Liepājas valsts tehnikums

**Cikls ar skaitītāju programmēšana valoda Java Sesijas eksāmens**

Darba autors: Alina Bokareva

Liepāja 2024

Saturs

[1. Kas ir cikls ar skaitītāju? 1](#_Toc168842452)

[1.1 Cikla ar skaitītāju darbības princips 1](#_Toc168842453)

[1.2 Sintakse 1](#_Toc168842454)

[2. For ciklu veidi 2](#_Toc168842455)

[2.1 Parastais For cikls 2](#_Toc168842456)

[2.2 Cikls for-each 2](#_Toc168842457)

[2.3 Cikls ciklā 3](#_Toc168842458)

[2.4 Bezgalīgs for cikls 4](#_Toc168842459)

[3. Operatori darbam ar ciklu 5](#_Toc168842460)

[4. Piemēri, kas parāda cikla ar skaitītāju nozīmi 6](#_Toc168842461)

[4.1 piemērs 6](#_Toc168842462)

[4.2 piemērs 7](#_Toc168842463)

[4.3 piemērs 8](#_Toc168842465)

[5. Avoti 9](#_Toc168842466)

# Kas ir cikls ar skaitītāju?

Bieži vien, rakstot programmu, ir nepieciešams vairākas reizes atkārtot vienu un to pašu darbību. Ja process ir jāatkārto divas vai trīs reizes, tas īpaši neietekmēs efektivitāti, bet ko darīt, ja veicamā darbība aizņem lielu skaitu koda rindu un tā ir jāatkārto vairāk nekā piecas vai pat desmit reizes? **Cikls ar skaitītāju** jeb **For cikls** ir optimālākais risinājums un pastāv tieši tādiem gadījumiem.

For cikls - ir cikls, kas ļauj veikt noteiktu darbību noteiktu reižu skaitu, neatkārtojot vienas un tās pašas koda rindas.

## 1.1 Cikla ar skaitītāju darbības princips

Dzīvē cikls ar skaitītāju tiek izmantots katru dienu. Piemēram, rakstot pārbaudes darbu, cilvēks analizē pirmo uzdevumu, izpilda to un pāriet pie nākamā uzdevuma, kamēr visi uzdevumi ir pabeigti. For cikls darbojas pēc līdzīga principa: tā darbība sākas no sākotnējās vērtības, katru reizi, kad darbība tiek izpildīta, tā palielina sākotnējo vērtību par noteiktu soli, līdz sākotnējā vērtība ir vienāda ar izpildīto darbību skaitu *(sk. pielikums 1.)*.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

pielikums 1.

## Sintakse

Cikla veikšanai ir jānorāda šādi parametri:

* sākuma vērtība - cikls sāk skaitīt darbības atkārtojumus no šī skaitļa.
* beigu vērtība - cikls beigsies, kad sākotnējā vērtība būs vienāda ar to vērtību.
* mainīgā vērtība izmaiņa (cikla solis) - skaitlis, kas tiks pievienots sākotnējai vērtībai pēc katras darbības izpildes.

Programmēšanas valodā Java For cikls izskatās sekojoši:

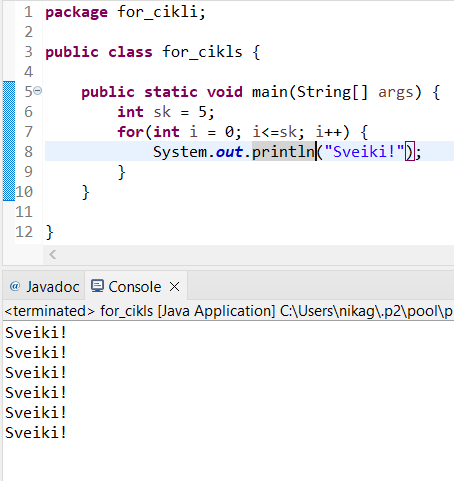
* **for(<sākuma vērtība >; <beigu vērtība>; <solis>){**

**<izpildāmās komandas>**

**};**

# For ciklu veidi

## 2.1 Parastais For cikls

 Visizplatītākais un ērtākais cikla ar skaitītāju veids. To izmanto, lai atkārtotu kādu darbību *(sk. pielikums 2.)*, attēlot skaitļu virkni un/vai aprēķinātu tās summu *(sk. pielikums 2.1.)*, utt. Lai to izpildītu, ir jāzina sākuma vērtība, beigu vērtība un solis (for cikla solis var būt arī dilstošs).

