
    ofstream izlaz;
    int a;
    unsigned int b;
    string c;

    ulaz.open("ulazna.cpp");
    izlaz.open("izlazna.cpp");

    while( !ulaz.eof() ) {
        getline(ulaz, c);
        a = c.find("//");
        if ( a >= 0 )
            c = c.substr( 0, a );
    }
}

```

```

        if( a < 0 ) {
            for( b = 0 ; b < c.length() ; b++ ) {
                if( c.at(b) != ' ' || c.at(b) != '\t' ) {
                    break;
                }
            }

            if( b != c.length() ) {
                izlaz << c << endl;
            }
        }
    }
    ulaz.close();
    izlaz.close();

    return 0;
}

```

73)

```

#include <iostream>
#include <fstream>
using namespace std;

int main(int argc, char** argv) {
    ifstream ulaz;
    char c;
    int br = 0;

    if( argc < 2 ) {
        cout << "Nije definirana ulazna datoteka.";
        return 1;
    }
    ulaz.open( argv[1] );
    if ( !ulaz.good() || !ulaz.is_open() ) {
        cout << "Ne moze se otvoriti ulazna datoteka";
        return -1;
    }

    while( !ulaz.eof() ) {
        ulaz >> c;
        br++;
        if( c != ' ' ) {
            cout << c;
        }
        if( br == 40 ) {
            cout << endl;
            br = 0;
        }
    }
    ulaz.close();

    return 0;
}

```

74)

```

int main() {
    ifstream ulaz;

```

```

float a, b; // radijus i kut u radijanimi
float x, y; //ovo su koordinate kartezijevog sustava
PointR t;

ulaz.open("ulazna.txt");
while (!ulaz.eof()) {
    ulaz >> x >> y;
    t.SetXYPosition(x, y);
    cout << t.GetRadius() << " " << t.GetAngle() << endl;
}
ulaz.close();

return 0;
}

```

75)

```

#include<stdlib.h>
#include<string>
#include<iostream>
#include<fstream>
using namespace std;

int main() {
    ifstream ulaz;
    ofstream izlaz;
    izlaz.open("izlazna_1.txt");
    ulaz.open("datoteka.txt");
    char c;
    char string[10];
    while (ulaz.get(c)) {
        if ((c >= 65 && c <= 90)) {
            itoa(c, string, 10);
            cout << string;
            izlaz << string;
        } else if ((c >= 97 && c <= 122)) {
            itoa(c, string, 10);
            cout << string;
            izlaz << string;
        } else {
            izlaz.put(c);
            cout << c;
        }
    }
    izlaz.close();
    ulaz.close();

    return 0;
}

```

76)

```

int P(int n, int k) {

    // temeljno pravilo
    if( ( k > 0 ) && ( n > k ) )
        return P( n - 1, k ) + P( n - 1, k - 1 );

    // granični slučaj
    if( ( k == 0 || k == n ) && ( n >= 0 ) ) {
        return 1;
    }
}

```

```

    }

    // praktični nije potrebno - služi da kompajler ne baci grešku
    return 0;
}

```

77)

```

class ColoredCircle : public Circle {
public:
    int m_color;

    ColoredCircle() { m_color = 0; };
    ColoredCircle& ColoredCircle::operator=
        (const ColoredCircle& c);
};

ColoredCircle& ColoredCircle::operator= (const ColoredCircle& c) {
    m_radius = c.m_radius;
    m_color = c.m_color;
    return *this;
}

```

78)

```

class PozicioniraniString : public Tocka {
public:
    PozicioniraniString() : Tocka() { m_tekst = ""; }

    void SetTekst(string&);
    string& GetTekst();

private:
    string m_tekst;
};

void PozicioniraniString::SetTekst(string& temp) {
    m_tekst = temp;
}

string& PozicioniraniString::GetTekst() {
    return m_tekst;
}

```

79)

```

class Krug1 : public Tocka {
    int m_radius;
public:
    Krug1() { m_radius = 0; };
    void SetRadius(int);
    int GetRadius();
};

```

80)

