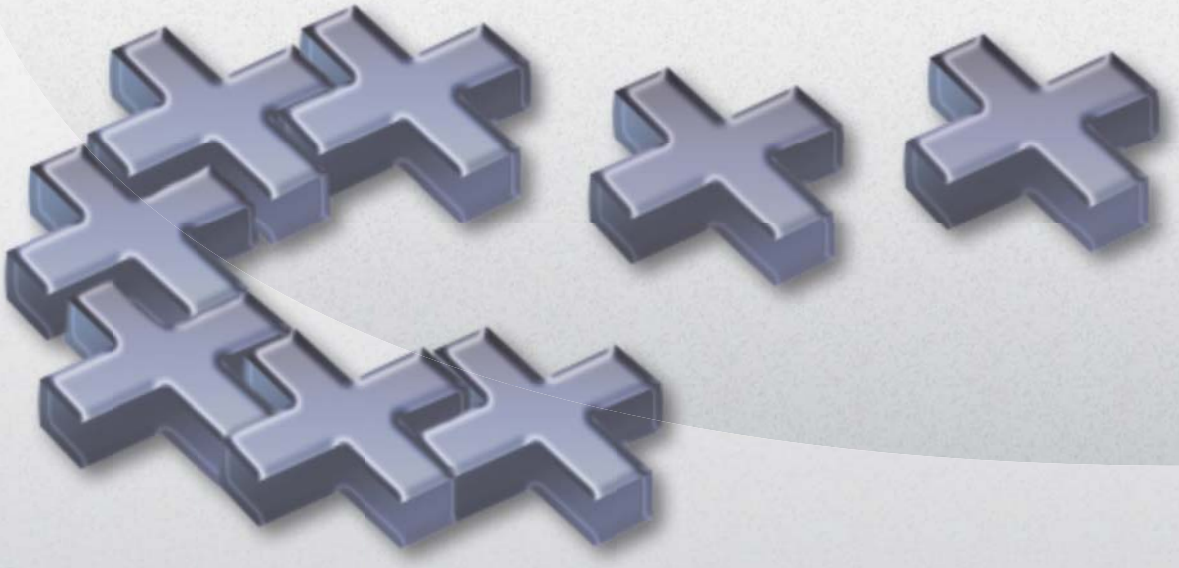
UNIVERZITET SINGIDUNUM



Goran Kunjadić

Zbirka zadataka iz programskog jezika C++

BEOGRAD, 2009.

UNIVERZITET SINGIDUNUM

Fakultet za informatiku i menadžment

mr Goran Kunjadić

ZBIRKA ZADATAKA IZ PROGRAMSKOG JEZIKA C++

Beograd, 2009

ZBIRKA ZADATAKA IZ PROGRAMSKOG JEZIKA C++

Autor:
mr Goran Kunjadić

Recenzent:
Prof. dr Ranko Popović

Izdavač:
UNIVERZITET SINGIDUNUM
FAKULTET ZA INFORMATIKU I MENADŽMENT
Beograd, Danijelova 32

Za izdavača:
Prof. dr Milovan Stanišić

Tehnička obrada:
Goran Kunjadić

Dizajn korica:
Milan Nikolić

Godina izdanja:
2009.

Tiraž:
250 primeraka

Štampa:
ČUGURA print, Beograd
www.cugura.rs

ISBN: 978-86-7912-161-5

Predgovor

Ova zbirka zadataka je namenjena studentima druge godine Fakulteta za Informatiku i Menadžment, Univerziteta Singidunum u Beogradu. U zbirci se obrađuje gradivo koje se izučava u okviru predmeta Uvod u programiranje II. Takođe, ovu zbirku mogu koristiti i drugi koji žele kroz primere da steknu ili prošire svoje znanje iz poznavanja C++ programskog jezika.
Imajući u vidu kontinuitet izučavanja C++ programskog jezika na Univerzitetu Singidunum, javila se potreba za kompaktnom zbirkom zadataka, koja bi pratila gradivo predmeta Uvod u programiranje II. Zbirka je nastala kao rezultat rada sa studentima u prethodnih nekoliko generacija. U zbirci su dati zadaci koji imaju za cilj da studentima olakšaju savladavanje osnova C++ programskog jezika. Potrebno je napomenuti da su u zbirci obrađeni osnovni koncepti objektnog programiranja i da predstavljaju dobru osnovu za dalje usavršavanje u praksi.
Radi lakšeg vežbanja zadataka, prateći deo ove zbirke je kompakt disk koji sadrži source kodove date u ovoj zbirci.
Zadaci su podeljeni u dve osnovne celine i to Osnove C++ programskog jezika i Objektno orijentisano programiranje u C++ programskom jeziku.
U prvom delu su obrađeni neobjektno orijentisani koncepti programiranja u C++ programskom jeziku, što omogućava savladavanje gradiva bez prethodnog znanja nekog od programskih jezika. Ipak, poznavanje C programskog jezika, koji se izučava u okviru predmeta Uvod u programiranje I, je više nego korisno.
Posle uvoda, sledi poglavlje posvećeno operacijama sa realnim brojevima, a zatim se obrađuju iteracije. Nakon toga se razmatra odlučivanje u toku izvršavanja programa, čemu sledi izučavaje funkcija i nizova. U završnom delu poglavlja su objašnjeni pokazivači, c-stringovi, nizovi i funkcije. Na samom kraju ovog poglavlja su opisani korisnički definisani tipovi podataka i tabele.
U drugom delu se obrađuju osnovni principi objektno orijentisanog programiranja, što zapravo čini suštinu C++ programskog jezika. Programski jezik C++ je prvenstveno razvijen radi primene objektnih metoda, iako se u njemu može programirati i na neobjektni način. Ukoliko programer programira u C++ programskom jeziku na neobjektni način, postavlja se pitanje pravilnog izbora jezika za programiranje. U tom slučaju vredi razmisliti o izboru nekog drugog programskog jezika, u kome se vrši neobjektno programiranje.
Drugi deo započinje obradom klase string i time ulazimo u problematiku klasa i objekata, zatim sami generišemo svoje klase, njihove objekte i učimo da ih

koristimo. Na samom kraju izučavamo izvedene klase i nasleđvanje, kao i rad sa fajlovima.

Ovom prilikom se zahvaljujem studentima koji su me naučili kako da ih učim. Veliku zahvalnost izražavam recenzentu, Prof. dr Ranku Popoviću koji mi je pružio niz korisnih sugestija prilikom oblikovanju ove zbirke. Takođe se zahvaljujem profesorima sa Fakulteta za poslovnu informatiku, Univerziteta Singidunum, posebno Prof. dr Milanu M. Milosavljeviću i Prof. dr Mladenu Veinoviću na nesebično pruženoj podršci i strpljenju. Naročito se zahvaljujem svojim prijateljima Rektoru Univerziteta Singidunum, Prof. dr Milovanu Stanišiću koji me je uvek i bezuslovno podržavao i hrabrio i mr Dobrivoju Cvijiću koji mi je pomogao kada mi je bilo najteže.

Posebnu zahvalnost izražavam svojoj supruzi Suzani, sinu Filipu i ćerki Kristini na razumevanju i podršci koju su mi pružali dok sam pripremao materijale za ovu zbirku.

Goran Kunjadić
Januar, 2009.

Sadržaj

Prvi deo - Osnove C++ jezika

Drugi deo - Objektno orijentisano programiranje u C++ programskom jeziku

Prvi deo - Osnove C++ jezika

I Uvod u C++

Primer 1.

Napisati program koji ispisuje poruku na ekranu u dve linije.
U programu se demonstrira ispisivanje poruke na ekranu. Ispisivanjem poruke na ekranu se korisniku saopštavaju informacije u toku izvršavanja programa.

```
// Primer 1

// Ispisivanje poruke na ekranu

#include <iostream>

using namespace std;

int main()
{
    // Ispisivanje poruke

    cout << "Ovo je moj prvi C++ program !" << endl;
    cout << "A jos i radi !!!" << endl;
    cout << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Ovo je moj prvi C++ program !
A jos i radi !!!
```

Znakom `//` se označava komentar. Kompajler ne uzima u obzir ispis nakon ovog znaka i služi isključivo programeru kao podsetnik. Naredba `#include` je petprocesorska direktiva koja govori kompajleru da uključi sadržaj navedenog fajla u kod. U ovom slučaju se koristi biblioteka `<iostream>` - input ouput stream. Opseg - scope važenja promenljivih i funkcija je definisan nakon izraza `namespace`. Glavni program je definisan ključnom reči `main()`. Reč `int` koja prethodi reči `main()` saopštava C++ kompajleru da `main()` predstavlja celobrojnu vrednost. Naredba `return 0;` ima dvostruku ulogu. Prvo, završava izvršavanje funkcije `main()`.

Drugo, šalje vrednost 0 operativnom sistemu kako bi sistem imao informaciju da je program završen regularno.

Primer 2.

Napisati program koji zahteva unos dva cela broja, a zatim ih: sabira, oduzima, množi, deli i prikazuje ostatak pri deljenju, a zatim ispisuje rezultate na ekranu.
U ovom programu se demonstrira način unošenja podataka sa konzole.

```
// Primer 2

// Program ilustruje upotrebu celobrojnih promenljivih u programskom
// jeziku C++.
// Korisnik unosi dva cela broja.
// Program sabira, oduzima, mnozi i deli unete cele brojeve
// i prikazuje rezultate izracunavanja.

#include <iostream>

using namespace std;

int main()
{
    // Deklarisanje promenljivih

    int i1,
        i2,
        zbir,
        razlika,
        proizvod,
        kolicnik,
        ostatak;

    // Ucitavanje podataka sa konzole

    cout << "Unesite ceo broj i pritisnite 'Enter' : ";
    cin  >> i1;

    cout << "Unesite ceo broj i pritisnite 'Enter' : ";
    cin  >> i2;

    // Izracunavanja

    zbir = i1 + i2;
    razlika = i1 - i2;
    proizvod = i1 * i2;
    kolicnik = i1 / i2;
    ostatak = i1 % i2;

    // Ispisivanje rezultata

    cout << endl;
```

```
    cout << "Zbir brojeva " << i1 << " i " << i2 << " je " << zbir <<
            endl;
    cout << "Razlika brojeva " << i1 << " i " << i2 << " je "
         << razlika << endl;
    cout << "Proizvod brojeva " << i1 << " i " << i2 << " je "
         << proizvod << endl;
    cout << "Kolicnik brojeva " << i1 << " i " << i2 << " je "
         << kolicnik << endl;
    cout << "Ostatak pri deljenju broja " << i1 << " brojem " << i2
         << " je " << ostatak << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite ceo broj i pritisnite 'Enter' : 15
Unesite ceo broj i pritisnite 'Enter' : 6

Zbir brojeva 15 i 6 je 21
Razlika brojeva 15 i 6 je 9
Proizvod brojeva 15 i 6 je 90
Kolicnik brojeva 15 i 6 je 2
Ostatak pri deljenju broja 15 brojem 6 je 3
```

Celobrojne promenljive su definisane tako što su nakon ključne reči `int` navedena imena promenljivih. Ukoliko se vrši aritmetička operacija u kojoj učestvuju dva cela broja, rezultat je takođe ceo broj.

Primer 3.

Napisati program koji izračunava cenu servisiranja uređaja ako se cena delova i broj radnih sati unose sa tastature, dok se cena radnog sata definiše kao konstanta. Sve vrednosti su celi brojevi. Koristiti formatirano ispisivanje iznosa, poravnati ih po desnoj strani.
U programu je pokazan način definisanja konstantne veličine – veličine koja ne menja svoju vednost za vreme izvršavanja programa.

```
// Primer 3

// Program izracunava ukupnu cenu delova i rada

#include <iostream>
#include <iomanip>

using namespace std;
```

```
int main()
{
    // Deklarisanje promenljivih

    const int CENA_PO_SATU = 44;

    int delovi,                 // Cena delova
        sati,                   // Broj utrosenih radnih sati
        rad,                    // Cena rada
        ukupno;                 // Ukupo za naplatu

    // Unos podataka

    cout << "Unesite cenu delova : ";
    cin >> delovi;
    cout << endl;

    cout << "Uneste broj utrosenih radnih sati : ";
    cin >> sati;
    cout << endl;

    // Izracunavanja

    rad = CENA_PO_SATU * sati;
    ukupno = delovi + rad;

    // Ispisivanje rezultata

    cout << "Cena delova : " << setw(6) << delovi << endl;
    cout << "Cena rada : " << setw(8) << rad << endl;
    cout << "-------------------" << endl;
    cout << "Ukupna cena : " << setw(6) << ukupno << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite cenu delova : 239

Uneste broj utrosenih radnih sati : 2

Cena delova :   239
Cena rada :      88
-------------------
Ukupna cena :   327
```

Ukoliko se prilikom deklarisanja promenljive, navede ključna reč `const`, na taj način se kompajleru saopštava da je deklarisana veličina konstantna za sve vreme izvršavanja programa. Manipulator `setw` omogućava formatirano ispisivanje brojne vrednosti, poravnate po desnoj strani. Broj u zagradi označava broj mesta

rezervisanih za ispisivanje brojčane vrednosti. Takođe je uvedena biblioteka `<iomanip>` - input output manipulation.

Primer 4.

Prethodni primer izmeniti tako da može računati sa većim iznosima, korišćenjem promenljivih tipa long integer.

```
// Primer 4

// Program izracunava ukupnu cenu delova i rada
// Razlika izmedju ovog i prethodnog primera je u koriscenju
// razlicitog tipa promenljivih

#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
    // Deklarisanje promenljivih

    const int CENA_PO_SATU = 44;

    long delovi,          // Cena delova
         sati,            // Broj utrosenih radnih sati
         rad,             // Cena rada
         ukupno;          // Ukupno za naplatu

    // Unos podataka

    cout << "Unesite cenu delova : ";
    cin >> delovi;
    cout << endl;
    cout << "Uneste broj utrosenih radnih sati : ";
    cin >> sati;

    // Izracunavanja

    rad = CENA_PO_SATU * sati;
    ukupno = delovi + rad;

    // Ispisivanje rezultata

    cout << endl;
    cout << "Cena delova : " << setw(9) << delovi << endl;
    cout << "Cena rada : " << setw(11) << rad << endl;
    cout << "-----------------------"           << endl;
    cout << "Ukupna cena : " << setw(9) << ukupno << endl << endl;

    return 0;
```

```
}
```

Rezultat izvršavanja programa:

```
Unesite cenu delova : 52731

Uneste broj utrosenih radnih sati : 97

Cena delova :     52731
Cena rada :        4268
----------------------
Ukupna cena :     56999
```

Razlika u odnosu na prethodni primer je jedino u tipu promenljivih koje se koriste. Korišćene su promenljive tipa long koje omogućavaju obradu vrednosti većih nego u prethodnom prmeru.

II Realni brojevi

Primer 5.

Napisati program koji izračunava porez na nabavnu cenu i maloprodajnu cenu proizvoda. Stopa poreza je konstantna, dok se nabavna cena unosi sa tastature. Rezultate prikazati u formatiranom ispisu.
Za promenljive koristiti realne brojeve tipa `double`.

```
// Primer 5

// Program izracunava porez i maloprodajnu cenu proizvoda

#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
    // Deklarisanje promenljivih

    const double STOPA_POREZA = 0.0825;
    double nabavna_cena,        // Nabavna cena proizvoda
           porez,               // Porez na posmatrani proizvod
           cena;                // Prodajna cena (nabavna_cena +
                                // porez)

    //Podesavanje izlaznog niza za prikazivanje iznosa

    cout << setprecision(2)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);

    // Unos podataka

    cout << "Unesite nabavnu cenu proizvoda : ";
    cin >> nabavna_cena;

    // Izracunavanja

    porez = nabavna_cena * STOPA_POREZA;
    cena = nabavna_cena + porez;

    // Ispisivanje rezultata

    cout << endl;
    cout << "Nabavna cena je : " << setw(11) << nabavna_cena << endl;
    cout << "Porez je : " << setw(18) << porez << endl;
    cout << "Prodajna cena je : " << setw(10) << cena << endl << endl;

    return 0;
```

```
}
```

Rezultat izvršavanja programa:

```
Unesite nabavnu cenu proizvoda : 100.00

Nabavna cena je :      100.00
Porez je :               8.25
Prodajna cena je :     108.25
```

U ovom primeru su uvedene promenljive tipa `double`, što su zapravo realni brojevi. Ispis realnih brojeva se uvek vrši u eksponencijalnom obliku, ukoliko drugačije ne naglasimo. Činjenica je da cene nije pogodno pisati u eksponencijalnom obliku već u fiksnom i to sa dva decimalna mesta. Naredba `setprecision(2)` definiše broj decimalnih mesta, `setiosflags(ios::fixed)` govori da će ispis biti u fiksnom obliku dok naredba `setiosflags(ios::showpoint)` 'traži' od kompajlera da upotrebi decimalnu tačku da bi razdvojio celobrojni deo od decimalnog.

Primer 6.

Napisati program za kasu u restoranu koji izračunava maloprodajnu cenu obroka, kao i kusur koji je potrebno vratiti gostu restorana na osnovu cene obroka i datog iznosa. Izvršiti ispisivanje pozdravne i završne poruke.
Za promenljive koristiti realne brojeve tipa `double`.

```
// Primer 6

// Program za kasu u restoranu

#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
    // Deklarisanje promenljivih

    const double STOPA_MALOPRODAJNOG_POREZA = 0.0825;

    double cena_obroka,    // Nabavna cena obroka za restoran
           iznos_poreza,   // Iznos poreza
           ukupno,         // Ukupno za naplatu
           dati_iznos,     // Iznos koji je dala musterija
           kusur;          // Kusur: dati_iznos - ukupno

    //Podesavanje izlaznog formata za ispisivanja iznosa
```

```
    cout << setprecision(2)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);

    // Ispisivanje naziva restorana i unos cene obroka

    cout << "***** Kod Radojka *****" << endl << endl;
    cout << "Ibarska magistrala b.b." <<endl << endl;
    cout << "Unesite cenu obroka : EUR ";
    cin >> cena_obroka;
    cout << endl;

    // Izracunavanje poreza i ukupne cene

    iznos_poreza = cena_obroka * STOPA_MALOPRODAJNOG_POREZA;
    ukupno = cena_obroka + iznos_poreza;

    // Ispisivanje poreza i ukupne cene

    cout << endl;
    cout << "Cena obroka : " << setw(7) << cena_obroka << endl;
    cout << "Porez : " << setw(13) << iznos_poreza << endl;
    cout << "---------------------" << endl;
    cout << "Ukupan uznos : " << setw(6) << ukupno << endl;

    // Unos datog iznosa

    cout << endl;
    cout << "Unesite iznos koji je musterija dala : EUR ";
    cin >> dati_iznos;
    cout << endl;

    // Izracunavanje kusura

    kusur = dati_iznos - ukupno;

    // Ispisivanje kusura

    cout << endl;
    cout << "Dati iznos :   EUR " << setw(9) << dati_iznos
         << endl;
    cout << "Ukupan iznos : EUR " << setw(9) << ukupno << endl;
    cout << "----------------------------" << endl;
    cout << "Kusur :        EUR " << setw(9) << kusur << endl;

    // Ispisivanje zavrsne poruke

    cout << endl << endl;
    cout << "*** Zivi bili i dosli nam opet ! ***" << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
***** Kod Radojka *****

Ibarska magistrala b.b.

Unesite cenu obroka : EUR 52.95

Cena obroka :   52.95
Porez :          4.37
----------------------
Ukupan uznos :  57.32

Unesite iznos koji je musterija dala : EUR 70.00

Dati iznos :   EUR  70.00
Ukupan iznos : EUR  57.32
-------------------------
Kusur :        EUR  12.68

*** Zivi bili i dosli nam opet ! ***
```

U ovm primeru su korišćene već objašnjene metode radi rešavanja praktičnog problema.

Primer 7.

Napisati program koji konvertuje dužine unete u miljama i stopama u dužine izražene u kilometrima i metrima.
U programu se demonstrira kombinovano korišćenje promenljivih različitog tipa.

```
// Primer 7

// Preracunavanje duzine iz anglosaksonskog u metricki sistem
// U programu se mogu videti primeri racunanja sa razlicitim tipovima
promenljivih

#include <iostream>

using namespace std;

int main()
{
    // Deklarisanje promenljivih

    const double METARA_PO_MILJI = 1609.35;
    const double METARA_PO_STOPI = 0.30480;

    int  milje,
         stope,
```

```
          kilometri,
          metri;
     double ukupno_metri,
            ukupno_kilometri;

     // Unos podataka

     cout << "Unesite broj milja : ";
     cin >> milje;
     cout << "Unesite broj stopa : ";
     cin >> stope;

     // pretvaranje unete duzine u metre

     ukupno_metri = milje * METARA_PO_MILJI + stope * METARA_PO_STOPI;

     // Izracunavanje broja kilometara

     ukupno_kilometri = ukupno_metri / 1000;
     kilometri = ukupno_kilometri;     // odbacivanje decimalnog dela i
                                       // dobijanje celog broja
                                       // kilometara

     // Preracunavanje decimalnog dela kilometara u metre

     metri = (ukupno_kilometri - kilometri) * 1000;

     // Ispisivanje rezultata

     cout << endl;
     cout << "Rastojanje je " << kilometri << " km i "
          << metri << " m" << endl << endl;

     return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite broj milja : 2
Unesite proj stopa : 1000

Rastojanje je 3 km i 523 m
```

Potrebno je primetiti da se rezultat izračunavanja koji se smešta u promenljivu određenog tipa konvertuje u tip kome promenljiva pripada.

Primer 8.

U narednom programu se demonstrira upotreba složenih binarnih opertora i to `+=`, `-=`, `*=` kao i `/=`.

```
// Primer 8

// Program ilustruje dodeljivanje vrednosti primenom slozenih
// operatora

#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
    // Deklarisanje promenljivih

    double stanje     = 545.50,
           ulog       = 300.00,
           podizanje  = 200.00,
           broj       =   7.5,
           cena       = 104.50;

    //Podesavanje izlaznog formata za ispisivanja iznosa

    cout << setprecision(2)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);

    // Izracunavanja i ispisivanje rezultata na ekranu

     stanje += ulog;
     cout << "Stanje je " << stanje << endl;

     stanje -= podizanje;
     cout << "Stanje je " << stanje << endl;

     broj *= broj;
     cout << "Broj je " << broj << endl;

     cena /= 2.0;
     cout << "Cena je " << cena << endl << endl;

     return 0;
}
```

Rezultat izvršavanja programa:

```
Stanje je 845.50
Stanje je 645.50
Broj je 56.25
Cena je 52.25
```

Upotrebom navedenih operatora postiže se kraće pisanje koda i predstavlja specifičnost koja je uvedena pojavom jezika C, C++ i C#.

Primer 9.

Naredni program ilustruje upotrebu unarnih operatora ispred i iza promenljive. Upotreba operatora kao u sledećem primeru je veoma česta u C++ jeziku. Zato je neophodno dobro razumeti sledeći primer.

```
// Primer 9

// Program ilustruje razliku izmedju primene opearotra pre
// i posle promenljive

#include <iostream>

using namespace std;

int main()
{
    // Deklarisanje promenljivih

    int i,
        j;

    // Izracunavanja i ispisivanje rezultata

    i = 7;                      // Dodeljivanje inicijalne vrednosti
    j = 4 + --i;                // prvo se od i oduzme 1 a zatim se
                                // sabere sa 4

    cout << "Pri koriscenju operatora pre promenljive i = "
    << i << " dok je j = " << j << endl;

    i = 7;                      // Ponovna inicijalizacija promenljive
                                // i na 7
    j = 4 + i--;                // prvo se i sabere sa 4 a zatim mu se
                                // oduzme 1

    cout << "Pri koriscenju operatora posle promenljive i = "
         << i << " dok je j = " << j << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Pri koriscenju operatora pre promenljive i = 6 dok je j = 10
Pri koriscenju operatora posle promenljive i = 6 dok je j = 11
```

Unarni operatori se odnose samo na jednu promenljivu i to tako što njenu vrednost povećavaju ili smanjuju za 1. Ukoliko se operator nalazi ispred promenljive, vrednost promenljive se smanji/poveća za 1, a zatim ulazi u aritmetičku operaciju. Ukoliko se operator nalazi iza promenljive, izvrši se aritmetička operacija, a zatim se vrednost promenljive smanji/poveća za 1.

III Iteracije

Primer 10.

Napisati program koji broji znakove unete sa konzole u jednoj liniji. U programu koristiti `while` petlju.
U prikazanoj `while` petlji, uslov koji mora biti ispunjen da bi se petlja izvršavala, nalazi se na početku petlje. Ukoliko uslov nije ispunjen, sekvenca naredbi u okviru petlje se neće ni jednom izvršiti.

```
// Primer 10

// Program broji unete znakove koje je upisao korisnik u jednoj liniji
// Linija se zavrsava pritiskom na Enter. Enter se ne racuna kao
karakter
// Primer iteracije.

#include <iostream>

using namespace std;

int main()
{
    // Deklarisanje promenljivih

    char znak;                  // Koristi se za prihvat unetog znaka
    int broj_znakova = 0;       // Broji unete znakove

    // Unos niza

    cout << "Unesite bilo koji broj znakova  i na kraju pritisnite "
         << "'Enter'"
         << endl << endl;

    znak = cin.get();           // Unos prvog znaka

    // Petlja za sabiranje znakova unetih u jednom redu

    while (znak != '\n')        // While uslov se nalazi na pocetku
                                // petlje
    {
        ++broj_znakova;
        znak = cin.get();       // Unos sledeceg znaka
    }

    // Ispisivanje rezultata

    cout << endl;
    cout << "Uneli ste " << broj_znakova << " znakova" << endl
         << endl;
```

```
    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite bilo koji broj znakova  i na kraju pritisnite 'Enter'

aaB1, 34

Uneli ste 8 znakova
```

U programu se koristi i escape sekvenca `\n` što označava kraj reda, odnosno proverava da li je pritisnut taster 'Enter'. Escape sekvence ili kontrolne sekvence su grupe karaktera koje menjaju stanje sistema.

Primer 11.

Izmeniti program iz primera 6, tako da može da obradi proizvoljan broj mušterija upotrebom `while` petlje.
Kod za izračunavanje se smešta u okviru `while` petlje tako da se može ponavljati željeni broj puta.

```
// Primer 11

// Program je namenjen za kasu u restoranu.
// Obradjuje proizvoljan broj musterija.
// Na kraju programa se ispisuje ukupan broj musterija
// kao i ukupan promet

#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
    // Deklarisanje promenljivih

    const double STOPA_MALOPRODAJNOG_POREZA = 0.0825;

    double cena_obroka,    // Nabavna cena obroka za restoran
           iznos_poreza,   // Iznos poreza
           ukupno,         // Ukupno za naplatu
           dati_iznos,     // Iznos koji je dala musterija
           kusur,          // Kusur: dati_iznos - ukupno
           ukupan_promet;  // Ukupan promet za posmatrani period

    int    broj_musterija,  // Sabira ukupan broj obradjenih musterija
```

```
       odgovor;        // da = 1, ne = 0

//Podesavanje izlaznog formata za ispisivanja iznosa

cout << setprecision(2)
     << setiosflags(ios::fixed)
     << setiosflags(ios::showpoint);

// Inicijalizacija brojaca i zbira

ukupan_promet = 0.0;
broj_musterija = 0;

// Ispisivanje naziva restorana

cout << "***** Kod Radojka *****" << endl << endl;
cout << "Ibarska magistrala b.b." <<endl << endl;

// Ispitivanje da li se nastavlja unos sledece musterije

cout << endl;
cout << "Da li ima jos neko ko hoce da plati ?" << endl;
cout << "Ako ima kucaj 1 a ako nema kucaj 0 : ";
cin >> odgovor;

// Glavna petlja

while (odgovor == 1)                    // While uslov se nalazi na
                                        // pocetku petlje
{
    // Unos cene obroka

    cout << endl;
    cout << "Za koliko para su pojeli i popili : EUR ";
    cin >> cena_obroka;
    cout << endl << endl;

    // Izracunavanje poreza i ukupne cene

    iznos_poreza = cena_obroka * STOPA_MALOPRODAJNOG_POREZA;
    ukupno = cena_obroka + iznos_poreza;

    // Ispisivanje poreza i ukupne cene

    cout << "Pojeli i popili : " << setw(7) << cena_obroka <<
            endl;
    cout << "Porez : " << setw(17) << iznos_poreza << endl;
    cout << "-------------------------" << endl;
    cout << "Ukupan uznos : " << setw(10) << ukupno << endl <<
            endl;

    // Unos datog iznosa

    cout << "Unesite iznos koji je musterija dala : EUR ";
    cin >> dati_iznos;
    cout << endl << endl;
```

```
        // Izracunavanje kusura

        kusur = dati_iznos - ukupno;

        // Ispisivanje kusura

        cout << endl;
        cout << "Dati iznos :   EUR " << setw(9) << dati_iznos
        << endl;
       cout << "Ukupan iznos : EUR " << setw(9) << ukupno << endl;
       cout << "-----------------------------" << endl;
       cout << "Kusur :        EUR " << setw(9) << kusur << endl;

        // Povecanje brojaca i ukupnog prometa

        ++broj_musterija;
        ukupan_promet += ukupno;

        // Ispitivanje da li se nastavlja unos sledece musterije

        cout << endl;
        cout << "Da li ima jos neko ko hoce da plati ?" << endl;
        cout << "Ako ima kucaj 1 a ako nema kucaj 0 : ";
        cin >> odgovor;
    }

    // Ispisivanje ukupnog broja musterija i ukpnog prometa

    cout << endl;
    cout << "Ukpno : " << endl << endl;
    cout << "Broj musterija : " << broj_musterija << endl;
    cout << "Promet :  " << ukupan_promet << endl;

    // Ispisivanje zavrsne poruke

    cout << endl;
    cout << "*** Zivi bili i dosli nam opet ! ***" << endl;

    cout << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
***** Kod Radojka *****

Ibarska magistrala b.b.

Da li ima jos neko ko hoce da plati ?
Ako ima kucaj 1 a ako nema kucaj 0 :  1

Za koliko para su pojeli i popili : EUR 34.50

Pojeli i popili :   34.50
Porez :              2.85
-------------------------
Ukupan uznos :      37.35

Unesite iznos koji je musterija dala : EUR 40.00

Dati iznos :   EUR 40.00
Ukupan iznos : EUR 37.35
-------------------------
Kusur :        EUR 2.65

Da li ima jos neko ko hoce da plati ?
Ako ima kucaj 1 a ako nema kucaj 0 :  1

Za koliko para su pojeli i popili : EUR 75.23

Pojeli i popili :   75.23
Porez :              6.21
-------------------------
Ukupan uznos :      81.44

Unesite iznos koji je musterija dala : EUR 100.00

Dati iznos :   EUR100.00
Ukupan iznos : EUR 81.44
-------------------------
Kusur :        EUR 18.56

Da li ima jos neko ko hoce da plati ?
Ako ima kucaj 1 a ako nema kucaj 0 :  0

Ukpno :

Broj musterija : 2
Promet : 118.78

*** Zivi bili i dosli nam opet ! ***
```

Kompletan blok naredbi je smešten unutar `while` petlje tako da se može izvršavati više puta sve dok je ispunjen uslov petlje, dat na početku petlje odnosno sve dok je `odgovor == 1`.

Primer 12.

Napisati program koji izračunava mesečnu kamatu na štednju primenom `for` petlje. Uslov za izvršavanje petlje se nalazi iza ključne reči `for`. Primetimo da se ispisivanje rezultata i izračunavanja takođe vrše u okviru `for` petlje, što se često koristi u cilju racionalizacije programskog koda.

```
// Primer 12

// Program ilustruje upotrebu for petlje za izracunavanje mesecne
// kamate.
// Korisnik unosi visinu uloga, rok i kamatnu stopu.
// Program daje listu mesecnih kamata, ukupnu kamatu
// i stanje racuna za svaki period.

#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
    //Deklarisanje promenljivih

    double godisnja_kamata, // Godisnja kamata u %
           mesecna_kamata,  // Mesecna kamata
           ulog,            // Visina uloga
           stanje_racuna,   // Mesecno stanje racuna
           kamatni_iznos,   // Kamatni iznos
           ukupna_kamata;   // Ukupna kamata
    int    mesec,           // Racuna period prikazan na izvodu sa
                            // racuna
           period;          // Period na izvodu sa racuna u mesecima

    //Podesavanje izlaznog formata za ispisivanja iznosa

    cout << setprecision(2)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);

    // Unos podataka

    cout << "Unesite visinu uloga : ";
    cin >> ulog;
    cout << endl;
    cout << "Unesite period izrazen u broju meseci : ";
    cin >> period;
    cout << endl;
    cout << "Unesite godisnju kamatu u % : ";
```

```
    cin >> godisnja_kamata;

    // Izracunavanja i ispisivanje rezultata

    mesecna_kamata = (godisnja_kamata / 100) / 12;
    ukupna_kamata = 0.00;
    stanje_racuna = ulog;

    cout << endl << endl;
    cout << "            MESECNI          UKUPAN      NOVO" << endl;
    cout << " MESEC   KAMATNI IZNOS   KAMATNI IZNOS   STANJE" << endl;
    cout << "-----------------------------------------------";

    for (mesec = 1; mesec <= period; ++mesec)
    {
        kamatni_iznos = mesecna_kamata * stanje_racuna;
        stanje_racuna += kamatni_iznos;
        ukupna_kamata += kamatni_iznos;

        cout << endl << setw(4) << mesec
             << setw(14) << kamatni_iznos
             << setw(16) << ukupna_kamata
             << setw(14) << stanje_racuna;
    }

    cout << endl;
    cout << "-----------------------------------------------" << endl
         << endl;
    cout << "\tUkupno" << endl << endl;
    cout << "Pocetni ulog :  " << setw(8) << ulog << endl;
    cout << "Kamata :        " << setw(8) << ukupna_kamata << endl;
    cout << "Krajnja suma :  " << setw(8) << stanje_racuna << endl
         << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite visinu uloga : 1000.00

Unesite period izrazen u broju meseci : 12

Unesite godisnju kamatu u % : 7.5

            MESECNI          UKUPAN      NOVO
 MESEC   KAMATNI IZNOS   KAMATNI IZNOS   STANJE
-------------------------------------------------
   1          6.25             6.25        1006.25
   2          6.29            12.54        1012.54
   3          6.33            18.87        1018.87
   4          6.37            25.24        1025.24
   5          6.41            31.64        1031.64
   6          6.45            38.09        1038.09
   7          6.49            44.58        1044.58
   8          6.53            51.11        1051.11
   9          6.57            57.68        1057.68
  10          6.61            64.29        1064.29
  11          6.65            70.94        1070.94
  12          6.69            77.63        1077.63
-------------------------------------------------

        Ukupno

Pocetni ulog :   1000.00
Kamata :           77.63
Krajnja suma :   1077.63
```

For petlja se definiše: početnom vrednošću brojača, uslovom koji treba biti ispunjen da bi se petlja izvršavala, kao i inkrementom/dekrementom samog brojača. Petlja će se izvršavati sve dok je ispunjen navedeni uslov. Korišćena escape sekvenca `\t` definiše tabulator.

Primer 13.