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

pielikums 2.1.

pielikums 2.

## Cikls for-each

**Изображение выглядит как текст, снимок экрана, дисплей, Шрифт

Автоматически созданное описание**Šāda veida ciklu izmanto tikai tad, ja nepieciešams veikt darbu ar masīva vai kolekcijas elementiem, piemēram, ArrayList. Kad cikls veic savu darbu, katra masīva/kolekcijas elementa vērtība tiek piešķirta cikla skaitītājam, līdz cikls ir izstaigājis visus elementus un izpildījis darbības ar tiem *(sk. pielikums 2.2).*

Atšķirība starp for-each ciklu un parasto for ciklu ir tāda, ka for-each var strādāt tikai ar masīviem un kolekcijām un neļauj mainīt to elementus. Tas nozīmē, ka, mēģinot ar to apstrādāt cita veida mainīgos, programma izraisīs kļūdu. For-each ciklam nav nepieciešams norādīt soli, jo tas to izdara pats, pēc noklusējuma tā solis ir 1.

pielikums 2.2.

For-each cikla sintakse ir sekojoša:

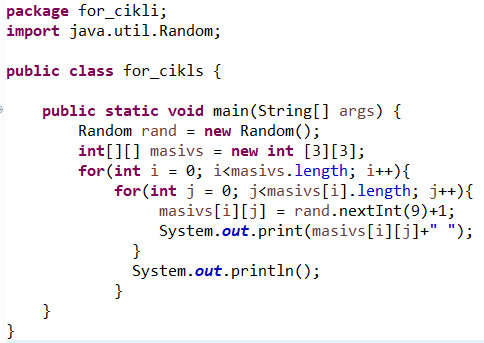
* **for (<skaitītāja tips un nosaukums> : < masīva vai kolekcijas nosaukums>){**

**<izpildāmās komandas>**

**};**

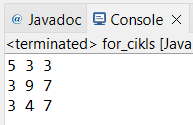
## Cikls ciklā

Cikla veids, kurā viens cikls tiek izpildīts citā ciklā, ļaujot veikt dažas darbības, kamēr tiek veiktas citas darbības. Šo veidu izmanto, lai aizpildītu un pārvietotos pa divdimensiju masīvu, lai izveidotu šķirošanas algoritmus (Bubble Sort, Selection Sort) un citiem mērķiem. Cikla ciklā darbības principu var labāk izprast, aplūkojot piemēru:

Aplūkosim piemēru par divdimensiju masīva piepildīšanu ar skaitļiem. (*sk. pielikums 2.3.*) Pirms cikla izveides tiek izveidots divdimensiju masīvs ar nosaukumu “masivs”, kā arī klase Random, lai nākotnē masīvu piepildītu ar nejaušiem skaitļiem. Pēc tam tiek izveidots ārējais cikls, kas iet cauri visiem elementiem masīva rindās, un vēl viens iekšējais for cikls, kas iet cauri elementiem masīva kolonnās. Abi cikli sākas no indeksa nulle un turpināsies, līdz beigsies masīva elementi. Ņemot vērā, ka masīva lielums ir trīs rindas un trīs kolonnas un ka ārējās un iekšējās ciklas atkārtojumu skaits ir jāreizina, šis cikls kopumā tiks izpildīts 9 reizes. Katram masīva elementam tiek nejauši izveidots skaitlis no 1 līdz 9, un tas pats elements tiek parādīts ekrānā. Pēc katras rindas programma veic rindas pārcelšanu, lai masīvs izdrukātā formā būtu pārskatāmāks.

pielikums 2.3.

