

# Osnove programiranja

## Osnovni pojmovi i elementi

-Pristupno predavanje-

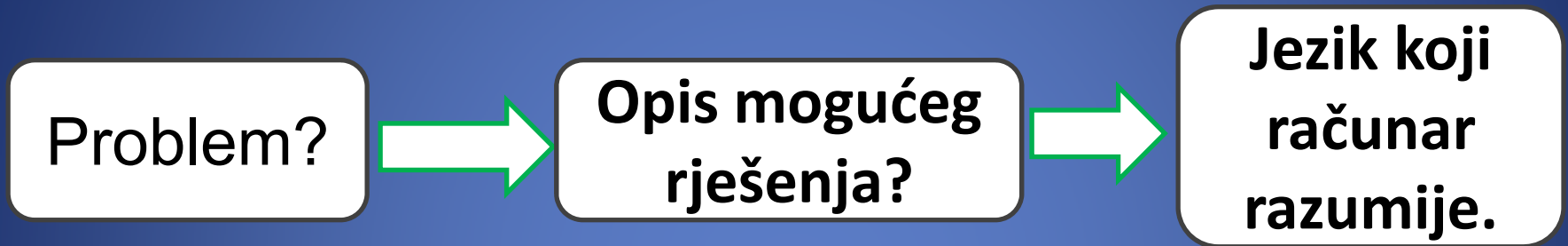
Dr. sc. Denis Čeke, dipl. inž. el.

Zenica, 03.09.2018. godine

# Sadržaj izlaganja

- Programiranje i faze u programiranju.
- Tipovi podataka, osnovni aritmetički operatori, relacioni operatori i operator dodjele.
- Struktura računarskog programa.
- Varijable (promjenjive).
- Konstante (nepromijenjive).
- Naredbe ulaza i izlaza, i naredbe grananja, if/switch.
- Programske petlje, for/while/do-while.
- Polja podataka ili nizovi.
- Funkcije.
- Komentari.

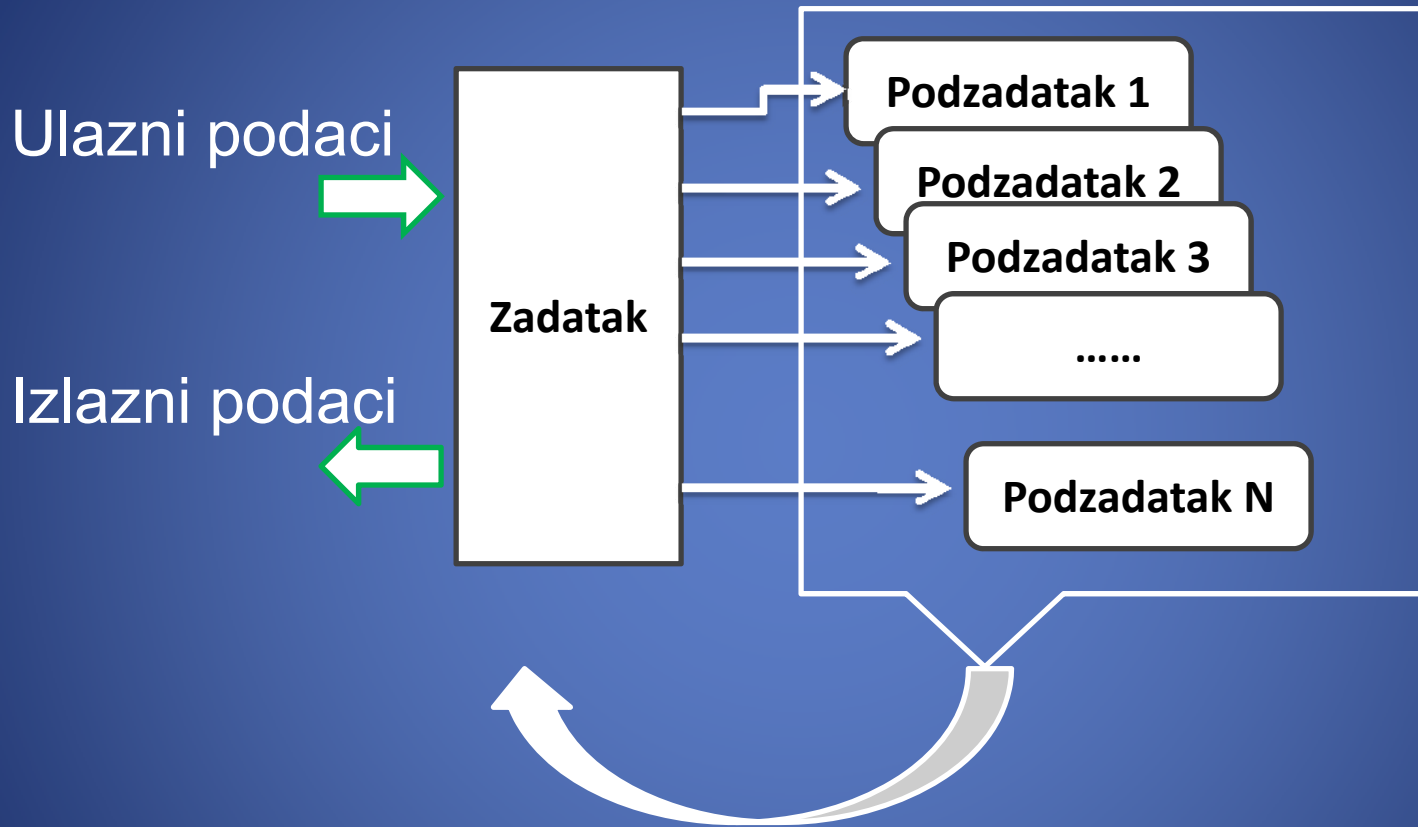
# Šta je to programiranje?



