8.5. `while` stavek

1

- `while` stavek spada med zanke
- pravimo mu tudi `while` zanka
- zanke so namenjene ponavljanju stavkov
  - pravimo jim tudi iterativni stavki
  - ena iteracija je ena ponovitev zanke

Kaj je zanka?

2

- zanka v programiranju
  - del programa, ki ga izvajamo vedno znova, dokler nimamo dovoljenja, da grem iz tega dela programa in izvedemo naslednji stavek
  - je programski stavek
  - uporabljamo za ponavljanje stavkov
  - kako dolgo ponavljamo je lahko odvisno od več stvari
    - imamo nek pogoj, ki to določa
    - imamo znano število ponovitev

Zanka - primeri

3

- poglejmo naslednje primere
  1. beri cela števila, dokler ne vnesemo sodega števila
  2. beri cela števila, dokler ne vnesemo sodega števila in preštej, koliko števil smo skupno vnesli
  3. na ekran izriši 99 zvezdic
- od česa je odvisna ponovitev
  1. od sodosti prebranega števila
  2. od sodosti prebranega števila
  3. od števila izrisanih zvezdic

while zanka

4

- zamislimo si naslednjo situacijo
  - v učilnici imamo skupino računalnikov
  - mi moramo izklopiti vse vklopljene računalnike
  - kako bi pristopili k problemu?
    1. Preveri, ali je v učilnici kak vklopljen računalnik?
       Če je, nadaljuj na koraku št. 2, v nasprotnem primeru pojdi iz učilnice.
    2. Ugasni vklopljen računalnik. Vrni se na korak št. 1.
  - če v učilnici že na začetku ni vklopljenih računalnikov, nimamo dela

Sintaksa `while` zanke

5

- sintaksa `while` zanke:

```
while (pogoj)
  stavek;
```

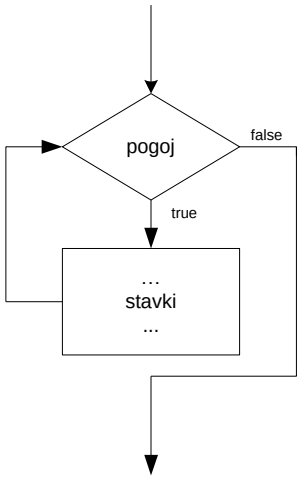
  - kadar želimo ponavljati več stavkov, uporabimo sestavljeni stavek:

```
while (pogoj) {
  ...
  stavki
  ...
}
```

Kako deluje while zanka? (1)

6

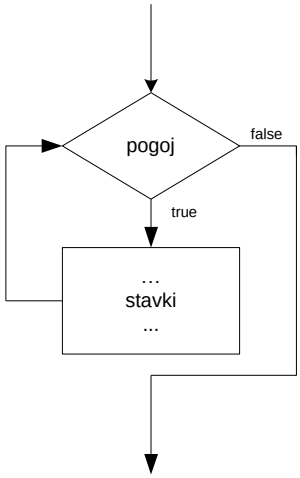
- delovanje while zanke:
  1. Ovrednoti pogoj.
  2. Če je pogoj enak true, izvedi stavek v telesu zanke. V nasprotnem primeru nadaljuj z izvajanjem za zanko.
  3. Vrni se na korak 1.



Kako deluje while zanka? (2)

7

- posledica oz. rezultat:
  - Telo zanke se izvaja tako dolgo, dokler se pogoj ne ovrednoti na false.
  - Ko se pogoj ovrednoti na false, se izvajanje nadaljuje na naslednjem stavku.
  - Na ta način se stavek v telesu zanke lahko izvede nič- ali večkrat.