Konsole izvades rezultāts pēc programmas palaišanas (*sk. pielikums 2.4.*):



pielikums 2.4.

## Bezgalīgs for cikls

For ciklu var izveidot arī bezgalīgu. To var izmantot, lai programma darbotos bezgalīgi vai lai veiktu kādu uzdevumu, līdz programma pārtrauc savu darbu. Ir divi veidi, kā padarīt ciklu bezgalīgu:

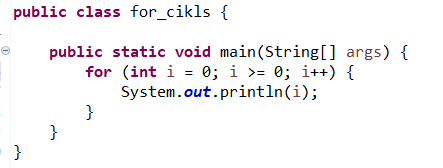
1. Nenoteikt parametrus *(sk. pielikums 2.5.).*

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

pielikums 2.5

1. Изображение выглядит как текст, Шрифт, снимок экрана, линия

   Автоматически созданное описаниеSākuma vērtību iestatīšana uz mainīgo, kas vienmēr būs true *(sk. pielikums 2.6. un pielikums 2.7.).*

pielikums 2.7.

pielikums 2.6.

Šajos piemēros ir redzams, ka noteiktajā iemesla dēļ galīgais nosacījums vienmēr paliek patiess: pirmajā piemērā *(pielikums 2.3.)* cikls turpinās tik ilgi, kamēr sākotnējā vērtība ir lielāka par nulli, bet, tā kā cikla solis ir viens, sākumvērtība vienmēr ir lielāka par nulli, savukārt otrajā gadījumā *(pielikums 2.4.)* cikls apstājas vienā punktā, jo pirms sākumvērtības palielināšanas par vienu no tās tiek atņemts tāds pats skaitlis, kas neļauj ciklam virzīties uz priekšu.

# Operatori darbam ar ciklu

Operatori tiek izmantoti, lai, strādājot ar cikliem, noteiktos gadījumos varētu mainīt ciklu darbību. Tas palīdz samazināt koda garumu un palielina programmas efektivitāti gadījumos, kad cikls tiek pārtraukts priekšlaicīgi.

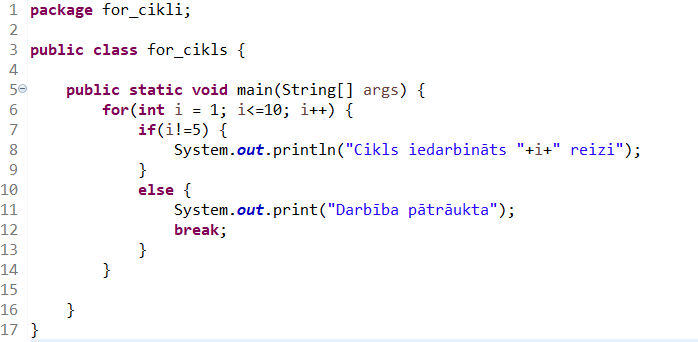
Ir daži populāri operatori, kas tiek izmantoti visbiežāk:

* break - tiek izmantots, lai pilnībā izietu no cikla.
* return - to izmanto arī, lai izietu no cikla.
* continue - izlaiž vienu cikla iterāciju un pāriet uz nākamo.

For ciklā tiek izmantoti arī operatori, kas salīdzina un piešķir vērtības, piemēram, gala un sākuma vērtību salīdzinājumā un soļa piešķiršanā. Piemēram:

* ++ - palielinājums (solim).
* -- - samazināšana (solim).
* += un -= saskaitīšana/atņemšana un piešķiršana (solim).
* > vai < - lielāks vai mazāks (beigu vērtību salīdzināšanai).
* >= vai <= - lielāks/mazāks vai vienāds (beigu vērtību salīdzināšanai).
* un daudzi citi.

Operatoru lietojuma piemērs (sk. pielikums 3.1.).:



pielikums 3.1.

