Bitne operacije: c = a^b

a	1	0	1	1	0	1	0	0
b	0	1	1	0	0	1	0	1
c								

Bitne operacije: c = a^b

a	1	0	1	1	0	1	0	0
b	0	1	1	0	0	1	0	1
c								**1**

0 XOR 1 = 1

Bitne operacije: c = a^b

a	1	0	1	1	0	1	0	0
b	0	1	1	0	0	1	0	1
c							**0**	**1**

0 XOR 0 = 0

Bitne operacije: c = a^b

a	1	0	1	1	0	1	0	0
b	0	1	1	0	0	1	0	1
c						**0**	**0**	**1**

1 XOR 1 = 0

Bitne operacije: c = a^b

a	1	0	1	1	0	1	0	0
b	0	1	1	0	0	1	0	1
c					**0**	**0**	**0**	**1**

0 XOR 0 = 0

Bitne operacije: c = a^b

a	1	0	1	1	0	1	0	0
b	0	1	1	0	0	1	0	1
c				**1**	**0**	**0**	**0**	**1**

1 XOR 0 = 1

Bitne operacije: c = a^b

a	1	0	1	1	0	1	0	0
b	0	1	1	0	0	1	0	1
c			**0**	**1**	**0**	**0**	**0**	**1**

1 XOR 1 = 0

Bitne operacije: c = a^b

a	1	0	1	1	0	1	0	0
b	0	1	1	0	0	1	0	1
c		**1**	**0**	**1**	**0**	**0**	**0**	**1**

0 XOR 1 = 1

Bitne operacije: c = a^b

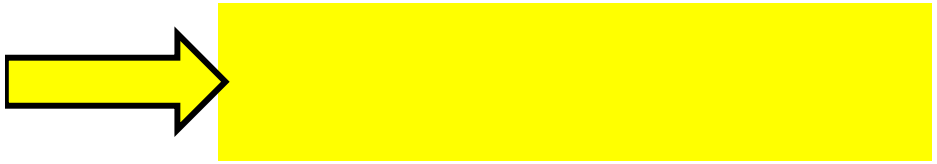
a	1	0	1	1	0	1	0	0
b	0	1	1	0	0	1	0	1
c	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**1**

1 XOR 0 = 1

Bitne operacije: c = a^b

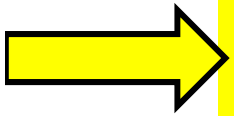
a	1	0	1	1	0	1	0	0
b	0	1	1	0	0	1	0	1
c	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**1**

Primer: Ukaz *break*



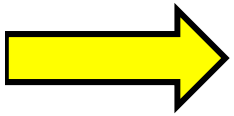
```
for(inicializacija;test;korak)
{
   operacija1;
   if(pogoj) break;
   operacija2;
}
```

Primer: Ukaz *break*



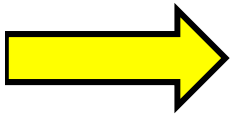
```
for(inicializacija;test;korak)
{
   operacija1;
   if(pogoj) break;
   operacija2;
}
```

Primer: Ukaz *break*



```
for(inicializacija;test;korak)
  {
    operacija1;
    if(pogoj) break;
    operacija2;
  }
```

Primer: Ukaz *break*



```
for(inicializacija;test;korak)
  {
    operacija1;
    if(pogoj) break;
    operacija2;
  }
```

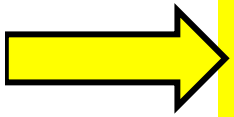
Primer: Ukaz *break*

```
for(inicializacija;test;korak)
{
    operacija1;
    if(pogoj) break;
    operacija2;
}
```

test = TRUE

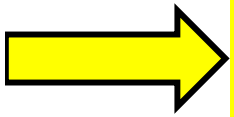
Pogoj za test je izpolnjen

Primer: Ukaz *break*



```
for(inicializacija;test;korak)
{
   operacija1;
   if(pogoj) break;
   operacija2;
}
```

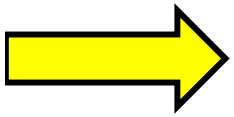
Primer: Ukaz *break*



```
for(inicializacija;test;korak)
{
   operacija1;
   if(pogoj) break;
   operacija2;
}
```

Primer: Ukaz *break*

```
for(inicializacija;test;korak)
{
   operacija1;
   if(pogoj) break;
   operacija2;
}
```

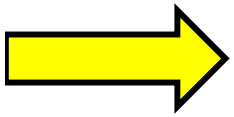


pogoj = TRUE

Pogoj je izpolnjen

Primer: Ukaz *break*

```
for(inicializacija;test;korak)
{
   operacija1;
   if(pogoj) break;
   operacija2;
}
```

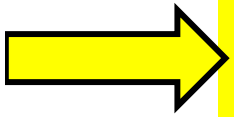


pogoj = TRUE

Pogoj je izpolnjen

Primer: Ukaz *break*

```
for(inicializacija;test;korak)
{
   operacija1;
   if(pogoj) break;
   operacija2;
}
```



Ukaz *break* povzroči izstop iz zanke.

Bitne operacije: Brisanje bita

Program:

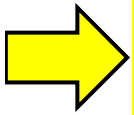
```
a &= ~(1<<b);
```

Razlaga:

Brisanje bita št. b (na primeru a=53, b=2)

Bitne operacije: Brisanje bita

Program:



```
a &= ~(1<<b);
```

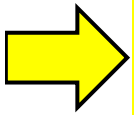
Spremenljivke:

a = 53

b = 2

Bitne operacije: Brisanje bita

Program:



```
a &= ~(1<<b);
```

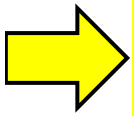
Spremenljivke:

a	0	0	1	1	0	1	0	1

b = 2

Bitne operacije: Brisanje bita

Program:



```
a &= ~(1<<b);
```

Spremenljivke:

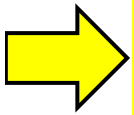
a	0	0	1	1	0	0	0	1

b = 2

Bitne operacije: Brisanje bita

Program:

```
a &= ~(1<<b);
```



Spremenljivke:

a	0	0	1	1	0	0	0	1

b = 2

Bitne operacije: Brisanje bita

Opis po korakih s programom:

```
c = 1<<b;

d = ~c;

e = a&d;

a = e;
```

Bitne operacije: Brisanje bita

Program:

```
c = 1<<b;
d = ~c;
e = a&d;
a = e;
```

Spremenljivke:

a = 53

b = 2

c = 0

d = 0

e = 0

Bitne operacije: Brisanje bita

Program:

```
c = 1<<b;
d = ~c;
e = a&d;
a = e;
```

Spremenljivke:

a	0	0	1	1	0	1	0	1
c	0	0	0	0	0	0	0	0
d	0	0	0	0	0	0	0	0
e	0	0	0	0	0	0	0	0

b = 2

Bitne operacije: Brisanje bita

Program:

```
c = 1<<b;
d = ~c;
e = a&d;
a = e;
```

Spremenljivke:

a	0	0	1	1	0	1	0	1
c	0	0	0	0	0	1	0	0
d	0	0	0	0	0	0	0	0
e	0	0	0	0	0	0	0	0

b = 2

Bitne operacije: Brisanje bita

Program:

```
c = 1<<b;
d = ~c;
e = a&d;
a = e;
```

Spremenljivke:

a	0	0	1	1	0	1	0	1
c	0	0	0	0	0	1	0	0
d	1	1	1	1	1	0	1	1
e	0	0	0	0	0	0	0	0

b = 2

Bitne operacije: Brisanje bita

Program:

```
c = 1<<b;
d = ~c;
e = a&d;
a = e;
```

Spremenljivke:

a	0	0	1	1	0	1	0	1
c	0	0	0	0	0	1	0	0
d	1	1	1	1	1	0	1	1
e	0	0	1	1	0	0	0	1

b = 2

Bitne operacije: Brisanje bita

Program:

```
c = 1<<b;
d = ~c;
e = a&d;
a = e;
```

Spremenljivke:

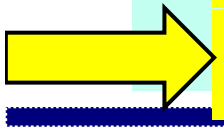
a	0	0	1	1	0	0	0	1
c	0	0	0	0	0	1	0	0
d	1	1	1	1	1	0	1	1
e	0	0	1	1	0	0	0	1

b = 2

Bitne operacije: Brisanje bita

Program:

```
c = 1<<b;
d = ~c;
e = a&d;
a = e;
```



Spremenljivke:

a	0	0	1	1	0	0	0	1
c	0	0	0	0	0	1	0	0
d	1	1	1	1	1	0	1	1
e	0	0	1	1	0	0	0	1

b = 2

Bitne operacije: Brisanje bita

Program:

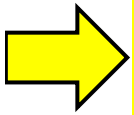
```
a &= ~(1<<b);
```

Razlaga:

Brisanje bita št. 3 (na primeru a=53, b=3)

Bitne operacije: Brisanje bita

Program:



```
a &= ~(1<<b);
```

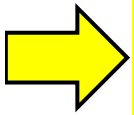
Spremenljivke:

a = 53

b = 2

Bitne operacije: Brisanje bita

Program:



```
a &= ~(1<<b);
```

Spremenljivke:

a	0	0	1	1	0	1	0	1

b = 3

Bit 3 je že na 0!

Bitne operacije: Brisanje bita

Program:

```
a &= ~(1<<b);
```

Spremenljivke:

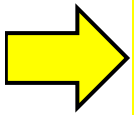
a	0	0	1	1	0	1	0	1

b = 3

Bitne operacije: Brisanje bita

Program:

```
a &= ~(1<<b);
```



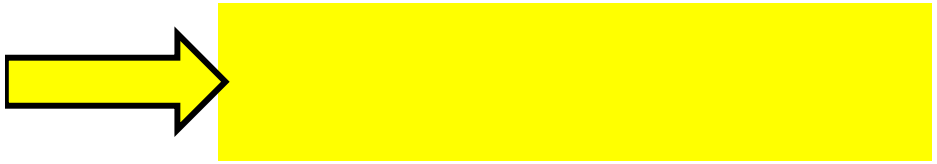
Spremenljivke:

a	0	0	1	1	0	1	0	1

b = 3

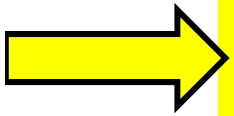
Bit 3 se ni spremenil!

Primer: Ukaz *continue*



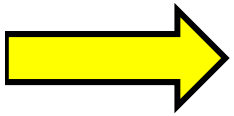
```
for(inicializacija;test;korak)
{
   operacija1;
   if(pogoj) continue;
   operacija2;
}
```

Primer: Ukaz *continue*



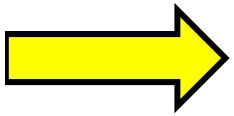
```
for(inicializacija;test;korak)
{
   operacija1;
   if(pogoj) continue;
   operacija2;
}
```

Primer: Ukaz *continue*



```
for(inicializacija;test;korak)
    {
       operacija1;
       if(pogoj) continue;
       operacija2;
    }
```

Primer: Ukaz *continue*



```
for(inicializacija;test;korak)
  {
    operacija1;
    if(pogoj) continue;
    operacija2;
  }
```

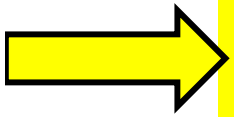
Primer: Ukaz *continue*

```
for(inicializacija;test;korak)
{
    operacija1;
    if(pogoj) continue;
    operacija2;
}
```

test = TRUE

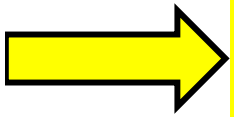
Pogoj za test je izpolnjen

Primer: Ukaz *continue*



```
for(inicializacija;test;korak)
{
   operacija1;
   if(pogoj) continue;
   operacija2;
}
```

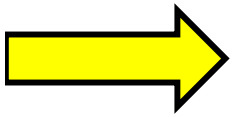
Primer: Ukaz *continue*



```
for(inicializacija;test;korak)
{
   operacija1;
   if(pogoj) continue;
   operacija2;
}
```

Primer: Ukaz *continue*

```
for(inicializacija;test;korak)
{
   operacija1;
   if(pogoj) continue;
   operacija2;
}
```

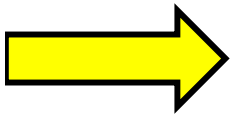


pogoj = TRUE

Pogoj je izpolnjen

Primer: Ukaz *continue*

```
for(inicializacija;test;korak)
{
   operacija1;
   if(pogoj) continue;
   operacija2;
}
```



pogoj = TRUE

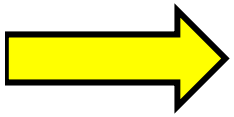
Pogoj je izpolnjen

Primer: Ukaz *continue*

```
for(inicializacija;test;korak)
  {
    operacija1;
    if(pogoj) continue;
    operacija2;
  }
```

Ukaz *continue* preskoči vse ukaze do konca zanke.

Primer: Ukaz *continue*

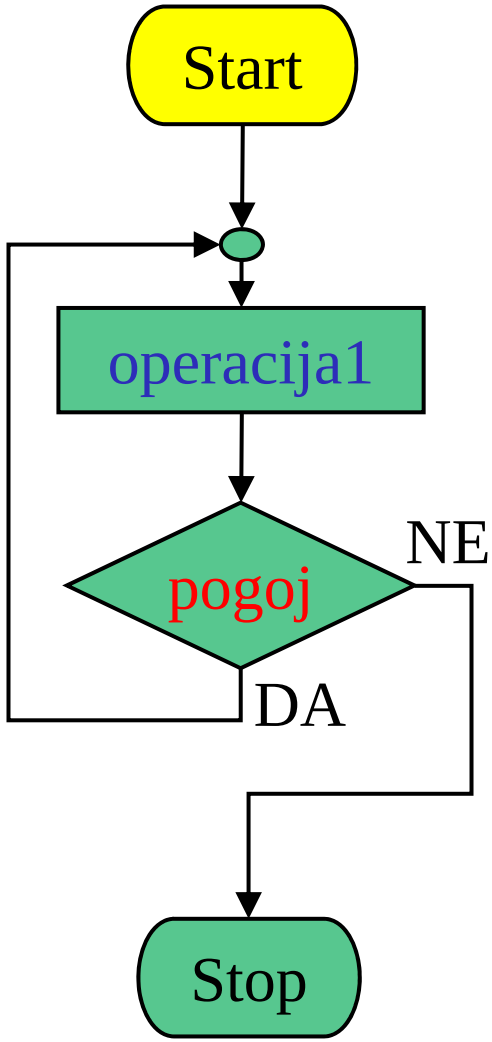


```
for(inicializacija;test;korak)
  {
    operacija1;
    if(pogoj) continue;
    operacija2;
  }
```

Ukaz *continue* povzroči skok na začetek zanke.

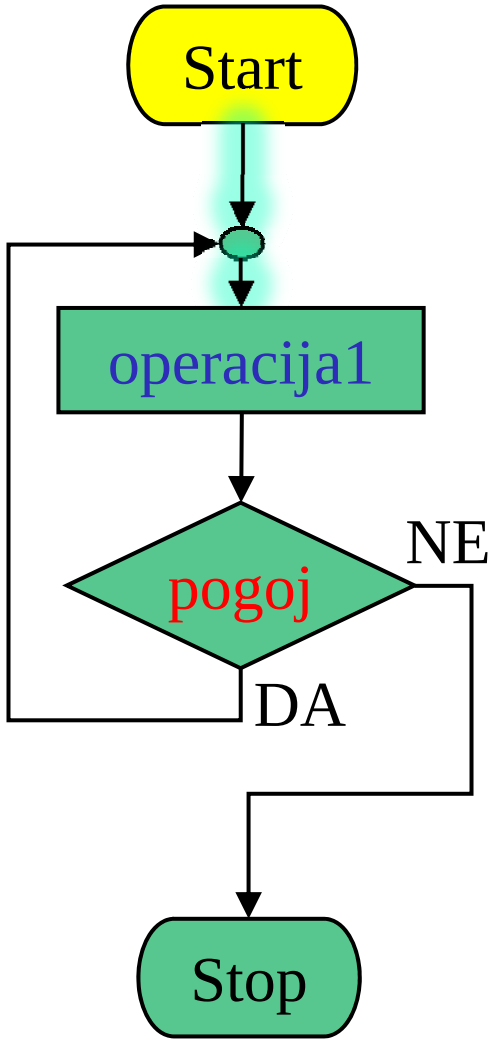
Primer: Zanka *do-while*

```
do
{
   operacija1;
} while(pogoj);
```



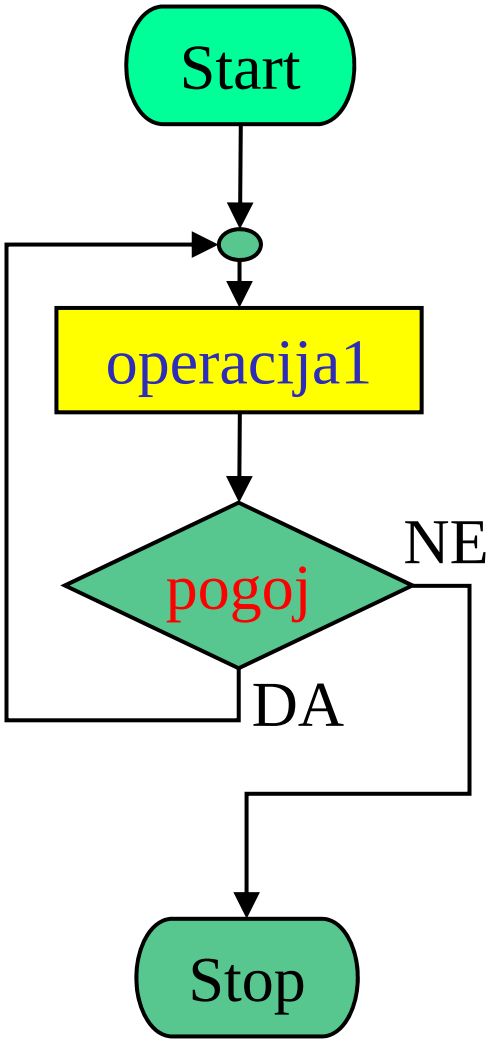
Primer: Zanka *do-while*

```
do
{
   operacija1;
} while(pogoj);
```



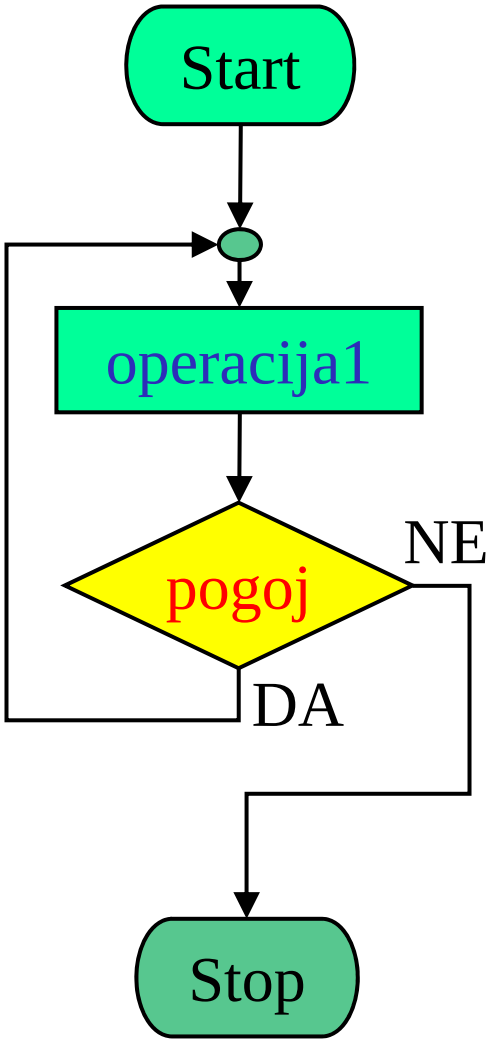
Primer: Zanka *do-while*

```
do
{
   operacija1;
} while(pogoj);
```



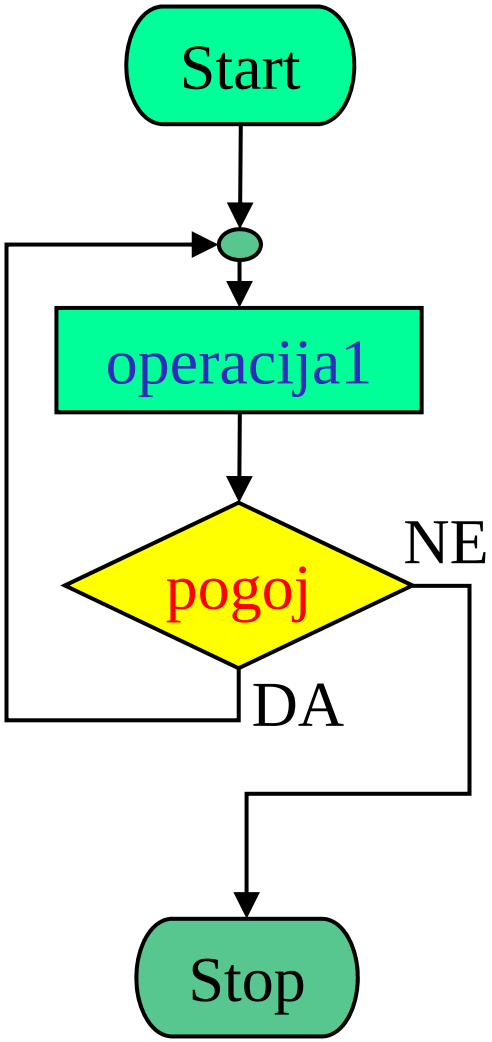
Primer: Zanka *do-while*

```
do
{
   operacija1;
} while(pogoj);
```



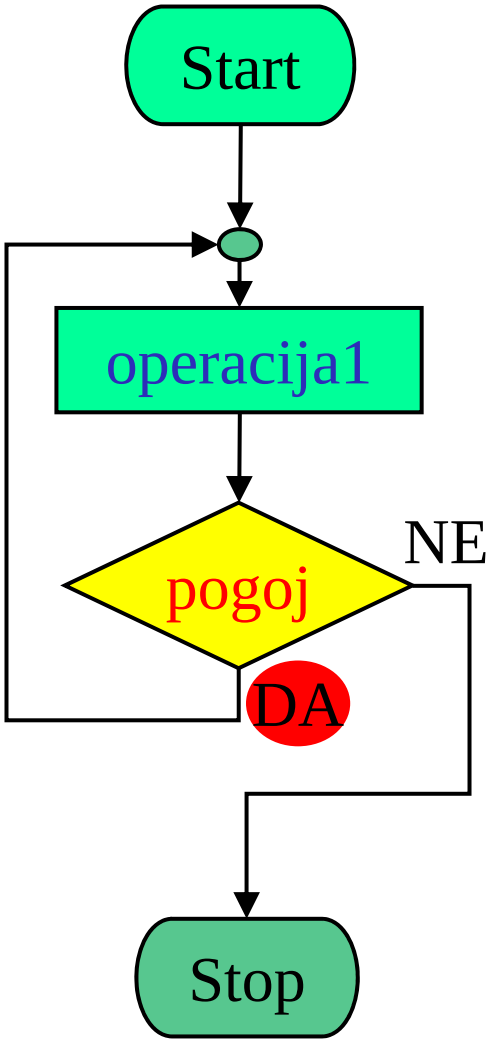
Primer: Zanka *do-while*

```
do
{
   operacija1;
} while(pogoj);
```



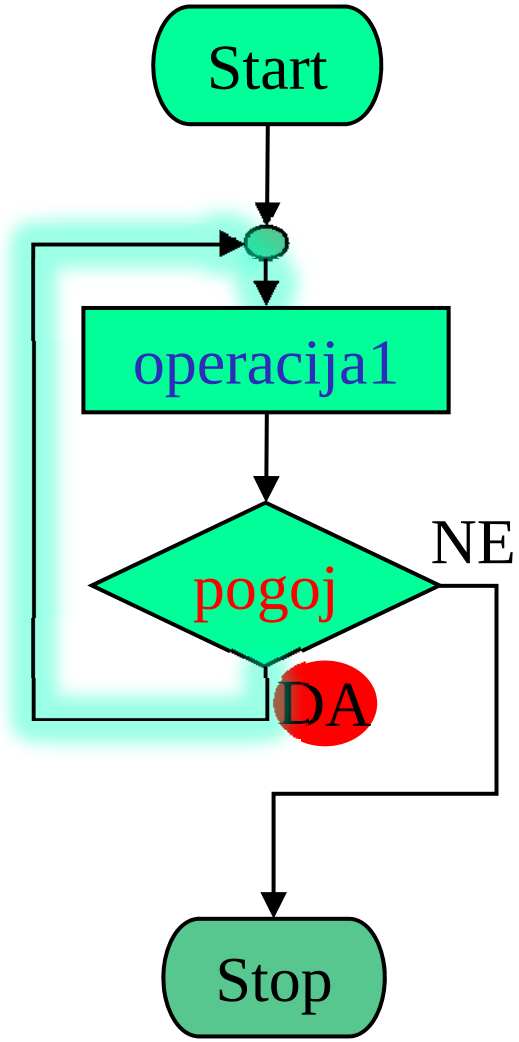
Primer: Zanka *do-while*

```
do
{
   operacija1;
} while(pogoj);
```



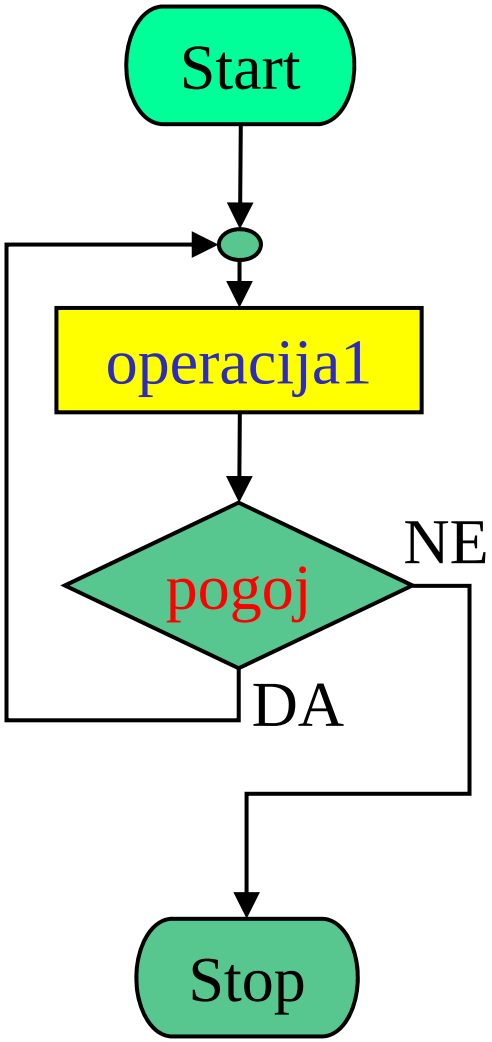
Primer: Zanka *do-while*

```
do
{
   operacija1;
} while(pogoj);
```



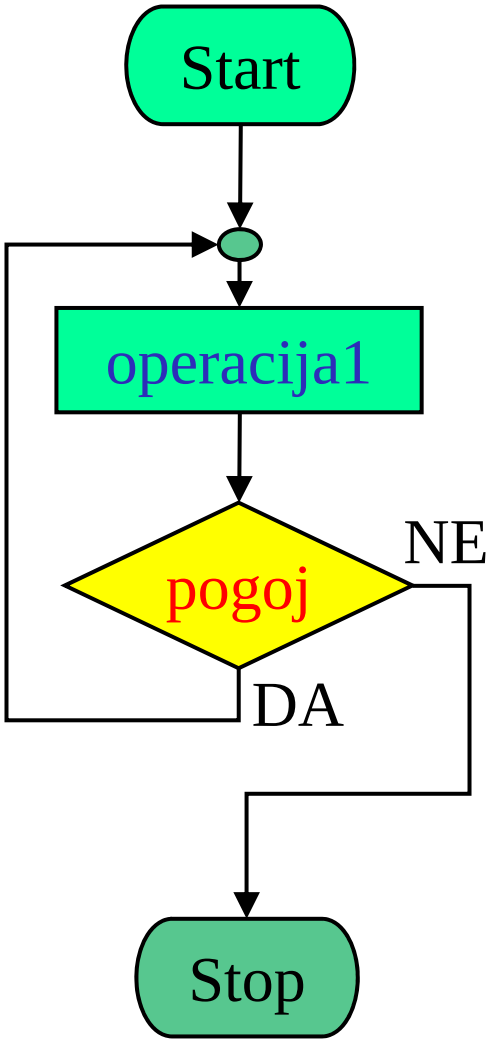
Primer: Zanka *do-while*

```
do
{
   operacija1;
} while(pogoj);
```



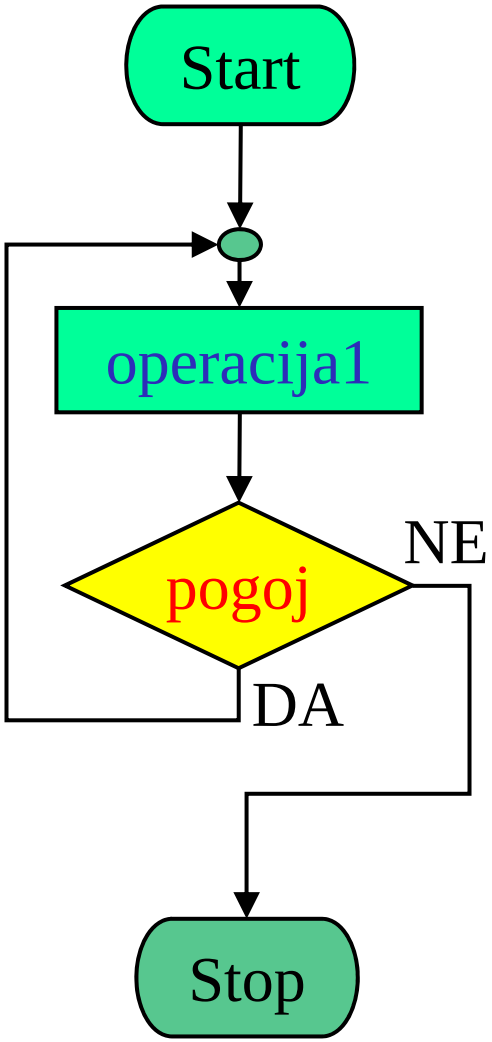
Primer: Zanka *do-while*

```
do
{
   operacija1;
} while(pogoj);
```



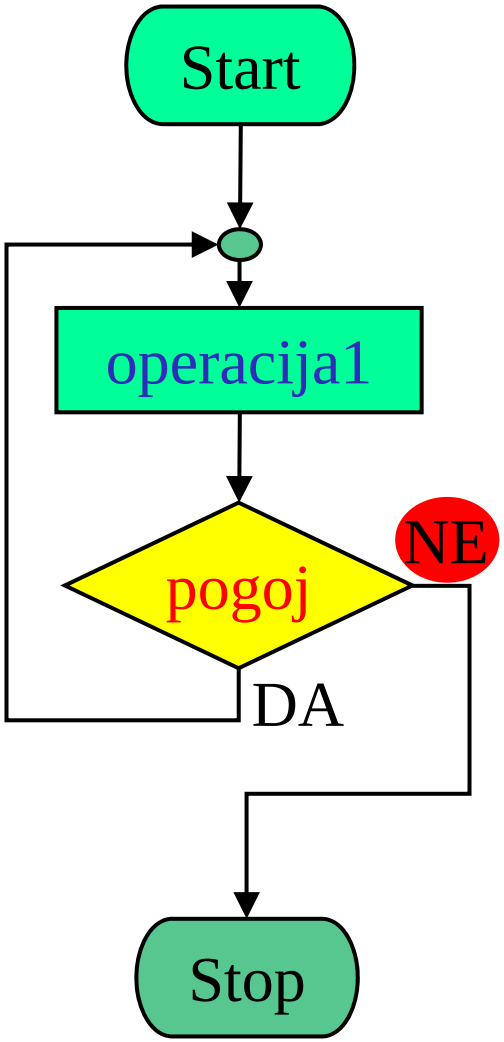
Primer: Zanka *do-while*

```
do
{
   operacija1;
} while(pogoj);
```



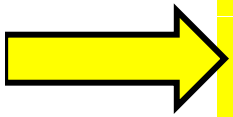
Primer: Zanka *do-while*

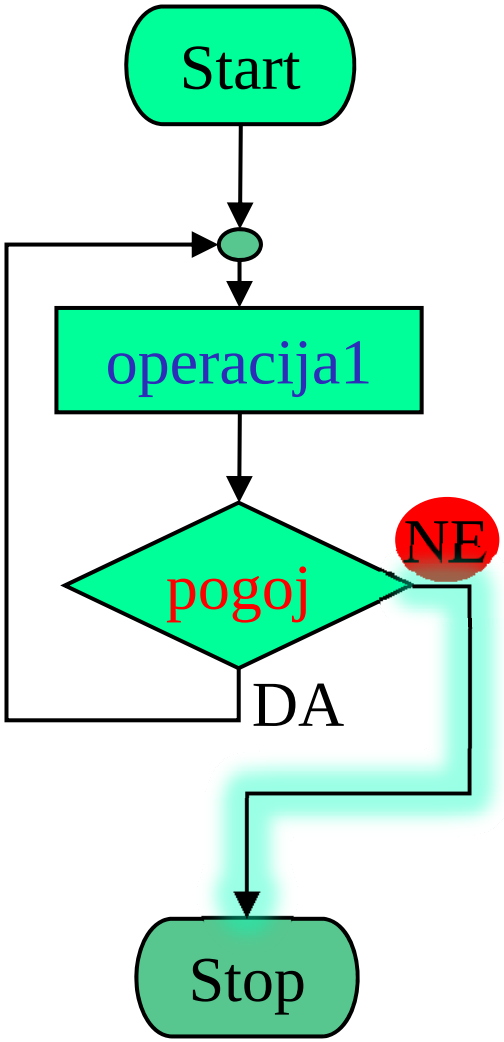
```
do
{
   operacija1;
} while(pogoj);
```



Primer: Zanka *do-while*

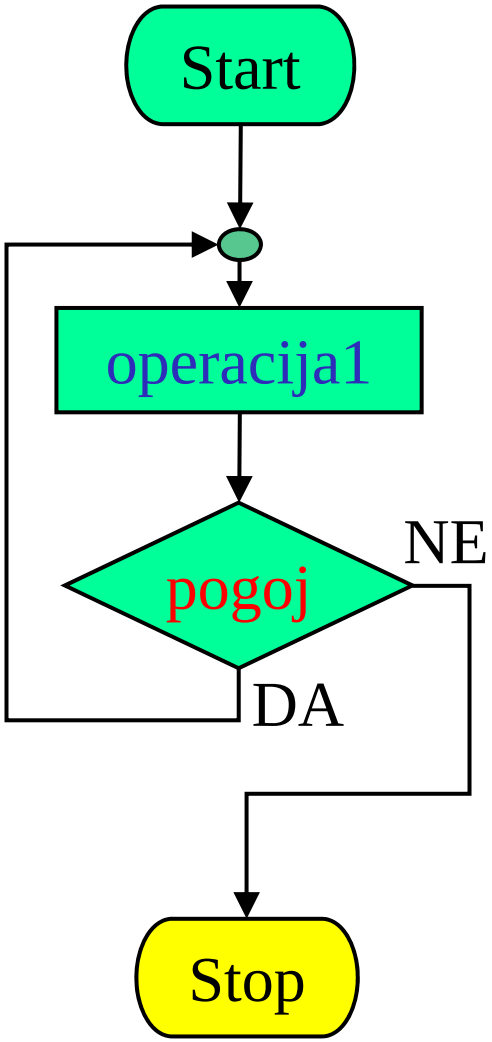
```
do
{
   operacija1;
} while(pogoj);
```





Primer: Zanka *do-while*

```
do
{
   operacija1;
} while(pogoj);
```

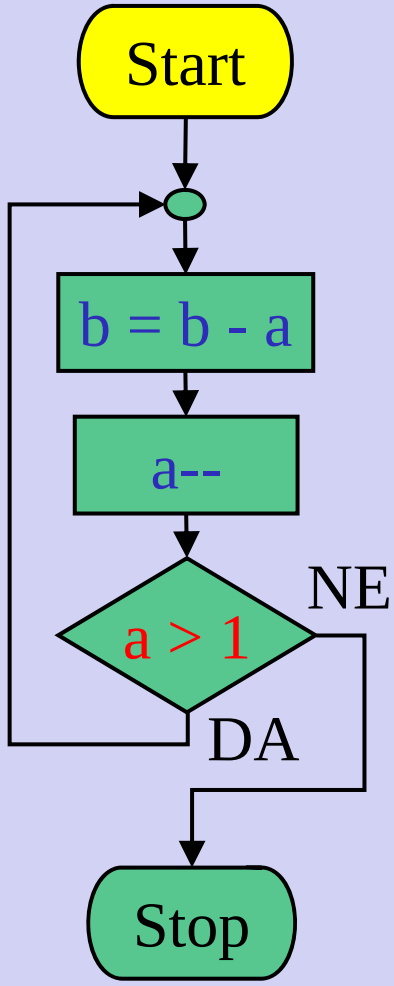


Praktični primer: Zanka *do-while*

Program:

```
do
{
   b -= a;
   a--;
} while(a > 1);
```

Diagram poteka:



Spremenljivki:

a = 3;

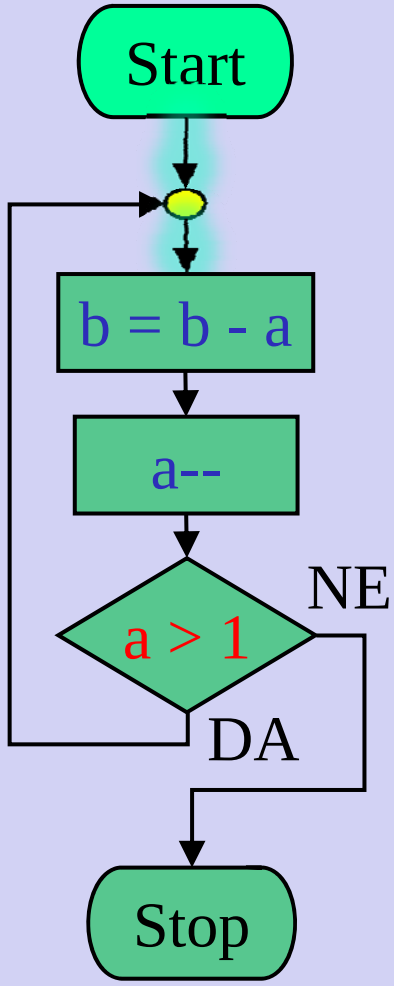
b = 10;

Praktični primer: Zanka *do-while*

Program:

```
do
{
   b -= a;
   a--;
} while(a > 1);
```

Diagram poteka:



Spremenljivki:

a = 3;

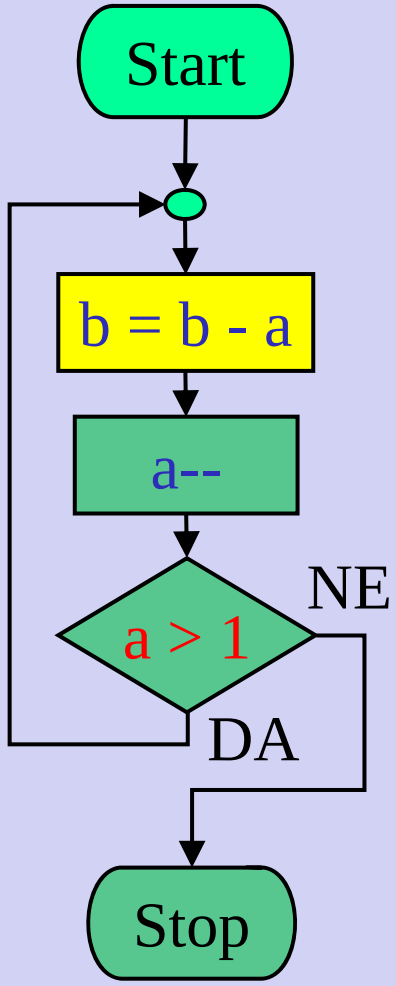
b = 10;

Praktični primer: Zanka *do-while*

Program:

```
do
{
   b -= a;
   a--;
} while(a > 1);
```

Diagram poteka:



Spremenljivki:

a = 3;

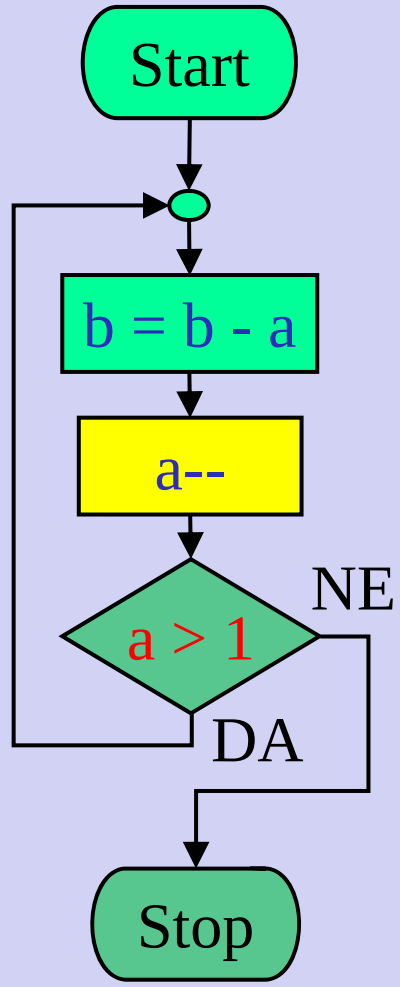
b = 7;

Praktični primer: Zanka *do-while*

Program:

```
do
{
   b -= a;
   a--;
} while(a > 1);
```

Diagram poteka:



Spremenljivki:

a = 2;

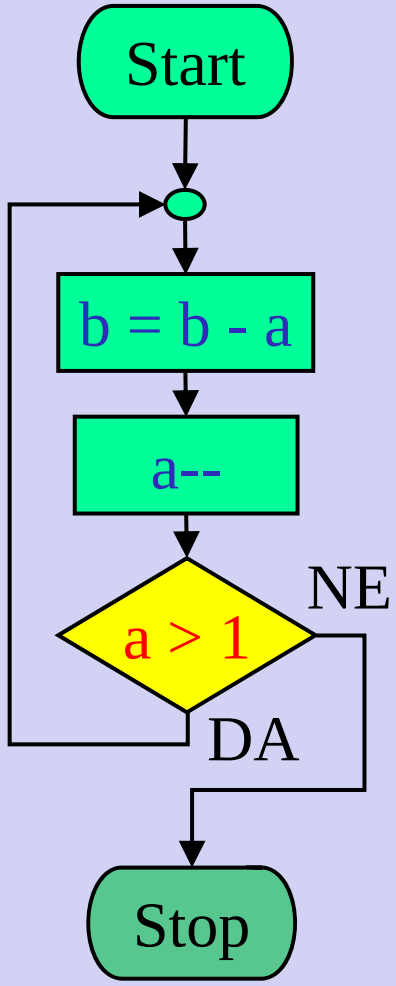
b = 7;

Praktični primer: Zanka *do-while*

Program:

```
do
{
    b -= a;
    a--;
} while(a > 1);
```

Diagram poteka:



Spremenljivki:

a = 2;

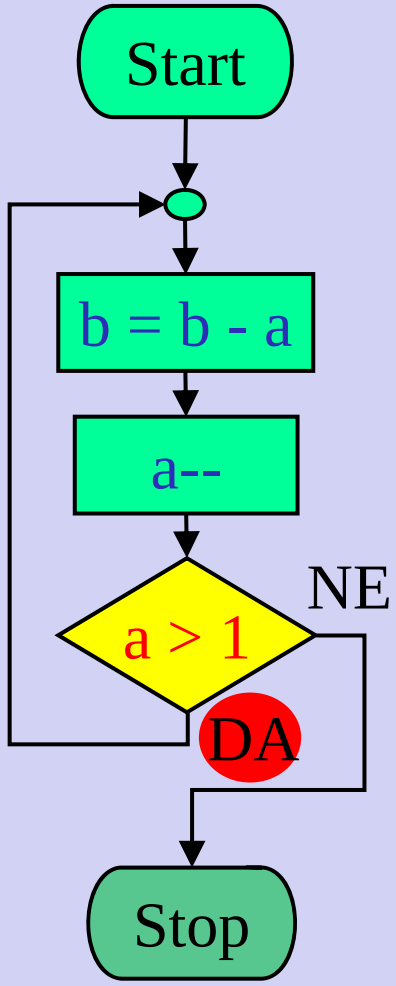
b = 7;

Praktični primer: Zanka *do-while*

Program:

```
do
{
   b -= a;
   a--;
} while(a > 1);
```

Diagram poteka:



Spremenljivki:

a = 2;

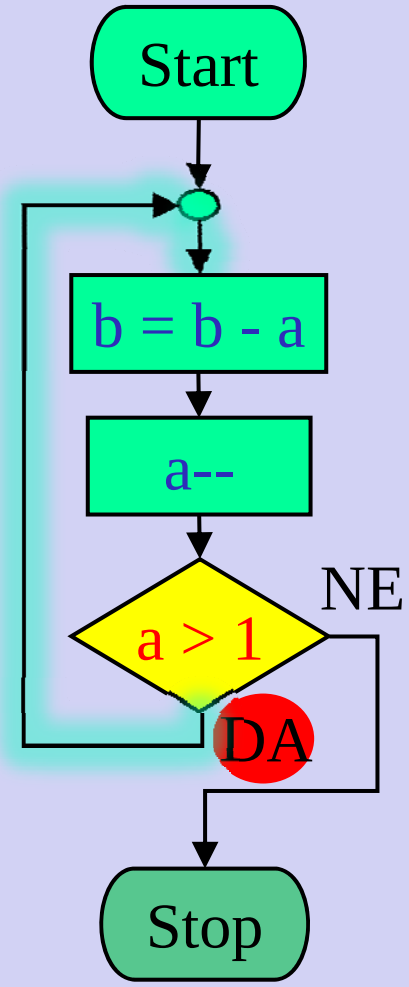
b = 7;

Praktični primer: Zanka *do-while*

Program:

```
do
{
   b -= a;
   a--;
} while(a > 1);
```

Diagram poteka:



Spremenljivki:

a = 2;

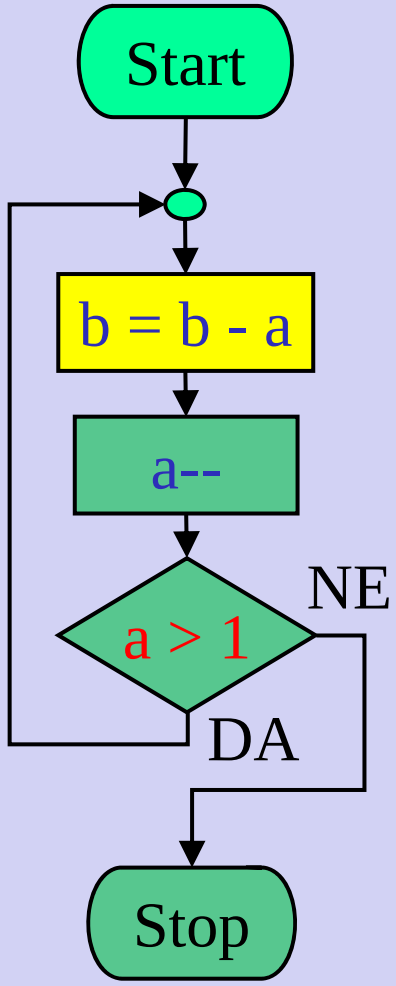
b = 7;

Praktični primer: Zanka *do-while*

Program:

```
do
{
   b -= a;
   a--;
} while(a > 1);
```

Diagram poteka:



Spremenljivki:

a = 2;

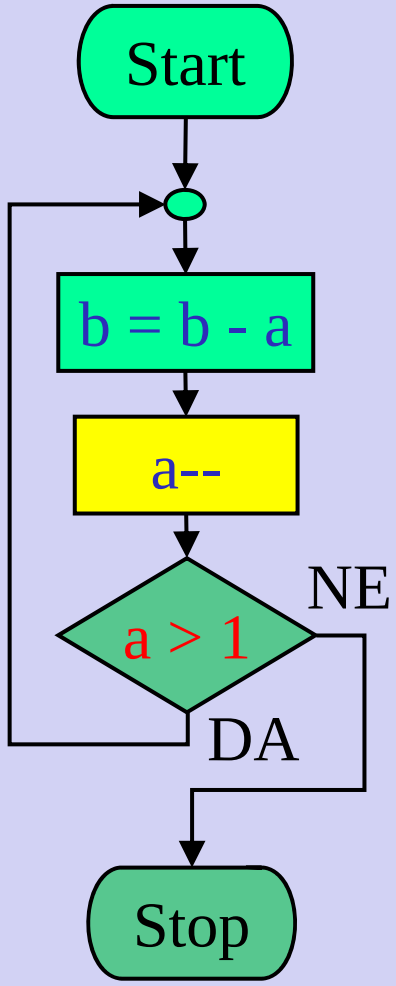
b = 5;

Praktični primer: Zanka *do-while*

Program:

```
do
{
   b -= a;
   a--;
} while(a > 1);
```

Diagram poteka:



Spremenljivki:

a = 1;

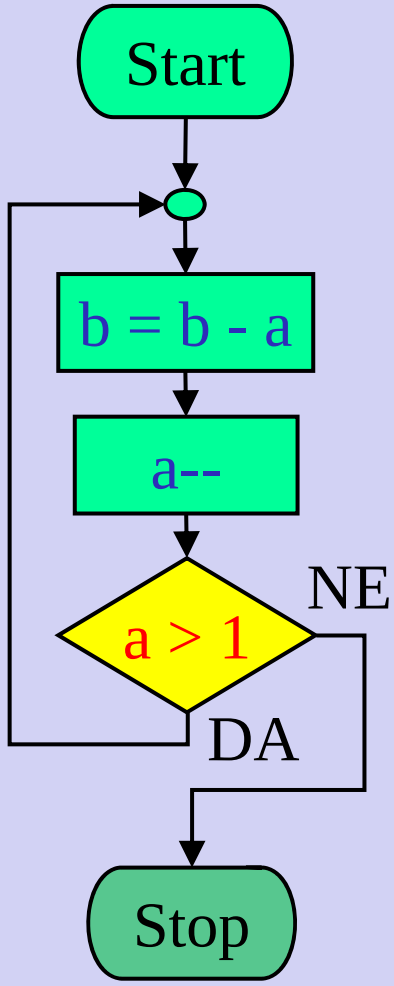
b = 5;

Praktični primer: Zanka *do-while*

Program:

```
do
{
    b -= a;
    a--;
} while(a > 1);
```

Diagram poteka:



Spremenljivki:

a = 1;

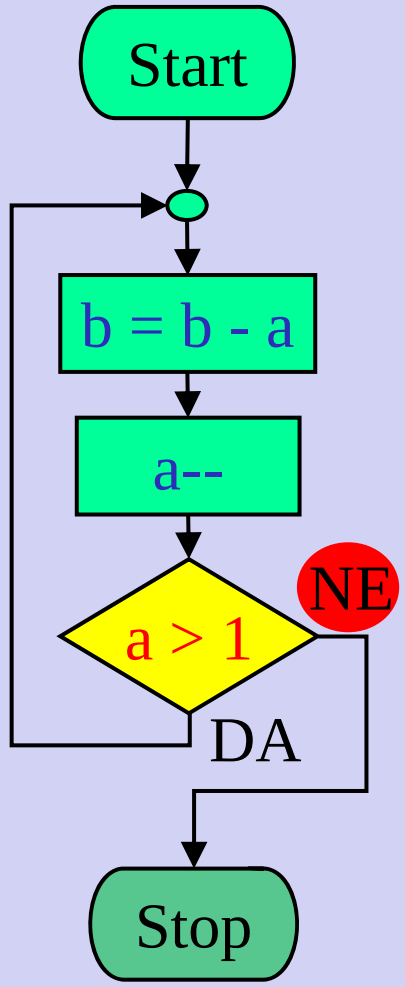
b = 5;

Praktični primer: Zanka *do-while*

Program:

```
do
{
   b -= a;
   a--;
} while(a > 1);
```

Diagram poteka:



Spremenljivki:

a = 1;

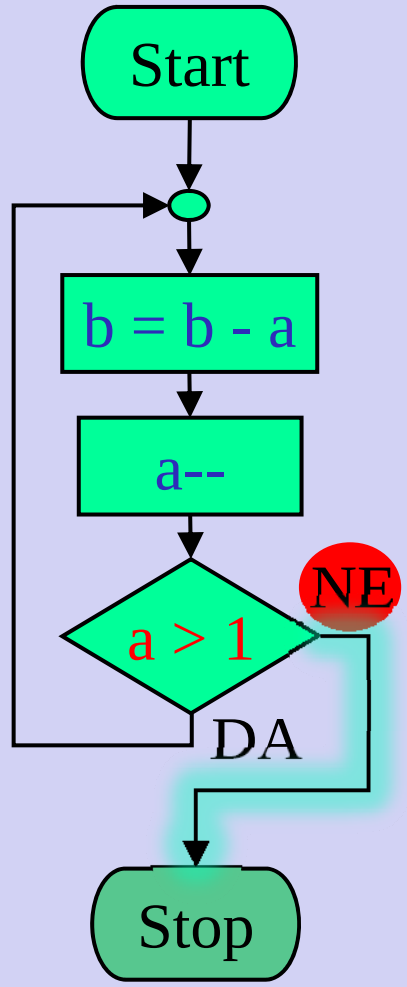
b = 5;

Praktični primer: Zanka *do-while*

Program:

```
do
{
   b -= a;
   a--;
} while(a > 1);
```

Diagram poteka:



Spremenljivki:

a = 1;

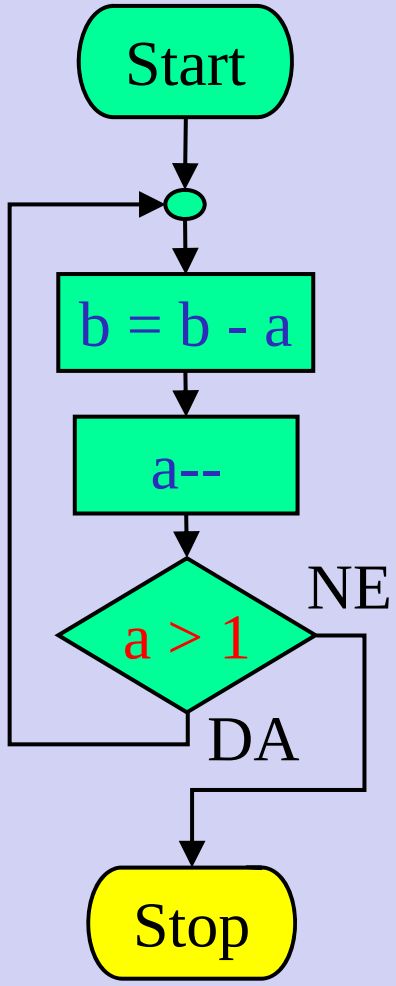
b = 5;

Praktični primer: Zanka *do-while*

Program:

```
do
{
   b -= a;
   a--;
} while(a > 1);
```

Diagram poteka:



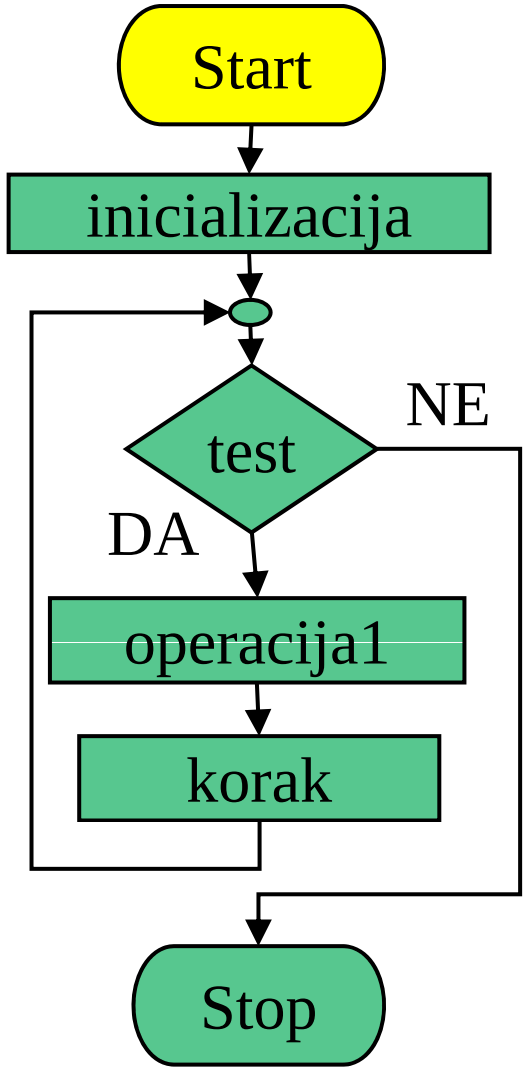
Spremenljivki:

a = 1;

b = 5;

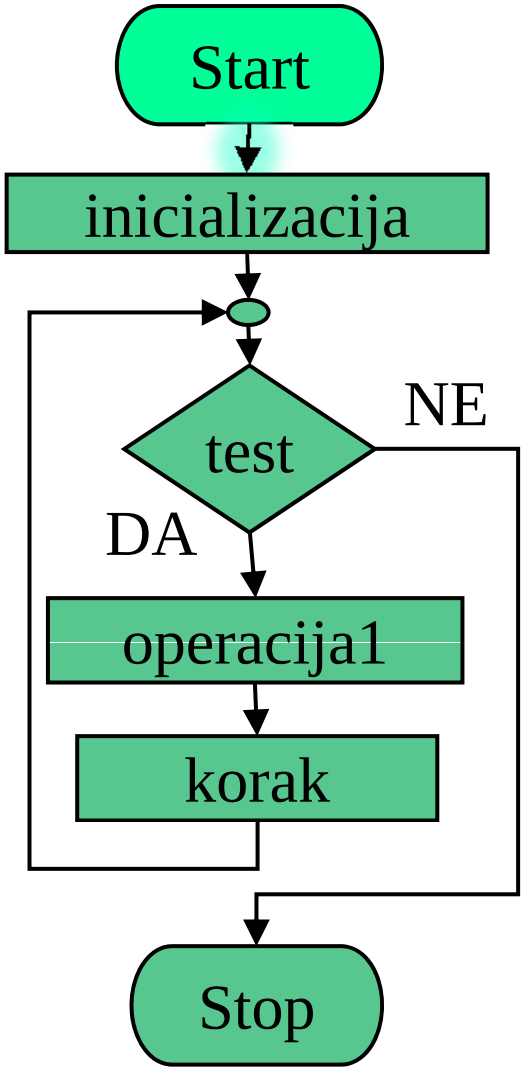
Primer: Zanka *for*

```
for(inicializacija;test;korak)
{
   operacija1;
}
```



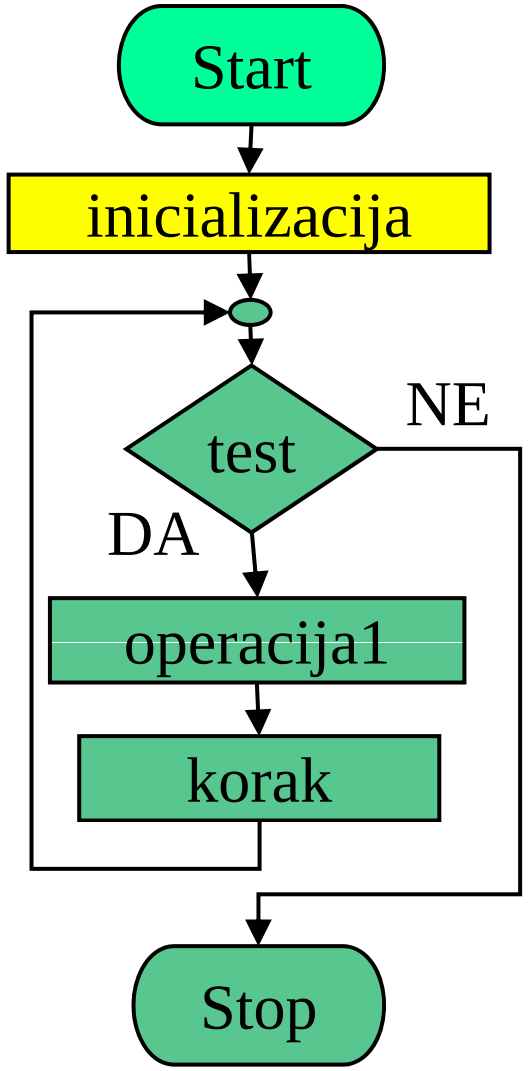
Primer: Zanka *for*

```
for(inicializacija;test;korak)
{
   operacija1;
}
```



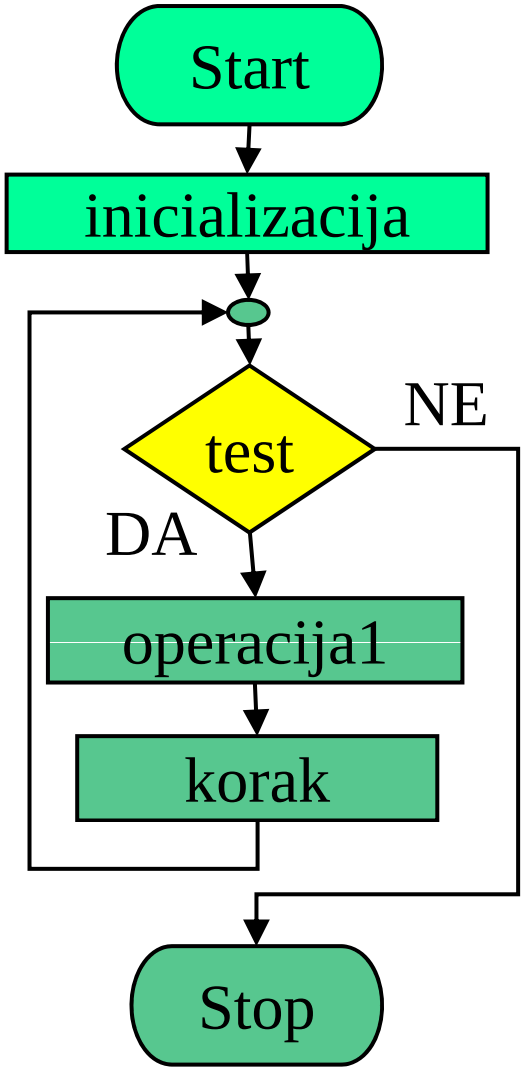
Primer: Zanka *for*

```
for(inicializacija;test;korak)
  {
    operacija1;
  }
```



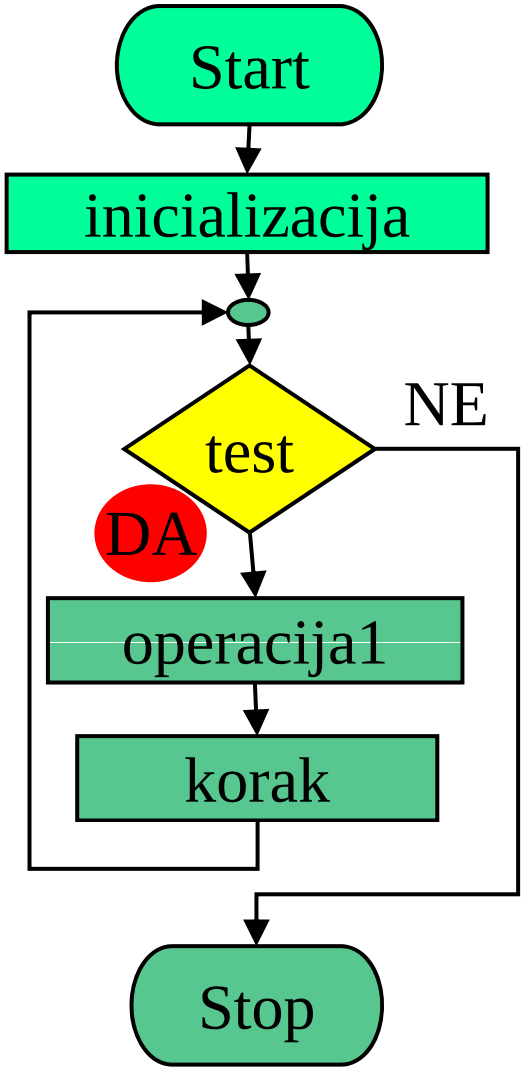
Primer: Zanka *for*

```
for(inicializacija;test;korak)
  {
    operacija1;
  }
```



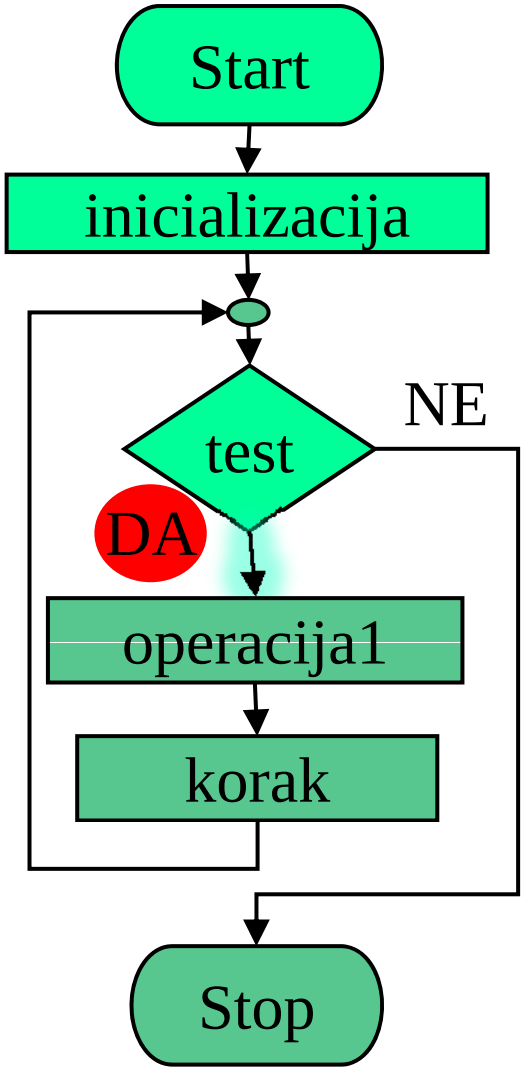
Primer: Zanka *for*

```
for(inicializacija;test;korak)
  {
    operacija1;
  }
```



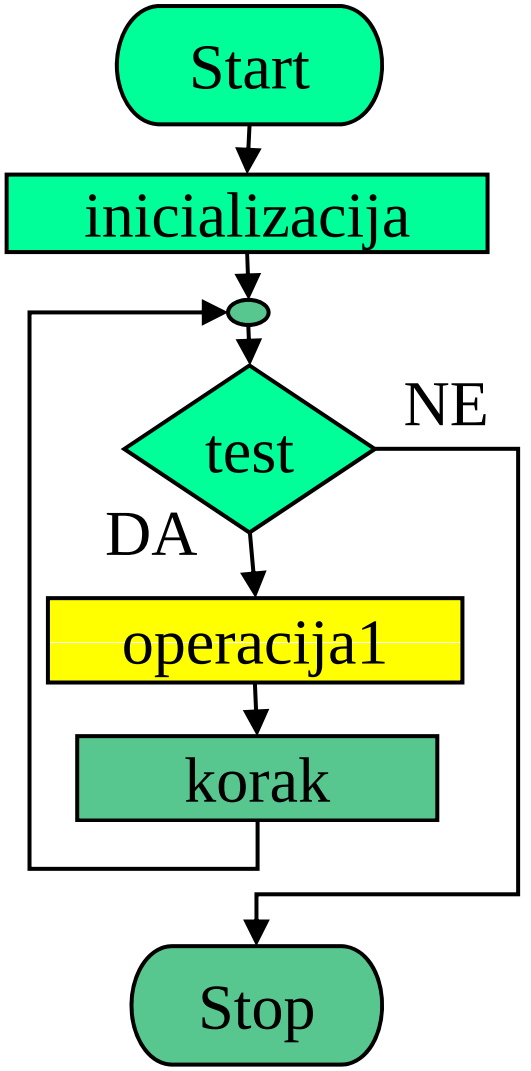
Primer: Zanka *for*

```
for(inicializacija;test;korak)
  {
    operacija1;
  }
```



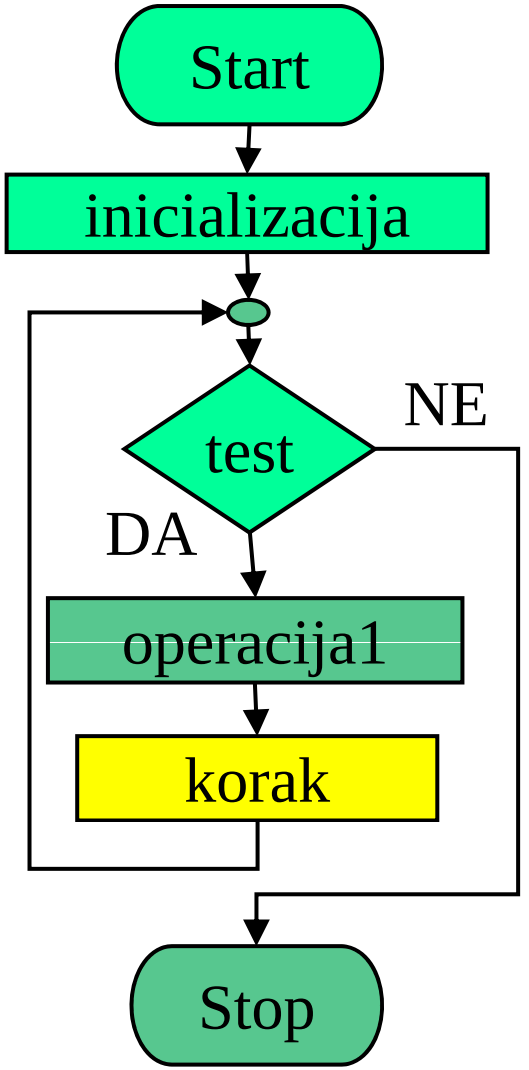
Primer: Zanka *for*

```
for(inicializacija;test;korak)
{
    operacija1;
}
```



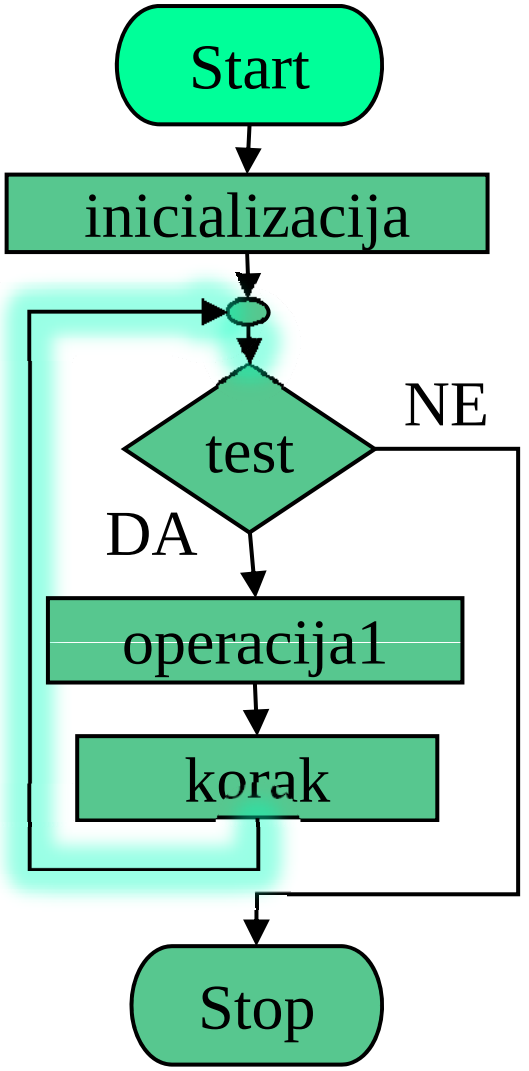
Primer: Zanka *for*

```
for(inicializacija;test;korak)
  {
    operacija1;
  }
```



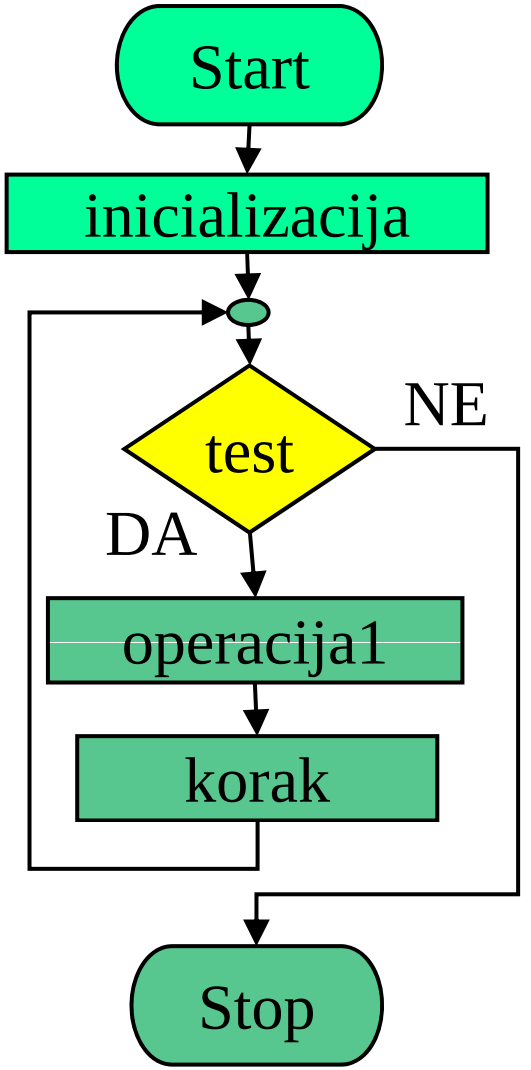
Primer: Zanka *for*

```
for(inicializacija;test;korak)
{
    operacija1;
}
```



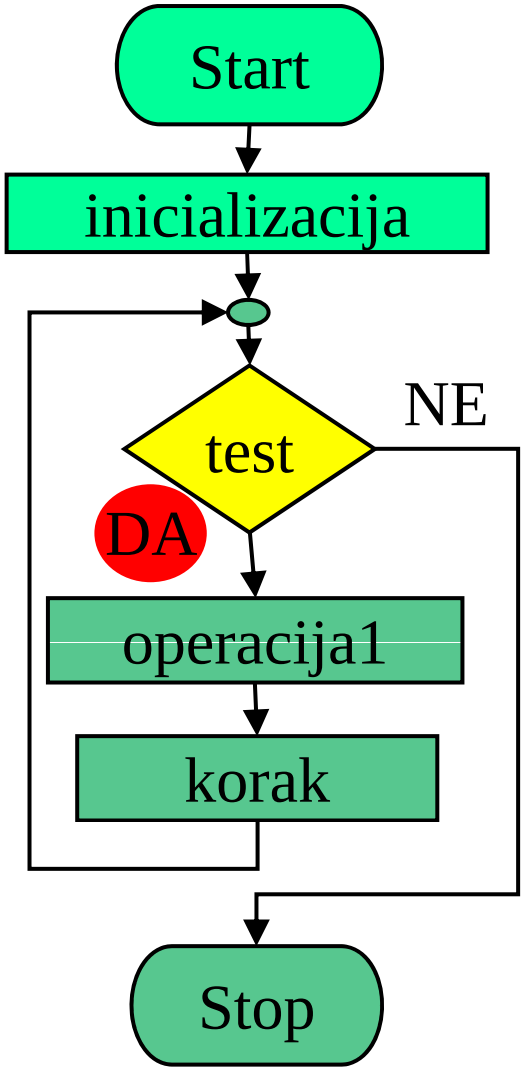
Primer: Zanka *for*

```
for(inicializacija;test;korak)
  {
    operacija1;
  }
```



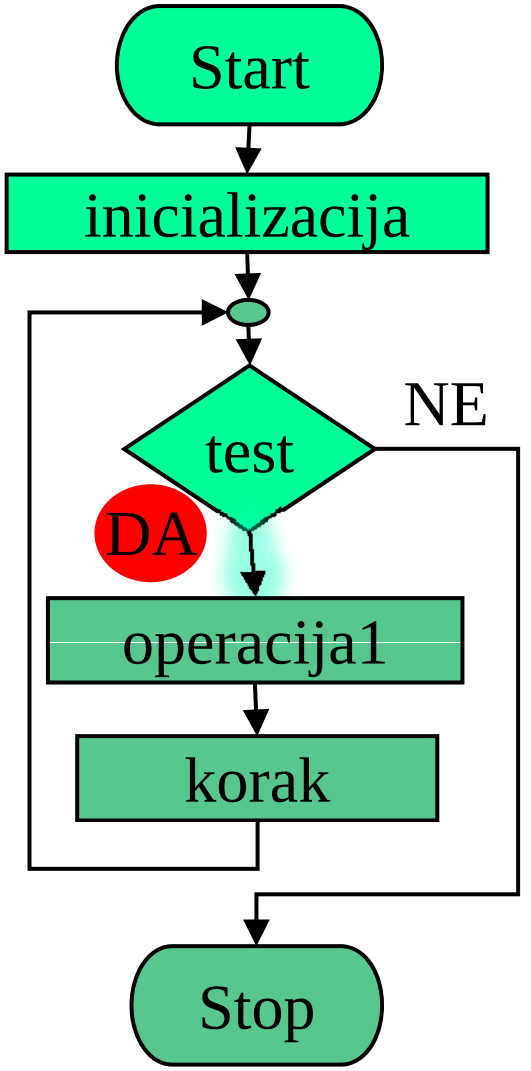
Primer: Zanka *for*

```
for(inicializacija;test;korak)
  {
    operacija1;
  }
```



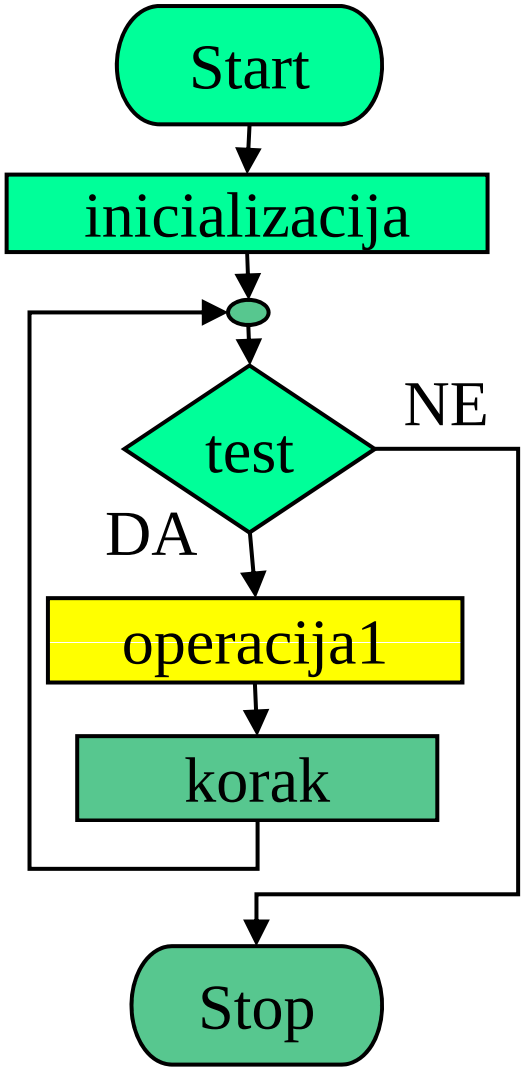
Primer: Zanka *for*

```
for(inicializacija;test;korak)
  {
    operacija1;
  }
```



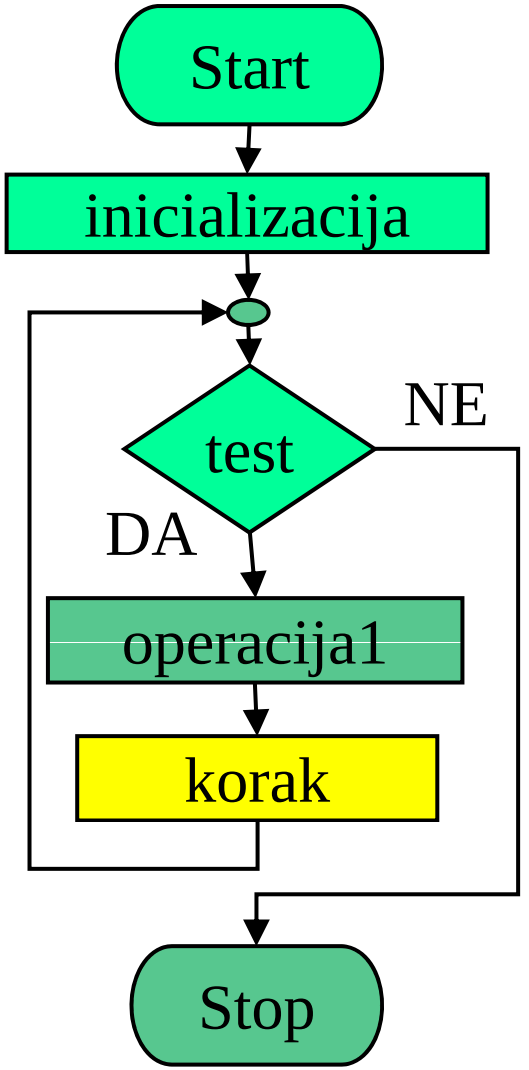
Primer: Zanka *for*

```
for(inicializacija;test;korak)
{
    operacija1;
}
```



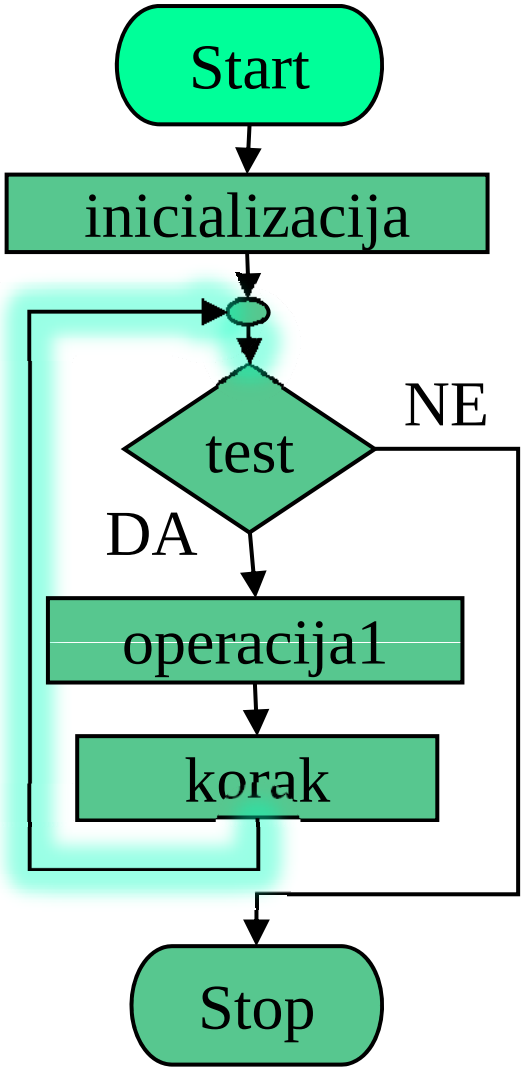
Primer: Zanka *for*

```
for(inicializacija;test;korak)
  {
    operacija1;
  }
```



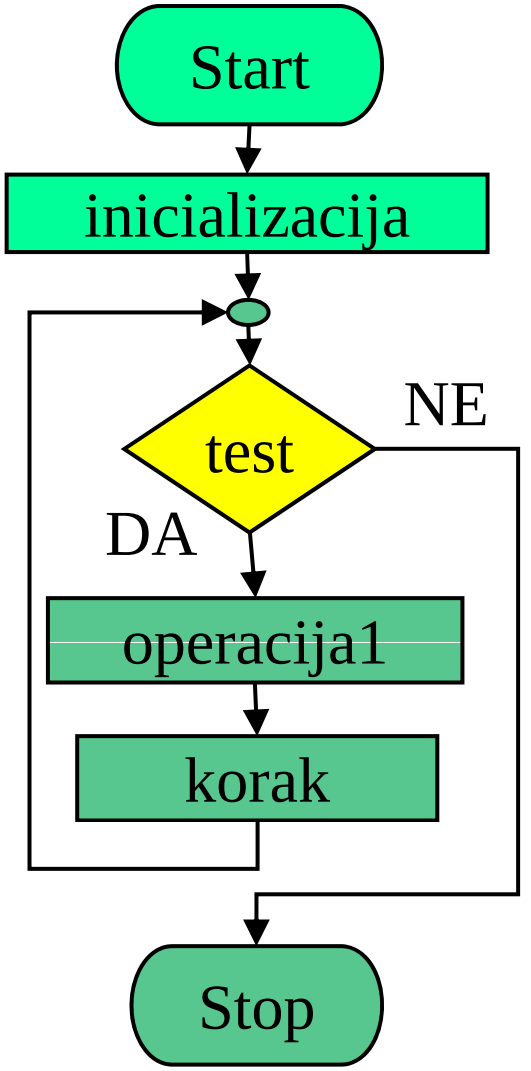
Primer: Zanka *for*

```
for(inicializacija;test;korak)
{
    operacija1;
}
```



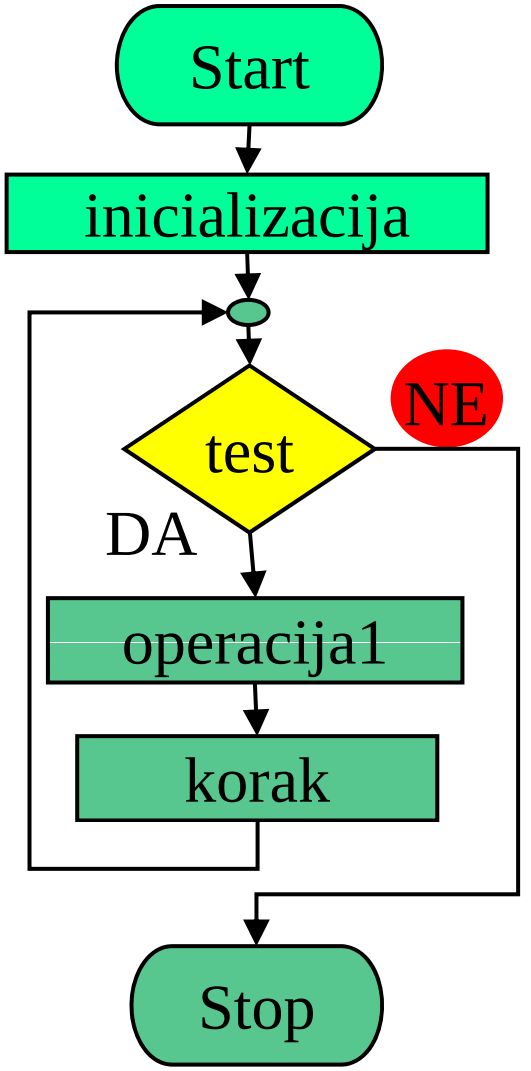
Primer: Zanka *for*

```
for(inicializacija;test;korak)
  {
    operacija1;
  }
```



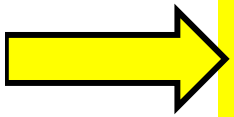
Primer: Zanka *for*

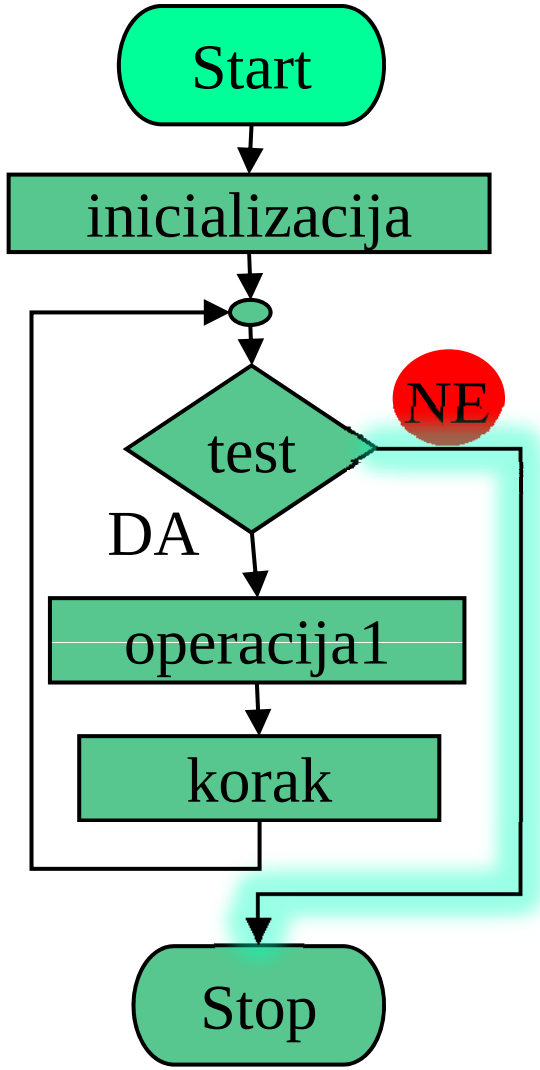
```
for(inicializacija;test;korak)
  {
    operacija1;
  }
```



Primer: Zanka *for*

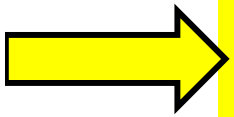
```
for(inicializacija;test;korak)
{
   operacija1;
}
```

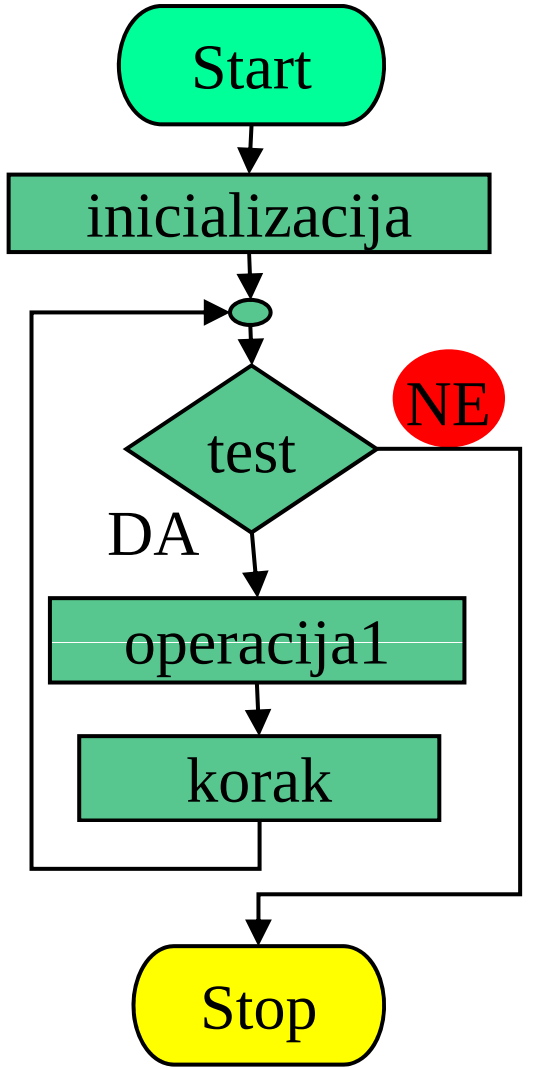




Primer: Zanka *for*

```
for(inicializacija;test;korak)
{
   operacija1;
}
```



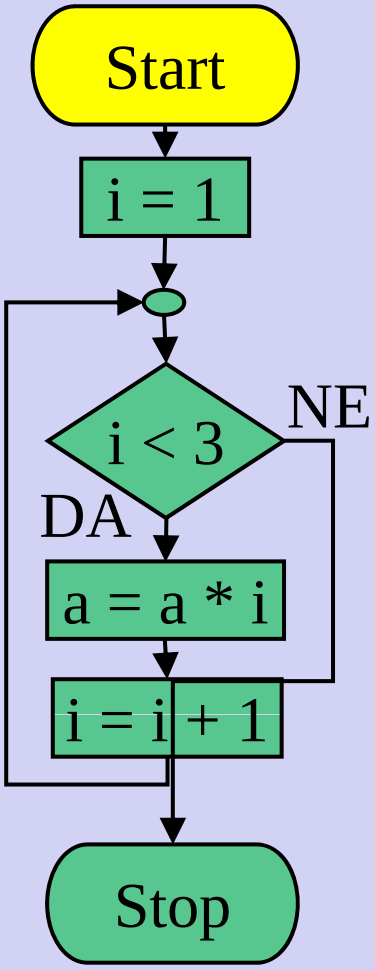


Praktični primer: Zanka *for*

Program:

```
for(i=1;i<3;i++)
{
    a *= i;
}
```

Diagram poteka:



Spremenljivki:

i = 3;

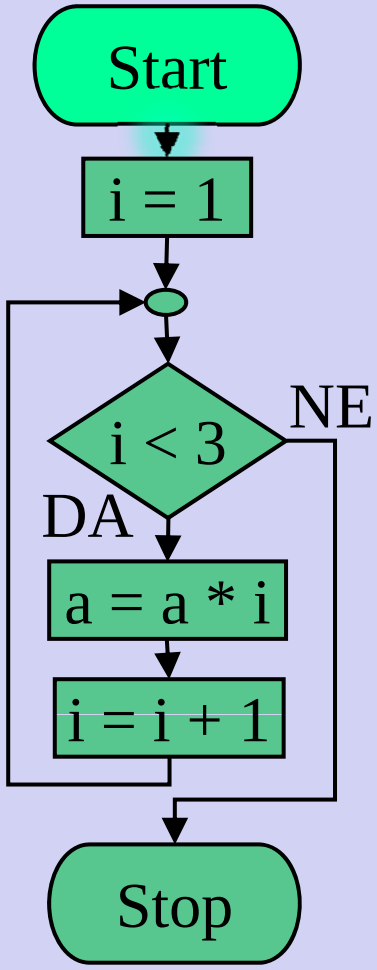
a = 2;

Praktični primer: Zanka *for*

Program:

```
for(i=1;i<3;i++)
  {
     a *= i;
  }
```

Diagram poteka:



Spremenljivki:

i = 3;

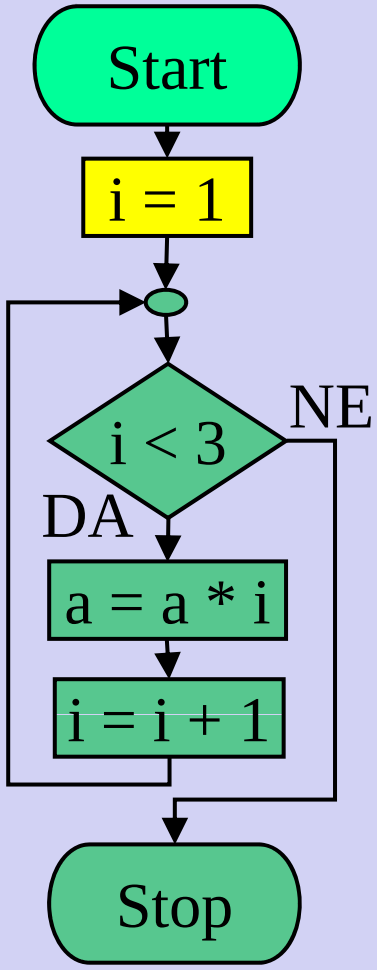
a = 2;

Praktični primer: Zanka *for*

Program:

```
for(i=1;i<3;i++)
  {
     a *= i;
  }
```

Diagram poteka:



Spremenljivki:

i = 1;

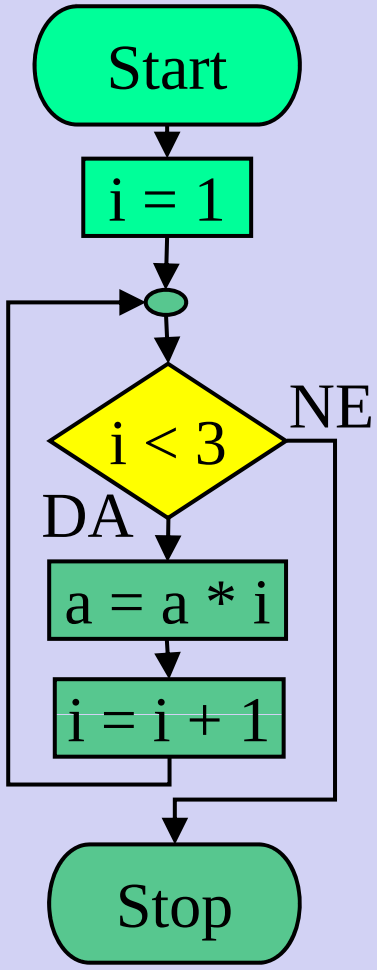
a = 2;

Praktični primer: Zanka *for*

Program:

```
for(i=1;i<3;i++)
  {
    a *= i;
  }
```

Diagram poteka:



Spremenljivki:

i = 1;

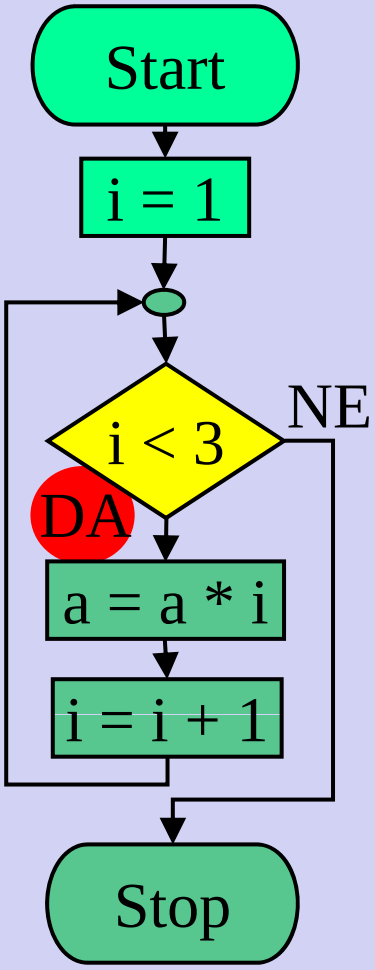
a = 2;

Praktični primer: Zanka *for*

Program:

```
for(i=1;i<3;i++)
  {
    a *= i;
  }
```

Diagram poteka:



Spremenljivki:

i = 1;

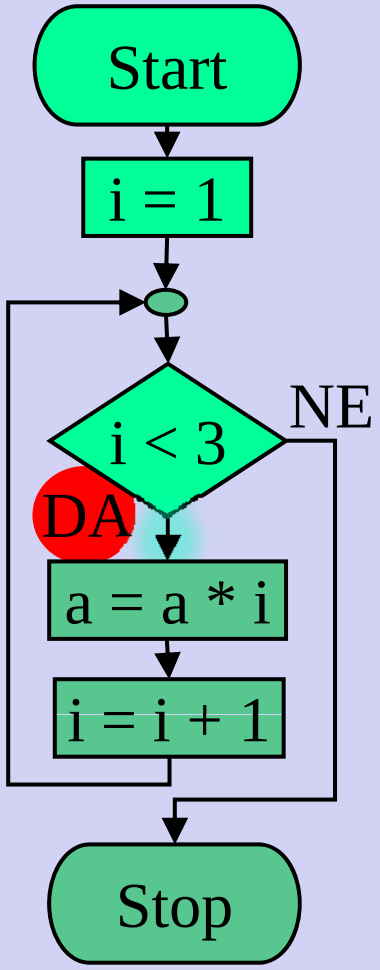
a = 2;

Praktični primer: Zanka *for*

Program:

```
for(i=1;i<3;i++)
{
   a *= i;
}
```

Diagram poteka:



Spremenljivki:

i = 1;

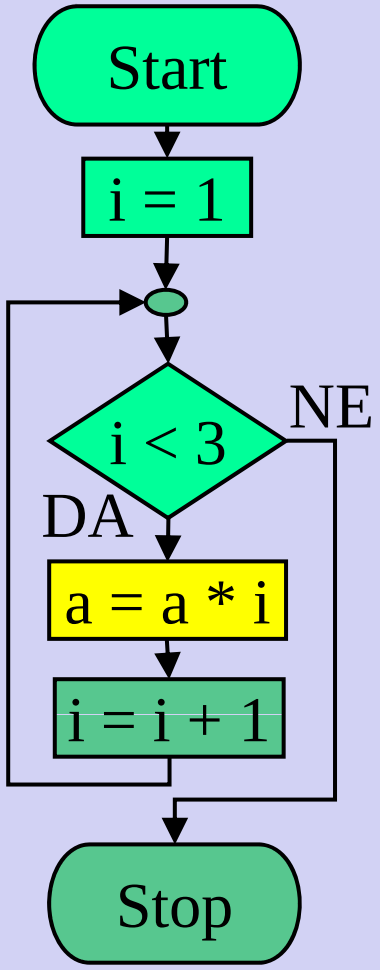
a = 2;

Praktični primer: Zanka *for*

Program:

```
for(i=1;i<3;i++)
{
   a *= i;
}
```

Diagram poteka:



Spremenljivki:

i = 1;

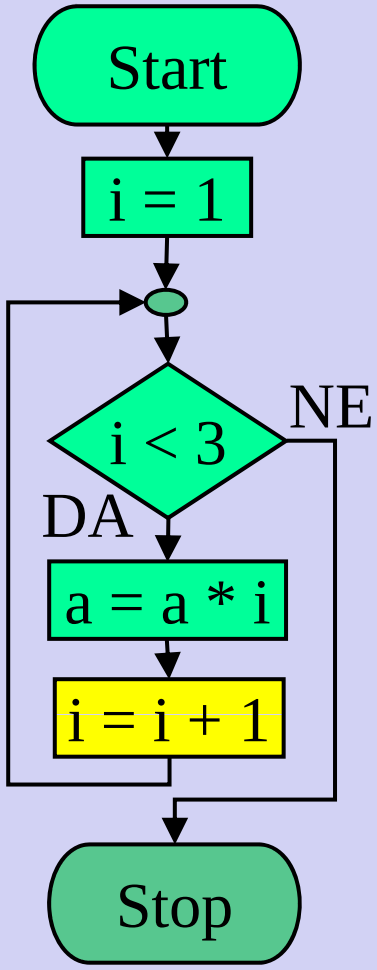
a = 2;

Praktični primer: Zanka *for*

Program:

```
for(i=1;i<3;i++)
  {
     a *= i;
  }
```

Diagram poteka:



Spremenljivki:

i = 2;

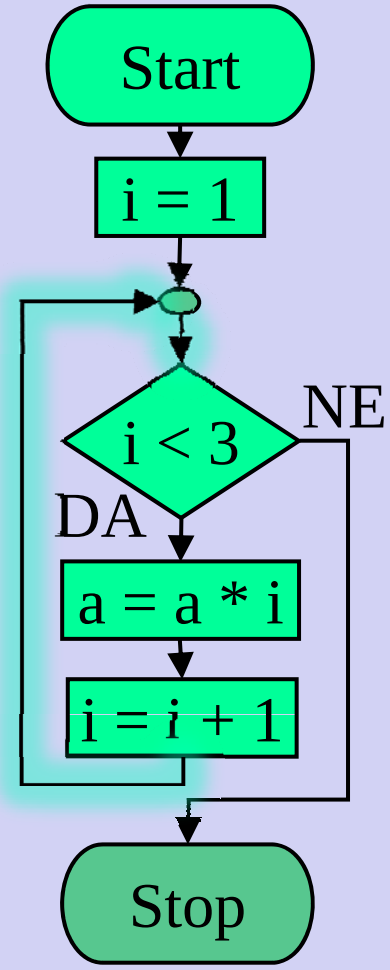
a = 2;

Praktični primer: Zanka *for*

Program:

```
for(i=1;i<3;i++)
  {
     a *= i;
  }
```

Diagram poteka:



Spremenljivki:

i = 2;

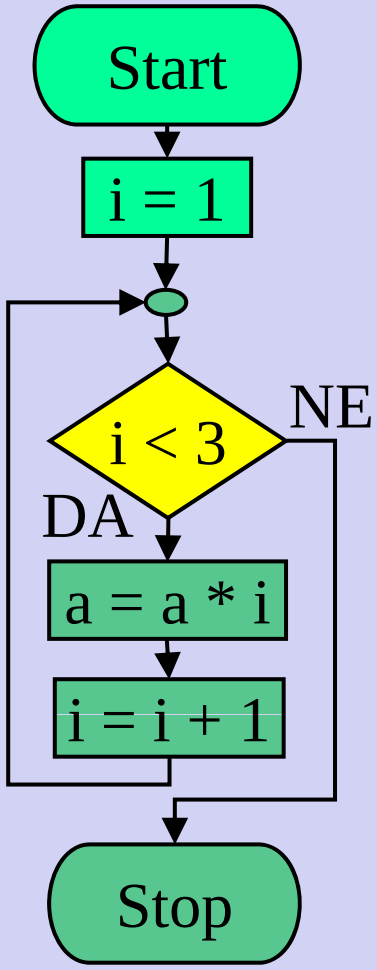
a = 2;

Praktični primer: Zanka *for*

Program:

```
for(i=1;i<3;i++)
  {
     a *= i;
  }
```

Diagram poteka:



Spremenljivki:

i = 2;

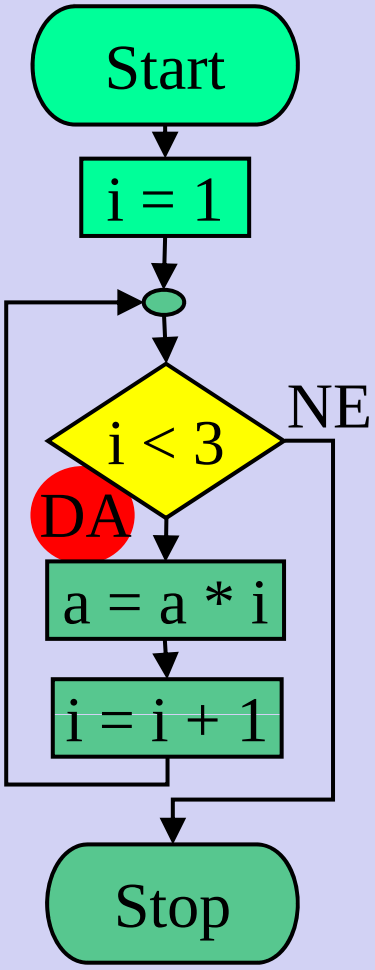
a = 2;

Praktični primer: Zanka *for*

Program:

```
for(i=1;i<3;i++)
  {
    a *= i;
  }
```

Diagram poteka:



Spremenljivki:

i = 2;

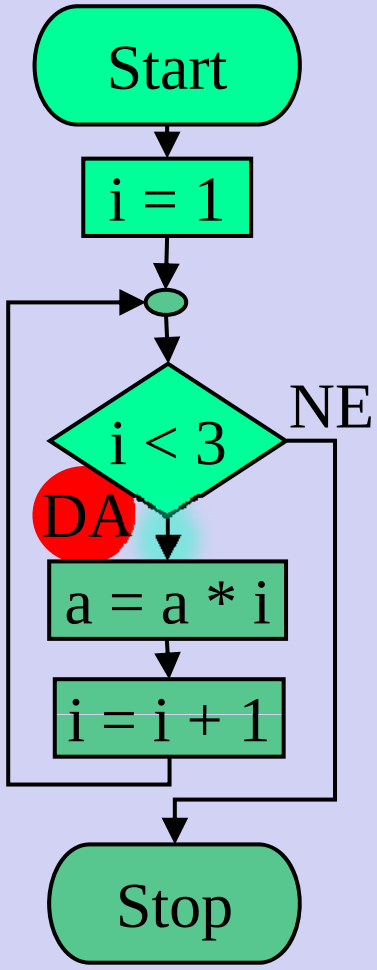
a = 2;

Praktični primer: Zanka *for*

Program:

```
for(i=1;i<3;i++)
{
    a *= i;
}
```

Diagram poteka:



Spremenljivki:

i = 2;

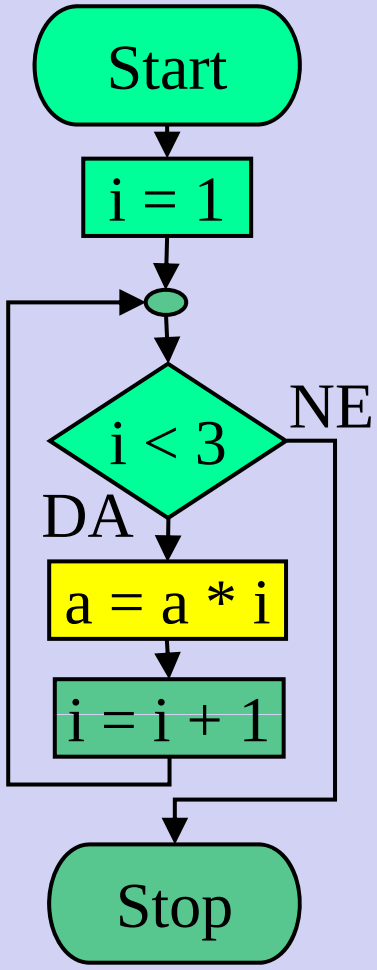
a = 2;

Praktični primer: Zanka *for*

Program:

```
for(i=1;i<3;i++)
{
   a *= i;
}
```

Diagram poteka:



Spremenljivki:

i = 2;

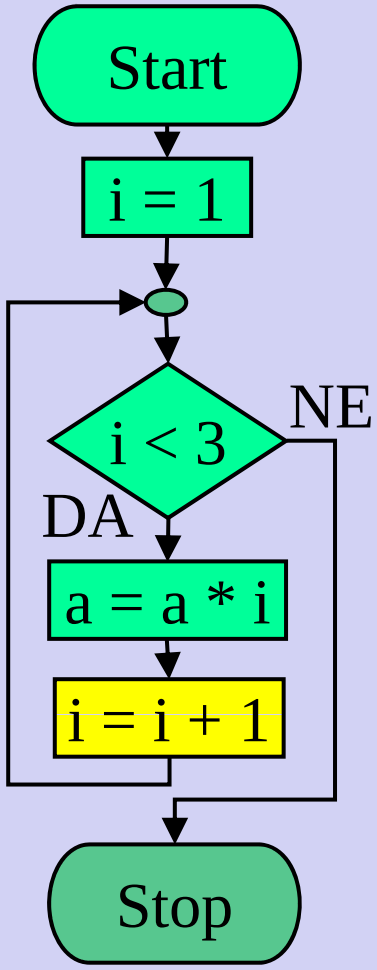
a = 4;

Praktični primer: Zanka *for*

Program:

```
for(i=1;i<3;i++)
  {
     a *= i;
  }
```

Diagram poteka:



Spremenljivki:

i = 3;

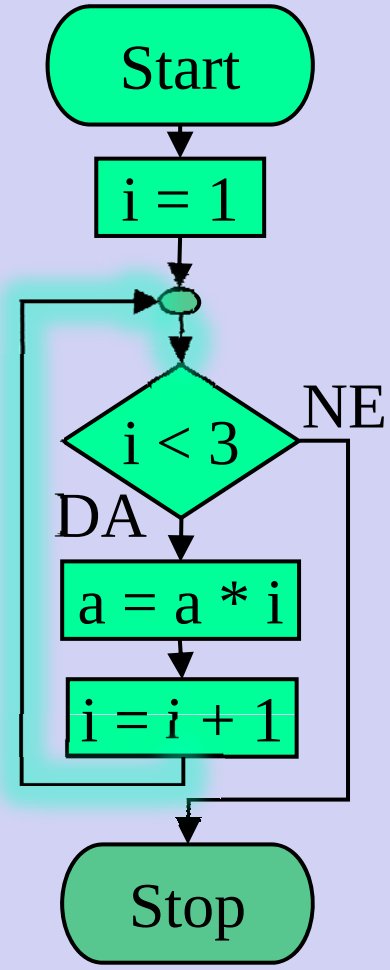
a = 4;

