

Лекция 9.1

Алгоритми за търсене и сортиране на масиви в Java

Основни теми

- Търсене на дадена стойност в масив, посредством алгоритми за последователно (линейно) и бинарно търсене.
- Сортиране на масив по метода на мехурчето
- Приложения на графични Swing компонети
- Задачи

10.1 Въведение

10.2 Алгоритми за търсене

10.2.1 Последователно (линейно) търсене

10.2.2 Бинарно търсене

10.3 Алгоритъм за сортиране

10.4 Приложение на Swing компоненти

Задачи

Литература:

Java How to Program, Sixth Edition, глава 16

10.1 Въведение

- **Търсене**

- **Зададена е стойност (или обект) . Проверява се за съвпадение с елемент на масив от същия тип както зададената стойност (или обект)**
- **Търсене на едно съвпадение- извежда се позицията на съвпадението в масива**
- **Търсене на всички съвпадения- извежда се масив с позициите на съвпадения в масива**

- **Сортиране**

- **Подреждане на елементите на масив по даден критерий за наредба в съответствие на зададени ключ(ове) за сортиране**

Примери

Алгоритми за търсене:

- Linear Search**
- Binary Search**
- Recursive Linear Search**
- Recursive Binary Search**
- Linear search of a List**
- Binary tree search**
- binarySearch method of class Collections**

Алгоритми за сортиране:

- Selection Sort**
- Insertion Sort**
- Recursive Merge Sort**
- Bubble Sort**
- Bucket Sort**
- Recursive Quicksort**
- Binary tree sort**
- sort method of class Collections**
- SortedSet collection**

Fig. 16.1 | Алгоритми за търсене и сортиране.

10.2 Алгоритми за търсене

- **Примери за търсене**
 - Търсене на телефонен номер
 - Търсене на линк до уеб сайт
 - Търсене на дума в речник, документ и пр.

10.2.1 Linear Search

- **Последователно (линейно) търсене**
 - **Сравняват се последователно всички елементи със зададения шаблон (**search key**) за търсене**
- **Алгоритъм**
 - **Всеки елемент на масива се сравнява със зададения шаблон, докато се намери съвпадение- извежда се позицията на съвпадение**
 - **Ако се стигне до края на масива без да е намерено съвпадение- извежда се сигнал за липса на съвпадение**

Outline

LinearArray.java

(1 of 2)

```
1 // Fig 16.2: LinearArray.java
2 // Class that contains an array of random integers and a method
3 // that will search that array sequentially
4 import java.util.Random;
5
6 public class LinearArray
7 {
8     private int[] data; // array of values
9     private static Random generator = new Random();
10
11     // create array of given size and fill with random numbers
12     public LinearArray( int size )
13     {
14         data = new int[ size ]; // create space for array
15
16         // fill array with random ints in range 10-99
17         for ( int i = 0; i < size; i++ )
18             data[ i ] = 10 + generator.nextInt( 90 );
19     } // end LinearArray constructor
20
```



Outline

LinearArray.java

(2 от 2)

```
21 // perform a linear search on the data
22 public int linearSearch( int searchKey )
23 {
24     // loop through array sequentially
25     for ( int index = 0; index < data.length; index++ )
26         if ( data[ index ] == searchKey )
27             return index; // return index of integer
28
29     return -1; // integer was not found
30 } // end method linearSearch
31
32 // method to output values in array
33 public String toString()
34 {
35     String temporary = " ";
36
37     // iterate through array
38     for ( int element : data )
39         temporary += element + " ";
40
41     temporary += "\n" ; // add newline character
42     return temporary;
43 } // end method toString
44 } // end class LinearArray
```

Обхожда се целия масив

Сравнява се последователно
всеки елемент с шаблона

Връща индекса на съвпадащия
елемент

Сигнализира за липса на
съвпадение



Outline

LinearSearchTest
.java

(1 от 2)

```
1 // Fig 16.3: LinearSearchTest.java
2 // Sequentially search an array for an item.
3 import java.util.Scanner;
4
5 public class LinearSearchTest
6 {
7     public static void main( String args[] )
8     {
9         // create Scanner object to input data
10        Scanner input = new Scanner( System.in );
11
12        int searchInt; // search key
13        int position; // location of search key in array
14
15        // create array and output it
16        LinearArray searchArray = new LinearArray( 10 );
17        System.out.println( searchArray ); // print array
18
19        // get input from user
20        System.out.print(
21            "Please enter an integer value (-1 to quit): " );
22        searchInt = input.nextInt(); // read first int from user
23
24        // repeatedly input an integer; -1 terminates the program
25        while ( searchInt != -1 )
26        {
27            // perform linear search
28            position = searchArray.linearSearch( searchInt );
29
```



Outline

LinearSearchTest.java

(2 от 2)

```
30     if ( position == -1 ) // integer was not found
31         System.out.println( "The integer " + searchInt +
32             " was not found.\n" );
33     else // integer was found
34         System.out.println( "The integer " + searchInt +
35             " was found in position " + position + ".\n" );
36
37     // get input from user
38     System.out.print(
39         "Please enter an integer value (-1 to quit): " );
40     searchInt = input.nextInt(); // read next int from user
41 } // end while
42 } // end main
43 } // end class LinearSearchTest
```

16 35 68 10 48 36 81 60 84 21

Please enter an integer value (-1 to quit): 48
The integer 48 was found in position 4.

