

Управляващи структури — оператори за цикъл

control flow loop statements while, for, do-while

ЦИКЪЛ

- последователност от инструкции, която се повтаря многократно, докато дадено условие е изпълнено
- Четири компонента
 - Инициализация – задава начално състояние при първата проверка на условието, изпълнява се еднократно и не е част от самия цикъл
 - Условие – до кога се изпълнява цикъла
 - Тяло – операциите, които се изпълняват всяко завъртане на цикъла
 - Промяна / актуализация – промяна на условието, така че цикълът да може да спре
- Итерация – едно завъртане на цикъла / едно изпълнение на тялото му

ЦИКЪЛ

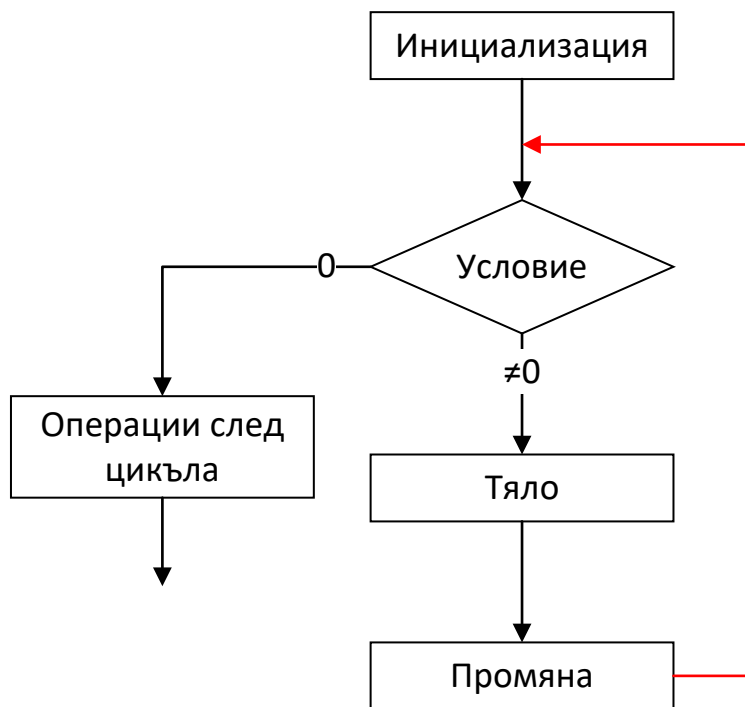
- Не всички компоненти присъстват задължително
 - Може инициализацията да е имплицитна
 - Може промяната да е част от тялото на цикъла
 - Някои оператори позволяват да няма условие – безкраен цикъл
- Тялото е задължително
- Тялото се състои от една операция или много операции, затворени в блок `{ }`
 - Забележка: Затварянето на тялото на цикъла в блок `{ }` е препоръчително дори и то да се състои само от една операция.

Оператори за прекъсване и прескачане на итерации

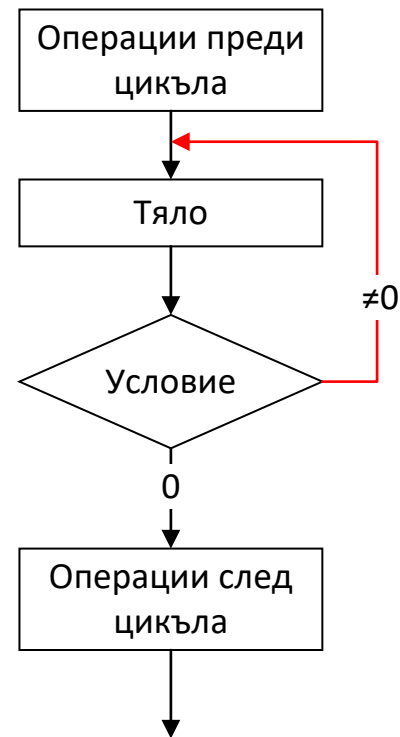
- Оператор break;
 - Прекъсва цикъла и програмата продължава с операциите след него
- Оператор continue;
 - Прекъсва текущата итерация на цикъла (прескача оставащите операции от тялото на цикъла след continue) и връща изпълнението на програмата към условието на цикъла

