Mira Nikolić

Zbirka primera iz programskog jezika C++

Sadržaj:

Objektno orijentisano programiranje
Klase
Podela koda programa u module
Konstruktor kopiranja
Preklapanje operacija
Prijateljske funkcije
Nasleđivanje
Virtuelne funkcije, polimorfizam

Objektno orijentisano programiranje

Osnovne karakteristike objektno orijentisanog programiranja su
- Kapsuliranje
- Nasleđivanje
- Polimorfizam

Klase sadrže polja podataka i funkcije članice. One definišu operacije na objektima.
Objekat je primerak klase.

Elemente klase čine podaci-članovi i funkcije-članice (metode)

1) Ako je u klasi samo prototip funkcije, ona se definiše izvan klase.
Sintaksa:

```
Povratni_tip ime_klase :: ime_funkcije(argumenti)
        {deklaracije_i_iskazi}
```

2) Ako je funkcija definisana u okviru klase, to je inline funkcija.

Sintaksa poziva funkcije:

```
Ime_objekta.ime_funkcije(stvarni_parametri);
```

Sintaksa klase:

```
Class <ime_klase>
{
[private:]      //neobavezno, podrazumeva se
        <opis skrivenih delova klase>
public:
        <opis javnih delova klase>
};
```


Klase

```
//  Demonstrira deklaraciju klase i  definiciju objekta u klasi

#include <iostream>
using namespace std;
class Cat                                   // deklaracija klase
{
public:                                     // slede javni clanovi klase
        int itsAge;
        int itsWeight;
};                                  // na kraju declaracije klase stavi se ;

void main()
{
        Cat Frisky;                         // definicija objekta u klasi
        Frisky.itsAge = 5;     // dodela vrednosti podatku clanu klase
        cout << "Frisky is a cat who is " ;
        cout << Frisky.itsAge << " years old.\n";
                // tacka upucuje na clana klase - kao kod struktura
}
Frisky is a cat who is 5 years old.
Press any key to continue

// Demonstrira deklaraciju klasa i definiciju metoda

#include <iostream>
using namespace std;

class Cat                           // pocetak deklaracije klase
{
  public:                           // pocetak javne sekcije
        int GetAge();               // ovo su funkcije clanice klase
        void SetAge (int age);      // to su prototipovi
        void Meow();
  private:                          // pocetak privatne sekcije
        int itsAge;
        // deklaracija podatka clana klase u kom je sadrzaj klase
};                                  // primeti: na kraju deklaracije klase stoji ;
```

```
// definicija GetAge, javne funkcije clanice klase
                //koja vraca vrednost clana klase itsAge
int Cat::GetAge()               // sintaksa: ime_klase::ime_funkcije
{// to znaci da je ova funkcija GetAge definisana  u ovoj klasi Cat
        return itsAge;
}

// def. SetAge, javne funkcije clanice klase koja postavlja
        //vrednost clana klase itsAge na vrednost  parametra age
void Cat::SetAge(int age)
{
        itsAge = age;
}

// definicija Meow, javne funkcije clanica klase koja stampa "meow"
void Cat::Meow()
{
        cout << "Meow.\n";
}

// glavni program kreira objekat - promenljivu tipa klase cat,
        // pise mjau, govori nam starost, ponovo mjau
void main()
{
        Cat Frisky;             // kreiranje objekta: Frisky klase Cat
        Frisky.SetAge(5);
                // tacka je u pozivu funkcije; broj godina Friskija je 5
        Frisky.Meow();          // za Friskija se ispisuje mjau
        cout << "Frisky is a cat who is " ;
        cout << Frisky.GetAge() << " years old.\n";
                                // za Friskija se ispisuje starost
        Frisky.Meow();          // za Friskija se ispisuje mjau
  }
```

