File Edit View Help

File Edit View Help

Arrays
Part 02

Arrays

- Arrays are a collection of variables of the same type
- Foundational Data Structure
- Contiguous Block of Memory
  - The size of the Array must be specified initially
  - Arrays cannot be resized
- In Java, Arrays are considered a special kind of Object
  - Container Object
  - Identifiers contain only the reference to its contents
  - The reference *points* to contents
  - "==" Does not check the contents of the array

Creating an Array Syntax

```
//Declaring an Array
<<type>>[] <<id>>;
//Initializing an Array]
<<id>> = new <<type>>[<<size>>];
//or
<<type>>[] <<id>> = new <<type>>[size];
```

Example

```
//Creates an array of 5 integers
int[] array = new int[5];
```

Arrays

- If the values are known, it is possible to both construct the array and initialize the values at the same time.
- Values are put inside of curly braces (“{}”)
- Each value is separated by a comma (“,”)

Creating an Array Syntax

```
//Declaring an Array and Initializing its Values
<<type>>[] <<id>> = {<<Value0>>,<<Value1>>,…};
```

Example

```
//Creates an array of 5 integers
int[] array = {0,1,2,3,4};
```

Searching and Sorting

- Search and Sorting Arrays are fundamental to their function
- Searching involves *looking* through an array for some "target" value
  - Target values can be specific values
  - They can also be values with special properties like the minimum or maximum

Searching for a Target Value

```
int[] a = {0,1,2,3,4};
boolean found = false;
int target = keyboard.nextInt();
for(int i=0;i<a.length;i++)
{
	if(a[i] == target)
	{
		found = true;
	}
}
```

Searching and Sorting

- Search and Sorting Arrays are fundamental to their function
- Searching involves *looking* through an array for some “target” value
  - Target values can be specific values
  - They can also be values with special properties like the minimum or maximum

Searching for the MAX Value

```
int[] a = {0,1,2,3,4};
int max = a[0];//Should not be arbitrary
for(int i=1;i<a.length;i++)
{
	if(a[i] > max)
	{
		max = a[i];
	}
}
```

Sorting

- We will assume ascending order (smallest value to largest)
- Selection Sort Algorithm
  1. Start at the first index
  2. Assume this value is in the correct location
  3. Check all other values after this index
  4. If a smaller value is found, then mark that index
  5. If the assumed index does not contain smallest value, then swap that with the marked index
  6. Repeat step 2 for all indices

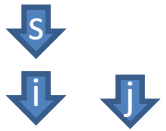
Selection Sort Example

Index	0	1	2	3	4	5	6	7
Value	9	2	1	3	5	4	8	7

Sorting

- We will assume ascending order (smallest value to largest)
- Selection Sort Algorithm
  1. Start at the first index
  2. Assume this value is in the correct location
  3. Check all other values after this index
  4. If a smaller value is found, then mark that index
  5. If the assumed index does not contain smallest value, then swap that with the marked index
  6. Repeat step 2 for all indices

Selection Sort Example



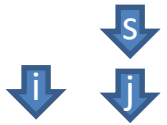
Index	0	1	2	3	4	5	6	7
Value	9	2	1	3	5	4	8	7

s = Index with smallest value

Sorting

- We will assume ascending order (smallest value to largest)
- Selection Sort Algorithm
  1. Start at the first index
  2. Assume this value is in the correct location
  3. Check all other values after this index
  4. If a smaller value is found, then mark that index
  5. If the assumed index does not contain smallest value, then swap that with the marked index
  6. Repeat step 2 for all indices

Selection Sort Example



Index	0	1	2	3	4	5	6	7
Value	9	2	1	3	5	4	8	7

s = Index with smallest value

Sorting

- We will assume ascending order (smallest value to largest)
- Selection Sort Algorithm
  1. Start at the first index
  2. Assume this value is in the correct location
  3. Check all other values after this index
  4. If a smaller value is found, then mark that index
  5. If the assumed index does not contain smallest value, then swap that with the marked index
  6. Repeat step 2 for all indices

