Arrays

by Chris Brown
under Prof. Susan Rodger
Duke University
June 2012
Arrays

• The purpose of this tutorial is to demonstrate how to use arrays in Alice worlds. An array is an ordered collection of objects stored by an index. Alice has two types of arrays: visual and nonvisual. Visual arrays put the items on the array index number to show their location in the array. This is useful for objects that need to stand in a line. In nonvisual arrays, however, the items can be put anywhere in the Alice world and still function as an array or they can be used for an array of numbers, strings, objects, etc.
Arrays vs. Lists

- Arrays are similar to lists, but elements in arrays are ordered and elements in a list are unordered. When traversing an array, one uses a loop (complicated version) to step through indexing particular items. Not all elements need to be processed. One could access every other element. When traversing a list, the order does not matter and the user wants to handle every element. Use “For all in order” or “For all together” to process the elements in a list.
Standards

CSTA Standard 5.3.B- Computer Science Concepts and Practices (CT):
Students will be able to…

“6. Compare and contrast simple data structures and their uses (e.g., arrays and lists).”
Getting Started

• You want to add in different people to put into this world. You can find them under the “People” tag or use the students under the “High School” tag. I used 8 people in my world to form the array as well as the coach to describe what’s going on (9 people total). Now we’re ready to build the array!
Visual Arrays

• To use a visual array, go to the “Visualizations” folder in the Alice Local Gallery and add an ArrayVisualization to your world.
Visual Arrays

• After adding the ArrayVisualization, a menu will pop up for you to add elements to the array. Select “new item” to add objects into the array and choose the items that you would like to add at the specified index. Make sure to add “the entire ___”. You should add 8 people and have one person left over.
Visual Arrays

• And here is your visual array! Click and drag the ArrayVisualization object to turn it and move it around to get the entire array to fit the screen and facing the camera. Note that the people in the array will move with it. Don’t click on the people and move them or this will cause problems accessing the array later on. Click “Undo” if you accidentally move a person.
Using the Array

• Now we will go over several uses for the visual array including modifying each element of the array, every other element of the array, choosing random elements in the array, accessing specified indices of the array, and swapping elements in the array

• (Note that for arrays with n objects, the first element is at index 0 and the last element has the position n-1)!
Using the Array

- Create a new world method by clicking on “world” in the object tree on the left and then under “world’s details” click on the methods tab and then “create new method”. Name this method *eachElement*. 
Loops

- Loops are very important for iterating through the objects of an array. In this method, drag in a Loop from the bottom of the screen into the “Do Nothing”. Have the loop go from 0 to the number of elements in the array, in this case we want to choose 8. When the menu comes up, go down to “other…” and enter 8 into the calculator.
To get each element to do something in order, click on any element of the array in the object tree and go to the methods tab. The will be your “default person” that you will use for all of the characters’ methods that you want the elements of the array to perform. Have your object move up ½ meter then move back down.
Loops

• If you click on “show complicated version”, you will see more information about how loops work. The index variable is used to traverse the array and its value will increase from 0 to 7 each time you go through the loop.
Each Element

Next, click on arrayVisualization in the object tree and go to the properties panel. Under properties, drag the elements property over your “default person” that you used before to get the instructions and select “ith item from array” → “expressions” → “index” to access the object at position index in the array.
And that concludes how to modify each element in an array in Alice! Make sure to test this method by changing the Events panel at top right corner select “When the world starts, do → world.eachElement” to run this when the world starts.
Each Element (Reverse)

- Now, we want to go through the elements of the array backwards. Unfortunately, the complicated loop will not let you increment by negative numbers, so we will have to use the formula $7 - \text{index}$ to process the correct position in the array. Note that as index increases, $7 - \text{index}$ will decrease.

<table>
<thead>
<tr>
<th>index</th>
<th>$7 - \text{index}$ (current position of the array)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
</tr>
</tbody>
</table>
Each Element (Reverse)

• To go through the array in reverse, you only need to change one thing. When you select the ith item from array, select the highest index of the array, in this case, 7. Then, click on the down-arrow next to the number and select “math” → “7 –” → “expressions” → “index”.

