ZBIRKA URAĐENIH ZADATAKA

KONTROLA ULAZA / IZLAZA I KONVERZIJA TIPOVA

Korišćenje direktive - Namespaces

Biblioteka **standard library** je proširenи set rutina koje su napisane da bi mogli izvršavati različite zadatke: naprimer, rad sa input i output sistemima, izvršavanje osnovnih matematičkih operacija, itd. Umesto da sami pišemo te rutine, jednostavno ih možemo pobrati iz standardnih biblioteka I koristiti ih u našem kodu.. Fajl iostream je samo jedan deo header files koji sadrži rutine standardne biblioteke. (Možemo videti ceo set fajlova standardne biblioteke u MSDN online help fajlovima u **Visual C++ Documentation \ Reference \ C/C++ Language and C++ Libraries \ Standard C++ Library Reference**.)

Kodovi svih rutina standardnih biblioteka sadržani su u naredbi **namespace** std. Tasko, svaka standardna rutina pripada opciji namespace std. Svaki header file standardne biblioteke prilaže nekoliko rutina pozivom namespace std.

Kod našeg programa ne pripada funkciji namespace std. Prema tome, prilikom korišćenja izlaznih rutina, (naprimer iostream header fajlova), moramo reći računaru da ove rutine pripadaju direktivi namespace std. Koristićemo:

using namespace std;

Ovom linijom u našem programu,kompajler zna da ćemo koristiti rutine koje pripadaju standardnoj biblioteci.

Međutim, u jeziku C++ možemo normalno raditi i sa već poznatim direktivama za korišćenje ulaza/izlaza

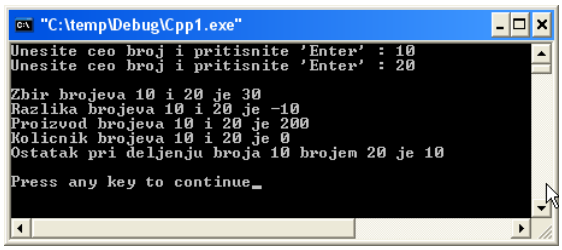
#include<iostream.h>

1 Zadatak:

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.

```
// 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;
}
```



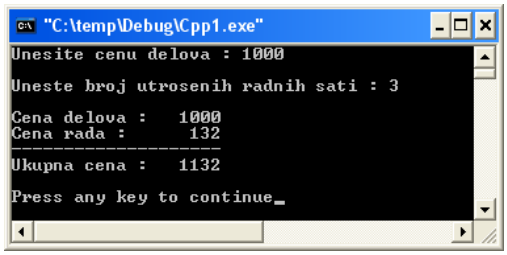
2 Zadatak:

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.

```
// Program izracunava ukupnu cenu delova i rada
#include <iostream.h>
#include <iomanip.h>
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;
}
```

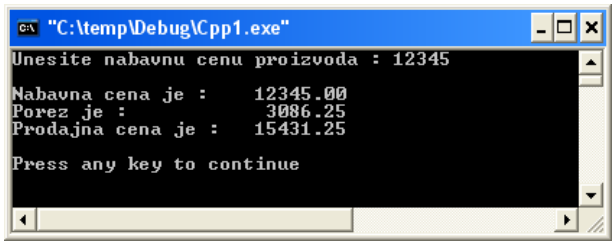


3 Zadatak:

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

```
// Program izracunava porez i maloprodajnu cenu proizvoda
#include <iostream.h>
#include <iomanip.h>
int main()
{
// Deklarisanje promenljivih
const double STOPA_POREZA = 0.25;
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)     // definiše broj decimalnih mesta
<< setiosflags(ios::fixed)  //govori da će ispis biti u fiksnom obliku
<< setiosflags(ios::showpoint);  /*‘traži’ od kompajlera da
                                    upotrebi decimalnu tačku da bi razdvojio celobrojni deo
od decimalnog*/
// 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;
}
```

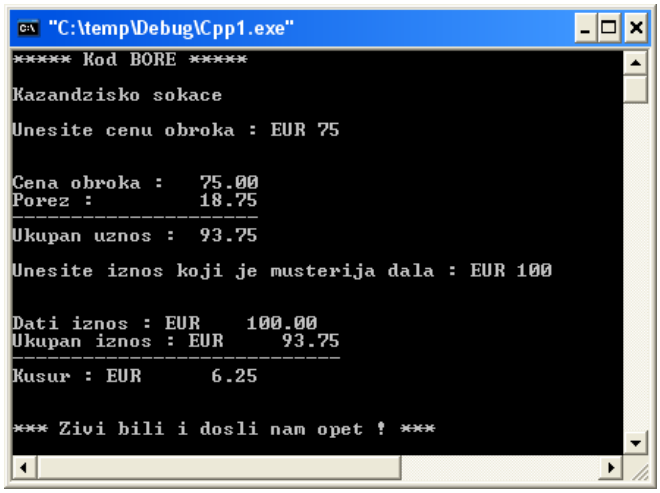


4 Zadatak:

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.

```
// Program za kasu u restoranu
#include <iostream.h>
#include <iomanip.h>
int main()
{
// Deklarisanje promenljivih
const double STOPA_MALOPRODAJNOG_POREZA = 0.25;
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)
```

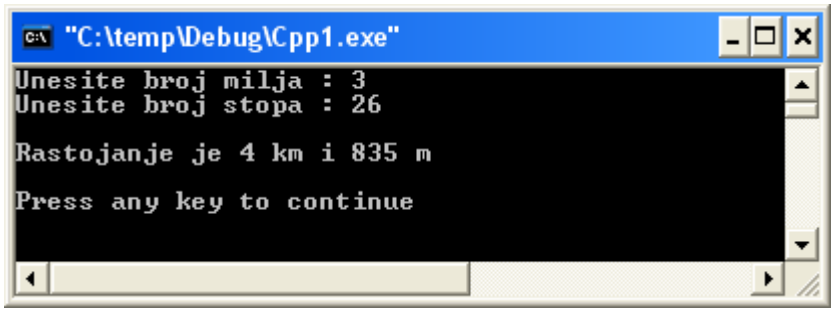
```
<< setiosflags(ios::showpoint);
// Ispisivanje naziva restorana i unos cene obroka
cout << "***** Kod BORE *****" << endl << endl;
cout << "Kazandzisko sokace" <<endl << endl;
cout << "Unesite cenu obroka : EUR ";
cin >> cena_obroka;
cout << endl;
// Izracunavanje poreza i ukupne cene
iznos_poreza = 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;
}
```



5 Zadatak:

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.

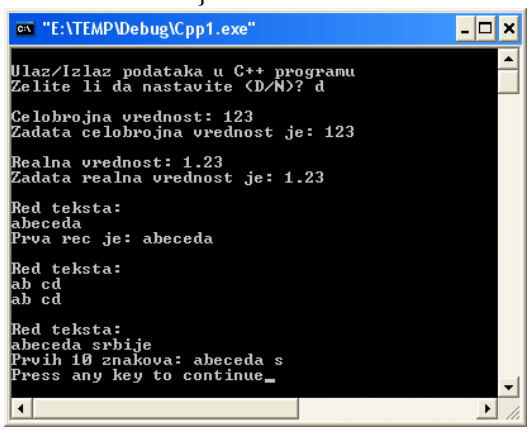
```
// Preracunavanje duzine iz anglosaksonskog u metricki sistem
#include <iostream.h>
int main()
{
// Deklarisanje promenljivih
const double METARA_PO_MILJI = 1609.35;
const double METARA_PO_STOPI = 0.30480;
int milje,
stope,
kilometri,
metri;
double ukupno_metri,
ukupno_kilometri;
// Unos podataka
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;
}
```



6 Zadatak:

//Realizacija ulaza/izlaza (tastatura/ekran) za razlicite vrsta podataka

```
#include <iostream.h>
#include <ctype.h>
#define MAX1 80
#define MAX2 10
main( )
{       int znak, broj_i;    float  broj_f;    char   odgovor, string[MAX1+1];

        cout << "\nUlaz/Izlaz podataka u C++ programu" << endl;
        cout << "Zelite li da nastavite (D/N)? ";
        cin  >> odgovor;
        cin.ignore();                          //prihvata znak za prelaz u novi red
                if( toupper(odgovor) == 'D' )
        {       cout << "\nCelobrojna vrednost: ";
                cin  >>  broj_i;
                cout << "Zadata celobrojna vrednost je: " << broj_i << endl;
                cout << "\nRealna vrednost: ";
                cin  >>  broj_f;
                cout << "Zadata realna vrednost je: " << broj_f << endl;
                //Citanje do prvog belog znaka od zadatog reda znakova
                cout << "\nRed teksta:" << endl;
                cin  >> string;
                cout << "Prva rec je: " << string << endl;
                //Citanje ostatka od zadatog reda znakova
                cin.get(string, MAX1);      //prihvata ostatak do kraja reda
                cin.ignore();               //prihvata znak za prelaz u novi red
                //Citanje znak po znak do znaka za prelaz u novi red
                cout << "\nRed teksta:" << endl;
                while((znak = cin.get()) != '\n')
                cout.put((char)znak);       //moze i: cout << (char)znak;
                //Citanje zadatog broja znakova ili do znaka za prelaz u novi red
                cout << "\n\nRed teksta:" << endl;
                cin.get(string, MAX2);
          cout << "Prvih " << MAX2<< " znakova: " << string << endl;          }
        return 0;
        }
```

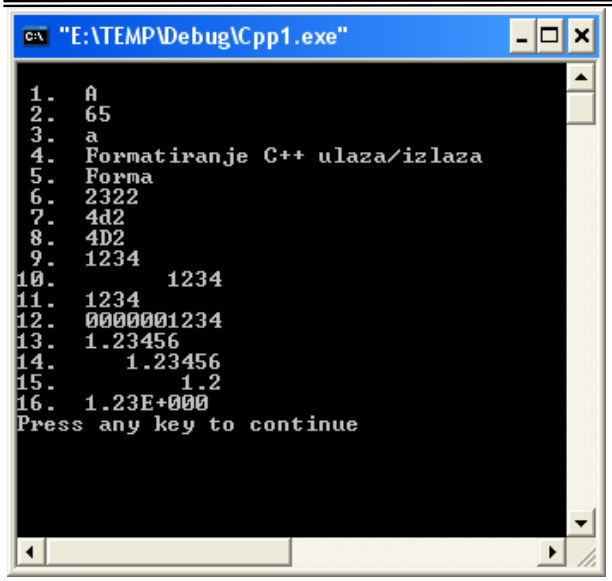


7 Zadatak:

//Primena funkcija za konverziju i formatiranje ulaza/izlaza

```
#include <iostream.h>
#include <iomanip.h>
void stampa_rbr( void );

main( )
{       char slovo = 'A';     char string[] = "Formatiranje C++ ulaza/izlaza";
        int broj_i = 1234;    double broj_d = 1.23456;

        //1. stampanje samog slova A
        stampa_rbr();
        cout << slovo;

        //2. stampanje ASCII koda slova A
        stampa_rbr();
        cout << (int)slovo;

        //3. stampanje znaka zadate ASCII vrednosti (slovo a)
        stampa_rbr();
        cout << (char)97;

        //4. stampanje niza znakova
        stampa_rbr();
        cout << string;

        //5. stampanje prvih 5 znakova od niza
        stampa_rbr();
        cout.write(string,5);

        //6. stampanje celog broja u oktalnom obliku
        stampa_rbr();
        cout.setf(ios::oct);
        cout << broj_i;                    //moze i: cout << oct << broj_i;
        cout.unsetf(ios::oct);

        //7. stampanje celog broja u heksa obliku (malim slovima)
        stampa_rbr();
        cout.setf(ios::hex);
        cout << broj_i;                    //moze i: cout << hex << broj_i;

        //8. stampanje celog broja u heksa obliku (velikim slovima)
        stampa_rbr();
        cout.setf(ios::uppercase);
        cout << broj_i;
        cout.unsetf(ios::hex);
```

```
        //9. stampanje celog broja u decimalnom obliku
        stampa_rbr();
        cout << broj_i;                    //moze i: cout << dec << broj_i;

        //10. stampanje celog broja,u polju zadate sirine, desno ravnanje
        stampa_rbr();
        cout.width(10);
        cout << broj_i;

        //11. stampanje celog broja, u polju zadate sirine, levo ravnanje
        stampa_rbr();
        cout.width(10);
        cout.setf(ios::left);
        cout << broj_i;
        cout.unsetf(ios::left);

        //12. stampanje celog broja, desno popravnanje, nule ispred broja
        stampa_rbr();
        cout.width(10);
        cout.fill('0');
        cout << broj_i;
        cout.fill(' ');

        //13. stampanje realnog broja, sa standardnom preciznoscu
        stampa_rbr();
        cout << broj_d;

        //14. stampanje realnog broja, u polju zadate sirine,
        stampa_rbr();
        cout.width(10);
        cout << broj_d;

        //15. stampanje realnog broja, sa 2. decimalne cifre
        stampa_rbr();
        cout.width(10);
        cout.precision(2);
        cout << broj_d;

        //16. stampanje realnog broja, sa eksponentom
        stampa_rbr();
        cout.setf(ios::scientific);
        cout << broj_d << endl;
        cout.unsetf(ios::scientific);

        return 0;
}
//Funkcija koja prikazuje redni broj na pocetku reda
void stampa_rbr (void)
{       static int redni_broj = 0;
        cout << "\n";
        cout.width(2);
        cout << ++redni_broj << ".  ";
}
```



8 Zadatak:

```
//Tabeliranje vrednosti izraza 10 stepenovan brojem x u zadatom opsegu
        // i sa zadatim korakom za x (levo i desno poravnanje vrednosti izraza)

        #include <iostream.h>
        #include <iomanip.h>
        #include <math.h>
        #define MIN             0
        #define MAX             20
        #define MAX1    2
        #define MAX2    10

        main()
        {       int xmin, xmax, dx;
                do
                {       cout << "\nUnesite cele brojeve ";
                cout << "xmin>=" << MIN << ", xmax<=" << MAX << " i dx: ";
                        cin >> xmin >> xmax >> dx;
                }while(xmin<MIN || xmax>MAX || dx>xmax);

                //Stampanje na ekranu zaglavlja tabele i tabele
                cout << "\nx\t10 stepenovano brojem x\t\t10 stepenovano brojem x"
                        << "\n\t(desno poravnanje)\t\t(levo poravnanje)"
                        << "\n================================"
                        << "================================\n\n";

                cout.setf(ios::fixed);
                cout.precision(0);

                for(int x = xmin; x<=xmax; x+=dx)
                {       //Desno poravnanje vrednosti x  u 1. koloni tabele
```

```
                        cout.width(MAX1);
                        cout.unsetf(ios::left);
                        cout << x << "\t\t";

                        //Desno poravnjanje vrednosti izraza u 2. koloni tabele
                        cout.width(MAX2);
                        cout.unsetf(ios::left);
                        cout << pow(10,x) << "\t\t";

                        //Levo poravnjanje vrednosti izraza u 3. koloni tabele
                        cout.width(MAX2);
                        cout.setf(ios::left);
                        cout << pow(10,x) << "\n";
                }
                cout << "\n";
                return 0;
        }
```

9 Zadatak:

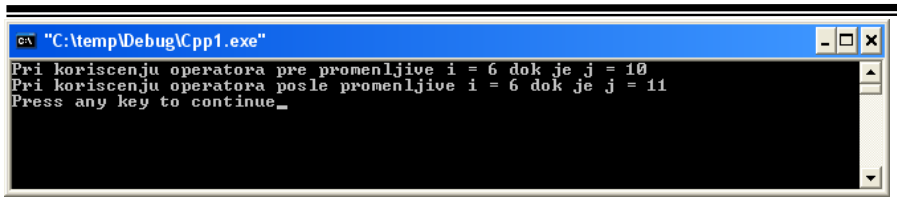
```
//Citanje niza celih brojeva iz komandne linije i prikaz na ekranu
//njegovih elemenata u oktalnom, heksadecimalnom i binarnom obliku

        #include <iostream.h>
        #include <iomanip.h>
        #include <stdlib.h>

        void stampa_bin( int decimal_value );

        main(int argc, char *argv[])
        {       int decimalni;

                if( argc < 2 )
                {       cerr << "\nUneti ime programa i niz celih brojeva." << endl;
                        exit(0);
                }

                cout << "\nZadati niz brojeva u raznim oblicima:\n\n";
```

```
        cout << "dec\tokt\theksa\tbin" << endl;

        for(int i=1; i<argc; i++)
        {       decimalni = atoi( argv[i]);
                cout << "\n" << decimalni << "\t";
                cout << oct << decimalni << "\t";
                cout << hex << decimalni << "\t";
                stampa_bin(decimalni);
        }

        return 0;
}
//Funkcija koja prikazuje na ekranu niz u binarnom obliku
void stampa_bin(int broj)
{       int broj_cifara = 0;
        int niz_binarnih[50];

        while( broj != 0 )
        {       niz_binarnih[broj_cifara++] = broj % 2;
                broj /= 2;
        }

        for( int i=broj_cifara-1; i>=0; i-- )
                cout << dec << niz_binarnih[i];
}
```

10 Zadatak:

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.

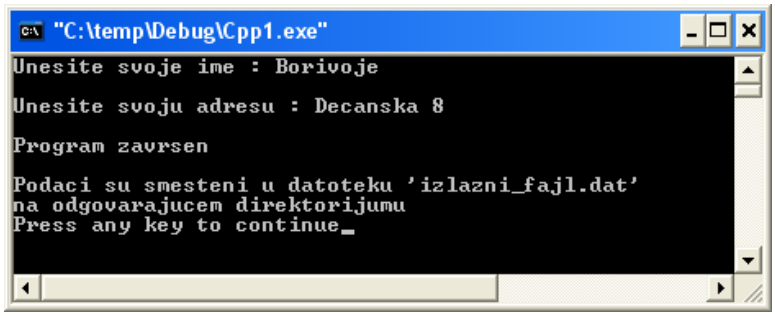
```
// Program ilustruje razliku izmedju primene opearotra pre
// i posle promenljive
#include <iostream.h>
int main()
{// Deklarisanje promenljivih
int i,j;
// Izracunavanja i ispisivanje rezultata
i = 7; // Dodeljivanje inicijalne vrednosti
j = 4 + --i; // prvo se od i oduzme 1 a zatim se
             // 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;}
```



11 Zadatak:

Napisati program koji upisuje podatke u fajl.

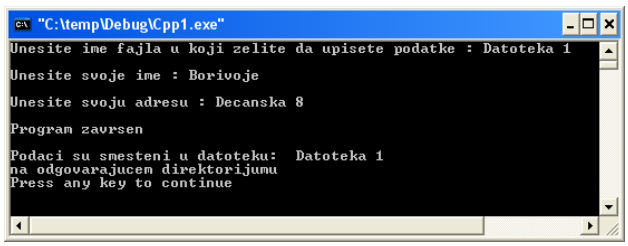
```
// Program prikazuje upis podataka u fajl
#include <fstream>
#include <iostream>
using namespace std;

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;
cout << "Podaci su smesteni u datoteku 'izlazni_fajl.dat'" << endl;
cout << "na odgovarajucem direktorijumu" << endl;
return 0;
}
```



12 Zadatak:

Napisati program koji upisuje podatke u fajl i proverava da li postoji greška prilikom otvaranja fajla.

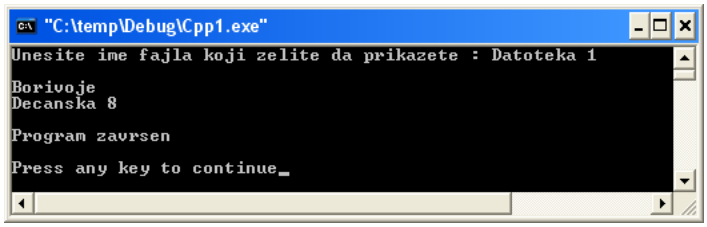
```
// Program demonstrira upis podataka u fajl
// Proverava da li postoji greska prilikom otvaranju fajla
#include <fstream>
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
char buffer[81];
char ime_fajla[81];
cout << "Unesite ime fajla u koji zelite da upisete podatke : ";
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;
cout << "Podaci su smesteni u datoteku:  "<< ime_fajla << endl;
cout << "na odgovarajucem direktorijumu" << endl;
return 0;
}
```



13 Zadatak:

Napisati program koji čita podatke iz fajla.

```
// 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;
}
```

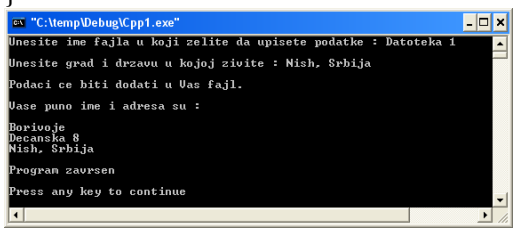


14 Zadatak:

Napisati program koji upisuje podatke u fajl, čita podatke iz fajla i dodaje podatke u fajl.

```
// 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;
}
```

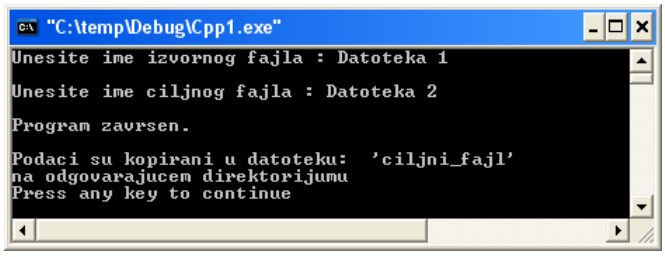


15 Zadatak:

Napisati program koji kopira jedan fajl u drugi koristeći get() and put() naredbe.

```
// 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;
cout << "Podaci su kopirani u datoteku:  'ciljni_fajl'"<<endl;
cout << "na odgovarajucem direktorijumu" << endl;

return 0;}
```



KONVERZIJA TIPOVA

Na primer, pogledajmo sledeći C++ fragment, koji izračunava i ispisuje zbir recipročnih vrednosti svih brojeva od 1 do 1000:

```
double suma = 0;
for(int i = 1; i <= 1000; i++)
suma =suma+ 1. / i;
cout<< suma;
```

U ovom primeru, promjenljiva "i", deklarisana unutar **"for"** petlje, postoji samo dok se petlja ne završi. U C++-u je preporuka da se brojačke promjenljive koje upravljaju radom **"for"** petlji deklarišu isključivo na ovakav način. Na taj način se sprečava mogućnost da se brojačka promjenljiva nehotice neispravno upotrebi ili čak zloupotrebi izvan tela petlje.

Obratimo pažnju da smo u prethodnom primeru pisali "1." umjesto "1". Da to nismo uradili, imali bismo pogrešan rezultat. Naime, kako je "1" celobrojnog tipa, i kako je promjenljiva "i" takođe celobrojna, operator "/" u izrazu kao "1 / i" interpretirao bi se kao *celobrojno deljenje.* Ovako, kako je "1." realnog tipa, operator "/" se interpretira kao *klasično deljenje.* Ovo je dobar povod da kažemo nešto o ***konverziji tipova*** u C++-u. C++ podržava skoro sve automatske konverzije tipova koji se dešavaju i u jeziku C (uz retke izuzetke koji će biti posebno istaknuti). Na primer, celobrojni izrazi se automatski konvertuju u realne ukoliko se upotrebe u kontekstu u kojem se očekuje realan izraz, npr. ukoliko želimo dodeliti neki celobrojni izraz realnoj promenljivoj. Ovakvu automatsku konverziju, pri kojoj se "uži" tip konvertuje u "širi" obično zovemo *promocija.* S druge strane, realan izraz često možemo upotrebiti u kontekstu u kojem se očekuje celobrojni izraz, npr. kao argument neke funkcije koja očekuje celobrojni argument, ili pri dodeli realnog izraza celobrojnoj promenljivoj. Takve konverzije praćene su *gubitkom informacija* (u konkretaom primeru *odsecanjem decimald),* i obično ih nazivamo *degradacija.* C++ kompajleri imaju pravo da nailaskom na degradirajuću automatsku konverziju emituju *poruku upozorenja,* ali ne smeju odbiti da prevedu program.

Operator ***(tip)izraz*** se može koristiti i u jeziku C++. Pogledajmo, na primer, sledeći programski isečak koji ispisuje količnik dve celobrojne vrednosti unesene sa tastature:

```
int broj_l, broj_2;
cin>> brojl >> broj_2;
cout<< (double)brojl / broj_2;
```

U ovom isečku, pomoću eksplicitne konverzije privremeno smo promenili tip promenljive "brojl" u realni tip **"double"** da bismo izbegli da operator "/" bude interpretiran kao celobrojno deljenje. Mada ovakva sintaksa i dalje radi u jeziku C++, ona se više *ne preporučuje.* C++ uvodi dve nove sintakse za konverziju tipova:

- Prva je takozvana ***funkcijska* ili *konstruktorska* notacija,** koja izgleda ovako:

tip(izraz)

Dakle, u ovoj sintaksi, ime tipa se tretira kao *funkcija koja svoj argument prevodi u rezultat iste vrednosti, ali odgovarajućeg tipa.* Prethodni primjer bi se, u skladu sa ovom sintaksom, napisao ovako:

```
int brojl, broj_2;
cin >> broj_l >> broj_2;
cout« double(broj 1) / broj_2;
```

Ova sintaksa može biti mnogo praktičnija kod složenijih izraza. Naime, zbog vrlo visokog prioriteta operatora konverzije tipa u C stilu, često su svakako potrebne dodatne

zagrade. Na primer, neka želimo izračunati i ispisati celi dio produkta "2.541 * 3.17". Koristeći C notaciju, to bismo uradili ovako:

cout<< (int)(2.541 * 3.17);

Dodatne zagrade su neophodne, jer bi se u suprotnom operator konverzije "(int)" odnosio samo na podatak "2.541" (odnosno, kao rezultat bismo dobili vrijednost produkta "2 * 3.17"). Korištenjem flinkcijske notacije, istu stvar možemo uraditi ovako:

cout<< int(2.541 * 3.17);

Na ovaj način, ime tipa **"int**' se ujedno može interpretirati i kao *funkcija* koja vraća kao rezultat celi deo svog argumenta.

• Druga sintaksa za konverziju tipa uvedena u jeziku C++, koristi novu ključnu reč **"static_cast",** a izgleda ovako:

static_cast<tip>(izraz)

U skladu sa ovom sintaksom, prethodno napisani primeri izgledali bi ovako:

```
int brojl, broj_2;
cin >> brojl>> broj_2;
cout<< static_cast<double>(broj_l) / broj_2;
```

odnosno

cout<< static_cast<int>(2.541 * 3.17);

Ova sintaksa je dosta rogobatna, ali ona ima svoje prednosti. Prvo, mesta gde se vrše konverzije tipa su potencijalno opasna mjesta, koja često prave probleme kada se program treba preneti sa jednog tipa računara na drugi, ili sa jednog na drugi operativni sistem. Korištenjem ove sintakse, takva mesta u programu je lako locirati prostim traženjem ključne riječi "static_cast". Drugo, konverzija tipa se. za različite tipove u C++-u često vrši na suštinski različite načine, na šta programer ranije nije imao nikakvog uticaja. Sada su u C++-U uvedene četiri ključne reči za konverziju tipova (pored već spomenute **"static_cast",** imamo i "const_cast", **"reinterpret_cast"** i **"dynamic_cast**'), čime programer može eksplicitnije da iskaže tačnu nameru kakvu konverziju želi.

