algorithmic thinking

we need names for the data

- a is the array of ordered data items, indexed starting at 1:
- a[1] is the first data item, a[2] is the second data item, and so on, up to a[15]
 query
- query is the item we are searching for
- our task: given query, output the index of query in array a

more names

- first and last refer to the indices bounding the part of the array we are searching,
- middle is the index halfway between first and last
- initially, first is 1 and last is 15

binary search algorithm

input: $\begin{array}{c|c} & query \\ \hline & input: \\ \hline & first \\ \hline & middle \\ \hline & last \\ \hline & search \\ \end{array}$

set middle to be halfway between first and last

if query == a[middle] then how to do

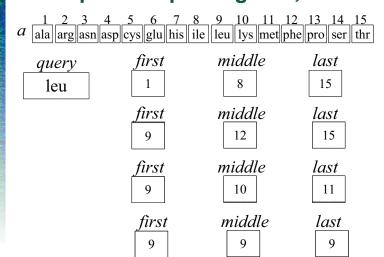
output middle

else

update first and last search in a between first and last

search

examples of updating first, last



binary search algorithm

middle = first + (last - first)/2
if query == a[middle] then
 output middle
else
 if query < a[middle] then last = middle - 1;
 if query > a[middle] then first = middle + 1;
 search in a between first and last

binary search algorithm

```
a la arg asn asp cys glu his ile leu lys met phe pro ser thr

input: query

output: index of query in array a

first = 1; last = 15;

search

middle = first + (last - first)/2

if query == a[middle] then

output middle

else

if query < a[middle] then last = middle - 1;

if query > a[middle] then first = middle + 1;

search in a between first and last
```

things in a program

- · variables: data items that may change over time
- identifiers: names of variables
- instructions/statements: actions on data items
 - compare data values
 - assignment statement: assign a new value to a variable
- control flow instructions
 - if ... then ... else
 - while / repeat

```
input: query
output: index of query in array a
first = 1; last = 15;

middle = first + (last - first)/2
if query == a[middle] then
  output middle
else
  if query < a[middle] then last = middle - 1;
  if query > a[middle] then first = middle + 1;
  search in a between first and last
  assignment
  statements
```

real java code for binary search!

```
private int search(int query, int first, int last)
{  int middle, result;
  middle := (first + last)/2;
  if (query == a[middle])  result = middle;
  else if (query < a[middle])
    result = search(query, first, middle-1);
  else result = search(query, middle+1, last);
  return result;
}</pre>
```

food for thought

 Under which conditions (regarding the input data) does our search algorithm work?

(Hint: Think about the number and sequence of entries in the array.)

 How can the algorithm be extended to work in cases where these assumptions don't hold?