Mācību materiāls “Viendimensiju masīvi programmēšanas valodā Java”

...

Anželika Starceva 2pt

Saturs:

Ievads

Kas ir masīvs?

- Masīvs programmēšanā ir datu struktūra (objekts), kas sevī apvieno līdzīgus mainīgos ar vienādu datu tipu. Masīvā var uzglabāt primitīvos datu tipus, kā arī objektus.

Ievads

Masīvu lietošanas priekšrocības:

- Koda optimizēšana: kods kļūst optimizētāks, īsāks un vieglāk ir atlasīt un kārtot datus.
- Viegla piekļuve elementiem: iespējams piekļūt jebkuram masīvā esošam elementam lietojot tā kārtas indeksu.

Masīvu lietošanas trūkumi:

- Lieluma limits: masīvā iespējams uzglabāt tikai fiksētu elementu daudzumu un masīva lielumu nav iespējams izmainīt pēc tā izveides (ja vien nestrādā ar dinamiskajiem masīviem).

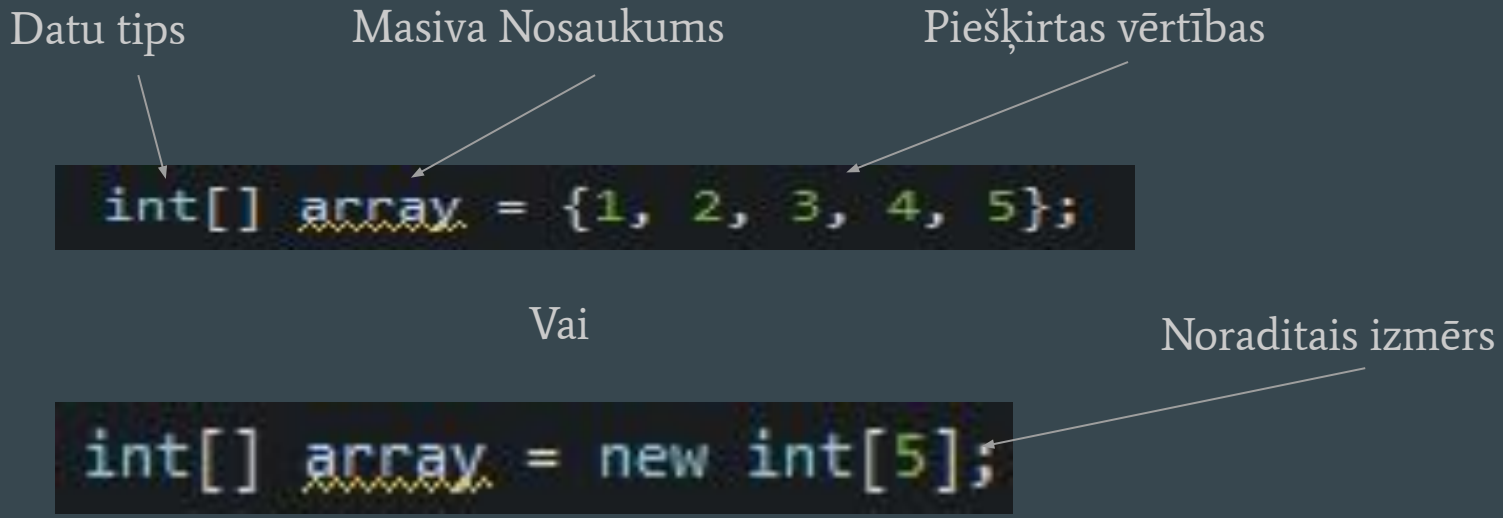
Ievads

Viendimensiju masīva pielietojums:

- Vienkāršības, efektivitātes un daudzpusības dēļ viendimensiju masīvus var izmantot dažādās jomās:
  - Saraksti un kolekcijas
  - Datu glabāšana un izguve
  - Krāvumi un rindas
  - Matricas un vektori
  - Dinamiskā programmēšana
  - Šķirošanas un meklēšanas algoritmi
  - Grafiku algoritmi
  - Histogrammas un frekvenču skaitīšana
  - Attēlu apstrāde
  - Kriptogrāfija

