Struktura petlje

Srednja škola fra Slavka Barbarića Čitluk

Izborna nastava

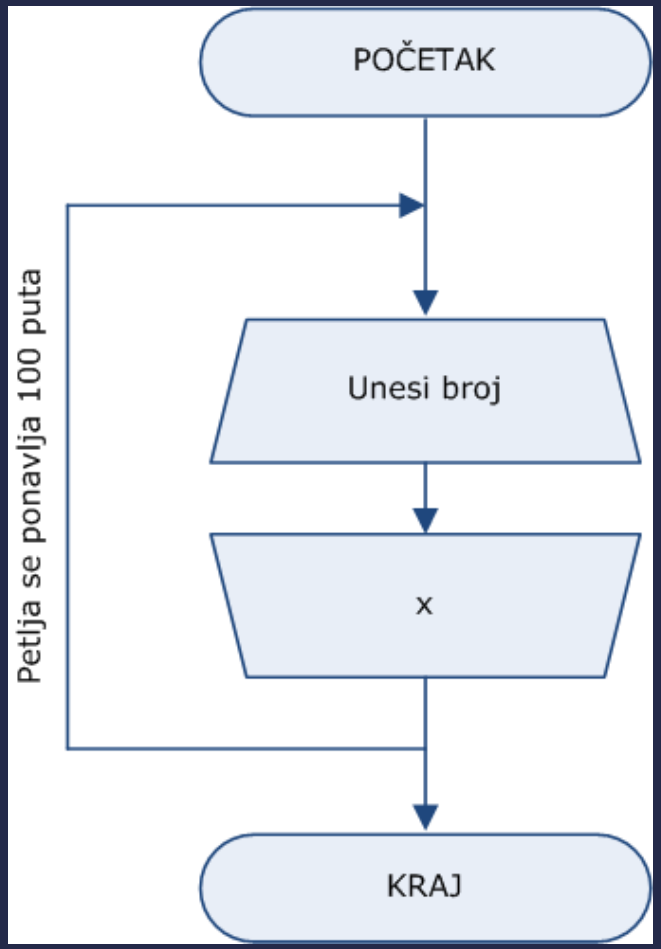
Mario Miletić

Programska struktura petlje

- Često je u programu potrebno ponoviti neku radnju više puta (iterirati). Takva se programska struktura naziva petlja.
- Ponavljati se može:
  - unaprijed zadani broj puta,
  - sve dok je ispunjen zadani uvjet.

Programska struktura petlje

- Kao primjer programske strukture petlje može poslužiti zadatak:
- Korisnik treba unijeti 100 cijelih brojeva, a prije svakog unesa broja ispisuje se tekst "Unesi broj".



Petlje

- Struktura petlje se može ostvariti naredbama:
  - *for*,
  - *while,*
  - *do – while.*

for petlja

- *for* petlja se najčešće koristi ako se dijelovi programa trebaju ponoviti unaprijed poznati broj puta.
- Osnovni oblik *for* petlje:

```
for (početno stanje; uvjet; prirast)
{
        blok naredbi
}
naredba iz bloka
```

for petlja

- Svaka *for* petlja ima svoju kontrolnu varijablu.
- Njena se vrijednost svakim prolaskom kroz petlju mijenja ovisno o vrijednosti prirasta.
- Kontrolnu varijablu petlje potrebno je prije ulaska u petlju deklarirati.
- U petlji se najprije zadaje vrijednost početnog stanja kontrolne varijable petlje (početno stanje).

for petlja

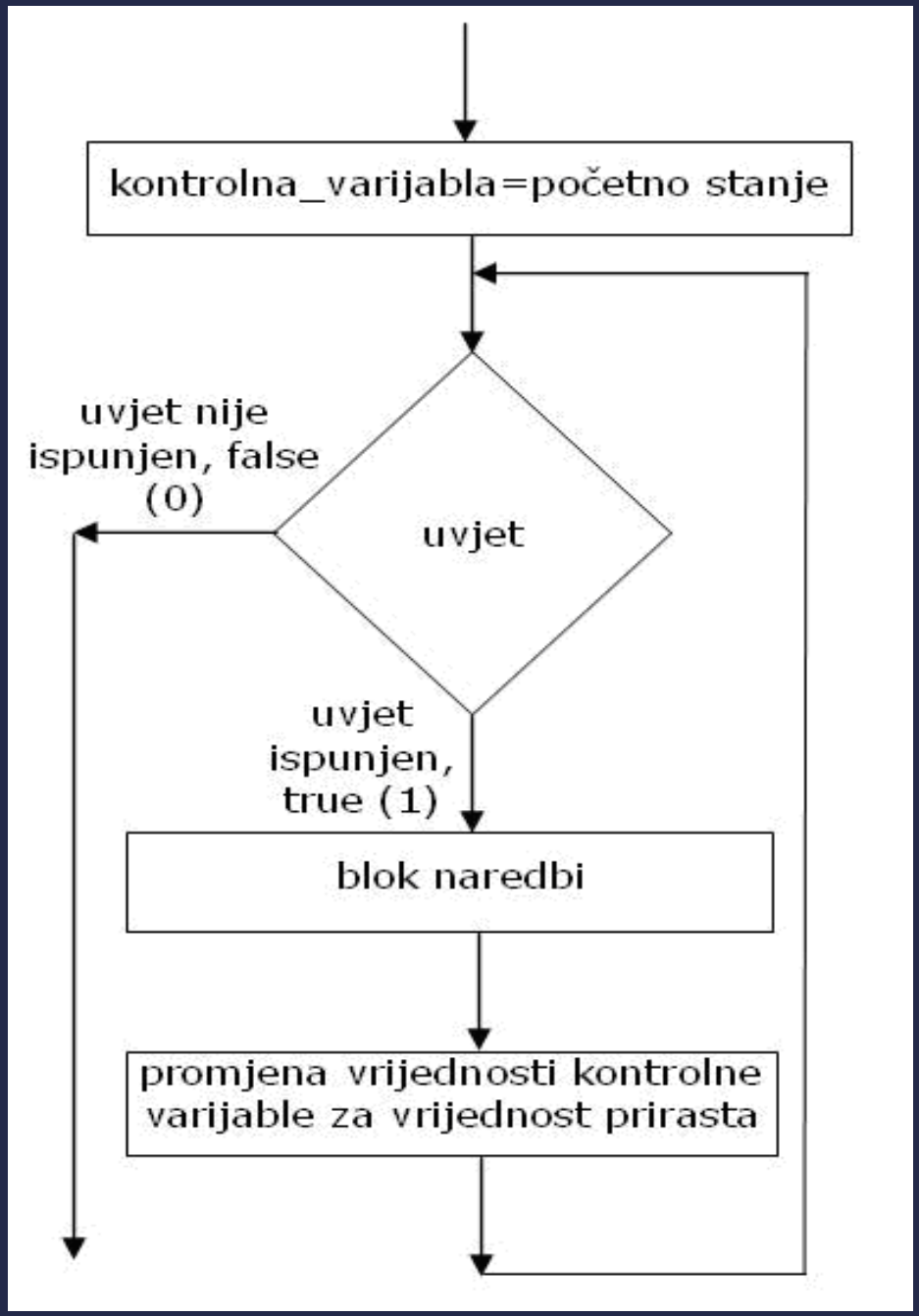
- Po zadavanju početnog stanja kontrolne varijable petlje zapisuje se uvjet.
- Rezultat uvjeta je podatak tipa bool (0 ili 1).
- Blok naredbi se izvršava sve dok je vrijednost uvjeta jednaka logičkoj istini (1).
- Kad vrijednost uvjeta postane neistina (0) petlja se prekida.