Meow.
Frisky is a cat who is 5 years old.
Meow.
Press any key to continue

Klase – konstruktori i destruktori

```
// Demonstrira deklaraciju konstruktora i destruktora za klasu Cat

#include <iostream>
using namespace std;


class Cat                               // deklaracija klase
  {
   public:                              //  public sekcija
     Cat(int initialAge);               // konstruktor
    ~Cat();                             // destruktor
    int GetAge();                       // funkcije članica
    void SetAge(int age);
    void Meow();
  private:                              // private sekcija
    int itsAge;                         // podatak član
 };


 Cat::Cat(int initialAge)               // konstruktor
 {
   itsAge = initialAge;
 }

 Cat::~Cat()                    // destruktor, ne radi ništa
 {
 }

 // GetAge, funkcija članica, vraća vrednost članu klase itsAge
  int Cat::GetAge()
 {
   return itsAge;
 }
```

```
// SetAge, funkcija članica, postavlja vrednost itsAge na vrednost
ulaznog parametra age
  void Cat::SetAge(int age)
  {
     itsAge = age;
  }

  // Meow metoda, štampa mjau na ekran
  void Cat::Meow()
{
     cout << "Meow.\n";
  }

void main()
  {
    Cat Frisky(5);        // kreira objekat Frisky,
                          // konstruktor inicijalizuje član klase itsAge na 5
    Frisky.Meow();     // poziv funkcije članice Meow.
    cout << "Frisky is a cat who is " ;
    cout << Frisky.GetAge() << " years old.\n";
                          // poziv funkcije članice GetAge.
    Frisky.Meow();     // poziv funkcije članice Meow.
    Frisky.SetAge(7);
          // poziv funkcije članice SetAge, stvarni param. u pozivu je 7
    cout << "Now Frisky is " ;
    cout << Frisky.GetAge() << " years old.\n";
                          // poziv funkcije članice GetAge
  }
```

Meow.
Frisky is a cat who is 5 years old.
Meow.
Now Frisky is 7 years old.
Press any key to continue

Klase – inline funkcije

```
#include <iostream>
using namespace std;
  class tacka                                   // deklaracija klase
{
private:
     float x,y;                                 // podaci clanovi
public:
      tacka();                                  // konstruktor
     ~tacka();                                  // destruktor
void translacija(float dx, float dy)
                          {x+=dx;y+=dy;};       // inline funkcije
void pozicija()                 {cout<<"x="<<x<<" y="<<y<<endl;}
};
          tacka::tacka()          // definicija konstruktora tacka
          {         x=0; y=0;
                    cout<<"Tacka je inicijalizovana!"<<endl;          }

          tacka::~tacka()         // destruktor
          {         cout<<"Tacka je uklonjena!"<<endl;        }

void main()                       // Glavni test program
{
          tacka t1;                             // inicijalizacija tacke
          t1.translacija(5.5,5.5);              // pomeranje tacke
          t1.pozicija();                        // ispis pozicije tacke
          tacka t2;                             // inicijalizacija tacke
          t2.translacija(10,10);                // pomeranje tacke
          t2.pozicija();                        // ispis pozicije tacke
 }
```

Tacka je inicijalizovana!
x=5.5 y=5.5
Tacka je inicijalizovana!
x=10 y=10
Tacka je uklonjena!
Tacka je uklonjena!
Press any key to continue

Podela koda programa u module

Kod velikih programa, kod programa može se podeliti u dve datoteke, datoteku zaglavlja i source kod sa definicijama funkcija.

//ovo je datoteka sa definicijom klase i funkcija, primer13.h

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

class tacka                                  // deklaracija klase
{
private:
    float x,y;                               // podaci clanovi
    public:
     tacka();                                // konstruktor
    ~tacka();                                // destruktor
    void translacija(float dx, float dy)
                        {x+=dx;y+=dy;};      // inline funkcije
    void pozicija()
                        {cout<<"x="<<x<<" y="<<y<<endl;}
};
```

// ovo je datoteka sa kodom glavnog programa, primer13.cpp

```
#include "primer13.h"
#include <iostream>
using namespace std;

tacka::tacka()                  // definicija konstruktora tacka
{
        x=0; y=0;
        cout<<"Tacka je inicijalizovana!"<<endl;
}

tacka::~tacka()                 // destruktor
{       cout<<"Tacka je uklonjena!"<<endl;      }
```

```
void main()                     // Glavni test program
{
        tacka t1;               // inicijalizacija tacke
        t1.translacija(5.5,5.5); // pomeranje tacke
        t1.pozicija();          // ispis pozicije tacke
        tacka t2;               // inicijalizacija tacke
        t2.translacija(10,10);  // pomeranje tacke
        t2.pozicija();          // ispis pozicije tacke
}
```

Tacka je inicijalizovana!
x=5.5 y=5.5
Tacka je inicijalizovana!
x=10 y=10
Tacka je uklonjena!
Tacka je uklonjena!
Press any key to continue

Vidljivost clanova klase

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

class tacka                                  // deklaracija klase
{
  float x,y;                                 // podaci clanovi
public:                                      // inline funkcije
  void inic(float a, float b)                {x=a; y=b;}

  void translacija(float dx, float dy)              {x+=dx;y+=dy;}
  void pozicija()            {cout<<"x="<<x<<" y="<<y<<endl;}
  float dist(tacka t);
 };

float tacka::dist(tacka t)
  {return sqrt((x-t.x)*(x-t.x)+(y-t.y)*(y-t.y));}

void main()              // Glavni test program
{       tacka t1,t2;
        t1.inic(0,0);                        // inicijalizacija tacke
        t1.translacija(2.5,3.0);             // pomeranje tacke
        t1.pozicija();                       // ispis pozicije tacke
        t2.inic(3,5);                        // inicijalizacija tacke
        t2.translacija(10,10);               // pomeranje tacke
        t2.pozicija();                       // ispis pozicije tacke
        cout<<"Rastojanje tacaka t1 i t2 je "<<t1.dist(t2)<<endl;       }
```

