Lesson

OZOBLOCKLY ARRAY PRIMER

CREATED BY

Richard Born
Associate Professor Emeritus
Northern Illinois University
rborn2@niu.edu

TOPICS

Robotics, Programming

GRADES

7-12

METHOD

OzoBlockly

DURATION

45-60 minutes

OzoBlockly Array Primer

By Richard Born
Associate Professor Emeritus
Northern Illinois University
rborn2@niu.edu

Introduction

The release of the Master Mode 5 of OzoBlockly contains numerous powerful programming features, one of which is *arrays*. This lesson takes the student through a sequence of ten short OzoBlockly programs that bring arrays to life in a way that helps students understand this data structure. Through the use of a provided Ozomap, Evo demonstrates array elements by *speaking* the contents of the element while following a line with the array's indexes and stopping at the intersections. See Figure 1.

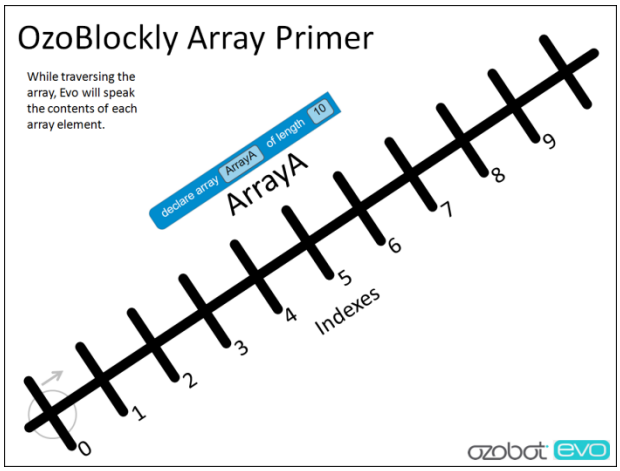

Figure 1

What is an array?

Before working with arrays in OzoBlockly, it might be a good idea to think of how the word *array* is used in everyday conversation. Here are some example sentences:

- There is a beautiful *array* of flowers in that vase.
- The *array* of solar panels on my neighbor's roof is soaking up the sun.
- Mark has an *array* of baseball bats in the corner of his bedroom.
- You can choose from an *array* of colors to paint your house.
- Sarah has an *array* of dresses in her closet.

These sentences all share a common thread. Each sentence references a *group of identical or similar things with the same name*: flowers, solar panels, bats, colors, or dresses. The concept of an array in computer science also shows this common feature. An array in computer science is *a group of sequential memory locations for storing similar information*. Examples include:

- An array storing names of customers
- An array storing restaurants within fifteen miles of your house
- An array storing prices for items that you purchase while visiting the grocery store
- An array storing colors that Ozobot encounters while following a line
- An array storing the direction that Ozobot takes for each step in a random walk

The name of the array for the "storing colors" example above might be *color*. Individual elements in an array are identified by an *index*. The first element in an array has an index 0. Languages such as JavaScript place the index in square brackets. So we could identify the first three elements of the array *color* by *color[0]*, *color[1]*, and *color[2]*. Array elements in OzoBlockly can hold any integer from -128 through 127. If we let 1 represent the color red, 2 green, and 3 blue, then the contents of *color[0]*, *color[1]*, and *color[2]* might be 2, 3, and 1 if Ozobot encountered the colors green, blue, and red in that order.

A *declare array* block in the OzoBlockly workspace is used to name and specify the length of an array. The maximum length of an array is 127, the minimum 1. You can have more than one array, if needed, in your program logic. It would be a good idea to study the array block references in the online OzoBlockly *Master Mode Reference*. OzoBlockly only supports one-dimensional arrays.

Array Declaration

The ten short programs that we study in this *OzoBlockly Array Primer* all have a single array of length 10 whose name we call *ArrayA*. See Figure 2 for the array declaration block.

Figure 2

It is recommended that the array declarations be placed in the upper left corner of the OzoBlockly workspace. It causes OzoBlockly to set aside space in Evo's memory, in this case 10 locations for storing any integers from -128 to 127. Figure 3 provides a pictorial view of this array.