Napisati program koji prebrojava znakove unete sa konzole, u više linija korišćenjem ugneždene `while` petlje.
Ugneždena petlja se nalazi u okviru spoljašnje petlje i može se izvršiti više puta ili ni jednom ukoliko uslov ugneždene petlje nije ispunjen, u toku jednog izvršavanja spoljašnje petlje.

```
// Primer 13

// Program koristi ugnezdenu petlju za izracunavanje broja
// znakova koji se unesu u nekoliko linija.
```

```
#include <iostream>

using namespace std;

int main()
{
    // Deklarisanje promenljivih

    int znak;          // Promenljiva u koju se smesta znak unet od
                       // strane korisnika
    int broj_znakova = 0;    // Broji unete karaktere

    // Ispisivanje poruke sa uputstvom za korisnika

    cout << "Ovaj program broji unete znakove sa tastaure." << endl
         << "Unesite  neki broj znakova u svaki red" << endl
         << "a dobicete ukupan broj unetih znakova. " << endl
         << "Zavrsite svaki red sa Enter." << endl
         << "Zavrsite unos priskom na Enter, [Ctrl]+Z, a zatim Enter."
         << endl << endl;

    // Unos i brojanje znakova

    while ((znak = cin.get()) != EOF)  // Petlja koja se izvrsava
                                       // sve dok nije kraj fajla
    {
        ++broj_znakova;

        while ((znak = cin.get()) != '\n')   // Petlja koja se
                                    // izvrsava sve dok nije kraj reda
            ++broj_znakova;
    }

    // Ispisivanje rezultata

    cout << endl;
    cout << "Uneli ste " << broj_znakova << " znakova." << endl
         << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Ovaj program broji unete znakove sa tastaure.
Unesite  neki broj znakova u svaki red
a dobicete ukupan broj unetih znakova..
Zavrsite svaki red sa Enter.
Zavrsite unos priskom na Enter, [Ctrl]+Z, a zatim Enter.

abc
defg
hijkl
^Z

Uneli ste 12 znakova.
```

Petlja unutar petlje ili ugneždena petlja se često koristi u C++ programskom jeziku. Korišćenjem ugneždene petlje omogućava se rešavanje problema sa dvodimenzionim i višedimenzionim nizovima/matricama, o čemu će biti reči u primerima koji slede.

Primer 14.

Napisati program koji crta pravougli trougao znacima 'X' korišćenjem ugneždene `for` petlje.
Kao i `while` petlja u prethodnom primeru, tako i `for` petlja može biti unutar spoljašnje `for` petlje.

```
// Primer 14

// Program demonstrira ugnezdenu petlju
// prikazujuci redove koji se sastoje od znaka X.
// Broj iteracija unutrasnje for petlje zavisi od vrednosti brojaca
// koji kontrolise spoljasnju for petlju.

#include <iostream>

using namespace std;

int main()
{
    // Deklarisanje promenljivih

    const int BROJ_REDOVA = 22;

    int trenutni_broj_koraka,
        brojac_x;

    // Ispisivanje rezultata
```

```
    for (trenutni_broj_koraka = 1; trenutni_broj_koraka <=
         BROJ_REDOVA; ++trenutni_broj_koraka)
    {
        cout << endl;

        for (brojac_x = 1; brojac_x <= trenutni_broj_koraka;
             ++brojac_x)
            cout << "X";
    }

    cout << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
X
XX
XXX
XXXX
XXXXX
XXXXXX
XXXXXXX
XXXXXXXX
XXXXXXXXX
XXXXXXXXXX
XXXXXXXXXXX
XXXXXXXXXXXX
XXXXXXXXXXXXX
XXXXXXXXXXXXXX
XXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
```

Veličina `BROJ_REDOVA` definiše broj izvršavanja spoljašnje petlje dok veličina `brojac_x` definiše broj izvršavanja unutrašnje petlje.

IV Odlučivanje

Primer 15.

Napisati program koji računa platu za zaposlene. Uzeti u obzir da se prekovremeni rad plaća 50% više od redovnog. Predvideti mogućnost obrade podataka za više zaposlenih, kao i sumiranje plata na kraju programa.

```
// Primer 15

// Program izracunava ukupnu platu za svakog zaposlenog
// ukljucujuci i prekovremeni rad koji se placa 1.5 puta vise.
// Program takodje izracunava ukupne plate za sve obradjne zaposlene.
// Program ilustruje upotrebu if naredbe.

#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
    // Deklarisanje promenljivih

    const double FAKTOR_PREKOVREMENOG_RADA = 1.5;
    const double REGULARAN_BROJ_SATI = 40.0;

    int   brojac_zaposlenih,
          sledeci_zaposleni;     // 1 ako postoji sledeci;
                                 // 0 ako ne postoji

    double radni_sati,
           satnica,
           regularna_plata,
           prekovremena_plata,
           ukupna_plata,
           sve_plate;

    // Inicijalizacija promenljivih

    sve_plate = 0.00;
    brojac_zaposlenih = 0;

    //Podesavanje izlaznog formata za ispisivanja iznosa

    cout << setprecision(2)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);

    do  // Pocetak while petlje
    {
        // Unos podataka
```

```
        cout << endl;
        cout << "Unesite broj ostvarenih radnih sati : ";
        cin >> radni_sati;
        cout << "\nUnesite satnicu : ";
        cin >> satnica;

        // Izracunavanje plate

        if (radni_sati > REGULARAN_BROJ_SATI)
        {
            regularna_plata = REGULARAN_BROJ_SATI * satnica;
            prekovremena_plata = (radni_sati - REGULARAN_BROJ_SATI) *
                                 FAKTOR_PREKOVREMENOG_RADA * satnica;
        }
        else
        {
            regularna_plata = radni_sati * satnica;
            prekovremena_plata = 0.00;
        }

        ukupna_plata = regularna_plata + prekovremena_plata;
        sve_plate += ukupna_plata;
        ++brojac_zaposlenih;

        // Ispisivanje plate

        cout << endl << endl;
        cout << "REGULARNO    PREKOVREMENO     UKUPNO";
        cout << endl << setw(9) << regularna_plata
             << setw(16) << prekovremena_plata
             << setw(11) << ukupna_plata << endl;
        cout << "-----------------------------------" << endl;

        // Pitanje korisniku da li zeli da nastavi sa sledecim
        // zaposlenim

        cout << endl << endl;
        cout << "Da li zelita da nastavite sa sledecim zaposlenim ?"
             << endl;
        cout << "Unesite 1 za Da ili 0 za Ne : ";
        cin >> sledeci_zaposleni;
    }
    while (sledeci_zaposleni);    // Uslov while petlje - na kraju
                                  // bloka naredbi

    // Ispisivanje zbira svih plata

    cout << endl << endl;
    cout << "Ukupna plata za " << brojac_zaposlenih
         << " zaposlenih je " << sve_plate << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite broj ostvarenih radnih sati : 53

Unesite satnicu : 6.50

REGULARNO    PREKOVREMENO     UKUPNO
   260.00          126.75     386.75
-------------------------------------

Da li zelita da nastavite sa sledecim zaposlenim ?
Unesite 1 za Da ili 0 za Ne : 1

Unesite broj ostvarenih radnih sati : 36

Unesite satnicu : 4.00

REGULARNO    PREKOVREMENO     UKUPNO
   144.00            0.00     144.00
-------------------------------------

Da li zelita da nastavite sa sledecim zaposlenim ?
Unesite 1 za Da ili 0 za Ne : 1

Unesite broj ostvarenih radnih sati : 45.5

Unesite satnicu : 10.00

REGULARNO    PREKOVREMENO     UKUPNO
   400.00           82.50     482.50
-------------------------------------

Da li zelita da nastavite sa sledecim zaposlenim ?
Unesite 1 za Da ili 0 za Ne : 0

Ukupna plata za 3 zaposlenih je 1013.25
```

U okviru programa se koristi `if` naredba iza koje se navodi uslov. Ukoliko je uslov uspunjen izvršava se blok naredbi koji sledi. Ukoliko uslov nije ispunjen blok naredbi koji sledi se ne izvršava već se izvršava blok naredbi koji je definisan iza ključne reči `else`. Na taj način se vrši odlučivanje u programu na osnovu zadatih uslova. U programu je obrađen konkretan problem izračunavanja plata.

Primer 16.

Napisati program koji odlučuje da li je pritisnuto dugme cifra.

```
// Primer 16

// Program prikazuje koriscenje slozene if naredbe
// koja odlucuje da li je pritisnutuo dugme broj.
// Program koristi cin.get() za unos znaka.

#include <iostream>

using namespace std;

int main()
{
    // Deklarisanje promenljivih

    char znak;

    // Unos znaka sa tastature

    cout << "Pritisnite neko dugme na tastaturi, a zatim pritisnite "
         << "Enter : ";

    znak = cin.get();

    cout << endl;

    // Provera unetog znaka i ispisivanje rezultata

    if ( (znak >= '0') && (znak <= '9'))
        cout << "Pritisnuli ste dugme koje je cifra." << endl;
    else
        cout << "Pritisnuli ste dugme koje nije cifra." << endl
             << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Pritisnite neko dugme na tastaturi, a zatim pritisnite Enter : a

Pritisnuli ste dugme koje nije cifra.
```

Program ispituje znak koji je unet naredbom `cin.get()` i donosi odluku da li je uneti znak cifra. Uslov koji je naveden zapravo upoređuje ASCII vrednosti unetih znakova. Primetimo da su u ASCII kodnom sistemu, vrednosti cifara od 0 do 9 poređane sukcesivno.

Primer 17.

Napisati program koji izračunava plate zaposlenih u manjoj kompaniji. U programu moraju biti posebno obrađivani domaći, a posebno strani radnici. Potrebno je izračunati i porez na platu savakog zaposlenog, a na kraju prikazati zbirne podatke.

```
// Primer 17

// Program obradjuje plate za malu kompaniju.
// Izracunava ukupnu platu ukljucujuci i moguci prekovremeni rad.
// Takodje racuna i visinu poreza na platu za
// domace i strane radnike. Program prikazuje ukupnu platu,
// porez, i neto platu za svakog zaposlenog. Kada nema vise
// zaposlenih za obradu, program ispisuje zbirne informacije.

#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
    // Deklarisanje promenljivih

    const double KOEFICIJENT_ZA_DOMACE_RADNIKE = 0.070;
    const double KOEFICIJENT_ZA_STRANE_RADNIKE = 0.045;

    int  broj_zaposlenih;

    char kod_zaposlenog,
         kod_drzave,
         odgovor;

    double    satnica,
              radni_sati,
              regularni_rad,
              prekovremeni_rad,
              ukupna_plata,
              porez,
              neto_plata,
              suma_ukupno_plate,
              suma_porez,
              suma_neto_plata;

    //Podesavanje izlaznog formata za ispisivanja iznosa

    cout << setprecision(2)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);

    // Ispisivanje poruke

    cout << "Ovaj program racuna neto platu za svakog zaposlenog."
         << endl << endl;
```

```
cout << "Pri svakom pitanju, unesite trazene podatke.";

// Inicijalizacija zbirova

broj_zaposlenih = 0;
suma_ukupno_plate = suma_porez = suma_neto_plata = 0.00;

// Glavna petlja

do  // Pocetak while petlje
{
    // Unos podataka

    cout << endl << endl << "Unesite satnicu : ";
    cin >> satnica;
    cout << endl << "Unesite broj ostvarenih radnih sati : ";
    cin >> radni_sati;
    cout << endl<< "Zaposleni moze biti oslobodjen placanja "
         << "poreza." << endl;
    cout << "Ako je oslobodjen placanja poreza unesite 'O' a ako "
         << "nije unesite 'N' : ";
    cin.get();
    kod_zaposlenog = cin.get();
    cout << endl<< "Zaposleni moze biti domaci ili strani "
         << "drzavljanin." << endl;
    cout << "Ako je domaci drzavljanin unesite 'D' a ako je "
         << "strani drzavljanin unesite 'S' : ";
    cin.get();
    kod_drzave = cin.get();

    // Racunanje ukupne plate

    if (radni_sati > 40.0)
    {
        regularni_rad = 40.0 * satnica;
        prekovremeni_rad = (radni_sati - 40.0) * 1.5 * satnica;
    }
    else
    {
        regularni_rad = radni_sati * satnica;
        prekovremeni_rad = 0.00;
    }

    ukupna_plata = regularni_rad + prekovremeni_rad;

    // Racunanje poreza

    if ((kod_zaposlenog == 'N') || (kod_zaposlenog == 'n'))
        if ((kod_drzave == 'D') || (kod_drzave == 'd'))
            porez = ukupna_plata * KOEFICIJENT_ZA_DOMACE_RADNIKE;
        else
            porez = ukupna_plata * KOEFICIJENT_ZA_STRANE_RADNIKE;
    else
    porez = 0.00;

    // Racunanje neto plate
```

```
        neto_plata = ukupna_plata - porez;

        // Ispisivanje rezultata

        cout << endl << endl;
        cout << "Redovna plata :" << setw(10) << regularni_rad <<
                endl;
        cout << "Prekovremeno : " << setw(10) << prekovremeni_rad <<
                endl;
        cout << "Porez :        " << setw(10) << porez << endl;
        cout << "Ukupno :       " << setw(10) << ukupna_plata << endl;
        cout << "-------------------------" << endl;
        cout << "Neto plata :   " << setw(10) << neto_plata << endl;

        // Povecanje zbirova

        ++broj_zaposlenih;
        suma_ukupno_plate += ukupna_plata;
        suma_porez += porez;
        suma_neto_plata += neto_plata;

        // Odluka o unosu sledeceg zaposlenog

        cout << endl << endl;
        cout << "Zelite li da nastavite sa sledecim zaposlenim ? "
             << "( d/n): ";
        cin.get();
        odgovor = cin.get();
    }
    while ( (odgovor == 'd') || (odgovor == 'D') );    // Uslov while
                                                       // petlje

    // Ispisivanje suma

    cout << endl << endl << endl;
    cout << "Ukupan broj zaposlenih : " << setw(8) <<broj_zaposlenih
         << endl;
    cout << "Zbir ukupnih plata : " << setw(12) << suma_ukupno_plate
         << endl;
    cout << "Zbir poreza :        " << setw(12) << suma_porez << endl;
    cout << "Zbir neto plata :    " << setw(12) << suma_neto_plata <<
            endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Ovaj program racuna neto platu za svakog zaposlenog.

Pri svakom pitanju, unesite trazene podatke.

Unesite satnicu : 7.25

Unesite broj ostvarenih radnih sati : 47

Zaposleni moze biti oslobodjen placanja poreza.
Ako je oslobodjen placanja poreza unesite 'O' a ako nije unesite 'N' :
N

Zaposleni moze biti domaci ili strani drzavljanin.
Ako je domaci drzavljanin unesite 'D' a ako je strani drzavljanin
unesite 'S' : D

Redovna plata :    290.00
Prekovremeno :      76.13
Porez :             25.63
Ukupno :           366.13
-------------------------
Neto plata :       340.50

Zelite li da nastavite sa sledecim zaposlenim ? (d/n):  D

Unesite satnicu : 8.50

Unesite broj ostvarenih radnih sati : 30

Zaposleni moze biti oslobodjen placanja poreza.
Ako je oslobodjen placanja poreza unesite 'O' a ako nije unesite 'N' :
N

Zaposleni moze biti domaci ili strani drzavljanin.
Ako je domaci drzavljanin unesite 'D' a ako je strani drzavljanin
unesite 'S' : S


Redovna plata :    255.00
Prekovremeno :       0.00
Porez :             11.48
Ukupno :           255.00
-------------------------
Neto plata :       243.53

Zelite li da nastavite sa sledecim zaposlenim ? (d/n):  N

Ukupan broj zaposlenih :       2
Zbir ukupnih plata :      621.13
Zbir poreza :              37.10
Zbir neto plata :         584.02
```

U programu se može videti uslov definisan unutar već postojećeg uslova - ugneždeni uslov. Zaključujemo da bi se izvršio blok naredbi definisan unutar unutrašnjeg uslova, neophodno je da oba uslova i spoljašnji i unutrašnji budu ispunjeni.

Primer 18.

Napisati program koji izračunava komisionu cenu u zavisnosti od tipa robe.

```
// Primer 18

// Program odredjuje komisionu cenu
// u zavisnosti od tipa robe.

#include <iostream>
#include <iomanip>
#include <cstdlib>

using namespace std;

int main()
{
    // Deklarisanje promenljivih

    const double    DOMACA_STOPA = 0.060;
    const double    REGIONALNA_STOPA = 0.050;
    const double    KOMERCIJALNA_STOPA = 0.045;

    int     kod_robe;

    double  prodajna_cena,
            komisiona_stopa,
            komisiona_cena;

    //Podesavanje izlaznog formata za ispisivanja iznosa

    cout << setprecision(2)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);

    // Unos podataka

    cout << "Unesite prodajnu cenu proizvoda: ";
    cin >> prodajna_cena;

    cout << endl;
    cout << "Unesite sifru proizoda na sledeci nacin."
         << endl << endl;
    cout << "Domaci proizvod,       unesite D" << endl;
    cout << "Regionani proizvod,    unesite R" << endl;
    cout << "Komercijalni proizvod, unesite C" << endl << endl;
    cout << "Izaberite opciju : ";
```

```
    cin.get();
    kod_robe = cin.get();

    // Izracunavanja

    switch (kod_robe)
    {
        case 'D':
        case 'd':
            komisiona_stopa = DOMACA_STOPA;
        break;

        case 'R':
        case 'r':
            komisiona_stopa = REGIONALNA_STOPA;
        break;

        case 'C':
        case 'c':
            komisiona_stopa = KOMERCIJALNA_STOPA;
        break;

        default:
            cout << endl << endl
            <<"Neispravna sifra proizvoda! Pokusajte ponovo" << endl;
        exit(1);
        break;
    }

    komisiona_cena = prodajna_cena * komisiona_stopa;

    // Ispisivanje rezultata

    cout << endl << endl;
    cout << "Komisiona cena je " << komisiona_cena << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite prodajnu cenu proizvoda: 270000

Unesite sifru proizoda na sledeci nacin.
Domaci proizvod,       unesite D
Regionani proizvod,    unesite R
Komercijalni proizvod, unesite C

Izaberite opciju :  D

Komisiona cena je 16200.00
```

U primeru je korišćeno odlučivanje korišćenjem `switch` naredbe. Iza ključne reči `switch` navodi se veličina koja se posmatra i na osnovu koje se donosi odluka. Dalje sledi `case` ključna reč koja definiše vrednost promenljive, da bi se izvršio blok naredbi koji sledi. Sekvence se ponavljaju onoliko puta koliko uslova želimo da ispitamo.

V Funkcije

Primer 19.

Napisati program koji poziva funkciju definisanu van funkacije `main()` i iscrtava pravougaonik znacima ‘X’ pri čemu koristi funkciju definisanu izvan funkcije `main()`.

```
// Primer 19

// Program ilustruje pozivanje funkcije
// koja je definisana van funkcije 'main'

#include <iostream>

using namespace std;

void Napisi_10_puta_po_X();    // Deklarisanje funkcije

int main()
{
    // Deklarisanje promenljivih

    int i;

    cout << endl;

    // Pozivanje funkcije za ispisivanje kroz petlju

    for (i = 1; i <= 5; ++i)
    {
        Napisi_10_puta_po_X();
        cout << endl;
    }

    cout << endl;

    return 0;
}

void Napisi_10_puta_po_X()    // Definisanje funkcije
{
    // Deklarisanje promenljivih

    int j;

    // Ispisivanje 10 znakova 'X' u jednom redu kroz petlju

    for (j = 1; j <= 10; ++j)
        cout << 'X';

    return;
}
```

Rezultat izvršavanja programa:

```
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
```

Funkcije predstavljaju sekvencu koda definisanog izvan funkcije `main`() koji se izvršava po pozivu funkcije. Funkcija vraća vrednost svojim imenom. Funkcija mora biti deklarisana što znači da se kompajleru mora 'reći' da funkcija postoji, njeno ime, tip kao i ulazni parametri. Deklarisanje funkcije mora biti izvršeno pre njenog korišćenja. Funkcija tipa `void` ne vraća vrednost.
Korišćenje funkcija je veoma uobičajeno i efikasno u C++ programskom jeziku čime se izbegava višestruko ponavljanje identičnog bloka naredbi i smanjuje se broj kodnih linija u samoj `main()` funkciji. Upotrebom funkcija se postiže bolja preglednost samog koda i smanjuje verovatnoća logičke greške.
U ovom primeru, funkcija nema ulazne parametre.

Primer 20.

Napisati program koji poziva funkciju definisanu van funkcije `main()` i crta trougao izabranim znacima.

```
// Primer 20

// Program ilustruje pozivanje funkcije
// koja je definisana van funkcije 'main'

#include <iostream>

using namespace std;

void Napisi_Znak(char, int);    // Deklarisanje funkcije

int main()
{
    // Deklarisanje promenljivih

    int i,
        broj_redova;

    char znak;

    // Unos podataka

    cout << "Unesite znak koji zelite da bude prikazan : ";
```

```
    cin >> znak;

    cout << endl;
    cout << "Unesite broj redova koji zelite da bude prikazan : ";
    cin >> broj_redova;

    // Pozivanje funkcije za ispisivanje kroz petlju

    cout << endl;
    for (i = 1; i <= broj_redova; ++i)
    {
        Napisi_Znak(znak, i);
        cout << endl;
    }

    cout << endl;

    return 0;
}

void Napisi_Znak(char izabrani_znak, int brojac)    // Definisanje
                                                    // funkcije
{
    // Deklarisanje promenljivih

    int j;

    // Ispisivanje izabranog znaka izabrani broj puta u jednom redu
    // kroz petlju

    for (j = 1; j <= brojac; ++j)
        cout << izabrani_znak;

    return;
}
```

Rezultat izvršavanja programa:

```
Unesite znak koji zelite da bude prikazan : Z

Unesite broj redova koji zelite da bude prikazan : 5

Z
ZZ
ZZZ
ZZZZ
ZZZZZ
```

U ovom primeru, funkcija ima ulazne parametre. To su promenljive koje se prosleđuju funkciji na osnovu kojih funkcija vrši zadata izračunavanja ili logičke operacije. Promenljive su o ovom slučaju prosleđene 'po vrednosti' što znači da se

funkciji prosleđuje vrednost neke promenljive. Funkcija može imati više ulaznih parametara.
Nazivi promenljivih kojima se funkcija poziva ne moraju biti isti kao i nazivi odgovarajućih promenljivih koje se koriste u funkciji. Bitan je njihov tip i njihov redosled, kako je to navedeno prilikom deklarisanja funkcije.

Primer 21.

Napisati program koji prosleđuje argument funkcije po vrednosti i analizirati povratnu vrednost funkcije.

```
// Primer 21

// Program ilustruje prosledjivanje argumenat funkcije po vrednosti.

#include <iostream>

using namespace std;

void Funkcija(int);     // Deklarisanje funkcije

int main()
{
    // Deklarisanje promenljivih

    int i;

    // Inicijalizacija promenljive i na 5

    i = 5;

    // Ispisivanje vrednosti promenljive i pre poziva funkcije

    cout << "Pre poziva fukcije, i = " << i << endl;

    // Vrednost i se prosledjuje funkciji Funkcija(),
    // i dodeljuje parametru argument (videti u definiciji funkcije
    // Funkcija()).
    // Funkcija povecava vrednost svoje promenljive argument za 1
    // i vraca izvrsavanje programa nazad u main().
    // Vrednost promenljive i se ne menja pozivom funkcije Funkcija().

    Funkcija(i);

    // Ispisivanje vrednosti promenljive i posle poziva funkcije

    cout << "Posle poziva funkcije, i = " << i << endl;

    cout << endl;

    return 0;
```

```
}

void Funkcija(int argument)    // Definisanje funkcije
{
    // Vrednost prosledjena funkciji se dodeljuje
    // parametru argument. Funkcija povecava vrednost
    // promenljive argument za jedan i vraca kontrolu funkciji main().

    ++argument;

    return;
}
```

Rezultat izvršavanja programa:

```
Pre poziva fukcije, i = 5
Posle poziva funkcije, i = 5
```

Vrednost promenljive `i` koja se prosleđuje funkciji se smešta u promenljivu `argument`, lokalnu promenljivu funkcije. Uočimo da promena lokalne promenljive `argument` ne utiče na vrednost promenljive `i` u `main()` funkciji.

Primer 22.

Napisati program koji izračunava ocenu na ispitu na osnovu broja bodova sa usmenog i pismenog dela ispita. Koristiti funkciju definisanu izvan funkcije `main()`.

```
// Primer 22

// Program izracunava krajnju ocenu na ispitu na osnovu rezultata
// pismenog i usmenog ispita. Koisti funkciju
// Izracunaj_Konacnu_Ocenu() za izracunavanje konacne ocene.
// Funkcija vraca vrednost u main().

#include <iostream>

using namespace std;

int Izracunaj_Konacnan_Broj_Bodova(int, int);    // Deklarisanje
                                                 // funkcije

int main()
{
    // Deklarisanje promenljivih

    int bodovi_sa_pismenog_ispita,
        bodovi_sa_usmenog_ispita,
        konacan_broj_bodova;
```

```
    // Unos podataka

    cout << "Unesite broj bodova sa pismenog ispita : ";
    cin >> bodovi_sa_pismenog_ispita;
    cout << endl;

    cout << "Unesite broj bodova sa usmenog ispita : ";
    cin >> bodovi_sa_usmenog_ispita;

    // izracunavanja pozivom funkcije

   konacan_broj_bodova =
   Izracunaj_Konacnan_Broj_Bodova(bodovi_sa_pismenog_ispita,
   bodovi_sa_usmenog_ispita);

    // Ispisivanje rezultata

    cout << endl;
    cout << "Konacan broj bodova je " << konacan_broj_bodova << endl;

    cout << endl;

    return 0;
}

int Izracunaj_Konacnan_Broj_Bodova(int pismeni, int usmeni)
// Definisanje funkcije
{
    // Deklarisanje promenljivih

    const double UTICAJ_PISMENOG_ISPITA = 0.40;
    const double UTICAJ_USMENOG_ISPITA = 0.60;

    double bodovi;

    int zaokruzeni_bodovi;

    // Izracunavanja

    bodovi = UTICAJ_PISMENOG_ISPITA * pismeni + UTICAJ_USMENOG_ISPITA
             * usmeni;
    zaokruzeni_bodovi = bodovi + 0.5;

    return zaokruzeni_bodovi;
}
```

Rezultat izvršavanja programa:

```
Unesite broj bodova sa pismenog ispita : 77

Unesite broj bodova sa usmenog ispita : 80

Konacan broj bodova je 79
```

Primer predstavlja praktičnu primenu korišćenja funkcije. Uočimo liniju koda `konacan_broj_bodova = Izracunaj_Konacnan_Broj_Bodova(bodovi_sa_pismenog_ispita, bodovi_sa_usmenog_ispita)` kojom se poziva funkcija iz `main()` programa, kao i liniju koda `return zaokruzeni_bodovi` u spoljašnjoj funkciji kojom definišemo povratnu vrednost same funkcije.

Primer 23.

Napisati program koji demonstrira upotrebu lokalnih i globalnih promenljivih.

```
// Primer 23

// Program ilustruje primenu lokalnih i globalnih promenljivih.

#include <iostream>

using namespace std;

// Deklarisanje globalnih promenljivih

int globalna_promenljiva_1,
    globalna_promenljiva_2;

void Funkcija();    // Deklarisanje funkcije

int main()
{
    // Sledeca dodeljivanja vrednosti su legalna jer su promenljive
    // globalna_promenljiva_1 i globalna_promenljiva_2
    // globalne promenljive deklarisane pre funkcije main().

    globalna_promenljiva_1 = 88;
    globalna_promenljiva_2 = 99;

    // Ispisivanje vrednosti globalnih promenljivih
    // globalna_promenljiva_1 i globalna_promenljiva_2.

    cout << "Inicijalne vrednosti globalnih promenljivih :" << endl;
    cout << "globalna promenljiva 1 = " << globalna_promenljiva_1
         << " globalna promenljiva 2 = " << globalna_promenljiva_2
         << endl << endl;

    // Poziv funkcije Funkcija()

    Funkcija();

    // Ispisivanje vrednosti globalnih promenljivih
    // globalna_promenljiva_1 i globalna_promenljiva_2
    // nakon poziva funkcije Funkcija().

    cout << endl;
```

```
    cout << "Nakon poziva funkcije Funkcija(), globalne promenljive "
         << "su :"
         << endl;
    cout << "globalna promenljiva 1 = " << globalna_promenljiva_1
         << " globalna promenljiva 2 = " << globalna_promenljiva_2
         << endl << endl;

    return 0;
}

// Deklarisanje globalne promenljive

int globalna_promenljiva_3;

void Funkcija()    // Definisanje funkcije
{
    //Deklarisanje lokalnih promenljivih

    int globalna_promenljiva_1,    // Lokalna promenljiva za funkciju
                                   // Funkcija().
    lokalna_promenljiva_1;         // Lokalna promenljiva
                                   // globalna_promenljiva_1 je
    // razlicita od globalne promenljive globalna_promenljiva_1 koja
    // je deklarisana pre funkcije main().

    globalna_promenljiva_3 = 1;    // Dodeljivanje vrednosti globalnoj
                                   // promenljivoj 3

    lokalna_promenljiva_1 = 2;     // Inicijalizacija lokalne
                                   // promenljive lokalna_promenljiva_1

    globalna_promenljiva_1 = 3;    // Ne menja vrednost promenljive
                                   // globalna_promenljiva_1
                                   // u main() funkciji.

    globalna_promenljiva_2 = 4;    // Posto globalna_promenljiva_2
                                   // nije deklarisana u ovoj
    // funkciji, radi se o globalnoj promenljivoj deklarisanoj pre
    // main() funkcije. Zbog toga, ova dodela vrednosti menja vrednost
    // promenljive globalna_promenljiva_2 u main() funkciji.

    // Ispisivanje vrednosti lokalnih promenljivih
    // globalna_promenljiva_1 i lokalna_promenljiva_1
    // i globalnih promenljivih globalna_promenljiva_2 i
    // globalna_promenljiva_3.

    cout;
    cout << "U funkciji Funkcija(), globalne promenljive su :"
         << endl;
    cout << "globalna promenljiva 2 = " << globalna_promenljiva_2
         << " globalna promenljiva 3 = " << globalna_promenljiva_3
         << endl << endl;

    cout << "U funkciji Funkcija(), lokalne promenljive su :" << endl;
    cout << "globalna promenljiva 1 = " << globalna_promenljiva_1
         << " lokalna promenljiva 1 = " << lokalna_promenljiva_1
```

```
          << endl;
}
```

Rezultat izvršavanja programa:

```
Inicijalne vrednosti globalnih promenljivih :
globalna promenljiva 1 = = 88 globalna promenljiva 2 = 99

U funkciji Funkcija(), globalne promenljive su :
globalna promenljiva 2 = 4 globalna promenljiva 3 = 1

U funkciji Funkcija(), lokalne promenljive su :
globalna promenljiva 1 = 3 lokalna promenljiva 1 = 2

Nakon poziva funkcije Funkcija(), globalne promenljive su :
globalna promenljiva 1 = 88 globalna promenljiva 2 = 4
```

Promenljive deklarisane izvan funkcije su globalne promenljive i mogu se pozivati i koristiti u svakoj funkciji, uključujući i `main()`. Promenljive deklarisane unutar neke funkcije su lokalne promenljive i mogu se pozivati i koristiti samo unutar te funkcije. Ukoliko se unutar neke funkcije deklariše lokalna promenljiva sa imenom koje već ima neka globalna promenljiva radi se o lokalnoj promenljivoj za tu funkciju, bez ikakvog uticaja na globalnu promenljivu.

Primer 24.

Napisati program koji broji iteracije u petlji primenom static promenljive.

```
// Primer 24

#include <iostream>

using namespace std;

void Brojac_Iteracija();    // Deklarisanje funkcije

int main()
{
    // Deklarisanje promenljivih

    int i;

    // for petlja koja poziva funkciju Brojac_Iteracija()
    // definisani broj puta

    for (i = 1; i <= 5; ++i)
        Brojac_Iteracija();
```

```
    cout << endl;

    return 0;
}

void Brojac_Iteracija()    // Definicija funkcije
{
    // Deklarisanje promenljivih

    static int brojac = 0;    // static oznacava da je memorijska
    // lokacija u koju se smesta promenljiva brojac nepromenljiva

    ++brojac;

    // Ispisivanje poruke

    cout << "Iteracija broj " << brojac << endl;

    return;
}
```

Rezultat izvršavanja programa:

```
Iteracija broj 1
Iteracija broj 2
Iteracija broj 3
Iteracija broj 4
Iteracija broj 5
```

Trajanje promenljive je vreme za koje program alocira memoriju promenljivoj.
automatic variable je lokalna varijabla, koja po default-u ima storage class auto, automatski pozivom funkcije dodeljuje joj se memorija .
static variable se dodeljuje memorija kada program počne. Ostaje da važi sve vreme dok se program izvršava, nezavisno koja je funkcija aktivna. Globalna variabla ima static storage class po default-u.

Primer 25.

Napisati program koji izračunava cenu selidbe nameštaja u zavisnosti od težine i udaljenosti na koju se prevozi. U programu korisiti funkcije, kao i globalne i lokalne promenljive.

```
// Primer 25

// Program izracunava cenu selidbe namestaja.
// Koristi funkcije za izracunavanja i ispisivanje rezultata.

#include <iostream>
```

```
#include <iomanip>

using namespace std;

double Izracunavanje_Cene_Rada(int);
double Izracunavanje_Cene_Prevoza(int);
void Ispisivanje_Troskova(double, double, double);

int main()
{
    // Deklarisanje promenljivih

    int    tezina,         // Tezina namestaja
           rastojanje;     // Rastojanje za prevoz

    double troskovi_rada,      // Troskovi prenosenja namestaja
           troskovi_prevoza,   // Troskovi prevoza
           ukupni_troskovi;    // Ukupni troskovi

    //Podesavanje izlaznog formata za ispisivanja iznosa

    cout << setprecision(2)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);

    // Unos podataka

    cout << "Unesite tezinu u kilogramima : ";
    cin >> tezina;

    cout << endl;
    cout << "Unesite rastojanje u kilometrima : ";
    cin >> rastojanje;

    // Izracunavanja pozivom funkcija

    troskovi_rada = Izracunavanje_Cene_Rada(tezina);
    troskovi_prevoza = Izracunavanje_Cene_Prevoza(rastojanje);
    ukupni_troskovi = troskovi_rada + troskovi_prevoza;

    // Ispisivanje rezultata pozivom funkcije

    Ispisivanje_Troskova(troskovi_rada, troskovi_prevoza,
                         ukupni_troskovi);

    cout << endl;

    return 0;

} // Kraj funkcije main()

double Izracunavanje_Cene_Rada(int tezina)
{
    // Deklarisanje promenljivih

    const double KOEFICIJENT_RADA = 4.00;
```

```
    double troskovi_rada;

    // Izracunavanja

    troskovi_rada = (tezina / 100) * KOEFICIJENT_RADA;

    return troskovi_rada;

}  // Kraj funkcije Izracunavanje_Cene_Rada()

double Izracunavanje_Cene_Prevoza(int rastojanje)
{
    // Deklarisanje promenljivih

    const double KOEFICIJENT_PREVOZA     = 1.75;
    const double FIKSNI_TROSKOVI_PREVOZA = 50.00;

    double troskovi_prevoza;

    // Izracunavanja

    troskovi_prevoza = FIKSNI_TROSKOVI_PREVOZA + rastojanje *
                       KOEFICIJENT_PREVOZA;

    return troskovi_prevoza;

}  // Kraj funkcije Izracunavanje_Cene_Prevoza()

void Ispisivanje_Troskova(double cena_rada, double cena_prevoza,
                          double ukupni_troskovi)
{
    // Ispisivanje rezultata

    cout << endl;
    cout << "Cena selidbe je :" << endl << endl;
    cout << "Prenosenje      : " << setw(9) << cena_rada << endl;
    cout << "Prevoz          : " << setw(9) << cena_prevoza << endl;
    cout << "--------------------------" << endl;
    cout << "Ukupno          : " << setw(9) << ukupni_troskovi
         << endl;
}  // Kraj funkcije Ispisivanje_Troskova()
```

Rezultat izvršavanja programa:

```
Unesite tezinu u kilogramima : 7400

Unesite rastojanje u kilometrima : 230

Cena selidbe je :

Prenosenje      :    296.00
Prevoz          :    452.50
-------------------------
Ukupno          :    748.50
```

Globalna varijabla važi sve vreme dok se program izvršava, nezavisno koja je funkcija aktivna. Globalna varijabla ima static storage class po default-u.

