SVEUČILIŠTE U ZAGREBU
Fakultet prometnih znanosti
Zavod za inteligentne transportne sustave
Vukelićeva 4, Zagreb, HRVATSKA

Računalstvo

Petlje

Doc. dr. sc. Edouard Ivanjko, dipl.ing.

Sadržaj

- Uvod
- Petlja while
- Petlja do - while
- Petlja for
- Primjeri

Uvod

- Računala se često koriste za automatizaciju zadataka koji se ponavljaju
  - Isti zadatak se izvrši više puta s različitim podacima
- Takav tijek izvođenja programa naziva se **petlja**
  - Jedan prolaz kroz petlju se naziva **iteracija**
  - Tijelo petlje je programski odsječak koji se izvrši u svakoj iteraciji

Uvod

- Vrste petlji
  - Petlje s ispitivanjem uvjeta na početku
    - Moguće da se tijelo petlje ne izvrši nijednom
    - Petlja **while**
  - Petlje s ispitivanjem uvjeta na kraju
    - Tijelo petlje se izvrši obavezno jednom
    - Petlja **do - while**
  - Petlje s poznatim brojem ponavljanja
    - Broj ponavljanja se može unaprijed odrediti
    - Broj ponavljanja najčešće ne ovisi o tijelu petlje
      - Po potrebi moguće je napraviti ovisnost
    - Petlja **for**

Uvod

- Izvršavanje petlje ovisi o logičkom uvjetu
  - Uvjet ispunjen (**istina**) tijelo petlje se izvršava
  - Uvjet nije ispunjen (**laž**) petlja završava
- Pravila kreiranja logičkog uvjeta analogna pravilima kod struktura grananja
  - Kombinacija aritmetičkog i logičkog dijela
- Završetkom petlje se izvršava prva naredba ili linija kôda nakon programskog bloka petlje
  - Program se dalje izvršava slijedno

Petlja while

- Pseudokôd

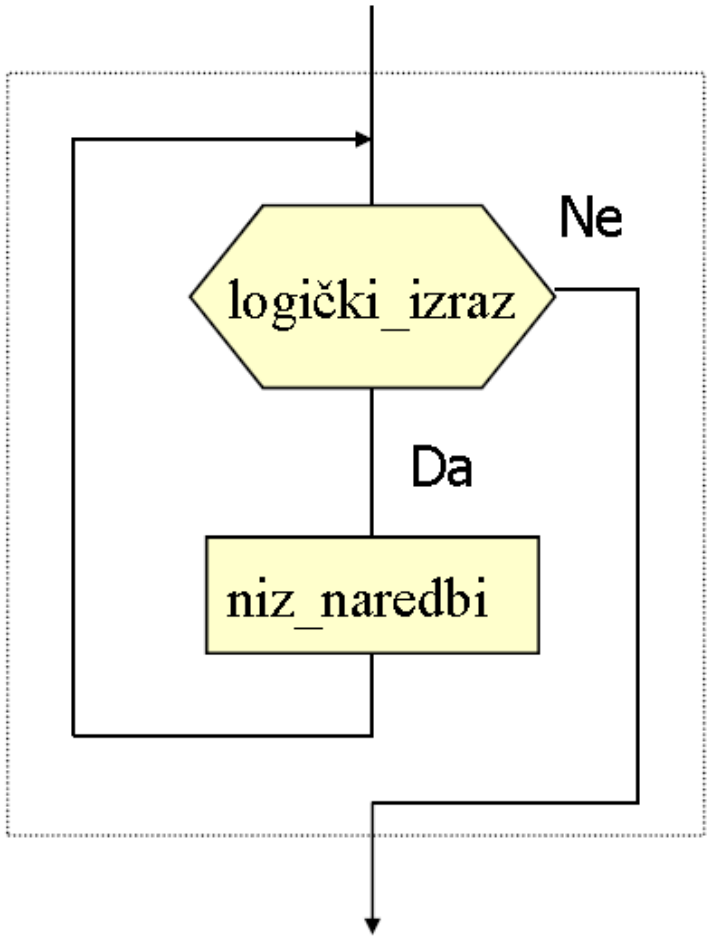
```
dok je ( logički_izraz )
    niz_naredbi
```