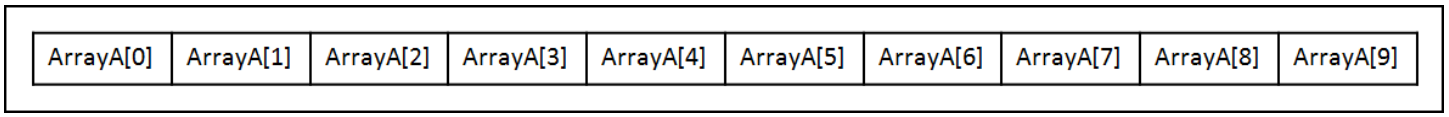

Figure 3

The indexes are shown in square brackets [ ]. Note that arrays are indexed starting at 0, so the last index in the array is always 1 less than the declared length of the array. In our array for these programs, the length is 10 and the last index is therefore 9. It should be noted that in the event the index is lower than 0 or higher than *the array length minus 1*, the program will terminate and an *index out of bound* error is indicated by a blue-red flashing LED. The length of an OzoBlockly array cannot be changed at program runtime.

With the discussion of array declaration complete, we can now begin looking at each of our ten short OzoBlockly programs, in which Evo is used to demonstrate some simple array manipulation concepts. Students are encouraged to build them in OzoBlockly, and then load/run them on the provided Ozomap.

Program 1: Zilch, Zip, Nada, Nothing

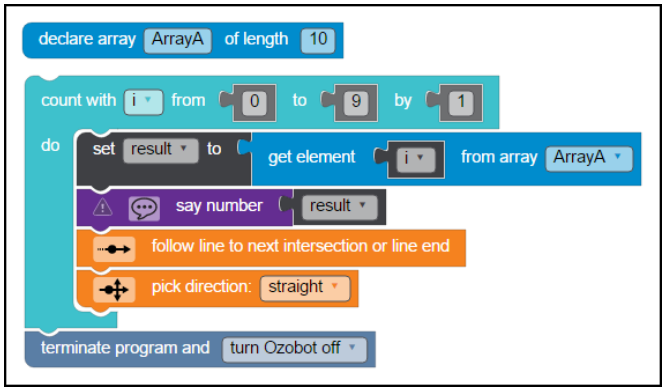

Figure 4

In the program shown in Figure 4, you will discover what the default values are for array elements. After the array declaration, we simply set up a loop in which the index (*i*) into the array varies from 0 through 9. Within the loop, we make use of the *"get element … from array …"* block to get the *value* of the array element. Then Evo speaks the *value* and follows the line to the next intersection. Fill in the values below that Evo speaks while the program is running. What do you conclude regarding the default values for array elements?

ArrayA[0]	ArrayA[1]	ArrayA[2]	ArrayA[3]	ArrayA[4]	ArrayA[5]	ArrayA[6]	ArrayA[7]	ArrayA[8]	ArrayA[9]

Program 2: All the Same to Me

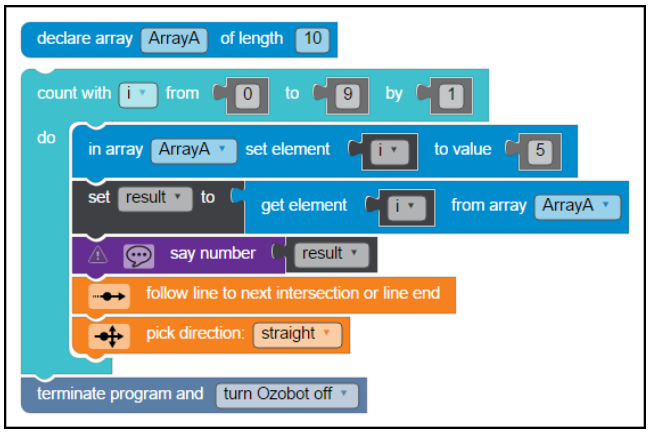

Figure 5

The program shown in Figure 5 introduces the "in array … set element … to value …" block. For each index *i* from 0 through 9, the value 5 is written to the element with that index. Other than that, the program is identical to the program of Figure 4. Fill in the values below that Evo speaks while the program is running.

