**Tutorial C# Basics 7**

**Array and List**

V1.0

07/08/2022

An array/list is a data structure for storing more than one data item that has a similar data type. By the end of this tutorial, you will be able to

* Tell what array is and what list is
* What are the differences between array and list
* Program array and list in C#

**Topics**

[**1.** **Data structure** 2](#_Toc110727571)

[**Array** 2](#_Toc110727572)

[**Tree** 2](#_Toc110727573)

[**Hash** 3](#_Toc110727574)

[**Graph** 4](#_Toc110727575)

[**2.** **Array in programming** 4](#_Toc110727576)

[**3.** **C# Array (1-D)** 5](#_Toc110727577)

[**Class activity 1** 6](#_Toc110727578)

[**Class activity 2** 6](#_Toc110727579)

[**Class activity 3** 6](#_Toc110727580)

[**Class activity 4** 7](#_Toc110727581)

[**Class activity 5** 7](#_Toc110727582)

[**4.** **List in programming** 8](#_Toc110727583)

[**5.** **C# List** 8](#_Toc110727584)

[**Class activity 6** 8](#_Toc110727585)

[**6.** **C# multidimensional arrays** 9](#_Toc110727586)

[**Challenge** 9](#_Toc110727587)

[**Model answers** 10](#_Toc110727588)

[**Class activity 1** 10](#_Toc110727589)

[**Class activity 2** 10](#_Toc110727590)

[**Class activity 3** 11](#_Toc110727591)

[**Class activity 4** 11](#_Toc110727592)

[**Class activity 5** 11](#_Toc110727593)

[**Class activity 6** 12](#_Toc110727594)

1. **Data structure**

In many cases, data to be processed in a program relate to each other. Thus, we need to organizing data in computer so that it can be used effectively. Here are some basic structures how data are organized

**Array**

An array can be of 1-D, 2-D, 3-D and n-D (n = 1, 2, 3, 4, 5 …).