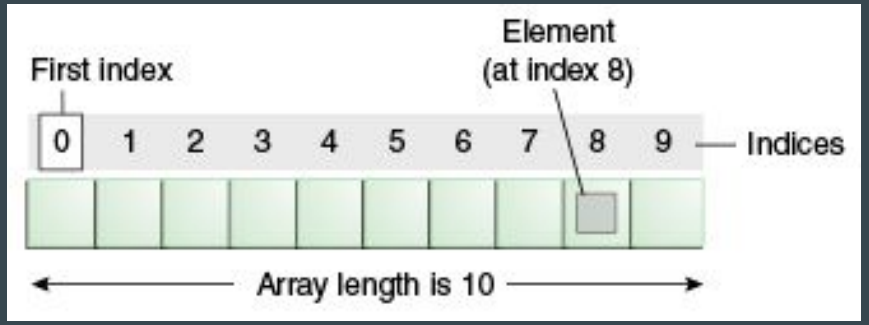
Masīvu Deklarācija un Iniciēšana

Lai deklarētu masīvu Java valodā, jums jānorāda elementu tips, sekojošs kvadrātiekavas pāris un mainīgā nosaukums. Masīva iniciēšanai nepieciešams norādīt tā izmēru vai arī tiešā veidā piešķirt vērtības.



Masīvu Piekļuve un Modificēšana

Katru masīva vienumu sauc par elementu, un katram elementam var piekļūt, izmantojot tā skaitlisko indeksu. Numerācija sākas ar 0!




```java
//Masīvs
int[] sk = {1, 2, 3, 4, 5};

//Piekļuve masīva elementiem
int pirmais = sk[0];
int otrais = sk[1];

//Elementu izvadīšana
System.out.println("Pirmais elements: "+ pirmais);
System.out.println("Otrais elements: "+ otrais);
```

```
Pirmais elements: 1
Otrais elements: 2
```

Masīva lieluma iegūšana

Masīva garumu var iegūt izmantojot īpašību length:

```
//Masīvs
int[] sk = {1, 2, 3, 4, 5};

//Masīva garums
int garums = sk.length;

//Izvada masīva garumu
System.out.println("Masīva garums: "+garums);
```

```
Masīva garums: 5
```

Viendimensiju masīvu datu tipi

- Datu tips: visiem masīva elementiem ir jābūt viena veida, piemeram:
  - byte
  - short
  - long
  - float
  - boolean
  - int
  - double
  - String
  - char

```
//int
int[] intArray = {1, 2, 3, 4, 5};

//double
double[] doubleArray = {1.1, 2.2, 3.3, 4.4, 5.5};

//char
char[] charArray = {'a', 'b', 'c', 'd', 'e'};

//boolean
boolean[] booleanArray = {true, false, true, false};
```

Operācijas ar masīviem

Populārākie veidi, kā manipulēt ar masīviem:

- Meklēt
- Atjaunināt
- Izvadīt
- Kārtot
- Apvienot
- Kopēt
- Apgriezt

Meklēšanas operācija

- Elementa meklēšana masīvā, izmantojot atslēgu; Atslēgu elements secīgi salīdzina katru masīva vērtību, lai pārbaudītu, vai atslēgs atrodas masīvā.

```
int LA[] = new int[5];
System.out.println("Masīvs:");
for (int i = 0; i < 5; i++) {
    LA[i] = i + 3;
    System.out.println("LA[" + i + "] = " + LA[i]);
}
for (int i = 0; i < 5; i++) {
    if (LA[i] == 6) {
        System.out.println("Elements " + 6 + " ir atrasts indeksā " + i);
    }
}
```

```
Masīvs:
LA[0] = 3
LA[1] = 4
LA[2] = 5
LA[3] = 6
LA[4] = 7
Elements 6 ir atrasts indeksā 3
```

Atjaunināšanas darbība

- Atjaunināšanas darbība attiecas uz esoša elementa atjaunināšanu no masīva noteiktā indeksā.

```
int LA[] = new int[5];
int item = 15;
System.out.println("Masīva elementi ir: ");
for (int i = 0; i < 5; i++) {
    LA[i] = i + 2;
    System.out.println("LA[" + i + "] = " + LA[i]);
}
LA[3] = item;
System.out.println("Masīva elementi pēc atjaunināšanas ir: ");
for (int i = 0; i < 5; i++) {
    System.out.println("LA[" + i + "] = " + LA[i]);
}
```

```
Masīva elementi ir:
LA[0] = 2
LA[1] = 3
LA[2] = 4
LA[3] = 5
LA[4] = 6
Masīva elementi pēc atjaunināšanas ir:
LA[0] = 2
LA[1] = 3
LA[2] = 4
LA[3] = 15
LA[4] = 6
```

