PRACTICE 10: ARRAYS

10

Just List It

Let's make a list! There are lots of reasons you might want to make a list: a wish list, a bucket list, a list of chores, and so on. In code, when you want to store a numbered (indexed) list of items, you use an array. In lesson 4, you learned about variables. You can think of an Array as a series of numbered variables.

What's an Array?

Let's take a moment to think about how we would do this using individual variables (from lesson 4), storing each item individually:

```
todo1 = "Learn to skateboard"
todo2 = "Visit Tokyo"
todo3 = "Stand on the Equator"
todo4 = "Skydiving"
todo5 = "Travel to Paris"
```

An array is a special kind of variable which can hold more than one value at a time. Basically, what it means is that instead of having to create five variables (`todo1`, `todo2`, etc.), we can use `todo` (the array) to store all five items. The way we store multiple values is by using something called an "index." For example, `todo[1]`, `todo[2]`, `todo[3]`, `todo[4]`, and `todo[5]` can all store a value. It's like if `todo` were a cabinet, and each number represented a different small drawer in the cabinet where you could put something. Those numbers 1, 2, 3, 4 and 5 are called indices for the array. You can now store all the items in your list in a single array named `todo`:

```
todo[1] = "Learn to skateboard"
todo[2] = "Visit Tokyo"
todo[3] = "Stand on the Equator"
todo[4] = "Go Skydiving"
todo[5] = "Travel to Paris"
```

Next let's combine this with loops that we learned about in Practice 6, so the number of items in your list depend on user input. Here is an example that reads what you input to create your to do list:

```
input = TextWindow.Read()
index = 1
While(Text.GetLength(input) > 0)
  todo[index] = input
  index = index + 1
EndWhile
```

Did you see how `index + 1` increments the index up to the next number each time the loop runs? Then the user enters the new text to store in that index.

What if you want to know the 10th item in your to do list? You can save it in a variable and then display it:

```
todo10 = todo[10]
TextWindow.WriteLine(todo10)
```

If you want to clear an item from the array, you can simply assign some default value to that index:

```
todo[1] = "Hello"
todo[2] = "world!"
todo[1] = ""
```

Now `todo[2]` still contains `"world!"` while `todo[1]` contains the empty string (`""`).

Challenge: To Do List

Can you use a loop to print out your to do list to the `TextWindow` and then make another loop to clear the array?

Congratulations! You can use arrays to store and read items!

Challenge: Counting Characters

What kind of characters do you like? Super heroes? Princesses? Star Wars characters? Write a program that asks your users to enter as many of the characters as they can think of. For example, "Enter as many Disney princesses as you can think of."

Then prompt them to enter another one, and to keep doing that. Let them know to just press Enter when they can't think of any more. Then tell them how many characters they came up with!

Arrays and Indexing

In the previous discussion, we saw how you can index an array. The primary form of indexing we reviewed was indexing through numbers (`todo[1]`, `todo[2]`, etc.) Another way that we can index arrays is through strings.

Imagine this, you are in a book donation club and you are trying to log the number of books each person brings to donate. So far, you've had three friends donate books: Fred donated 2, Lucy donated 4, and Sam donated 5. You can easily store this data in your array and index it using the names of the individuals:

```
DonorArray["Fred"] = 2
DonorArray["Lucy"] = 4
DonorArray["Sam"] = 5
```

Here's a way to list the elements through a for loop, as learned in Practice 7:

```
indices = Array.GetAllIndices(DonorArray)
For i = 1 To Array.GetItemCount(indices)
      name = indices[i] 'obtains the name of the person
      TextWindow.WriteLine(name + " donated " +
      DonorArray[name] + " books")
EndFor
```

In the first line, `Array.GetAllIndices` gets the list of indices of `DonorArray`, which in this case would be `"Fred"`, `"Lucy"`, and `"Sam"`, and assigns this to a new array, named `indices`. So now, `indices[2]` contains the string value `"Lucy"`, which we can then use to find how many books lucy donated, with `DonorArray["Lucy"]` still containing the value 4.

Challenge: Pet Adoption List

Here is a log of a pet adoption center where the staff wants to log how many animals were adopted. Currently, five dogs and six cats have been adopted:

```
Numberof["Dogs"] = 5
Numberof["Cats"] = 6
```

Your challenge is to add another element in this array where you log the number of hamsters. So far two have been adopted.

After adding the number of adopted hamsters, print the array using a for loop.

Discussion Questions

- Where might you use an array in real life?
- When using an array, can you think of things that might go wrong? How many different types of indices can you use?
- When a software engineer refers to a list, they are referring to a different data type than an array. What do you think the difference might be between a list and an array?

Additional Resources

- Small Basic Getting Started Guide: Chapter 10: Arrays
  - https://aka.ms/sbguidechapter10
- Small Basic Array Basics
  - https://aka.ms/sbarraybasics
- Small Basic Documentation: Array
  - https://aka.ms/sbarray