1-D Array

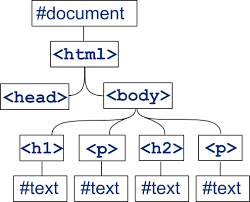
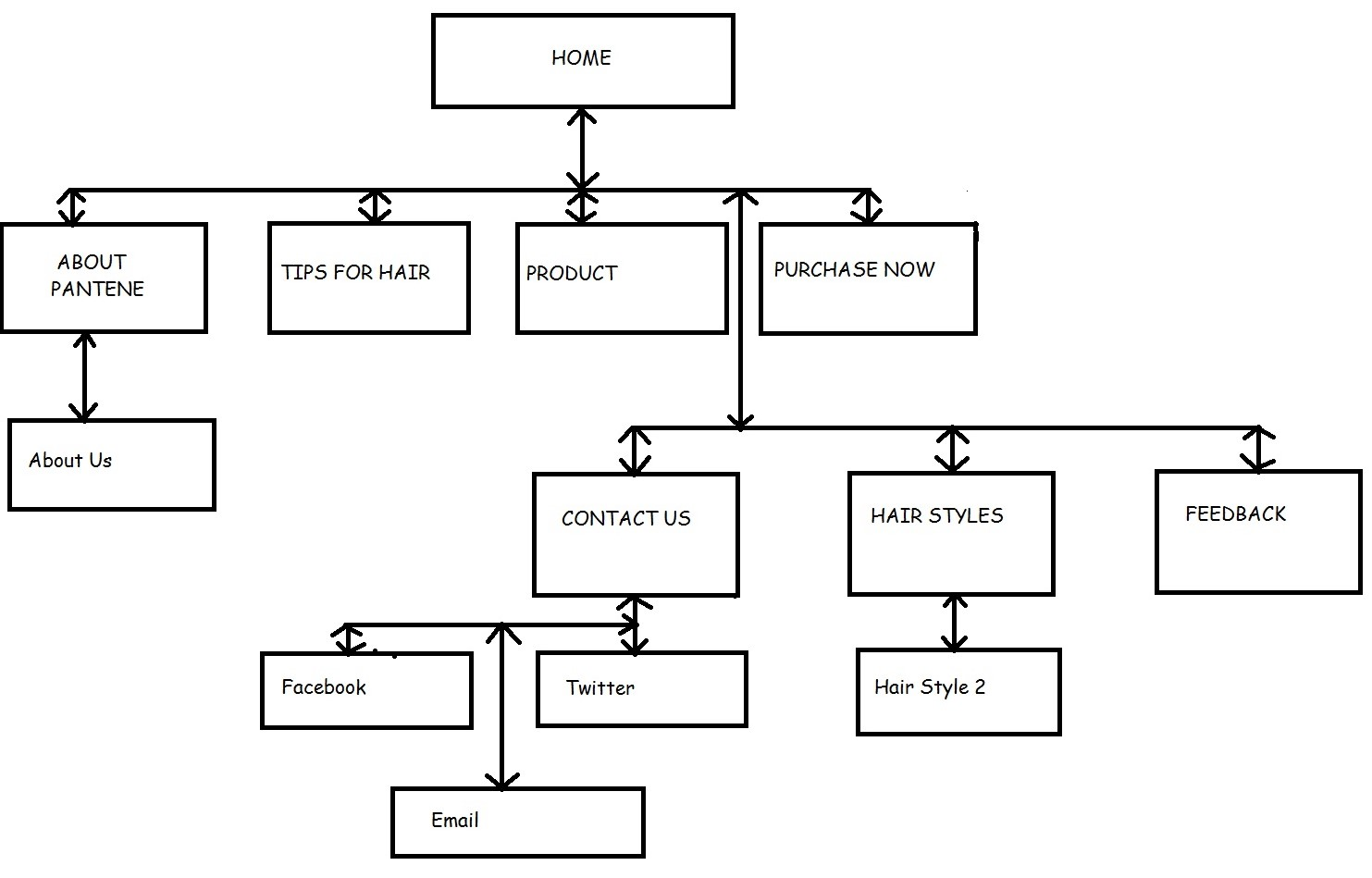
