Uvod u programiranje

- predavanja -

listopad 2025.

5. Kontrola toka programa

Vodič „04 Prije petog predavanja”

- Očekuje se da ste savladali (na razini na kojoj je obrađeno u vodiču):
  - Petlje:
    - while
    - do while
    - for

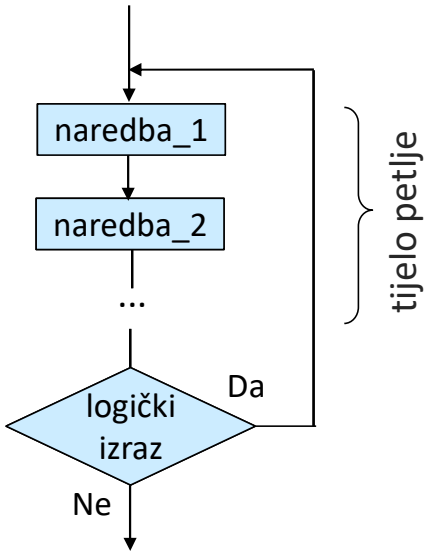
- Pitanja?

2. Programska petlja do while

S ispitivanjem uvjeta na kraju

Programska petlja s ispitivanjem uvjeta na kraju

Dijagram toka



C program - sintaksa

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

jedna naredba!

Što ako tijelo petlje sadrži više od jedne naredbe?

Rješenje: koristiti složenu naredbu.

C program - primjer

tijelo petlje

```
...
broj = 1;
do  {
   printf("%d ", broj);
   broj = broj + 1;
} while (broj <= 20);
...
```

Primjer

- Programski zadatak
  - Učitavati i sumirati cijele brojeve dok se ne upiše cijeli broj 0. Ispisati sumu učitanih brojeva.

```
#include <stdio.h>

int main(void) {
   int broj, suma = 0;

   do {
      printf("Upisite broj > ");
      scanf("%d", &broj);
      suma = suma + broj;
   } while (broj != 0);
   printf("Suma = %d\n", suma);
   return 0;
}
```

Primjer

- Programski zadatak
  - Učitati pozitivni cijeli broj koji određuje gornju granicu sume brojeva (ne treba provjeravati ispravnost učitanog broja). Zatim učitavati i sumirati cijele brojeve sve dok njihova suma ne prekorači zadanu gornju granicu sume. Nakon toga ispisati dosegnutu sumu, broj učitanih brojeva i aritmetičku sredinu učitanih brojeva.
  - Primjer izvršavanja programa

```
Upisite gornju granicu > 11↵
2↵
4↵
1↵
8↵
15 4 3.750000↵
```

Rješenje

```
#include <stdio.h>

int main(void) {
   int brojac = 0, suma = 0, broj, gg;
   float as;

   printf("Upisite gornju granicu > ");
   scanf("%d", &gg);

   do {
      scanf("%d", &broj);
      suma = suma + broj;
      brojac = brojac + 1;
   } while (suma <= gg);

   as = 1.f * suma / brojac;
   printf("%d %d %f\n", suma, brojac, as);
   return 0;
}
```

tijelo petlje obavit će se barem jednom jer se koristi petlja s ispitivanjem uvjeta ponavljanja na kraju

Množenje s 1.f potrebno radi realnog dijeljenja

Rješenje

```
#include <stdio.h>

int main(void) {
   int brojac = 0, suma = 0, broj, gg;
   float as;

   printf("Upisite gornju granicu > ");
   scanf("%d", &gg);

   while (suma <= gg) {
      scanf("%d", &broj);
      suma = suma + broj;
      brojac = brojac + 1;
   }

   as = 1.f * suma / brojac;
   printf("%d %d %f\n", suma, brojac, as);
   return 0;
}
```

tijelo petlje obavit će se barem jednom jer je u ovom slučaju uvjet za obavljanje tijela petlje na početku sigurno zadovoljen

Množenje s 1.f potrebno radi realnog dijeljenja

Primjer

- Programski zadatak
  - Učitavati cijele brojeve iz intervala [-100, 100]. Učitavanje brojeva prekinuti kada se učita broj izvan intervala [-100, 100]. Ispisati broj učitanih pozitivnih brojeva, broj učitanih negativnih brojeva i broj učitanih nula. U obzir uzeti samo brojeve iz intervala [-50, 50].