ArrayA[0]	ArrayA[1]	ArrayA[2]	ArrayA[3]	ArrayA[4]	ArrayA[5]	ArrayA[6]	ArrayA[7]	ArrayA[8]	ArrayA[9]

Program 3: To Err is Human (a.k.a., to really foul things up requires a computer)

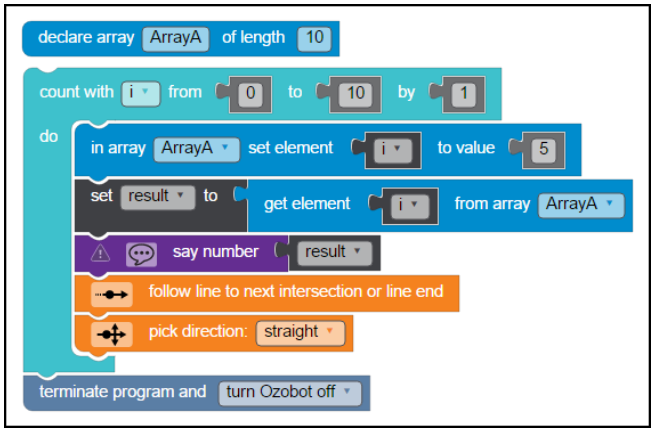

Figure 6

The program of Figure 6 has one small difference when compared to the program blocks of Figure 5. Can you find the difference? That's right—the loop goes from 0 to 10 rather than from 0 to 9. 10 is larger than the highest index allowed for an array of length 10. (Remember that the allowed indexes range from 0 to *length-1* for an array whose number of elements is *length*.) How does Evo respond to this when running the program on the Ozomap? _____

Program 4: The Count

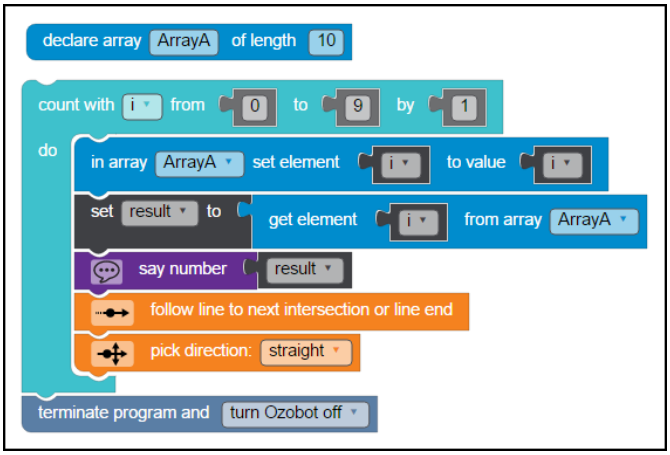

Figure 7

The program of Figure 7 has one small difference when compared to the program blocks of Figure 5. Can you find the difference? That's right—the *"in array … set element … to value …"* block has a *"to value"* with the variable *i* rather than the number 5. What do you then expect the value in each element to be? Fill in the values below that Evo speaks while the program is running. Are you expectations correct?

ArrayA[0]	ArrayA[1]	ArrayA[2]	ArrayA[3]	ArrayA[4]	ArrayA[5]	ArrayA[6]	ArrayA[7]	ArrayA[8]	ArrayA[9]

Program 5: Counting by 5's

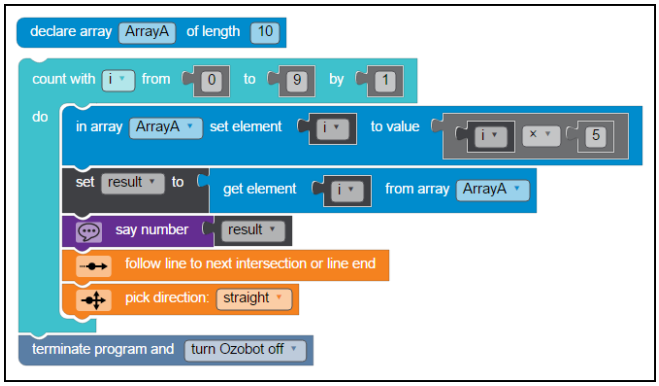

Figure 8

The program of Figure 8 differs from that of Figure 7 only in the *"to value"* argument of the *"set element"* block. Instead of the variable *i*, the *"to value"* is the arithmetic expression *i x 5*. What do you expect the value in each array element to be? Fill in the values below that Evo speaks while the program is running. Are you expectations correct?

