

OSNOVI PROGRAMIRANJA

Dr Dinu Dragan, dinud@uns.ac.rs

Dr Dušan Gajić, dusan.gajic@uns.ac.rs



POTPROGRAMI

Osnovno o potprogramima ...

- Omogućava dekompoziciju problema na manje, lakše rešive probleme
- Omogućuju **modularnost**, ponovno **korišćenje**
- Implementiraju **semantički zatvoren posao** tako da se mogu koristiti u drugom rešenju
- Čine program mnogo razumljivijim i čitljivijim
- Omogućuje jedan nivo apstrakcije (možemo koristiti potprogram, a da ne znamo kako je on implementiran)
- Generalizuje često korišćeni deo koda
- Mogu biti ugrađene (deo ugrađenih Java paketa) ili ih program implementira i koristi

... Osnovno o potprogramima ...

- Identifikuje se **nazivom**
- Potrebne podatke za rad dobija putem **ulaznih** i/ili **ulazno/izlaznih** parametara
- Rezultate rada prosleđuje **nazivom** i/ili **izlaznim** parametrima
- Telo potprograma predstavlja njegovu implementaciju
- Trebalo bi da izgledaju kao crne kutije
 - Kada koristimo potprogram, ne zanima nas njegov sadržaj, već samo kako radi
 - Umesto da brinete o ogromnoj količini koda, vi ga zatvorite u potprogram i brinete samo o rezultatu tog potprograma
 - Tu gomilu koda vidite i pozivate samo preko imena potprograma

... Osnovno o potprogramima ...

- Ono što je još važno za uspeh crne kutije
 - Interfejs potprograma – mehanizam preko kojeg se unose podaci koje će potprogram obraditi i za koje će izbaciti rezultat; kod Java potprograma su to parametri koji se prosleđuju u pozivu potprograma
 - Da nema interfejsa i mogućnosti zadavanja parametara, potprogram bi uvek radio samo isti zadatak za iste podatke i verovatno davao samo jedan, isti, rezultat (to je ok samo u retkom broju slučajeva, npr. ispisivanje uvek iste poruke)
 - Implementacija potprograma ne treba ništa da zna o spoljnom svetu i programu i/ili potprogramu koji ga poziva, sva komunikacija ide preko interfejsa
 - Interfejs je ujedno i specifikacija potprograma i onoga šta on radi

... Osnovno o potprogramima

- Interfejs potprogram ima sintaksno, ali i semantičko značenje
- Da bismo pravilno pozivali (izvršavali) potprogram moramo znati sintaksu njegov interfejsa
- Da bismo pravilno koristili potprogram, moramo znati i njegovu semantiku (kako bismo ga upotreбили za pravi zadatak)
- *"Here is what you have to do to use me, and here is what I will do for you, guaranteed."*

Statički i nestatički potprogrami

- U Javi svaki potprogram mora biti deklarisan unutar neke klase (u drugim programskim jezicima, postoji mogućnost slobodnih potprograma – koji nisu vezani za neku klasu)
- Klasa može sadržati statičke i nestatičke potprograme
- U programu koji se izvršava statički potprogram pripada samoj klasi (mogu da se pozivaju i ako nije napravljen objekat)
- Nestatičke se mogu koristiti samo kada postoje objekti, jer postaju potprogrami samih tih, konkretnih, objekata
- Više o svemu na sledećem kursu za sada to koristiti plastično (to tako mora...)

Deklarisanje potprograma

tip povratne vrednosti

naziv

formalni parametri

```
static int sab(int a, int b)
```


zaglavlje
potprograma

```
{  
    int rezultat;  
    rezultat = a + b;  
    return rezultat;  
}
```

telo
potprograma

Povratna vrednost potprograma ...

tip povratne vrednosti



```
static int sab(int a, int b)
{
    int rezultat;
    rezultat = a + b;
    return rezultat;
}
```

... Povratna vrednost potprograma ...