Primer 26.

Napisati program koji izračunava mesečnu ratu prilikom otplate kredita upotrebom funkcija.

```
// Primer 26

// Program za izracunavanje kamate

#include <iostream>
#include <iomanip>
#include <cstdlib>

using namespace std;

void   Ispisivanje_Pozdravne_Poruke();
double Unos_Visine_Kredita();
double Unos_Kamatne_Stope();
int    Unos_Roka_Otplate();

int main()
{
    double kamata,
           kredit,
           kamatna_stopa,
           ukupno,
           mesecna_rata;

    int    rok_otplate;

    cout << setprecision(2)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);
```

```
    // Ispisivanje pozdravne poruke

    Ispisivanje_Pozdravne_Poruke();

    // Unos podataka

    kredit        = Unos_Visine_Kredita();
    kamatna_stopa = Unos_Kamatne_Stope();
    rok_otplate   = Unos_Roka_Otplate();

    // Izracunavanja

    kamata          = kredit * (kamatna_stopa/100) * rok_otplate;
    ukupno          = kredit + kamata;
    mesecna_rata    = ukupno / (12 * rok_otplate);

    // Ispisivanje rezultata

    cout << endl;
    cout << "Kamatna stopa :       "  << setw(5) << kamatna_stopa
         << "%" << endl;
    cout << "Rok otplate :         "  << rok_otplate << " godina"
         << endl << endl;
    cout << "Visina kredita :      EUR" << setw(9) << kredit << endl;
    cout << "Kamata :              EUR" << setw(9) << kamata << endl;
    cout << "-----------------------------------" << endl;
    cout << "Iznos koji se vraca : EUR" << setw(9) << ukupno << endl;
    cout << "Mesecna rata :        EUR" << setw(9) << mesecna_rata
         << endl;

    cout << endl;

    return 0;
} // kraj funkcije main()

void Ispisivanje_Pozdravne_Poruke()
{
    int broj_znakova;

    cout << endl << endl;
    for (broj_znakova = 1; broj_znakova <= 70; ++broj_znakova)
        cout << "*";

    cout << endl << endl;
    cout << "\t\t     Progam za izracunavanje kamate" << endl << endl;
    cout << " Ovaj program izracunava kamatu, ukupan iznos i ";
    cout << "mesecnu ratu kredita" << endl << endl;
    cout << "\t\t       Pazljivo sledite uputstva" << endl << endl;

    for (broj_znakova = 1; broj_znakova <= 70; ++broj_znakova)
        cout << "*";

    cout << endl << endl << endl;
} // Kraj funkcije Ispisivanje_Pozdravne_Poruke()
```

```
double Unos_Visine_Kredita()
{
    const double MIN_KREDIT = 1000.00;
    const double MAX_KREDIT = 20000.00;

    double kredit;

    cout << "Unesite zeljeni iznos kredita : EUR ";
    cin >> kredit;

    if (kredit < MIN_KREDIT)
    {
        cout << endl;
        cout << "Zao nam je, ne odobravamo kredite manje od 1,000.00 "
             << "EUR" << endl;
        exit(1);
    }
    else
    if (kredit > MAX_KREDIT)
    {
        cout << endl;
        cout << "Zao nam je, ne odobravamo kredite";
        cout << " vece od 20,000.00 EUR" << endl << endl;
        cout << "Pokusajte sa drugom vrstom kredita\n";
        exit(1);
    }
    else

        return kredit;
} // Kraj funkcije Unos_Visine_Kredita()

double Unos_Kamatne_Stope()
{
    const double MAX_KAMATNA_STOPA = 18.70;

    double kamatna_stopa;

    cout << endl;
    cout << "Unesite godisnju kamatnu stopu (u %) : ";

    cin >> kamatna_stopa;

    if (kamatna_stopa > MAX_KAMATNA_STOPA)
    {
        cout << endl;
        cout << "Zao nam je, kamatna stopa prevazilazi zakonski "
             << "maksimum od " << MAX_KAMATNA_STOPA << "%" << endl;
        exit(1);
    }
    else

        return kamatna_stopa;
} // Kraj funkcije Unos_Kamatne_Stope()

int Unos_Roka_Otplate()
{
```

```
    const int MAX_ROK = 5;

    int rok;

    cout << endl;
    cout <<"Unesite broj godina otplate : ";
    cin >> rok;

    if (rok > MAX_ROK)
    {
        cout << endl;
        cout << "Zao nam je, rok otplate je veci od"
             << MAX_ROK << "godina" << endl;
        exit(1);
    }
    else

    return rok;
}
```

Rezultat izvršavanja programa:

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

                    Progam za izracunavanje kamate

 Ovaj program izracunava kamatu, ukupan iznos i mesecnu ratu kredita

                      Pazljivo sledite uputstva

**********************************************************************


Unesite zeljeni iznos kredita : EUR 10000.00

Unesite godisnju kamatnu stopu (u %) : 10.7

Unesite broj godina otplate : 4

Kamatna stopa :       10.70%
Rok otplate :         4 godina

Visina kredita :      EUR 10000.00
Kamata :              EUR  4280.00
-----------------------------------
Iznos koji se vraca : EUR 14280.00
Mesecna rata :        EUR   297.50
```

Program ilustruje upotrebu lokalnih i globalnih varijabli, kao i upotrebu funkcija.

Primer 27.

Napisati program koji stepenuje uneti broj željenim eksponentom.

```
// Primer 27

// Program ilustruje naredbu za stepenovanje pow()

#include <iostream>
#include <cmath>

using namespace std;

int main()
{
    double osnova;
    double eksponent;

    cout << "Unesite osnovu : ";
    cin >> osnova;

    cout << "Unesite eksponent : ";
    cin >> eksponent;

    cout << endl << osnova << " na stepen " << eksponent
    << " je " << pow(osnova, eksponent) << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite osnovu : 4.2
Unesite eksponent : 6.49

4.2 na stepen 6.49 je 11088.9
```

Program ilustruje upotrebu matematičke funkcije za stepenovanje. Uočimo da je korišćena biblioteka `cmath` u kojoj su definisane matematičke funkcije.

Primer 28.

Napisati program koji pronalazi kvadratni koren unetog broja.

```
// Primer 28

// Program prikazuje pronalazi kvadratni koren broja - sqrt()
```

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

using namespace std;

int main()
{
    double broj;

    cout << "Unesite broj ciji kvadratni koren zelite da nadjete : ";
    cin >> broj;
    cout << endl << "Kvadratni koren broja " << broj
         << " je " << sqrt(broj) << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite broj ciji kvadratni koren zelite da nadjete : 873.47

Kvadratni koren broja 873.47 je 29.5545
```

Program ilustruje upotrebu matematičke funkcije za izračunavanje kvadratnog korena.

VI Nizovi

Primer 29.

Napisati program koji računa iznos koji se plaća prilikom preuzimanja poštanske pošiljke. Koristiti jednodimenzioni niz.

```
// Primer 29

// Program racuna iznos koji je potrebno platiti prilikom prijema
posiljke.
// U cenu ulazi cena posiljke kao i postarina, koja zavisi od regiona
// u kome se isporucuje posiljka.
// U programu se demonstrira upotreba promenljive tipa niz - array.

#include <iostream>
#include <iomanip>

using namespace std;

int Unesi_Korektan_Region();

int main()
{
    // Sledeca deklaracija definise niz celih brojeva stopa[]
    // koji ima 5 elemenata.

    double stopa[5] = {0.075, 0.080, 0.082, 0.085, 0.088};

    int    region;

    double cena_posiljke,
           postarina,
           ukupno_za_naplatu;

    cout << setprecision(2)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);

    cout << "Unesite cenu posiljke : EUR ";
    cin >> cena_posiljke;

    region = Unesi_Korektan_Region();

    postarina = cena_posiljke * stopa[region - 1];
    ukupno_za_naplatu = cena_posiljke + postarina;

    cout << endl << endl;
    cout << "Cena posiljke :     " << setw(9) << cena_posiljke
         << endl;
    cout << "Postarina :         " << setw(9) << postarina << endl;
```

```
cout << "Ukupno za naplatu : " << setw(9) << ukupno_za_naplatu
     << endl << endl;

    return 0;
} //Kraj funkcije main()

int Unesi_Korektan_Region()
{
    bool pogresan_unos;

    int region;

    do
    {
        cout << endl;
        cout << "Unesite region u koji se posiljka salje (1-5) : ";
        cin >> region;
        if (region < 1 || region > 5)
        {
            cerr << endl;
            cerr << "Pogresan unos regiona - Pokusajte ponovo.";
            pogresan_unos = true;
        }
        else
            pogresan_unos = false;
    }
    while(pogresan_unos);

    return region;
}
```

Rezultat izvršavanja programa:

```
Unesite cenu posiljke : EUR 100

Unesite region u koji se posiljka salje (1-5) : 4

Cena posiljke :        100.00
Postarina :              8.50
Ukupno za naplatu :    108.50
```

Niz je kolekcija fiksnog broja objekata istog tipa, koji su uskladišteni sekvencijalno u računarsku memoriju. Objekti u jednom nizu su elementi niza.
Na ovaj način se štedi memorijski prostor, broj promenljivih, a takođe se postiže i brži rad programa. Vrednost promenljive na koju se trenutno referiše je definisana indeksom koji se navodi uz ime promenljive. Početni indeks je 0.

Primer 30.

Napisati program koji pronalazi prvi element u zadatom nizu koji zadovoljava postavljeni uslov.

```
// Primer 30

// Program pokazuje kako se nalazi prvi element u nizu
// koji zadovoljava postavljeni uslov. Uslov koji je postavljen
// u ovom primeru je da je broj bodova na testu veci ili jednak 85.

#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
    const int BROJ_TESTOVA = 10;
    const int MIN_BROJ_BODOVA = 85;

    int broj_bodova[BROJ_TESTOVA];     // Niz koji cuva broj bodova na
                                       // svakom od testova
    int redni_broj_testa;              // Koeficijent niza -
                                       // test po redu

    cout << "Unesite broj bodova za svaki od " << BROJ_TESTOVA
         << " testova." << endl << endl;

    for (redni_broj_testa = 0; redni_broj_testa < BROJ_TESTOVA;
         ++redni_broj_testa)
    {
        cout << endl;
        cout << "Unesite broj bodova za test broj "
             << redni_broj_testa + 1 << " : ";
        cin >> broj_bodova[redni_broj_testa];
    }

    for (redni_broj_testa = 0; redni_broj_testa < BROJ_TESTOVA;
         ++redni_broj_testa)
        if (broj_bodova[redni_broj_testa] >= MIN_BROJ_BODOVA)
        {
            cout << endl;
            cout << "Prvo pojavljivanje od najmanje "
                 << MIN_BROJ_BODOVA
                 << " bodova je :" << endl << endl;
            cout << "Test broj " << redni_broj_testa + 1 << " sa "
                 << setw(3) << broj_bodova[redni_broj_testa]
                 << " bodova." << endl << endl;
            break;
        }
        if (redni_broj_testa >= BROJ_TESTOVA)
        {
            cout << endl;
            cout << "Ne postoji test sa vise od " << MIN_BROJ_BODOVA
```

```
                    << " bodova." << endl << endl;
        }

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite broj bodova za svaki od 10 testova.

Unesite broj bodova za test broj 1 : 40

Unesite broj bodova za test broj 2 : 50

Unesite broj bodova za test broj 3 : 80

Unesite broj bodova za test broj 4 : 70

Unesite broj bodova za test broj 5 : 80

Unesite broj bodova za test broj 6 : 94

Unesite broj bodova za test broj 7 : 85

Unesite broj bodova za test broj 8 : 68

Unesite broj bodova za test broj 9 : 73

Unesite broj bodova za test broj 10 : 90

Prvo pojavljivanje od najmanje 85 bodova je :

Test broj 6 sa  94 bodova.
```

U primeru je prikazan način pretraživanja niza korišćenjem `for` petlje. Očigledno je koliko je prikazani postupak jednostavniji od identičnog postupka ukoliko se ne bi koristio niz za smeštanje veličina.

Primer 31.

Napisati program koji sortira niz primenom bubble sort algoritma.

```
// Primer 31

// Program demonstrira sortiranje elemenata niza
// koriscenjem bubble sort algoritma.

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

```
using namespace std;

int main()
{
    const int BROJ_TESTOVA = 10;

    int broj_bodova[BROJ_TESTOVA];    // Niz koji cuva broj bodova na
                                      // svakom od testova

    int redni_broj_testa,    // Koeficijent niza - test po redu
        pomocna,             // Pomocna promenljiva koja sluzi za
                             // zamenu vrednosti elemenata niza
        prolaz_po_redu,      // Redni broj izvrsavanja sekvence
                             // u petlji za sortiranje
        max_broj_uporedjivanja;    // Najveci broj iteracija

    // Unos brojeva niza

    cout << "Unesite broj bodova za svaki od " << BROJ_TESTOVA
         << " testova." << endl << endl;

    for (redni_broj_testa = 0; redni_broj_testa < BROJ_TESTOVA;
         ++redni_broj_testa)
    {
        cout << endl;
        cout << "Unesite broj bodova za test broj "
             << redni_broj_testa + 1 << ": ";
        cin >> broj_bodova[redni_broj_testa];
    }

    // Ispisivanje niza u unetom poretku

    cout << endl << endl;
    cout << "Uneli ste broj bodova sledecim redom :" << endl;

    for (redni_broj_testa = 0; redni_broj_testa < BROJ_TESTOVA;
         ++redni_broj_testa)
        cout << setw(6) << broj_bodova[redni_broj_testa];

    cout << endl;

    // Sortiranje niza bubble sort algoritmom

    max_broj_uporedjivanja = BROJ_TESTOVA - 2;

    for (prolaz_po_redu = 1; prolaz_po_redu <= BROJ_TESTOVA - 1;
         ++prolaz_po_redu)
    {
        for (redni_broj_testa = 0; redni_broj_testa <=
             max_broj_uporedjivanja;
             ++redni_broj_testa)
        if (broj_bodova[redni_broj_testa] >
            broj_bodova[redni_broj_testa + 1])
        {
            pomocna = broj_bodova[redni_broj_testa];
```

```
            broj_bodova[redni_broj_testa] =
            broj_bodova[redni_broj_testa + 1];
            broj_bodova[redni_broj_testa + 1] = pomocna;
        }
        max_broj_uporedjivanja;
    }

    // Ispisivanje sortiranih brojeva

    cout << endl << endl;
    cout << "Broj bodova u rastucem rasporedu je :" << endl;

    for (redni_broj_testa = 0; redni_broj_testa < BROJ_TESTOVA;
         ++redni_broj_testa)
        cout << setw(6) << broj_bodova[redni_broj_testa];

    cout << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite broj bodova za svaki od 10 testova.

Unesite broj bodova za test broj 1: 40

Unesite broj bodova za test broj 2: 50

Unesite broj bodova za test broj 3: 80

Unesite broj bodova za test broj 4: 70

Unesite broj bodova za test broj 5: 80

Unesite broj bodova za test broj 6: 94

Unesite broj bodova za test broj 7: 85

Unesite broj bodova za test broj 8: 68

Unesite broj bodova za test broj 9: 73

Unesite broj bodova za test broj 10: 90

Uneli ste broj bodova sledecim redom :
    40    50    80    70    80    94    85    68    73    90

Broj bodova u rastucem rasporedu je :
    40    50    68    70    73    80    80    85    90    94
```

Bubble sort algoritam se pokazao kao najbrži metod sortiranja niza. Potrebno je obratiti pažnju na međusobnu zamenu vrednosti promenljivih. Uočimo da je neophodno uvođenje pomoćne promenljive da bi se navedena operacija izvela.

Primer 32.

Napisati program koji računa prosečan broj bodova za svakog studenta korišćenjem dvodimenzionog niza.

```
// Primer 32

// Program koristi dvodimenzioni niz za smestanje podataka
// o broju bodova studenta na ispitima. Nakon toga
// racuna prosecan broj bodova za svakog studenta.

#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
    const int BROJ_TESTOVA = 10;
    const int BROJ_STUDENATA = 5;

    int broj_bodova[BROJ_STUDENATA][BROJ_TESTOVA];

    int redni_broj_studenta,
        redni_broj_testa,
        ukupan_broj_bodova;

    double prosecan_broj_bodova;

    cout << setprecision(1)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);

    // Unos broja bodova za svaki test i za svakog studenta

    cout << "Unesite ostvarene bodove za " << BROJ_TESTOVA
         << " testova za svakog studenta." << endl;
    cout << "Razdvojte bodove jednim ili vise blankova." << endl;

    for (redni_broj_studenta = 0; redni_broj_studenta <
         BROJ_STUDENATA;
         ++redni_broj_studenta)
    {
        cout << endl;
        cout << "Broj bodova za studenta" << redni_broj_studenta + 1
             << ": ";
        for (redni_broj_testa = 0; redni_broj_testa < BROJ_TESTOVA;
             ++redni_broj_testa)
```

```
            cin >> broj_bodova[redni_broj_studenta][redni_broj_testa];
        }

        // Izracunavanje i ispisivanje prosecnog broja bodova
        // za svakog studenta

        for (redni_broj_studenta = 0; redni_broj_studenta <
             BROJ_STUDENATA;
             ++redni_broj_studenta)
        {
            ukupan_broj_bodova = 0;

            for (redni_broj_testa = 0; redni_broj_testa < BROJ_TESTOVA;
                 ++redni_broj_testa)
                ukupan_broj_bodova +=
                broj_bodova[redni_broj_studenta][redni_broj_testa];

            prosecan_broj_bodova = (double) ukupan_broj_bodova /
                                   BROJ_TESTOVA;

            cout << endl << endl;
            cout << "Student : " << setw(3) << redni_broj_studenta + 1
                 << "   Prosecan broj bodova : " << setw(5)
                 << prosecan_broj_bodova;
        }

        cout << endl << endl;

        return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite ostvarene bodove za 10 testova za svakog studenta.
Razdvojte bodove jednim ili vise blankova.

Broj bodova za studenta 1 : 50 56 87 67 98 90 68 54 67 30

Broj bodova za studenta 2 : 70 68 64 78 97 57 68 90 67 74

Broj bodova za studenta 3 : 64 76 87 67 95 67 56 83 60 78

Broj bodova za studenta 4 : 76 65 84 47 86 65 46 66 87 65

Broj bodova za studenta 5 : 76 57 65 45 90 76 76 44 67 82

Student:   1   Prosecan broj bodova :  66.7

Student:   2   Prosecan broj bodova :  73.3

Student:   3   Prosecan broj bodova :  73.3

Student:   4   Prosecan broj bodova :  68.7

Student:   5   Prosecan broj bodova :  67.8
```

Dvodimenzioni nizovi se veoma često koriste u rešavanju raznovrsnih problema.

VII Pokazivači i c-stringovi

Primer 33.

Napisati program koji dodeljuje vrednost promenljivoj primenom pokazivača.

```
// Primer 33

// Program ilustruje upotrebu pokazivaca - pointera
// i indirektnog operatora.

#include <iostream>

using namespace std;

int main()
{
  int   i;
  int* pokazivac_na_i = &i;

  cout << "Memorjska lokacija u koju je smestena promenljiva i je "
       << &i << endl;
  cout << "Vrednost promenljive pokazivac_na_i je "
       << pokazivac_na_i << endl << endl;

  // Dodeljivanje vrednosti koriscenjem indirekcije

  *pokazivac_na_i = 17;

  cout << "Vrednost i je " << i << endl;
  cout << "Vrednost  *pokazivac_na_i je " << *pokazivac_na_i
       << endl << endl;

  return 0;
}
```

Rezultat izvršavanja programa:

```
Memorjska lokacija u koju je smestena promenljiva i je 0012FF7C
Vrednost promenljive pokazivac_na_i je 0012FF7C

Vrednost i je 17
Vrednost  *pokazivac_na_i je 17
```

Pokazivač je promenljiva čija je vrednost adresa druge promenljive.
Operator indirekcije * je unarni operator. Sledi, ima isti prioritet i (sa desna-u-levo) asocijativnost, kao i drugi unarni operatori.
Čita se kao "the target of." Cilj pokazivača je promenljiva na koju on pokazuje.

Primer 34.

Napisati program koji vrši operacije sa pokazivačima na nizove, kao i samim nizovima.

```
// Primer 34

// Program demonstrira aritmeticke operacije sa pokazivacima
// i redovima.

#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
    int  niz_brojeva[5] = {3, 4, 5, 6, 7};

    int* pokazivac_na_niz_brojeva;

    int  i;

    cout << "Ispisivanje niza koriscenjem sintakse: niz_brojeva[i]"
         << endl;

    for (i = 0; i < 5; ++i)
        cout << setw(4) << niz_brojeva[i];

    cout << endl << endl;

    cout << "Ispisivanje niza koriscenjem sintakse: "
         << "(niz_brojeva + i)" << endl;

    for (i = 0; i < 5 ; ++i)
        cout << setw(4) << *(niz_brojeva + i);

    cout << endl << endl;

    cout << "Ispisivanje niza koriscenjem sintakse: "
     << "pokazivac_na_niz_brojeva" << endl;

     for (pokazivac_na_niz_brojeva = niz_brojeva;
          pokazivac_na_niz_brojeva < niz_brojeva + 5;
          ++pokazivac_na_niz_brojeva)
          cout << setw(4) << *pokazivac_na_niz_brojeva;

    cout << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Ispisivanje niza koriscenjem sintakse: niz_brojeva[i]
   3   4   5   6   7

Ispisivanje niza koriscenjem sintakse: *(niz_brojeva + i)
   3   4   5   6   7

Ispisivanje niza koriscenjem sintakse: pokazivac_na_niz_brojeva
   3   4   5   6   7
```

Pokazivač na niz pokazuje na prvi element niza. Da bi pokazivač pokazao na sledeći element niza potrebno ga je uvećati za 1.

Primer 35.

Napisati program koji za unos niza znakova koristeći naredbu cin.getline().

```
// Primer 35

// program demonstrira koriscenje naredbe cin.getline()
// za unos stringa

#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
  double const STOPA_POREZA = 0.0825;

  char naziv_stavke[51];

  double cena,
         porez,
         ukupno;

  cout << setprecision(2)
       << setiosflags(ios::fixed)
       << setiosflags(ios::showpoint);

  cout << "Unesite naziv stavke : ";
  cin.getline(naziv_stavke, 51);
  cout << endl;
  cout << "Unesite cenu stavke : ";
  cin >> cena;

  porez = cena * STOPA_POREZA;
  ukupno = cena + porez;
```

```
  cout << endl;
  cout << "Stavka :  " << naziv_stavke << endl << endl;;
  cout << "Cena :    " << setw(9) << cena << endl;
  cout << "Porez :   " << setw(9) << porez << endl;
  cout << "-------------------" << endl;
  cout << "Ukupno :  " << setw(9) << ukupno << endl << endl;

  return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite naziv stavke : Procesor

Unesite cenu stavke : 1459.95

Stavka :  Procesor

Cena :      1459.95
Porez :      120.45
-------------------
Ukupno :    1580.40
```

Klasa `string` je definisana u okviru VS6. Nije potrebno ulaziti u detalje implementacije klase, već je potrebno ovladati korišćenjem funkcija date klase. Možemo uočiti da je objakat klase string definisan kao niz pojedinačnih znakova. Kao u svakom nizu, moguće je definisati svaki pojedinačni element niza.

Napomena:

Ispravljanje greške u Microsoft Visual C++ 6.0
Ukoliko kompajlirate source kod korišćenjem Microsoft Visual C++, version 6.0, i koristite funkciju getline za učitavanje vrednosti objekta klase string, verovatno ćete primetiti da korisnik mora pritisnuti Enter dva puta kako bi nastavio izvršavanje programa. Ovo je kompajlerska greška.
Da bi se ispravila getline greška u Microsoft Visual C++, version 6.0, pažljivo pratite sledeće korake:

1. *Kliknite desnim klikom na reč string u #include <string> vašeg hedera*
2. *Izaberite open document string iz padajućeg menija*
3. *Kliknite na Edit, a zatim Find*
4. *Unesite sledeće kao tekst za pretraživanje:* `(_Tr::eq((_E)_C, _D))`
5. *Uradite izmene kako je prikazano u sledećoj tabeli:*

Izvorno	*Modifikovano*

`else if (_Tr::eq((_E)_C, _D))`
`{_Chg = true;`
`_I.rdbuf()->snextc();`
`break; }`
	`else if (_Tr::eq((_E)_C, _D))`
	`{_Chg = true;`
	`// _I.rdbuf()->snextc();`
	`// (ovaj komentar izbacuje neispravnu naredbu)`
	`_I.rdbuf()->sbumpc(); // ispravan kod`
	`break; }`

6. *Save i close string header file.*

Kraj napomene.

Primer 36.

Napisati program koji broji znakove unete u jednom redu korišćenjem klase C-sring i pokazivača.

```
// Primer 36

// Progam broji znakove koji se unose u jednoj liniji
// Koristi pokazivac za kretanje kroz string kao i while petlju za
prebrojavanje znakova

#include <iostream>

using namespace std;

int main()
{
  char linija[81];
  char* pokazivac_na_znak = linija;
  int  brojac = 0;

  cout << "Unesite niz znakova :" << endl << endl;
  cin.getline(linija, 81);

  while ( *pokazivac_na_znak != '\0' )
  {
      ++brojac;
      ++pokazivac_na_znak;
  }
  cout << endl;
  cout << "Niz koji ste uneli sadrzi " << brojac << " znakova."
       << endl << endl;

  return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite niz znakova :

Vec smo dosta naucili o C++ programskom jeziku.

Niz koji ste uneli sadrzi 47 znakova.
```

Određeni specijalni karakteri se nazivaju escape sekvencama. Escape sekvence počinju sa \ , obrnuta kosa crta ili backslash, iza koje sledi alfanumerički karakter. Na primer \n escape sekvenca predstavlja novi red, dok \0 predstavlja null karakter kojim se završava string.

Primer 37.

Napisati program koji zahteva unos znakova u jednom redu korišćenjem klase C-string i pokazivača. Program nakon toga ispisuje uneti niz u obrnutom redosledu od redosleda unosa.

```
// Primer 37

// Ovaj program zahteva unos stringa
// a zatim ga ispisuje i obrnutom redosledu.

#include <iostream>

using namespace std;

int main()
{
    char linija[81];
    char* pokazivac_na_znak = linija;

    cout << "Unesite niz znakova :" << endl << endl;
    cin.getline(linija, 81);

    // Pronalazenje kraja stringa

    while ( *pokazivac_na_znak != '\0' )
        ++ pokazivac_na_znak;

    // ch_ptr sada pokazuje na null
    -- pokazivac_na_znak;

    // pokazivac_na_znak sada pokazuje na poslednji znak u stringu

    cout << endl;
    cout << "Niz u obrnutom redosledu je :" << endl << endl;
```

```
    // while petlja ispisuje sve znakove osim prvog

    while (pokazivac_na_znak != linija )
    {
        cout << *pokazivac_na_znak;
        --pokazivac_na_znak;
    }

    // Ispisivanje prvog znaka

    cout << *pokazivac_na_znak;

    cout << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite niz znakova :

ana voli milovana

Niz u obrnutom redosledu je :

anavolim ilov ana
```

Uočavamo da izvorni niz ostaje neizmenjen, dok se prilikom ispisa koristi pokazivač koji počinje od poslednjeg znaka, a brojač broji znakove umanjujući svoju vrednost za 1 u svakoj iteraciji.

VIII Pokazivači, nizovi i funkcije

Primer 38.

Napisati program koji prosleđuje pokazivač kao argument funkcije.

```
// Primer 38

// Program demonstrira prosledjivanje argumenta po adresi.

#include <iostream>

using namespace std;

void Funkcija_Po_Adresi(int*);

int main()
{
    int i = 4;

    Funkcija_Po_Adresi(&i);

    cout << "i = " << i << endl << endl;

    return 0;
}

void Funkcija_Po_Adresi(int* pokazivac_na_a)
{
    *pokazivac_na_a = -(*pokazivac_na_a);
}
```

Rezultat izvršavanja programa:

```
i = -4
```

U ovom primeru smo videli da i pokazivač može biti argument koji se prosleđuje funkciji. Funkciji je prosleđena vrednost pokazivača. Opisani način prosleđivanja agumenta se naziva prosleđivanje argumenta po adresi.

Primer 39.

Napisati program koji prosleđuje memorijsku adresu kao parametar funkcije.

```
// Primer 39

// Program demonstrira prosledjivanje argumenta po referenci

#include <iostream>

using namespace std;

void Funkcija_Po_Referenci(int&);

int main()
{
    int i = 4;

    Funkcija_Po_Referenci(i);

    cout << "i = " << i << endl << endl;

    return 0;
}

void Funkcija_Po_Referenci(int&  a)
{
    a = -a;
}
```

Rezultat izvršavanja programa:

```
i = -4
```

U ovom primeru smo videli da i memorijska adresa može biti argument koji se prosleđuje funkciji. Opisani način prosleđivanja agumenta se naziva prosleđivanje argumenta po referenci.

Primer 40.

Napisati program koji izračunava broj unetih znakova u jednoj liniji. Koristiti funkciju kojoj se string prosleđuje po adresi.

```
// Primer 40

// Program izracunava broj unetih znakova u jednoj liniji
// Koristi funkciju Duzina_Niza() za brojanje unetih znakova

#include <iostream>

using namespace std;

int Duzina_Niza(char*);
```

```
int main()
{
    char linija[81];

    cout << "Unesite niz znakova u jednoj liniji : " << endl;
    cin.getline(linija,81);

    cout << endl;
    cout << "String koji ste uneli ima " << Duzina_Niza(linija)
         << " znakova" << endl << endl;

    return 0;
}

int Duzina_Niza(char* pokazivac_na_znak)
{
    int duzina_niza = 0;

    while (*pokazivac_na_znak != '\0')
    {
        ++duzina_niza;
        ++pokazivac_na_znak;
    }

    return duzina_niza;
}
```

Rezultat izvršavanja programa:

```
Unesite niz znakova u jednoj liniji :
Tekst za probu.

String koji ste uneli ima 15 znakova
```

Program predstavlja jednu od mogućnosti prebrojavanja znakova unetih u jednom redu. Specifičnost je korišćenje pokazivača na niz kao argumenta funkcije kojoj se niz prosleđuje.

Primer 41.

Napisati program koji klasifikuje unete karaktere po tipu.

```
// Primer 41

// Program zahteva od korisnika da unese string.
// Zatim poziva funkcija za klasifikaciju unetih znakova,
// broji znakove u nekoliko katregorija i ispisuje rezultate.
```

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

using namespace std;

int main()
{
    char linija[81];

    char* pokazivac_na_znak;

    int broj_cifara = 0,
        broj_malih_slova = 0,
        broj_znakova_interpunkcije = 0,
        broj_blankova = 0,
        broj_velikih_slova = 0,
        ukupno_znakova = 0;

    cout << "Unesite niz znakova u jednoj liniji : " << endl;
    cin.getline(linija, 81);

    pokazivac_na_znak = linija;

    while (*pokazivac_na_znak != '\0')
    {
        if (isdigit(*pokazivac_na_znak))
            ++broj_cifara;

        else if (islower(*pokazivac_na_znak))
            ++broj_malih_slova;

        else if (ispunct(*pokazivac_na_znak))
            ++broj_znakova_interpunkcije;

        else if (isspace(*pokazivac_na_znak))
            ++broj_blankova;

        else if (isupper(*pokazivac_na_znak))
            ++broj_velikih_slova;

        ++pokazivac_na_znak;
    }

    ukupno_znakova = broj_cifara + broj_malih_slova +
                     broj_znakova_interpunkcije
                     + broj_blankova + broj_velikih_slova;

    cout << endl;
    cout <<"Uneti string sadrzi sledece : " << endl << endl;
    cout <<"Cifara :                " << setw(2) << broj_cifara
         << endl;
    cout <<"Malih slova :           " << setw(2)
         << broj_malih_slova <<endl;
    cout <<"Znakova interpunkcije : " << setw(2)
         << broj_znakova_interpunkcije << endl;
```

```
    cout <<"Blankova :              " << setw(2) << broj_blankova
     << endl;
    cout <<"Velikih slova :         " << setw(2)
         << broj_velikih_slova << endl;
    cout <<"-------------------------" << endl;
    cout <<"Ukupno znakova :        " << setw(2) << ukupno_znakova
         << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite niz znakova u jednoj liniji :
ah1Ad,,.>  B33@?,a   ;jjccMM.

Uneti string sadrzi sledece :

Cifara :                3
Malih slova :           8
Znakova interpunkcije : 9
Blankova :              5
Velikih slova :         4
-------------------------
Ukupno znakova :       29
```

Primer ilustruje upotrebu biblioteke `cctype` korišćenjem nekih njenih funkcija. I u ovom primeru se može videti upotreba pokazivača umesto samih vrednosti koje se ispituju.

Primer 42.

Napisati program koji konvertuje velika u mala slova i obrnuto.

```
// Primer 42

// Zahteva se unos stringa a zatim se pozivaju funkcije
// toupper() i tolower() kako bi prikazao uneti niz velikim
// odnosno malim slovima.

#include <iostream>
#include <cctype>

using namespace std;

int main()
{
    char linija[81];
```

```
    char* pokazivac_na_znak;

    cout <<"Unesite niz znakova u jednoj liniji : " << endl;
    cin.getline(linija, 81);

    cout << endl;
    cout <<"Sva velika slova :" << endl;
    pokazivac_na_znak = linija;
    while (*pokazivac_na_znak != '\0')
    {
        cout << char(toupper(*pokazivac_na_znak));
        ++pokazivac_na_znak;
    }
    cout << endl;
    cout <<"\nSva mala slova :" << endl;

    pokazivac_na_znak = linija;
    while (*pokazivac_na_znak != '\0')
    {
        cout << char(tolower(*pokazivac_na_znak));
        ++pokazivac_na_znak;
    }

    cout << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite niz znakova u jednoj liniji :
abc123.;]XYZ

Sva velika slova :
ABC123.;]XYZ

Sva mala slova :
abc123.;]xyz
```

Program ilustruje upotrebu funkcija `toupper()` i `tolower()` klase string za prevođenje malih slova u velika i obrnuto.

Primer 43.

Napisati program koji prevodi string u ceo broj.

```
// Primer 43

// Program ilustruje upotrebu funkcije atoi()
```

```
// koja prevodi string u ceo broj.

#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
    char buffer[51];

    int i;

    cout << "\nUnesite ceo broj : ";
    cin.getline(buffer, 51);

    i = atoi(buffer);

    cout << endl;
    cout << "Vrednost i je " << i << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite ceo broj : 881

Vrednost i je 881
```

Postoje tri standardne funkcije kojima se prevodi string u broj i to:
- `atoi()` - funkcija koja prevodi `string` u `integer`
- `atol()` - funkcija koja prevodi `string` u `long`
- `atof()` - funkcija koja prevodi `string` u `double`

Metoda koja je prikazana u ovom primeru ima primenu u C# programskom jeziku. Naime u C# programskom jeziku, svaki unos sa konzole se inicijalno smešta u objekat klase string, a zatim se prevodi u objekat odgovarajuće klase.

Primer 44.