ArrayA[0]	ArrayA[1]	ArrayA[2]	ArrayA[3]	ArrayA[4]	ArrayA[5]	ArrayA[6]	ArrayA[7]	ArrayA[8]	ArrayA[9]

Program 6: All Squared Up

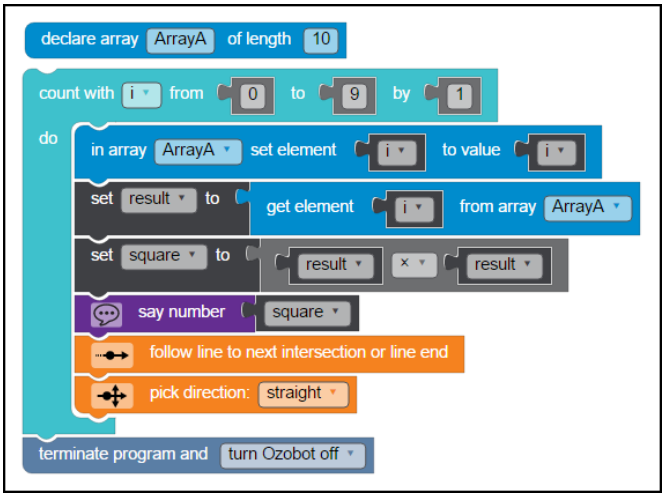

Figure 9

Study the program of Figure 9. What do you expect the value in each array element to be? Fill in the values below that Evo speaks while the program is running. Are you expectations correct?

ArrayA[0]	ArrayA[1]	ArrayA[2]	ArrayA[3]	ArrayA[4]	ArrayA[5]	ArrayA[6]	ArrayA[7]	ArrayA[8]	ArrayA[9]

Program 7: Blast off!

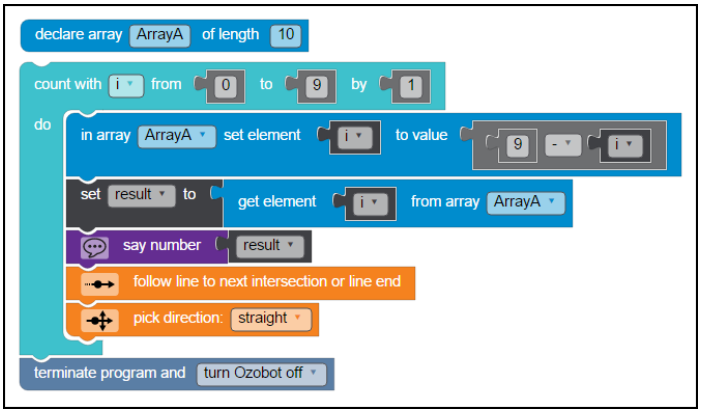

Figure 10

How does the program of Figure 10 differ from the program of Figure 7? What do you expect the value in each array element to be? Fill in the values below that Evo speaks while the program is running. Are you expectations correct?

ArrayA[0]	ArrayA[1]	ArrayA[2]	ArrayA[3]	ArrayA[4]	ArrayA[5]	ArrayA[6]	ArrayA[7]	ArrayA[8]	ArrayA[9]

