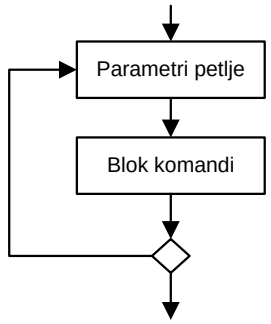
Naredbe ponavljanja

U većini programa se javljaju situacije kada je potrebno neku naredbu ili grupu naredbi izvršiti više puta. Ukoliko je naredbu potrebno izvršiti konačan i mali broj puta, problem je moguće razrešiti i korišćenjem linijskih struktura, tako što bi se naredba jednostavno ponovila određeni broj puta uzastopno. Međutim, može se desiti da je naredbu potrebno ponoviti veliki broj puta, a veoma često je taj broj promenljiv u zavisnosti od izvršenja ostatka programa. U takvim slučajevima nije moguće iskoristiti linijsku strukturu, već je neophodno uvesti takozvane **ciklične strukture**. Ciklične strukure omogućavaju izvršavanje jedne ili više naredbi određeni broj puta, pri čemu broj ponavljanja može biti definisan prirodnim brojem ili uslovom koji određuje kada se ponavljanje prekida.

Napomenimo i to da se ciklične strukture vrlo često nazivaju i **ciklusima** ili **petljama**.

FOR – ciklus

Naredba **for** umogućava bezuslovno ponavljanje nekog dela programa određeni broj puta. Na taj način moguće je ostvariti cikličnu strukturu prikazanu na slici:



Slika ### Algoritamska šema **for** ciklusa

FOR petlja se sastoji iz tri dela. U okviru prvog dela može se definisati brojačka promenljiva (brojač) koja se u svakom prolasku kroz petlju (iteraciji) može menjati. U okviru drugog dela može se definisati uslov izlaska iz petlje. To je logički izraz koji u slučaju da je **netačan** prekida izvršavanje petlje. U okviru trećeg dela se zadaje promena brojača petlje nakon svake iteracije.

Sintaksa

```
for ([opcioni_izraz_1] ; [opcioni_logicki_izraz] ; [opcioni_izraz_2]) <naredba>;
```

Nije obavezno definisati sve delove **for** petlje, ali iako neki deo ne postoji mora biti odvojen rezervisanim znakom **;**.

Primer 1

```
for (i = 1; i <= 5; i++) printf("A");
```

Rezultat ovog primera je 5 puta ispisano slovo "A" na ekranu na sledeći način:

```
AAAAA
```

Primer 2

```
for (i = 1; i <= 10; i++) printf("%5d", i);
```

Nakon izvršenja ove petlje na ekranu će biti ispisani brojevi od 1 do 10:

```
    1    2    3    4    5    6    7    8    9   10
```

Ukoliko bismo želeli da brojevi budu ispisani obrnutim redom, onda bi petlja trebala da izgleda ovako:

```
for (i = 10; i > 0; i--) printf("%5d", i);
```

a rezultat bi bio niz brojeva ispisan na sledeći način:

```
   10    9    8    7    6    5    4    3    2    1
```

Primer 3

Sledeći primer pokazuje da početna i krajnja vrednost brojačke promenljive mogu biti i izrazi.

```
for (i = -4+6; i <= 4*3-5; i++) printf("%5d", i);
```

Rezultat prethodnog primera je sledeći niz brojeva ispisan na ekranu:

```
    2    3    4    5    6    7
```