```

class PravokutnikHasATocka{
public:
    PravokutnikHasATocka ();
}

```

```

        void SetSirina(int sirina) {m_sirina = sirina;};
        void SetVisina(int visina) {m_visina = visina;};
        int GetSirina() {return m_sirina;};
        int GetVisina() {return m_visina;};

        int Povrsina();

        void SetX(int x) {m_p.SetX(x);};
        void SetY(int y) {m_p.SetY(y);};
        int GetX() {return m_p.GetX(); };
        int GetY() {return m_p.GetY(); };

    private:
        int m_sirina;
        int m_visina;
        Tocka m_p;
};

PravokutnikHasATocka::PravokutnikHasATocka() {
    m_sirina = 1;
    m_visina = 1;

    m_p.SetX(0);
    m_p.SetY(0);
}

class PravokutnikIsATocka : public Tocka {
    public:
        PravokutnikIsATocka();

        void SetSirina(int sirina) {m_sirina = sirina;};
        void SetVisina(int visina) {m_visina = visina;};
        int GetSirina() {return m_sirina;};
        int GetVisina() {return m_visina;};

        int Povrsina();

        void SetX(int x) {m_x = x;};
        void SetY(int y) {m_y = y;};
        int GetX() {return m_x; };
        int GetY() {return m_y; };

    private:
        int m_sirina;
        int m_visina;
};

```

81)

```

class Point3D : public Point2D {
    public:
        void SetZ(int z) { m_z = z; }
        int GetZ() { return m_z; }
        Point3D() { m_z = 0; }

        bool operator==(const Point3D& temp) {
            if( m_x == temp.m_x && m_y == temp.m_y
                && m_z == temp.m_z )
                return true;
        }
};

```



```

        else
            return false;
    }

protected:
    int m_z;
};

istream& operator >> (istream& in, Point3D& p) {
    int x, y, z;
    if( in >> x >> y >> z ) {
        p.SetX(x);
        p.SetY(y);
        p.SetZ(z);
    }
    return in;
}

```

82)

```

class CStudent : public CUcenik {
//nasljeđivanje od klase ucenik funkcija i varijabli koje su protected
    CKolokvij m_kolokvij;
public:
    CStudent();
    void SetOcjenaKolokvija(int brojKolokvija, int ocjena);
    int GetOcjenaKolokvija(int brojKolokvija);
};

CStudent::CStudent() {
//postavljanje na primarnu ocjenu svih kolokvija
    m_ime = "";
    m_ocjena = 0;
    m_kolokvij.m_ocjena[0] = 0;
    m_kolokvij.m_ocjena[1] = 0;
    m_kolokvij.m_ocjena[2] = 0;
}

void CStudent::SetOcjenaKolokvija(int brojKolokvija, int ocjena) {
//provjera ocjena i njihovog intervala
    if (ocjena >= 0 && ocjena <= 5) {
        m_kolokvij.m_ocjena[brojKolokvija] = ocjena;
    }
}

int CStudent::GetOcjenaKolokvija(int brojKolokvija) {
//get funkcija za ocjenu kolokvija
    return m_kolokvij.m_ocjena[brojKolokvija];
}

ostream& operator << (ostream &out, CStudent &s) {
//definicija operatora ispisa cijelog studenta i njegovih ocjena
    out << s.GetIme() << " ";
    out << s.GetOcjenaKolokvija(0) << " ";
    out << s.GetOcjenaKolokvija(1) << " ";
    out << s.GetOcjenaKolokvija(2) << " ";
    out << s.GetOcjena() << " ";

    return out;
}

```

83)

```
#include <iostream>
#include<fstream>
using namespace std;

template <class T>
class Circle {
public:
    Circle() : m_radius(0){};
    Circle(T c) :m_radius(c){};
    void SetRadius(T r){ m_radius = r; }
    T GetRadius() { return m_radius; }