Izvadīšanas darbība

- Šī darbība parāda visus elementus visā masīvā, izmantojot drukas paziņojumu.

```
int LA[] = new int[5];
LA = new int[5];
System.out.println("Masiva elementi ir: ");
for (int i = 0; i < 5; i++) {
    LA[i] = i + 2;
    System.out.println("LA[" + i + "] = " + LA[i]);
}
```

```
Masīva elementi ir:
LA[0] = 2
LA[1] = 3
LA[2] = 4
LA[3] = 5
LA[4] = 6
```

Masīva Kopēšana

- Java valodā var izmantot System.arraycopy metodi, lai kopētu masīvus.

```java
int[] arr = {1, 2, 3, 4, 5};
int[] merkis = new int[5];
System.arraycopy(arr, 0, merkis, 0, arr.length);

System.out.println("Arr array: ");
for (int i = 0; i < arr.length; i++) {
    System.out.print(arr[i] + " ");
}

System.out.println("\n\nMerkis array: ");
for (int i = 0; i < merkis.length; i++) {
    System.out.print(merkis[i] + " ");
}
```

```
Arr array:
1 2 3 4 5

Merkis array:
1 2 3 4 5
```

Masīva Kārtošana

- Izmantojot Arrays.sort metodi, varam sakārtot masīva elementus alfabēta secībā.

```
String[] augli = {"ābols", "banāns", "apelsīns"};
Arrays.sort(augli);

System.out.println(Arrays.toString(augli)); // ["apelsīns", "banāns", "ābols"]
```

```
[apelsīns, banāns, ābols]
```

Masīva Apvienošana

- Bieži vien ir nepieciešams apvienot divus vai vairākus String masīvus vienā.

```
String[] augli1 = {"ābols", "banāns"};
String[] augli2 = {"apelsīns", "citroni"};

// Izveidojam jaunu masīvu, kas var saturēt abu masīvu elementus
String[] apvienots = new String[augli1.length + augli2.length];

// Kopējam elementus no pirmā masīva
System.arraycopy(augli1, 0, apvienots, 0, augli1.length);

// Kopējam elementus no otrā masīva
System.arraycopy(augli2, 0, apvienots, augli1.length, augli2.length);

System.out.println(Arrays.toString(apvienots)); // ["ābols", "banāns", "apelsīns", "citroni"]
```

```
[ābols, banāns, apelsīns, citroni]
```

Masīva Elementu Apgriešana

- Apgriezīsim String masīva elementus pretējā secībā.

```
String[] augli = {"ābols", "banāns", "apelsīns"};
String[] apgriezts = new String[augli.length];

for (int i = 0; i < augli.length; i++) {
    apgriezts[i] = augli[augli.length - 1 - i];
}

System.out.println(Arrays.toString(apgriezts)); // ["apelsīns", "banāns", "ābols"]
```

```
[apelsīns, banāns, ābols]
```

Error ArrayIndexOutOfBoundsException

Situācijā ja cenšamies piekļūt masīva elementam ar indeksu, kurš nemaz nepastāv, JVM atgriež ArrayIndexOutOfBoundsException izņēmumu. Vispiežāk šādu situāciju nākas sastapt, ja ciklā kurš apstrādā masīvu, tiek norādīts nokorekts cikla nosacījums.

```
int[] arr = new int[4];

arr[0] = 10;
arr[1] = 20;
arr[2] = 30;
arr[3] = 40;

System.out.println(arr[5]);
```

Masīva lielums ir 4, un mēs vēlamies izvadīt elementu indeksā 5.

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 4
        at nana.da.main(da.java:16)
```

Avoti

- https://www.geeksforgeeks.org/one-dimensional-array-in-java/
- https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html
- https://www.tutorialspoint.com/data_structures_algorithms/array_data_structure.htm