Цикъл с пред- и с пост-условие

Цикъл с пред-условие



Цикъл с пост-условие



Оператор while

Цикъл с пред-условие

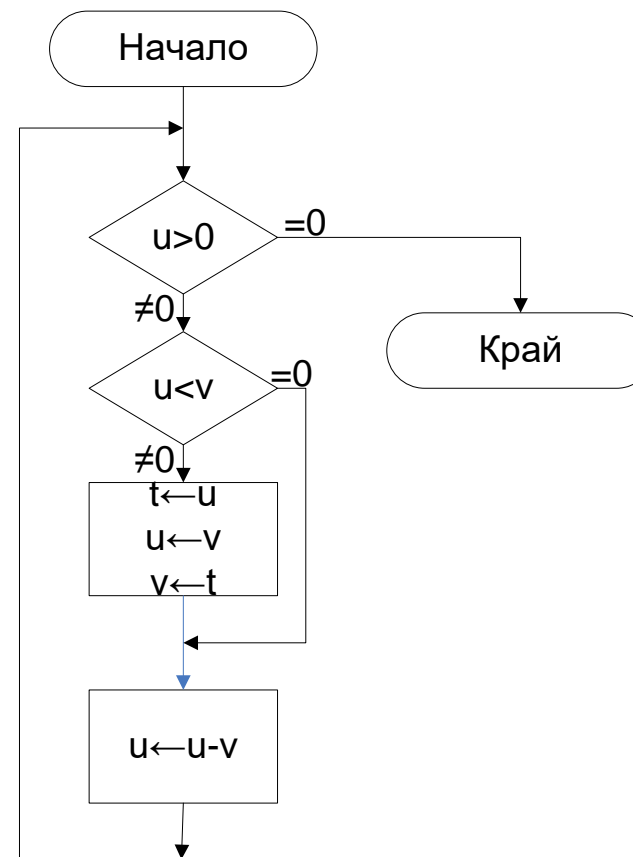
Синтаксис

```
while(условие)
{
  Операции от тялото на цикъла
}
```

- Условието се записва в скоби
- След скобите няма ; (ако има ; тя се приема за тялото на цикъла)
- Приема една подчинена операция
- Повече операции трябва да се затворят в блок { }

Алгоритъм на Евклид за намиране на най-големия общ делител

- Приложение – при съкращаване на дроби.
- Ако $u > v$, то най-големият общ делител е същият както най-големия общ делител на v и $u - v$.



Алгоритъм на Евклид – най-голям общ делител

```
#include <stdio.h>
int main ()
{ int u, v, t;
  printf ("Vavedi dve celi chisla: ");
  scanf ("%d%d", &u, &v);
  if (u > 0 && v > 0)
  {
    while (u > 0)
    {
      if (u < v)
      {
        t = u; u = v; v = t; // u и v си разменят стойностите
      }
      u = u - v;
    }
    printf ("Nai-golemiqt obst delitel e %d.\n", v);
  }
  else
    printf ("Dvete chisla trqbva da sa polojitelni!\n");
  return 0;
}
```

Поточна обработка

- Всяка итерация се чете стойност и веднага се обработва
- Следващата итерация се чете и обработва ново число
- Всяка обработвана стойност се съхранява само в рамките на итерацията, в която се обработва
- Обработката спира при настъпване на определено събитие
 - Въвеждане на конкретна терминираща стойност
 - Обработка на определен брой стойности

Намиране на сума на поток от числа, до въвеждане на 999

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

```
int main()
```

```
{
```

```
    int number, s;
```

```
    s = 0;
```

```
    printf ("Въведи последователност от цели числа (999 за край): ");
```

```
    scanf ("%d", &number); // инициализация
```

```
    while (number != 999) // условие
```

```
    {
```

```
        s += number; // тяло
```

```
        scanf ("%d", &number); // промяна
```

```
    }
```

```
    printf ("Сума = %d\n", s);
```

```
    return 0;
```

```
}
```

Намиране на средно-аритметично на положителните стойности

```
#include <stdio.h>
int main()
{
    int number; //входни данни
    float avr; //изходни данни
    int sum, broi; //помощни променливи
    sum = broi = 0;
    printf ("Vavedete chislo (999 za krai): ");
    scanf ("%d", &number); // инициализация
    while (number != 999) // условие
    {
        if(number > 0) //средно-аритметично само на положителни числа
        {
            sum += number; // тяло
            broi++;
        }
        printf ("Vavedete chislo (999 za krai): ");
        scanf ("%d", &number); // промяна
    }

    if(broi > 0) //proverka, za da nqma delenie na 0
    {
        avr = (float)sum / broi; //float cast, за да не е целочислено делението
        printf ("Srednoto aritmetichno na polojitelnite chisla e %.2f.\n", avr);
    } else
    {
        printf("Nqma vavedeni stoinosti.");
    }
    return 0;
}
```