for petlja

```
for (početno stanje; uvjet; prirast)
```

- Prirast je iznos za koji se mijenja vrijednost kontrolne varijable petlje.

for petlja


kontrolna_varijabla=početno stanje
uvjet nije ispunjen, false (0)
uvjet
uvjet ispunjen, true (1)
blok naredbi
promjena vrijednosti kontrolne varijable za vrijednost prirasta

Primjer 1

- Potrebno je ispisati brojeve od 1 do 20.

```
Ispis brojeva od 1 do 20:
1 2 3 4 5 6 7 8 9 ..............20
```

Primjer 1

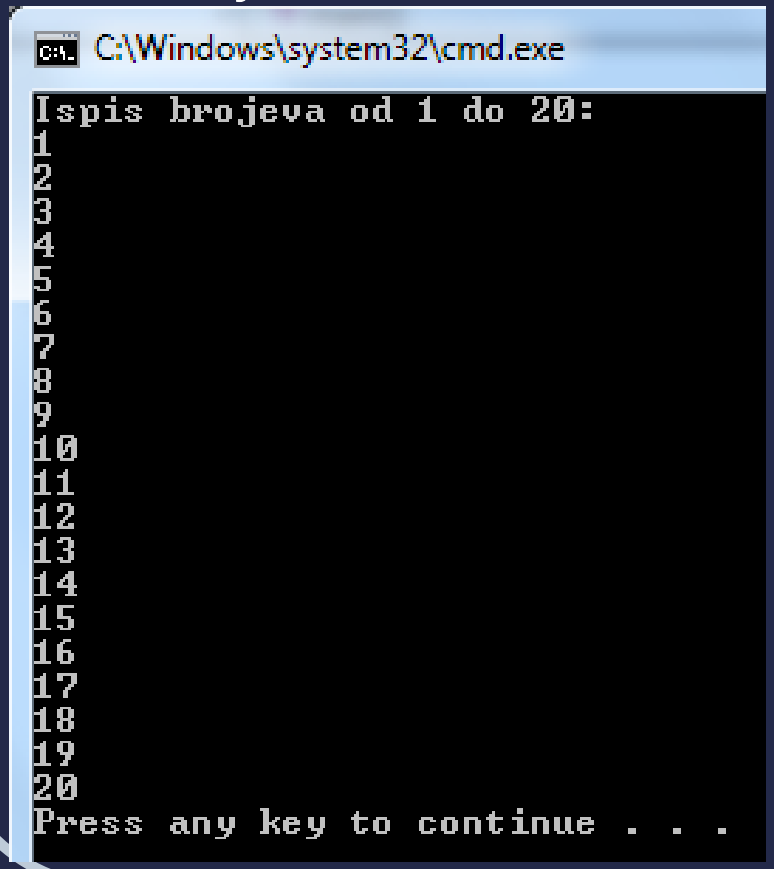
- Treba zadati početnu vrijednost kontrolne varijable petlje (brojac=1), postaviti uvjet (brojac<=20) i zadati prirast (brojac++).
- Naredba koja se izvodi konačan broj puta je ispis trenutne vrijednosti varijable brojac.

Primjer 1

```c
#include<stdio.h>

void main() {

    int brojac;
    printf("Ispis brojeva od 1 do 20:\n");

    for (brojac=1; brojac<=20; brojac++) {
        printf("%d\n", brojac);
    }
}
```

Primjer 1

- Provjera:



Primjer 2

- Potrebno je dopuniti prethodni primjer tako da se ispisuju brojevi iz raspona od M do N (raspon bira korisnik).

```
Ispis pocinje od broja:
Ispis zavrsava brojem:
Ispis brojeva od ... do ...:
... ... ...          .....          ...
```

Primjer 2

- Napomena:
  - Početna vrijednost kontrolne varijable brojac je ovisna o unesenoj početnoj vrijednosti raspona (M), a uvjet o unesenoj završnoj vrijednosti raspona (N). Izraz prirasta (brojac++) se ne mijenja.

