Цикли

Изпадаме в следната ситуация: намираме се на работното си място, 10 минути преди да приключи нашата работна седмица и наш колега ни възлага следната задача, от която зависи издигането ни в йерархията - (напишете програма, която за въведено число извежда следващите 1000 числа, тоест всяко е с 1 по-голямо от предходното). С моментните ни знания ние ще започнем да пишем нещо от сорта:

```
int x; cin > x;
std::cout << x + 1 << std::endl;
std::cout << x + 2 << std::endl;
…
std::cout << x + 1000 << std::endl;
```

След 45 минути копиране и поставяне на код ние вече сме изпълнили задачата, отиваме да се похвалим на пичагата, който ни я възложи. Вървейки с гордо вдигната глава, докато не долавяме крясъци от съседния офис. Повишават колегата, на когото бяха възложили същата задача. Аз започнах да се извинявам с това, че компютърът ми е по-бавен и мишката ми прекъсва и затова съм закъснял толкова с решаването на проблема. Излизам с наглостта и аз да получа повишение и колегите ми се изсмяха подигравателно. Чуваше се нещо от сорта: този младеж не е ли чувал за нещо наречено лууп. Облян в сълзи, вървейки с наведена глава, замерян с опаковки от snickers, отидох в кабинета си. Намерих въпросното нещо в интернет. Оказа се, че нещото се нарича loop и означава цикъл. Зачетох се и попаднах на следната информация…

Цикличен процес (цикъл) - многократно изпълнение на оператори за различни стойности на дадени параметри. По-лесно казано: прилагане на едни и същи оператори за различни параметри , докато не се наруши дадено условие.

Части на цикъл - всеки цикъл се състои от 4 части:

- инициализация: задава се начална стойност на даден брой променливи, които участват в цикъла. Могат да бъдат и наричани “корекция”.
- тяло: блок, съдържащ операторите, които искаме да прилагаме многократно.
- актуализация: обновяване на стойностите на някои променливи с цел използването им при следващата итерация.
- условие за край: израз, който чиято стойност може да се представи чрез булева, гарантиращ приключването на цикъла (ако го няма ще попаднем в безкраен цикъл).

For-loop

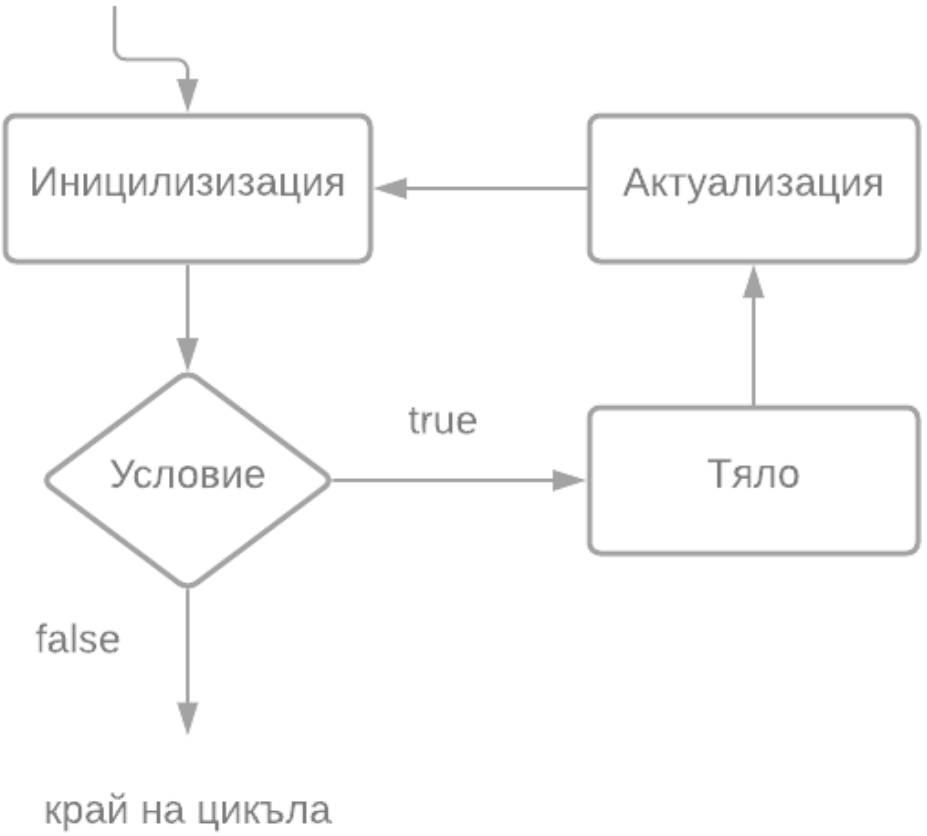
Синтаксис:

```
for (<инициализация>; <условие>; <актуализация>) {
    // тяло
}
```