// Vidljivost clanova klase, 2. nacin, upotreba prijateljskih funkcija

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

 class tacka                         // deklaracija klase
{
   float x,y;                        // podaci clanovi
public:                              // inline funkcije
   void inic(float a, float b)              {x=a; y=b;}

   void translacija(float dx, float dy)     {x+=dx;y+=dy;}
   void pozicija()            {cout<<"x="<<x<<" y="<<y<<endl;}
   friend float dist(tacka t1, tacka t2);
 };

float dist(tacka t1, tacka t2)
  {return sqrt((t1.x-t2.x)*(t1.x-t2.x)+(t1.y-t2.y)*(t1.y-t2.y));}

void main()              // Glavni test program
{       tacka t1,t2;
        t1.inic(0,0);                        // inicijalizacija tacke
        t1.translacija(2.5,3.0);             // pomeranje tacke
        t1.pozicija();                       // ispis pozicije tacke
        t2.inic(3,5);                        // inicijalizacija tacke
        t2.translacija(10,10);               // pomeranje tacke
        t2.pozicija();                       // ispis pozicije tacke
        cout<<"Rastojanje tacaka t1 i t2 je "<<dist(t1,t2)<<endl;     }
```

Pokazivac this

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

class DvaZnaka                  // deklaracija klase
{
   char c1,c2;                  // podaci clanovi
public:                         // inline funkcije
        void init(char b)               {c2=b; c1=b+1;}

        DvaZnaka &increment()   {c1++;c2++; return (*this);}
        void *MojaAdresaJe()    {return this;}
        void print()            {cout<<c1<<c2<<endl;}
 };

void main()
{
        DvaZnaka a,b;
        a.init('A');            // inicijalizacija objekta
        b.init('B');            // inicijalizacija objekta
        cout<<"Na adresi: "<<a.MojaAdresaJe()<<" je: ";
                                // vraca adresu objekta a
        a.print();              //ispis BA
        cout<<"Na adresi: "<<b.MojaAdresaJe()<<" je: ";
                                // vraca adresu objekta b
        b.print();              //ispis CB
        cout<<"Na adresi: "<<b.MojaAdresaJe()<<" je: ";
                                // vraca adresu objekta b
        b.increment().print();  //ispis DC
 }
```

```
Na adresi: 0012FED4 je: BA
Na adresi: 0012FEC8 je: CB
Na adresi: 0012FEC8 je: DC
  Press any key to continue
```

Konstruktor kopiranja

Služi da se kreira objekat iz postojećeg objekta. Da se pri oslobađanju memorije (operatori new i delete) ne bi izbrisali podaci za oba objekta, mora se objekat proslediti preko referenci ili pokazivača!

Sintaksa za konstruktor kopiranja:

```
Ime_klase::ime_klase(const ime_klase&)
                        {……}
```

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

 class CAT
 {
  public:
        CAT();                  // podrazumevani konstruktor
        CAT (const CAT &);      // konstruktor kopiranja
        ~CAT();                 // destruktor
        int GetAge() const { return *itsAge; }      // inline funkcije
        int GetWeight() const { return *itsWeight; }
        void SetAge(int age) { *itsAge = age; }
 private:
         int *itsAge;           // pokazivaci na podatke
         int *itsWeight;
 };

 CAT::CAT()                     // definicija konstruktora
{
     itsAge = new int;
     itsWeight = new int;
     *itsAge = 5;
     *itsWeight = 9;
 }
```

```
CAT::CAT(const CAT & rh)        // definicija konstruktora kopiranja
{    itsAge = new int;
     itsWeight = new int;
    *itsAge = rh.GetAge();
    *itsWeight = rh.GetWeight();  }

CAT::~CAT()                                  // definicija destruktora
{   delete itsAge;
     itsAge = 0;
     delete itsWeight;
     itsWeight = 0;  }

void main()
{
     CAT frisky;
     cout << "frisky's age: " << frisky.GetAge() << endl;
     cout << "Setting frisky to 6...\n";
     frisky.SetAge(6);
     cout << "Creating piksy from frisky\n";
     CAT piksy(frisky);        //kreira se objekat piksy iz objekta frisky
     cout << "frisky's age: " <<     frisky.GetAge() << endl;
     cout << "piksy's age: " << piksy.GetAge() << endl;
     cout << "setting frisky to 7...\n";
     frisky.SetAge(7);                  //samo frisky menja godine
     cout << "frisky's age: " <<     frisky.GetAge() << endl;
     cout << "piksy's age: " << piksy.GetAge() << endl;
}
```

frisky's age: 5
Setting frisky to 6...
Creating piksy from frisky
frisky's age: 6
piksy's age: 6
setting frisky to 7...
frisky's age: 7
piksy's age: 6
Press any key to continue

Preklapanje operacija

Može se u okviru klase definisati nova operacija @ nad objektima.
Sintaksa:
tip_rezultata ime_klase::operator@(spisak_parametara)
{telo_funkcije}