FORMATIRANJE IZLAZA

Možemo li pisati početak programa sa "void main()"?

Definicija void main() { /* ... */ } nije preporučljiva u konotaciji C++.

Uglavnom se koristi konotacija

```
int main() { /* ... */ }
```

ili

```
int main(int argc, char* argv[]) { /* ... */ }
```

Možemo implementirati više verzija main() konotacije, ali sve one moraju imati povratni tip **int. Int** vraćen iz main() je način da program vrati vrednost sistemu i obrati joj se. Kod sistema koji ne poseduju takvu olakšicu vrednost return se ignoriše, što ne čini opciju "void main()" legalnom za C++ ili C. I u slučaju da kompajler prihvati "void main()", to treba izbegavati, jer u suprotnom rizikujete da vas C i C++ programeri smatraju neznalicom.

U C++, main() ne mora da sadrži explicitno return naredbu. U ovom slučaju, vraćena vrednost 0, podrazumeva uspešno izvršenje. Naprimer:

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

int main()
{
std::cout << "This program returns the integer value 0\n";
```

Imajte u vidu da ni ISO C++ ni C99 neomogućuju da izostavimo tip deklaracije. Tojest u poređenju sa C89 i ARM C++ ,"int" nije dozvoljeno da se ne pojavi u deklaraciji. Kao posledica toga:

```
#include<iostream.h>

main() { /* ... */ }
```

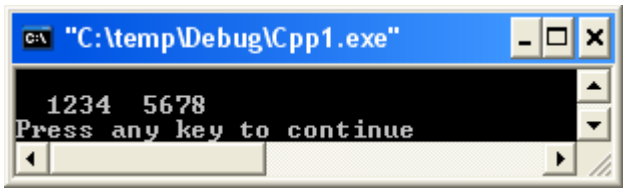
javiće se greška, jer return tip funkcije main() nedostaje.

Return naredba završava izvršenje funkcije i vraća kontrolu na funkciju koja ju je pozvala. Ako izostavimo return vrednost deklarisaćemo funkciju da ima **void** povratni tip i da takva funkcija ne vraća nikakvu vrednost, dok je u ostalim slučajevima povratni tip funkcije po difoltu **int.**

1 Zadatak:

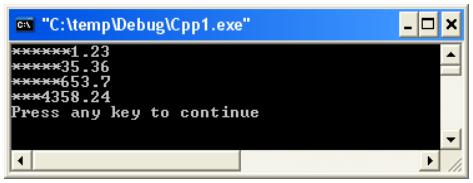
```
// manipulator setw()
#include <iostream.h>
#include <iomanip.h>
int main()
{
  int num1 =1234, num2 =5678;
  cout << endl;                    //Start sa novom linijom
```

```
 cout << setw(6) << num1 << setw(6) << num2;   /*Izlaz dve vrednosti sa     manipulatorom
širine setw*/
   cout << endl;                          //Start novom linijom
   return 0;                              //Exit programa
}
```



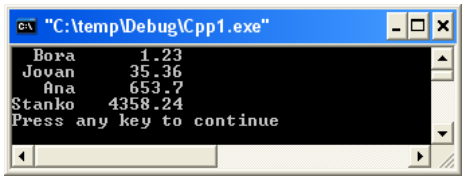
2 Zadatak:

```
//primena operatora width() i  metode fill
#include <iostream.h>
void main()
{
   double values[] = { 1.23, 35.36, 653.7, 4358.24 };
   for( int i = 0; i < 4; i++ )
{ cout.width(10); //formatiranje širine podatka funkcijom članicom width ( slično kao setw )
cout.fill( '*' );  //popuna praznih mesta f-je članice width u izlaznom formatu, naprimer
zvezdicama
cout << values[i] << '\n';  }
}
```



3 Zadatak:

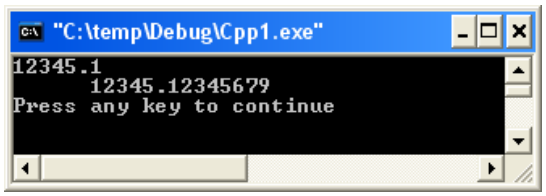
```
#include <iostream.h>
#include <iomanip.h>
void main()
{
   double values[] = { 1.23, 35.36, 653.7, 4358.24 };
   char *names[] = { "Bora", "Jovan", "Ana", "Stanko" };
   for( int i = 0; i < 4; i++ )
      cout << setw( 6 )  << names[i]
           << setw( 10 ) << values[i] << endl;
}
```



4 Zadatak:

```
//primena width i precision formata za IOS
#include <iostream.h>
#include <iomanip.h>
#include <stdlib.h>
int main()
{const double prom=12345.123456789;
cout << prom<<endl;
cout.width(20);     /* "20" nije broj razmaka koji se ispisuju ispred podatka, nego broj mesta
                       koje ce          zauzeti          podatak*/
cout<<setiosflags(ios::fixed);
cout.precision(8);
cout << prom<<endl;
return 0;}
```

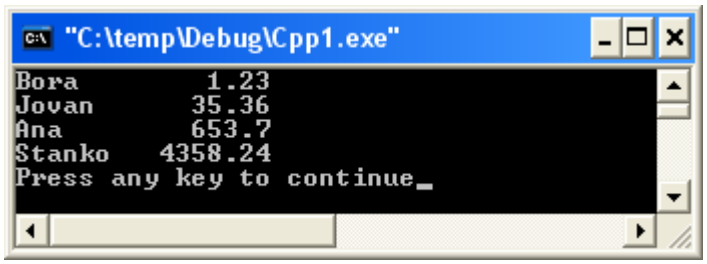
IZLAZ:



5 Zadatak:

```
//ravnanje izlaza levo ili desno
#include <iostream.h>
#include <iomanip.h>
int main()
{
  double values[] = { 1.23, 35.36, 653.7, 4358.24 };
  char *names[] = { "Bora", "Jovan", "Ana", "Stanko" };

for ( int i = 0; i < 4; i++ )
  cout << setiosflags( ios::left )
       << setw( 6 )  << names[i]
       << resetiosflags( ios::left )
       << setw( 10 ) << values[i] << endl;
return 0;}
```

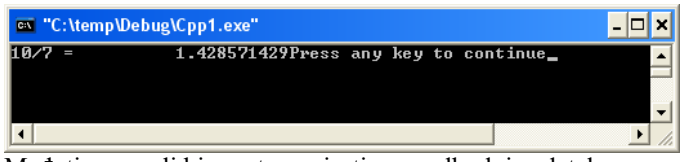
IZLAZ:



6 Zadatak:

```
#include <iostream.h>
#include <iomanip.h>
int main()
{cout << "10/7 = ";
cout.width(20);     /* “20” nije broj razmaka, nego broj mesta koje će zauzeti podatak*/
cout.precision(10);
cout << 10./7.;}
```

IZLAZ:



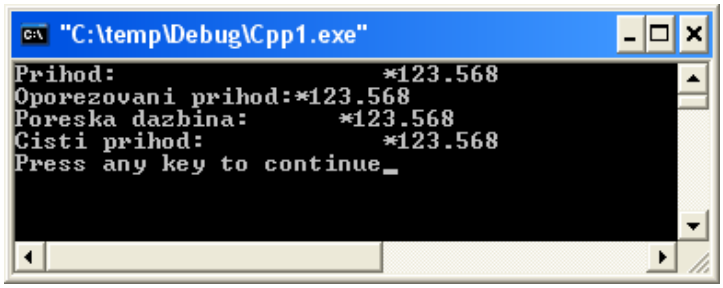
Međutim, mogli bi prosto napisati za predhodni zadatak:

```
cout << "1/7 = " << setw(20) << setprecision(10) << 10./7;
```

7 Zadatak:

```
#include <iostream.h>
#include <iomanip.h>

int main()
{
double const prihod=123.56789;
double const oporezovani_prihod=123.56789;
double const porez=123.56789;
double const prihod_porez=123.56789;
cout.fill( '*' );  // popuna praznih mesta u formatu zvezdicama
cout << "Prihod:          " << setw(8) << prihod << endl;  //    prikaz osam mesta sa tackom
cout << "Oporezovani prihod:" << setw(8) << oporezovani_prihod << endl;
cout << "Poreska dazbina:      " << setw(8) << porez << endl;
cout << "Cisti prihod:              " << setw(8) << prihod_porez << endl;
return 0;}
```

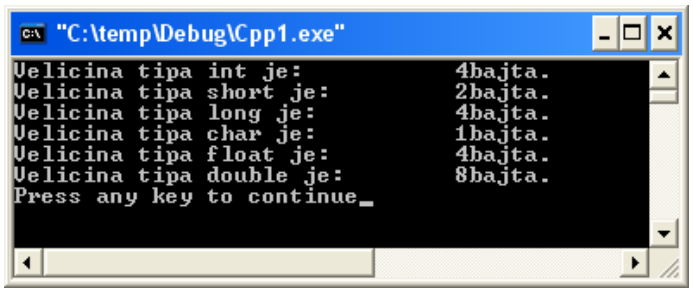
IZLAZ:



8 Zadatak:

```
//funkcije sizeof()
#include <iostream.h>
int main()
{cout<< "Velicina tipa int je: \t\t" << sizeof(int)     <<"bajta.\n";
cout << "Velicina tipa short je:\t\t" << sizeof(short) <<"bajta.\n";
cout << "Velicina tipa long je: \t\t" << sizeof(long)   <<"bajta. \n";
cout << "Velicina tipa char je: \t\t" << sizeof(char)   <<"bajta. \n";
cout << "Velicina tipa float je:\t\t" << sizeof(float) <<"bajta. \n";
cout << "Velicina tipa double je:\t" << sizeof(double) <<"bajta. \n";
return 0;}
```

IZLAZ:

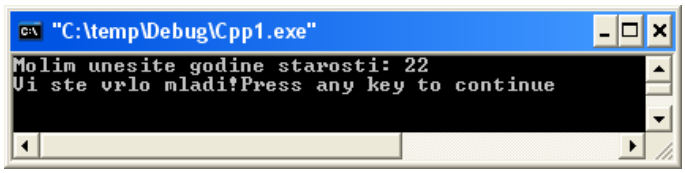


KONTROLA TOKA PROGRAMA

1 Zadatak:

Primer za IF naredbu uz upotrebu else strukture:

```
#include <iostream.h>
int main()                  //Početak programa!
{
 int age;                                          //Definišemo promenljivu...
 cout<<"Molim unesite godine starosti: ";     //Upit na monitoru
 cin>>age;                                         //Unosi se broj godina
 if(age<100)                          //Ako su manje od 100
 {    cout<<"Vi ste vrlo mladi!";    }
 else if(age==100)      //Koristi se  else za novi odnos
 {     cout<<"Vi ste stari";  }
 else
 {    cout<<" Vi ste vrlo stari ";      }
 return 0;}
```

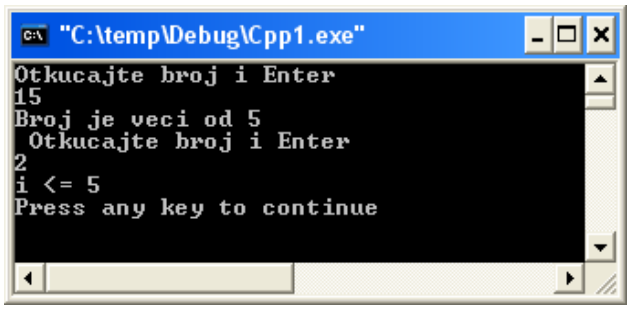


2 Zadatak:

Primer za IF naredbu uz upotrebu lokalnih promenljivih:

```
#include<iostream.h>
#include<math.h>
void main()
{
int i;
cout << "Otkucajte broj i Enter" << endl;
cin >> i;
if(i > 5)
cout << "Broj je veci od 5" << endl;
else
if(i < 5)
cout << "Broj je manji od 5 " << endl;
else
cout << "Broj je jednak 5 " << endl;
cout << " Otkucajte broj i Enter " << endl;
cin >> i;
if(i < 10)
if(i > 5) // "if" je samo sledeca naredba
```

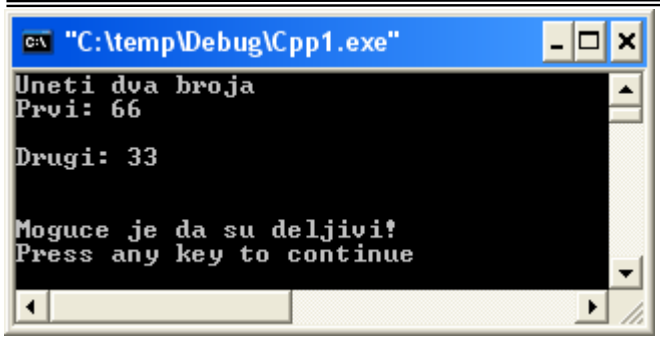
```
cout << "5 < i < 10" << endl;
else
cout << "i <= 5" << endl;
else // Poklapa se "if(i < 10)"
cout << "i >= 10" << endl;
}
```



3 Zadatak:

Kompleksno ugnježdena if naredba

```
#include <iostream.h>
 int main()
 {   // Unosimo dva broja
     // Brojevima dodeljujemo Veliki broj ili Mali broj
     // Ako je Veliki broj veci od  Mali broj
     // videti dali su moyda deljivisee if they are evenly divisible
     // I ako jesu videti dali su mozda to isti brojevi

    int prviBroj, drugiBroj;
    cout << "Uneti dva broja\nPrvi: ";
    cin >> prviBroj;
    cout << "\nDrugi: ";
    cin >> drugiBroj;
    cout << "\n\n";
    if (prviBroj >= drugiBroj)
    {
      if ( (prviBroj % drugiBroj) == 0) // eventualno deljivi?
      {
          if (prviBroj == drugiBroj)
              cout << "Ovo su isti brijevi!\n";
          else
              cout << "Moguce je da su deljivi!\n";
      }
      else
          cout << "Nije moguce da su deljivi!\n";
    }
    else
          cout << "Hey! Drugi broj je veci!\n";
    return 0;
 }
```



4 Zadatak:

Listing za demonstraciju zašto su zagrade važne za ugnježdene if naredbe - KOJI JE PROGRAM DOBAR?

```
  #include <iostream.h>
  int main()
  {
    int x;
 cout << "Unesi broj manji od 10 ili veći
od  100: ";
    cin >> x;
    cout << "\n";

    if (x >= 10)
      if (x > 100)
          cout << " Veći od 100, Hvala!\n";
    else
      cout << " Manji od 10, Hvala!\n";

    return 0;
  }
```

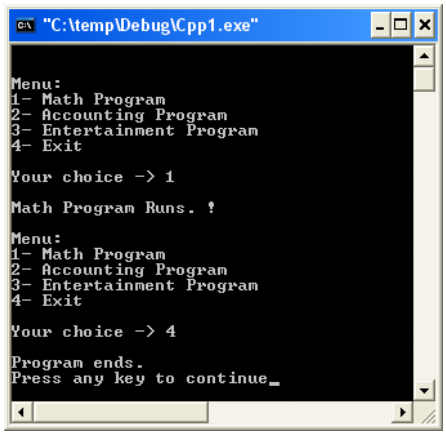
```
#include <iostream.h>
   int main()
   {
     int x;
cout << "Unesi broj manji od 10 ili veći od
100: ";
     cin >> x;
     cout << "\n";

     if (x >= 10)
       if (x > 100)
{cout << "Veći od 100, Hvala!\n";goto kraj;}
    // else                    // nepotrebno
       cout << "Manji od 10, Hvala!\n";
     kraj:
     return 0;
   }
```

5 Zadatak:

Prikaz kompleksne "if" statement

```
#include <iostream.h>
#include <stdlib.h>
main()
{int choice;
while(1)
  {
  cout<<"\n\nMenu:\n";
  cout<<"1- Math Program\n2- Accounting Program\n";
  cout<<"3- Entertainment Program\n4- Exit";
```

```
cout<<"\n\nYour choice -> ";
cin>>choice;
if(choice==1)
  cout<<"\nMath Program Runs. !";
else if(choice==2)
  cout<<"\nAccounting Program Runs. !";
else if(choice==3)
  cout<<"\nEntertainment Program Runs. !";
else if(choice==4)
  {    cout<<"\nProgram ends.\n";
  exit(0);
  }
else
  cout<<"\nInvalid choice";  }}
```

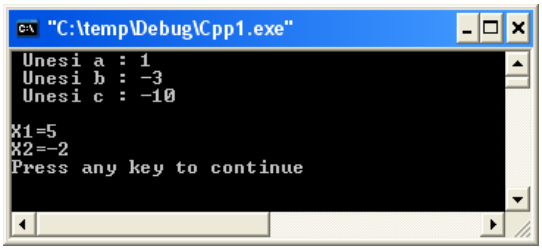


6 Zadatak:

Program za rešavanje kvadratne j-ne $ax^2+bx+c=0$.

```
#include <iostream.h>
#include <math.h>
#include <stdlib.h>
void main()
{
double delta,a,b,c,x1,x2;
cout<<" Unesi a : ";
cin>>a;
cout<<" Unesi b : ";
cin>>b;
cout<<" Unesi c : ";
cin>>c;
delta=b*b-(4*a*c);
if(delta<0)
  {  cout<<"Jednacina nema resenja !\n";
  exit(0);  }
if(delta==0)
  {  x1=-b/(2*a);
```

```
 cout<<"Postoje dva identicna resenja !\n";
 cout<<"x1=x2=%"<<x1;
 exit(0);  }
x1=(-b+sqrt(delta))/(2*a);
x2=(-b-sqrt(delta))/(2*a);
cout<<"\nX1="<<x1;
cout<<"\nX2="<<x2<<endl;}
```

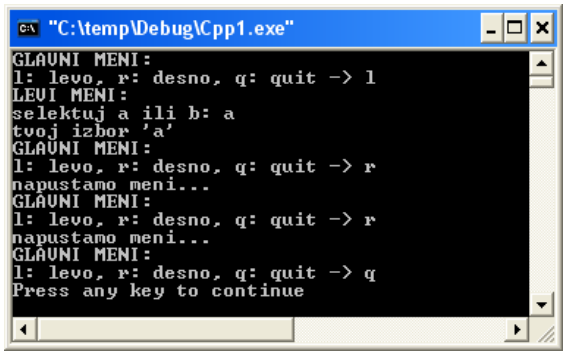


7 Zadatak:

Program za demonstraciju "break" i "continue"

```
#include <iostream.h>
void main()
{
char c; // Za definisanje izbora
while(1)
{
cout << "GLAVNI MENI:" << endl;
cout << "l: levo, r: desno, q: quit -> ";
cin >> c;
if(c == 'q')
break; // Izlaz iz "while(1)"
if(c == 'l') {
cout << "LEVI MENI:" << endl;
cout << "selektuj a ili b: ";
cin >> c;
if(c == 'a') {
cout << "tvoj izbor 'a'" << endl;
continue; // Povratak u glavni meni}
if(c == 'b') {
cout << " tvoj izbor 'b'" << endl;
continue; // Povratak u glavni meni}
else {cout << "nisi izabrao a ili b!"<< endl;
continue; // Povratak u glavni meni}}
if(c == 'r') {
cout << "DESNI MENI:" << endl;
cout << "selektuj c ili d: ";
cin >> c;
if(c == 'c') {
cout << " tvoj izbor 'c'" << endl;
continue; // Povratak u glavni meni}
```

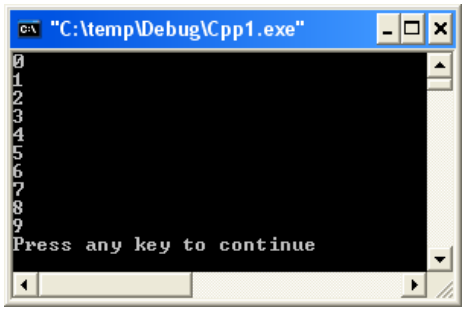
```
if(c == 'd') {
cout << " tvoj izbor 'd'" << endl;
continue; // Povratak u glavni meni}
else {
cout << " nisi izabrao c ili d!"
<< endl;
continue; // Povratak u glavni meni}}
cout << "moras otkucati l ili r ili q!" << endl;}
cout << "napustamo meni..." << endl;}}
```



8 Zadatak:

Primer za FOR naredbu uz upotrebu inkrementa:

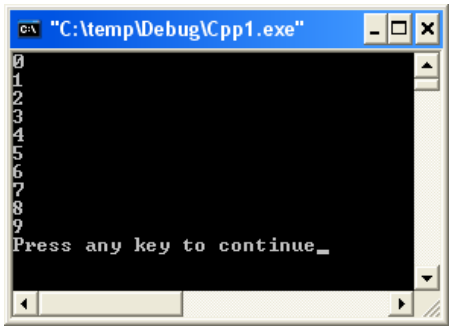
```
#include <iostream.h>
int main()
 {                          //Petlja se vrto dok je x<100, i x se u svakoj petlji povećava za 1
for(int x=0;x<10;x++) //Zapamtite da uslov petlje proverava promenljivu u petlji pre
{cout<<x<<endl;  }                 //Kada je  x jednako 100 petlja se prekida
                                   //Izlazi za  x
  return 0;
}
```



9 Zadatak:

Primer za WHILE naredbu :

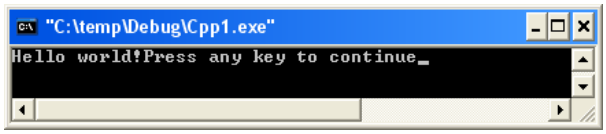
```
#include <iostream.h>
int main()
{
 int x=0;                          //Ne zaboravite da deklarišete promenljivu
 while(x<10)                                    //Dok je x manje od 100 uradi
 {    cout<<x<<endl;               //Isti izlaz kao u petlji gore
   x++;                            //Dodaje 1 na x svaki put kad se petlja ponovi

 }
 return 0;}
```



10 Zadatak:

Primer za DO WHILE naredbu :

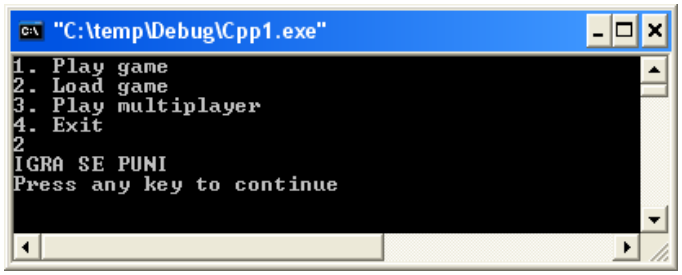
```
#include <iostream.h>
int main()
{  int x;
  x=0;
  do
  { cout<<"Hello world!"; }
while(x!=0);                        //Petlja radi dok x nije nula, ali se prvo izvrši kod u
sekciji:
                                                //Izlaz je "Hello world" jednom
  return 0;}
```



11 Zadatak:

Primer za SWITCH naredbu uz upotrebu case strukture:

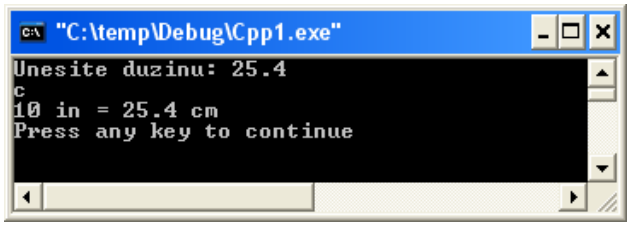
```
#include <iostream.h>
#include <conio.h>
void playgame();
void loadgame();
void playmultiplayer();
int main()
{  int input;
  cout<<"1. Play game"<<endl;
  cout<<"2. Load game"<<endl;
  cout<<"3. Play multiplayer"<<endl;
  cout<<"4. Exit"<<endl;
  cin>>input;
  switch (input)
  {  case 1: playgame();// Paziti na prototipove - obezbediti funkcije
              break;
   case 2:
              loadgame();
       break;
   case 3:
       playmultiplayer();
              break;
   case 4:
       return 0;
  default:   cout<<"GRESKA Los ulaz";
  }
  return 0;}

void playgame()
{cout<<"POCINJE IGRA"<<endl;}
void loadgame()
{cout<<"IGRA SE PUNI"<<endl;}
void playmultiplayer()
{cout<<"IGRA ZA VISE IGRACA"<<endl;}
```



12 Zadatak:

Još jedan primer upotrebe case naredbe

```
#include<iostream.h>
#include<math.h>
int main()
{const double factor = 2.54;                    // 1 inch je 2.54 cm
double x, in, cm;
char ch = 0;
cout << "Unesite duzinu: ";
cin >> x;                                       // citanje floating-point broja
cin >> ch;                                      // citanje sufix
switch (ch)
{
case 'i': // ako se unese i radi se o inch
in = x;
cm = x*factor;
break;
case 'c': // ako se unese i radi se o cm
in = x/factor;
cm = x;
break;
default:
in = cm = 0;
break;
}
cout << in << " in = " << cm << " cm\n";
return 0;}
```



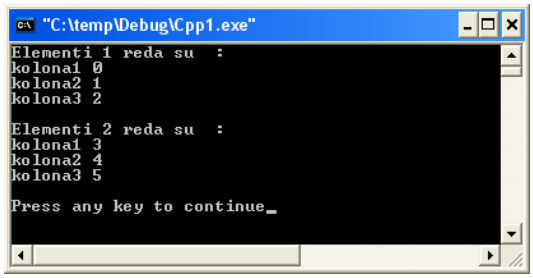
13 Zadatak:

Elementi matrice, kombinovanje for naredbi

```
#include<iostream.h>
#include<math.h>
void main()
{
static int mat[2][3]={{0,1,2},{3,4,5}};
int i,j;
for(i=1;i<=2;++i)
{
cout<<"Elementi "<< i<<" reda su  :\n";
```

```
for(j=1;j<=3;++j)
cout<<"kolona"<<j<<" "<<mat[i-1][j-1]<<"\n";
cout<<"\n";
}}
```

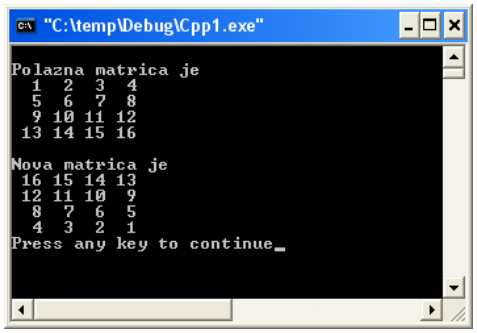
IZLAZ:



14 Zadatak:

Štampanje članova matrice i njenih članova unazad

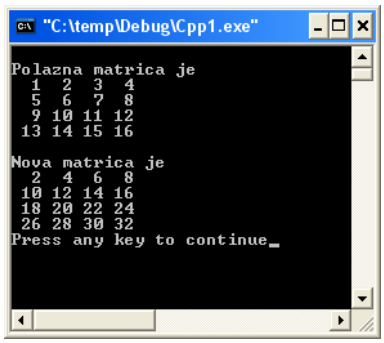
```
#include <iostream.h>
#include<math.h>
#include <iomanip.h>
void main()
{int mat[4][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12},{13,14,15,16}};
int i,j;
cout<<"\nPolazna matrica je\n";
for(i=0;i<4;++i)
{
 for(j=0;j<4;++j)
{cout<<setw(3)<<mat[i][j];}
cout<<"\n";}
cout<<"\nNova matrica je\n";
for(i=3;i>=0;--i)
{
 for(j=3;j>=0;--j)
{cout<<setw(3)<<mat[i][j];}
cout<<"\n";}}
```



15 Zadatak:

Štampanje članova matrice i njenih članova pomnoženih skalarom