Please enter an integer value (-1 to quit): 60
The integer 60 was found in position 7.

Please enter an integer value (-1 to quit): 33
The integer 33 was not found.

Please enter an integer value (-1 to quit): -1



10.2.2 Бинарно търсене

- **Бинарно търсене**
 - По- бързо в сравнение с последователното търсене
 - Изисква елементите на масива да са предварително сортирани
 - Започва със сравнение на елемента от масива, който е в средата
 - Ако средния елемент на масива съвпада с шаблона- то извеждаме позицията в масива, където е настъпило съвпадението
 - В протевен случай определяме дали шаблона е по- малък или по- голям от средния елемент и продължаваме да търсим по същия начин съответно в половината с по- малките или по- големите елементи от средния
 - На всяка итерация се елеминира половина от оставащите елементи за сравнение

Outline

BinaryArray.java

(1 of 3)

```
1 // Fig 16.4: BinaryArray.java
2 // Class that contains an array of random integers and a method
3 // that uses binary search to find an integer.
4 import java.util.Random;
5 import java.util.Arrays;
6
7 public class BinaryArray
8 {
9     private int[] data; // array of values
10    private static Random generator = new Random();
11
12    // create array of given size and fill with random integers
13    public BinaryArray( int size )
14    {
15        data = new int[ size ]; // create space for array
16
17        // fill array with random ints in range 10-99
18        for ( int i = 0; i < size; i++ )
19            data[ i ] = 10 + generator.nextInt( 90 );
20
21        Arrays.sort( data );
22    } // end BinaryArray constructor
23
```



Пресмята лява, дясна и средна позиция от оставащия масив за сравнение

BinaryArray.java

(2 от 3)

Цикли, докато се получи съвпадение или няма повече елементи за сравнение

Ако сравнявания елемент съвпада със средния

Връща индекса на средата

Ако шаблонът е по- малък от средния елемент

Търсим наляво от средата

Иначе, търсим надясно от средата

```

24 // perform a binary search on the data
25 public int binarySearch( int searchElement )
26 {
27     int low = 0; // low end of the search area
28     int high = data.length - 1; // high end of the search area
29     int middle = ( low + high + 1 ) / 2; // middle element
30     int location = -1; // return value; -1 if not found
31
32     do // loop to search for element
33     {
34         // print remaining elements of array
35         System.out.print( remainingElements( low, high ) );
36
37         // output spaces for alignment
38         for ( int i = 0; i < middle; i++ )
39             System.out.print( "   " );
40         System.out.println( " * " ); // indicate current middle
41
42         // if the element is found at the middle
43         if ( searchElement == data[ middle ] )
44             location = middle; // location is the current middle
45
46         // middle element is too high
47         else if ( searchElement < data[ middle ] )
48             high = middle - 1; // eliminate the higher half
49         else // middle element is too low
50             low = middle + 1; // eliminate the lower half
51

```



Намираме новата среда

BinaryArray.java

(3 от 3)

Връщаме позицията на
съвпадение

locaton остава **-1** ако не е
намерено съвпадение

```

52     middle = ( low + high + 1 ) / 2; // recalculate the middle
53 } while ( ( low <= high ) && ( location == -1 ) )
54
55 return location; // return location of search key
56 } // end method binarySearch
57
58 // method to output certain values in array
59 public String remainingElements( int low, int high )
60 {
61     String temporary = " ";
62
63     // output spaces for alignment
64     for ( int i = 0; i < low; i++ )
65         temporary += " ";
66
67     // output elements left in array
68     for ( int i = low; i <= high; i++ )
69         temporary += data[ i ] + " ";
70
71     temporary += "\n" ;
72     return temporary;
73 } // end method remainingElements
74
75 // method to output values in array
76 public String toString()
77 {
78     return remainingElements( 0, data.length - 1 );
79 } // end method toString
80 } // end class BinaryArray

```



Outline

BinarySearchTest.java

(1 от 3)

```
1 // Fig 16.5: BinarySearchTest.java
2 // Use binary search to locate an item in an array.
3 import java.util.Scanner;
4
5 public class BinarySearchTest
6 {
7     public static void main( String args[] )
8     {
9         // create Scanner object to input data
10        Scanner input = new Scanner( System.in );
11
12        int searchInt; // search key
13        int position; // location of search key in array
14
15        // create array and output it
16        BinaryArray searchArray = new BinaryArray( 15 );
17        System.out.println( searchArray );
18
```