- C#

```
while ( logički_izraz )
    naredba;
```

  – ili

```
while ( logički_izraz )
{
    niz_naredbi;
}
```



Petlja while

- Formalno napisano, tijek petlje **while** je slijedeći
  - Izračunava se uvjet u zagradi (rezultat je uvijek ili istina ili laž)
  - Ako je uvjet lažan program izlazi iz petlje while i nastavlja izvršavanje iza nje
  - Ako je uvjet istinit, redom se izvršavaju naredbe unutar vitičastih zagrada (tijelo petlje) nakon čega se ponavlja provjera uvjeta

Petlja while

- Ukoliko u prvoj iteraciji uvjet nema vrijednost **istina** (engl. **true**), naredbe unutar tijela petlje se nikada neće izvršiti
  - Izvršava se prva naredba nakon programskog bloka petlje
- Naredbe unutar tijela petlje trebaju promijeniti vrijednosti varijabli tako da u nekom trenutku uvjet postane vrijednost **laž** (engl. **false**) te da se petlja prekine

Petlja while

```
static void Main(string[] args)
{
    int n = 10;
    while (n > 0)
    {
        Console.WriteLine(n);
        n = n - 1;
    }
    Console.ReadKey();
}
```

Izračunava se **uvjet** u zagradi (rezultat je uvijek ili istina ili laž)

Ako je uvjet istinit, redom se izvršavaju naredbe unutar vitičastih zagrada nakon čega se ponavlja provjera **uvjeta**

Ako je **uvjet** laž, program izlazi iz petlje **`while`** i nastavlja izvršavanje iza nje

Petlja while - Primjer

- Napisati program koji ispisuje ostatke uzastopnog dijeljenja učitanog nenegativnog cijelog broja s 2

📂 **DijeljenjeBroja**

```
// deklaracija varijabli
int broj, ostatak;

// unos broja
Console.Write("Unesite nenegativan cijeli broj > ");
broj = Convert.ToInt32(Console.ReadLine());

// obrada broja
while (broj != 0)
{
    ostatak = broj % 2;
    Console.WriteLine("Ostatak cjelobrojnog dijeljenja je " + ostatak);
    broj = broj / 2;
}

// da ispis ostane vidljiv
System.Console.ReadKey();
```

Petlja do - while

- Pseudokôd

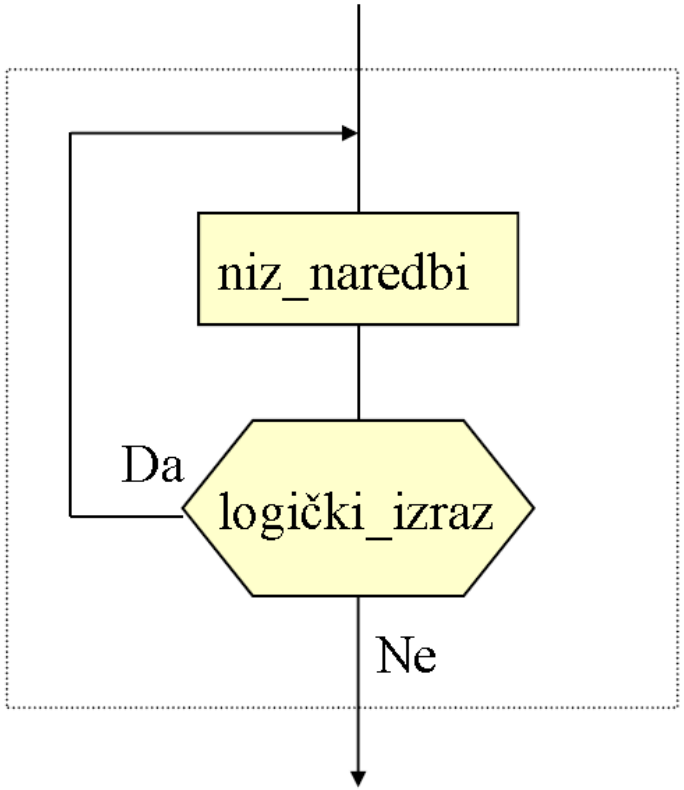
```
ponavljaj
    niz_naredbi
dok je ( logički_izraz )
```