Rješenje

```
#include <stdio.h>
int main(void) {
   int broj;
   int brojPozitivnih = 0, brojNegativnih = 0, brojNula = 0;
   printf("Upisite brojeve > ");
   do {
      scanf("%d", &broj);
      if (broj >= -50 && broj <= 50) {
         if (broj == 0) brojNula = brojNula + 1;
         else if (broj > 0) brojPozitivnih = brojPozitivnih + 1;
         else brojNegativnih = brojNegativnih + 1;
      }
   } while (broj >= -100 && broj <= 100);

   printf("Pozitivnih je %d, negativnih je %d, nula je %d\n"
         , brojPozitivnih, brojNegativnih, brojNula);
   return 0;
}
```

Primjer

- Programski zadatak
  - S tipkovnice učitati 10 cijelih brojeva, odrediti i ispisati najveći broj.
  - Primjer izvršavanja programa

```
Upisite 10 cijelih brojeva >↵
3 -15 8 45 0 72 -99 72 0 11↵
Najveci broj je 72↵
```

Oblikovanje algoritma

- Kod traženja najveće (slično i kod traženja najmanje) vrijednosti u nekom nizu vrijednosti koristi se sljedeći algoritam:
  - prvu vrijednost proglasiti najvećom i pohraniti u pomoćnu varijablu koja predstavlja trenutačni maksimum
  - redom ispitivati jednu po jednu preostalu vrijednost i svaki puta kada se nađe vrijednost koja je veća od trenutačnog maksimuma, trenutačni maksimum ažurirati na tu vrijednost
  - nakon što se ispitaju sve vrijednosti, u pomoćnoj varijabli će se nalaziti najveća vrijednost
- Primjer: `3 -15 8 45 0 72 -99 72 0 11`

broj	3	-15	8	45	0	72	-99	72	0	11
		-15>3?	**8>3?**	**45>8?**	**0>45?**	**72>45?**	**-99>72?**	**72>72?**	**0>72?**	**11>72?**
maks	3	3	8	45	45	72	72	72	72	72

Rješenje (1. varijanta)

```
#include <stdio.h>
#define UKUP_BROJEVA 10                                   mora biti najmanje 1

int main(void) {
   int korak = 1, broj, maks;

   printf("Upisite %d cijelih brojeva >\n", UKUP_BROJEVA);
   scanf("%d", &broj);                                    prvi broj
   maks = broj;

   while (korak < UKUP_BROJEVA) {                         preostali brojevi
      korak = korak + 1;
      scanf("%d", &broj);
      if (broj > maks)
         maks = broj;
   }

   printf("Najveci broj je %d", maks);
   return 0;
}
```

Rješenje (2. varijanta)

```
#include <stdio.h>
#define UKUP_BROJEVA 10                                   mora biti najmanje 1

int main(void) {
   int korak = 0, broj, maks;

   printf("Upisite %d cijelih brojeva >\n", UKUP_BROJEVA);

   do {
      korak = korak + 1;
      scanf("%d", &broj);
      if (korak == 1)                                     prvi broj
         maks = broj;
      else                                                preostali brojevi
         if (broj > maks)
            maks = broj;
   } while (korak < UKUP_BROJEVA);

   printf("Najveci broj je %d", maks);
   return 0;
}
```