Steve Jobs (1955-2011) je jednom rekao: *"I think everyone in this country should learn to program a computer, because it teaches you how to think."*

- Potrebno je „programersko razmišljanje“.

# Šta znači „znati programirati“?



- Kako, pomoću čega?
- Kroz praksu, vježbu, od jednostavnijih ka složenijim primjerima?

# Program i programski jezik

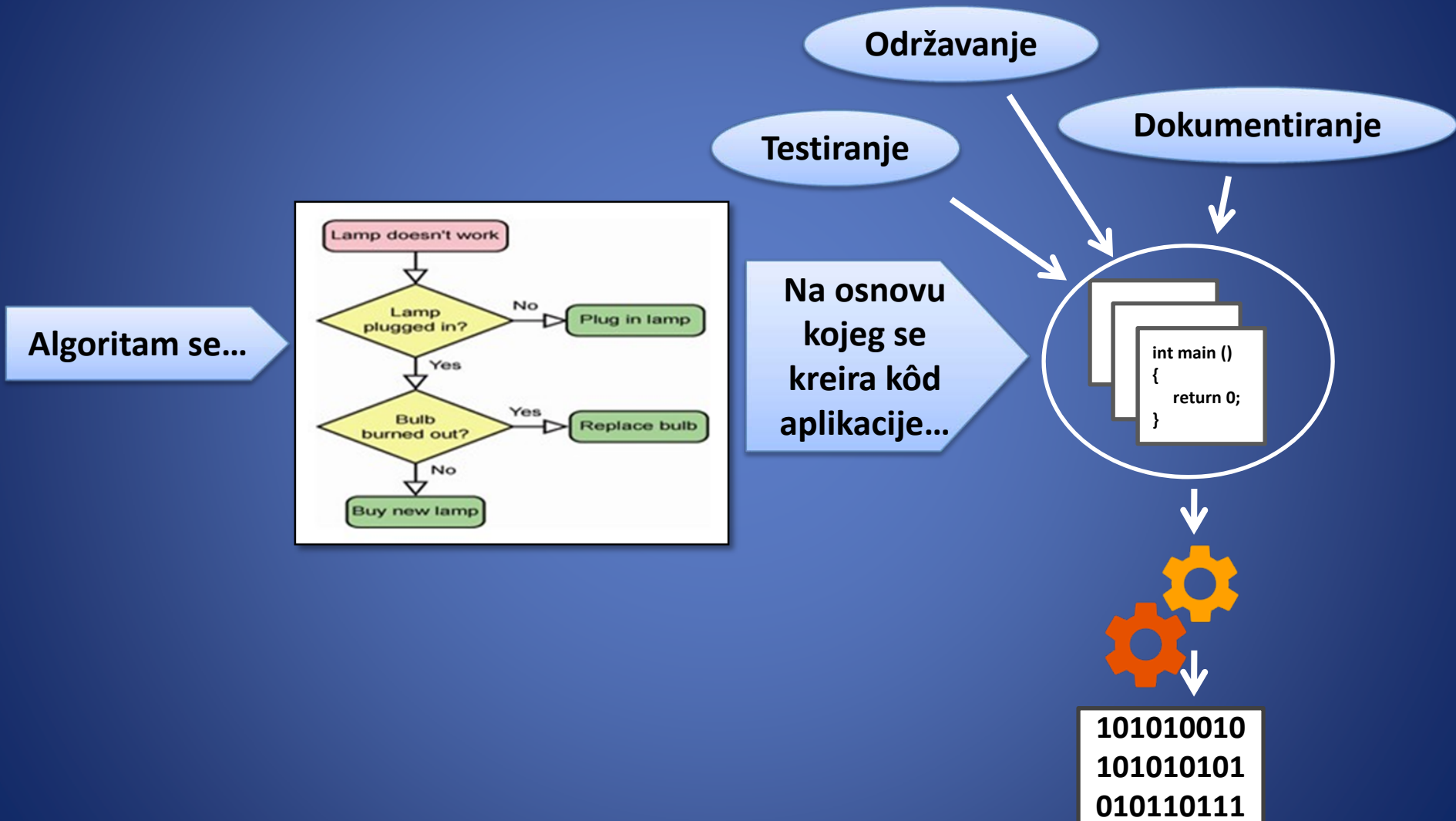


```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {int x=5;
5 cout << "Value of x is " << x << endl;
6 return 0;
7 }
8
```