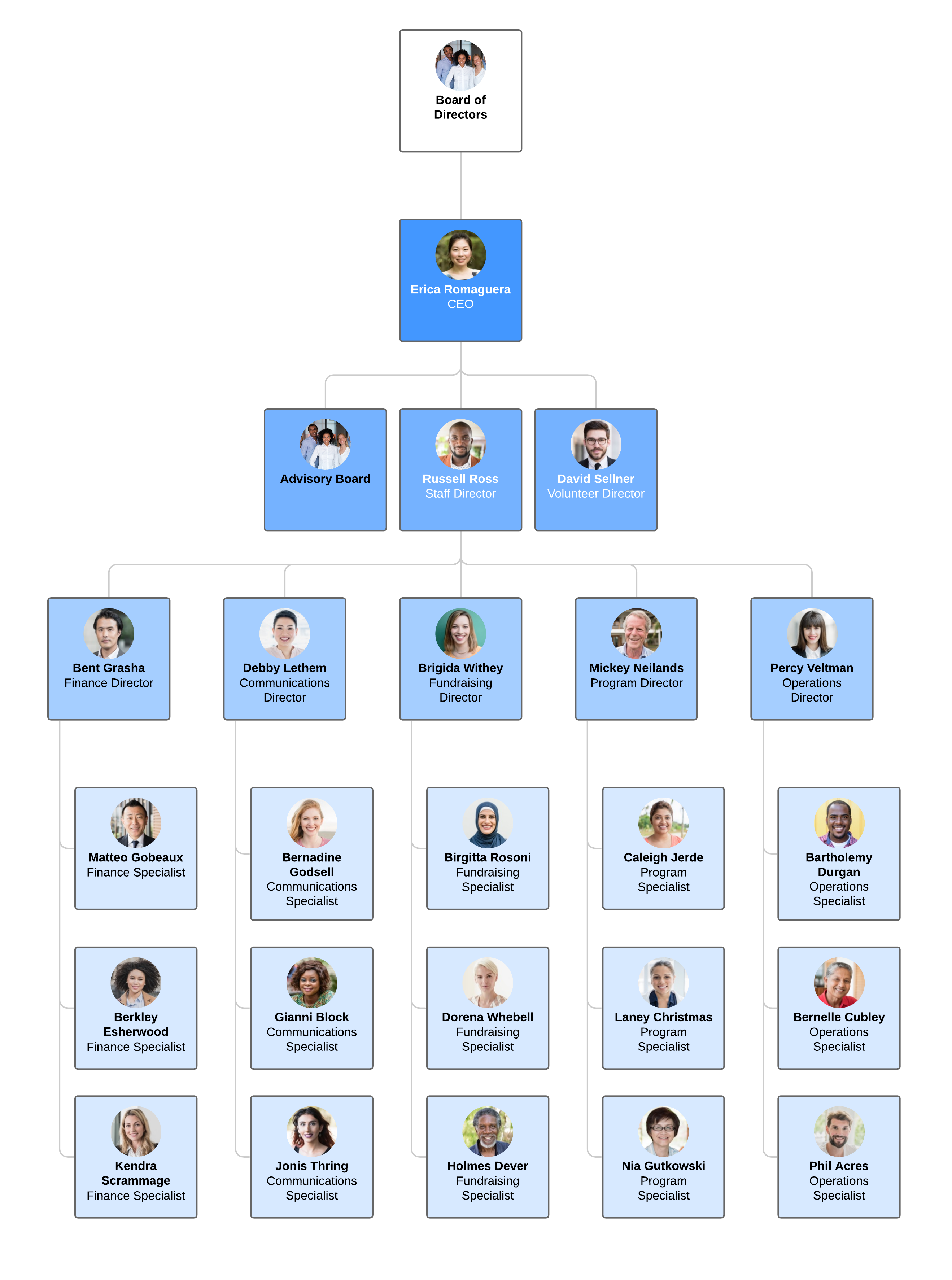
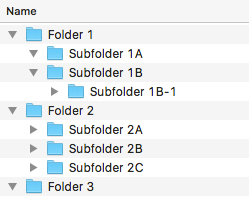
2-D Array