Zadatak

- ***The Simplest Math Problem No One Can Solve - Collatz Conjecture***

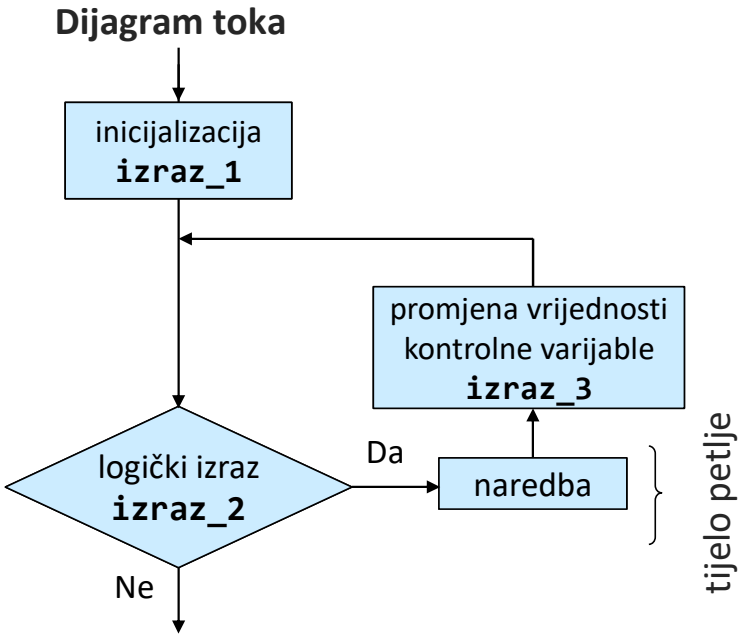
- *The **Collatz conjecture**[a] is one of the most famous unsolved problems in mathematics. The conjecture asks whether repeating two simple arithmetic operations will eventually transform every positive integer into 1. It concerns sequences of integers in which each term is obtained from the previous term as follows:*
  - *if the previous term is even, the next term is one half of the previous term.*
  - *If the previous term is odd, the next term is 3 times the previous term plus 1*

  The conjecture is that these sequences always reach 1, no matter which positive integer is chosen to start the sequence. The conjecture has been shown to hold for all positive integers up to 2.95×10^{20}, but no general proof has been found.

3. Programska petlja for

S poznatim brojem ponavljanja

Programska petlja s poznatim brojem ponavljanja



C program - sintaksa

```
for (izraz_1; izraz_2; izraz_3)
   naredba;
```

jedna naredba!

Što ako tijelo petlje sadrži više od jedne naredbe?

Rješenje: koristiti složenu naredbu.

C program - primjer Ispis neparnih brojeva iz intervala [1, 10]

```
...
  int brojac;
  for (brojac = 1; brojac <= 10; brojac = brojac + 2)
      printf("%d\n", brojac);
...
```

može se staviti 9

Značenje izraza u zagradama


```
for (izraz_1; izraz_2; izraz_3)
   naredba;
```

- **izraz_1** je izraz koji će se izvršiti samo jednom, prije ulaska u prvu iteraciju petlje. Najčešće se koristi za inicijalizaciju brojača.
- **izraz_2** je logički izraz. Tijelo petlje se izvršava ako je izraz_2 zadovoljen (istinit). Ako nije, petlja se prekida (nastavlja se s prvom sljedećom naredbom iza petlje).
- **izraz_3** se obavlja nakon svakog prolaska kroz tijelo petlje. Najčešće se koristi za povećavanje/smanjenje vrijednosti kontrolne varijable/brojača. Nakon obavljanja izraza izraz_3, ponovo se testira uvjet u izraz_2
- Svaki od izraza (izraz_1, izraz_2, izraz_3) se može izostaviti. Ako se izostavi izraz_2, smatra se da je rezultat izraza izraz_2 uvijek istina.

Programska petlja s poznatim brojem ponavljanja

- svaka petlja s poznatim brojem ponavljanja (for-petlja) može se realizirati pe s ispitivanjem uvjeta na početku. Glavni razlozi za korištenje for-petlje su:
  - uputa ostalim programerima da se radi o petlji za koju se odmah na početku može izračunati koliko puta će se tijelo petlje izvršiti
  - mogućnost pisanja kompaktnijeg koda

```
for (izraz_1; izraz_2; izraz_3)
   naredba;
```

=

```
izraz_1;
while (izraz_2) {
   naredba;
   izraz_3;
}
```

Primjer

- Programski zadatak
  - Učitati pozitivan cijeli broj *n* (nije potrebno provjeravati je li učitan ispravan broj). Zatim učitati *n* cijelih brojeva, izračunati i na zaslon ispisati njihovu aritmetičku sredinu
  - Koja vrsta petlje je najprikladnija za rješavanje ovog zadatka?
  - Primjer izvršavanja programa:

```
Koliko brojeva zelite ucitati? > 5↵
Upisite 1. broj > 3↵
Upisite 2. broj > -15↵
Upisite 3. broj > 8↵
Upisite 4. broj > 45↵
Upisite 5. broj > 7↵
Aritmeticka sredina je 9.600↵
```