Primer 06

8

```
#include <iostream>
using namespace std;

int main()
{
    int i=1, vsota = 0;

    while (i<=10) {
        vsota += i;
        ++i;
    }

    cout << "Vsota je " << vsota << endl;
    return 0;
}
```

V `while` zanki povečujemo vrednost spremenljivke `vsota` za trenutno vrednost spremenljivke `i`. Zatem povečamo vrednost spremenljivke `i` za 1. Ko se je telo zanke izvedlo že 10-krat, ima `i` vrednost 11, zato se pogoj `(i<=10)` ovrednoti na `false`. S tem se zanka preneha izvajati in se izvajanje nadaljuje na naslednjem stavku (`cout`).

while zanka – primer in problem

9

```
#include <iostream>
using namespace std;

int main()
{
    // beremo cela števila, dokler ne vnesemo sodega števila

    int stevilo;

    // ponavljamo, če število ni sodo
    while (stevilo % 2 != 0) {
        cout << "Vnesi celo sodo število: ";
        cin >> stevilo;
    }

    return 0;
}
```

Ne vemo, koliko je število!

Kaj se zgodi, če je v pomnilniku ravno sodo število?
Potem se zanka nikoli ne izvede – nikoli ne vtipkamo števila.

while zanka – primer 07

10

```
#include <iostream>
using namespace std;

int main()
{
    // beremo cela števila, dokler ne vnesemo sodega števila
    // zagotovimo, da število ni sodo
    int stevilo = 1;

    // ponavljamo, če število ni sodo
    while (stevilo % 2 != 0) {
        cout << "Vnesi celo sodo število: ";
        cin >> stevilo;
    }

    return 0;
}
```

8.6. do…while stavek

11

- v prejšnjem primeru smo želeli, da se zanka vsaj enkrat izvede
- to dosežemo z *do…while* stavkom
- tudi do…while stavek spada med zanke
- je zelo podoben while zanki
- razlika med do…while in while zanko je:
  - v do..while zanki pogoj vrednotimo na koncu zanke
  - v while zanki pogoj vrednotimo na začetku zanke
- ker se pogoj v do…while zanki preverja na koncu, se telo zanke zagotovo izvede vsaj enkrat

Sintaksa do…while zanke

12

- sintaksa do...while zanke:

```
do
  stavek;
while (pogoj);
```

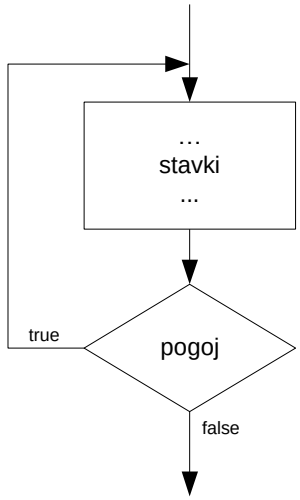
  - kadar želimo ponavljati več stavkov, uporabimo sestavljeni stavek:

```
do {
  ...
  stavki
  ...
} while (pogoj);
```

Kako deluje do…while zanka? (1)

13

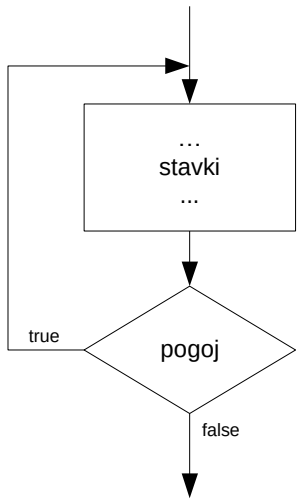
- delovanje `do..while` zanke:
  1. Izvedi stavke v zanki.
  2. Ovrednoti pogoj.
  3. Če je pogoj enak `true`, se vrni na korak 1. V nasprotnem primeru nadaljuj z izvajanjem za zanko.



Kako deluje do…while zanka? (2)

14

- posledica oz. rezultat:
  - Telo zanke se izvaja tako dolgo, dokler se pogoj ne ovrednoti na false.
  - Ko se pogoj ovrednoti na false, se izvajanje nadaljuje na naslednjem stavku.
  - Na ta način se stavki v telesu zanke izvedejo enkrat ali večkrat.