```
#include <iostream.h>
#include <math.h>
#include <iomanip.h>
void main()
 {int mat[4][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12},{13,14,15,16}};
 int i,j;
 cout<<"\nPolazna matrica je\n";
 for(i=0;i<4;++i)
{
 for(j=0;j<4;++j)
{cout<<setw(3)<<mat[i][j];}
cout<<"\n";}
 cout<<"\nNova matrica je\n";
for(i=0;i<4;++i)
{
 for(j=0;j<4;++j)
{cout<<setw(3)<<2*mat[i][j];}
cout<<"\n";}}
```



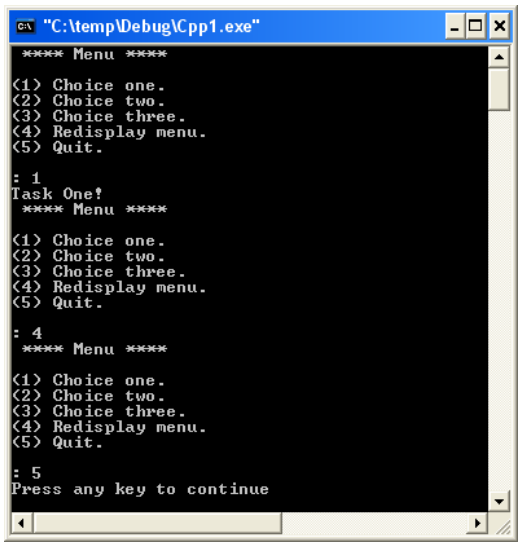
16 Zadatak:

Korišćenje neprekidne petlje koja se upravlja korisničkom interakcijom

```
#include <iostream.h>
// prototypes
int menu();
void DoTaskOne();
void DoTaskMany(int);

int main()
{

   bool exit = false;
   for (;;)
   {
```

```
        int choice = menu();
        switch(choice)
        {
          case (1):
              DoTaskOne();
              break;
          case (2):
              DoTaskMany(2);
              break;
          case (3):
              DoTaskMany(3);
              break;
          case (4):
              continue;  // redundant!
              break;
          case (5):
              exit=true;
              break;
          default:
              cout << "Please select again!\n";
              break;
        }         // end switch

        if (exit)
             break;
    }               // end forever
   return 0;
}                   // end main()



int menu()//definisanje funkcije
  {
     int choice;

     cout << " **** Menu ****\n\n";
     cout << "(1) Choice one.\n";
     cout << "(2) Choice two.\n";
     cout << "(3) Choice three.\n";
     cout << "(4) Redisplay menu.\n";
     cout << "(5) Quit.\n\n";
     cout << ": ";
     cin >> choice;
     return choice;
  }

  void DoTaskOne()
  {
     cout << "Task One!\n";
  }

  void DoTaskMany(int which)
  {
```

```
    if (which == 2)
        cout << "Task Two!\n";
    else
        cout << "Task Three!\n";
}
```

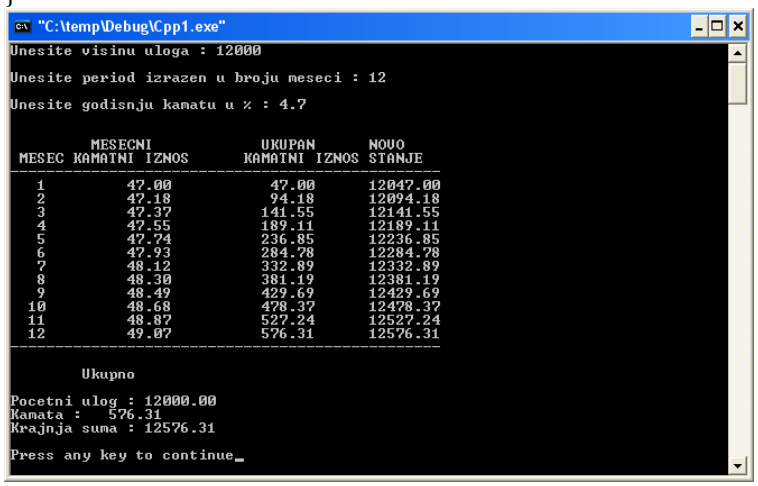


17 Zadatak:

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.

```
// 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.h>
#include <iomanip.h>
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)
```

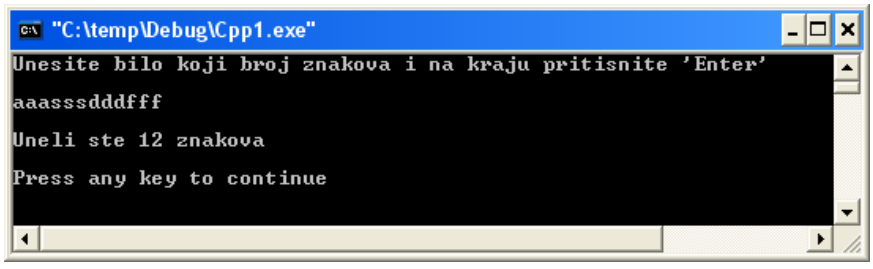
```
<< setiosflags(ios::showpoint);
// Unos podataka
cout << "Unesite visinu uloga : ";
cin >> ulog;
cout << endl;
cout << "Unesite period izrazen u broju meseci : ";
cin >> period;
cout << endl;
cout << "Unesite godisnju kamatu u % : ";
cin >> godisnja_kamata;
// Izracunavanja i ispisivanje rezultata
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;
}
```



18 Zadatak:

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.

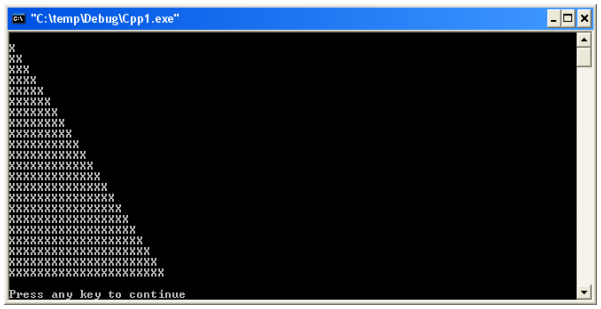
```
// Program broji unete znakove koje je upisao korisnik u jednoj liniji
// Linija se zavrsava pritiskom na Enter. Enter se ne racuna kao karakter
// Primer iteracije.
#include <iostream.h>
int main()
{
// Deklarisanje promenljivih
char znak; // Koristi se za prihvat unetog znaka
int broj_znakova = 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;
}
```



19 Zadatak:

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.

```
// 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;
}
```



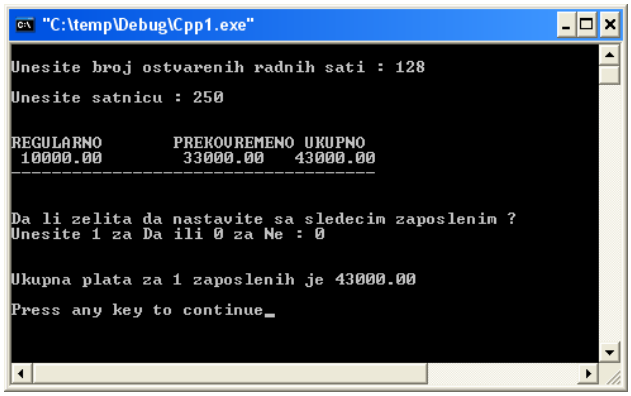
20 Zadatak:

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.

```
// 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.h>
#include <iomanip.h>
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;
}
```

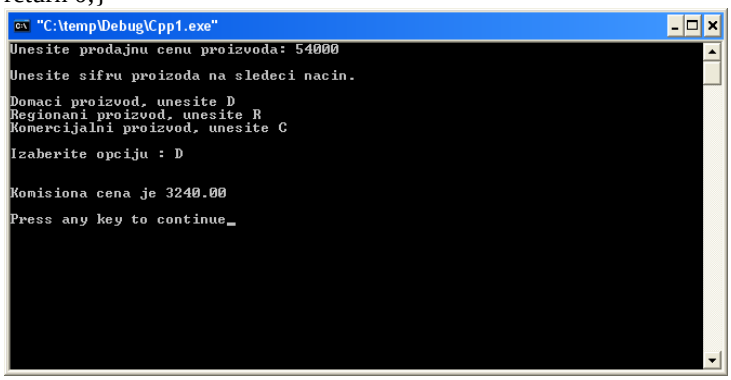


21 Zadatak:

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

```
#include <iostream.h>
#include <iomanip.h>
#include <stdlib.h>
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;}
```

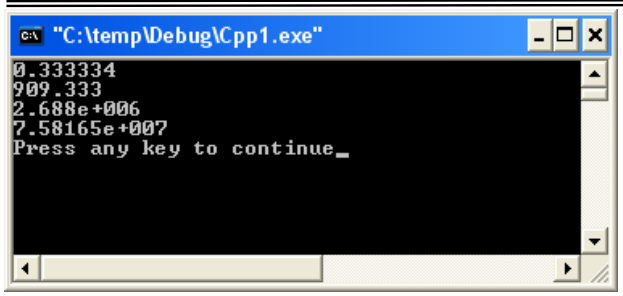


22 Zadatak:

Izračunati vrednost integrala

$\int_a^b x^2 dx$ za granice integraljenja a=0, b=1; a=-10, b=12; a=-200, b=40 i a=121, b=612 funkcijom **double integrate** sa brojem iteracija 500.

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

double xSquared(double x)
{    // power funkcija, se mo\e yameniti sa x * x
   return pow(x, 2);}

double integrate(double (*pFunc)(double), double nDonjaGranica,
       double nGornjaGranica, long nIteracija = 500)
{
   // Formiranje totalnog broja intervala koji se koristi
   double nInterval = abs(nGornjaGranica - nDonjaGranica) * nIteracija;

   // Specifikacija sirine svakog intervala (delta x)
   double nIntervalWidth = (nGornjaGranica - nDonjaGranica) / nInterval;

   // variabla u kojoj se cuva totalna suma
   double nTotalSum = 0;

   for(double i = 0, j = nDonjaGranica; i < nInterval; i++)
   {
      // temporary variabla za cuvanje sledeceg subintervala
      double k = j + nIntervalWidth;

      // dodavanje  sumi
      nTotalSum += (nIntervalWidth / 2) * ( pFunc(j) + pFunc(k) );

      // setovanje j na sledeci subinterval
      j = k;
   }

   return nTotalSum;
}

int main(int argc, char *argv[])
{

   cout << integrate(xSquared, 0, 1) << endl;
   cout << integrate(xSquared, -10, 12) << endl;
   cout << integrate(xSquared, -200, 40) << endl;
   cout << integrate(xSquared, 121, 612) << endl;

   return 0;}
```



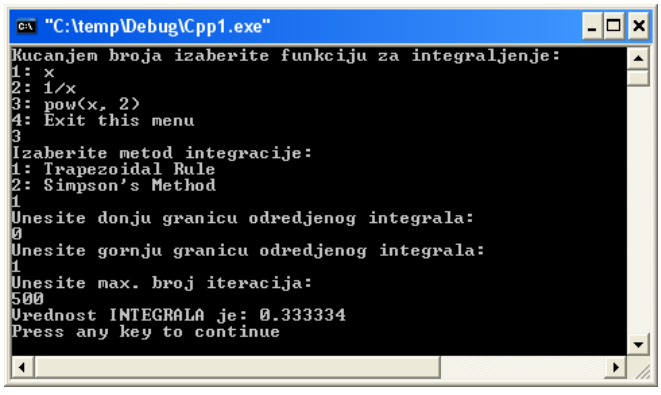
23 Zadatak:

Program nudi korisnicima 2 menija sa listom opcija koje moze da izabere. Prva opcija je u izboru jedne od podintegralnih funkcija koju resavamo a finalna opcija je u izboru metoda integracije. Rezultat takve numericke integracije se na izlazu pojavljuje u decimalnoj formi.

```
#include <iostream>
#include <iomanip>
#include <cstdlib>
#include <string>
#include <cmath>
using namespace std;

void getlimit(double& lower, double& upper, int& imax); // prototip funkcije ogranicenja


 double trapezoid(double(*f)(double), double a, double b, int imax)  // kodni blok za upotrebu trapezoidalnog metoda integracije
                                    {
                                       double r, dx, x;
                                       r = 0.0;
                                       dx = (b-a)/static_cast<float>(imax);

                                       for (int i = 1; i <= imax-1; i = i+1)
                                       {
                                        x = a + static_cast<float>(i)*dx;
                                        r = r + f(x);
                                       }
                                       r = (r + (f(a)+f(b))/2.0)* dx;
                                       return r;
                                    }

        double simpson(double(*f)(double), double a, double b, int imax) // kodni blok za upotrebu simpsonove metode integracije
                                    {
                                       double s, dx, x;
                                    // za neparno n dodaje se +1 da bi interval bio paran
                                       if((imax/2)*2 != imax) {imax=imax+1;}
```

```
            s = 0.0;
            dx = (b-a)/static_cast<float>(imax);
            for (int i = 2; i <= imax-1; i = i+2)
            {
               x = a+static_cast<float>(i)*dx;
               s = s + 2.0*f(x) + 4.0*f(x+dx);
            }
            s = (s + f(a)+f(b)+4.0*f(a+dx) )*dx/3.0;
            return s;
         }

double fun1(double x) // definise funkciju 1
{
   return x;
}
double fun2(double x) // definise funkciju 2
{
   return 1 / x;
}
double fun3(double x) // definise funkciju 3
{
   return x * x;
}
int main()
{
   string str1 = "x";
   string str2 = "1/x";
   string str3 = "pow(x, 2)";
   double(*f)(double)  = 0;
   double a, b, lower, upper,  integralValue;
   int method, equation, imax;
   const int PICK_EQUATION = 4;
   const int MAXMETHODS = 2;

      cout << "Kucanjem broja izaberite funkciju za integraljenje:\n";
      cout << "1: " << str1 << " \n";
      cout << "2: " << str2 << " \n";
      cout << "3: " << str3 << " \n";
      cout << "4: Exit this menu\n";
      cin >> equation;
      if (equation == 1) // govori programu da ukljuci funkciju #1
         f = fun1;
      else if (equation == 2) // govori programu da ukljuci funkciju #2
         f = fun2;
      else              // govori programu da ukljuci funkciju #3
         f = fun3;

      cout << "Izaberite metod integracije:\n";
      cout << "1: Trapezoidal Rule" << endl; // govori programu da ukljuci
trapezoidalni metod integracije
      cout << "2: Simpson's Method" << endl; // govori programu da ukljuci
simpsonov metod integracije
      cin >> method;
```

```
        getlimit(lower, upper, imax);

        if(method <= MAXMETHODS)
                {
                switch(method)
                      {
                      case 1:
                         integralValue = trapezoid(f, lower,upper, imax);
                         break;
                      case 2:
                         integralValue = simpson(f, lower, upper, imax);
                         break;
                      }
                   cout << "Vrednost INTEGRALA je: " << integralValue << endl;
                }



        return 0;
    }

    void getlimit(double& lower, double& upper, int& imax)
    {
        cout << "Unesite donju granicu odredjenog integrala: \n"; // takes the users input
for lower bound
        cin >> lower;
        cout << "Unesite gornju granicu odredjenog integrala: \n"; // takes the users input
for upper bound
        cin >> upper;
        cout << "Unesite max. broj iteracija: \n"; // takes the users input for maximum
number of iterations
        cin >> imax;

        return;
    }
```



RAD SA STRINGOM

ASCII string

Do sada smo radili sa nizovima znakova u obliku tekstualnih konstanti koje smo zvali literalni string ("Hello World!"). Cilj nam je da nizove znakova tretiramo kao varijable. Na taj način moći ćemo vršiti obradu tekstualnih zapisa.
U C++ jeziku se koriste dva načina rada sa stringovima.

• Prvi način je da se string tretira kao niz znakova kojem poslednji znak mora biti jednak nuli. Ovakovi stringovi se nazivaji i ASCIIZ stringovi (ASCII +zero). U standardnoj biblioteci C jezika postoji više funkcija za rad s ovakovim stringovima, a deklarisane su u datoteci <string.h> odnosno <cstring>.

• Drugi je način, standardiziran u C++ jeziku, da se koristi klasa string koja je deklarisana u datoteci <string>.

Najpre ćemo upoznati kako se manipuliše stringom u C jeziku. Funkcije C- jezika tretiraju string kao za memorijski objekt koji sadrži niz znakova, uz uslov da poslednji znak u nizu mora biti jednak nuli. Nula na kraju stringa se koristi kao oznaka kraja stringa. Naprimer, u stringu koji sadrži tekst: Hello, World!, indeks nultog znaka je 13, što je jednako dužini stringa. Ovaj string u memoriji zauzima 14 bajta.

0	1	2	3	4	5	6	7	8	9	0	1	2	3
'H'	'e'	'l'	'l'	'o'	', '	' '	'W'	'o'	'r'	'l'	'd'	'!'	'\0'

Indeks nultog znaka je upravo jednak broju znakova u stringu.
Prema ovoj definiciji ASCIIZ string je svaki niz koji se deklariše kao:

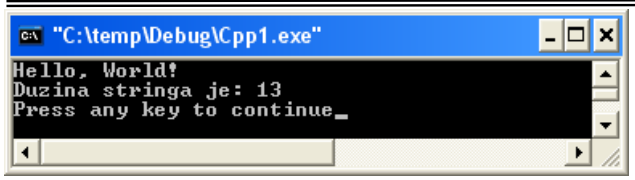
• niz znakova (npr. char str[10];) , ili kao
• pokazivač na znak (char *str;),

pod uslovom da se pri inicijalizaciji niza, i kasnije u radu s nizom uvek vodi računa o tome da polednji element niza mora biti jednak nuli.

Pogledajmo primer u kojem se string tretira kao obični niz znakova.

```
//Prvi C++ program - drugi put.
#include <iostream>
#include <cstring> // deklaracija funkcije strlen
using namespace std;
int main()
{
char hello[14] = { 'H', 'e', 'l', 'l', 'o', ',', ' ', 'W', 'o', 'r', 'l', 'd', '!', '\0' };
cout << hello << endl;
cout << "Duzina stringa je: " << strlen(hello) << endl;
return 0;
}
```

Posle izvršenja programa, dobije se poruka:



Ovaj program vrši istu funkciju kao i naš prvi C-program, štampa poruku: Hello, World!. Prvo je definisana i inicijalizovana varijabla hello. Ona je tipa znakovnog niza od 14 elemenata. Inicijalizacijom se u prvih 13 elemenata upisuju znakovi (Hello World), poslednji element se inicira na nultu vrednost.
Ispis ove varijable : Za ispis dužine stringa korišćena je standardna funkcija size_t strlen(char *s); koja vraća vrednost dužine stringa. Kao argument funkcija prihvata vrednost pokazivača na char, odnosno adresu početnog elementa stringa. (size_t je sinonim za unsigned).
Funkcija strlen se može implementirati na sledeći način:

1.verzija:

```
unsigned strlen(char s[])
{
unsigned i=0;
while (s[i] != '\0')          // petlja se prekida kada je s[i]==0, inače
i++;                          // inkrementira se brojač znakova, koji na kraju
return i;                     // sadrži duzinu stringa (bez nultog znaka)
}
```

2.verzija: pomoću pokazivača

```
unsigned strlen(char *s)
{
unsigned len = 0;
while (*s++ != '\0')          // petlja se prekida kada je *s==0, inače
len++;                        // inkrementira se brojač znakova i pokazivač
return len;                   // len sadrži duljinu stringa (bez nultog znaka)
}
```

String se može inicijalizirati navođenjem znakova od kojih se sastoji, kao u prethodnom primeru, ili pomoću literalne konstante:

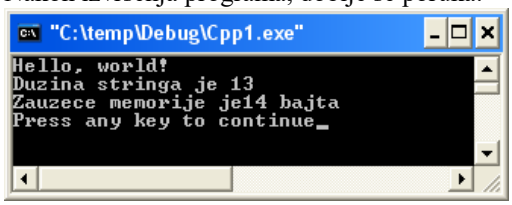
```
char hello[] = "Hello, World!"; // kompajler rezerviđe mesto u memoriji
```

Primer:

```
//Prvi C++ program - drugi put.
#include <iostream>
using namespace std;
unsigned strlen(char *s)
{
unsigned len = 0;
while (*s++ != '\0') // petlja se prekida kada je *s==0, inače
len++;               // inkrementira se brojač znakova i pokazivač
return len;          // len sadrži duzinu stringa (bez nultog znaka)
}
```

```
int main()
{
char hello[] = "Hello, world!";
cout << hello << endl;
cout << "Duzina stringa je " << strlen(hello) << endl;
cout << "Zauzece memorije je" << sizeof(hello)*sizeof(char)
<< " bajta" << endl;
return 0;
}
```

Nakon izvršenja programa, dobije se poruka:



Elementi stringa se mogu menjati naredbom dodele vrednosti, naprimer:

```
hello[0] = 'h';
hello[6] = 'w';
```
menjaju sadržaj stringa u "hello world";

Ako se procenjuje da će trebati više mesta za string, nego što se to navodi inicijalnim literalnim stringom, tada treba eksplicitno navesti dimenziju stringa. Deklaracijom,

```
char hello[50] = "Hello, World!";
```

kompajler rezerviše 50 mesta u memoriji i inicira prvih 14 elemenata na "Hello, World!", zaključno s nultim znakom.
Dozvoljeno je literalni string koristiti kao referencu niza:

```
cout << "0123456789ABCDEF"[n]; <=> char digits[] ="0123456789ABCDEF";
cout << digits[n];
```

Sledeća dva primera pokazuju rad s stringovima. Biće opisano kako se mogu napisati standardne funkcije za kopiranje stringa (strcpy) i upoređenje dva stringa (strcmp).

Funkcija strcpy kopira znakove stringa src u string dest, a vraća adresu od dest.

1. Verzija sa nizovima:

```
char *strcpy( char dst[], char src[])
{
int i;
for (i = 0; src[i] != '\0'; i++)
dst[i] = src[i];
dst[i] = '\0';
return dst;
}
```

2. Verzija sa pokazivačkim varijablama

```
char *strcpy(char *dest, const char *src)
{
char *dp = dest;
while((*dp++ = *src++) != '\0') // kopira se i '\0'
; /*prazna naredba*/
return dest;
}
```

Ova verzija sa inkrementiranjem pokazivača će se brže izvršavati od verzije koja je zapisana u indeksnoj notaciji.
Funkcija strcmp služi za upoređenje dva stringa s1 i s2. Deklarisana je s:

```
int strcmp(const char *s1, const char *s2)
```

Funkcija vraća vrednost 0 ako je sadržaj oba stringa isti, negativnu vrednost ako je s1 leksički manji od s2, ili pozitivnu vrednost ako je s1 leksički veći od s2. Leksičko poređenje se izvršava znak po znak prema kodnoj vrednosti ASCII standarda.

```
/*1. verzija*/
int strcmp( char s1[], char s2[])
{
int i = 0;
while(s1[i] == s2[i] && s1[i] != '\0')
i++;
if (s1[i] < s2[i])
return -1;
else if (s1[i] > s2[i])
return +1;
else
return 0;
}
```

Najprije se u for petlji poređuje da li su znakovi sa istim indeksom isti i da li je dostignut kraj niza (znak '\0'). Kad petlja završi, ako su oba znaka jednaka, to znači da su i svi prethodni znakovi jednaki. U tom slučaju funkcija vraća vrednost 0. U suprotnom funkcija vraća 1 ili –1 zavisno od numeričkog koda znakova koji se razlikuju.

```
/*2. verzija*/
int strcmp(char *s1, char *s2)
{
for(; *s1 == *s2; s1++, s2++)
{
if(*s1 == '\0')
return 0;
}
return *s1 - *s2;
}
```

Verzija s pokazivačima predstavlja optimiranu verziju prethodne funkcije. Uočite da se u **for** petlji ne koristi izraz inicijalizacije petlje. Najpre se poređuju znakovi na koje pokazuju s1 i s2. Ako su znakovi isti inkrementira se vrednost oba pokazivača. Telo petlje će

se izvršiti samo u slučaju ako su stringovi isti (tada s1 i s2 konačno pokazuju na znak '\0'). U suprotnom petlja se prekida, a funkcija vraća numeričku razliku ASCII koda znakova koji se razlikuju.

Napomena: verzije s inkrementiranjem pokazivača često rezultiraju bržim izvršenjem programa, ali predstavljaju programe koje je nešto teže razumeti od verzija sa indeksiranim nizovima.

