Programiranje za elektrotehnike 1

Uvod v programski jezik C

Vsebina poglavja

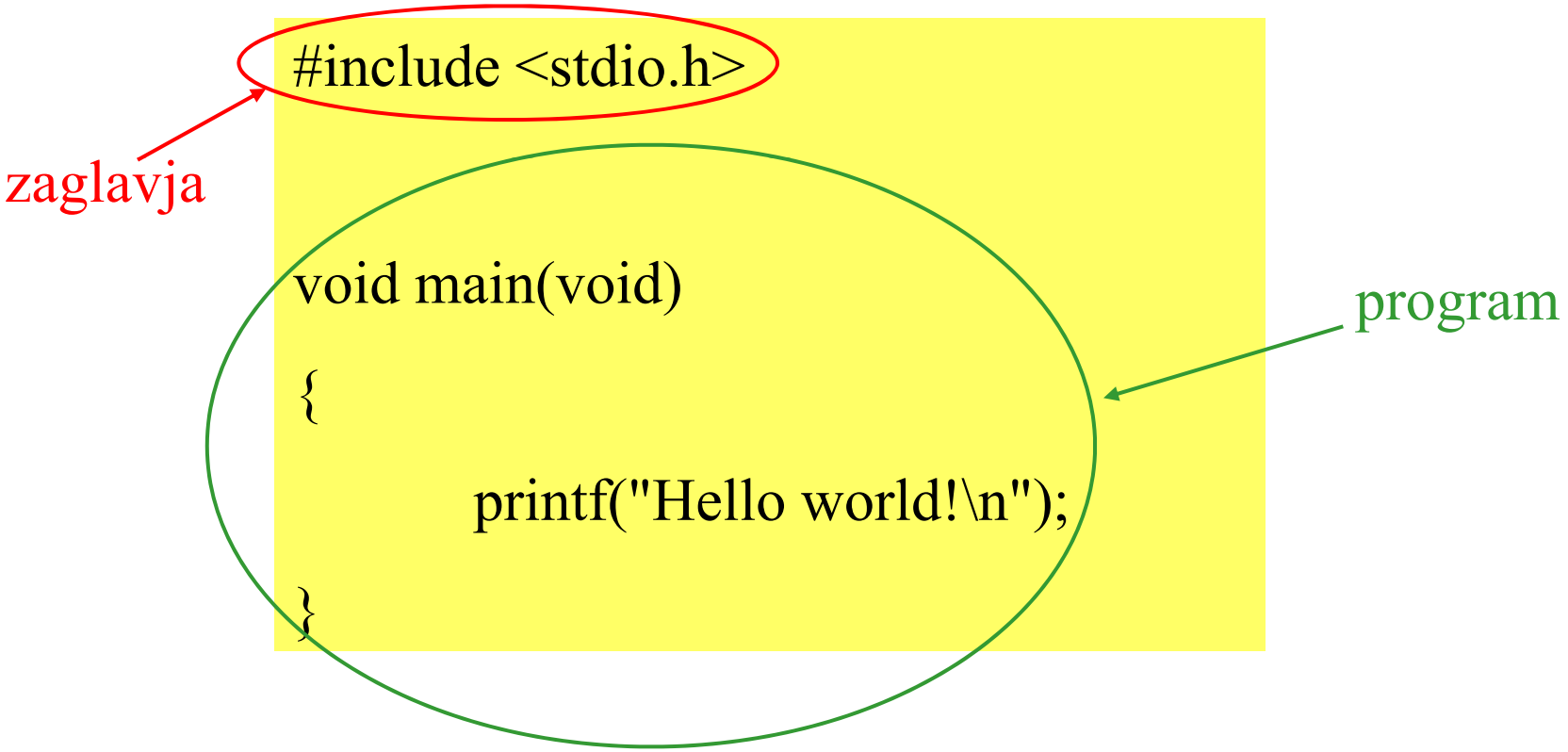
- Kratek opis jezika C
- Prvi program v C
- Pisanje programa v C
- Prevajanje programa v C
- Spremenljivke
  - Prireditve
  - Osnovne računske operacije
    - Vrstni red izvajanja
- Standardne knjižnice
  - Vejitve - odločitve
    - Osnovne logične operacije
  - Zanke - ponavljanje

Kratek opis jezika C

- Višji programski jezik.
- Blizu strojni opremi.
- Omogoča pisanje ’čistih’ uporabniških, pa tudi sistemskih programov, kot so operacijski sistemi
  - začetki C so v UNIXu,
  - Linux je pisan v C.
- Osnova je ANSI C jezik.
- Za vse C razen ANSI ’C’-ja velja, da niso povsem združljivi (kompatibilni) med sabo. ANSI C naj bi razumele vse ostale izvedenke, obratno pa ne velja.
- Nadaljnji razvoj je v C++, objektno orientiran C.