Napisati program koji unetom nizu dinamički dodeljuje memoriju.

```
// Primer 44

// Program ilustruje upotrebu naredbi new i delete
// za dinamicku dodelu memorije nizu.

#include <iostream>
```

```
#include <cstdlib>
#include <cstring>

using namespace std;

int main()
{
    const int BROJ_OSOBA = 5;

    char buffer[81];

    char* osoba[BROJ_OSOBA];

    int i;

    cout << "Unsite " << BROJ_OSOBA << " imena :" << endl;

    for (i = 0; i < BROJ_OSOBA; ++i)
    {
        cout << endl;
        cout << "Ime " << i + 1 << " : ";
        cin.getline(buffer, 81);

        osoba[i] = new char [strlen(buffer) + 1];
        strcpy(osoba[i], buffer);
    }

    cout << endl;
    cout << "Uneli ste sledeca imena :" << endl;
    for (i = 0; i < BROJ_OSOBA; ++i)
    cout << endl << i+1 << " " << osoba[i];

    for (i = 0; i < BROJ_OSOBA; ++i)
        delete [] osoba[i];

    cout << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite 5 imena :

Ime 1 : Mr. Bean

Ime 2 : Del boy

Ime 3 : Rodney

Ime 4 : Trigger

Ime 5 : Boycie

Uneli ste sledeca imena :

1 Mr. Bean
2 Del boy
3 Rodney
4 Trigger
5 Boycie
```

Prilikom statičog dodeljivanja memorije, memorijski prostor se rezerviše za sve moguće članove niza, bez obzira da li će rezervisani prostor biti upotrebljen ili ne. Za razliku od statičke dodele memorije, dinamička dodela memorije nizu može znatno da uštedi memorijski prostor i ubrza rad aplikacije. Dinamičkom dodelom, u memoriju se upisuju samo postojeći elementi niza.

IX Korisnički definisani tipovi podataka i tabele

Primer 45.

Napisati program koji evidentira prodaju za određenog prodavca u toku jedne sedmice, zatim određuje ukupnu prodaju za sedam dana, kao i dan u kome je ostvarena najveća prodaja. Koristiti `typedef` i `enum`.

```
// Primer 45

// Program izracunava ukupan iznos prodaje u toku nedelje
// i pronalazi dan u nedelji kada je prodaja bila najveca
// Koristi niz pokazivaca za cuvanje dana u nedelji.
// Program koristi typedef i enum

#include <iostream>
#include <iomanip>

using namespace std;

typedef char* CSTRING;
typedef char IME [41];
typedef double IZNOS;
enum DAN {pon, uto, sre, cet, pet, sub, ned};

int main()
{
    CSTRING imena_dana[7] = {"Ponedeljak", "Utorak", "Sreda",
                             "Cetvrtak", "Petak", "Subota",
                             "Nedelja"};


    IZNOS  prodaja[7];
    IME    prodavac;
    IZNOS  najveca_prodaja,
           ukupna_prodaja;
    DAN    dan,
           dan_sa_najvecom_prodajom;

    cout << setprecision(2)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);

    cout << "Unesite ime prodavca : ";
    cin.getline(prodavac, 41);
    for (dan = pon; dan <= ned; dan = (DAN)((int)dan + 1))
    {
        cout << endl << endl;
        cout << "Unsite prodaju za dan : " << imena_dana [dan]
             << " : ";
        cin >> prodaja[dan];
    }
```

```
    ukupna_prodaja = 0;
    dan_sa_najvecom_prodajom = pon;
    najveca_prodaja = prodaja[pon];

    for (dan = pon; dan <= ned; dan = (DAN)((int)dan + 1))
    {
        if (prodaja[dan] > najveca_prodaja)
        {
             najveca_prodaja = prodaja[dan];
             dan_sa_najvecom_prodajom = dan;
        }
        ukupna_prodaja += prodaja[dan];
    }

    cout << endl << endl;
    cout << "Ukupna prodaja koju je ostvario prodavac " << prodavac
         << " je " << ukupna_prodaja << "." << endl << endl;
    cout << "Najveca dnevna prodaja je " << najveca_prodaja << ". "
         << endl << endl;
    cout << "Dan u kome je prodaja bla najveca je "
         << imena_dana [dan_sa_najvecom_prodajom] << "."
         << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite ime prodavca : Mr. Bean

Unsite prodaju za dan : Ponedeljak : 345.76

Unsite prodaju za dan : Utorak : 239.89

Unsite prodaju za dan : Sreda : 100.00

Unsite prodaju za dan : Cetvrtak : 563.99

Unsite prodaju za dan : Petak : 0.0

Unsite prodaju za dan : Subota : 789.99

Unsite prodaju za dan : Nedelja : 533.90

Ukupna prodaja koju je ostvario prodavac Mr. Bean je 2573.53.

Najveca dnevna prodaja je 789.99.

Dan u kome je prodaja bla najveca je Subota.
```

U primeru je korišćena ključna reč `typedef` kojom su definisani novi tipovi podataka, prema potrebi zadatka. Takođe je korišćena ključna reč `enum` pomoću koje su podaci definisani nabrajanjem elemenata.

Primer 46.

Napisati program koji formira strukturu podataka radi evidentiranja podataka u prodavnici delova. Zatim program zahteva unos podataka koji su definisani u strukturi i vrši ispisivanje unetih podataka na ekranu.

```
// Primer 46

// Program ilustruje deklarisanje i upotrebu strukture promenljivih
// kao i opratora clanica za pristupanje promenljivima clanicama.

#include <iostream>
#include <iomanip>

using namespace std;

struct OPIS_DELA
{
    char   sifra_dela[8];
    int    kolicina_na_stanju;
    double jedinicna_cena;
};

int main()
{
    OPIS_DELA deo;

    cout << setprecision(2)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);

    cout << "Unesite sifru dela sastavljenu od sedam znakova : ";
    cin.getline(deo. sifra_dela, 8);

    cout << endl;
    cout << "Unesite broj delova na zalihama : ";
    cin >> deo. kolicina_na_stanju;

    cout << endl;
    cout << "Unesite jedinicnu cenu dela : ";
    cin >> deo. jedinicna_cena;

    cout << endl;
    cout << "Uneli ste sledece podatke :" << endl << endl;
    cout << "Sifra dela :        " << deo. sifra_dela << endl;
    cout << "Staje na zalihama : " << setw(7)
         << deo. kolicina_na_stanju << endl;
    cout << "Jedinicna cena :    " << setw(7)
```

```
     << deo. jedinicna_cena << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite sifru dela sastavljenu od sedam znakova : AB12345

Unesite broj delova na zalihama : 62

Unesite jedinicnu cenu dela : 34.95

Uneli ste sledece podatke :

Sifra dela :        AB12345
Staje na zalihama :      62
Jedinicna cena :      34.95
```

Primetimo da je struktura podataka definisana izvan funkcije `main()`, i da se kasnije tretira kao tip promenljive. Pristup pojedinom članu strukure se vrši sintaksom `ime_strukture.ima_clana`.

Drugi deo - Objektno orijentisano programiranje u C++ programskom jeziku

X Klasa string, uvod u klase i objekte

Primer 47.

Napisati program koji deklariše objekte klase string na više različitih načina.

```
// Primer 47

// Program prikazijue razlicite nacine deklarisanja
// objekta klase string.

#include <iostream>
#include <string>

using namespace std;

int main()
{
    string niz1;
    string niz2 = "Hello";
    string niz3("Hello");
    string niz4(50, '*');

    cout << "niz1 = " << niz1 << endl;
    cout << "niz2 = " << niz2 << endl;
    cout << "niz3 = " << niz3 << endl;
    cout << "niz4 = " << niz4 << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
niz1 =
niz2 = Hello
niz3 = Hello
niz4 = **************************************************
```

Program prikazuje deklarisanje objekata klase string, kao i dodeljivanje vrednosti objektima klase string na više različitih načina. Klasa string je definisana u VS6 i u njenu implementaciju nećemo ulaziti. Biće objašnjene neke od najčešće korišćenih metoda klase string.

Primer 48.

Napisati program koji vrši unos stringa korišćenjem ključne reči getline(), pri čemu uneti string može sadržati više reči.

```
// Primer 48

// Program ilustruje upotrebu getline()
// za unos stringa koji sadrzi vise reci.

#include <iostream>
#include <string>

using namespace std;

int main()
{
    string puno_ime;
    string ime_drzave;

    cout << "Upisite svoje puno ime : ";
    getline(cin, puno_ime);

    cout << endl;
    cout << "Upisite ime drzave u kojoj zivite : ";
    getline(cin, ime_drzave);

    cout << endl << endl;
    cout << "Dobrodosli, " << puno_ime << endl;
    cout << ime_drzave << " je lepo mesto za zivot." << endl << endl;

    return 0;
 }
```

Rezultat izvršavanja programa:

```
Upisite svoje puno ime : Mr. Bean

Upisite ime drzave u kojoj zivite : United Kingdom

Dobrodosli, Mr. Bean
United Kingdom je lepo mesto za zivot.
```

Pod pojmom više reči, podrazumevamo više grupa znakova odvojenih blankovima.

Primer 49.

Napisati program koji računa prosek bodova na testovima. Ime i prezime studenta smestiti u string.

```
// Primer 49

// Program racuna prosek bodova sa tri testa,
// i zahteva unos imena i prezimena studenta.

#include <iostream>
#include <iomanip>
#include <string>

using namespace std;

int main()
{
    int  test1,             // Broj bodova sa 1. testa
         test2,             // Broj bodova sa 2. testa
         test3;             // Broj bodova sa 3. testa
    double prosek;          // Prosek bodova sa tri testa
    string ime_prezime;     // String u koji se smesta
                            // ime i prezime studenta

    cout << setprecision(1)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);

    cout << "Unesite ime i prezime studenta : ";
    getline(cin, ime_prezime);

    cout << endl;
    cout << "Unesite broj bodova sa 1. testa : ";
    cin >> test1;
    cout << "Unesite broj bodova sa 2. testa : ";
    cin >> test2;
    cout << "Unesite broj bodova sa 3. testa : ";
    cin >> test3;

    // Type cast (promena tipa) double() je
    // neophodna radi konverzije sume
    // broja bodova sa testova, koji su celi brojevi u realan broj.
    // Ukoliko to ne bi uradili izraz na desnoj strani bi dao
    // rezultat proseka kao integer bez decimalnog dela.

    prosek = double (test1 + test2 + test3) / 3;

    cout << endl;
    cout << "Prosecan broj bodova koje je ostvario student "
         << ime_prezime << " je " << prosek << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite ime i prezime studenta : Rodney Trotter

Unesite broj bodova sa 1. testa : 90
Unesite broj bodova sa 2. testa : 85
Unesite broj bodova sa 3. testa : 95

Prosecan broj bodova koje je ostvario student Rodney Trotter je 90.0
```

Uočimo promenu tipa promenljive prilikom izračunavanja prosečne ocene. Pošto je ocena ceo broj i broj ocena je ceo broj, samim tim i rezultat će biti ceo broj. Pogodno je da se prosečna ocena predstavlja kao realan broj, iz tih razloga je ispred izraza koji se izračunava navedena reč `double`. Opisana operacija se naziva promena tipa ili `type cast`.

Primer 50.

Napisati program koji vrši sabiranje-nadovezivanje stringova.

```
// Primer 50

// Progam ilustruje dodelu vrednosti stringu, nadovezivanje,
// kao i length() metodu.

#include <iostream>
#include <string>

using namespace std;

int main()
{
    string ime;
    string prezime;
    string puno_ime;

    int duzina_imena;

    cout << "Unesite svoje ime : ";
    getline(cin, ime);

    cout << endl;
    cout << "Unesite svoje prezime : ";
    getline(cin, prezime);

    puno_ime = ime + " " + prezime;
    duzina_imena = puno_ime.length() - 1;
    cout << endl <<endl;
    cout << "Pozdrav, " << puno_ime << endl;
```

```
    cout << "Vase ime sadrzi " << duzina_imena << " znakova."
         << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite svoje ime : Derek

Unesite svoje prezime : Trotter

Pozdrav, Derek Trotter
Vase ime sadrzi 12 znakova.
```

U primeru je pokazano da se sabiranje stringova vrši binarnim operatorom + kao da se radi o brojčanim vrednostima. Opisana osobina se naziva preklapanje operatora. Izvodimo zaključak da je negde u implementacji klase string definisana metoda sabiranja-nadovezivanja stringova.

Primer 51.

Napisati program koji pronalazi znak u stringu na traženoj poziciji koristeći `at()` metodu klase string.

```
// Primer 51

// Program ilustruje upotrebu at()
// metode klase string.

#include <iostream>
#include <string>

using namespace std;

int main()
{
    string rec = "Singidunum";

    int poslednji_index = rec.length() - 1;

    cout << "Prvi znak u stringu je " << rec.at(0) << endl;
    cout << "Poslednji znak u stringu je " << rec.at(poslednji_index)
         << endl;

    rec.at(0) = 'X';

    cout << "Novi string je " << rec << endl << endl;
```

```
    return 0;
}
```

Rezultat izvršavanja programa:

```
Prvi znak u stringu je S
Poslednji znak u stringu je  m
Novi string je Xingidunum
```

Metodom `at()` se ne vrši samo pristupanje određenom znaku objekta klase string, već se može vršiti i izmena posmatranog znaka.

Primer 52.

Napisati program koji vrši različite operacije nad stringovima.

```
// Primer 52

// Program ilustruje nekoliko operacija sa stringovima

#include <iostream>
#include <string>

using namespace std;

int main()
{
    string ime;
    string prezime;
    string puno_ime;
    string puno_ime_za_proveru;

    cout << "Unesite vase ime : ";
    getline(cin, ime);

    cout << endl;
    cout << "Unesite vase prezime : ";
    getline(cin, prezime);

    puno_ime = ime + " " + prezime;

    cout << endl << endl;
    cout << "Pozdrav " << puno_ime << ". Dobrodosli u program."
         << endl << endl;
    cout << "Vase ime sadrzi " << ime.length()
         << " znakova." << endl << endl;
    cout << "Vase prezime sadrzi "<< prezime.length()
         << " znakova." << endl << endl;
```

```
    cout << "Sada unesite svoje ime i prezime" << endl;
    cout << "razdvojene jednim blankom : ";

    getline(cin, puno_ime_za_proveru);

    if (puno_ime == puno_ime_za_proveru)
    {
        cout << endl << endl;
        cout << "Cestitamo ! Pratili ste uputstva." << endl << endl;
    }
    else
    {
        cout << endl;
        cout << "Niste pratili uputstva !" << endl;
        cout << "Pokrenite program ponovo." << endl << endl;
    }

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite vase ime : Rodney

Unesite vase prezime : Trotter

Pozdrav Rodney Trotter. Dobrodosli u program.

Vase ime sadrzi 6 znakova.

Vase prezime sadrzi 7 znakova.

Sada unesite svoje ime i prezime
razdvojene jednim blankom : Rodney Trotter


Cestitamo ! Pratili ste uputstva.
```

U programu je prikazano više metoda klase string: sabiranje, određivanje dužine i upoređivanje.

Primer 53.

Napisati program koji proverava da li je uneta vrednost null ili prazan string.

```
// Primer 53

// Program koristi funkciju za proveru podataka koja kao argument
```

```
// ima string i koja vraca string kojim se potvrdjuje da unos
// nije null, ili prazan string.

#include <iostream>
#include <string>

using namespace std;

string Unesi_Ime(string);

int main()
{
    string ime;
    string prezime;
    string puno_ime;
    string puno_ime_za_proveru;
    string pocetno_slovo_imena;
    string pocetno_slovo_prezimena;

    ime = Unesi_Ime("Unesite svoje ime : ");

    prezime = Unesi_Ime("Unesite svoje prezime : ");

    puno_ime = ime + " " + prezime;
    pocetno_slovo_imena = ime.at(0);
    pocetno_slovo_prezimena = prezime.at(0);

    cout << endl << endl;
    cout << "Pozdrav " << puno_ime << ". Dobrodosli u program."
         << endl << endl;
    cout << "Vase ime sadrzi " << ime.length()
          << " znakova." << endl << endl;
    cout << "Vase prezime sadrzi "<< prezime.length()
         << " znakova." << endl << endl;
    cout << "Vasi inicijali su "
         << pocetno_slovo_imena + pocetno_slovo_prezimena
         << endl << endl;

    cout << "Sada unesite svoje ime i prezime" << endl;
    cout << "razdvojene jednim blankom : ";

    getline(cin, puno_ime_za_proveru);

    if (puno_ime == puno_ime_za_proveru)
    {
        cout << endl;
        cout << "Cestitamo ! Pratili ste uputstva." << endl << endl;
    }
    else
    {
        cout << endl;
        cout << "Niste pratili uputstva !" << endl;
        cout << "Pokrenite program ponovo." << endl;
    }

    return 0;
```

```
}

string Unesi_Ime(string prosledjeni_niz)
{
    string unos;

    bool neisparavan_unos;

    do
    {
        cout << endl << prosledjeni_niz;
        getline(cin, unos);

        if (unos.empty())
        {
            cerr << "Niste uneli ime - Pokusajte ponovo." << endl;
            neisparavan_unos = true;
        }
        else
            neisparavan_unos = false;
    }
    while(neisparavan_unos);

    return unos;
}
```

Rezultat izvršavanja programa:

```
Unesite svoje ime : Derek

Unesite svoje prezime :
Niste uneli ime - Pokusajte ponovo.

Unesite svoje prezime : Trotter


Pozdrav Derek Trotter. Dobrodosli u program.

Vase ime sadrzi 5 znakova.

Vase prezime sadrzi 7 znakova.

Vasi inicijali su DT

Sada unesite svoje ime i prezime
razdvojene jednim blankom : Derek Trotter

Cestitamo ! Pratili ste uputstva.
```

Program predstavlja proširenu verziju prethodnog, s tom razlikom što je napisana funkcija za proveru da li je unet prazan string.

Primer 54.

Napisati program koji međusobno zamenjuje vrednosti dva objekta klase string korišćenjem funkcije sa pozivom argumenta po referenci.

```
// Primer 54

// Program ilustruje prosledjivanje objekta tipa string po referenci.
// Medjusobno zamenjuje vrednosti dva objekta tipa string
// koriscenjem funkcije.

#include <iostream>
#include <string>

using namespace std;

void Zamena_Stringova(string&, string&);

int main()
{
    string niz1;
    string niz2;

    cout << "Unesite prvi niz : ";
    getline(cin, niz1);

    cout << "Unesite drugi niz : ";
    getline(cin, niz2);

    cout << endl;

    cout << "Pre zamene :" << endl;
    cout << "Prvi niz je " << niz1 << endl;
    cout << "Drugi niz je " << niz2 << endl;

    Zamena_Stringova(niz1, niz2);

    cout << endl;
    cout << "Posle zamene :" << endl;
    cout << "Prvi niz je " << niz1 << endl;
    cout << "Drugi niz je " << niz2 << endl << endl;

    return 0;
}

void Zamena_Stringova(string& n1, string& n2)
{
    string privremena_promenljiva;

    privremena_promenljiva = n1;
    n1 = n2;
    n2 = privremena_promenljiva;
}
```

Rezultat izvršavanja programa:

```
Unesite prvi niz : Zdravo
Unesite drugi niz : Dovidjenja

Pre zamene :
Prvi niz je Zdravo
Drugi niz je Dovidjenja

Posle zamene :
Prvi niz je Dovidjenja
Drugi niz je Zdravo
```

Uočimo da se zamena vrednosti objekata vrši na već opisani način. Karakteristično u ovom zadatku je to što su funkciji parametri prosleđeni po referenci.

Primer 55.

Napisati program koji koristi različite metode definisane nad klasom string.

```
// Primer 55

// Program ilustruje neke od metoda koje postoje
// za operacije nad objektima klase string.

#include <iostream>
#include <string>

using namespace std;

int main()
{
    string niz1 = "This is an example string";
    string niz2 = "sample";
    string niz3;
    string interpunkcija = ".,:;!?";

    int pozicija;

    cout << "Originalni string, niz1 je : "
         << niz1 << endl << endl;

    //  Pronalazi prvo pojavljivanje reci "is" u stringu niz1
    pozicija = niz1.find("is");

    cout << "Prvo pojavljivanje reci \"is\" je na poziciji "
         << pozicija << endl << endl;

    // Pronalazi prvo pojavljivanje reci "is" u stringu
    // niz1 nakon pozicije 3
```

```
pozicija = niz1.find("is", 3);

cout << "Prvo pojavljivanje reci \"is\" nakon pozicije 3 je "
     << "na poziciji " << pozicija << endl << endl;

// Pronalazi prvo pojavljivanje reci "is" posmatrano
// sa kraja stringa niz1
pozicija = niz1.rfind("is");

cout << "Prvo pojavljivanje reci \"is\" posmatrano "
     << "sa kraja stringa niz1 je "
     << "na poziciji " << pozicija << endl << endl;

// Pokusaj pronalazenja niza koji se ne nalazi u nizu1
pozicija = niz1.find("Hello");

if (pozicija == -1)
    cerr << "Niz \"Hello\" nije pronadjen u nizu1."
         << endl << endl;

// Izdvaja niz od 8 znakova u stringu niz1 pocevsi od pozcije 11
niz3 = niz1.substr(11, 8);
cout << "Niz od 8 znakova u stringu niz1 pocevsi od pozicije "
     << "11 je : " << niz3 << endl << endl;

// Zamenjuje 7 znakova pocevsi od pozicije 11 stringom niz2
niz1.replace(11, 7, niz2);

cout << "Nakon zamene reci \"example\" " << endl
     << "recju \"sample\" u nizu1, string niz1 postaje : "
     << niz1 << endl << endl;

// Brise niz od 7 znakova iz niza1 pocevsi od pozicije 11
niz1.erase(11, 7);

cout << "Posle brisanja reci \"sample \" u stringu, "
     << "niz1 postaje : " << niz1 << endl << endl;

// Ubacuje niz "example " u niz1 pocevsi od pozicije 11
niz1.insert(11, "example ");

cout << "Nakon vracanja reci \"example \", niz1 postaje : "
     << endl << niz1 << endl << endl;

// Dodaje tacku na kraj niza
niz1.append(".");

cout << "Nakon dodavanja tacke, niz1 postaje : "
     << niz1 << endl << endl;

// Pronalazi prvo pojavljivanje znaka interpunkcije u stringu niz1
pozicija = niz1.find_first_of(interpunkcija);

cout << "Prvi znak interpunkcije u stringu niz1 je "
     << "na poziciji " << pozicija << endl << endl;
```

```
    return 0;
}
```

Rezultat izvršavanja programa:

```
Originalni string, niz1 je : This is an example string

Prvo pojavljivanje reci "is" je na poziciji 2

Prvo pojavljivanje reci "is" nakon pozicije 3 je 5

Prvo pojavljivanje reci "is" posmatrano sa kraja stringa niz1 je na
poziciji 5

Niz "Hello" nije pronadjen u nizu1.

Niz od 8 znakova u stringu niz1 pocevsi od pozicije 11 je : example

Nakon zamene reci "example"
recju "sample" u nizu1, string niz1 postaje : This is an sample string

Posle brisanja reci "sample " u stringu, niz1 postaje : This is an
string

Nakon vracanja reci "example ", niz1 postaje :
This is an example string

Nakon dodavanja tacke, niz1 postaje : This is an example string.

Prvi znak interpunkcije u stringu niz1 je na poziciji 25
```

Metode klase string koje su date u ovom primeru su karakteristične i često se koriste prilikom manipulacije objektima klase string. Naročito je važno dobro razumeti opisani primer odnosno metode: `find`, `substr`, `replace`, `erase`, `insert`, `append`.

Primer 56.

Napisati program koji određuje prodaju po danima u nedelji korišćenjem klase string.

```
// Primer 56

// Program odredjuje prodaju po danima u nedelji i pronalazi najvisi
// promet u toku nedelje. Koristi niz stringova za smestanje
// imena dana u nedelji.

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

```
using namespace std;

int main()
{
    string dani_u_nedelji[] = {"Ponedeljak", "Utorak", "Sredu",
                               "Cetvrtak","Petak", "Subotu",
                               "Nedelju"};
    double prodaja[7];
    string prodavac;

    double maksimalna_prodaja,
           ukupna_prodaja;
    int    dan,
           najbolji_dan;

    cout << setprecision(2)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);

    cout << "Unesite ime prodavca : ";
    getline(cin, prodavac);


    for (dan = 0; dan < 7; ++dan)
    {
        cout << endl;
        cout << "Unesite prodaju za "<< dani_u_nedelji[dan]<< " : ";
        cin >> prodaja[dan];
    }

    ukupna_prodaja = 0;
    najbolji_dan = 0;
    maksimalna_prodaja = prodaja[0];

    for (dan = 0; dan < 7; ++dan)
    {
        if (prodaja[dan] > maksimalna_prodaja)
        {
            maksimalna_prodaja = prodaja[dan];
            najbolji_dan = dan;
        }
        ukupna_prodaja += prodaja[dan];
    }

    cout << endl << endl;
    cout << "Ukupna prodaja koju je ostvario " << prodavac

         << " je " << ukupna_prodaja << "." << endl << endl;
    cout << "Najveca prodaja je " << maksimalna_prodaja << "."
         << endl << endl;
    cout << "Najveca prodaja je ostvarena u "
         << dani_u_nedelji[najbolji_dan]<< ".";

    cout << endl << endl;
```

```
    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite ime prodavca : Derek Trotter

Unesite prodaju za Ponedeljak: 379.90

Unesite prodaju za Utorak: 2877.95

Unesite prodaju za Sredu: 2661.90

Unesite prodaju za Cetvrtak: 178.45

Unesite prodaju za Petak: 3066.20

Unesite prodaju za Subotu: 0.00

Unesite prodaju za Nedelju: 2904.77

Ukupna prodaja koju je ostvario Derek Trotter je 12069.17.

Najveca prodaja je 3066.20.

Najveca prodaja je ostvarena u Petak.
```

Primer predstavlja rešavanje problema definisanog u primeru 45 na drugi način, bez korišćenja definisanih struktura podataka.

XI Korisničko definisanje klasa i objekata

Primer 57.

Napisati program koji kreira klasu bankovni račun i u funkciji `main()` izvršiti poziv njenih public funkcija-članica.

```
// Primer 57

// Program ilustruje upotrebu funkcija koje su clanice klase

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

class Bankovni_Racun
{
    private:
        string broj_racuna;
        double stanje_na_racunu;
        double kamatna_stopa;

    public:
        Bankovni_Racun(string, double, double);
        double Izracunaj_Kamatu();
};

Bankovni_Racun::Bankovni_Racun(string  broj,  double  stanje,  double
kamata)
{
    broj_racuna      = broj;
    stanje_na_racunu = stanje;
    kamatna_stopa    = kamata;
}

double Bankovni_Racun::Izracunaj_Kamatu()
{
    return stanje_na_racunu * kamatna_stopa;
}

int main()
{
    cout << setprecision(2)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);

    string racun;     // Promenljive za smestanje unetih podtaka
    double na_racunu;
    double stopa;

    // Podaci o racunu koje unosi korisnik
```

```
    cout << "\nUnesite broj racuna : ";
    getline(cin, racun);

    cout << endl;
    cout << "Unesite stanje na racunu : ";
    cin >> na_racunu;

    cout << endl;
    cout << "Unesite kamatnu stopu : ";
    cin >> stopa;

    // Inicijalizacija objekta
    Bankovni_Racun Racun_1(racun, na_racunu, stopa);

    cout << endl;
    cout << "Kamata na novac koji se nalazi na racunu je : "
         << Racun_1.Izracunaj_Kamatu() << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite broj racuna : 1234

Unesite stanje na racunu : 100.00

Unesite kamatnu stopu : 0.06

Kamata na novac koji se nalazi na racunu je : 6.00
```

Primetimo da se iz spoljašnje funkcije, u ovom slučaju `main()` može direktno pristupiti isključivo javnim članicama klase. Privatnim članicama klase se može pristupiti isključivo korišćenjem javni članica-funkcija, ukoliko je to dozvoljeno prilikom implementacije klase. Funkcije koje pristupaju privatnim članicama klase i ne menjaju njihovu vrednost su funkcije aksesori, dok se funkcije koje mogu da promene vrednost privatne članice nazivaju mutatori. Funkcije članice se nazivaju još i metode klase.

Primer 58.

Napisati program koji koristi proširenu verziju klase iz prethodnog primera. Klasu proširiti novim metodama.

```
// Primer 58

// Program koristi prosirenu verziju klase Bankovni_Racun
// iz Primera 51

#include <iostream>
#include <iomanip>
#include <string>

using namespace std;

class Bankovni_Racun
{
    private:
        string broj_racuna;
        double stanje_na_racunu;
        double kamatna_stopa;

    public:
        Bankovni_Racun(string, double, double);
        double Izracunaj_Kamatu();
        double Prosledi_Stanje();
        string Prosledi_Broj_Racuna();
        void Uplata(double);
        bool Isplata(double);
};

Bankovni_Racun::Bankovni_Racun(string broj, double stanje,
                               double kamata)
{
    broj_racuna      = broj;
    stanje_na_racunu = stanje;
    kamatna_stopa    = kamata;
}

double Bankovni_Racun::Prosledi_Stanje()
{
    return stanje_na_racunu;
}

string Bankovni_Racun::Prosledi_Broj_Racuna()
{
    return broj_racuna;
}

double Bankovni_Racun::Izracunaj_Kamatu()
{
    double kamatni_iznos;

    kamatni_iznos = stanje_na_racunu * kamatna_stopa;
    stanje_na_racunu += kamatni_iznos;

    return kamatni_iznos;
}

void Bankovni_Racun::Uplata(double iznos)
```

```
{
    stanje_na_racunu += iznos;
}

bool Bankovni_Racun::Isplata(double iznos)
{
    bool dopusteno;
    if (iznos <= stanje_na_racunu)
    {
        stanje_na_racunu -= iznos;
        dopusteno = true;
    }
    else
        dopusteno = false;

    return dopusteno;
}

int main()
{
    cout << setprecision(2)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);
    string racun;
    double na_racunu;
    double stopa;
    double iznos;

    cout << endl;
    cout << "Unesite broj racuna : ";
    getline(cin, racun);

    cout << endl;
    cout << "Unesite stanje na racunu : ";
    cin >> na_racunu;

    cout << endl;
    cout << "Unesite kamatnu stopu : ";
    cin >> stopa;

    Bankovni_Racun Racun_1(racun, na_racunu, stopa);

    cout << endl;
    cout << "Broj racuna je " << Racun_1.Prosledi_Broj_Racuna()
         << endl << endl;

    cout << "Trenutno stanje na racunu je "
         << Racun_1.Prosledi_Stanje()
         << endl << endl;

    cout << "Unesite iznos koji ulazete : ";
    cin >> iznos;

    Racun_1.Uplata(iznos);

    cout << endl << endl;
```

```
    cout << "Uplata od " << iznos << " je izvrsena." << endl;
    cout << "Novo stanje na racunu je " << Racun_1.Prosledi_Stanje();

    Racun_1.Izracunaj_Kamatu();

    cout << endl << endl;
    cout << "Primenjuje se kamatna stopa na iznos na racunu." << endl;
    cout << "Novo stanje na racunu je " << Racun_1.Prosledi_Stanje();

    cout << endl << endl;
    cout << "Unesite iznos koji zelite da podignete : ";
    cin >> iznos;

    if (Racun_1.Isplata(iznos))
    {
        cout << endl << endl;
        cout << "Podizanje iznos od " << iznos << " je izvrseno.";
    }
    else
    {
        cout << endl << endl;
        cout << "PODIZANJE NIJE IZVRSENO: Nemate dovoljno "
             << "novca na racunu.";
    }
    cout << endl;
    cout << "Novo stanje na racunu je " << Racun_1.Prosledi_Stanje()
         << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite broj racuna : 1234

Unesite stanje na racunu : 100.00

Unesite kamatnu stopu : 0.06

Broj racuna je 1234

Trenutno stanje na racunu je 100.00

Unesite iznos koji ulazete : 55.42

Uplata od 55.42 je izvrsena.
Novo stanje na racunu je 155.42

Primenjuje se kamatna stopa na iznos na racunu.
Novo stanje na racunu je 164.75

Unesite iznos koji zelite da podignete : 120.00

Podizanje iznos od 120.00 je izvrseno.
Novo stanje na racunu je 44.75
```

U ovom primeru su, za razliku od prethodnog, definisane i funkcije mutatori.

Primer 59.

Napisati program koji koristi proširenu verziju klase iz prethodnog primera i koji koristi preklapanje konstruktora.

```
// Primer 59

// Program koristi prosirenu verziju klase Bankovni_Racun
// Prikazuje metodu preklapanja konstruktora

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

class Bankovni_Racun
{
    private:
        string broj_racuna;
        double stanje_na_racunu;
        double kamatna_stopa;

    public:
```

```
        Bankovni_Racun(string, double, double);
        Bankovni_Racun(string, double);
        Bankovni_Racun(string);
        double Izracunaj_Kamatu();
        double Prosledi_Stanje();
        string Prosledi_Broj_Racuna();
        void Uplata(double);
        bool Isplata(double);
};

Bankovni_Racun::Bankovni_Racun(string  broj,  double  stanje,  double
kamata)
{
    broj_racuna      = broj;
    stanje_na_racunu = stanje;
    kamatna_stopa    = kamata;

    cout << "Konstruktor sa tri argumenta za racun " << broj_racuna
         << endl;
}

Bankovni_Racun::Bankovni_Racun(string broj, double stanje)
{
    broj_racuna = broj;
    stanje_na_racunu = stanje;
    kamatna_stopa = 0.02;

    cout << "Konstruktor sa dva argumenta za racun " << broj_racuna
         << endl;
}

Bankovni_Racun::Bankovni_Racun(string broj)
{
    broj_racuna = broj;
    stanje_na_racunu = 0.0;
    kamatna_stopa = 0.02;

    cout << "Konstruktor sa jednim argumentom za racun "
         << broj_racuna << endl;
}

double Bankovni_Racun::Prosledi_Stanje()
{
    return stanje_na_racunu;
}

string Bankovni_Racun::Prosledi_Broj_Racuna()
{
    return broj_racuna;
}

double Bankovni_Racun::Izracunaj_Kamatu()
{
    double kamatni_iznos;

    kamatni_iznos = stanje_na_racunu * kamatna_stopa;
```

```
    stanje_na_racunu += kamatni_iznos;

    return kamatni_iznos;
}

void Bankovni_Racun::Uplata(double iznos)
{
    stanje_na_racunu += iznos;
}

bool Bankovni_Racun::Isplata(double iznos)
{
    bool dopusteno;
    if (iznos <= stanje_na_racunu)
    {
        stanje_na_racunu -= iznos;
        dopusteno = true;
    }
    else
        dopusteno = false;

    return dopusteno;
}

int main()
{
    Bankovni_Racun Racun_3("3333", 100.00, 0.06);
    Bankovni_Racun Racun_2("2222", 200.00);
    Bankovni_Racun Racun_1("1111");

    cout << endl;
    cout << "*** Kraj Programa ***" << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Konstruktor sa tri argumenta za racun  3333
Konstruktor sa dva argumenta za racun 2222
Konstruktor sa jednim argumentom za racun 1111

*** Kraj Programa ***
```

Namena konstruktora je da generiše objekat posmatrane klase. Primećujemo da se konstruktor javlja u tri oblika, u zavisnosti od toga koji se argumenti prosleđuju konstruktoru prilikom kreiranja objekta. Kompajler donosi odluku koji konstruktor da koristi na osnovu broja i tipa prosleđenih argumenata. Podrazumevani elementi koji su navedeni u konstruktoru znatno olakšavaju rad prilikom kreiranja objekta, jer ne zahtevaju unos svih članica prilikom svakog kreiranja objekta.

Primer 60.

Napisati program koji koristi proširenu verziju klase iz prethodnog primera i u kome se prikazuje upotreba destruktora.