Standardne funkcije za rad s ASCIIZ stringovima

U standardnoj biblioteci postoji niz funkcija za manipulisanje sa stringovima . One su deklarisane u datoteci <cstring> ili <string.h>. Funkcija im je:

size_t strlen(const char *s)
Vraća dužinu stringa s.

char *strcpy(char *s, const char *t)
Kopira string t u string s, uključujući '\0'; vraća s.

char *strncpy(char *s, const char *t, size_t n)
Kopira najviše n znakova stringa t u s; vraća s. Dopunjava string s sa '\0' znakovima ako t ima manje od n znakova.

char *strcat(char *s, const char *t)
Dodaje string t na kraj stringa s; vraća s.

char *strncat(char *s, const char *t, size_t n)
Dodaje najviše n znakova stringa t na string s, i znak '\0'; vraća s.

int strcmp(const char *s, const char *t)
Upoređuje string s sa stringom t, vraća <0 ako je s<t, 0 ako je s==t, ili >0 ako je s>t. Poređenje je leksikografsko, prema ASCII "abecedi".

int strncmp(const char *s, const char *t, size_t n)
Upoređuje najviše n znakova stringa s sa stringom t; vraća <0 ako je s<t, 0 ako je s==t, ili >0 ako je s>t.

char *strchr(const char *s, int c)
Vraća pokazivač na prvu pojavnost znaka c u stringu s, ili NULL znak ako c nije sadržan u stringu s.

char *strrchr(const char *s, int c)
Vraća pokazivač na zadnju pojavnost znaka c u stringu s, ili NULL znak ako c nije sadržan u stringu s.

char *strstr(const char *s, const char *t)
Vraća pokazivač na prvu pojavu stringa t u stringu s, ili NULL ako string s ne sadrži string t.

size_t strspn(const char *s, const char *t)
Vraća dužinu prefiksa stringa s koji sadrži znakove koji čine string t.

size_t strcspn(const char *s, const char *t)
Vraća dužinu prefiksa stringa s koji sadrži znakove koji nisu prisutni u stringu t.

char *strpbrk(const char *s, const char *t)
Vraća pokazivač na prvu pojavu bilo kojeg znaka iz string t u stringu s, ili NULL ako nije prisutan ni jedan znak iz string t u stringu s.

char *strerror(int n)
Vraća pokazivač na string kojeg interno generiše kompajler za dojavu greške u nekim sistemskim operacijama. Argument je obično globalna varijabla errno, čiju vrednost takođe postavlja kompajler pri sistemskim operacijama.

char *strtok(char *s, const char *sep)
strtok je funkcija kojom se može izvršiti razlaganje stringa na niz leksema koji su razdvojeni znakovima-separatorima. Skup znakova-separatora se zadaje u stringu sep. Funkcija vraća pokazivač na leksem ili NULL ako nema leksema.
Korištenje funkcije strtok je specifično jer u strigu može biti više leksema, a ona vraća pokazivač na jedan leksem. Da bi se dobili sedeći leksemi treba ponovo zvati istu funkciju, ali sa prvim argumentom jednakim NULL. Primeri, za string
char * s = "Prvi drugi,treci";
ako odaberemo znakove separatore: razmak, tab i zarez, tada sledeći iskazi daju ispis tri leksema (Prvi drugi i treci):

```
char *leksem = strtoken(s, " ,\t"); /* dobavlja prvi leksem */
while( leksem != NULL) {          /* ukoliko postoji */
printf("", leksem);               /* ispiši ga i */
lexem = strtok(NULL, " ,\t");     /* dobavi sledeći leksem */
}                                 /* pa ponovi postupak */
```

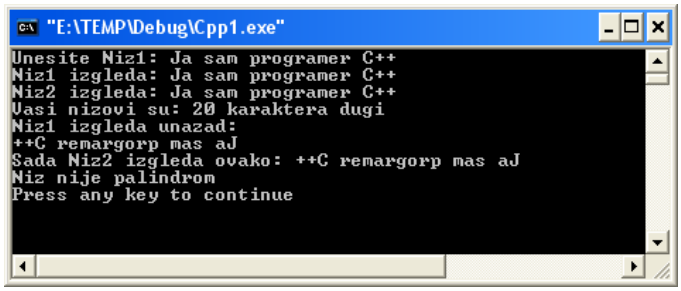
1 Zadatak:

Sastaviti program na C++ jeziku za unos skupa karaktera ”anavolimilovana” u promenljivu Niz1, pa ga onda iskopirati u promenljivu Niz2 korišćenjem funkcije strcpu. Odštampati sadržaj unete promenljive Niz 1, programski realizovane promenljive Niz 2 i odrediti njihovu dužinu primenom funkcije strlen. Dalje odštampati kako izgleda Niz1 unazad (od zadnjeg do prvog karaktera) pa takav niz smestiti u promenljivu Niz2. Odštampati kako sada izgleda promenljiva Niz2.

Opciono - primenom funkcije strcmp(Niz1,Niz2) utvrditi da li su sada ovako dobijeni nizovi palindromi?

```
#include<iostream.h>
#include<string.h>
int main()
{
int i;
char Niz1[40];
char Niz2[40];
cout << "Unesite Niz1: ";
cin.get(Niz1,39); //uzima do 39 karaktera ili novog reda - return
strcpy(Niz2,Niz1);
```

```
cout << "Niz1 izgleda: " << Niz1 << endl;
cout << "Niz2 izgleda: " << Niz2 << endl;
cout << "Vasi nizovi su: " << strlen(Niz1) << " karaktera dugi " << endl;
cout << "Niz1 izgleda unazad: " << endl;
for(i=strlen(Niz1)-1;i>=0;i--)
{cout<<Niz1[i];
Niz2[strlen(Niz1)-i-1]=Niz1[i];
Niz2[strlen(Niz1)]='\0';}
cout<<endl;
cout << "Sada Niz2 izgleda ovako: " << Niz2 << endl;
if(!strcmp(Niz1, Niz2)) cout<<"Niz2 je palindrom sa Niz1" <<endl;
else cout<<"Niz nije palindrom\n";
return 0;
}
```



Mnogi programi koriste jedinstvena imena za sve entitiete što može da izazove velike probleme. To je istina, kada se aplikacija paralelno razvijaa sa strane nekoliko programera, ili kada distributer hoće da produkuje biblioteku koja se odnosi na te programe. Mehanizam koji može pomoći da se ovi problemi reše sastoji se u upotrebi funkcije **nemespace**, u obliku:

using namespace std, ili sa drugim atributima;

NAPRIMER:

```
// Using a namespace
namespace myRegion
{
.....
.....
}

// Using a namespace
namespace data
{
  extern const double pi = 3.14159265;
  extern const int primes[] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31};
}
```

Ovde smo definisali pi i primes[] u okviru namespace data. Možemo koristiti ove promenljive i u drugoj programskoj jedinici koja sada sadrži:

```
// Using a namespace
#include <iostream>
```

```
namespace data
{
  extern const double pi; // Variable su definisane u drugom fajlu
  extern const int primes[]; // Array je definisano u drugom fajlu
}

int main()
{
  std::cout << std::endl
            << "pi ima vrednost "
            << data::pi << std::endl;  //Posmatra se kao konstruktor
  std::cout << "Četvrti prime broj je "
            << data::primes[3] << std::endl;
  return 0;
}
```

Prvo moramo postaviti relaciju imezmeđu naredbe ***namespaces*** i načina na koji će se uključivati hederski fajlovi. Pre nego što je standardizovan novi stil hederskog fajla **<iostream>** (gde nema ekstenzije '**.h**'), tipični način uključivanja hederskog fajla bio je sa ekstenzijom '**.h**', kao naprimer **<iostream.h>**. U tom slučaju, ***namespaces*** nije deo predeprocesorskih direktiva. Na taj način se može utvrditi analogija između ova dva vida pisanja koda, pa ako napišemo:

```
#include <iostream.h>
```

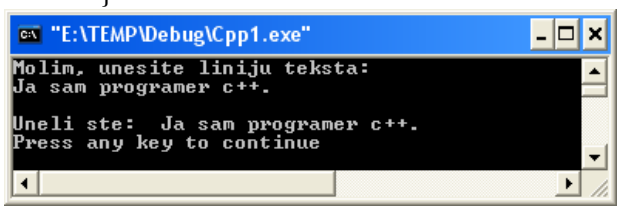
to znači

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

Dakle da bi koristili deklaraciju promenljive strings uključićemo hedersku datoteku <string>. Klasa string je definisana sa namespace std tako da je korišćenje ove direktive neophodno.

2 Zadatak:

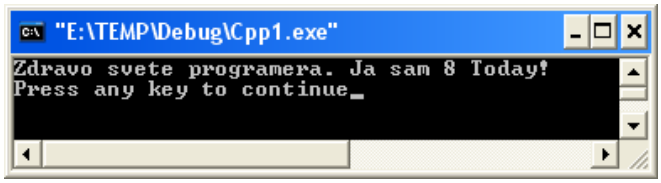
```
#include<iostream>
#include<string>
using namespace std;
                int main()
        {       cout << "Molim, unesite liniju teksta:\n";
                string s;
                getline(cin,s);
                cout << "Uneli ste:  " << s << '\n';
                return 0;
        }
```



3 Zadatak:

Prva dva stringa, s1 i s2, startuju prazni, dok s3 i s4 prikazuju dva ekvivalentna nacina da inicijalizuju objekat stringa iz polja karaktera.

```
#include <string>
#include <iostream>
using namespace std;
int main() {
string s1, s2; // Empty strings
string s3 = "Zdravo svete programera."; // Initialized
string s4("Ja sam"); // Also initialized
s2 = "Today"; // Assigning to a string
s1 = s3 + " " + s4; // Combining strings
s1 += " 8 "; // Appending to a string
cout << s1 + s2 + "!" << endl;
}
```



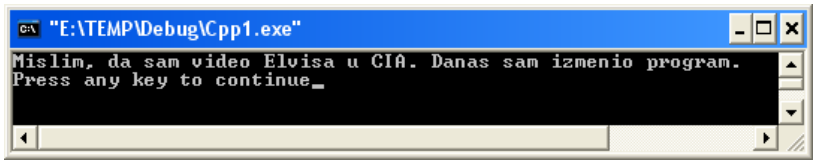
4 Zadatak:

Primer zamene karaktera u stringu - complete demonstration replace( )

```
#include <string>
#include <iostream>
using namespace std;
void replaceChars(string& modifyMe,string findMe, string newChars)
{
// Look in modifyMe for the "find string"
// starting at position 0
int i = modifyMe.find(findMe, 0);
// Did we find the string to replace?
if(i != string::npos)
// Replace the find string with newChars
modifyMe.replace(i,newChars.size(),newChars);
}

int main()
{
string bigNews =
"Mislim, da sam video Elvisa u UFO. "
"Danas sam izmenio program.";
string replacement("CIA");
string findMe("UFO");
// Find "UFO" in bigNews and overwrite it:
```

```
replaceChars(bigNews, findMe, replacement);
cout << bigNews << endl;
}
```

IZLAZ:

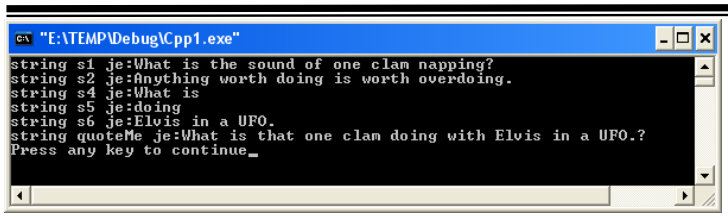


5 Zadatak:

Najprostija forma inicijalizacije stringa, koja nudi varijacije koje nude fleksibilnost i bolju kontrolu.. Mozemo :

- » Koristiti ma koji deo C char polja ili C++ stringa
- » Kombinovati razlicite delove inicijalizacije podataka operatorom +
- » Koristiti string objekta substr( ) funkciju clanicu za kreiranjea substringa

```
#include <string>
#include <iostream>
using namespace std;
int main() {
string s1
("What is the sound of one clam napping?");
cout<<"string s1 je:"<<s1<<endl;
string s2
("Anything worth doing is worth overdoing.");
string s3("I saw Elvis in a UFO.");
cout<<"string s2 je:"<<s2<<endl;
// Copy the first 8 chars
string s4(s1, 0, 8);
cout<<"string s4 je:"<<s4<<endl;
// Copy 6 chars from the middle of the source
string s5(s2, 15, 6);
cout<<"string s5 je:"<<s5<<endl;
// Copy from middle to end
string s6(s3, 6, 15);
cout<<"string s6 je:"<<s6<<endl;
// Copy all sorts of stuff
string quoteMe = s4 + "that" +
// substr() copies 10 chars at element 20
s1.substr(20, 10) + s5 +
// substr() copies up to either 100 char
// or eos starting at element 5
"with" + s3.substr(5, 100) +
// OK to copy a single char this way
s1.substr(37, 1);
cout<<"string quoteMe je:"<<quoteMe<<endl;
}
```

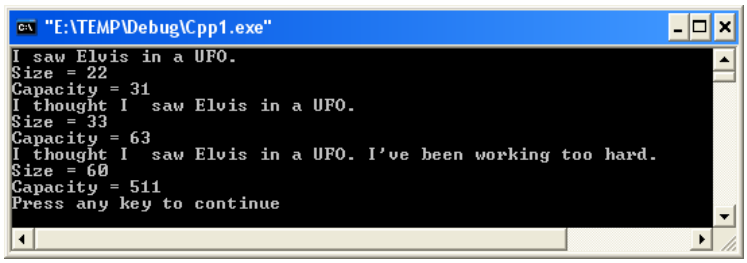


6 Zadatak:

Slično kao predhodni primer: C++ za string obezbedjuje nekoliko alata za monitoring i upravljanje njegove veličine. Tako size( ), resize( ),capacity( ), i reserve( ) funkcije članice mogu biti vrlo korisne za rad sa ovakvim podacima.

```
#include <string>
#include <iostream>
using namespace std;
int main() {
string bigNews("I saw Elvis in a UFO. ");
cout << bigNews << endl;
// How much data have we actually got?
cout << "Size = " << bigNews.size() << endl;
// How much can we store without reallocating
cout << "Capacity = "<< bigNews.capacity() << endl;
// Insert this string in bigNews immediately
// following bigNews[1]
bigNews.insert(1, " thought I ");
cout << bigNews << endl;
cout << "Size = " << bigNews.size() << endl;
cout << "Capacity = "<< bigNews.capacity() << endl;
// Make sure that there will be this much space
bigNews.reserve(500);
// Add this to the end of the string
bigNews.append("I've been working too hard.");
cout << bigNews << endl;
cout << "Size = " << bigNews.size() << endl;
cout << "Capacity = "<< bigNews.capacity() << endl;
}
```

IZLAZ:

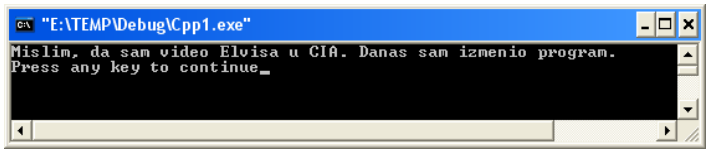


7 Zadatak:

Ovom prilikom proveravamo da li nesto postoji, pre nego izvrsimo replace( ).

```
#include <string>
#include <iostream>
using namespace std;
void replaceChars(string& modifyMe,string findMe, string newChars)
{// Look in modifyMe for the "find string"
// starting at position 0
int i = modifyMe.find(findMe, 0);
// Did we find the string to replace?
if(i != string::npos)
// Replace the find string with newChars
modifyMe.replace(i,newChars.size(),newChars);}

int main()
{
string bigNews =
"Mislim, da sam video Elvisa u UFO. "
"Danas sam izmenio program.";
string replacement("CIA");
string findMe("UFO");
// Find "UFO" in bigNews and overwrite it:
replaceChars(bigNews, findMe, replacement);
cout << bigNews << endl;}
```

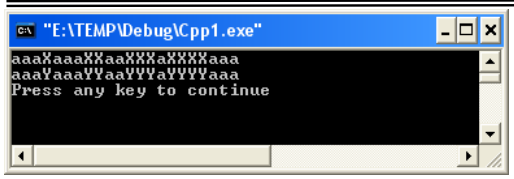
IZLAZ:



8 Zadatak:

Jednostavna zamena karaktera koriscenjem STL replace( ) algoritma

```
#include <string>
#include <algorithm>
#include <iostream>
using namespace std;
int main() {
string s("aaaXaaaXXaaXXXaXXXXaaa");
cout << s << endl;
replace(s.begin(), s.end(), 'X', 'Y');
cout << s << endl;
}
```

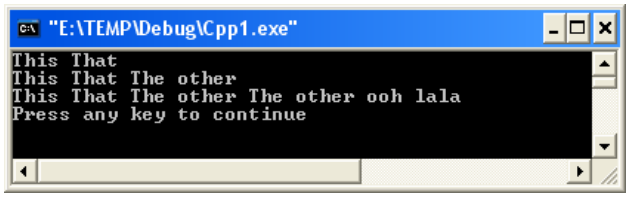


9 Zadatak:

Koriscenje operatora + i += za sabiranje stringova.

```
#include <string>
#include <iostream>
using namespace std;
int main() {
string s1("This ");
string s2("That ");
string s3("The other ");
// operator+ concatenates strings
s1 = s1 + s2;
cout << s1 << endl;
// Another way to concatenates strings
s1 += s3;
cout << s1 << endl;
// You can index the string on the right
s1 += s3 + s3[4] + "oh lala";
cout << s1 << endl;
}
```

IZLAZ:



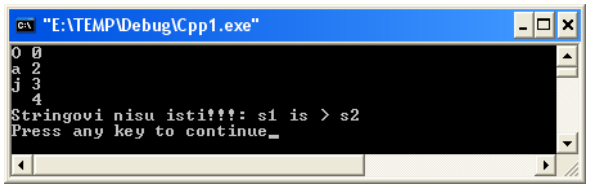
10 Zadatak:

C++ obezbedjuje nekoliko nacina za uporedjenje stringova. Najjednostavnije je koristiti funkcijeske operatore ==, operator != operator >, operator <, operator >=, i operator <=.

```
#include <string>
#include <iostream>
using namespace std;
int main() {
// Strings to compare
string s1("Ovaj ");
string s2("Onaj ");
```

```
for(int i = 0; i< s1.size() && i < s2.size(); i++)
// See if the string elements are the same:
if(s1[i] == s2[i])
cout << s1[i] << " " << i << endl;
// Use the string inequality operators
if(s1 != s2)
        {
cout << "Stringovi nisu isti!!!:" << " ";
if(s1 > s2)
cout << "s1 is > s2" << endl;
else
cout << "s2 is > s1" << endl;
        }
}
```

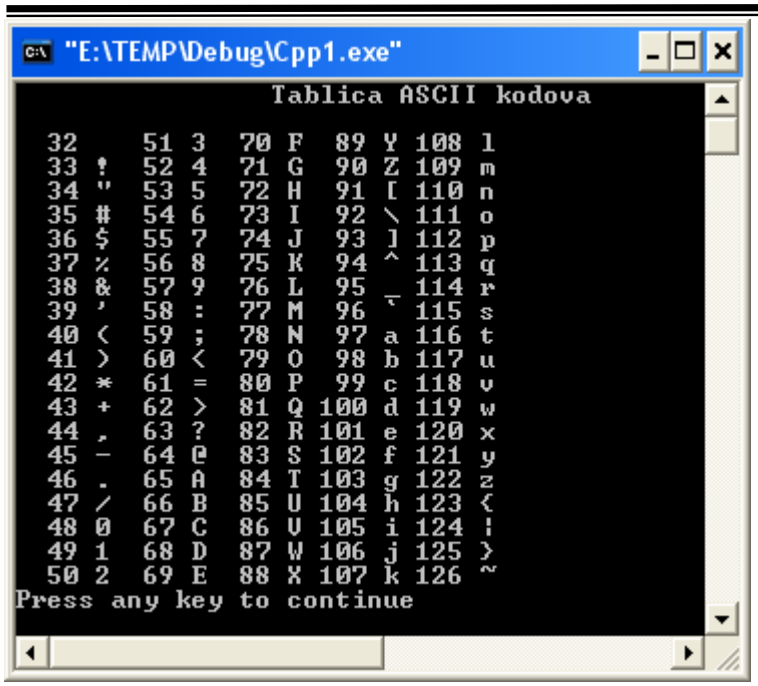
IZLAZ:



11 Zadatak:

Program za prikazivanje tablice ASCII kodova:

```
#include <iostream.h>
#include <iomanip.h>
main()
{
        char c=' ';
        int i;

        cout<<"\t\tTablica ASCII kodova \n \n";
linija:
        i=0;
znak:
        cout<<setw(4)<<(int)(c+i)<<" "<<setw(1)<<(char)(c+i);
        i=i+19;
        if (i<95) goto znak;
        cout<<"\n";
        c=c+1;
        if (c<' '+19) goto linija;
}
```

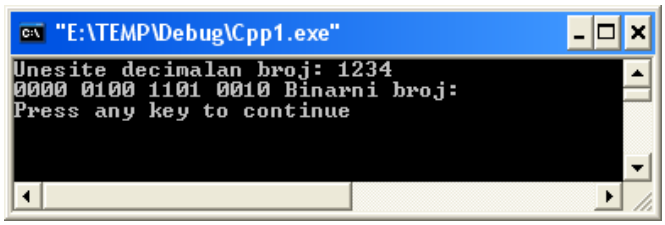


12 Zadatak:

Sastaviti program na programskom jeziku C++ koji učitava decimalan pozitivan celi broj u obliku niza znakova i ispisuje njegovu vrednost u binarnom obliku. Pretpostaviti da se za interno predstavljanje celih brojeva koristi 16 bitova.

```
#include <iostream.h>
#include <iomanip.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
main()
{
        char dec[10];
        short int bin,i;
        cout<<"Unesite decimalan broj: ";
cin>>dec;       /* ucitava string sa standardnog ulaza (dec=&dec[0]) */
                /*      atoi vraca 0 ukoliko nije uspela konverzija
                        Ukoliko je strlen(dec)=0 onda se drugi deo USLOVA (iza &&
        operatora)              nece ni proveravati. Po postavci zadatka ocekuje se
        pozitivan broj,  i takav uslov se jednostavno, ovim putem, moze proveriti.      */
if(strlen(dec) && (bin=atoi(dec)))
        {
                cout<<"Binarni broj: ";
                i=-1;
                while (++i<16)
                {
                        putchar((bin & 0x8000) ? '1' : '0');
```

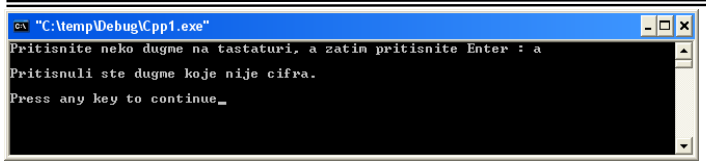
```
                    /*      0x8000 ima jedinicu na najvisem 15-om bitu
                            Gornja naredba ce ispisati 15-i bit broja bin!
                            Najpre ispisujemo najvise bitove, jer takav prikaz zelimo na
                            ekranu bit15 bit14 ... bit2 bit1 bit0   */
                    bin <<= 1 ;
/* bin = bin shl 1 ; pomeramo bin ulevo za 1 bit */
                            if (i%4 == 3)
                            putchar(' ');
/* prikaz razmaka izmedju svake polovine bajta */
                    }
          cout<<"\n";
          }
          else
                    cout<<"Neispravan broj ili nula\n";
}
```



13 Zadatak:

Napisati program koji odlučuje da li je pritisnuto dugme cifra.

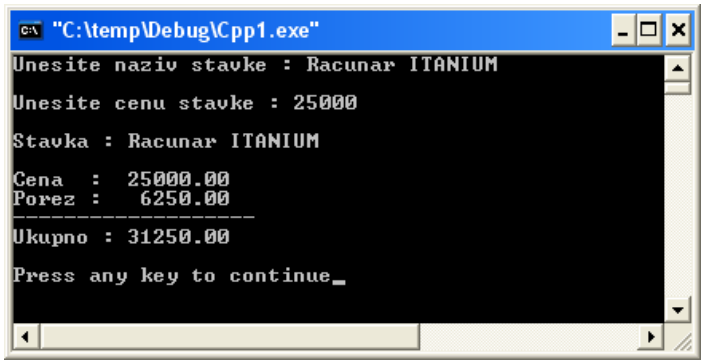
```
// Program prikazuje koriscenje slozene if naredbe
// koja odlucuje da li je pritisnutuo dugme broj.
// Program koristi cin.get() za unos znaka.
#include <iostream.h>

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;
}
```



14 Zadatak:

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

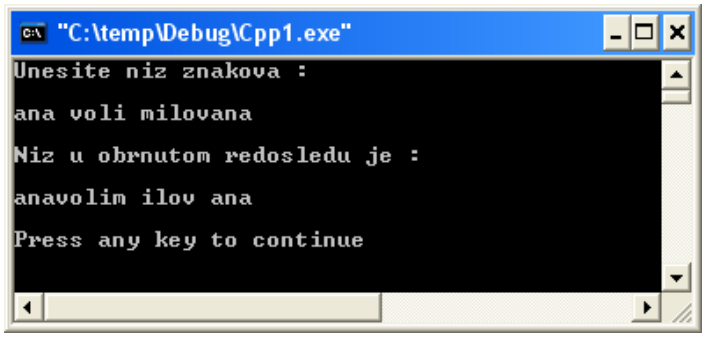
```
// program demonstrira koriscenje naredbe cin.getline()
// za unos stringa
#include <iostream.h>
#include <iomanip.h>
int main()
{
double const STOPA_POREZA = 0.25;
char naziv_stavke[51];
double cena,
porez,
ukupno;
cout << setprecision(2)
<< setiosflags(ios::fixed)
<< setiosflags(ios::showpoint);
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;
}
```



15 Zadatak:

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

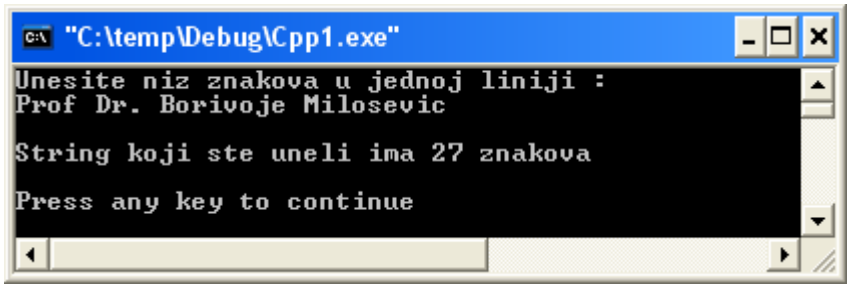
```
// Ovaj program zahteva unos stringa
// a zatim ga ispisuje i obrnutom redosledu.
#include <iostream.h>
int main()
{
char linija[81];
char* pokazivac_na_znak = linija;
cout << "Unesite niz znakova :" << endl << endl;
cin.getline(linija, 81);
// Pronalazenje kraja stringa
while ( *pokazivac_na_znak != '\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;
}
```



16 Zadatak:

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

```
// Program izracunava broj unetih znakova u jednoj liniji
// Koristi funkciju Duzina_Niza() za brojanje unetih znakova
#include <iostream.h>
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;
}
```

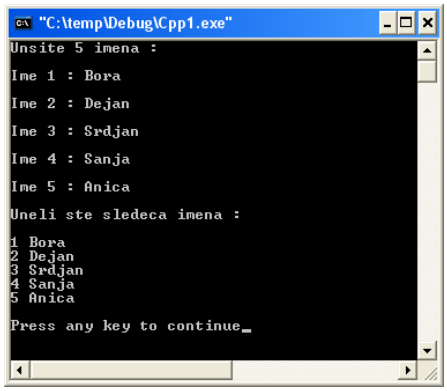


17 Zadatak:

Napisati program koji unetom nizu dinamički dodeljuje memoriju.

```
// Program ilustruje upotrebu naredbi new i delete
// za dinamicku dodelu memorije nizu.
#include <iostream.h>
#include <stdlib.h>
#include <string.h>
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;
}
```



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.

F U N K C I J E

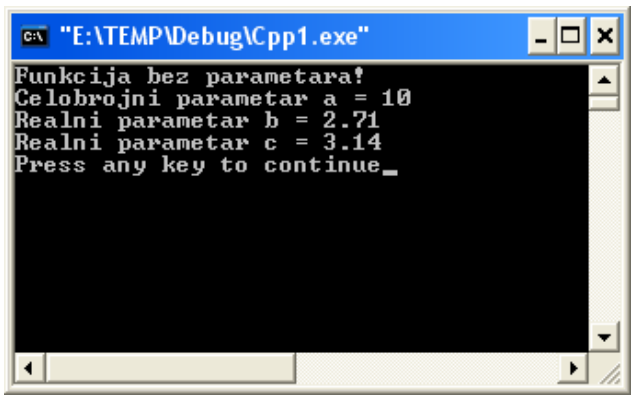
Ima mnogo razloga za upotrebu funkcija:

a.) Deo koda može biti iskorišćen mnogo puta u različitim delovima programa.
b.) Program se deli na niz blokova. Svaki blok će raditi specijalni posao. Razumevanje i dizajn programa biće na ovaj način pojednostavljeno.
c.) Blok koda se može izvršavati sa različitim brojem inicijalnih parametara. Ti parametri su postavljeni u funkciji sa argumentima.
d.) C++ omogucuje da više funkcija ima isti naziv, pod uslovom da su im liste parametara različite.
Prema listi parametara kod poziva funkcije odreduje se koja će funkcija biti pozvana.