Instrukcije koje računar razumije

# Pregled faza u programiranju

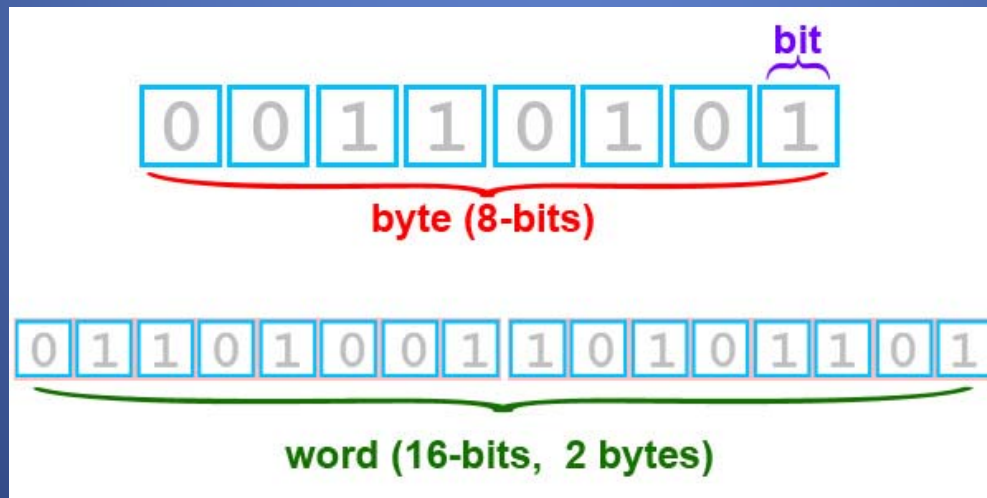


# Koji programski jezik koristiti, razvojna okruženja?

- Vječno pitanje!
- Zavisí od potreba.
- Programski jezik C, jezik visokog nivoa.
- Ima široku primjenu:
  - Igre, komercijalni sistemi, vještačka inteligencija.
  - Sistemi u stvarnom vremenu “Real-Time Systems”.
  - Operativni sistem, prevodioci odnosno kompajleri.
- Razvojna okruženja: za razvoj C/C++ može se koristiti besplatno razvojno okruženje Dev C++.

# Tipovi podataka

- Svaki tip podataka ima svoju veličinu koja je izražena brojem bita koje taj podataka zauzima u memoriji računara.





# Tipovi podataka

- Tri su osnovna tipa podataka:
  1. Cjelobrojni tip:
    - int (16 ili 32 bita).
  2. Realni tip:
    - float i double (32, odnosno 64 bita).
  3. Znakovni tip:
    - char (8 bita, odnosno 1 bajt).

# Tipovi podataka

- Varijante: *short*, *long*, *signed* i *unsigned*.
- *short* - memorijski prostor za predstavljanje podataka se smanjuje na pola.
- *long* - memorijski prostor za predstavljanje podataka se udvostručuje.
- *unsigned* - radi se o neoznačenom podatku, nema predznaka.
- *signed* - radi se označenom podatku, ima predznak.

# Osnovni aritmetički operatori

- Operatori osnovnih aritmetičkih operacija su prikazani u sljedećoj tabeli:

Operacija	Operator
Zbrajanje	+
Oduzimanje	-
Množenje	*
Dijeljenje	/
Ostatak cjelobrojnog dijeljenja. Na primjer $7\%2=1$ , jer je 7 podijeljeno sa 2 jednako 3, a ostatak je jedan.	%

# Poredbeni operatori (relacijski)

- Koriste za uspoređivanje dvije vrijednosti.
- Pretežno u slučajevima kontrole toka programa, petlje i grananja.

Operator	Značenje
<	Manje
<=	Manje ili jednako
>	Veće
>=	Veće ili jednako
==	Jednako
!=	Različito

# Operator dodjele

- Operator proste dodjele ima sljedeći oblik: =
- Lijevi operand je lijeva vrijednost (ime promjenjive) a desni operand je proizvoljan izraz.

**Primjer: X = 5;**

- **Važno pravilo:** Uvijek se prvo izračunava izraz na desnoj strani operatora i dobivena vrijednost se zapisuje na memorijsku lokaciju rezerviranu za čuvanje te promjenjive!
- Nije isto što i relacioni operator: ==

# Struktura računarskog programa

- Jedan C/C++ program se sastoji od:
  - varijabli (promjenjivih),
  - konstanti (nepromjenjivih),
  - tipova podataka i
  - funkcija.
- Svaki C/C++ program sadrži tačno i samo jednu *main* funkciju i nju poziva operativni sistem računara u trenutku poziva programa.
- *main* funkcija je tzv. “ulazna tačka” programa.

