Увод в програмирането

с Java

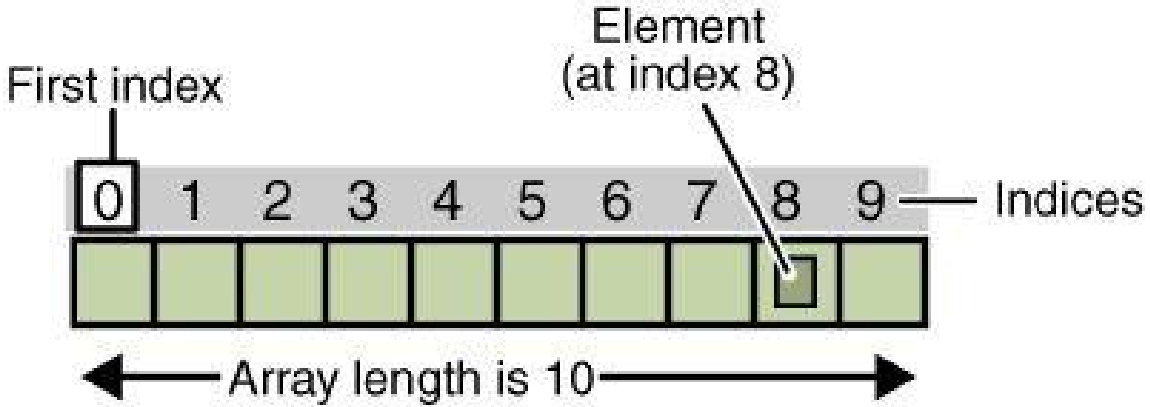
Масиви (II част)

Съдържание

- Преговор
- Сортиране
- Многомерни масиви
- Задачи
- Домашно

Преговор

Масив


Element
(at index 8)
First index
0 1 2 3 4 5 6 7 8 9
Indices
Array length is 10

Декларация

Има два начина:

•Когато знаем кои са елементите:

–int[] myArray = {1, 2, 3, 4, 5};

•Когато знаем само броя им:

–int[] myArray = new int[5];

– Операторът “new” заделя памет за масива.

Достъп

- Могат да се достъпват чрез []
- Индексът на първия елемент е 0
- Индексът на последния елемент е дължината на масива - 1

String[] names = {“Pesho”, “Gosho”, “Petkan”}; names[0]; //

“Pesho”

names[names.length - 1]; // “Petkan”

Обхождане

Често ни се налага да минем през всички елементи на масив и да направим нещо с тях. Този процес се казва обхождане.

Възможен е чрез цикли. int[] array = {3 ,5 ,6, 6,7,8};

```
for (int i = 0; i < array.length; i++){  array[i]=5+i;


}
```

Сортиране

Сортиране

Да сортираме един масив, означава да подредим елементите му според някакъв критерий.

Пример: Да наредим едно тесте от карти

Пример: Имаме масив от оценки на студенти и трябва да ги подредим по нарастване - от най-ниските към най-високите.

Пример: Имаме масив от имена на студенти и трябва да ги подредим по азбучен ред.

Задача

Имаме числата от 1 до 20, записани в масив с разбъркан ред. Как да ги подредим в нарастващ ред?

Алгоритми за сортиране

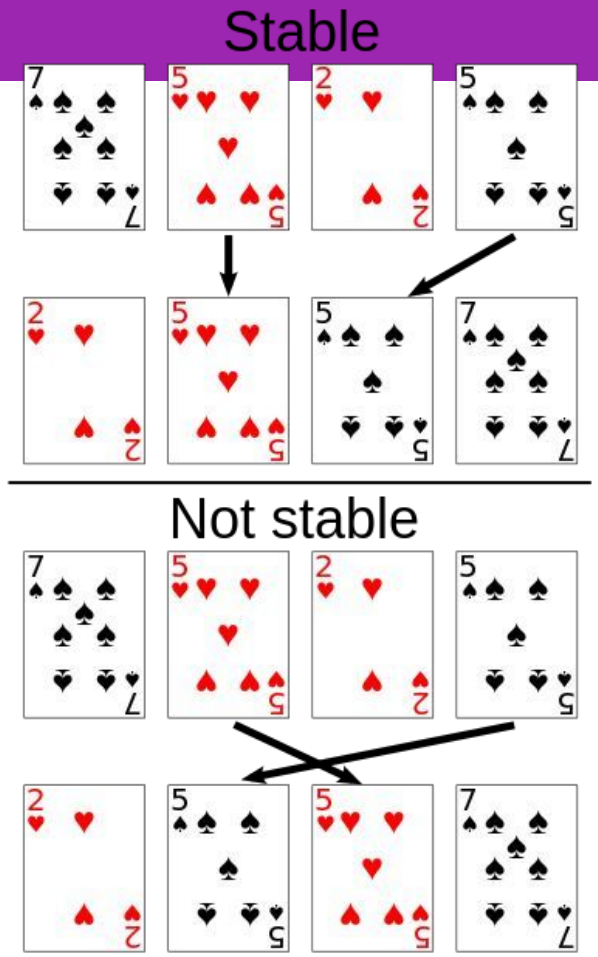
Алгоритмите за сортиране ни дават решение на задачата как да сортираме един масив от данни. Те са едни от основните в компютърните науки и се използват често, както за обучителни цели, така и в практиката.

Класифицираме ги според няколко критерия:

- стабилност
- бързина (времева сложност)
- използвана памет
- ефективност
- др.

Алгоритми за сортиране

Name	Best	Average	Worst	Memory	Stable
Quicksort	$n \log n$	$n \log n$	n^2	$\log n$	Depends
Merge sort	$n \log n$	$n \log n$	$n \log n$	Depends	Yes
In-place Merge sort	—	—	$n (\log n)^2$	1	Yes
Heapsort	$n \log n$	$n \log n$	$n \log n$	1	No
Insertion sort	n	n^2	n^2	1	Yes
Selection sort	n^2	n^2	n^2	1	Depends
Bubble sort	n	n^2	n^2	1	Yes



Алгоритми за сортиране

Наивни:

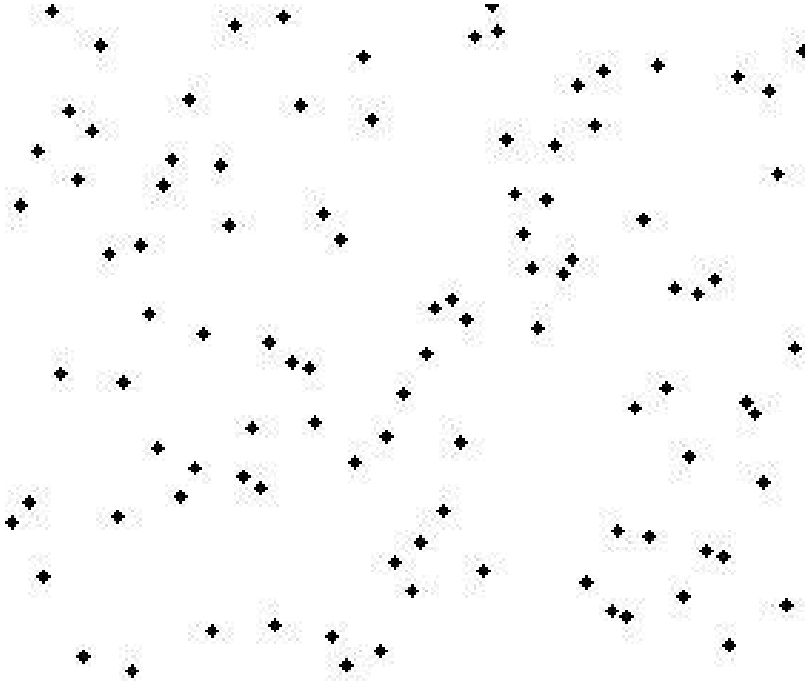
- “метод на мехурчето” (Bubble Sort)
- сортиране чрез вмъкване (Insertion Sort)
- сортиране чрез пряка селекция (Selection Sort)

Ефективни:

- бърза сортировка (Quick Sort)
- сортиране чрез сливане (Merge Sort)
- сортиране чрез двоична пирамида (Heap Sort)

“Метод на мехурчето” (Bubble Sort)

6 5 3 1 8 7 2 4



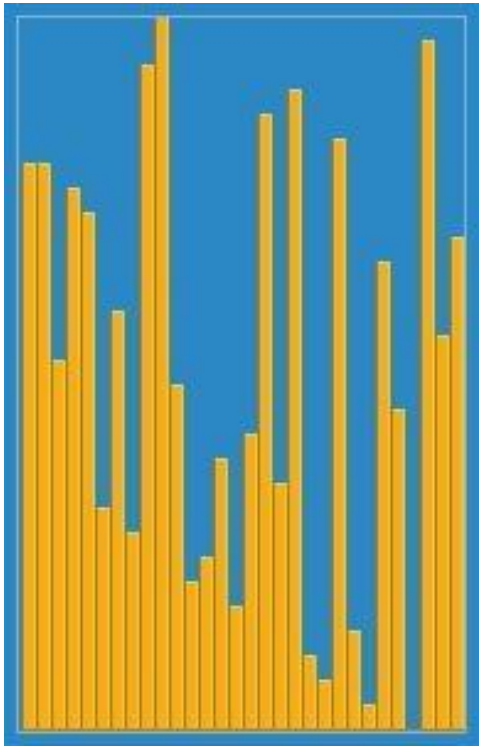
Bubble Sort (algorithm)

```
int[] array = { 6, 9, 3, 1, 8 };
int temp = 0;

for (int i = 0; i < array.length; i++) {
   for (int j = 1; j < (array.length - i); j++) {

      if (array[j - 1] > array[j]) {
         // swap the elements!
         temp = array[j - 1];
         array[j - 1] = array[j];
         array[j] = temp;
      }

   }
}
System.out.println(Arrays.toString(array));
```

Сортиране чрез вмъкване (Insertion Sort)

Псевдокод:

```
for i ← 1 to length(A) - 1  j ←
    i
    while j > 0 and A[j-1] > A[j]
        swap A[j] and A[j-1]  j
        ← j - 1
    end while  end for
```



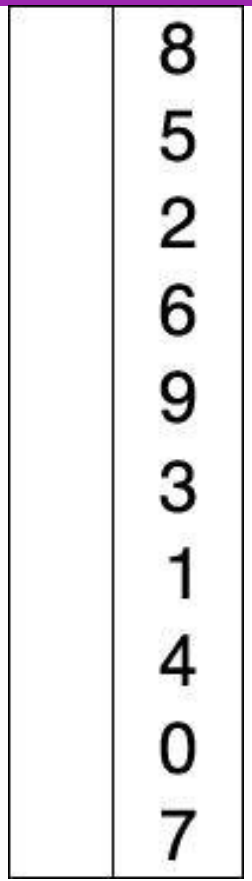
Insertion Sort (algorithm)

```
int[] array = { 3, 9, 8, 1, 6 };
for (int i = 1; i < array.length; i++) {
int j = i;
int temp;
while (j > 0 && array[j - 1] > array[j]) {
temp = array[j - 1];
array[j - 1] = array[j];
array[j] = temp;
j = j - 1;
}
}
System.out.println(Arrays.toString(array));
```

Сортиране чрез пряка селекция (Selection Sort)

Код:

```
for (int j = 0; j < n-1; j++) {  int

        iMin = j;

        for (int i = j+1; i < n; i++) {
        if (a[i] < a[iMin]) {
            iMin = i;
        }
                        }
       if(iMin != j) {
              swap(a[j], a[iMin]);
        }
}
```



Selection Sort (algorithm)

```
int[] array = { 3, 9, 2, 1, 6 };

for (int j = 0; j < array.length; j++) {
int minIndex = j;
for (int i = j + 1; i < array.length; i++) {
if (array[i] < array[minIndex]) {
minIndex = i;
}
}
int temp;
if (minIndex != j) {
temp = array[minIndex];
array[minIndex] = array[j];
array[j] = temp;
}
}
System.out.println(Arrays.toString(array));
```

За любознателните :)

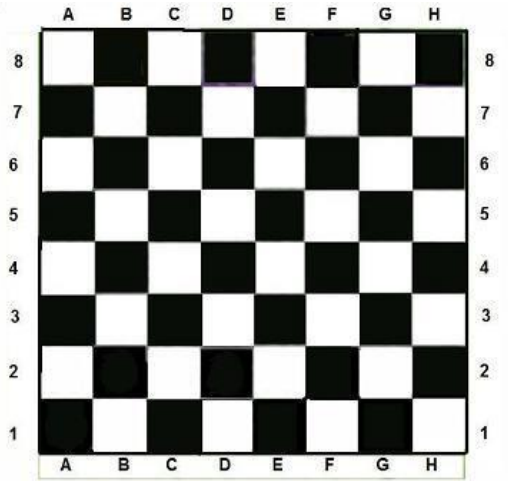
- Анимации + псевдокод на най-често използваните алгоритми

Многомерни масиви

Многомерен масив

Масивите, с които се занимавахме досега, представят един ред обекти от някакъв тип. Често обаче ни се налага да представяме данните под формата на таблици (напр., таблица с оценки за всеки студент, в която всеки ред е даден студент, а всяка колона - оценка по даден предмет.).

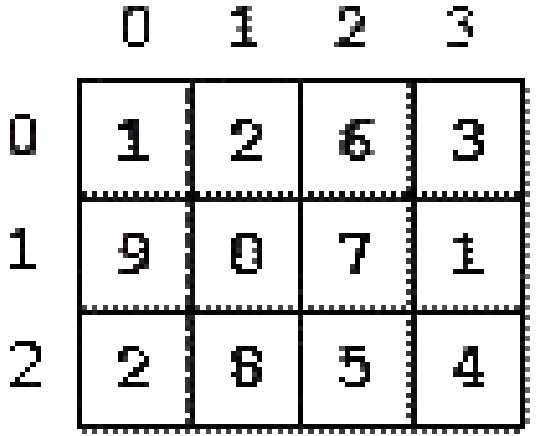
Id	Name	Email	Investments
231	Albert Master	albert.master@gmail.com	Bonds
210	Alfred Alan	aalan@gmail.com	Stocks
256	Alison Smart	asmart@biztalk.com	Residential Property
211	Ally Emery	allye@easymail.com	Stocks
248	Andrew Phips	andyp@mycorp.com	Stocks
234	Andy Mitchel	andym@hotmail.com	Stocks
226	Angus Robins	arobins@robins.com	Bonds
241	Ann Melan	ann_melan@iinet.com	Residential Property
225	Ben Bessel	benb@hotmail.com	Stocks
235	Bensen Romanolf	benr@albert.net	Bonds



Многомерен масив - декларация

Многомерните масиви са нищо друго, освен масив от масиви. Могат да имат n на брой измерения, но рядко в практиката се използват повече от 2.

```
int[][] twoDimentionalArray ;

int[][][] threeDimentionalArray ;

int[][] intMatrix = new int[3][4];
float[][] floatMatrix = new float[8][2];
String[][][] stringCube = new String[5][5][5];
```



Многомерен масив - инициализация

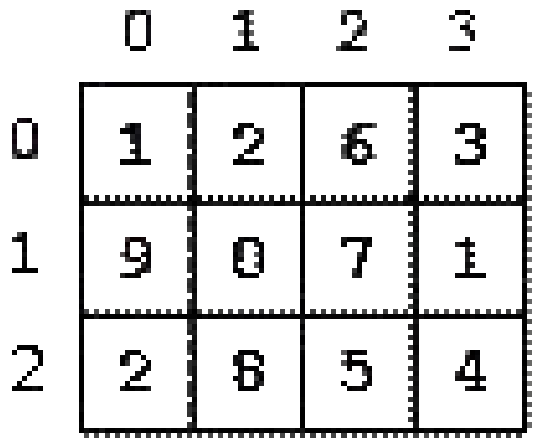
Инициализираме многомерните масиви по същия начин, както и едномерните.

```
int[][]   matrix {
    =                 // row 0
    {1, 2,  3,  4}, values
    {5, 6,  7,  8}, // row 1
};                    values
```

Матрица

Матриците са двумерни масиви, т.е. масив от едномерни масиви. Имат редове и колони, като всеки ред е масив от елементи, а всяка колона - съвкупност от елементи с еднакъв индекс.

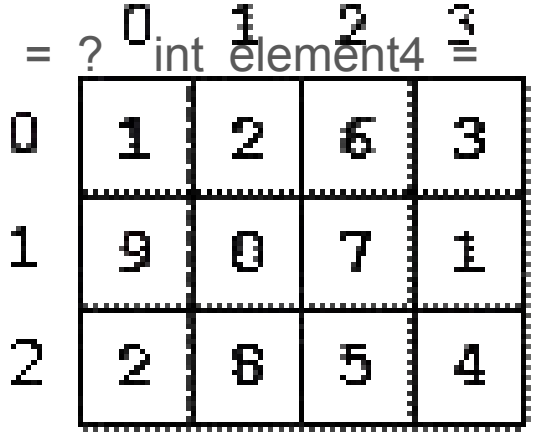
```
int[][] matrix = {
        {1, 2, 6, 3},
        {9, 0, 7, 1},
        {2, 8, 5, 4}
};
```



Достъп до елементите на многомерен масив

Както при едномерните масиви, можем да достъпваме елементите и на многомерен масив. За да вземем даден елемент, трябва да посочим номер на ред и номер на колона:
matrix[i][j]; Например:

int element1 = matrix[0][1]; // element1 = 2 int element2 = matrix[2][2]; // element2 = 5 int element3 = matrix[1][2]; // element3 = ? int element4 = matrix[3][0]; // element4 = ?



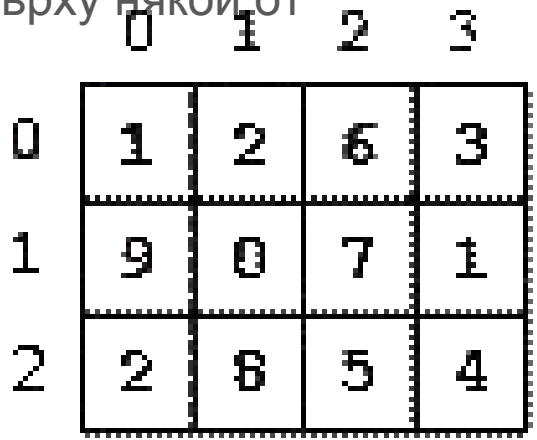
Размер на матрица

За да намерим броя на редовете на една матрица, използваме метода . length. Тъй като матрицата е просто масив от едномерни масиви, length ни дава размера на този масив:

```
int rows = matrix.length; // 3
```

За да намерим броя на колоните, прилагаме length върху някой от редовете, например:

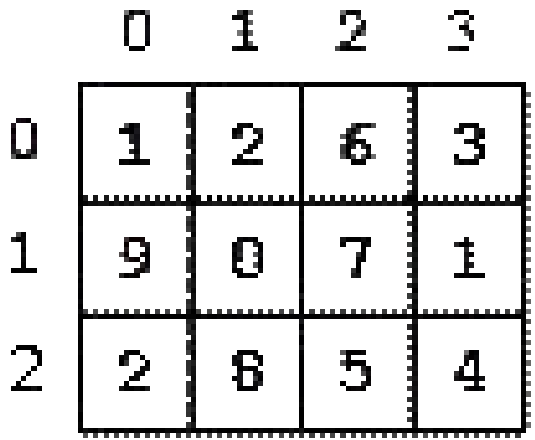
```
int columns =  matrix[0].length; // 4
```