Изпълнение на цикъла:

1. Инициализиране на променливите
2. Проверка за край на цикъл (условие)
    a. Ако стойността от условието е true, то продължаваме към стъпка 3
    b. Ако стойността от условието е false, то цикълът приключва

3. Изпълнява се тялото на цикъла
4. Извършва се актуализация, която променя някои от стойностите на променливите.
5. Връщаме се на стъпка 2 и така докато цикъла приключи.



While-loop

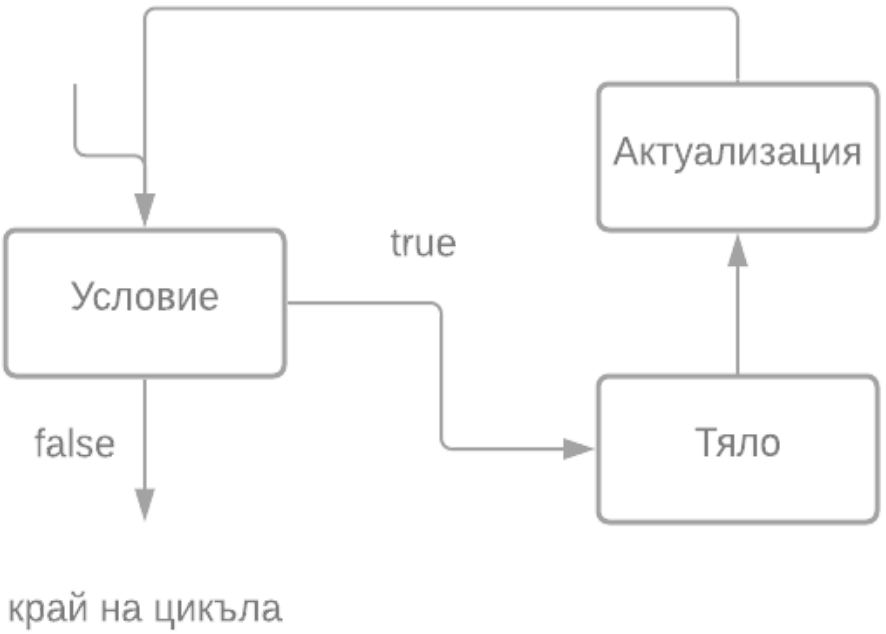
Синтаксис:

```
while (<условие>) {
        // тяло
}
```

Изпълнение на цикъла:

1. Проверява се условието

    a. ако има стойност false, цикълът приключва изпълнението си
    b. ако има стойност true, продължаваме със стъпка номер 2
2. Изпълнява се тялото на цикъла, където трябва да има актуализация, иначе ще попаднем в безкрайно изпълнение на оператора while.
3. Връщаме се отново на стъпка 1 и така, докато условието не придобие стойност false.



