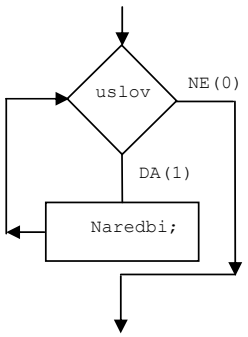
4 Контрола на текот на програмата
While, do… while, for

1. Структури за повторување, циклуси (while и do. . . while)

While структурата за повторување е една од трите структури за повторување во C и воедно наједноставна. Овозможува повторување на една или повеќе наредби.

Слично како и кај if стуктурата, така и во while структурата се задава услов (**задолжително во мали загради ( )**) што се евалуира, односно проверува. Наредбата (наредбите) од циклусот се извршуваат се додека условот од while структурата е точен (враќа ненулта вредност). Повторувањето на наредбите од циклусот завршува кога условот ќе врати неточна вредност (нула 0) и програмата продолжува со извршување на следната наредба од телото на програмата (што не е дел од while структурата). Доколку треба да се изврши само една наредба, таа едноставно се пишува по while делот, доколку, пак, треба да се извршат повеќе наредби (блок наредби) по while делот, тие се стават во блок ограничен со големи загради **{ }**.

while (**услов**)
тело (наредба/наредби за повторување)**;**




```
int j = 5;

while (j > 0)
      printf("j = %d\n",j--);
```

```
while (j > 0)                    j = 5
{                                j = 4
   printf("j = %d\n",j);         j = 3
   j--;                          j = 2
}                                j = 1
```

ВАЖНО!!!
Доколку по грешка се напише знакот **;** по while делот со условот, наредбата/наредбите од while структурата стануваат дел “**неправи ништо**”. Бидејќи условот останува непроменет програмата влегува во бесконечна јамка (**loop**). Во ваков случај програмата нема да врати резултат, бидејќи е зафатена со “неправењето ништо”, поради што следните наредби по while структурата никогаш не се извршуваат.

Пример:

```
int j = 5;
while (j>0);
      printf("j = %d\n",j--);
```

ЕКРАН:

ВАЖНО!!!

Многу важно е да се запомни дека сите C услови се while услови, односно "**се додека е исполнет условот се извршуваат наредбите**". Наредбите од циклусот while се извршуваат се додека е точен условот (вредност различна од 0).

Пример:

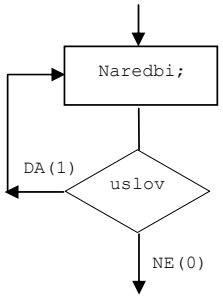
Во следниот пример, програмерот најверојатно имал намера наредбата од while структурата да се извршува се додека j не стане еднакво на 0. Сепак ова не е начинот да се постави условот, точен услов би бил се додека j е различно од 0 (j != 0).

```
int j = 5;
printf("start\n");
while (j == 0)
      printf("j = %d\n",j--);
printf("end\n");
```

ЕКРАН:
start
end

Do. . . while структурата во C е "превртена" верзија на структурата за повторување while. Ако во while циклусот по условот следува телото со наредби што се повторуваат, кај do. . . while циклусот прво се пишува телото со наредби за повторување, па следува условот. Од ова произлегува дека наредбите од циклусот ќе бидат извршени сигурно еднаш, пред да се провери условот. Ако условот е неточен, наредбите од циклусот do. . . while нема да бидат повторно извршени.

do
тело (наредба/наредби за повторување)**;**
while (услов);




```
int j = 5;
printf("start\n");
do
      printf("j = %d\n",
j--);
while (j > 0);
printf("stop\n");
```

ЕКРАН:

```
start
j = 5
j = 4
j = 3
j = 2
j = 1
stop
```

```
int j = -10;
printf("start\n");
do {
      printf("j = %d\n",);
      j--;
} while (j > 0);
printf("stop\n");
```

ЕКРАН:

```
start
j = -10
stop
```

ВАЖНО!!!!!

Да не се погреши и наместо = = да се напише =.