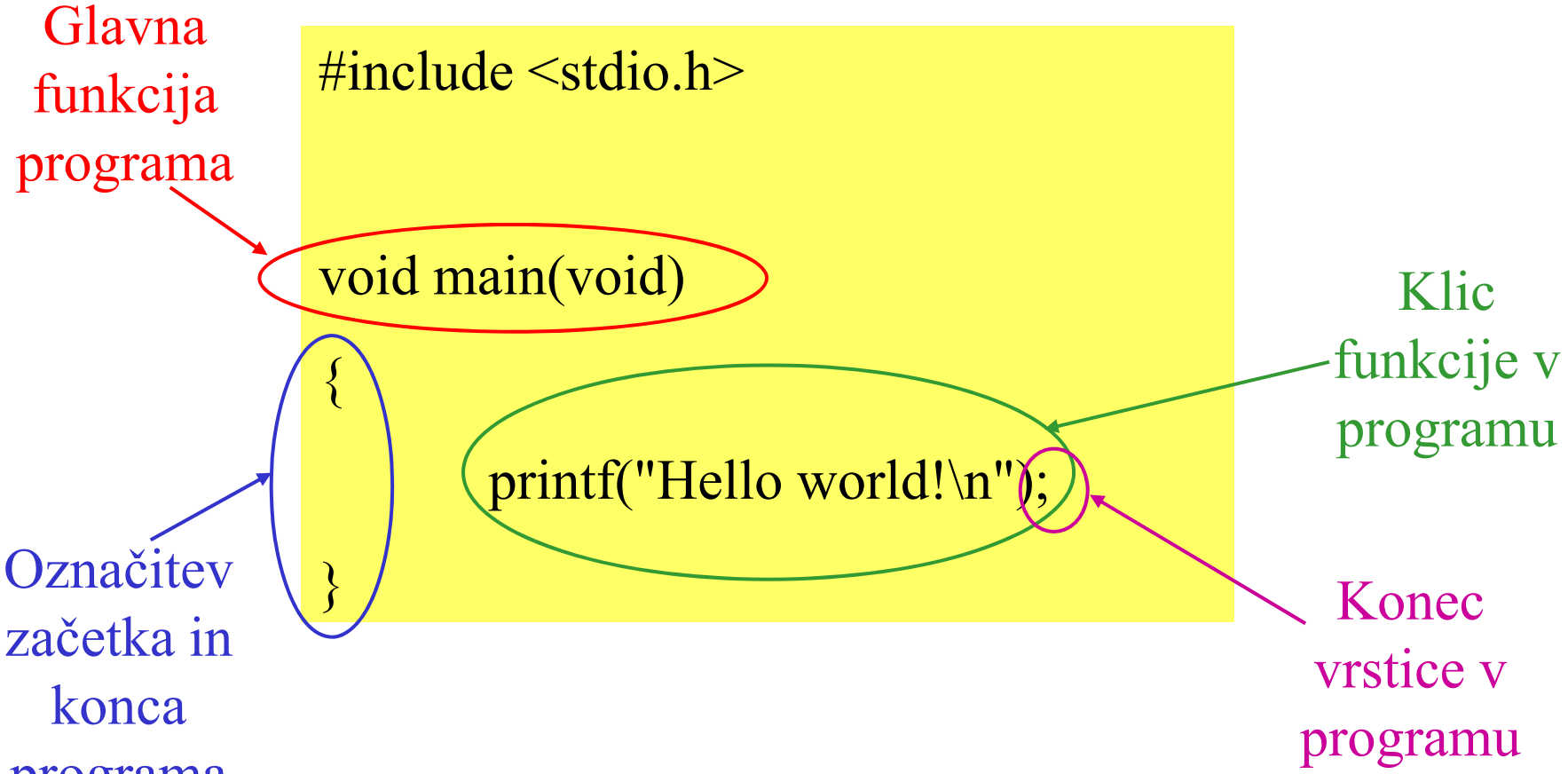
Prvi program v C

- Klasični “Hello world!”:



Prvi program v C

- Klasični “Hello world!”:



Pisanje programa v C

- **Deklariranje:**
  - Pokažemo, da spremenljivke in funkcije obstajajo in kakšne so.
- **Definiranje:**
  - Opis objekta programa
  - Rezervacija pomnilniškega prostora
  - Lahko tudi prireditev začetne vrednosti
  - Definicija velja tudi kot deklaracija, obratno pa ne velja.
- Pri pisanju velja da imamo:
  - bloke
    - v zavitih oklepajih {}
  - stavke
    - zaključeni s podpičjem ;
  - komentarje
    - /*komentar*/
    - novejši C (in C++) uporabljajo za komentar "//" na začetku vrstice, velja do konca vrstice

Prevajanje programa v C

- Program v jeziku C zapišemo v datoteko, ki ima končnico .c
- Za prevajanje uporabimo prevajalnik, v našem primeru je za to uporabljeno orodje *Bloodshed Dev-C++*.

Spremenljivke

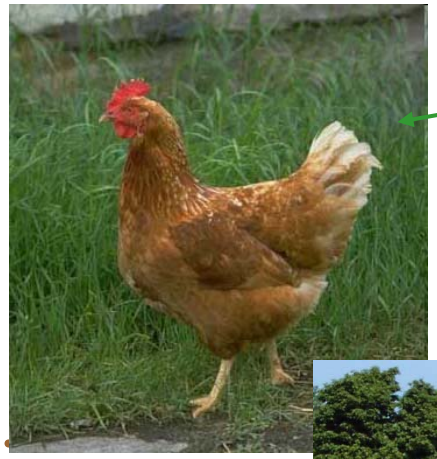
- Spremenljivka je definirana:
  - ime spremenljivke
    - lahko ima poljubno število znakov
    - prevajalnik loči prvih x znakov
  - tip podatka (celo, realno, znak,...)
  - način hranjenja spremenljivke (kje in koliko časa)
- Ime spremenljivke:
  - Začne se s črko.
  - Poleg črk lahk vsebujejo številke in znak _ .
  - Razlikujemo velike in male črke!
  - Ne smemo uporabljati rezerviranih besed.