- Potprogram može svojim nazivom vratiti neku vrednost
- Ova vrednost postaje vrednost izraza u kome je poziv potprograma
- Tip povratne vrednosti u zaglavlju potprograma određuje kojeg tipa će biti vrednost koju potprogram vraća
- Ako potprogram ne vraća vrednost, povratni tip treba da bude **void**

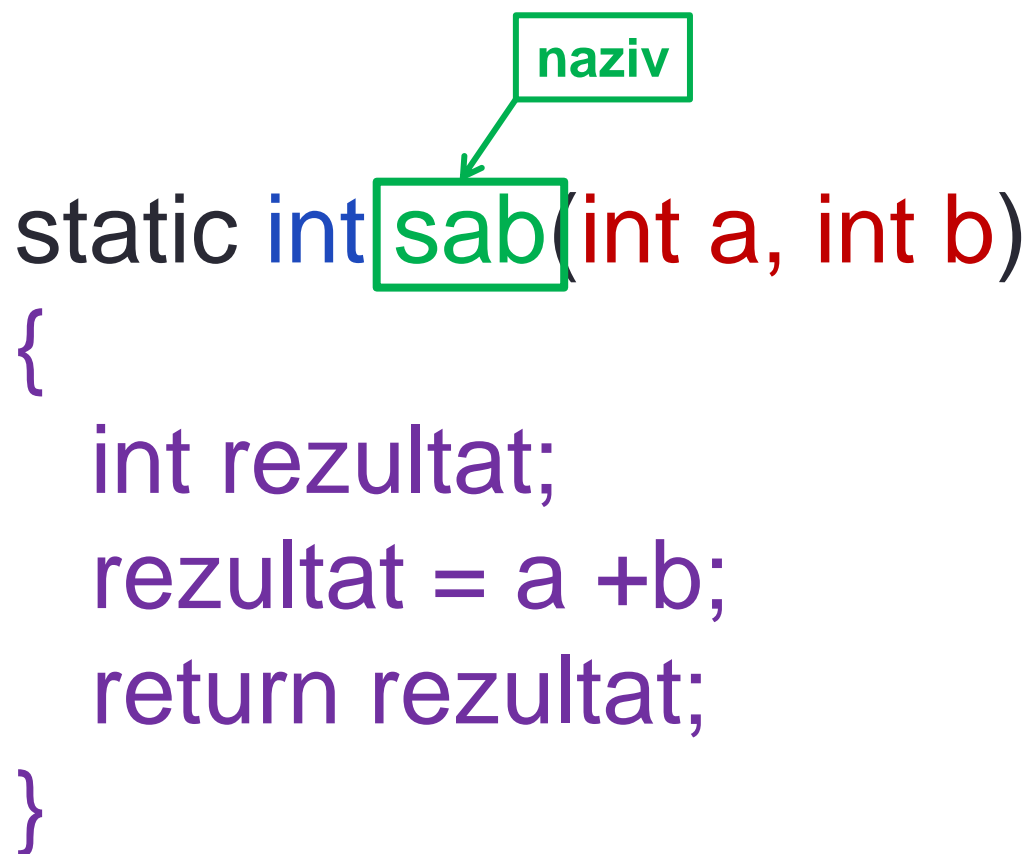
... Povratna vrednost potprograma

- Ako potprogram vraća vrednost, u telu potprograma se mora nalaziti **return** iza koje sledi izraz čiji se rezultat vraća kao povratna vrednost potprograma
- **return** vraća vrednost tipa koji odgovara tipu povratne vrednosti naveden u zaglavlju potprograma (moraju se slagati)
- **return završava izvršavanje potprograma. Sav kod posle return se ignoriše!**
- Može biti više **return** naredbi, ali prva na koju se naiđe završava izvršavanje potprograma i vraća vrednost
- Ako potprogram ne vraća vrednost, ne mora sadržati **return**

Povratna vrednost – primeri

```
static void f1( );    // nema povratne vrednosti
static boolean f2( ); // povratna vrednost tipa boolean
static byte f3( );   // povratna vrednost tipa byte
static char f4( );   // povratna vrednost tipa char
static short f5( );  // povratna vrednost tipa short
static int f6( );    // povratna vrednost tipa int
static long f7( );   // povratna vrednost tipa long
static float f8( );  // povratna vrednost tipa float
static double f9( ); // povratna vrednost tipa double
static String f10( ); // povratna vrednost tipa String
```

Naziv potprograma ...