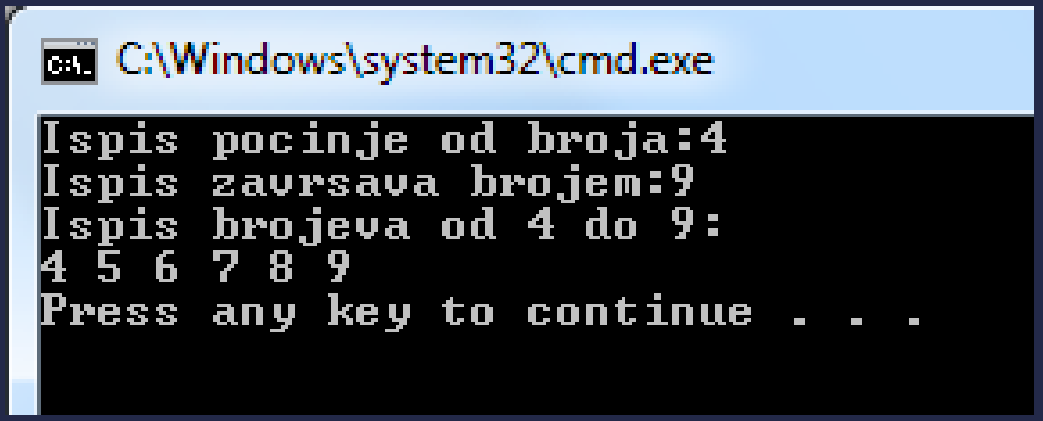
Primjer 2

unos početne i završne vrijednosti raspona

```c
#include<stdio.h>

void main() {

    int poc_broj, zav_broj, brojac;

    printf("Ispis pocinje od broja:");
    scanf("%d", &poc_broj);
    printf("Ispis zavrsava brojem:");
    scanf("%d", &zav_broj);
    printf("Ispis brojeva od %d do %d:\n", poc_broj, zav_broj);

    for (brojac=poc_broj; brojac<=zav_broj; brojac++) {
        printf("%d ", brojac);
    }
    printf("\n");
}
```

Primjer 2

- Provjera:



Oprez - beskonačna petlja

- Potrebno je ovako promijeniti prethodni program:

```
for (brojac=1;brojac<=n;)
{
  printf("%d", brojac);
}
```

Oprez - beskonačna petlja

Oprez - beskonačna petlja

- Petlja se izvodi neograničen broj puta jer je uvjet uvijek ispunjen.
- Ako se dogodi da se pokrene program u kome je beskonačna petlja, program se može prekinuti zatvaranjem prozora u kom se izvršava program.

Oprez - beskonačna petlja

- Da bi se izbjegla beskonačna petlja treba:
  - zadati uvjet koji jamči konačan broj ponavljanja petlje,
  - navesti sva tri izraza u zagradi naredbe *for*,
  - izbjegavati promjenu vrijednosti kontrolne varijable unutar bloka naredbi *for* petlje.

Primjer 3

- Treba ispisati parne brojeve iz intervala od 50 do 100.

```
Parni brojevi iz intervala od 50 do
100 su:
50 52 54 56....          98 100
```

Primjer 3

- Napomene:
  - Pošto je razlika dva susjedna parna broja 2, problem je moguće riješiti tako da se vrijednost kontrolne varijable petlje mijenja za 2 (prirast 2).

```
brojac=brojac+2 ili brojac+=2
```

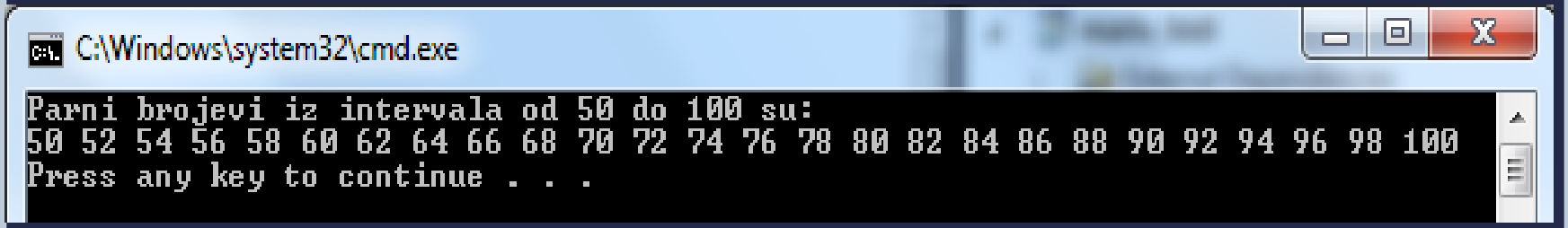
(Neće trebati provjeravati parnost!)

Primjer 3

```c
#include<stdio.h>

void main() {

    int brojac;

    printf("Parni brojevi iz intervala od 50 do 100 su:\n");

    for (brojac=50; brojac<=100; brojac+=2) {
        printf("%d ", brojac);
    }
    printf("\n");
}
```

Primjer 3

- Provjera.