- C#

```
do
    naredba;
while ( logički_izraz );
```

– ili

```
do
{
    niz_naredbi;
} while ( logički_izraz );
```



Petlja do - while

- Razlika prema while petlji je mjesto provjere uvjeta
  - Kod petlje **while** se provjera uvjeta radi na početku (prije tijela petlje)
    - Mogućnost da se tijelo petlje nikada ne izvrši
  - Kod petlje **do – while** se provjera uvjeta radi na kraju (nakon tijela petlje)
    - Tijelo petlje se izvrši najmanje jednom
    - Sljedeće izvršavanje samo ako je logički uvjet ispunjen (istina)

```
static void Main(string[] args)
{
    int brojac = 0;
    do
    {
        brojac++;
        Console.WriteLine(brojac);
    }
    while (brojac < 5) ;

    Console.ReadKey();
}
```

Petlja do - while - Primjer

- Napisati program koji će čitati pozitivne brojeve s tipkovnice dok se ne upiše broj nula i zatim ispisati najveći broj, zanemariti negativne brojeve

```
// deklaracija varijabli
int broj, najveci;

// inicijalizacija varijabli
najveci = 0;

// unos i obrada podataka
do
{
    Console.Write("Unesite broj (0 za kraj) > ");
    broj = Convert.ToInt32(Console.ReadLine());
    if (broj > 0 && broj > najveci)
        najveci = broj;
} while (broj != 0);

// ispis rezultata
if (najveci > 0)
    Console.WriteLine("\nNajveci ucitani broj je > " + najveci);
else
    Console.WriteLine("\nNije ucitan broj veci od 0!");

// da ispis ostane vidljiv
System.Console.ReadKey();
```

📂 **NajveciBrojDoWhile**

Petlja for

- Pseudokôd

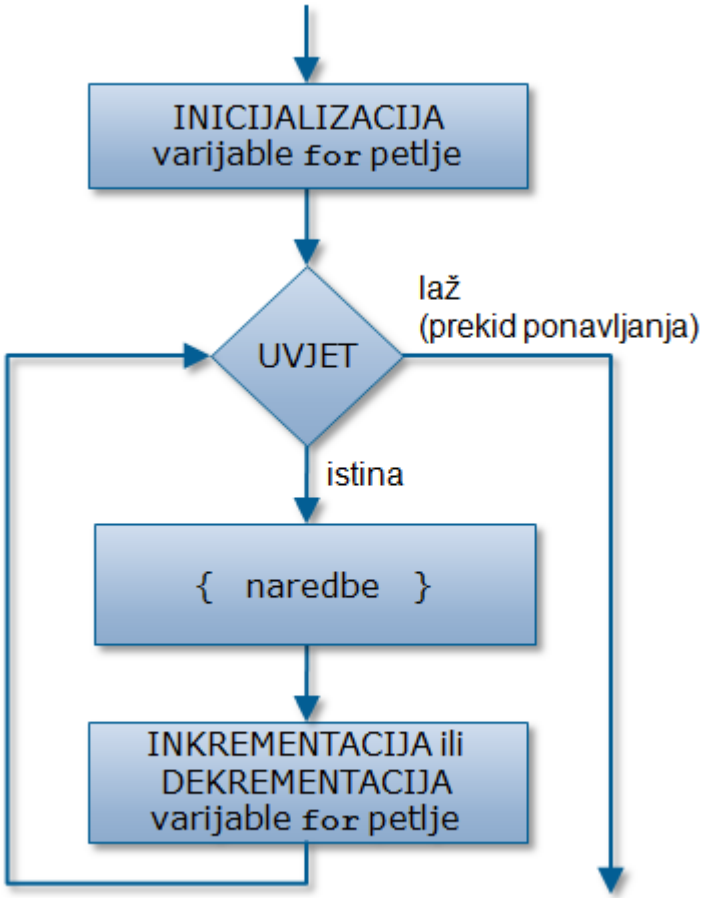
```
za i = početak do kraj (korak k)
    niz_naredbi
```

