**Cikls “while” (Java)**

**Ievads**

Programmas, ko mēs rakstām mācību laikā, kļūst arvien sarežģītākas un apjomīgākas. Tās joprojām ir ļoti tālu no reālām programmām, kuru kodola rindu skaits sasniedz desmitiem un simtiem tūkstošu, bet pašreizējā sarežģītība jau var izraisīt stresu cilvēkiem bez pieredzes.

Sākot no šīs nodarbības, mēs pārejam pie vienas no visgrūtākajām pamattemām programmēšanā - cikliem.

Jebkura lietojumprogramma kalpo ļoti pragmatiskiem mērķiem. Tās palīdz pārvaldīt darbiniekus, finanses, un galu galā, tās izklaidē. Neskatoties uz atšķirībām, visām šīm programmām ir kopīgi iestrādātie algoritmi, kas ļoti līdzinās viens otram.

Algoritms ir darbību vai norādījumu secība, kas mūs ved uz kādu gaidāmu rezultātu. Šī definīcija attiecas uz jebkuru programmu, bet ar algoritmiem parasti saprot kaut ko specifiskāku.

Iedomājieties, ka mums ir grāmata un mēs vēlamies atrast konkrētu frāzi tajā. Mēs atceramies pašu frāzi, bet nezinām, uz kuras lapas tā atrodas. Kā atrast nepieciešamo lapu?

Vienkāršākais un ilgākais veids ir secīgi pārskatīt grāmatu, līdz mēs atradīsim vēlamo lapu. Sliktākajā gadījumā mums būs jāpārskata visas lapas, bet rezultātu mēs tomēr iegūsim.

Tieši šis process tiek saukts par algoritmu. Tas ietver lapu pārbaudi un loģiskās pārbaudes, vai mēs atradām frāzi vai nē. Lapu skaits, ko būs jāpārskata, nav zināms iepriekš, bet pats pārskatīšanas process atkārtojas vienādi katru reizi.

Lai veiktu atkārtojošas darbības, tieši tāpēc ir vajadzīgi cikli. Katra tāda atkārtošanās tiek saukta par iterāciju.

**Teorija**

**1.Darbības princips "WHILE"**

**While** cikls Java valodā ļauj izpildīt atkārtojošu kodu bloku tik ilgi, kamēr noteiktā loģiskā nosacījuma vērtība ir patiesa. Kods iekš **while** cikla tiks izpildīts, kamēr nosacījums ir patiess. Kad nosacījums kļūst nepatiess, cikla izpilde tiek pārtraukta, un vadība tiek nodota nākamajam kodu blokam pēc **while** cikla.

**2.No kā sastāv “WHILE”**

**"WHILE "sastāv no trim elementiem**

* **Galvenā atslēgvārda -**"while".
* **Prēdikāts -** nosacījums, kas norādīts iekavās pēc "while" un tiek novērtēts katrā iterācijā.
* **Cikla ķermenis** - koda bloks, kas atrodas figūriņu iekšpusē un ir līdzīgs metodes koda blokam. Visas konstantes vai mainīgie, kas definēti šajā blokā, būs redzami tikai šajā blokā.

**3.Kā rakstīt "WHILE" kodā?**

while (nosacījums) {

// izpildāmā koda bloks

}

**4. Kā pārtraukt ciklas "WHILE" darbību?**

Lai apturētu "while" cikla darbību, ir nepieciešams izmainīt nosacījumu tā, lai tas kļūst nepaties. Kad nosacījums kļūst nepaties, cikls tiek pārtraukts un programmas vadība tiek nodota pēc cikla bloka. Bet arī var izmantot komandu "break;".

**5.Kurš simbols norāda cikla "WHILE" sākumu un beigas?**

“{“ - Sākums

“}” - Beigas

**6.Bezgalīgs cikls “WHILE.**

Dažos gadījumos bezgalīgas ciklas ir noderīgas. Šādi izskatās bezgalīgas ciklas nosacījumi:

while (true) {

// kaut ko darām

}

**7.Kas notiek, ja ciklas "WHILE" nosacījums ir apzināti nepatiess?**

Ja nosacījums ir zināmā veidā nepareizs, tad cikls neizpildīsies ne reizi.

**8.Varētu būt interesanti**

-1 == -- / mīnus viens var ierakstīt kā divus mīnusus.

+1==++/ plus viens var ierakstīt kā divus plusus.

Lai apvienotu mainīgā vērtību ar citu vērtību while cilpā, varat izmantot operatoru "+=".

**PIEMĒRI**

**1. Izdrukājiet skaitļus no 1 līdz 5, izmantojot while ciklu.**

int i = 1; // deklarē mainīgo "i" un piešķir tam vērtību 1. Tas norāda uz to, ka cikla izpilde sāksies no skaitļa 1.

while (i <= 5) // sāk ciklu "while" ar nosacījumu, ka cikls turpināsies, kamēr "i" būs mazāks vai vienāds ar 5.

{

System.out.println(i); // izvada mainīgā "i" vērtību uz ekrāna, izmantojot "println" funkciju. Tā rezultātā tiek izvadīts skaitlis, kas atbilst aktuālajai "i" vērtībai.

i++; // Šī rindiņa palielina mainīgā "i" vērtību par 1 pēc katras iterācijas. Tas nodrošina, ka cikls pārejot no vienas iterācijas uz nākamo, "i" vērtība palielinās par 1.