Spremenljivke – rezervirane besede

auto	break	case	char	const
continue	default	do	double	else
enum	extern	float	for	goto
if	int	long	register	return
short	signed	sizeof	static	struct
switch	typedef	union	unsigned	void
volatile	while			

Celoštevilčne spremenljivke

- Cela števila, npr.:
  - Število nog, študentov, ...
- Označevanje v C:
  - int a
- Primer uporabe v programu:



1 kokoš

2 nogi

X peres

1 krava

4 noge

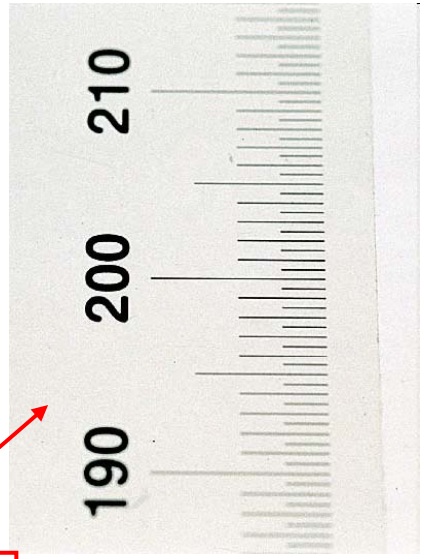
1 rep

```
#include <stdio.h>
void main(void)
{
        int krava_nog;
        krava_nog = 4;
        printf("a=%d\n",krava_nog);
}
```

Spremenljivke s plavajočo vejico

- Racionalna števila, npr.:
  - 1/10, 1/2, ...
- Ne realna števila – ločljivost!
- Označevanje v C:
  - float b
- Primer uporabe v programu:

termometer



merila

```
#include <stdio.h>
void main(void)
{
        float temperatura = 36.5;
        printf("Temperatura=%f\n",temperatura);
}
```

Prireditve

- Izraz ima levo in desno stran:
  - a = b + c;
- Najprej se izvede desna stran, nato pa še prireditev.
  - To omogoča ukaze, kot je npr.:
    - a = a+1;
  - Obstajajo izjeme!
- Vrednost na desni strani se samodejno prilagaja tipu spremenljivke na levi strani, lahko pa izvedemo tudi določitev (casting):
  - a = (int)b;

Prireditve - primer

- Zaokroževanje:
  - Na desetinko mm merimo nivo tekočine, želimo samo informacijo, kateri nivo je presežen, npr. Za opozarjanje:

```
#include <stdio.h>
void main(void)
{
        float nivo;
        int presezen_nivo;
        nivo = 15.3;            /* namesto meritve */
        presezen_nivo = (int)nivo;
        printf("Presežen je bil nivo %d mm!\n",presezen_nivo);
}
```

Osnovne računske operacije

- Seštevanje
  - c = a+b;
- Odštevanje
  - c = a-b;
- Množenje
  - c = a*b;
- Deljenje
  - c = a/b;
- Negacija
  - c = -b;

- Primer uporabe v programu:

```
#include <stdio.h>
void main(void)
{
        int a,b,c;
        a=3; b=1;
        c = a+b;     /* seštevanje */
        printf("%d + %d = %d\n",a,b,c);
        c = a-b;     /* odštevanje */
        printf("%d * %d = %d\n“,a,b,c);
        c = a*b;     /* množenje */
        printf("%d * %d = %d\n",a,b,c);
        c = a/b;     /* deljenje */
        printf("%d / %d = %d\n",a,b,c);
        c = -b;      /* negacija */
        printf("negativna vrednost %d = %d\n",b,c);
}
```

Nekaj dodatnih računskih operacij v C

- Ostanek deljenja:
  - c = a%b;
- Inkrement
  - c = a++;
    - Priredi in povečaj za 1
  - c = ++a;
    - Povečaj za 1 in priredi
- Dekrement
  - c = a--;
    - Priredi in zmanjšaj za 1
  - c = --a;
    - Zmanjšaj za 1 in priredi