Primer 08

15

```
/*
    Program bere celo število tako dolgo,
    dokler ne vnesemo pozitivnega števila.
*/
#include <iostream>
using namespace std;

int main()
{
    int stevilo;

    do {
        cout << "Vnesi pozitivno celo stevilo: ";
        cin >> stevilo;
    } while(stevilo<=0);

    cout << "Vnesel si naslednje poz. stevilo: " << stevilo;
    return 0;
}
```

Primer 09 – opis problema

16

- Napišite program, ki bere cela števila tako dolgo, dokler ne vnesemo sodega celega števila. Program naj **prešteje**, koliko števil smo skupno vnesli.
- za štetje potrebujemo spremenljivko, ki ji pravimo **števec**

Kaj je števec?

17

- števec je spremenljivka s katero štejemo
  - običajno je to celo število
- primeri (prejšnja prosojnica):
  - primer 09 – prešteti moramo koliko števil smo prebrali
  - primer 10 – štejemo število izrisanih zvezdic
- vsakemu števcu določimo
  - začetno vrednost
  - korak s katerim se spremeni
  - kdaj se števec spremeni

Primer 09

18

```
#include <iostream>
using namespace std;

int main()
{
    int stevilo;
    int stevec = 0; // števec postavimo na začetno vrednost 0
                    // saj do sem še nismo prebrali nobenega števila

    do {
        cout << "Vnesi celo sodo stevilo: ";
        cin >> stevilo;    // preberemo število
        stevec = stevec + 1;  // zato na tem mestu povečamo števec
                              // imamo eno število več
    } while (stevilo % 2 != 0); // ponovimo, če število ni sodo

    cout << "Skupno si vnesel " << stevec << " stevil.";
    return 0;
}
```

Primer 10

19

```
// na ekran izriši 30 zvezdic
#include <iostream>
using namespace std;

int main()
{
    int stZvezdic = 0; // števec postavimo na začetno vrednost 0
                       // nismo še izrisali nobene zvezdice

    do {
        cout << "*";           // izrišemo eno zvezdico
        stZvezdic++;           // na ekranu je ena zvezdica več
    } while (stZvezdic < 30);  // ponovimo, če je na ekranu manj
                               // kot 30 zvezdic
    return 0;
}
```

8.7. for stavek

20

- v prejšnjem primeru smo vedeli, **kolikokrat** se zanka ponovi
- ustreznejša izbira je for zanka
- je zanka namenjena za štetje
- sintaksa

```
for (inicializacija; pogoj; sprememba) {
  stavki;
}
```

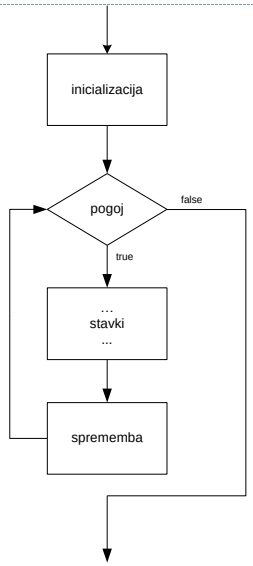
- če imamo v telesu zanke samo en stavek, lahko seveda zavite oklepaje izpustimo

for zanka – kako deluje?

21

```
for (inicializacija; pogoj; sprememba) {
     stavki;
};
```

1. Izvedi inicializacijo.
2. Ovrednoti pogoj. Če je pogoj resničen (*true*), skoči na korak 3. V nasprotnem primeru nadaljuj za zanko.
3. Izvedi stavke.
4. Izvedi spremembo.
5. Skoči na korak 1.