Selection Sort Example

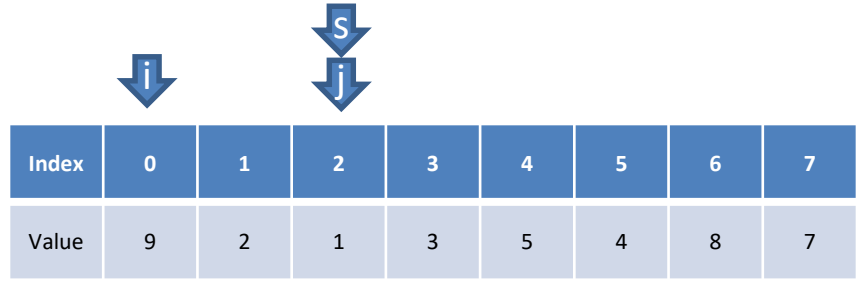
Index	0	1	2	3	4	5	6	7
Value	9	2	1	3	5	4	8	7

s = Index with smallest value

Sorting

- We will assume ascending order (smallest value to largest)
- Selection Sort Algorithm
  1. Start at the first index
  2. Assume this value is in the correct location
  3. Check all other values after this index
  4. If a smaller value is found, then mark that index
  5. If the assumed index does not contain smallest value, then swap that with the marked index
  6. Repeat step 2 for all indices

Selection Sort Example

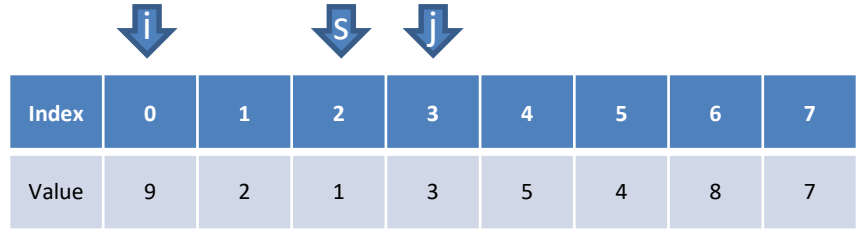


s = Index with smallest value

Sorting

- We will assume ascending order (smallest value to largest)
- Selection Sort Algorithm
  1. Start at the first index
  2. Assume this value is in the correct location
  3. Check all other values after this index
  4. If a smaller value is found, then mark that index
  5. If the assumed index does not contain smallest value, then swap that with the marked index
  6. Repeat step 2 for all indices

Selection Sort Example



s = Index with smallest value

Sorting

- We will assume ascending order (smallest value to largest)
- Selection Sort Algorithm
  1. Start at the first index
  2. Assume this value is in the correct location
  3. Check all other values after this index
  4. If a smaller value is found, then mark that index
  5. If the assumed index does not contain smallest value, then swap that with the marked index
  6. Repeat step 2 for all indices

Selection Sort Example

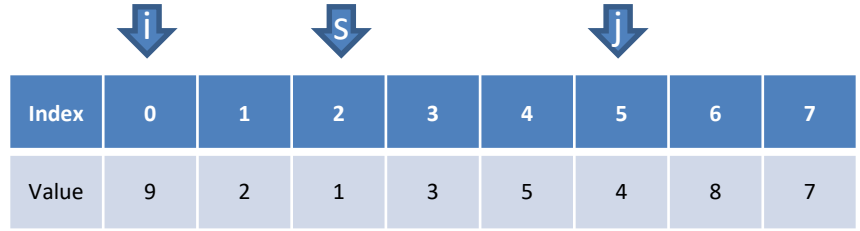
Index	0	1	2	3	4	5	6	7
Value	9	2	1	3	5	4	8	7

s = Index with smallest value

Sorting

- We will assume ascending order (smallest value to largest)
- Selection Sort Algorithm
  1. Start at the first index
  2. Assume this value is in the correct location
  3. Check all other values after this index
  4. If a smaller value is found, then mark that index
  5. If the assumed index does not contain smallest value, then swap that with the marked index
  6. Repeat step 2 for all indices