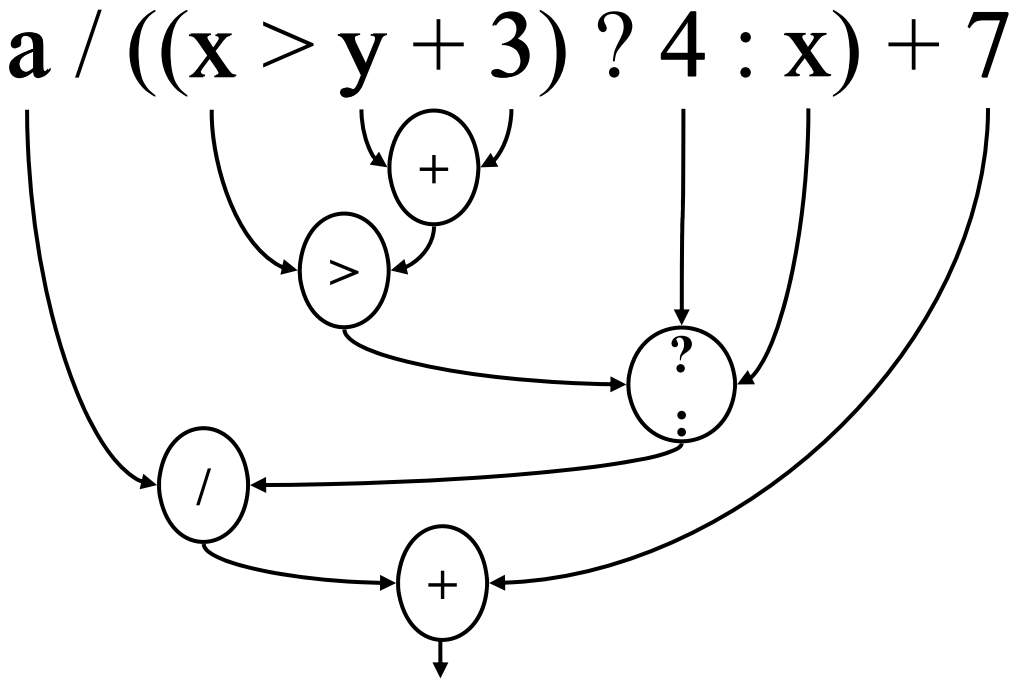
- Inkrement in dekrement sta pogosto uporabljena:
  - Povečevanje in zmanjševanje vrednosti
    - Štetje,
    - Meritve časa, ...
    - Zanke!
- Leva stran ukaza ni potrebna:
  - Primer ukaza v C:
    - a++;

Pogojni operator

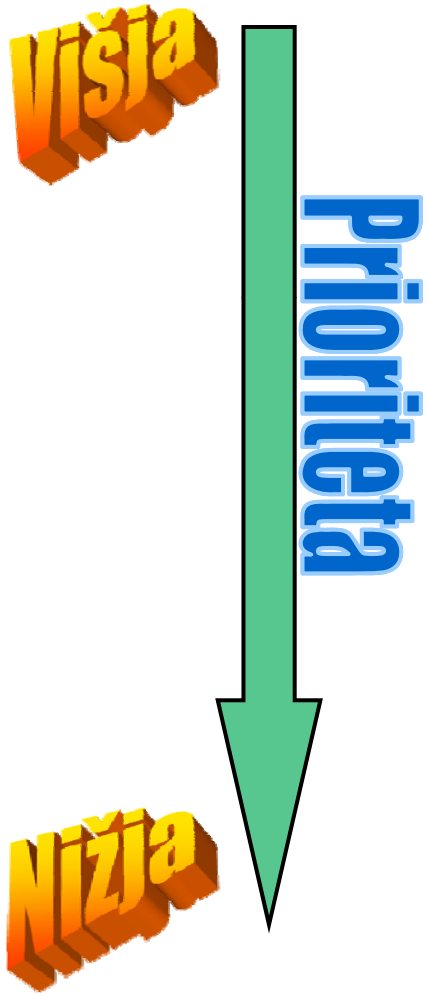
- Splošna oblika:
  - izraz1 ? izraz2 : izraz3
- Pomen:
  - Če je vrednost izraz1 TRUE (ni nič), potem je celotni izraz po vrednosti in tipu enak izrazu2 sicer je celotni izraz po tipu in vrednosti enak izrazu3.
- Primer:
  - c = (a>b)?a:b;
    - Spremenljivka c dobi večjo od vrednosti a in b.

Vrstni red izvajanja

- Veljajo pravila matematičnega zapisa
- Oklepaji imajo vedno najvišji nivo
- Primer:

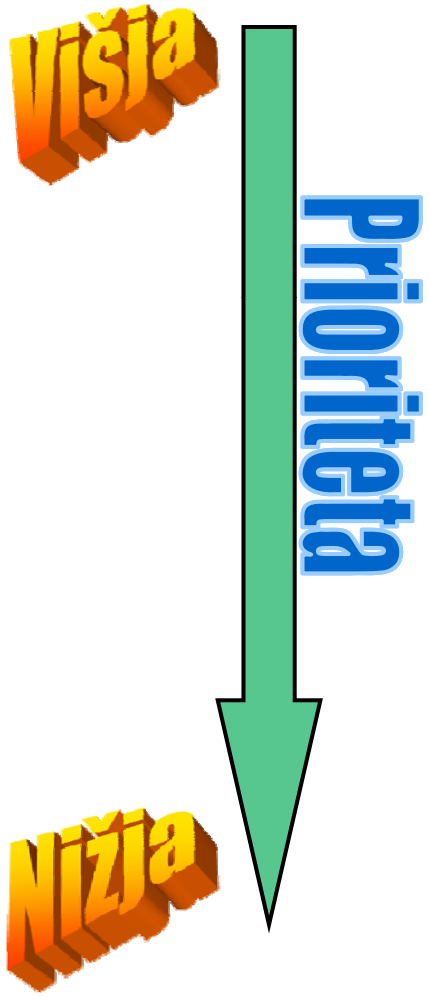


Operatorji in njihove prioritete



element polja	[]
klic funkcije	()
postfiksni inkrement, dekrement	*spr*++, *spr*--
element strukture	*ime.element*
element str. na katero kaže kazalec	*kazalec->element*
prefiksni inkrement, dekrement	++*spr*, --*spr*
unarni plus in minus	+, -
logični ne, bitni ne	!, ~
vsebina kazalca	**kazalec*
naslov spremenljivke	&*spremenljivka*
pretvorba tipa	(*nov_tip*)*spremenljivka*
velikost objekta	sizeof(*tip*)
množenje, deljenje, ostanek	*, /, %
seštevanje, odštevanje	+, -