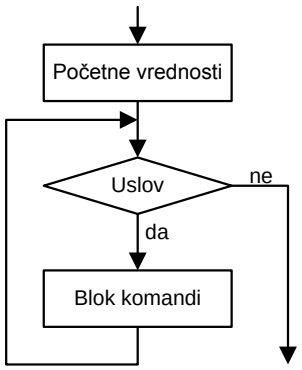
Primer 4

Napisati program koji za *n* zadatih vrednosti poluprečnika *r* izračunava obim i površinu kruga i prikazuje ih na ekranu.

```
#include <stdio.h>
#include <math.h>

main()
{
   const float PI = 3.14;

   int i,n;
   float r,O,P;

   printf("Unesite koliko izracunavanja zelite:\n");
   scanf("%d", &n);

   for(i = 0; i < n; i++)
   {
      printf("Unesite poluprecnik:\n");
      scanf("%f", &r);

      O=2.0*r*PI;
      P=pow(r, 2)*PI;

      printf("Obim      Povrsina\n");
      printf("%10.2f%10.2f\n", O, P);
   }
}
```


WHILE – ciklus

Osnovna karakteristika **while** petlje je da je to **petlja sa preduslovom**, što znači da se ispunjenost uslova proverava pre izvršavanja svake iteracije. Posledica toga je da je moguće da se naredbe unutar petlje ne izvrše ni jednom, ukoliko uslov u startu nije zadovoljen. Šematski prikaz ciklusa sa preduslovom prikazan je na slici:



Slika ### Šematski prikaz ciklusa sa preduslovom

Naredba **while** omogućava ponavljanje određenog dela programa sve **dok je navedeni uslov ispunjen**. Na početku svake iteracije proverava se da li je uslov ispunjen, tj. da li je vrednost logičkog iskaza **tačno** (**true**), a zatim, ukoliko je jeste, izvršava se jedna naredba ili blok naredbi navedenih unutar komande **while**. Nakon izvršenja naredbi unutar **while** petlje otpočinje nova iteracija i nova provera uslova. U trenutku kada na početku neke iteracije vrednost logičkog izraza postane **netačno** (**false**), prekida se izvršavanje petlje i nastavlja se izvršavanje ostatka programa.

Sintaksa

```
while (<logicki_izraz>) <naredba>;
```

NAPOMENA: **While** petlja mora biti tako napisana da garantuje da će u konačnom broju iteracija navedeni logički izraz postati netačan (**false**). Na taj način obezbeđuje se mehanizam izlaska iz petlje nakon konačnog broja iteracija. Ukoliko logički izraz nikada ne bi dobio vrednost **false** došlo bi do beskonačnog broja ponavljanja (tzv. **mrtva petlja**), odnosno do blokade izvršenja ostatka programa.

Primer 1

```
int i = 2;
while(i < 10)
{
   printf("%5d", i);
   i = i + 2;
}
```

Prethodni primer štampa na ekranu sve jednocifrene parne brojeve na sledeći način:

```
    2    4    6    8
```

Primer 2

Ukoliko bismo prethodni primer promenili tako da izgleda ovako:

```
int i = 20;
while(i < 10)
{
   printf("%5d", i);
   i = i + 2;
}
```

u tom slučaju se naredbe unutar **while** petlje ne bi izvršile ni jednom, iz razloga što uslov *i<10* nije ispunjen već na početku prve iteracije.

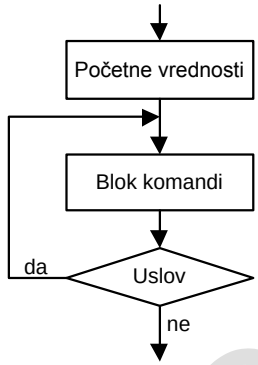
Primer 3

Napisati program koji zadatom celom broju *n* uklanja sve nule sa desne strane. Na primer, ukoliko unesemo broj 21000, rezultat treba da bude 21.

```
#include <stdio.h>

main()
{
   int n;

   printf("Unesite broj n:\n");
   scanf("%d", &n);

   while(n % 10 == 0)
      n = n / 10;

   printf("Dobijeni broj je %d\n", n);
}
```

DO-WHILE – ciklus

Za razliku od **while** petlje, naredba **do-while** omogućava realizaciju ciklusa sa **postuslovom**, kao što je prikazano na slici:



Slika ### Šematski prikaz postuslovnog ciklusa

Ciklus sa postuslovom podrazumeva to da se uslov za dalje izvršavanje petlje proverava na kraju svake iteracije. Na ovaj način naredbe unutar petlje će biti izvršene bar jednom, bez obzira na to da li je uslov bio ispunjen pre ulaska u petlju.