Praktični primer: Zanka *for*

Program:

```
for(i=1;i<3;i++)
  {
     a *= i;
  }
```

Diagram poteka:



Spremenljivki:

i = 3;

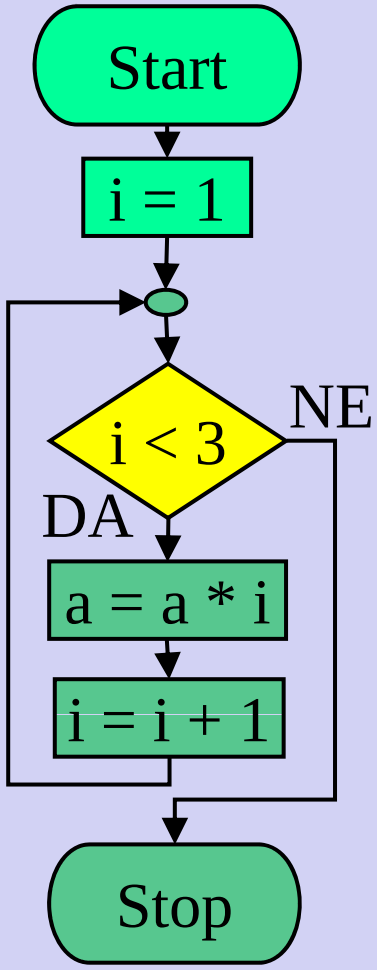
a = 4;

Praktični primer: Zanka *for*

Program:

```
for(i=1;i<3;i++)
  {
     a *= i;
  }
```

Diagram poteka:



Spremenljivki:

i = 3;

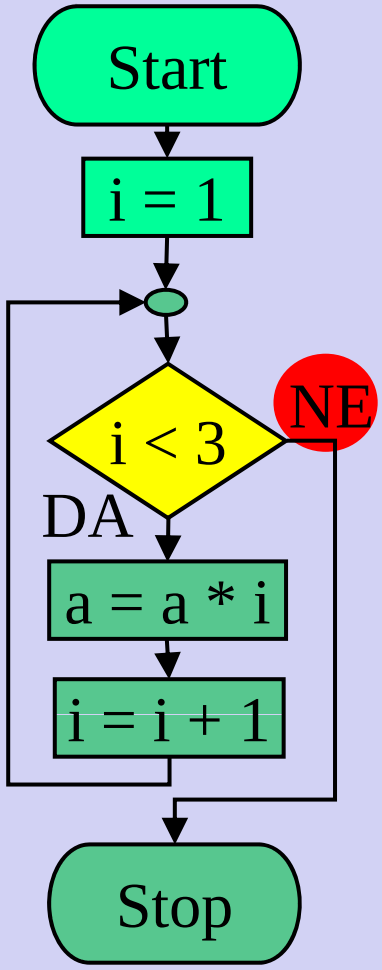
a = 4;

Praktični primer: Zanka *for*

Program:

```
for(i=1;i<3;i++)
  {
    a *= i;
  }
```

Diagram poteka:



Spremenljivki:

i = 3;

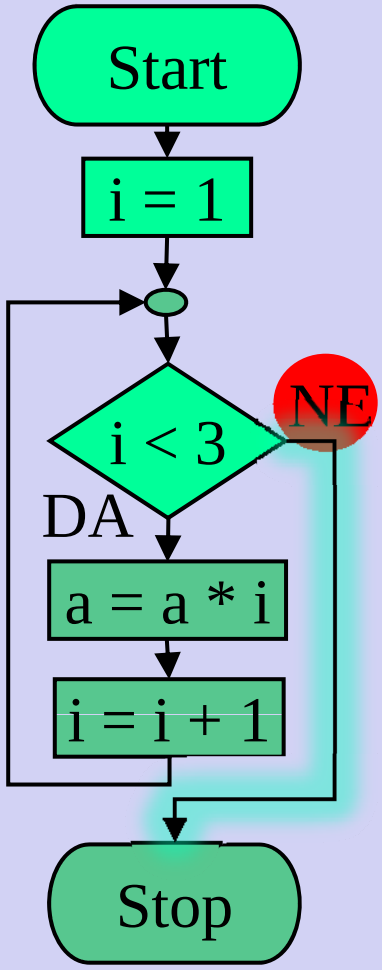
a = 4;

Praktični primer: Zanka *for*

Program:

```
for(i=1;i<3;i++)
  {
     a *= i;
  }
```

Diagram poteka:



Spremenljivki:

i = 3;

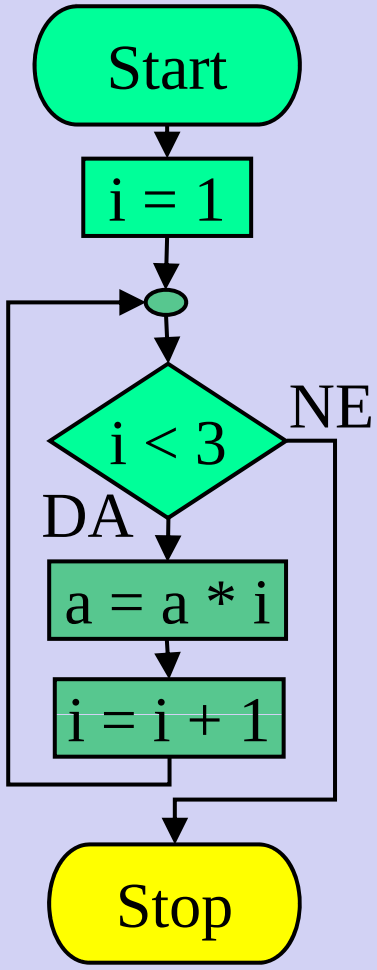
a = 4;

Praktični primer: Zanka *for*

Program:

```
for(i=1;i<3;i++)
{
   a *= i;
}
```

Diagram poteka:



Spremenljivki:

i = 3;

a = 4;

Primer: Funkcija

```
…
f = vsota(g,h);
…

int vsota(int a, int b)
{
    int c;
    c=a+b;
    return c;
}
```

a = nedefiniran

b = nedefiniran

c = nedefiniran

f = 0

g = 1

h = 2

Primer: Funkcija

```
…
f = vsota(g,h);
…

int vsota(int a, int b)
{
    int c;
    c=a+b;
    return c;
}
```

a = nedefiniran

b = nedefiniran

c = nedefiniran

f = 0

g = 1

h = 2

Primer: Funkcija

```
…
f = vsota(g,h);
…

int vsota(int a, int b)
{
    int c;
    c=a+b;
    return c;
}
```

a = 1

b = 2

c = nedefiniran

f = 0

g = 1

h = 2

Primer: Funkcija

```
…
f = vsota(g,h);
…

int vsota(int a, int b)
{
    int c;
    c=a+b;
    return c;
}
```

a = 1

b = 2

c = nedefiniran

f = 0

g = 1

h = 2

Primer: Funkcija

```
…
f = vsota(g,h);
…

int vsota(int a, int b)
{
    int c;
    c=a+b;
    return c;
}
```

a = 1

b = 2

c = 3

f = 0

g = 1

h = 2

Primer: Funkcija

```
…
f = vsota(g,h);
…

int vsota(int a, int b)
{
    int c;
    c=a+b;
    return c;
}
```

a = 1

b = 2

c = 3

f = 0

g = 1

h = 2

Primer: Funkcija

```
…
f = vsota(g,h);
…

int vsota(int a, int b)
{
    int c;
    c=a+b;
    return c;
}
```

a = nedefiniran

b = nedefiniran

c = nedefiniran

f = 3

g = 1

h = 2

Primer: Funkcija

```
…
f = vsota(g,h);
…

int vsota(int a, int b)
{
    int c;
    c=a+b;
    return c;
}
```

a = nedefiniran

b = nedefiniran

c = nedefiniran

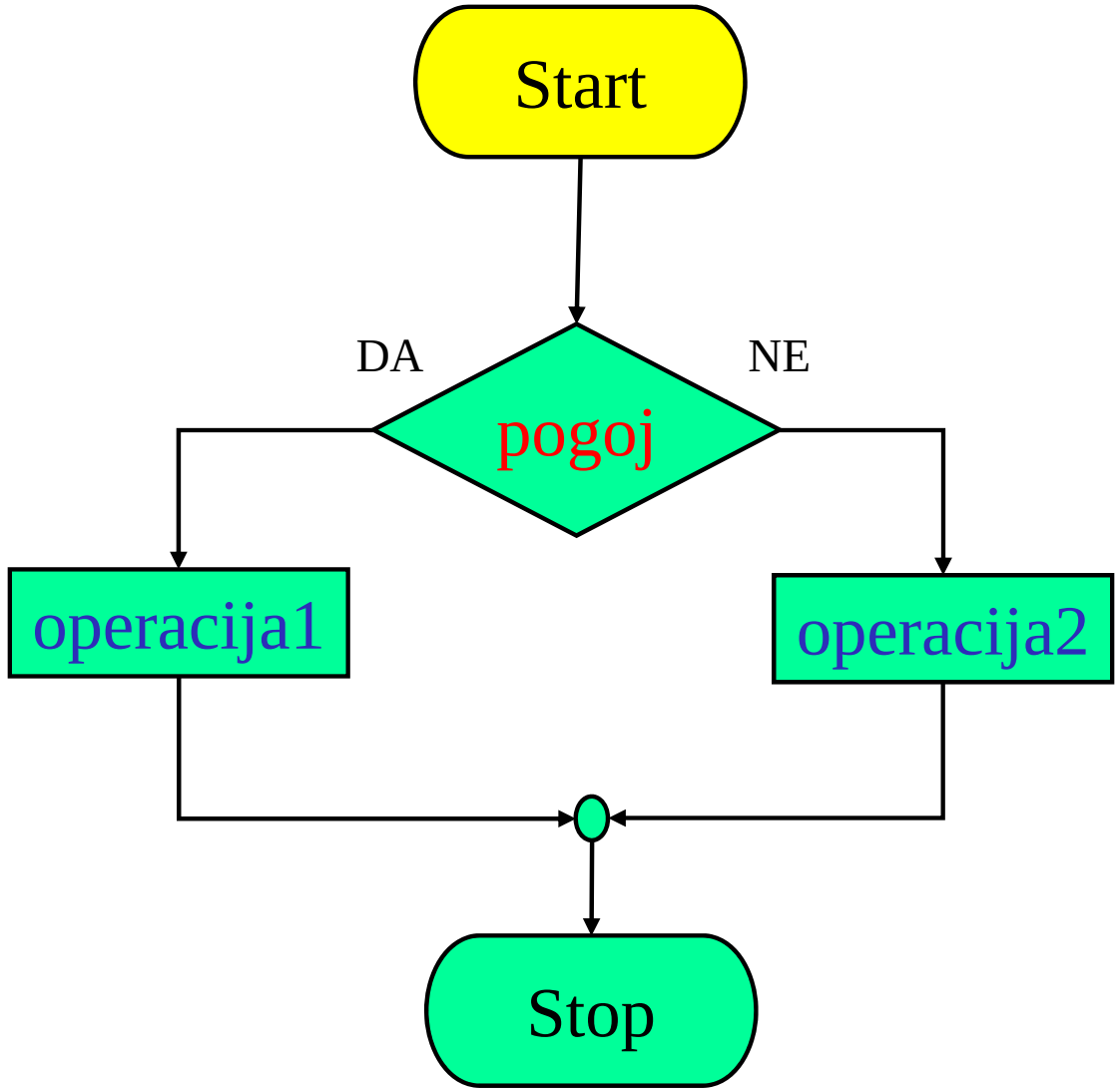
f = 3

g = 1

h = 2

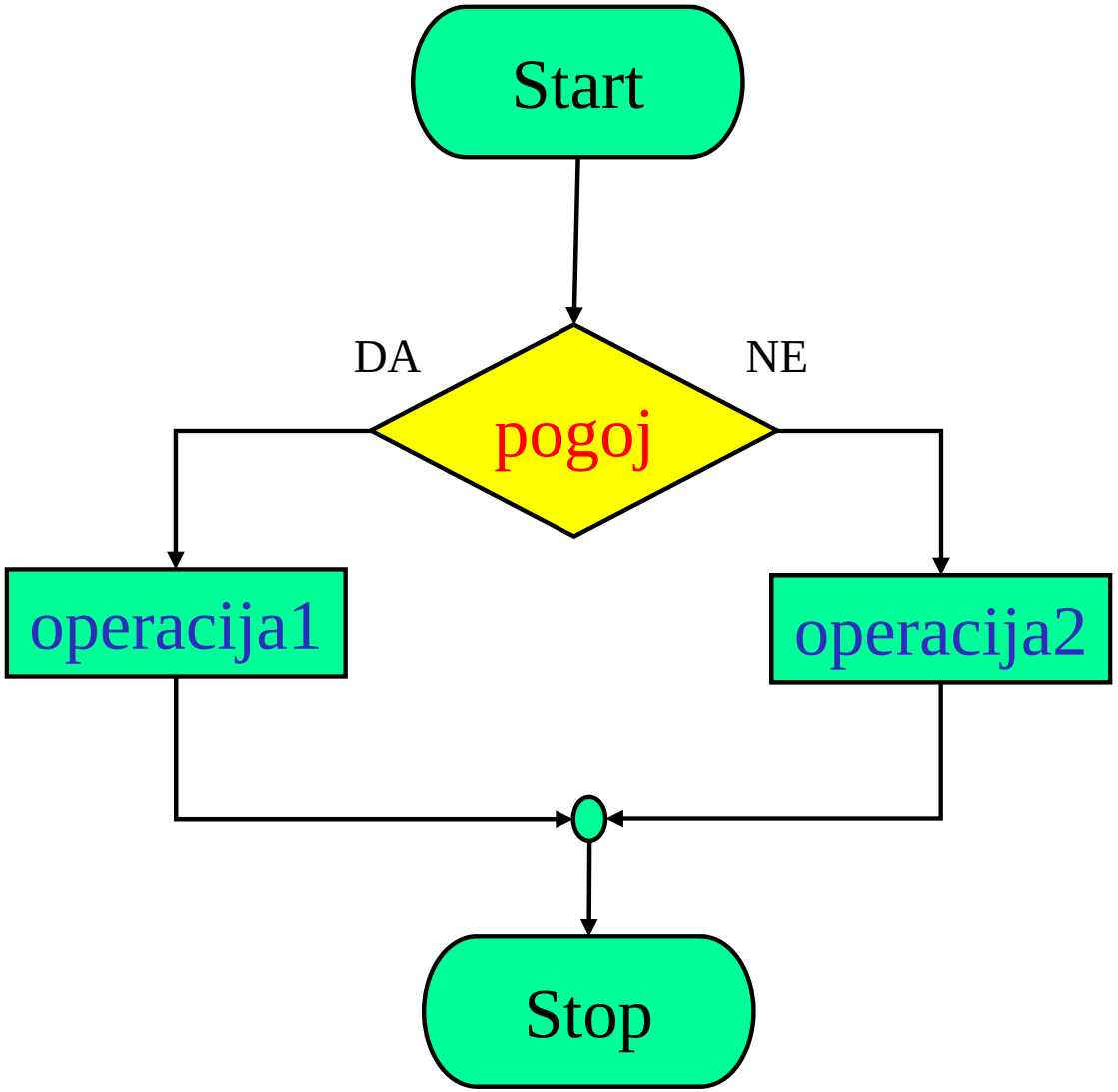
Primer: Stavek *if-else*

```
if(pogoj)
{
    operacija1;
}
else
{
    operacija2;
}
```



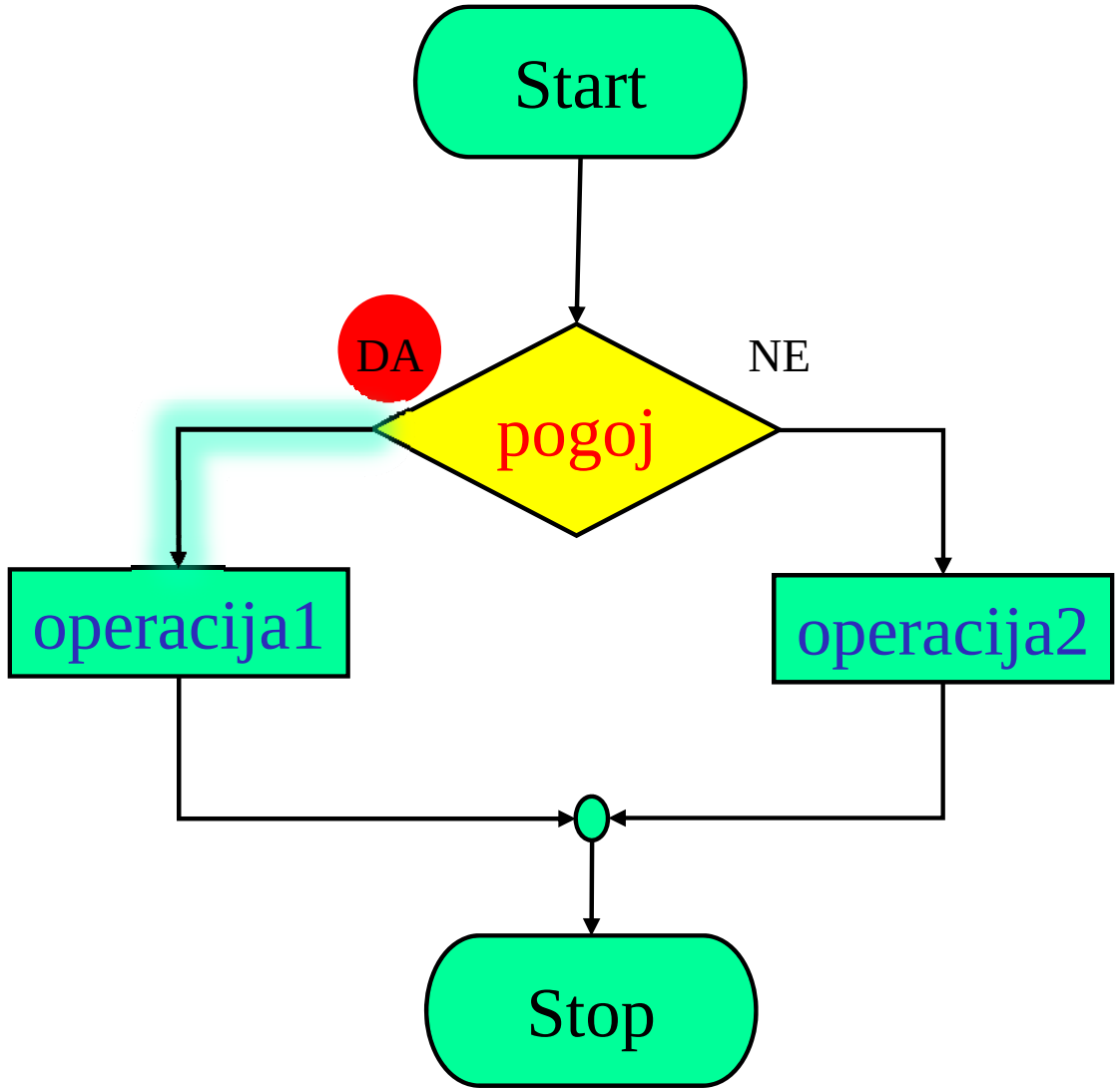
Primer: Stavek *if-else*

```
if(pogoj)
{
    operacija1;
}
else
{
    operacija2;
}
```



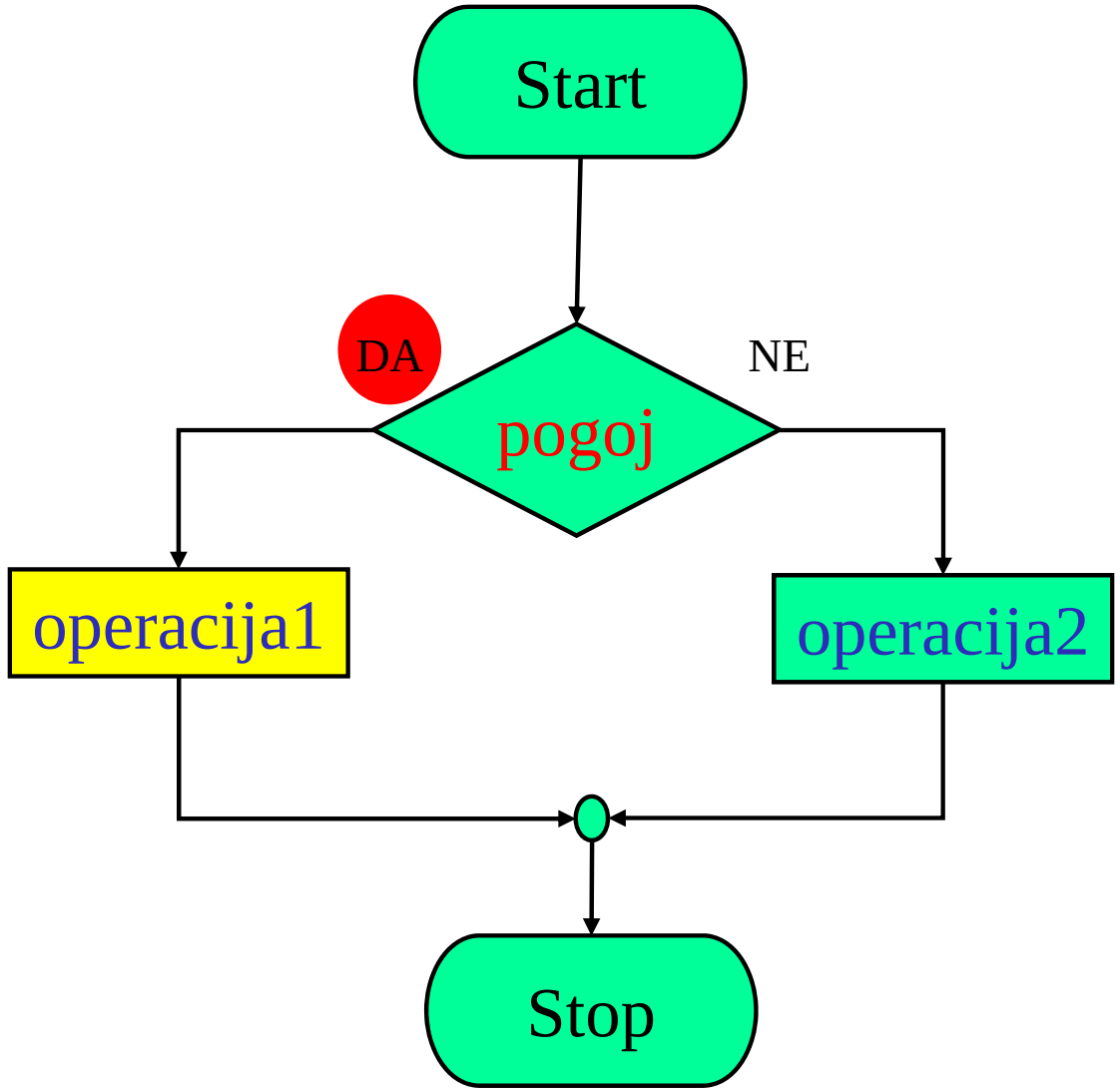
Primer: Stavek *if-else*

```
if(pogoj)
{
    operacija1;
}
else
{
    operacija2;
}
```



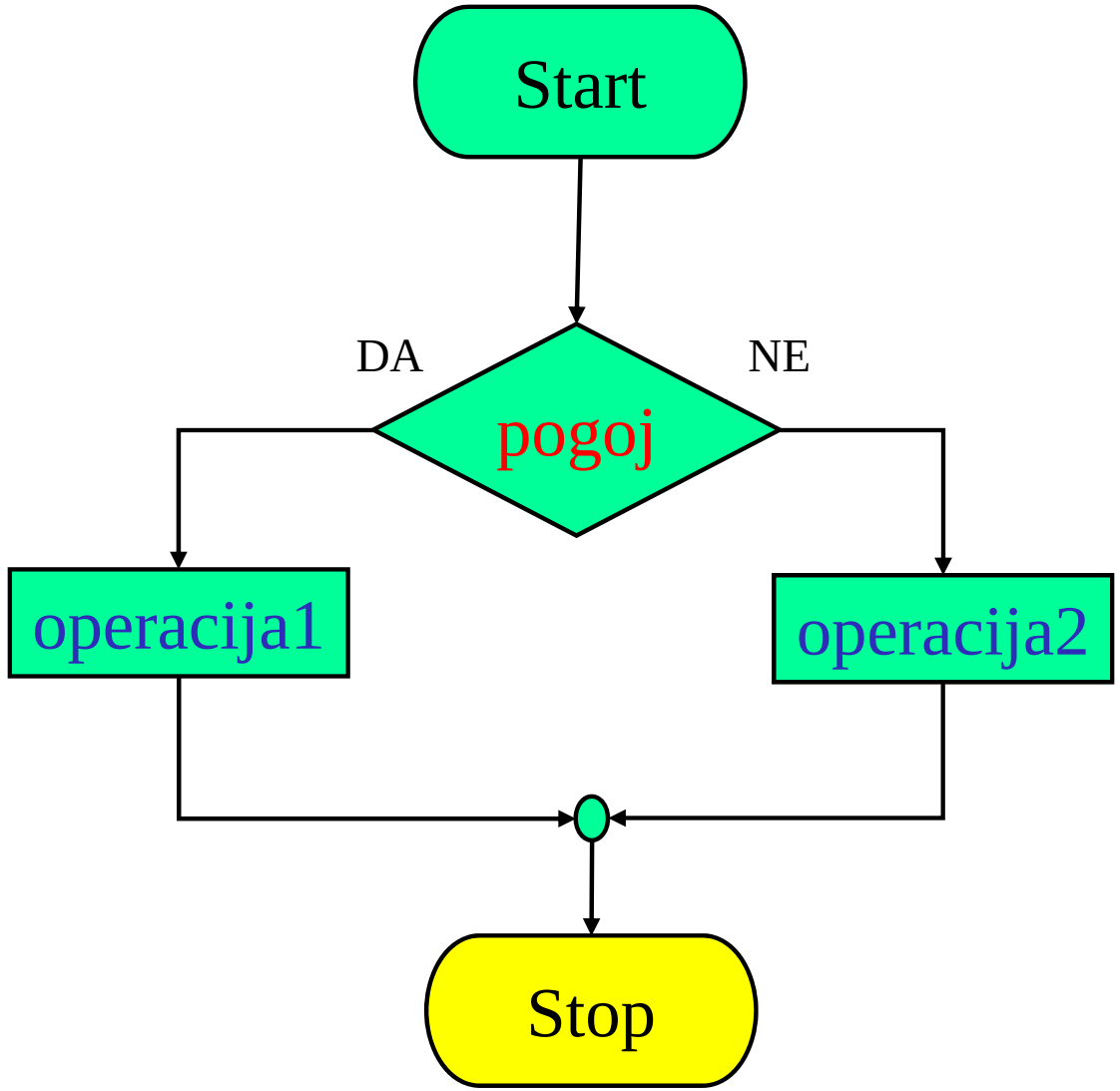
Primer: Stavek *if-else*

```
if(pogoj)
{
    operacija1;
}
else
{
    operacija2;
}
```



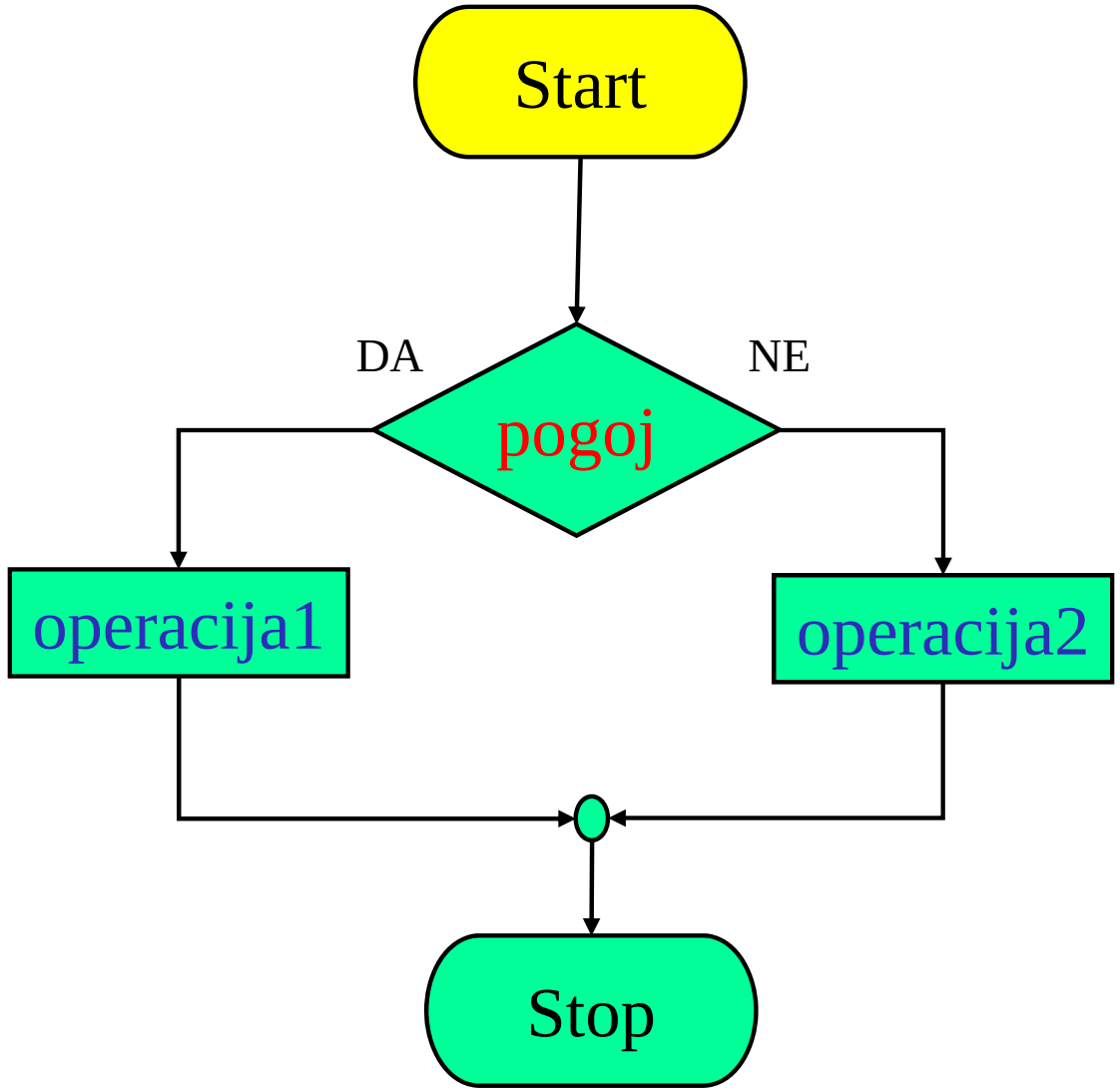
Primer: Stavek *if-else*

```
if(pogoj)
{
    operacija1;
}
else
{
    operacija2;
}
```



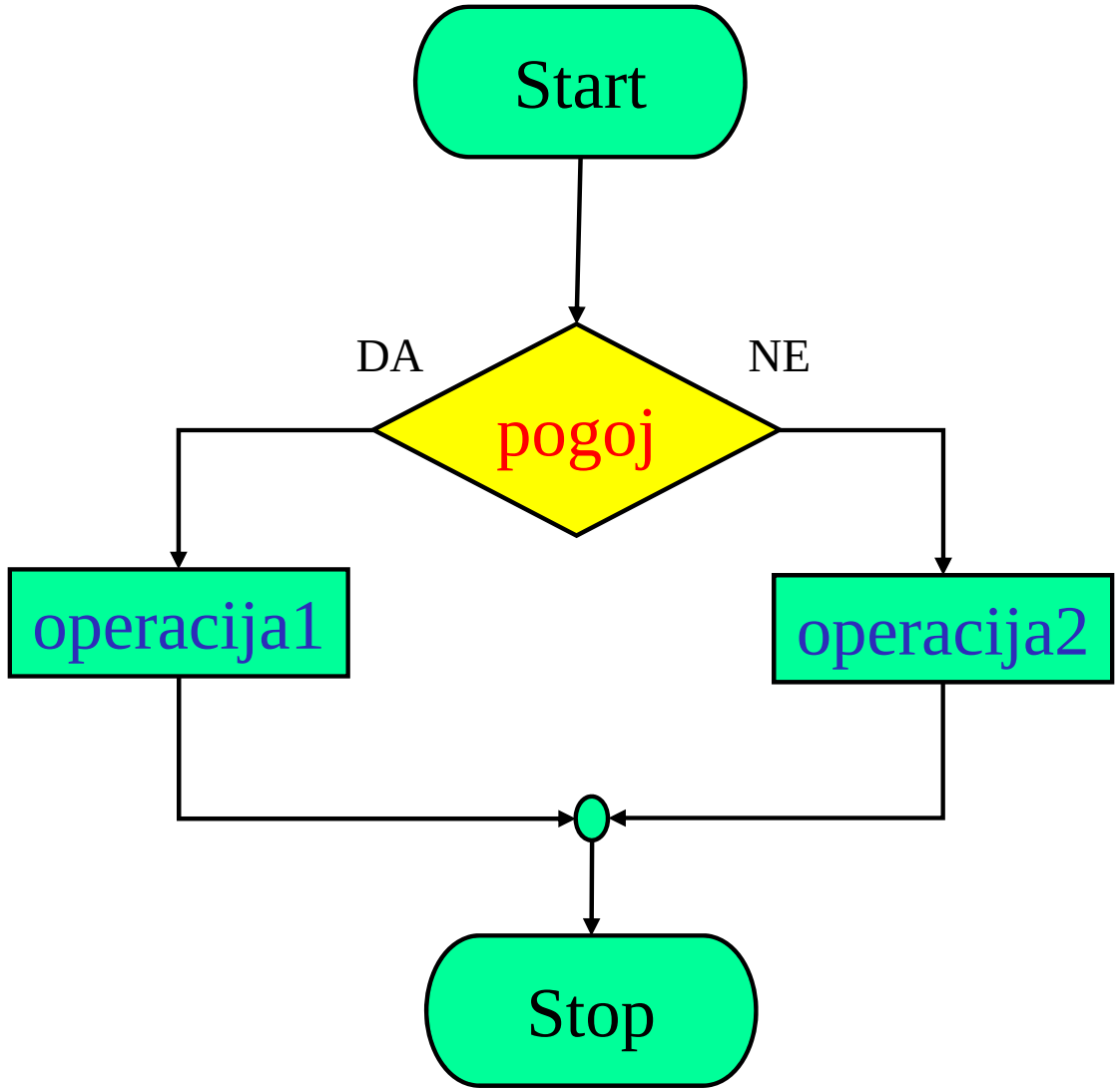
Primer: Stavek *if-else*

```
if(pogoj)
{
    operacija1;
}
else
{
    operacija2;
}
```



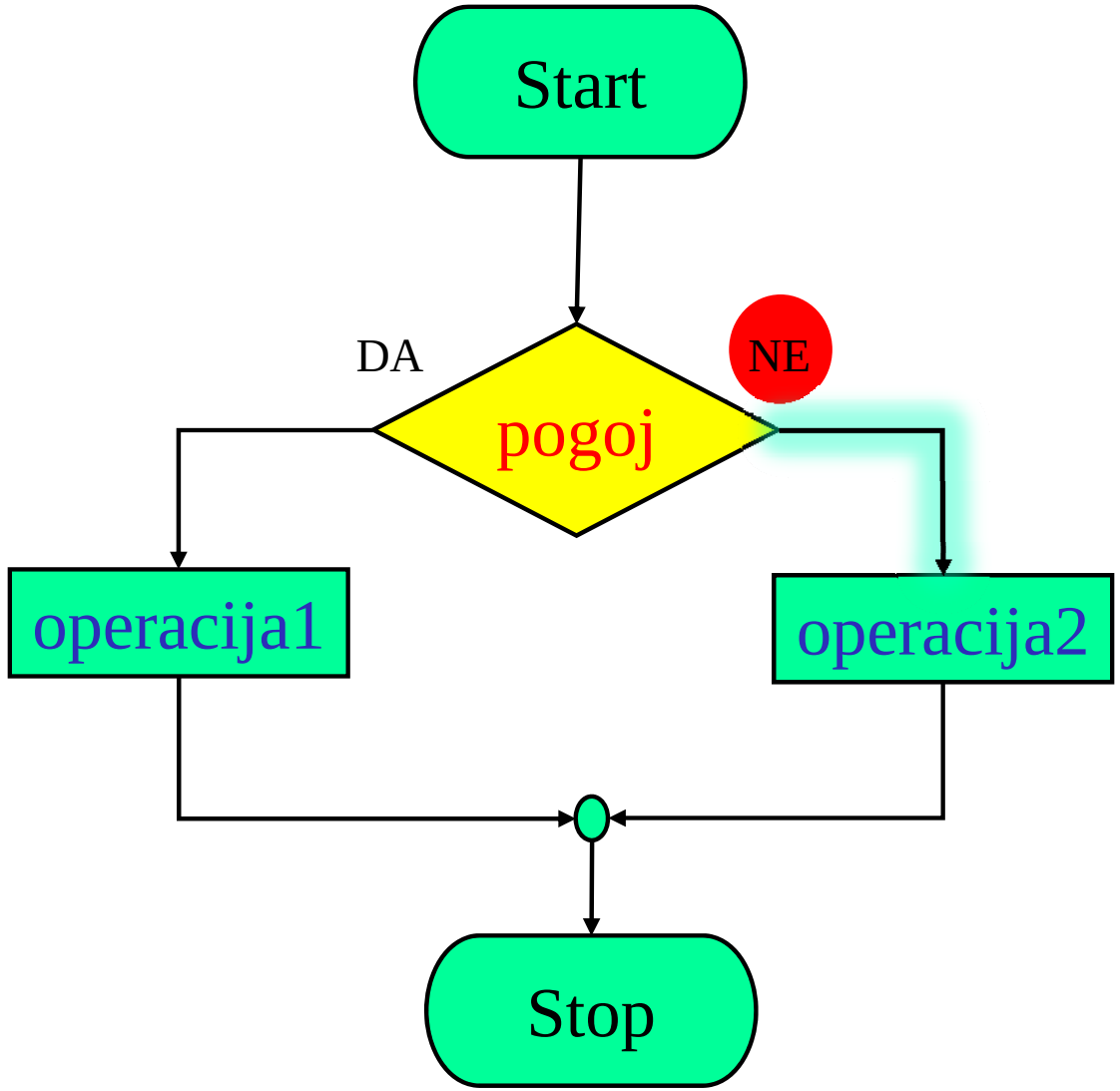
Primer: Stavek *if-else*

```
if(pogoj)
{
    operacija1;
}
else
{
    operacija2;
}
```



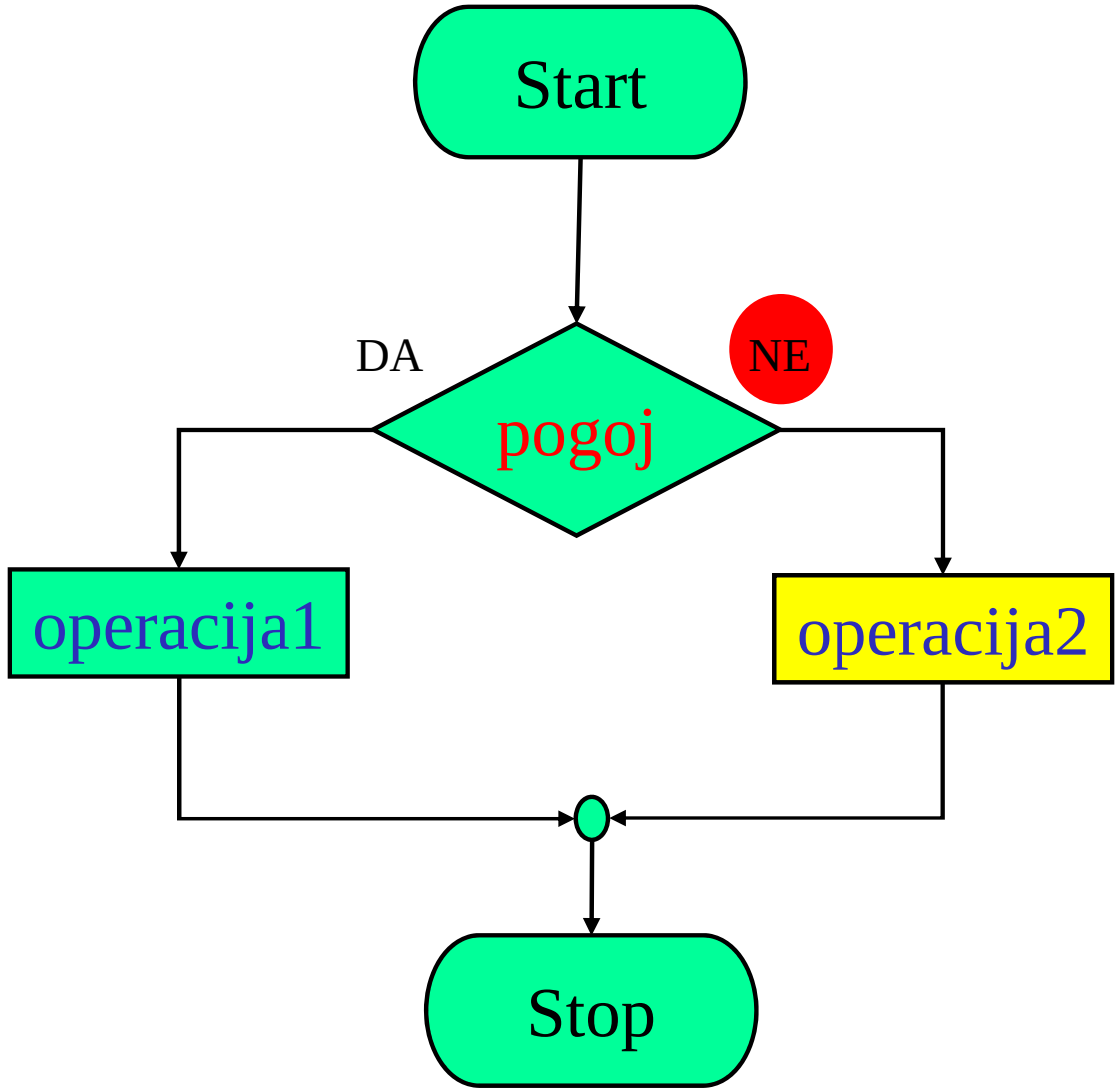
Primer: Stavek *if-else*

```
if(pogoj)
{
    operacija1;
}
else
{
    operacija2;
}
```



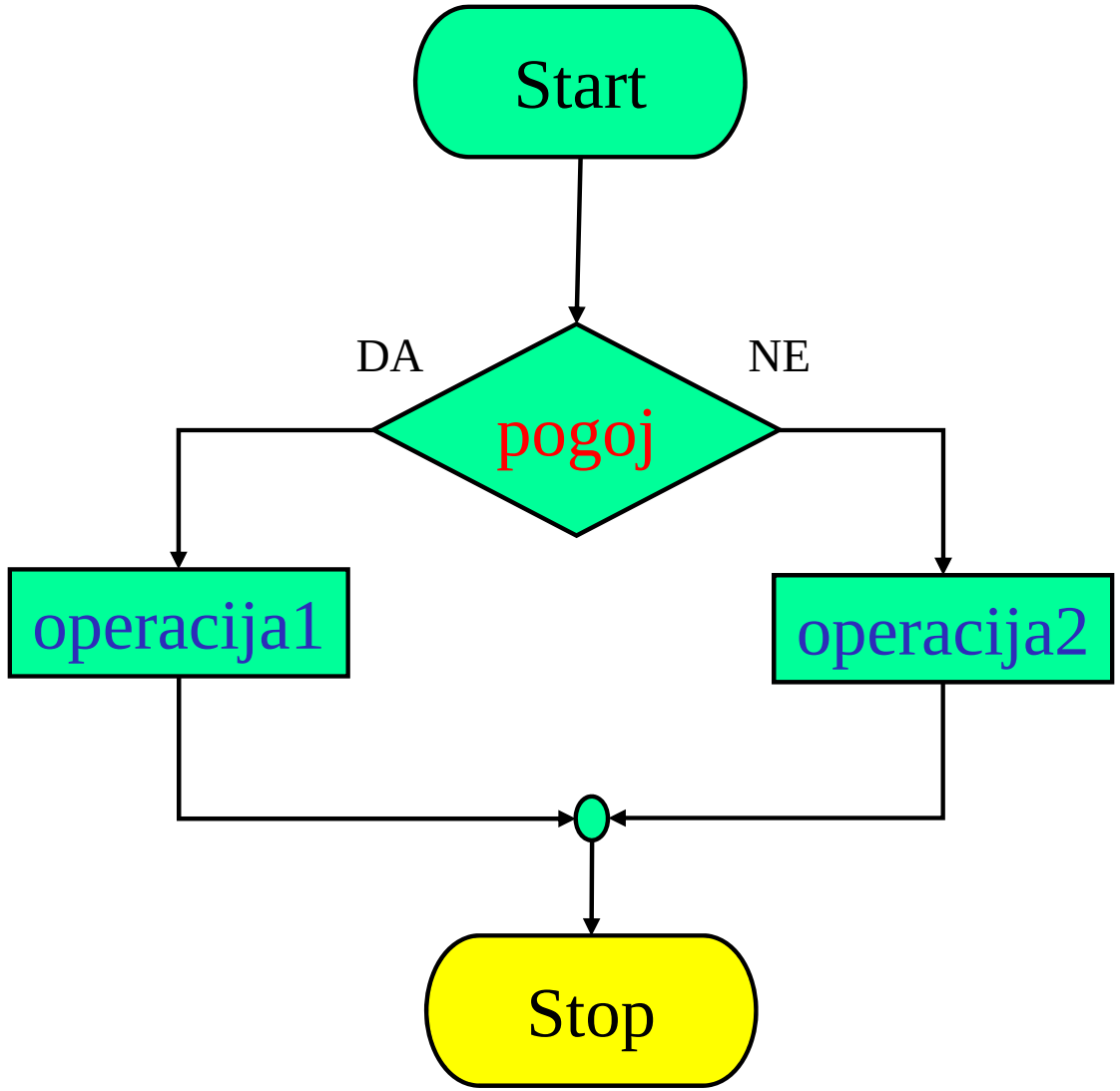
Primer: Stavek *if-else*

```
if(pogoj)
{
    operacija1;
}
else
{
    operacija2;
}
```



Primer: Stavek *if-else*

```
if(pogoj)
{
    operacija1;
}
else
{
    operacija2;
}
```

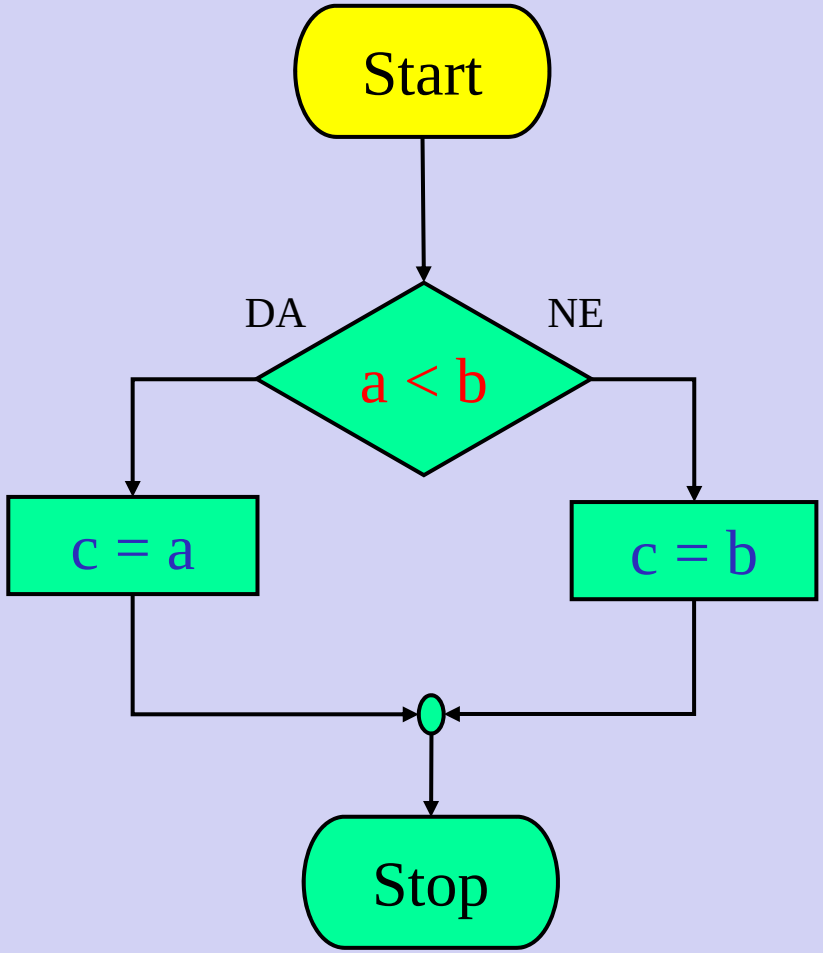


Praktični primer: Stavek *if-else*

Program:

```
if(a<b)
{
    c=a;
}
else
{
    c=b;
}
```

Diagram poteka:



Spremenljivke:

a = 2

b = 3

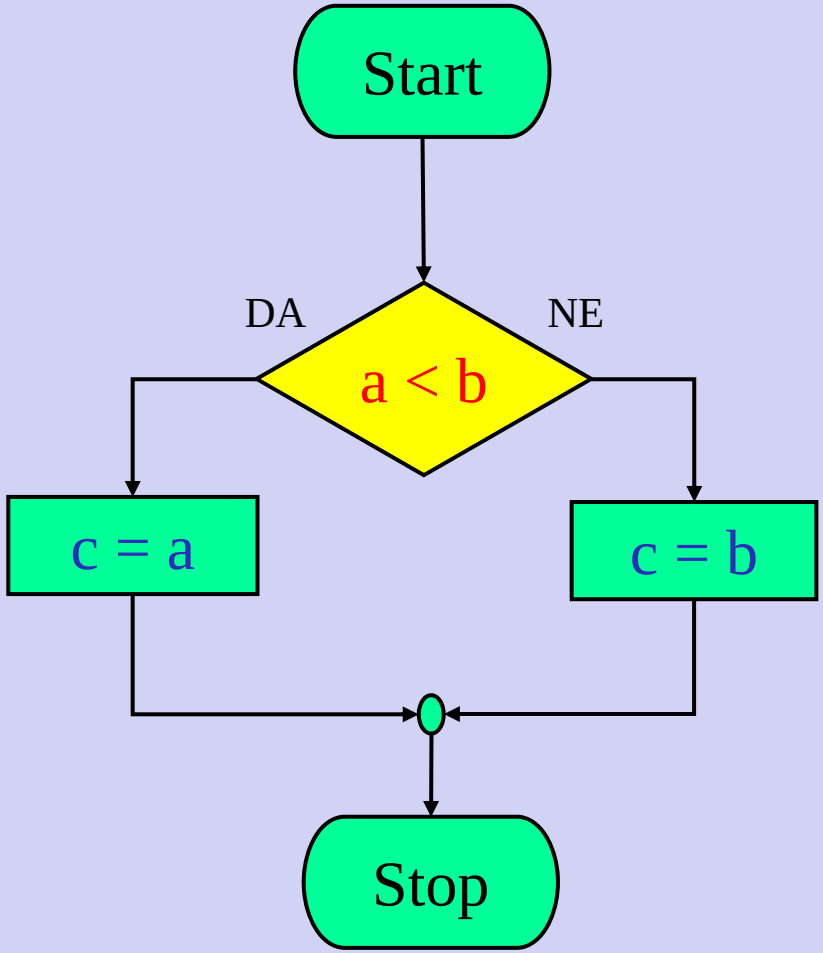
c = 1

Praktični primer: Stavek *if-else*

Program:

```
if(a<b)
{
    c=a;
}
else
{
    c=b;
}
```

Diagram poteka:



Spremenljivke:

a = 2

b = 3

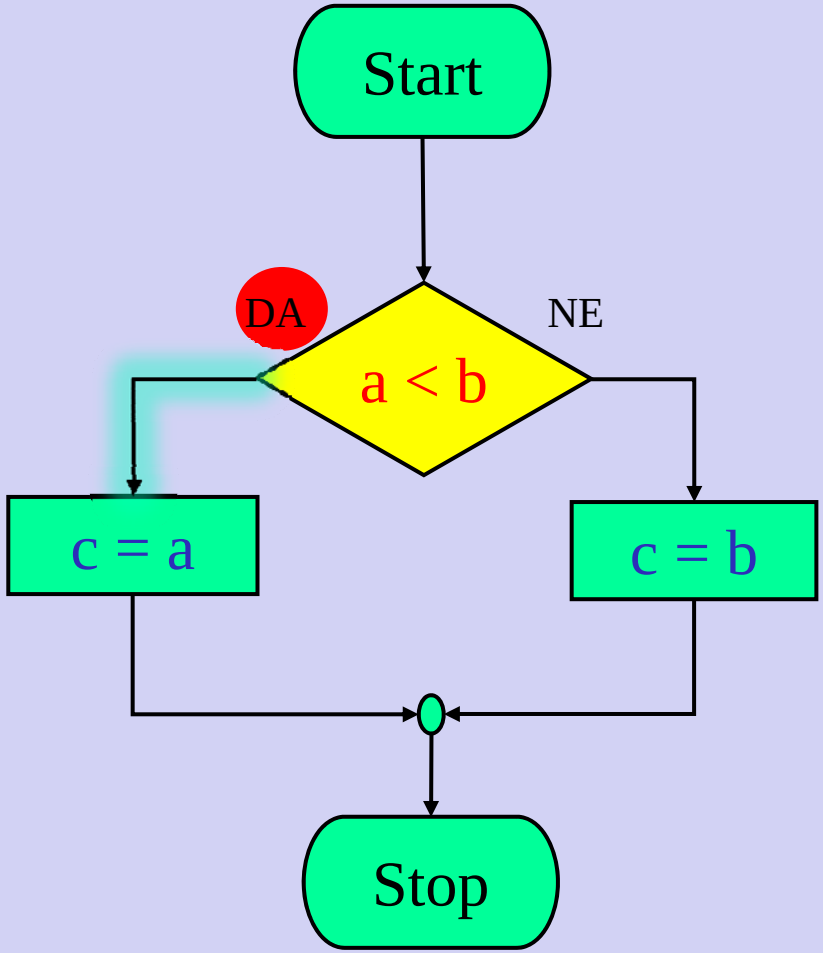
c = 1

Praktični primer: Stavek *if-else*

Program:

```
if(a<b)
{
    c=a;
}
else
{
    c=b;
}
```

Diagram poteka:



Spremenljivke:

a = 2

b = 3

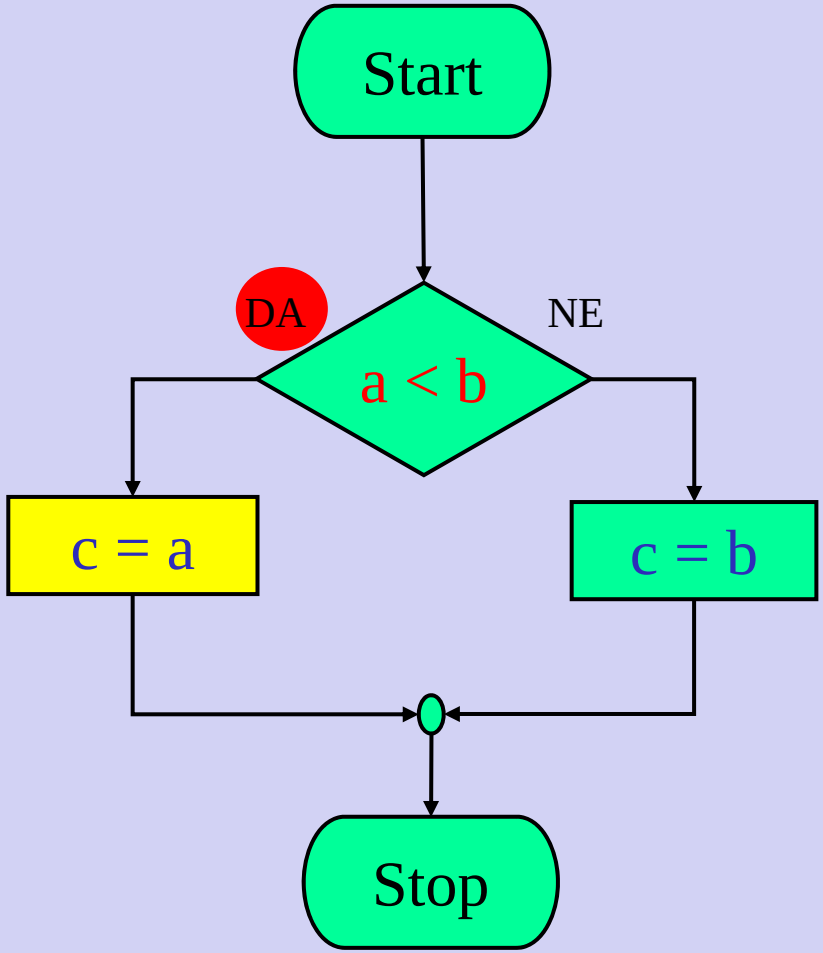
c = 1

Praktični primer: Stavek *if-else*

Program:

```
if(a<b)
{
    c=a;
}
else
{
    c=b;
}
```

Diagram poteka:



Spremenljivke:

a = 2

b = 3

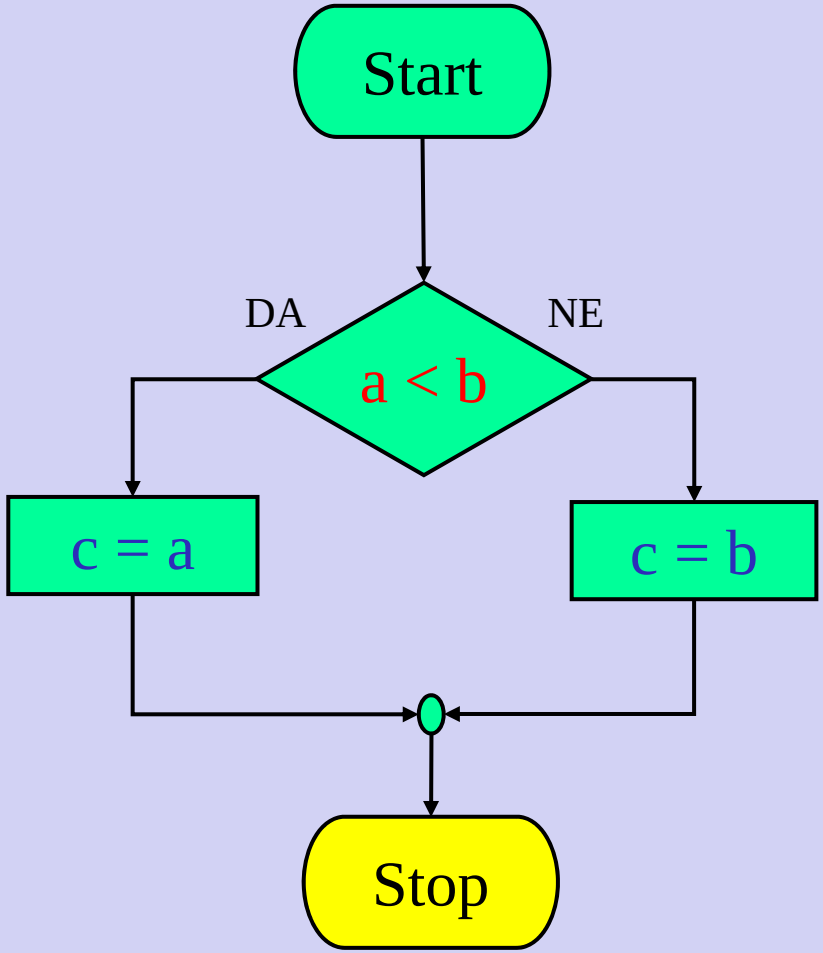
c = 2

Praktični primer: Stavek *if-else*

Program:

```
if(a<b)
{
    c=a;
}
else
{
    c=b;
}
```

Diagram poteka:



Spremenljivke:

a = 2

b = 3

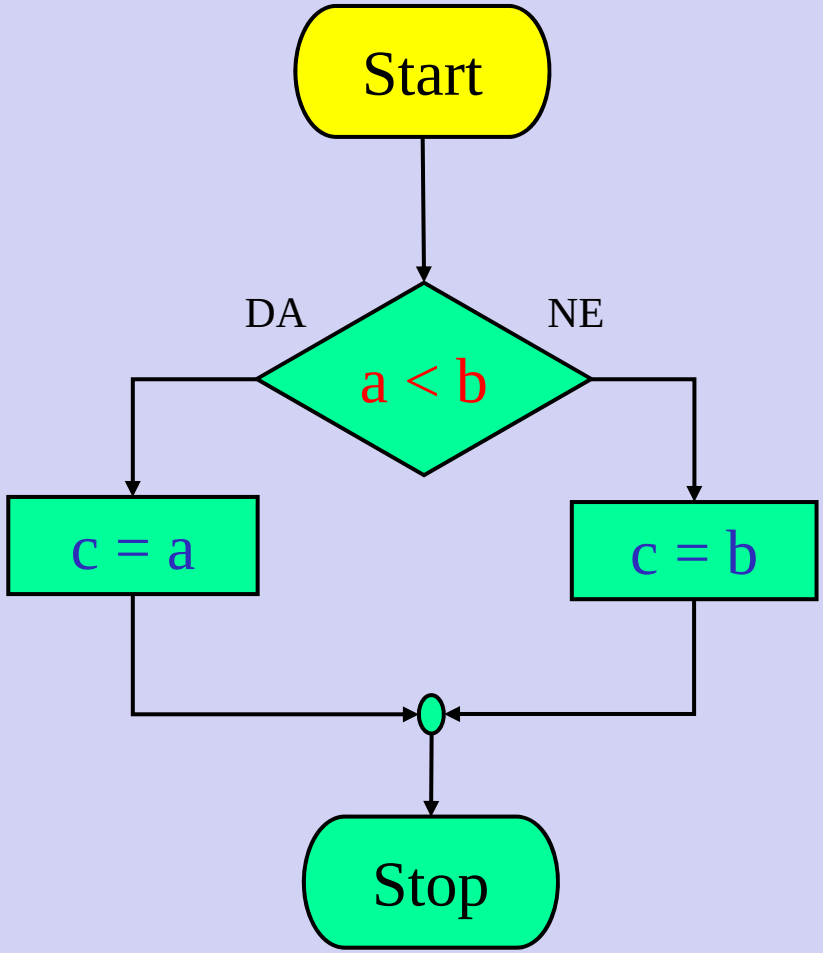
c = 2

Praktični primer: Stavek *if-else*

Program:

```
if(a<b)
{
    c=a;
}
else
{
    c=b;
}
```

Diagram poteka:



Spremenljivke:

a = 7

b = 3

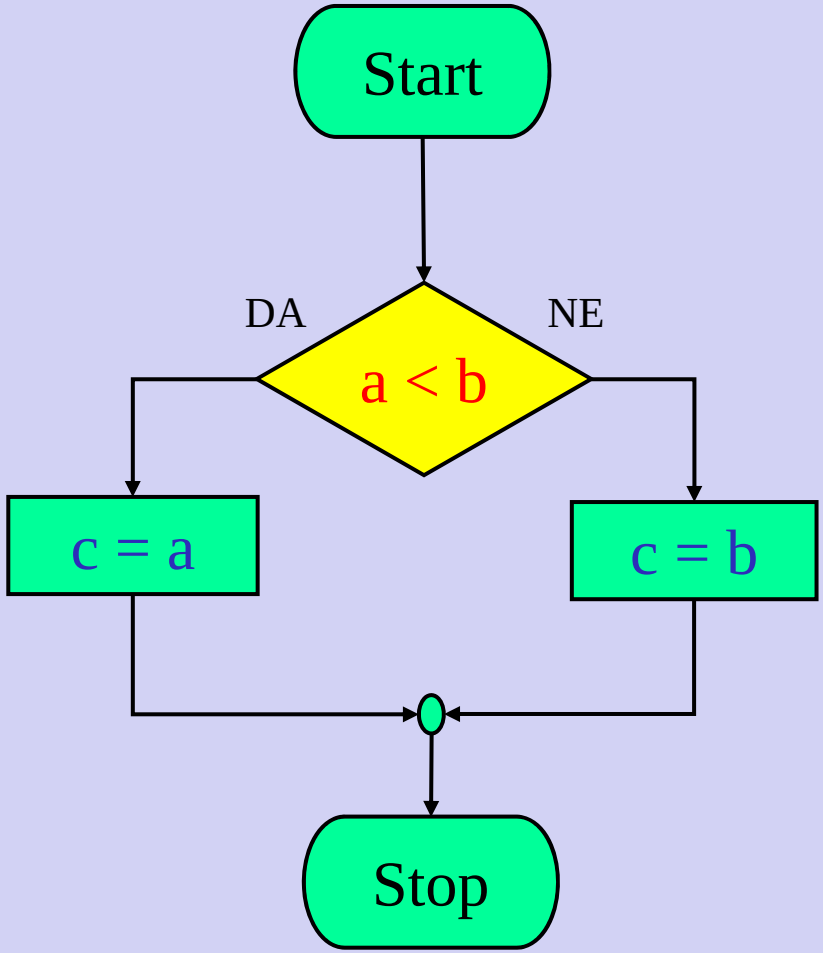
c = 1

Praktični primer: Stavek *if-else*

Program:

```
if(a<b)
{
    c=a;
}
else
{
    c=b;
}
```

Diagram poteka:



Spremenljivke:

a = 7

b = 3

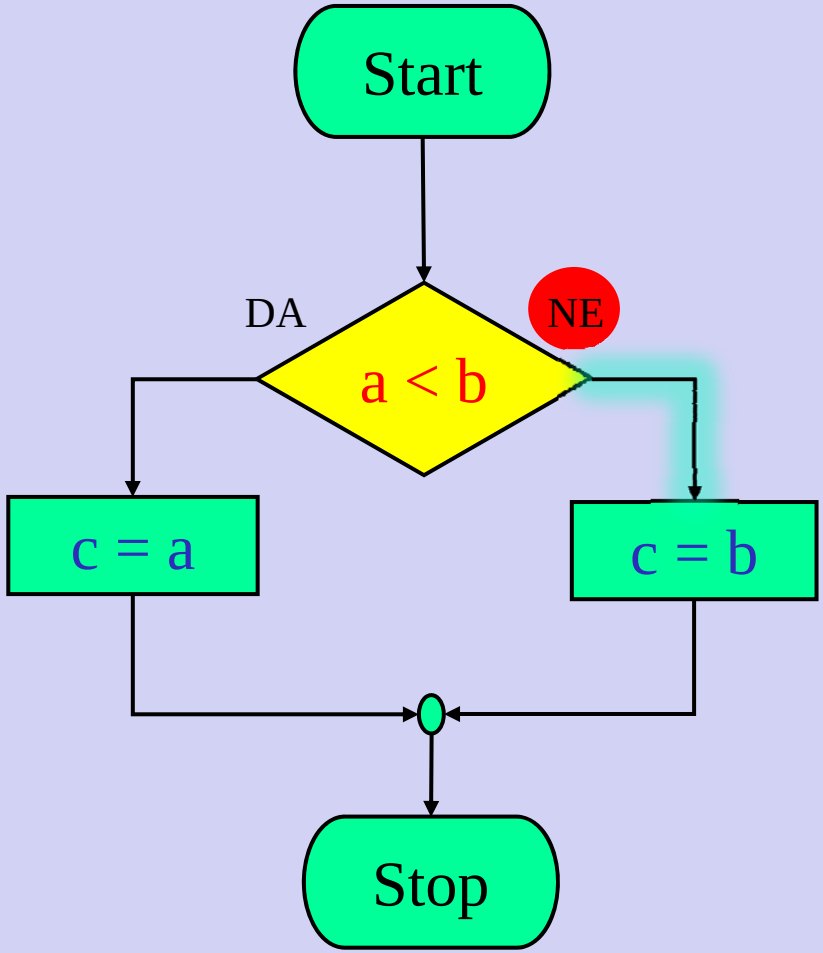
c = 1

Praktični primer: Stavek *if-else*

Program:

```
if(a<b)
{
    c=a;
}
else
{
    c=b;
}
```

Diagram poteka:



Spremenljivke:

a = 7

b = 3

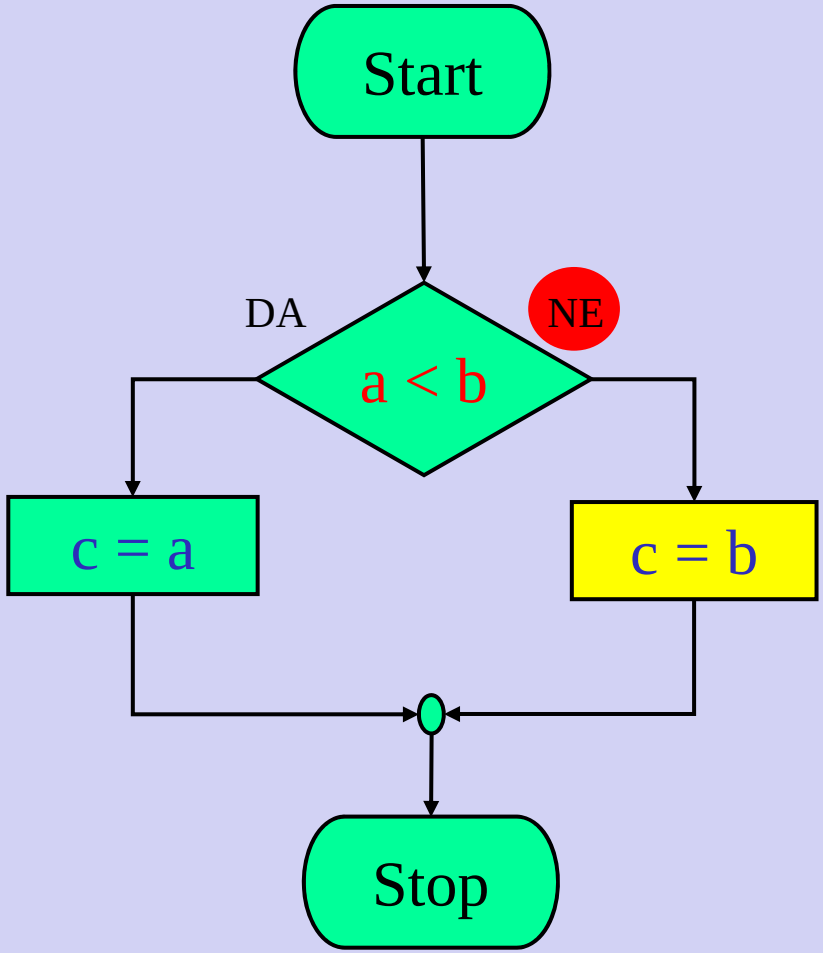
c = 1

Praktični primer: Stavek *if-else*

Program:

```
if(a<b)
{
    c=a;
}
else
{
    c=b;
}
```

Diagram poteka:



Spremenljivke:

a = 7

b = 3

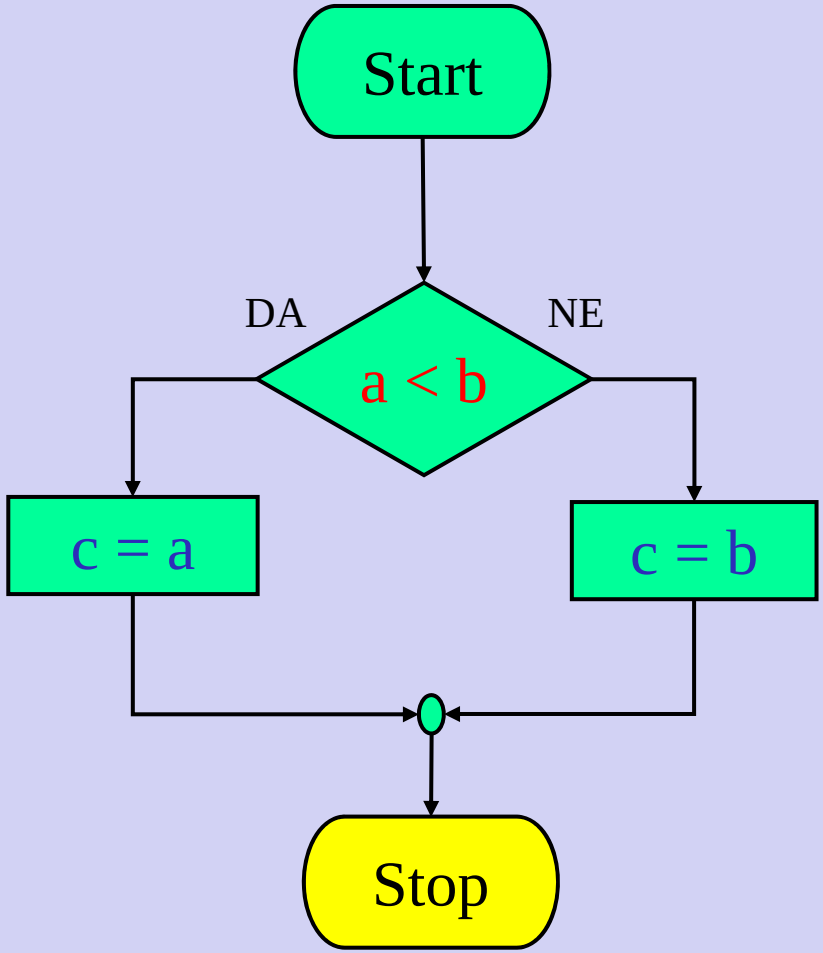
c = 3

Praktični primer: Stavek *if-else*

Program:

```
if(a<b)
{
    c=a;
}
else
{
    c=b;
}
```

Diagram poteka:



Spremenljivke:

a = 7

b = 3

c = 3

a++ in ++a

Program:

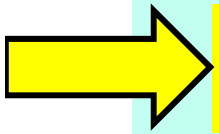
```
c = a++;

d = ++a;
```

Spremenljivke:

```
a = 1;

c = 2;

d = 0;
```

a++ in ++a

Program:



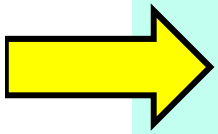
```
c = a++;

d = ++a;
```

Spremenljivke:

```
a = 1;

c = 1;

d = 0;
```

a++ in ++a

Program:



```
c = a++;

d = ++a;
```

Spremenljivke:

```
a = 2;

c = 1;

d = 0;
```

a++ in ++a

Program:

c = a++;

d = ++a;

Spremenljivke:

a = 2;

c = 1;

d = 0;

a++ in ++a

Program:

```
c = a++;

d = ++a;
```

Spremenljivke:

```
a = 3;

c = 1;

d = 0;
```

a++ in ++a

Program:

```
c = a++;

d = ++a;
```

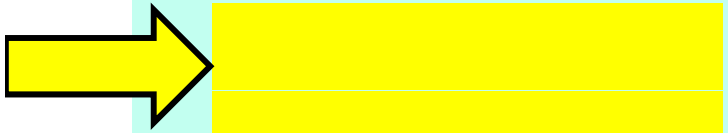
Spremenljivke:

```
a = 3;

c = 1;

d = 3;
```

a++ in ++a

Program:

```
c = a++;

d = ++a;
```



Spremenljivke:

```
a = 3;

c = 1;

d = 3;
```

Primer: Kazalec na strukturo

```
struct Op
{
    int leta;      /* 16 b – dva zloga */
    char spol;     /* 8b – en zlog */
    int otrok;     /* 16b – dva zloga */
}
```

Pomnilnik:

Naslov	Vsebina
1	0
2	100
3	’m’
4	0
5	2

Imamo strukturo Op (osebni podatki), ki vsebuje več podatkov o osebi.

Primer: Kazalec na strukturo

```
struct Op
{
    int leta;      /* 16 b – dva zloga */
    char spol;     /* 8b – en zlog */
    int otrok;     /* 16b – dva zloga */
}
```

Pomnilnik:

Naslov	Vsebina
1	0
2	100
3	'm'
4	0
5	2

leta – starost osebe – zajema 16b (dve mesti v pomnilniku)

Primer: Kazalec na strukturo

```
struct Op
{
    int leta;      /* 16 b – dva zloga */
    char spol;     /* 8b – en zlog */
    int otrok;     /* 16b – dva zloga */
}
```

Pomnilnik:

Naslov	Vsebina
1	0
2	100
3	'm'
4	0
5	2

spol – spol osebe – zajema 8b (eno mesto v pomnilniku)

Primer: Kazalec na strukturo

```
struct Op
{
    int leta;      /* 16 b – dva zloga */
    char spol;     /* 8b – en zlog */
    int otrok;     /* 16b – dva zloga */
}
```

Pomnilnik:

Naslov	Vsebina
1	0
2	100
3	'm'
4	0
5	2

otrok – število otrok osebe – zajema 16b (dve mesti v pomnilniku)

Primer: Kazalec na strukturo

```
Struct Op str;

...
zamenjaj_str(&str);
...

void zamenjaj_str(struct Op *op)
{
    op->otrok = 3;
}
```

str.leta = 100
str.spol = 'm'
str.otrok = 2

Pomnilnik:

Naslov	Vsebina
1	0
2	100
3	'm'
4	0
5	2

Primer: Kazalec na strukturo

```
Struct Op str;

…
zamenjaj_str(&str);
…

void zamenjaj_str(struct Op *op)
{
    op->otrok = 3;
}
```

str.leta = 100
str.spol = 'm'
str.otrok = 2

&str = 1

Pomnilnik:

Naslov	Vsebina
1	0
2	100
3	'm'
4	0
5	2

Primer: Kazalec na strukturo

```
Struct Op str;

…
zamenjaj_str(&str);
…

void zamenjaj_str(struct Op *op)
{
    op->otrok = 3;
}
```

str.leta = 100
str.spol = 'm'
str.otrok = 2

&str = 1

Pomnilnik:

Naslov	Vsebina
1	0
2	100
3	'm'
4	0
5	2

Primer: Kazalec na strukturo

```
Struct Op str;

…
zamenjaj_str(&str);
…

void zamenjaj_str(struct Op *op)
{
    op->otrok = 3;
}
```

str.leta = 100
str.spol = 'm'
str.otrok = 2

op = 1

Pomnilnik:

Naslov	Vsebina
1	0
2	100
3	'm'
4	0
5	2

Primer: Kazalec na strukturo

```
Struct Op str;

…
zamenjaj_str(&str);
…

void zamenjaj_str(struct Op *op)
{
    op->otrok = 3;
}
```

str.leta = 100
str.spol = 'm'
str.otrok = 2

&(op.otrok) = 4

Pomnilnik:

Naslov	Vsebina
1	0
2	100
3	'm'
4	0
5	2

Primer: Kazalec na strukturo

```
Struct Op str;

…
zamenjaj_str(&str);
…

void zamenjaj_str(struct Op *op)
{
    op->otrok = 3;
}
```

str.leta = 100
str.spol = 'm'
str.otrok = 3

op->otrok = 3

Pomnilnik:

Naslov	Vsebina
1	0
2	100
3	'm'
4	0
5	3

Primer: Kazalec na strukturo

```
Struct Op str;

…
zamenjaj_str(&str);
…

void zamenjaj_str(struct Op *op)
{
    op->otrok = 3;
}
```

op->otrok = 3

str.leta = 100
str.spol = 'm'
str.otrok = 3

Pomnilnik:

Naslov	Vsebina
1	0
2	100
3	'm'
4	0
5	3

Primer: Kazalec na strukturo

```
Struct Op str;


…
zamenjaj_str(&str);
…


void zamenjaj_str(struct Op *op)
{
    op->otrok = 3;
}
```

str.leta = 100
str.spol = 'm'
str.otrok = 3

str.otrok = 3

Pomnilnik:

Naslov	Vsebina
1	0
2	100
3	'm'
4	0
5	3

Bitne operacije: Postavljanje bita

Program:

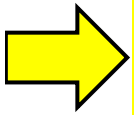
```
a |= (1<<b);
```

Razlaga:

Postavljanje bita št. b
(na primeru a=53, b=3)

Bitne operacije: Postavljanje bita

Program:



```
a |= (1<<b);
```

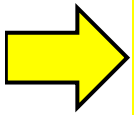
Spremenljivke:

a = 53

b = 3

Bitne operacije: Postavljanje bita

Program:



```
a |= (1<<b);
```

Spremenljivke:

a	0	0	1	1	0	1	0	1

b = 3

Bitne operacije: Postavljanje bita

Program:

```
a |= (1<<b);
```

Spremenljivke:

a	0	0	1	1	1	1	0	1

b = 3

Bitne operacije: Postavljanje bita

Program:

```
a |= (1<<b);
```

Spremenljivke:

a	0	0	1	1	1	1	0	1

b = 3

Bitne operacije: Postavljanje bita

Opis po korakih s programom:

```
c = 1<<b;

e = a|c;

a = e;
```

Bitne operacije: Postavljanje bita

Program:

```
c = 1<<b;
e = a|c;
a = e;
```

Spremenljivke:

a = 53

b = 3

c = 0

e = 0

Bitne operacije: Postavljanje bita

Program:

```
c = 1<<b;
e = a|c;
a = e;
```

Spremenljivke:

a	0	0	1	1	0	1	0	1
c	0	0	0	0	0	0	0	0
e	0	0	0	0	0	0	0	0

b = 3

Bitne operacije: Postavljanje bita

Program:

```
c = 1<<b;
e = a|c;
a = e;
```

Spremenljivke:

a	0	0	1	1	0	1	0	1
c	0	0	0	0	1	0	0	0
e	0	0	0	0	0	0	0	0

b = 3

Bitne operacije: Postavljanje bita

Program:

```
c = 1<<b;
e = a|c;
a = e;
```

Spremenljivke:

a	0	0	1	1	0	1	0	1
c	0	0	0	0	1	0	0	0
e	0	0	1	1	1	1	0	1

b = 3

Bitne operacije: Postavljanje bita

Program:

```
c = 1<<b;
e = a|c;
a = e;
```

Spremenljivke:

a	0	0	1	1	1	1	0	1
c	0	0	0	0	1	0	0	0
e	0	0	1	1	1	1	0	1

b = 3

Bitne operacije: Postavljanje bita

Program:

```
c = 1<<b;
e = a|c;
a = e;
```

Spremenljivke:

a	0	0	1	1	1	1	0	1
c	0	0	0	0	1	0	0	0
e	0	0	1	1	1	1	0	1

b = 3

Bitne operacije: Postavljanje bita

Program:

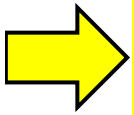
```
a |= (1<<b);
```

Razlaga:

Postavljanje bita št. 2
(na primeru a=53, b=2)

Bitne operacije: Postavljanje bita

Program:



```
a |= (1<<b);
```

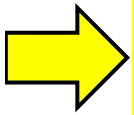
Spremenljivke:

a = 53

b = 2

Bitne operacije: Postavljanje bita

Program:



```
a |= (1<<b);
```

Spremenljivke:

a	0	0	1	1	0	1	0	1

b = 2

Bit 2 je že postavljen!

Bitne operacije: Postavljanje bita

Program:

```
a |= (1<<b);
```

Spremenljivke:

a	0	0	1	1	0	1	0	1

b = 2

Bitne operacije: Postavljanje bita

Program:

```
a |= (1<<b);
```

Spremenljivke:

a	0	0	1	1	0	1	0	1

b = 2

Bit 2 se ni spremenil!

Prireditev: c=a+b in a=a+1

Program:

c = a+b;

a = a+1;

Spremenljivke:

a = 3;

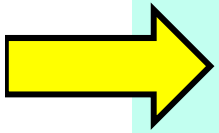
b = 2;

c = 1;

Rezultat: ?

Prireditev: c=a+b in a=a+1

Program:



c = a+b;

a = a+1;

Spremenljivke:

a = 3;

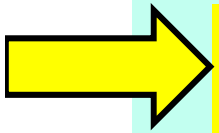
b = 2;

c = 1;

Rezultat: 5

Prireditev: c=a+b in a=a+1

Program:



c = a+b;

a = a+1;

Spremenljivke:

a = 3;

b = 2;

c = 5;

Rezultat: 5

Prireditev: c=a+b in a=a+1

Program:

```
c = a+b;

a = a+1;
```

Spremenljivke:

```
a = 3;

b = 2;

c = 5;
```

Rezultat: 5

Prireditev: c=a+b in a=a+1

Program:

```
c = a+b;

a = a+1;
```

Spremenljivke:

```
a = 3;
b = 2;
c = 5;
```

Rezultat: 4

Prireditev: c=a+b in a=a+1

Program:

```
c = a+b;

a = a+1;
```

Spremenljivke:

a = 4;

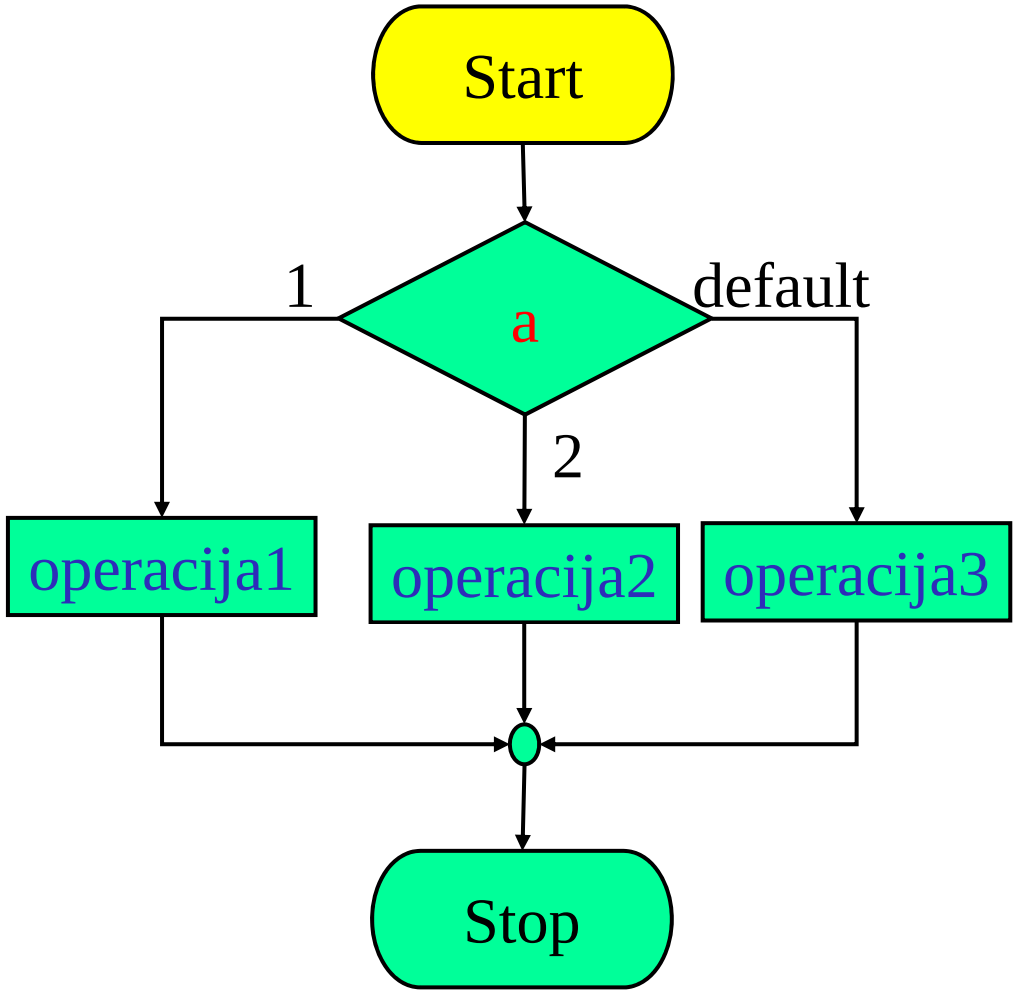
b = 2;

c = 5;

Rezultat: 4

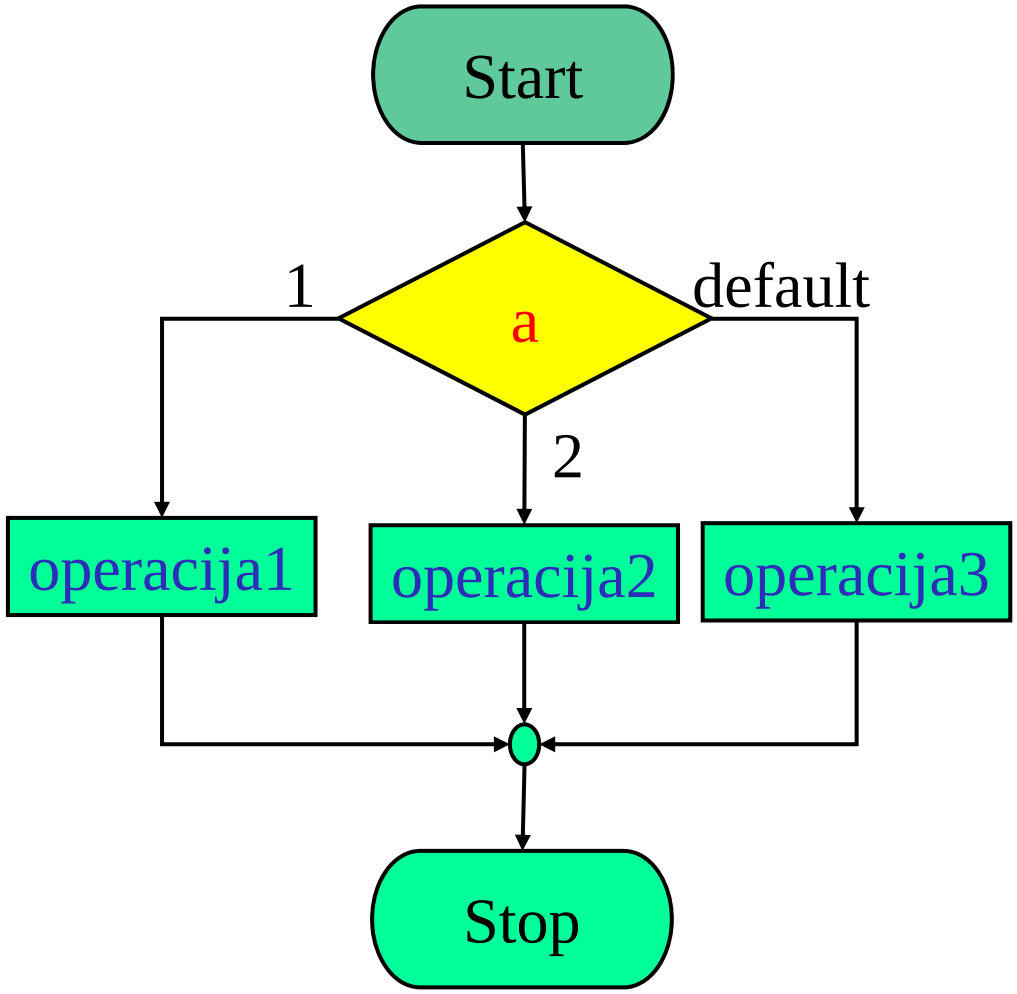
Primer: Stavek *switch* (a=1)

```
switch(a)
{
    case 1:
        operacija1;
        break;
    case 2:
        operacija2;
        break;
    default:
        operacija3;
        break;
}
```



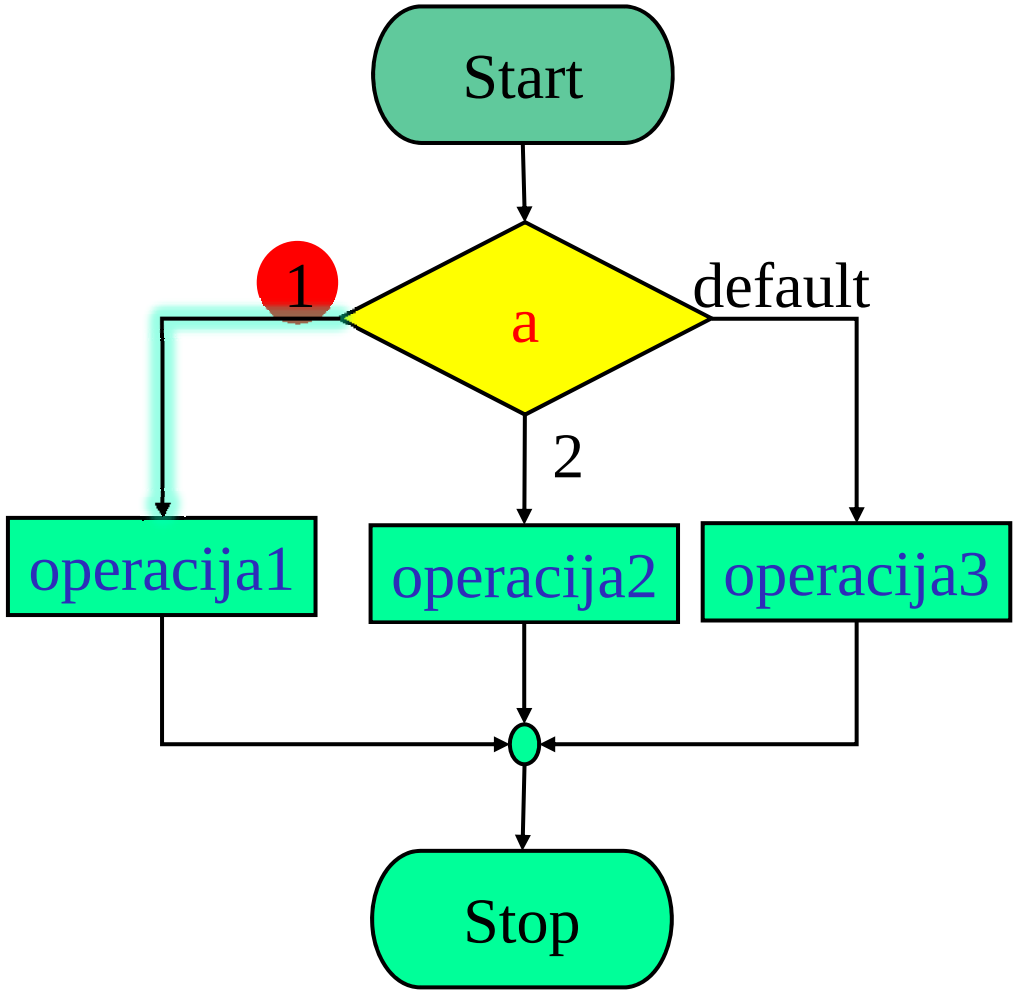
Primer: Stavek *switch* (a=1)

```
switch(a)
{
    case 1:
        operacija1;
        break;
    case 2:
        operacija2;
        break;
    default:
        operacija3;
        break;
}
```



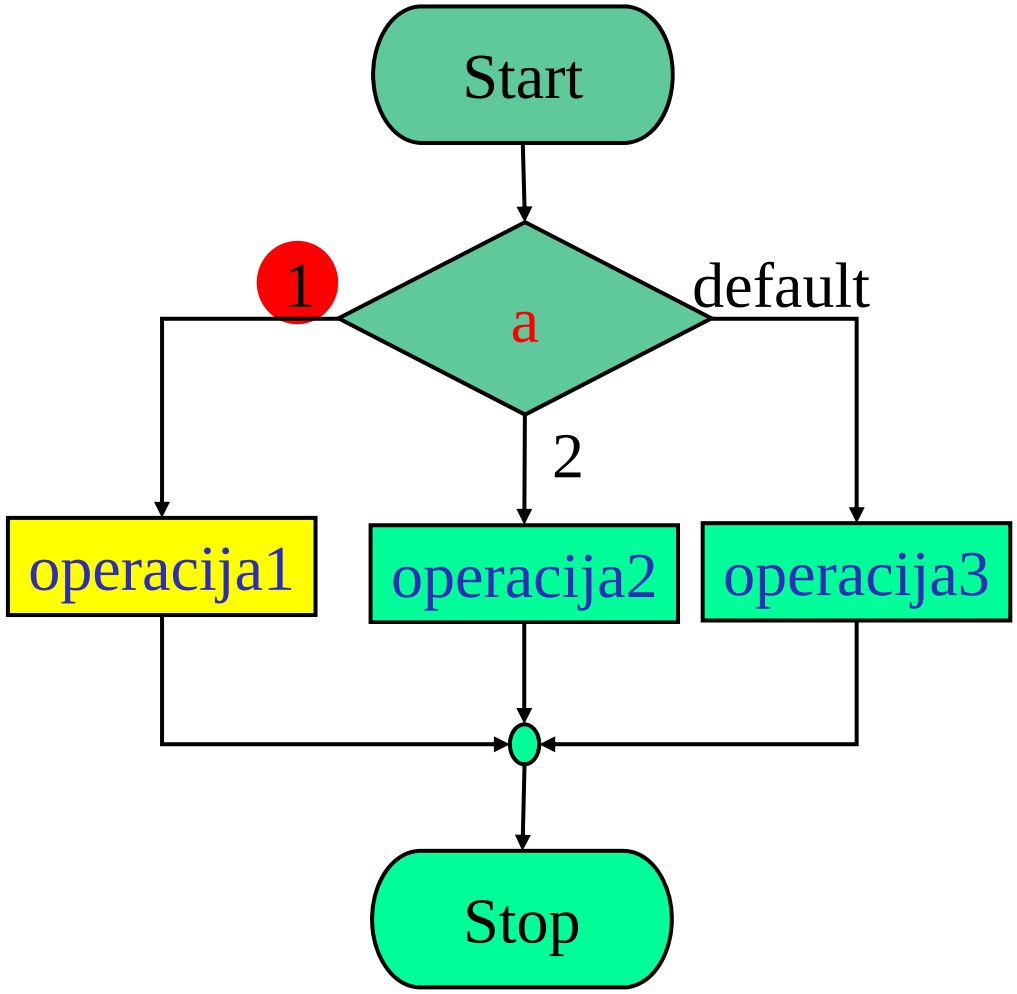
Primer: Stavek *switch* (a=1)

```
switch(a)
{
    case 1:
        operacija1;
        break;
    case 2:
        operacija2;
        break;
    default:
        operacija3;
        break;
}
```



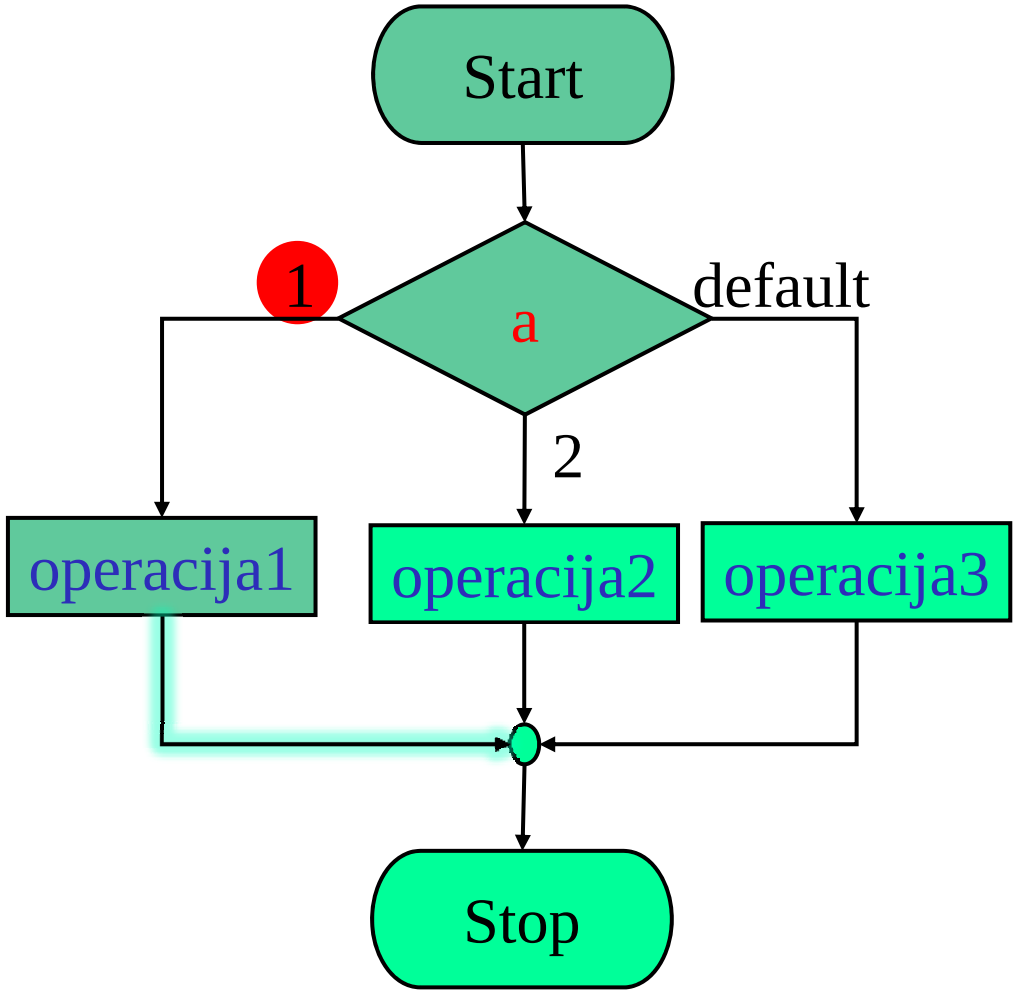
Primer: Stavek *switch* (a=1)

```
switch(a)
{
    case 1:
        operacija1;
        break;
    case 2:
        operacija2;
        break;
    default:
        operacija3;
        break;
}
```



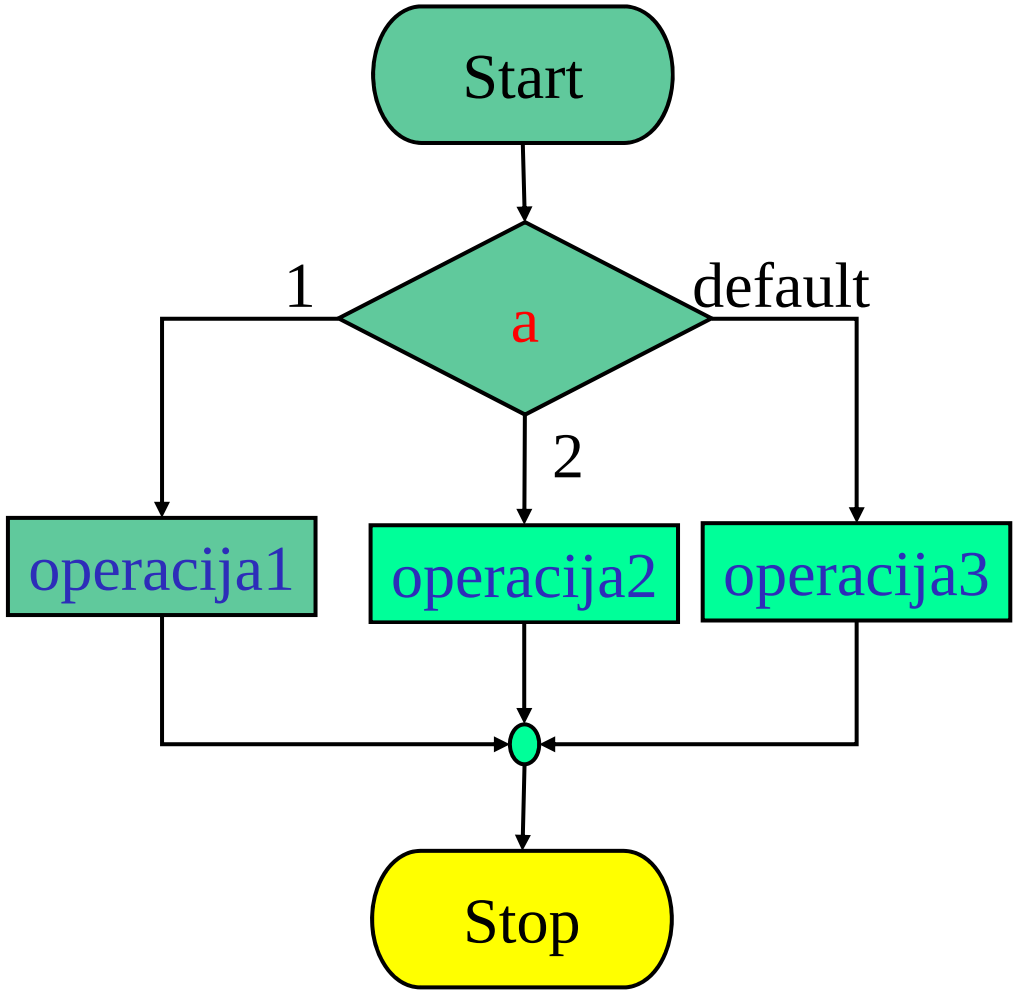
Primer: Stavek *switch* (a=1)

```
switch(a)
{
    case 1:
        operacija1;
        break;
    case 2:
        operacija2;
        break;
    default:
        operacija3;
        break;
}
```



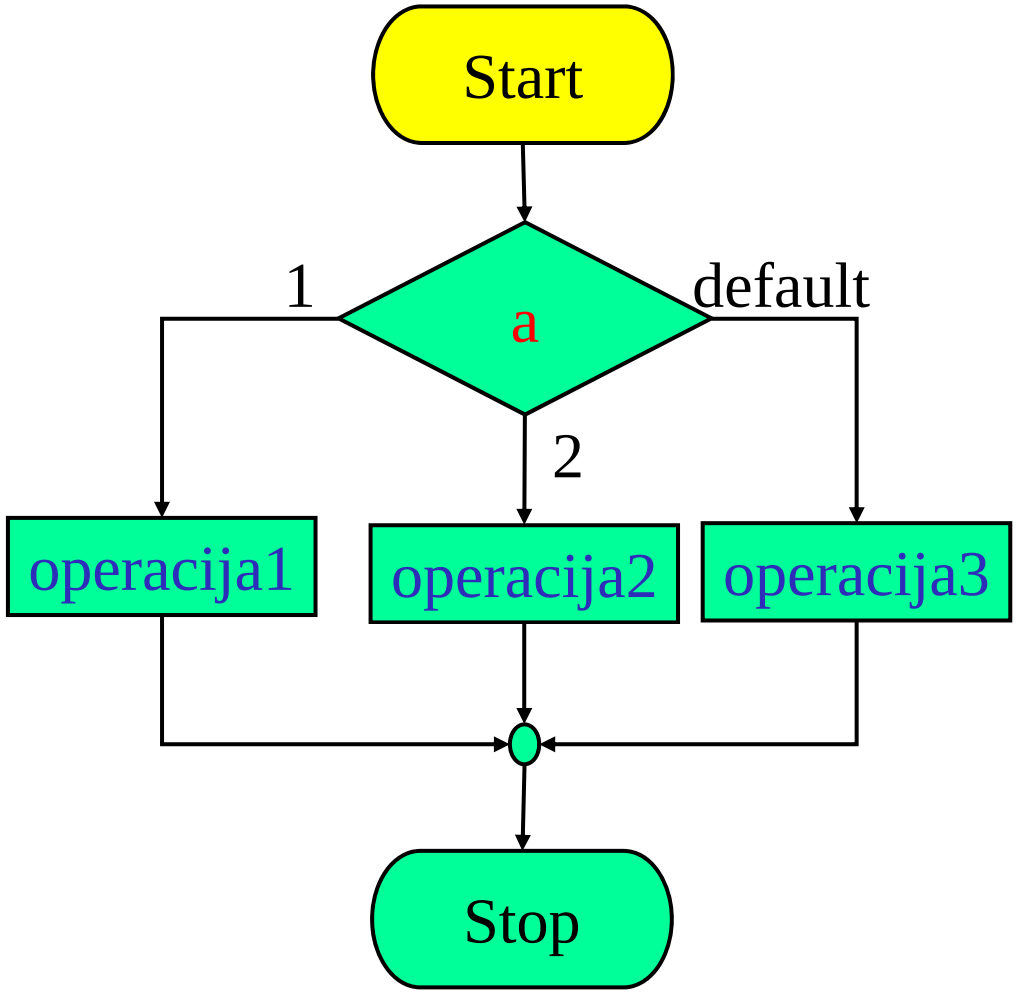
Primer: Stavek *switch* (a=1)

```
switch(a)
{
    case 1:
        operacija1;
        break;
    case 2:
        operacija2;
        break;
    default:
        operacija3;
        break;
}
```



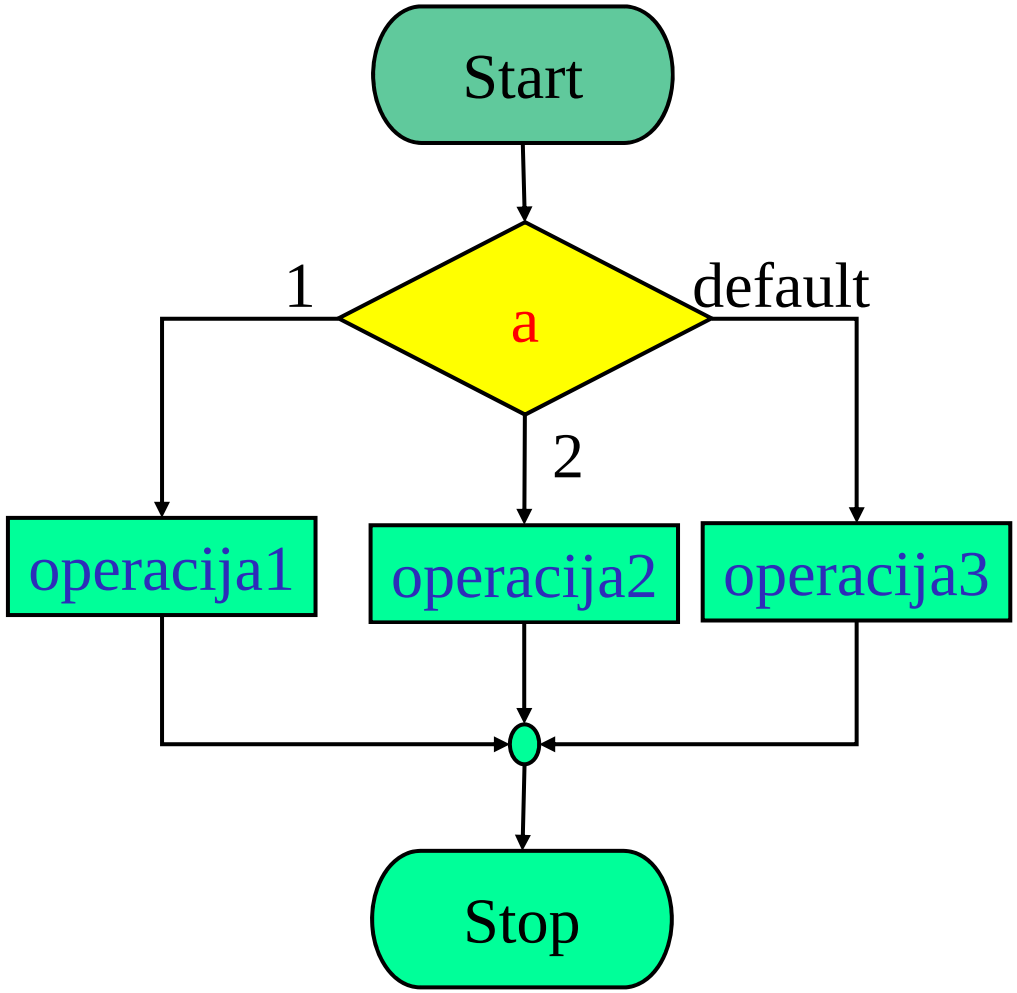
Primer: Stavek *switch* (a=2)

```
switch(a)
{
    case 1:
        operacija1;
        break;
    case 2:
        operacija2;
        break;
    default:
        operacija3;
        break;
}
```