# Varijable (promjenjive)

- Deklaracija varijable ima sljedeći oblik:

**tip\_podatka** *ime\_varijable*;

- Primjer:

<b>tip_podatka</b>	<b>ime_varijable</b>
<b>int</b>	x, y, z
<b>long int</b>	a
<b>double</b>	povrsina_kvadra

# Konstante (nepromjenjive)

- Deklaracija konstante ima sljedeći oblik:

**const** tip\_konstante *izraz*;

- Primjer:

Ključna riječ	tip_konstante	<i>izraz</i>
<b>const</b>	<b>int</b>	x=1, y=2, z=3
<b>const</b>	<b>float</b>	Pi=3.14

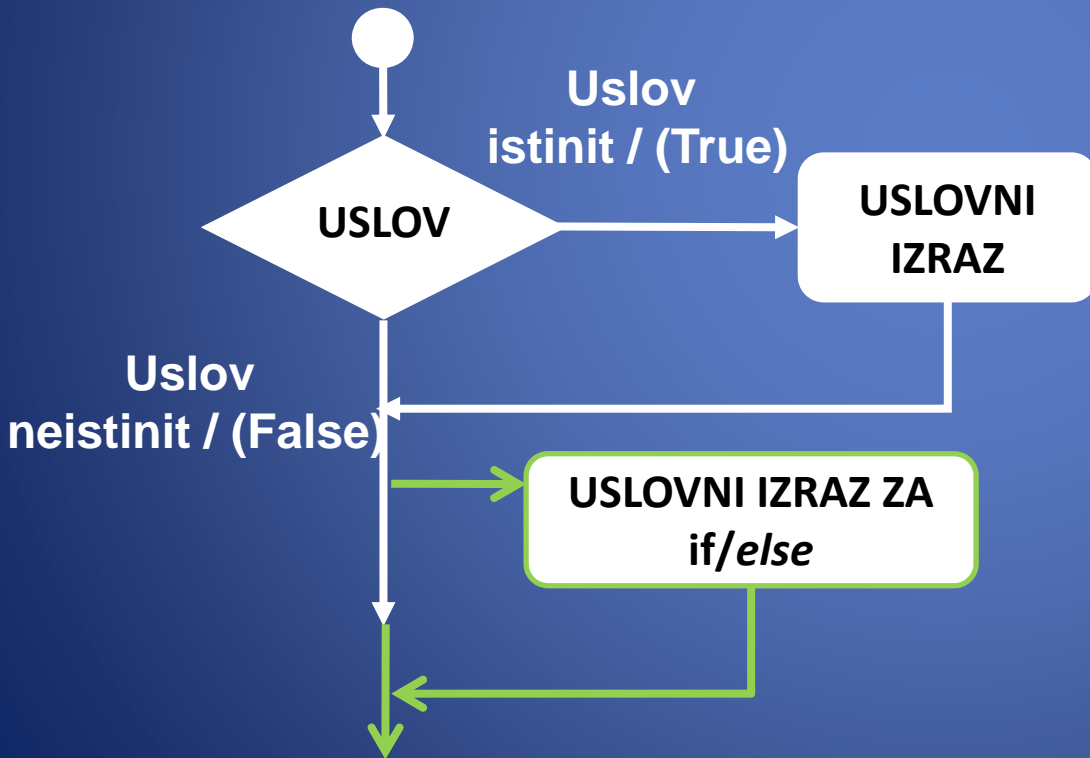


# Naredbe ulaza i izlaza

- C programski jezik: *scanf* i *printf*.
- C++ programski jezik: *cin* i *cout*.
- Biblioteke *stdio.h* i *iostream.h* sadrže funkcije za upravljanje ulazno/izlaznim resursima.
- Uključivanje ovih biblioteka u kod aplikacije se vrši pretprocesorskom direktivom: *include*.
- Primjer: *#include <naziv\_fajla>*
- C programski jezik: *#include <stdio.h>*
- C++ programski jezik: *#include <iostream.h>*

# Naredbe grananja - *if*

- Osnovni tip grananja implementira se naredbom *if*.
- Prikaz dijagrama toka:



## Ekvivalent zapisan u kôdu

```
if(<logički_izraz>) {  
    <blok_naredbi 1 ...>  
}
```

## Primjer za if/else

```
if(x==5) {  
    y=y+x;  
} else {y=0;}
```

# Naredbe grananja - *switch*

- Ako uslov grananja može dati više različitih rezultata preglednije je koristiti *switch* grananje.