Selection Sort Example

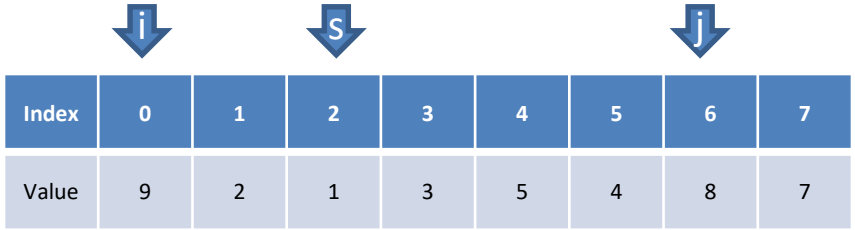


s = Index with smallest value

Sorting

- We will assume ascending order (smallest value to largest)
- Selection Sort Algorithm
  1. Start at the first index
  2. Assume this value is in the correct location
  3. Check all other values after this index
  4. If a smaller value is found, then mark that index
  5. If the assumed index does not contain smallest value, then swap that with the marked index
  6. Repeat step 2 for all indices

Selection Sort Example

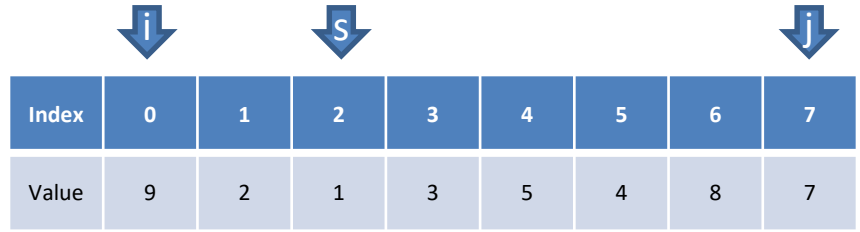


s = Index with smallest value

Sorting

- We will assume ascending order (smallest value to largest)
- Selection Sort Algorithm
  1. Start at the first index
  2. Assume this value is in the correct location
  3. Check all other values after this index
  4. If a smaller value is found, then mark that index
  5. If the assumed index does not contain smallest value, then swap that with the marked index
  6. Repeat step 2 for all indices

Selection Sort Example



s = Index with smallest value

Sorting

- We will assume ascending order (smallest value to largest)
- Selection Sort Algorithm
  1. Start at the first index
  2. Assume this value is in the correct location
  3. Check all other values after this index
  4. If a smaller value is found, then mark that index
  5. If the assumed index does not contain smallest value, then swap that with the marked index
  6. Repeat step 2 for all indices

Selection Sort Example

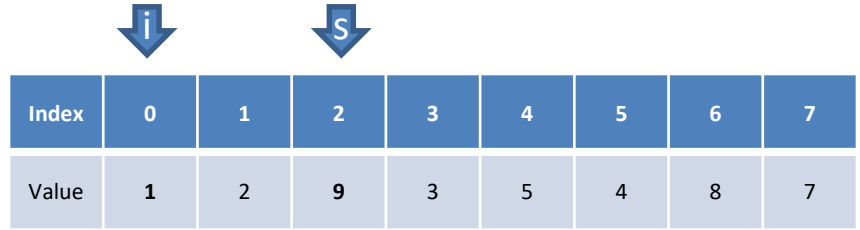
Index	0	1	2	3	4	5	6	7
Value	9	2	1	3	5	4	8	7

s = Index with smallest value

Sorting

- We will assume ascending order (smallest value to largest)
- Selection Sort Algorithm
  1. Start at the first index
  2. Assume this value is in the correct location
  3. Check all other values after this index
  4. If a smaller value is found, then mark that index
  5. If the assumed index does not contain smallest value, then swap that with the marked index
  6. Repeat step 2 for all indices

Selection Sort Example

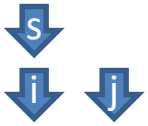


s = Index with smallest value

Sorting

- We will assume ascending order (smallest value to largest)
- Selection Sort Algorithm
  1. Start at the first index
  2. Assume this value is in the correct location
  3. Check all other values after this index
  4. If a smaller value is found, then mark that index
  5. If the assumed index does not contain smallest value, then swap that with the marked index
  6. Repeat step 2 for all indices

Selection Sort Example



Index	0	1	2	3	4	5	6	7
Value	1	2	9	3	5	4	8	7

s = Index with smallest value

File Edit View Help

Example

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
  1. Start at the first index
  2. Examine that index's *neighbor*
  3. If the neighbor has a smaller value, then swap values
  4. Move to the next index
  5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

Bubble Sort Example

Index	0	1	2	3	4	5	6	7
Value	9	2	1	3	5	4	8	7

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
  1. Start at the first index
  2. Examine that index's *neighbor*
  3. If the neighbor has a smaller value, then swap values
  4. Move to the next index
  5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