Outline

BinarySearchTest.java

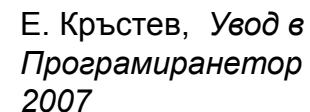
(2 от 3)

```
19 // get input from user
20 System.out.print(
21     "Please enter an integer value (-1 to quit): " );
22 searchInt = input.nextInt(); // read an int from user
23 System.out.println();
24
25 // repeatedly input an integer; -1 terminates the program
26 while ( searchInt != -1 )
27 {
28     // use binary search to try to find integer
29     position = searchArray.binarySearch( searchInt );
30
31     // return value of -1 indicates integer was not found
32     if ( position == -1 )
33         System.out.println( "The integer " + searchInt +
34             " was not found.\n" );
35     else
36         System.out.println( "The integer " + searchInt +
37             " was found in position " + position + ".\n" );
38
39     // get input from user
40     System.out.print(
41         "Please enter an integer value (-1 to quit): " );
42     searchInt = input.nextInt(); // read an int from user
43     System.out.println();
44 } // end while
45 } // end main
46 } // end class BinarySearchTest
```



```
Please enter an integer value (-1 to quit): -1
```

(3 OT 3)



10.3 Сортиране

- Сортиране на масив
 - Често срещано изискване
 - **Bubble sort** “*метод на мехурчето*”
 - По- малките стойности “изплуват” в началото на масива
 - По- големите стойности “потъват” в дъното на масива
 - Използват се вложени цикли за изпълнение на няколко паса по елементите на масива
 - Всеки пас сравнява последователно всички двойки от елементи
 - Двойките елементи не се разменят ако са в изискваната наредба (нарастваща или намаляваща големина) или равни
 - Двойките елементи се разменят ако не са в изискваната наредба (нарастваща или намаляваща големина)



BubbleSort.java

Ред 19

Декларираме **10-int array** с инициализиращ списък

Ред 27

Предаваме **array** на метод **bubbleSort** за сортирането му

```

1  // Fig. 7.11: BubbleSort.java
2  // Sort an array's values into ascending order.
3
4
5
6
7
8
9
10 public class BubbleSort{
11
12     // initialize algorithm
13     public static void main(String[] args)
14     {
15
16
17         BubbleSort app = new BubbleSort();
18
19         int array[] = { 2, 6, 4, 8, 10, 12, 89, 68, 45, 37 };
20
21         String output = "Data items in original order\n";
22
23         // append original array values to String output
24         for ( int counter = 0; counter < array.length; counter++ )
25             output += "    " + array[ counter ];
26
27         app.bubbleSort( array ); // sort array
28
29         output += "\n\nData items in ascending order\n";
30
31         // append sorted\ array values to String output
32         for ( int counter = 0; counter < array.length; counter++ )
33             output += "    " + array[ counter ];
34

```

Декларираме **10-int array** с инициализиращ списък

Предаваме **array** на метод **bubbleSort** за сортирането му





```
35 System.out.println( output );
```

```
36 }
```

Метод **bubbleSort** взима **array** като аргумент

```
38 // sort elements of array with bubble sort
```

```
39 public void bubbleSort( int array2[] )
```

```
40 {
```

```
41 // loop to control number of passes
```

```
42 for ( int pass = 1; pass < array2.length; pass++ ) {
```

```
44 // loop to control number of comparisons
```

```
45 for ( int element = 0;
46       element < array2.length - 1;
47       element++ ) {
```

Използва вложени цикли за пасове по масива **array**

```
49 // compare side-by-side elements and swap them if
```

```
50 // first element is greater than second element
```

```
51 if ( array2[ element ] > array2[ element + 1 ] )
```

```
52 swap( array2, element, element + 1 );
```

```
54 } // end loop to control comparisons
```

```
56 } // end loop to control passes
```

Ако двойката елементи е в нарастващ ред, извикай метод **swap** за размяната им

```
58 } // end method bubbleSort
```

```
60 // swap two elements of an array
```

```
61 public void swap( int array3[], int first, int second )
```

```
62 {
```

```
63 int hold; // temporary holding area for swap
```

```
65 hold = array3[ first ];
```

```
66 array3[ first ] = array3[ second ];
```

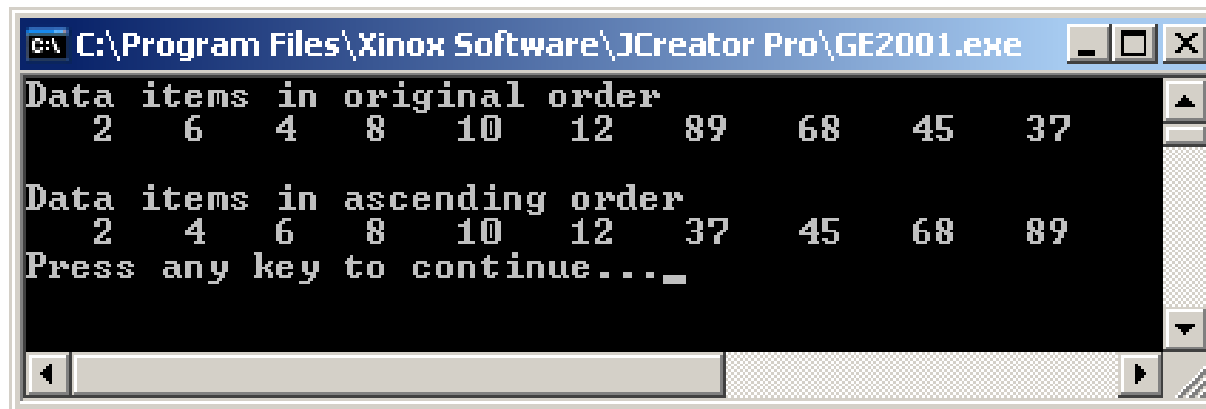
```
67 array3[ second ] = hold;
```

```
68 }
```

