Procedure binary\_search

A ← sorted array

n ← size of array

x ← value to be searched

Set lowerBound = 1

Set upperBound = n

while x not found

if upperBound < lowerBound

EXIT: x does not exists.

set midPoint = lowerBound + ( upperBound - lowerBound ) / 2

if A[midPoint] < x

set lowerBound = midPoint + 1

if A[midPoint] > x

set upperBound = midPoint - 1

if A[midPoint] = x

EXIT: x found at location midPoint

end while

end procedure

**Algorithm**

Algorithm is quite simple. It can be done either recursively or iteratively:

1. get the middle element;
2. if the middle element equals to the searched value, the algorithm stops;
3. otherwise, two cases are possible:
   * searched value is less, than the middle element. In this case, go to the step 1 for the part of the array, before middle element.
   * searched value is greater, than the middle element. In this case, go to the step 1 for the part of the array, after middle element.

Now we should define, when iterations should stop. First case is when searched element is found. Second one is when subarray has no elements. In this case, we can conclude, that searched value doesn't present in the array.

### C++

/\*

\* searches for a value in sorted array

\*   arr is an array to search in

\*   value is searched value

\*   left is an index of left boundary

\*   right is an index of right boundary

\* returns position of searched value, if it presents in the array

\* or -1, if it is absent

\*/

int binarySearch(int arr[], int value, int left, int right) {

      while (left <= right) {

            int middle = (left + right) / 2;

            if (arr[middle] == value)

                  return middle;

            else if (arr[middle] > value)

                  right = middle - 1;

            else

                  left = middle + 1;

      }

      return -1;

}