1 Zadatak:

Kako radi funkcija bez parametara?

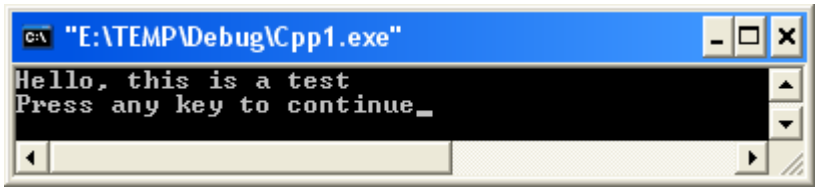
```
#include <iostream.h>
void funkcija()
{cout << "Funkcija bez parametara!" << endl;
}
void funkcija(int a)
{cout << "Celobrojni parametar a = " << a << endl;
}
void funkcija(float b, float c)
{cout << "Realni parametar b = " << b << endl;
cout << "Realni parametar c = " << c << endl;
}

void main()
{
funkcija();
funkcija(10);
funkcija(2.71,3.14);
}
```



2 Zadatak:

Pokazati kako funkcioniše prototip funkcije.

```
#include<iostream.h>
myfunc();/*inicijalizacija funkcije bez argumenata - prototip*/
main()
{
  myfunc();/*pozivanje funkcije bez argumenata*/
}
                        myfunc()/*definisanje funkcije bez argumenata*/
{
  cout<<"Hello, this is a test\n";
}
```



3 Zadatak:

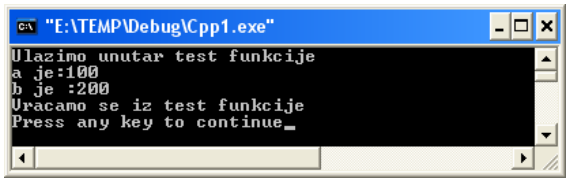
Korišćenje prototipa funkcije sa argumentom.

```
#include <iostream.h>
sayhello(int count);
main()
{sayhello(4);}
sayhello(int count)
{int c;
for(c=0;c<count;c++)
  cout<<"\nHello";}
```

4 Zadatak:

Test funkcije

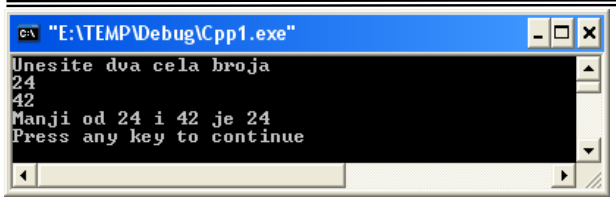
```
#include<iostream.h>
void test();
int main()
{
cout<<"Ulazimo unutar test funkcije\n";
test();
cout<<"Vracamo se iz test funkcije\n";
return 0;
}

void test()
{int a,b;a=100;
b=a+100;
cout<<"a je:"<<a<<"\nb je :"<<b<<"\n";}
```



5 Zadatak:

Naći minimum od dva cela broja. U glavnom programu obezbediti štampanje, a u funkciji njihovo poređenje.

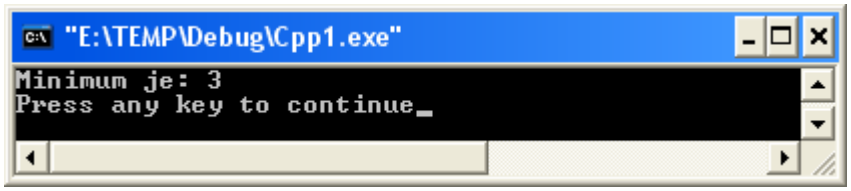
```
#include <iostream.h>
int imin(int n,int m);
int main()/* glavni program */
{
        int broj1,broj2;
        cout<<"Unesite dva cela broja\n";
        cin>>broj1>>broj2;
        cout<<"Manji od "<< broj1<<" i "<< broj2<<" je "<< imin(broj1,broj2)<<"\n";
        return 0;}

int imin(int n,int m)/* potprogram */
{
        int min;
        if (n<m)
                min=n;
        else
                min=m;
        return min;}
```



6 Zadatak:

Isti kao 4. samo na drugi način.

```
#include <iostream.h>
int min(int a,int b);

main()
{
int m;
m=min(3,6);
cout<<"Minimum je: "<<m<<"\n";
return 0;
}

int min(int a, int b)
{
if(a<b)
 return a;
else
 return b;}
```

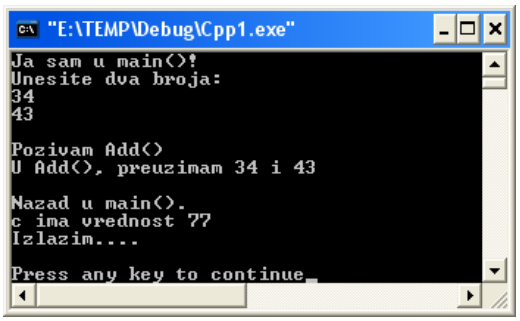


7 Zadatak:

Formiranje zbira preko funkcije

```
#include <iostream.h>
int Add (int x, int y);
int main()
{
cout << "Ja sam u main()!\n";
int a, b, c;
cout << "Unesite dva broja:\n";
cin >> a;
cin >> b;
cout << "\nPozivam Add()\n";
```

```
c=Add(a,b);
cout << "\nNazad u main().\n";
cout << "c ima vrednost " << c;
cout << "\nIzlazim....\n\n";
return 0;
}

int Add (int x, int y)
{
cout << "U Add(), preuzimam " << x << " i " << y << "\n";
return (x+y);
}
```

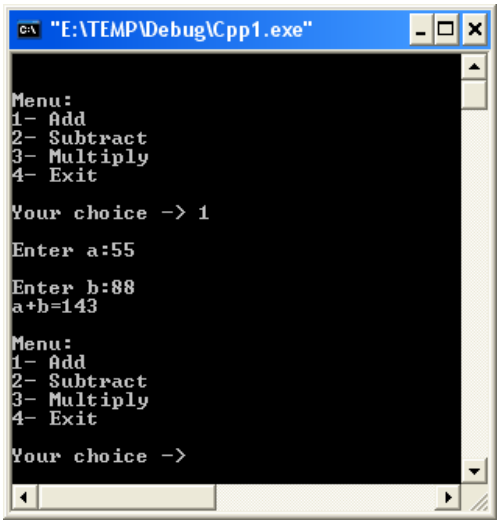


8 Zadatak:

Uraditi meni sa izborom operacija za sabiranje, oduzimanje i množenje dva cela realna broja. U funkciji izvesti opisane računske radnje.

```
#include <iostream.h>
#include <stdlib.h>
add();
subtract();
multiply();
main()
{
int choice;
while(1)
 {
 cout<<"\n\nMenu:\n";
 cout<<"1- Add\n2- Subtract\n";
 cout<<"3- Multiply\n4- Exit";
 cout<<"\n\nYour choice -> ";
 cin>>choice;
            switch(choice)
                    {    case 1 : add();
          break;
                        case 2 : subtract();
          break;
                        case 3 : multiply();
```

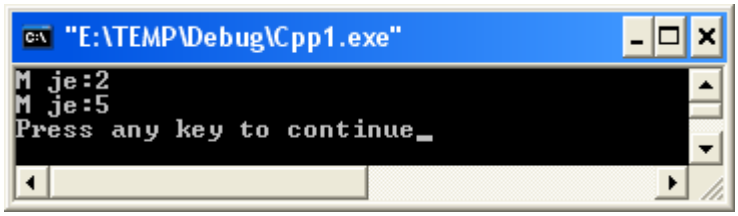
```
        break;
                case 4 : cout<<"\nProgram Ends. !";
        exit(0);

default:
        cout<<"\nInvalid choice";     }  }}

add()
{float a,b;
cout<<"\nEnter a:";
cin>>a;
cout<<"\nEnter b:";
cin>>b;
cout<<"a+b="<<a+b;}

subtract()
{float a,b;
cout<<"\nEnter a:";
cin>>a;
cout<<"\nEnter b:";
cin>>b;
cout<<"a-b="<<a-b;}

multiply()
{float a,b;
cout<<"\nEnter a:";
cin>>a;
cout<<"\nEnter b:";
cin>>b;
cout<<"a*b="<<a*b;}
```



8 Zadatak:

Pozivanje funkcije po vrednosti

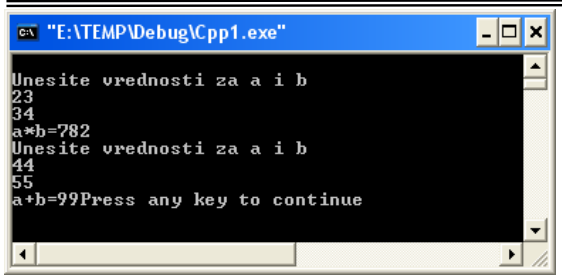
```
/*Pozivanje po vrednosti*/
#include <iostream.h>
int test(int a);
int main()
{int m,y;
m=2;
cout<<"M je:"<<m;
y=test(m);
cout<<"\nM je:"<<y<<endl;
return 0;}
int test(int a)
{a=5;
return a ;}
```



9 Zadatak:

MNOŽENJE I SABIRANJE DVA REALNA BROJA PREKO FUNKCIJE

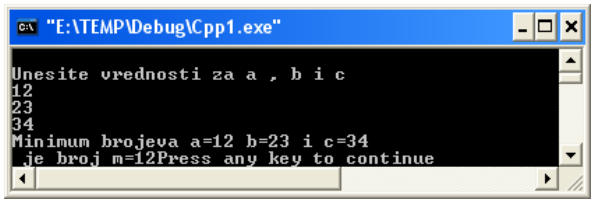
```
#include <iostream.h>
#include <math.h>
mno();
sab();
main()
{float a,b;
mno(); sab();}
mno()
{float a,b;
cout<<"\nUnesite vrednosti za a i b\n";
cin>>a>>b;
cout<<"a*b="<<a*b;}
sab()
{float a,b;
cout<<"\nUnesite vrednosti za a i b\n";
cin>>a>>b;
cout<<"a+b="<<a+b;}
```



10 Zadatak:

MINIMUM TRI CELA BROJA PREKO FUNKCIJE

```
#include <iostream.h>
#include <math.h>
min();
main()
{int a,b,c,m;
min();}
min()
{int a,b,c,m;
cout<<"\nUnesite vrednosti za a , b i c\n";
cin>>a>>b>>c;
m=a;
if(b<m)  m=b;
if(c<m)  m=c;
cout<<"Minimum brojeva a="<<a<<" b="<<b<<" i c="<<c<<"\n"
<<" je broj m="<<m;}
```

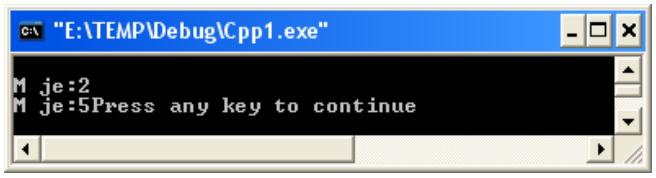


11 Zadatak:

Poziv funkcije salje vrednost promenljive m funkciji a ne salje promenljivu samu sebi: m=2, M=2

```
/*Pozivanje po referenci*/
#include <iostream.h>
void test(int *ptr);
main()
{
int m;
m=2;
cout<<"\nM je:"<<m;
```

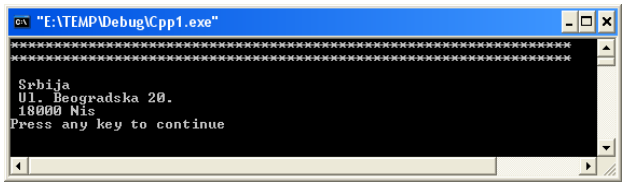
```
test(&m);
cout<<"\nM je:"<<m;
return 0;
}
void test(int *ptr)
{
*ptr=5;
}
/*Poziv funkcije salje vrednost promenljive m po referenci: m=2, M=5*/
```



12 Zadatak:

Poziv funkcije bez argumenta

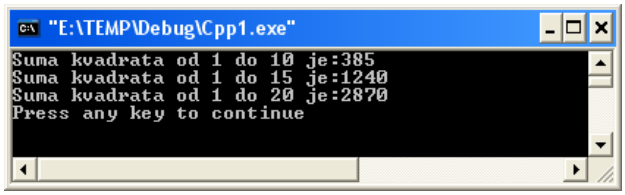
```
#include <iostream.h>
#include <stdio.h>
#define IME "Srbija"
#define ADRESA "Ul. Beogradska 20."
#define MESTO "18000 Nis"
#define LIMIT 65
void main()
{void zvezde(void);/*deklaracija funkcije bez argumenata*/
zvezde(); /*poziv korisnicke funkcije */
cout<<"\n "<<IME ; /*poziv funkcije iz standardne biblioteke*/
cout<<"\n "<<ADRESA;
cout<<"\n "<<MESTO<<"\n";
zvezde() ;} /*definicija korisnicke funkcije */
void zvezde() /*funkcija nema argumenata*/
{ int brojac;
for(brojac=1;brojac<=LIMIT;brojac++)
putchar ('*');
putchar ('\n') ; }
```



13 Zadatak:

Suma kvadrata preko funkcije

```
/*Program za izracunavanje sume
kvadrata celih brojeva od 1 do n*/
#include <iostream.h>
suma_kvadrata(int p);
main()
        {suma_kvadrata(10);
        suma_kvadrata(15);
        suma_kvadrata(20);
    return 0;}
suma_kvadrata(n)                        /*f ja za izracunavanje*/
        int n;                          /*sume kvadrata*/
        {int i;
        long suma=0;
        for (i=1; i<=n; suma+=(long)i*i, ++i);
        cout<<"Suma kvadrata od 1 do "<<n<<" je:"<<suma<<"\n";}
```

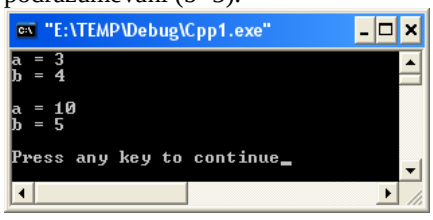


14 Zadatak:

Prilikom poziva funkcije potrebno je navesti listu argumenata koja odgovara listi argumenata u zaglavlju funkcije. C++ dopušta da se neki od argumenata u pozivu funkcije izostave, tako da se umesto njih koriste podrazumevane vrednosti.

```
#include <iostream.h>
void funkcija(int a, int b=5){
cout << "a = " << a << endl;
cout << "b = " << b << endl << endl;}

void main(){
funkcija (3,4);
funkcija (10);}
```

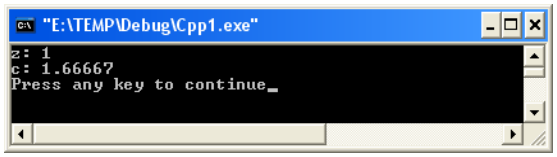
U drugom pozivu funkcije izostavljen je drugi argument, umesto kojeg se koristi podrazumevani (b=5).



15 Zadatak:

Preko dve funkcije prikazati operaciju deljenja i njen rezultat kada se u glavni program poziva kao INT tip i FLOAT tip.

```
#include <iostream.h>

 void intDiv(int x, int y)
 {
      int z = x / y;
      cout << "z: " << z << endl;
 }

 void floatDiv(int x, int y)
{
    float a = (float)x;                  // stari stil
    float b = static_cast<float>(y);    // napredni stil
    float c = a / b;

    cout << "c: " << c << endl;
}

 int main()
 {
     int x = 5, y = 3;
     intDiv(x,y);
     floatDiv(x,y);
     return 0;
 }
```

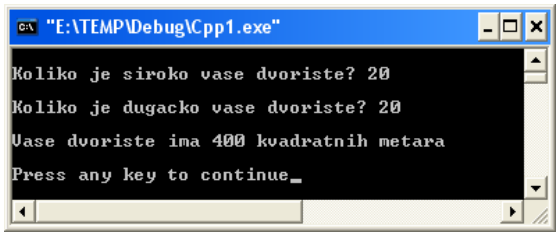


16 Zadatak:

Sastaviti program za izračunavanje površine dvorišta, kada se sirina i dužina dvorišta unose u glavnomprogramu a površina izračunava u funkciji.

```
#include <iostream.h>
   int Povrsina(int length, int width); //function prototype

   int main()
   {
     int lengthOfYard;
    int widthOfYard;
    int areaOfYard;
```

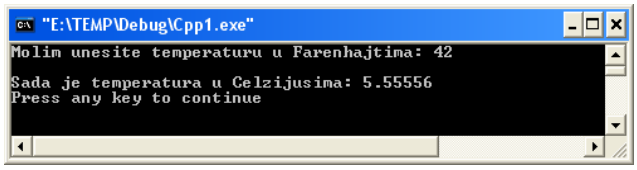
```
   cout << "\nKoliko je siroko vase dvoriste? ";
   cin >> widthOfYard;
   cout << "\nKoliko je dugacko vase dvoriste? ";
   cin >> lengthOfYard;

   areaOfYard= Povrsina (lengthOfYard,widthOfYard);

   cout << "\nVase dvoriste ima ";
   cout << areaOfYard;
   cout << " kvadratnih metara\n\n";
          return 0;
 }

 int Povrsina (int yardLength, int yardWidth)
 {
     return yardLength * yardWidth;
 }
```



17 Zadatak:

Sastaviti program za konverziju stepeni Farenhajta u Celzijus. Konverziju organizovati u funkciji.

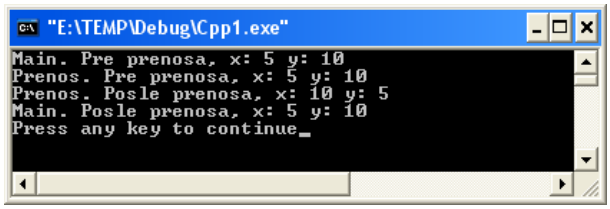
```
#include <iostream.h>

    float Konverzija(float);
    int main()
    {
       float TempFer;//Temperatura u Farenhajtima
       float TempCel;//Temperatura u Stepenima

       cout << "Molim unesite temperaturu u Farenhajtima: ";
      cin >> TempFer;
      TempCel = Konverzija(TempFer);
      cout << "\nSada je temperatura u Celzijusima: ";
      cout << TempCel << endl;
             return 0;
    }
    float Konverzija(float TempFer)
    {
      float TempCel;
      TempCel = ((TempFer - 32) * 5) / 9;
      return TempCel;    }
```



18 Zadatak:

Demonstracija prenosa promenljivih po vrednosti.

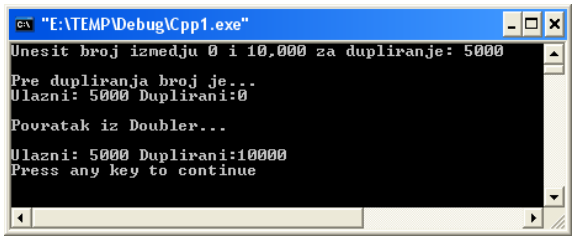
```
#include <iostream.h>

    void prenos(int x, int y);//Funkcijski prototip

    int main()
    {
      int x = 5, y = 10;

cout << "Main. Pre prenosa, x: " << x << " y: " << y << "\n";
      prenos(x,y);//Poziv funkcije za prenos vrednosti
cout << "Main. Posle prenosa, x: " << x << " y: " << y << "\n";
        return 0;
    }

    void prenos (int x, int y)//Funkcija
    {
      int temp;

cout << "Prenos. Pre prenosa, x: " << x << " y: " << y << "\n";

      temp = x;
      x = y;
      y = temp;

cout << "Prenos. Posle prenosa, x: " << x << " y: " << y << "\n";

    }
```



19 Zadatak:

Demonstraciju višeznačne naredbe return izvesti programom za dupliranje vrednosti promenljive.

```
#include <iostream.h>
    int Doubler(int IznosZaDupliranje);
    int main()
    {   int rezultat = 0;
        int input;
  cout << "Unesit broj izmedju 0 i 10,000 za dupliranje: ";
  cin >> input;
  cout << "\nPre dupliranja broj je... ";
cout << "\nUlazni: " << input << " Duplirani:" << rezultat << "\n";
        rezultat = Doubler(input);
        cout << "\nPovratak iz Doubler...\n";
cout << "\nUlazni: " << input << " Duplirani:" <<rezultat<< "\n";
        return 0; }

   int Doubler(int original)
   {
        if (original <= 10000)
            return original * 2;
        else
                        return -1;
        cout << "Ovo stampanje je nemoguce!\n";
   }
```

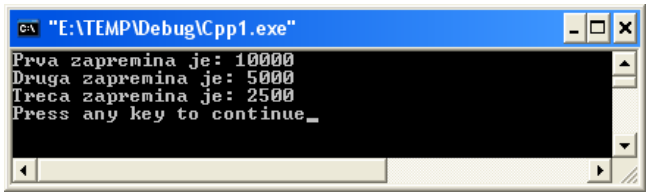


20 Zadatak:

Demonstracija korišćenja parametara po difoltu kada se rešava zapremina kocke u funkciji.

```
#include <iostream.h>

  int ZapreminaKocke(int length, int width = 25, int height = 1);
          /*Kada u programu nisu definisane vrednosti promenljivih se uzimaju po difoltu,
          onako kako su ovde specificirane*/
  int main()
  {
     int length = 100;
     int width = 50;
     int height = 2;
     int zapremina;
```

```
    zapremina = ZapreminaKocke(length, width, height);
    cout << "Prva zapremina je: " << zapremina << "\n";

    zapremina = ZapreminaKocke(length, width);
    cout << "Druga zapremina je: " << zapremina << "\n";

    zapremina = ZapreminaKocke(length);
    cout << "Treca zapremina je: " << zapremina << "\n";
     return 0;
}

ZapreminaKocke(int length, int width, int height)
{
   return (length * width * height);
 }
```

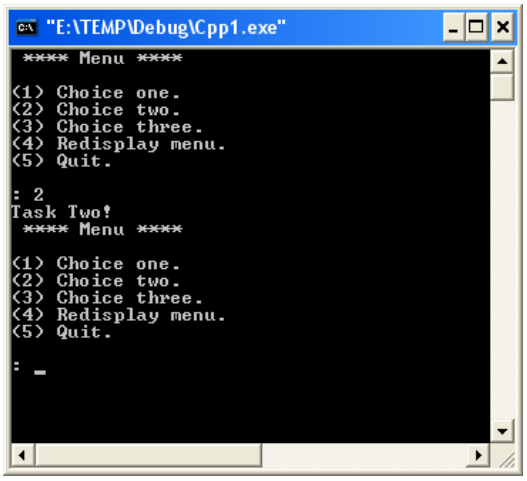


21 Zadatak:

Korišćenje beskonačne petlje za menadžment korisničke interakcije.

```
#include <iostream.h>
   // prototipovi
   int menu();
   void DoTaskOne();
   void DoTaskMany(int);
   int main()
   {   bool exit = false;
      for (;;)
      {    int choice = menu();
          switch(choice)
          {  case (1):
                DoTaskOne();
                break;
             case (2):
                DoTaskMany(2);
                break;
             case (3):
                DoTaskMany(3);
                break;
             case (4):
                continue;  // redundant!
                break;
```

```
            case (5):
                exit=true;
                break;
            default:
                cout << "Please select again!\n";
                break;
        }         // end switch

        if (exit)
            break;
    }             // end forever
  return 0;
}                 // end main()

int menu()
{int choice;
   cout << " **** Menu ****\n\n";
   cout << "(1) Choice one.\n";
   cout << "(2) Choice two.\n";
   cout << "(3) Choice three.\n";
   cout << "(4) Redisplay menu.\n";
   cout << "(5) Quit.\n\n";
   cout << ": ";
   cin >> choice;
   return choice;}

void DoTaskOne()
{       cout << "Task One!\n";    }

void DoTaskMany(int which)
{
   if (which == 2)
       cout << "Task Two!\n";
   else
       cout << "Task Three!\n";
}
```



22 Zadatak:

Prilikom poziva funkcije potrebno je navesti listu argumenata koja odgovara listi argumenata u zaglavlju funkcije. C++ dopušta da se neki od argumenata u pozivu funkcije izostave, tako da se umesto njih koriste podrazumevane vrednosti.

Primer:

```
#include <iostream.h>
void funkcija(int a, int b=5){
cout << "a = " << a << endl;
cout << "b = " << b << endl << endl;
}
void main(){
funkcija (3,4);
funkcija (10);}
```

23 Zadatak:

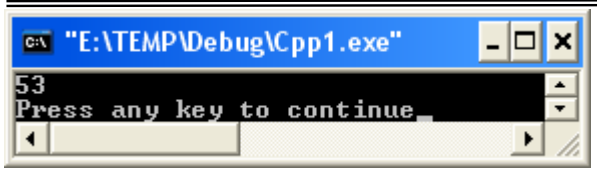
Deklaracije referenci
U C-u se često koriste pokazivači za prosledivanje argumenata funkcijama. U C++ može se koristiti referentni operator (&) u listi argumenata, što čini programski kod čišcim.

C:

```
void zamena (int *a, int *b){
int t = *a;
*a = *b;
*b = t;
}
void main()
{int x = 3, y = 5;
zamena (&x, &y);
printf ("%i %i\n",x,y);}
```

C++:

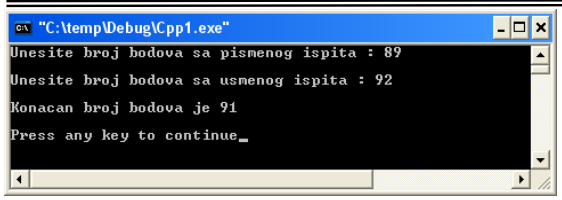
```
#include <iostream.h>
void zamena (int &a, int &b)
{int t = a;
a = b;
b = t;}
void main()
{int x = 3, y = 5;
zamena (x,y);
cout << x << y << endl;}
```



24 Zadatak:

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

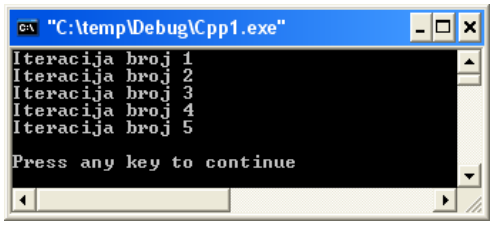
```
// Program izracunava krajnju ocenu na ispitu na osnovu rezultata
// pismenog i usmenog ispita. Koisti funkciju
// Izracunaj_Konacnu_Ocenu() za izracunavanje konacne ocene.
// Funkcija vraca vrednost u main().
#include <iostream.h>
int Izracunaj_Konacnan_Broj_Bodova(int, int); // Deklarisanje funkcije
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;}
```



25 Zadatak:

Napisati program koji broji iteracije u petlji primenom static promenljive.

```
#include <iostream.h>
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;
}
```



Trajanje promenljive je vreme za koje program alocira memoriju promenljivoj. Automatic variable je lokalna varijabla, koja po default-u ima auto storage class, 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.