Rješenje

```
#include <stdio.h>

int main(void) {
   int n, brojac, ucitani_broj, suma = 0;
   float arit_sred;

   printf("Koliko brojeva zelite ucitati? > ");
   scanf("%d", &n);

   for (brojac = 1; brojac <= n; brojac = brojac + 1) {
      printf("Upisite %d. broj > ", brojac);
      scanf("%d", &ucitani_broj);
      suma = suma + ucitani_broj;
   }

   arit_sred = 1.f * suma / n;
   printf("Aritmeticka sredina je %.3f\n", arit_sred);
   return 0;
}
```

radi realnog dijeljenja

Primjer

- Programski zadatak
  - Učitati pozitivan cijeli broj *n* (nije potrebno provjeravati je li učitan ispravan broj). Zatim od većih prema manjim, ispisati sve prirodne brojeve djeljive sa 7, 13 ili 19, koji su manji od broja *n*.
  - Koja je vrsta petlje najprikladnija za rješavanje ovog zadatka?
  - Primjer izvršavanja programa:

```
Upisite broj n > 30↵
28↵
26↵
21↵
19↵
14↵
13↵
7↵
```

Rješenje

```
#include <stdio.h>

int main(void) {
   int i, n;

   printf("Upisite broj n > ");
   scanf("%d", &n);

   for (i = n - 1; i >= 7; i = i - 1) {
      if ((i % 7 == 0) ||
          (i % 13 == 0) ||
          (i % 19 == 0)) {
         printf("%d\n", i);
      }
   }
   return 0;
}
```

u konkretnom slučaju oba para vitičastih zagrada mogu se ispustiti

Primjer

- Programski zadatak
  - Učitati nenegativan cijeli broj n (nije potrebno provjeravati je li učitan ispravan broj). Ispisati prvih n članova Fibonaccijevog niza.
  - Primjer izvršavanja programa:

```
Upisite broj clanova Fibonaccijevog niza > 15↵
1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610
```

i = 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

  - definicija niza:

$a_1 = a_2 = 1$

$a_i = a_{i-1} + a_{i-2}$ za $i > 2$

Rješenje

```
#include <stdio.h>

int main(void) {
   int n, i, a_i_minus2 = 1, a_i_minus1 = 1, a_i;

   printf("Upisite broj clanova Fibonaccijevog niza > ");
   scanf("%d", &n);

   if (n >= 1) printf("%d ", 1);
   if (n >= 2) printf("%d ", 1);
   for (i = 3; i <= n; i = i + 1) {
      a_i = a_i_minus1 + a_i_minus2;
      printf("%d ", a_i);
      a_i_minus2 = a_i_minus1;
      a_i_minus1 = a_i;
   }
   return 0;
}
```

```
1 1 2 3 5 8 13 ...
```

Pojednostavljeno rješenje:

- prva dva člana se ispisuju na jedan, a ostali članovi na drugi način
- uočiti "if" kod ispisa prva dva člana
- nije riješeno pitanje ispisa zareza između članova

Rješenje

```
#include <stdio.h>

int main(void) {
   int n, i, a_i_minus2 = 1, a_i_minus1 = 1, a_i = 1;

   printf("Upisite broj clanova Fibonaccijevog niza > ");
   scanf("%d", &n);

   for (i = 1; i <= n; i = i + 1) {
      if (i > 2) {
         a_i = a_i_minus1 + a_i_minus2;
         a_i_minus2 = a_i_minus1;
         a_i_minus1 = a_i;
      }
      printf("%d ", a_i);
   }
   return 0;
}
```

```
1 1 2 3 5 8 13 ...
```

Unaprijeđeno rješenje:

- svi članovi se ispisuju istom naredbom
- u prva dva koraka petlje ispisuje se inicijalna vrijednost a_i, a u ostalim koracima se vrijednost a_i prije ispisa izračunava
- još uvijek nije riješeno pitanje ispisa zareza između članova

Rješenje

```
#include <stdio.h>

int main(void) {
   int n, i, a_i_minus2 = 1, a_i_minus1 = 1, a_i = 1;

   printf("Upisite broj clanova Fibonaccijevog niza > ");
   scanf("%d", &n);

   for (i = 1; i <= n; i = i + 1) {
      if (i > 2) {
         a_i = a_i_minus1 + a_i_minus2;
         a_i_minus2 = a_i_minus1;
         a_i_minus1 = a_i;
      }
      if (i > 1) printf(", ");
      printf("%d", a_i);
   }
   return 0;
}
```

```
1, 1, 2, 3, 5, 8, 13, ...
```