```
#include <iostream>
using namespace std;
class vektor                              // deklaracija klase
{
        int x,y;                                  // podaci clanovi
public:
        vektor operator+(vektor t);          // preklapanje operacije +
        const vektor &operator=(const vektor &t);  // preklapanje  =
        void show();                              // prikaz koordinata
        void assign(int x1, int y1);          // dodela vrednosti
};

vektor vektor::operator+(vektor t)// definicija preklapanja operacije +
{       vektor temp;                    //lokalni vektor
        temp.x=x+t.x;
                //koordinate x i y se uvecavaju za koord. vektora t
        temp.y=y+t.y;
        return temp;  }                 //vracaju se temp.x i temp.y

const vektor &vektor::operator=(const vektor &t)
{                                  // definicija preklapanja operacije =
        if (&t==this) return *this;
                //ako se dodeljuje samom sebi, tada ne radi nista
        x=t.x;  // x i y dobijaju vrednosti vektora argumenta t
        y=t.y;
        return *this;   //vraca se vrednost tekuceg objekta, a to je x,y
}

void vektor::show()               // ostale funkcije, ispisi x i y
        {       cout<<"x= "<<x<<"\ty= "<<y<<endl;       }
void vektor::assign(int x1, int y1)  // dodeli vrednost za x i y
        {       x=x1;  y=y1;  }
```

```
void main()             // Glavni test program
{
        vektor a,b,c;                           // inicijalizacija vektora
        a.assign(5,5);
        b.assign(10,10);
        a.show();       b.show();
        c=a+b;          c.show();               // preklopljeni operatori
        c=a+b+c;        c.show();
        c=b=a;                          // visestruka dodela
        c.show();       b.show();
        cout<<"============================"<<endl;
        vektor v[5],vs;
        for (int i=0;i<5;i++)
        {     v[i].assign(i,i+1);
            cout<<"vektor v["<<i<<"]->\t";
            v[i].show();          }
        vs.assign(0,0);                 // inicijalizacija vektora zbira
        cout<<"Zbir vektora ->"<<endl;
        for (i=0;i<5;i++)         vs=vs+v[i];
        vs.show();
        }
```

```
x= 5    y= 5
x= 10   y= 10
x= 15   y= 15
x= 30   y= 30
x= 5    y= 5
x= 5    y= 5
============================
vektor v[0]->   x= 0    y= 1
vektor v[1]->   x= 1    y= 2
vektor v[2]->   x= 2    y= 3
vektor v[3]->   x= 3    y= 4
vektor v[4]->   x= 4    y= 5
Zbir vektora ->
x= 10   y= 15
Press any key to continue
```

```
// Preklapanje operacija, ++ prefiksno i postfiksno

#include <iostream>
using namespace std;

class vektor                            // deklaracija klase
{
   int x,y;                             // podaci clanovi
public:
  vektor operator+(vektor t);           // preklapanje operacije +
  const vektor &operator=(const vektor &t);
                        // preklapanje operacije =
  vektor operator++();          // preklapanje prefiksne operacije ++
  vektor operator++(int);       // preklapanje postfiksne operacije ++
  void show();                                  // prikaz koordinata
  void assign(int x1, int y1);                  // dodela vrednosti
};

vektor vektor::operator+(vektor t)// definicija preklapanja operacije +
{       vektor temp;                    //lokalni vektor
        temp.x=x+t.x;
                //koordinate x i y se uvecavaju za koord. vektora t
        temp.y=y+t.y;
        return temp;            }       //vracaju se temp.x i temp.y

const vektor &vektor::operator=(const vektor &t)
                // definicija preklapanja operacije=
{       if (&t==this) return *this;
                //ako se dodeljuje samom sebi, tada ne radi nista
        x=t.x;
        //koordinate x i y dobijaju vrednosti vektora argumenta t
        y=t.y;
        return *this;           }
                //vraca se vrednost tekuceg objekta, a to je x,y

vektor vektor::operator++()
                // definicija preklapanja operacije ++, prefiksno
{       x++;    y++;
        return *this;           }//vraca vrednost izmenjenog vektora
```