Primjer 4

- Treba ispisati parne brojeve u rasponu od 100 do 50.

```
Parni brojevi iz intervala od 100 do
50 su:
100 98 96 94 .......      52 50
```

Primjer 4

- Napomene:
  - Prirast može biti i negativan pa se tada vrijednost kontrolne varijable petlje smanjuje.
  - U ovom primjeru početna vrijednost kontrolne varijable petlje mora biti veća od završne.
  - Npr. prirast može biti:

```
brojac=brojac-2 (brojac-=2)
```

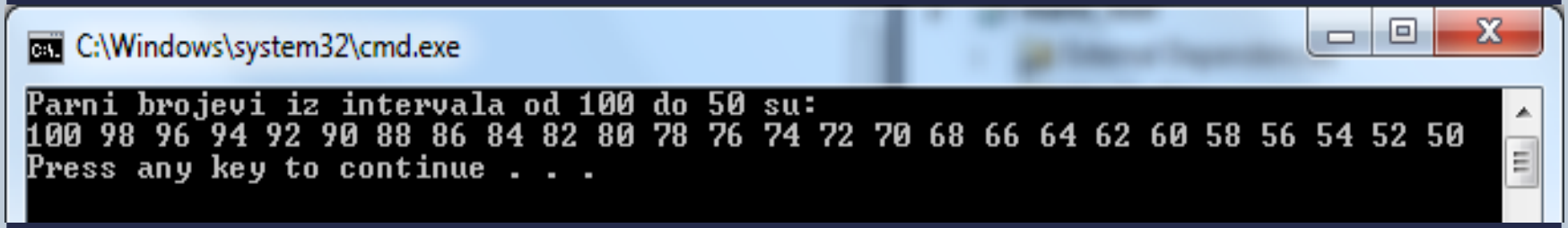
Primjer 4

```c
#include<stdio.h>

void main() {

    int brojac;

    printf("Parni brojevi iz intervala od 100 do 50 su:\n");

    for (brojac=100; brojac>=50; brojac-=2) {
        printf("%d ", brojac);
    }
    printf("\n");
}
```

prirast

Primjer 4

- Provjera:



Primjer 5

- Treba potražiti pa ispisati brojeve djeljive sa 7 unutar raspona od M do N.

```
Raspon pocinje od broja:
Raspon zavrsava brojem:

Brojevi djeljivi sa 7 iz raspona od … do … su:
... ... ... ... ...
```

Primjer 5

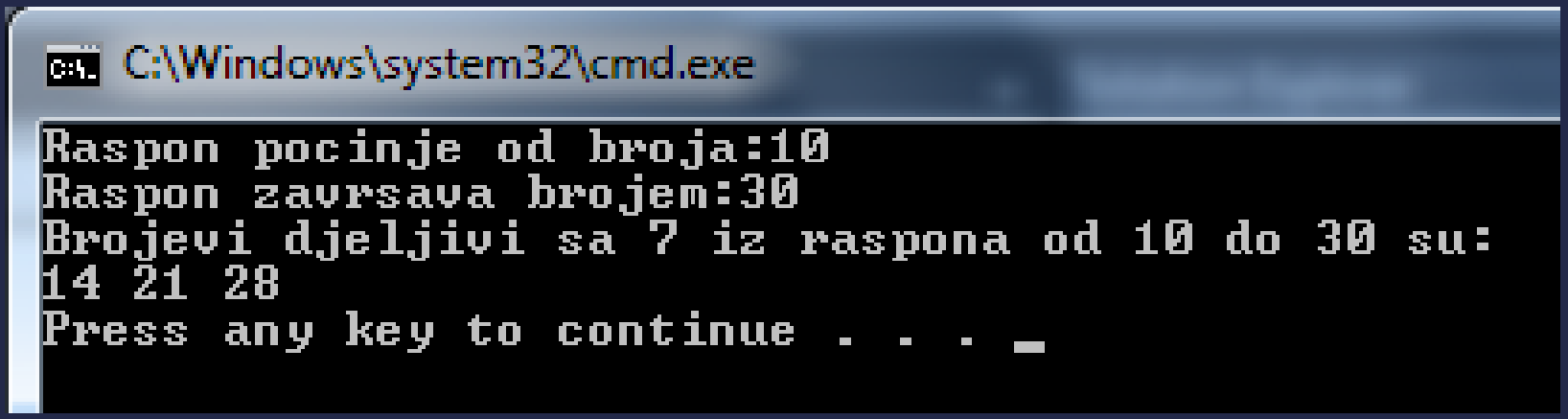
- Napomene:
  - Blok naredbi u petlji se izvršava za svaki broj iz zadanog raspona.
  - Ispisuju se samo brojevi djeljivi s brojem 7.
  - Djeljivost brojeva se provjerava operatorom modulo (%).

Primjer 5

```c
#include<stdio.h>

void main() {

    int poc_broj, zav_broj, brojac;

    printf("Raspon pocinje od broja:");
    scanf("%d", &poc_broj);

    printf("Raspon zavrsava brojem:");
    scanf("%d", &zav_broj);

    printf("Brojevi djeljivi sa 7 iz raspona od %d do %d su:\n", poc_broj, zav_broj);

    for (brojac=poc_broj; brojac<=zav_broj; brojac++) {
        if(brojac%7==0) {
            printf("%d ", brojac);
        }
    }
    printf("\n");
}
```