3-D Array

**Tree**

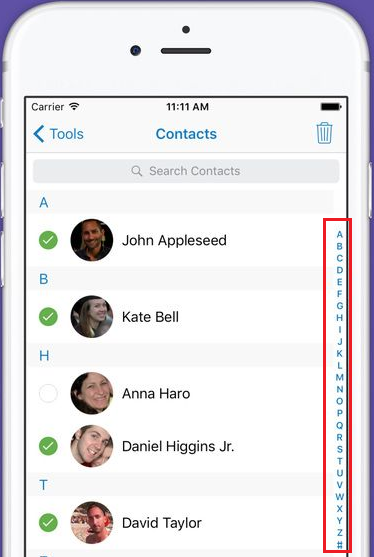
Trees are used for representing tree-like structures, such as folder structure, organization structure, web site menu structure, DOM structure.

Tree Structure



**Hash**

Hash tables are used for representing dictionary-like structures, such as dictionary, contact list in a phone book app.



A

B

C

D

…

X

Y

Z

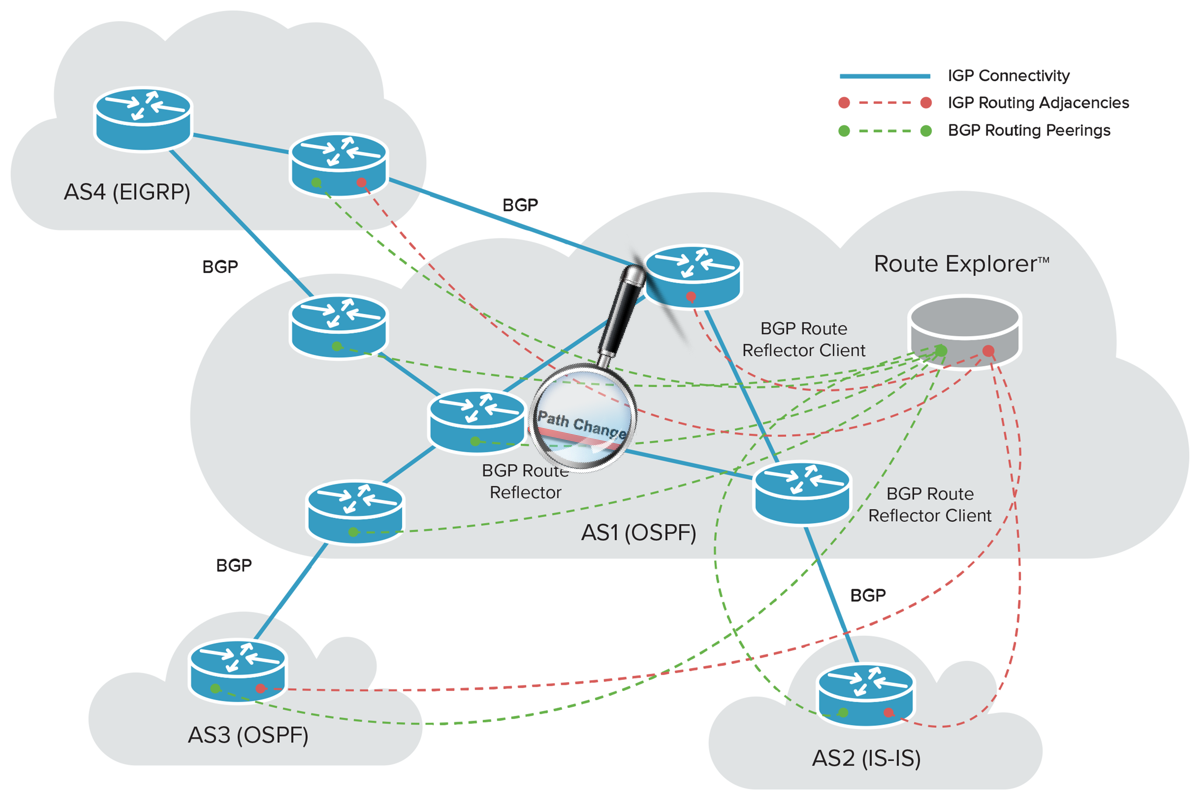
#

Hash Structure

**Graph**

Graphs are used for representing networks (both real and electronic), such as networked computers, streets on a map or social media relationships.

Graph Structure



In this tutorial, you will study the concepts of “array” and “list”. And how to use array and list in C#.

1. **Array in programming**

Consider a situation where we need to store five integer numbers. If we use programming's simple variable and data type concepts, then we need five variables of int data type and the program will look like this

Text

Description automatically generated

Well, creating 5 variables are not too bad. But how about creating 5000 variables for 5000 integer numbers. And how about x number of integers, where x is a dynamic number in the program?

To handle such situations, almost all the programming languages provide a concept called array. An array is a data structure, which can store a **fixed-size** collection of elements of the same data type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the **same type**. Each individual data in an array can be accessed using its **index**.

For the example above, ***X*** is an array of 5000 integer numbers - ***X1****,* ***X2****,* ***X3****, … ,* ***X4999****,* ***X5000***.

Moreover, array can be used to describe a collection of data of 2-dimension, 3-dimension, and so on. For example,

* ***Y*** is 2-D array represents all grids on a chess board, and ***Ym,n*** is the grid in row ***m*** and column ***n***.

Graphical user interface

Description automatically generated

* A Rubik's Cube can be described using a 3-D array ***Z***, and the pieces can be accessed by ***Zi,j,k***.

A picture containing toiletry, cosmetic

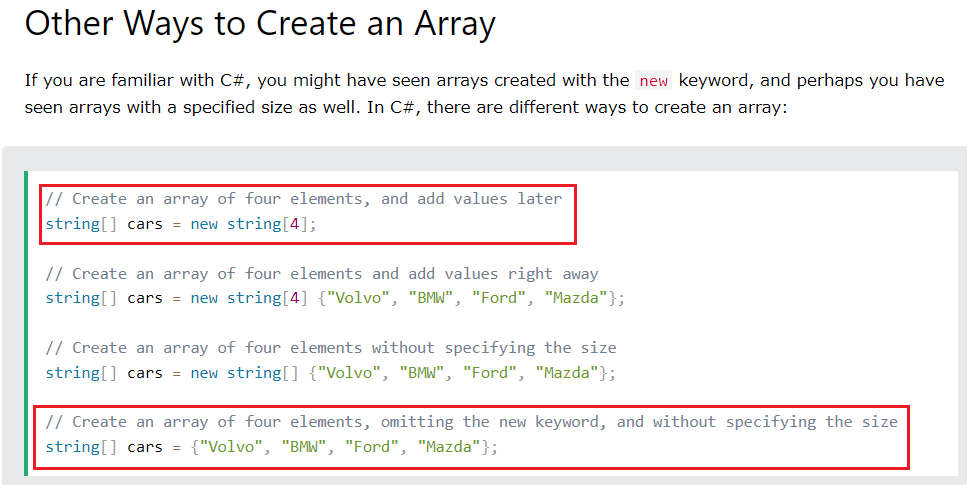
Description automatically generated

1. **C# Array (1-D)**

Study the [C# Arrays](https://www.w3schools.com/cs/cs_arrays.php) tutorial on w3school.com for how to create and operate an (1-D) array in C#. Here are the takeaway notes

* The size of an array is fixed in declaration and cannot be altered later in the program.
* An item in an array can be accessed using its index.
* In the programming world, array indexes start with 0: [0] is the first element. [1] is the second element.
* “**for**” and “**foreach**” are the best loop syntax to iterate an array. You can of course use “**while**”, but not as handy as the “**for**” and “**foreach**”.