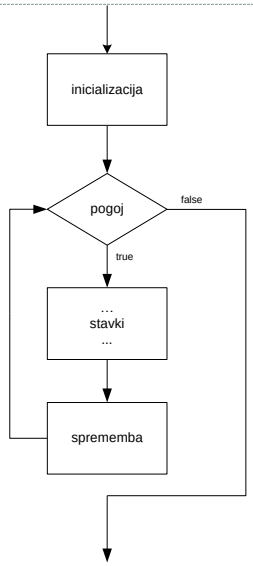


for zanka – kako deluje?

22

```
for (inicializacija; pogoj; sprememba) {
     stavki;
};
```

- inicializacija je lahko prazen stavek ali več stavkov ločenih z **vejicami**
- pogoj je nek logični izraz
- sprememba je lahko nič, en ali več stavkov ločenih z **vejicami**
- podpičja v for( ) pišemo vedno



Primer 11


```
#include <iostream>
using namespace std;

int main()
{
    int vsota = 0;

    for(int i=1; i<=10; i++)
        vsota = vsota + i;

    cout << "Vsota stevil od 1 do 10 je " << vsota;
    return 0;
}
```

še o for zanki...

- inicializacija je lahko tudi deklaracijski stavek
  - spremenljivka deklarirana v for zanki je dosegljiva le znotraj te zanke
- katerikoli del znotraj ( ) - glave for - zanke so lahko prazni, moramo pa pisati podpičja ;
- pomen praznih stavkov v glavi for zanke:
  - inicializacija ... če jo izpustimo, se ne izvede nobena inicializacija ob začetku izvajanja zanke
  - pogoj ... če izpustimo pogoj, velja pravilo, da je vedno true
  - sprememba ... če jo izpustimo, se ne izvede nobena sprememba

še več o for zanki...

25

- inicializacija in sprememba lahko vsebujeta več stavkov
  - v tem primeru jih ne ločimo s podpičjem, temveč z vejicami

PRIMER:

```
for(int i=1, vsota=0;  ; i++)
  vsota = vsota + i;
```

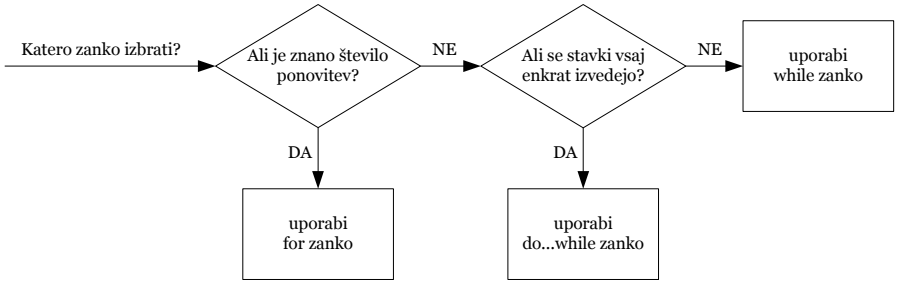
zanke – katero naj uporabim?

26

- pri zankah moramo določiti
  1. kaj se ponavlja – kateri stavki
  2. kdaj se ponavlja
     - bodisi je znano število ponovitev
     - bodisi je podan pogoj ustavitve
- na podlagi točke 2 se odločimo za ustrezno zanko (diagram na naslednji strani)

zanke – katero naj uporabim?

27



Gnezdene zanke

28

- stavki znotraj zank so poljubni stavki
  - deklaracijski stavek
  - prireditveni stavek
  - pogojni stavek
  - zanka
  - ...
- zanka znotraj zanke – gnezdena zanka

Primer 12 – opis problema

- Napišite program, ki prebere 10 celih pozitivnih števil in izračuna vsoto prebranih števil. Če uporabnik vnese negativno število ali število 0, se naj branje števila ponavlja, dokler prebrano število ne ustreza zahtevam.