Primjer 5

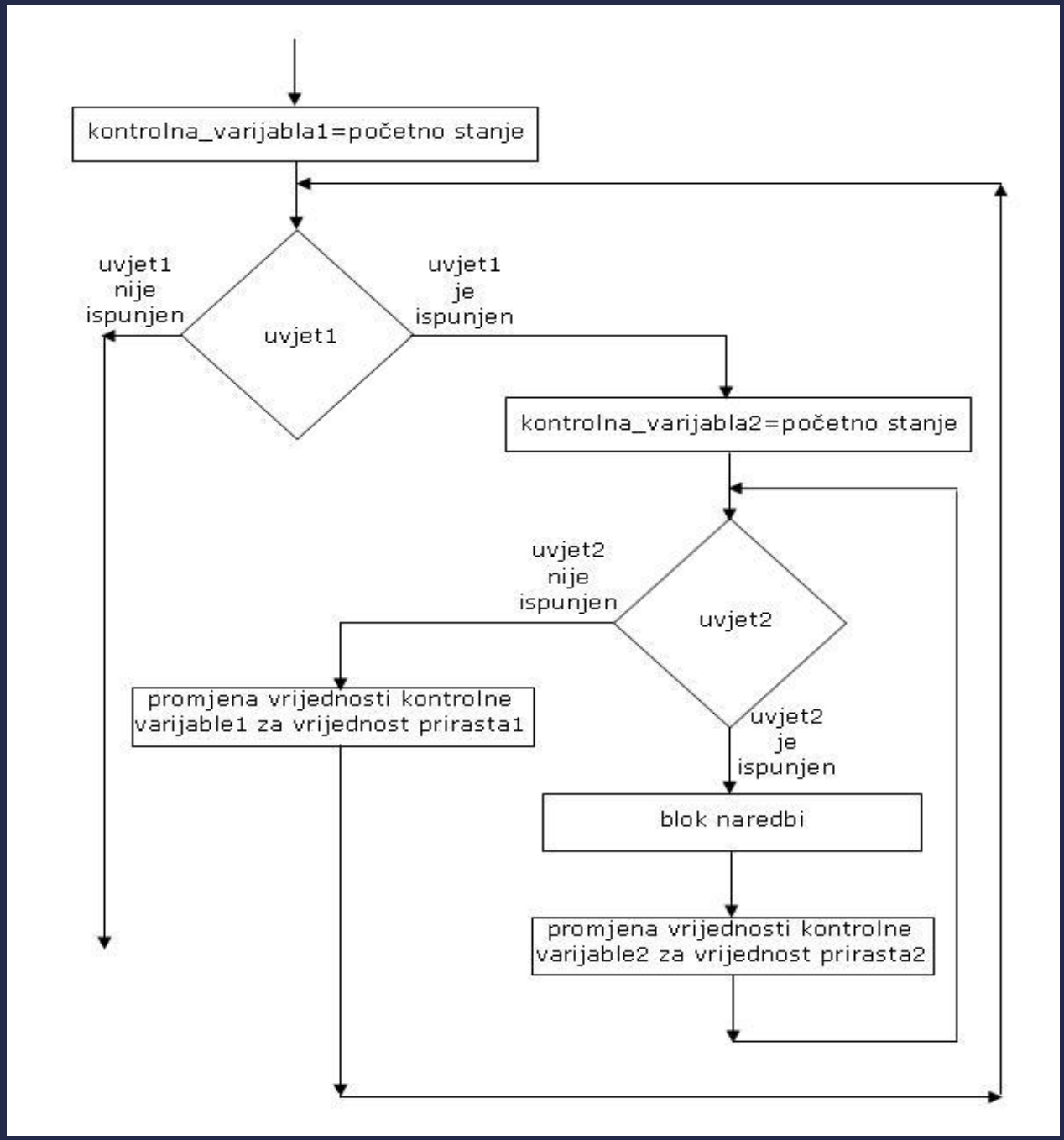
- Provjera:



Ugniježđene *for* petlje

- *for* petlje mogu biti ugniježđene jedna unutar druge.
- Pri ulazu u vanjsku petlju, njena se kontrolna varijabla postavi na početnu vrijednost.
- S tom se vrijednošću ulazi u unutarnju petlju.

Ugniježđene *for* petlje



Ugniježđene *for* petlje

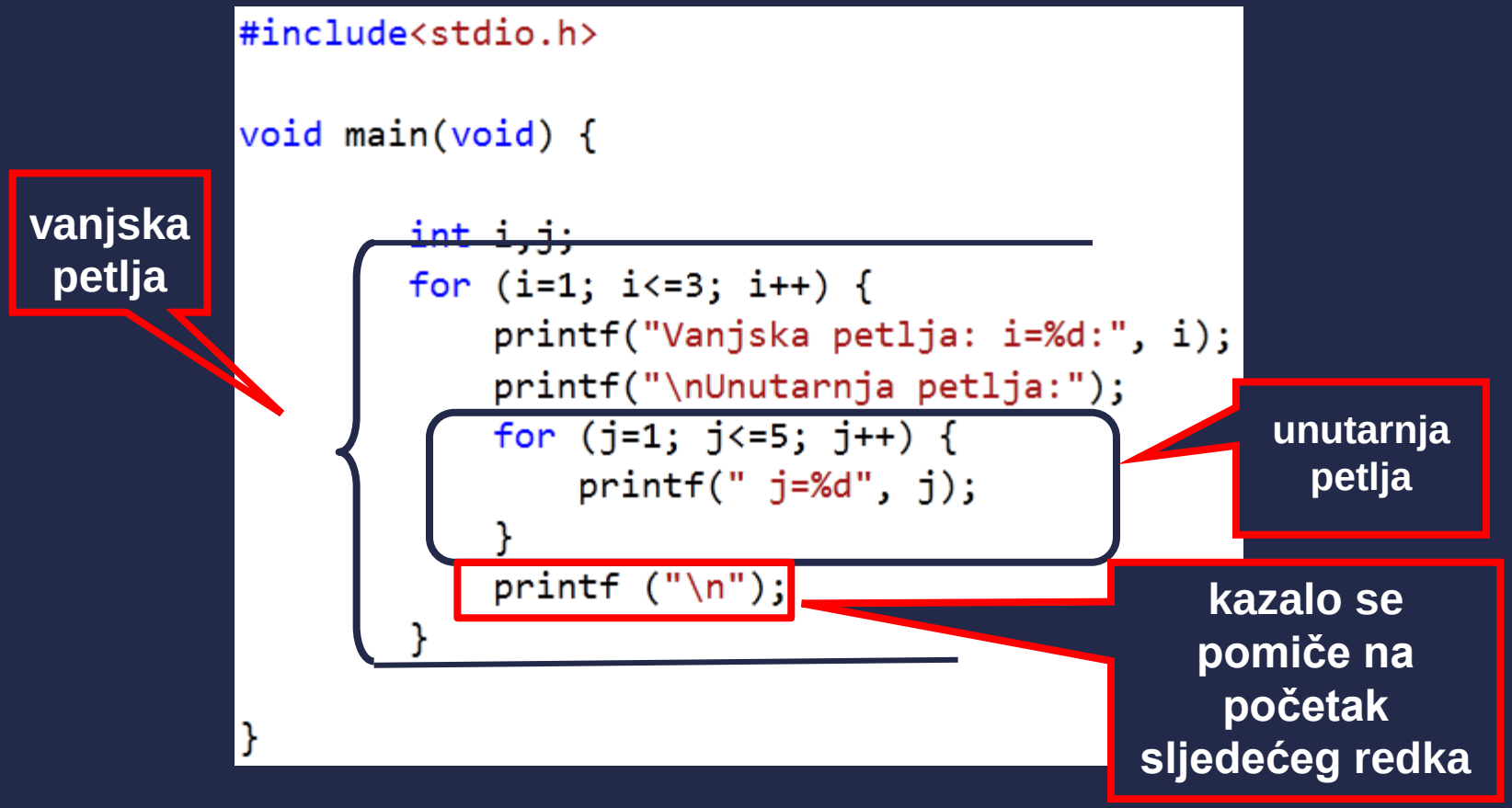
- Pri ulazu u unutarnju petlju, kontrolna varijabla unutarnje petlje se postavi na početnu vrijednost
  i sve dok je uvjet zadovoljen, izvršava se blok naredbi unutarnje petlje.
- Po završenoj unutarnjoj petlji, kontrolna varijabla vanjske petlje se mijenja za vrijednost prirasta i provjerava se uvjet vanjske petlje.
- Za svaku vrijednost kontrolne varijable vanjske petlje izvodi se cjelokupna unutarnja petlja.