Program 8: Getting Dicey

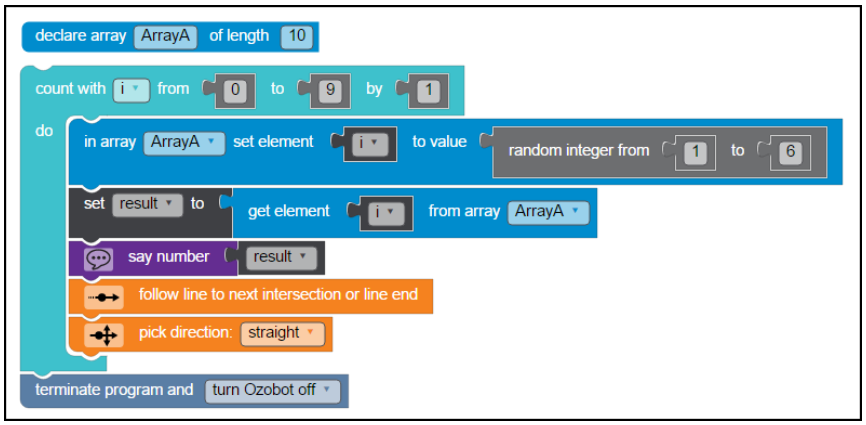

Figure 11

How does the program of Figure 11 differ from the program of Figure 5? What do you expect the values in each array element to be? Fill in the values below that Evo speaks while the program is running. Are you expectations correct? Why do you think the title of this program is "Getting Dicey"? What happens to the values in the array elements each time you run the program with Evo on the Ozomap?

ArrayA[0]	ArrayA[1]	ArrayA[2]	ArrayA[3]	ArrayA[4]	ArrayA[5]	ArrayA[6]	ArrayA[7]	ArrayA[8]	ArrayA[9]

Program 9: One by One

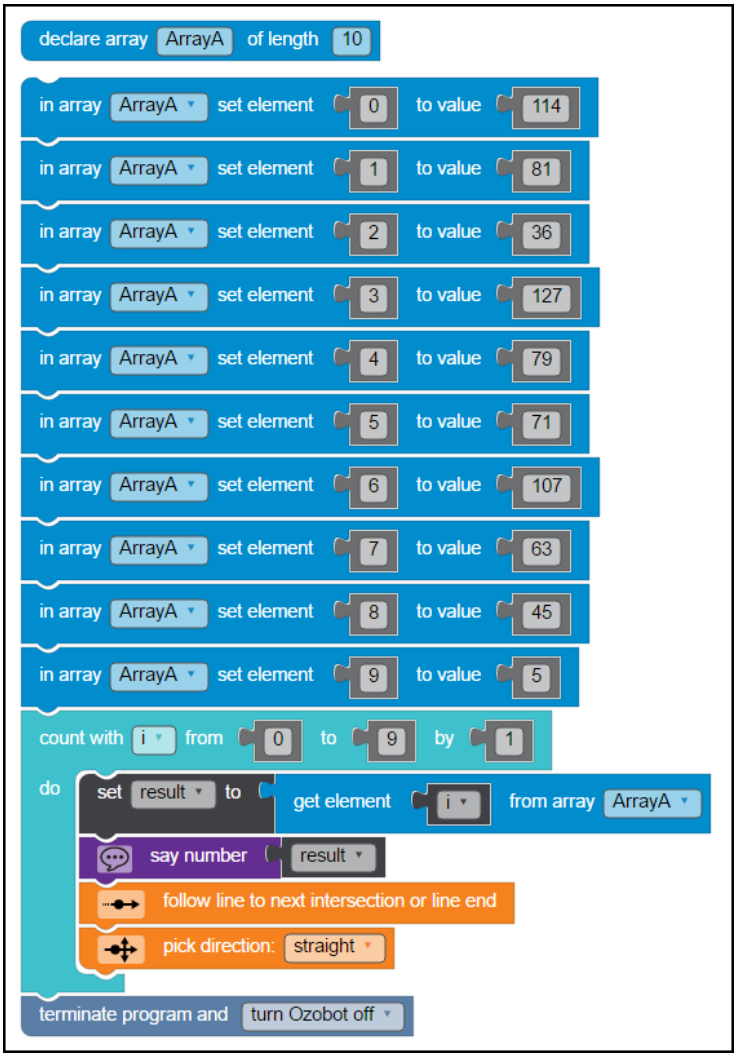

Figure 12

The program of Figure 12 has more blocks than all of the others in this lesson, but it is really not any more complex. The purpose of the ten "set element" blocks that begin the program is simply for the user to populate the array with whatever values are desired. What do you expect the values in each array element to be? Fill in the values below that Evo speaks while the program is running. Are you expectations correct?