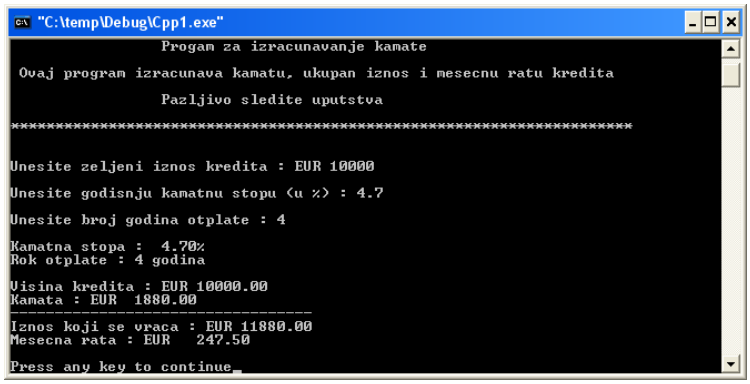
26 Zadatak:

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

```
// Program za izracunavanje kamate
#include <iostream.h>
#include <iomanip.h>
#include <stdlib.h>
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;
}
```

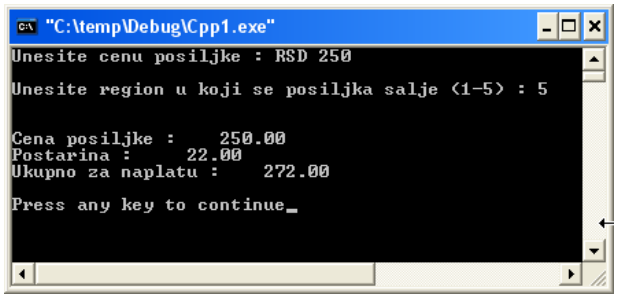


27 Zadatak:

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

```
// 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.h>
#include <iomanip.h>
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 : RSD ";
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;
}
```



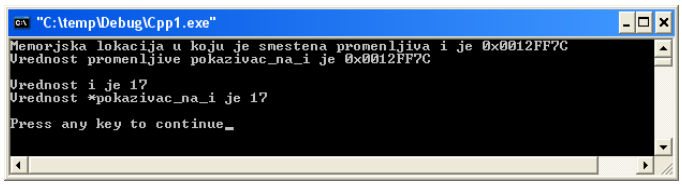
POINTERI

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.

1 Zadatak:

Napisati program koji dodeljuje vrednost promenljivoj primenom pokazivača.

```
// Program ilustruje upotrebu pokazivaca - pointera
// i indirektnog operatora.
#include <iostream.h>
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;
}
```

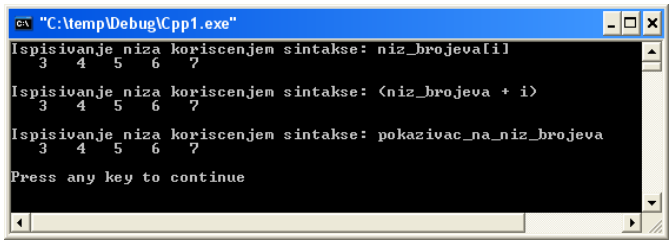


2 Zadatak:

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

```
// Program demonstrira aritmeticke operacije sa pokazivacima
// i redovima.
#include <iostream.h>
#include <iomanip.h>
int main()
{
int niz_brojeva[5] = {3, 4, 5, 6, 7};
int* pokazivac_na_niz_brojeva;
```

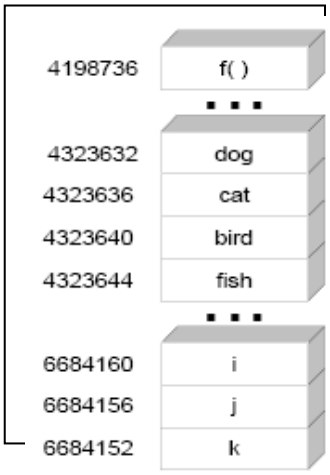
```
int i;
cout << "Ispisivanje niza koriscenjem sintakse: niz_brojeva[i]"<< endl;
for (i = 0; i < 5; ++i)
cout << setw(4) << niz_brojeva[i];
cout << endl << endl;
cout << "Ispisivanje niza koriscenjem sintakse: "<< "(niz_brojeva + i)" << endl;
for (i = 0; i < 5 ; ++i)
cout << setw(4) << *(niz_brojeva + 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;}
```

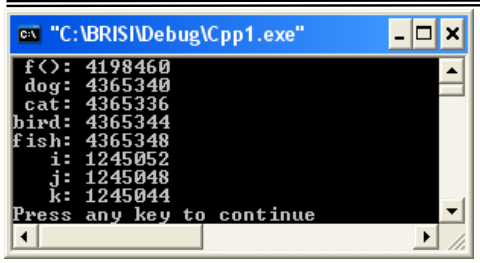


3 Zadatak:

Napraviti program za prikaz adresa funkcije i promenljivih u operativnoj memoriji.

```
#include <iostream.h>
#include<iomanip.h>
int dog, cat, bird, fish;
void f(int pet);

int main()
{int i, j, k;
cout << setw(6)<<"f(): " << (long)&f << endl;
cout << setw(6)<<"dog: " << (long)&dog << endl;
cout << setw(6)<<"cat: " << (long)&cat << endl;
cout << setw(6)<<"bird: " << (long)&bird << endl;
cout << setw(6)<<"fish: " << (long)&fish << endl;
cout << setw(6)<<"i: " << (long)&i << endl;
cout << setw(6)<<"j: " << (long)&j << endl;
cout << setw(6)<<"k: " << (long)&k << endl;
return 0;}

void f(int pet)
{cout << "pet id number: " << pet << endl;}
IZLAZ:
```



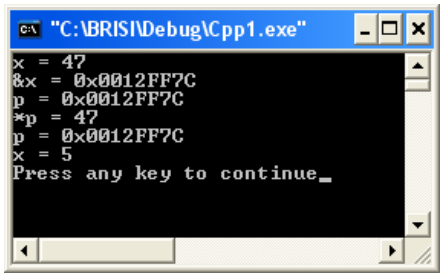


4 Zadatak:

Korišćenje ukazivača na adresu i indirekcije:

```
#include <iostream.h>
void f(int* p) ;
int main() {
int x = 47;
cout << "x = " << x << endl;
cout << "&x = " << &x << endl;
f(&x);     //poziv funkcije
cout << "x = " << x << endl;
return 0;}
void f(int* p) {
cout << "p = " << p << endl;
cout << "*p = " << *p << endl;
*p = 5;
cout << "p = " << p << endl;}
```

IZLAZ:



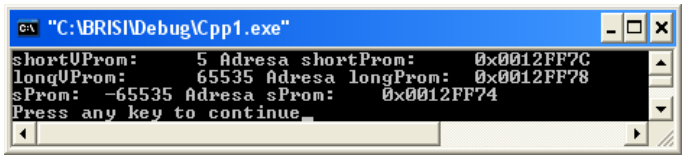
5 Zadatak:

Demonstriranje operatora adresa od, i adresa lokalnih promenljivih

```
 #include <iostream.h>
      int main()
      {unsigned short shortProm=5;
      unsigned long longProm=65535;
      long sProm = -65535;
       cout <<"shortVProm:\t" << shortProm;
      cout << " Adresa shortProm:\t";
      cout << &shortProm<< "\n";
      cout << "lonqVProm:\t" << longProm;
      cout << " Adresa longProm:\t" ;
      cout << &longProm << "\n";
```

```
    cout << "sProm:\t"<< sProm;
    cout << " Adresa sProm:\t" ;
    cout << &sProm << "\n";
   return 0;}
```

IZLAZ:

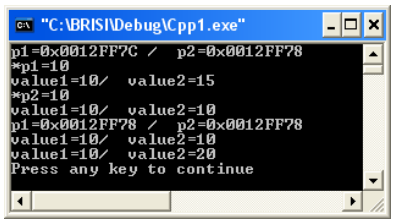


6 Zadatak:

Primena operatora adresa i ukazatelja:

```
#include <iostream.h>
main ()
{ int value1 = 5, value2 = 15;
int *p1, *p2;
p1 = &value1; // p1 = adresa value1
p2 = &value2; // p2 = adresa value2
cout << "p1=" << p1 << " /  p2=" << p2<<"\n" ;
*p1 = 10; // ukazatelj na p1=10
cout<<"*p1="<<*p1<<endl;
cout << "value1=" << value1 << "/  value2=" << value2<<endl;
*p2 = *p1; // ukazatelj p2 jednak je ukazatelju p1
cout<<"*p2="<<*p2<<endl;
cout << "value1=" << value1 << "/  value2=" << value2<<endl;
p1 = p2; // p1 = p2
cout << "p1=" << p1 << " /  p2=" << p2<<"\n" ;
cout << "value1=" << value1 << "/  value2=" << value2<<endl;
*p1 = 20; // ukazatelj na p1=20
cout << "value1=" << value1 << "/  value2=" << value2<<endl;
return 0; }
```

IZLAZ:



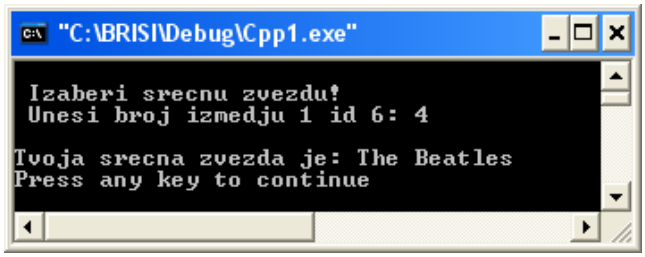
7 Zadatak:

Inicijalizacija pointera sa stringovima

```
#include <iostream.h>
int main() { // Inicijalizacija pointerskog polja
char *pstr[ ]= { "Robert Redford",
                 "Sting",
                 "Lassie",
                 "The Beatles",
```

```
                    "Boris Karloff",
                    "Oliver Hardy" };
    char *pstart = "Tvoja srecna zvezda je: ";
    int izbor = 0;
    cout << endl << " Izaberi srecnu zvezdu!"<<endl;
    cout << " Unesi broj izmedju 1 id 6: ";
    cin >> izbor;
    cout << endl;
    if(izbor >= 1 && izbor <= 6) // Provera ulaznih podataka
    cout << pstart << pstr[izbor-1]; // Izabrano ime zvezde
    else
    // Invalid input
    cout << "Zalim, nisi izabrao srecnu zvezdu.";
    cout << endl;
    return 0; }
```

IZLAZ:

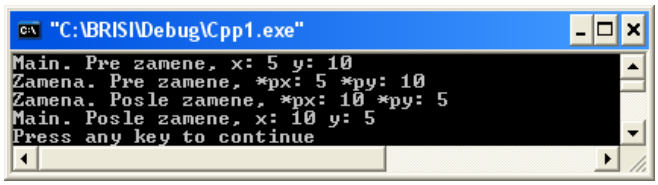


8 Zadatak:

Demonstrcija prelaza po referenci korišćenjem Pointera

```
#include <iostream.h>
    void zamena(int *x, int *y);
    int main()
    {       int x = 5, y = 10;
     cout << "Main. Pre zamene, x: " << x << " y: " << y << "\n";
     zamena(&x,&y);
     cout << "Main. Posle zamene, x: " << x << " y: " << y << "\n";
   return 0;     }
    void zamena (int *px, int *py)
    {       int temp;
     cout << "Zamena. Pre zamene, *px: " << *px << " *py: " << *py << "\n";
     temp = *px;
     *px = *py;
     *py = temp;
     cout << "Zamena. Posle zamene, *px: " << *px << " *py: " << *py << "\n";     }
```

IZLAZ:



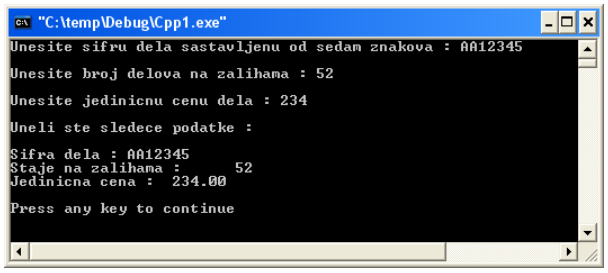
STRUKTURE

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.ime_clana.

1 Zadatak:

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.

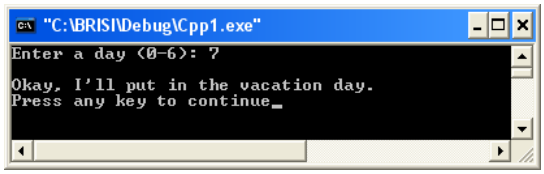
```
// Program ilustruje deklarisanje i upotrebu strukture promenljivih
// kao i opratora clanica za pristupanje promenljivima clanicama.
#include <iostream.h>
#include <iomanip.h>
struct OPIS_DELA
{char sifra_dela[8];
int kolicina_na_stanju;
double jedinicna_cena;};
int main()
{OPIS_DELA deo;
cout << 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;}
```



2 Zadatak:

Primena izvedenog tipa podataka enum kao uvod u strukture:

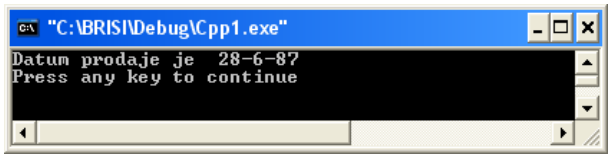
```
#include <iostream.h>
 int main()
 {
    enum Days { Sunday, Monday, Tuesday,
       Wednesday, Thursday, Friday, Saturday };
    int choice;
    cout << "Enter a day (0-6): ";
    cin >> choice;
    if (choice == Sunday || choice == Saturday)
      cout << "\nYou're already off on weekends!\n";
   else
     cout << "\nOkay, I'll put in the vacation day.\n";
   return 0;
 }
```



2 Zadatak:

Program za ilustraciju struktura

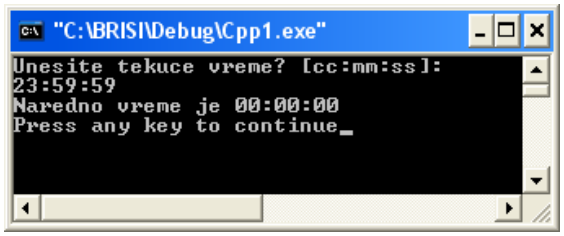
```
#include<iostream.h>
main()
        {
        struct datum {
                int dan;
                int mesec;
                int godina;
                  };
        struct datum prodaja;
        prodaja.dan=28;
        prodaja.mesec=6;
        prodaja.godina=1987;
        cout<< ''Datum prodaje je  ''<< prodaja.dan<<''-''<<prodaja.mesec<<''-''
<< prodaja.godina%100<<endl;,
        }
```



3 Zadatak:

Program za korekciju tekuceg vremena

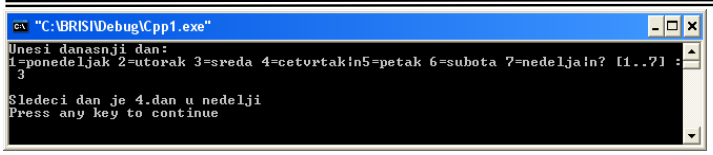
```
#include<iostream.h>
#include<iomanip.h>
void main()
        {
        struct vreme {      int sat;    int minut;          int sekund;          }_vreme[2];
        cout<<"Unesite tekuce vreme? [cc:mm:ss]:\n";
        cin>>_vreme[0].sat;
        cin>>_vreme[0].minut;
        cin>>_vreme[0].sekund;
        _vreme[1]=_vreme[0];
        if(++_vreme[1].sekund==60)
                {           _vreme[1].sekund=0;
                    if(++_vreme[1].minut==60)
                        {  _vreme[1].minut=0;
                            if(++_vreme[1].sat==24)
                                        _vreme[1].sat=0; }                      }
                cout << setiosflags( ios::fixed );
                cout.precision(2);
                cout<<"Naredno vreme je    "<<_vreme[1].sat<<":"<<_vreme[1].minut
                        <<":"<<_vreme[1].sekund<<endl;
        }
```



4 Zadatak:

Program za odredjivanje sledeceg dana

```
main()
        {           int k;
        enum dani {ponedeljak, utorak, sreda, cetvrtak, petak, subota, nedelja} sledeci_dan;
        printf("Unesi danasnji dan:\n");
        printf("1=ponedeljak 2=utorak 3=sreda 4=cetvrtak|n");
        printf("5=petak 6=subota 7=nedelja|n? [1..7] : ");
        scanf("%d", &k);
        k+=1; k%=7;
        sledeci_dan=(enum dani) k;
        printf("\nSledeci dan je %d.dan u nedelji\n", (int)sledeci_dan); }
```



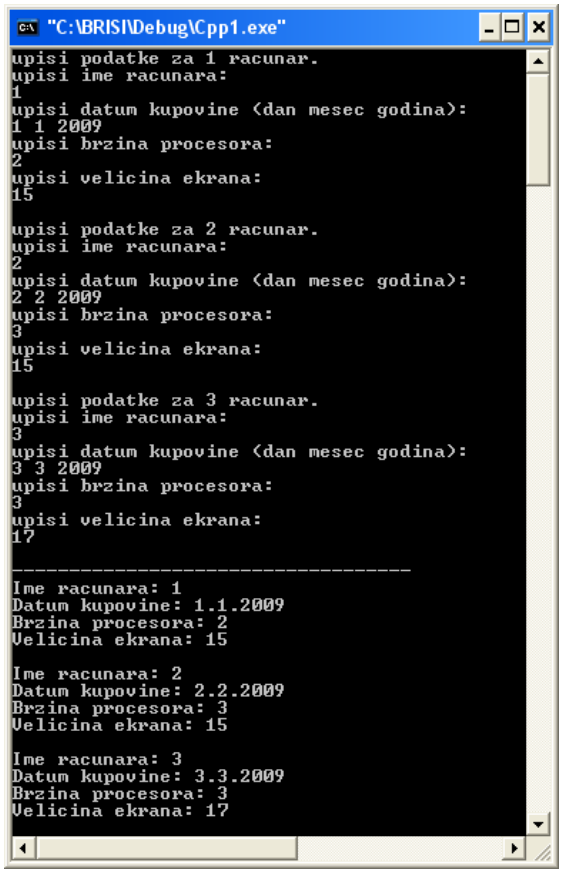
5 Zadatak:

Sastaviti program za unos karakteristika računara primenom struktura tipa Datum {int dan, mesec, godina;} i predmet { char ime[32];Datum d_kupovine ;int brzina_procesora; int velicina_ekrana; }. Broj računara za unos ograničiti.

```
#include<iostream.h>
struct Datum //Struktura datum
{int dan, mesec, godina;};

struct predmet //struktura tipa predmet
{char ime[32];
Datum d_kupovine;//ugnjezdena struktura tipa Datum
int brzina_procesora;
int velicina_ekrana; };

void upis(predmet racunar[]) //funkcija za upis podataka
{
for (int i=0; i<3; i++)
{
cout<< "upisi podatke za "<< i+1<< " racunar." <<endl;
cout <<"upisi ime racunara: "<<endl;
cin>> racunar[i].ime;
cout<< "upisi datum kupovine (dan mesec godina): "<<endl;
cin>> racunar[i].d_kupovine.dan >> racunar[i].d_kupovine.mesec>>
racunar[i].d_kupovine.godina;
cout<< "upisi brzina procesora: "<<endl;
cin>> racunar[i].brzina_procesora;
cout<< "upisi velicina ekrana: "<<endl;
cin>> racunar[i].velicina_ekrana;
cout<< endl;
}}

void izpis(predmet racunar[]) //funkcija za izpis podataka
{
for (int i=0; i<3; i++)
{
cout<< "Ime racunara: " <<racunar[i].ime<< endl;
cout<< "Datum kupovine: "<< racunar[i].d_kupovine.dan <<"."<<
racunar[i].d_kupovine.mesec<< "."
<<racunar[i].d_kupovine.godina<< endl;
cout<< "Brzina procesora: "<< racunar[i].brzina_procesora<< endl;
cout<< "Velicina ekrana: " <<racunar[i].velicina_ekrana<< endl;
cout<< endl;
}}
```

```
int main()
{predmet racunar[10]; //definicija polja racunar tipa predmet
upis (racunar); //poziv funkcije upis
cout<< "-----------------------------------"<<endl;
izpis(racunar); //poziv funkcije izpis
return 0;}
```

Izlaz :



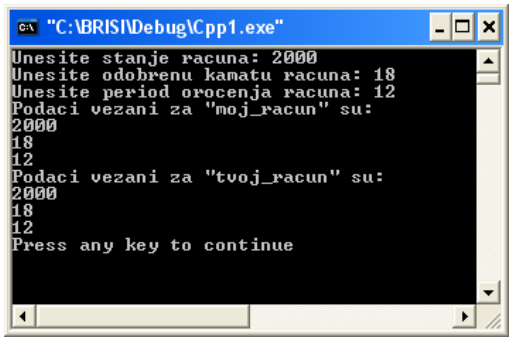
6 Zadatak:

Da vidimo kako bi sve to izgledalo u programu, koji će samo učitati od korisnika podatke o nekom računu, a zatim ih odštampati na ekran ali preuzete i u drugoj strukturi..

```
#include<iostream.h>
struct Racun{double stanje;double kamata;int period;};
int main ()
{
Racun moj_racun, tvoj_racun;//dve strukture tipa Racun
cout<<"Unesite stanje racuna: ";
cin>>moj_racun.stanje;
cout<<"Unesite odobrenu kamatu racuna: ";
```

```
cin>>moj_racun.kamata;
cout<<"Unesite period orocenja racuna: ";
cin>>moj_racun.period;
tvoj_racun=moj_racun;
cout<<"Podaci vezani za \"moj_racun\" su: \n";
cout<<moj_racun.stanje<<endl;
cout<<moj_racun.kamata<<endl;
cout<<moj_racun.period<<endl;
cout<<"Podaci vezani za \"tvoj_racun\" su: \n";
cout<<tvoj_racun.stanje<<endl;
cout<<tvoj_racun.kamata<<endl;
cout<<tvoj_racun.period<<endl;
return 0;}
```

Izlaz:



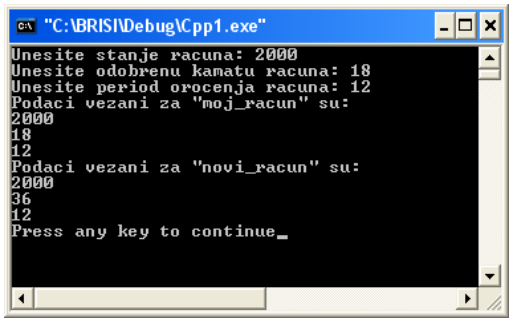
7 Zadatak:

Program demonstrira upotrebu funkcija u radu sa strukturama

```
#include <iostream.h>
struct Racun{ double stanje;double kamata;int period; };
Racun udvostruciKamatu( Racun );  //funkcija
int main ( )
{Racun moj_racun, novi_racun;//dve strukture tipa Racun
cout<<"Unesite stanje racuna: ";
cin>>moj_racun.stanje;
cout<<"Unesite odobrenu kamatu racuna: ";
cin>>moj_racun.kamata;
cout<<"Unesite period orocenja racuna: ";
cin>>moj_racun.period;
novi_racun=udvostruciKamatu(moj_racun);
cout<<"Podaci vezani za \"moj_racun\" su: \n";
cout<<moj_racun.stanje<<endl;
cout<<moj_racun.kamata<<endl;
cout<<moj_racun.period<<endl;
cout<<"Podaci vezani za \"novi_racun\" su: \n";
cout<<novi_racun.stanje<<endl;
cout<<novi_racun.kamata<<endl;
cout<<novi_racun.period<<endl;
```

```
return 0;}

Racun udvostruciKamatu(Racun stari_racun)    //funkcija
{ Racun temp=stari_racun;
temp.kamata=2*stari_racun.kamata;
return temp; }
```

Izlaz:

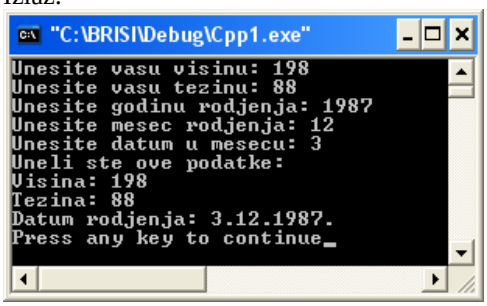


8 Zadatak:

Program demonstrira upotrebu struktura unutar struktura

```
#include <iostream.h>
struct Datum
{ int mesec;
int dan;
int godina;};
struct Osoba
{ double visina;
int tezina;
Datum rodjendan; };
int main ( )
{ Osoba osoba1;
cout<<"Unesite vasu visinu: ";
cin>>osoba1.visina;
cout<<"Unesite vasu tezinu: ";
cin>>osoba1.tezina;
cout<<"Unesite godinu rodjenja: ";
cin>>osoba1.rodjendan.godina;
cout<<"Unesite mesec rodjenja: ";
cin>>osoba1.rodjendan.mesec;
cout<<"Unesite datum u mesecu: ";
cin>>osoba1.rodjendan.dan;
cout<<"Uneli ste ove podatke: \n";
cout<<"Visina: "<<osoba1.visina<<endl;
cout<<"Tezina: "<<osoba1.tezina<<endl;
cout<<"Datum rodjenja: "<<osoba1.rodjendan.dan;
cout<<"."<<osoba1.rodjendan.mesec;
cout<<"."<<osoba1.rodjendan.godina<<"."<<endl;
return 0;}
```

Izlaz:

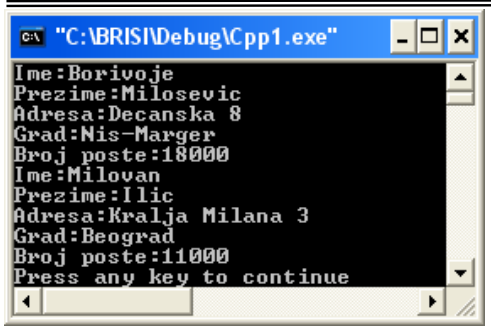


9 Zadatak:

Napraviti strukturu zaposleni i inicijalizovati je primenom komande za kopiranje stringa.

```
#include <string.h>
#include <iostream.h>
struct Zaposleni // Deklaracija strukture
{char ime[20]; //Deklaracija polja char s dvadeset znakova
char prezime[20];
char adresa[50];
char grad[20];
int BrojPoste;};
void main()
{
struct Zaposleni slog1; //Instanca strukture Zaposleni, a njen naziv je slog1
strcpy(slog1.ime, "Borivoje"); //Kopirati odgovarajuce vrednosti
strcpy(slog1.prezime, "Milosevic");
strcpy(slog1.adresa, "Decanska 8");
strcpy(slog1.grad, "Nis-Marger");
slog1.BrojPoste = 18000;
struct Zaposleni slog2 ={"Milovan","Ilic","Kralja Milana 3","Beograd",11000,};
 /* Kraci oblik kreiranja i  instanciranja strukture*/
cout<<"Ime:" <<slog1.ime<<endl;
cout<<"Prezime:" <<slog1.prezime<<endl;
cout<<"Adresa:" <<slog1.adresa<<endl;
cout<<"Grad:" <<slog1.grad<<endl;
cout<<"Broj poste:" <<slog1.BrojPoste<<endl;
cout<<"Ime:" <<slog2.ime<<endl;
cout<<"Prezime:" <<slog2.prezime<<endl;
cout<<"Adresa:" <<slog2.adresa<<endl;
cout<<"Grad:" <<slog2.grad<<endl;
cout<<"Broj poste:" <<slog2.BrojPoste<<endl;
}
```

Izlaz :



10 Zadatak:

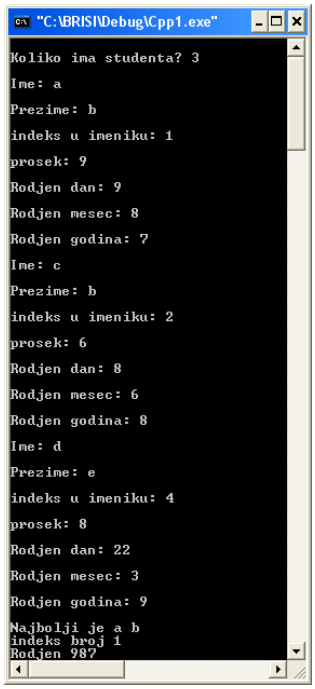
Primer programa u kojem se unose podaci za: ime, prezime, matični broj, prosek i datum rođenja za sve studente jednog godine (najviše 40). Posle unosa svih podataka traži se student s najboljim prosekom pa se na kraju programa na ekranu računara ispisuju svi podaci za studenta koji ima najbolji prosek.