```
// Primer 60

// Program koristi prosirenu verziju klase Bankovni_Racun
// Ilustruje upotrebu destruktora

#include <iostream>
#include <iomanip>
#include <string>

using namespace std;

class Bankovni_Racun
{
    private:
        string broj_racuna;
        double stanje_na_racunu;
        double kamatna_stopa;

    public:
        Bankovni_Racun(string, double, double);
        Bankovni_Racun(string, double);
        Bankovni_Racun(string);
        ~Bankovni_Racun();
        double Izracunaj_Kamatu();
        double Prosledi_Stanje();
        string Prosledi_Broj_Racuna();
        void Uplata(double);
        bool Isplata(double);
};

Bankovni_Racun::Bankovni_Racun(string broj, double stanje,
                               double kamata)
{
    broj_racuna      = broj;
    stanje_na_racunu = stanje;
    kamatna_stopa    = kamata;

    cout << "Konstruktor sa tri argumenta za racun " << broj_racuna
         << endl;
}

Bankovni_Racun::Bankovni_Racun(string broj, double stanje)
{
    broj_racuna      = broj;
    stanje_na_racunu = stanje;
    kamatna_stopa    = 0.02;

    cout << "Konstruktor sa dva argumenta za racun " << broj_racuna
         << endl;
}
```

```
Bankovni_Racun::Bankovni_Racun(string broj)
{
    broj_racuna      = broj;
    stanje_na_racunu = 0.0;
    kamatna_stopa    = 0.02;

    cout << "Konstruktor sa jednim argumentom za racun "
         << broj_racuna << endl;
}

Bankovni_Racun::~Bankovni_Racun()
{
    cout << "Destruktor izvrsen za " << broj_racuna << endl;
}

double Bankovni_Racun::Prosledi_Stanje()
{
    return stanje_na_racunu;
}

string Bankovni_Racun::Prosledi_Broj_Racuna()
{
    return broj_racuna;
}

double Bankovni_Racun::Izracunaj_Kamatu()
{
    double kamatni_iznos;

    kamatni_iznos = stanje_na_racunu * kamatna_stopa;
    stanje_na_racunu += kamatni_iznos;

    return kamatni_iznos;
}

void Bankovni_Racun::Uplata(double iznos)
{
    stanje_na_racunu += iznos;
}

bool Bankovni_Racun::Isplata(double iznos)
{
    bool dopusteno;
    if (iznos <= stanje_na_racunu)
    {
        stanje_na_racunu -= iznos;
        dopusteno = true;
    }
    else
        dopusteno = false;

    return dopusteno;
}

int main()
```

```
{
    Bankovni_Racun Racun_3("3333", 100.00, 0.06);
    Bankovni_Racun Racun_2("2222", 200.00);
    Bankovni_Racun Racun_1("1111");

    cout << endl;
    cout << "*** Kraj Programa ***" << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Konstruktor sa tri argumenta za racun 3333
Konstruktor sa dva argumenta za racun 2222
Konstruktor sa jednim argumentom za racun 1111

*** Kraj Programa ***

Destruktor izvrsen za 1111
Destruktor izvrsen za 2222
Destruktor izvrsen za 3333
```

Uloga destruktora je da obriše objekat nakon prestanka njegovog korišćenja i na taj način oslobodi memorijski prostor. Primećujemo da se destruktor izvršava automatski, čak i ako nije definisan. U ovom primeru je u telu destruktora definisano ispisivanje poruke, kako bismo znali da je destruktor izvršen.

Primer 61.

Napisati program koji koristi proširenu verziju klase iz prethodnog primera i u kome se koriste podrazumevani argument i njihovo dodeljivanje objektima.

```
// Primer 61

// Program koristi prosirenu verziju klase Bankovni_Racun
// Program ilustruje podrazumevane (default) argumente
// u deklaraciji konstruktora

#include <iostream>
#include <iomanip>
#include <string>

using namespace std;

class Bankovni_Racun
{
    private:
```

```
        string broj_racuna;
        double stanje_na_racunu;
        double kamatna_stopa;

    public:
        Bankovni_Racun(string broj, double stanje = 0.00,
                       double kamata = 0.02);
        ~Bankovni_Racun();
        double Izracunaj_Kamatu();
        double Prosledi_Stanje();
        string Prosledi_Broj_Racuna();
        void Uplata(double);
        bool Isplata(double);
};

Bankovni_Racun::Bankovni_Racun(string  broj,  double  stanje,  double
kamata)
{
    broj_racuna      = broj;
    stanje_na_racunu = stanje;
    kamatna_stopa    = kamata;

    cout << "Konstruktor sa podrazumevanim argumentima za racun "
         << broj_racuna << endl;
    cout << "Stanje na racunu je " << stanje_na_racunu << endl;
    cout << "Kamatna stopa je " << kamatna_stopa << endl << endl;
}

Bankovni_Racun::~Bankovni_Racun()
{
    cout << "Destruktor izvrsen za " << broj_racuna << endl;
}

double Bankovni_Racun::Prosledi_Stanje()
{
    return stanje_na_racunu;
}

string Bankovni_Racun::Prosledi_Broj_Racuna()
{
    return broj_racuna;
}

double Bankovni_Racun::Izracunaj_Kamatu()
{
    double kamatni_iznos;

    kamatni_iznos = stanje_na_racunu * kamatna_stopa;
    stanje_na_racunu += kamatni_iznos;

    return kamatni_iznos;
}

void Bankovni_Racun::Uplata(double iznos)
{
    stanje_na_racunu += iznos;
```

```
}

bool Bankovni_Racun::Isplata(double iznos)
{
    bool dopusteno;
    if (iznos <= stanje_na_racunu)
    {
        stanje_na_racunu -= iznos;
        dopusteno = true;
    }
    else
        dopusteno = false;

    return dopusteno;
}

int main()
{
    cout << setprecision(2)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);

    Bankovni_Racun Racun_3("3333", 100.00, 0.06);
    Bankovni_Racun Racun_2("2222", 200.00);
    Bankovni_Racun Racun_1("1111");

    cout << endl;
    cout << "*** Kraj Programa ***" << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Konstruktor sa podrazumevanim argumentima za racun 3333
Stanje na racunu je 100.00
Kamatna stopa je 0.06

Konstruktor sa podrazumevanim argumentima za racun 2222
Stanje na racunu je 200.00
Kamatna stopa je 0.02

Konstruktor sa podrazumevanim argumentima za racun 1111
Stanje na racunu je 0.00
Kamatna stopa je 0.02

*** Kraj Programa ***

Destruktor izvrsen za 1111
Destruktor izvrsen za 2222
Destruktor izvrsen za 3333
```

Podrazumevani argumenti mogu znatno da ubrzaju kreiranje objekata. Slično je već opisano u prethodnim primerima.

Primer 62.

Napisati program koji koristi proširenu verziju klase iz prethodnog primera u kome se vrednost dodeljuje direktno objektu.

```
// Primer 62

// Program koristi prosirenu verziju klase Bankovni_Racun
// Program ilustruje dodeljivanje vrednosti objektu

#include <iostream>
#include <iomanip>
#include <string>

using namespace std;

class Bankovni_Racun
{
    private:
        string broj_racuna;
        double stanje_na_racunu;
        double kamatna_stopa;

    public:
        Bankovni_Racun(string broj, double stanje = 0.00,
                       double kamata = 0.02);
        double Izracunaj_Kamatu();
        double Prosledi_Stanje();
        double Prosledi_Kamatnu_Stopu();
        string Prosledi_Broj_Racuna();
        void Uplata(double);
        bool Isplata(double);
};

Bankovni_Racun::Bankovni_Racun(string broj, double stanje,
                               double kamata)
{
    broj_racuna      = broj;
    stanje_na_racunu = stanje;
    kamatna_stopa    = kamata;
}

double Bankovni_Racun::Prosledi_Stanje()
{
    return stanje_na_racunu;
}

string Bankovni_Racun::Prosledi_Broj_Racuna()
{
```

```
    return broj_racuna;
}

double Bankovni_Racun::Prosledi_Kamatnu_Stopu()
{
    return kamatna_stopa;
}

double Bankovni_Racun::Izracunaj_Kamatu()
{
    double kamatni_iznos;
    kamatni_iznos = stanje_na_racunu * kamatna_stopa;
    stanje_na_racunu += kamatni_iznos;

    return kamatni_iznos;
}

void Bankovni_Racun::Uplata(double iznos)
{
    stanje_na_racunu += iznos;
}

bool Bankovni_Racun::Isplata(double iznos)
{
    bool dopusteno;
    if (iznos <= stanje_na_racunu)
    {
        stanje_na_racunu -= iznos;
        dopusteno = true;
    }
    else
        dopusteno = false;

    return dopusteno;
}

int main()
{
    cout << setprecision(2)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);

    Bankovni_Racun Racun_1("1111", 100.00, 0.03);
    Bankovni_Racun Racun_2("2222", 200.00);

    Racun_2 = Racun_1;

    cout << "Nove vrednosti za Racun_2 su : " << endl << endl;
    cout << "broj_racuna = " << Racun_2.Prosledi_Broj_Racuna()
         << endl;
    cout << "stanje_na_racunu = " << Racun_2.Prosledi_Stanje()
         << endl;
    cout << "kamatna_stopa = " << Racun_2.Prosledi_Kamatnu_Stopu()
         << endl << endl;

    return 0;
```

```
}
```

Rezultat izvršavanja programa:

```
Nove vrednosti za Racun_2 su :

broj_racuna = 1111
stanje_na_racunu = 100.00
kamatna_stopa = 0.03
```

Prilikom dodele vrednosti jednog objekta drugom, svi podaci članice jednog objekta se direktno preslikavaju u podatke članice drugog objeka. Ponovo vidimo korišćenje preklapanja operatora.

Primer 63.

Napisati program koji kreira klasu i koristi različite funkcije sa istim imenom - preklapanje funkcija.

```
// Primer 63

// Program ilustruje preklapanje funkcija

#include <iostream>

using namespace std;

void Zamena(int&, int&);
void Zamena(double&, double&);

int main()
{
  int    n = 3,
         m = 9;
  double d = 5.7,
         e = 3.14 ;

  cout << "n = " << n << " m = " << m << endl;
  cout << "d = " << d << " e = " << e << endl;

  Zamena(n, m);
  Zamena(d, e);

  cout << endl;
  cout << "Nakon zamene :" << endl << endl;
  cout << "n = " << n << " m = " << m << endl;
  cout << "d = " << d << " e = " << e << endl << endl;

  return 0;
```

```
}
void Zamena(int& i, int& j)
{
  int temp;

  temp = i;
  i = j;
  j = temp;
}
void Zamena(double& x, double& y)
{
  double temp;

  temp = x;
  x = y;
  y = temp;
}
```

Rezultat izvršavanja programa:

```
n = 3 m = 9
d = 5.7 e = 3.14

Nakon zamene :

n = 9 m = 3
d = 3.14 e = 5.7
```

Veoma je pogodno koristiti isto ime funkcije za analogne operacije nad objektima različitih klasa. Funkcija za svaku klasu mora biti posebno definisana, iako je istog imena. Kompajler donosi odluku o tome koju će funkciju koristiti na osnovu tipa (klase) parametara koji se funkciji prosleđuju.

Primer 64.

Napisati program koji kreira klasu i koristi template za metodu klase.

```
// Primer 64

// Program ilustruje sablon (template) funkcije

#include <iostream>

using namespace std;

template <class Tip>
Tip Veci(Tip x, Tip y)
{
    return (x > y)? x: y;
```

```
}

template <class Tip>
void Zamena(Tip& x, Tip& y)
{
    Tip temp;
    temp = x;
    x = y;
    y = temp;
}

template <class TipA, class TipB>
TipA Funkcija(TipA x, TipB y)
{
    TipA r;

    r = 3 * x + 2 * y;

    return r;
}

template <class Tip>
Tip Zbir(Tip a[], int n)
{
    Tip t = 0;

    for (int i = 0; i < n; ++i)
        t += a[i];

    return t;
}

int main()
{
    int i = 3,
        j = 5;

    double d = 8.62,
           e = 4.14;

    float float_niz[6] = {1.2, 2.3, 3.4, 4.5, 5.6, 6.7};
    int   integer_niz[4] = {4, 5, 6, 7};

    cout << "i = " << i << " , j = " << j << endl;
    cout << "d = " << d << " , e = " << e << endl << endl;

    cout << "Veci od i , j je " << Veci(i, j) << endl;
    cout << "Veci od d , e je " << Veci(d, e) << endl << endl;

    Zamena(i, j);
    Zamena(d, e);

    cout << "i = " << i << " , j = " << j << endl;
    cout << "d = " << d << " , e = " << e << endl << endl;

    cout << "Funkcija() primenjena na i , d je " << Funkcija(i, d)
```

```
        << endl;
    cout << "Funkcija() primenjena na d , i je " << Funkcija(d, i)
        << endl << endl;

    cout << "Zbir clanova niza float_niz[] je "
        << Zbir(float_niz, 6) << endl;
    cout << "Zbir clanova niza integer_niz[] je "
        << Zbir(integer_niz, 4) << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
i = 3 , j = 5
d = 8.62 , e = 4.14

Veci od i , j je 5
Veci od d , e je 8.62

i = 5 , j = 3
d = 4.14 , e = 8.62

Funkcija() primenjena na i , d je 23
Funkcija() primenjena na d , i je 22.42

Zbir clanova niza float_niz[] je 23.7
Zbir clanova niza integer_niz[] je 22
```

Za razliku od prethodnog primera, u ovom primeru definišemo samo jednu funkciju koja je sposobna da prihvati objekte različitih klasa - template funkcija.

XII Korišćenje objekata

Primer 65.

Napisati program koji koristi klasu bankovni račun, kao i njene metode. Pozivom iz funkcije `main()` demonstrirati korišćenje metoda kreirane klase.

```
// Primer 65

// Program koristi prosirenu verziju klase Bankovni_Racun.
// Koristi se konstruktor sa default argumentima.
// Broj racuna se smesta u niz a u klasi Bankovni_Racun
// ime vlasnika racuna se cuva upotrebom caracter pointer-a.
// Konstruktor koristi dinamicko dodeljivanje (alokaciju) memorije.

#include <iostream>
#include <iomanip>
#include <string>

using namespace std;

class Bankovni_Racun
{
    private:
        char   broj_racuna[5];
        char*  ime_i_prezime;
        double stanje_na_racunu;
        double kamatna_stopa;

    public:
        Bankovni_Racun(char broj[], char* i_p, double stanje = 0.00,
                       double kamata = 0.04);
        ~Bankovni_Racun();
        double Izracunaj_Kamatu();
        double Prosledi_Stanje();
        void Uplata(double);
        bool Isplata(double);
        void Prikazi_Broj_Racuna();
        void Prikazi_Imei_I_Prezime();
};

Bankovni_Racun::Bankovni_Racun(char broj[], char* i_p, double stanje,
                               double kamata)
{
    strcpy(broj_racuna, broj);   // kopira prvi argument
                                 // u broj_racuna[]

    ime_i_prezime = new char[strlen(i_p) + 1];  // kreira prostor
                                                // za ime

    strcpy(ime_i_prezime, i_p);  // kopira drugi argument
                                 // u novi memorijski prostor
```

```
    stanje_na_racunu = stanje;
    kamatna_stopa    = kamata;
}

Bankovni_Racun::~Bankovni_Racun()
{
    cout << endl << endl;
    cout << "Racun broj " << broj_racuna << " je obrisan." << endl
         << endl;
    delete [] ime_i_prezime;
}

double Bankovni_Racun::Prosledi_Stanje()
{
    return stanje_na_racunu;
}

double Bankovni_Racun::Izracunaj_Kamatu()
{
    double kamatni_iznos;

    kamatni_iznos = stanje_na_racunu * kamatna_stopa;
    stanje_na_racunu += kamatni_iznos;

    return kamatni_iznos;
}

void Bankovni_Racun::Uplata(double iznos)
{
    stanje_na_racunu += iznos;
}

bool Bankovni_Racun::Isplata(double iznos)
{
    bool dopusteno;

    if (iznos <= stanje_na_racunu)
    {
        stanje_na_racunu -= iznos;
        dopusteno = true;
    }
    else
        dopusteno = false;

    return dopusteno;
}

void Bankovni_Racun::Prikazi_Broj_Racuna()
{
    cout << broj_racuna;
}

void Bankovni_Racun::Prikazi_Imei_I_Prezime()
{
    cout << ime_i_prezime;
```

```
}

int main()
{
    cout << setprecision(2)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);

    char   racun[5];
    char   buffer[81];  // Promenljiva za privremeno smestanje imena
    double na_racunu;
    double stopa;
    double iznos;

    cout << endl;
    cout << "Unesite broj racuna : ";
    cin.getline(racun, 5);

    cout << endl;
    cout << "Unesite ime vlasnika racuna : ";
    cin.getline(buffer, 81);

    cout << endl;
    cout << "Unesite stanje na racunu : ";
    cin >> na_racunu;

    cout << endl;
    cout << "Unesite kamatnu stopu : ";
    cin >> stopa;
    Bankovni_Racun Racun_1(racun, buffer, na_racunu, stopa);

    cout << endl;
    cout << "Broj racuna je ";
    Racun_1.Prikazi_Broj_Racuna();
    cout << endl;
    cout << "Vlasnika racuna je ";
    Racun_1.Prikazi_Imei_I_Prezime();

    cout << endl << endl;
    cout << "Trenutno stanje na racunu je "
         << Racun_1.Prosledi_Stanje();

    cout << endl << endl;
    cout << "Unesite iznos koji ulazete : ";
    cin >> iznos;

    Racun_1.Uplata(iznos);
    cout << endl << endl;
    cout << "Uplata od " << iznos << " je izvrsena." << endl;
    cout << "Novo stanje na racunu je "
         << Racun_1.Prosledi_Stanje() << endl;

    Racun_1.Izracunaj_Kamatu();

    cout << endl;
    cout << "Primenjuje se kamatna stopa na iznos na racunu." << endl;
```

```
    cout << "Novo stanje na racunu je "
         << Racun_1.Prosledi_Stanje() << endl << endl;

    cout << "Unesite iznos koji zelite da podignete : ";
    cin >> iznos;

    if (Racun_1.Isplata(iznos))
    {
        cout << endl <<endl;
        cout << "Podizanje iznos od " << iznos << " je izvrseno.";
    }
    else
    {
        cout << endl << endl;
        cout << "PODIZANJE NIJE IZVRSENO: Nemate dovoljno "
             << "novca na racunu";
    }
    cout << endl;
    cout << "Novo stanje na racunu je " << Racun_1.Prosledi_Stanje()
         << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite broj racuna : 1234

Unesite ime vlasnika racuna : Mr. Bean

Unesite stanje na racunu : 100.00

Unesite kamatnu stopu : 0.06

Broj racuna je 1234
Vlasnika racuna je Mr. Bean

Trenutno stanje na racunu je 100.00

Unesite iznos koji ulazete : 200.00

Uplata od 200.00 je izvrsena.
Novo stanje na racunu je 300.00

Primenjuje se kamatna stopa na iznos na racunu.
Novo stanje na racunu je 318.00

Unesite iznos koji zelite da podignete : 175.00

Podizanje iznos od 175.00 je izvrseno.
Novo stanje na racunu je 143.00

Racun broj 1234 je obrisan.
```

Program ilustruje manipulaciju objektima pozivom metoda klase.

Primer 66.

Napisati program koji međusobno dodeljuje vrednosti objektima.

```
// Primer 66

// Program ilustruje dodeljivanje jednog objekta drugome.
// Inicijalizacija objekta ne koristi konstruktor
// sa jednim argumentom.

#include <iostream>

using namespace std;

class Klasa_Za_Testiranje
{
    private:
        int n;
    public:
        Klasa_Za_Testiranje(int);
        ~Klasa_Za_Testiranje();
        int Prosledi_Vrednost();
};

Klasa_Za_Testiranje::Klasa_Za_Testiranje(int i)
{
    n = i;

    cout << endl;
    cout << "Konstruktor je izvrsen: Clanica klase je "
         << "inicijalizovana na " << n << endl;
}

Klasa_Za_Testiranje::~Klasa_Za_Testiranje()
{
    cout << endl;
    cout << "Destruktor je izvresn za objekat sa clanicom "
         << n << endl;
}

int Klasa_Za_Testiranje::Prosledi_Vrednost()
{
    return n;
}

int main()
{
    Klasa_Za_Testiranje objekat1(4);
    Klasa_Za_Testiranje objekat2(0);

    cout << endl;
```

```
    cout << "Nakon kreiranja objekata :" << endl <<endl;
    cout << "Objekat 1 : " << objekat1.Prosledi_Vrednost() << endl;
    cout << "Objekat 2 : " << objekat2.Prosledi_Vrednost()
         << endl << endl;

    objekat2 = objekat1;

    cout << "Nakon dodele vrednosti objektu :" << endl << endl;
    cout << "Objekat 1 : " << objekat1.Prosledi_Vrednost() << endl;
    cout << "Objekat 2 : " << objekat2.Prosledi_Vrednost()
         << endl << endl;

    Klasa_Za_Testiranje objekat3 = objekat1;

    cout << "Nakon inicijalizacija Objekta 3 :" << endl << endl;
    cout << "Objekat 3 : " << objekat3.Prosledi_Vrednost() << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Konstruktor je izvrsen: Clanica klase je inicijalizovana na 4
Konstruktor je izvrsen: Clanica klase je inicijalizovana na 0

Nakon kreiranja objekata :

Objekat 1 : 4
Objekat 2 : 0

Nakon dodele vrednosti objektu :

Objekat 1 : 4
Objekat 2 : 4

Nakon inicijalizacija Objekta 3 :

Objekat 3 : 4

Destruktor je izvrsen za objekat sa clanicom 4

Destruktor je izvrsen za objekat sa clanicom 4

Destruktor je izvrsen za objekat sa clanicom 4
```

U programu je demonstrirano direktno dodeljivanje vrednosti objektima na ranije opisani način.

Primer 67.

Napisati program koji koristi opciju kopije konstruktora.

```
// Primer 67

// Program ilustruje upotrebu kopije konstruktora.

#include <iostream>

using namespace std;

class Klasa_Za_Testiranje
{
    private:
        int n;

    public:
        Klasa_Za_Testiranje(int);  // Konstruktor sa jednim argumantom
        Klasa_Za_Testiranje(Klasa_Za_Testiranje&);  // Kopija
                                                    // konstruktora
        ~Klasa_Za_Testiranje();
        int Prosledi_Vrednost();
};

Klasa_Za_Testiranje::Klasa_Za_Testiranje(int i)
{
    n = i;

    cout << endl;
    cout << "Konstruktor je izvrsen: Clanica klase je "
         << "inicijalizovana na " << n << endl;
}

Klasa_Za_Testiranje::Klasa_Za_Testiranje(Klasa_Za_Testiranje& tc_r)
{
    n = tc_r.n;

    cout << endl << endl;
    cout << "Kopija konstruktora je izvrsena : "
               << "Clanica klase je inicijalizovana na " << n
               << endl << endl;
}

Klasa_Za_Testiranje::~Klasa_Za_Testiranje()
{
    cout << endl << endl;
    cout << "Destruktor je izvrsen za objekat sa clanicom " << n
         << endl;
}

int Klasa_Za_Testiranje::Prosledi_Vrednost()
{
    return n;
}
```

```
int main()
{
    Klasa_Za_Testiranje objekat1(4);
    Klasa_Za_Testiranje objekat2(0);

    cout << "Nakon kreiranja objekata :" << endl << endl;
    cout << "Objekat 1 : " << objekat1.Prosledi_Vrednost() << endl;
    cout << "Objekat 2 : " << objekat2.Prosledi_Vrednost()
         << endl << endl;

    objekat2 = objekat1;
    cout << "Nakon dodele vrednosti objektu :" << endl << endl;
    cout << "Objekat 1 : " << objekat1.Prosledi_Vrednost() << endl;
    cout << "Objekat 2 : " << objekat2.Prosledi_Vrednost()
         << endl << endl;

    Klasa_Za_Testiranje objekat3 = objekat1;

    cout << "Nakon inicijalizacija Objekta 3 :" << endl << endl;
    cout << "Objekat 3 : " << objekat3.Prosledi_Vrednost() << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Konstruktor je izvrsen: Clanica klase je inicijalizovana na 0

Nakon kreiranja objekata :

Objekat 1 : 4
Objekat 2 : 0

Nakon dodele vrednosti objektu :

Objekat 1 : 4
Objekat 2 : 4

Kopija konstruktora je izvrsena : Clanica klase je inicijalizovana na
4

Nakon inicijalizacija Objekta 3 :

Objekat 3 : 4

Destruktor je izvrsen za objekat sa clanicom 4

Destruktor je izvrsen za objekat sa clanicom 4

Destruktor je izvrsen za objekat sa clanicom 4
```

Korišćenjem opcije kopije konstruktora, istovremeno se vrši inicijalizacija objekta i dodeljivanje vrednosti njenim podacima članicama već postojećeg objekta.

Primer 68.

Napisati program koji koristi kopiju konstruktorau klasi koja ima pokazivač kao članicu.

```
// Primer 68

// Program ilustruje upotrebu copy constructor.
// u klasi koja ima pokazivac (pointer) kao clanicu.

#include <iostream>
#include <iomanip>
#include <string>

using namespace std;

class Bankovni_Racun
{
    private:
        char   broj_racuna[5];
        char*  ime_i_prezime;
        double stanje_na_racunu;
        double kamatna_stopa;

    public:
        Bankovni_Racun(char broj[], char* i_p, double stanje = 0.00,
                       double kamata = 0.04);
        Bankovni_Racun(Bankovni_Racun&);
        ~Bankovni_Racun();
        double Izracunaj_Kamatu();
        double Prosledi_Stanje();
        void Uplata(double);
        bool Isplata(double);
        void Prikazi_Broj_Racuna();
        void Prikazi_Imei_I_Prezime();
};

Bankovni_Racun::Bankovni_Racun(char broj[], char* i_p, double stanje,
                               double kamata)
{
    strcpy(broj_racuna, broj);   // kopira prvi argument
                                 // u broj_racuna[]

    ime_i_prezime = new char[strlen(i_p) + 1];  // kreira prostor
                                                // za ime

    strcpy(ime_i_prezime, i_p);   // kopira drugi argument
                                  // u novi memorijski prostor
```

```
    stanje_na_racunu = stanje;
    kamatna_stopa    = kamata;

    cout << endl << endl;
    cout << "Konstruktor izvrsen." << endl;
}

Bankovni_Racun::Bankovni_Racun(Bankovni_Racun& Racun)
{
    strcpy(broj_racuna, Racun.broj_racuna);  // kopira prvi argument
                                             // u broj_racuna[]

    ime_i_prezime = new char[strlen(Racun.ime_i_prezime) + 1];
                                             // kreira prostor za ime

    strcpy(ime_i_prezime, Racun.ime_i_prezime);
                    // kopira drugi argument u novi memorijski prostor

    stanje_na_racunu = Racun.stanje_na_racunu;
    kamatna_stopa    = Racun.kamatna_stopa;

    cout << endl << endl;
    cout << "Kopija konstruktora izvrsena." << endl;
}

Bankovni_Racun::~Bankovni_Racun()
{
    cout << endl << endl;
    cout << "Racun " << broj_racuna << " obrisan." << endl;
    delete [] ime_i_prezime;
}

double Bankovni_Racun::Prosledi_Stanje()
{
    return stanje_na_racunu;
}

double Bankovni_Racun::Izracunaj_Kamatu()
{
    double kamatni_iznos;

    kamatni_iznos = stanje_na_racunu * kamatna_stopa;
    stanje_na_racunu += kamatni_iznos;

    return kamatni_iznos;
}

void Bankovni_Racun::Uplata(double iznos)
{
    stanje_na_racunu += iznos;
}

bool Bankovni_Racun::Isplata(double iznos)
{
    bool dopusteno;
```

```
    if (iznos <= stanje_na_racunu)
    {
        stanje_na_racunu -= iznos;
        dopusteno = true;
    }
    else
        dopusteno = false;

    return dopusteno;
}

void Bankovni_Racun::Prikazi_Broj_Racuna()
{
    cout << broj_racuna;
}

void Bankovni_Racun::Prikazi_Imei_I_Prezime()
{
    cout << ime_i_prezime;
}

int main()
{
    cout << setprecision(2)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);

    char racun[5];
    char buffer[81];  // Promenljiva za privremeno smestanje imena
    double na_racunu;
    double stopa;

    cout << "Unesite broj racuna : ";
    cin.getline(racun, 5);

    cout << endl;
    cout << "Unesite ime vlasnika racuna : ";
    cin.getline(buffer, 81);

    cout << endl;
    cout << "Unesite stanje na racunu : ";
    cin >> na_racunu;

    cout << endl;
    cout << "Unesite kamatnu stopu : ";
    cin >> stopa;

    Bankovni_Racun Racun_1(racun, buffer, na_racunu, stopa);

    cout << endl << endl;
    cout << "Kreiran je racun sa brojem ";
    Racun_1.Prikazi_Broj_Racuna();
    cout << endl;
    cout << "ciji je vlasnik ";
    Racun_1.Prikazi_Imei_I_Prezime();
```

```
    cout << endl << endl;
    cout << "Racun je kreiran inicijalizacijom.";

    Bankovni_Racun Racun_2 = Racun_1;

    cout << endl << endl;
    cout << "Inicijalizovan je racun sa brojem ";
    Racun_2.Prikazi_Broj_Racuna();
    cout << endl;
    cout << "\ni vlasnikom ";
    Racun_2.Prikazi_Imei_I_Prezime();
    cout << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite broj racuna : 1234

Unesite ime vlasnika racuna : Mr. Bean

Unesite stanje na racunu : 100.00

Unesite kamatnu stopu : 0.06

Konstruktor izvrsen.

Kreiran je racun sa brojem 1234
ciji je vlasnik Mr. Bean

Racun je kreiran inicijalizacijom.

Kopija konstruktora izvrsena.

Inicijalizovan je racun sa brojem 1234
i vlasnikom Mr. Bean

Racun 1234 obrisan.

Racun 1234 obrisan.
```

U primeru je pokazano da se kopija konstruktora može koristiti, iako je pokazivač članica klase.

Primer 69.

Napisati program koji koristi constant member funkciju.

```
// Primer 69

// Program ilustruje constant member funkciju.
// Treba primetiti da naredba u main() funkciji
// prijavljuje kompajlersku gresku.

#include <iostream>

using namespace std;

class Klasa_Za_Testiranje
{
    private:
        int n;

    public:
        Klasa_Za_Testiranje(int i = 0);     // Konstruktor sa jednim
                                            // argumentom
        int Prosledi_Vrednost() const;
        void Promeni_Vrednost(int);
};

Klasa_Za_Testiranje::Klasa_Za_Testiranje(int i)
{
    n = i;

    cout << endl;
    cout << "Konstruktor je izvrsen: Clanica klase je "
         << "inicijalizovana na "
         << n << endl << endl;
}

int Klasa_Za_Testiranje::Prosledi_Vrednost() const
{
    return n;
}

void Klasa_Za_Testiranje::Promeni_Vrednost(int i)
{
    n = i;
}

int main()
{
    const Klasa_Za_Testiranje objekat1(4);

    cout << "Vrednost objekta je " << objekat1.Prosledi_Vrednost()
         << endl << endl;

    // Dodavanjem sledeceg reda ce izazvati kompajlersku gresku
    // objekat1.Promeni_Vrednost(7);

    return 0;
}
```

Rezultat izvršavanja programa:

```
Konstruktor je izvrsen: Clanica klase je inicijalizovana na 4

Vrednost objekta je 4
```

Objekat neke klase se može definisati kao `constant`, analogno promenljivima tipa `integer`, `double`… što smo videli u prethodnim primerima. To znači da se vrednosti članica objekta ne mogu menjati i da će pokušaj promene njihovih vrednosti biti prijavljen kao greška.

Primer 70.

Napisati program koji koristi kopiju konstruktora u klasi čija je jedna članica pokazivač. Omogućiti pristup privatnoj članici - imenu vlasnika računa preko pokazivača.

```
// Primer 70

// Program ilustruje upotrebu kopije konstruktora
// u klasi cija je clanica pokazivac.
// Klasa sadrzi accessor funkciju koja vraca const pointer.

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

class Bankovni_Racun
{
    private:
        char   broj_racuna[5];
        char*  ime_i_prezime;
        double stanje_na_racunu;
        double kamatna_stopa;

    public:
        Bankovni_Racun(char broj[], char* i_p, double stanje = 0.00,
                       double kamata = 0.04);
        Bankovni_Racun(const Bankovni_Racun &);
        ~Bankovni_Racun();
        double Izracunaj_Kamatu();
        double Prosledi_Stanje();
        char*  Prosledi_Ime();
        void Uplata(double);
        bool Isplata(double);
};
```

```
Bankovni_Racun::Bankovni_Racun(char broj[], char* i_p, double stanje,
                               double kamata)
{
    strcpy(broj_racuna, broj);   // kopira prvi argument
                                 // u broj_racuna[]

    ime_i_prezime = new char[strlen(i_p) + 1];  // kreira prostor
                                                // za ime

    strcpy(ime_i_prezime, i_p);  // kopira drugi argument
                                 // u novi memorijski prostor

    stanje_na_racunu = stanje;
    kamatna_stopa    = kamata;

    cout << endl << endl;
    cout << "Konstruktor izvrsen." << endl << endl;
}

Bankovni_Racun::Bankovni_Racun(const Bankovni_Racun& Racun)
{
    strcpy(broj_racuna, Racun.broj_racuna);  // kopira prvi argument
                                             // u broj_racuna[]

    ime_i_prezime = new char[strlen(Racun.ime_i_prezime) + 1];
                                             // kreira prostor za ime

    strcpy(ime_i_prezime, Racun.ime_i_prezime);
                // kopira drugi argument u novi memorijski prostor

    stanje_na_racunu = Racun.stanje_na_racunu;
    kamatna_stopa    = Racun.kamatna_stopa;

    cout << endl << endl;
    cout << "Kopija konstruktora izvrsena." << endl << endl;
}

Bankovni_Racun::~Bankovni_Racun()
{
    cout << endl << endl;
    cout << "Racun " << broj_racuna << " obrisan." << endl;
    delete [] ime_i_prezime;
}

double Bankovni_Racun::Prosledi_Stanje()
{
    return stanje_na_racunu;
}

char* Bankovni_Racun::Prosledi_Ime()
{
    return ime_i_prezime;
}

double Bankovni_Racun::Izracunaj_Kamatu()
{
```

```
    double kamatni_iznos;

    kamatni_iznos = stanje_na_racunu * kamatna_stopa;
    stanje_na_racunu += kamatni_iznos;

    return kamatni_iznos;
}

void Bankovni_Racun::Uplata(double iznos)
{
    stanje_na_racunu += iznos;
}

bool Bankovni_Racun::Isplata(double iznos)
{
    bool dopusteno;

    if (iznos <= stanje_na_racunu)
    {
        stanje_na_racunu -= iznos;
        dopusteno = true;
    }
    else
        dopusteno = false;

    return dopusteno;
}

int main()
{
    cout << setprecision(2)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);

    char racun[5];
    char buffer[81]; // Promenljiva za privremeno smestanje imena
    double na_racunu;
    double stopa;

    cout << "Unesite broj racuna : ";
    cin.getline(racun, 5);

    cout << endl;
    cout << "Unesite ime vlasnika racuna : ";
    cin.getline(buffer, 81);

    cout << endl;
    cout << "Unesite stanje na racunu : ";
    cin >> na_racunu;

    cout << endl;
    cout << "Unesite kamatnu topu :  ";
    cin >> stopa;

    cout << endl;
    cout << "Racun je kreiran.";
```

```
    Bankovni_Racun Racun_1(racun, buffer, na_racunu, stopa);

    cout << "Vlasnik racuna : " << Racun_1.Prosledi_Ime()
         << endl << endl;

    char* name = Racun_1.Prosledi_Ime();

    strcpy(name, "Neko Drugi");

    cout << "Vlasnik racuna : " << Racun_1.Prosledi_Ime();

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite broj racuna : 1234

Unesite ime vlasnika racuna : Derek Trotter

Unesite stanje na racunu : 1000.00

Unesite kamatnu stopu : 0.06

Racun je kreiran.

Konstruktor izvrsen.

Vlasnik racuna : Derek Trotter

Vlasnik racuna : Neko Drugi

Racun 1234 obrisan.
```

U ovom primeru smo prekršili postavku privatnosti podataka-članica klase tako što smo privatnoj članici pristupili preko pokazivača. Pristupom preko pokazivača, uspeli smo da izmenimo vrednost privatne članice, što nije dopušteno. U ovom primeru je izvršen direktan pristup imenu vlasnika računa.

Primer 71.

Napisati program koji koristi kopiju konstruktora u klasi čija je jedna članica pokazivač. Omogućiti pristup privatnoj članici - broju računa preko pokazivača.