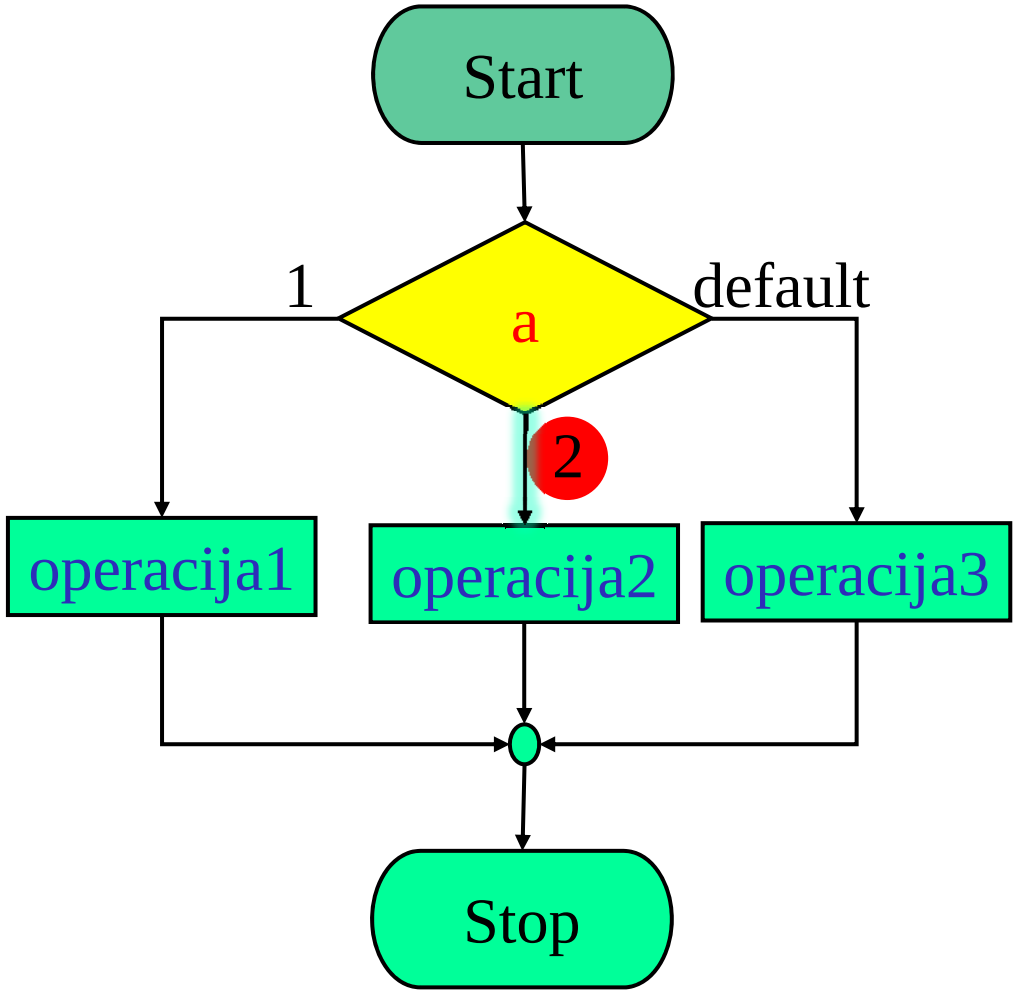
Primer: Stavek *switch* (a=2)

```
switch(a)
{
    case 1:
        operacija1;
        break;
    case 2:
        operacija2;
        break;
    default:
        operacija3;
        break;
}
```



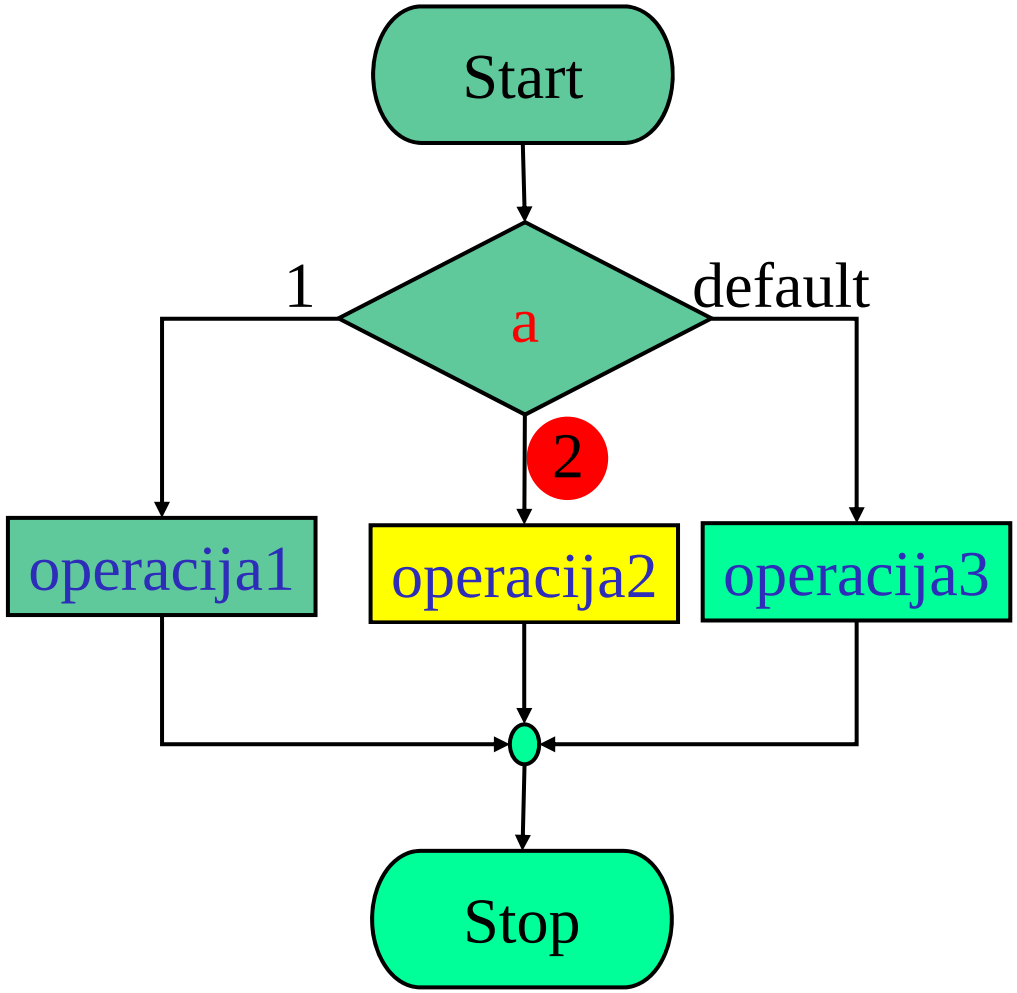
Primer: Stavek *switch* (a=2)

```
switch(a)
{
    case 1:
        operacija1;
        break;
    case 2:
        operacija2;
        break;
    default:
        operacija3;
        break;
}
```



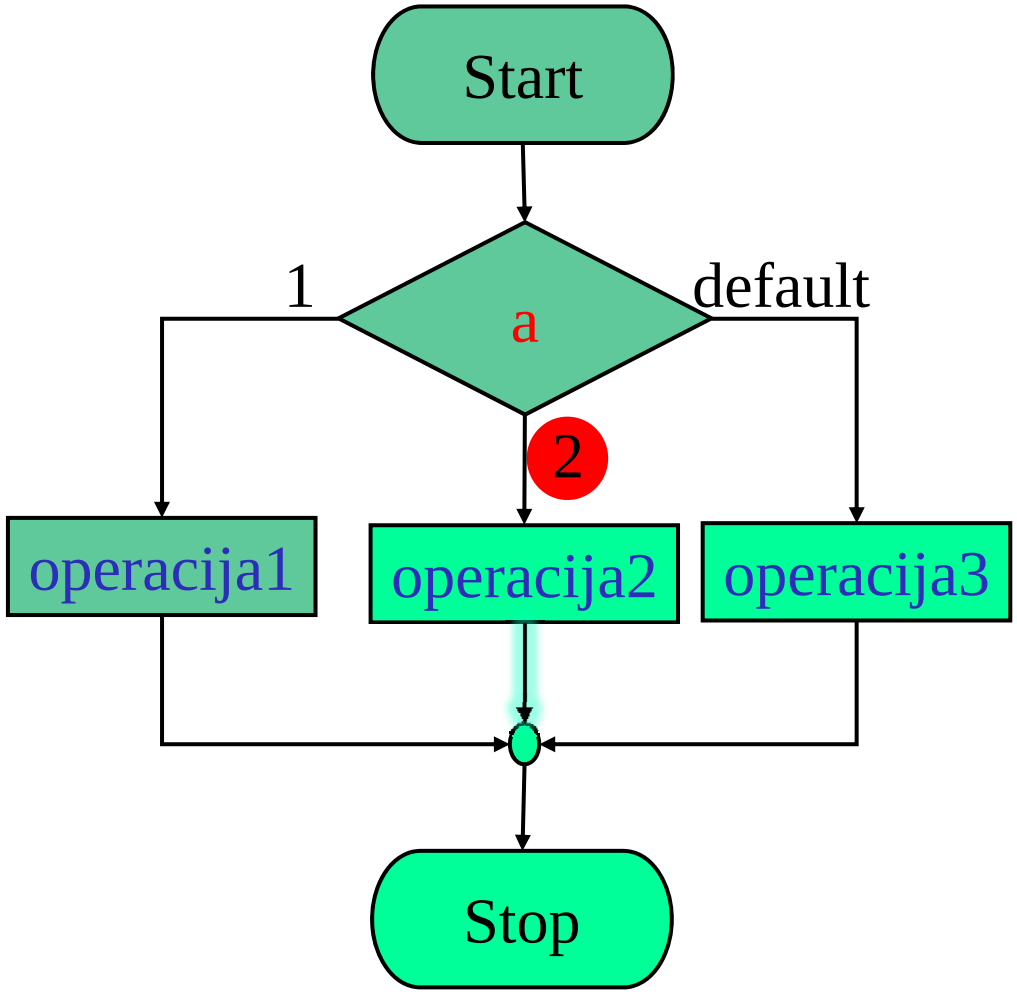
Primer: Stavek *switch* (a=2)

```
switch(a)
{
    case 1:
        operacija1;
        break;
    case 2:
        operacija2;
        break;
    default:
        operacija3;
        break;
}
```



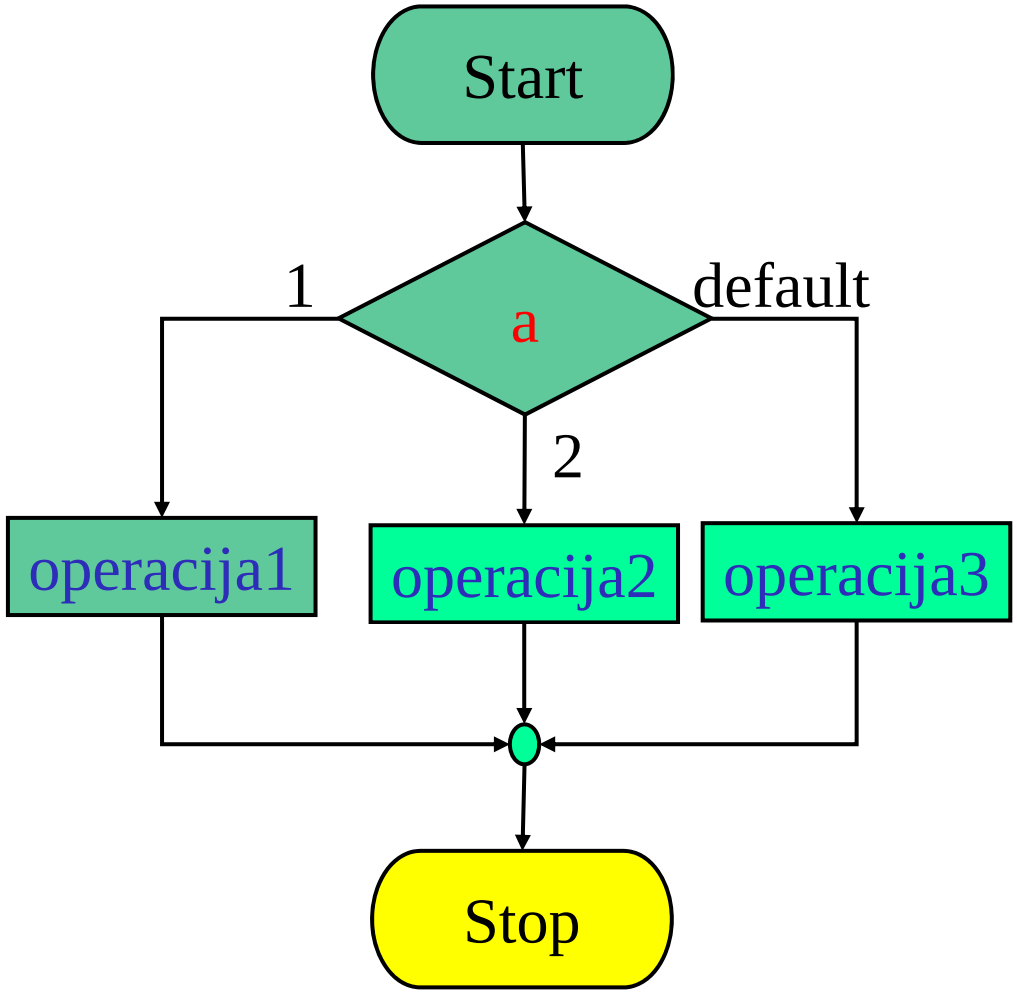
Primer: Stavek *switch* (a=2)

```
switch(a)
{
    case 1:
        operacija1;
        break;
    case 2:
        operacija2;
        break;
    default:
        operacija3;
        break;
}
```



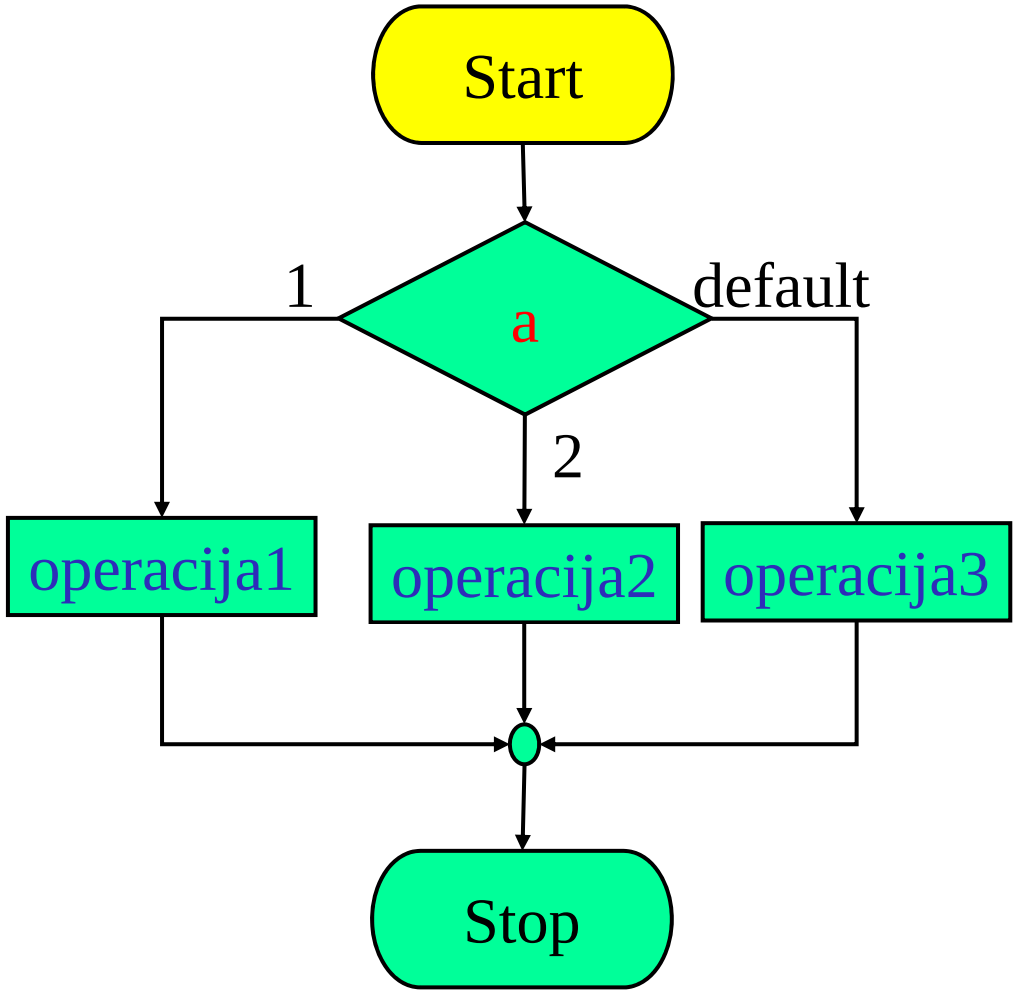
Primer: Stavek *switch* (a=2)

```
switch(a)
{
    case 1:
        operacija1;
        break;
    case 2:
        operacija2;
        break;
    default:
        operacija3;
        break;
}
```



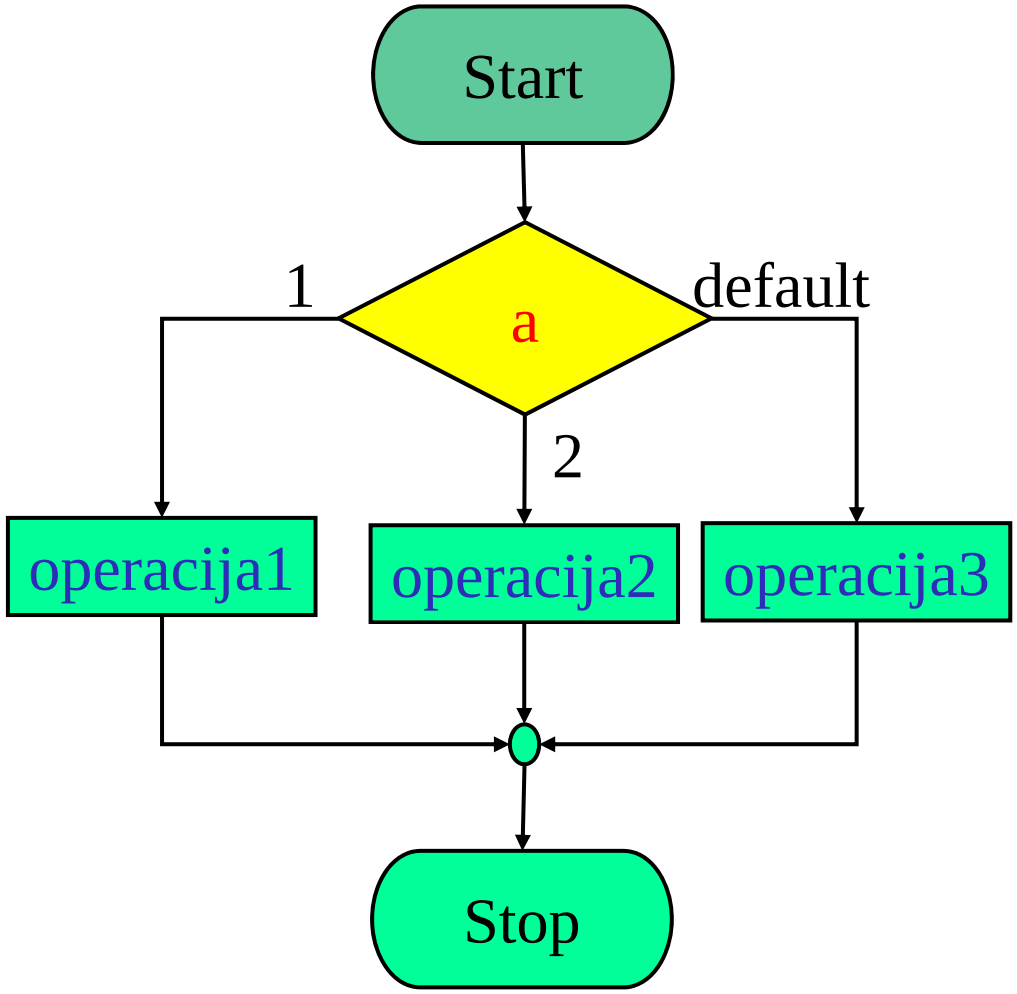
Primer: Stavek *switch* (a≠1 in a≠2)

```
switch(a)
{
    case 1:
        operacija1;
        break;
    case 2:
        operacija2;
        break;
    default:
        operacija3;
        break;
}
```



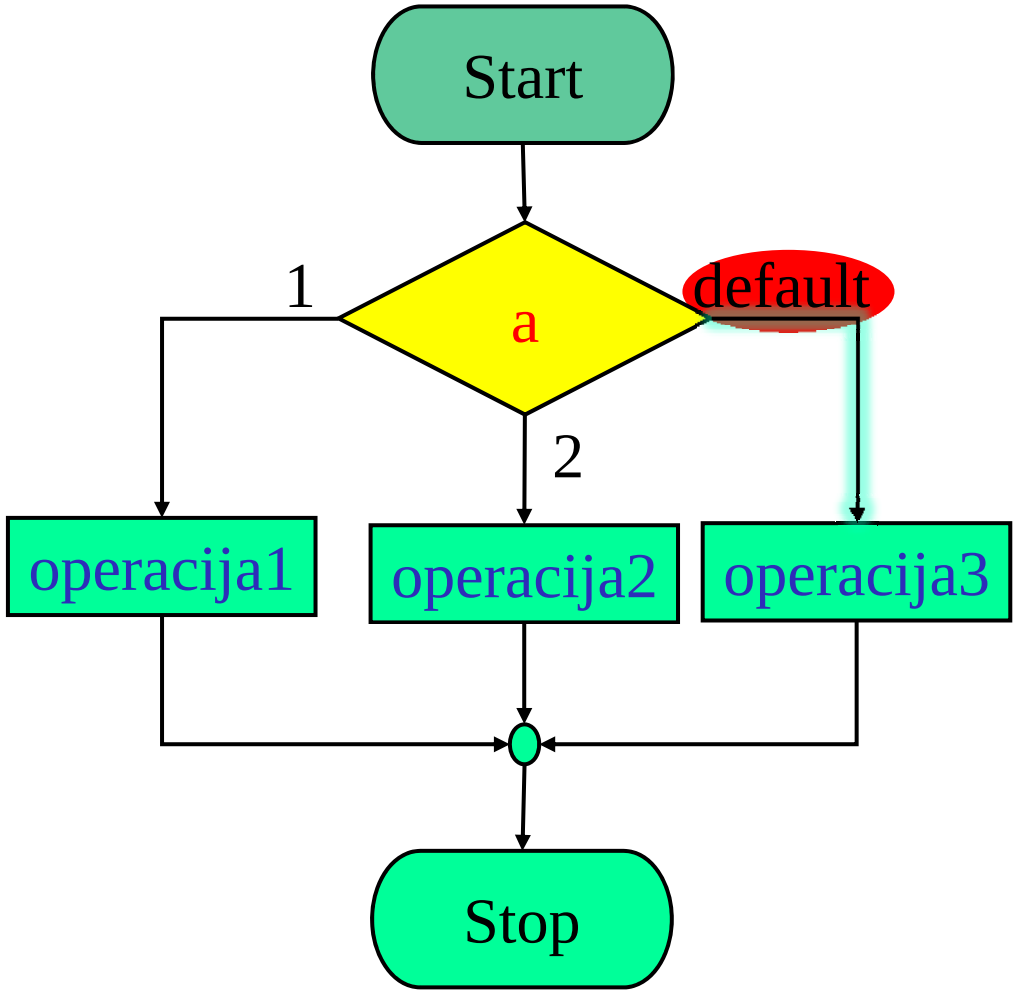
Primer: Stavek *switch* (a≠1 in a≠2)

```
switch(a)
{
    case 1:
        operacija1;
        break;
    case 2:
        operacija2;
        break;
    default:
        operacija3;
        break;
}
```



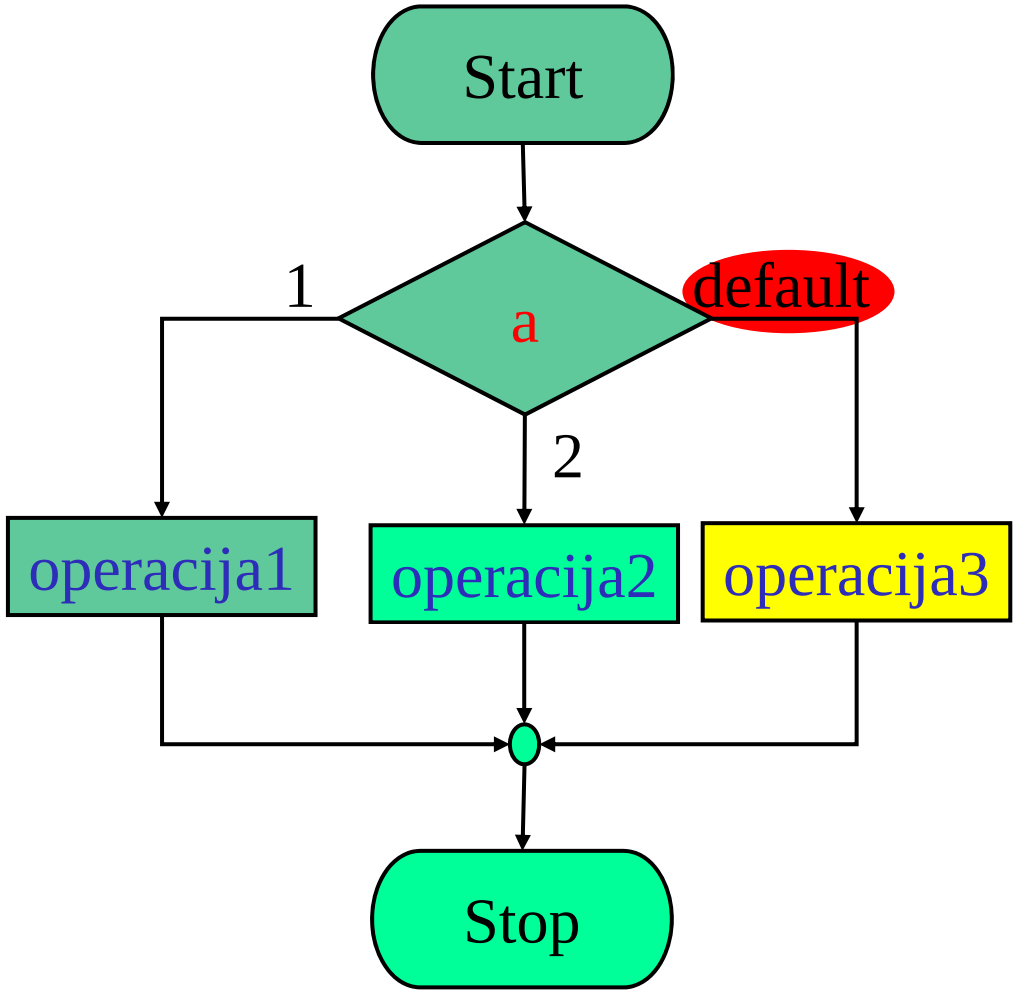
Primer: Stavek *switch* (a≠1 in a≠2)

```
switch(a)
{
    case 1:
        operacija1;
        break;
    case 2:
        operacija2;
        break;
    default:
        operacija3;
        break;
}
```



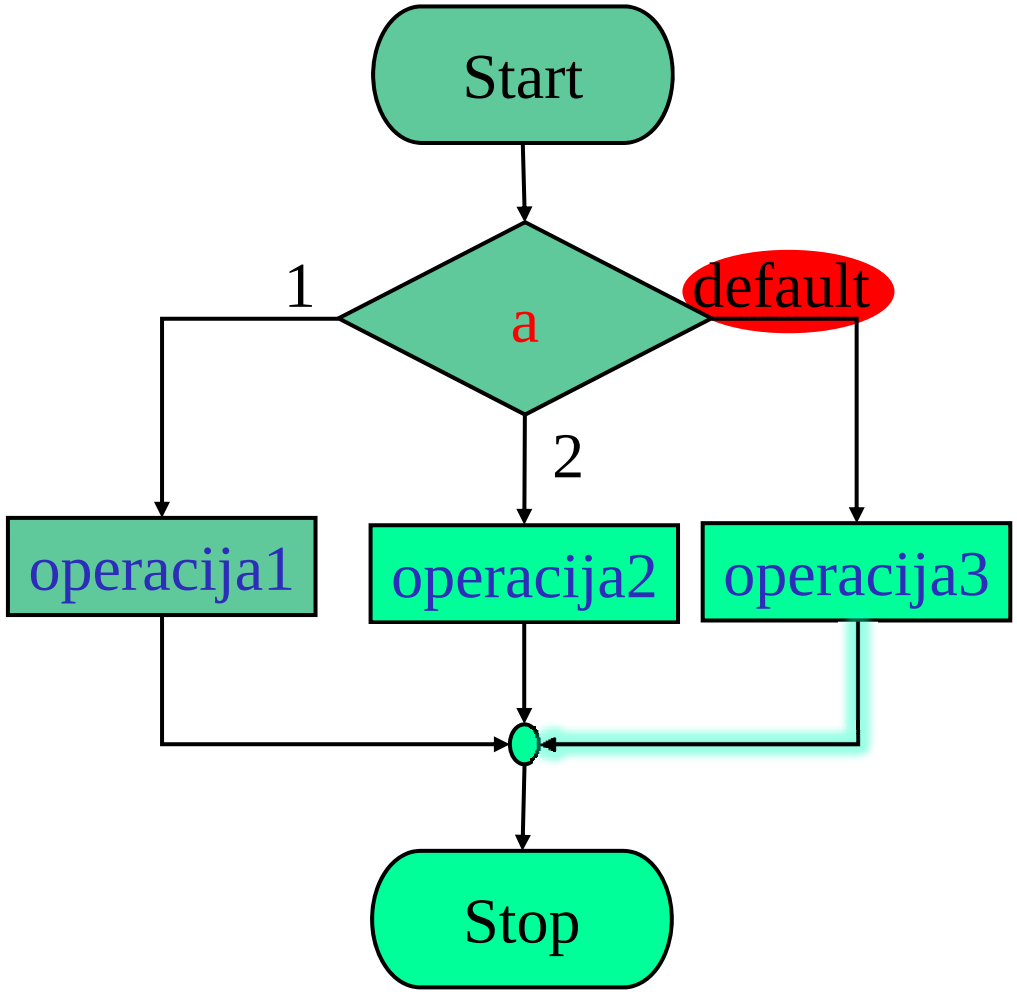
Primer: Stavek *switch* (a≠1 in a≠2)

```
switch(a)
{
    case 1:
        operacija1;
        break;
    case 2:
        operacija2;
        break;
    default:
        operacija3;
        break;
}
```



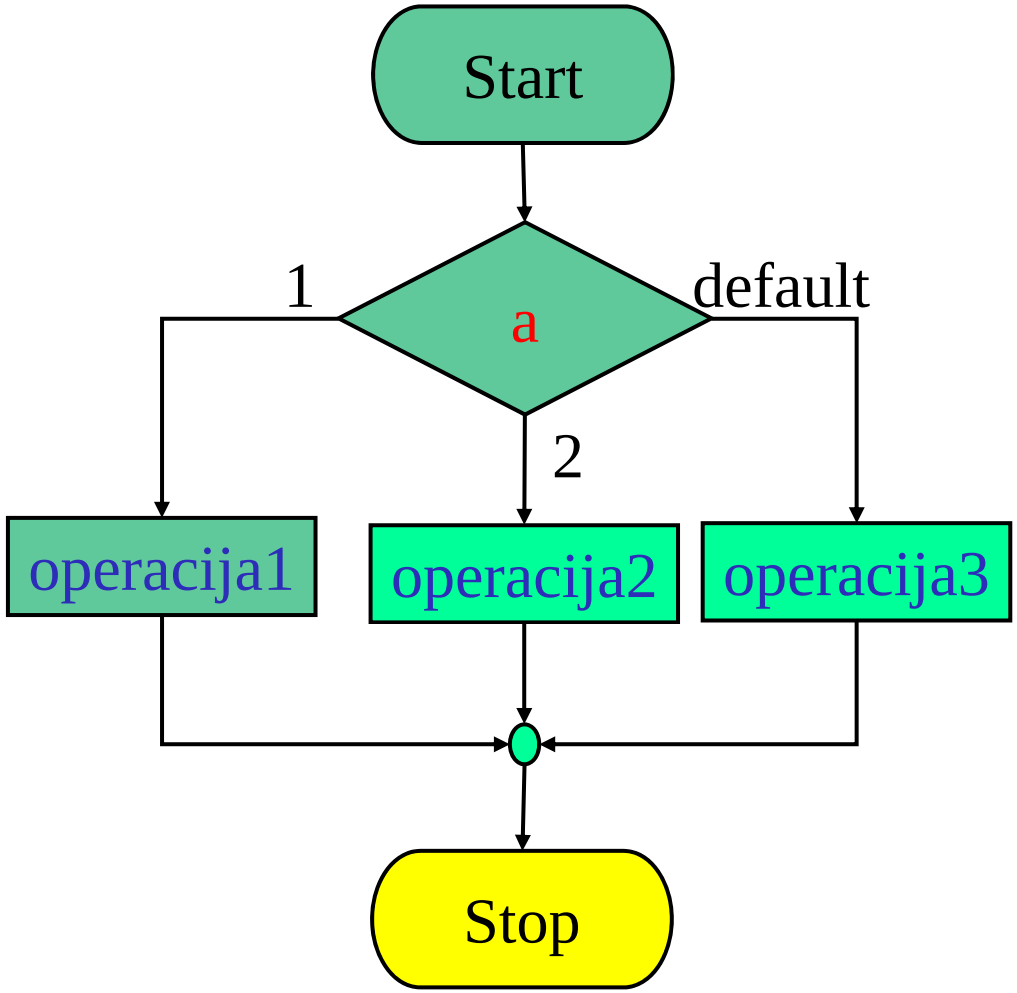
Primer: Stavek *switch* (a≠1 in a≠2)

```
switch(a)
{
    case 1:
        operacija1;
        break;
    case 2:
        operacija2;
        break;
    default:
        operacija3;
        break;
}
```



Primer: Stavek *switch* (a≠1 in a≠2)

```
switch(a)
{
    case 1:
        operacija1;
        break;
    case 2:
        operacija2;
        break;
    default:
        operacija3;
        break;
}
```



Primer: Stavek *switch* brez *break* (a=1)

```
switch(a)
{
    case 1:
        operacija1;
    case 2:
        operacija2;
    default:
        operacija3;
}
```

Če ni ukaza break, jemlje program vse nadaljnje *case* (in *default*) za izpolnjene.

Primer: Stavek *switch* brez *break* (a=1)

```
switch(a)
{
    case 1:
        operacija1;
    case 2:
        operacija2;
    default:
        operacija3;
}
```

Če ni ukaza break, jemlje program vse nadaljnje *case* (in *default*) za izpolnjene.

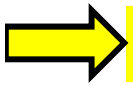
Primer: Stavek *switch* brez *break* (a=1)

```
switch(a)
{
    case 1:
        operacija1;
    case 2:
        operacija2;
    default:
        operacija3;
}
```

Če ni ukaza break, jemlje program vse nadaljnje *case* (in *default*) za izpolnjene.

Primer: Stavek *switch* brez *break* (a=1)

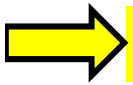
```
switch(a)
{
    case 1:
        operacija1;
    case 2:
        operacija2;
    default:
        operacija3;
}
```



Če ni ukaza break, jemlje program vse nadaljnje *case* (in *default*) za izpolnjene.

Primer: Stavek *switch* brez *break* (a=1)

```
switch(a)
{
    case 1:
        operacija1;
    case 2:
        operacija2;
    default:
        operacija3;
}
```



Če ni ukaza break, jemlje program vse nadaljnje *case* (in *default*) za izpolnjene.

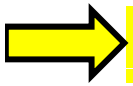
Primer: Stavek *switch* brez *break* (a=1)

```
switch(a)
{
    case 1:
        operacija1;
    case 2:
        operacija2;
    default:
        operacija3;
}
```

Če ni ukaza break, jemlje program vse nadaljnje *case* (in *default*) za izpolnjene.

Primer: Stavek *switch* brez *break* (a=1)

```
switch(a)
{
    case 1:
        operacija1;
    case 2:
        operacija2;
    default:
        operacija3;
}
```



Če ni ukaza break, jemlje program vse nadaljnje *case* (in *default*) za izpolnjene.

Primer: Stavek *switch* brez *break* (a=1)

```
switch(a)
{
    case 1:
        b = b + 1;
    case 2:
        c = c + 1;
    default:
        d = d + 1;
}
```

a = 1

b = 0

c = 0

d = 0

Primer: Stavek *switch* brez *break* (a=1)

```
switch(a)
{
    case 1:
        b = b + 1;
    case 2:
        c = c + 1;
    default:
        d = d + 1;
}
```

a = 1

b = 0

c = 0

d = 0

Primer: Stavek *switch* brez *break* (a=1)

```
switch(a)
{
    case 1:
        b = b + 1;
    case 2:
        c = c + 1;
    default:
        d = d + 1;
}
```

a = 1

b = 0

c = 0

d = 0

Primer: Stavek *switch* brez *break* (a=1)

```
switch(a)
{
    case 1:
        b = b + 1;
    case 2:
        c = c + 1;
    default:
        d = d + 1;
}
```

a = 1

b = 1

c = 0

d = 0

Primer: Stavek *switch* brez *break* (a=1)

```
switch(a)
{
    case 1:
        b = b + 1;
    case 2:
        c = c + 1;
    default:
        d = d + 1;
}
```

a = 1

b = 1

c = 1

d = 0

Primer: Stavek *switch* brez *break* (a=1)

```
switch(a)
{
    case 1:
        b = b + 1;
    case 2:
        c = c + 1;
    default:
        d = d + 1;
}
```

a = 1

b = 1

c = 1

d = 1

Primer: Stavek *switch* brez *break* (a=1)

```
switch(a)
{
    case 1:
        b = b + 1;
    case 2:
        c = c + 1;
    default:
        d = d + 1;
}
```

a = 1

b = 1

c = 1

d = 1

Primer: Stavek *switch* brez *break* (c='a')

```
switch(c)
{
    case 'a':
    case 'A':
        operacija1;
        break;
    case 'b':
    case 'B':
        operacija2;
        break;
    default:
        operacija3;
        break;
}
```

To je mogoče tudi koristno uporabiti, ko imamo za več možnosti enake operacije.

Primer: Stavek *switch* brez *break* (c='a')

```
switch(c)
{
    case 'a':
    case 'A':
        operacija1;
        break;
    case 'b':
    case 'B':
        operacija2;
        break;
    default:
        operacija3;
        break;
}
```

To je mogoče tudi koristno uporabiti, ko imamo za več možnosti enake operacije.

Primer: Stavek *switch* brez *break* (c='a')

```
switch(c)
{
    case 'a':
    case 'A':
        operacija1;
        break;
    case 'b':
    case 'B':
        operacija2;
        break;
    default:
        operacija3;
        break;
}
```

To je mogoče tudi koristno uporabiti, ko imamo za več možnosti enake operacije.

Primer: Stavek *switch* brez *break* (c='a')

```
switch(c)
{
    case 'a':
    case 'A':
        operacija1;
        break;
    case 'b':
    case 'B':
        operacija2;
        break;
    default:
        operacija3;
        break;
}
```

To je mogoče tudi koristno uporabiti, ko imamo za več možnosti enake operacije.

Primer: Stavek *switch* brez *break* (c='a')

```
switch(c)
{
    case 'a':
    case 'A':
        operacija1;
        break;
    case 'b':
    case 'B':
        operacija2;
        break;
    default:
        operacija3;
        break;
}
```

To je mogoče tudi koristno uporabiti, ko imamo za več možnosti enake operacije.

Primer: Stavek *switch* brez *break* (c='a')

```
switch(c)
{
    case 'a':
    case 'A':
        operacija1;
        break;
    case 'b':
    case 'B':
        operacija2;
        break;
    default:
        operacija3;
        break;
}
```

To je mogoče tudi koristno uporabiti, ko imamo za več možnosti enake operacije.

Primer: Stavek *switch* brez *break* (c='A')

```
switch(c)
{
    case 'a':
    case 'A':
        operacija1;
        break;
    case 'b':
    case 'B':
        operacija2;
        break;
    default:
        operacija3;
        break;
}
```

To je mogoče tudi koristno uporabiti, ko imamo za več možnosti enake operacije.

Primer: Stavek *switch* brez *break* (c='A')

```
switch(c)
{
    case 'a':
    case 'A':
        operacija1;
        break;
    case 'b':
    case 'B':
        operacija2;
        break;
    default:
        operacija3;
        break;
}
```

To je mogoče tudi koristno uporabiti, ko imamo za več možnosti enake operacije.

Primer: Stavek *switch* brez *break* (c='A')

```
switch(c)
{
    case 'a':
    case 'A':
        operacija1;
        break;
    case 'b':
    case 'B':
        operacija2;
        break;
    default:
        operacija3;
        break;
}
```

To je mogoče tudi koristno uporabiti, ko imamo za več možnosti enake operacije.

Primer: Stavek *switch* brez *break* (c='A')

```
switch(c)
{
    case 'a':
    case 'A':
        operacija1;
        break;
    case 'b':
    case 'B':
        operacija2;
        break;
    default:
        operacija3;
        break;
}
```

To je mogoče tudi koristno uporabiti, ko imamo za več možnosti enake operacije.

Primer: Stavek *switch* brez *break* (c='A')

```
switch(c)
{
    case 'a':
    case 'A':
        operacija1;
        break;
    case 'b':
    case 'B':
        operacija2;
        break;
    default:
        operacija3;
        break;
}
```

To je mogoče tudi koristno uporabiti, ko imamo za več možnosti enake operacije.

Primer: Stavek *switch* brez *break* (c='A')

```
switch(c)
{
    case 'a':
    case 'A':
        operacija1;
        break;
    case 'b':
    case 'B':
        operacija2;
        break;
    default:
        operacija3;
        break;
}
```

To je mogoče tudi koristno uporabiti, ko imamo za več možnosti enake operacije.

Bitne operacije: Testiranje bita

Program:

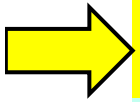
```
if(a & (1<<b))
        Operacija1;
Else
        Operacija2;
```

Razlaga:

Testiranje, če je bit b postavljen.

Bitne operacije: Testiranje bita

Program:



```
if(a & (1<<b))
        Operacija1;
else
        Operacija2;
```

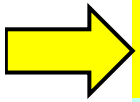
Spremenljivke:

a = 53

b = 2

Bitne operacije: Testiranje bita

Program:



```
if(a & (1<<b))
        Operacija1;
else
        Operacija2;
```

Spremenljivke:

a = 53

b = 2

Bitne operacije: Testiranje bita

Program:

```
if(a & (1<<b))
        Operacija1;
else
        Operacija2;
```

Spremenljivke:

a	0	0	1	1	0	1	0	1

b = 2

Bitne operacije: Testiranje bita

Program:

```
if(a & (1<<b))
        Operacija1;
else
        Operacija2;
```

Spremenljivke:

a	0	0	1	1	0	1	0	1

b = 2

$$(a \;\&\; (1<<b)) \neq 0$$

Bitne operacije: Testiranje bita

Program:

```
if(a & (1<<b))
        Operacija1;
else
        Operacija2;
```

Spremenljivke:

a	0	0	1	1	0	1	0	1

b = 2

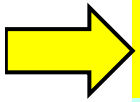
(a & (1<<b)) ≠ 0

Bit 2 je postavljen!

Bitne operacije: Testiranje bita

Program:

```
if(a & (1<<b))
        Operacija1;
else
        Operacija2;
```



Spremenljivke:

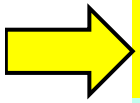
a	0	0	1	1	0	1	0	1

b = 2

Bitne operacije: Testiranje bita

Program:

```
if(a & (1<<b))
        Operacija1;
else
        Operacija2;
```



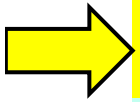
Spremenljivke:

a	0	0	1	1	0	1	0	1

b = 2

Bitne operacije: Testiranje bita

Program:



```
if(a & (1<<b))
        Operacija1;
else
        Operacija2;
```

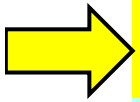
Spremenljivke:

a = 53

b = 3

Bitne operacije: Testiranje bita

Program:



```
if(a & (1<<b))
        Operacija1;
else
        Operacija2;
```

Spremenljivke:

$a = 53$

$b = 3$

Bitne operacije: Testiranje bita

Program:

```
if(a & (1<<b))
        Operacija1;
else
        Operacija2;
```

Spremenljivke:

a	0	0	1	1	0	1	0	1

b = 3

Bitne operacije: Testiranje bita

Program:

```
if(a & (1<<b))
        Operacija1;
else
        Operacija2;
```

Spremenljivke:

a	0	0	1	1	0	1	0	1

b = 3

(a & (1<<b)) = 0

Bitne operacije: Testiranje bita

Program:

```
if(a & (1<<b))
        Operacija1;
else
        Operacija2;
```

Spremenljivke:

a	0	0	1	1	0	1	0	1

b = 3

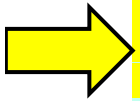
(a & (1<<b)) = 0

Bit 3 ni postavljen!

Bitne operacije: Testiranje bita

Program:

```
if(a & (1<<b))
        Operacija1;
else
        Operacija2;
```



Spremenljivke:

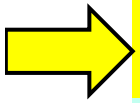
a	0	0	1	1	0	1	0	1

b = 3

Bitne operacije: Testiranje bita

Program:

```
if(a & (1<<b))
        Operacija1;
else
        Operacija2;
```



Spremenljivke:

a	0	0	1	1	0	1	0	1

b = 3

Primer: Vejitev

začni izvajanje

preveri pogoj

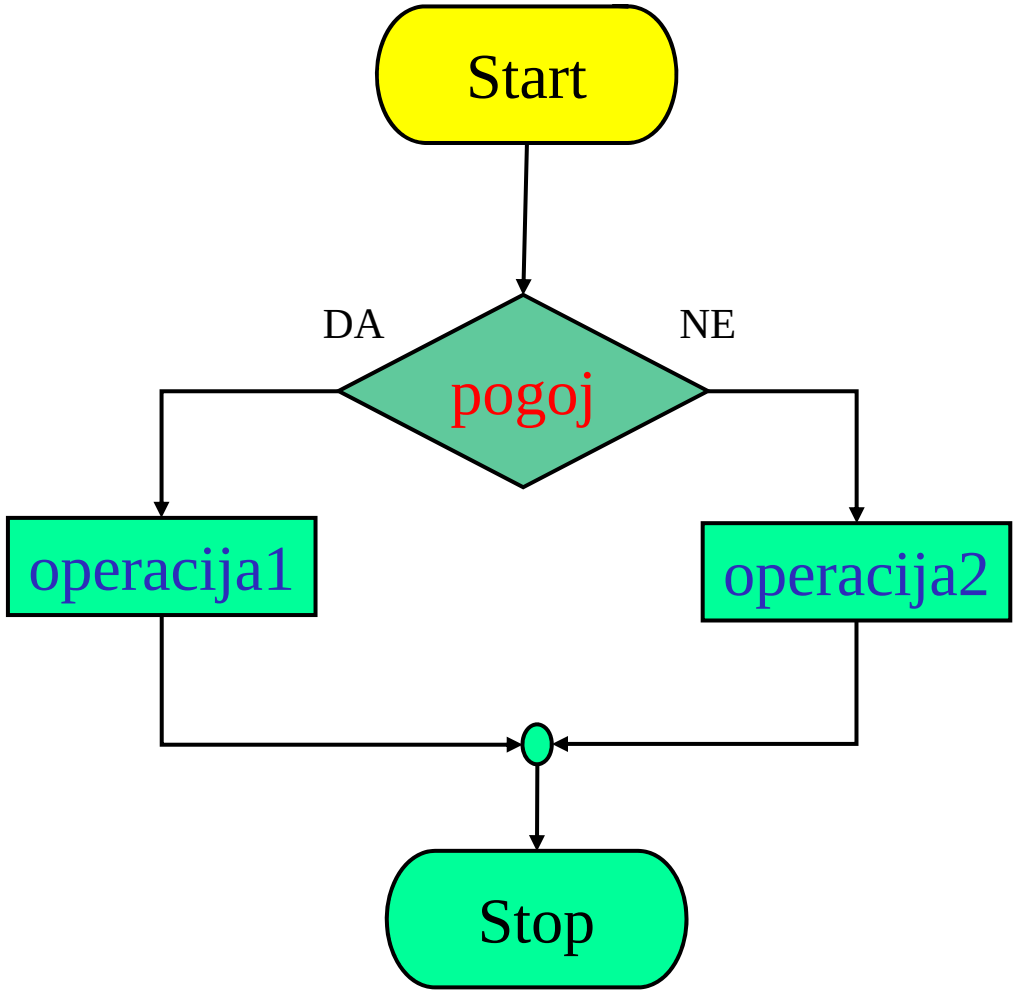
pogoj je izpolnjen

izvedi operacijo 1

pogoj ni izpolnjen

izvedi operacijo 2

končaj izvajanje



Primer: Vejitev

začni izvajanje

preveri pogoj

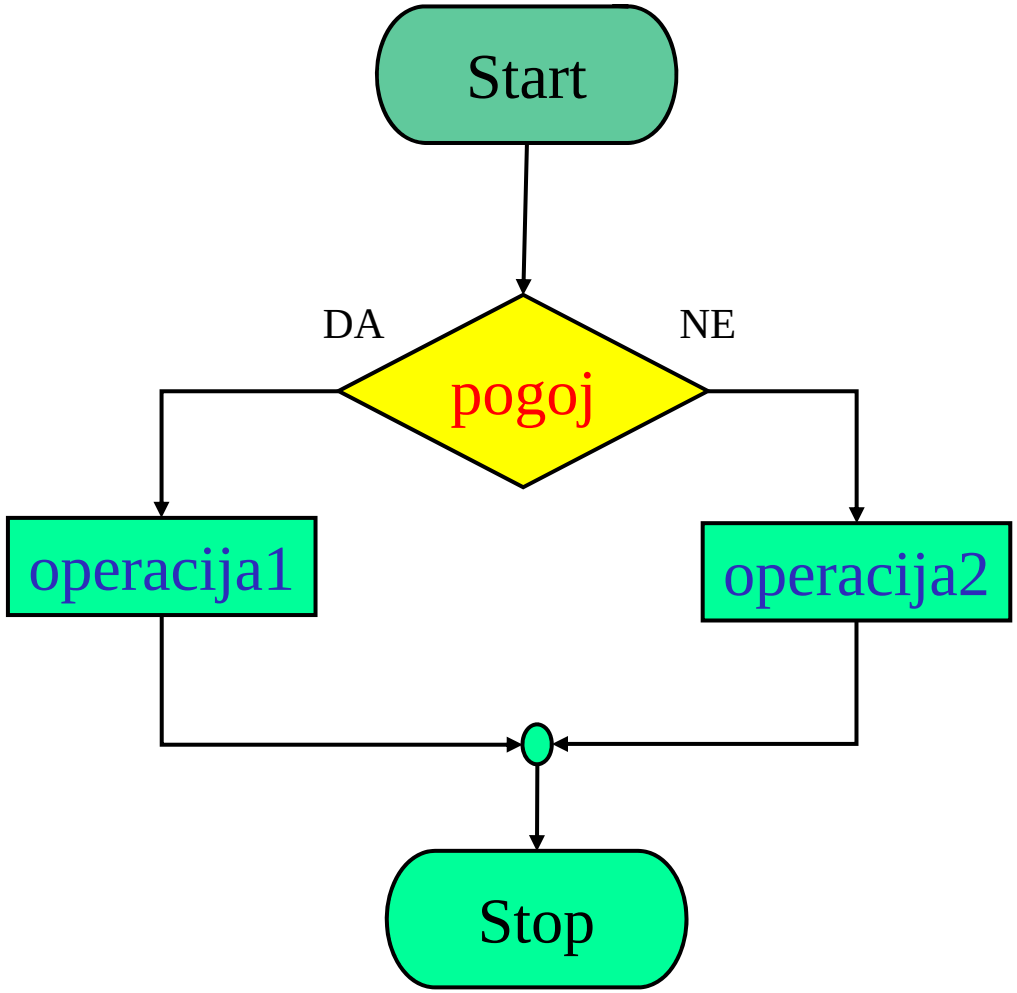
pogoj je izpolnjen

izvedi operacijo 1

pogoj ni izpolnjen

izvedi operacijo 2

končaj izvajanje



Primer: Vejitev

začni izvajanje

preveri pogoj

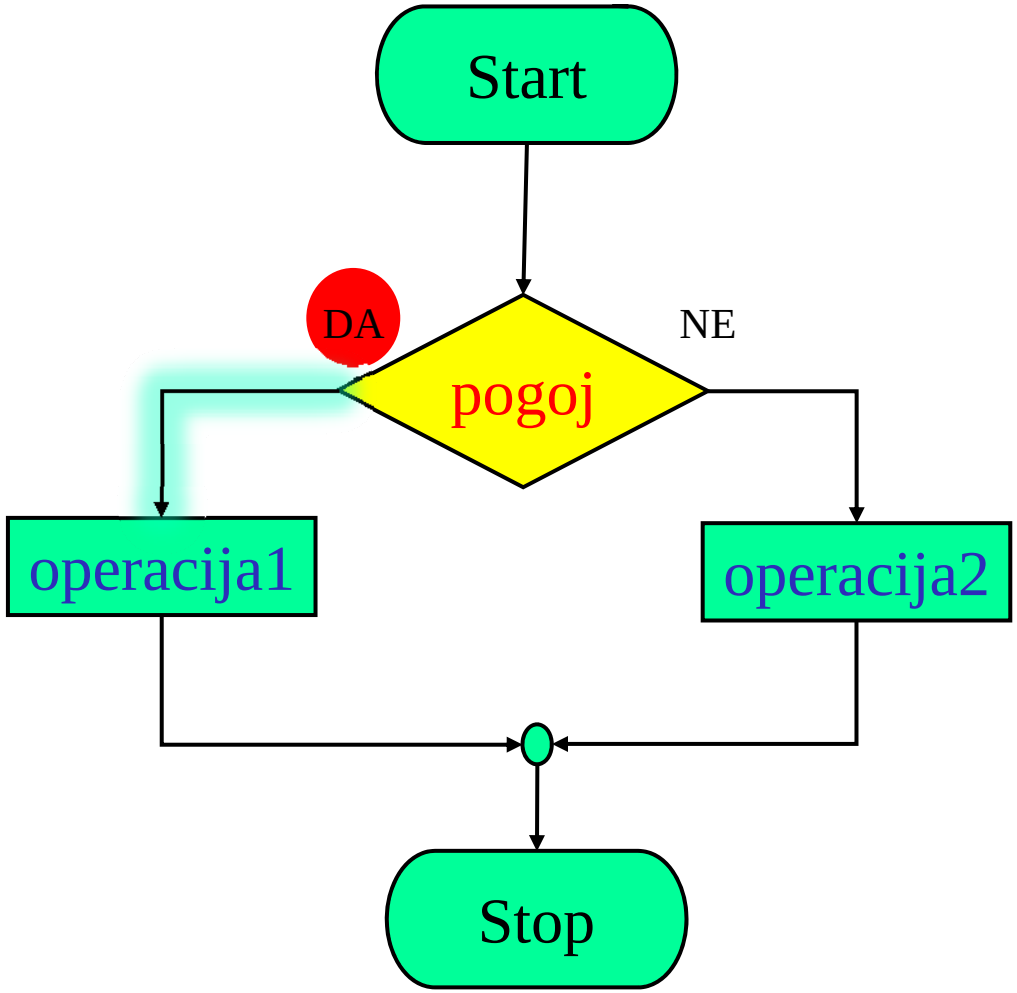
pogoj je izpolnjen

izvedi operacijo 1

pogoj ni izpolnjen

izvedi operacijo 2

končaj izvajanje



Primer: Vejitev

začni izvajanje

preveri pogoj

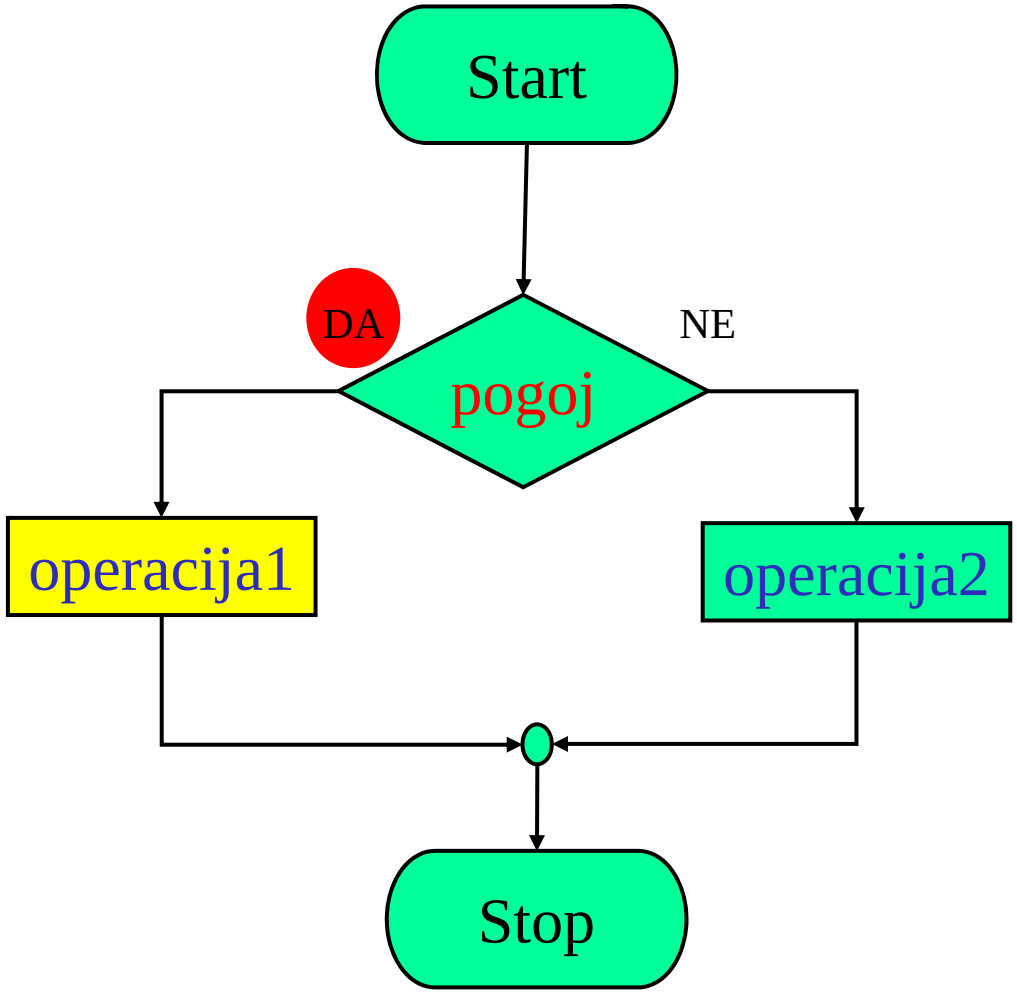
pogoj je izpolnjen

izvedi operacijo 1

pogoj ni izpolnjen

izvedi operacijo 2

končaj izvajanje



Primer: Vejitev

začni izvajanje

preveri **pogoj**

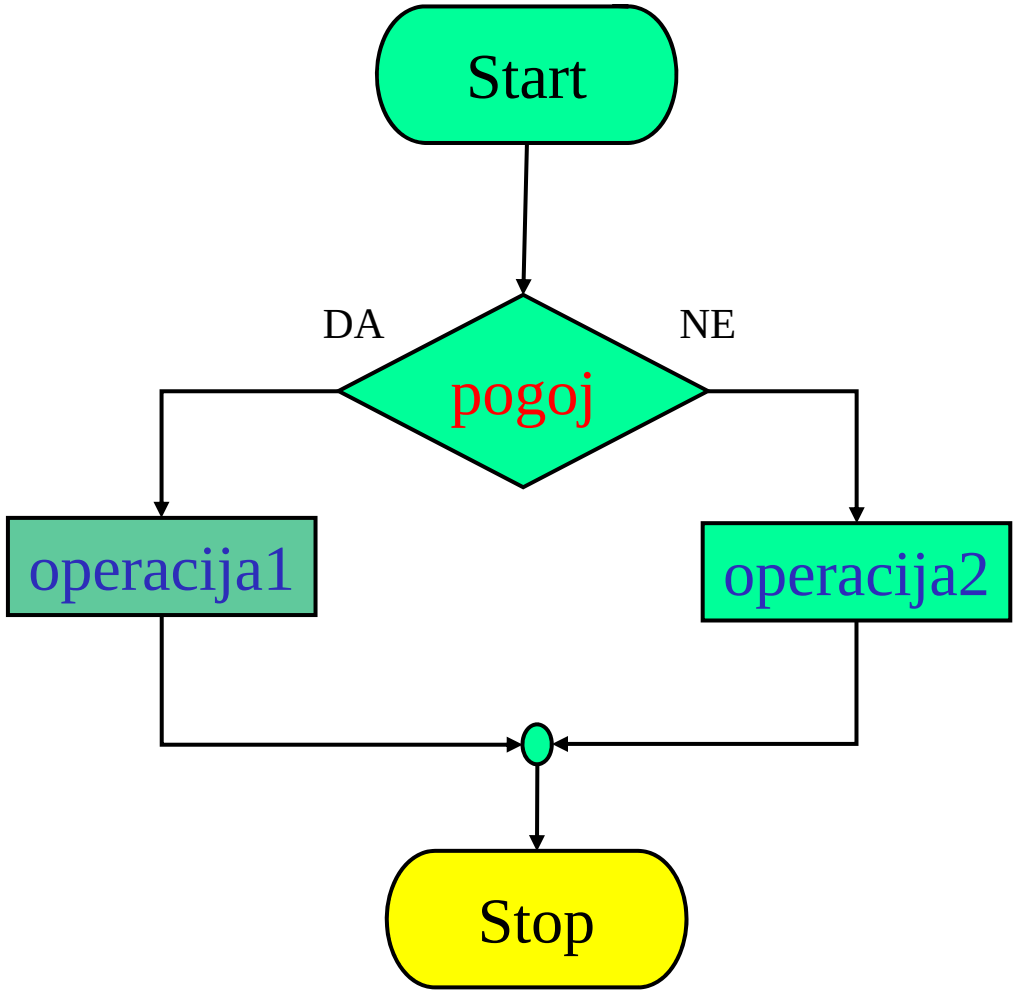
pogoj je izpolnjen

izvedi **operacijo 1**

pogoj ni izpolnjen

izvedi **operacijo 2**

končaj izvajanje



Primer: Vejitev

začni izvajanje

preveri pogoj

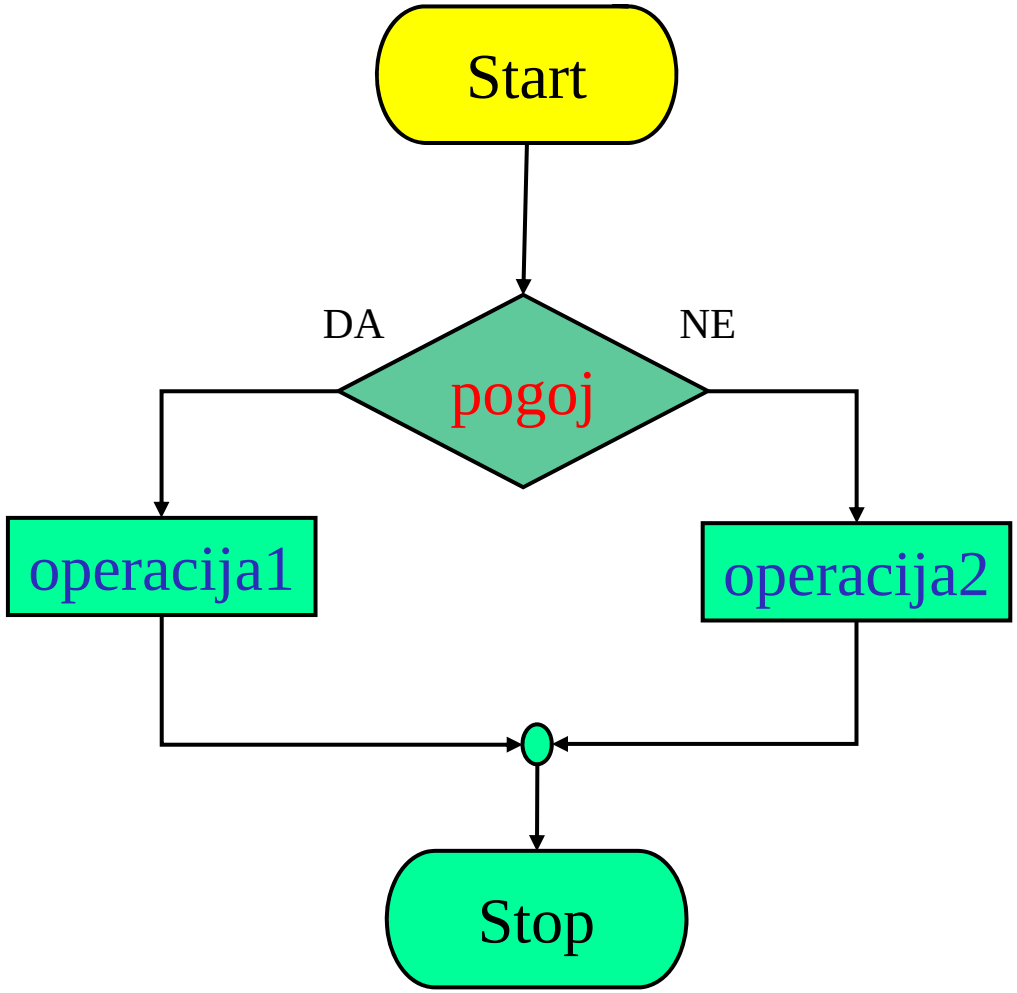
pogoj je izpolnjen

izvedi operacijo 1

pogoj ni izpolnjen

izvedi operacijo 2

končaj izvajanje



Primer: Vejitev

začni izvajanje

preveri pogoj

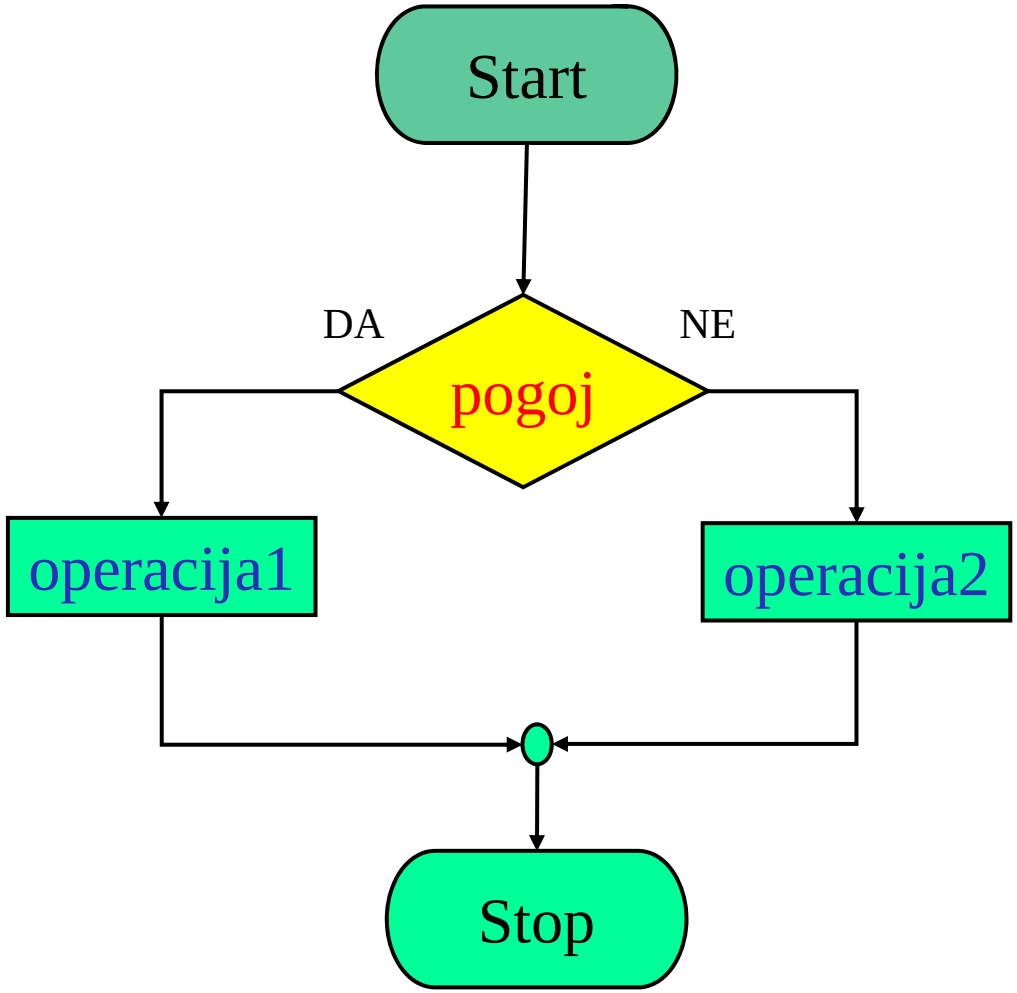
pogoj je izpolnjen

izvedi operacijo 1

pogoj ni izpolnjen

izvedi operacijo 2

končaj izvajanje



Primer: Vejitev

začni izvajanje

preveri pogoj

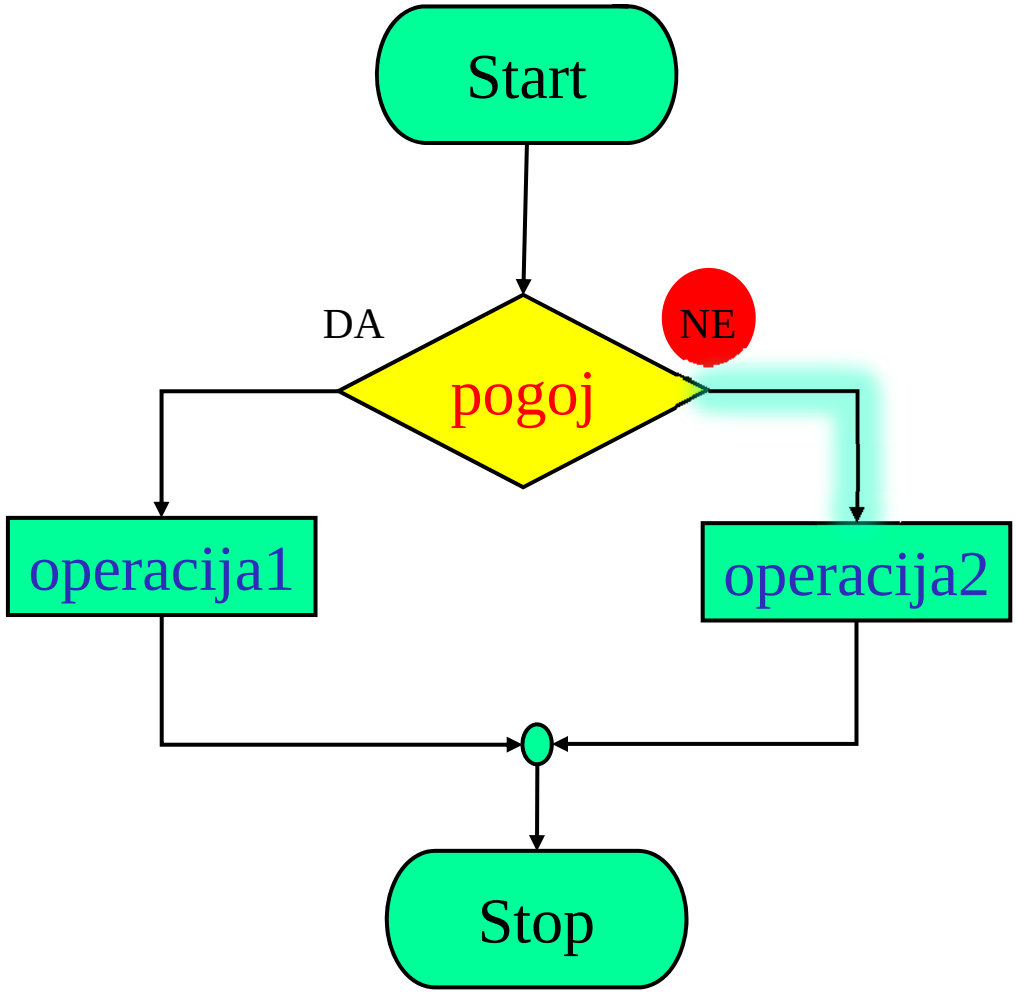
pogoj je izpolnjen

izvedi operacijo 1

pogoj ni izpolnjen

izvedi operacijo 2

končaj izvajanje



Primer: Vejitev

začni izvajanje

preveri pogoj

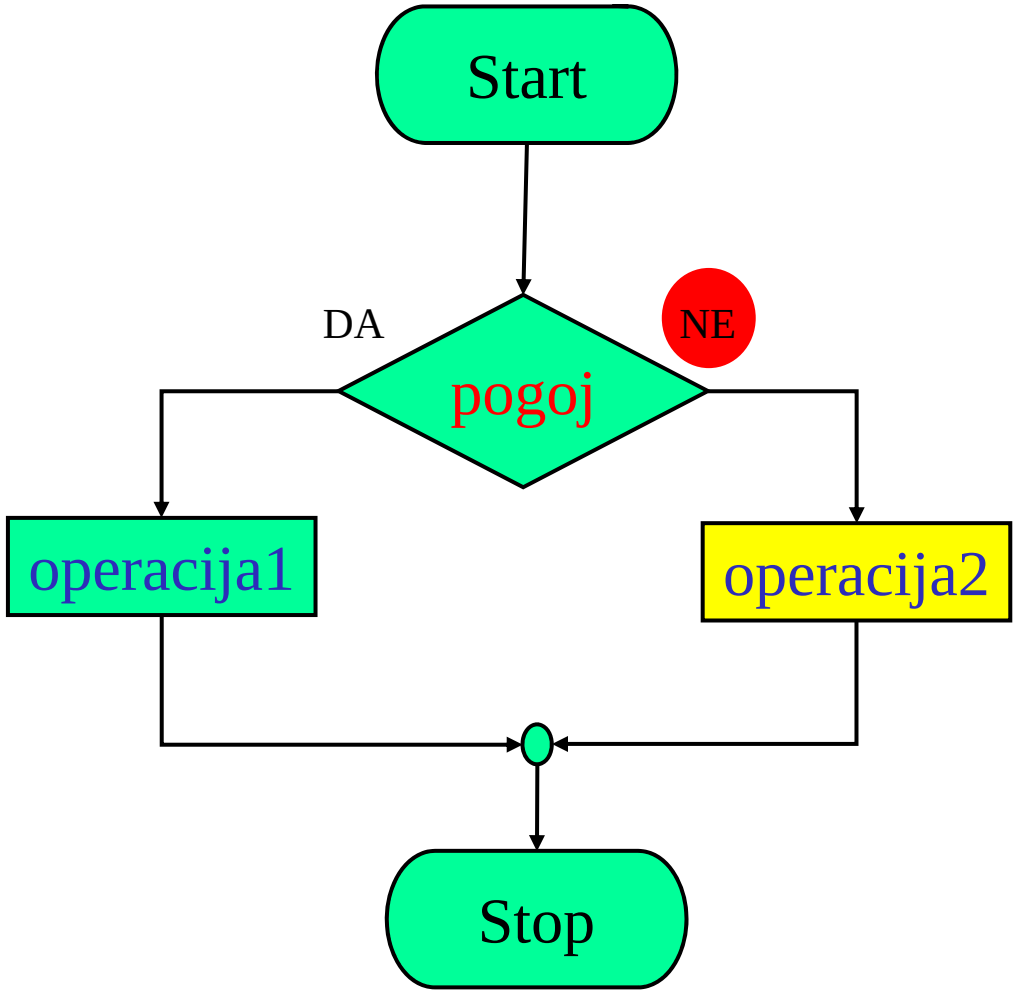
pogoj je izpolnjen

izvedi operacijo 1

pogoj ni izpolnjen

izvedi operacijo 2

končaj izvajanje



Primer: Vejitev

začni izvajanje

preveri **pogoj**

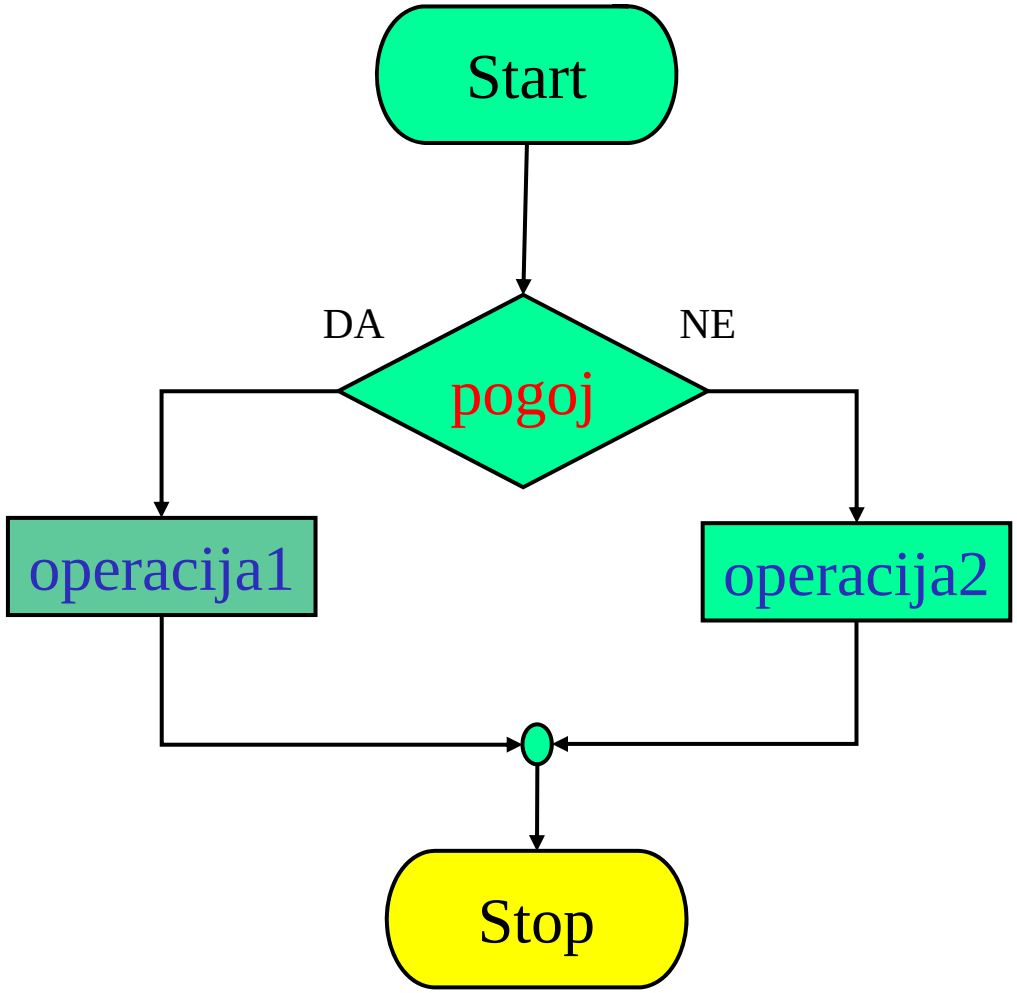
pogoj je izpolnjen

izvedi **operacijo 1**

pogoj ni izpolnjen

izvedi **operacijo 2**

končaj izvajanje



Primer: Vejitev – primer za minimum

začni izvajanje

ali je a manjši od b?

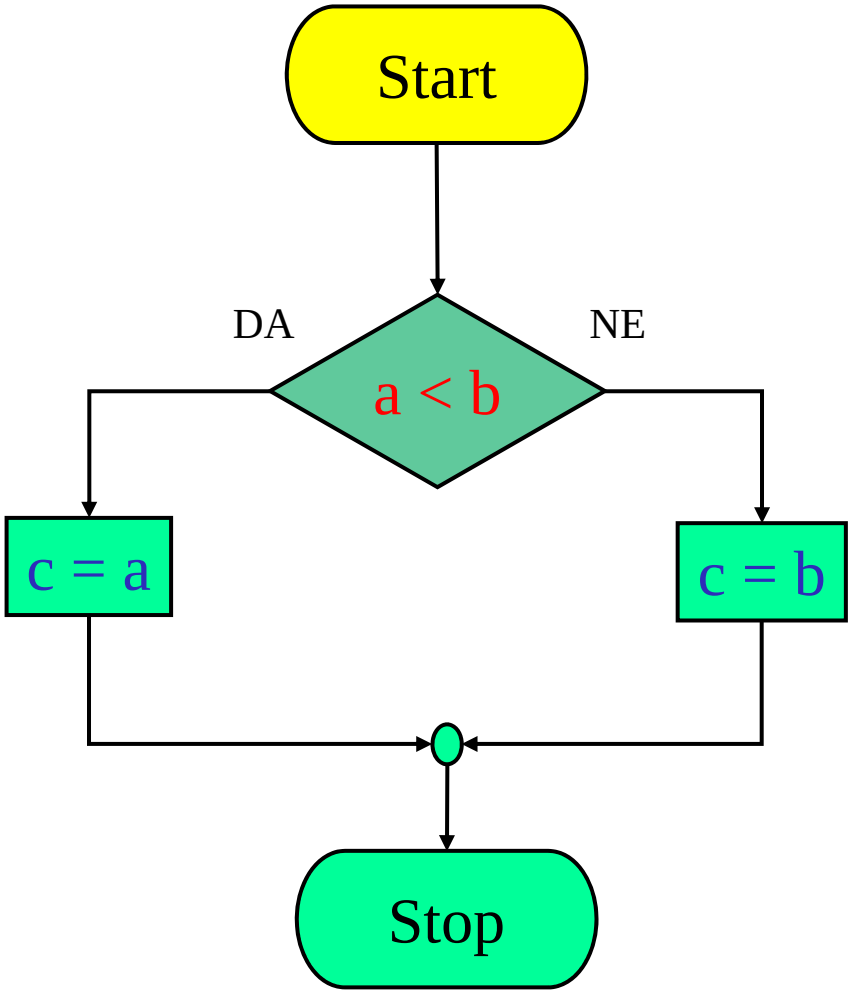
a je manjši od b

c=a

a ni manjši od b

c=b

končaj izvajanje



Primer: Vejitev – primer za minimum

začni izvajanje

ali je a manjši od b?

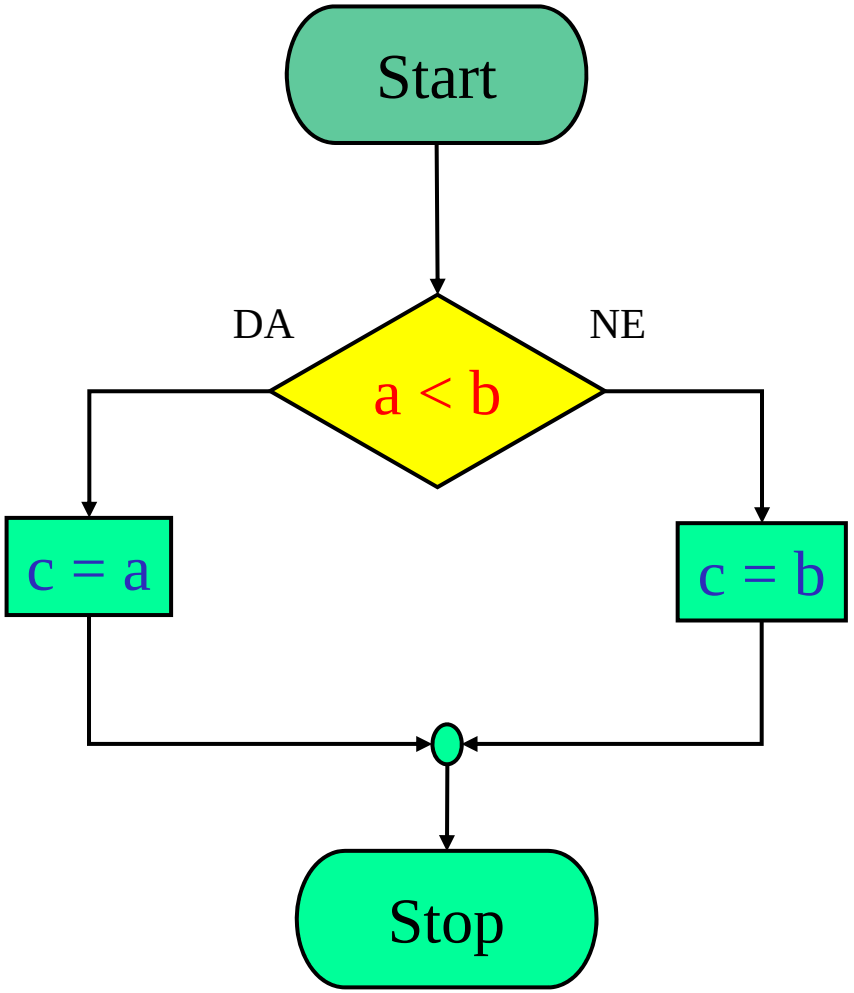
a je manjši od b

c=a

a ni manjši od b

c=b

končaj izvajanje



Primer: Vejitev – primer za minimum

začni izvajanje

ali je a manjši od b?

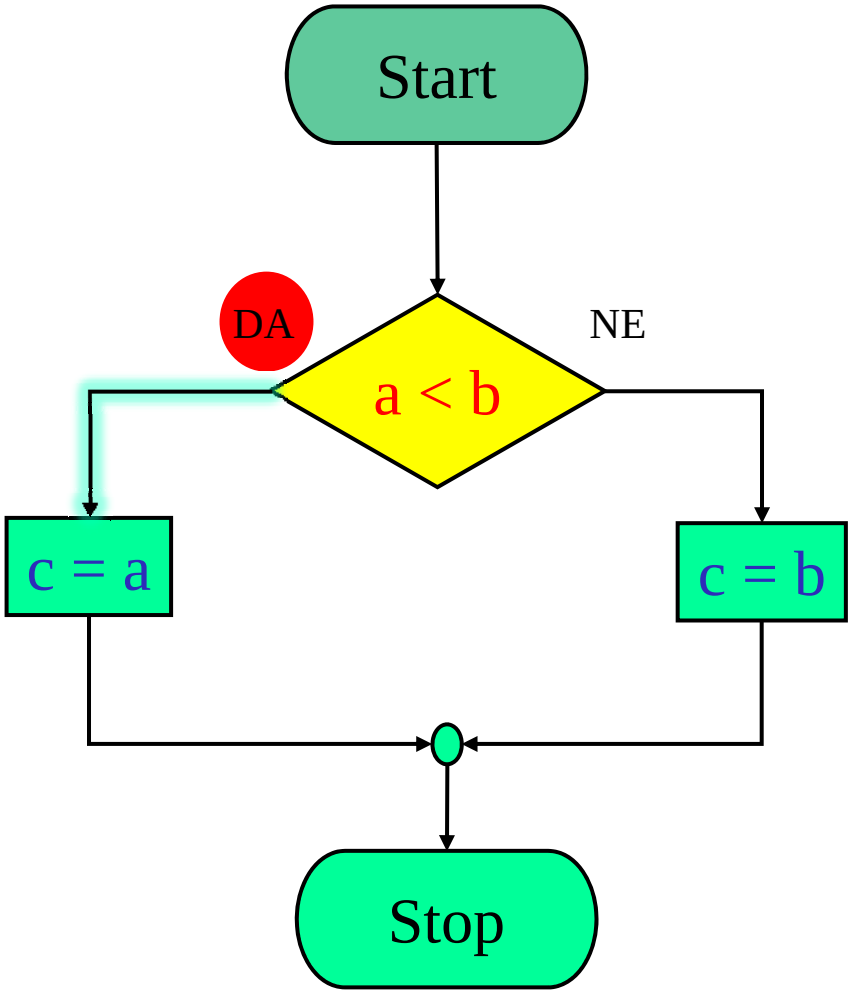
a je manjši od b

c=a

a ni manjši od b

c=b

končaj izvajanje



Primer: Vejitev – primer za minimum

začni izvajanje

ali je a manjši od b?

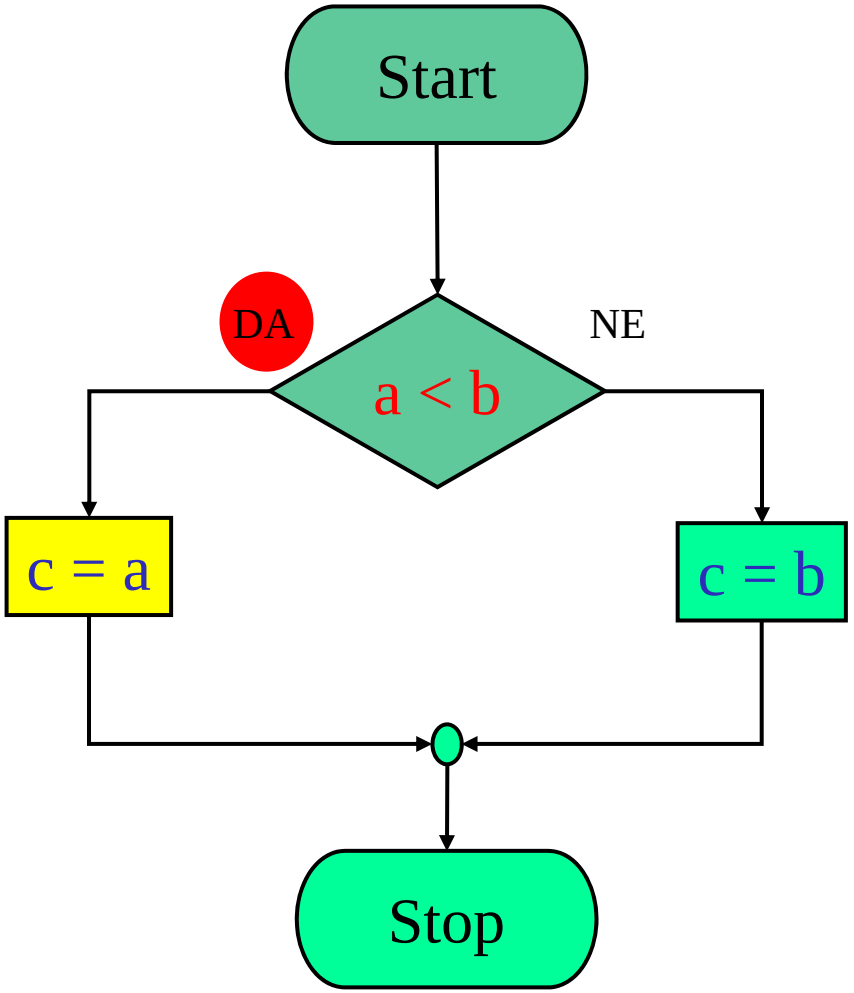
a je manjši od b

c=a

a ni manjši od b

c=b

končaj izvajanje



Primer: Vejitev – primer za minimum

začni izvajanje

ali je a manjši od b?

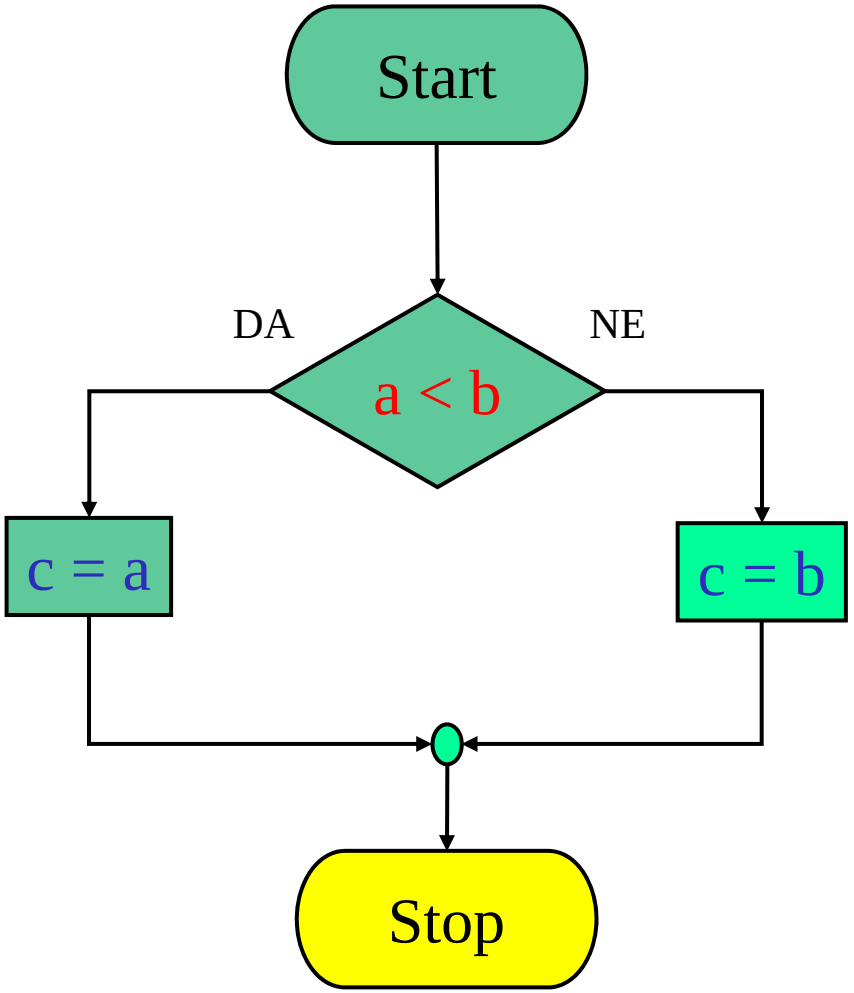
a je manjši od b

c=a

a ni manjši od b

c=b

končaj izvajanje



Primer: Vejitev – primer za minimum

začni izvajanje

ali je a manjši od b?

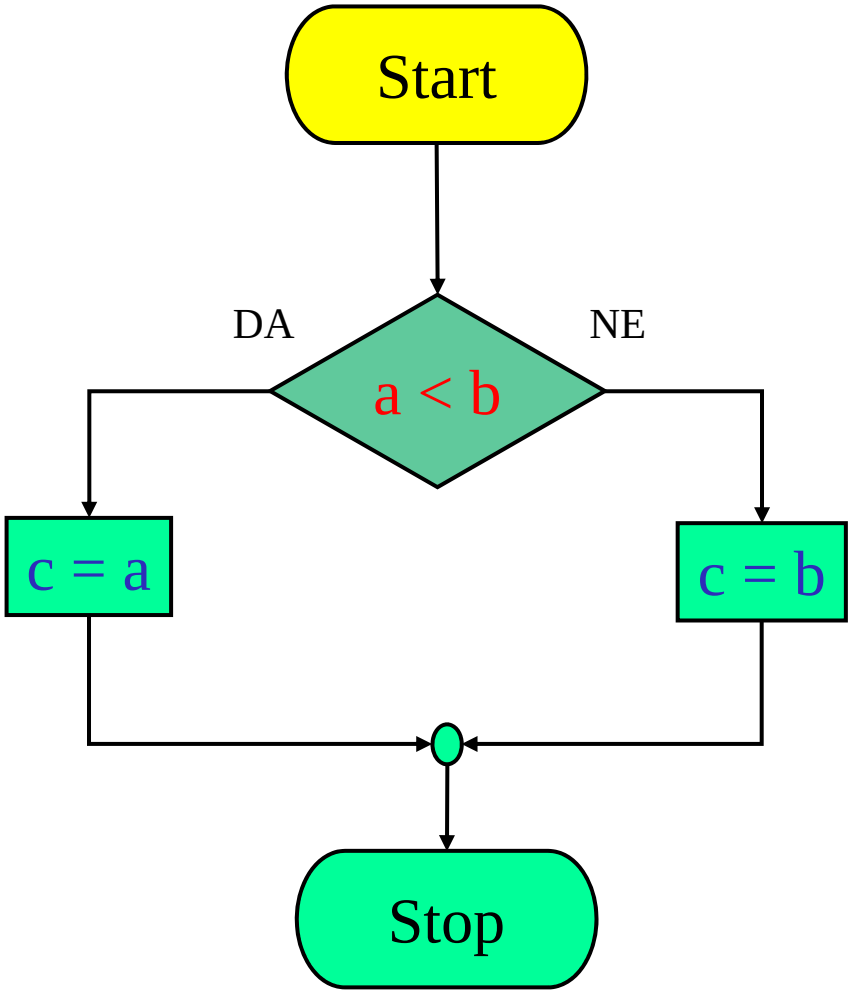
a je manjši od b

c=a

a ni manjši od b

c=b

končaj izvajanje



Primer: Vejitev – primer za minimum

začni izvajanje

ali je a manjši od b?

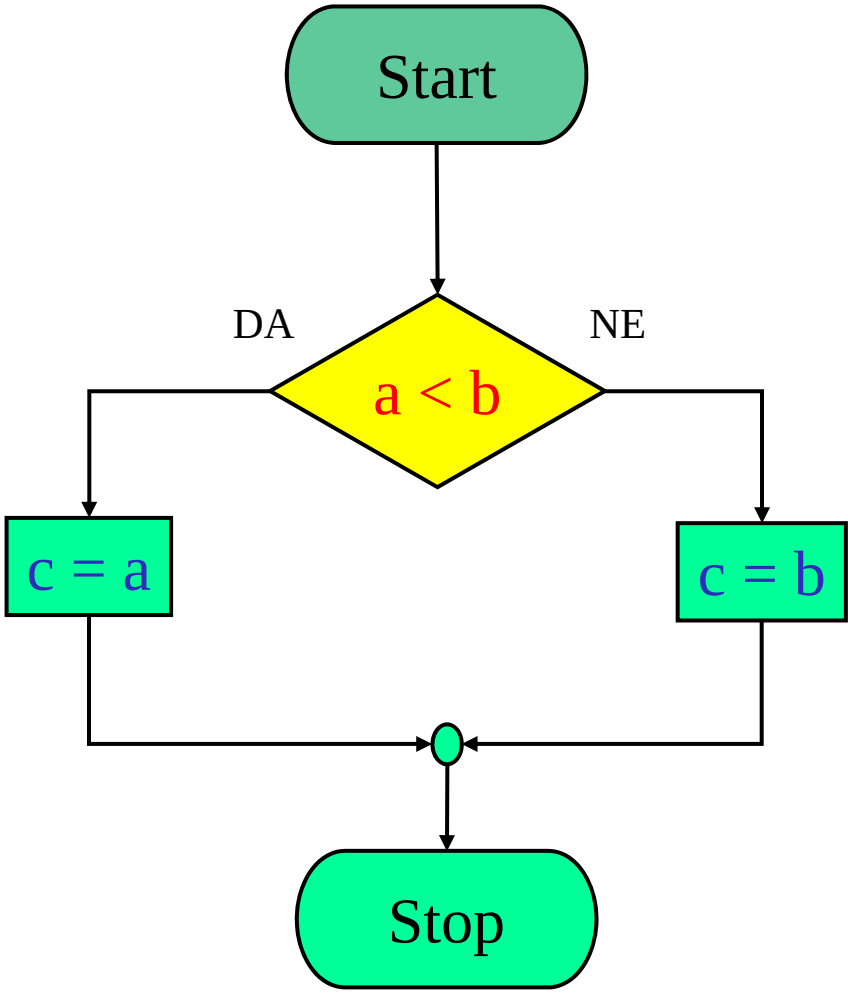
a je manjši od b

c=a

a ni manjši od b

c=b

končaj izvajanje



Primer: Vejitev – primer za minimum

začni izvajanje

ali je a manjši od b?

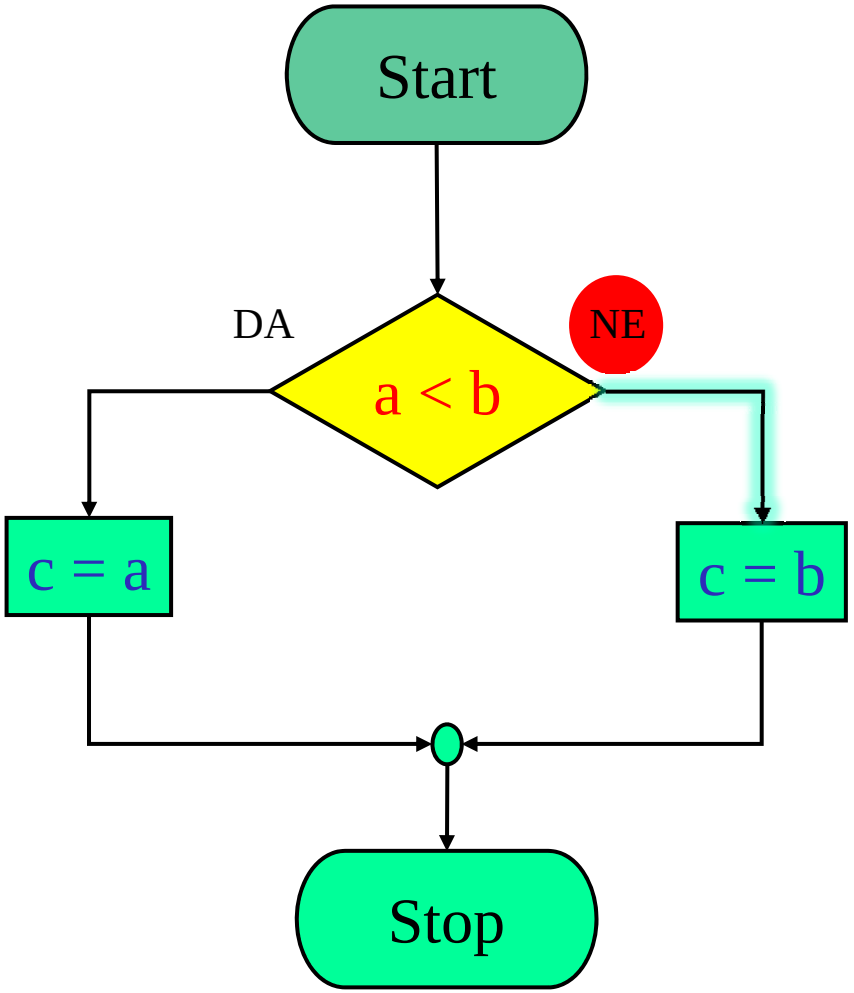
a je manjši od b

c=a

a ni manjši od b

c=b

končaj izvajanje



Primer: Vejitev – primer za minimum

začni izvajanje

ali je a manjši od b?

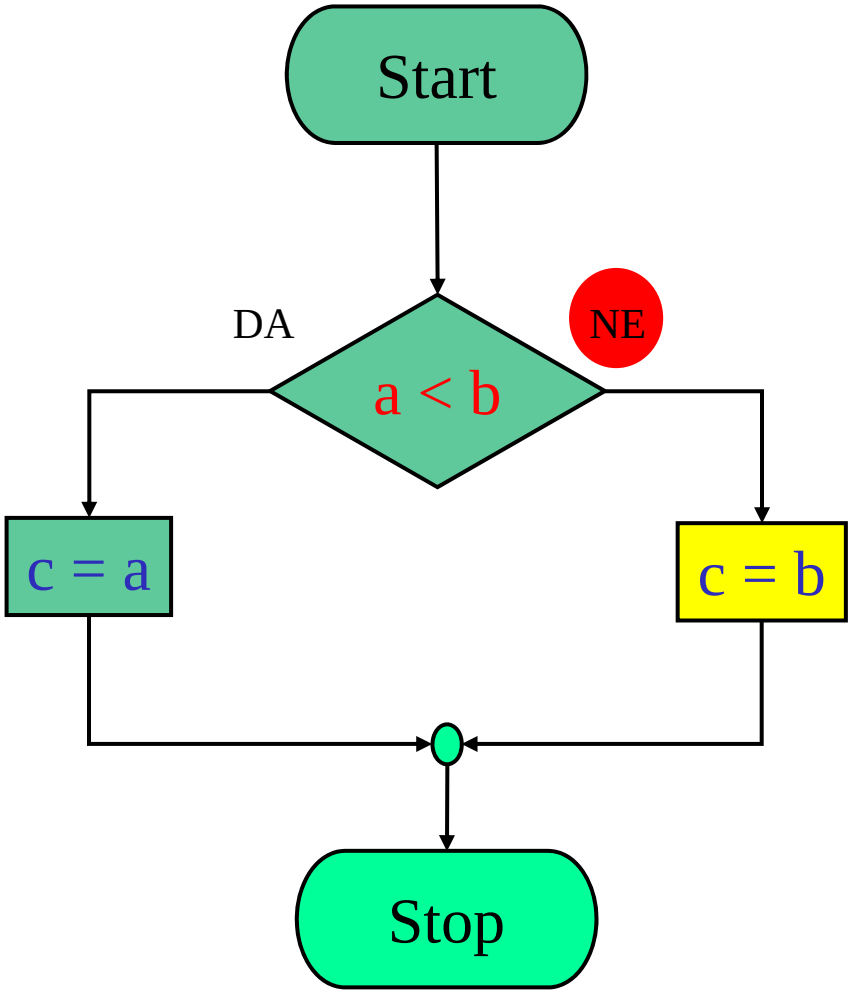
a je manjši od b

c=a

a ni manjši od b

c=b

končaj izvajanje



Primer: Vejitev – primer za minimum

začni izvajanje

ali je a manjši od b?