![Diagram showing array visualization and selection options](image-url)
Each Element (Reverse)

- If you create a method called \textit{reverse} to try this out, the two lines inside of the loop should look like this (Have the object move up and down again):

- Now, add a line at the beginning of the method to have the coach say, “Now in reverse”, under the coach’s methods before the loop starts and you are done with the \textit{reverse} method. Test this method out by changing “When the world starts, do $\rightarrow$ world.reverse” in the Events editor.
Every Other Element

• Next, we will discover how to change every other element in an array. Start by creating a new world method called `everyOtherElement`, and adding a loop into the method from 0 to 8. Make sure to hit the “show complicated version” button of the loop, and your loop should be a longer structure like this:
Every Other Element

- There are a couple of ways to modify the elements of an array at a particular interval. The simplest way is to change the “incrementing by” portion of the loop to 2 or whatever interval you want. Add the methods that you would like done in the loop and only every other element will act out the method. We chose to have them turn left and then right 1 revolution each direction. Remember, when asked for the \( i \)th item from array after dragging the elements property into the method, go to “expressions” \( \rightarrow \) “index”.

![Image of a programming interface showing the use of the elements property and loop settings]
Every Other Element

• On your own, try creating a method that has every third element in the array do the action of your choice. Name this method `everyThirdElement`. Don’t forget to test your new method to see if it works by changing the event “When the world starts, do –”…
Every Other Element Different

- Another way to modify every other element in an array is to check if the index is even or odd, or is divisible by some number. This is better if you want every object to do an action, then every other object to do a different action. Create a new world method called everyOtherElementDiff.
Every Other Element Different

- To do this, you first need to drag an If/Else block inside of your loop from the bottom of the window and select true for now.
Every Other Element Different

- Next, go to the world’s functions tab and select “a == b”. Let a be any arbitrary number (I chose 0.25 but this value will be replaced) and let b = 0. Remember that you may need to select “other…” and type in 0.
Every Other Element Different

- Scroll down in the functions tab, select “IEEERemainder of a/b” and drag it over the arbitrary value that you used for a before. When asked for the value of a, go to “expressions” → “index” OR just drag the index variable from the loop, and let b = 2. This will check to see if the remainder of the index divided by 2 returns a remainder of zero or not, determining if the index is odd or even.
Now, have the objects at the even indices turn backward then forward ¼ revolution, and the objects at odd indices just turn left one revolution. (Remember to use the “default person”, then replace them with the elements property of the array). And that concludes modifying every other element in an array! Have your extra person say “Every other element different.” and test this method by changing the event when the world starts.
Random Elements

- Now, we’ll go over how to choose random elements from an array. First, create a new world method called `randomElements` and add a loop from 0 to the number of random positions that you want in this method (I chose 5). You will also want to create a new number variable and call it `random` that will be used to calculate the random indices to be modified. You should see your variable show up to the left.
Random Elements

• Now drag the *random* variable into the loop and when prompted, set the value of *random* to any value for now. In the example below I set the value to 1.

• Then you want to go to the world’s functions tab and find the “random number” function under the random heading. Drag that over the number that you originally set as the value for *random*. 
Random Elements

- Click on “more…” next to the random item that you just dragged into the method. We only need integer values from 0 to 7 to get the positions in the array, so set the minimum value to 0, the maximum value to 8, and integerOnly to be true.

- Note: This statement generates random numbers up to but not including the maximum value (8), making 7 the largest possible integer produced.
Random Elements

- This finishes creating the random number generator. Now, fill in the code with the instructions that you want, using your “default person” and then replacing him with the elements of the array from arrayVisualization’s properties. But this time, instead of choosing “index” as the ith element from array, go to “expressions” → “random”. Our random person moves forward 1 meter, turns and says the value of random, and then moves back to their position.
Random Elements

• And that concludes the portion on random elements in an array. The code below has the coach say “5 random elements”, then selects 5 random objects from the array that move forward, turn and say their position number, then move back. Make sure you don’t forget to change the event “When the world starts do” → randomElements and run your world.
Specific Elements

• Now we will see how to modify certain elements of an array. For example, if you only want the certain positions in the array to do something. We want the user to choose 2 objects and then have them switch places. To accomplish this, you will first need to create a new world method we will call `specificElements` and then create a second method to swap the two elements chosen.
Specific Elements

• Now we need to create a variable to obtain the user input. Click on “create new variable” in the top right corner of the method. Make this a number variable and call it $\text{num}$. 