Bubble Sort Example

Index	0	1	2	3	4	5	6	7
Value	9	2	1	3	5	4	8	7

j = i+1

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
  1. Start at the first index
  2. Examine that index's *neighbor*
  3. If the neighbor has a smaller value, then swap values
  4. Move to the next index
  5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

Bubble Sort Example

Index	0	1	2	3	4	5	6	7
Value	**2**	**9**	1	3	5	4	8	7

j = i+1

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
  1. Start at the first index
  2. Examine that index's *neighbor*
  3. If the neighbor has a smaller value, then swap values
  4. Move to the next index
  5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

Bubble Sort Example

Index	0	1	2	3	4	5	6	7
Value	2	9	1	3	5	4	8	7

j = i+1

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
  1. Start at the first index
  2. Examine that index's *neighbor*
  3. If the neighbor has a smaller value, then swap values
  4. Move to the next index
  5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

Bubble Sort Example

Index	0	1	2	3	4	5	6	7
Value	2	**1**	**9**	3	5	4	8	7

j = i+1

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
  1. Start at the first index
  2. Examine that index's *neighbor*
  3. If the neighbor has a smaller value, then swap values
  4. Move to the next index
  5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

Bubble Sort Example

Index	0	1	2	3	4	5	6	7
Value	2	1	9	3	5	4	8	7

j = i+1

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
  1. Start at the first index
  2. Examine that index's *neighbor*
  3. If the neighbor has a smaller value, then swap values
  4. Move to the next index
  5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

Bubble Sort Example

Index	0	1	2	3	4	5	6	7
Value	2	1	**3**	**9**	5	4	8	7

j = i+1

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
  1. Start at the first index
  2. Examine that index's *neighbor*
  3. If the neighbor has a smaller value, then swap values
  4. Move to the next index
  5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

Bubble Sort Example

Index	0	1	2	3	4	5	6	7
Value	2	1	3	9	5	4	8	7

j = i+1

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
  1. Start at the first index
  2. Examine that index's *neighbor*
  3. If the neighbor has a smaller value, then swap values
  4. Move to the next index
  5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

Bubble Sort Example

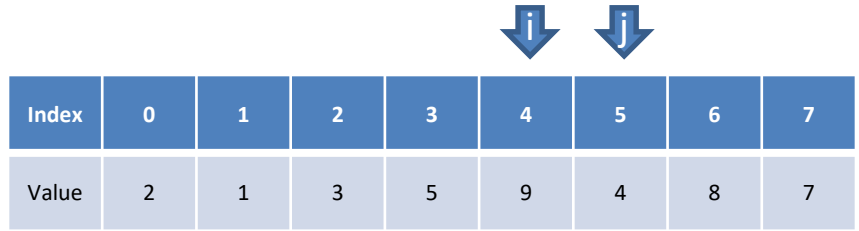
Index	0	1	2	3	4	5	6	7
Value	2	1	3	**5**	**9**	4	8	7

j = i+1

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
  1. Start at the first index
  2. Examine that index's *neighbor*
  3. If the neighbor has a smaller value, then swap values
  4. Move to the next index
  5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

Bubble Sort Example

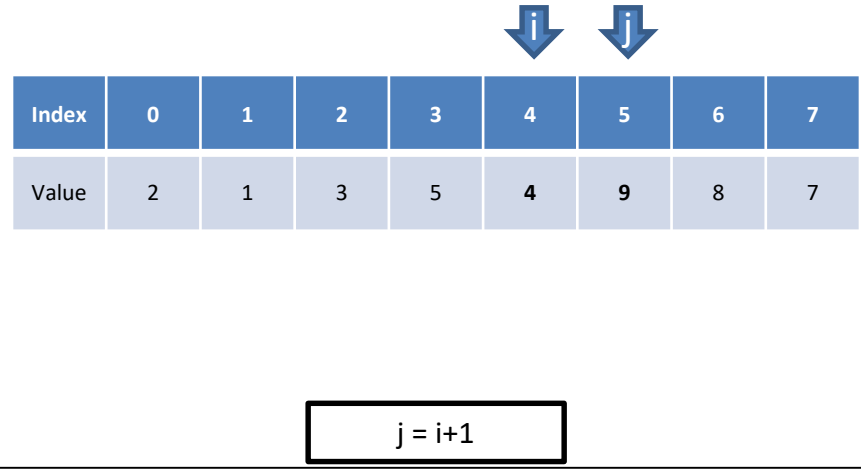


j = i+1

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
  1. Start at the first index
  2. Examine that index's *neighbor*
  3. If the neighbor has a smaller value, then swap values
  4. Move to the next index
  5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