Primer 12


```
#include <iostream>
using namespace std;

int main()
{
    int vsota = 0, stevilo;
    cout << "Vnesi 10 celih pozitivnih stevil: " << endl;

    for(int i=0; i<10; i++) {
        // ponavljaj branje, dokler število ni pozitivno
        do {
           cout << "Vnesi celo poz. stevilo: ";
           cin >> stevilo;
        } while (stevilo <= 0);

        vsota = vsota + stevilo;
    }

    cout << "Vsota prebranih 10 celih poz. stevil je " << vsota << endl;
    return 0;
}
```

Primer 13 – opis problema

31

- Napiši program, ki prebere dolžino *a* in širino *b* pravokotnika. Nato na ekran izpišite pravokotnik zvezdic, ki bo imel v vsaki vrstici *a* zvezdic, vsega skupaj pa naj bo *b* vrstic.

PRIMER: a=6, b=4

```
******
******
******
******
```

Primer 13 – koraki do rešitve

32

Začetni algoritem:

- `ponovi b-krat`
  - `izriši vrstico`

Podrobnejši algoritem:

- `ponovi b-krat`
  - `ponovi a-krat`
    - `izriši *`
  - `skoči v novo vrstico`

Primer 13 (1/2)

33

```
#include <iostream>
using namespace std;

int main() {
  cout << "Vnesi stevilo zvezdic v vrstici: ";
  int a;
  cin >> a;

  cout << "Vnesi stevilo vrstic: ";
  int b;
  cin >> b;
```

Primer 13 (2/2)

34

```
  for(int v=0; v<b; v++) {       // ponovi b-krat
       for(int s=0; s<a; s++) {  // ponovi a-krat
              cout << "*";
       }
       cout << endl;             // skoci v novo vrstico
  }

  return 0;
}
```

Primer 13 - delovanje

35

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

pomnilnik

a	4
b	2
v	?
s	?

izpis na ekran

_

Primer 13 - delovanje

36

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

pomnilnik

a	4
b	2
v	0
s	?

izpis na ekran

_

Primer 13 - delovanje

37

Delovanje na primeru a=4, b=2.

```
                    true
  for(int v=0; v<b; v++) {
       for(int s=0; s<a; s++) {
              cout << "*";
       }
       cout << endl;
  }
```

pomnilnik

a	4
b	2
v	0
s	?

izpis na ekran

_

Primer 13 - delovanje

38

Delovanje na primeru a=4, b=2.

```
  for(int v=0; v<b; v++) {
       for(int s=0; s<a; s++) {
              cout << "*";
       }
       cout << endl;
  }
```