U slučaju **do-while** petlje, izvršava se jedna naredba ili blok naredbi navedenih unutar komande **do**. Nakon izvršenja ovih naredbi vrši se provera uslova navedenog iza rezervisane reči **while**. Ukoliko je vrednost logičkog izraza **tačno** (**true**), kreće se u izvršavanje sledeće iteracije. U suprotnom prekida se izvršavanje petlje i nastavlja se izvršavanje ostatka programa.

Sintaksa

```
do
{
   <naredba_1>;
   [naredba_2];
      ...
```

```
   [naredba_n];
} while (<logicki_izraz>);
```

NAPOMENA: **do-while** petlja mora biti tako napisana da garantuje da će u konačnom broju iteracija navedeni logički izraz postati netačan (**false**). Na taj način obezbeđuje se mehanizam izlaska iz petlje nakon konačnog broja iteracija. Ukoliko logički izraz nikada ne bi dobio vrednost **false** došlo bi do beskonačnog broja ponavljanja (tzv. **mrtva petlja**), odnosno do blokade izvršenja ostatka programa.

Primer 1

```
int i = 2;

do
{
   printf("%5d", i);
   i = i + 2;
} while(i < 10);
```

Prethodni primer štampa na ekranu sve jednocifrene parne brojeve, na sledeći način:

```
    2    4    6    8
```

Primer 2

Napisati program koji izračunava broj π sa zadatom tačnošću ε, korišćenjem formule:

$$\pi = 4 \cdot \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \ldots\right)$$

```
#include <stdio.h>

main()
{
   float pi,epsilon,delilac,znak,sabirak;
   printf("Unesite zeljenu tacnost:\n");
   scanf("%f", &epsilon);

   pi = 4.0;
   delilac = 1.0;
   znak = 1.0;

   do
   {
      delilac=delilac+2.0;
      znak=-znak;
      sabirak=4.0/delilac;
      pi=pi+znak*sabirak;
   } while(sabirak >= epsilon);

   printf("PI=%f\n", pi);
}
```

Naredbe BREAK i CONTINUE

C kompajler podržava korišćenje komandi za prevremeni prekid čitave petlje ili samo iteracije koja se trenutno izvršava. Obe ove komande su primenjive na **for**, **while** i **do-while** petlje.

Naredba BREAK

Naredba **break** se koristi za bezuslovno prekidanje izvršavanja čitave petlje. Nakon izvršenja ove komande prekida se izvršavanje naredbi unutar petlje i nastavlja se sa izvršavanjem ostatka programa.

Primer

Napisati program koji za *n* realnih brojeva sa ulaza izračunava funkciju $f(x)=\frac{1}{x}$. U slučaju da je na ulazu uneta nula, obavestiti korisnika da deljenje nulom nije moguće i prekinuti izvršavanje programa.

```
#include <stdio.h>

main()
{
   int i,n;
   float x,f;

   printf("Unesite koliko brojeva zelite:\n");
   scanf("%d", &n);

   for(i = 0; i < n; i++)
   {
      printf("Unesite x:\n");
      scanf("%f", &x);

      if(x != 0.0)
      {
         f=1.0/x;
         printf("f(x)=%10.4f\n", f);
      }
      else
      {
         printf("Deljenje nulom nije moguce.\n");
         break;
      }
   }
}
```

Naredba CONTINUE

Naredba **continue** se koristi za bezuslovno prekidanje izvršavanje trenutne i otpočinjanje sledeće iteracije. Time se praktično postiže preskakanje naredbi koje slede u nastavku iza komande **continue**, a naredna iteracija se normalno izvršava ukoliko su zadovoljeni uslovi definisani petljom.