Operations on Arrays

https://csci-1301.github.io/about#authors

February 20, 2023 (05:13:40 PM)

Contents

This lab serves multiple goals:

- To reinforce your understanding of arrays of different datatypes,
- To reinforce your understanding of the `Length` property of array,
- To introduce you to more advanced array manipulations,
- To introduce you to simple algorithms that search for values in two arrays,
- (Optional) to introduce you to simple algorithms that merge arrays.

1 Operations on Numeric Arrays

This first part will ask you to declare and initialize an array, and then to display, sum, count occurrences and retrieve information from this array. In a second moment, we will assess whenever your solution is "universal", that is, whether it produces correct results with any array.

Start by declaring and initialize an `int` array called `numbers`:

```
int[] numbers = {4, 2, 6, 1, 7, 5, 3, 4, 2, 2, 8, 6, 3, 11, 7, 2, 9, 3, 1, 9, 7};
```

1.1 Displaying Values

After declaring and initializing the `numbers` array, write statements to:

1. Display every value in order, left to right,
2. Display every value at an even index (skip odd indices),
3. Display all values that are greater than 5.

1.2 Counting Values

Next, write statements that do the following:

1. Calculate the sum of all numbers in `numbers`, then display the result. The expected answer is 102.
2. Count how many times `7` occurs in `numbers`, then display that count. The expected answer is 3.

1.3 Finding Values

Next, write statements that do the following:

1. Find the *index* of the first `7`, then display that index. The expected answer is 4, but your statements should be such that if the value is not found, it would display `-1`.
2. Find the greater value in `numbers`. The expected answer is 11.

1.4 Evaluate Your Solution: Is It Universal?

After implementing these methods, and assuming your program obtained the expected answers, *ideally* the solution still works even if the values in the `numbers` array change, or even if the array length changes.

To test your program, go back to the beginning where you declared the `numbers` array, then change the initialization so that the new array values are:

```
int[] numbers = {55, 92, 12, 90, 37, 18, 6, 20, 80, 18, 46, 19, 65, 68, 18};
```

Then re-execute the program.

Check that you obtain the expected values:

- the sum should now be `644`
- since `7` does not occur in the array anymore,
  - count should be `0`
  - first index of `7` should be `-1`
- maximum value is now `92`

2 Working With Two Arrays

For this part, declare and initialize the following two `char` arrays:

```
char[] chars1 = {'K', 's', 'Q', 'U', 'i', 'N', 'K', 'N', 'h', 't', 'u'};
char[] chars2 = {'?', 'E', 'U', 'a', 'j', 'X', 'L', 'G', '@', 'L', 'l', 'C', 'w', 'J',
 ↪  'U' };
```

Next, write statements that answer these two questions:

1. Does the value `'w'` occur in both arrays?
2. What is the first value of the array `chars1` that also occurs in the second array `chars2`, searching from left to right? If none is found, display `no match`.

After completing these two problems, make sure the program answers these questions correctly. The expected results are:

- Does `'w'` occur in both arrays → `false`
- First value that occurs in both arrays → `'U'`

Again, evaluate your work by changing the array initializations to:

```
char[] chars1 = {'s', 'p', 'd', 'P', 'y', 'D', 'w', '?'};
char[] chars2 = {'V', 'D', 'l', 'P', 'w', 'O', 'y', 'k', 'D', 'Z' };
```

Then execute the program again. Ideally the program does not crash and should still produce correct answers:

- Does `'w'` occur in both arrays → **`true`**
- First value that occurs in both arrays → `'P'`

If the program does not produce these expected answers after changing the array values, review your program and try to determine how to write a solution that works for *any* two char arrays.

3 Pushing Further (Optional)

This short exercise will require you to manipulate two arrays at the same time to construct a third one.

Start with two integer arrays:

```
int[] left = { 101, 76, 74, 94, 94 };
int[] right = { 73, 74, 67, 107, 111, 108, 66 };
```

Then, implement statements to merge those `left` and `right` arrays by creating a new, larger array that holds both of their values, in this order:

```
101, 76, 74, 94, 94, 73, 74, 67, 107, 111, 108, 66
```

Then, change the values in the `left` and `right` arrays, and make sure that your program still create the correct array.