The diagram illustrates the naming of a subprogram in a C++ code snippet. A green box labeled "naziv" (name) points to the identifier "sab" in the function signature "static int sab(int a, int b)". The code is as follows:

```
static int sab(int a, int b)
{
    int rezultat;
    rezultat = a + b;
    return rezultat;
}
```

... Naziv potprograma

- Identifikuje potprogram
- Mora biti jedinstven
- Koristi se za pozivanje potprograma negde drugde u programu (u nekom drugom potprogramu, main-u, ili klasi/objektu)
- Trebalo bi da ima semantičko značenje tako da se iz naziva lako identifikuje zadatak potprograma

Parametri potprograma ...

formalni parametri

```
static int sab(int a, int b)
{
    int rezultat;
    rezultat = a + b;
    return rezultat;
}
```


... Parametri potprograma ...

- Lista parametra služi da potprogram dobije podatke potrebne za rad ili da prosledi rezultate svog rada
- Potrebno je njihovu vrednost definisati (saopštiti potprogramu) svaki put kada se poziva potprogram
- Postoje potprogrami koje nemaju parametre
- Svaki parametar deklariše lokalnu promenljivu koja je vidljiva samo u telu potprograma
- Vrednost ove promenljive postavlja se na vrednost koja je saopštena prilikom poziva potprograma
- Prestaje da važi neposredno nakon izlaska iz potprograma

... Parametri potprograma ...

- Parametar potprograma nije isto što i promenljiva čija vrednost se prosleđuje potprogramu
- Prilikom poziva funkcije, **stvarni parametri** moraju se poklapati sa **formalnim parametrima**:
 - u broju parametara,
 - tipu podatka parametra, i
 - moraju biti u odgovarajućem redosledu
- **Nazivi formalnih i stvarnih parametara ne moraju se poklapati!**

... Parametri potprograma

Deklaracija potprograma

```
static int funk1(int par1, float par2)
{
    // telo
    // potprograma
}
```

Pozivanje potprograma

```
// kod iz kojeg se poziva
// potprogram
int arg1 = 21;
float arg2 = 22.23;
int rez;
rez = funk1(arg1, arg2);
```

- **Formalni parametri** se navode u deklaraciji potprograma i nemaju konkretne vrednosti
- **Stvarni parametri** se navode u pozivu potprograma i donose potprogramu konkretne vrednosti (nazivaju se još i argumenti potprograma)

Potprogrami – primer 1

- **ZADATAK1:**

Implementirati program koji računa sumu, razliku, proizvod i količnik dva broja. Brojeve zadaje korisnik preko tastature. Implementirati unos, sabiranje, oduzimanje, množenje, deljenje i prikaz rezultata kao zasebne potprograme.

- **VEŽBA1:**

Izmeniti prethodni program tako da korisnik može da bira željenu operaciju (sabiranje, oduzimanje, množenje ili deljenje), pri čemu za svaku operaciju ponovo definiše operande. Omogućiti ponavljanje ovih operacija sve dok korisnik ne odluči da izađe iz programa.

Potprogrami – zadatak1 ...

```
/* **** */
* Primer programa koji računa sumu, razliku, proizvod i količnik dva broja. Brojeve zadaje korisnik preko tastature.
* Unos, sabiranje, oduzimanje, množenje, deljenje i prikaz rezultata implementirane su kao zasebne funkcije.
/* **** */
public class PotprogramaKalkulatorPrimer {
    public static void main(String[] args) {
        double operand1 = 0, operand2 = 0;
        double zbir = 0, razlika = 0, proizvod = 0, kolicnik = 0;
        System.out.println("--Program za simuliranje kalkulatora--");
        operand1 = unosOperanda("Unesite prvi operand:");
        operand2 = unosOperanda("Unesite drugi operand:");
        System.out.println("-----");
        zbir = sabiranje(operand1, operand2);
        razlika = oduzimanje(operand1, operand2);
        proizvod = mnozenje(operand1, operand2);
        kolicnik = deljenje(operand1, operand2);
        prikaziRezultate(operand1, operand2, zbir, razlika, proizvod, kolicnik);
    }
    static double unosOperanda(String poruka){
        double operand = 0.0;
        System.out.println(poruka);
        operand = TextIO.getLnDouble();
        return operand;
    }
}
```

... Potprogrami – zadatak1

```
static double sabiranje(double op1, double op2){
    return op1 + op2;
}
static double oduzimanje(double op1, double op2){
    return op1 - op2;
}
static double mnozenje(double op1, double op2){
    return op1 * op2;
}
static double deljenje(double op1, double op2){
    return op1 / op2;
}
static void prikaziRezultate(double op1, double op2, double zb, double rz, double pr,
                             double kl){
    System.out.println("\n---Rezultati osnovnih operacija su:---");
    System.out.printf("\n%8.2f + %8.2f = %8.2f\n", op1, op2, zb);
    System.out.printf("\n%8.2f - %8.2f = %8.2f\n", op1, op2, rz);
    System.out.printf("\n%8.2f * %8.2f = %8.2f\n", op1, op2, pr);
    System.out.printf("\n%8.2f / %8.2f = %8.2f\n\n", op1, op2, kl);
    System.out.println("-----");
}
}
```

- Šta sa deljenjem?

