Task: Find in rotated sorted array index of target for example 15, if number not exist return -1

I) Binary search for finding pivot, where sorted array was rotated

Create three variables: left, right and mid
left = 0, right = length of array - 1, mid = 0

Create loop, where statement to stop: left < right

Inside loop calculate mid of the current range: mid = left + (right - left) / 2

Also inside loop create statement: if mid value more than right index value set left index equal to midpoint + 1, else right = midpoint

After this steps, we have right = left = pivot, which divide array on two sorted ranges

Example:

Array: [20, 25, 10, 15, 17, 18, 19]

Target: 15

Step 0: left = 0, right = 6, mid = 0

Step 1: left = 0, right = 6, mid = 3 -> array[mid] (15) < array[right] (19) -> right = mid = 3

Step 2: left = 0, right = 3, mid = 1 -> array[mid] (25) > array[right] (15) -> left = mid + 1 = 2

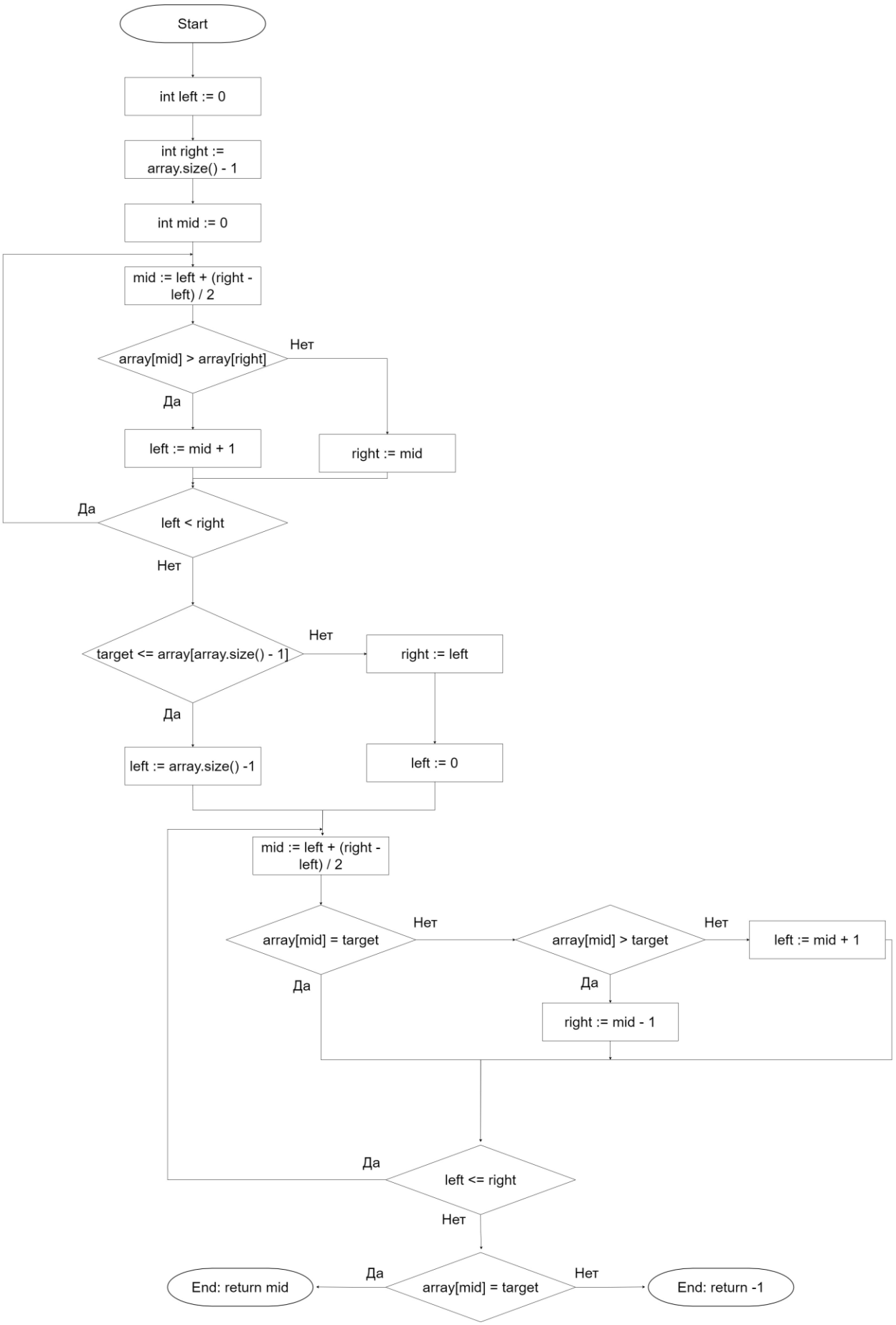
Step 3: left = 2, right = 3, mid = 2 -> array[mid] (10) < array[right] (15) -> right = mid = 2

Step 4: left = right = 2, loop end.

II) Define in which range target can be

If target smaller or equal last element, target can be located in second range (pivot, index of last element), else target can be located in first range (index of first element, pivot)

III) For defined range implement binary search


Start
int left := 0
int right := array.size() - 1
int mid := 0
mid := left + (right - left) / 2
array[mid] > array[right]
Нет
Да
left := mid + 1
right := mid
Да
left < right
Нет
target <= array[array.size() - 1]
Нет
right := left
Да
left := array.size() -1
left := 0
mid := left + (right - left) / 2
array[mid] = target
Нет
array[mid] > target
Нет
left := mid + 1
Да
Да
right := mid - 1
Да
left <= right
Нет
End: return mid
Да
array[mid] = target
Нет
End: return -1