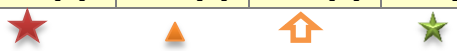
Binary Search

When searching an array, the **binary search** process utilizes the concept of splitting intervals in half as a means of finding the "**key**" value as quickly as possible.
If the array that contains your data is **in order** (ascending or descending), you can search for the **key** item much more quickly by using a **binary search algorithm** ("**divide and conquer**"). Consider the following array of integers. We will be searching for the integer 77:

Array of integers, named num, arranged in "ascending order"

13	24	34	46	52	63	77	89	91	100
num[0]	num[1]	num[2]	num[3]	num[4]	num[5]	num[6]	num[7]	num[8]	num[9]



First, find the middle of the array by adding the array subscript of the first value to the subscript of the last value and dividing by two: (0 + 9) / 2 = 4 **Integer division** is used to arrive at the **4th subscript** as the middle. (The actual mathematical middle would be between the 4th and 5th subscripts, but we must work with integer subscripts.)	The 4th subscript holds the integer 52, which comes before 77. We know that 77 will be in that portion of the array to the right of 52. We now find the middle of the right portion of the array by using the same approach. (5 + 9) / 2 = 7	The 7th subscript holds the integer 89, which comes after 77. Now find the middle of the portion of the array to the right of 52, but to the left of 89. (5 + 6) / 2 = 5	The 5th subscript holds the integer 63, which comes before 77, so we subdivide again (6 + 6) / 2 = 6 and the 6th subscript holds the integer 77.

```
public static void binarySearch(int[ ] array, int lowerbound, int upperbound, int key)
  {
     int position;
     int comparisonCount = 1;    // counting the number of comparisons (optional)

   position = ( lowerbound + upperbound) / 2;    // To start, find the subscript of the middle position.

    while((array[position] != key) && (lowerbound <= upperbound))
    {
       comparisonCount++;
       if (array[position] > key)            // If the number is > key, ..
       {                                    // decrease position by one.
           upperbound = position - 1;
       }
           else
      {
           lowerbound = position + 1;    // Else, increase position by one.
      }
     position = (lowerbound + upperbound) / 2;
    }
    if (lowerbound <= upperbound)
    {
         System.out.println("The number was found in array subscript" + position);
         System.out.println("The binary search found the number after " + comparisonCount +  "comparisons.");
    }
    else
        System.out.println("Sorry, the number is not in this array.  The binary search made " +comparisonCount  + "
comparisons.");
  }

}
```