ArrayA[0]	ArrayA[1]	ArrayA[2]	ArrayA[3]	ArrayA[4]	ArrayA[5]	ArrayA[6]	ArrayA[7]	ArrayA[8]	ArrayA[9]

Program 10: An Improvement

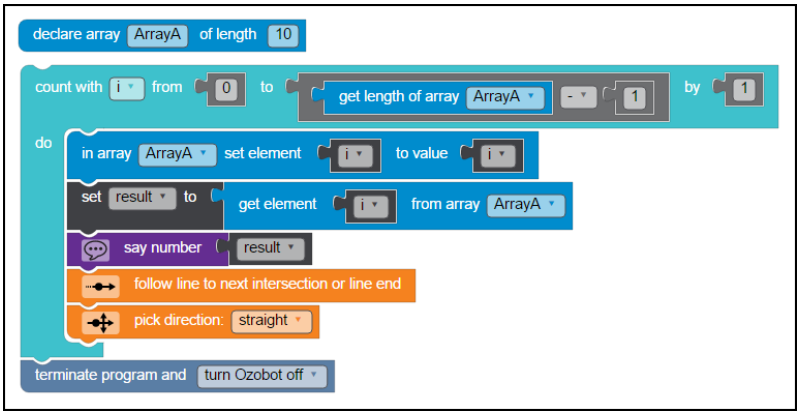

Figure 13

In this program the output is the same as it was for the program of Figure 7. However, an important change has been introduced by including the *"get length of array"* block instead of hard-coding the number 9 in the count with block, as was done in most of the previous programs. In this program the length of the array is declared as 10 in the "declare array" block. We then subtract 1 to get the index of the last element in the array. What is the main advantage of using the "get length of array block" here? _____

_____

Running the Programs

1. Students should enter and then load their array programs onto Evo and use a printed copy of the Ozomap on the last page of this document with Evo.
2. To run the loaded program on Evo, begin with Evo powered down. Place Evo centered on the gray circle in the lower left corner of the Ozomap and facing the direction shown by the arrow. Press Evo's button once to turn Evo on. Then when the front lights show blue, double-press the button. The OzoBlockly program has started running.
3. Evo will speak the array element values while stopping at each intersection. The numbers 0 through 9 at the intersections are the indexes for the array elements.

Answers to Program Questions

Program 1: Zilch, Zip, Nada, Nothing – the default value for array elements is 0, the same as for any OzoBlockly variable. Element values: 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.

Program 2: All the Same to Me – Element values: 5, 5, 5, 5, 5, 5, 5, 5, 5, 5.

Program 3: To Err is Human – Any time that an array references an index that is out-of-bounds (i.e.,less than zero or greater than *length – 1*) the program will stop and Evo will flash red and blue alternately.

Program 4: The Count – Expected array element values: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Program 5: Counting by 5's – The expected array element values: 0, 5, 10, 15, 20, 25, 30, 35, 40, 45

Program 6: All Squared Up – Expected array element values: 0, 1, 4, 9, 16, 25, 36, 49, 64, 81.

Program 7: Blast off! – In Figure 7, array elements are set to the value *i*, whereas in Figure 10, they are set to the value *i – 1*. Expected array element values: 9, 8, 7, 6, 5, 4, 3, 2, 1, 0.

Program 8: Getting Dicey – The "set element" block in Figure 11 is setting the value to a random integer between 1 and 6 rather than to the constant 5, as in the program of Figure 5. The element values will be random numbers between 1 and 6, as in rolling a single ordinary 6-sided die. The random numbers will be different each time the program is run.

Program 9: One by One – Expected array element values: 114, 81, 36, 127, 79, 71, 107, 63, 45, 5.

Program 10: An Improvement – The most significant advantage of using the "get length of array" block is that in the event that we want to change the length of the array, we only need to adjust the value in the "declare array" block. No adjustments need to be made elsewhere in the program.

OzoBlockly Array Primer

While traversing the array, Evo will speak the contents of each array element.