Konsole izvades rezultāts pēc programmas palaišanas (sk. pielikums 3.2.):

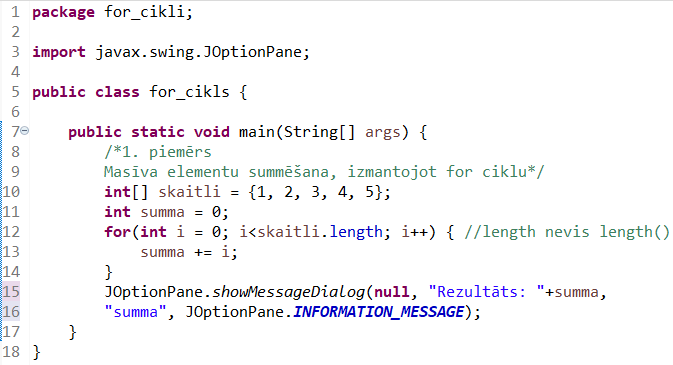
Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

pielikums 3.2.

# Piemēri, kas parāda cikla ar skaitītāju nozīmi

## 4.1 piemērs

Pirmajā piemērā parādīts (*sk. pielikums 4.1.*), kā For cikls samazina koda rindu skaitu. Šim nolūkam tika izdomāts neliels un ļoti vienkāršs uzdevums: aprēķināt visu masīva elementu summu. Sākumā izveidoju masīvu “skaitli” ar datu tipu int un pievienoju tam 5 skaitļus no 1 līdz 5. Pēc tam izveidojām atsevišķu mainīgu “summa”, kas ir vienāds ar 0: to izmantosim kā mainīgo, kuram pievienosim katru masīva "skaitli" elementu. Tālāk tika izveidota pati cikla ar skaitītāju, kura sākotnējā vērtība ir vienāda ar 0, pēdējai vērtībai jābūt mazākai par masīva "skaitli" garumu (***svarīgi!*** *length ir īpašība, kas atgriež masīva lielumu, bet length() ir metode, kas atdod simbolu skaitu teksta virknē*), un noslēgumā cikla solis ir viens. Cikla ķermenī i vērtība, kas satur masīva elementu, tiks pakāpeniski pievienota mainīgajam summa, un šī darbība turpināsies, līdz būs saskaitīti visi masīva elementi.

pielikums 4.1.

Pēc tam logā JOptionPane uz ekrāna tiek parādīts izpildes rezultāts pēc programmas palaišanas (*sk. pielikums 4.2*.):

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

pielikums 4.2

Neizmantojot ciklu for, vienu un to pašu koda rindu piecas reizes bija jāraksta, lai katru masīva “skaitli” elementu pievienotu mainīgajam “summa”. Tas nozīmē, ka cikls ar skaitītāju patiešām optimizē programmu, un tas ir skaidri redzams.

## 4.2 piemērs

Otrajā piemērā (*sk. pielikums 4.3*.) parādīts, kā operatorus break un continue var izmantot ciklā for. Uzdevums ir vienkāršs: ir jāatrod konkrēts elements masīvā. Tiek izveidots masīvs “skaitli”, kurā tiek saglabāti skaitļi no 1 līdz 10. Tad tiek izveidots mainīgais "atrast", kurā tiek saglabāts nejaušs skaitlis no 1 līdz 10 un teksta virkne “apskatitie”. Ciklā ar skaitītāju visi nosacījumi ir tādi paši kā iepriekšējā uzdevumā: cikls sākas ar 0 elementu, darbojas tik ilgi, kamēr sākotnējā vērtība ir mazāka par masīva “skaitli” izmēru un cikla solis ir viens. Pats cikls satur pārbaudi: ja i vērtība ir vienāda ar mainīgā “atrast” vērtību, cikls pārtrauc savu darbu un parāda programmas rezultātu (*sk. pielikums 4.4*.), pretējā gadījumā cikls pievieno i elementu virknei “apskatitie” un turpina darbu, līdz tiek atrasts vajadzīgais elements.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание.