Operatorji in njihove prioritete



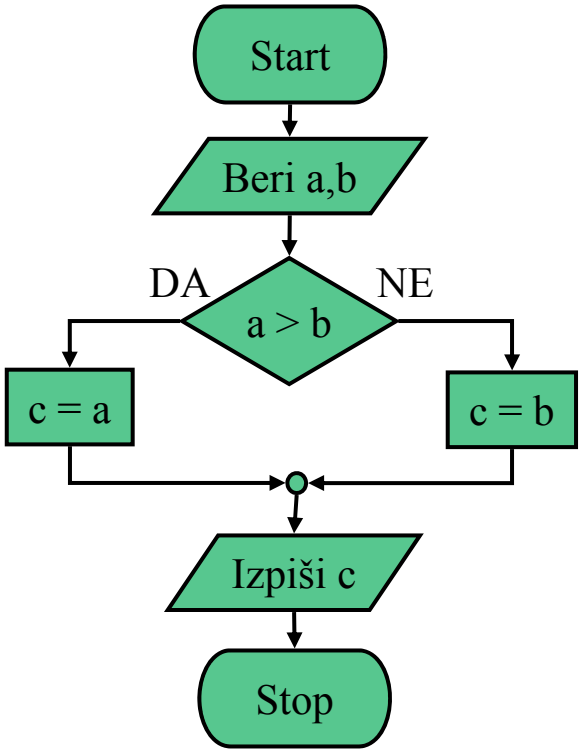
bitni premik levo, desno	<<, >>
primerjave	<, <=, >, >=
primerjave	==, !=
bitni in	&
bitni ekskluzivni ali	^
bitni ali	|
logični in	&&
logični ali	||
pogojni operator	*izraz* ? *izr1* : *izr2*
prireditveni operatorji	=, +=, -=, *=, /=, ...
ločilo med izrazi	,

Standardne knjižnice

- **if** stavek
- **switch** stavek
- **while do** zanka
- **do while** zanka
- **for** stavek
- **go to** (se izogibamo)
- Moč C dajejo funkcije iz knjižnic!

Vejitve - odločitve

- Uporabljamo ukaz **if**
- Primer – izbira večje vrednosti:

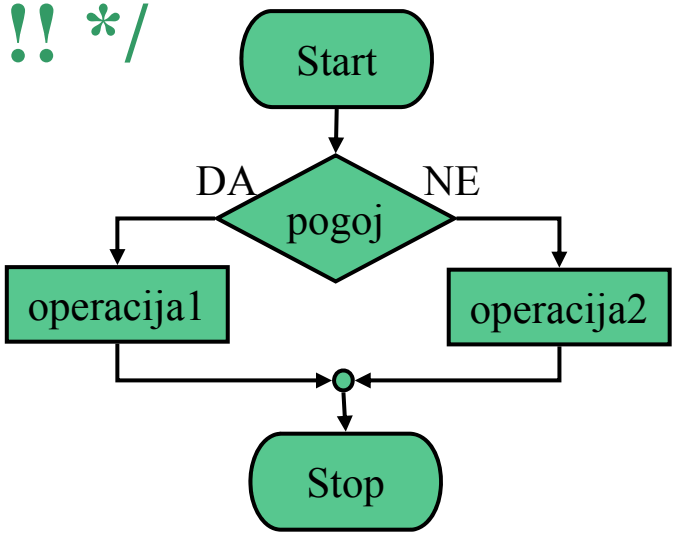


- Primer programa:

```
#include <stdio.h>
void main(void)
{
        int a,b,c;
        printf("\nVpisi prvo vrednost: ");
        scanf("%d",&a);
        printf("\nVpisi drugo vrednost: ");
        scanf("%d",&b);
        if(a>b)
        {
                c=a;
        }
        else
        {
                c=b;
        }
        printf("\nVecja vrednost je %d.\n",c);
}
```

Stavek *if-else*

```
if(pogoj)                    /* ni podpičja!!! */
{
   operacija1;               /* DA */
}
else                         /* ni podpičja!!! */
{
   operacija2;               /* NE */
}
```



Osnovne logične operacije

- Večje (>):
  - (a > b)
- Manjše (<):
  - (a < b)
- Enako (==):
  - (a == b)
- Različno (!=):
  - (a != b)
- Večje ali enako (>=):
  - (a >= b)
- Manjše ali enako (<=):
  - (a <= b)

Kompleksnejše operacije:

- Logični ALI (||)
  - Če je izpolnjen kateri od pogojev
    - (pogoj1) || (pogoj2)
- Logični IN (&&)
  - Če sta izpolnjena oba pogoja
    - (pogoj1) && (pogoj2)
- Logični NE
  - Če pogoj ni izpolnjen
    - (!(pogoj))
    - Primer:
      - if(!(a>b))

A	B	A|B
0	0	0
0	1	1
1	0	1
1	1	1

A	B	A&B
0	0	0
0	1	0
1	0	0
1	1	1

A	!A
0	1
1	0

Primer programa – najmanjša vrednost

```
#include <stdio.h>
void main(void)
{
        int v1, v2, v3, min_v;
        printf("\nVpisi prvo vrednost: ");
        scanf("%d",&v1);
        printf("\nVpisi drugo vrednost: ");
        scanf("%d",&v2);
        printf("\nVpisi tretjo vrednost: ");
        scanf("%d",&v3);
        if((v1<v2)&&(v1<v3))
        {
                min_v = v1;
        }
        else
        {
                if(v2 < v3) min_v = v2;
                else  min_v = v3;
        }
        printf("\nNajmanjša vrednost je %d.\n",min_v);
}
```

Program izpiše najmanjšo od treh vpisanih vrednosti.