Обхождане на матрица

Обхождаме матриците по същия начин, както и едномерните масиви, само че тук трябва да използваме вложени цикли (съответно за да минем през всеки ред и всяка колона):

```
for (int i = 0; i < rows; i++){
        for (int j = 0; j < cols; j++){  matrix[i][j] += 1;


        }

}
```



Задачи

Задача

Имате 3 критика, всеки от които е дал оценка за 4 филма. Оценките са представени чрез следната таблица:

reviewer \ movie	0	1	2	3
0	4	6	2	5
1	7	9	4	8
2	6	9	3	7

Представете данните по подходящ начин (използвайте двумерен масив).

Задача

Изведете на екрана данните от таблицата.

reviewer \ movie	0	1	2	3
0	4	6	2	5
1	7	9	4	8
2	6	9	3	7

Решение:

```
int[][] reviews = {
{4, 6, 2, 5},
{7, 9, 4, 8},
{6, 9, 3, 7},
};
int rows = reviews.length;
int cols = reviews[0].length;

for (int i = 0; i < rows; i++) {
for (int j = 0; j < cols; j++) {
System.out.print(reviews[i][j] + " ");
}
System.out.println();
}
```

Задача

Намерете каква е средната стойност на оценките, дадени от рецензент #2.

reviewer \ movie	0	1	2	3
0	4	6	2	5
1	7	9	4	8
2	6	9	3	7

Задача

```
int sum = 0;
for (int i = 0; i < reviews[2].length; i++) {
sum += reviews[2][i];
}
System.out.println("Average score for reviewer 2: " + (double) sum/cols);
```

Задача

Намерете броя на оценките над 6.

reviewer \ movie	0	1	2	3
0	4	6	2	5
1	7	9	4	8
2	6	9	3	7

Задача

Запишете всички оценки в едномерен масив и го сортирайте, използвайки някоя от разгледаните сортировки.

		movie			
		0	1	2	3
reviewer	0	4	6	2	5
	1	7	9	4	8
	2	6	9	3	7

Домашно

Задача 1:

Имате 3 критика, всеки от които е дал оценка за 4 филма. Оценките са представени чрез следната таблица:

Принтирайте средната оценка за филм #3.

reviewer \ movie	0	1	2	3
0	4	6	2	5
1	7	9	4	8
2	6	9	3	7

Задача 2:

Една квадратна таблица от числа се нарича магически квадрат, когато е изпълнено следното условие: всички суми, получени поотделно от сбора на елементите по всеки ред, всеки стълб и всеки от двата диагонала са равни.
Да се състави програма, която проверява дали матрицата дадена по-долу е магически квадрат.

16,3,2,13
5,10,11,8
9,6,7,12
4,15,14,1

Задача 3:

Имате предварително въведени стойности от цели числа, принадлежащи на интервала [10..99]. числата са въведени в матрица с размери 6 реда и 6 колони.
Да се състави програма, чрез която се намира сумата на всички елементи в колони с нечетни номера: 1, 3 и 5 по отделно.

Матрица:
11,12,13,14,15,16,
21,22,23,24,25,26,
31,32,33,34,35,36,
41,42,43,44,45,46,
51,52,53,54,55,56,
61,62,63,64,65,66

Изход:
11, 21, 31, 41, 51, 61 сума от елементите на колоната е: (сумата от числата в реда)
13, 23, 33, 43, 53, 63 сума от елементите на колоната е: (сумата от числата в реда)
15, 25, 35, 45, 55, 65 сума от елементите на колоната е: (сумата от числата в реда)