Primjer

- Programski zadatak
  - Ispisati tablicu množenja do 100, u 10 redaka i 10 stupaca.
  - Primjer izvršavanja programa:

```
  1   2   3   4   5   6   7   8   9  10
  2   4   6   8  10  12  14  16  18  20
  3   6   9  12  15  18  21  24  27  30
  4   8  12  16  20  24  28  32  36  40
  5  10  15  20  25  30  35  40  45  50
  6  12  18  24  30  36  42  48  54  60
  7  14  21  28  35  42  49  56  63  70
  8  16  24  32  40  48  56  64  72  80
  9  18  27  36  45  54  63  72  81  90
 10  20  30  40  50  60  70  80  90 100
```

Primjer

- Je li ovakvo rješenje prihvatljivo?

```
int stupac;
for (stupac = 1; stupac <= 10; stupac = stupac + 1)
   printf("%4d", 1 * stupac);
printf("\n");

for (stupac = 1; stupac <= 10; stupac = stupac + 1)
   printf("%4d", 2 * stupac);
printf("\n");

for (stupac = 1; stupac <= 10; stupac = stupac + 1)
   printf("%4d", 3 * stupac);
printf("\n");

... itd. za 4, 5, 6, 7, 8, i 9. redak

for (stupac = 1; stupac <= 10; stupac = stupac + 1)
   printf("%4d", 10 * stupac);
printf("\n");
```

Rješenje

Dobro rješenje

```
#include <stdio.h>

int main(void) {
   int redak, stupac;

   for (redak = 1; redak <= 10; redak = redak + 1) {
      for (stupac = 1; stupac <= 10; stupac = stupac + 1) {
         printf("%4d", redak * stupac);
      }
      printf("\n");
   }
   return 0;
}
```

```
   int i;
   for (i = 0; i < 100; i = i + 1) {
      printf("%4d", (i / 10 + 1) * (i % 10 + 1));
      if ((i + 1) % 10 == 0) {
         printf("\n");
      }
   }
```

Primjer: odabir vrste petlje

- Programski zadatak
  - Učitati nenegativan cijeli broj n (nije potrebno provjeravati je li učitan ispravan broj). Izračunati i ispisati n faktorijela
  - Zadatak riješiti
    - petljom s ispitivanjem uvjeta na početku
    - petljom s ispitivanjem uvjeta na kraju
    - petljom s poznatim brojem ponavljanja
  - Procijeniti koja je vrsta petlje najprikladnija
  - Primjeri izvršavanja programa:

```
Upisite n > 12↵
12! = 479001600
```

```
Upisite n > 0↵
0! = 1
```

Rješenje (1)

- programska petlja s ispitivanjem uvjeta na početku

```
#include <stdio.h>

int main(void) {
   int n, i, fakt;

   scanf("%d", &n);
   fakt = 1;
   i = 2;
   while (i <= n) {
      fakt = fakt * i;
      i = i + 1;
   }

   printf("%d! = %d", n, fakt);
   return 0;
}
```

```
...
if (n >= 2) {
   fakt = 1;
   i = 2;
   while (i <= n) {
      fakt = fakt * i;
      i = i + 1;
   }
} else {
   fakt = 1;
}
...
```

Rješenje (2)

- programska petlja s poznatim brojem ponavljanja

```
#include <stdio.h>

int main(void) {
   int n, i, fakt;

   scanf("%d", &n);
   fakt = 1;
   for (i = 2; i <= n; i = i + 1)  {
      fakt = fakt * i;
   }
   printf("%d! = %d", n, fakt);
   return 0;
}
```

```
i = 2;
while (i <= n) {
   fakt = fakt * i;
   i = i + 1;
}
```

Rješenje (3)

- programska petlja s ispitivanjem uvjeta na kraju

```
#include <stdio.h>

int main(void) {
   int n, i, fakt;

   scanf("%d", &n);
   fakt = 1;
   i = 1;                                  ne smije se početi od 2 zbog do-while
   do {
      fakt = fakt * i;
      i = i + 1;
   } while (i <= n);

   printf("%d! = %d", n, fakt);
   return 0;
}
```

Prije sljedećeg predavanja

- Edgar:
  - Tutorial: **05. Prije šestog predavanja**
  - Public exams: **5. vježbe uz predavanja – kontrola toka programa**