```
// Primer 71

// Program ilustruje upotrebu kopije konstruktora
```

```
// u klasi koja ima pokazivac (pointer) kao clanicu.
// Klasa sadrzi accessor funkciju koja vraca const pointer.

#include <iostream>
#include <iomanip>
#include <string>

using namespace std;

class Bankovni_Racun
{
    private:
        char   broj_racuna[5];
        char*  ime_i_prezime;
        double stanje_na_racunu;
        double kamatna_stopa;
    public:
        Bankovni_Racun(char broj[], char* i_p, double stanje = 0.00,
                       double kamata = 0.04);
        Bankovni_Racun(const Bankovni_Racun&);
        ~Bankovni_Racun();
        double Izracunaj_Kamatu();
        double Prosledi_Stanje();
        const char* Prosledi_Broj_Racuna();
        const char* Prosledi_Ime();
        void Uplata(double);
        bool Isplata(double);
};

Bankovni_Racun::Bankovni_Racun(char broj[], char* i_p, double stanje,
                               double kamata)
{
    strcpy(broj_racuna, broj);    // kopira prvi argument
                                  // u broj_racuna[]

    ime_i_prezime = new char[strlen(i_p) + 1];  // kreira prostor
                                                // za ime

    strcpy(ime_i_prezime, i_p);  // kopira drugi argument
                                 // u novi memorijski prostor

    stanje_na_racunu = stanje;
    kamatna_stopa    = kamata;

    cout << endl << endl;
    cout << "Konstruktor izvrsen." << endl;
}

Bankovni_Racun::Bankovni_Racun(const Bankovni_Racun& Racun)
{
    strcpy(broj_racuna, Racun.broj_racuna);   // kopira prvi argument
                                              // u broj_racuna[]

    ime_i_prezime = new char[strlen(Racun.ime_i_prezime) + 1];
                                              // kreira prostor za ime
```

```
    strcpy(ime_i_prezime, Racun.ime_i_prezime);
                    // kopira drugi argument u novi memorijski prostor

    stanje_na_racunu = Racun.stanje_na_racunu;
    kamatna_stopa    = Racun.kamatna_stopa;

    cout << endl << endl;
    cout << "Kopija konstruktora izvrsena." << endl;
}

Bankovni_Racun::~Bankovni_Racun()
{
    cout << endl;
    cout << "Racun " << broj_racuna << " obrisan." << endl;
    delete [] ime_i_prezime;
}

double Bankovni_Racun::Prosledi_Stanje()
{
    return stanje_na_racunu;
}

const char* Bankovni_Racun::Prosledi_Broj_Racuna()
{
  return broj_racuna;
}

const char* Bankovni_Racun::Prosledi_Ime()
{
    return ime_i_prezime;
}

double Bankovni_Racun::Izracunaj_Kamatu()
{
    double kamatni_iznos;

    kamatni_iznos = stanje_na_racunu * kamatna_stopa;
    stanje_na_racunu += kamatni_iznos;

    return kamatni_iznos;
}

void Bankovni_Racun::Uplata(double iznos)
{
    stanje_na_racunu += iznos;
}

bool Bankovni_Racun::Isplata(double iznos)
{
    bool dopusteno;

    if (iznos <= stanje_na_racunu)
    {
        stanje_na_racunu -= iznos;
        dopusteno = true;
    }
```

```
    else
        dopusteno = false;

    return dopusteno;
}

int main()
{
    cout << setprecision(2)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);

    char racun[5];
    char buffer[81];  // Promenljiva za privremeno smestanje imena
    double na_racunu;
    double stopa;

    cout << "Unesite broj racuna : ";
    cin.getline(racun, 5);

    cout << endl;
    cout << "Unesite ime vlasnika racuna : ";
    cin.getline(buffer, 81);

    cout << endl;
    cout << "Unesite stanje na racunu : ";
    cin >> na_racunu;

    cout << endl;
    cout << "Unesite kamatnu stopu : ";
    cin >> stopa;

    cout << endl;
    cout << "Racun je kreiran.";

    Bankovni_Racun Racun_1(racun, buffer, na_racunu, stopa);

    cout << endl;
    cout << "Broj racuna : " << Racun_1.Prosledi_Broj_Racuna()
         << endl <<endl;
    cout << "Vlasnik racuna : " << Racun_1.Prosledi_Ime()
         << endl <<endl;

    cout << "Racun je kreiran inicijalizacijom.";

    Bankovni_Racun Racun_2 = Racun_1;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite broj racuna : 1234

Unesite ime vlasnika racuna : Derek Trotter

Unesite stanje na racunu : 1000.00

Unesite kamatnu stopu : 0.06

Racun je kreiran.

Konstruktor izvrsen.

Broj racuna : 1234

Vlasnik racuna : Derek Trotter

Racun je kreiran inicijalizacijom.

Kopija konstruktora izvrsena.

Racun 1234 obrisan.

Racun 1234 obrisan.
```

Analogno prethodnom zadatku.

Primer 72.

Napisati program koji koristi kopiju konstruktora u klasi čija je članica pokazivač. Kreirati privremenu kopiju objekta.

```
// Primer 72

// Program ilustruje upotrebu kopije konstruktora
// u klasi koja ima pokazivac kao clanicu klase.
// Kopija konstruktora se koristi za kreiranje privremene kopije
// prosledjenog objekta iz funkcije Kreiraj_Racun().

#include <iostream>
#include <iomanip>
#include <string>

using namespace std;

class Bankovni_Racun
{
    private:
        char   broj_racuna[5];
```

```
        char*  ime_i_prezime;
        double stanje_na_racunu;
        double kamatna_stopa;

    public:
        Bankovni_Racun(char broj[], char* i_p, double stanje = 0.00,
                       double kamata = 0.04);
        Bankovni_Racun(const Bankovni_Racun&);
        ~Bankovni_Racun();
        double Izracunaj_Kamatu();
        double Prosledi_Stanje();
        void Uplata(double);
        bool Isplata(double);
};

Bankovni_Racun::Bankovni_Racun(char broj[], char* i_p, double stanje,
                               double kamata)
{
    strcpy(broj_racuna, broj);  // kopira prvi argument
                                // u broj_racuna[]

    ime_i_prezime = new char[strlen(i_p) + 1];
                                // kreira prostor za ime

    strcpy(ime_i_prezime, i_p);  // kopira drugi argument
                                 // u novi memorijski prostor

    stanje_na_racunu = stanje;
    kamatna_stopa    = kamata;

    cout << endl;
    cout << "Konstruktor izvrsen." << endl;
}

Bankovni_Racun::Bankovni_Racun(const Bankovni_Racun& Racun)
{
    strcpy(broj_racuna, Racun.broj_racuna);  // kopira prvi argument
                                             // u broj_racuna[]
    ime_i_prezime = new char[strlen(Racun.ime_i_prezime) + 1];
                                            // kreira prostor za ime

    strcpy(ime_i_prezime, Racun.ime_i_prezime);
                    // kopira drugi argument u novi memorijski prostor

    stanje_na_racunu = Racun.stanje_na_racunu;
    kamatna_stopa    = Racun.kamatna_stopa;

    cout << endl;
    cout << "Kopija konstruktora izvrsena." << endl;
}

Bankovni_Racun::~Bankovni_Racun()
{
    cout << endl;
    cout << "Racun " << broj_racuna << " obrisan" << endl;
    delete [] ime_i_prezime;
```

```
}

double Bankovni_Racun::Prosledi_Stanje()
{
    return stanje_na_racunu;
}

double Bankovni_Racun::Izracunaj_Kamatu()
{
    double kamatni_iznos;

    kamatni_iznos = stanje_na_racunu * kamatna_stopa;
    stanje_na_racunu += kamatni_iznos;

    return kamatni_iznos;
}

void Bankovni_Racun::Uplata(double iznos)
{
    stanje_na_racunu += iznos;
}

bool Bankovni_Racun::Isplata(double iznos)
{
    bool dopusteno;

    if (iznos <= stanje_na_racunu)
    {
        stanje_na_racunu -= iznos;
        dopusteno = true;
    }
    else
        dopusteno = false;

    return dopusteno;
}

Bankovni_Racun Kreiraj_Racun();    // Prototip za Kreiraj_Racun()

int main()
{
    cout << setprecision(2)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);

    cout << "Racun se kreira.";
    cout << endl;

    Bankovni_Racun Racun_1 = Kreiraj_Racun();

    cout << endl;
    cout << "Racun se kreira inicijalizacijom." << endl;

    Bankovni_Racun Racun_2 = Racun_1;

    return 0;
```

```
}

Bankovni_Racun Kreiraj_Racun()
{
    char racun[5];
    char buffer[81];
    double na_racunu;
    double stopa;

    cout << endl;
    cout << "Unesite broj racuna : ";
    cin.getline(racun, 5);

    cout << endl;
    cout << "Unesite ime vlasnika racuna : ";
    cin.getline(buffer, 81);

    cout << endl;
    cout << "Unesite stanje na racunu : ";
    cin >> na_racunu;

    cout << endl;
    cout << "Unesite kamatnu stopu : ";
    cin >> stopa;

    Bankovni_Racun Racun(racun, buffer, na_racunu, stopa);

    return Racun;
}
```

Rezultat izvršavanja programa:

```
Racun se kreira.

Unesite broj racuna : 1111

Unesite ime vlasnika racuna : Mr. Bean

Unesite stanje na racunu : 100.00

Unesite kamatnu stopu : 0.06

Konstruktor izvrsen.

Kopija konstruktora izvrsena.

Racun 1111 obrisan.

Racun se kreira inicijalizacijom.

Kopija konstruktora izvrsena.

Racun 1111 obrisan.

Racun 1111 obrisan.
```

U navedenom primeru potrebno je obratiti pažnju na kreiranje privremene kopije objekta.

Primer 73.

Napisati program koji prosleđuje objekat funkciji po vrednosti.

```
// Primer 73

// Program ilustruje prosledjivanje objekta po vrednosti.
// Neophodan je copy constructor kako bi se osiguralo da se korektna
// kopija objekta prosledi funkciji.

#include <iostream>
#include <iomanip>
#include <string>

using namespace std;

class Bankovni_Racun
{
    private:
        char   broj_racuna[5];
        char*  ime_i_prezime;
```

```
        double stanje_na_racunu;
        double kamatna_stopa;

    public:
        Bankovni_Racun(char broj[], char* i_p, double stanje = 0.00,
                       double kamata = 0.04);
        ~Bankovni_Racun();
        Bankovni_Racun(const Bankovni_Racun&);
        const char* Prosledi_Broj_Racuna() const;
        const char* Prosledi_Ime() const;
        double Izracunaj_Kamatu();
        double Prosledi_Stanje();
        void Uplata(double);
        bool Isplata(double);
};

Bankovni_Racun::Bankovni_Racun(char broj[], char* i_p, double stanje,
double kamata)
{
    strcpy(broj_racuna, broj);   // kopira prvi argument
                                 // u broj_racuna[]

    ime_i_prezime = new char[strlen(i_p) + 1];  // kreira prostor
                                                // za ime

    strcpy(ime_i_prezime, i_p);  // kopira drugi argument
                                 // u novi memorijski prostor

    stanje_na_racunu = stanje;
    kamatna_stopa    = kamata;
}

Bankovni_Racun::Bankovni_Racun(const Bankovni_Racun& Racun)
{
    strcpy(broj_racuna, Racun.broj_racuna);  // kopira prvi argument
                                             // u broj_racuna[]

    ime_i_prezime = new char[strlen(Racun.ime_i_prezime) + 1];
                                               // kreira prostor za ime

    strcpy(ime_i_prezime, Racun.ime_i_prezime);
                    // kopira drugi argument u novi memorijski prostor

    stanje_na_racunu = Racun.stanje_na_racunu;
    kamatna_stopa    = Racun.kamatna_stopa;
    cout << endl;
    cout << "Kopija konstruktora izvrsena." << endl;
}

Bankovni_Racun::~Bankovni_Racun()
{
    cout << endl << endl;
    cout << "Racun " << broj_racuna << " obrisan " << endl;
    delete [] ime_i_prezime;
}
```

```
const char* Bankovni_Racun::Prosledi_Broj_Racuna() const
{
    return broj_racuna;
}

const char* Bankovni_Racun::Prosledi_Ime() const
{
    return ime_i_prezime;
}

double Bankovni_Racun::Prosledi_Stanje()
{
    return stanje_na_racunu;
}

double Bankovni_Racun::Izracunaj_Kamatu()
{
    double kamatni_iznos;

    kamatni_iznos = stanje_na_racunu * kamatna_stopa;
    stanje_na_racunu += kamatni_iznos;

    return kamatni_iznos;
}

void Bankovni_Racun::Uplata(double iznos)
{
    stanje_na_racunu += iznos;
}

bool Bankovni_Racun::Isplata(double iznos)
{
    bool dopusteno;

    if (iznos <= stanje_na_racunu)
    {
        stanje_na_racunu -= iznos;
        dopusteno = true;
    }
    else
        dopusteno = false;

    return dopusteno;
}

void Prikazi_Racun(Bankovni_Racun);     // Prototip funkcije

int main()
{
    cout << setprecision(2)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);

    char racun[5];
    char buffer[81];
    double na_racunu;
```

```
double stopa;
double iznos;

cout << endl;
cout << "Unesite broj racuna : ";
cin.getline(racun, 5);

cout << endl;
cout << "Unesite ime vlasnika racuna : ";
cin.getline(buffer, 81);

cout << endl;
cout << "Unesite stanje na racunu : ";
cin >> na_racunu;

cout << endl;
cout << "Unesite kamatnu stopu : ";
cin >> stopa;

Bankovni_Racun Racun_1(racun, buffer, na_racunu, stopa);

Prikazi_Racun(Racun_1);

cout << endl;
cout << "Unesite iznos koji uplacujete : ";
cin >> iznos;

Racun_1.Uplata(iznos);

cout << endl;
cout << "Izvrsena je uplata od " << iznos;
cout << endl;

Prikazi_Racun(Racun_1);

Racun_1.Izracunaj_Kamatu();

cout << endl;
cout << "Izracunata je kamata na novac na racunu.";
cout << endl;

Prikazi_Racun(Racun_1);

cout << endl;
cout << "Ime i prezime : " << Racun_1.Prosledi_Ime();

cout << endl << endl;
cout << "Unesite iznos koji zelite da podignete : ";
cin >> iznos;
if (Racun_1.Isplata(iznos))
{
    cout << endl;
    cout << "Podigli ste iznos od " << iznos;
}
else
{
```

```
        cout << endl;
        cout << "ISPLATA NIJE IZVRSENA: Nemate "
     << "dovoljno novca na racunu.";
    }
    cout << endl << endl;

    cout << "Ime vlasnika racuna: " << Racun_1.Prosledi_Ime();
    cout << endl;

    Prikazi_Racun(Racun_1);

    cout << endl;

    return 0;
}

void Prikazi_Racun(Bankovni_Racun Racun)
{
    cout << endl;
    cout << "Podaci za racun broj : " << Racun.Prosledi_Broj_Racuna();

    cout << endl << endl;
    cout << "Ime vlasnika racuna : " << Racun.Prosledi_Ime();

    cout << endl << endl;
    cout << "Stanje na racunu : " << Racun.Prosledi_Stanje();
}
```

Rezultat izvršavanja programa:

```
Unesite broj racuna : 1234

Unesite ime vlasnika racuna : Mr. Bean

Unesite stanje na racunu : 100.00

Unesite kamatnu stopu : 0.06

Kopija konstruktora izvrsena.

Podaci za racun broj : 1234

Ime vlasnika racuna : Mr. Bean

Stanje na racunu : 100.00

Racun 1234 obrisan.

Unesite iznos koji uplacujete : 50.00

Izvrsena je uplata od 50.00

Kopija konstruktora izvrsena.
```

```
Podaci za racun broj : 1234

Ime vlasnika racuna : Mr. Bean

Stanje na racunu : 150.00

Racun 1234 obrisan.

Izracunata je kamata na novac na racunu.

Kopija konstruktora izvrsena.

Podaci za racun broj : 1234

Ime vlasnika racuna : Mr. Bean

Stanje na racunu : 159.00

Racun 1234 obrisan.

Ime i prezime : Mr. Bean

Unesite iznos koji zelite da podignete : 60.00

Podigli ste iznos od 60.00

Ime vlasnika racuna : Mr. Bean

Kopija konstruktora izvrsena.

Podaci za racun broj : 1234

Ime vlasnika racuna : Mr. Bean

Stanje na racunu : 99.00

Racun 1234 obrisan.

Racun 1234 obrisan.
```

Analogno prosleđivanju argumenata po vrednosti za promenljive tipa `int`, `double`…

Primer 74.

Napisati program koji koristi pokazivač na objekat, kao i operator ->.

```
// Primer 74

// Program prikazuje upotrebu pokazivaca na objekat
// kao i upotrebu operatora -> za pozivanje metode.
```

```
#include <iostream>

using namespace std;

class Klasa_Za_Testiranje
{
    private:
        int n;

    public:
        Klasa_Za_Testiranje(int i = 0);        // Konstrukor sa jednim
                                               // clanom i default vrednoscu
        int Prosledi_Vrednost() const;
        void Promeni_Vrednost(int);
};

Klasa_Za_Testiranje::Klasa_Za_Testiranje(int i)
{
    n = i;

    cout << endl << endl;
    cout << "Konstruktor izvrsen: Clanica je inicijalizovana na "
         << n << endl;
}

int Klasa_Za_Testiranje::Prosledi_Vrednost() const
{
    return n;
}

void Klasa_Za_Testiranje::Promeni_Vrednost(int i)
{
    n = i;
}

int main()
{
    Klasa_Za_Testiranje objekat1(5);
    Klasa_Za_Testiranje* pokazivac_na_objekat = &objekat1;

    cout << "Vrednost objekta je "
         << pokazivac_na_objekat -> Prosledi_Vrednost();

    pokazivac_na_objekat -> Promeni_Vrednost(7);
                            // Menja vrednost na koju pokazuje pointer

    cout << endl;

    cout << "Nova vrednost objekta je "
         << pokazivac_na_objekat -> Prosledi_Vrednost();

    cout << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Constructor izvrsen: Clanica je inicijalizovana na 5
Vrednost objekta je 5
Nova vrednost objekta je 7
```

Pokazivači na objekat se mogu koristiti analogno pokazivačima na tipove promenljivih koje smo do sada razmatrali. Moguća sintaksa za pozivanje metoda klase, preko pokazivača na objekat je `->`.

Primer 75.

Napisati program koji koristi klasu `Bankovni_Racun` pozivanjem metoda sintaksom `->`. Objekat klase `Bankovni_Racun` proseditи funkciji `Prikazi_Racun()` preko pokazivača.

```
// Primer 75

// Program koristi prosirenu verziju klase Bankovni_Racun.
// Program koristi constructor sa default argumentima.
// Broj racuna se smesta u niz a ime vlasnika racuna je
// character pointer. Konstruktor koristi dinamicku dodelu memorije.
// Objekat klase Bankovni_Racun se prosledjuje preko pointera
// funkciji Prikazi_Racun()

#include <iostream>
#include <iomanip>
#include <string>

using namespace std;

class Bankovni_Racun
{
    private:
        char   broj_racuna[5];
        char*  ime_i_prezime;
        double stanje_na_racunu;
        double kamatna_stopa;

    public:
        Bankovni_Racun(char broj[], char* i_p, double stanje = 0.00,
                       double kamata = 0.04);
        ~Bankovni_Racun();
        const char* Prosledi_Broj_Racuna() const;
        const char* Prosledi_Ime() const;
        double Izracunaj_Kamatu();
        double Prosledi_Stanje();
        void Uplata(double);
```

```
        bool Isplata(double);
};

Bankovni_Racun::Bankovni_Racun(char broj[], char* i_p, double stanje,
                               double kamata)
{
    strcpy(broj_racuna, broj);   // kopira prvi argument
                                 // u broj_racuna[]

    ime_i_prezime = new char[strlen(i_p) + 1];
                                 // kreira prostor za ime

    strcpy(ime_i_prezime, i_p);  // kopira drugi argument
                                 // u novi memorijski prostor

    stanje_na_racunu = stanje;
    kamatna_stopa    = kamata;
}

Bankovni_Racun::~Bankovni_Racun()
{
    cout << endl << endl;
    cout << "Racun " << broj_racuna << " obrisan." << endl;
    delete [] ime_i_prezime;
}

const char* Bankovni_Racun::Prosledi_Broj_Racuna() const
{
    return broj_racuna;
}

const char* Bankovni_Racun::Prosledi_Ime() const
{
    return ime_i_prezime;
}

double Bankovni_Racun::Prosledi_Stanje()
{
    return stanje_na_racunu;
}

double Bankovni_Racun::Izracunaj_Kamatu()
{
    double kamatni_iznos;

    kamatni_iznos = stanje_na_racunu * kamatna_stopa;
    stanje_na_racunu += kamatni_iznos;

    return kamatni_iznos;
}

void Bankovni_Racun::Uplata(double iznos)
{
    stanje_na_racunu += iznos;
}
```

```
bool Bankovni_Racun::Isplata(double iznos)
{
    bool dopusteno;

    if (iznos <= stanje_na_racunu)
    {
        stanje_na_racunu -= iznos;
        dopusteno = true;
    }
    else
        dopusteno = false;

    return dopusteno;
}

void Prikazi_Racun(Bankovni_Racun*);     // Prototip funkcije

int main()
{
    cout << setprecision(2)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);

    char racun[5];
    char buffer[81];
    double na_racunu;
    double stopa;
    double iznos;

    cout << "Unesite broj racuna : ";
    cin.getline(racun, 5);

    cout << endl;
    cout << "Unesite ime vlasnika racuna : ";
    cin.getline(buffer, 81);

    cout << endl;
    cout << "Unesite stanje na racunu : ";
    cin >> na_racunu;

    cout << endl;
    cout << "Unesite kamatnu stopu : ";
    cin >> stopa;

    Bankovni_Racun Racun_1(racun, buffer, na_racunu, stopa);

    Prikazi_Racun(&Racun_1);

    cout << endl << endl;
    cout << "Unesite iznos koji uplacujete : ";
    cin >> iznos;

    Racun_1.Uplata(iznos);

    cout << endl <<endl;
    cout << "Izvrsena je uplata od " << iznos << endl;
```

```
    Prikazi_Racun(&Racun_1);

    Racun_1.Izracunaj_Kamatu();

    cout << endl << endl;
    cout << "Izracunata je kamata na novac na racunu." << endl;
    Prikazi_Racun(&Racun_1);

    cout << endl << endl;
    cout << "Ime i prezime : " << Racun_1.Prosledi_Ime();

    cout << endl << endl;
    cout << "Unesite iznos koji zelite da podignete : ";
    cin >> iznos;

    if (Racun_1.Isplata(iznos))
    {

        cout << endl << endl;
        cout << "Podigli ste iznos od " << iznos << endl;
    }
    else
    {
        cout << endl << endl;
        cout << "ISPLATA NIJE IZVRSENA: Nemate "
             << "dovoljno novca na racunu." << endl;
    }
    cout << endl << endl;

    Prikazi_Racun(&Racun_1);

    cout << endl;

    return 0;
}

void Prikazi_Racun(Bankovni_Racun* pokazivac_na_racun)
{
    cout << endl;

    cout << "Podaci za racun broj : "
         << pokazivac_na_racun -> Prosledi_Broj_Racuna()
         << endl << endl;

    cout << "Ime vlasnika racuna : "
         << pokazivac_na_racun -> Prosledi_Ime()
         << endl << endl;

    cout << "Stanje na racunu : "
         << pokazivac_na_racun -> Prosledi_Stanje();
}
```

Rezultat izvršavanja programa:

```
Unesite broj racuna : 1234
Unesite ime vlasnika racuna : Mr. Bean
Unsite stanje na racunu : 100.00
Unesite kamatnu stopu : 0.06
Podaci za racun broj : 1234
Ime vlasnika racuna : Mr. Bean
Stanje na racunu : 100.00
Unesite iznos koji uplacujete : 50.00
Izvrsen je uplata od 50.00
Podaci za racun broj : 1234
Ime vlasnika racuna : Mr. Bean
Stanje na racunu : 150.00
Izracunata je kamata na novac na racunu.
Podaci za racun broj : 1234
Ime vlasnika racuna : Mr. Bean
Stanje na racunu : 159.00
Ime i prezime : Mr. Bean
Unesite iznos koji zelite da podignete : 90
Podigli ste : 90.00
Podaci za racun broj : 1234
Ime vlasnika racuna : Mr. Bean
Stanje na racunu : 69.00
Racun 1234 obrisan.
```

U primeru je prikazano prosleđivanje objekta kao parametra funkcije po adresi.

Primer 76.

Napisati program koji koristi klasu `Bankovni_Racun` pozivanjem metoda sintaksom `->`. Objekat klase `Bankovni_Racun` proslediti funkciji `Prikazi_Racun()` po referenci.

```
// Primer 76

// Program koristi prosirenu verziju klase Bankovni_Racun.
// Koristi konstruktor sa default argumentima.
// Broj racuna se cuva kao niz a ime vlasnika racuna je
// character pointer. Konstruktor koristi dinamicu dodelu memorije
// Objekat klase Bankovni_Racun se prosledjuje po referenci
// funkciji Prikazi_Racun()

#include <iostream>
#include <iomanip>
#include <string>

using namespace std;

class Bankovni_Racun
{
    private:
        char   broj_racuna[5];
        char*  ime_i_prezime;
        double stanje_na_racunu;
        double kamatna_stopa;

    public:
        Bankovni_Racun(char broj[], char* i_p, double stanje = 0.00,
                       double kamata = 0.04);
        ~Bankovni_Racun();
        const char* Prosledi_Broj_Racuna() const;
        const char* Prosledi_Ime() const;
        double Izracunaj_Kamatu();
        double Prosledi_Stanje();
        void Uplata(double);
        bool Isplata(double);
};

Bankovni_Racun::Bankovni_Racun(char broj[], char* i_p, double stanje,
                               double kamata)
{
    strcpy(broj_racuna, broj);   // kopira prvi argument
                                 // u broj_racuna[]

    ime_i_prezime = new char[strlen(i_p) + 1];
                                 // kreira prostor za ime

    strcpy(ime_i_prezime, i_p);  // kopira drugi argument
                                 // u novi memorijski prostor

    stanje_na_racunu = stanje;
```

```
    kamatna_stopa    = kamata;
}

Bankovni_Racun::~Bankovni_Racun()
{
    cout << endl << endl;
    cout << "Racun " << broj_racuna << " obrisan." << endl;
    delete [] ime_i_prezime;
}

const char* Bankovni_Racun::Prosledi_Broj_Racuna() const
{
    return broj_racuna;
}

const char* Bankovni_Racun::Prosledi_Ime() const
{
    return ime_i_prezime;
}

double Bankovni_Racun::Prosledi_Stanje()
{
    return stanje_na_racunu;
}

double Bankovni_Racun::Izracunaj_Kamatu()
{
    double kamatni_iznos;

    kamatni_iznos = stanje_na_racunu * kamatna_stopa;
    stanje_na_racunu += kamatni_iznos;

    return kamatni_iznos;
}

void Bankovni_Racun::Uplata(double iznos)
{
    stanje_na_racunu += iznos;
}

bool Bankovni_Racun::Isplata(double iznos)
{
    bool dopusteno;

    if (iznos <= stanje_na_racunu)
    {
        stanje_na_racunu -= iznos;
        dopusteno = true;
    }
    else
        dopusteno = false;

    return dopusteno;
}

void Prikazi_Racun(Bankovni_Racun&);     // Prototip funkcije
```

```
int main()
{
    cout << setprecision(2)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);

    char racun[5];
    char buffer[81];
    double na_racunu;
    double stopa;
    double iznos;

    cout << "Unesite broj racuna : ";
    cin.getline(racun, 5);

    cout << endl;
    cout << "Unesite ime vlasnika racuna : ";
    cin.getline(buffer, 81);

    cout << endl;
    cout << "Unesite stanje na racunu : ";
    cin >> na_racunu;

    cout << endl;
    cout << "Unesite kamatnu stopu : ";
    cin >> stopa;

    Bankovni_Racun Racun_1(racun, buffer, na_racunu, stopa);

    Prikazi_Racun(Racun_1);

    cout << endl << endl;
    cout << "Unesite iznos koji uplacujete : ";
    cin >> iznos;

    Racun_1.Uplata(iznos);

    cout << endl << endl;
    cout << "Izvrsena je uplata od " << iznos << endl;

    Prikazi_Racun(Racun_1);

    Racun_1.Izracunaj_Kamatu();

    cout << endl << endl;
    cout << "Izracunata je kamata na novac na racunu." << endl;

    Prikazi_Racun(Racun_1);

    cout << endl << endl;
    cout << "Ime i prezime : " << Racun_1.Prosledi_Ime();

    cout << endl << endl;
    cout << "Unesite iznos koji zelite da podignete : ";
    cin >> iznos;
```

```
    if (Racun_1.Isplata(iznos))
    {
        cout << endl << endl;
        cout << "Podigli ste iznos od " << iznos << endl;
    }
    else
    {
        cout << endl << endl;
        cout << "ISPLATA NIJE IZVRSENA: Nemate "
             << "dovoljno novca na racunu." << endl;
    }
    cout << endl << endl;
    Prikazi_Racun(Racun_1);

    cout << endl;

    return 0;
}

void Prikazi_Racun(Bankovni_Racun& Racun)
{
    cout << endl;
    cout << "Podaci za racun broj : "
         << Racun.Prosledi_Broj_Racuna() << endl << endl;

    cout << "Ime vlasnika racuna : "
         << Racun.Prosledi_Ime() << endl << endl;

    cout << "Stanje na racunu : " << Racun.Prosledi_Stanje();
}
```

Rezultat izvršavanja programa:

```
Unesite broj racuna : 1234
Unesite ime vlasnika racuna : Mr. Bean
Unesite stanje na racunu : 100.00
Unesite kamatnu stopu : 0.06
Podaci za racun broj : 1234
Ime vlasnika racuna : Mr. Bean
Stanje na racunu : 100.00
Unesite iznos koji uplacujete : 50.00
Izvrsena je uplata od 50.00
Podaci za racun broj : 1234
Ime vlasnika racuna : Mr. Bean
Stanje na racunu : 150.00
Izracunata je kamata na ovac na racunu.
Podaci za racun broj : 1234
Ime vlasnika racuna : Mr. Bean
Stanje na racunu : 159.00
Ime i prezime : Mr. Bean
Unesite iznos koji zelite da podignete : 90.00
Podigli ste iznos od 90.00
Podaci za racun broj : 1234
Ime vlasnika racuna : Mr. Bean
Stanje na racunu : 69.00
Racun 1234 obrisan.
```

U primeru je prikazano prosleđivanje objekta kao parametra funkcije po referenci.

Primer 77.

Napisati program koji koristi klasu `Bankovni_Racun` koristeći dinamičku alokaciju objekta.

```
// Primer 77

// Program ilustruje dinamicku alokaciju objekata.

#include <iostream>
#include <iomanip>
#include <string>
#include <cctype>

using namespace std;

class Bankovni_Racun
{
    private:
        char   broj_racuna[5];
        char*  ime_i_prezime;
        double stanje_na_racunu;
        double kamatna_stopa;

    public:
        Bankovni_Racun(char broj[], char* i_p, double stanje = 0.00,
                       double kamata = 0.04);
        ~Bankovni_Racun();
        const char* Prosledi_Broj_Racuna() const;
        const char* Prosledi_Ime() const;
        double Izracunaj_Kamatu();
        double Prosledi_Stanje();
        void Uplata(double);
        bool Isplata(double);
};

Bankovni_Racun::Bankovni_Racun(char broj[], char* i_p, double stanje,
                               double kamata)
{
    strcpy(broj_racuna, broj);   // kopira prvi argument
                                 // u broj_racuna[]

    ime_i_prezime = new char[strlen(i_p) + 1];
                                 // kreira prostor za ime

    strcpy(ime_i_prezime, i_p);  // kopira drugi argument
                                 // u novi memorijski prostor

    stanje_na_racunu = stanje;
    kamatna_stopa    = kamata;
}

Bankovni_Racun::~Bankovni_Racun()
{
    cout << endl;
```

```
    cout << "Racun " << broj_racuna << " obrisan." << endl;
    delete [] ime_i_prezime;
}

const char* Bankovni_Racun::Prosledi_Broj_Racuna() const
{
    return broj_racuna;
}

const char* Bankovni_Racun::Prosledi_Ime() const
{
    return ime_i_prezime;
}

double Bankovni_Racun::Prosledi_Stanje()
{
    return stanje_na_racunu;
}

double Bankovni_Racun::Izracunaj_Kamatu()
{
    double kamatni_iznos;

    kamatni_iznos = stanje_na_racunu * kamatna_stopa;
    stanje_na_racunu += kamatni_iznos;

    return kamatni_iznos;
}

void Bankovni_Racun::Uplata(double iznos)
{
    stanje_na_racunu += iznos;
}

bool Bankovni_Racun::Isplata(double iznos)
{
    bool dopusteno;
    if (iznos <= stanje_na_racunu)
    {
        stanje_na_racunu -= iznos;
        dopusteno = true;
    }
    else
        dopusteno = false;

    return dopusteno;
}

void Prikazi_Racun(Bankovni_Racun&);     // Prototipovi funkcija

Bankovni_Racun* Kreiraj_Racun();

int main()
{
    cout << setprecision(2)
         << setiosflags(ios::fixed)
```

```
          << setiosflags(ios::showpoint);

    Bankovni_Racun* tabela_racuna[10];

    int brojac = 0;
    char izbor;

    cout << endl;
    cout << "Da li zelite da kreirate racun ? (D/N) : ";
    izbor = cin.get();
    cin.get();            // Brisanje ulaznog buffer-a

    while (toupper(izbor) == 'D' && brojac < 10)
    {
        tabela_racuna[brojac] = Kreiraj_Racun();
        ++brojac;
        cout << endl;
        cout << "Da li zelite da kreirate racun ? (D/N) : ";
        izbor = cin.get();
        cin.get();          // Brisanje ulaznog buffer-a
    }

    // Prikazivanje racuna

    for (int i = 0; i < brojac; ++i)
        Prikazi_Racun(*tabela_racuna[i]);

    // Brisanje memorije

    for (i = 0; i < brojac; ++i)
        delete tabela_racuna[i];

    cout << endl;

    return 0;
}

void Prikazi_Racun(Bankovni_Racun& Racun)
{
    cout << endl << endl << endl;

    cout << "Podaci za racun broj : "
         << Racun.Prosledi_Broj_Racuna() << endl << endl;

    cout << "Ime vlasnika racuna : "
         << Racun.Prosledi_Ime() << endl << endl;

    cout << "Stanje na racunu : " << Racun.Prosledi_Stanje();
}

Bankovni_Racun* Kreiraj_Racun()
{
    char racun[5];
    char buffer[81];
    double na_racunu;
    double stopa;
```

```
    Bankovni_Racun* pokazivac_na_racun;
    cout << endl;
    cout << "Unesite broj racuna : ";
    cin.getline(racun, 5);

    cout << endl;
    cout << "Unesite ime vlasnika racuna : ";
    cin.getline(buffer, 81);

    cout << endl;
    cout << "Unesite stanje na racunu : ";
    cin >> na_racunu;

    cout << endl;
    cout << "Unesite kamatnu stopu : ";
    cin >> stopa;

    cin.get();    // Ciscenje ulaznog buffer-a

    pokazivac_na_racun = new Bankovni_Racun (racun, buffer, na_racunu,
                                             stopa);

    return pokazivac_na_racun;
}
```

Rezultat izvršavanja programa:

```
Da li zelite da kreirate racun ? (D/N) : d

Unesite broj racuna : 1111

Unesite ime vlasnika racuna : Mr. Bean

Unesite stanje na racunu : 1000.00

Unesite kamatnu stopu : 0.06

Da li zelite da kreirate racun ? (D/N) : d

Unesite broj racuna : 2222

Unesite ime vlasnika racuna : Derek Trotter

Unesite stanje na racunu : 2000.00

Unesite kamatnu stopu : 0.07

Da li zelite da kreirate racun ? (D/N) : d

Unesite broj racuna : 3333

Unesite ime vlasnika racuna : Rodney Trotter
```

```
Unesite stanje na racunu : 3000.00

Unesite kamatnu stopu : 0.07

Da li zelite da kreirate racun ? (D/N) : n

Podaci za racun broj : 1111

Ime vlasnika racuna : Mr. Bean

Stanje na racunu : 1000.00

Podaci za racun broj : 2222

Ime vlasnika racuna : Derek Trotter

Stanje na racunu : 2000.00

Podaci za racun broj : 3333

Ime vlasnika racuna : Rodney Trotter

Stanje na racunu : 3000.00

Racun 1111 obrisan.