Метод **swap** разменя два елемента на масива **array**



```
69  
70 } // end class BubbleSort
```



C:\Program Files\Xinox Software\JCreator Pro\GE2001.exe

Data items in original order
2 6 4 8 10 12 89 68 45 37

Data items in ascending order
2 4 6 8 10 12 37 45 68 89

Press any key to continue..._



10.4 Приложение на Swing компоненти

- Сортиране на масив- извеждане на резултата

- Многоредово текстово поле ***JTextArea***

- Принадлежи на библиотека ***Swing***

- import javax.swing.*;***

- Деклариране и създаване на ***JTextArea***

- JTextArea jtxtInput = new JTextArea();***

- Методи за писане и четене в ***JTextArea***

- void setText(String txt);***

- String getText();***

- Пример: *JTextArea* в диалогов прозорец**



BubbleSort.java

Ред 19
Декларираме **10-int**
array с
инициализиращ
списък

Ред 27
Предаваме **array** на
метод **bubbleSort** за
сортирането му

```

1  // Fig. 7.11a: BubbleSort.java
2  // Sort an array's values into ascending order.
3  // Display results in a Multi line textbox JTextArea
4  // Display the textbox JTextArea in dialog box
5  // import Swing library
6  import javax.swing.*;
7
8
9
10 public class BubbleSort{
11
12     // initialize algorithm
13     public static void main(String[] args)
14     {
15         // Declare a multiline textbox
16         JTextArea outputArea = new JTextArea();
17         BubbleSort app = new BubbleSort();
18
19         int array[] = { 2, 6, 4, 8, 10, 12, 89, 68, 45, 37 };
20
21         String output = "Data items in original order\n";
22
23         // append original array values to String output
24         for ( int counter = 0; counter < array.length; counter++ )
25             output += "    " + array[ counter ];
26
27         app.bubbleSort( array ); // sort array
28
29         output += "\n\nData items in ascending order\n";
30
31         // append sorted\ array values to String output
32         for ( int counter = 0; counter < array.length; counter++ )
33             output += "    " + array[ counter ];
34

```

Декларира и създава
JTextArea

Предаваме **array** на метод
bubbleSort за сортирането му




```

35 outputArea.setText( output );
36 JOptionPane.showMessageDialog( null, outputArea,
37     "Initializing an Array of int Values",
38     JOptionPane.INFORMATION_MESSAGE );
39 System.exit(0);
36 }
37
38 // sort elements of array with bubble sort
39 public void bubbleSort( int array2[] )
40 {
41     // loop to control number of passes
42     for ( int pass = 1; pass < array2.length; pass++ ) {
43
44         // loop to control number of comparisons
45         for ( int element = 0;
46             element < array2.length - 1;
47             element++ ) {
48
49             // compare side-by-side elements and swap them if
50             // first element is greater than second element
51             if ( array2[ element ] > array2[
52                 element + 1 ] )
53                 swap( array2, element, element + 1 );
54         } // end loop to control comparisons
55
56     } // end loop to control passes
57
58 } // end method bubbleSort
59
60 // swap two elements of an array

```

Пише String в JTextArea

Метод **showMessageDialog**
 ВМЪКВА **outputArea**

Метод **bubbleSort** взима
array като аргумент

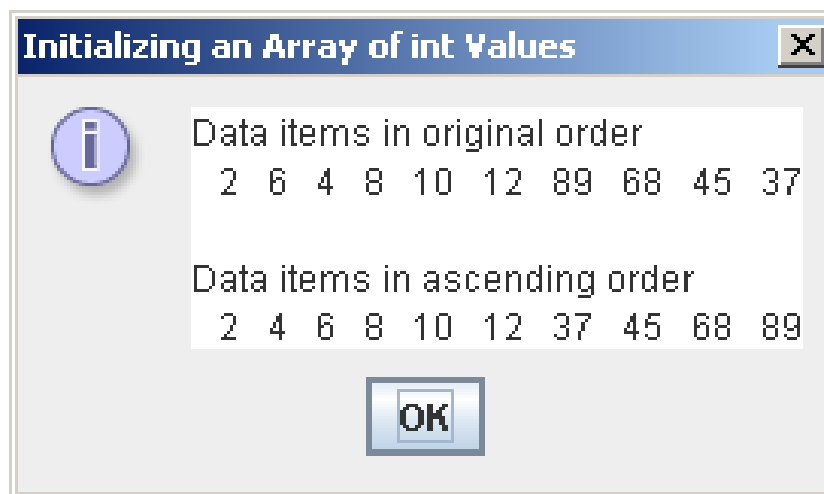
Исползва вложени цикли за
 пасове по масива **array**