pomnilnik

a	4
b	2
v	0
s	0

izpis na ekran

_

Primer 13 - delovanje

39

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

true

pomnilnik

a	4
b	2
v	0
s	0

izpis na ekran

_

Primer 13 - delovanje

40

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

pomnilnik

a	4
b	2
v	0
s	0

izpis na ekran

*_

Primer 13 - delovanje

41

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

pomnilnik

a	4
b	2
v	0
s	0

izpis na ekran

```
*
```

Primer 13 - delovanje

42

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

pomnilnik

a	4
b	2
v	0
s	1

izpis na ekran

```
*
```

Primer 13 - delovanje

43

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

true

pomnilnik

a	4
b	2
v	0
s	1

izpis na ekran

```
*
```

Primer 13 - delovanje

44

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

pomnilnik

a	4
b	2
v	0
s	1

izpis na ekran

```
**
```

Primer 13 - delovanje

45

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

pomnilnik

a	4
b	2
v	0
s	1

izpis na ekran

```
**
```

Primer 13 - delovanje

46

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

pomnilnik

a	4
b	2
v	0
s	2

izpis na ekran

```
**
```

Primer 13 - delovanje

47

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

true

pomnilnik

a	4
b	2
v	0
s	2

izpis na ekran

```
**
```

Primer 13 - delovanje

48

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

pomnilnik

a	4
b	2
v	0
s	2

izpis na ekran

```
***
```

Primer 13 - delovanje

49

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

pomnilnik

a	4
b	2
v	0
s	2

izpis na ekran

```
***
```

Primer 13 - delovanje

50

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

pomnilnik

a	4
b	2
v	0
s	3

izpis na ekran

```
***
```

Primer 13 - delovanje

51

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

true

pomnilnik

a	4
b	2
v	0
s	3

izpis na ekran

```
***
  _
```

Primer 13 - delovanje

52

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

pomnilnik

a	4
b	2
v	0
s	3

izpis na ekran

```
****
   _
```

Primer 13 - delovanje

53

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

pomnilnik

a	4
b	2
v	0
s	3

izpis na ekran

```
****
```

Primer 13 - delovanje

54

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

pomnilnik

a	4
b	2
v	0
s	4

izpis na ekran

```
****
```

Primer 13 - delovanje

55

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

false

pomnilnik

a	4
b	2
v	0
s	4

izpis na ekran

```
****
```

Primer 13 - delovanje

56

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

pomnilnik

a	4
b	2
v	0
s	?

izpis na ekran

```
****

```

Primer 13 - delovanje

57

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
          cout << "*";
     }
     cout << endl;
}
```

pomnilnik

a	4
b	2
v	0
s	?

izpis na ekran

```
****
_
```

Primer 13 - delovanje

58

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
          cout << "*";
     }
     cout << endl;
}
```

pomnilnik

a	4
b	2
v	1
s	?

izpis na ekran

```
****
_
```

Primer 13 - delovanje

59

Delovanje na primeru a=4, b=2.

```
                    true
  for(int v=0; v<b; v++) {
       for(int s=0; s<a; s++) {
              cout << "*";
       }
       cout << endl;
  }
```

pomnilnik

a	4
b	2
v	1
s	?

izpis na ekran

```
****
_
```

Primer 13 - delovanje

60

Delovanje na primeru a=4, b=2.

```
  for(int v=0; v<b; v++) {
       for(int s=0; s<a; s++) {
              cout << "*";
       }
       cout << endl;
  }
```

