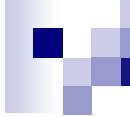


# Programming

## Recursive functions

5<sup>th</sup> lecture



# Algorithm

- What is an algorithm?
  1. It is a function that converts input to output
  2. The algorithm calculates a result for **each** data, it always stops

# Example

```
public int sestejDo0(int par) {  
    int vsota;  
    vsota = 0;  
    while (par != 0) {  
        vsota += par;  
        par--;  
    }  
    return vsota;  
} /* sestejDo0 */
```

- Is this an algorithm?



# Sum numbers to 0

```
Int sestejDo0(int par) {  
    int vsota;  
    vsota = 0;                      /* initialization to 0 */  
    while (par != 0) {                /* while par is bigger than 0  
     */  
        vsota += par;                /* add par to sum */  
        par--;                      /* decrease par */  
    }  
    return vsota;  
} /* sestejDo0 */
```

# And now for something completely different

```
Int sestejDo0(int par) {    Int sestejDo0(int par) {
    int vsota;
    vsota = 0;
    if (par <= 0)
        return 0;
    else {
        while (par != 0) {
            vsota += par;
            par--;
        }
        return vsota;
    }
} /* sestejDo0 */
```

```
Int sestejDo0(int par) {
    if (par <= 0)
        return 0;
    else
        return
            sestejDo0(par-1)+par;
} /* sestejDo0 */
```



# What the function really does

$$\text{resultat} = (\sum_{i=0..par} i) = (\sum_{i=0..par-1} i) + par$$

```
Int sestejDo0(int par) {  
    if par is smaller or  
    equalt to 0, the  
    result is 0;  
    else  
        the result is equal to  
        the sum of numbers  
        from 0 to par-1,  
        and add par  
} /* sestejDo0 */
```

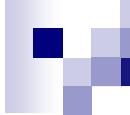
```
Int sestejDo0(int par) {  
    if (par <= 0)  
        return 0;  
    else  
        return  
            sestejDo0(par-1)  
            + par;  
} /* sestejDo0 */
```

# Recursive definition

```
Int sestejDo0(int par) {  
    if (par <= 0)  
        return 0;  
    else  
        return  
            sestejDo0(par-1)+par;  
} /* sestejDo0 */
```

```
Int sestejDo0(int par) {  
    if (par <= 0)  
        result is 0;  
    else  
        result is sum of the  
        smaller problem and par  
    } /* sestejDo0 */
```

- Stoping condition
- Step divide and conquer (divide et impera)  
**(deli in vladaj)**



# Again - algorithm

```
Int sestejDo0(int par) {  
    if (par <= 0)  
        return 0;  
    else  
        return  
            sestejDo0(par-1)+par;  
} /* sestejDo0 */
```

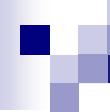
- Is this recursive function an algorithm?
- Yes, it stops for every value of `par`.
- *Proof* (induction on parameter `par`):
  - *Basis*: for each `par <= 0` from the **source code**
  - *Hypothesys*: if the statement `par = n-1` is true, than it is true also for `par = n`
  - *Step*: say, the function stops for `par-1`; then it certainly stops for `par` (from the **source code**)



# Recursion and iteration

- Each iteration can be translated into recursion

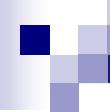
```
public static void print(int[] polje) {  
    for(int i = 0; i < polje.length; i++)  
    {  
        System.out.print(polje[i]);  
        System.out.print(" ");  
    }  
    System.out.println();  
} // print
```



# Output all elements of the array

```
public static void print(int[] polje) {  
    for(int i = 0; i < polje.length; i++)  
        System.out.print(polje[i] + " ");  
    System.out.println();  
} // print
```

```
public static void print(int[] polje, int i) {  
    if (i >= polje.length)  
        System.out.println();  
    else {  
        System.out.print(polje[i] + " ");  
        print(polje, i+1);  
    }  
} // print
```



# Output all elements of the array

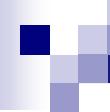
- if we come to the end of the array, print last element, else
- print the next character (the first in the current array) and the rest of the field

```
public static void print(int[] polje, int i)
{
    if (i >= polje.length)
        System.out.println();
    else {
        System.out.print(polje[i] + " ");
        print(polje, i+1);
    }
} // print
```



# Search of an element

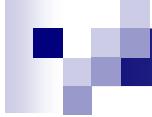
```
public static boolean  
member(int[] polje, int elt) {  
    for(int i = 0; i < polje.length; i++)  
        if (polje[i] == elt) return true;  
    return false;  
} // member  
  
public static boolean  
member(int[] polje, int elt, int i) {  
    if (polje[i] == elt)      return true;  
    if (i >= polje.length)  return false;  
    return member(polje, elt, i+1);  
} // member
```



# Search of an element

- if we find the element, return true
- if we come to the end of the array, return false
- else search the rest of the array

```
public static boolean  
member(int[] polje, int elt, int i) {  
    if (polje[i] == elt)    return true;  
    if (i >= polje.length) return false;  
    return member(polje, elt, i+1);  
} // member
```



# Search for the biggest element

- If we are at the end of the array, return the biggest element, else:
- look at whether the current element is bigger than previously found element and, if so, this element becomes the new current biggest element
- then look in the rest of the field, if there a bigger element

# Search for the biggest element

```
public static  
int maximum(int[] polje, int i, int trenMaks) {  
    if (i >= polje.length) return trenMaks;  
    if (polje[i] > trenMaks) trenMaks = polje[i];  
    return maximum(polje, i+1, trenMaks);  
} // maximum
```

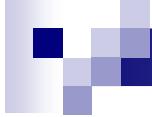
- Call like this:

what about

**maximum(polje, 0, polje[0])**

this call?

**maximum(polje, 1, polje[0])?**



# Search for the biggest element - 2.

- If this is the last item in the array, then this is the biggest element in the array, otherwise:
- find the biggest element in the remainder of the array
- look at whether the current element is bigger than the current biggest element in the remainder of the field and, if so, we claim this is the current biggest element

# Search for the biggest element - 2.

```
public static int maximum(int[] polje, int i) {  
    int maksPreost;  
    if ((i+1) == polje.length) return polje[i];  
    maksPreost = maximum(polje, i+1);  
    if (polje[i] > maksPreost) maksPreost = polje[i];  
    return maksPreost;  
} // maximum
```

- Call like this:

```
maximum(polje, 0)
```