Prenos parametara potprograma ...

- **Java dozvoljava prenos parametara samo po vrednosti**
- Prilikom poziva potprograma, vrednost promenljive primitivnog tipa kopira se u parametar potprograma (koliko god da ta promenljiva zauzima memorije), te potprogram nema pristup promenljivoj
- Zbog toga ako potprogram i promeni vrednost jednom od parametara, to ne utiče na promenljivu čija vrednost je prosleđena potprogramu

... Prenos parametara potprograma ...

- Primer kako se vrednost promenljive ne menja:

```
public class PotprogramiPrimerPrenosParam1 {  
    public static void main(String[] args) {  
        int broj = 5;  
        povecajBroj(broj);  
        System.out.println("Vrednost broja nakon povratka u main: " + broj);  
    }  
    static void povecajBroj(int brojZaUvecavanje){  
        System.out.println("Vrednost broja koji stize u potprogram: " + brojZaUvecavanje);  
        brojZaUvecavanje++;  
        System.out.println("Vrednost broja nakon uvecanja u potprogramu: " + brojZaUvecavanje);  
    }  
}
```

Vrednost broja koji stize u potprogram: 5
Vrednost broja nakon uvecanja u potprogramu: 6
Vrednost broja nakon povratka u main: 5

... Prenos parametara potprograma ...

- Ovo ne važi za klase/objekte jer se one prenose po referenci, tako da se njihove vrednosti mogu menjati unutar potprograma i to je vidljivo i van potprograma
- Ovo je značajno kada se koriste nizovi

... Prenos parametara potprograma

- Primer kako se vrednost promenljive menja (niz):

```
public class PotprogramiPrimerPrenosParam2 {  
    public static void main(String[] args) {  
        int[] intNiz = {1, 2, 3};  
        promeniNiz(intNiz);  
        System.out.println("Vrednost niza nakon povratka u main: ");  
        System.out.print(" [" + intNiz[0] + "," + intNiz[1] + "," + intNiz[2] + "]");  
        System.out.println();  
    }  
    static void promeniNiz(int[] intNizPar){  
        int i = 0;  
        System.out.println("Vrednost niza koji stize u potprogram: ");  
        System.out.print(" [" + intNizPar[0] + "," + intNizPar[1] + "," + intNizPar[2] + "]");  
        System.out.println();  
        for (i = 0; i < 3; i++)  
            intNizPar[i]++;  
        System.out.println("Vrednost niza nakon uvecanja u potprogram: ");  
        System.out.print(" [" + intNizPar[0] + "," + intNizPar[1] + "," + intNizPar[2] + "]");  
        System.out.println();  
    }  
}
```

Vrednost niza koji stize u potprogram: [1,2,3]

Vrednost niza nakon uvecanja u potprogramu: [2,3,4]

Vrednost niza nakon povratka u main: [2,3,4]

Potprogrami – primer 2

- **ZADATAK2:**

Implementirati program za računanje sume vrednosti elemenata niza prirodnih brojeva koji sadrži maksimum 50 elemenata. Program prihvata od korisnika broj elemenata ($0 < N \leq 50$) i vrednost svakog pojedinačnog elementa. Unos elemenata i računanje sume realizovati kao zasebne potprograme.

- **VEŽBA2:**

Implementirati program koji od korisnika prihvata N elemenata niza (koji maksimalno može sadržati do 30 elemenata). Omogućiti korisniku da bira neku od sledećih operacija: izračunavanje sume elemenata niza, računanje srednje vrednosti niza, nalaženje minimuma, nalaženje maksimuma. Omogućiti ponavljeno izvršavanje ovih akcija.