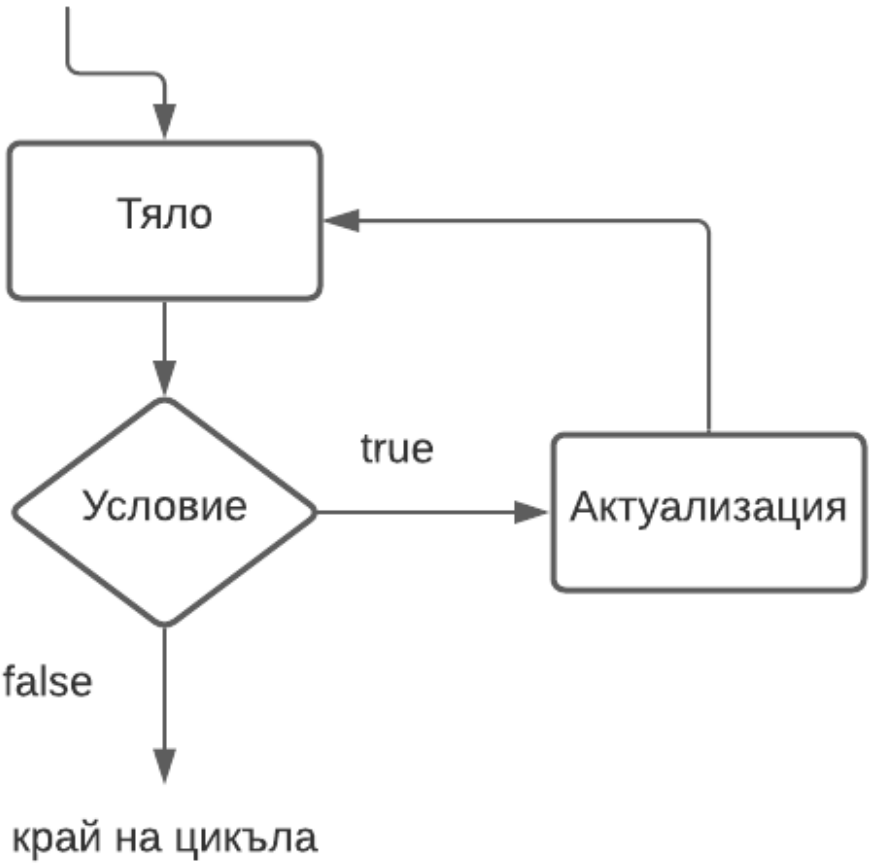
Do-while-loop

Синтаксис:

```
do {
        // тяло
}
while (<условие>);
```

Изпълнение на цикъла:

1. Изпълнява се тялото на цикъла, където трябва да има някаква актуализация.
2. Проверява се условието
    a. ако то има стойност true - връщаме се на стъпка 1
    b. ако то има стойност false - цикълът приключва



Важно: актуализацията при while и do-while се намира в тялото на цикъла, а инициализацията - извън него. Всеки един от 4-те елементи може да бъде пропуснат,

обаче това може да доведе до неправилно изпълнение на програмата ни. Задължително трябва да имаме граничен случай, тоест такъв случай, който прекратява цикъла, в противен случай изпадаме в безкраен такъв.

Вложени цикли - когато в тялото на един цикъл имаме друг цикъл, то ние ще казваме, че имаме вложени цикли (nested loops). Най-често ще ги срещаме при обхождане или каквато и да е работа с матрица.

Оператор break - да предположим че имаме крайна редица от числа и търсим число, примерно числото 3. За пример нека имаме следната редица: 8 7 2 9 3 20 10 30 23 100. Започваме последователно да сравняваме всеки елемент от редицата с нашето число. След като вече сме достигнали търсеното число, на нас вече не ни е нужно да обиколим и останалите елементи. При такъв случай се използва оператора break. Ние принудително прекъсваме цикъла.

Използване на flag - освен break, можем да използваме помощни флагове. Един флаг служи за проследяване на някакво условие, като сменя стойността си измежду 0 и 1. По принцип флаговете биват от булев тип.

Оператор continue - за разлика от оператора break, който прекратява цикъла, то continue преминава на следваща итерация, тоест текущата се прекъсва. Тоест частта от кода, която се намира след continue бива прескочена/игнорирана.

Блок и видимост на данни - Видимост на променливите, се нарича частта от програмата, където можем да достъпим дадените променливи. Видимостта, спрямо това как я разглеждаме, можем да я разделим в 3 категории:

- Вътре във функция или блок, се нарича локална променлива.
- В дефиниция на функция - нарича се формален параметър (ще го разгледаме подробно скоро).
- Вън от всички функции, тогава се нарича глобална променлива.

Локалните променливи са видими само в конкретния блок в който са декларирани и във всеки негов подблок. Глобалните променливи са дефинирани извън всички функции, обикновено най - отгоре в програмата. Те могат да бъдат достъпни от всички блокове в програмата.