## Ekvivalent zapisan u kodu

```
switch (cjelobrojni_izraz)
{
    case konstantan_izraz 1 :
        // kôd za Slučaj 1

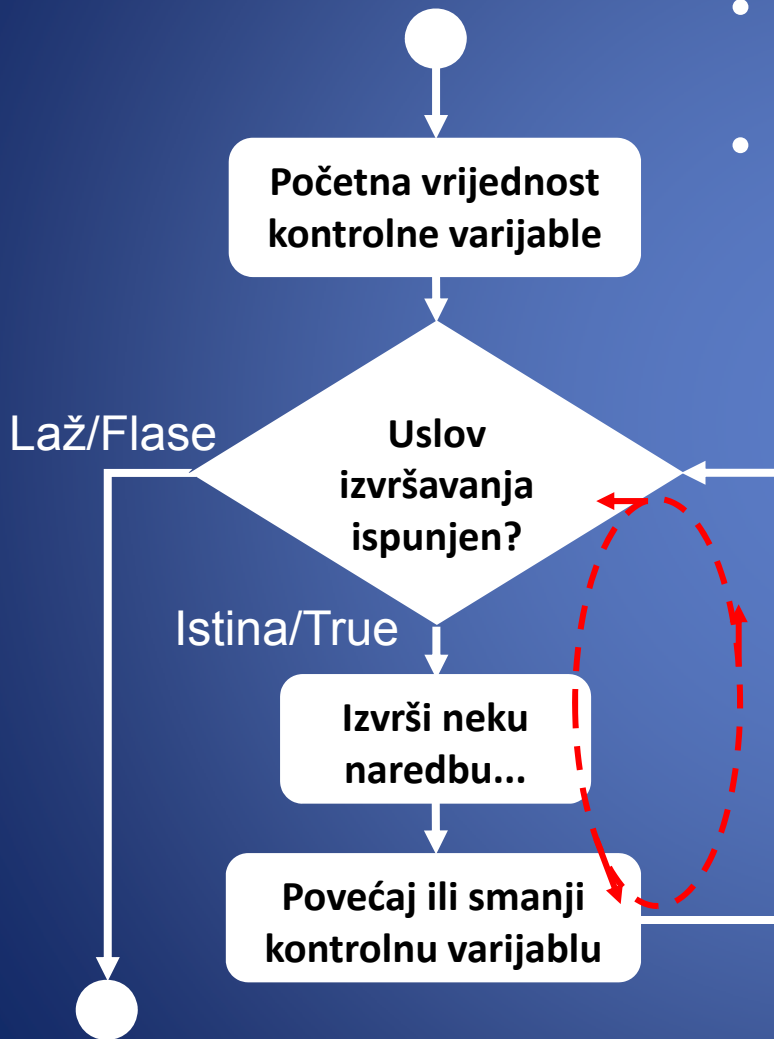
    case konstantan_izraz n :
        // kôd za Slučaj n
        break;

    default:
        // podrazumijevani kôd
}
```

# Programske petlje

- Koriste se za implementaciju višestrukog ponavljanja izvršavanja nekog dijela programskog koda.
- Tri osnovne vrste programskih petlji su:
  1. *for*
  2. *while*
  3. *do-while*

# Programska petlja *for*



- Posjeduje tri izraza za kontrolu toka izvršavanja.
- Početno stanje kontrolne varijable, uslov izvršavanja i povećanje ili smanjenje kontrolne varijable.

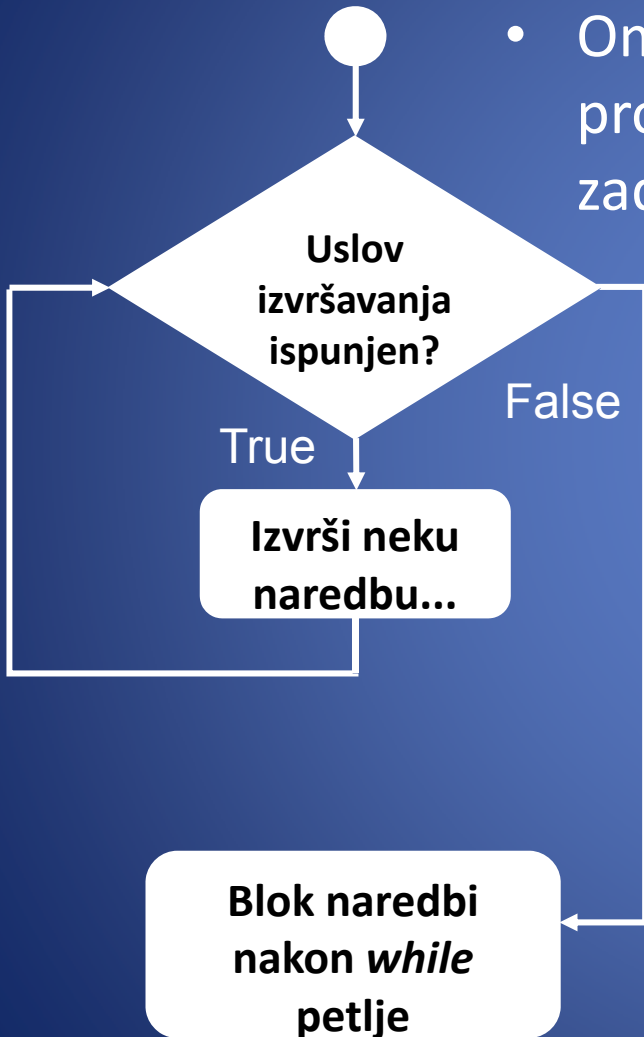
```
for(<izraz1>; <izraz2>; <izraz3>)  
{  
    <naredba ili ...>  
}
```

```
for(i=1; i<4; i++)  
{  
    x=i+5;  
}
```

$i=1, x=1+5=6;$ (1 iter.)
$i=2, x=2+5=7;$ (2 iter.)
$i=3, x=3+5=8;$ (3 iter.)