Pay special attention to this section in the w3schools tutorial



It is recommended to use method 1 and 4 only to make your code clean and do not use method 2 and 3. We will talk more about the key word “**new**” in the future tutorials.

**Class activity 1**

For a given array of numbers, calculate the sum and average of all values.

{10, 13.2, 14.3, 11, 9.89, 12.98, 15.76, 11.6, 14.21, 10.89, 8, 7.1}

|  |
| --- |
| Your codes |
| Test results |

**Class activity 2**

For the array in the previous activity, calculate the 3-point moving average, and store the results in a new array.

Hint: if the size of original array is n, then the size of new array is n-2.

**Class activity 3**

For the array in the previous activity, search for all numbers greater than a given value, and display all of them.

**Class activity 4**

The Fibonacci Sequence is the series of numbers:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

The next number is found by adding up the two numbers before it:

* the 2 is found by adding the two numbers before it (1+1),
* the 3 is found by adding the two numbers before it (1+2),
* the 5 is (2+3),
* and so on!

You are required to found and store the first 20 Fibonacci number and store them in an array. The program has been started, and you may use the code below and complete the program.

|  |
| --- |
| static void Main(string[] args)  {  // array F is used to store the first 20 Fibonacci numbers  int[] F = new int[20];  F[0] = 0;  F[1] = 1;  // you code to generate F[2] to F[19]  // display the whole array  Console.WriteLine("The first 20 Fibonacci numbers:");  foreach(int f in F)  {  Console.Write(f + ", ");  }  } |

|  |
| --- |
| Your codes |
| Test results |

**Class activity 5**

Create a program that allow user to enter information for several clients. Each client has a name, phone number, and email address. Here is the workflow of the program:

* User enters the number of clients
* The program creates three arrays for storing the name, phone, and email information
* User enters the information for all clients
* After all client information has been entered, the program displays the information of all clients

|  |
| --- |
| **Your codes** |
| **The results** |

1. **List in programming**

List (or linked-list) is another way of representing 1 dimensional data. Watch the following video to understand the differences between **array** and **list**.

<https://www.youtube.com/watch?v=Muo0UdwESn4>

Also study the following article to understand the limitations and advantages of both array and list

<https://www.geeksforgeeks.org/linked-list-vs-array/>

Here are the takeaway notes

Table

Description automatically generated

1. **C# List**

Use the following tutorial to study how to use list in C#.

<https://www.c-sharpcorner.com/article/c-sharp-list/>

**Class activity 6**

Below are the codes of a mini cli program. Extend it to build a “to-do list” program, with the following functions

* The program maintains two list: **to-do** and **done**
* Command “**add**” allow user to append a to-do item at the end of the to-do list
* Command “**done**” allow user to specify a to-do item index, and move the item from the **to-do** list to the **done** list
* Command “**show**” display both **to-do** and **done** list

|  |
| --- |
| static void Main(string[] args)  {  bool exit = false;  while (!exit)  {  Console.WriteLine("Enter a command");  string command = Console.ReadLine();  switch (command)  {    case "exit":  exit = true;  break;  default:  Console.WriteLine("invalid input! try again:");  break;  }  }  Console.WriteLine("Byebye");  } |

|  |
| --- |
| **Your codes** |
| **The results** |

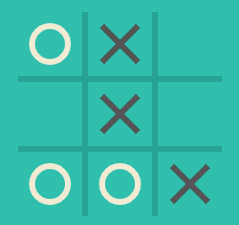
1. **C# multidimensional arrays**

Use the following article to study how to use C# 2-D and 3-D array.