Задачи:

1. Да се напише програма за пресметување на $y = x^n$ за даден природен број n и релаен број x.

```
#include <stdio.h>

int main ()
{
      int  brojac, n;
      float x, y;
      printf("vnesi ja osnovata: ");scanf("%f", &x);
      printf("vnesi go eksponentot: ");scanf("%d", &n);
      y = x;
      brojac = 1;
      while (brojac < n)
      {
            y *= x;
            brojac++;
      }
      printf("%f^%d = %f\n", x, n, y);

return 0;
}
```

2. Да се напише програма која од n броеви (внесени од тастатура) ќе го определи бројот на броеви што се деливи со 3, при делењето со 3 имаат остаток 1, односно 2.

```
#include <stdio.h>

int main ()
{
      int  n, i, broj, del, os1, os2;
      del = os1 = os2 = 0;
      printf("Kolku broevi treba da se proveruvaat za delivost so 3?\n");
      scanf("%d", &n);
      i = 1;
      do {
            printf("Vnesete broj za proverka: ");
            scanf("%d", &broj);
            if ( broj % 3 == 0) del++;
            else if ( broj % 3 == 1) os1++;
                 else os2++;
            i++;
      } while (i <= n);
      printf("%d broj(a) se delivi so 3.\n", del);
      printf("%d broj(a) imaat ostatok 1, pri delenje so 3.\n", os1);
      printf("%d broj(a) imaat ostatok 2, pri delenje so 3.\n", os2);

return 0;
}
```

3. Да се напише програма која за даден природен број m ќе ги испише сите броеви помеѓу 1 и 1000 за кои сумата цифрите изнесува m.

Comment [Д.Ѓ.1]: на

```
#include <stdio.h>

int main ()
```

```
{
      int  m, i, j, n, suma, cifra;
      printf("Vnesete go brojot m za sporedba: ");
      scanf("%d", &m);
      if (m > 27) printf("Ne postoi broj od 1 do 1000 so suma %d.\n", m);
      else {
            printf("Broevi cij zbir na cifri e %d se: \n", m);
            i = 1;
            while (i<=1000) {
                  n = i; suma = 0;
                  while (n > 0) {
                        cifra = n % 10;
                        suma += cifra;
                        n /= 10;
                  }
                  if (suma == m) printf("%d\t", i);
                  i++;
            }
           }

return 0;
}
```

Comment [Д.Ѓ.2]: на вежби да им се нагласи зошто е потребна помошна променлива n во која се копира i

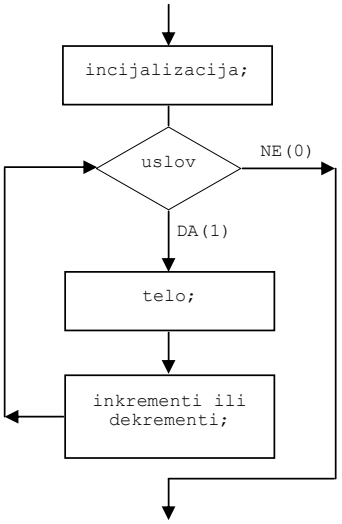
2. Структура за повторување (for)

Од аспект на синтакса **for** циклусот за повторување е најкомплициран. Сепак, сличен е на **while** циклусот, бидејќи и самиот содржи while тип на услов (се додека е исполнет, односно точен условот се извршуваат наредбите од телото).

Comment [Д.Ѓ.3]: та

for (**иницијализација; услови; инкременти или декременти**)
тело (наредба/наредби за повторување)**;**

Во for делот содржани се почетните услови (иницијализација на променливи), услов/и за повторување, како и наредби за инкрементирање или декрементирање, од кои зависи исполнувањето на условот.



Наредбите од делот **иницијализација** се извршуваат точно еднаш, на почетокот - пред влезот во циклусот. Наредбите од делот **услови** се извршуваат пред почетокот на секој нов циклус и доколку резултираат со вредност која се интерпретира како логичка вистина се влегува во повторување на наредбите од циклусот, инаку се завршува повторувањето на циклусот и се продолжува со наредбите што следуваат по циклусот во програмата. Наредбите од делот **инкременти или декременти** се извршуваат на крајот на секој циклус (по извршувањето на сите наредби од **телото** на циклусот) по што се извршуваат наредбите од делот **услови** и доколку се задоволени, циклусот се повторува. Исто како и кај if и while, доколку треба да се изврши само една наредба, таа едноставно се пишува по for делот, доколку, пак, треба да се извршат повеќе наредби (блок наредби) по for делот, тие се стават во блок ограничен со големи загради **{ }**.

