Operations on Arrays

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

March 7, 2022 (01:27:18 PM)

Contents

1 Operations on numeric arrays

Start by creating a new C# solution.

After creating the solution, declare and initialize an `int` array called `numbers`. Initialize the array so that it holds the following values, in the same order:

```
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. Check that your program outputs the correct answer, which you can determine by visually observing the array values.

1.3 Finding values

Now implement statements to answer the following questions:

1. Find the *index* of the first `7`, then display that index. If the value does not exist, display `-1` to indicate it was not found. Check that your solution is correct by comparing what you obtain from the program with what you know by visually observing the array.
2. Find the maximum value in `numbers`. Check that the solution you implement obtains the expected value.

1.4 Evaluate your solution

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:

```
55, 92, 12, 90, 37, 18, 6, 20, 80, 18, 46, 19, 65, 68, 18
```

Then re-run 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? Display the answer, true/false.
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 run 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)

Start with two integer arrays with the following values:

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

Implement statements to merge `left` and `right` 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
```

Do not use built-in array methods from the `Array` class.