    Circle& operator++();

    template<class T>
    friend ostream& operator << (ostream& s, const Circle<T>& c1);
    template<class T>
    friend istream& operator >> (istream& s, Circle<T>& c1);

protected:
    T m_radius;
};

template<class T> Circle<T>& Circle<T>::operator++() {
    ++m_radius;
    return *this;
}

template<class T> ostream& operator << (ostream& s, const Circle<T>&
c1) {
    return s << c1.m_radius;
}

template <class T> istream& operator >> (istream& s, Circle<T>& c1)
{
    T r;
    if (s >> r)
        c1.SetRadius(r);
    return s;
}

int main() {
    Circle <double> c(3.14);
    Circle <int> d(5);

    cout << "Unesite radijus neobojanog kruga:" << endl;
    cin >> c;
    cout << "Unesite radijus obojanog kruga:" << endl;
    cin >> d;

    cout << "Radijus neobojanog kruga" << " " << c << endl;
    cout << "Radijus obojanog kruga " << " " << d << endl << endl;

    ++c;
    ++d;

    cout << "Radijus neobojanog kruga" << " " << c << endl;
    cout << "Radijus obojanog kruga " << " " << d << endl << endl;
```

```

        return 0;
    }

```

84)

```

template <class T> int PronadjiElement(const vector<T> V, T x) {
    for( int i = 0 ; i < V.size() ; i++ ) {
        if( V[i] == x )
            return i;
    }
    return -1;
}

template <class T> vector<T> SortirajVektor (const vector<T> V){
    vector<T> vecTemp(V);

    // sortiranje selection sortom
    T temp, s1, s2;
    int i, j;
    int imin;
    for (i = 0 ; i < (vecTemp.size() - 1) ; i++) {
        imin = i;
        for (j = i+1 ; j < vecTemp.size() ; j++) {
            s1 = vecTemp[j];
            s2 = vecTemp[imin];
            if ( s1 < s2 )
                imin = j;
        }

        temp = vecTemp[i];
        vecTemp[i] = vecTemp[imin];
        vecTemp[imin] = temp;
    }

    return vecTemp;
}

```

85)

```

void Pomnozi_elemente_liste( LIST& L, double multiplikator) {

    // za šetanje po listi se koristi iterator
    LIST::iterator Iter;

    for ( Iter = L.begin() ; Iter != L.end() ; Iter++ ) {
        *Iter = *Iter * multiplikator;
    }
}

```

86)

```

void Razdijeli_vektor( Vec& V, Vec& Vshort, Vec& Vlong ) {

    // zadatak se može riješiti i bez iteratora
    // tako da se po vektoru šeta pomoću cjelobrojnog indeksa

    Vec::iterator iter;

    for( iter = V.begin() ; iter != V.end() ; iter++ ){
        if( (*iter).length() < 10 ) {
            Vshort.push_back(*iter);

```

```

        } else {
            Vlong.push_back(*iter);
        }
    }
}

```

87)

```

void Razdijeli_vektor( Vec& V, Vec& V1, Vec& V2 ) {
    Vec::iterator Iter;

    for( Iter = V.begin() ; Iter != V.end() ; Iter++ ) {
        if( *Iter % 2 == 0 ) {
            V2.push_back(*Iter);
        } else {
            V1.push_back(*Iter);
        }
    }
}

```

88)

```

void SortStrVector( vector<string>& strVec ) {
    // za sortiranje se koristi selection sort

    string temp, s1, s2;
    int i, j;
    int imin; /* indeks najmanjeg elementa u A[i..n-1] */

    for ( i = 0 ; i < (strVec.size() - 1) ; i++) {
        // Odredi najmanji element vektora
        imin = i; /* pretpostavi da je to element indeksa i */
        for ( j = i+1 ; j < strVec.size() ; j++) {
            string s1 = strVec[j];
            string s2 = strVec[imin];
            if ( s1 < s2 ) /* ako je j najmanji */
                imin = j; /* zapamti njegov indeks */
        }
        // Sada je imin najmanji element od i do n-1,
        // njega zamjenjujemo sa i.
        temp = strVec[i];
        strVec[i] = strVec[imin];
        strVec[imin] = temp;
    }
}

```

89)