Gnezditev!

Možen je tudi krajši zapis, brez {}

Razlika med = in ==

- Pogosta napaka pri programiranju:
- (a = b) pomeni nekaj drugega kot (a==b)
  - Pogoj (a=b) bo izpolnjen, ko bo b različen od 0
  - Pogoj (a==b) bo izpolnjen, ko bo a enak b

```
v1 = 0; v2 = 0;
if(v1 = v2)
{
        printf("Vrednosti sta enaki!\n");
}
else
{
        printf("Vrednosti sta razlicni!\n");
}
```

Izpiše: Vrednosti sta razlicni!

```
v1 = 0; v2 = 0;
if(v1 == v2)
{
        printf("Vrednosti sta enaki!\n");
}
else
{
        printf("Vrednosti sta razlicni!\n");
}
```

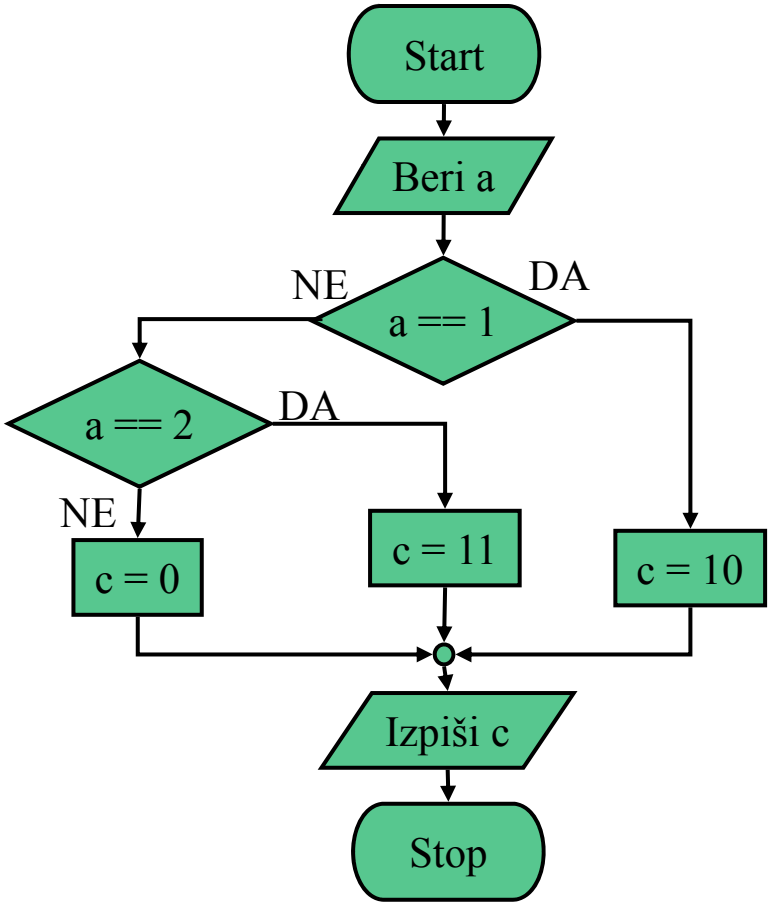
Izpiše: Vrednosti sta enaki!

Vejitev – več možnosti

- Možen je še en tip vejitve.
- Uporabljamo ga, ko imamo več možnosti.
  - Tipke
  - Režimi delovanja
  - ...
- Dosežemo boljšo preglednost programa.
- Uporabimo ukaz **switch.**

Vejitve z ukazom *switch*

- Primer – izbira izmed več možnostmi:

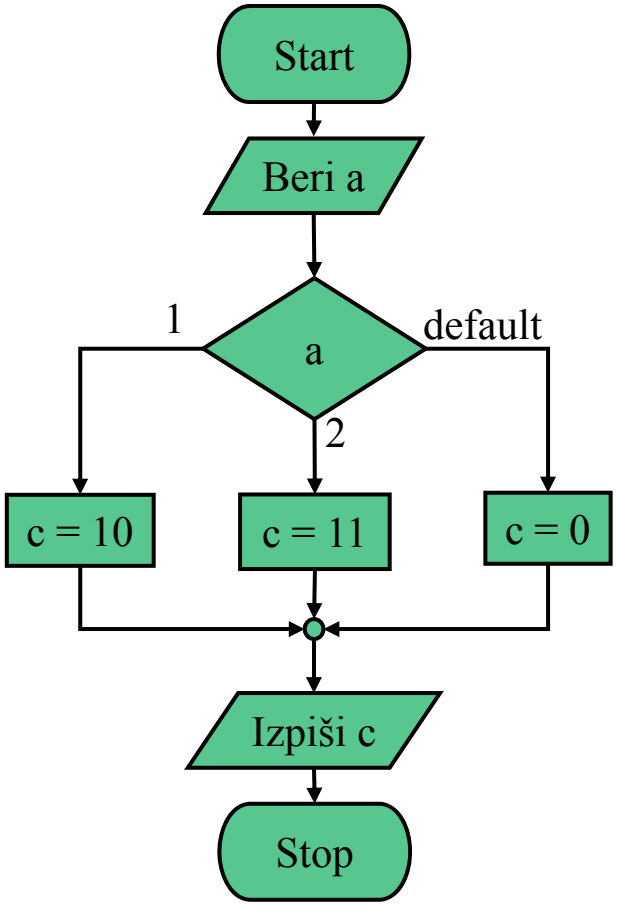


- Primer programa:

```
#include <stdio.h>
void main(void)
{
        int a,c;
        printf("\nVpisi izbiro: ");
        scanf("%d",&a);
        switch(a)
        {
                case 1:
                                c=10;
                                break;
                case 2:
                                c=11;
                                break;
                default:
                                c=0;
                                break;
        }
   }
   printf("c=%d.\n",c);
}
```

Vejitve z ukazom *switch*

- Primer – diagram poteka za *switch*:



- Primer programa:

```
#include <stdio.h>
void main(void)
{
        int a,c;
        printf("\nVpisi izbiro: ");
        scanf("%d",&a);
        switch(a)
        {
                case 1:
                                c=10;
                                break;
                case 2:
                                c=11;
                                break;
                default:
                                c=0;
                                break;
        }
   }
   printf("c=%d.\n",c);
}
```

Stavek *switch*

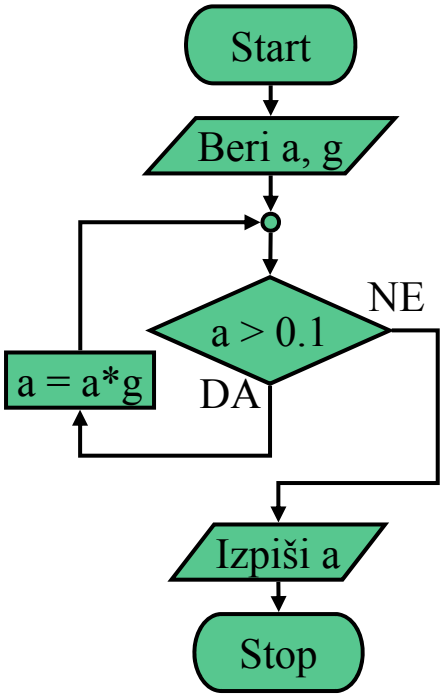
```
switch(a)                  /* ni podpičja!! a je tipa int ali char */
{
case 1:                          /* a je enak 1 */
    operacija1;
    break;                       /* izhod iz zanke */
case 2:                          /* a je enak 2 */
    operacija2;
    break;                       /* izhod iz zanke */
default:                         /* a ni enak 1 ali 2 */
    operacija3;
    break;                       /* tukaj break ni potreben */
}
```

Zanke - ponavljanje

- Tipi zank v jeziku C
  - **while do** zanka
    - Ponavljaj, dokler je pogoj izpolnjen.
    - Najprej preveri pogoj, nato izvedi zanko.
  - **do while** zanka
    - Ponavljaj, dokler je pogoj izpolnjen.
    - Najprej izvedi zanko, nato preveri pogoj.
  - **for** stavek
    - Ponavljaj.
    - Ponovi x krat ali dokler je pogoj izpolnjen!

Zanke – while do

- Primer – padajoče geometrijsko zaporedje:
  - $a_k = a_{k-1} * g$
    - g<1, iskanje prvega a, ki je manjši ali enak 0.1



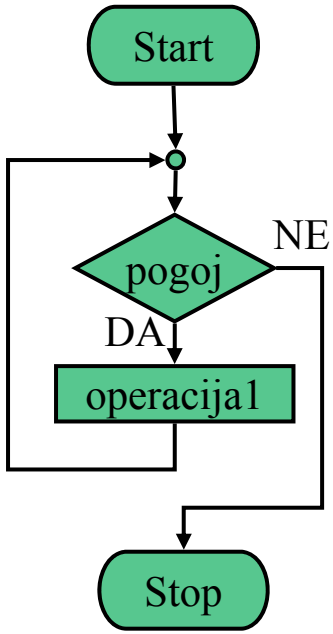
- Primer programa:

```
#include <stdio.h>
void main(void)
{
        float a,g;
        printf("\nVpisi zacetno vrednost: ");
        scanf("%f",&a);
        printf("\nVpisi koeficient zaporedja: ");
        scanf("%f",&g);
        while(a>0.1)
        {
                a=a*g;
        }
        printf("a = %f\n",a);
}
```

Zanka se v nekaterih primerih ne izvede!