}

**2. Summējiet skaitļus no 1 līdz 10, izmantojot while ciklu.**

int sum = 0;// Deklarē mainīgo "sum" un piešķir tam sākotnējo vērtību 0. Šis mainīgais tiks izmantots, lai uzglabātu summu, kā arī veikt papildinājumus.

int i = 1; // Šī rindiņa deklarē mainīgo "i" un piešķir tam vērtību 1. Tas norāda, ka cikla izpilde sāksies no skaitļa 1.

while (i <= 10) // Šī rindiņa sāk ciklu "while" ar nosacījumu, ka cikls turpināsies, kamēr "i" būs mazāks vai vienāds ar 10.

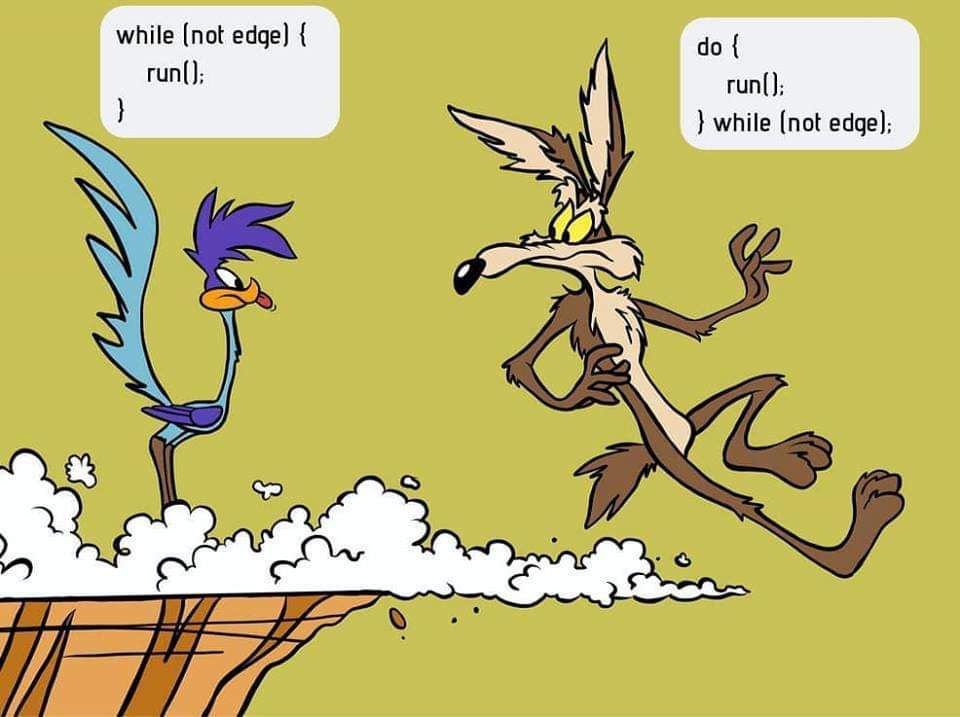
{

sum += i; // Palielina mainīgā "sum" vērtību, pievienojot tam "i" vērtību. Tas nozīmē, ka katrā iterācijā tiek veikts papildinājums, pievienojot pa vienam skaitlim no 1 līdz 10.

i++;

}

System.out.println(" Summa skaitļus no 1 līdz 10: " + sum);// Šī rindiņa izvada rezultātu uz ekrāna, izmantojot "println" funkciju. Tiek izvadīts teksts "Summa skaitļus no 1 līdz 10: ", kam seko "sum" mainīgā vērtība, kas ir aprēķinātā summa no 1 līdz 10.



**3.Nejaušo skaitļu ģenerēšana, līdz tiek ģenerēts skaitlis 7, izmantojot while ciklu.**

import java.util.Random;// Šī rindiņa importē klasi "Random" no "java.util" pakotnes. Tā ir nepieciešama, lai varētu izmantot gadījuma skaitļu ģeneratoru.

Random random = new Random();// Izveido jaunu objektu "random" no klases "Random". Tas ļauj mums izmantot gadījuma skaitļu ģeneratoru, kas tiks izmantots vēlāk.

int randomNumber;// Šī rindiņa deklarē mainīgo "randomNumber", kas tiks izmantots, lai uzglabātu gadījuma skaitli, kas tiks ģenerēts.

while (true) // Šī rindiņa sāk bezgalīgu ciklu "while", jo "true" vienmēr ir patiesa vērtība. Tas nozīmē, ka cikls turpināsies, līdz tam tiek piemērots "break" operators.

{

randomNumber = random.nextInt(10);// Šī rindiņa ģenerē gadījuma skaitli, izmantojot "nextInt(10)" metodi, kur "10" norāda, ka gadījuma skaitļa diapazons ir no 0 līdz 9. Tad šis gadījuma skaitlis tiek piešķirts mainīgajam "randomNumber".

System.out.println(randomNumber);// Šī rindiņa izvada gadījuma skaitli uz ekrāna, izmantojot "println" funkciju.

if (randomNumber == 7) { // Šī rindiņa pārbauda, vai gadījuma skaitlis ir vienāds ar 7.

break; // Ja nosacījums ir patiess, tad izpildās "break" operators, kas pārtrauc cikla izpildi un pārejai uz nākamo pēc cikla esošo kodu.

}

}

**Izmantotie avoti**

* <https://ru.hexlet.io/courses/java-basics/lessons/while/theory_unit>
* <https://www.w3schools.com/java/java_break.asp>
* <https://itproger.com/spravka/java/while>
* https://skolo.lv/course/view.php?id=103573