Objašnjenje: U programu su definisane dve strukture: datum i student. Struktura datum služi za unos datuma rođenja studenta i sastoji se od dana, meseca i godine rođenja. Druga struktura se zove student i sadrži 2 niza znakova (za ime i prezime), broj indeksa i prosek ocena. Niz (polje) godina se sastoji od struktura tipa student. Na primer poziv za ispis promenljive *razred[0].rodjendan.godina* će dati za ispis godine rođenja za 1. studenta, a *razred[1].prosek* će dati ispis proseka za 2. studenta.

```
#include <iostream.h>
#include <string.h>
        struct datum
{int dan;
int mesec;
int godina;};
        struct student
{char ime[15];
char prezime[15];
int indeks;
float prosek;
struct datum rodjendan;// struktura rodjendan tipa datum
}razred[40];        /*polje razred ciji su clanovi strukture tipa student*/

int main()
{int n,i,k;
float max;
cout<<"\nKoliko ima studenta? ";
cin>>n;
if (n>40)
cout<<"Broj studenta ne moze biti >40";
else
{for(i=0;i<n;i++)        /*unos podataka za pojedinog studenta u polje razred*/
{cout<<"\nIme: ";
cin>>razred[i].ime;
cout<<"\nPrezime: ";
```

```
cin>>razred[i].prezime;
cout<<"\nindeks u imeniku: ";
cin>>razred[i].indeks;
cout<<"\nprosek: ";
cin>>razred[i].prosek;
cout<<"\nRodjen dan: ";
cin>>razred[i].rodjendan.dan;
cout<<"\nRodjen mesec: ";
cin>>razred[i].rodjendan.mesec;
cout<<"\nRodjen godina: ";
cin>>razred[i].rodjendan.godina;}
max=razred[0].prosek;    /*odredjivanje najboljeg proseka*/
k=0;
for(i=0;i<n;i++)
if (max<razred[i].prosek)
{max=razred[i].prosek;k=i;}
/*ispis najboljeg*/
cout<<"\nNajbolji je "<<razred[k].ime<<" " <<razred[k].prezime;
cout<<"\nindeks broj "<<razred[k].indeks;
cout<<"\nRodjen "<<razred[k].rodjendan.dan<<razred[k].rodjendan.mesec<<
razred[k].rodjendan.godina;
cout<<"\nS prosekom "<<razred[k].prosek;
}
return 0;
}
```

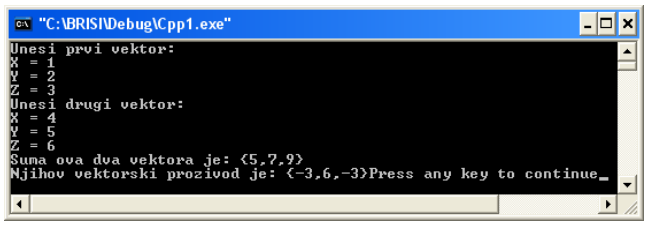
Izlaz:



11 Zadatak:

Strukture se *mogu prenositi kao parametri u funkcije*, kako po vrednosti, tako i po referenci. Takođe, za razliku od nizova, strukture se mogu i *vraćati kao rezultati iz funkcija*. Ova tehnika ilustrovana je u sledećem primeru, u kojem je definisan strukturni tip podataka "Vektor3d" koji opisuje vektor u prostoru koji se, kao što znamo, može opisati sa tri koordinate *x, y* i *z*. U prikazanom programu, prvo se čitaju koordinate dva vektora sa tastature, a zatim se računa i ispisuje njihova suma i njihov vektorski prozivod (pri čemu se vektor ispisuje u formi tri koordinate međusobno razdvojene zarezima, unutar vitičastih zagrada):

```
#include <iostream.h>
struct Vektor3d {double x, y, z;};
void UnesiVektor(Vektor3d &v)
{cout << "X = "; cin >> v.x;
cout << "Y = "; cin >> v.y;
cout << "Z = "; cin >> v.z;}
Vektor3d ZbirVektora(const Vektor3d &v1, const Vektor3d &v2)
{Vektor3d v3;
v3.x = v1.x + v2.x; v3.y = v1.y + v2.y; v3.z = v1.z + v2.z;
return v3;}

Vektor3d VektorskiProizvod(const Vektor3d &v1, const Vektor3d &v2)
{Vektor3d v3;
v3.x = v1.y * v2.z - v1.z * v2.y;
v3.y = v1.z * v2.x - v1.x * v2.z;
v3.z = v1.x * v2.y - v1.y * v2.x;
return v3;}

void IspisiVektor(const Vektor3d &v)
{cout << "{" << v.x << "," << v.y << "," << v.z << "}";}

int main()
{Vektor3d a, b;
cout << "Unesi prvi vektor:\n";
UnesiVektor(a);
cout << "Unesi drugi vektor:\n";
UnesiVektor(b);
cout << "Suma ova dva vektora je: ";
IspisiVektor(ZbirVektora(a, b));
cout << endl << "Njihov vektorski prozivod je: ";
IspisiVektor(VektorskiProizvod(a, b));
return 0;}
```

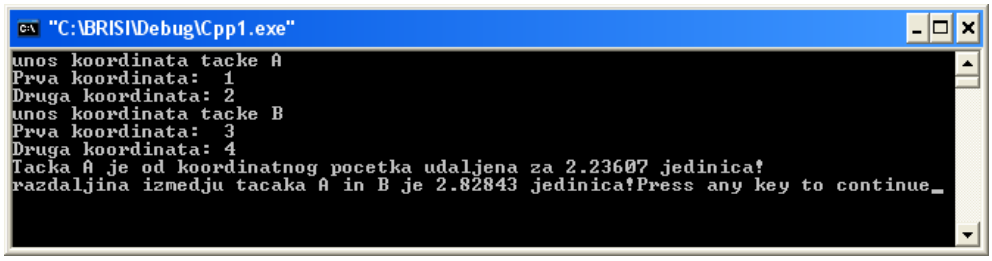


12 Zadatak:

Kreiraj strukturu tacka2D sa dve koordinate tipa //int. Napisi funkcije za unos podataka, izracunati udaljenje ove tacke od koorinatnog pocetka i funkciju za izracunavanje razdaljine izmedju tacaka!

```
#include <iostream.h>
#include <iostream.h>
#include <conio.h>
#include <math.h>
#include <stdio.h>

struct tacka2D { int x,y;};
tacka2D A,B,C;
void unos(tacka2D &T) { //parameter je pozvan po referenci
  cout<<"Prva koordinata:  ";cin>>T.x;
  cout<<"Druga koordinata: ";cin>>T.y;}
float razdaljina (tacka2D T)
{ return (sqrt(float(pow(T.x,2)+pow(T.y,2))));}
float razdaljina(tacka2D T1, tacka2D T2)
{return (sqrt(float(pow(T1.x-T2.x,2)+pow(T1.y-T2.y,2))));}

void main(){
   puts("unos koordinata tacke A");unos(A);
   puts("unos koordinata tacke B");unos(B);
   cout<<"Tacka A je od koordinatnog pocetka udaljena za ";
   cout<<razdaljina(A)<<" jedinica!"<<endl;
   cout<<"razdaljina izmedju tacaka A in B je "<<razdaljina(A,B)<<" jedinica!";
   //Pretrazi podatke jos za tacku C! Koliki je obim trougaonika ABC?

   //Koja od tacaka A, B i C je najvise udaljena od koordinatnog pocetka?
   //Napisi funkciju HERON, koja ima za parametre tri tacke, koja vraca
   //povrsinu trougla koju dobijamo pozivom funkcije cout<<HERON(A,B,C);
   //Heronova formula p=sqrt(s*(s-a)*(s-b)*(s-c)), s=(a+b+c)/2

}
```

Izlaz:

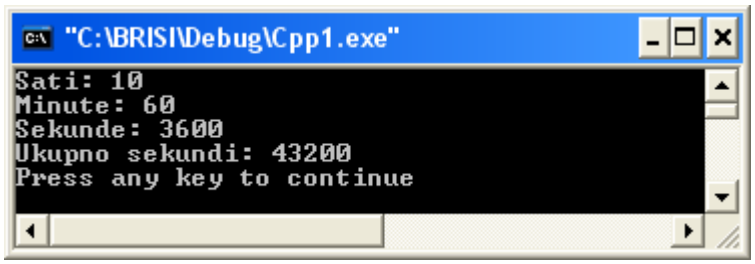


13 Zadatak:

Primer strukture ČAS, koja pamti podatke o trenutnom času. Sledi funkcija za pretvaranje časa u sekunde:

```
#include <iostream.h>
#include <conio.h>
struct Cas {
   int sati;
   int minute;
   int sekunde;
};

int VSekunde(Cas trenutni) {
   return 3600*trenutni.sati + 60*trenutni.minute + trenutni.sekunde;
}

void main() {
   Cas t;
   cout<<"Sati: ";cin >> t.sati;
   cout<<"Minute: ";cin >> t.minute;
   cout<<"Sekunde: ";cin >> t.sekunde;
   cout << "Ukupno sekundi: " << VSekunde(t) << endl;
}
```

Izlaz:



14 Zadatak:

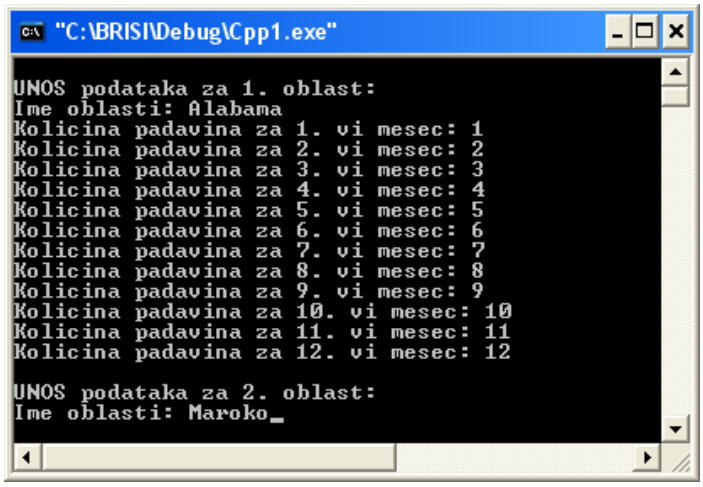
Za vodjenje evidencije o padavinama u zadnjoj godini potrebni su sledeci podaci:
-ime oblasti (do 20 char) i podaci o kolicini padavina u zadnjih 12 meseci
(12 realnih brojeva u polju/tabeli) (6. float)

Kreiraj strukturu podataka PADAVINE
Kreiraj tabelu OBLASTI, koja najvise ima 10 takvih struktura
Napisi funkciju UNOS(N)
Napisi funkciju POVP(OBLASTI,N),

```
#include <iostream.h>
#include <iomanip.h>
#include <stdio.h>
#include <conio.h>
#include <ctype.h>
```

```
struct PADAVINE
{   char ime_oblasti[30];
   float kolicina[12];};

PADAVINE OBLASTI[10];

void UNOS(int N)
{  cout<<"\nUNOS podataka za "<<N+1<<". oblast: \n";
  cout<<"Ime oblasti: ";
  cin>>OBLASTI[N].ime_oblasti;
  for (int i=0;i<12;i++)
{    cout<<"Kolicina padavina za "<<i+1<<". vi mesec: ";
    cin>>OBLASTI[N].kolicina[i];  }}

float POVP(PADAVINE OBLASTI[10], int N
{  float suma=0;
  for (int i=0;i<12;i++)
    suma=suma + OBLASTI[N].kolicina[i];
  return suma; }

void main()
{  for (int i=0;i<10;i++)
    UNOS(i);
  cout<<"Prosecna kolicina padavina oblasti sa indeksom 5 je "<<POVP(OBLASTI,5); }
```

Izlaz:



15 Zadatak:

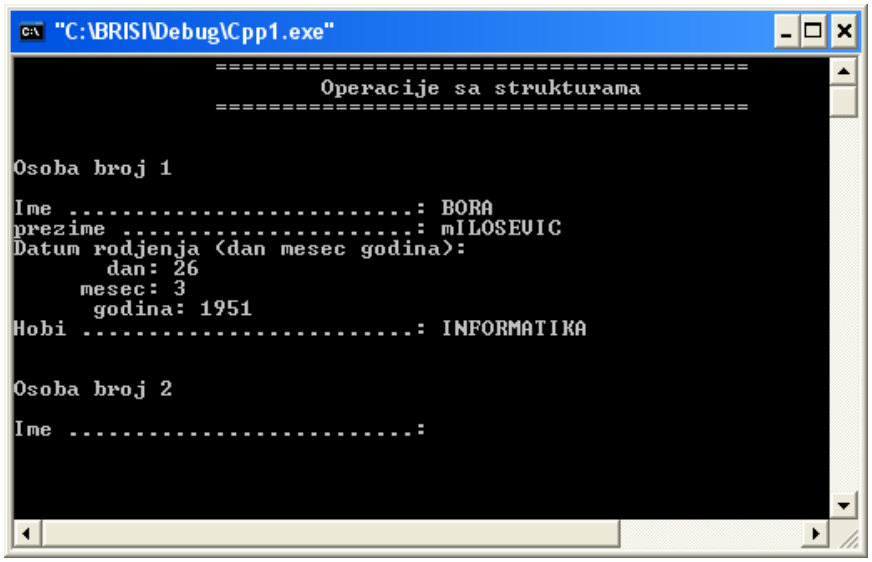
Napisite program, koji sakuplja podatke o 5 osoba. Svaka osoba ima sledece podatke: ime, prezime, datum rodjenja, najdrazi hobi.
Program ispisuje podatke o osobama uredjene po prezimenu i imenu, pa zatim uredjene po hobiju.
Ulazni podaci o osobama su: ime, prezime, dan, mesec, godina, hobi.

```
#include <iostream.h>
#include <iomanip.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <conio.h>
//inicializacija globalnih konstanti
const int n1 = 51;
//inicilazacija struktura
struct dat
{  int dan;
  int mesec;
  int godina;};


struct osoba
{  char ime[n1];
  char prezime[n1];
  dat datum;
  char hobi[n1];};
//preostale funkcije
//void izpis(osoba osobe[ ], int f1);

void swp(char *a, char *b)
   {char tmp[n1];
   strcpy(tmp, a);
   strcpy(a, b);
   strcpy(b, tmp);}

void swp_dat(int& a, int& b)
   {int temp;
   temp = a;
   a = b;
   b = temp;}

void swap(osoba& x, osoba& y)
  //funkcija ,koja menja dve strukture
  //x <=> y (samo nizi)
  {swp(x.ime, y.ime);
  swp(x.prezime, y.prezime);
  swp(x.hobi, y.hobi);
  //zamena datuma
  swp_dat(x.datum.dan ,y.datum.dan);
  swp_dat(x.datum.mesec, y.datum.mesec);
```

```
  swp_dat(x.datum.godina, y.datum.godina);}
//funkcija main
                void main() {
  //glava programa
  cout<< setw(55) << "======================================" <<endl;
  cout<< setw(55) << "        Operacije sa strukturama       " <<endl;
  cout<< setw(55) << "======================================" <<endl;
  cout<<endl;
  //inicializacija potrebnih struktura
  osoba osobe[5];
  //unos podataka u strukture pomocu for petlje
  for(int i = 0; i < 5; i++) {
    cout<<endl;
    cout<< "Osoba broj " << i + 1 <<endl;
    cout<<endl;
    cout<< "Ime ..........................: " ;
    cin>> osobe[i].ime ;
    cout<< "prezime ......................: " ;
    cin>> osobe[i].prezime ;
    cout<< "Datum rodjenja (dan mesec godina): \n" ;
    cout<< "       dan: "; cin>> osobe[i].datum.dan;
    cout<< "     mesec: "; cin>> osobe[i].datum.mesec;
    cout<< "      godina: "; cin>> osobe[i].datum.godina ;
    cout<< "Hobi .........................: " ;
    cin>> osobe[i].hobi ;
    cout<<endl;  }
  //uredjenje struktura po prezimenu i imenu
  for(int f2 = 0; f2 < 5; f2++) {
   for(int f3 = 0; f3 < 5; f3++){
     if(strcmp(osobe[ f2 ].prezime ,osobe[ f3 ].prezime) < 0) {//ce je prvi niz krajsi od drugega
se izvrsi zamenjava struktur
      swap(osobe[ f2 ], osobe[ f3 ] );
     }
     else if(strcmp(osobe[ f2 ].prezime ,osobe[ f3 ].prezime) == 0 && f2 != f3) {//ce sta niza
enaka se zacne gledati po imenu
      if(strcmp(osobe[ f2 ].ime, osobe[ f3 ].ime) < 0) {
        swap(osobe[ f2 ], osobe[ f3 ] );      }     }   }  }
  //ispis struktura
  cout<<endl;
  cout<< "IZPIS OSOBA UREDJENIH PO PREYIMENU I IMENU" <<endl;
  for(int f1 = 0; f1 < 5; f1++) {
    cout<<endl;
    cout<< osobe[ f1 ].prezime << " " ;
    cout<< osobe[ f1 ].ime << " ";
    cout<< osobe[ f1 ].datum.dan << " "<< osobe[f1].datum.mesec << " "
    << osobe[ f1 ].datum.godina << " ";
    cout<< osobe[ f1 ].hobi <<endl;
  }
  //uredjenje strukture po hobiju
  for(int f7 = 0; f7 < 5; f7++)  {
    for(int f8 = 0; f8 < 5; f8++) {
      if(strcmp(osobe[f7].hobi ,osobe[f8].hobi) < 0)
                {  swap(osobe[f7], osobe[f8]); }
```

```
   }
  }
  //ispis struktura
  cout<<endl;
  cout<< "IZPIS OSOBA UREDJENIH PO HOBIJU" <<endl;
  for(int f5 = 0; f5 < 5; f5++)
{    cout<<endl;
    cout<< osobe[ f5 ].prezime << " " ;
    cout<< osobe[ f5 ].ime << " ";
    cout<< osobe[ f5 ].datum.dan << " "<< osobe[ f5 ].datum.mesec << " "
    << osobe[ f5 ].datum.godina << " ";
    cout<< osobe[ f5 ].hobi <<endl;
  }
  cout<<endl;
  getch();
}
```

Izlaz:



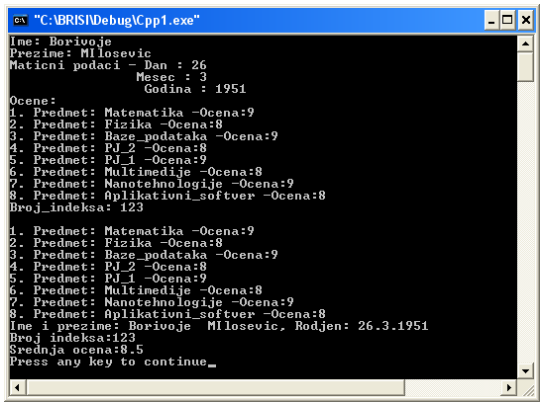
16 Zadatak:

Date su strukture datum, Predmet i apsolvent. Datum definise potrebne datume, struktura Predmet predmete {"Matematika","Fizika","Baze_podataka","PJ_2","PJ_1", "Multimedije", "Nanotehnologije","Aplikativni_softver"}, promenljiva M pamti podatke o jednom apsolventu, svakoj pojedinačnoj oceni koju je postigao na izabranim predmetima.
- napisi funkciju unos(M), za unos svih studentskih podataka
- napisi for petlju, koja ispisuje izabrane predmete i ocene
- napisi deo programa, koji ispisuje ime, prezime i datum rodjenja studenta, broj indeksa i srednju ocenu*/

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
```

```
#include<string.h>

struct datum {   int dan,mesec,godina;  };

struct Predmet {char imepredmeta[20];};
struct Predmet predmeti[8] ={"Matematika","Fizika","Baze_podataka","PJ_2","PJ_1",
"Multimedije", "Nanotehnologije","Aplikativni_softver"};
struct apsolvent
{  char ime[20],prezime[20];
  datum datum_rodjenja;      //Ugnježđena struktura
  int ocene[8],broj_indeksa; };
void unos(apsolvent &M)
{ cout<<"Ime: "; cin>>M.ime;
  cout<<"Prezime: "; cin>>M.prezime;
  cout<<"Maticni podaci - Dan : ";cin>>M.datum_rodjenja.dan;
  cout<<"               Mesec : ";cin>>M.datum_rodjenja.mesec;  // Poziv strukture u strukturi
  cout<<"               Godina : ";cin>>M.datum_rodjenja.godina;
  cout<<"Ocene:\n" ;
  for (int i=0;i<8;i++)
{   cout<<i+1<<". Predmet: "<<predmeti[i].imepredmeta<<" -Ocena:";
    cin>>M.ocene[i];  }
  cout<<"Broj_indeksa: ";cin>>M.broj_indeksa; }

void main()
{ float prolaz=0.0;
  apsolvent M;  // definisanje instance M strukture tipa apsolvent
  //funkcija za unos podataka
  unos(M);
  //Ocene na predmetima:
  for (int i=0;i<8;i++)
  { cout<<"\n"<<i+1<<". Predmet: "<<predmeti[i].imepredmeta<<" -Ocena:"
  <<M.ocene[i];prolaz=prolaz+M.ocene[i]; }
  //stavka, koja zapisuje ime, prezime i datum rodjenja studenta
  cout<<"\nIme i prezime: "<<M.ime<<"  "<<M.prezime
<<", Rodjen: "<<M.datum_rodjenja.dan<<"."<<M.datum_rodjenja.mesec
<<"."<<M.datum_rodjenja.godina<<endl;
  cout<<"Broj indeksa:"<<M.broj_indeksa<<endl;
  cout<<"Srednja ocena:"<<prolaz/8.<<endl;}
```

Izlaz:



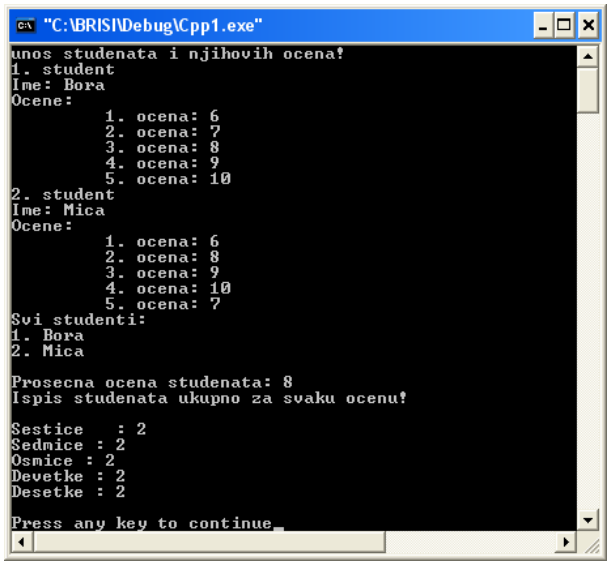
17 Zadatak:

Napisite program, koji najpre sakuplja podatke o N studenata i za svakoga
posebno po deset ocena.
- napisi funkciju za ispis studenata, koji imaju jednu ili vise negativnih ocena
- napisi funkciju, koja vraca prosecnu ocenu svih studenata u tabeli
- napisi funkciju, koja ispisuje imena svih studenata

```
#include <iostream.h>
#include <iomanip.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
//inicializacija globalnih konstanti
const int N = 2;
struct student {  //deklaracija strukture
  char ime[50];
  int ocene[5];
};
student tabela[N];  //tabela N struktur
void unos(student tabela[N]){  //unos podataka u tabelu
 cout<<"unos studenata i njihovih ocena!"<<endl;
 for (int i=0;i<N;i++){
   cout<<i+1<<". student"<<endl;
   cout<<"Ime: ";cin>>tabela[i].ime; //unos imena
   cout<<"Ocene:"<<endl;
   for (int j=0;j<5;j++){ //unos ocena za svakog studenta
       cout<<setw(10)<<j+1<<". ocena: ";
       cin>>tabela[i].ocene[j];
   }
 }
}
void ispisinepolozeno(student tabela[N]){ //i nepolozenim predmetima!\n"<<endl;
  for (int i=0;i<N;i++){
    for (int j=0;j<5;j++)
      if (tabela[i].ocene[j]==5){
        cout<<tabela[i].ime<<endl;
        break;
      }
  }
}
float prosecnaocena(student *D){  //parameter funkcije je ukazatelj na strukturu
  int suma=0; //pocetna vrednost
  for (int i=0;i<N;i++){
    for (int j=0;j<5;j++)
      suma=suma+D[i].ocene[j];  //sabiranje ocena
  }
 return float(suma)/(5*N);  //funkcija vraca prosecnu vrednost
}
void izpisimen(student *D) {  //funkcija ispise imena studenata
  cout<<"Svi studenti:"<<endl;
  for (int i=0;i<N;i++)
```

```
    cout<<i+1<<". "<<D[i].ime<<endl;
}
void pregledocen(student tabela[N]){ //Pregled i ispis prosecnih ocena
 cout<<"\nIspis studenata ukupno za svaku ocenu!\n"<<endl;
 int sest=0,sedam=0,osam=0,devet=0,deset=0;
 for (int i=0;i<N;i++)
{   for (int j=0;j<5;j++)
    switch (tabela[i].ocene[j])
 {     case 6: sest++;
            break;
       case 7: sedam++;
            break;
       case 8: osam++;
            break;
       case 9: devet++;
            break;
      case 10: deset++;
            break;     } }
 cout<<"Sestice   : "<<sest<<endl;
 cout<<"Sedmice : "<<sedam<<endl;
 cout<<"Osmice : "<<osam<<endl;
 cout<<"Devetke : "<<devet<<endl;
 cout<<"Desetke : "<<deset<<endl; }
void main()
 {  unos(tabela);
 ispisinepolozeno(tabela);
 izpisimen(tabela);
 cout<<"\nProsecna ocena studenata: "<<prosecnaocena(tabela);
 pregledocen(tabela);
 cout<<endl; }
```

Izlaz:

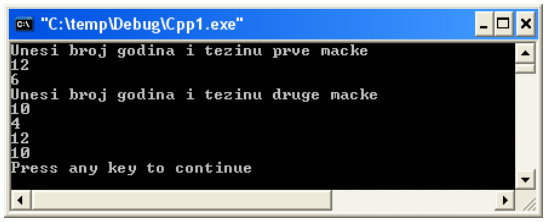


KLASE

1 Zadatak:

Primer klase macka, sa dodatim funkcijama koje vraćaju godine i težinu.

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

class macka
{
public:
void jedi() {cout<<"mljac, mljac, mljac"<<endl;}
void spavaj() {cout<<"zzz..zzzzz"<<endl;}
void predi() {cout<<"prrr...."<<endl;}
void postavi_god(int x) {_god=x;}
void postavi_tezinu(int t) {_tezina=t;}
void vrati_god() {cout<<_god<<endl;}
void vrati_tezinu() {cout<<_tezina<<endl;}

private:
int _god;
int _tezina;
};

int main()
{
macka Garfild, Tom;
int x, t;

cout<<"Unesi broj godina i tezinu prve macke"<<endl;
cin>>x>>t;
Garfild.postavi_god(x);
Garfild.postavi_tezinu(t);
cout<<"Unesi broj godina i tezinu druge macke"<<endl;
cin>>x>>t;
Tom.postavi_god(x);
Tom.postavi_tezinu(t);
Garfild.vrati_god();
Tom.vrati_god();
return 0;
}
```



Svaki objekat sadrži sopstvenu kopiju podataka članova klase. Tom ima sopstvene vrednosti za težinu i godine, isto tako I Garfild. S druge strane postoji samo jedna kopija svake funkcije članice klase. Objekat Tom i Garfild pozivaju istu kopiju bilo koje određene funkcije članice. Videli smo da funkcija članica može da koristi članove svoje klase bez primene operatora za pristup članovima. Ako se funkcija vrati_god() pozove za objekat Tom onda podaci članovi kojima pristupa funkcija vrati_god() su podaci članovi objekta Tom. Ako se funkcija vrati_god() pozove za objekat Garfild, podaci članovi kojima ona pristupa su podaci članovi objekta Garfild. Kako funkcija vrati_god() zna kojim podacima treba da pristupi?