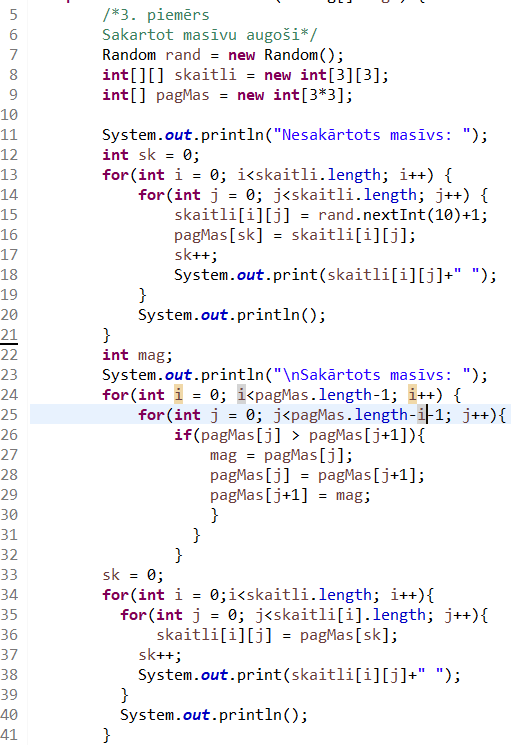
pielikums 4.3.

# 

pielikums 4.4.

Šis piemērs parāda, ka dažos gadījumos operatori break un continue patiešām ir jāizmanto un tie var ietekmēt programmas darbību.

## 4.3 piemērs

Trešajā piemērā (*sk. pielikums 4.5*.) šķirošana tiek veikta, izmantojot burbuļu šķirošanas algoritmu. Ir izveidoti divi masīvi: divdimensiju masīvs “skaitli”, kura lielums ir 3 rindas un 3 kolonnas, un viendimensiju masīvs “pagMas”, kura lielums ir masīva “skaitli” rindu reizinājums ar tā paša masīva kolonnām. Masīvs “pagMas” tika izveidots, lai uzglabātu masīva “skaitli” elementus, un nākotnē būs vieglāk kārtot skaitļus viendimensiju masīvā. Pēc tam nesašķiroto masīvu izvada ekrānā, izmantojot ciklu ciklā (darbības principu var aplūkot 2.3. tēmā).

pielikums 4.5.

Tālāk tiek izveidots mainīgais “mag”, kurā turpmāk īslaicīgi glabāsies masīva elementi. Tālāk tiek izveidota ārējais cikls, kas pakāpeniski apstaigās katru masīva “pagMas” elementu: cikla sākuma elements ir 0, un cikls beigsies pirms pēdējā elementa, kas nozīmē, ka tas darbosies vienu reizi mazāk, jo pēdējais elements automātiski sakārtosies pats. Iekšējais cikls arī sākas ar 0 elementu, un galīgā vērtība norāda, ka ar katru cikla atkārtojumu pārbaudāmo elementu skaits samazinās, jo katrs elements ieņem savu pozīciju. Iekšējās cikls pārbauda: vai pašreizējais elements ir lielāks par nākamo elementu? Ja tas tā ir, tad pašreizējais elements tiek saglabāts mainīgajā “mag”, nākamā masīva vērtība tiek ievietota pašreizējā elementa vietā, un pēc tam vērtība, ko esam saglabājuši mainīgajā “mag”, tiek pārvietota uz nākamā elementa vietu, tas nozīmē: pašreizējais un nākamais elements mainās vietām. Pēdējā koda daļa parāda sakārtoto masīvu uz ekrāna.

Programmas darbības rezultāts (*sk. pielikums 4.6*.):

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

pielikums 4.6.

Сikls ar skaitītāju tiek izmantots arī masīva elementu šķirošanai. Tā nav visefektīvākā metode, tā var būt mulsinoša, bet, neskatoties uz to, tā tiek izmantota un labi parāda, kā darbojas For cikls.

# 5. Avoti

1. <https://sky.pro/media/cikl-for-v-java/>
2. https://javarush.com/groups/posts/for-v-java
3. <https://itproger.com/course/java/7>
4. <https://vertex-academy.com/tutorials/ru/cikl-for-each/>
5. <https://gb.ru/blog/czikly-v-java-kak-sozdat-i-prervat/>
6. <https://www.programiz.com/java-programming/for-loop>