Primjer 6

- Vrijednost kontrolne varijable vanjske petlje mijenja se od 1 do 3, a unutarnje od 1 do 5.
- U programu se ispisuju trenutne vrijednosti kontrolnih varijabli vanjske i unutarnje petlje.

```
Vanjska petlja: i=1
Unutarnja petlja:   j=1   j=2   j=3   j=4   j=5

Vanjska petlja: i=2
Unutarnja petlja:   j=1   j=2   j=3   j=4   j=5

Vanjska petlja: i=3
Unutarnja petlja:   j=1   j=2   j=3   j=4   j=5
```

Primjer 6

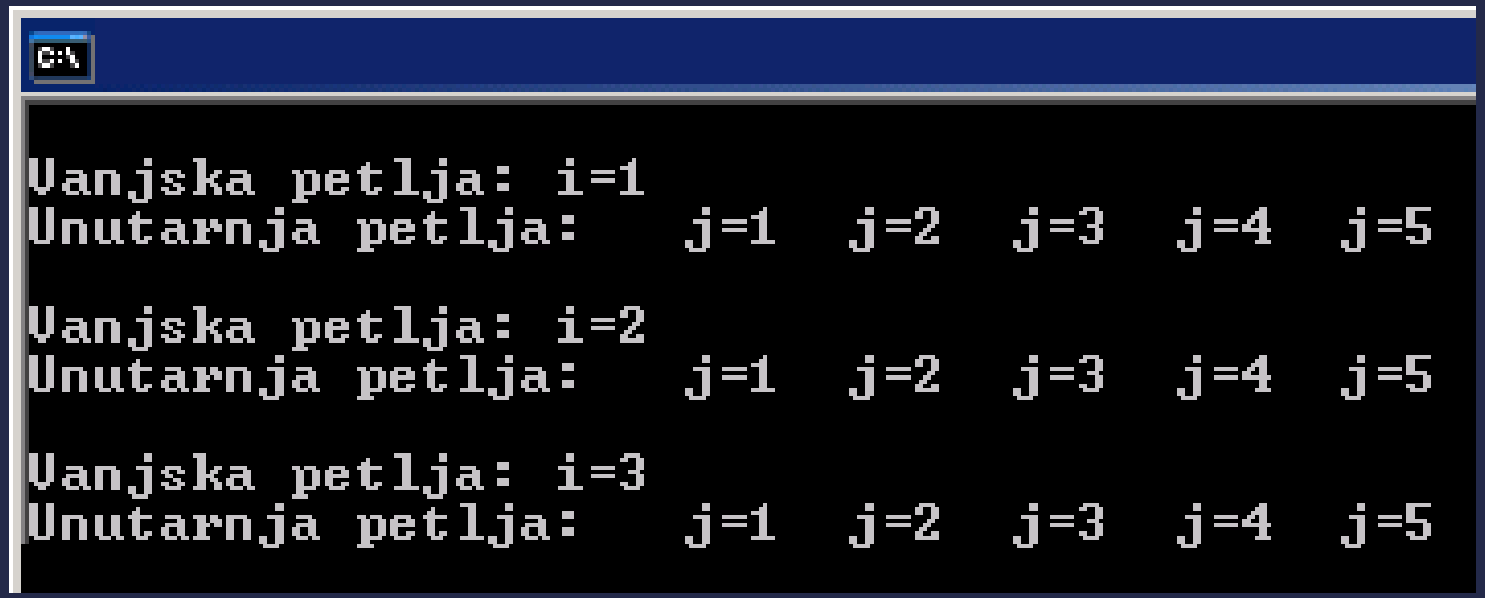

#include<stdio.h>
void main(void) {
int i,j;
for (i=1; i<=3; i++) {
printf("Vanjska petlja: i=%d:", i);
printf("\nUnutarnja petlja:");
for (j=1; j<=5; j++) {
printf(" j=%d", j);
}
printf ("\n");
}
}
vanjska petlja
unutarnja petlja
kazalo se pomiče na početak sljedećeg redka

Primjer 6

- Vanjska petlja omogućit će :
  - ispis teksta: `Vanjska petlja:`
  - ispis trenutne vrijednosti kontrolne varijable vanjske petlje (i)
  - ispis teksta: `Unutarnja petlja:`
- Unutarnja petlja :
  - ispisuje trenutne vrijednosti kontrolne varijable unutarnje petlje (j, od 1 do 5).
- Po ispisu svakoga od redaka, kazalo se prebacuje na početak novog redka i postupak se ponavlja.