```
vektor vektor::operator++(int)  // definicija preklapanja ++, postfiksno
{       vektor temp=*this;
        x++;    y++;
        return temp;          }//vraca vrednost vektora pre uvecavanja

void vektor::show()                     // ostale funkcije, ispisi x i y
{       cout<<"x= "<<x<<"\ty= "<<y<<endl;       }

void vektor::assign(int x1, int y1)  // dodeli vrednost za x i y
{       x=x1;  y=y1;  }

void main()    // Glavni test program
{       vektor a,b,c;                           // inicijalizacija vektora
        a.assign(5,5);
        b.assign(10,10);
        cout<<"a:   ";a.show();
        cout<<"b:   ";b.show();
        c=a+b;                                  // preklopljeni operatori
        cout<<"c=a+b:   ";c.show();
        cout<<"postfiksno inkrementiranje c=a++ za a=5"<<endl;
        c=a++;                          // postfiksno inkrementiranje
        cout<<"a:   "; a.show();
        cout<<"c:   ";c.show();
        cout<<"prefiksno inkrementiranje c=++b za b=10"<<endl;
        c=++b;                          // prefiksno inkrementiranje
        cout<<"b:   ";b.show();
        cout<<"c:   ";c.show();
}
```

```
a:   x= 5       y= 5
b:   x= 10      y= 10
c=a+b:   x= 15  y= 15
postfiksno inkrementiranje c=a++ za a=5
a:   x= 6       y= 6
c:   x= 5       y= 5
prefiksno inkrementiranje c=++b za b=10
b:   x= 11      y= 11
c:   x= 11      y= 11
Press any key to continue
```



```
// Preklapanje operacije [ ] koriscenjem funkcije clanice klase

#include <iostream>
using namespace std;

class vektor                            // deklaracija klase
{    int v[3];                  // vektor dimenzije 3, privatni podatak
public:
   vektor() {for(int i=0;i<3;i++)v[i]=0;}       // konstruktor
   vektor(const int a[ ])                       // konstruktor
                {for(int i=0;i<3;i++)v[i]=a[i];}
   vektor operator+(vektor t);                  // preklapanje operacije +
   const vektor &operator=(const vektor &t);
                                        // preklapanje operacije =
   int &operator[](int i);                      // preklapanje operacije [ ]
   void show(void);                             // ispis vektora
};

vektor vektor::operator+(vektor t)
                                // definicija preklapanja operacije +
{
        vektor temp=*this;              //lokalni vektor
        for (int i=0;i<3;i++)temp.v[i]+=t.v[i];
                        //koordinate v[i] se uvecavaju za argument t.v[i]
        return temp;            //vraca se lokalni vektor
}

const vektor &vektor::operator=(const vektor &t)
                        // definicija preklapanja operacije=
{
        if (&t==this) return *this;
                //ako se dodeljuje samom sebi, tada ne radi nista
        for (int i=0;i<3;i++) v[i]+=t.v[i];
          return *this;//vraca se vrednost tekuceg objekta, a to je v[i]
}

int &vektor::operator[](int i) // definicija  operacije indeksiranja [ ]
{       return v[i];     }        //vraca vrednost v[i] za argument i
```

```
void vektor::show(void)              // definicija ispisa vektora
{       for (int i=0;i<3;i++)
                        cout<<"v["<<i<<"]= "<<v[i]<<"\t";
        cout<<endl;
}

void main()                     // Glavni test program
{
        int a[3]={1,2,3};                       // inicijalizacija vektora
        int b[3]={10,20,30};
        vektor v1(a), v2(b), v3;
        v3=v1+v2;
                        //ispis koriscenjem preklopljene operacije
        cout<<"Ispis v3 koriscenjem preklopljene operacije"<<endl;
        for (int i=0;i<3;i++)
                cout<<"v["<<i<<"]= "<<v3[i]<<"\t";
        cout<<endl;
                        //ispis koriscenjem funkcije clana
        cout<<"Ispis v3 koriscenjem funkcije clana"<<endl;
        v3.show();
                v1[0]=100;//upucivanje na v1.v[0] kojem se dodeljuje
                // vrednost 100. v1[i] je poziv funkcije u levom delu
                // operacije dodele: v1.operator[](i)
        v1[1]=201;
        v1[2]=302;      //ispis koriscenjem funkcije clana
        cout<<"Ispis v1 koji je inicijalizovan koriscenjem operacije
indeksiranja"<<endl;
        v1.show();
}
```

Ispis v3 koriscenjem preklopljene operacije
v[0]= 11 v[1]= 22 v[2]= 33
Ispis v3 koriscenjem funkcije clana
v[0]= 11 v[1]= 22 v[2]= 33
Ispis v1 koji je inicijalizovan koriscenjem operacije indeksiranja
v[0]= 100 v[1]= 201 v[2]= 302
Press any key to continue

