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

- 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 and Initializing its Values
<<type>>[] <<id>> = {<<Value0>>,<<Value1>>,...};

Example

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

```java
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;
    }
}
```
• 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

```java
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];
    }
}
```
• 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

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>9</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>8</td>
<td>7</td>
</tr>
</tbody>
</table>
• 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

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>9</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>8</td>
<td>7</td>
</tr>
</tbody>
</table>

s = Index with smallest value
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
• 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

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>9</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>8</td>
<td>7</td>
</tr>
</tbody>
</table>

$s =$ Index with smallest value
• 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
• 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

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>9</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>8</td>
<td>7</td>
</tr>
</tbody>
</table>

$s =$ Index with smallest value
• 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

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>9</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>8</td>
<td>7</td>
</tr>
</tbody>
</table>

\[ s = \text{Index with smallest value} \]
• 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

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>9</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>8</td>
<td>7</td>
</tr>
</tbody>
</table>

\[ s = \text{Index with smallest value} \]
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
• 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
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

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>9</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>8</td>
<td>7</td>
</tr>
</tbody>
</table>

\[ s = \text{Index with smallest value} \]
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

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>1</td>
<td>2</td>
<td>9</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>8</td>
<td>7</td>
</tr>
</tbody>
</table>

s = Index with smallest value
• 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
Example
• 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

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>9</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>8</td>
<td>7</td>
</tr>
</tbody>
</table>
• 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
• 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

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>2</td>
<td>9</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>8</td>
<td>7</td>
</tr>
</tbody>
</table>

\[ j = i + 1 \]
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

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>2</td>
<td>9</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>8</td>
<td>7</td>
</tr>
</tbody>
</table>

\[ j = i + 1 \]
• 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

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>2</td>
<td>1</td>
<td>9</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>8</td>
<td>7</td>
</tr>
</tbody>
</table>

\[ j = i + 1 \]
• 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
• 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

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>9</td>
<td>5</td>
<td>4</td>
<td>8</td>
<td>7</td>
</tr>
</tbody>
</table>

\[ j = i + 1 \]
• 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

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>9</td>
<td>5</td>
<td>4</td>
<td>8</td>
<td>7</td>
</tr>
</tbody>
</table>

![](j=i+1)
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

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>9</td>
<td>4</td>
<td>8</td>
<td>7</td>
</tr>
</tbody>
</table>

\[ j = i + 1 \]
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

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>9</td>
<td>4</td>
<td>8</td>
<td>7</td>
</tr>
</tbody>
</table>

\[ j = i + 1 \]
• 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

---

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>9</td>
<td>8</td>
<td>7</td>
</tr>
</tbody>
</table>

\[ j = i + 1 \]
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
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

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>8</td>
<td>9</td>
<td>7</td>
</tr>
</tbody>
</table>

\[ j = i + 1 \]
• 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**

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>8</td>
<td>9</td>
<td>7</td>
</tr>
</tbody>
</table>

```
j = i + 1
```
• 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

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>8</td>
<td>7</td>
<td>9</td>
</tr>
</tbody>
</table>

\[ j = i + 1 \]
• 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
• 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

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>8</td>
<td>7</td>
<td>9</td>
</tr>
</tbody>
</table>

\[
j = i + 1\]
Example