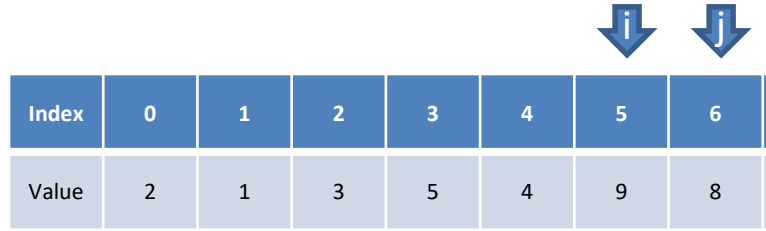
Bubble Sort Example



Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
  1. Start at the first index
  2. Examine that index's *neighbor*
  3. If the neighbor has a smaller value, then swap values
  4. Move to the next index
  5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

Bubble Sort Example



j = i+1

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
  1. Start at the first index
  2. Examine that index's *neighbor*
  3. If the neighbor has a smaller value, then swap values
  4. Move to the next index
  5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

Bubble Sort Example

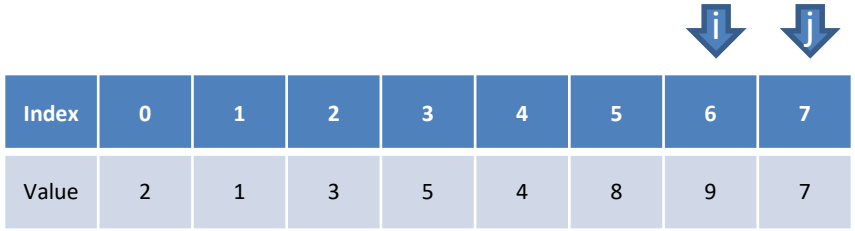
Index	0	1	2	3	4	5	6	7
Value	2	1	3	5	4	**8**	**9**	7

j = i+1

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
  1. Start at the first index
  2. Examine that index's *neighbor*
  3. If the neighbor has a smaller value, then swap values
  4. Move to the next index
  5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

Bubble Sort Example

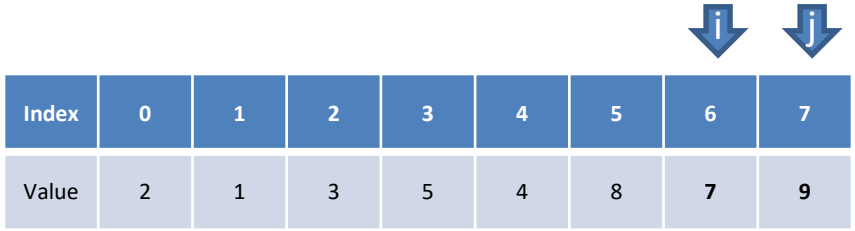


j = i+1

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
  1. Start at the first index
  2. Examine that index's *neighbor*
  3. If the neighbor has a smaller value, then swap values
  4. Move to the next index
  5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

Bubble Sort Example



j = i+1

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
  1. Start at the first index
  2. Examine that index's *neighbor*
  3. If the neighbor has a smaller value, then swap values
  4. Move to the next index
  5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

Bubble Sort Example

Index	0	1	2	3	4	5	6	7
Value	2	1	3	5	4	8	7	9

j = i+1

Sorting

- We will assume ascending order (smallest value to largest)
- Bubble Sort Algorithm
  1. Start at the first index
  2. Examine that index's *neighbor*
  3. If the neighbor has a smaller value, then swap values
  4. Move to the next index
  5. If the next index is the last index and there has been at least 1 swap, then repeat Step 2

Bubble Sort Example

Index	0	1	2	3	4	5	6	7
Value	2	1	3	5	4	8	7	9

j = i+1

File Edit View Help

Example