



Увод в програмирането с Java



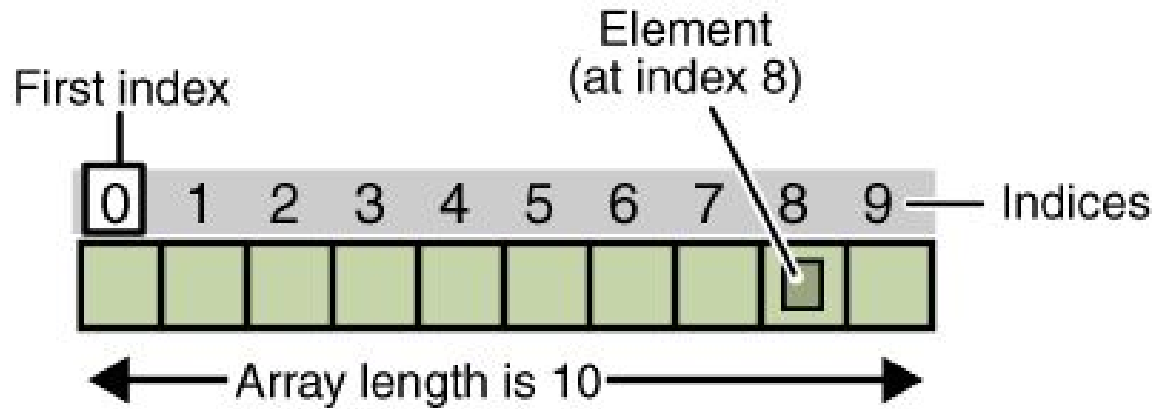
Масиви (II част)

Съдържание

- Преговор
- Сортиране
- Многомерни масиви
- Задачи
- Домашно

Преговор

Массив



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

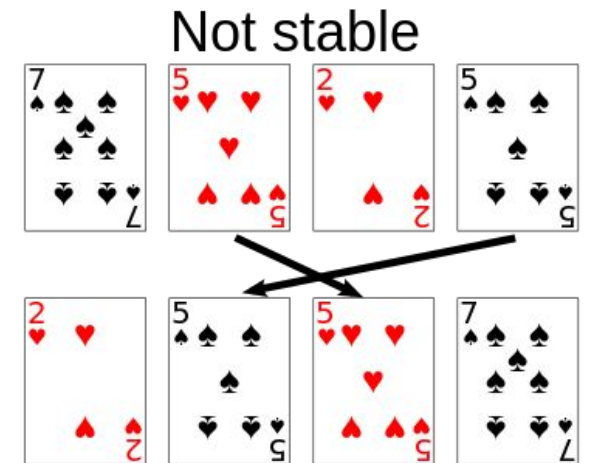
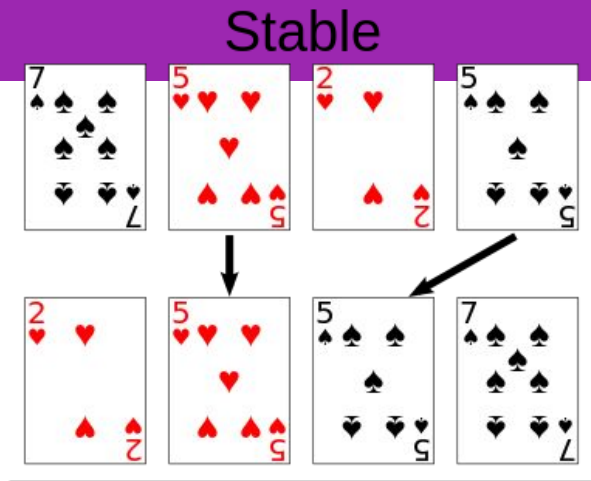
[Алгоритмите за сортиране](#) ни дават решение на задачата как да сортираме един масив от данни. Те са едни от основните в компютърните науки и се използват често, както за обучителни цели, така и в практиката.

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:

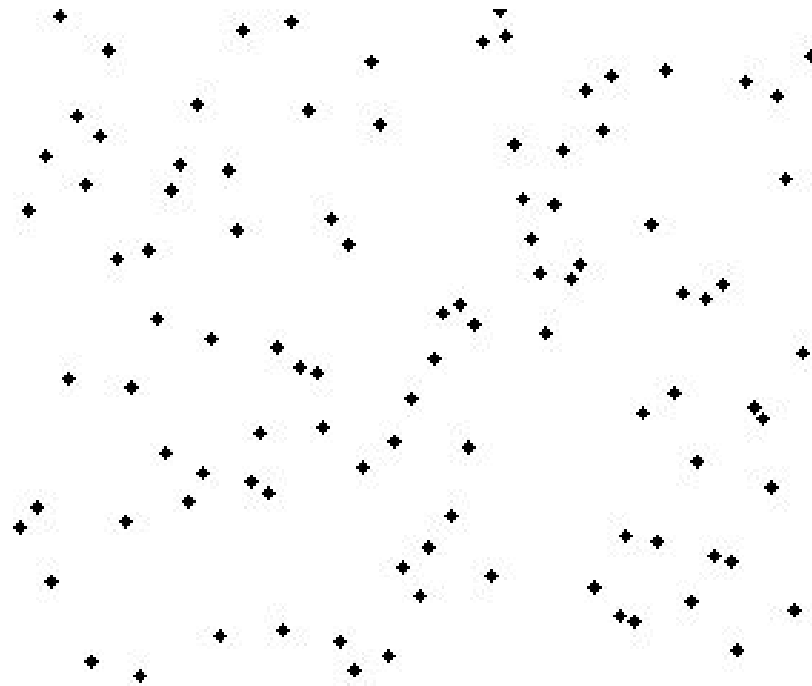
- [“метод на мехурчето” \(Bubble Sort\)](#)
- [сортиране чрез вмъкване \(Insertion Sort\)](#)
- [сортиране чрез пряка селекция \(Selection Sort\)](#)

Ефективни:

- [бърза сортировка \(Quick Sort\)](#)
- [сортиране чрез сливане \(Merge Sort\)](#)
- [сортиране чрез двоична пирамида \(Heap Sort\)](#)

“Метод на мехурчето” (Bubble Sort)

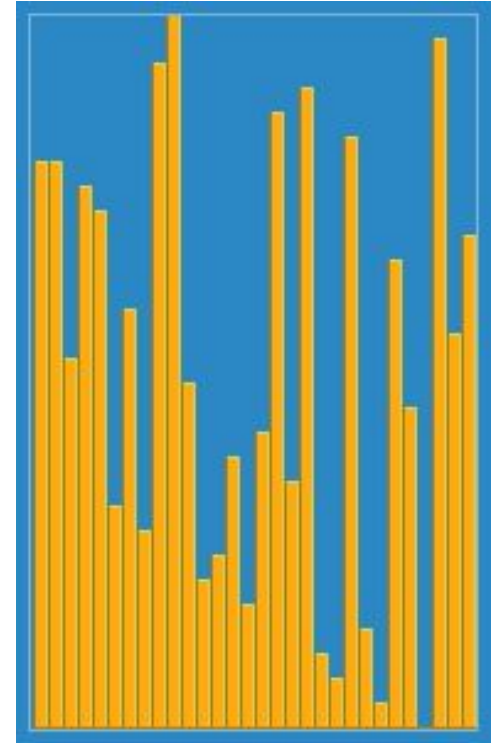
6 5 3 1 8 7 2 4



Сортиране чрез вмъкване (Insertion Sort)

Псевдокод:

```
for  $i \leftarrow 1$  to  $\text{length}(A) - 1$   
   $j \leftarrow i$   
  while  $j > 0$  and  $A[j-1] > A[j]$   
    swap  $A[j]$  and  $A[j-1]$   
     $j \leftarrow j - 1$   
  end while  
end for
```



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]);  
    }  
}
```

8
5
2
6
9
3
1
4
0
7

За любознателните :)

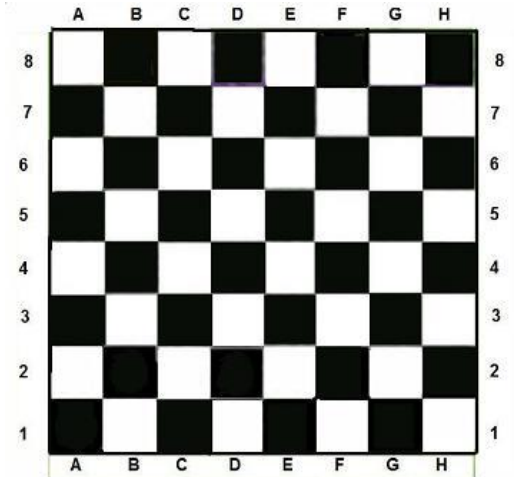
- [Анимации + псевдокод на най-често използваните алгоритми](#)

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[][] twoDimensionalArray;
```

```
int[][][] threeDimensionalArray;
```

```
int[][] intMatrix = new int[3][4];
```

```
float[][] floatMatrix = new float[8][2];
```

```
String[][][] stringCube = new String[5][5][5];
```

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

Многомерен масив - инициализация

Инициализираме многомерните масиви по същия начин, както и едномерните.

```
int[][] matrix = {  
    {1, 2, 3, 4}, // row 0 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},  
}
```

	0	1	2	3
0	1	2	6	3
1	9	0	7	1
2	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 = ?
```

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

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
```

	0	1	2	3
0	1	2	6	3
1	9	0	7	1
2	2	8	5	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;  
    }  
}
```

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

Задачи

Задача

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

		<i>movie</i>			
		<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>
<i>reviewer</i>	<i>0</i>	4	6	2	5
	<i>1</i>	7	9	4	8
	<i>2</i>	6	9	3	7

Представете данните по подходящ начин (използвайте двумерен масив).

Задача

Изведете на екрана данните от таблицата.

		<i>movie</i>			
		<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>
<i>reviewer</i>	<i>0</i>	4	6	2	5
	<i>1</i>	7	9	4	8
	<i>2</i>	6	9	3	7

Задача

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

		<i>movie</i>			
		<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>
<i>reviewer</i>	<i>0</i>	4	6	2	5
	<i>1</i>	7	9	4	8
	<i>2</i>	6	9	3	7

Задача

Намерете броя на оценките над 6.

		<i>movie</i>			
		<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>
<i>reviewer</i>	<i>0</i>	4	6	2	5
	<i>1</i>	7	9	4	8
	<i>2</i>	6	9	3	7

Задача

Принтирайте средната оценка за филм #3.

		<i>movie</i>			
		<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>
<i>reviewer</i>	<i>0</i>	4	6	2	5
	<i>1</i>	7	9	4	8
	<i>2</i>	6	9	3	7

Задача

Запишете всички оценки в едномерен масив и го сортирайте, използвайки някоя от разгледаните сортировки (или `quick sort` и `merge sort`, като разгледате имплементациите им в `Google`).

	<i>movie</i>			
	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>
<i>reviewer</i> 0	4	6	2	5
1	7	9	4	8
2	6	9	3	7

Домашно