4420-AP1-AS - ALGORITHMS AND PROGRAMMING – COMPUTER SCIENCE TECHNOLOGY – PROGRAMMING
Section 07218

ASSIGNMENT 7 – Arrays (one dimension)

XIMENA CARRILLO

LaSalle College
October 19th, 2023

Assignment 7 – Arrays (One dimension)

For the following exercises:

- Create a new project called “Assignment7” into your solution “AP1_2023”
- Write all the code in a unique C# file.
- You must validate entries from the console (input data validation).
- Send the whole project, compressed in a “zip” or “rar” file by Lea.

Exercise:

Hints:

- Declare all your variables, constants and arrays before the while loop.
- Create a variable to manage the index where you want to storage the student's information.
- You could create a Boolean variable to know if there are students created or not.

Complete the following steps:

1. Declare an array to store 5 names.
2. Declare an array to store 5 ages.
3. Create a menu and a corresponding switch or conditional statement that allows the user to select any of the questions below. Each question should correspond to one case in the switch statement. Wrap your entire program in a while loop (or do-while loop) that allows the user to run it multiple times if they so choose.
    a. Option 1 - Enter the names: Read 5 names from the console and save it into the array of names (1).
    b. Option 2 - Enter the ages: Read 5 ages from the console and save it into the array of ages (2).
    c. Option 3 – Display information: Display all the names and the ages together. Example:

Index	Name	Age
0	Sam	10
1	Alex	8
2	Maria	9
3	Bob	7
4	Anne	11

Note: If there are no names and ages created, you should display a message “Please create names and ages first, and then try again”

d. Option 4 – Display smallest and largest age: Get the smallest and the largest age and display it in the console. Example:

   Largest Age: Anne – 9
   Smallest Age: Bob – 7

   Note: If there are no names and ages created, you should display a message “Please create names and ages first, and then try again”

e. Option 5 – Get Average: Get the average of the ages and display it.

   Note: If there are no names and ages created, you should display a message “Please create names and ages first, and then try again”

f. Option 6 – Display name by index: Ask to the user for an index (n) and display the name and the age is stored in the array in the index (n).
   - Validate the index entered is between 0 and 4.
   - If the index doesn’t exist, you should display the message “Please enter a number between 0 and 4”

g. Option 7. Exit the program.

To Practice:

These exercises are optional, but they are helpful to understand how the arrays works in C#.

1. Write a program that reads an array of real numbers, and then initializes the elements of the array with one value “*x”* (also reading by console). So that every element of the array is initialized to the value of *x*. Display the initial array and the result.
2. Write a program that reads an array (A) of integer numbers and create another array (B) with the cube of each element of the first array (A). Display both arrays.
3. Write a program that multiplies each element of an array of doubles by a double number read by console.
4. Write a program that reads an array of strings. Then the program asks for a string "*s*". The program retrieves if “s” is in the array or not.
5. Write a program that reads an array of integer numbers and an integer (n). The program should display the number of times that this integer *n* is found in the array (number of occurrences).
6. Write a program that reads an array of integer numbers and displays the indices and values of all elements that have a negative value (< 0).
7. Write a program that receives an array of integers and an integer *n*. The program should display the number of elements in the array that are greater or equal to *n*.

Bonus:

Write a program that receives an array of integers and an integer *n*. The program applies *n* <u>rightward</u> rotations to the values contained in the elements of the array. Upon each rotation, the value of the last element of the array is placed in the first element, and the others are moved one element to the right.

Example: If the program receives an array containing: 4 3 8 3 6 4 9 5 0 1 5 2, and if *n* is 4, then after the call, the array will contain: 0 1 5 2 4 3 8 3 6 4 9 5.