```

bool SpisakSortiran(const vector<Student>& spisak) {
    unsigned int j, i = 0;

    while (i < spisak.size() - 1) {
        for (j = i + 1; j < spisak.size(); j++) {
            if (spisak[i].m_prezime > spisak[j].m_prezime) {
                return false;
            } else if (spisak[i].m_prezime ==
                spisak[j].m_prezime && spisak[i].m_ime > spisak[j].m_ime) {
                return false;
            }
        }
        i++;
    }
}

```

```

        }
        i++;
    }

    return true;
}

```

90)

```

string MatBroj(const vector<Student>& spisak, const string& prezime){
// po vektoru šetamo sa indeksom

    for( unsigned int i = 0 ; i < spisak.size() ; i++ ) {
        if( spisak[i].m_prezime == prezime ) {
            return spisak[i].m_broj;
        }
    }
    return "00000";
}

```

91)

```

#include <iostream>
#include <vector>
#include <string>
using namespace std;

int main() {
    vector <string> strVec; // definiramo vektor tipa string imena strVec
    string s; // u string s unosimo string

    do {
        cin >> s;
        strVec.push_back(s); // kreiramo objekt tipa strVec koji ima
sve karakteristike vektora
    } while (s != "kraj");

    strVec.pop_back(); // izbacujemo string "kraj"

    for (int i = strVec.size() - 1; i >= 0; i--)
        cout << i << " " << strVec[i] << endl;

    return 0;
}

```

92)

```

#include <iostream>
#include <list>

using namespace std;

bool IsSorted(const list<char>& L) {
    list<char>::const_iterator pos;
    int n = L.size();
    char* A = new char[n];
    int i = 0;

    for (pos = L.begin(); pos != L.end(); ++pos) {
        A[i] = *pos;
    }
}

```

```

        i++;
    }
    int j;
    for (i = 0; i < (n - 1); i++) {
        for (j = i + 1; j < n; j++) {
            if (A[i] > A[j])
                return false;
        }
    }
    return true;
}

int main() {
    list<char> L;

    L.push_back('A');
    L.push_back('B');
    L.push_back('D');
    L.push_back('C');

    cout << IsSorted(L) << endl;

    return 0;
}

```

93)

```

#include <iostream>
#include <list>
#include <string>
using namespace std;

list<int> KopirajStringLista(string var) {
    list<int> c;
    int i;

    for (i = 0; i < var.size(); i++) {
        c.push_back(int(var[i]));
    }
    return c;
}

int main() {
    list<int> lista;
    string ante = "asd";
    lista = KopirajStringLista(ante);

    return 0;
}

```

94)

```

#include <list>
#include <iostream>

using namespace std;

int main() {
    double x;          // broj koji se unosi
    list<double> L;     // lista u kojoj pamtimos unesene brojeve
}

```

```

double sumapozitivnih = 0, sumanegativnih = 0;

while ( cin >> x ) {
    if ( x == 0 )
        break;

    if ( x > 0 ) {
        L.push_front(x);
        sumapozitivnih += x;
    } else {
        L.push_back(x);
        sumanegativnih += x;
    }
}

cout << "suma pozitivnih brojeva: " << sumapozitivnih << endl;
cout << "suma negativnih brojeva: " << sumanegativnih << endl;

return 0;
}

```

95)

```

void SortListToVector(list<double>& L, vector<double>& V) {
    list<double> :: iterator pos;

    // dinamički alocirati ću jedno privremeno polje
    // u koje ću smjestiti članove liste
    int n = L.size();
    double* A = new double[n];

    // kopiram listu u polje
    int i = 0;
    for( pos = L.begin(); pos != L.end(); ++pos ) {
        A[i] = *pos;
        i++;
    }

    // sortiram polje sa selection sort
    int t, j, imin;
    for( i = 0; i < (n-1); i++ ) {
        imin = i;
        for( j = (i+1); j < n ; j++ )
            if( A[j] < A[imin] ) {
                imin=j;
            }
        t = A[i];
        A[i] = A[imin];
        A[imin] = t;
    }

    // kopiram sortirano polje u vektor
    for( i=0 ; i<n ; i++ ) {
        V.push_back( A[i] );
    }
}

```

96)