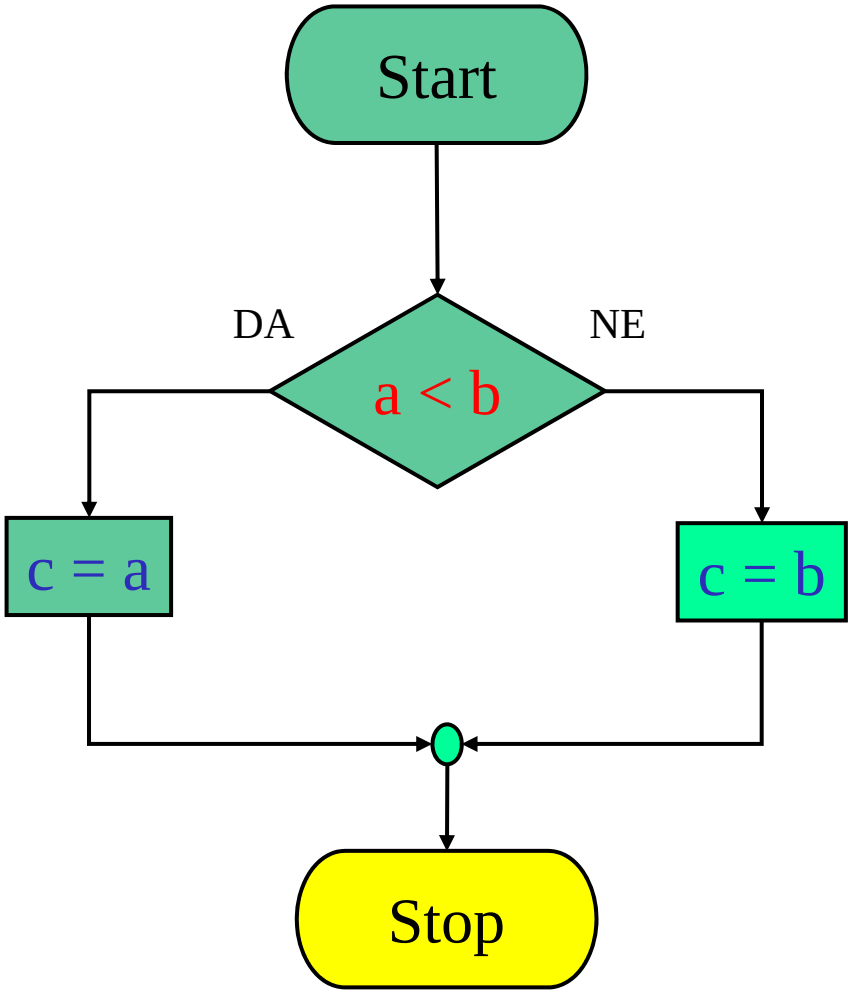
a je manjši od b

c=a

a ni manjši od b

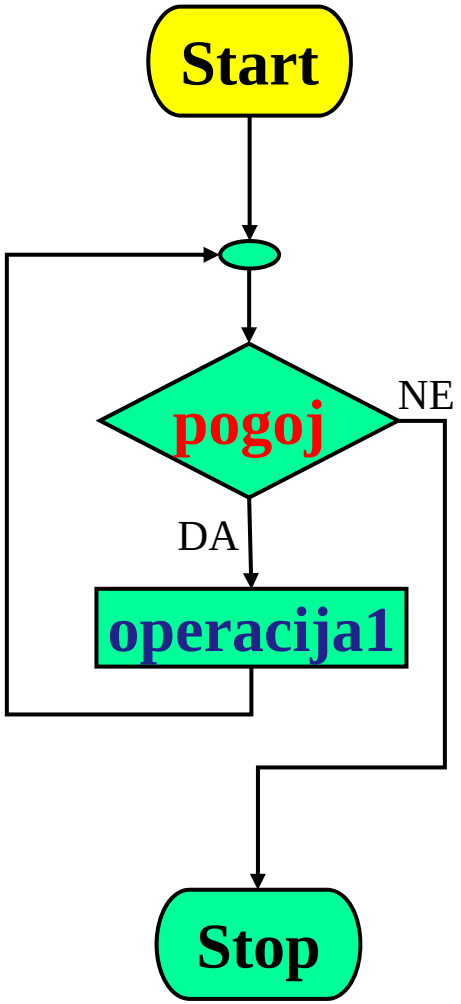
c=b

končaj izvajanje



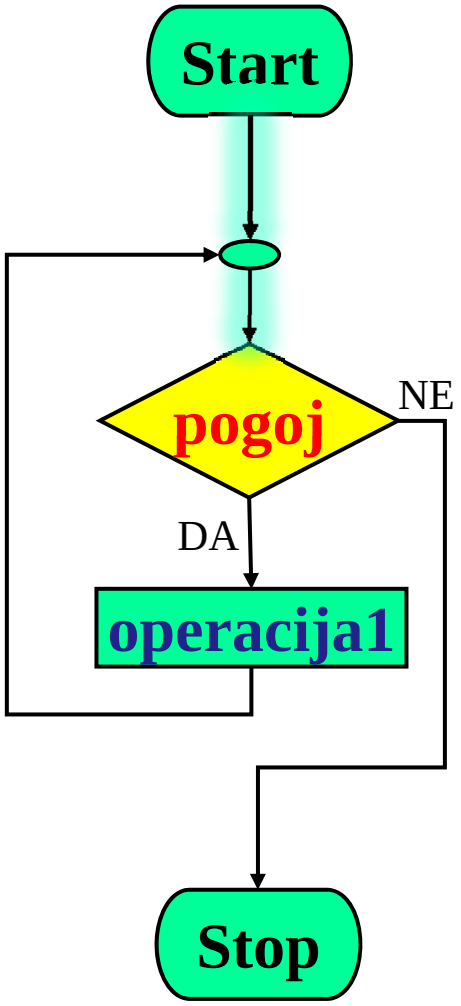
Primer: Zanka *while-do*

```
while(pogoj)
{
   operacija1;
}
```



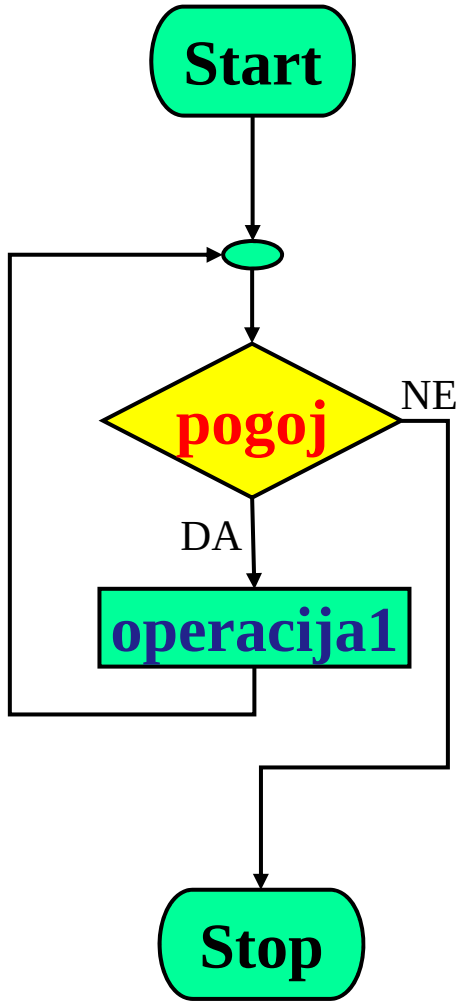
Primer: Zanka *while-do*

```
while(pogoj)
{
   operacija1;
}
```



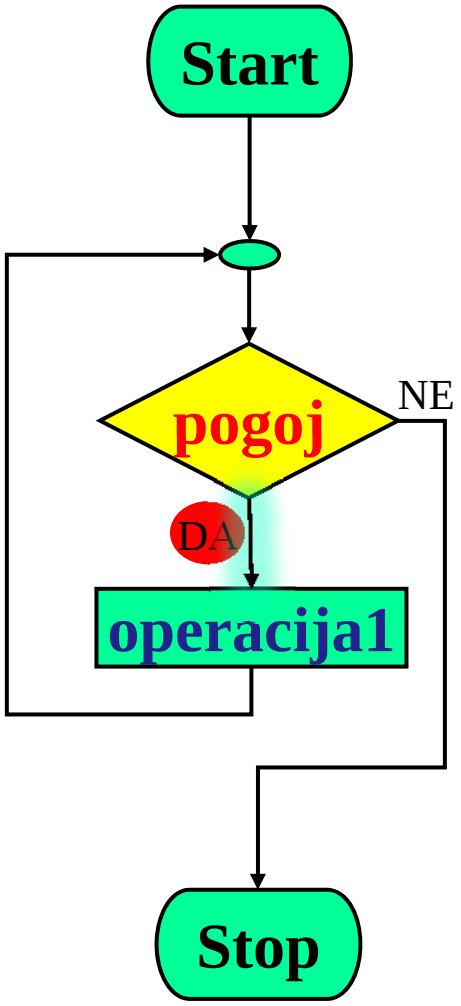
Primer: Zanka *while-do*

```
while(pogoj)
  {
    operacija1;
  }
```



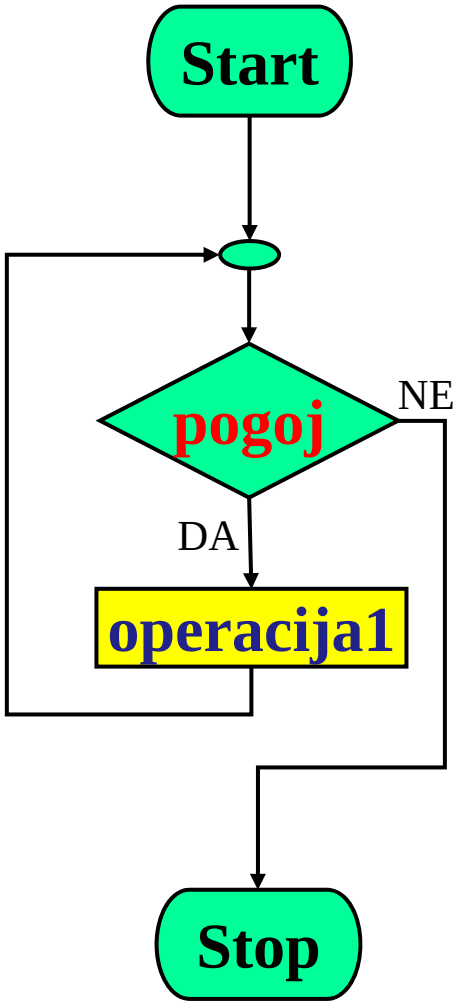
Primer: Zanka *while-do*

```
while(pogoj)
  {
    operacija1;
  }
```



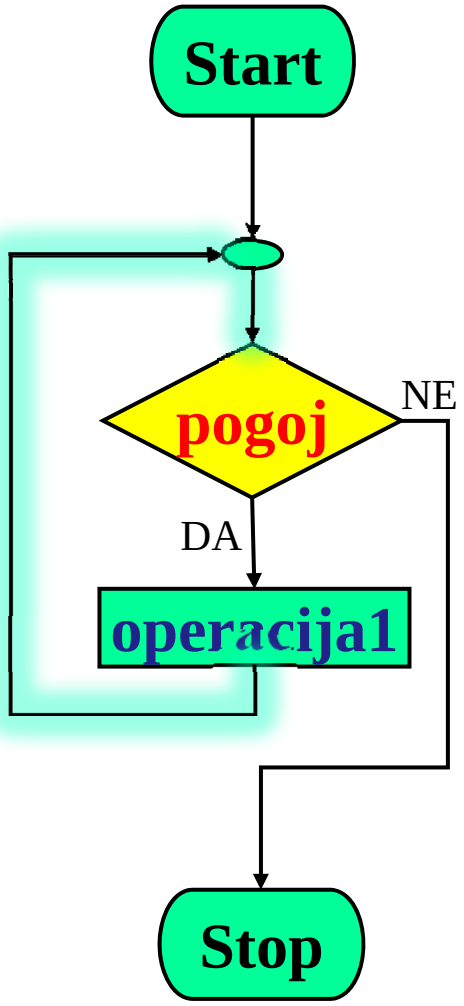
Primer: Zanka *while-do*

```
while(pogoj)
{
   operacija1;
}
```



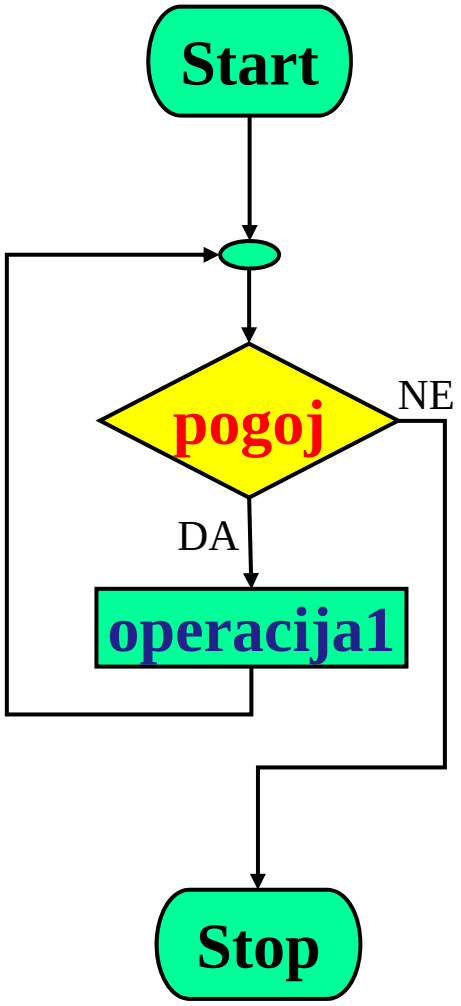
Primer: Zanka *while-do*

```
while(pogoj)
{
   operacija1;
}
```



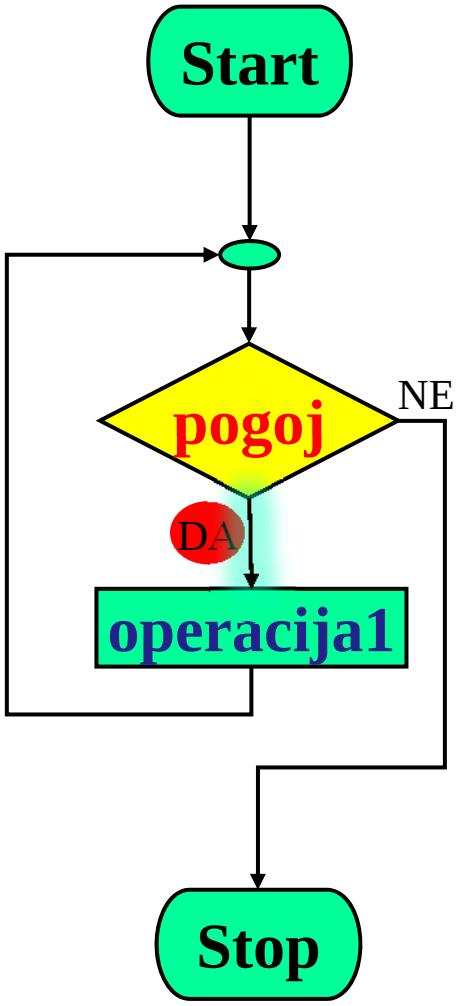
Primer: Zanka *while-do*

```
while(pogoj)
  {
    operacija1;
  }
```



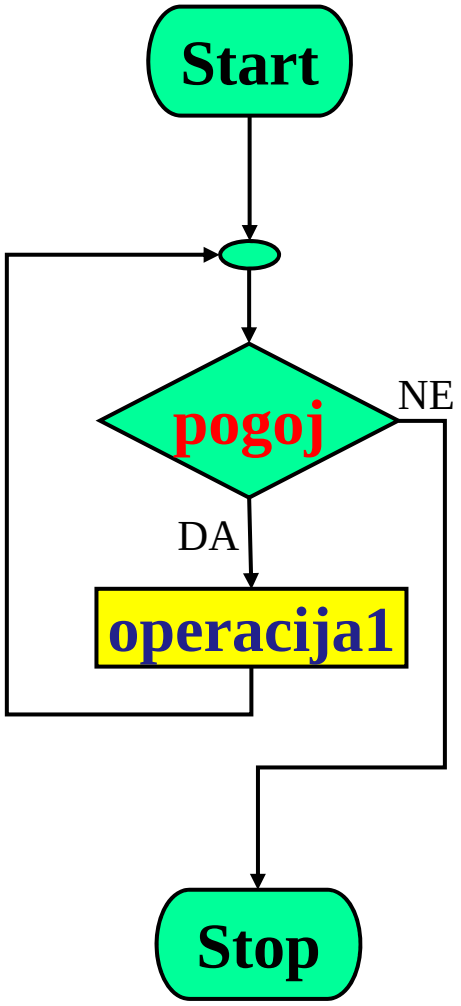
Primer: Zanka *while-do*

```
while(pogoj)
  {
    operacija1;
  }
```



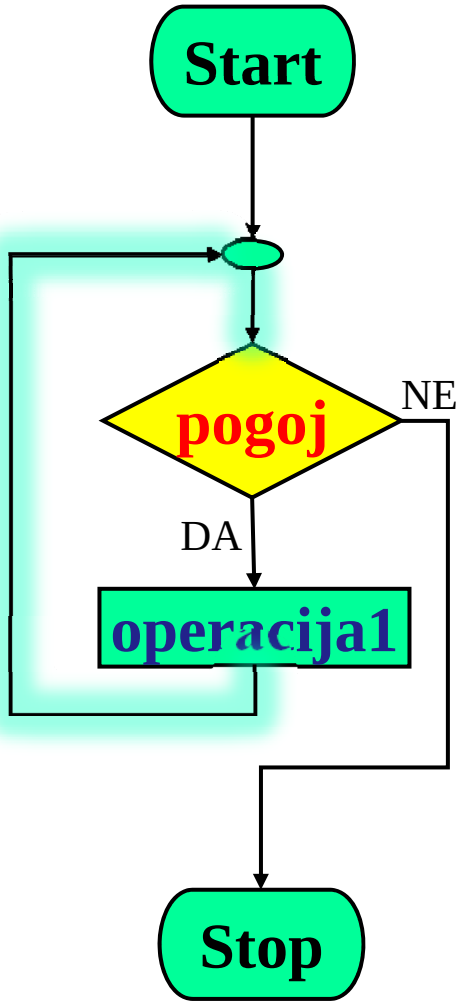
Primer: Zanka *while-do*

```
while(pogoj)
{
   operacija1;
}
```



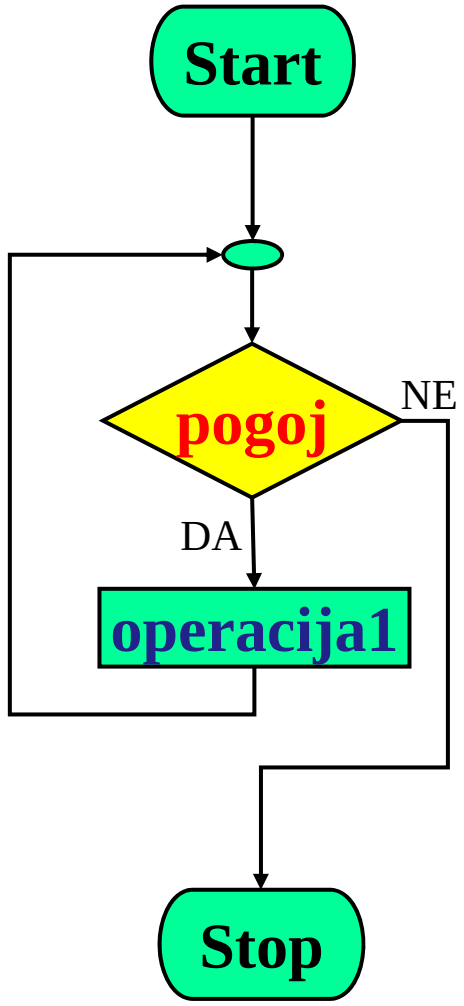
Primer: Zanka *while-do*

```
while(pogoj)
{
   operacija1;
}
```



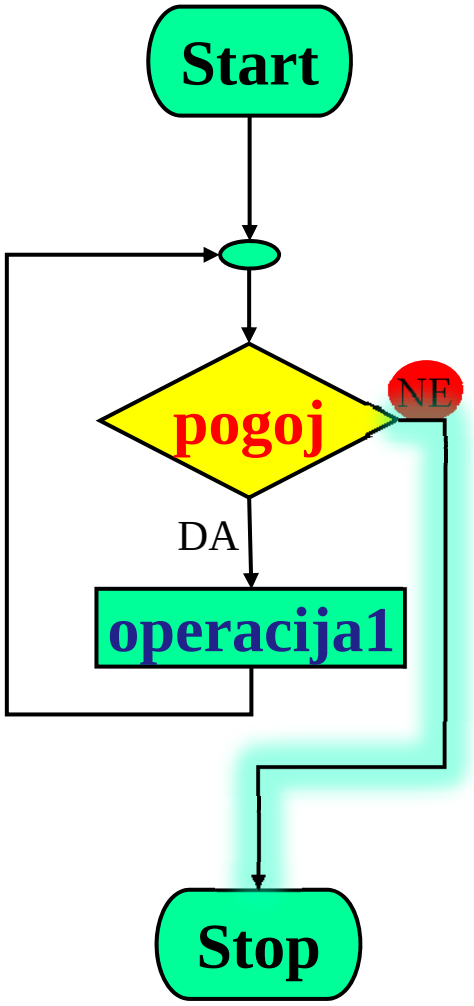
Primer: Zanka *while-do*

```
while(pogoj)
  {
    operacija1;
  }
```



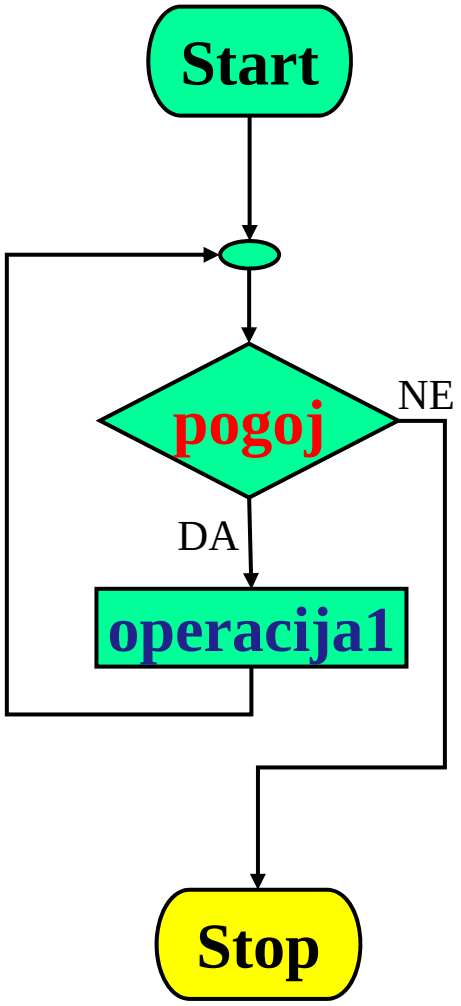
Primer: Zanka *while-do*

```
while(pogoj)
  {
    operacija1;
  }
```



Primer: Zanka *while-do*

```
while(pogoj)
{
   operacija1;
}
```

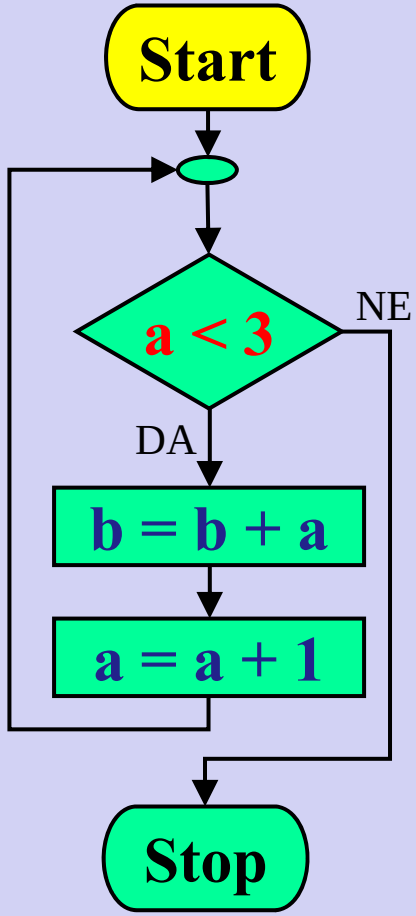


Praktični primer: Zanka *while-do*

Program:

```
while(a < 3)
{
   b += a;
   a++;
}
```

Diagram poteka:



Spremenljivki:

a = 1;

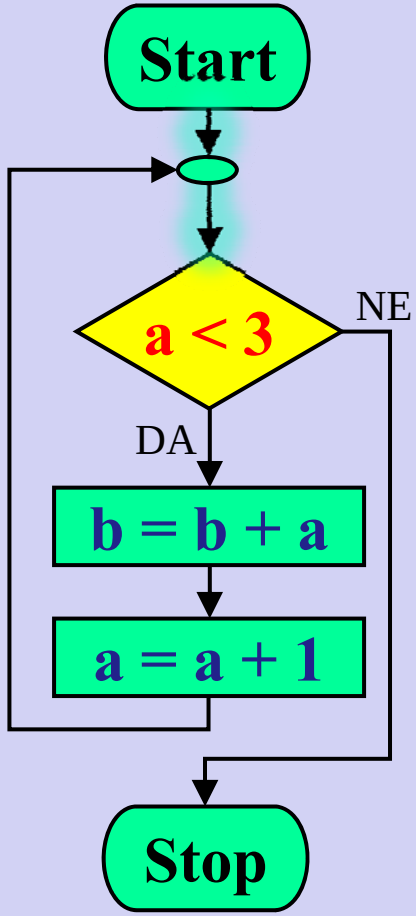
b = 0;

Praktični primer: Zanka *while-do*

Program:

```
while(a < 3)
{
    b += a;
    a++;
}
```

Diagram poteka:



Spremenljivki:

a = 1;

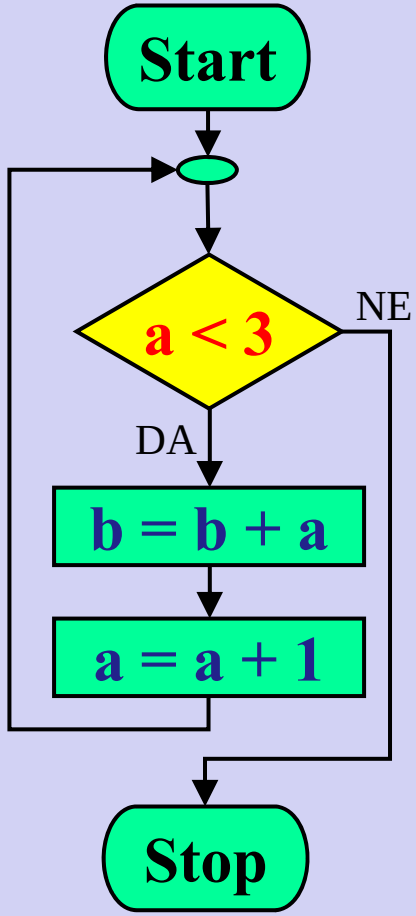
b = 0;

Praktični primer: Zanka *while-do*

Program:

```
while(a < 3)
  {
    b += a;
    a++;
  }
```

Diagram poteka:



Spremenljivki:

a = 1;

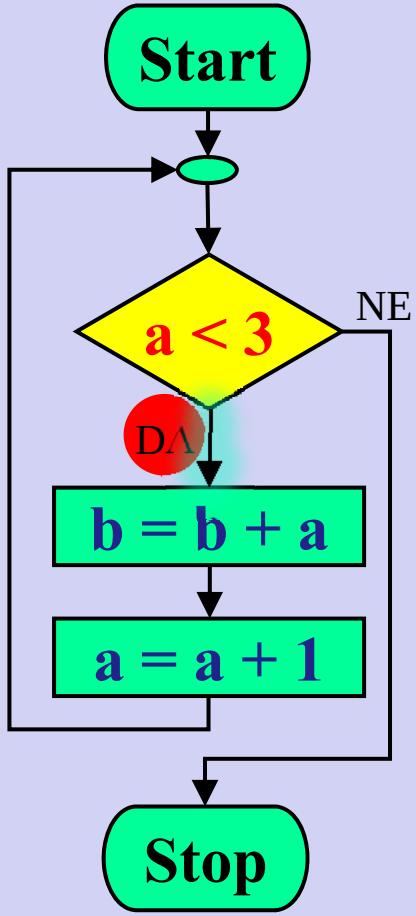
b = 0;

Praktični primer: Zanka *while-do*

Program:

```
while(a < 3)
  {
    b += a;
    a++;
  }
```

Diagram poteka:



Spremenljivki:

a = 1;

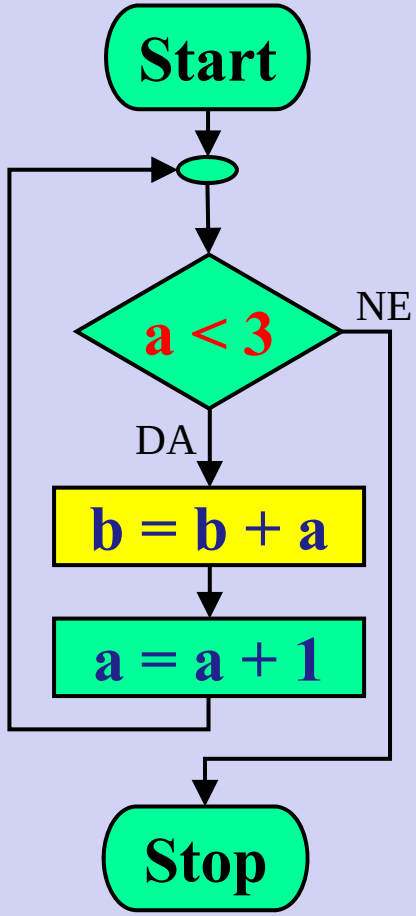
b = 0;

Praktični primer: Zanka *while-do*

Program:

```
while(a < 3)
{
   b += a;
   a++;
}
```

Diagram poteka:



Spremenljivki:

a = 1;

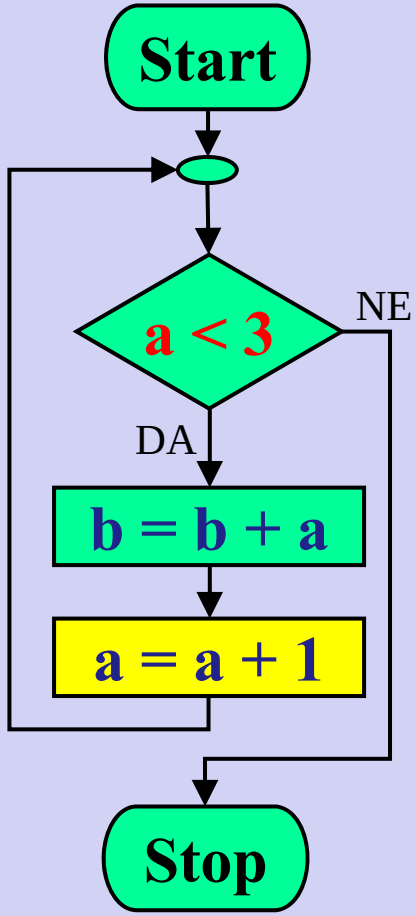
b = 1;

Praktični primer: Zanka *while-do*

Program:

```
while(a < 3)
{
   b += a;
   a++;
}
```

Diagram poteka:



Spremenljivki:

a = 1;

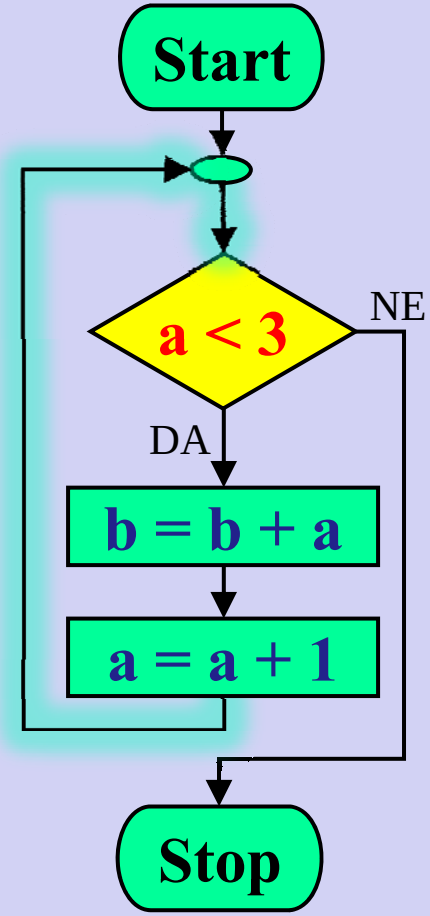
b = 1;

Praktični primer: Zanka *while-do*

Program:

```
while(a < 3)
{
    b += a;
    a++;
}
```

Diagram poteka:



Spremenljivki:

a = 1;

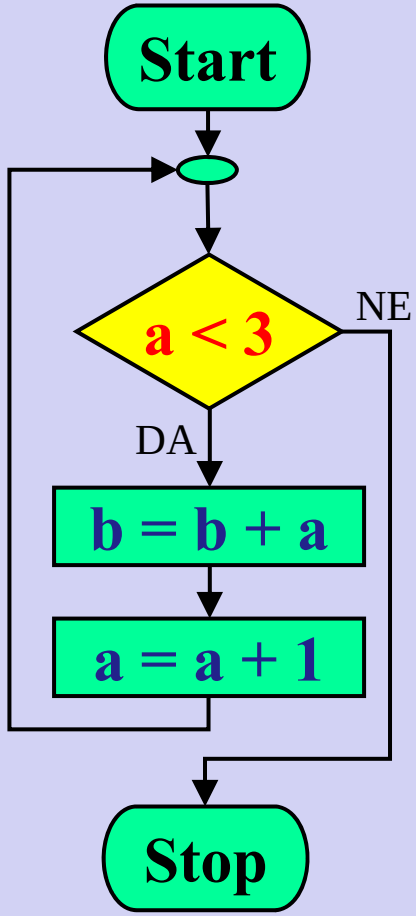
b = 1;

Praktični primer: Zanka *while-do*

Program:

```
while(a < 3)
  {
    b += a;
    a++;
  }
```

Diagram poteka:



Spremenljivki:

a = 1;

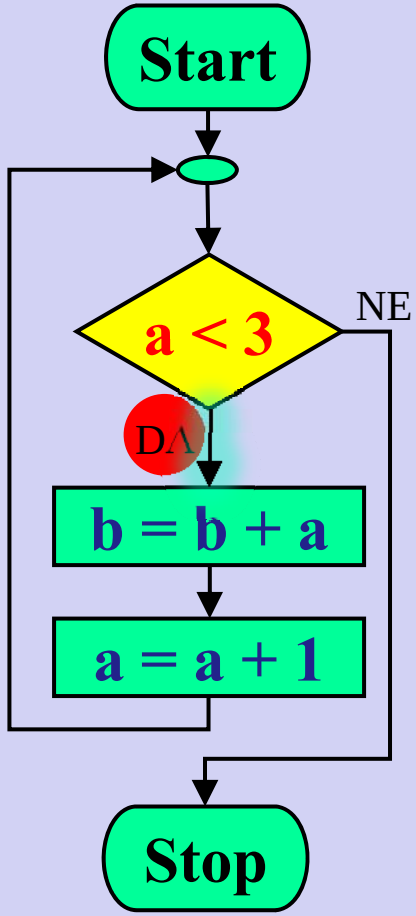
b = 1;

Praktični primer: Zanka *while-do*

Program:

```
while(a < 3)
  {
    b += a;
    a++;
  }
```

Diagram poteka:



Spremenljivki:

a = 1;

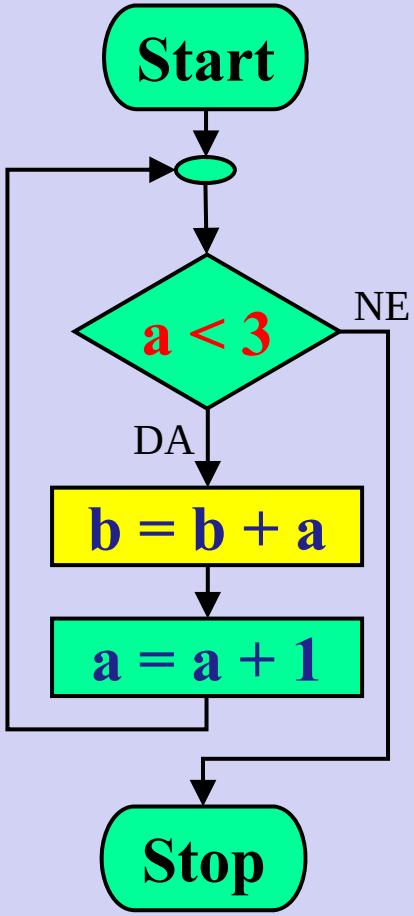
b = 1;

Praktični primer: Zanka *while-do*

Program:

```
while(a < 3)
{
   b += a;
   a++;
}
```

Diagram poteka:



Spremenljivki:

a = 1;

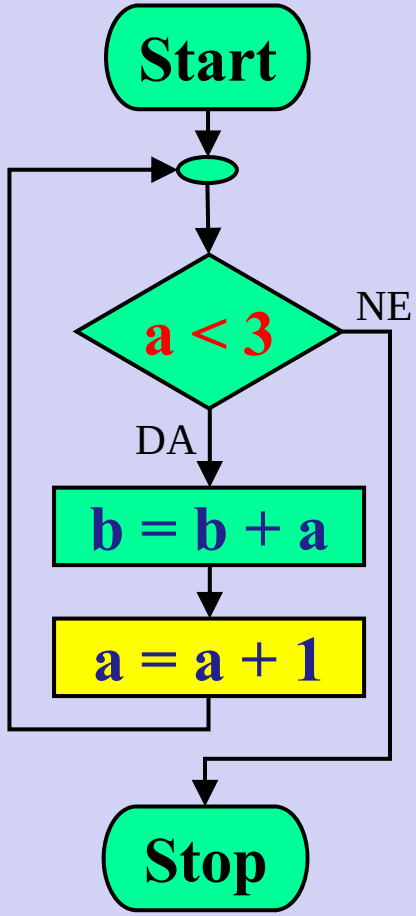
b = 2;

Praktični primer: Zanka *while-do*

Program:

```
while(a < 3)
{
   b += a;
   a++;
}
```

Diagram poteka:



Spremenljivki:

a = 2;

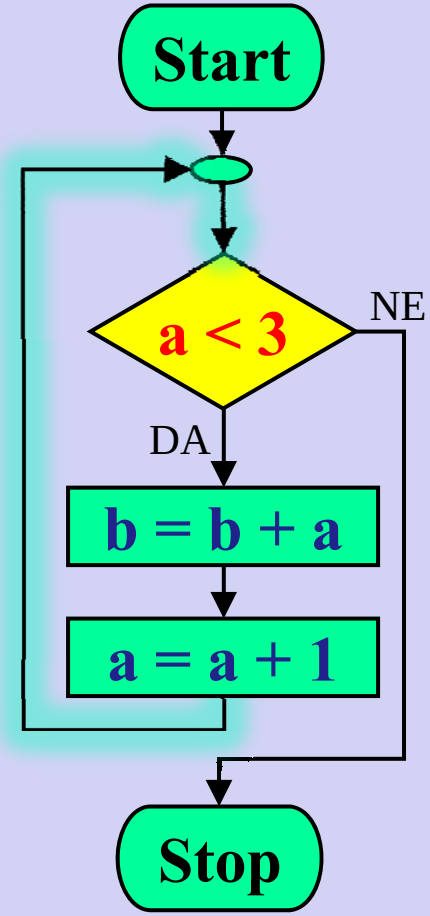
b = 2;

Praktični primer: Zanka *while-do*

Program:

```
while(a < 3)
{
    b += a;
    a++;
}
```

Diagram poteka:



Spremenljivki:

a = 2;

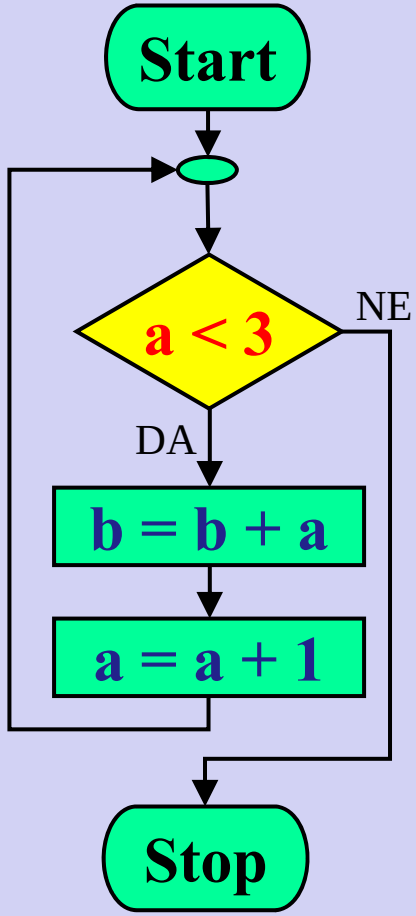
b = 2;

Praktični primer: Zanka *while-do*

Program:

```
while(a < 3)
  {
    b += a;
    a++;
  }
```

Diagram poteka:



Spremenljivki:

a = 2;

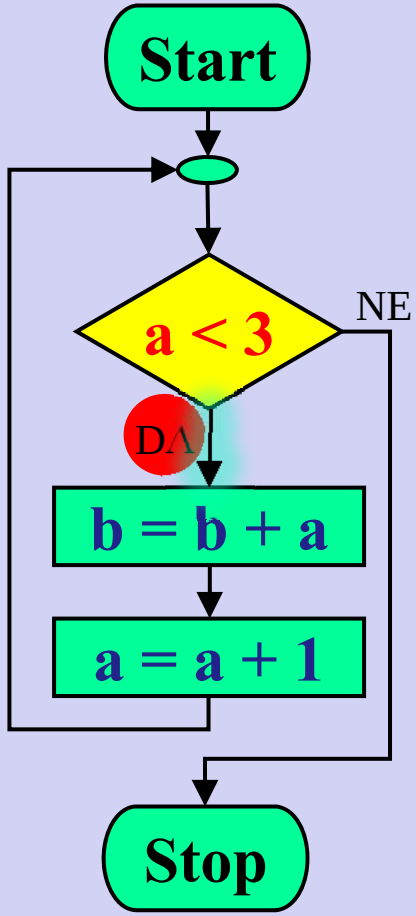
b = 2;

Praktični primer: Zanka *while-do*

Program:

```
while(a < 3)
  {
    b += a;
    a++;
  }
```

Diagram poteka:



Spremenljivki:

a = 2;

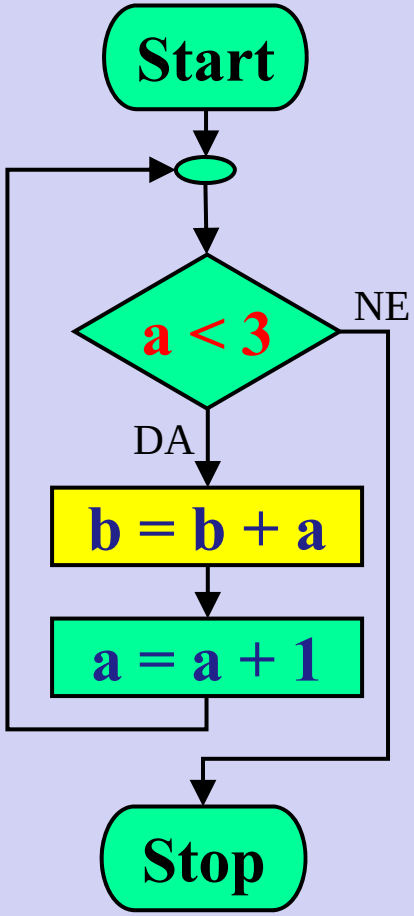
b = 2;

Praktični primer: Zanka *while-do*

Program:

```
while(a < 3)
{
   b += a;
   a++;
}
```

Diagram poteka:



Spremenljivki:

a = 2;

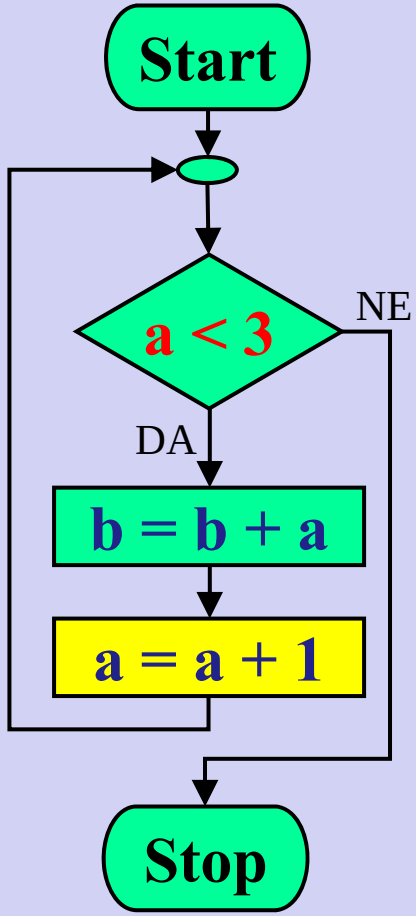
b = 4;

Praktični primer: Zanka *while-do*

Program:

```
while(a < 3)
{
   b += a;
   a++;
}
```

Diagram poteka:



Spremenljivki:

a = 3;

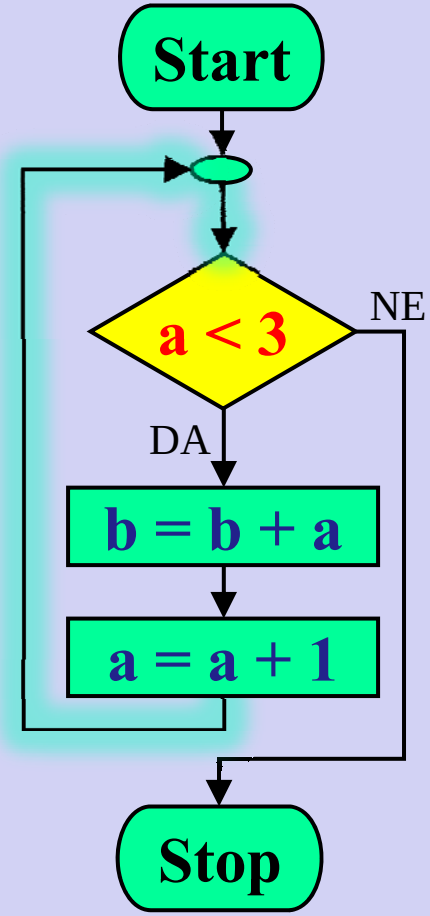
b = 4;

Praktični primer: Zanka *while-do*

Program:

```
while(a < 3)
{
    b += a;
    a++;
}
```

Diagram poteka:



Spremenljivki:

a = 3;

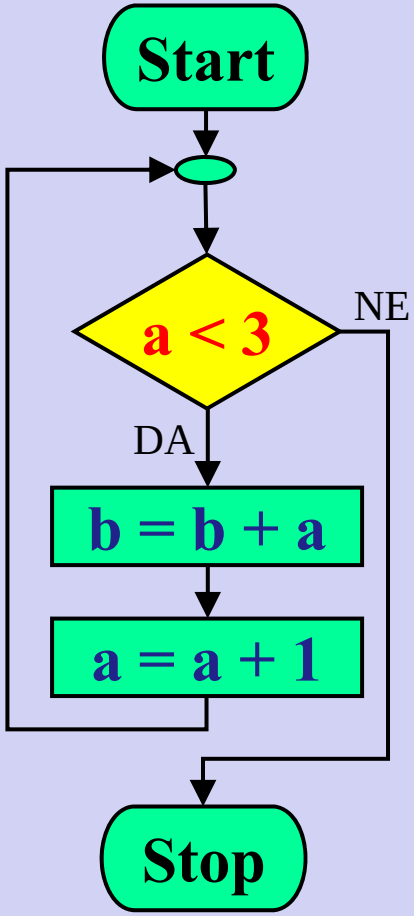
b = 4;

Praktični primer: Zanka *while-do*

Program:

```
while(a < 3)
  {
    b += a;
    a++;
  }
```

Diagram poteka:



Spremenljivki:

a = 3;

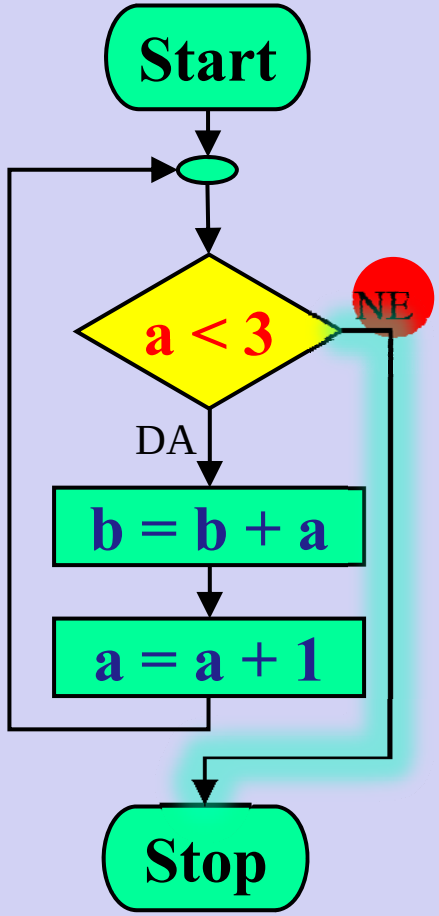
b = 4;

Praktični primer: Zanka *while-do*

Program:

```
while(a < 3)
  {
    b += a;
    a++;
  }
```

Diagram poteka:



Spremenljivki:

a = 3;

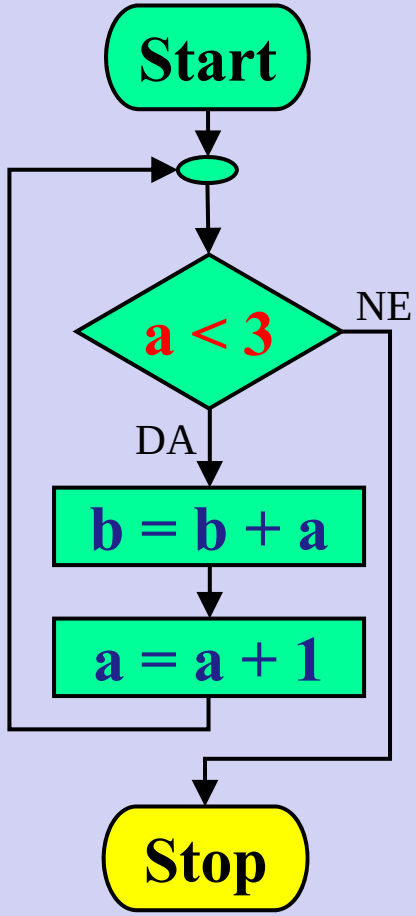
b = 4;

Praktični primer: Zanka *while-do*

Program:

```
while(a < 3)
{
   b += a;
   a++;
}
```

Diagram poteka:



Spremenljivki:

a = 3;

b = 4;

Primer: Kazalci – zamenjava vrednosti

```
…
zamenjaj(&a, &b);
…

void zamenjaj(int *s1, int *s2)
{
int zacas;
    zacas=*s1;
    *s1=*s2;
    *s2=zacas;
}
```

a	5
b	6
&a	1
&b	2
s1	nedefinirano
s2	nedefinirano
*s1	nedefinirano
*s2	nedefinirano
zacas	nedefinirano

Pomnilnik:

Naslov	Vsebina
1	5
2	6

Primer: Kazalci – zamenjava vrednosti

```
…
zamenjaj(&a, &b);
…

void zamenjaj(int *s1, int *s2)
{
int zacas;
    zacas=*s1;
    *s1=*s2;
    *s2=zacas;
}
```

a	5
b	6
&a	1
&b	2
s1	nedefinirano
s2	nedefinirano
*s1	nedefinirano
*s2	nedefinirano
zacas	nedefinirano

Pomnilnik:

Naslov	Vsebina
1	5
2	6

Primer: Kazalci – zamenjava vrednosti

```
…
zamenjaj(&a, &b);
…

void zamenjaj(int *s1, int *s2)
{
int zacas;
    zacas=*s1;
    *s1=*s2;
    *s2=zacas;
}
```

a	5
b	6
&a	1
&b	2
s1	nedefinirano
s2	nedefinirano
*s1	nedefinirano
*s2	nedefinirano
zacas	nedefinirano

Pomnilnik:

Naslov	Vsebina
1	5
2	6

Primer: Kazalci – zamenjava vrednosti

```
…
zamenjaj(&a, &b);
…

void zamenjaj(int *s1, int *s2)
{
int zacas;
    zacas=*s1;
    *s1=*s2;
    *s2=zacas;
}
```

a	5
b	6
&a	1
&b	2
s1	nedefinirano
s2	nedefinirano
*s1	nedefinirano
*s2	nedefinirano
zacas	nedefinirano

Pomnilnik:

Naslov	Vsebina
1	5
2	6

Primer: Kazalci – zamenjava vrednosti

```
…
zamenjaj(&a, &b);
…

void zamenjaj(int *s1, int *s2)
{
int zacas;
    zacas=*s1;
    *s1=*s2;
    *s2=zacas;
}
```

a	5
b	6
&a	1
&b	2
s1	nedefinirano
s2	nedefinirano
*s1	nedefinirano
*s2	nedefinirano
zacas	nedefinirano

Pomnilnik:

Naslov	Vsebina
1	5
2	6

Primer: Kazalci – zamenjava vrednosti

```
…
zamenjaj(&a, &b);
…

void zamenjaj(int *s1, int *s2)
{
int zacas;
    zacas=*s1;
    *s1=*s2;
    *s2=zacas;
}
```

a	5
b	6
&a	1
&b	2
s1	nedefinirano
s2	nedefinirano
*s1	nedefinirano
*s2	nedefinirano
zacas	nedefinirano

Pomnilnik:

Naslov	Vsebina
1	5
2	6

Primer: Kazalci – zamenjava vrednosti

```
…
zamenjaj(&a, &b);
…

void zamenjaj(int *s1, int *s2)
{
int zacas;
    zacas=*s1;
    *s1=*s2;
    *s2=zacas;
}
```

a	5
b	6
&a	1
&b	2
s1	1
s2	2
*s1	5
*s2	6
zacas	nedefinirano

Pomnilnik:

Naslov	Vsebina
1	5
2	6

Primer: Kazalci – zamenjava vrednosti

```
…
zamenjaj(&a, &b);
…

void zamenjaj(int *s1, int *s2)
{
int zacas;
    zacas=*s1;
    *s1=*s2;
    *s2=zacas;
}
```

a	5
b	6
&a	1
&b	2
s1	1
s2	2
*s1	5
*s2	6
zacas	nedefinirano

Pomnilnik:

Naslov	Vsebina
1	5
2	6

Primer: Kazalci – zamenjava vrednosti

```
…
zamenjaj(&a, &b);
…

void zamenjaj(int *s1, int *s2)
{
int zacas;
    zacas=*s1;
    *s1=*s2;
    *s2=zacas;
}
```

a	5
b	6
&a	1
&b	2
s1	1
s2	2
*s1	5
*s2	6
zacas	nedefinirano

Pomnilnik:

Naslov	Vsebina
1	5
2	6

Primer: Kazalci – zamenjava vrednosti

```
…
zamenjaj(&a, &b);
…

void zamenjaj(int *s1, int *s2)
{
int zacas;
    zacas=*s1;
    *s1=*s2;
    *s2=zacas;
}
```

a	5
b	6
&a	1
&b	2
s1	1
s2	2
*s1	5
*s2	6
zacas	nedefinirano

Pomnilnik:

Naslov	Vsebina
1	5
2	6

Primer: Kazalci – zamenjava vrednosti

```
…
zamenjaj(&a, &b);
…

void zamenjaj(int *s1, int *s2)
{
int zacas;
    zacas=*s1;
    *s1=*s2;
    *s2=zacas;
}
```

a	5
b	6
&a	1
&b	2
s1	1
s2	2
*s1	5
*s2	6
zacas	nedefinirano

Pomnilnik:

Naslov	Vsebina
1	5
2	6

Primer: Kazalci – zamenjava vrednosti

```
…
zamenjaj(&a, &b);
…

void zamenjaj(int *s1, int *s2)
{
int zacas;
    zacas=*s1;
    *s1=*s2;
    *s2=zacas;
}
```

a	5
b	6
&a	1
&b	2
s1	1
s2	2
*s1	5
*s2	6
zacas	nedefinirano

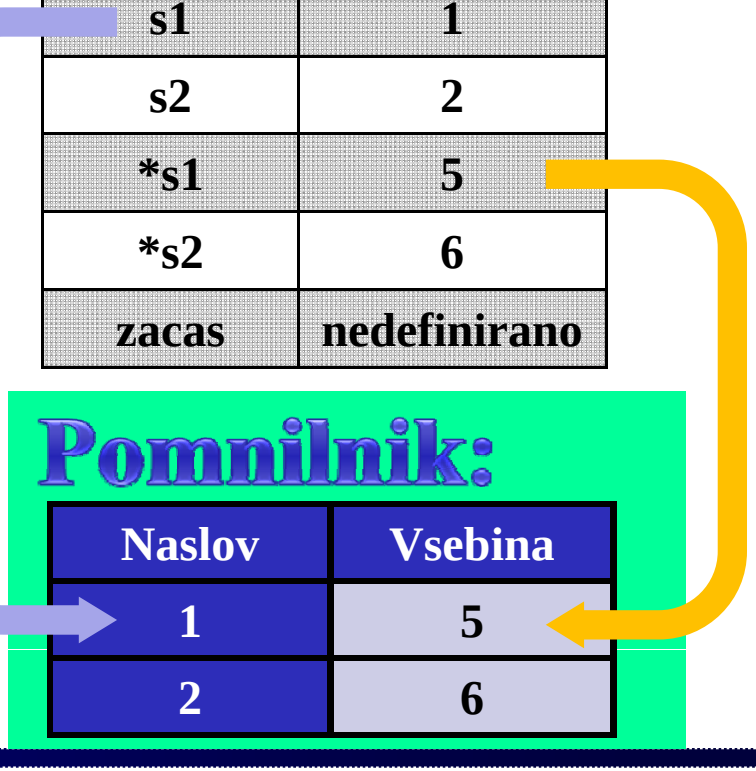
Pomnilnik:

Naslov	Vsebina
1	5
2	6

Primer: Kazalci – zamenjava vrednosti

```
…
zamenjaj(&a, &b);
…

void zamenjaj(int *s1, int *s2)
{
int zacas;
    zacas=*s1;
    *s1=*s2;
    *s2=zacas;
}
```

a	5
b	6
&a	1
&b	2
s1	1
s2	2
*s1	5
*s2	6
zacas	nedefinirano



Primer: Kazalci – zamenjava vrednosti

```
…
zamenjaj(&a, &b);
…

void zamenjaj(int *s1, int *s2)
{
int zacas;
    zacas=*s1;
    *s1=*s2;
    *s2=zacas;
}
```

a	5
b	6
&a	1
&b	2
s1	1
s2	2
*s1	5
*s2	6
zacas	nedefinirano

Primer: Kazalci – zamenjava vrednosti

```
…
zamenjaj(&a, &b);
…

void zamenjaj(int *s1, int *s2)
{
int zacas;
    zacas=*s1;
    *s1=*s2;
    *s2=zacas;
}
```

a	5
b	6
&a	1
&b	2
s1	1
s2	2
*s1	5
*s2	6
zacas	nedefinirano

Pomnilnik:

Naslov	Vsebina
1	5
2	6

Primer: Kazalci – zamenjava vrednosti

```
…
zamenjaj(&a, &b);
…

void zamenjaj(int *s1, int *s2)
{
int zacas;
    zacas=*s1;
    *s1=*s2;
    *s2=zacas;
}
```

a	5
b	6
&a	1
&b	2
s1	1
s2	2
*s1	5
*s2	6
zacas	nedefinirano

Pomnilnik:

Naslov	Vsebina
1	5
2	6

Primer: Kazalci – zamenjava vrednosti

```
…
zamenjaj(&a, &b);
…

void zamenjaj(int *s1, int *s2)
{
int zacas;
    zacas=*s1;
    *s1=*s2;
    *s2=zacas;
}
```

a	5
b	6
&a	1
&b	2
s1	1
s2	2
*s1	5
*s2	6
zacas	nedefinirano

Pomnilnik:

Naslov	Vsebina
1	5
2	6

Primer: Kazalci – zamenjava vrednosti

```
…
zamenjaj(&a, &b);
…

void zamenjaj(int *s1, int *s2)
{
int zacas;
    zacas=*s1;
    *s1=*s2;
    *s2=zacas;
}
```

a	5
b	6
&a	1
&b	2
s1	1
s2	2
*s1	5
*s2	6
zacas	nedefinirano

Pomnilnik:

Naslov	Vsebina
1	5
2	6

Primer: Kazalci – zamenjava vrednosti

```
…
zamenjaj(&a, &b);
…

void zamenjaj(int *s1, int *s2)
{
int zacas;
    zacas=*s1;
    *s1=*s2;
    *s2=zacas;
}
```

a	5
b	6
&a	1
&b	2
s1	1
s2	2
*s1	5
*s2	6
zacas	nedefinirano

Pomnilnik:

Naslov	Vsebina
1	5
2	6

Primer: Kazalci – zamenjava vrednosti

```
…
zamenjaj(&a, &b);
…

void zamenjaj(int *s1, int *s2)
{
int zacas;
    zacas=*s1;
    *s1=*s2;
    *s2=zacas;
}
```

a	5
b	6
&a	1
&b	2
s1	1
s2	2
*s1	5
*s2	6
zacas	nedefinirano

Pomnilnik:

Naslov	Vsebina
1	5
2	6

Primer: Kazalci – zamenjava vrednosti

```
…
zamenjaj(&a, &b);
…

void zamenjaj(int *s1, int *s2)
{
int zacas;
    zacas=*s1;
    *s1=*s2;
    *s2=zacas;
}
```

a	5
b	6
&a	1
&b	2
s1	1
s2	2
*s1	5
*s2	6
zacas	nedefinirano

Primer: Kazalci – zamenjava vrednosti

```
…
zamenjaj(&a, &b);
…

void zamenjaj(int *s1, int *s2)
{
int zacas;
    zacas=*s1;
    *s1=*s2;
    *s2=zacas;
}
```

a	5
b	6
&a	1
&b	2
s1	1
s2	2
*s1	5
*s2	6
zacas	nedefinirano

Primer: Kazalci – zamenjava vrednosti

```
…
zamenjaj(&a, &b);
…

void zamenjaj(int *s1, int *s2)
{
int zacas;
    zacas=*s1;
    *s1=*s2;
    *s2=zacas;
}
```

a	5
b	6
&a	1
&b	2
s1	1
s2	2
*s1	5
*s2	6
zacas	nedefinirano

Primer: Kazalci – zamenjava vrednosti

```
…
zamenjaj(&a, &b);
…

void zamenjaj(int *s1, int *s2)
{
int zacas;
    zacas=*s1;
    *s1=*s2;
    *s2=zacas;
}
```

a	5
b	6
&a	1
&b	2
s1	1
s2	2
*s1	5
*s2	6
zacas	5

Pomnilnik:

Naslov	Vsebina
1	5
2	6

Primer: Kazalci – zamenjava vrednosti

```
…
zamenjaj(&a, &b);
…

void zamenjaj(int *s1, int *s2)
{
int zacas;
    zacas=*s1;
    *s1=*s2;
    *s2=zacas;
}
```

a	6
b	6
&a	1
&b	2
s1	1
s2	2
*s1	6
*s2	6
zacas	5

Pomnilnik:

Naslov	Vsebina
1	6
2	6

Primer: Kazalci – zamenjava vrednosti

```
…
zamenjaj(&a, &b);
…

void zamenjaj(int *s1, int *s2)
{
int zacas;
    zacas=*s1;
    *s1=*s2;
    *s2=zacas;
}
```

a	6
b	5
&a	1
&b	2
s1	1
s2	2
*s1	6
*s2	5
zacas	5

Pomnilnik:

Naslov	Vsebina
1	6
2	5

Primer: Kazalci – zamenjava vrednosti

```
…
zamenjaj(&a, &b);
…

void zamenjaj(int *s1, int *s2)
{
int zacas;
    zacas=*s1;
    *s1=*s2;
    *s2=zacas;
}
```

a	6
b	5
&a	1
&b	2
s1	nedefinirano
s2	nedefinirano
*s1	nedefinirano
*s2	nedefinirano
zacas	nedefinirano

Pomnilnik:

Naslov	Vsebina
1	6
2	5

Primer: Zanka

začni izvajanje

ali je pogoj izpolnjen?

