**Linear Search**

Look at each element of array in range \([first, \ last)\) until the target is found or run out of elements. Return index of target or last.

**Analysis**

<table>
<thead>
<tr>
<th>Objects</th>
<th>Type</th>
<th>Kind</th>
<th>Movement</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>array of values</td>
<td>int[ ]</td>
<td>varying</td>
<td>received</td>
<td>arr</td>
</tr>
<tr>
<td>index of lower bound</td>
<td>int</td>
<td>varying</td>
<td>received</td>
<td>first</td>
</tr>
<tr>
<td>index of upper bound</td>
<td>int</td>
<td>varying</td>
<td>received</td>
<td>last</td>
</tr>
<tr>
<td>search target</td>
<td>int</td>
<td>varying</td>
<td>received</td>
<td>target</td>
</tr>
<tr>
<td>index of target</td>
<td>int</td>
<td>varying</td>
<td>returned</td>
<td>position</td>
</tr>
</tbody>
</table>

**Algorithm**

1. For position from first to last-1 do
   
   1.1 If \(arr[\text{position}] = \text{target}\) then
      
      1.1.1 Return \text{position}
   
2. Return last

**Code**

```cpp
// Precondition: first <= last
int LinearSearch (const int arr[], int first, int last, int target) {
    // scan indices from first <= position < last for a match
    for (int position = first; position < last; position++)
        if (arr[position] == target) // found target
            return position;
    return last; // target not found
} // end LinearSearch
```

1