ВАЖНО!!!
Мора да се запомни дека знакот **;** мора да ги одделува трите дела на **for** наредбата. И условот во for наредбата мора да биде од типот "**се додека е исполнет условот**".

```
int j;                                  j = 5
                                        j = 4
for (j = 5; j > 0; j--)                 j = 3
      printf("j = %d\n",j--);           j = 2
                                        j = 1
```

За разлика од програмскиот јазик PASCAL каде чекорот во for циклусот можеше да биде само 1, во C не постои ограничување за чекорот.

Пример:

```
#include <stdio.h>
#include <math.h>
int main()
{
      double agol;
      for(agol = 0.0; agol < 3.14159; agol += 0.2)
            printf("sinus od %.1lf e %.2lf\n", agol, sin(agol));
      return 0;
}
```

Во овој случај променливата agol се зголемува за 0.2 при секое извршување на наредбата for, се додека е исполнет условот agol < 3.14159.

Во деловите за иницијализација и менување на вредности од for наредбата може да се наведат повеќе изрази по потреба одвоени со знакот запирка(**,**). Запирката гарантира секвенцијално извршување на наведените изрази.

Пример:

```
int i, j, k;
for (i = 0, j = 5, k = -1; i < 10; i++, j++, k--)
```

Ако нема потреба од наведување на изрази во овие два дела, тие можат да бидат изоставени, но **НЕ СМЕАТ** да се заборават знаците **;**.

`for(; i < 10; i++, j++, k--)`

`for(; i < 10;)` подобро да се користи `while(i < 10)`

`for(;;)` се креира бесконечна јамка и се чита "засекогаш"(for ever)

Comment [Д.Ѓ.4]: пример
#define EVER (;;)
for EVER
{
...
}

3. break и continue

Од претходните примери за наредбите while и for се гледа можноста за креирање на бесконечен циклус. Знаејќи го ефектот од користењето на бесконечниот циклус произлегува дека треба да се одбегнува по секоја цена. Сепак, во C може да се користат бесконечни циклуси така што ќе се излегува, односно скокнува надвор од нив. Користењето на клучниот збор **break** овозможува излегување од било кој циклус, независно од условот и продолжување со првата следна наредба по циклусот.

Пример:

```
for(;;) {
     printf("vnesi int vrednost:");
     if(scanf("%d", &j) == 1)
        break;
     while((c = getchar()) != '\n')
           ;
}
printf("j = %d\n", j);
```

ЕКРАН:

vnesi int vrednost: an int
vnesi int vrednost: no
vnesi int vrednost: 16
j = 16

Во овој случај, ако scanf врати 1 (односно се вчита преку тастатура цел број), се излегува од бесконечната јамка.

И додека break предизвикува моментално излегување од циклусот, клучниот збор continue предизвикува извршување на следната наредба од циклусот. Во случај на while и do. . . while циклусите, continue предизвикува директно реевалуирање, односно извршување на условот. Кај for циклусите пак, предизвикува извршување на делот за менување на вредностите на променливите од for наредбата, па реевалуирање на условот.

Пример:

```
for(j = 1; j <= 10; j++) {
     if(j % 3 == 0)
        continue;
     printf("j = %d\n", j);
}
```

ЕКРАН:

j = 1
j = 2
j = 4
j = 5
j = 7
j = 8
j = 10

Задачи:

1. Да се напише програма која ќе ги испечати сите броеви од зададен опсег кои исто се читаат и од лево на десно и од десно на лево.

```
#include <stdio.h>

int main ()
{
      int  i, odb, dob, pom, prev, cifra;
      printf("Vnesete vrednost za opsegot.\n");
      printf("Od koj broj?\n"); scanf("%d", &odb);
      printf("Do koj broj?\n"); scanf("%d", &dob);
      for (i = odb; i <= dob; i++) {
            pom = i;
            prev = 0;
            while (pom > 0) {
                  cifra = pom % 10;
                  prev = prev*10 + cifra;
                  pom /= 10;
            }
            if (prev == i) printf("%d\t", i);
      }

return 0;
}
```