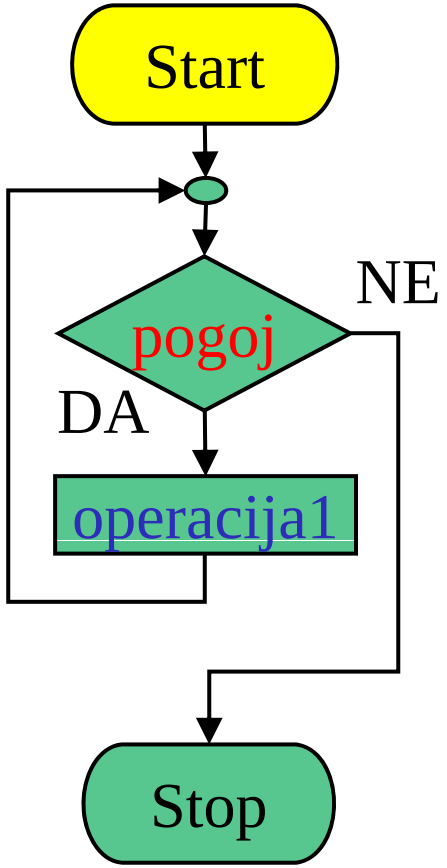
pogoj je izpolnjen

izvedi operacijo 1

vrni se pred preverjanje

pogoj ni izpolnjen

končaj izvajanje



Primer: Zanka

začni izvajanje

ali je pogoj izpolnjen?

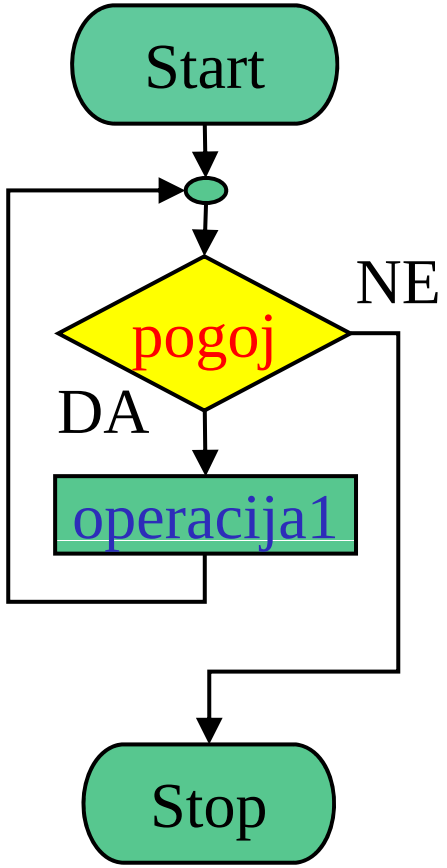
pogoj je izpolnjen

izvedi operacijo 1

vrni se pred preverjanje

pogoj ni izpolnjen

končaj izvajanje



Primer: Zanka

začni izvajanje

ali je pogoj izpolnjen?

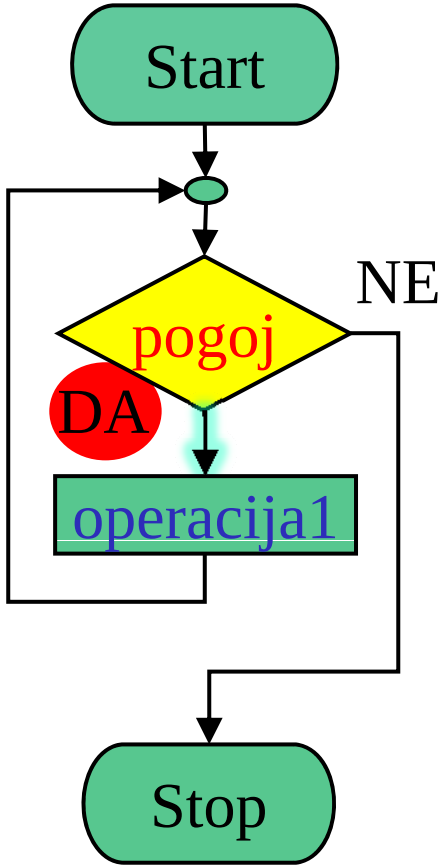
pogoj je izpolnjen

izvedi operacijo 1

vrni se pred preverjanje

pogoj ni izpolnjen

končaj izvajanje



Primer: Zanka

začni izvajanje

ali je pogoj izpolnjen?

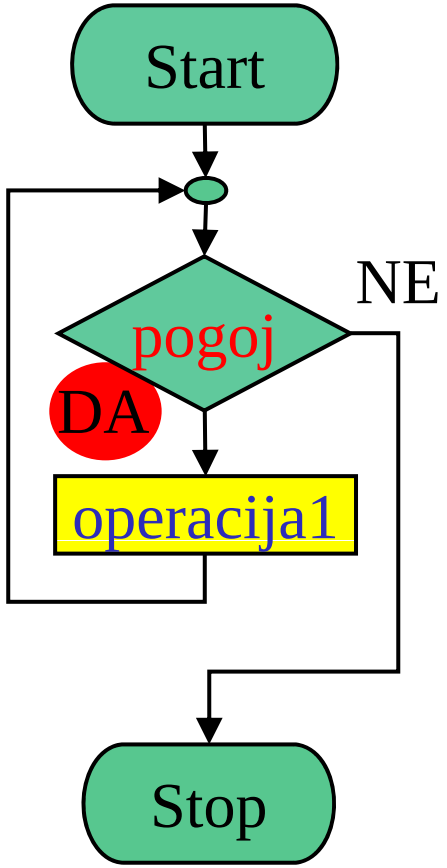
pogoj je izpolnjen

izvedi operacijo 1

vrni se pred preverjanje

pogoj ni izpolnjen

končaj izvajanje



Primer: Zanka

začni izvajanje

ali je pogoj izpolnjen?

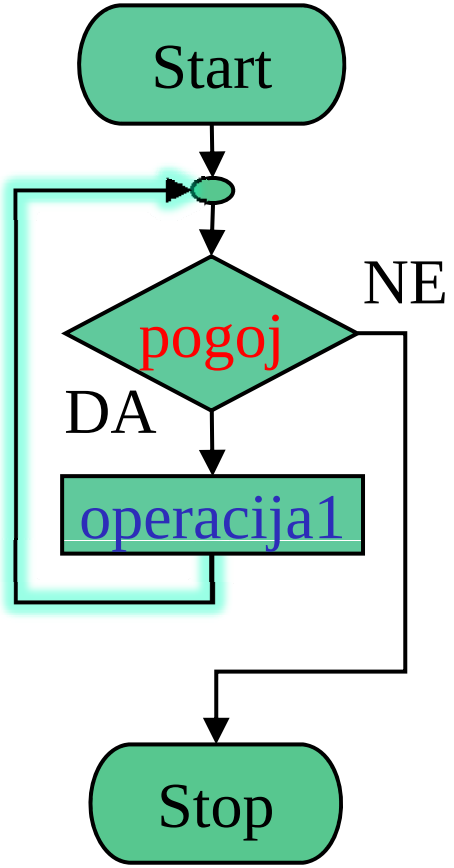
pogoj je izpolnjen

izvedi operacijo 1

vrni se pred preverjanje

pogoj ni izpolnjen

končaj izvajanje



Primer: Zanka

začni izvajanje

ali je pogoj izpolnjen?

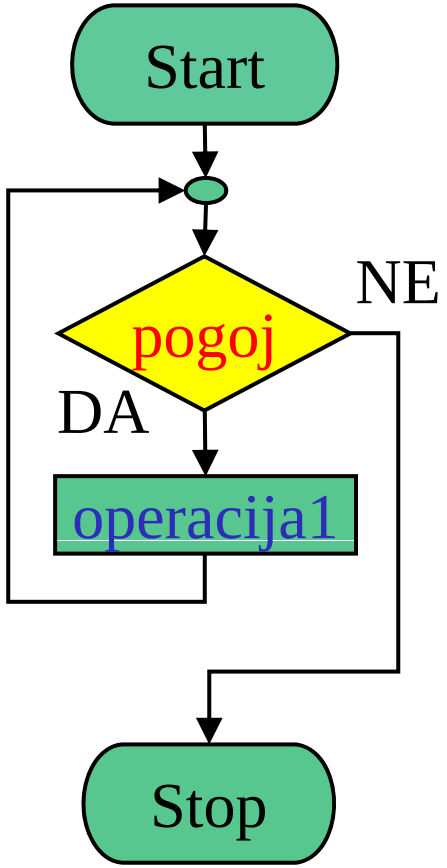
pogoj je izpolnjen

izvedi operacijo 1

vrni se pred preverjanje

pogoj ni izpolnjen

končaj izvajanje



Primer: Zanka

začni izvajanje

ali je pogoj izpolnjen?

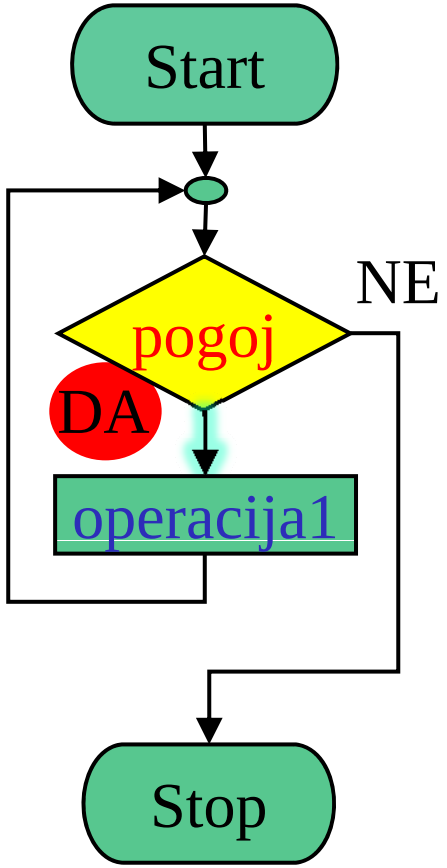
pogoj je izpolnjen

izvedi operacijo 1

vrni se pred preverjanje

pogoj ni izpolnjen

končaj izvajanje



Primer: Zanka

začni izvajanje

ali je pogoj izpolnjen?

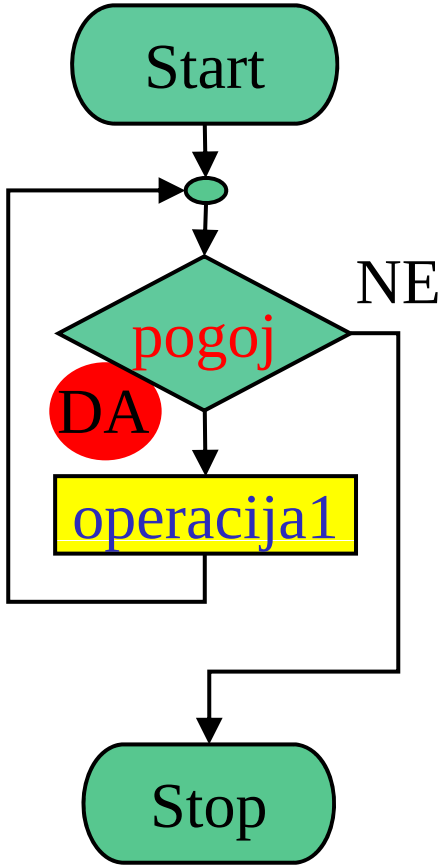
pogoj je izpolnjen

izvedi operacijo 1

vrni se pred preverjanje

pogoj ni izpolnjen

končaj izvajanje



Primer: Zanka

začni izvajanje

ali je pogoj izpolnjen?

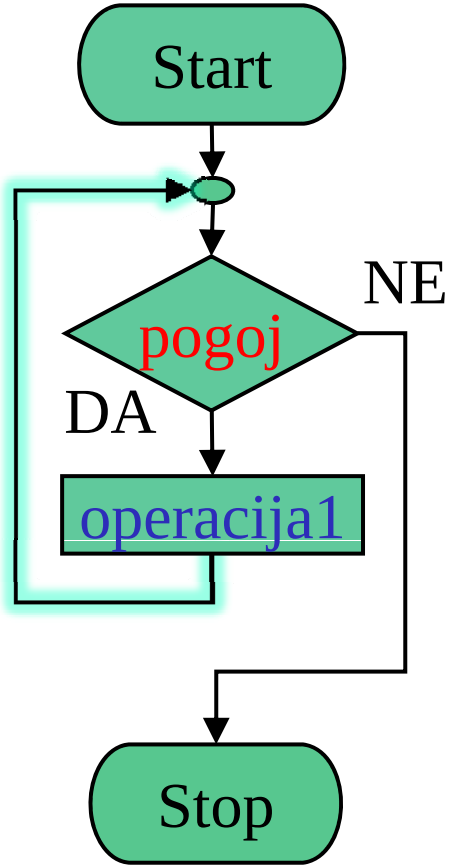
pogoj je izpolnjen

izvedi operacijo 1

vrni se pred preverjanje

pogoj ni izpolnjen

končaj izvajanje



Primer: Zanka

začni izvajanje

ali je pogoj izpolnjen?

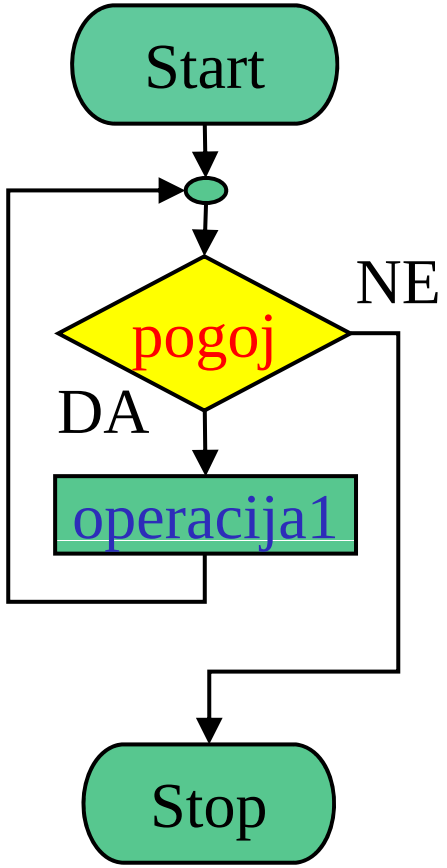
pogoj je izpolnjen

izvedi operacijo 1

vrni se pred preverjanje

pogoj ni izpolnjen

končaj izvajanje



Primer: Zanka

začni izvajanje

ali je pogoj izpolnjen?

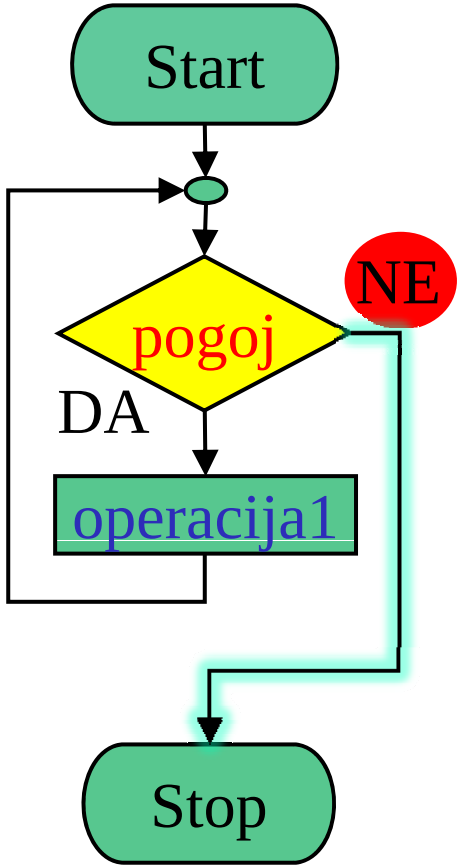
pogoj je izpolnjen

izvedi operacijo 1

vrni se pred preverjanje

pogoj ni izpolnjen

končaj izvajanje



Primer: Zanka

začni izvajanje

ali je pogoj izpolnjen?

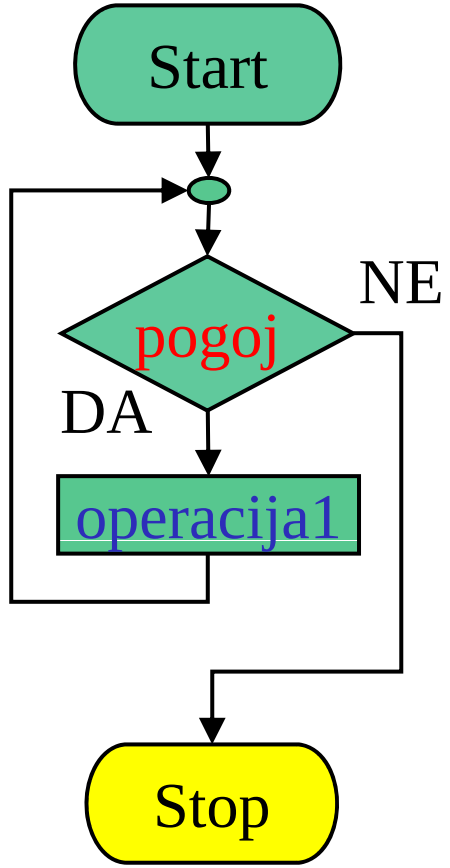
pogoj je izpolnjen

izvedi operacijo 1

vrni se pred preverjanje

pogoj ni izpolnjen

končaj izvajanje



Primer: Zanka – ponovi 2 krat

začni izvajanje

sta bili že dve ponovitvi?

ni bilo dveh ponovitev

izvedi operacijo 1

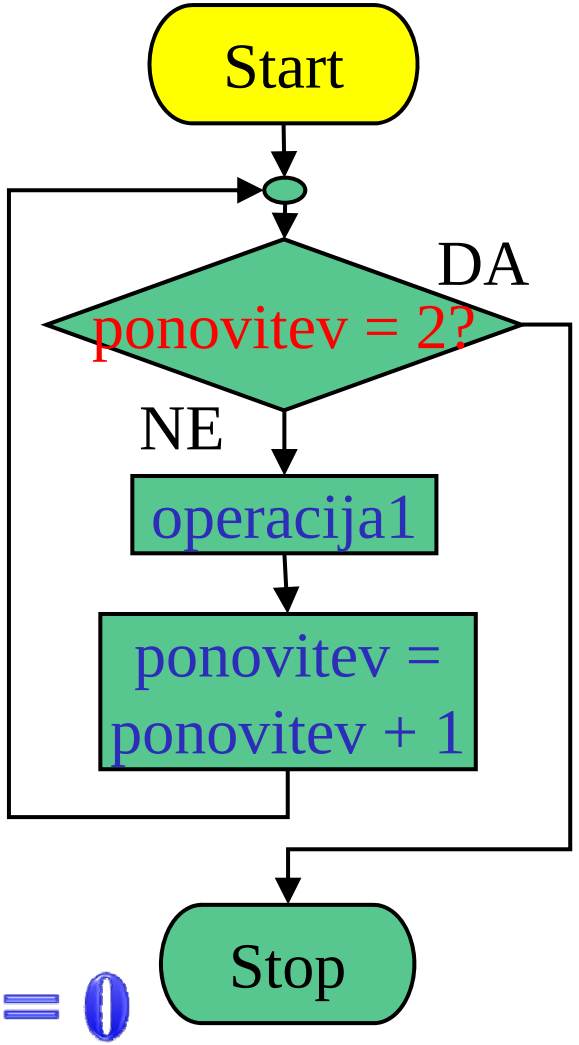
povečaj števec ponovitev

vrni se pred preverjanje

bili sta dve ponovitvi

končaj izvajanje

ponovitev = 0



Primer: Zanka – ponovi 2 krat

začni izvajanje

sta bili že dve ponovitvi?

ni bilo dveh ponovitev

izvedi operacijo 1

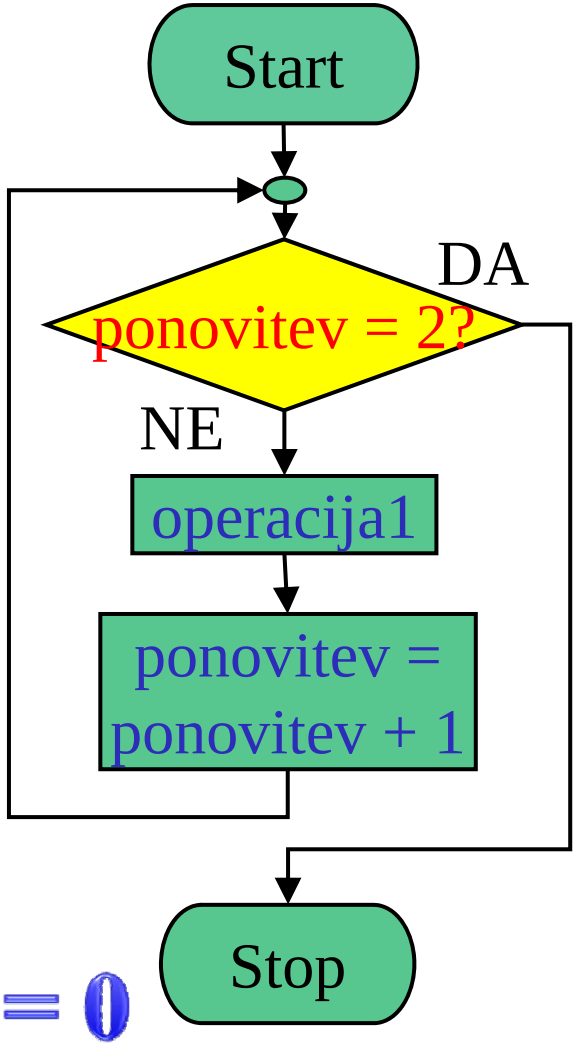
povečaj števec ponovitev

vrni se pred preverjanje

bili sta dve ponovitvi

končaj izvajanje

ponovitev = 0



Primer: Zanka – ponovi 2 krat

začni izvajanje

sta bili že dve ponovitvi?

ni bilo dveh ponovitev

izvedi operacijo 1

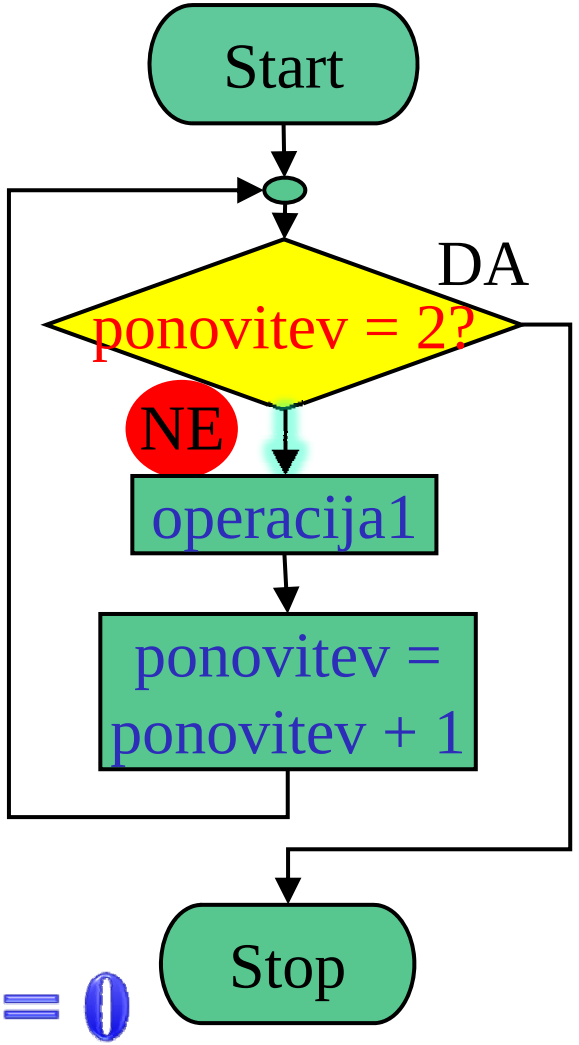
povečaj števec ponovitev

vrni se pred preverjanje

bili sta dve ponovitvi

končaj izvajanje

ponovitev = 0



Primer: Zanka – ponovi 2 krat

začni izvajanje

sta bili že dve ponovitvi?

ni bilo dveh ponovitev

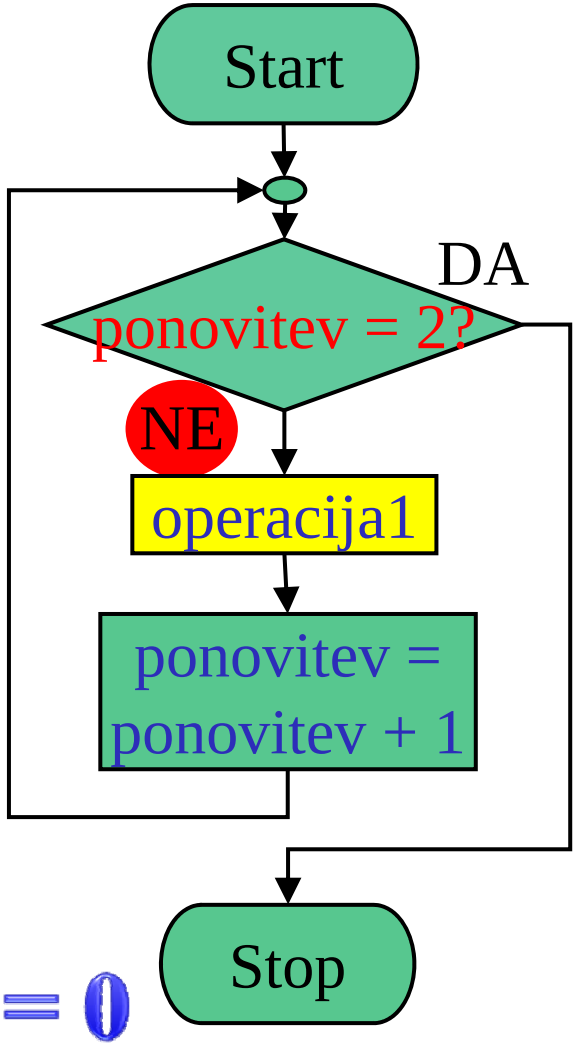
izvedi operacijo 1

povečaj števec ponovitev

vrni se pred preverjanje

bili sta dve ponovitvi

končaj izvajanje



ponovitev = 0

Primer: Zanka – ponovi 2 krat

začni izvajanje

sta bili že dve ponovitvi?

ni bilo dveh ponovitev

izvedi operacijo 1

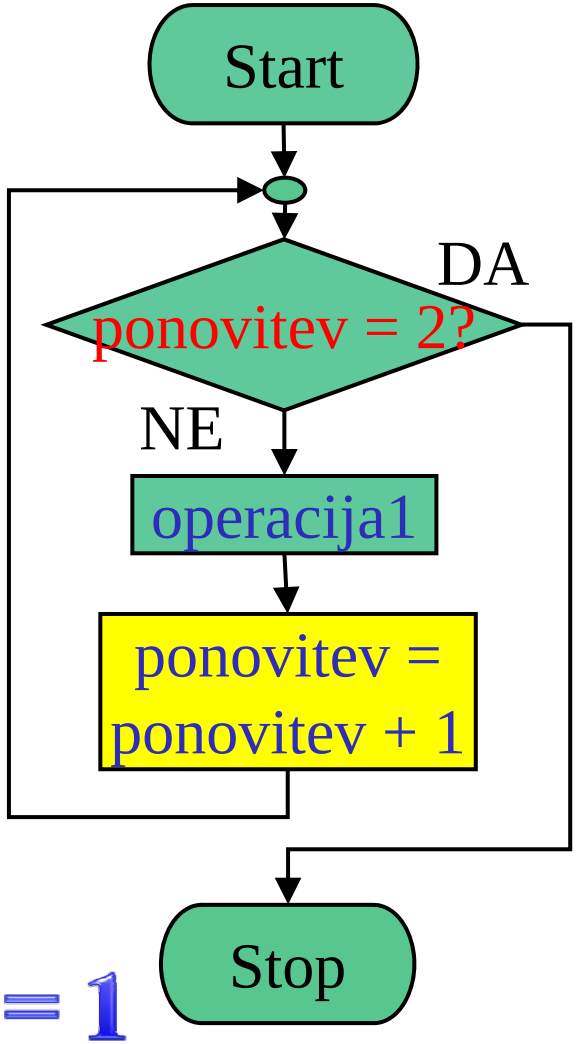
povečaj števec ponovitev

vrni se pred preverjanje

bili sta dve ponovitvi

končaj izvajanje

ponovitev = 1



Primer: Zanka – ponovi 2 krat

začni izvajanje

sta bili že dve ponovitvi?

ni bilo dveh ponovitev

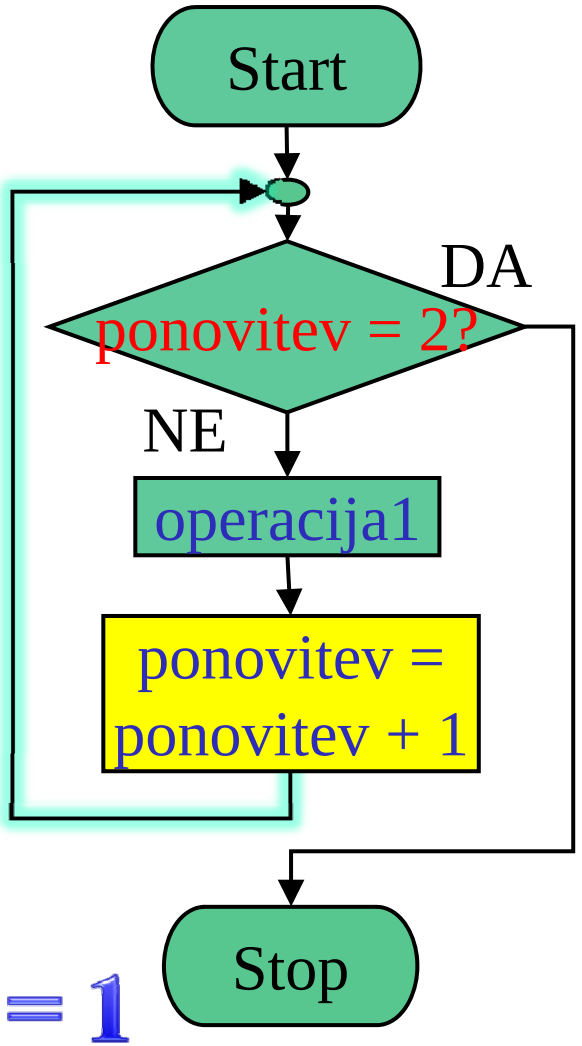
izvedi operacijo 1

povečaj števec ponovitev

vrni se pred preverjanje

bili sta dve ponovitvi

končaj izvajanje



ponovitev = 1

Primer: Zanka – ponovi 2 krat

začni izvajanje

sta bili že dve ponovitvi?

ni bilo dveh ponovitev

izvedi operacijo 1

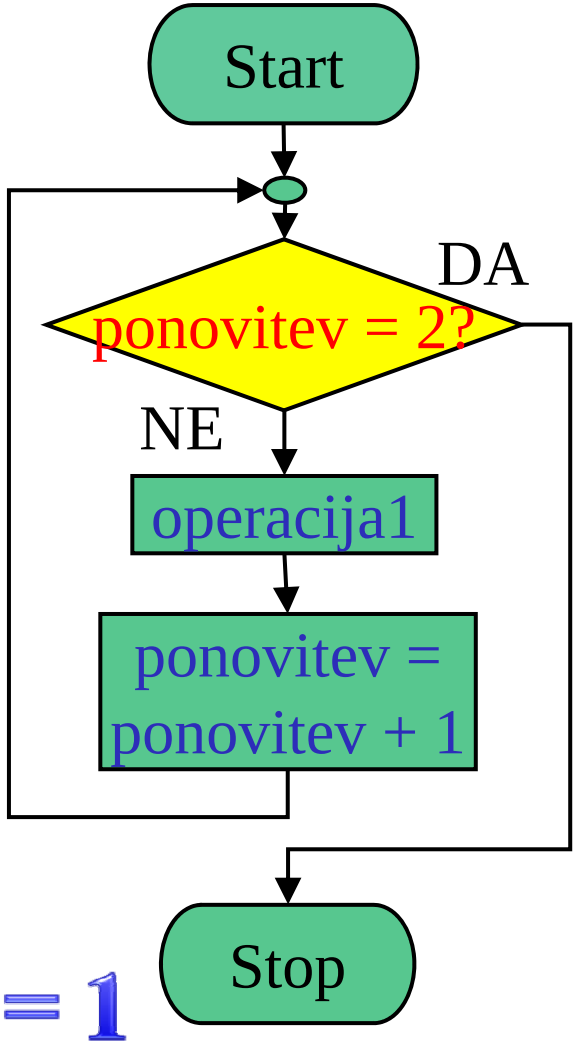
povečaj števec ponovitev

vrni se pred preverjanje

bili sta dve ponovitvi

končaj izvajanje

ponovitev = 1



Primer: Zanka – ponovi 2 krat

začni izvajanje

sta bili že dve ponovitvi?

ni bilo dveh ponovitev

izvedi operacijo 1

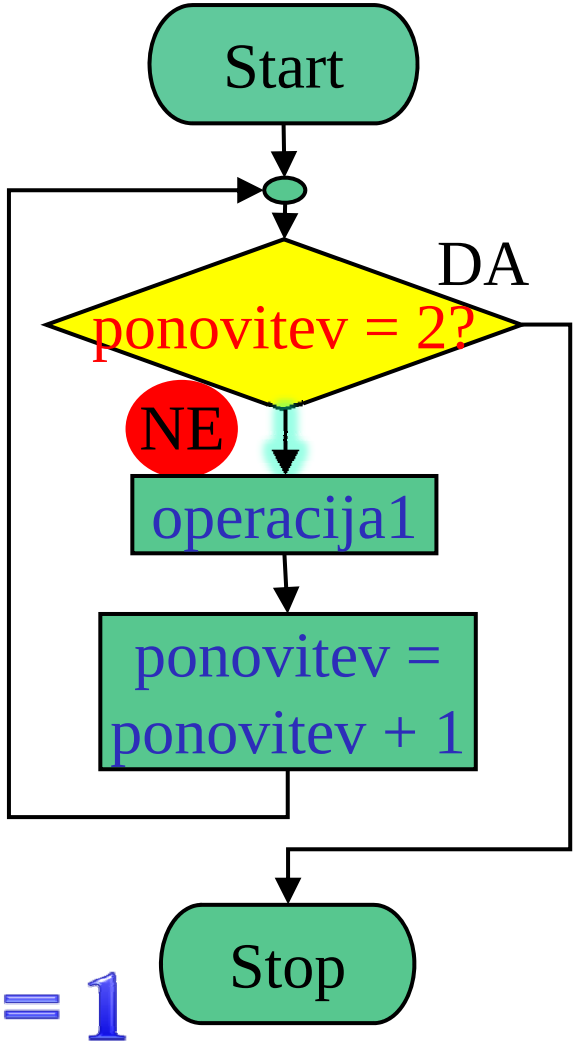
povečaj števec ponovitev

vrni se pred preverjanje

bili sta dve ponovitvi

končaj izvajanje

ponovitev = 1



Primer: Zanka – ponovi 2 krat

začni izvajanje

sta bili že dve ponovitvi?

ni bilo dveh ponovitev

izvedi operacijo 1

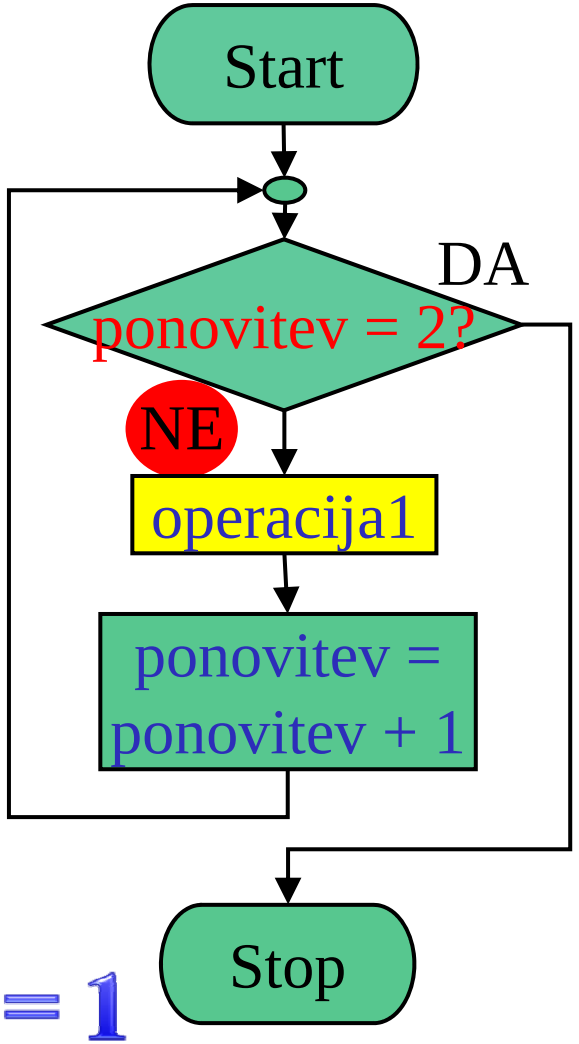
povečaj števec ponovitev

vrni se pred preverjanje

bili sta dve ponovitvi

končaj izvajanje

ponovitev = 1



Primer: Zanka – ponovi 2 krat

začni izvajanje

sta bili že dve ponovitvi?

ni bilo dveh ponovitev

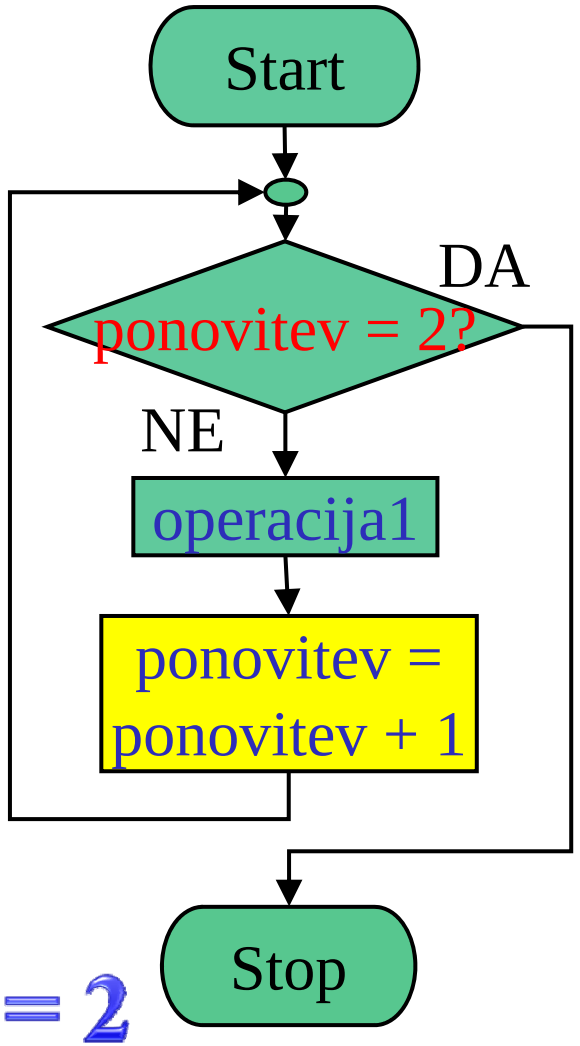
izvedi operacijo 1

povečaj števec ponovitev

vrni se pred preverjanje

bili sta dve ponovitvi

končaj izvajanje



ponovitev = 2

Primer: Zanka – ponovi 2 krat

začni izvajanje

sta bili že dve ponovitvi?

ni bilo dveh ponovitev

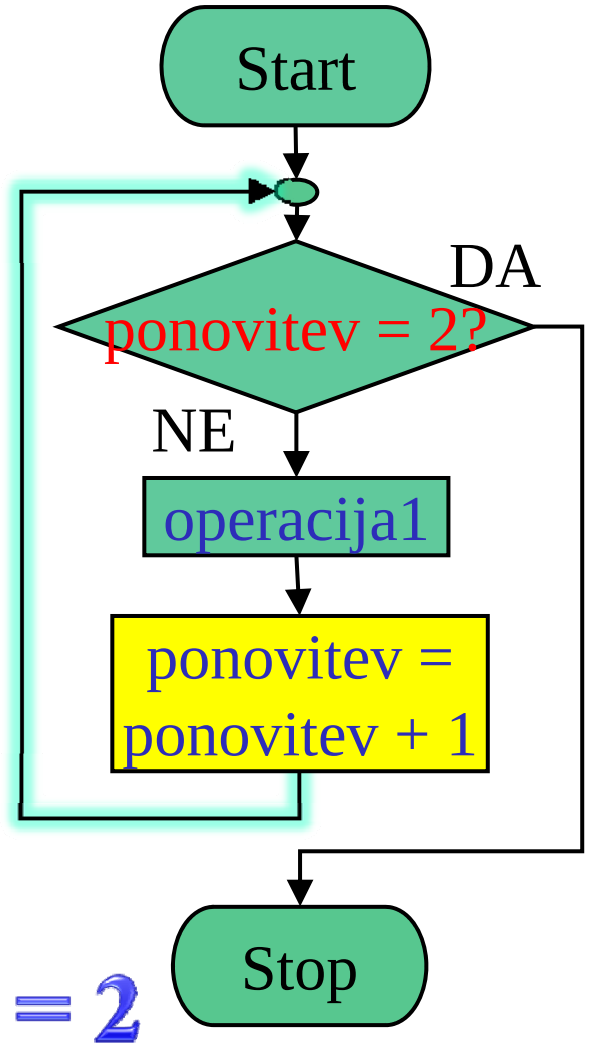
izvedi operacijo 1

povečaj števec ponovitev

vrni se pred preverjanje

bili sta dve ponovitvi

končaj izvajanje



ponovitev = 2

Primer: Zanka – ponovi 2 krat

začni izvajanje

sta bili že dve ponovitvi?

ni bilo dveh ponovitev

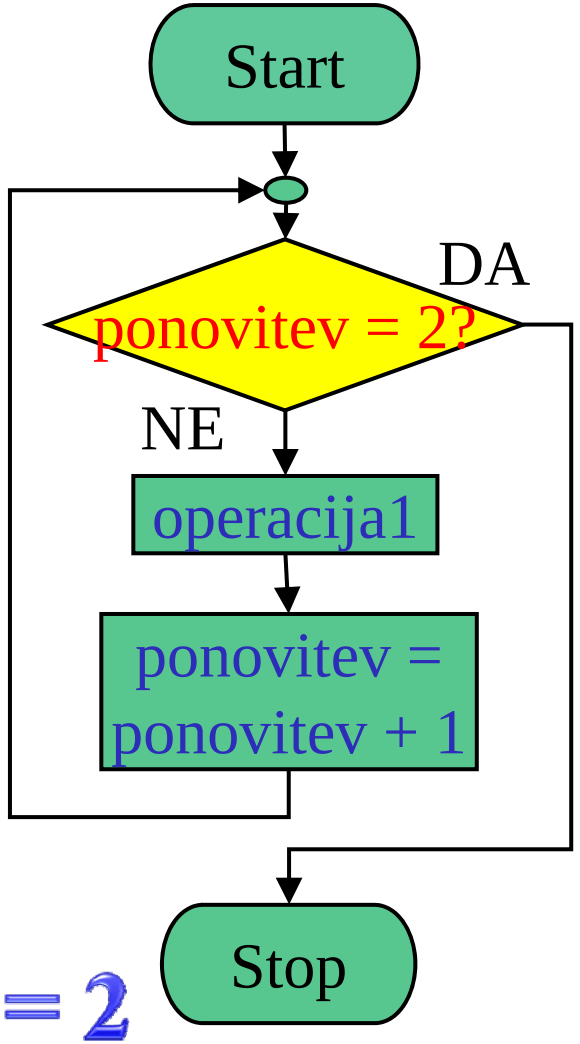
izvedi operacijo 1

povečaj števec ponovitev

vrni se pred preverjanje

bili sta dve ponovitvi

končaj izvajanje



ponovitev = 2

Primer: Zanka – ponovi 2 krat

začni izvajanje

sta bili že dve ponovitvi?

ni bilo dveh ponovitev

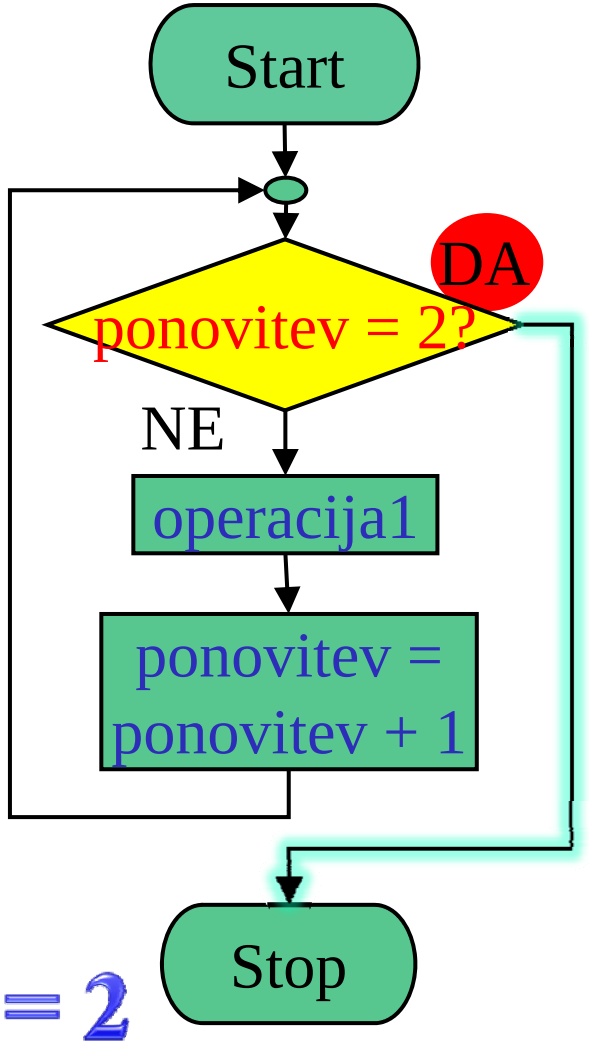
izvedi operacijo 1

povečaj števec ponovitev

vrni se pred preverjanje

bili sta dve ponovitvi

končaj izvajanje



ponovitev = 2

Primer: Zanka – ponovi 2 krat

začni izvajanje

sta bili že dve ponovitvi?

ni bilo dveh ponovitev

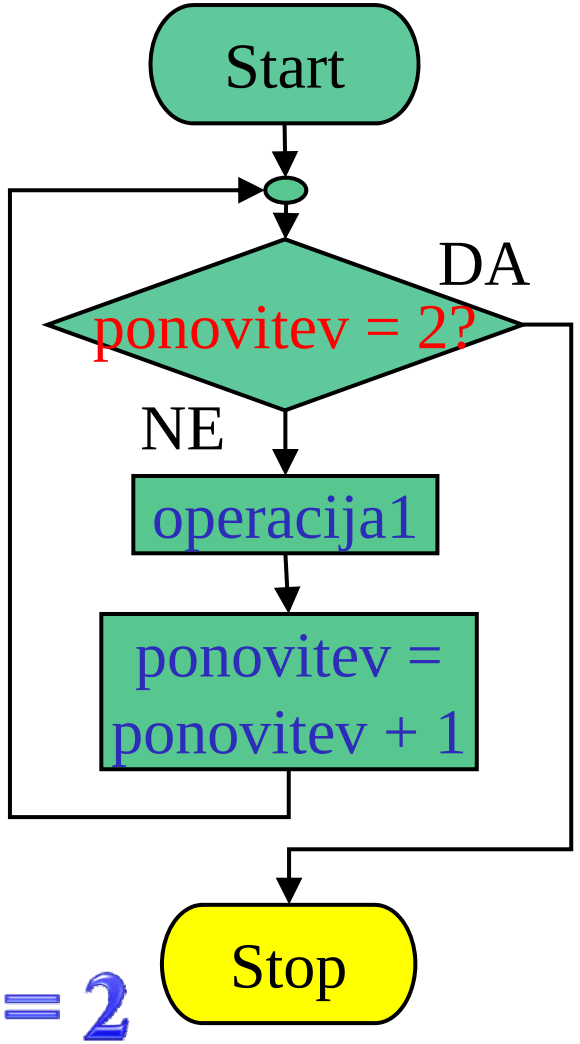
izvedi operacijo 1

povečaj števec ponovitev

vrni se pred preverjanje

bili sta dve ponovitvi

končaj izvajanje



ponovitev = 2

Bitne operacije: c = a|b

a	1	0	1	1	0	1	0	0
b	0	1	1	0	0	1	0	1
c								

Bitne operacije: c = a|b

a	1	0	1	1	0	1	0	0
b	0	1	1	0	0	1	0	1
c								**1**

0 ALI 1 = 1

Bitne operacije: c = a|b

a	1	0	1	1	0	1	0	0
b	0	1	1	0	0	1	0	1
c							**0**	**1**

0 ALI 0 = 0

Bitne operacije: c = a|b

a	1	0	1	1	0	1	0	0
b	0	1	1	0	0	1	0	1
c						**1**	**0**	**1**

1 ALI 1 = 1

Bitne operacije: c = a|b

a	1	0	1	1	0	1	0	0
b	0	1	1	0	0	1	0	1
c					**0**	**1**	**0**	**1**

0 ALI 0 = 0

Bitne operacije: c = a|b

a	1	0	1	1	0	1	0	0
b	0	1	1	0	0	1	0	1
c				**1**	**0**	**1**	**0**	**1**

1 ALI 0 = 1

Bitne operacije: c = a|b

a	1	0	1	1	0	1	0	0
b	0	1	1	0	0	1	0	1
c			**1**	**1**	**0**	**1**	**0**	**1**

1 ALI 1 = 1

Bitne operacije: c = a|b

a	1	0	1	1	0	1	0	0
b	0	1	1	0	0	1	0	1
c		**1**	**1**	**1**	**0**	**1**	**0**	**1**

0 ALI 1 = 1

Bitne operacije: c = a|b

a	1	0	1	1	0	1	0	0
b	0	1	1	0	0	1	0	1
c	**1**	**1**	**1**	**1**	**0**	**1**	**0**	**1**

1 ALI 0 = 1

Bitne operacije: c = a|b

a	1	0	1	1	0	1	0	0
b	0	1	1	0	0	1	0	1
c	**1**	**1**	**1**	**1**	**0**	**1**	**0**	**1**

Bitne operacije: c = a&b

a	1	0	1	1	0	1	0	0
b	0	1	1	0	0	1	0	1
c								

Bitne operacije: c = a&b

a	1	0	1	1	0	1	0	0
b	0	1	1	0	0	1	0	1
c								**0**

0 IN 1 = 0

Bitne operacije: c = a&b

a	1	0	1	1	0	1	0	0
b	0	1	1	0	0	1	0	1
c							**0**	**0**

0 IN 0 = 0

Bitne operacije: c = a&b

a	1	0	1	1	0	1	0	0
b	0	1	1	0	0	1	0	1
c						**1**	**0**	**0**

1 IN 1 = 1

Bitne operacije: c = a&b

a	1	0	1	1	0	1	0	0
b	0	1	1	0	0	1	0	1
c					**0**	**1**	**0**	**0**

0 IN 0 = 0

Bitne operacije: c = a&b

a	1	0	1	1	0	1	0	0
b	0	1	1	0	0	1	0	1
c				**0**	**0**	**1**	**0**	**0**

1 IN 0 = 0

Bitne operacije: c = a&b

a	1	0	1	1	0	1	0	0
b	0	1	1	0	0	1	0	1
c			**1**	**0**	**0**	**1**	**0**	**0**

1 IN 1 = 1

Bitne operacije: c = a&b

a	1	0	1	1	0	1	0	0
b	0	1	1	0	0	1	0	1
c		**0**	**1**	**0**	**0**	**1**	**0**	**0**

0 IN 1 = 0

Bitne operacije: c = a&b

a	1	0	1	1	0	1	0	0
b	0	1	1	0	0	1	0	1
c	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**0**

1 IN 0 = 0

Bitne operacije: c = a&b

a	1	0	1	1	0	1	0	0
b	0	1	1	0	0	1	0	1
c	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**0**

Bitne operacije: c = ~a

a	1	0	1	1	0	1	0	0
c								

Bitne operacije: c = ~a

a	1	0	1	1	0	1	0	0
c								

NE 0 = 1

Bitne operacije: c = ~a

a	1	0	1	1	0	1	0	0
c							**1**	**1**

NE 0 = 1

Bitne operacije: c = ~a

a	1	0	1	1	0	1	0	0
c						**0**	**1**	**1**

NE 1 = 0

Bitne operacije: c = ~a

a	1	0	1	1	0	1	0	0
c					**1**	**0**	**1**	**1**

NE 0 = 1

Bitne operacije: c = ~a

a	1	0	1	1	0	1	0	0
c				**0**	**1**	**0**	**1**	**1**

NE 1 = 0

Bitne operacije: c = ~a

a	1	0	1	1	0	1	0	0
c			**0**	**0**	**1**	**0**	**1**	**1**

NE 1 = 0

Bitne operacije: c = ~a

a	1	0	1	1	0	1	0	0
c		**1**	**0**	**0**	**1**	**0**	**1**	**1**

NE 0 = 1

Bitne operacije: c = ~a

a	1	0	1	1	0	1	0	0
c	**0**	**1**	**0**	**0**	**1**	**0**	**1**	**1**

NE 1 = 0

Bitne operacije: c = ~a

a	1	0	1	1	0	1	0	0
c	**0**	**1**	**0**	**0**	**1**	**0**	**1**	**1**