```

void StringStog::push(const string& str) {
    m_stog.push_back(str);
}

void StringStog::pop(string& str) {
    if( m_stog.size() > 0 ) {
        str = m_stog.back();
        m_stog.pop_back();
    }
}

bool StringStog::operator ==( StringStog& temp) {
    if( temp.m_stog.size() != m_stog.size() ) {
        return false;
    }
    for( int i = 0; i < temp.size() ; i++ ) {
        if( temp.m_stog[i] != m_stog[i] ) {
            return false;
        }
    }
    return true;
}

```

97)

```

int main() {
    StringStog a;
    string s;

    cout << "Unesite proizvoljan broj stringova zaključno sa
            stringom kraj:" << endl;
    cin >> s;
    while( s != "kraj" ) {
        a.push(s);
        cin >> s;
    }
    if( a.size() == 0 ) {
        cout << "prazan" << endl;
    } else {
        cout << a.size() << endl;
        for( int i = 0 ; a.size() > 0 ; i++ ) {
            a.pop(s);
            cout << s << endl;
        }
    }

    return 0;
}

```

98)

```

void SortStringStog(StringStog &s) {
    string tmp;

    int imin;
    for (int i = 0; i < s.size() - 1; i++){
        imin = i;
        for (int j = i + 1; j < s.size(); j++){
            if (s.m_stog[i] < s.m_stog[j]){
                tmp = s.m_stog[i];

```



```

        s.m_stog[i] = s.m_stog[j];
        s.m_stog[j] = tmp;
    }
}
}

```

99)

```

void RedStringova::put(const string& str) {
    m_red.push_back(str);
}

void RedStringova::remove() {
    if( !m_red.empty() ) {
        m_red.erase( m_red.begin() );
    }
}

string RedStringova::get() {
    if( !m_red.empty() ) {
        return m_red[0];
    }
    return "";
}

```

100)

```

RedStringova rs;
string s;

cout << "Unesite proizvoljan broj stringova, zaključno sa praznim
stringom." << endl;

cin >> s;
while( s != "kraj" ) {
    rs.put(s);
    cin >> s;
}
if( rs.empty() ) {
    cout << "nije izvršen unos" << endl;
} else {
    while( !rs.empty() ) {
        cout << rs.get() << endl;
        rs.remove();
    }
}

```

101)

```

void ReverseVector(vector<int>& v) {
    int temp;
    int i;

    for( i = 0 ; i < v.size()/2 ; i++ ) {
        temp = v[i];
        v[i] = v[v.size()-i-1];
        v[v.size()-i-1] = temp;
    }
}

```

102)

```
template <class T> void ReverseArray( T a[], int n ) {
    T temp;
    int i;

    for( i = 0 ; i < n/2 ; i++ ) {
        temp = a[i];
        a[i] = a[n-i-1];
        a[n-i-1] = temp;
    }
}
```

103)

```
template <class T> vector<T> FormirajVektorBezDuplikata(const
vector<T> V) {
    int i, j;
    vector<T> tempV;
    bool found = false;

    for (i = 0; i < V.size(); i++) {
        for (j = 0; j < tempV.size(); j++) {
            if (V[i] == tempV[j]) {
                found = true;
                break;
            }
        }
        if (!found) {
            tempV.push_back(V[i]);
        }
        else {
            found = false;
        }
    }
    return tempV;
}
```

104)

```
int OcjenaUcenika(const vector<CUcenik>& spisak, const string& ime) {
    // u funkciju saljemo vektor te string preko reference

    // PRED: spisak sadrži podatke o ucenicima
    // argumenti: spisak - sadrži vektor podataka o ucenicima
    //             ime - ime i prezime studenta kojem se traži ocjena
    // POST: funkcija vraća ocjenu studenta, ukoliko se ime nalazi
    //       u vektoru spisak, a ako ga nema funkcija vraća 0

    for( unsigned int i = 0 ; i < spisak.size() ; i++ ) {
        if( spisak[i].GetIme() == ime ) {
            return spisak[i].GetOcjena();
        }
    }

    return 0;
}
```