![Create New Local Variable dialog box](image-url)
Specific Elements

- The new variable should appear at the top of the method editor. Drag the `num` variable into the method, and set the value to 1 for now.
Specific Elements

Then, go to the world functions tab and drag the “ask user for a number” function over the “1” that we previously put for the value of $num$. This will create a pop-up box to ask the user for a number while the world is running. When prompted for a question to enter, type in a string to ask the user for a number between 0 and 7.
Specific Elements

• Now, add whatever instructions that you want into the method by using the methods of the “default object” in the array. When dragging the elements of the array from arrayVisualization’s properties over the object name this time, select *num* under “ith element from array” → “expressions”.

```
num set value to ask user for a number question = Enter a number between 0 and 7.
item num from arrayVisualization.elements say You chose me!
```
Specific Elements

- Have the character move forward and say “You chose me” at the same time, then have them move back to their position. Now, create a new number variable called \textit{first}. This variable will save the number that the user enters, because we will need it later on. Drag \textit{first} into the method and set this variable to “expressions” $\Rightarrow \text{num}$.
Next, you are going to want to copy all of the instructions that we just added and move them all to the bottom and replace all of the <None> subjects with *num* as they are in the first half. Create a new number variable called *second* to store the second value the user will enter and put it where *first* was in the top part to save the second value. The entire method code can be seen on the next slide.
num set value to ask user for a number question = Enter a number between 0 and 7.

Do together
- item num from arrayVisualization.elements say You chose me! duration = 2 seconds
- item num from arrayVisualization.elements move forward 1 meter
- item num from arrayVisualization.elements move backward 1 meter

first set value to num

num set value to ask user for a number question = Enter another number between 0 and 7.

Do together
- item num from arrayVisualization.elements say You chose me! duration = 2 seconds
- item num from arrayVisualization.elements move forward 1 meter
- item num from arrayVisualization.elements move backward 1 meter

second set value to num
Specific Elements

• That concludes how to access a specific element of an array. Make sure to test this method by changing the Event. Below is a picture of the pop-up box to ask the user for an index at runtime.

[Image of pop-up box]

• Note: Entering a number that does not exist in the array (anything less than 0 or greater than 7) will cause the program to crash and you will see an “index out of bounds exception” error.
Swapping Elements

• Now, we want two elements in the array to swap places. To do this, we’ll need to add another visualization object. Click on Add Objects and go to Visualizations in the Local Gallery. Import an ObjectVisualization.
Swapping Elements

- Back in the method editor, create a new world method called \textit{swap}. In this method create two number parameters that will pass in the indices that will switch places from \textit{specificElements}. To create a new parameter, click on the “create new parameter” button to the right of the editor. Name the first parameter \textit{index1} and the second one \textit{index2}. The next two slides provide an example of how we will swap two elements in the array using these parameters and the \textit{objectVisualization}. 
Swapping Elements

• In this example, the positions that will be swapping places are 0 and 7.

• The algorithm for swapping elements in an array has 3 steps: The first is to move the object from the first index to the ObjectVisualization.
Swapping Elements

• Next, move the object at the second index of the array to the space at the first index.
• Then, move the object on the objectVisualization to the second index in the array.
Swapping Elements

- To move the object from the first index of the array to the ObjectVisualization, go to the methods of objectVisualization and drag “let objectVisualization = item” into the editor. For now choose your “default person”. Then, go to arrayVisualization properties and drag the elements of the array over the object as the item. Choose “ith item from array” → “expressions” → index1.
Swapping Elements

- Next, in arrayVisualization methods, drag “let arrayVisualization [index] = item” into the editor. Choose \textit{index1} as the index and select your “default person” again. Go to the elements of the array under and drag it over the object you used for item, selecting \textit{index2} as the index. This can be seen on the next slide.
Swapping Elements

• Resulting Code:

let objectVisualization = item index1 from arrayVisualization.elements

let arrayVisualization[ index1 ] = item index2 from arrayVisualization.elements
Swapping Elements

- For the last step, you will need to drag “let arrayVisualization[index] = item” into the editor. Select index2 as the index, and the item will be replaced by the element on objectVisualization under the properties tab OR go to “expressions” → “objectVisualization.item”.
Swapping Elements

- Drag the `swap` method at the bottom of the `specificElements` method where the user chooses 2 elements, and have those two inputs be the indices of the objects that are switched in the array. Choose the parameters to be "expressions" $\rightarrow$ `index1` $\rightarrow$ "expressions" $\rightarrow$ `index2`. 
world.my first method

- The very last thing we need to do is add all of these new methods that we created into world.my first method. They should all be listed under world’s methods and you want to drag all of them in except for *swap*, which is called in *specificElements*. Change the event to run world.my first method when the world starts and this world is complete.
Conclusion

• Arrays are very useful in programming and iterating through a group of objects. This tutorial explains how to go through the objects in an array in order, in reverse order, every other element, elements at different intervals, random elements, selected elements, and how to swap elements.