```
// Preklapanje operacije + koriscenjem prijateljske funkcije

#include <iostream>
using namespace std;

class vektor                            // deklaracija klase
{
private:
    int x,y;                            // podaci clanovi
public:
    friend vektor operator+(vektor t1,vektor t2);
                                // preklapanje operacije +
    const vektor &operator=(const vektor &t);
                                // preklapanje operacije =
    void show();                // prikaz koordinata
    void assign(int x1, int y1)         // dodela vrednosti
                {x=x1;y=y1;};
};

vektor operator+(vektor t1,vektor t2) // definicija preklapanja op. +
{
        vektor temp;                    //lokalni vektor
        temp.x=t1.x+t2.x;
                //koordinate x i y se uvecavaju za koord. vektora t
        temp.y=t1.y+t2.y;
        return temp;            //vracaju se temp.x i temp.y
}

const vektor &vektor::operator=(const vektor &t)
{                               // definicija preklapanja operacije=
        if (&t==this) return *this;
                //ako se dodeljuje samom sebi, tada ne radi nista
        x=t.x;
        //koordinate x i y dobijaju vrednosti vektora argumenta t
        y=t.y;
        return *this;   //vraca se vrednost tekuceg objekta, a to je x,y
}
```

```
void vektor::show()          // ostale funkcije, ispisi x i y
{       cout<<"x= "<<x<<"\ty= "<<y<<endl;       }

void main()             // Glavni test program
{
        vektor a,b,c;           // inicijalizacija vektora
        a.assign(5,5);
        b.assign(10,10);
        cout<<"a:   ";a.show();
        cout<<"b:   ";b.show();
        c=a+b;                                  // preklopljeni operatori
        cout<<"c=a+b:   ";c.show();
        c=a+b+c;                                // visestruka dodela
        cout<<"c=a+b+c:   ";c.show();
}
```

a: x= 5 y= 5
b: x= 10 y= 10
c=a+b: x= 15 y= 15
c=a+b+c: x= 30 y= 30
Press any key to continue

Nasleđivanje

```
// Bazne i izvedene klase - klasa tacka i klasa krug

#include <iostream>
using namespace std;

class tacka                                     // deklaracija bazne klase
{
protected:
   float x,y;                                   // podaci - polozaj tacke
public:
   tacka(float x1=0.0, float y1=0.0)            // konstruktor
        {       cout<<"Pozvan je konstruktor klase tacka"<<endl;
                x=x1; y=y1;   }
   ~tacka()                                     // destruktor
        {cout<<"Pozvan je destruktor klase tacka"<<endl;}

void inicXY(float x1=0.0, float y1=0.0)
        {x=x1;y=y1;};                   // proizvoljan polozaj tacke
void translacija(float dx, float dy)
        {x+=dx;y+=dy;};                 // pomeraj tacke
void zaglavlje()
        {cout<<"Tacka se nalazi na poziciji"<<endl;}
void pozicija()
        {cout<<"x="<<x<<" y="<<y<<endl;}
void izvestaj()                 {zaglavlje();pozicija();}
};                                              // kraj bazne klase tacka

class krug:public tacka                 // deklaracija izvedene klase krug
{
  protected:
        float r;                        // podaci - poluprecnik kruga
  public:
      krug(float x1=0.0, float y1=0.0, float r1=0.0):       // konstruktor
                tacka(x1,y1)                    // poziv konstruktora tacka
        {       cout<<"Pozvan je konstruktor klase krug"<<endl;
                r=r1;   }
```

```
    ~krug()                                          // destruktor
       {cout<<"Pozvan je destruktor klase krug"<<endl;}

void inicR(float r1=0.0)      {r=r1;};// promena velicine poluprecnika
void zaglavlje()
       {cout<<"Krug poluprecnika "<<r<<" se nalazi na
poziciji"<<endl;}
void izvestaj()               {zaglavlje();pozicija();}
                                    // preklapanje funkcija
};                                  // kraj izvedene klase krug


void main()  // Glavni test program
{
            krug A(1.5,2.5,5.0);          // inicijalizacija kruga
            A.translacija(5.5,5.5);       // pomeranje kruga
            A.izvestaj();                 // ispis pozicije kruga
            A.inicR(10.0);                // promena poluprecnika
            A.izvestaj();                 // ispis pozicije kruga
            A.translacija(3.0,3.0);       // pomeranje kruga
            A.inicXY(2.0,2.0);            // promena pozicije kruga
            A.izvestaj();                 // ispis pozicije kruga
   }
```

Pozvan je konstruktor klase tacka
Pozvan je konstruktor klase krug
Krug poluprecnika 5 se nalazi na poziciji
x=7 y=8
Krug poluprecnika 10 se nalazi na poziciji
x=7 y=8
Krug poluprecnika 10 se nalazi na poziciji
x=2 y=2
Pozvan je destruktor klase krug
Pozvan je destruktor klase tacka
Press any key to continue