- C#

```
for (i = poc; i <= kraj; i = i + k)
    naredba;
```

– ili

```
for (i = poc; i <= kraj; i = i + k)
{
    niz_naredbi;
}
```



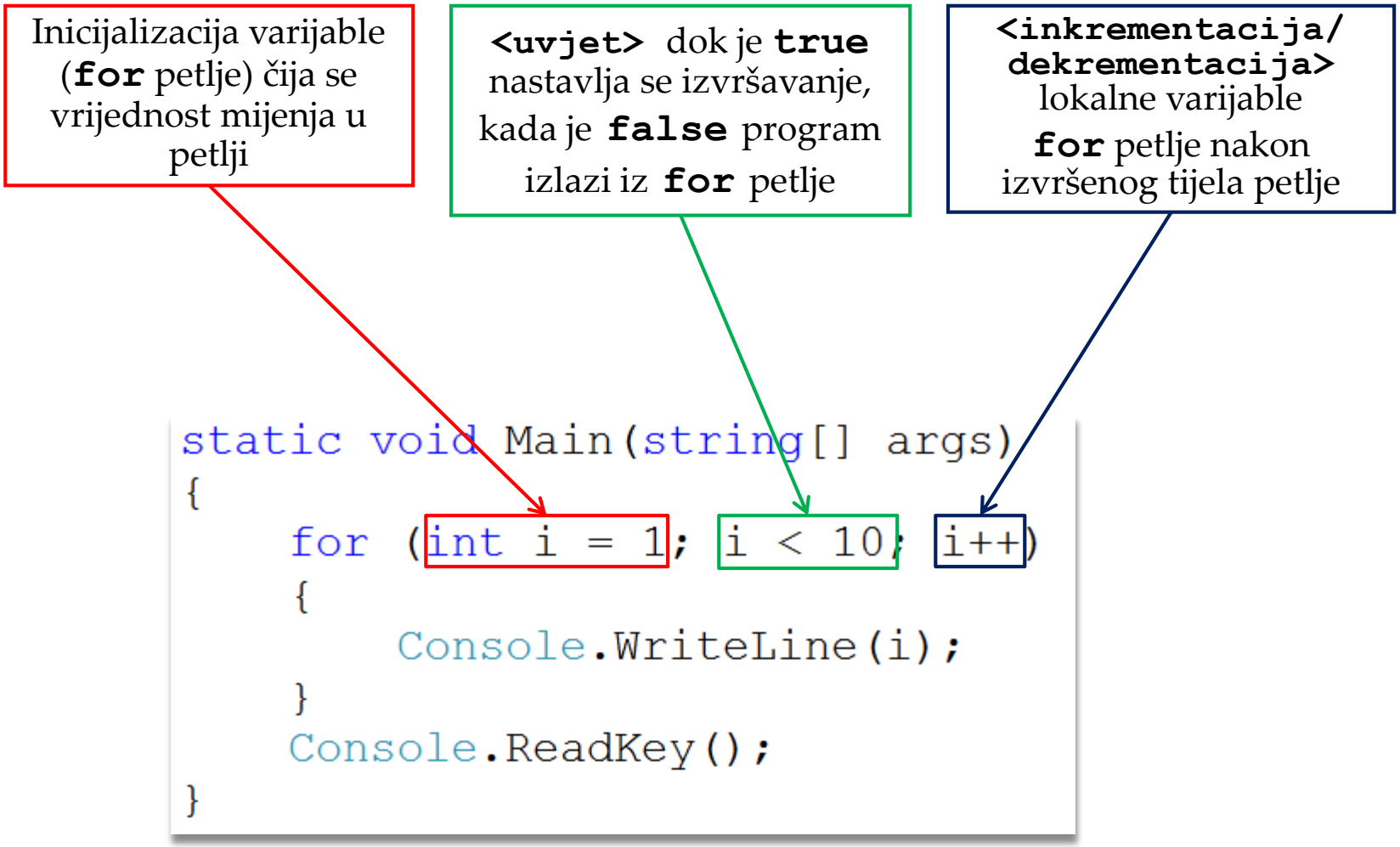
Petlja for

- Formalno napisano, tijek petlje for je slijedeći

```
for(<inicijalizacija>;<uvjet>;<inkrementacija/dekrementacija>)
{
    <naredbe>
}
```

  - **<inicijalizacija>** varijable for petlje čija se vrijednost u pravilu mijenja u svakoj iteraciji (inkrementira ili dekrementira)
  - **<uvjet>** dok je true nastavlja se izvršavanje petlje, kada je false program izlazi iz for petlje
  - **<inkrementacija/dekrementacija>** varijable for petlje

Petlja for



Petlja for - Inkrement

- INKREMENTIRANJE varijable i=i+1 u C# ima svoj skraćeni zapis i++

Inkrement

```
static void Main(string[] args)
{
    int i = 1;
    Console.WriteLine("brojac prije inkrementiranja -> " + i);
    i++;
    Console.WriteLine("brojac poslije inkrementiranja i++ -> " + i);
    i = i + 1;
    Console.WriteLine("brojac poslije inkrementiranja i = i + 1 -> " + i);
    Console.ReadKey();
}
```

```
Brojac prije inkrementiranja -> 1
Brojac poslije inkrementiranja i++ -> 2
Brojac poslije inkrementiranja i = i + 1 -> 3
```

