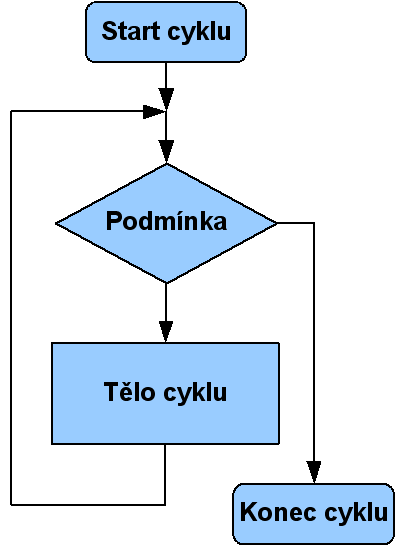
3. Řídící struktury – cykly

Využíváme, jakmile chceme něco několikrát za sebou opakovat.

Řízeny řídící proměnnou.

Jedno vykonání těla cyklu se nazývá **iterace.**

## **Cyklus s podmínkou na začátku (while cyklus)**

1. Nejprve se vyhodnotí (Booleovský) výraz v závorce za while a provede se bod 2.
2. Pokud:
3. Je výraz pravdivý, tak se provede blok cyklu a dále se pokračuje opět bodem 1.
4. Není výraz pravdivý, tak se provede bod 3.
5. Program pokračuje za cyklem while.

Tento cyklus nemusí proběhnout ani jednou.

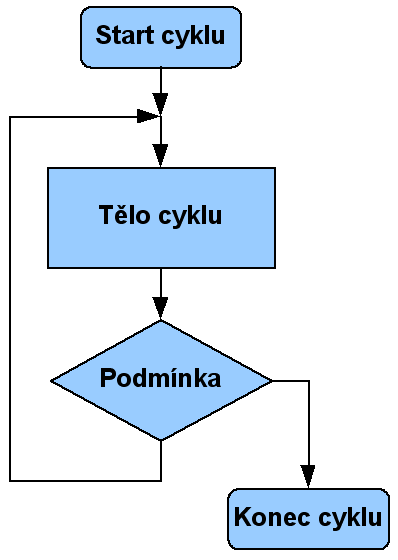
Zvláštním případem tohoto cyklu je cyklus **for**.

while(podmínka){

//kod

}

## **Cyklus s podmínkou na konci (do – while)**

1. Provede se tělo cyklu 
2. vyhodnotí (Booleovský) výraz v závorce za while
3. Je výraz pravdivý, tak se provede blok cyklu a dále se pokračuje opět bodem 1.
4. Není výraz pravdivý, tak se provede bod 3.
5. Program pokračuje za cyklem do – while.

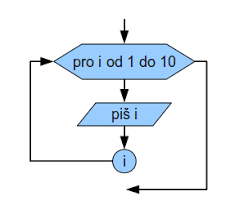
Tento cyklus proběhne alespoň jednou.

do{

//kod

}while(podmínka);

## **Cyklus s určeným počtem opakování (for)**

Tento cyklus má stanovený pevný počet opakování a hlavně obsahuje tzv. řídící proměnnou (celočíselnou), ve které se postupně během běhu cyklu mění hodnoty. 

for (promenna; podminka; prikaz){

//kod

}

**promenna** je řídící proměnná cyklu, které nastavíme počáteční hodnotu (nejčastěji 0).

**podminka** je podmínka vykonání dalšího kroku cyklu. Jakmile nebude platit, cyklus se ukončí.

**prikaz** nám říká co se má v každém kroku s řídící proměnnou stát. Tedy zda se má zvýšit nebo snížit.

## **Řídící proměnná**

Provést úplně identický kód opakovaně se někdy hodí, ale většinou chceme provést v těle cyklu trochu jiné příkazy, v závislosti na tom, která iterace se zrovna vykonává.

K tomu můžeme použít proměnnou, která si budeme pamatovat, v jaké iteraci cyklu se nacházíme, a podle ní se poté provede odpovídající operace.

Této proměnné jsou během cyklu měněny hodnoty (obvykle těsně za nebo před podmínkou).

Řídí chování cyklu.

## **Skokové příkazy**

Slouží pro alteraci chodu cyklů a smyček.

### **Break**



Ukončí nejbližší uzavřený iterační příkaz (tj for. , foreach, , while, nebo do smyčku) nebo switch příkaz. Příkaz break přenese řízení na příkaz, který následuje za ukončeným příkazem, pokud existuje.



### **Continue**



Spustí novou iteraci nejbližšího uzavřeného příkazu iterace (to znamená , for, foreach, while, nebo do smyčka).



### **Return**

Ukončí provádění funkce, ve které se zobrazí, a vrátí řízení a výsledek funkce, pokud existuje, volajícímu. Lze i použít i bez návratové hodnoty.



Využíván především u funkcí.



### **Goto**

Přenese řízení na příkaz, který je označen popiskem.



V případě nepřeskakování program pokračuje dále.



Pouze odignoruje název labelu ale vyplní příkazy pod ním.