```
// Bazne i izvedene klase - klasa tacka,klasa krug i klasa prsten

#include <iostream>
using namespace std;

class tacka                   // deklaracija bazne klase
{
protected:
   float x,y;                 // podaci - polozaj tacke
public:
    tacka(float x1=0.0, float y1=0.0)       // konstruktor
       {      cout<<"Pozvan je konstruktor klase tacka"<<endl;
       x=x1; y=y1;   }
    ~tacka()                                // destruktor
       {      cout<<"Pozvan je destruktor klase tacka"<<endl;       }
    void inicXY(float x1=0.0, float y1=0.0)        {x=x1;y=y1;};
                                    // proizvoljan polozaj tacke
    void translacija(float dx, float dy)           {x+=dx;y+=dy;};
                                    // pomeraj tacke
    void zaglavlje()  {cout<<"Tacka se nalazi na poziciji"<<endl;}
    void pozicija()   {cout<<"x="<<x<<" y="<<y<<endl;}
    void izvestaj()   {zaglavlje();pozicija();}
  };                                // kraj bazne klase tacka


class krug:public tacka       // deklaracija izvedene klase krug
{
protected:
   float r;                   // podaci - poluprecnik kruga
public:
   krug(float x1=0.0, float y1=0.0, float r1=0.0): // konstruktor
              tacka(x1,y1)          // poziv konstruktora tacka
       {      cout<<"Pozvan je konstruktor klase krug"<<endl;
    r=r1;     }

    ~krug()                                 // destruktor
       {      cout<<"Pozvan je destruktor klase krug"<<endl;       }
```

```
    void inicR(float r1=0.0)
                {r=r1;};// promena velicine poluprecnika
    void zaglavlje()
     {cout<<"Krug poluprecnika "<<r<<" se nalazi na poziciji"<<endl;}

    void izvestaj()
                {zaglavlje();pozicija();}
   };                                          // kraj izvedene klase krug

class prsten:public krug     // deklaracija izvedene klase prsten
{
protected:
    float r1;                // podaci - poluprecnik spoljasnjeg kruga
    public:
                             // konstruktor
    prsten(float x1=0.0, float y1=0.0, float ru=0.0, float rs=0.0):
                 krug(x1,y1,ru)       // poziv konstruktora krug
        {        cout<<"Pozvan je konstruktor klase prsten"<<endl;
                 r1=rs; }
   ~prsten()                                                 // destruktor
        {        cout<<"Pozvan je destruktor klase prsten"<<endl;       }

    void inicRuRs(float ru,float rs) {r=ru;r1=rs;};
                             // promena dimenzija prstena
    void zaglavlje()
        {cout<<"Prsten poluprecnika "<<r<<","<<r1<<
                 " se nalazi na poziciji"<<endl;}
    void izvestaj()    {zaglavlje();pozicija();}
   };                                  // kraj izvedene klase prsten
```

```
void main()          // Glavni test program
{       prsten A(1.5,2.5,5.0,10);          // inicijalizacija prstena
        A.translacija(5.5,5.5);            // pomeranje prstena
        A.izvestaj();                      // ispis pozicije prstena
        A.inicRuRs(10.0,15.0);             // promena poluprecnika
        A.izvestaj();                      // ispis pozicije prstena
        A.translacija(3.0,3.0);            // pomeranje prstena
        A.inicXY(2.0,2.0);                 // promena pozicije prstena
        A.izvestaj();                      // ispis pozicije prstena
 }
```

Pozvan je konstruktor klase tacka
Pozvan je konstruktor klase krug
Pozvan je konstruktor klase prsten
Prsten poluprecnika 5,10 se nalazi na poziciji
x=7 y=8
Prsten poluprecnika 10,15 se nalazi na poziciji
x=7 y=8
Prsten poluprecnika 10,15 se nalazi na poziciji
x=2 y=2
Pozvan je destruktor klase prsten
Pozvan je destruktor klase krug
Pozvan je destruktor klase tacka
Press any key to continue

Virtuelne funkcije, polimorfizam

```
// Virtuelne funkcije - funkcija zaglavlje ima isti naziv,
// a razlicitu funkciju u razlicitim klasama