# Programska petlja *while*

- Omogućava višestruko ponavljanje programskog koda dok je definirani uslov zadovoljen (istina / *True*).

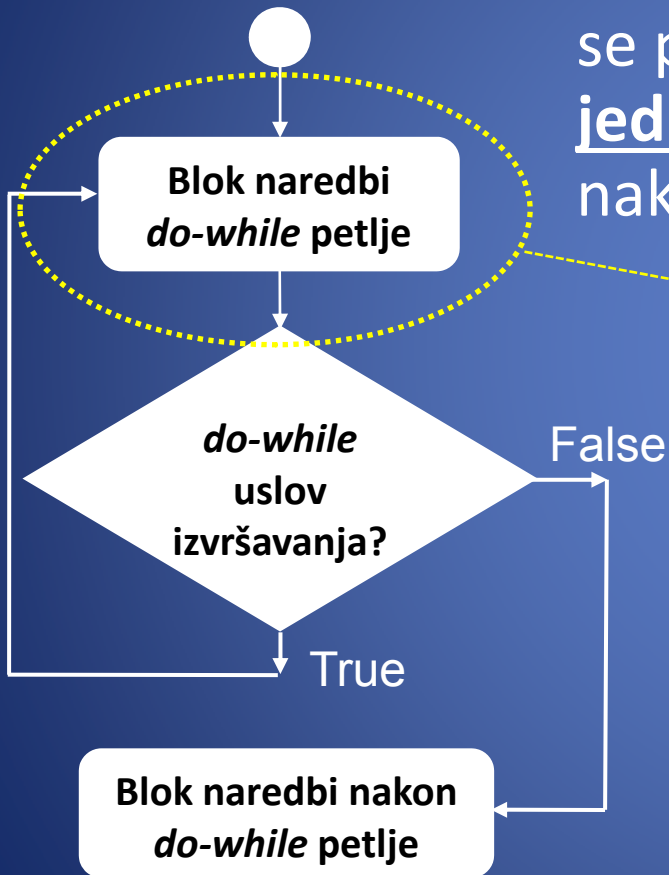


```
while (<izraz>)  
{  
    <naredba >  
}
```

```
while (i<10)  
{  
    i=i+5;  
}
```

# Programska petlja *do-while*

- Identična *while* petlji s tom razlikom da će se programski kod u petlji izvršiti sigurno jednom jer se tačnost izraza provjerava nakon jednog prolaska kroz petlju.



```
do {  
    <naredba...>  
} while (<izraz>);
```

```
do {  
    i=i+5;  
} while ( i>5);
```

# Polja podataka (*engl. arrays*) ili nizovi

- Konačni skupovi istih tipova podataka - članova polja.
- Pristup članovima polja ostvaruje se putem indeksa ili rednog broja kojeg svaki od njih zauzima u nizu.
- Prvi član u polju ima indeks 0, a zadnji član ima indeks za 1 manji od dužine polja.
- Osnovna podjela:
  - jednodimenzionalna i
  - dvodimenzionalna polja.



# Polja podataka - jednodimenzionalna

- Jednodimenzionalna polja - članovi su složeni u jednom kontinuiranom nizu i dohvaćaju se samo jednim indeksom.
- Primjer deklaracije jednodimenzionalnog polja:

`float x[5];`

X[0]:	X[1]:	X[2]:	X[3]:	X[4]:
float	float	float	float	float

Slika. Jednodimenzionalno polje ili niz.