`i++;` **Identični izrazi (skraćeni zapis)** `i = i + 1;`

Petlja for - Dekrement

- DEKREMENTIRANJE varijable i=i-1 u C# ima svoj skraćeni zapis i--

Dekrement

```
static void Main(string[] args)
{
    int i = 3;
    Console.WriteLine("brojac prije dekrementiranja -> " + i);
    i--;
    Console.WriteLine("brojac poslije dekrementiranja i-- -> " + i);
    i = i - 1;
    Console.WriteLine("brojac poslije dekrementiranja i = i - 1 -> " + i);
    Console.ReadKey();
}
```

```
brojac prije dekrementiranja -> 3
brojac poslije dekrementiranja i-- -> 2
brojac poslije dekrementiranja i = i - 1 -> 1
```

`i--;` **Identični izrazi (skraćeni zapis)** `i = i - 1;`

Petlja for - Inicijalizacija

- Osim inicijalizacije već postojeće varijable

```
int i;
for (i = 1; i <= 10; i++) {
    Console.WriteLine(i);
}
```

- Moguće je istovremeno i deklarirati i inicijalizirati lokalnu varijablu for petlje

```
for (int j = 1; j <= 10; j++) {
    Console.WriteLine(j);
}
```

  - Varijabla **j** je vidljiva jedino unutar tijela petlje
    - Nakon završetka petlje, varijabla **j** postaje nevidljiva

Petlja for - Uvjet

- Vrijednost koja određuje koliko će se puta varijabla inkrementirati (dekrementirati) ne mora biti konstanta, npr.

```
int x = 10;
for(int i = 0; i < x; i++) {
    Console.Write(i + " ");
    x = x – 2;
}
```

- Nakon izvršavanja gornjeg primjera, na zaslonu će se ispisati **0 1 2 3**
- Općenito, uvjet može biti bilo koji logički izraz odnosno vrijednost tipa **bool**

Petlja for

- Moguće je zarezom odvojiti naredbe u for petlji
- Kao operator odvajanja se zarez koristi tamo gdje je dopuštena samo jedna naredba

```
for ( i=0, j=60; i>j; i++, j--) {
    . . .
}
```

Petlja for – Česte greške

Kôd	Ispis

`int i;`
`for (i = 1; i = 10; i++)`
`{`
`  Console.WriteLine(i);`
`}`
	Kôd se ne može prevesti (C# ne može implicitno pretvoriti **int** u **bool**, C/C++ mogu)

`int i;`
`for (i = 1; i == 10; i++)`
`{`
`  Console.WriteLine(i);`
`}`
	Neće se ništa ispisati (naredba u tijelu petlje se neće niti jednom izvršiti)

`int i;`
`for (i = 1; i < 10; i++);`
`{`
`  Console.WriteLine(i);`
`}`
	Jednom se ispisuje broj 10

Petlja for - Primjer

- Uzlazno mijenjanje kontrolne varijable

```
for (i = poc; i <= kraj; i = i + k)
```

- Silazno mijenjanje kontrolne varijable

```
for (cv = 10; cv >0; cv = cv - 1)
```

- Dvije kontrolne varijable

```
for (si = 100, uz = 0; si >= uz; si = si-1, uz = uz + 1)
```

Petlja for - Primjer

- Napisati program koji će izračunati prosječnu starost vozila voznog parka tvrtke

ProsjecnaStarost

```
// deklaracija varijabli
int brojVozila;
double starostVozila, prosjecnaStarost;

// inicijalizacija varijabli
starostVozila = 0.0;

// dohvat podataka
Console.Write("Unesite broj vozila u voznom parku tvrtke > ");
brojVozila = Convert.ToInt32(Console.ReadLine());
Console.WriteLine();

// obrada podataka
for (int i = 0; i < brojVozila; i++)
{
    Console.Write("Unesite starost " + (i + 1) + ". vozila > ");
    starostVozila = starostVozila + Convert.ToDouble(Console.ReadLine());
}
prosjecnaStarost = starostVozila / brojVozila;

// ispis rezultata
Console.WriteLine("\nProsjecna starost vozila u voznom parku je " + prosjecnaStarost + " godina.");
```