Potprogrami – zadatak2 ...

```
/*
 * Primer programa za računanje sume vrednosti elemenata niza prirodnih brojeva koji sadrži maksimum 50 elemenata.
 * Program prihvata od korisnika broj elemenata (0 < N ≤ 50) i vrednost svakog pojedinačnog elementa.
 * Unos elemenata i računanje sume su realizovani kao zasebne potprogrami.
 */
public class PotprogramiPrimerNiz {
    static final int MAXELNIZA = 50;
    public static void main(String[] args) {
        int N = 1;
        int[] nizPrirodnihBrojeva = new int[MAXELNIZA];
        System.out.println("Program za racunanje sume elemenata niza N prirodnih brojeva.");
        N = unosBrojaElemNiza();
        unosElemNiza(N, nizPrirodnihBrojeva);
        System.out.println("Suma elemenata niza iznosi " + sumaVrednostNiza(N,
            nizPrirodnihBrojeva));
    }
    static int unosBrojaElemNiza(){
        int brojEl = 0;
        System.out.println("Unesite N:");
        brojEl = TextIO.getLnInt();
        return brojEl;
    }
}
```

... Potprogrami – zadatak2

```
static void unosElemNiza(int n, int[] nizElem){
    int i = 0;
    for (i = 0; i < n; i++){
        System.out.print("Unesite " + (i+1) + ". element niza: ");
        nizElem[i] = TextIO.getlnInt();
    }
    System.out.println();
}

static int sumaVrednostNiza(int n, int[] nizElem){
    int suma = 0;
    int i = 0;
    for (i = 0; i < n; i++)
        suma += nizElem[i];
    return suma;
}
}
```

Zadaci za vežbu

- **VEŽBA3:**

Napraviti program koji prihvata podatke o polaznicima (ime, prezime, jmbg, grad), dobijene podatke sortira po jmbg-u i prikazuje ih korisniku. Podaci se smeštaju u niz, može biti maksimalno 40 polaznika. Implementirati unos, prikaz i sortiranje kao zasebne potprograme.

- **VEŽBA4:**

Modifikovati prethodnu vežbu tako da korisnik može da bira kriterijum po kojem će se vršiti sortiranje. Implementirati program oslanjajući se na funkcije.

Vežba 3

```
/*
 * Primer programa koji prihvata podatke o polaznicima (ime, prezime, jmbg, grad), dobijene podatke sortira po
 * jmbg-u i prikazuje ih korisniku. Podaci se smeštaju u niz, može biti maksimalno 40 polaznika.
 * Implementirati unos, prikaz i sortiranje kao zasebne potprograme.
 */
public class Polaznici {
    static int MAXPOLAZNIKA = 25;
    public static void main(String[] args) {
        String[][] listaPolaznika = new String[MAXPOLAZNIKA][4];
        int brPolaznika = 0;
        System.out.println("-- Program za unos i sortiranje liste polaznika --");
        brPolaznika = unosBrojaPolaznika();
        unosPolaznika(listaPolaznika, brPolaznika);
        sortiranjeJmbg(listaPolaznika, brPolaznika);
        prikazPolaznika(listaPolaznika, brPolaznika, "JMBG");
    }
    static int unosBrojaPolaznika(){
        int brojEl = 0;
        System.out.println("Unesite broj polaznika (maksimalno 40):");
        brojEl = TextIO.getLnInt();
        return brojEl;
    }
}
```

Vežba 3

```
static void unosPolaznika(String[][] polaznici, int brPolaznika){
    int i = 0;
    System.out.println("-----");
    for (i = 0; i < brPolaznika; i++){
        System.out.println("Unesite podatke za " + (i+1) + ". polaznika:");
        System.out.print("Ime:\t\t");
        polaznici[i][0] = TextIO.getLnWord();
        System.out.print("Prezime:\t");
        polaznici[i][1] = TextIO.getLnWord();
        System.out.print("JMBG:\t\t");
        polaznici[i][2] = TextIO.getLnWord();
        System.out.print("Grad:\t\t");
        polaznici[i][3] = TextIO.getLnWord();
        System.out.println("-----");
    }
}

static void sortiranjeJmbg(String[][] polaznici, int brPolaznika){
    int i, j;
    for(i = 0; i < brPolaznika-1; i++)
        for(j = i+1; j < brPolaznika; j++)
            if (polaznici[i][2].compareTo(polaznici[j][2]) > 0)
                zamena(polaznici, i, j);
}
```