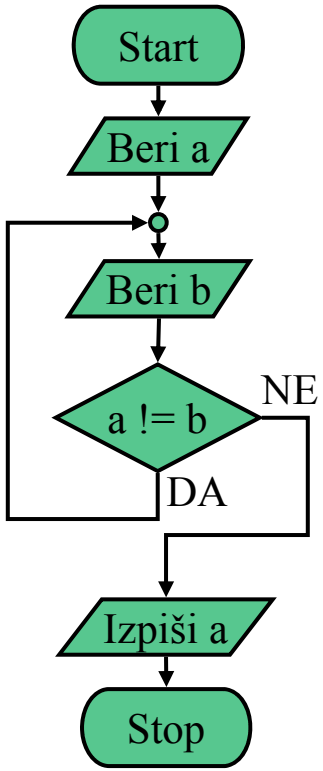
Zanka *while-do*

```
while(pogoj)                    /* ni podpičja! */
{
   operacija1;
}
```



Zanke – do while

- Primer – ugibanje števila:



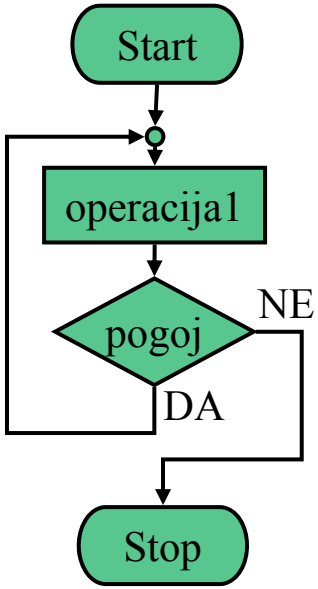
- Primer programa:

```
#include <stdio.h>
void main(void)
{
        int a,b;
        printf("\nVpisi stevilo: ");
        scanf("%d",&a);
        do
        {
                printf("\nUgibaj: ");
                scanf("%d",&b);
        }while(a!=b);
        printf("Uganil si, stevilo je = %d\n",a);
}
```

Zanka se vedno izvede vsaj enkrat!

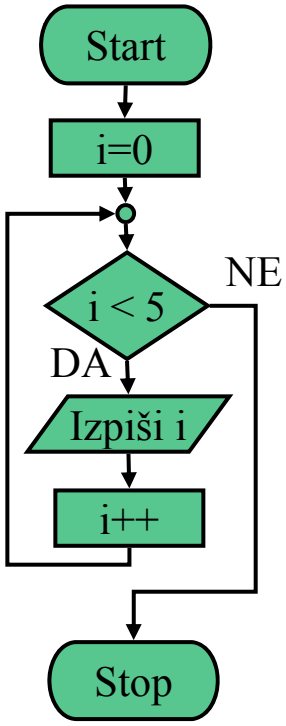
Zanka *do-while*

```
do                          /* ni podpičja! */
{
   operacija1;
} while(pogoj);
```



Zanke – for

- Brezpogojno ponavljanje
- Primer – pet krat izpiši vrednost spremenljivke:



- Primer programa:

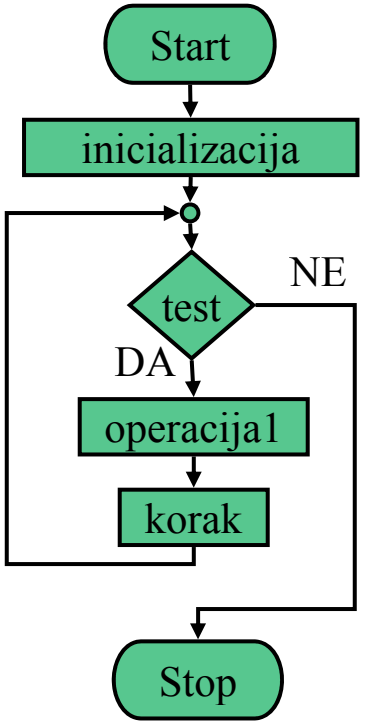
```
#include <stdio.h>

void main()
{
        int i;
        for(i=0;i<5;i++)
        {
                printf("i=%d\n",i);
        }
}
```

Spremenljivko *i* lahko spreminjamo tudi v telesu funkcije!

Zanka *for*

```
for(inicializacija;test;korak) /* ni podpičja! */
{
   operacija1;
}
```



Stavki break, continue in goto

- Stavek break
  - Povzroči izstop iz (najbolj notranje) zanke tipa for, while ali do..while. Uporabljamo ga tudi pri zaključku alternativ v odločitvenih stavkih switch.
- Stavek continue
  - Je podoben stavku break in ga uporabljamo v zankah (for, while, do..while). Za razliko od stavka break ne povzroči izstopa iz zanke ampak le preskok vseh stavkov (ki so za njim) v dani iteraciji.
- Stavek goto
  - Povzroči direkten prehod na stavek z dano etiketo.
  - Stavek goto ni dovoljen v strukturiranem programiranju!

Domača naloga

Možna vprašanja na izpitu

- Definicija in deklaracija – opiši razliko.
- Opiši definicijo spremenljivke!
- Katera števila lahko zajemajo celoštevilčne spremenljivke?
- Katera števila lahko zajemajo spremenljivke s plavajočo vejico?
- Kako so v jeziku C izvedene prireditve vrednosti?
- Razlika med == in =.
- Pomen operatorjev ++ in --.
- Kakšna je razlika med a++ in ++a?
- Sintaksa in opis stavka if.
- Sintaksa in opis stavka switch (case).
- Sintaksa in opis zanke do-while.
- Sintaksa in opis zanke while-do.
- Sintaksa in opis zanke for.
- Primer uporabe if, switch, while-do, do-while, for.