pomnilnik

a	4
b	2
v	1
s	0

izpis na ekran

```
****
_
```

Primer 13 - delovanje

61

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

true

pomnilnik

a	4
b	2
v	1
s	0

izpis na ekran

```
****
_
```

Primer 13 - delovanje

62

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

pomnilnik

a	4
b	2
v	1
s	0

izpis na ekran

```
****
*
 _
```

Primer 13 - delovanje

63

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

pomnilnik

a	4
b	2
v	1
s	0

izpis na ekran

```
****
*
```

Primer 13 - delovanje

64

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

pomnilnik

a	4
b	2
v	1
s	1

izpis na ekran

```
****
*
```

Primer 13 - delovanje

65

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

true

pomnilnik

a	4
b	2
v	1
s	1

izpis na ekran

```
****
*
```

Primer 13 - delovanje

66

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

pomnilnik

a	4
b	2
v	1
s	1

izpis na ekran

```
****
**
```

Primer 13 - delovanje

67

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

pomnilnik

a	4
b	2
v	1
s	1

izpis na ekran

```
****
**_
```

Primer 13 - delovanje

68

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

pomnilnik

a	4
b	2
v	1
s	2

izpis na ekran

```
****
**_
```

Primer 13 - delovanje

69

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

true

pomnilnik

a	4
b	2
v	1
s	2

izpis na ekran

```
****
**_
```

Primer 13 - delovanje

70

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

pomnilnik

a	4
b	2
v	1
s	2

izpis na ekran

```
****
***_
```

Primer 13 - delovanje

71

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

pomnilnik

a	4
b	2
v	1
s	2

izpis na ekran

```
****
***_
```

Primer 13 - delovanje

72

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

pomnilnik

a	4
b	2
v	1
s	3

izpis na ekran

```
****
***_
```

Primer 13 - delovanje

73

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

true

pomnilnik

a	4
b	2
v	1
s	3

izpis na ekran

```
****
***_
```

Primer 13 - delovanje

74

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

pomnilnik

a	4
b	2
v	1
s	3

izpis na ekran

```
****
****_
```

Primer 13 - delovanje

75

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

pomnilnik

a	4
b	2
v	1
s	3

izpis na ekran

```
****
****
```

Primer 13 - delovanje

76

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

pomnilnik

a	4
b	2
v	1
s	4

izpis na ekran

```
****
****
```

Primer 13 - delovanje

77

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

false

pomnilnik

a	4
b	2
v	1
s	4

izpis na ekran

```
****
****
```

Primer 13 - delovanje

78

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

pomnilnik

a	4
b	2
v	1
s	?

izpis na ekran

```
****
****
```

Primer 13 - delovanje

79

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

pomnilnik

a	4
b	2
v	1
s	?

izpis na ekran

```
****
****
_
```

Primer 13 - delovanje

80

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
            cout << "*";
     }
     cout << endl;
}
```

pomnilnik

a	4
b	2
v	2
s	?

izpis na ekran

```
****
****
_
```

Primer 13 - delovanje

81

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
          cout << "*";
     }
     cout << endl;
}
```

false

pomnilnik

a	4
b	2
v	2
s	?

izpis na ekran

```
****
****
_
```

Primer 13 - delovanje

82

Delovanje na primeru a=4, b=2.

```
for(int v=0; v<b; v++) {
     for(int s=0; s<a; s++) {
          cout << "*";
     }
     cout << endl;
}
// nadaljuje za zanko
```

pomnilnik

a	4
b	2
v	?
s	?

izpis na ekran

```
****
****
_
```

Sled programa

83

- sled programa
  - zapišemo vse vrednosti spremenljivk na vsakem koraku izvajanja programa
  - najlažje predstavimo s tabelo
- Primer: Kaj izpiše naslednji program, če preko tipkovnice vnesemo 14 in 4. Zapiši sled programa.

Sled programa

84

```
int main() {
  int a, b, c;
  double j;
  cin >>a >> b;
  c = b%a;

  for(int i=0; i<c; i++) {
    a = a + c;
    j = 1;
    while (j<i) {
      b = b + i;
      j = j + 1.8;
    }
  }
  cout << a << " " << b;
  return 0;
}
```

Postopek bomo naredili na tablo.
Končna rešitev:

a	b	c	i	j	i<c	j<i
14	4	4	?	?		
			0		true	
18				1		false
			1		true	
22				1		false
			2		true	
26				1		true
	6			2.8		false
			3		true	
30				1		true
	9			2.8		true
	12			4.6		false
			4		false	

Program izpiše: 30 12

8.8. Stavka `break` in `continue`

- stavka `break` in `continue` uporabimo, kadar želimo prekiniti oz. spremeniti običajen red izvajanja znotraj zank
- stavek break lahko uporabimo tudi v switch stavku
- stavek break prekine izvajanje zanke, v kateri se nahaja
  - v primeru gnezdenih zank to pomeni skok iz najbolj notranje zanke, v kateri je break stavek
- stavek continue prekine samo trenutno iteracijo (ponovitev) zanke in nemudoma prične z naslednjo iteracijo

Primer 14


```
#include <iostream>
using namespace std;

int main() {
    int stevec = 0;
    while (true) { // neskoncna zanka
        int stevilo;
        cout << "Vnesi celo stevilo (0 za konec): ";
        cin >> stevilo;

        if (stevilo==0)
            break;
        else if (stevilo < 0)
            continue;

        stevec++;
        if (stevilo % 2 == 0)
            cout << "Stevilo " << stevilo << " je sodo." << endl;
        else
            cout << "Stevilo " << stevilo << " je liho." << endl;
    }
    cout << "Vnesel si " << stevec << " pozitivnih stevil.";
    return 0;
}
```