Racun 2222 obrisan.

Racun 3333 obrisan.
```

U primeru je demonstrirana dinamička dodela memorij objektu, analogno ostalim tipovima promenljivih.

Primer 78.

Napisati program koji koristi klasu `Bankovni_Racun` koristeći `static` članice.

```
// Primer 78

// Proram ilustruje korscene static podataka clanica i funkcija.

#include <iostream>
#include <iomanip>
#include <string>
#include <cctype>

using namespace std;

class Bankovni_Racun
{
    private:
```

```
        char   broj_racuna[5];
        char*  ime_i_prezime;
        double stanje_na_racunu;
        static double kamatna_stopa;
            // static podatak clanica klase
        static int broj_otvorenih_racuna;
            // static podatak clanica klase

    public:
        Bankovni_Racun(char broj[], char* i_p, double stanje = 0.00);
        ~Bankovni_Racun();
        const char* Prosledi_Broj_Racuna() const;
        const char* Prosledi_Ime() const;
        double Izracunaj_Kamatu();
        double Prosledi_Stanje();
        void Uplata(double);
        bool Isplata(double);

        static int Ukupo_Racuna();   // deklarisanje static metode
};

// Definisanje static podataka clanica klase

double Bankovni_Racun::kamatna_stopa = 0.040;

int Bankovni_Racun::broj_otvorenih_racuna = 0;

Bankovni_Racun::Bankovni_Racun(char broj[], char* i_p, double stanje)
{
    strcpy(broj_racuna, broj);  // kopira prvi argument
                                // u broj_racuna[]

    ime_i_prezime = new char[strlen(i_p) + 1];
                                // kreira prostor za ime

    strcpy(ime_i_prezime, i_p);  // kopira drugi argument
                                 // u novi memorijski prostor

    stanje_na_racunu = stanje;

    ++broj_otvorenih_racuna;
}

Bankovni_Racun::~Bankovni_Racun()
{
    cout << endl << endl;
    cout << "Racun " << broj_racuna << " obrisan." << endl;
    delete [] ime_i_prezime;

    --broj_otvorenih_racuna;

    cout << endl;
    cout << "Preostali broj otvorenih racuna : "
         << broj_otvorenih_racuna << endl;
}
```

```
const char* Bankovni_Racun::Prosledi_Broj_Racuna() const
{
    return broj_racuna;
}

const char* Bankovni_Racun::Prosledi_Ime() const
{
    return ime_i_prezime;
}

double Bankovni_Racun::Prosledi_Stanje()
{
    return stanje_na_racunu;
}

double Bankovni_Racun::Izracunaj_Kamatu()
{
    double kamatni_iznos;

    kamatni_iznos = stanje_na_racunu * kamatna_stopa;
    stanje_na_racunu += kamatni_iznos;

    return kamatni_iznos;
}

void Bankovni_Racun::Uplata(double iznos)
{
    stanje_na_racunu += iznos;
}

bool Bankovni_Racun::Isplata(double iznos)
{
    bool dopusteno;

    if (iznos <= stanje_na_racunu)
    {
        stanje_na_racunu -= iznos;
        dopusteno = true;
    }
    else
        dopusteno = false;

    return dopusteno;
}

int Bankovni_Racun::Ukupo_Racuna()
{
   return broj_otvorenih_racuna;
}

void Prikazi_Racun(Bankovni_Racun&);    // Prototipovi funkcija

Bankovni_Racun* Kreiraj_Racun();

int main()
{
```

```
    cout << setprecision(2)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);

    Bankovni_Racun* tabela_racuna[10];

    int brojac = 0;
    char izbor;

    cout << endl;
    cout << "Da li zelite da kreirate racun ? (D/N) : ";
    izbor = cin.get();
    cin.get();            // Brisanje ulaznog buffer-a

    while (toupper(izbor) == 'D' && brojac < 10)
    {
        tabela_racuna[brojac] = Kreiraj_Racun();
        tabela_racuna[brojac] -> Izracunaj_Kamatu();
        ++brojac;

        cout << endl;
        cout << "Da li zelite da kreirate racun ? (D/N) : ";
        izbor = cin.get();
        cin.get();          // Brisanje ulaznog buffer-a
    }

    // Prikazivanje racuna

    for (int i = 0; i < brojac; ++i)
        Prikazi_Racun(*tabela_racuna[i]);

    // Brisanje memorije

    for (i = 0; i < brojac; ++i)
        delete tabela_racuna[i];

    cout << endl;

    return 0;
}

void Prikazi_Racun(Bankovni_Racun& Racun)
{
    cout << endl << endl << endl;
    cout << "Podaci za racun broj : "
         << Racun.Prosledi_Broj_Racuna() << endl << endl;

    cout << "Ime vlasnika racuna : "
         << Racun.Prosledi_Ime() << endl << endl;

    cout << "Stanje na posmatranom racunu : "
         << Racun.Prosledi_Stanje();
}

Bankovni_Racun* Kreiraj_Racun()
{
```

```
    char racun[5];
    char buffer[81];
    double na_racunu;

    Bankovni_Racun* pokazivac_na_racun;

    cout << endl;
    cout << "Unesite broj racuna : ";
    cin.getline(racun, 5);
    cout << endl;

    cout << "Unesite ime vlasnika racuna : ";
    cin.getline(buffer, 81);
    cout << endl;

    cout << "Unesite stanje na racunu : ";
    cin >> na_racunu;

    cin.get();    // Ciscenje ulaznog buffer-a

    pokazivac_na_racun = new Bankovni_Racun (racun,
                                             buffer, na_racunu);

    cout << endl;
    cout << "Broj otvorenih racuna je : "
         << Bankovni_Racun::Ukupo_Racuna() << endl;

    return pokazivac_na_racun;
}
```

Rezultat izvršavanja programa:

```
Da li zelite da kreirate racun ? (D/N) : d

Unesite broj racuna : 1111

Unesite ime vlasika racuna : Mr. Bean

Unesite stanje na racunu : 100.00

Broj otvorenih racuna je : 1

Da li zelite da kreirate racun ? (D/N) : d

Unesite broj racuna : 2222

Unesite ime vlasika racuna : Derek Trotter

Unesite stanje na racunu : 200.00

Broj otvorenih racuna je : 2

Da li zelite da kreirate racun ? (D/N) : d
```

```
Unesite broj racuna : 3333

Unesite ime vlasika racuna : Rodney Trotter

Unesite stanje na racunu : 300.00

Broj otvorenih racuna je : 3

Da li zelite da kreirate racun ? (D/N) : n

Podaci za racun broj : 1111

Ime vlasnika racuna : Mr. Bean

Stanje na posmatranom racunu : 104.00

Podaci za racun broj : 2222

Ime vlasnika racuna : Derek Troter

Stanje na posmatranom racunu : 208.00

Podaci za racun broj : 3333

Ime vlasnika racuna : Rodney Trotter

Stanje na posmatranom racunu : 312.00

Racun 1111 obrisan.
Preostali broj otvorenih racuna : 2

Racun 2222 obrisan.
Preostali broj otvorenih racuna: 1

Racun 3333 obrisan.
Preostali broj otvorenih racuna: 0
```

Primer ilustruje upotrebu `static` članica u klasi.

XIII Nasleđivanje

Primer 79.

Napisati program koji izvršava konstruktore i destruktore u osnovnoj i izvedenoj klasi.

```
// Primer 79

// Program pokazuje  kako se izvrsavaju konstruktori i destruktori
// u osnovnoj klasi

#include <iostream>

using namespace std;

class Osnovna_Klasa
{
    protected:
        int n;

    public:
        Osnovna_Klasa(int i = 0);  // Konstruktor sa jednim argumentom
        ~Osnovna_Klasa();
};

Osnovna_Klasa::Osnovna_Klasa(int i) : n(i)
{
    cout << endl;
    cout << "Izvrsen konstruktor klase Osnovna_Klasa: "
         << "Podatak clanica inicijalizovana na " << n << endl;
}

Osnovna_Klasa::~Osnovna_Klasa()
{
    cout << endl;
    cout << "Izvrsen destruktor klase Osnovna_Klasa "
         << "za objekat sa podatkom clanicom " << n << endl;
}

class Izvedena_Klasa : public Osnovna_Klasa
{
    protected:
        int m;

    public:
        Izvedena_Klasa(int j = 0);
        ~Izvedena_Klasa();
};

Izvedena_Klasa::Izvedena_Klasa(int j) : Osnovna_Klasa(j+1), m(j)
{
    cout << endl;
    cout << "Izvrsen konstruktor za izvedenu klasu Izvedena_Klasa: "
```

```
          << "Podatak clanica inicijalizovana na "
          << m <<endl;
}

Izvedena_Klasa::~Izvedena_Klasa()
{
    cout << endl;
    cout << "Izvrsen destruktor klase Izvedena_Klasa "
          << "za objekat sa podatkom clanicom "
          << m << endl;
}

int main()
{
    cout << "Kreira se objekat klase Osnovna_Klasa : " << endl;
    Osnovna_Klasa obekt_osnovne_klase(4);
    cout << endl;
    cout << "Kreira se objekat klase Izvedena_Klasa : " << endl;
    Izvedena_Klasa obekt_izvedene_klase(7);

    return 0;
}
```

Rezultat izvršavanja programa:

```
Kreira se objekat klase Osnovna_Klasa :

Izvrsen konstruktor klase Osnovna_Klasa: Podatak clanica
inicijalizovana na 4

Kreira se objekat klase Izvedena_Klasa :

Izvrsen konstruktor klase Osnovna_Klasa: Podatak clanica
inicijalizovana na 8

Izvrsen konstruktor za izvedenu klasu Izvedena_Klasa: Podatak clanica
inicijalizovana na 7

Izvrsen destruktor klase Izvedena_Klasa za objekat sa podatkom
clanicom 7

Izvrsen destruktor klase Osnovna_Klasa za objekat sa podatkom clanicom
8

Izvrsen destruktor klase Osnovna_Klasa za objekat sa podatkom clanicom
4
```

Prvo se izvršava konstruktor osnovne klase, a zatim konstruktor izvedene klase. Izvršavanje destruktora se odvija obrnutim redom.

Primer 80.

Napisati program koji koristi kopiju konstruktora ukoliko postoji izvedena klasa.

```
// Primer 80

// Program ilustruje primenu kopije konstruktora u izvedenoj klasi.

#include <iostream>

using namespace std;

class Osnovna_Klasa
{
    protected:
        int n;

    public:
        Osnovna_Klasa(int i = 0);  // Konstruktor sa jednim argumentom
        Osnovna_Klasa(const Osnovna_Klasa &);  // Kopija konstruktora
        ~Osnovna_Klasa();
};

Osnovna_Klasa::Osnovna_Klasa(int i) : n(i)
{
    cout << endl;
    cout << "Izvrsen konstruktor klase Osnovna_Klasa: "
         << "Podatak clanica inicijalizovana na " << n << endl;
}

Osnovna_Klasa::Osnovna_Klasa(const Osnovna_Klasa &
                             referenca_na_objekat_osnovne_klase)
              : n(referenca_na_objekat_osnovne_klase.n)
{
    cout << endl;
    cout << "Izvrsena Kopija konstruktora za klasu Osnovna_Klasa: "
         << "Podatak clanica inicijalizovana na " << n << endl;
}

Osnovna_Klasa::~Osnovna_Klasa()
{
    cout << endl;
    cout << "Izvrsen destruktor klase Osnovna_Klasa "
         << "za objekat sa podatkom clanicom " << n << endl;
}

class Izvedena_Klasa : public Osnovna_Klasa
{
    protected:
        int m;

    public:
        Izvedena_Klasa(int j = 0);
        Izvedena_Klasa(const Izvedena_Klasa &);
        ~Izvedena_Klasa();
```

```
};

Izvedena_Klasa::Izvedena_Klasa(int j) : Osnovna_Klasa(j+1), m(j)
{
    cout << endl;
    cout << "Izvrsen konstruktor za izvedenu klasu Izvedena_Klasa: "
         << "Podatak clanica inicijalizovana na "
         << m << endl;
}

Izvedena_Klasa::Izvedena_Klasa(const Izvedena_Klasa &
                               referenca_na_objekat_izvedene_klase)
               : Osnovna_Klasa(referenca_na_objekat_izvedene_klase),
                 m(referenca_na_objekat_izvedene_klase.m)
{
    cout << endl;
    cout << "Izvrsena je Kopija konstruktora "
         << "za izvedenu klasu Izvedena_Klasa: "
         << "Podatak clanica inicijalizovana na " << m << endl;
}

Izvedena_Klasa::~Izvedena_Klasa()
{
    cout << endl;
    cout << "Izvrsen destruktor klase Izvedena_Klasa "
         << "za objekat sa podatkom clanicom "
         << m << endl;
}

int main()
{
    Izvedena_Klasa objekat_izvedene_klase_1(7);
    Izvedena_Klasa objekat_izvedene_klase_2 =
        objekat_izvedene_klase_1;
    cout << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Izvrsen konstruktor klase Osnovna_Klasa: Podatak clanica
inicijalizovana na 8

Izvrsen konstruktor za izvedenu klasu Izvedena_Klasa: Podatak clanica
inicijalizovana na 7

Izvrsena Kopija konstruktora za klasu Osnovna_Klasa: Podatak clanica
inicijalizovana na 8

Izvrsena Kopija konstruktora za izvedenu klasu Izvedena_Klasa: Podatak
clanica inicijalizovana na 7

Izvrsen destruktor klase Izvedena_Klasa za objekat sa podatkom
clanicom 7

Izvrsen destruktor klase Osnovna_Klasa za objekat sa podatkom clanicom
8
```

Program ilustruje kopiju konstruktora u klasi koja sadrži izvedenu klasu.

Primer 81.

Napisati program koji kreira funkciju u osnovnoj, ali ne i izvedenoj klasi.

```
// Primer 81

// Progam ilustruje nasledjivanje funkcija.

#include <iostream>

using namespace std;

class Osnovna_Klasa
{
    protected:
        int n;

    public:
        Osnovna_Klasa(int i = 0);  // Konstruktor sa jednim argumentom
        ~Osnovna_Klasa();
        int Prosledi_Vrednost();
};

Osnovna_Klasa::Osnovna_Klasa(int i) : n(i)
{
    cout << endl;
    cout << "Izvrsen konstruktor klase Osnovna_Klasa: "
         << "Podatak clanica inicijalizovana na " << n << endl;
```

```
}

Osnovna_Klasa::~Osnovna_Klasa()
{
    cout << endl;
    cout << "Izvrsen destruktor klase Osnovna_Klasa "
         << "za objekat sa podatkom clanicom " << n << endl;
}

int Osnovna_Klasa::Prosledi_Vrednost()
{
    return n;
}

class Izvedena_Klasa : public Osnovna_Klasa
{
    protected:
        int m;

    public:
        Izvedena_Klasa(int j = 0);
        ~Izvedena_Klasa();
};

Izvedena_Klasa::Izvedena_Klasa(int j) : Osnovna_Klasa(j+1), m(j)
{
    cout << endl;
    cout << "Izvrsen konstruktor za izvedenu klasu Izvedena_Klasa: "
         << "Podatak clanica inicijalizovana na " << m << endl;
}

Izvedena_Klasa::~Izvedena_Klasa()
{
    cout << endl;
    cout << "Izvrsen destruktor klase Izvedena_Klasa "
         << "za objekat sa podatkom clanicom " << m << endl;
}

int main()
{
    Izvedena_Klasa objekat_izvedene_klase(7);

    cout << endl;
    cout << "Vrednost koju vraca funkcija Prosledi_Vrednost() je "
         << objekat_izvedene_klase.Prosledi_Vrednost();

    cout << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Izvrsen konstruktor klase Osnovna_Klasa: Podatak clanica
inicijalizovana na 8

Izvrsen konstruktor za izvedenu klasu Izvedena_Klasa: Podatak clanica
inicijalizovana na 7

Vrednost koju vraca funkcija Prosledi_Vrednost() je 8

Izvrsen destruktor klase Izvedena_Klasa za objekat sa podatkom
clanicom 7

Izvrsen destruktor klase Osnovna_Klasa za objekat sa podatkom clanicom
8
```

Uočavamo da funkcijom `Prosledi_Vrednost()` možemo pristupiti isključivo članici osnovne klase. Članici izvedene klase ne možemo pristupiti na način prikazan u ovom primeru.

Primer 82.

Napisati program koji koristi preklapanje metoda.

```
// Primer 82

// Program ilustruje preklapanje metoda.

#include <iostream>

using namespace std;

class Osnovna_Klasa
{
    protected:
        int n;

    public:
        Osnovna_Klasa(int i = 0);  // Konstruktor sa jednim argumentom
        Osnovna_Klasa(Osnovna_Klasa &);  // Copy constructor
        ~Osnovna_Klasa();
       int Prosledi_Vrednost();
};

Osnovna_Klasa::Osnovna_Klasa(int i) : n(i)
{
    cout << endl;
    cout << "Izvrsen konstruktor klase Osnovna_Klasa: "
         << "Podatak clanica inicijalizovana na " << n << endl;
```

```
}

Osnovna_Klasa::Osnovna_Klasa(Osnovna_Klasa                                    &
referenca_na_objekat_osnovne_klase)
             : n(referenca_na_objekat_osnovne_klase.n)
{
    cout << endl;
    cout << "Izvrsena Kopija konstruktora za klasu Osnovna_Klasa: "
         << "Podatak clanica inicijalizovana na " << n << endl;
}

Osnovna_Klasa::~Osnovna_Klasa()
{
    cout << endl;
    cout << "Izvrsen destruktor klase Osnovna_Klasa "
         << "za objekat sa podatkom clanicom " << n << endl;
}

int Osnovna_Klasa::Prosledi_Vrednost()
{
    return n;
}

class Izvedena_Klasa : public Osnovna_Klasa
{
    protected:
        int m;

    public:
        Izvedena_Klasa(int j = 0);
        Izvedena_Klasa(Izvedena_Klasa &);
        ~Izvedena_Klasa();
        int Prosledi_Vrednost();
};

Izvedena_Klasa::Izvedena_Klasa(int j) : Osnovna_Klasa(j+1), m(j)
{
    cout << endl;
    cout << "Izvrsen konstruktor za izvedenu klasu Izvedena_Klasa: "
         << "Podatak clanica inicijalizovana na " << m << endl;
}

Izvedena_Klasa::Izvedena_Klasa(Izvedena_Klasa &
                               referenca_na_objekat_izvedene_klase)
                  :Osnovna_Klasa(referenca_na_objekat_izvedene_klase),
                   m(referenca_na_objekat_izvedene_klase.m)
{
    cout << endl;
    cout << "Izvrsena Kopija konstruktora "
         << "za izvedenu klasu Izvedena_Klasa: "
         << "Podatak clanica inicijalizovana na " << m << endl;
}

Izvedena_Klasa::~Izvedena_Klasa()
{
    cout << endl;
```

```
    cout << "Izvrsen destruktor klase Izvedena_Klasa "
         << "za objekat sa podatkom clanicom " << m << endl;
}

int Izvedena_Klasa::Prosledi_Vrednost()
{
    return m;
}

int main()
{
    Osnovna_Klasa objekat_osnovne_klase(3);
    Izvedena_Klasa objekat_izvedene_klase(7);

    cout << endl;
    cout << "Vrenost n u objekatu klase Osnovna_Klasa je "
         << objekat_osnovne_klase.Prosledi_Vrednost() << endl <<endl;
    cout << "Vrenost m u objekatu klase Izvedena_Klasa je "
         << objekat_izvedene_klase.Prosledi_Vrednost();
    cout << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Izvrsen konstruktor klase Osnovna_Klasa: Podatak clanica
inicijalizovana na 3

Izvrsen konstruktor klase Osnovna_Klasa: Podatak clanica
inicijalizovana na 8

Izvrsen konstruktor za izvedenu klasu Izvedena_Klasa: Podatak clanica
inicijalizovana na 7

Vrenost n u objektu klase Osnovna_Klasa je  3

Vrenost m u objektu klase Izvedena_Klasa je 7

Izvrsen destruktor klase Izvedena_Klasa za objekat sa podatkom
clanicom 7

Izvrsen destruktor klase Osnovna_Klasa za objekat sa podatkom clanicom
8

Izvrsen destruktor klase Osnovna_Klasa za objekat sa podatkom clanicom
3
```

Ukoliko definišemo funkciju i u izvedenoj klasi, moguće je, u ovom slučaju, pristupiti i članici izvedene klase. Uočimo da funkcija `Prosledi_Vrednost()` prosleđuje vrednost članice osnovne klase, kada se primenjuje nad objektom osnovne klase. Ista

funkcija prosleđuje vrednost članice izvedene klase kada se primenjuje nad objektom izvedene klase.

Primer 83.

Napisati program u kome se koristi scope operatora.

```
// Primer 83

// Program ilustruje oblast primene (scope) operatora.

#include <iostream>

using namespace std;

class Osnovna_Klasa
{
    protected:
        int n;

    public:
        Osnovna_Klasa(int i = 0);  // Konstruktor sa jednim argumentom
        Osnovna_Klasa(const Osnovna_Klasa &);  // Kopija konstruktora
        ~Osnovna_Klasa();
        int Prosledi_Vrednost();
};

Osnovna_Klasa::Osnovna_Klasa(int i) : n(i)
{
    cout << endl;
    cout << "Izvrsen konstruktor klase Osnovna_Klasa: "
         << "Podatak clanica inicijalizovana na " << n << endl;
}

Osnovna_Klasa::Osnovna_Klasa(const Osnovna_Klasa &
                             referenca_na_objekat_osnovne_klase)
                             :n(referenca_na_objekat_osnovne_klase.n)
{
    cout << endl;
    cout << "Izvrsena Kopija konstruktora za klasu Osnovna_Klasa: "
         << "Podatak clanica inicijalizovana na " << n << endl;
}

Osnovna_Klasa::~Osnovna_Klasa()
{
    cout << endl;
    cout << "Izvrsen destruktor klase Osnovna_Klasa "
         << "za objekat sa podatkom clanicom " << n << endl;
}

int Osnovna_Klasa::Prosledi_Vrednost()
{
    return n;
```

```
}

class Izvedena_Klasa : public Osnovna_Klasa
{
    protected:
        int m;

    public:
        Izvedena_Klasa(int j = 0);
        Izvedena_Klasa(const Izvedena_Klasa &);
        ~Izvedena_Klasa();
        int Prosledi_Vrednost();
};

Izvedena_Klasa::Izvedena_Klasa(int j) : Osnovna_Klasa(j+1), m(j)
{
    cout << endl;
    cout << "Izvrsen konstruktor za izvedenu klasu Izvedena_Klasa: "
         << "Podatak clanica inicijalizovana na " << m << endl;
}

Izvedena_Klasa::Izvedena_Klasa(const Izvedena_Klasa &
                               referenca_na_objekat_izvedene_klase)
                  :Osnovna_Klasa(referenca_na_objekat_izvedene_klase),
                   m(referenca_na_objekat_izvedene_klase.m)
{
    cout << endl;
    cout << "Izvrsena Kopija konstruktora "
         << "za izvedenu klasu Izvedena_Klasa: "
         << "Podatak clanica inicijalizovana na " << m << endl;
}

Izvedena_Klasa::~Izvedena_Klasa()
{
    cout << endl;
    cout << "Izvrsen destruktor klase Izvedena_Klasa "
         << "za objekat sa podatkom clanicom " << m << endl;
}

int Izvedena_Klasa::Prosledi_Vrednost()
{
    return m;
}

int main()
{
    Osnovna_Klasa objekat_osnovne_klase(3);
    Izvedena_Klasa objekat_izvedene_klase(7);

    cout << endl << endl;
    cout << "Vrenost n u objektu klase Osnovna_Klasa je "
         << objekat_osnovne_klase.Prosledi_Vrednost() << endl << endl;

    cout << "Vrenost m u objektu klase Izvedena_Klasa je "
         << objekat_izvedene_klase.Prosledi_Vrednost()
         << endl << endl;
```

```
    cout << "Vrenost n u objektu klase Osnovna_Klasa je "
         << objekat_izvedene_klase.Osnovna_Klasa::Prosledi_Vrednost();

    cout << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Izvrsen konstruktor klase Osnovna_Klasa: Podatak clanica
inicijalizovana na 3

Izvrsen konstruktor klase Osnovna_Klasa: Podatak clanica
inicijalizovana na 8

Izvrsen konstruktor za izvedenu klasu Izvedena_Klasa: Podatak clanica
inicijalizovana na 7

Vrenost n u objektu klase Osnovna_Klasa je 3

Vrenost m u objektu klase Izvedena_Klasa je 7

Vrenost n u objektu klase Osnovna_Klasa je 8

Izvrsen destruktor klase Izvedena_Klasa za objekat sa podatkom
clanicom 7

Izvrsen destruktor klase Osnovna_Klasa za objekat sa podatkom clanicom
8

Izvrsen destruktor klase Osnovna_Klasa za objekat sa podatkom clanicom
3
```

Scope predstavlja oblast važenja neke promenljive ili neke funkcije.

Primer 84.

Napisati program u kome se definišu geometrijske figure korišćenjem izvedenih klasa.

```
// Primer 84

// Program prikzuje upotrebu poitera i metoda
// u hijerarhiji klase.

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

```
class Geometrijska_Figura
{
    protected:
        string tip_geometrijske_figure;

    public:
        Geometrijska_Figura
        (string tip_figure = "Geometrijska figura"):
        tip_geometrijske_figure(tip_figure)
        {cout << endl << "Konstruktor Geometrijske figure" << endl;}
        string Prosledi_Tip() {return tip_geometrijske_figure;}
        ~Geometrijska_Figura()
        {cout << endl << "Destruktor Geometrijske figure" << endl;}
        double Povrsina() {return 0.0;}
};

class Pravougaonik : public Geometrijska_Figura
{
    protected:
        double visina, sirina;

    public:
        Pravougaonik(double v, double s):
        Geometrijska_Figura("Pravougaonik"), visina(v),sirina(s)
        {cout << endl << "Konstruktor Pravougaonika" << endl;}
        ~Pravougaonik()
        {cout << endl << "Destruktor Pravougaonika" << endl;}
        double Povrsina() {return visina * sirina;}
};

class Krug : public Geometrijska_Figura
{
    protected:
        double poluprecnik;

    public:
        Krug(double p) : Geometrijska_Figura("Krug"), poluprecnik(p)
        {cout << endl << "Konstruktor Kruga" << endl;}
        ~Krug() {cout << endl << "Destruktor Kruga" << endl;}
        double Povrsina() {return 3.1416 * poluprecnik * poluprecnik;}
};

class Trapez : public Geometrijska_Figura
{
    protected:
        double osnova1, osnova2, visina;

    public:
        Trapez(double o1, double o2, double v):
        Geometrijska_Figura("Trapez"), osnova1(o1), osnova2(o2),
        visina(v)
        {cout << endl << "Konstruktor Trapeza" << endl;}
        ~Trapez() {cout << endl << "Destruktor Trapeza" << endl;}
        double Povrsina() {return 0.5 * visina * (osnova1 + osnova2);}
};
```

```
int main()
{
    cout << setprecision(4)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);

    Geometrijska_Figura* pokazivac_na_geometrijsku_figuru_1;
    Krug* pokazivac_na_krug_1;
    Krug krug_1(5);

    pokazivac_na_geometrijsku_figuru_1 = &krug_1;
        // Pokazivac na figuru pokazuje na krug

    pokazivac_na_krug_1 = &krug_1;
        // Pokazivac na krug pokazuje na krug

    cout << endl;
    cout << "Funkcija Povrsina() primenjena preko pointera na Krug : "
         << pokazivac_na_krug_1 -> Povrsina() << endl << endl;

    cout << "Funkcija Povrsina() primenjena "
         << "preko pointera na Geometrijska_Figura : "
         << pokazivac_na_geometrijsku_figuru_1 -> Povrsina()
         << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Konstruktor Geometrijske figure

Konstruktor kruga

Funkcija Povrsina() primenjena preko pointera na Krug : 78.5400

Funkcija Povrsina() primenjena preko pointera na Geometrijska_Figura :
0.0000

Destruktor Kruga

Destruktor Geometrijske figure
```

Primer ilustruje upotrebu osnovne i izvedenih klasa.

Primer 85.

Napisati program u kome se koristi `virtual` funkcija.

```
// Primer 85

// Program ilustruje upotrebu virtual funkcije
// u hijerarhihi klase.

#include <iostream>
#include <iomanip>
#include <string>

using namespace std;

class Geometrijska_Figura
{
    protected:
        string tip_geometrijske_figure;

    public:
        Geometrijska_Figura(string tip_figure= "Geometrijska_Figura"):
        tip_geometrijske_figure(tip_figure)
        {cout << endl << "Konstruktor Geometrijske figure" << endl;}
        string Prosledi_Tip() {return tip_geometrijske_figure;}
        ~Geometrijska_Figura()
        {cout << endl << "Destruktor Geometrijske figure" << endl;}
        virtual double Povrsina() {return 0.0;}
};

class Pravougaonik : public Geometrijska_Figura
{
    protected:
        double visina, sirina;

    public:
        Pravougaonik(double v,double s):
        Geometrijska_Figura("Pravougaonik"), visina(v),sirina(s)
        {cout << endl << "Konstruktor Pravougaonika" << endl;}
        ~Pravougaonik()
        {cout << endl << "Destruktor Pravougaonika" << endl;}
        double Povrsina() {return visina * sirina;}
};

class Krug : public Geometrijska_Figura
{
    protected:
        double poluprecnik;

    public:
        Krug(double p) : Geometrijska_Figura("Krug"), poluprecnik(p)
        {cout << endl << "Konstruktor Kruga" << endl;}
        ~Krug() {cout << endl << "Destruktor Kruga" << endl;}
        double Povrsina() {return 3.1416 * poluprecnik * poluprecnik;}
};

class Trapez : public Geometrijska_Figura
{
    protected:
        double osnova1, osnova2, visina;
```

```
    public:
        Trapez(double o1, double o2, double v):
        Geometrijska_Figura("Trapez"), osnova1(o1), osnova2(o2),
                            visina(v)
        {cout << endl << "Trapezoid Constructor" << endl;}
        ~Trapez() {cout << endl << "Trapezoid Destructor" << endl;}
        double Povrsina() {return 0.5 * visina * (osnova1 + osnova2);}
};

int main()
{
    cout << setprecision(4)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);

    Geometrijska_Figura* pokazivac_na_geometrijsku_figuru_1;
    Krug* pokazivac_na_krug_1;

    Krug krug_1(5);

    pokazivac_na_geometrijsku_figuru_1 = &krug_1;
        // Pokazivac na figuru pokazuje na krug

    pokazivac_na_krug_1 = &krug_1;
        // Pokazivac na krug pokazuje na krug

    cout << endl;
    cout << "Funkcija Povrsina() primenjena preko pointera na Krug : "
         << pokazivac_na_krug_1 -> Povrsina() << endl << endl;

    cout << "Funkcija Povrsina() primenjena "
         << "preko pointera na Geometrijska_Figura : "
         << pokazivac_na_geometrijsku_figuru_1 -> Povrsina()
         << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Konstruktor Geometrijske figure

Konstruktor kruga

Funkcija Povrsina() primenjena preko pointera na Krug : 78.5400

Funkcija Povrsina() primenjena preko pointera na Geometrijska_Figura :
78.5400

Destruktor Kruga

Destruktor Geometrijske figure
```

Ključna reč `virtual` ispred funkcije u osnovnoj klasi saopštava kompajleru da postoji odgovarajuća funkcija u izvedenoj klasi i da nju treba izvršiti prilikom poziva objekta izvedene klase. Ukoliko se ne navede ključna reč `virtual`, program će izvršiti funkciju osnovne klase, jer kompajler neće imati informaciju o postojanju odgovarajuće funkcije u izvedenoj klasi.

Primer 86.

Napisati program u kome se pokazuje nepravilna upotreba destruktora u hijerarhiji klasa.

```
// Primer 86

// Program ilustruje nepravilnu upotrebu destruktora
// u hijerarhiji klase.

#include <iostream>
#include <iomanip>
#include <string>

using namespace std;

class Geometrijska_Figura
{
    protected:
        string tip_geometrijske_figure;

    public:
        Geometrijska_Figura(string tip_figure= "Geometrijska_Figura"):
        tip_geometrijske_figure(tip_figure)
        {cout << endl << "Konstruktor Geometrijske figure" << endl;}
        string Prosledi_Tip()
        {return tip_geometrijske_figure;}
        ~Geometrijska_Figura()
        {cout << endl << "Destruktor Geometrijske figure" << endl;}
        virtual double Povrsina() {return 0.0;}
};

class Pravougaonik : public Geometrijska_Figura
{
    protected:
        double visina, sirina;

    public:
        Pravougaonik(double v, double s):
        Geometrijska_Figura("Pravougaonik"), visina(v),sirina(s)
        {cout << endl << "Konstruktor Pravougaonika" << endl;}
        ~Pravougaonik() {cout << endl << "Destruktor Pravougaonika"
                              << endl;}
        double Povrsina() {return visina * sirina;}
};
```

```
class Krug : public Geometrijska_Figura
{
    protected:
        double poluprecnik;

    public:
        Krug(double p) : Geometrijska_Figura("Krug"), poluprecnik(p)
        {cout << endl << "Konstruktor Kruga" << endl;}
        ~Krug() {cout << endl << "Destruktor Kruga" << endl;}
        double Povrsina() {return 3.1416 * poluprecnik * poluprecnik;}
};

class Trapez : public Geometrijska_Figura
{
    protected:
        double osnova1, osnova2, visina;

    public:
        Trapez(double o1, double o2, double v):
        Geometrijska_Figura("Trapez"), osnova1(o1), osnova2(o2),
                                visina(v)
        {cout << endl << "Konstruktor Trapeza" << endl;}
        ~Trapez() {cout << endl << "Destruktor Trapeza" << endl;}
        double Povrsina() {return 0.5 * visina * (osnova1 + osnova2);}
};

int main()
{
    cout << setprecision(4)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);

    Geometrijska_Figura* pokazivac_na_geometrijsku_figuru_1 =
    new Krug (5);

    cout << endl;
    cout << "Funkcija Povrsina() primenjena "
         << "preko pointera na Geometrijska_Figura : "
         << pokazivac_na_geometrijsku_figuru_1 -> Povrsina() << endl;

    delete pokazivac_na_geometrijsku_figuru_1;

    cout << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Konstruktor Geometrijske figure

Konstruktor Kruga

Funkcija Povrsina() primenjena preko pointera na Geometrijska_Figura :
78.5400

Destruktor Geometrijske figure
```

Greška se sastoji u tome što je prvo izvršen destruktor osnovne klase, dok destruktor izvedene klase nije izvršen. Kao posledica ovakve akcije, deo objekta izvedene klase ostaće zapisan u memoriji i bespotrebno će zauzimati memorijski prostor.

Primer 87.