2. Да се напише програма која ќе ги испечати сите броеви помали од број N составени само од парни цифри и ќе врати колку такви броја има.

```
#include <stdio.h>

int main ()
{
      int  i, n, pom, prov, cifra, brojac;
      printf("Do koj broj treba da se proveruva?\n");
      scanf("%d", &n);
      brojac = 0;
      for (i = 1; i <= n; i++) {
            prov = 1;
            pom = i;
            while ((pom > 0) && prov) {
                  cifra = pom % 10;
                  if (cifra % 2) prov = 0;
                  pom /= 10;
            }
            if (prov) {
                        printf("%d\t", i);
                        brojac++;
                      }
      }
      printf("\nIma vkupno %d broevi so samo parni cifri\n", brojac);

return 0;
}
```

3. Да се напише програма која пресметува просек на еден студент, како и број на испити на кои студентот паднал. Бројот на испити што ги полагал студентот не е познат. Програмата завршува кога ќе се внесе вредност различна од број.

```
#include <stdio.h>

int main ()
{
      int  i, ocena, suma, broceni, brpadnal;
      i = 1;
      suma = broceni = brpadnal = 0;
      printf("Vnesete bilo koja bukva za kraj!!!\n");
      for (;i != 0;) {

      //citanje na ocenka

            do {
                  printf("Vnesete ocena :");
                  i = scanf("%d", &ocena);
            } while ((ocena < 5 || ocena > 10) && i);

      //presmetuvanje za validna ocenka pomegu 5 i 10

            if (ocena >= 6 && ocena <= 10 && i) {
                              suma += ocena;
                              broceni++;
                                          }
            else if (ocena == 5 && i) brpadnal++;
      }

      //pecatenje na dobienite rezultati

      if   (broceni)   printf("Prosekot   na   studentot   e   %.2f.\n",
(float)suma/broceni);
      else printf("Nema polozeno ispit studentot.\n");
      if (brpadnal) printf("Studentot padnal na %d.\n", brpadnal);
      else printf("Nema padnato na ispit studentot.\n");

return 0;
}
```

Comment [Д.Ѓ.5]: може да стои и само i наместо i!=0

Comment [Д.Ѓ.6]: Можеби е подобро наместо да се влечка && i во сите услови да им се даде и втора варијанта во која по првиот do...while ќе има if(!i) continue; (или break;)

4. Да се напише програма која пресметува плоштина на правоаголник. Вредностите на страните a и b, се менуваат во опсези зададени преку тастатура, со чекори 0.5 и 2.5 соодветно. Резултатите треба да бидат прикажани во форма на табела (вредности a и b, вредност на плоштината со точност од две децимали) за вредности на плоштината помали од производот на кубовите на почетните вредности на страните на правоаголникот. Задачата да се реши со користење на break.

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

int main ()
{
      float  a, b, c, d, gorg1, gorg2, dolg1, dolg2, P, pom;
      printf("Vnesete dolna i gorna granica za stranata a?\n");
      scanf("%f %f", &dolg1, &gorg1);
      printf("Vnesete dolna i gorna granica za stranata b?\n");
```

```
      scanf("%f %f", &dolg2, &gorg2);
      c = dolg1 * sqr(dolg1);
      d = dolg2 * sqr(dolg2);
      for (a=dolg1,b=dolg2; a <= gorg1 && b <= gorg2; a+=0.5,b+=2.5)
      {
            P = a * b;
            pom = c * d;
            if (P < pom) printf("a= %.2f\t\tb= %.2f\t\tP= %.2f\n",a,b,P);
            else {
                  printf("Nadminata e dozvolenata vrednost za plostinata
na pravoagolnikot od %.2f.\n", pom);
                  break;
                 }
      }

return 0;
}
```

Comment [Д.Ѓ.7]: Vo standardnata math.h ne postoi funkcija sqr()