Ако двойката елементи е в нарастващ ред,
 извикай метод **swap** за размяната им



```
61 public void swap( int array3[], int first, int second )
62 {
63     int hold; // temporary holding area for swap
64
65     hold = array3[ first ];
66     array3[ first ] = array3[ second ];
67     array3[ second ] = hold;
68 }
69
70 } // end class BubbleSort
```

Метод **swap** разменя два
елемента на масива **array**





BubbleSort.java

Ред 19
Декларираме **10-int**
array с
инициализиращ
списък

Ред 27
Предаваме **array** на
метод **bubbleSort** за
сортирането му

```

1  // Fig. 7.11b: BubbleSort.java
2  // Sort an array's values into ascending order.
3  // Display results in a Multi line textbox JTextArea
4  // Display the textbox JTextArea in a window
5  // import Swing library
6  import javax.swing.*;
7
8
9
10 public class BubbleSortTest{
11
12     // initialize algorithm
13     public static void main(String[] args)
14     {
15         // Declare a multiline textbox
16         JTextArea outputArea = new JTextArea();
17         JFrame app          = new JFrame ("BubbleSort Method");
18
19         int array[] = { 2, 6, 4, 8, 10, 12, 89, 68, 45, 37 };
20
21         String output = "Data items in original order\n";
22
23         // append original array values to String output
24         for ( int counter = 0; counter < array.length; counter++ )
25             output += "    " + array[ counter ];
26
27         app.bubbleSort( array ); // sort array
28
29         output += "\n\nData items in ascending order\n";
30
31         // append sorted\ array values to String output
32         for ( int counter = 0; counter < array.length; counter++ )
33             output += "    " + array[ counter ];
34

```

Декларира и създава
JTextArea

Предаваме **array** на метод
bubbleSort за сортирането му



```

35  outputArea.setText( output );
36  app.add(outputArea);
37  app.setSize(250, 150);
38  app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
39  app.setVisible(true);
36  }
37 } // end class BubbleSortTest
38 public class BubbleSort{
39     public void bubbleSort( int array2[] )
40     {
41         // loop to control number of passes
42         for ( int pass = 1; pass < array2.length; pass++ ) {
43
44             // loop to control number of comparisons
45             for ( int element = 0;
46                 element < array2.length - 1;
47                 element++ ) {
48
49                 // compare side-by-side elements and swap them if
50                 // first element is greater than second element
51                 if ( array2[ element ] > array2[
52                     element + 1 ] )
53                     swap( array2, element, element + 1 );
54             } // end loop to control comparisons
55
56         } // end loop to control passes
57
58     } // end method bubbleSort
59
60     // swap two elements of an array

```

Пише String в JTextArea

Метод add() вмъква outputArea в прозореца app

Метод bubbleSort взима array като аргумент

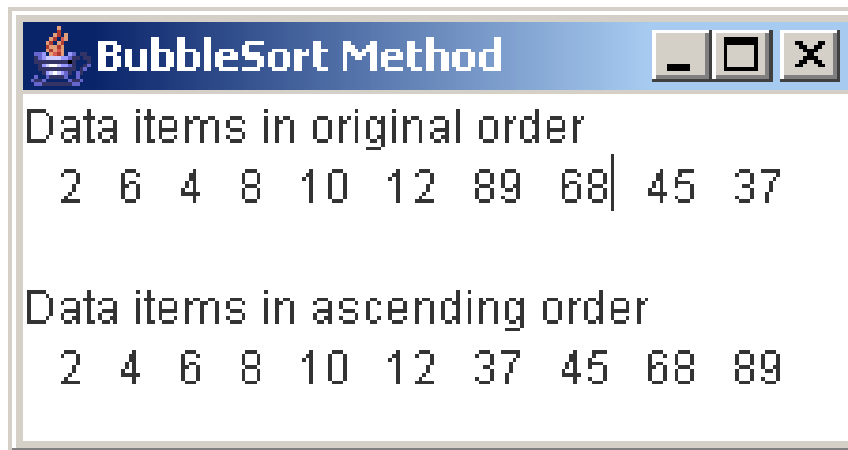
Използва вложени цикли за пасове по масива array

Ако двойката елементи е в нарастващ ред, извикай метод swap за размяната им



```
61 public void swap( int array3[], int first, int second )
62 {
63     int hold; // temporary holding area for swap
64
65     hold = array3[ first ];
66     array3[ first ] = array3[ second ];
67     array3[ second ] = hold;
68 }
69
70 } // end class BubbleSort
```

Метод **swap** разменя два
елемента на масива **array**



10.4 Приложение на Swing КОМПОНЕНТИ

- Търсене в масив- извеждане на резултата
 - Използване на етикети за текстово поле *JLabel*
 - Едноредово текстово поле *JTextField*
 - Принадлежат на библиотека *Swing*

```
import javax.swing.*;
```

- Деклариране и създаване

```
JLabel lblInput = new JLabel("strLabelText");  
JTextField      = new JTextField(intTextFieldSize);
```