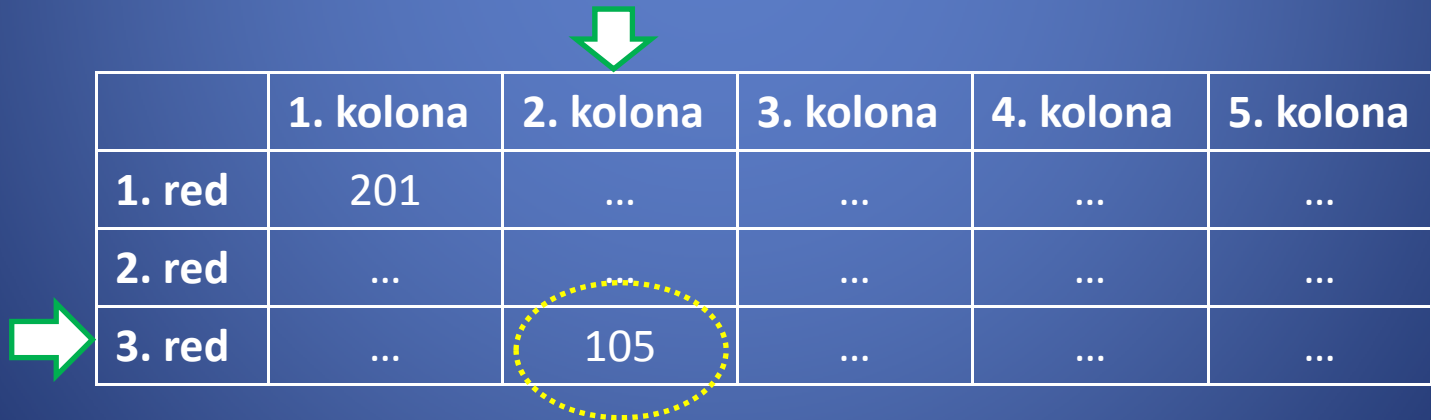
# Polja podataka - dvodimenzionalna

- Često postoji potreba za pohranjivanjem podataka u dvodimenzionalne ili višedimenzionalne strukture.
- Primjer: želimo pohraniti podatak u tablicu u 3 red i 5 kolonu - upotreba dvodimenzionalnih polja.
- Primjer deklaracije dvodimenzionalnog polja:

```
int Tabela [3][5];
```

# Polja podataka - dvodimenzionalna

- Pristup članovima polja se vrši preko dva indeksa:
  - prvi određuje red,
  - drugi određuje kolonu u tabelu u kojoj se podatak nalazi.
- Primjer: Tabela [2][1] = 105; // 3.red, 2. kolona.



A diagram illustrating a 3x5 data table. The table has three rows and five columns. The first row contains the values 201, ..., ..., ..., and ... in its five cells. The second row contains ..., ..., ..., ..., and ... in its five cells. The third row contains ..., 105, ..., ..., and ... in its five cells. A green arrow points to the third row, and a yellow dashed circle highlights the value 105 in the second column of the third row.

	1. kolona	2. kolona	3. kolona	4. kolona	5. kolona
1. red	201	...	...	...	...
2. red	...	...	...	...	...
3. red	...	105	...	...	...

Slika. Primjer tabele sa tri reda i pet kolona.

# Funkcije

- Izdvojeni nizovi naredbi koji čine logičke cjeline, a obavljaju tačno utvrđene zadatke.
- Moguće je napraviti vlastite funkcije, pa ih koristiti u programu, ili koristiti postojeće funkcije koje se nalaze u određenim bibliotekama funkcija.
- Primjer: *sprintf* i *printf* su funkcije koje su već napravljene i nalaze se u standardnoj *stdio.h* biblioteci.

# Funkcije

- Prednosti:
  - Jednom napravljena funkcija, može se koristiti željeni broj puta.
  - Rastavljanjem izvornog kôda u formu funkcija, doprinosi modularnosti programa.
  - Program se dijeli na manje samostalne cjeline koje je lakše koristiti (dekompozicija programa).
  - Kôd ujedno postaje čitljiviji i razumljiviji.
- Imaju ulazne parametre (argumente) (većinom).
- Vraćaju (ili ne) neku vrijednost nakon izvršavanja.

# Funkcije

- Opći oblik deklaracije funkcije:

*tip\_funkcije naziv\_funkcije (popis argumenata)*

- Primjer deklaracije funkcije:

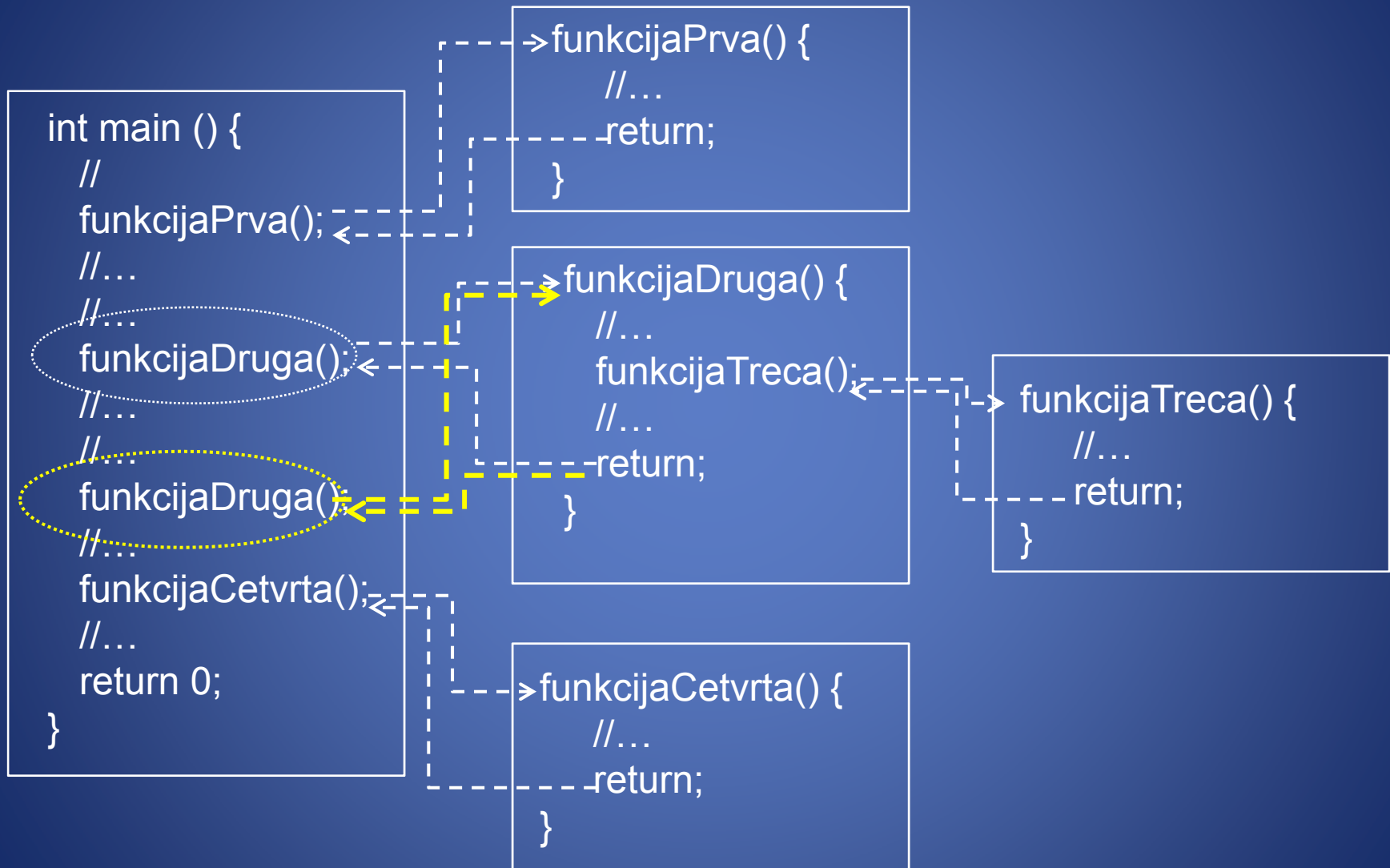
*float zapremina\_kvadra (float a, float b, float c)*