Vežba 3

```
static void zamena(String[][] polaznici, int koga, int kim){
    int i = 0;
    String podatakOPolazniku;
    for(i = 0; i < 4; i++){
        podatakOPolazniku = polaznici[koga][i];
        polaznici[koga][i] = polaznici[kim][i];
        polaznici[kim][i] = podatakOPolazniku;
    }
}

static void prikazPolaznika(String[][] polaznici, int brPolaznika, String vrstaSorta){
    int i;
    System.out.println("Lista polaznika sortirana prema " + vrstaSorta + ":");
    System.out.println("-----");
    for (i = 0; i < brPolaznika; i++){
        System.out.println("R.br:\t\t_ " + (i+1) + " _");
        System.out.println("Ime:\t\t" + polaznici[i][0]);
        System.out.println("Prezime:\t" + polaznici[i][1]);
        System.out.println("JMBG:\t\t" + polaznici[i][2]);
        System.out.println("Grad:\t\t" + polaznici[i][3]);
        System.out.println("-----");
    }
}
}
```

Kompleksni zadaci za vežbanje ...

1. Program za praćenje poslovanja agenta za osiguranje

- Evidencija prodaje osiguranja organizovana je u 5 kategorija i beleži se za svaki dan u mesecu (pretpostaviti da ih je 30).
- Program treba da omogući:
 - Unos ostvarenog prihoda za zadatu kategoriju osiguranja i dan u mesecu.
 - Izmenu ostvarenog prihoda za zadatu kategoriju osiguranja i dan u mesecu.
 - Prikaz rezultata sortiran po kategorijama osiguranja za izabrani dan u mesecu.
 - Prikaz rezultata sortiran po danima u mesecu za izabranu kategoriju.
 - Izračunavanje i prikaz dana kad je ostvaren najmanji prihod i koliko on iznosi.
 - Izračunavanje i prikaz ukupnog mesečnog prihod za svaku kategoriju osiguranja.
 - Izračunavanje i prikaz prosečnog mesečnog prihoda.
 - Meni iz kojeg se biraju ponuđene funkcije.
 - **Obavezno koristiti funkcije za implementaciju traženih funkcionalnosti.**
- Izabrati korektne tipove podataka i strukture podataka i ograničiti unos tako da nije moguće uneti neispravne vrednosti.

... Kompleksni zadaci za vežbanje ...

2. Program za praćenje rada kelnera

- Kelner prodaje 10 vrsta pića. Svakog sata radnog vremena (8 sati) beleži se kelnerov prihod po vrsti pića.
- Program treba da omogućiti:
 - Unos ostvarenog prihoda za zadati sat i vrstu pića.
 - Izmenu ostvarenog prihoda za zadati sat i vrstu pića.
 - Prikaz rezultata sortiran po satima za izabranu vrstu pića.
 - Prikaz rezultata sortiran po vrstama pića za izabrani sat.
 - Izračunavanje i prikaz najprodavanije vrste pića i prihod ostvaren njenom prodajom.
 - Izračunavanje i prikaz ukupnog dnevnog prihoda.
 - Izračunavanje i prikaz prosečnog prihoda za svaki sat.
 - Meni iz kojeg se biraju ponuđene funkcije.
 - **Obavezno koristiti funkcije za implementaciju traženih funkcionalnosti.**
- Izabrati korektne tipove podataka i strukture podataka i ograničiti unos tako da nije moguće uneti neispravne vrednosti.

... Kompleksni zadaci za vežbanje

3. Program za praćenje godišnjeg poslovanja banke

- Banka posluje u gradu putem 10 poslovnica i za svaki mesec beleži se njen prihod.
- Program treba da omogućiti:
 - Unos ostvarenog prihoda za zadatu poslovnicu i mesec.
 - Izmenu ostvarenog prihoda za zadatu poslovnicu i mesec.
 - Prikaz rezultata sortiran po mesecima za izabranu poslovnicu.
 - Prikaz rezultata sortiran po poslovnicama za izabrani mesec.
 - Izračunavanje i prikaz meseca u kojem je ostvaren najveći prihod i koliko on iznosi.
 - Izračunavanje i prikaz ukupnog godišnjeg prihoda.
 - Izračunavanje i prikaz prosečnog prihoda za svaku poslovnicu.
 - Meni iz kojeg se biraju ponuđene funkcije.
 - **Obavezno koristiti funkcije za implementaciju traženih funkcionalnosti.**
- Izabrati korektne tipove podataka i strukture podataka i ograničiti unos tako da nije moguće uneti neispravne vrednosti.