- Методи за писане и четене

```
void setText(String txt);  
String getText();
```

Пример: *JTextArea* в диалогов прозорец

```
1 // Fig 07.12: LinearArray.java
2 // Class that contains an array of random integers and a method
3 // that will search that array sequentially
4 import java.util.Random;
5
6 public class LinearArray
7 {
8     private int[] data; // array of values
9     private static Random generator = new Random();
10
11     // create array of given size and fill with random numbers
12     public LinearArray( int size )
13     {
14         data = new int[ size ]; // create space for array
15
16         // fill array with random ints in range 10-99
17         for ( int i = 0; i < size; i++ )
18             data[ i ] = 10 + generator.nextInt( 90 );
19     } // end LinearArray constructor
20
```

LinearArray.java

(1 от 2)

Същият от
преди

```
21 // perform a linear search on the data
22 public int linearSearch( int searchKey )
23 {
24     // loop through array sequentially
25     for ( int index = 0; index < data.length; index++ )
26         if ( data[ index ] == searchKey )
27             return index; // return index of integer
28
29     return -1; // integer was not found
30 } // end method linearSearch
31
32 // method to output values in array
33 public String toString()
34 {
35     String temporary = " ";
36
37     // iterate through array
38     for ( int element : data )
39         temporary += element + " ";
40
41     temporary += "\n" ; // add newline character
42     return temporary;
43 } // end method toString
44 } // end class LinearArray
```

Обхожда се целия масив

Сравнява се последователно
всеки елемент с шаблона

Връща индекса на съвпадащия
елемент

Сигнализира за липса на
съвпадение

LinearArray.java
(2 от 2)

Същият от
преди

LinearArrayTest.java (1 от 2)

```

1 // Fig 07.12: LinearArrayTest.java
2 // import packages
3 import java.awt.*;
3 import java.awt.event.*;
4 import javax.swing.*;

5 public class LinearArrayTest extends JFrame implements ActionListener
6 {
7     JLabel      enterLabel, resultLabel, infoLabel;
8     JTextField  enterField, resultField, infoText;
9     LinearArray  searchArray ;

10    public static void main( String args[] )
11    {
12        LinearArrayTest app = new LinearArrayTest();
13        app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
14        app.setSize(300, 120);
15        app.setVisible(true);
16    } //end main()
17

```

extends и implements ни казват
съответно, че LinearArrayTest е
JFrame прозорец и ще слуша за

събития

Дефинира етикети и текстови
полета като клас данни

Декларира референция към обект
с метод за последователно
търсене- има клас обхват на
достъп

Декларира и създава прозорец
от тип JFrame

Показва прозореца LinearArrayTest

```

18 public LinearArrayTest() // LinearArrayTest constructor
19 {
20     super("Linear Search");
21     // create a LinearSearch object with an array of 10 elements
22     searchArray = new LinearArray( 10 );
10     setLayout(new FlowLayout());
11
12     infoLabel = new JLabel("The array:" ); //info label
13     add( infoLabel ); // add label to window
14
15     infoText = new JTextField( 18 );
16     infoText.setEditable( false );
17     // print array
18     add( infoText );
19     infoText.setText(searchArray.toString())
20
21     enterLabel = new JLabel( "Enter integer search key" );
22     add( enterLabel );
23
24     enterField = new JTextField( 10 );
25     add( enterField );
26     // register this textfield with an action listener
27     enterField.addActionListener( this );
28
29     // set up JLabel and JTextField for displaying results
30     resultLabel = new JLabel( "Result" );
31     add( resultLabel );
32
33     resultField = new JTextField( 20 );
34     resultField.setEditable( false );
35     add( resultField );
36 } // end LinearArrayTest constructor

```

Създава обект с метод за последователно търсене

LinearArrayTest.java
(2 от 3)

Създава етикет и го добавя към прозореца

Създава текстово поле и го добавя към прозореца

Пише в текстовото поле infoText произволно генерираните елементи на масив, клас данна в searchArray

Конструкторът на класа създава и инициализира основните графични компоненти и ги добавя към прозореца

Позволява да се реагира на Return в текстовото поле enterField, какво става при Return се задава в метод actionPerformed() – ВИЖ

Следващия сайд



```
36 public void actionPerformed( ActionEvent actionEvent )
37 {
38     int searchInt; // search key
39     int element; // location of search key in array
40
41
42     // input also can be obtained with actionEvent.getActionCommand()
43     String searchKey = enterField.getText();
44     searchInt = Integer.parseInt(searchKey);
45
46
47     // perform linear search
48     element = searchArray.linearSearch( searchInt );
49     // display search result
50     if ( element != -1 )
51         resultField.setText( "Found value in element " + (element + 1) );
52     else
53         resultField.setText( "Value not found!" );
54 } // end actionPerformed()
55 } // end class LinearArrayTestr
```