Primjer 6

- Provjera:



Primjer 7

- Treba ispisati tablicu množenja za brojeve od 1 do 10.

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

Primjer 7

- Napomene:
  - Vanjska petlja omogućit će stvaranje 10 redaka.
  - Unutarnja petlja će u svakome redku stvoriti 10 stupaca.
  - Naredba koja se izvršava u unutarnjoj petlji je ispis umnoška trenutnih vrijednosti kontrolnih varijabli vanjske i unutarnje petlje.
  - Po ispisu svakoga od redaka, kazalo se prebacuje na početak novog redka.

Primjer 7

```c
#include<stdio.h>

void main(void) {

    int red,stup;
    for (red=1; red<=10; red++) {
        for (stup=1; stup<=10; stup++) {
            printf("%d ", red*stup);
        }
        printf ("\n");
    }

}
```

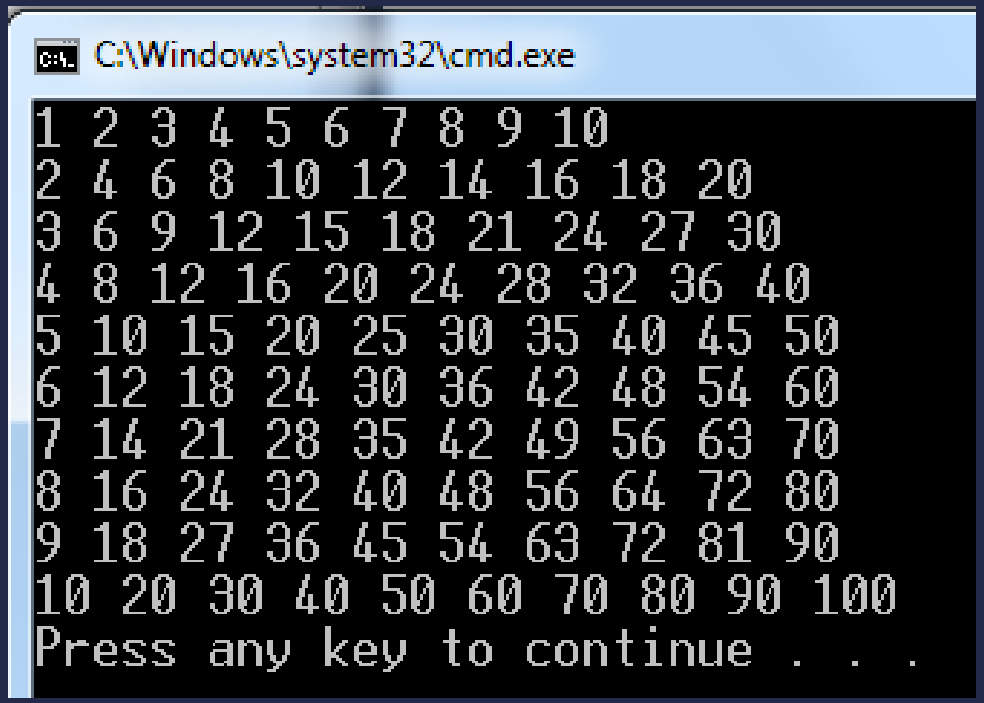
vanjska petlja

unutarnja petlja

kazalo se pomiče na početak sljedećeg redka

Primjer 7

- Ispis je dosta nepregledan, trebalo bi ga oblikovati tako da bude u pravilnim stupcima.



Manipulatori

- Ispis se može oblikovati manipulatorima (operatorima za rukovanje ispisom).
- Da bi ispis bio u pravilnim stupcima, potrebno umnožak u unutarnjoj petlji ispisati u sljedećem obliku:

  %5d

- *Broj 5* određuje koliki će se prostor predvidjeti za ispis podatka koji slijedi u izlaznom toku.