#include <iostream>
using namespace std;

class tacka                                  // deklaracija bazne klase
{
protected:
        float x,y;                           // podaci - polozaj tacke
public:
        tacka(float x1=0.0, float y1=0.0)   {       x=x1; y=y1;   }
        void inicXY(float x1=0.0, float y1=0.0)    {x=x1;y=y1;};

        void translacija(float dx, float dy)  {x+=dx;y+=dy;};

        virtual void zaglavlje()
                {cout<<"Tacka se nalazi na poziciji"<<endl;}
        void pozicija()
                {cout<<"x="<<x<<" y="<<y<<endl;}
        void izvestaj()                 {zaglavlje();pozicija();}
};                                           // kraj bazne klase tacka

class krug:public tacka              // deklaracija izvedene klase krug
{
protected:
        float r;                             // podaci - poluprecnik kruga
public:
        krug(float x1=0.0, float y1=0.0, float r1=0.0):     tacka(x1,y1)
                {r=r1; }
        void inicR(float r1=0.0)       {r=r1;};
        void zaglavlje()
        {cout<<"Krug poluprecnika "<<r;
        cout<<" se nalazi na poziciji"<<endl;}
        void izvestaj()
                {zaglavlje();pozicija();}
};                                   // kraj izvedene klase krug
```

```
 class prsten:public krug     // deklaracija izvedene klase prsten
{
protected:
        float r1;                    // podaci - poluprecnik spoljasnjeg kruga
public:
        prsten(float x1=0.0, float y1=0.0, float ru=0.0, float rs=0.0):
                krug(x1,y1,ru)
                {r1=rs;          }
        void inicRuRs(float ru,float rs)       {r=ru;r1=rs;};
        void zaglavlje()
                {cout<<"Prsten poluprecnika "<<r<<","<<r1;
                cout<<" se nalazi na poziciji"<<endl;}
        void izvestaj()
        {zaglavlje();pozicija();}
 };                                          // kraj izvedene klase prsten

 // Glavni test program
void main()
{
        tacka T(5.0,5.0);              // inicijalizacija tacke
        krug K(10.0,10.0,5.0);         // inicijalizacija kruga
        prsten P(2.5,2.5,10,20);       // inicijalizacija prstena
        T.izvestaj();                  // ispis pozicije tacke
        K.izvestaj();                  // ispis pozicije kruga
        P.izvestaj();                  // ispis pozicije prstena
}
```

Tacka se nalazi na poziciji
x=5 y=5
Krug poluprecnika 5 se nalazi na poziciji
x=10 y=10
Prsten poluprecnika 10,20 se nalazi na poziciji
x=2.5 y=2.5
Press any key to continue

Podela koda programa u module 2

```
// Modul primer14.h - Bazne i izvedene klase -  klasa figura, trougao, pravougaonik i krug

#include <iostream>
using namespace std;

class figura                              // deklaracija bazne klase
{
 protected:
     double x,y;                                  // podaci
 public:
     void set_dim(double x1, double y1=0)        {        x=x1; y=y1;   }
     virtual void print()
         {cout<<"Za ovu klasu nije definisana povrsina."<<endl;         }
};

class trougao:public figura  // deklaracija izvedene klase trougao
{
public:
     void print()
         {        cout<<"Trougao sa osnovom "<<x<<" i visinom "<<y;
                  cout<<" ima povrsinu "<<x*y*0.5<<endl; }
};

class pravougaonik:public figura   // deklaracija izvedene klase pravougaonik
{
public:
     void print()
         {        cout<<"Pravougaonik sa stranicama "<<x<<" i "<<y;
                  cout<<" ima povrsinu "<<x*y<<endl;       }
         };
```

```
class krug:public figura              // deklaracija izvedene klase krug
{
public:
      void print()
         {        cout<<"Krug poluprecnika "<<x;
                  cout<< " ima povrsinu "<<x*x*3.14159<<endl;   }
         };


// Kod  sa definicijama funkcija,  ovo je datoteka primer13.cpp
#include "primer14.h"
#include <iostream>
using namespace std;

void izvestaj(figura *f, double x, double y=0)     // Definicija funkcija
         {        f->set_dim(x,y);       f->print();                }


// Glavni test program
void main()
{
         figura f;                          // inicijalizacija objakata
         trougao t;
         pravougaonik p;
         krug k;
         izvestaj(&f,1,2);                  // ispis povrsine figure
         izvestaj(&t,3,4);                  // ispis povrsine trougla
         izvestaj(&p,5,6);                  // ispis povrsine pravougaonika
         izvestaj(&k,8);                    // ispis povrsine kruga
                                            }
```

Za ovu klasu nije definisana povrsina.
Trougao sa osnovom 3 i visinom 4 ima povrsinu 6
Pravougaonik sa stranicama 5 i 6 ima povrsinu 30
Krug poluprecnika 8 ima povrsinu 201.062
Press any key to continue