# Funkcije

- Primjer: funkcije koja vraća izračun zapreminu kvadra.

```
float zapremina_kvadra (float a,float b,float c)
{
    float v;
    v=a*b*c;
    return v;
}
```

# Primjer izvršavanja funkcija





# I na kraju... (ne manje važno)

## Upotreba komentara u izvornom kôdu

- Jako je bitno i korisno opisati šta se kojim od dijelova kôda želi postići, šta su argumenti, objasniti deklaraciju varijabli i slično.
- Komentar započinje s **dvostrukom kosom crtom //**, a završava **krajem reda**.
- Druga vrsta komentara počinje sa **/\*** i završava sa **\*/**. Ovakvi komentari mogu se protezati kroz više redova.
- Komentari u izvornom kôdu bi trebali biti primjer dobre prakse svakom programeru.

# Primjer upotrebe komentara u izvornom kôdu

```
int x, y, z;  
/* Komentar  
u više linija.  
Ovo je treća linija komentara. */
```

```
// komentar u jednoj liniji  
z=x-y; // komentar u jednoj liniji
```

- Pri prevođenju izvornog kôda komentar se **ne prevodi**.
- Komentari se osim za opis dijelova izvornog kôda katkad upotrebljavaju i za **privremeno isključivanje dijelova izvornog kôda**.

# O čemu smo danas govorili?

- Programi i programiranje.
- Faze programiranja, programski jezici.
- Sadržaji svakog programa: tipovi podataka, varijable, konstante, naredbe ulaza i izlaza iz programa.
- Naredbe grananja i programske petlje.
- Polja ili nizovi podataka.
- Funkcije u programu i upotreba komentara.

# Literatura

- *Demistificirani C++*, Julijan Šribar i Boris Motik, "Element" Zagreb, 2014.
- *Programiranje 1, Osnove programiranja kroz programski jezik C*, Filip Marić i Predrag Janičić, Matematički fakultet Univerziteta u Beogradu, 2015.
- *C++ How to program*, 7th edition, Deitel & Deitel, Prentice Hall, 2009.
- Razvojno okruženje Dev C++,  
<https://sourceforge.net/projects/orwelldevcpp/>.
- Razvojno okruženje Microsoft Visual Studio Code:  
<https://code.visualstudio.com/docs/languages/cpp>.
- *Osnove programiranja u jeziku C++*, Sanda Šutalo i Darko Grundler, <https://sites.google.com/site/sandasutalo/>.

“Vježba čini majstora”

Hvala na pažnji!

# Pitanja?

- Dr. sc. Denis Čeke, dipl.inž.el.
- E-mail: [ceke.denis@gmail.com](mailto:ceke.denis@gmail.com)
- Web: <http://www.denisceke.com>
- **LinkedIn:**  
<https://www.linkedin.com/in/denis-ceke-b3b30989?trk=hp-identity-name>
- **ResearchGate:**  
[https://www.researchgate.net/profile/Denis\\_Ceke](https://www.researchgate.net/profile/Denis_Ceke)
- **Google Scholar:**  
<https://scholar.google.com/citations?user=HrWIThYAAAAJ&hl=en>