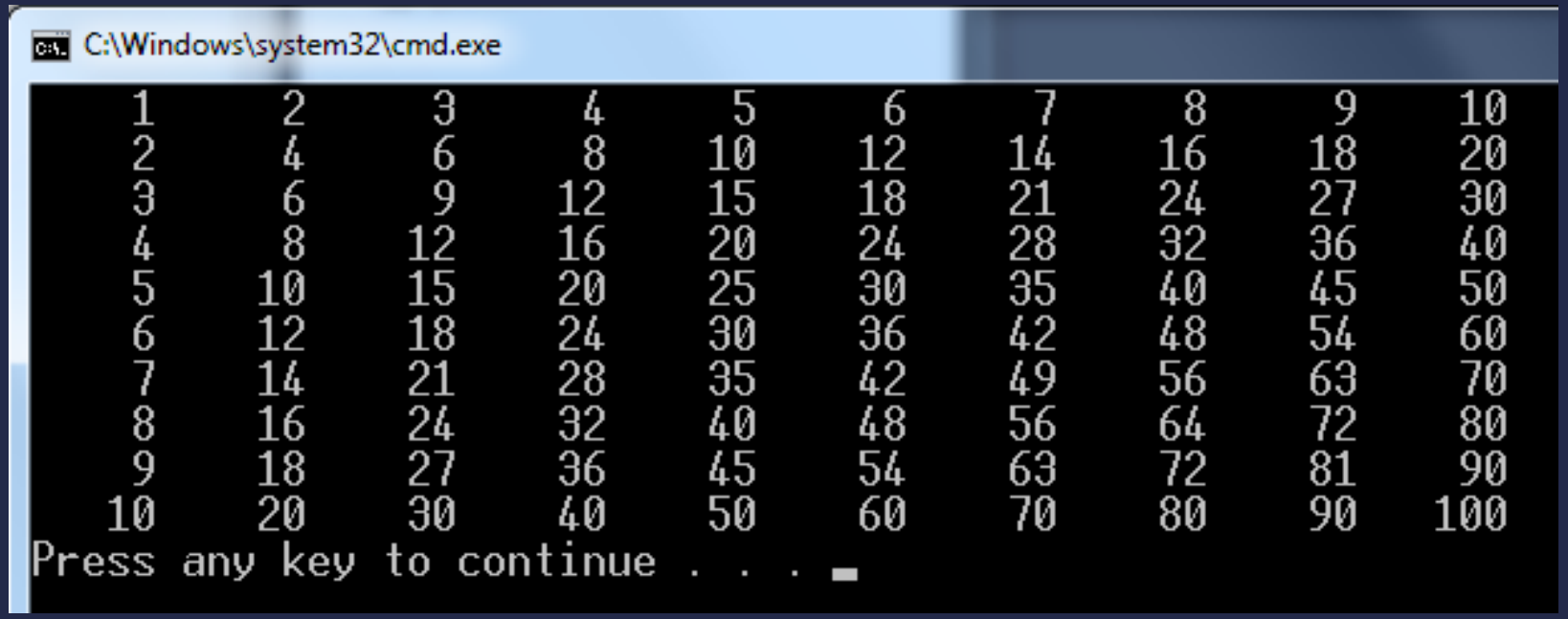
Primjer 7

```
#include<stdio.h>

void main(void) {

        int red,stup;
        for (red=1; red<=10; red++) {
            for (stup=1; stup<=10; stup++) {
                printf("%5d ", red*stup);
            }
            printf ("\n");
        }

}
```

za ispis svakog podatka predviđa se 5 mjesta

Primjer 7

- Provjera:



Primjer 8

- Potrebno je zbrojiti prvih 100 prirodnih brojeva.

```
Zbroj prvih 100 prirodnih brojeva je
....
```

Primjer 8

- Prije ulaska u petlju treba:
  - Deklarirati varijablu koja će sadržavati trenutne vrijednosti kontrolne varijable petlje (npr. *zbroj*) pri svakom prolasku kroz petlju.
  - Varijabli *zbroj* pridružiti vrijednost 0.
- Naredba u bloku omogućava uvećavanje vrijednosti varijable *zbroj* za tekuću vrijednost kontrolne varijable petlje (brojac).

Primjer 8

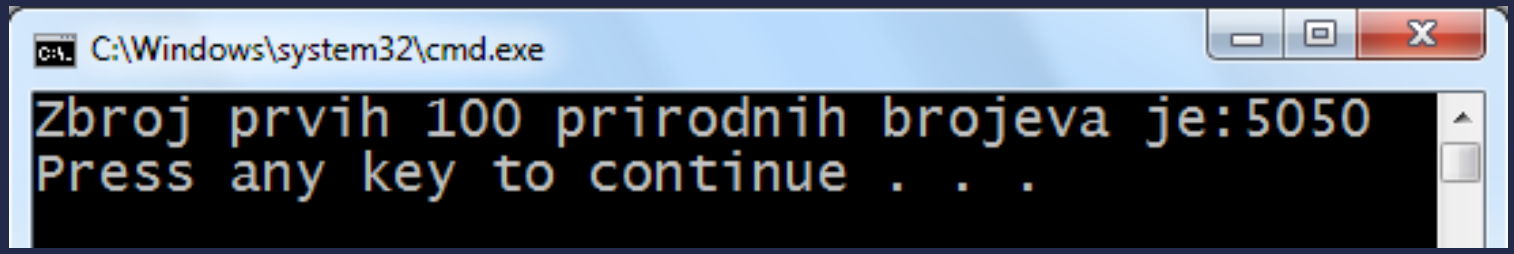
```c
#include<stdio.h>

void main() {
    int brojac, zbroj=0;

    for (brojac=1; brojac<=100; brojac++) {
        zbroj += brojac;
    }

    printf("Zbroj prvih 100 prirodnih brojeva je:%d\n", zbroj);
}
```

početna vrijednost

blok

Primjer 8

- Provjera:



Primjer 9

- Treba prebrojiti koliko brojeva unutar raspona od M do N ima znamenku jedinice vrijednosti 9.

```
Raspon pocinje od broja:
Raspon zavrsava brojem:

U rasponu od … do … ima … brojeva sa
znamenkom jedinice vrijednosti 9.
```

Primjer 9

- Prije ulaska u petlju treba:
  - Deklarirati varijablu (npr. *N*) u kojoj će se prebrojavati pronađeni brojevi koji zadovoljavaju uvjet.
  - Varijabli *N* pridružiti vrijednost 0.
- Naredba u bloku omogućava uvećavanje vrijednosti varijable *N* za 1 svaki put kada se pronađe broj koji zadovoljava uvjet.

Primjer 9

```c
#include<stdio.h>

void main() {
    int brojac, zbroj=0, m, n;

    printf("Raspon pocinje od broja:");
    scanf("%d", &m);
    printf("Raspon zavrsava brojem:");
    scanf("%d", &n);

    for (brojac=m; brojac<=n; brojac++) {
        if (brojac%10==9){
            zbroj++;
        }
    }

    printf("U rasponu od %d do %d ima %d brojeva sa znamenkom jedinice vrijednosti 9.\n", m, n, zbroj);
}
```

Primjer 9

- Provjera: