Programming

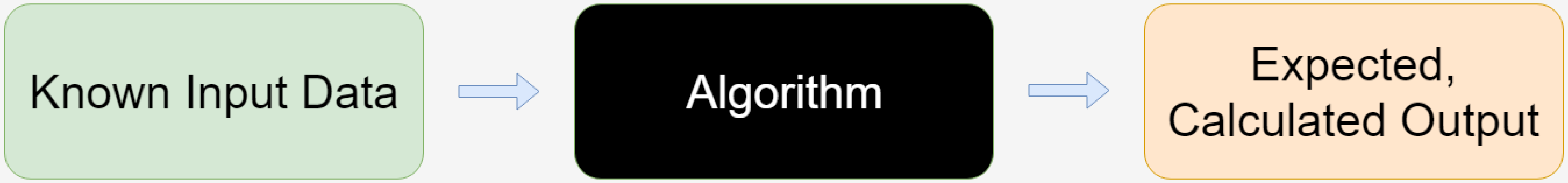
Algorithms and Their Complexity

Presentation Scheme

Presentation Scheme

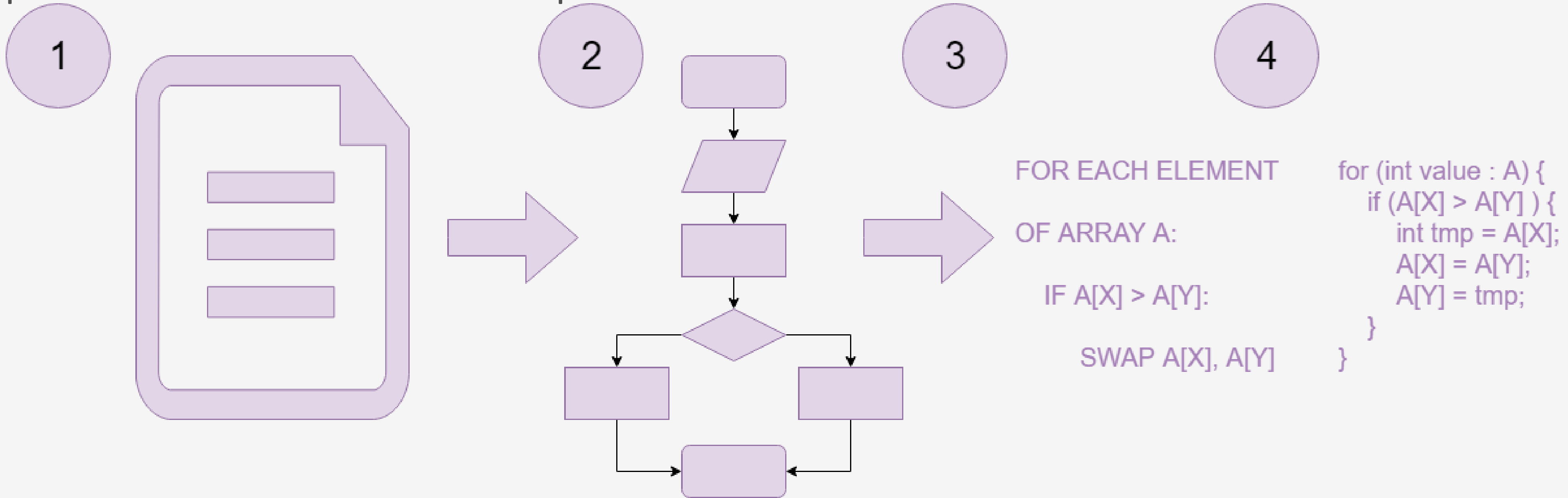
Algorithm is:

- a finite sequence of rules applied to a finite number of data, allowing you to solve similar problem classes
- a set of rules specific to certain calculations or IT operations



How to Create an Algorithm

1. Taking **input data** and **expected result** (output data) we list the sequence of actions that we need to take in order to achieve the desired effect.
2. Transform text version of the algorithm (created in pt 1) into a **block diagram**.
3. (optional) Write a **pseudocode** based on the block diagram.
4. **Implement the algorithm in the selected programming language**. Check the compliance with the assumptions.



```
FOR EACH ELEMENT

OF ARRAY A:

   IF A[X] > A[Y]:

      SWAP A[X], A[Y]
```

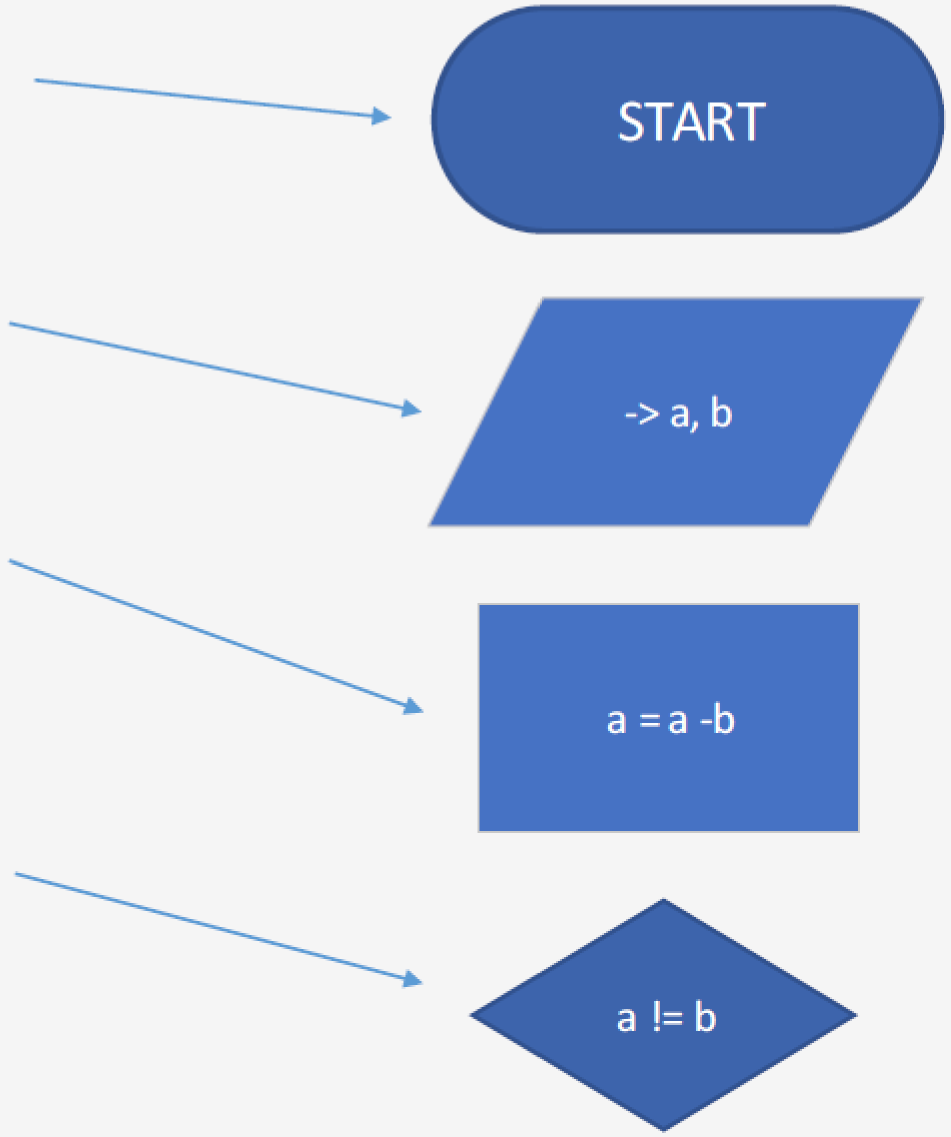
```
for (int value : A) {
   if (A[X] > A[Y] ) {
      int tmp = A[X];
      A[X] = A[Y];
      A[Y] = tmp;
   }
}
```

Benefits of Algorithms

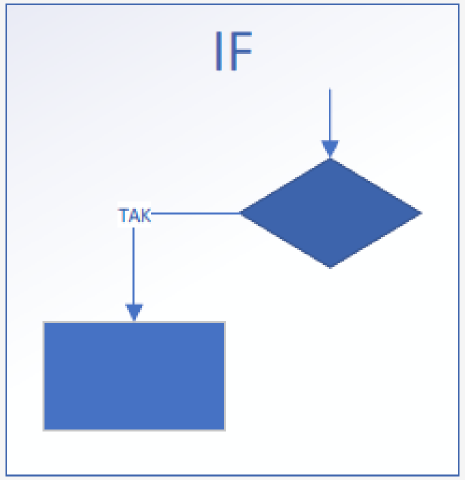
1. **Repeatability of results** - implementation in compliant with the recipe.
2. **Portability** - algorithm is not strictly associated with the programming language.
3. Specified **Computational Complexity** - comparison of algorithms in terms of performance.
4. **Optimization** - its development before the implementation of the solution allows to shorten the actual time spent on the task.

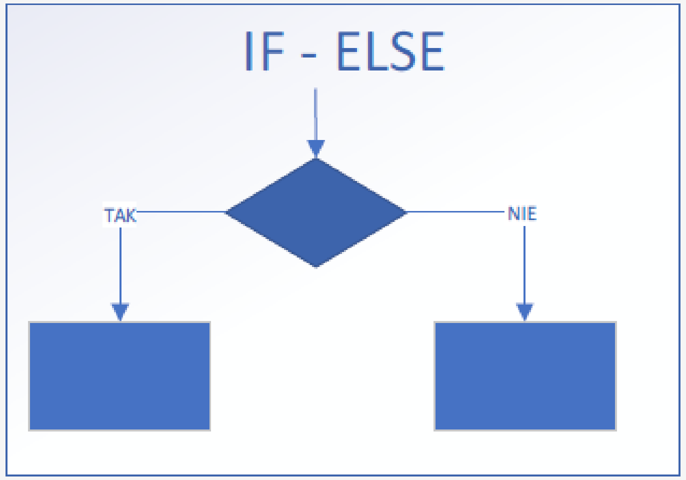
Block Diagrams

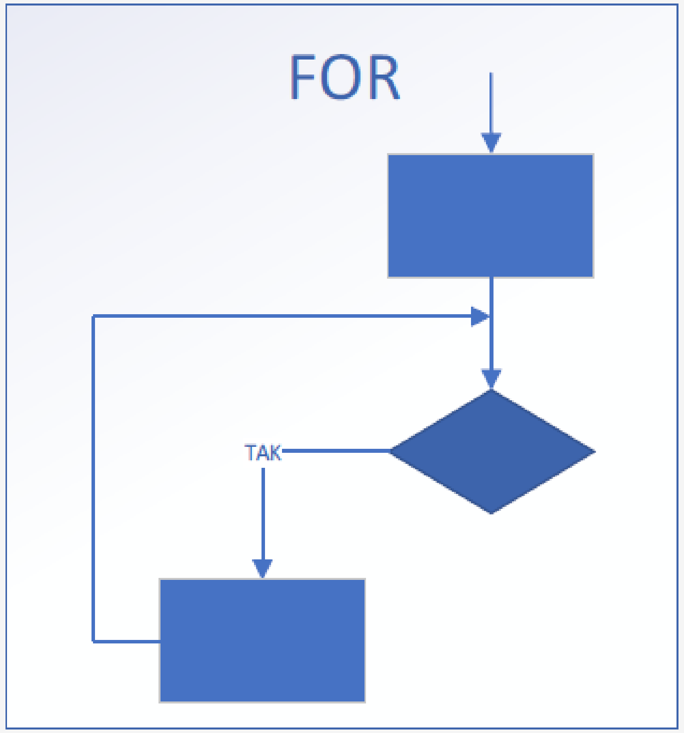
1. Start/Stop block - indicates start or end of algorithm work
2. Input/Output block - indicates taking or returning data
3. Process block - set of instructions - variable assignments, calling functions, arithmetic operations
4. Conditional block - contains a condition or a set of conditions. In case all conditions are fulfilled it continues through „Yes” branch and with „No” branch otherwise.

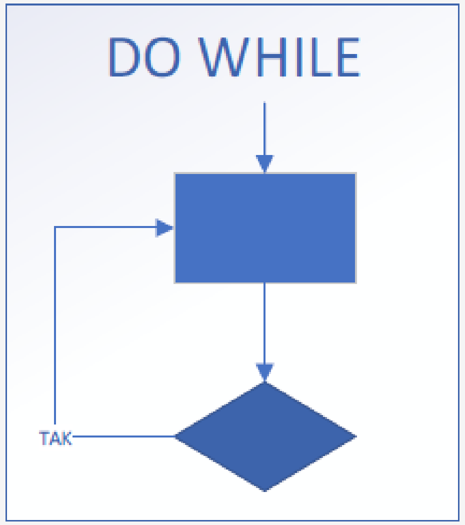


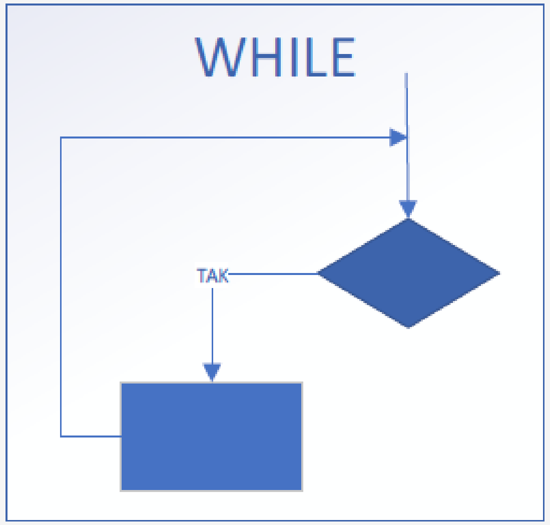
Block Diagrams - Loops and Conditions











Assignment - Greatest Common Divisor

Greatest Common Divisor (GCD) - for two integers, the largest integer which divides both numbers.

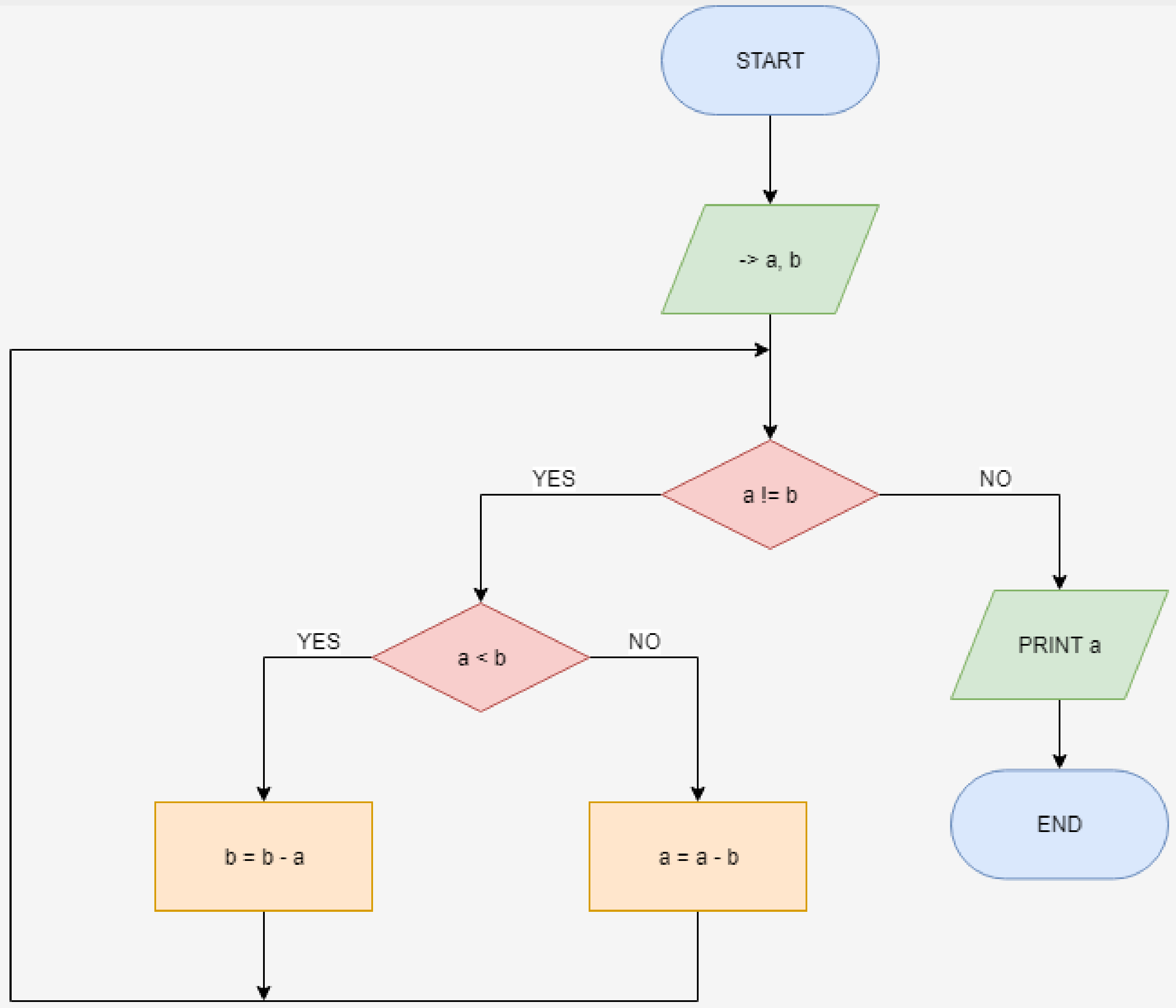
Implement the algorithm returning the greatest common divisor of two numbers using the Euclidean Algorithm.

Input: 18, 24

Output: 6

Block diagram on the next page.

Block Diagrams - Loops and Conditions



Assignment - Greatest Common Divisor

```
1. Until b ≠ 0 do 2-4
2. t <- b            // hold divisor
3. b <- a mod b      // calculate the rest from dividing which becomes the divisor
4. a <- t            // previous divisor becomes the divisive
5. print a
6. end
```

Assignment - Linear Function

Create and implement an algorithm that finds the zero place of the linear function $f(x) = ax + b$.

1. IF a != 0, the zero place is -b/a
2. IF a == 0 AND b != 0, no zero places
3. IF a == 0 AND b == 0, every number is a zero place

Prepare your own block diagram and implement the algoritm. You can use draw.io to draw your diagrams. Create and implement an algorithm that finds

Assignment - Square Function

Create and implement an algorithm that finds the zero place of the square function $f(x) = ax^2 + bx + c$.

1. IF a === 0, the function is linear
2. ELSE Google for finding zero place of the square function

Prepare your own block diagram and implement the algoritm. You can use draw.io to draw your diagrams. Create and implement an algorithm that finds

Assignment - Prime Numbers

Number n is a prime number if and only if it is divided by exactly two different numbers. 1 and n itself.

Implement an algorithm to find all prime numbers between 1 and n (n given by user).

Prepare your own block diagram and implement the algoritm. You can use draw.io to draw your diagrams. Create and implement an algorithm that finds

Assignment - Reverse Array

Create and implement an algorithm which will reverse values in given array.

Prepare your own block diagram and implement the algoritm. You can use draw.io to draw your diagrams. Create and implement an algorithm that finds

Assignment - *Second Smallest Value in an Array

Create and implement an algorithm which will find the second smallest value in an array

Prepare your own block diagram and implement the algoritm. You can use draw.io to draw your diagrams. Create and implement an algorithm that finds

Computational Complexity

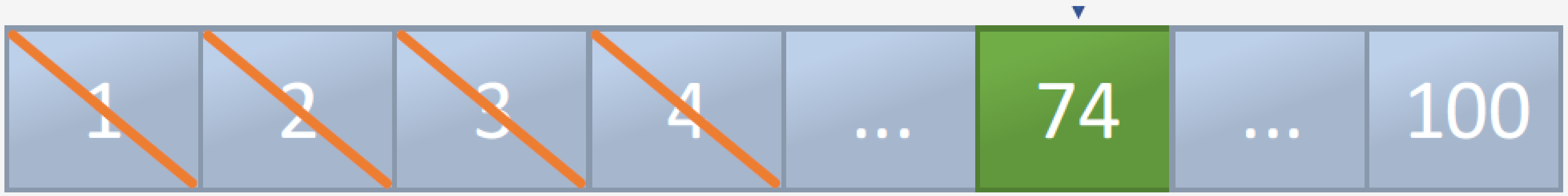
Assignment

Our goal is to achieve 16 squares on a piece of paper.

1. Draw 16 squares on a piece of paper. Each square is a single operation.
2. Fold another piece of paper to achieve 16 squares. Each fold is a single operation.
3. Compare number of operations in both cases.
4. Define, how many operations you need to perform to achieve certain amount of squares in both scenarios.

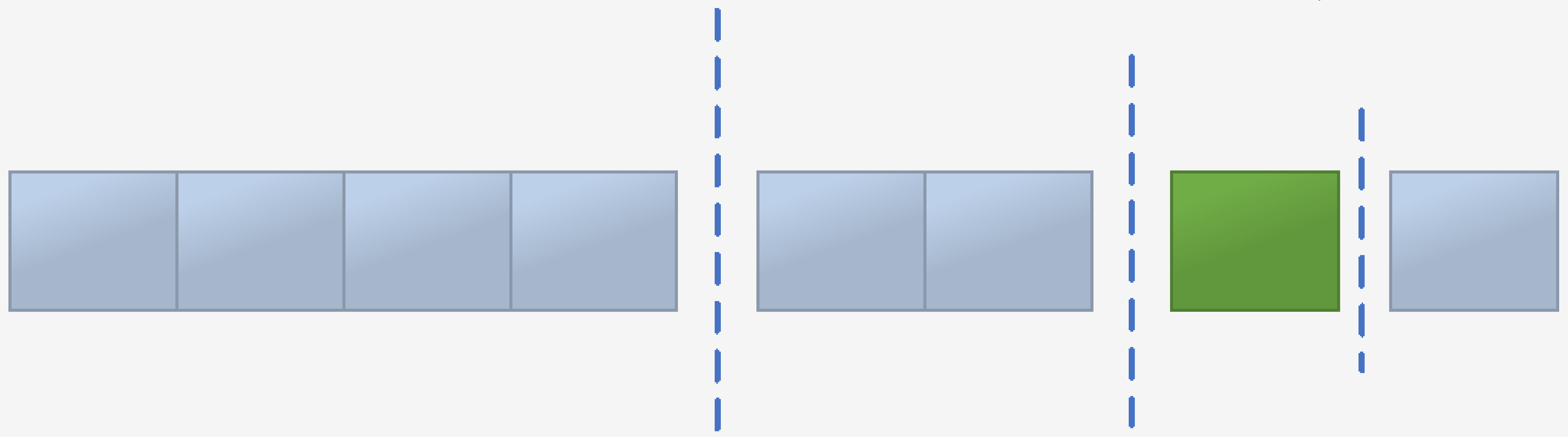
Simple Search

1. Simple search (linear, sequential) consists in searching a set of elements element by element.
2. Is the simplest way to search in a set.
3. Number of operations require from 1 up to n operations, where n is the size of the set.
4. Because in the worst-case scenario we would perform n operations, we say that the complexity is O(n).



Binary Search

1. Given sorted array of elements we divide the array by 2 and search in one of the parts.
2. To find the 74th element in an array of size 100, we only need 7 operations.
3. The complexity of this algorithm is $O(\log_2 n)$.



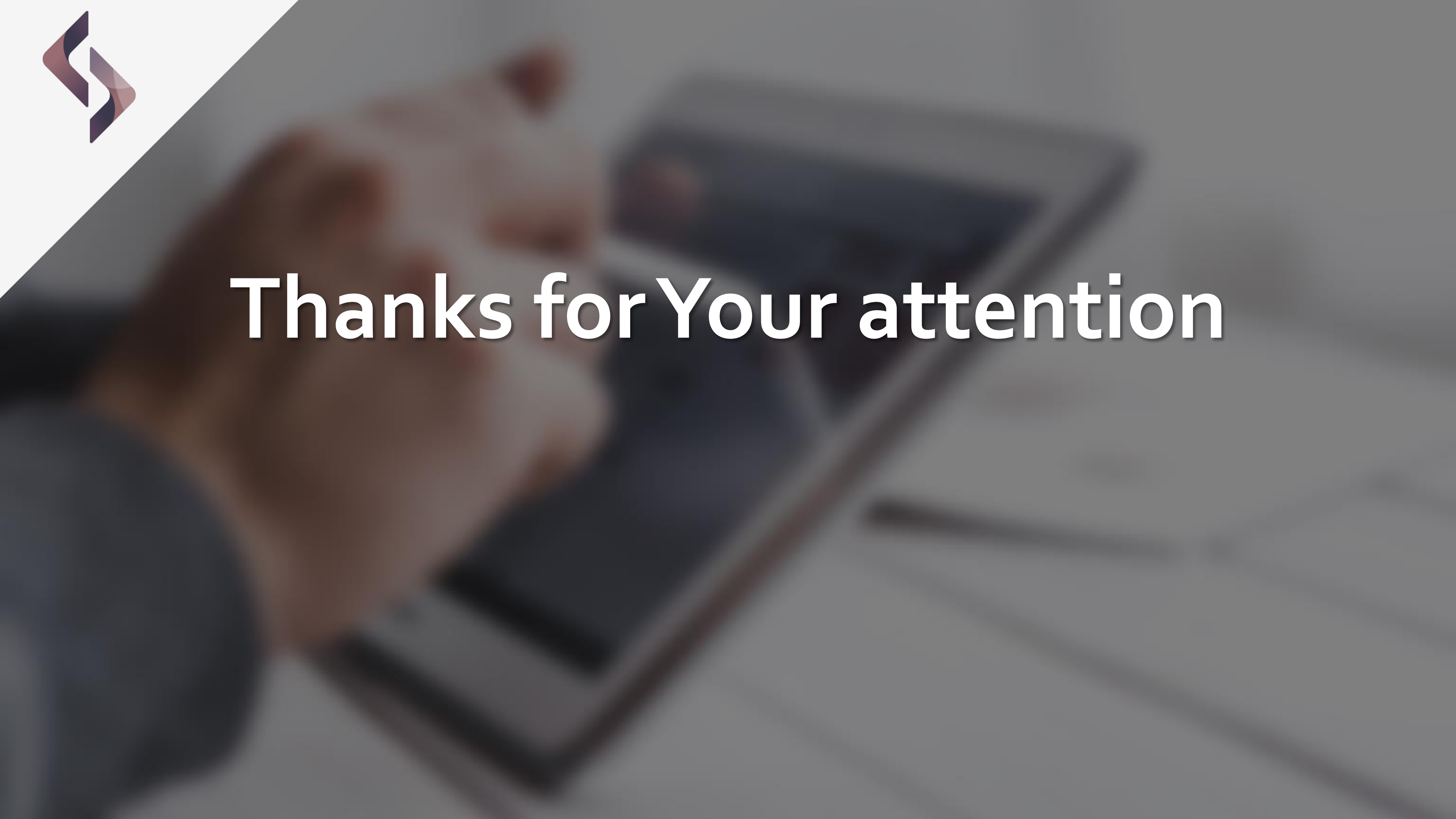
Computational Complexity

1. Computational Complexity - describes the amount of resources (time or memory) required to solve computational problems.
2. Is usually described with big-O notation.
3. Big-O notation allows us to compare the number of operations to be performed.
4. The speed of algorithms execution is not expressed in seconds, but in the rate of increase in the number of operations.
5. Common algorithm execution times:
    1. O(1) - single operation or accessing element in an array using index (arr[i])
    2. O(logn) - binary search
    3. O(n) - reading n numbers
    4. O(n*logn) - quicksort,
    5. $O(n^2)$ - bubble sort,
    6. O(n!) - Traveling salesman problem

Assignment - Simple and Binary Search

Create and implement Simple Search and Binary Search algorithms. Test them on:

1. Array of random ints,
2. Array of Strings,
3. *Array of Objects (make sure that you get familiar with Comparator vs Comparable)



Thanks for Your attention