Odgovor na ovo pitanje je pokazivač this. Svaka funkcija članica sadrži pokazivač na adresu objekta za koji je ta funkcija pozvana i taj pokazivač se naziva this.

Ako imamo funkciju:

void vrat_god() {cout<<_god<<endl;} nju kompajler (interno) prevodi u: void vrati_god(macka *this) {cout<<this->_god<<endl;} dok poziv funkcije zapravo postaje umesto

Tom.vrati_god(); postaje vrati_god(&Tom);

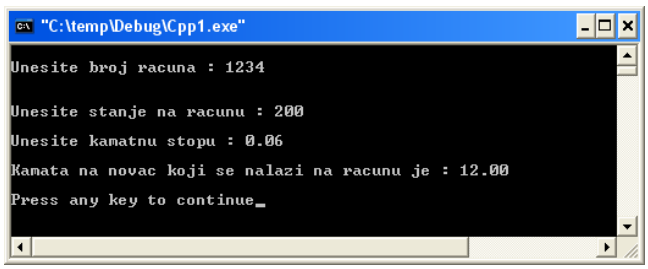
Dakle, svaki objekat ima kao član i pokazivač na njega samog, pokazivač this. Pokazivač this može se koristiti i eksplicitno unutar funkcije članice klase i o tome ćemo tek da pričamo.

2 Zadatak:

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

```
// 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;}
```



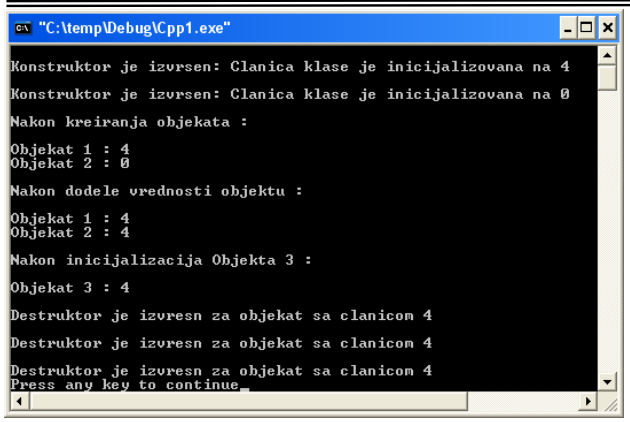
Primetimo da se iz spoljašnje funkcije, u ovom slučaju main() može direktno pristupiti isključivo javnim članicama klase. Privatnim članicama klase se može pristupiti isključivo korišćenjem javnih članica-funkcija, ukoliko je to dozvoljeno prilikom implementacije klase. Funkcije koje pristupaju privatnim 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.

3 Zadatak:

Napisati program koji međusobno dodeljuje vrednosti objektima.

```
// 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;
}
```

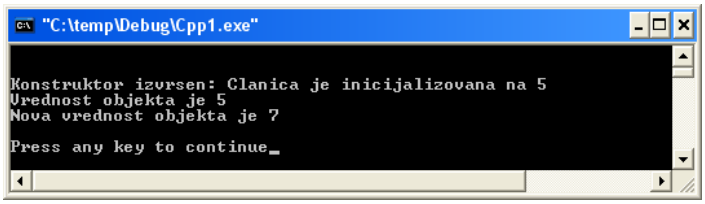


4 Zadatak:

Napisati program koji koristi pokazivač na objekat, kao i operator ->.

```
// 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;
}
```

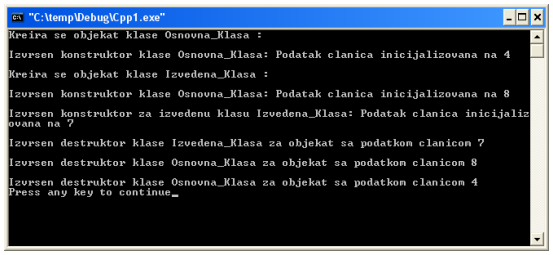


5 Zadatak:

Napisati program koji izvršava konstruktore i destruktore u osnovnoj i izvedenoj klasi.

```
// 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;
}
```



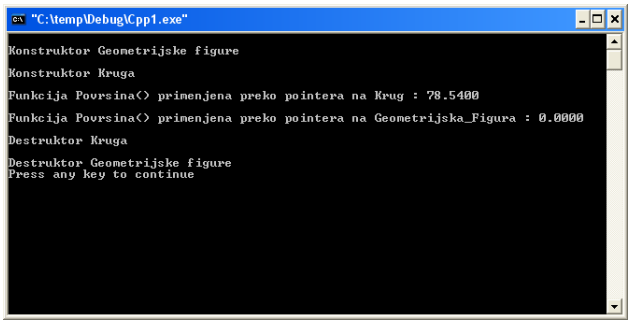
6 Zadatak:

Napisati program u kome se definišu geometrijske figure korišćenjem izvedenih klasa.

```
// 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;
}
```

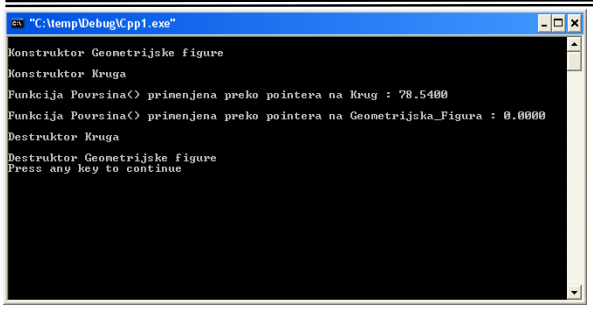


7 Zadatak:

Napisati program u kome se koristi virtual funkcija.

```
// 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;}
```



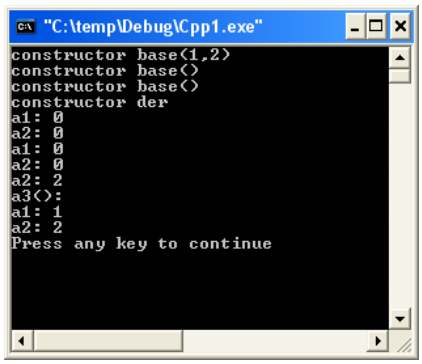
Ključna reč virtual ispred funkcije u osnovnoj klasi saopštava kompajleru da postoji odgovarajuća funkcija u izvedenoj klasi i da nju treba izvršiti prilikom poziva objekta izvedene klase. Ukoliko se ne navede ključna reč virtual, program će izvršiti funkciju osnovne klase, jer kompajler neće imati informaciju o postojanju odgovarajuće funkcije u izvedenoj klasi.

8 Zadatak:

Primer konstruktora osnovne klase

```
#include <iostream.h>
class base
{private: int a1;
protected: int a2;
public:
base()
{cout << "constructor base() \n";
a1 = 0;
a2 = 0; }
base(int x, int y)
{cout << "constructor base(" << x << "," << y << ")\n";
a1 = x;
a2 = y;}
void a3()
{cout << "a1: " << a1 << endl
<< "a2: " << a2 << endl;}
};
class der : public base
{private: base d1;
protected: base d2;
public:
der(int x, int y) : base(x, y)
{cout << "constructor der\n"; }
void d3()
{d1.a3() ;
d2.a3 () ;
cout << "a2: " << a2 << endl;
cout << "a3():" << endl;
a3();}
} ;
```

```
void main()
{der x(1, 2);
x.d3() ; }
```



9 Zadatak:

Primer destruktora osnovne klase

Program 10.2 prikazuje primer klase sa destruktorom za znakovne nizove. Pošto su znakovni nizovi međusobno vrlo različitih dužina, objekti za njihovo predstavljanje obično sadrže samo pokazivač na sam tekst. To je učinjeno i u ovom primeru.

Jedini atribut klase String je pokazivač na sam niz znakova. Zbog toga se prilikom inicijajizacije (konstruktorom String (char*)) vrši dodela memorijskog prostora potrebne veličine i kopiranje niza znakova u tako dodeljenu memoriju. Zadatak destruktora (~String( ) je da oslobodi taj memorijski prostor prilikom uništavanja objekta.

Podrazumevani konstruktor (String ()) osigurava da svaki objekat tipa String bude inicijalizovan do te mere da bi kasnije smelo da se na njega primeni destruktor. Naime, posledice primene operatora delete na pokazivač slučajnog sadržaja su nepredvidljive, ali primena operatora delete na pokaziva ~ NULL (0) je uvek bezbedna.

Naredba niz=0; u destruktoru nije potrebna ako se destruktor poziva samo implicitno prilikom uništavanja objekata. Me|utim, poželjno je da objekat bude ostavljen u " ispravnom praznom " stanju zbog eventualnog eksplicitnog pozivanja destruktora. Tada će objekat, verovatno, i dalje postojati pa bi moglo da zasmeta ako pokazivački atribut pokazuje na nešto u dinamičkoj zoni memorije što više ne postoji.

Konstruktor kopije (String(String&)) samo treba da prekopira celokupan sadržaj svog pravog argumenta u skriveni argument (to je argument koji se inicijalizuje). Pošto se vrši stvaranje novog objekta, ne pretpostavlja se ništa o zatečenom sadržaju parčeta memorije koje se inicijalizuje.

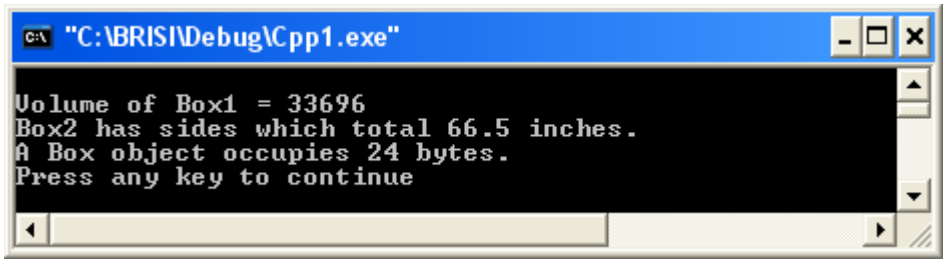
U glavnom programu na kraju programa, prvo se stvara objekat pozdrav tipa String koji se inicijalizuje običnim znakovnim nizom (char*). Tu će se pozivati konstruktor String(char*) koji je, u stvari, konverzija iz tipa char* u tip String. Posle toga, objekat a tipa String se inicijalizuje kopijom sadržaja objekta pozdrav pomoću konstruktora kopije String (String&) , dok objekat b se podrazumevanim konstruktorom String ( ) inicijalizuje kao "prazan" objekat. Ono što je bitno, atribut niz u njemu se postavlja na nulu, pa implicitno pozivanje destruktora (b.~String()) na kraju programa, neće da napravi nikakvu štetu.

10 Zadatak:

Kreiranje klase Box I postavljanje vrednosti njenih clanova:

```
#include <iostream.h>
class Box            // Class definition at global scope
{
  public:
    double length;   // Length of a box in inches
    double breadth;  // Breadth of a box in inches
    double height;   // Height of a box in inches
};

int main(void)
{
  Box Box1;              // Declare Box1 of type Box
  Box Box2;              // Declare Box2 of type Box
  double volume = 0.0;     // Store the volume of a box here
  Box1.height = 18.0;      // Define the values
  Box1.length = 78.0;      // of the members of
  Box1.breadth = 24.0;     // the object Box1
  Box2.height = Box1.height - 10;  // Define Box2
  Box2.length = Box1.length/2.0;   // members in
  Box2.breadth = 0.25*Box1.length; // terms of Box1

// Calculate volume of Box1
  volume = Box1.height * Box1.length * Box1.breadth;
  cout << endl
       << "Volume of Box1 = " << volume;
  cout << endl
       << "Box2 has sides which total "
       << Box2.height + Box2.length + Box2.breadth
       << " inches.";
  cout << endl      // Display the size of a box in memory
       << "A Box object occupies "
       << sizeof Box1 << " bytes.";
  cout <<endl;

  return 0;
}
```

IZLAZ:



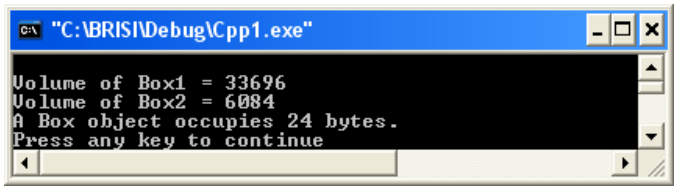
11 Zadatak:

PRIMER kako pristupati clanovima klase I kako klasi Box pridruziti funkciju.

// Izracunavanje zapremine koriscenjem funkcije clasnice klase

```
#include <iostream.h>

class Box               // Class definition at global scope
{
public:
  double length;        // Length of a box in inches
  double breadth;       // Breadth of a box in inches
  double height;        // Height of a box in inches
  // Function to calculate the volume of a box
  double Volume(void)
  {
    return length * breadth * height;
  }
};

int main(void)
{
  Box Box1;             // Declare Box1 of type Box
  Box Box2;             // Declare Box2 of type Box
  double volume = 0.0;  // Store the volume of a box here
  Box1.height = 18.0;   // Define the values
  Box1.length = 78.0;   // of the members of
  Box1.breadth = 24.0;  // the object Box1
  Box2.height = Box1.height - 10;  // Define Box2
  Box2.length = Box1.length/2.0;   // members in
  Box2.breadth = 0.25*Box1.length; // terms of Box1
  volume = Box1.Volume();          // Calculate volume of Box1
  cout << endl
       << "Volume of Box1 = " << volume;
  cout << endl
       << "Volume of Box2 = "
       << Box2.Volume();
  cout << endl
       << "A Box object occupies "
       << sizeof Box1 << " bytes.";
  cout << endl;

  return 0;
}
```

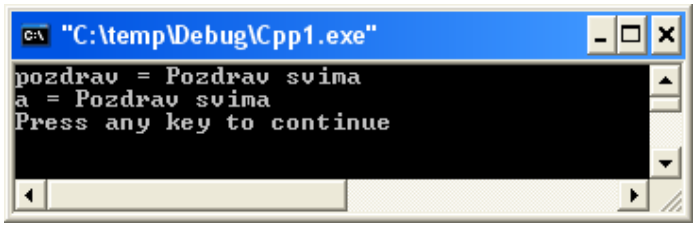
IZLAZ:



12 Zadatak:

Klasa String sa konstruktorima i destruktorima

```
#include <iostream.h>
class String {
char *niz; // Pokazivac na sam tekst.
public:
String () { niz = 0; } // Inicijalizacija praznog niza.
String (const char *) ; // Inicijalizacija nizom znakova.
String (const String &) ; // Inicijalizacija tipom String.
~String () ; // Unistavanje objekta tipa String.
void pisi () const; // Ispisivanje niza.
};
#include <string.h>
#include <iostream.h>
String:: String (const char *t)
{ if ( (niz = new char [strlen(t) +1] ) != 0) strcpy(niz,t) ; }
String::String (const String &s)
{if ((niz = new char [strlen(s.niz)+1]) != 0) strcpy (niz, s.niz); }
String::~String ()
{ delete [] niz; niz = 0; }
void String::pisi () const
{ cout << niz; }
int main () {
String pozdrav ("Pozdrav svima") ; // Poziva se String (char *)
String a = pozdrav; // Poziva se String (String &).
String b; // Poziva se String ().
cout << "pozdrav = "; pozdrav.pisi (); cout << endl;
cout << "a = "; a.pisi () ; cout << endl;
return 0;
}
// Ovde se poziva destruktor za sva tri objekta.
```



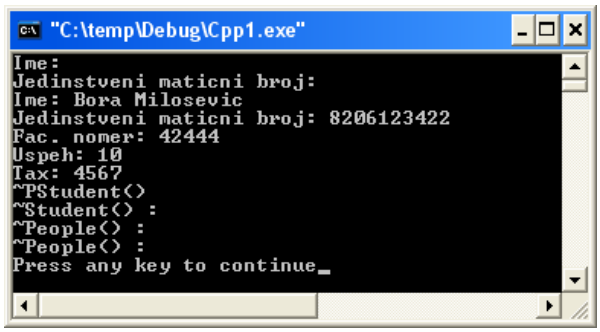
13 Zadatak:

U definisanim klasama People, Student i PStudent sagledati upotrebu konstruktora i destruktora.

```
#include <iostream.h>
#include <string.h>
//deklaracija klase People
```

```
class People
{public:
People(char * = "", char * = "");
void PrintPeople() const;
~People();
private:
char * name;
char * egn;
};
//definicija konstruktora klase People
People::People(char *str, char *num)
{name = new char[strlen(str)+1];
strcpy(name, str);
egn = new char[11];
strcpy(egn, num); }
//definicija metode PrintPeople
void People::PrintPeople() const
{cout << "Ime: " << name << endl;
cout << "Jedinstveni maticni broj: " << egn << endl; }
//definicija destruktora klase People
People::~People()
{cout << "~People() : " << endl;
delete name;
delete egn; }
//deklaracija klase Student
class Student : public People
{ public:
Student(char * = "", char * = "", long = 0, double = 0);
void PrintStudent() const;
~Student()
{cout << "~Student() : " << endl; }
private:
long facnom;
double usp; };
//definicija konstruktora klase Student
Student::Student(char *str, char * num, long facn,
double u) : People(str, num)
{facnom = facn;
usp = u; }
//definicija metode PrintStudent
void Student::PrintStudent() const
{PrintPeople();
cout << "Fac. nomer: " << facnom << endl;
cout << "Uspeh: " << usp << endl; }
// deklaracija klase PStudent
class PStudent : public Student
{ public:
PStudent(char * = "", char * = "", long = 0,
double = 0, double = 0);
~PStudent()
{cout << "~PStudent() \n"; }
void PrintPStudent() const;
protected:
```

```
double tax; } ;
//definicija konstruktora klase PStudent
PStudent::PStudent(char *str, char *num, long facn,
double u, double t) : Student(str, num, facn, u)
{tax = t; }
//definicija metode PrintPStudent
void PStudent::PrintPStudent() const
{PrintStudent();
cout << "Tax: " << tax << endl; }
void main()
{People pe;
pe.PrintPeople();
PStudent PStud("Bora Milosevic", "8206123422", 42444, 10.0, 4567);
PStud.PrintPStudent();
}
```

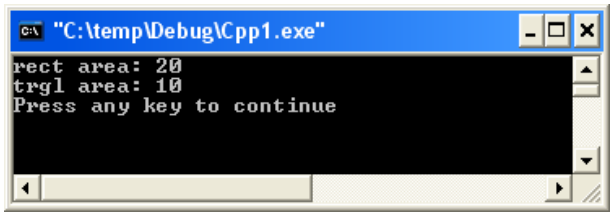


14 Zadatak:

PRIMER klase CPolygon i naslednih klasa CRectangle I CTriangle

```
#include<iostream.h>
class CPolygon {
protected:
int width,height;
public:
void set_values (int a,int b)
{ width=a;height=b; }
};
class CRectangle: public CPolygon { //klasa naslednik:public osnovna klasa
public:
int area (void)
{ return (width * height); }};
class CTriangle: public CPolygon { //klasa naslednik:public osnovna klasa
public:
int area (void)
{ return (width * height/2); }};
main() {
    CRectangle rect;//instanca, novi objekat
    CTriangle trgl; //instanca, novi objekat
    rect.set_values(4,5);
    trgl.set_values(4,5);
```

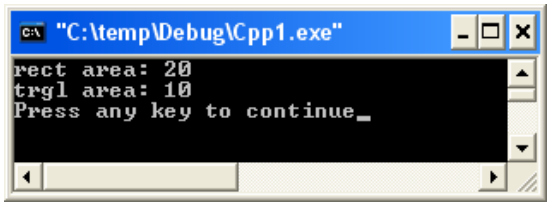
```
        cout<< "rect area: "<<rect.area()<<endl;
        cout<< "trgl area: "<<trgl.area()<<endl;
    return 0;
    }
```



15 Zadatak:

PRIMER - Polimorfizam:

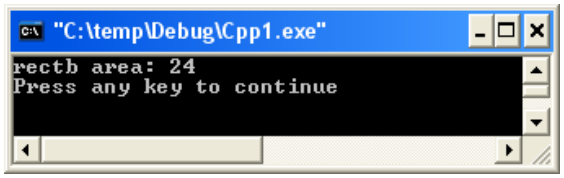
```
#include<iostream.h>
class CPolygon {
protected:
int width,height;
public:
void set_values (int a,int b)
{ width=a;height=b; }
};
class CRectangle: public CPolygon { //klasa naslednik:public osnovna klasa
public:
int area (void)
{ return (width * height); }};
class CTriangle: public CPolygon { //klasa naslednik:public osnovna klasa
public:
int area (void)
{ return (width * height/2); }};

main() {      CRectangle rect;
    CTriangle trgl;
    CPolygon *ppoly1=&rect;
    CPolygon *ppoly2=&trgl;
    ppoly1->set_values(4,5);
    ppoly2->set_values(4,5);
    cout<< "rect area: "<<rect.area()<<endl;
    cout<< "trgl area: "<<trgl.area()<<endl;
return 0;}
```



16 Zadatak:

PRIMER Prijateljske funkcije:

```
//PRIMER klase CRectangle sa uvodjenjem prijateljske funkcije
#include<iostream.h>
class CRectangle {
int width,height;
public:
void set_values (int,int);  //konstruktor kao funkcija
int area (void) { return (width*height); }//f-ja clanica klase
friend CRectangle duplicate (CRectangle);
};
void CRectangle::set_values (int a, int b)
{ width=a; height=b; }
CRectangle duplicate (CRectangle rectparam)
{CRectangle rectres;
rectres.width=rectparam.width*2;
rectres.height=rectparam.height*2;
return (rectres); }

main(){
      CRectangle recta,rectb;//novi objekat, instanca klase
      recta.set_values(2,3);
      rectb=duplicate (recta);
      cout<< "rectb area: "<<rectb.area()<<endl;
}
```

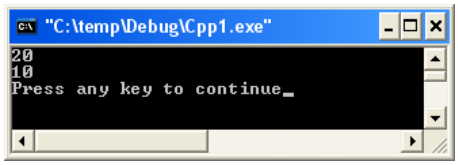


16 Zadatak:

PRIMER Virtualne funkcije:

```
#include<iostream.h>
class CPolygon
{protected:
 int width,height;
 public:
 void set_values (int a,int b)
{ width=a;height=b; }
 virtual int area (void) =0;};
class CRectangle: public CPolygon //klasa naslednik:public osnovna klasa
{public:
int area (void)
{ return (width * height); }};
class CTriangle: public CPolygon { //klasa naslednik:public osnovna klasa
```

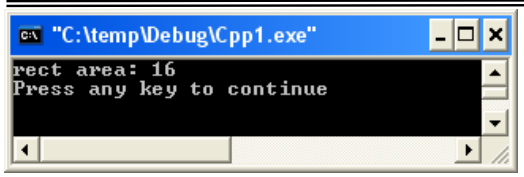
```
public:
int area (void)
{ return (width * height/2); }};

main() {
      CRectangle rect;
      CTriangle trgl;
      CPolygon *ppoly1=&rect;
      CPolygon *ppoly2=&trgl;
      ppoly1->set_values(4,5);
      ppoly2->set_values(4,5);
      cout<< ppoly1->area()<<endl;
      cout<< ppoly2->area()<<endl;
return 0;}
```



17 Zadatak:

PRIMER 13 – Prijateljske klase:

```
//PRIMER klase CRectangle sa uvodjenjem prijateljske klase CSquare
#include<iostream.h>
class CSquare;
class CRectangle
{int width,height;
public:
int area (void) {return (width*height);}    //f-ja clanica klase
void convert (CSquare a);};
class CSquare
{private:
int side;
public:
void set_side (int a)
{ side=a; }
friend class CRectangle; };
void CRectangle::convert (CSquare a)
{ width=a.side;  height=a.side; }

main(){
CSquare sqr;
CRectangle rect;
      sqr.set_side(4);
      rect.convert(sqr);
      cout<< "rect area: "<<rect.area()<<endl;
return 0;}
```



18 Zadatak:

Demonstracija deklaracije klase i definicija objekata

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

class person
{
public:
  string name;
  int age;
};

int main ()
{
  person a, b;
  a.name = "Bora";
  b.name = "Srdjan";
  a.age = 60;
  b.age = 40;
  cout << a.name << ": " << a.age << endl;
  cout << b.name << ": " << b.age << endl;
  return 0;
}
```

Izlaz