Проверка дали поток числа е геометрична прогресия – пример break

```
#include <stdio.h>

int main()
{
    int k, broi, isGP;
    float N, prevN, r; //pazim tekusta stoinost (N) i predishna stoinost (prevN), za da moje da se izchislqva r za vseki dve (posledovatelni) chisla

    isGP = 1; //priemame, che chislata ste formirat geom. progresiq

    printf("Vavedete broi chisla:");
    scanf("%d", &broi);

    printf("Vavedete 1 chislo ot redicata:");
    scanf("%f", &prevN);
    printf("Vavedete 2 chislo ot redicata:");
    scanf("%f", &N);

    k = 2;
    while (k < broi)
    {
        if (k == 2)
        {
            r = N / prevN;
        }
        else
        {
            if (N / prevN != r)
            {
                isGP = 0; //chislata ne obrazuvat geom. progresiq
                break;
            }
        }
        prevN = N; //tekustoto chislo stava predishno chislo za sledvastata iteraciq ot cikala
        printf("Vavedete %d chislo:", ++k);
        scanf("%f", &N);
    }

    if (isGP)
    {
        printf("Chislata obrazuvat geometrichna progresiq.");
    }
    else
    {
        printf("Chislata ne obrazuvat geometrichna progresiq.");
    }

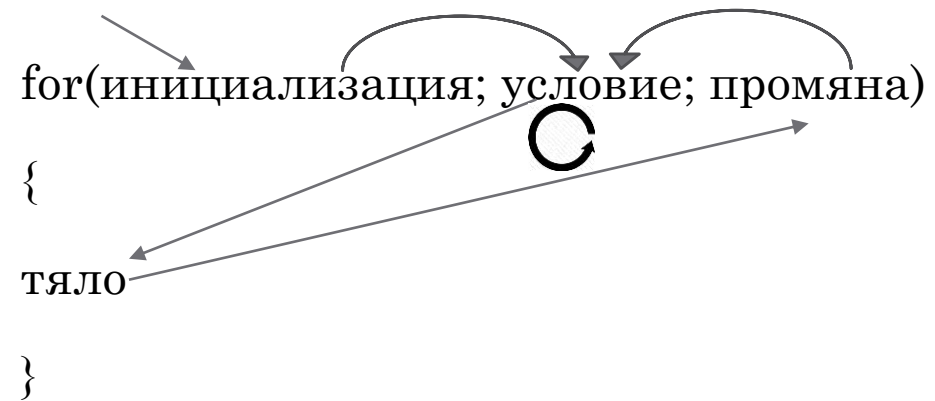
    return 0;
}
```

Оператор for

Цикъл с пред-условие

Синтаксис

for(инициализация; условие; промяна)
{
 тяло
}



The diagram illustrates the syntax of a C-style for loop. It features the text 'for(инициализация; условие; промяна)' on the first line, followed by a block of code enclosed in curly braces: '{', ' тяло', and '}'. Annotations include: a straight arrow pointing from the word 'Синтаксис' to the opening parenthesis of the 'for' statement; two curved arrows above the opening parenthesis, one from 'инициализация' to 'условие' and another from 'условие' to 'промяна'; a circular arrow with the letter 'C' inside, positioned between the opening and closing braces; and two straight arrows originating from the 'C' circle, one pointing to the 'тяло' (body) and the other pointing to the closing parenthesis of the 'for' statement.

Пример – проверка дали число е просто

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

```
int main()
{
    int i, n;
    printf("Vavedete chislo: ");
    scanf("%d", &n);
    for(i = 2; i < n; i++)
        if(n % i == 0) //if(!(n%i))
        {
            printf("Chisloto %d ne e prosto!", n);
            return 0; //izhod ot programata
        }

    printf("Chisloto %d e prosto!", n);

    return 0;
}
```


Специални случаи

- Няколко операции в частите за инициализация / актуализация
 - `for(i = 0, k = 1; i < n; i++, k+=2)`
 - Няколко операции се разделят с ,
- Няколко условия се обединяват с логически операции
 - `for(i=n-1; i >= 0 && k < m; i—)`
- Без инициализация / актуализация
 - `for(; var > 0;)`
 - Инициализацията, условието, актуализацията могат да бъдат пропуснати, но ; са задължителни

Оператор do-while

Цикъл с пост-условие

Синтаксис

```
do  
{  
Операции от тялото на цикъла  
}while(условие);
```

- Тялото на цикъла винаги се изпълнява поне един път (за разлика от цикъл с пред-условие, когато може да не се изпълни нито веднъж)

Проверяване на вход от потребителя

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main()
{
    int number;

    do
    {
        printf("Vavedete chislo mejdu 1 i 25: ");
        scanf("%d", &number);
    }while(number < 1 || number > 25); //докато числото не е между 1 и 25 цикълът се върти

    return 0;
}
```

Програма за отгатване на число – с МНОГО ОПИТИ

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main()
{
    int number, guess;

    srand(time(NULL));
    number = rand() % 100;

    do
    {
        printf("Otgatnete chisloto: ");
        scanf("%d", &guess);
        if (guess == number)
            printf("Poznahte!\n");
        else if (guess > number)
            printf("Chisloto e mnogo golqmo!\n");
        else
            printf("Chisloto e mnogo malko!\n");
    }
    while(guess != number);

    printf("Chisloto e %d", number);

    return 0;
}
```

Програма за отгатване на число – ограничаване на броя опити

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main()
{
    int number, guess, tries = 0;
    const int max_tries = 3;

    srand(time(NULL));
    number = rand() % 100; //ограничаване на стойностите между 0 и 99

    do
    {
        printf("Отгатнете числото: ");
        scanf("%d", &guess);
        tries++;
        if (guess == number)
            printf("Познахте!\n");
        else if (guess > number)
            printf("Числото е много голямо!\n");
        else
            printf("Числото е много малко!\n");
    }
    while(guess != number && tries < max_tries);

    printf("Числото е %d", number);

    return 0;
}
```