Napisati program u kome se koristi klasa `Geometrijska_Figura` pozivanjem metoda preko pokazivača.

```
// Primer 87

// U narednom kodu dodati funkciju koja izracunava obim pravougaonika
// sa nazivom Obim().
// Pozvati funkciju Obim() za stranice pravougaonika
// visina = 4 i sirina = 5, u funkciju main()
// i ispisati rezultat - vrednost obima pravougaonika.

#include <iostream>
#include <iomanip>
#include <string>

using namespace std;

class Geometrijska_Figura
{
    protected:
        string tip_geometrijske_figure;

    public:
        Geometrijska_Figura(string tip_figure= "Geometrijska_Figura"):
        tip_geometrijske_figure(tip_figure)
        {cout << endl << "Konstruktor Geometrijske figure" << endl;}
        string Prosledi_Tip() {return tip_geometrijske_figure;}
        virtual ~Geometrijska_Figura()
        {cout << endl << "Destruktor Geometrijske figure" << endl;}
        virtual double Povrsina() {return 0.0;}
};
```

```
class Pravougaonik : public Geometrijska_Figura
{
    protected:
        double visina, sirina;

    public:
        Pravougaonik(double v, double s):
        Geometrijska_Figura("Pravougaonik"), visina(v),sirina(s)
        {cout << endl << "Konstruktor Pravougaonika" << endl;}
        virtual ~Pravougaonik()
        {cout << endl << "Destruktor Pravougaonika" << endl;}
        double Povrsina() {return visina * sirina;}
};

class Krug : public Geometrijska_Figura
{
    protected:
        double poluprecnik;

    public:
        Krug(double p) : Geometrijska_Figura("Krug"), poluprecnik(p)
        {cout << endl << "Konstruktor Kruga" << endl;}
        virtual ~Krug() {cout << endl << "Destruktor Kruga" << endl;}
        double Povrsina() {return 3.1416 * poluprecnik * poluprecnik;}
};

class Trapez : public Geometrijska_Figura
{
    protected:
        double osnova1, osnova2, visina;
    public:
        Trapez(double o1, double o2, double v):
         Geometrijska_Figura("Trapez"), osnova1(o1), osnova2(o2),
                                visina(v)
        {cout << endl << "Konstruktor Trapeza" << endl;}
        virtual ~Trapez() {cout << endl << "Destruktor Trapeza"
                                << endl;}
        double Povrsina() {return 0.5 * visina * (osnova1 + osnova2);}
};

int main()
{
    cout << setprecision(4)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);

    Geometrijska_Figura* pokazivac_na_geometrijsku_figuru_1 = new Krug
(5);

    cout << endl;

    cout << "Povrsina() za "
         << pokazivac_na_geometrijsku_figuru_1 -> Prosledi_Tip()
         << " primenjena preko pointera na Geometrijska_Figura : "
         << pokazivac_na_geometrijsku_figuru_1 -> Povrsina() << endl;
```

```
    delete pokazivac_na_geometrijsku_figuru_1;

    cout << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Konstruktor Geometrijske figure

Konstruktor Kruga

Povrsina() za Krug primenjena preko pointera na Geometrijska_Figura :
78.5400

Destruktor Kruga

Destruktor Geometrijske figure
```

U primeru se demonstrira pozivanje metoda korisnički kreirane klase primenom pokazivača.

Primer 88.

Napisati program u kome će se na primeru klase `Geometrijska_Figura` definisati funkcija `Povrsina()` radi ilustracije svojstva polimorfizma.

```
// Primer 88

// Program prikazuje upotrebu polimorfizma.

#include <iostream>
#include <iomanip>
#include <string>

using namespace std;

class Geometrijska_Figura
{
    protected:
        string tip_geometrijske_figure;

    public:
        Geometrijska_Figura(string tip_figure= "Geometrijska_Figura"):
        tip_geometrijske_figure(tip_figure) {}
        string Prosledi_Tip() {return tip_geometrijske_figure;}
        virtual ~Geometrijska_Figura() {}
        virtual double Povrsina() {return 0.0;}
```

```
};

class Pravougaonik : public Geometrijska_Figura
{
    protected:
        double visina, sirina;

    public:
        Pravougaonik(double v, double s):
        Geometrijska_Figura("Pravougaonik"), visina(v),sirina(s) {}
        virtual ~Pravougaonik() {}
        double Povrsina() {return visina * sirina;}
};

class Krug : public Geometrijska_Figura
{
    protected:
        double poluprecnik;

    public:
        Krug(double p) :
        Geometrijska_Figura("Krug"), poluprecnik(p) {}
        virtual ~Krug() {}
        double Povrsina() {return 3.1416 * poluprecnik * poluprecnik;}
};

class Trapez : public Geometrijska_Figura
{
    protected:
        double osnova1, osnova2, visina;
    public:
        Trapez(double o1, double o2, double v):
        Geometrijska_Figura("Trapez"), osnova1(o1), osnova2(o2),
                            visina(v) {}
        virtual ~Trapez() {}
        double Povrsina() {return 0.5 * visina * (osnova1 + osnova2);}
};

Geometrijska_Figura* Prosledi_Tip();

int main()
{
    cout << setprecision(4)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);

    const int broj_geometrijskih_figura = 5;

    Geometrijska_Figura* figura[broj_geometrijskih_figura];

    for (int i = 0; i < broj_geometrijskih_figura; ++i)
        figura[i] = Prosledi_Tip();

    cout << endl;

    for (i = 0; i < broj_geometrijskih_figura; ++i)
```

```
        cout << "\nPovrsina " << figura[i] -> Prosledi_Tip() << " je "
             << figura[i] -> Povrsina();

    cout << endl;

    for (i = 0; i < broj_geometrijskih_figura; ++i)
        delete figura[i];

    cout << endl;

    return 0;
}

Geometrijska_Figura* Prosledi_Tip()
{
    Geometrijska_Figura* pokazivac_na_geometrijsku_figuru;
        // Vraca se u funkciju

    double visina,
           sirina,
           poluprecnik,
           osnova1,
           osnova2;

    cout << endl;
    cout << "Unesite broj figure koju zelite da kreirate :"
         << endl << endl;
    cout << "Za pravougaonik unesite 1" << endl;
    cout << "Za krug unesite 2" << endl;
    cout << "Za trapez unesite 3" << endl << endl;
    cout << "Biram : ";

    int izbor;

    cin >> izbor;

    switch (izbor)
    {
        case 1:
            cout << endl << endl;
            cout << "Unesite visinu pravougaonika : ";
            cin >> visina;
            cout << endl << endl;
            cout << "Unesite sirinu pravougaonika : ";
            cin >> sirina;

            pokazivac_na_geometrijsku_figuru =
            new Pravougaonik(visina, sirina);
        break;

        case 2:
            cout << endl << endl;
            cout << "Unesite poluprecnik kruga : ";
            cin >> poluprecnik;

            pokazivac_na_geometrijsku_figuru =
```

```
            new Krug(poluprecnik);
        break;

        case 3:
            cout << endl << endl;
            cout << "Unesite prvu osnovu trapeza : ";
            cin >> osnova1;
            cout << endl << endl;
            cout << "Unesite drugu osnovu trapeza : ";
            cin >> osnova2;
            cout << endl << endl;
            cout << "Unesite visinu trapeza : ";
            cin >> visina;

            pokazivac_na_geometrijsku_figuru =
            new Trapez(osnova1, osnova2, visina);
        break;

        default:
            cout << endl << endl;
            cout << "Pogresan izbor. Kreiranje default objekta.";

            pokazivac_na_geometrijsku_figuru =
            new Geometrijska_Figura();
        break;
  }

    return pokazivac_na_geometrijsku_figuru;
}
```

Rezultat izvršavanja programa:

```
Unesite broj figure koju zelite da kreirate :

Za pravougaonik unesite 1
Za krug unesite 2
Za trapez unesite 3

Biram :  1

Unesite visinu pravougaonika : 3

Unesite sirinu pravougaonika : 5

Unesite broj figure koju zelite da kreirate :

Za pravougaonik unesite 1
Za krug unesite 2
Za trapez unesite 3

Biram : 2

Unesite poluprecnik kruga : 5.6
```

```
Unesite broj figure koju zelite da kreirate :

Za pravougaonik unesite 1
Za krug unesite 2
Za trapez unesite 3

Biram : 3

Unesite prvu osnovu trapeza : 7

Unesite drugu osnovu trapeza : 8

Unesite visinu trapeza : 9

Unesite broj figure koju zelite da kreirate :

Za pravougaonik unesite 1
Za krug unesite 2
Za trapez unesite 3

Biram : 5

Pogresan izbor. Kreiranje default objekta.

Unesite broj figure koju zelite da kreirate :

Za pravougaonik unesite 1
Za krug unesite 2
Za trapez unesite 3

Biram : 1

Unesite visinu pravougaonika : 1

Unesite sirinu pravougaonika : 1

Povrsina Pravougaonik is 15.0000
Povrsina Krug is 98.5206
Povrsina Trapez is 67.5000
Povrsina Geometrijska_Figura je 0.0000
Povrsina Pravougaonik je 1.0000
```

Svojstvo da svaki objekat izvedene klase, čak i kada mu se pristupa kao objektu osnovne klase, izvršava metod tačno onako kako je to definisano u njegovoj izvedenoj klasi, naziva se polimorfizam.

Polimorfizam predstavlja različite odzive objekata različitih izvedenih klasa pozivom iste metode. Prilikom poziva funkcije `Povrsina()` svaki objekat različite izvedene klase, daje tačnu vrednost prema tipu geometrijske figure, koju izvedena klasa definiše.

Primer 89.

Napisati program koji izračunava mesečnu ratu kredita korišćenjem korisnički definisane klase `Kredit`.

```
// Primer 89

#include <iostream>
#include <iomanip>
#include <string>
#include <cmath>

using namespace std;

class Kredit
{
    protected:
        double osnovica;
        double godisnja_kamata;   // u procentima
        int    broj_godina;
        double mesecna_rata;
        string tip_kredita;

    public:
        Kredit (double osnova, double kamata, int rok, const char *
                tip):
        osnovica(osnova), godisnja_kamata(kamata),broj_godina(rok),
        tip_kredita(tip), mesecna_rata(0.0) {};
        virtual ~Kredit() {}
        virtual void Izracunavanje_Rate() = 0;   // virtuelna funkcija
        void Prikazi_Informacije_O_Kreditu();
};

void Kredit::Prikazi_Informacije_O_Kreditu()
{
    cout << endl << endl;
    cout << "Tip kredita : " << tip_kredita << endl;
    cout << "Osnovica : " << osnovica << endl;
    cout << "Godisnja kamata u % : " << godisnja_kamata <<endl;
    cout << "Broj godina za otplatu : " << broj_godina
         << endl << endl;
    cout << "Mesecna rata : " << mesecna_rata << endl;
}
// Kraj klase Kredit

class Obican_Kredit : public Kredit
{
    public:
        Obican_Kredit(double osnovica, double kamata, int rok):
        Kredit(osnovica, kamata, rok, "Obican kredit") {}
        ~Obican_Kredit() {}
        void Izracunavanje_Rate();
};

void Obican_Kredit::Izracunavanje_Rate()
```

```
{
    double kamata = osnovica * broj_godina * (godisnja_kamata / 100);
    double iznos_kredita = osnovica + kamata;

    mesecna_rata = iznos_kredita / (broj_godina * 12);
}
// Kraj klase Obican_Kredit

class Amortizovani_Kredit : public Kredit
{
    public:
        Amortizovani_Kredit(double osnovica, double kamata, int rok):
        Kredit(osnovica, kamata, rok, "Amortizovani kredit") {}
        ~Amortizovani_Kredit() {}
        void Izracunavanje_Rate();
};

void Amortizovani_Kredit::Izracunavanje_Rate()
{
    double mesecna_kamata = godisnja_kamata / 12 / 100;
    int rok_otplate_u_mesecima = broj_godina * 12;

    double stepen = pow(1 + mesecna_kamata, rok_otplate_u_mesecima);

    mesecna_rata = (osnovica * mesecna_kamata * stepen) / (stepen -
1);
}
// Kraj klase Amortizovani_Kredi

Kredit* Prosledi_Podatke_O_Kreditu();
    // Prototip funkcije koju koristi main()

int main()
{
    cout << setprecision(2)
         << setiosflags(ios::fixed)
         << setiosflags(ios::showpoint);

    const int broj_kredita = 3;

    Kredit* kredit[broj_kredita];

    for (int i = 0; i < broj_kredita; ++i)
        kredit[i] = Prosledi_Podatke_O_Kreditu();

    for (i = 0; i < broj_kredita; ++i)
        kredit[i] -> Izracunavanje_Rate();

    for (i = 0; i < broj_kredita; ++i)
        kredit[i] -> Prikazi_Informacije_O_Kreditu();

    for (i = 0; i < broj_kredita; ++i)
        delete kredit[i];

    cout << endl;
```

```
    return 0;
}
// Kraj funkcije main()

Kredit* Prosledi_Podatke_O_Kreditu()
{
    double osnovica;
    double kamata;
    int rok_u_godinama;

    Kredit* pokazivac_na_kredit;

    cout << endl << endl;
    cout << "Unesite prvo slovo tipa kredita." << endl << endl;
    cout << "(O)bican" << endl;
    cout << "(A)mortizovani" << endl << endl;
    cout << "Biram : ";
    char odgovor = cin.get();

    cout << endl << endl;
    cout << "Osnovica : ";
    cin >> osnovica;
    cout << endl;
    cout << "Godisnja kamata u % : ";
    cin >> kamata;
    cout << endl;
    cout << "Rok otplate u godinama : ";
    cin >> rok_u_godinama;

    cin.get();   // brisanje ulaznog buffer-a

    switch (odgovor)
    {
        case 'O': case 'o':
            pokazivac_na_kredit =
            new Obican_Kredit(osnovica, kamata, rok_u_godinama);
        break;

        case 'A': case 'a':
            pokazivac_na_kredit =
            new Amortizovani_Kredit(osnovica, kamata,
            rok_u_godinama);
        break;
    }

    return pokazivac_na_kredit;
}
// Kraj funkcije Prosledi_Podatke_O_Kreditu()
```

Rezultat izvršavanja programa:

```
Unesite prvo slovo tipa kredita.

(O)bican
```

```
(A)mortizovani

Biram : o

Osnovica : 1000

Godisnja kamata u % : 12

Rok otplate u godinama : 3

Unesite prvo slovo tipa kredita.

(O)bican
(A)mortizovani

Biram : A

Osnovica : 1000

Godisnja kamata u % : 12

Rok otplate u godinama : 3

Unesite prvo slovo tipa kredita.

(O)bican
(A)mortizovani

Biram : A

Osnovica : 150000

Godisnja kamata u % : 8.5

Rok otplate u godinama : 30

Tip kredita : Obican kredit
Osnovica : 1000.00
Godisnja kamata u % : 12.00
Broj godina za otplatu : 3

Mesecna rata : 37.78

Tip kredita : Amortizovani kredit
Osnovica : 1000.00
Godisnja kamata u % : 12.00
Broj godina za otplatu : 3

Mesecna rata : 33.21

Tip kredita : Amortizovani kredit
Osnovica : 150000.00
Godisnja kamata u % : 8.50
Broj godina za otplatu : 30

Mesecna rata : 1153.37
```

U primeru je kreirana korisnička klasa sa odgovarajućim izvedenim klasama, koja može imati prktičnu primenu u bankarstvu.

XIV Rad sa fajlovima

Primer 90.

Napisati program koji upisuje podatke u fajl.

```
// Primer 90

// Program prikazuje upis podataka u fajl

#include <fstream>
#include <iostream>

using namespace std;

int main()
{
    char buffer[81];

    ofstream izlazni_fajl("izlazni_fajl.dat");

    cout << "Unesite svoje ime : ";
    cin.getline(buffer, 81);

    izlazni_fajl << buffer << endl;

    cout << endl;
    cout << "Unesite svoju adresu : ";
    cin.getline(buffer, 81);
    izlazni_fajl << buffer << endl;

    izlazni_fajl.close();

    cout << endl;
    cout << "Program zavrsen" << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite svoje ime : Derek Trotter

Unesite svoju adresu : Pekam, London

Program zavrsen
```

Sadržaj fajla izlazni_fajl.dat:

```
Derek Trotter
Pekam, London
```

Program demonstrira upisivanje podataka unetih sa konzole u fajl.

Primer 91.

Napisati program koji upisuje podatke u fajl i proverava da li postoji greška prilikom otvaranja fajla.

```
// Primer 91

// Program demonstrira upis podataka u fajl
// Proverava da li postoji greska prilikom otvaranju fajla

#include <fstream>
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
    char buffer[81];
    char ime_fajla[81];

    cout << "Unesite ime fajla u koji zelite da upisete podatke : ";
    cin.getline(ime_fajla, 81);
    ofstream izlazni_fajl(ime_fajla);

    if (!izlazni_fajl)
    {
        cerr << endl;
        cerr << "GRESKA: Fajl ne moze biti otvoren." << endl;
        cerr << "Program zavrsen" << endl << endl;
        exit(1);
    }

    cout << endl;
    cout << "Unesite svoje ime : ";
    cin.getline(buffer, 81);

    izlazni_fajl << buffer << endl;

    cout << endl;
    cout << "Unesite svoju adresu : ";
    cin.getline(buffer, 81);

    izlazni_fajl << buffer << endl;
```

```
    izlazni_fajl.close();

    cout << endl;
    cout << "Program zavrsen" << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

U slučaju ispravnog otvaranja fajla:

```
Unesite ime fajla u koji zelite da upisete podatke : izlaz.dat

Unesite svoje ime : Derek Trotter

Unesite svoju adresu : Pekam, London

Program zavrsen
```

Sadržaj fajla izlaz.dat:

```
Derek Trotter
Pekam, London
```

U slučaju neispravnog otvaranja fajla:

```
Unesite ime fajla u koji zelite da upisete podatke : m:izlaz.dat

GRESKA: Fajl ne moze biti otvoren.
Program zavrsen
```

U slučaju greške fajl nije formiran.

U pokazanom primeru vrši se logička kontrola da li fajl može biti formiran/otvoren.

Primer 92.

Napisati program koji čita podatke iz fajla.

```
// Primer 92

// Program prikazuje ucitavanje podataka iz fajla

#include <fstream>
```

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
    char buffer[81];
    char ime_fajla[81];

    cout << "Unesite ime fajla koji zelite da prikazete : ";
    cin.getline(ime_fajla, 81);
    ifstream ulazni_fajl(ime_fajla);

    if (!ulazni_fajl)
    {
        cerr << endl;
        cerr << "GRESKA: Fajl ne moze biti otvoren." << endl;
        cerr << "Program zavrsen" << endl << endl;
        exit(1);
    }

    cout << endl;

    while (!ulazni_fajl.eof())
    {
        ulazni_fajl.getline(buffer, 81);
        cout << buffer << endl;
    }

    ulazni_fajl.close();

    cout << "Program zavrsen" << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite ime fajla koji zelite da prikazete : izlaz.dat

Derek Trotter
Pekam, London

Program zavrsen
```

Program demonstrira čitanje podataka iz fajla kreiranog u prethodnom primeru kao i ispisivanje pročitanih podataka na ekranu.

Primer 93.

Napisati program koji upisuje podatke u fajl i čita podatke iz fajla.

```
// Primer 93

// Program ilustruje upis podataka u fajl kao i citanje iz fajla.
// Fajl je otvoren u ios::in i ios::out modu.

#include <fstream>
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
    char buffer[81];
    char ime_fajla[81];

    cout << "Unesite ime fajla u koji zelite da upisete podatke : ";
    cin.getline(ime_fajla, 81);
    fstream fajl(ime_fajla, ios::out);

    if (!fajl)
    {
        cerr << endl;
        cerr << "GRESKA: Fajl ne moze biti otvoren." << endl;
        cerr << "Program zavrsen" << endl;
        exit(1);
    }

    cout << endl;
    cout << "Unesite svoje ime : ";
    cin.getline(buffer, 81);

    fajl << buffer << endl;

    cout << endl;
    cout << "Unesite svoju adresu : ";
    cin.getline(buffer, 81);

    fajl << buffer << endl;

    fajl.close();

    cout << endl;
    cout << "Sadrzaj fajla koji ste kreirali je :" << endl << endl;
    fajl.open(ime_fajla, ios::in);

    if (!fajl)
    {
        cerr << endl;
        cerr << "GRESKA: Fajl ne moze biti otvoren." << endl;
        cerr << "Program zavrsen" << endl << endl;
```

```
        exit(1);
    }

    while (!fajl.eof())
    {
        fajl.getline(buffer, 81);
        cout << buffer << endl;
    }

    fajl.close();

    cout << "Program zavrsen" << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite ime fajla u koji zelite da upisete podatke : test.abc

Unesite svoje ime : Mr. Bean

Unesite svoju adresu :  Belgrade street 18

Sadrzaj fajla koji ste kreirali je :

Mr. Bean
Belgrade street 18

Program zavrsen
```

Program predstavlja kombinaciju prethodna tri primera.

Primer 94.

Napisati program koji upisuje podatke u fajl, čita podatke iz fajla i dodaje podatke u fajl.

```
// Primer 94

// Program ilustruje upis podataka u fajl kao i citanje iz fajla.
// Fajl je otvoren u modu za upis, citanje i dodavanje podataka.

#include <fstream>
#include <iostream>
#include <cstdlib>

using namespace std;
```

```
int main()
{
    char buffer[81];
    char ime_fajla[81];

    cout << "Unesite ime fajla u koji zelite da upisete podatke : ";
    cin.getline(ime_fajla, 81);

    cout << endl;
    cout << "Unesite grad i drzavu u kojoj zivite : ";
    cin.getline(buffer, 81);

    cout << endl;
    cout << "Podaci ce biti dodati u Vas fajl." << endl;

    ofstream mod_za_dodavanje(ime_fajla, ios::app);

    if (!mod_za_dodavanje)
    {
        cerr << endl;
        cerr << "GRESKA: Fajl ne moze biti otvoren "
             << "u modu za dodavanje podataka." << endl;
        cerr << "Program zavrsen" << endl << endl;
        exit(1);
    }

    mod_za_dodavanje << buffer << endl;

    mod_za_dodavanje.close();

    cout << endl;
    cout << "Vase puno ime i adresa su :" << endl << endl;

    ifstream mod_za_upisivanje_i_citanje(ime_fajla);

    if (!mod_za_upisivanje_i_citanje)
    {
        cerr << endl;
        cerr << "GRESKA: Fajl ne moze biti otvoren." << endl;
        cerr << "Program zavrsen" << endl << endl;
        exit(1);
    }

    while (!mod_za_upisivanje_i_citanje.eof())
    {
        mod_za_upisivanje_i_citanje.getline(buffer, 81);
        cout << buffer << endl;
    }

    mod_za_upisivanje_i_citanje.close();
    cout << "Program zavrsen" << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite ime fajla u koji zelite da upisete podatke : test.abc

Unesite grad i drzavu u kojoj zivite : London, England

Podaci ce biti dodati u Vas fajl.

Vase puno ime i adresa su :

Mr. Bean
Belgrade street 18
London, England

Program zavrsen
```

Razlika u odnosu na prethodni primer je dodavanje podataka fajl iz prethodnog primera. Podaci se dodaju na kraju fajla.

Primer 95.

Napisati program koji kopira jedan fajl u drugi koristeći get() and put() naredbe.

```
// Primer 95

// Program kopira jedan fajl u drugi koristeci get() and put()

#include <fstream>
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
    char ime_fajla[81];
    char znak;

    cout << "Unesite ime izvornog fajla : ";
    cin.getline(ime_fajla, 81);
    ifstream izvorni_fajl(ime_fajla);

    if (!izvorni_fajl)
    {
        cout << endl;
        cout << "GRESKA: Fajl ne moze biti otvoren." << endl;
        exit(1);
    }

    cout << endl;
```

```
    cout << "Unesite ime ciljnog fajla : ";
    cin.getline(ime_fajla, 81);

    ofstream ciljni_fajl(ime_fajla);

    if (!ciljni_fajl)
    {
        cout << endl;
        cout << "GRESKA: Fajl ne moze biti otvoren." << endl;
        exit(1);
    }

    while ( (znak = izvorni_fajl.get()) != EOF )
        ciljni_fajl.put(znak);

    izvorni_fajl.close();
    ciljni_fajl.close();

    cout << endl;
    cout << "Program zavrsen." << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite ime izvornog fajla : test.abc

Unesite ime ciljnog fajla : test.cba

Program zavrsen.
```

Napomena:
Program ne prikazuje podatke izlaznog fajla na ekranu.
Sledi listing fajla test.cba:

```
Mr. Bean
Belgrade street 18
London, England
```

Program kopira sadržaj fajla iz prethodnog primera u novi fajl i to znak po znak.

Primer 96.

Napisati program koji štampa sadržaj fajla sa dvostrukim razmakom na printeru.

```
// Primer 96

// Program stampa sadrzaj fajla sa dvostrukim razmakom na printeru

#include <fstream>
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
    char ime_fajla[81];
    char znak;

    cout << "Unesite ime izvornog fajla : ";
    cin.getline(ime_fajla, 81);

    ifstream izvorni_fajl(ime_fajla);

    if (!izvorni_fajl)
    {
        cout << endl;
        cout << "GRESKA: Fajl ne moze biti otvoren.";
        exit(1);
    }

    ofstream fajl_za_printer("prn");

    if (!fajl_za_printer)
    {
        cout << endl;
        cout << "GRESKA: Printer nije pronadjen." << endl << endl;
        exit(1);
    }

    fajl_za_printer << "Sadrzaj izvornog fajla "
                    << "sa dvostrukim razmakom."
                    << endl << endl;

    while ((znak = izvorni_fajl.get()) != EOF)
    {
        fajl_za_printer.put(znak);

        if (znak == '\n')
            fajl_za_printer.put(znak);
    }

    izvorni_fajl.close();
    fajl_za_printer.close();

    cout << endl << endl;
    cout << "Program zavrsen." << endl << endl;

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite ime izvornog fajla : test.abc

Program zavrsen.
```

Izlaz na printeru

```
Sadrzaj izvornog fajla sa dvostrukim razmakom.

Mr. Bean

Belgrade street 18

London, England
```

Za testiranje ovog primera neophodno je imati konektovan štampač na računaru.

Primer 97.

Napisati program koji prevodi postojeći tekst u fajlu iz malih u velika slova.

```
// Primer 97

#include <iostream>
#include <fstream>
#include <cstdlib>
#include <cctype>
#include <iomanip>

using namespace std;

int main()
{
    char ime_fajla[51];
    char znak;

    cout << "Unesite ime fajla koji zelite da prevedete "
         << "u velika slova : ";
    cin.getline(ime_fajla, 51);

    fstream fajl(ime_fajla, ios::in | ios::out | ios::binary);

    if (!fajl)
    {
        cout << endl;
        cout << "GRESKA: Fajl ne moze biti otvoren.";
```

```
        exit(1);
    }

    while ( (znak = fajl.get()) != EOF)
    {
        znak = toupper(znak);
        fajl.seekg(-1L, ios::cur);
        fajl.put(znak);
        cout << fajl.tellg() << setw(6) << znak << endl;
    }

    cout << endl;
    cout << "Program zavrsen." << endl;
    cout << endl;

    fajl.close();

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite ime fajla koji zelite da prevedete u velika slova : test.ttt
1     A
2     B
3     C
4
5     D
6     E
7
8

9     F
10     G

Program zavrsen.
```

Izlaz u fajl:

```
ABC DE
FG
```

Program čita sadržaj zadatog fajla, znak po znak. Prevodi svaki znak u veliko slovo i upisuje ga u izlazni fajl u istom rasporedu kao u izvornom fajlu. Fajl iz koga se čita sadržaj neophodno je prethodno kreirati.

Primer 98.

Napisati program koji kreira sekvencijalni fajl.

```
// Primer 98

// Program kreira sekvencijalni fajl od strukture iz tekstualnog
// fajla sa podacima. Koristi znak ">>" za citanje podataka
// iz tekstualnog fajla
// i koristi write() metodu za upis strukture u izlazni fajl.

#include <iostream>
#include <fstream>
#include <cstdlib>

using namespace std;

struct Sekvencijalni_Zapis
{
    char   identifikator[3];
    int    kolicina;
    double cena;
};

void Procitaj_Sekvencijalni_Zapis(ifstream&, Sekvencijalni_Zapis&);
int main()
{
    int brojac_zapisa = 0;

    char ime_ulaznog_fajla[51];
    char ime_izlaznog_fajla[51];

    Sekvencijalni_Zapis sekvenca;

    cout << "Unesite ime tekstualnog fajla : ";
    cin.getline(ime_ulaznog_fajla, 51);

    ifstream ulazni_fajl(ime_ulaznog_fajla);

    if (!ulazni_fajl)
    {
        cout << endl;
        cout << "GRESKA: Tekstualni fajl ne moze biti otvoren.";
        exit(1);
    }

    cout << endl;
    cout << "Unesite ime sekvencijalnog fajla : ";
    cin.getline(ime_izlaznog_fajla, 51);

    ofstream izlazni_fajl(ime_izlaznog_fajla, ios::binary);

    if (!izlazni_fajl)
    {
        cout << endl;
```

```
        cout << "GRESKA: Sekvencijalni fajl ne moze biti otvoren. ";
        exit(1);
    }

    // Citanje prvog zapisa

    Procitaj_Sekvencijalni_Zapis(ulazni_fajl, sekvenca);

    while (!ulazni_fajl.eof())
    {
        izlazni_fajl.write( (char *)&sekvenca,
                            sizeof(Sekvencijalni_Zapis) );
        ++brojac_zapisa;

        Procitaj_Sekvencijalni_Zapis(ulazni_fajl, sekvenca);
    }

    cout << endl << brojac_zapisa << " zapisa upisano.";

    ulazni_fajl.close();
    izlazni_fajl.close();

    cout << endl << endl;

    return 0;
}

void Procitaj_Sekvencijalni_Zapis(ifstream& input_strim,
Sekvencijalni_Zapis& deo)
{
    input_strim >> deo.identifikator;
    input_strim >> deo.kolicina;
    input_strim >> deo.cena;
}
```

Rezultat izvršavanja programa:

```
Unesite ime tekstualnog fajla : ulaz.abc

Unesite ime sekvencijalnog fajla : izlaz.abc

9 zapisa upisano.
```

Ulazni fajl: ulaz.abc

```
23 145   24.95
15 46    67.99
65 230   11.95
56 37    73.95
59 1074   3.95
12 26   104.99
84 19    55.95
03 81    18.95
44 39    31.95
```

Sekvencijalni fajl predstavlja jednostavnu bazu podataka, u kojoj se svaki zapis tretira odvojeno.

Primer 99.

Napisati program koji čita sekvencijalni fajl.

```
// Primer 99

// Program pokazuje kako se cita sekvencijalni fajl sa strukturama.
// Koristi funkciju claicu read() za citanje zapisa iz fajla
// i koristi znak ">>"  za stampnje na disk u formi
// tekst fajla koji je pogodan za kasnije stampanje.
// Program pretpostavlja da je ulazni fajl
// fajl kreiran u prethodnom primeru.

#include <iostream>
#include <iomanip>
#include <fstream>
#include <cstdlib>

using namespace std;

struct Sekvencijalni_Zapis
{
    char identifikator[3];
    int kolicina;
    double cena;
};

int main()
{
    Sekvencijalni_Zapis sekvenca;

    char ime_ulaznog_fajla[51];
    char ime_izlaznog_fajla[51];
    int brojac_zapisa = 0;

    cout << "Unesite ime fajla za stampanje : ";
    cin.getline(ime_izlaznog_fajla, 51);
```

```
    cout << endl;

    ofstream izlazni_fajl(ime_izlaznog_fajla);

    if (!izlazni_fajl)
    {
        cout << "GRESKA: Fajl za stampanje ne moze biti otvoren.";
        exit(1);
    }

    izlazni_fajl << setw(15) << "Sifra dela" << setw(15) << "Kolicina"
                 << setw(15) << "Cena" << endl << endl;

    cout << "Unesite ime sekvencijalnog fajla : ";
    cin.getline(ime_ulaznog_fajla, 51);

    ifstream ulazni_fajl(ime_ulaznog_fajla, ios::binary);

    if (!ulazni_fajl)
    {
        cout << "GRESKA: Sekvencijalni fajl ne moze biti otvoren.";
        exit(1);
    }

    ulazni_fajl.read( (char*) &sekvenca,sizeof(Sekvencijalni_Zapis) );

    while (!ulazni_fajl.eof())
    {
        ++brojac_zapisa;
        izlazni_fajl << setw(15) << sekvenca.identifikator
                     << setw(15) << sekvenca.kolicina
                     << setw(15) << sekvenca.cena << endl;

        ulazni_fajl.read( (char *) &sekvenca,
                          sizeof(Sekvencijalni_Zapis) );
    }

    cout << endl << brojac_zapisa << "  zapisa procitano.";
    cout << endl << endl;

    ulazni_fajl.close();
    izlazni_fajl.close();

    return 0;
}
```

Rezultat izvršavanja programa:

```
Unesite ime fajla za stampanje : izlaz.hhh

Unesite ime sekvencijalnog fajla : izlaz.abc

9 zapisa procitano.
```

Izlazni fajl: izlaz.hhh

```
 Sifra dela       Kolicina           Cena

         23            145          24.95
         15             46          67.99
         65            230          11.95
         56             37          73.95
         59           1074           3.95
         12             26         104.99
         84             19          55.95
         03             81          18.95
         44             39          31.95
```

Program prvo zahteva unos fajla za štampanje i otvara ga. Zatim otvara sekvencijalni fajl iz pethodnog zadatka i čita zapise koji se u njemu nalaze. Na ovaj način smo se približili realnoj bazi podataka, obzirom da pored samih zapisa kreiramo i zaglavlje sa opisom polja. Koristi se sekvencijalni fajl kreiran u prethodnom primeru.

Primer 100.

Napisati program koji kreira fajl sa praznim zapisima.

```
// Primer 100

// Program kreira fajl sa praznim zapisima.

#include <fstream>
#include <iostream>

using namespace std;

struct Sekvencijalni_Zapis
{
    char   identifikator[3];
    int    kolicina;
    double cena;
};

int main()
{
    fstream sekvencijalni_fajl("sekvencijalnifajl.dat",
                               ios::out | ios::binary);
    Sekvencijalni_Zapis null_part = {" ", 0, 0.0};

    for (long broj_zapisa = 1L; broj_zapisa <= 99L; ++broj_zapisa)
        sekvencijalni_fajl.write( (char *) &null_part,
                                  sizeof(Sekvencijalni_Zapis) );

    cout << endl;
    cout << "Kreiran je Null fajl." << endl << endl;
```

```
    // Zatvaranje fajla
    sekvencijalni_fajl.close();
    return 0;
}   // Kraj main() funkcije
```

Rezultat izvršavanja programa:

```
Kreiran je Null fajl.
```

Program kreira fajl sa praznim zapisima u koje se kasnije mogu upisivati podaci. Pod praznim zapisima se podrazumeva da se na mesto brojanih veličina upisuju nule a na mesto tekstualnih veličina upisuju ‘blank’ karakteri.

Literatura:

1. C++ for Business Programming, John C. Molluzzo, Pace University, New York, Prentice Hall, 2005

2. Accelerated C++, Practical Programming by Example, Andrew Koenig and Barbara E. Moo, New York, Addison-Wesley, 2000

3. C++ Common Knowledge: Essential Intermediate Programming, Stephen C. Dewhurst, New York, Addison-Wesley, 2005

4. Essential C++, Stanley B. Lipman, New York, Addison-Wesley, 2002

5. C/C++ Programmer's Reference, Herbert Schildt, New York, McGraw-Hill/Osborne 2003

6. Practical C++ Programming, Steve Oualline, Sebastopol, CA, O'Reilly, 2002

7. http://www.microsoft.com/en/us/default.aspx

8. http://www.learncpp.com/

9. http://www.cplusplus.com/doc/tutorial/