Задава действието,
което да се изпълни при
натискане на Return

Прочита въведения текст в
текстовото поле

Изпълнява поделователно
търсене

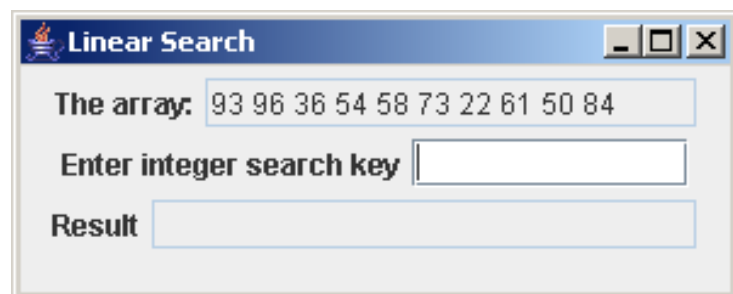
Извежда крайния резултат в
текстовото поле

resultField

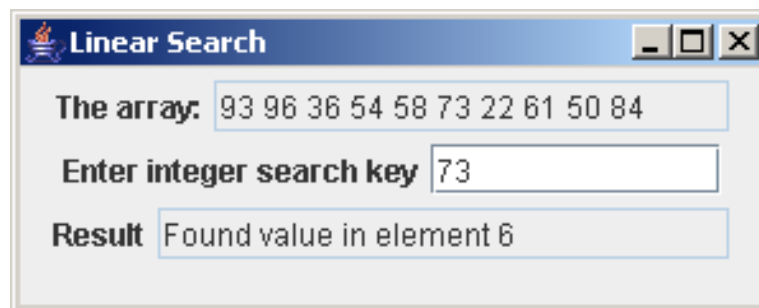
LinearArrayTest.java

(3 от 3)

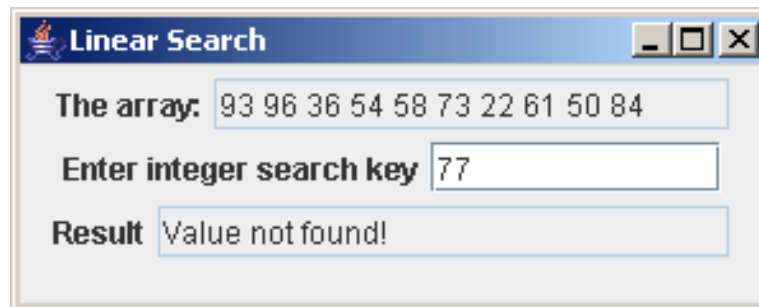
Примери



A screenshot of a Java Swing window titled "Linear Search". It contains three text fields: "The array:" with the value "93 96 36 54 58 73 22 61 50 84", "Enter integer search key" which is empty, and "Result" which is also empty.



A screenshot of a Java Swing window titled "Linear Search". It contains three text fields: "The array:" with the value "93 96 36 54 58 73 22 61 50 84", "Enter integer search key" with the value "73", and "Result" with the value "Found value in element 6".



A screenshot of a Java Swing window titled "Linear Search". It contains three text fields: "The array:" with the value "93 96 36 54 58 73 22 61 50 84", "Enter integer search key" with the value "77", and "Result" with the value "Value not found!".

Задачи

1. Методът за сортиране по метода на мехурчето, показан на Fig. 7.11 не е ефективен за големи масиви. **Направете следните промени** за да подобрите бързодействието на този алгоритъм.
 - След първия пас е сигурно, че най- голямото число се намира в края на масива, след втория пас двете най- големи числа са *“на местата си”* и т.н. Така вместо да се правят 9 сравнения на всеки пас за сортиране на 10 числа, променете този алгоритъм да прави 8 сравнения на втория пас, 7 сравнения на третия пас и т.н.
 - Напишете кода на програмата, така че изпълнението ѝ да не зависи от дължината на масива, който се подава за сортиране
 - Ако масиваът е сортиран при задаването си, нужно ли е да се правят 9 паса за сортиране на 10 числа или по- малко паса ще свършат същата работа? Променете кода за сортиране да проверява в края на всеки пас дали са правени размени на елементи. Ако не са правени размени, то масива и вече нареден и програмата може да приключи. В противен случай се преминава към следващия пас.

Задачи

2. Програмната реализация на методите за търсене, представени в *class LinearArray* на Fig. 16. 2 (*последователно търсене*) и в *class BinaryArray* на Fig. 16. 4 (*бинарно търсене*) позволяват да се открие точно едно съвпадение на зададения шаблон с елемент от масива. Тази програмна реализация не позволява да се открият всички елементи от масива, които съвпадат с дадения шаблон.
 - Добавете метод


```
public String linearSearchAll( int searchKey )
```

 в *class LinearArray* на Fig. 07. 12, който да прилага *метода на последователно търсене* за съвпадение с `searchKey` и да връща `String` с всички индекси на елементи от клас данната `int[] data` , които съвпадат с `searchKey` (индексите да са разделени със празен символ). *Преценете каква стойност да връща метода при липса на съвпадение*

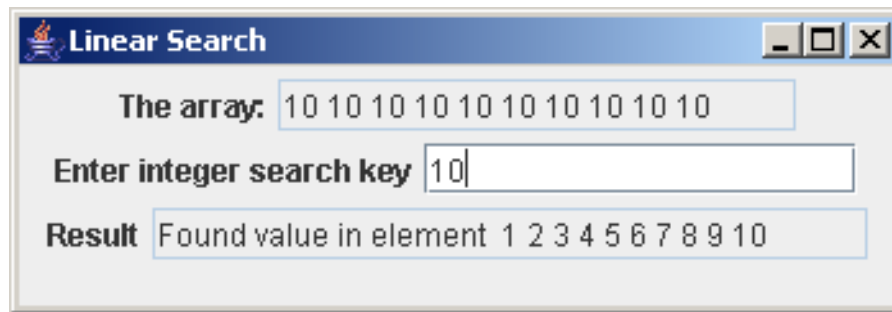
Задачи

- Променете метод

`public void actionPerformed(ActionEvent event)`
 в *class LinearArrayTest* на **Fig. 07. 12**, така че да тествате
 новосъздадения метод *linearSearchAll()* по аналогия с
 тестването на метод *linearSearch()*

Изпълнете и тествайте приложението с различни данни

(предварително инициализирайте масива *data* в *class LinearArray* с инициализиращ списък, съдържащ дублиращи се стойности)



Задачи

- По аналогичен начин, добавете метод
`public String binarySearchAll(int searchKey)`
 в `class BinaryArray` на **Fig16.04**, който да прилага *метода на бинарно търсене* за съвпадение с `searchKey` и да връща `String` с всички индекси на елементи от клас данната `int[] data`, които съвпадат с `searchKey` (индексите да са разделени със празен символ). *Преценете каква стойност да връща метода при липса на съвпадение*
[Упътване: Използвайте, че вече написания метод `binarySearch()`, а също и, че масива `data` е сортиран, за да получите списък с индекси на повторенията]

Задачи

- Пренапишете *BinarySearchTest.java* от Fig16.05 по аналогия с *class LinearArrayTest* на Fig. 07. 12 , за да използва същите такива етикети, текстови полета и възможност за въвеждане на шаблон за търсене, както *class LinearArrayTest*

- Променете метод

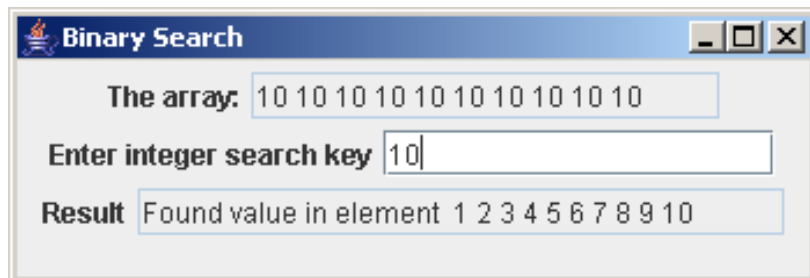
```
public void actionPerformed( ActionEvent event)
```

 в променения *class BinarySearchTest*, за да тествате новосъздадения метод *binarySearchAll()* по аналогия с тестването на метод *binarySearch()*

Изпълнете и тествайте приложението с различни данни

(предварително инициализирайте масива *data* в *class BinaryArray* с инициализиращ списък, съдържащ дублиращи се стойности)

Задачи

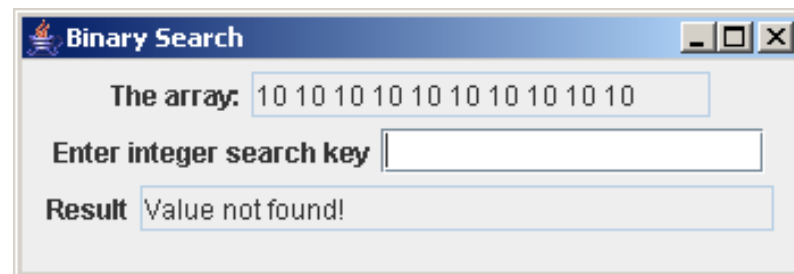


Binary Search

The array: 10 10 10 10 10 10 10 10 10 10

Enter integer search key 10

Result Found value in element 1 2 3 4 5 6 7 8 9 10

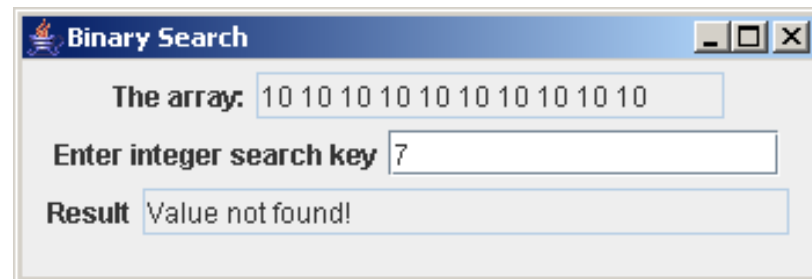


Binary Search

The array: 10 10 10 10 10 10 10 10 10 10

Enter integer search key

Result Value not found!



Binary Search

The array: 10 10 10 10 10 10 10 10 10 10

Enter integer search key 7

Result Value not found!