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/arrays/multidimensional-arrays>

**Challenge**

Develop a CLI based tic-tac-toe game

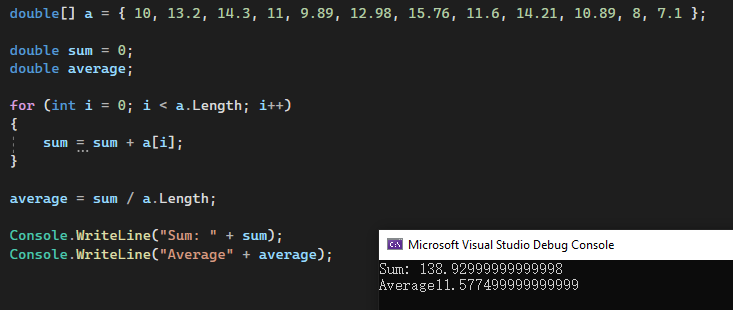


Hint: the codes below define, initiate and display a game.

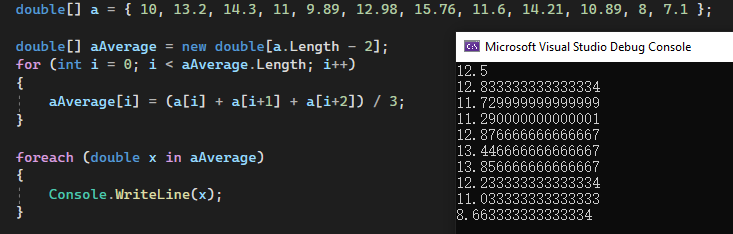


**Model answers**

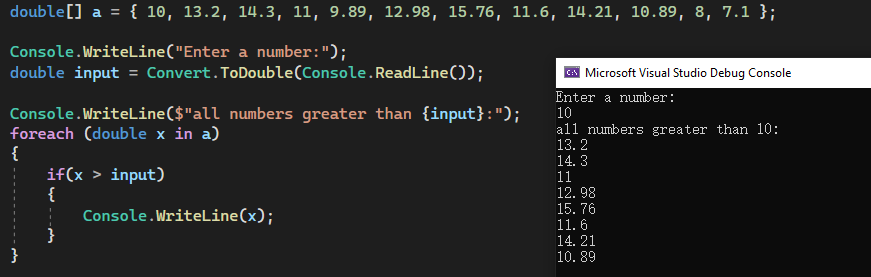
**Class activity 1**



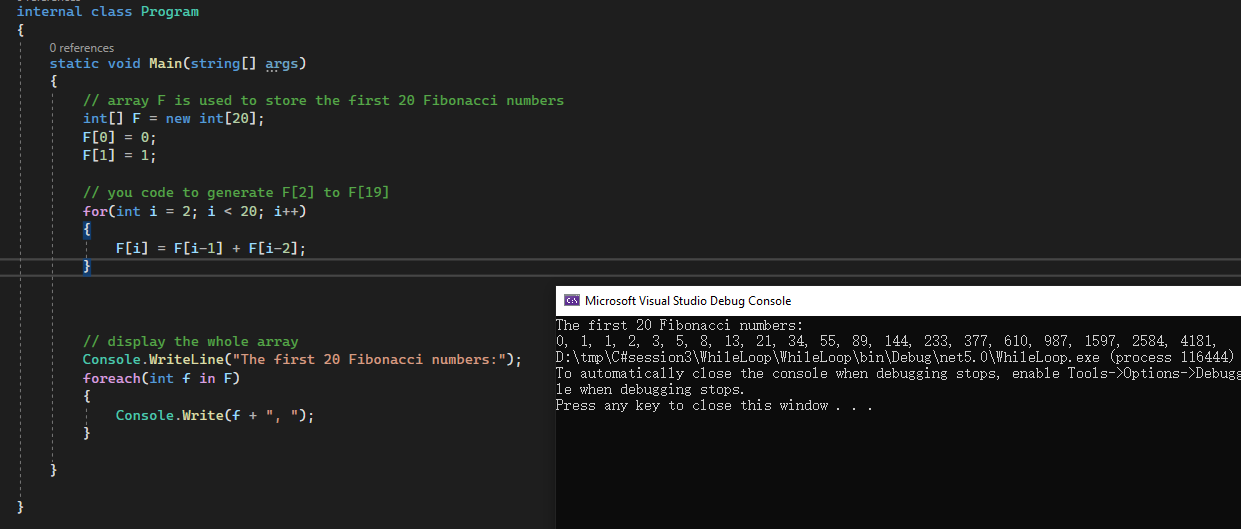
**Class activity 2**



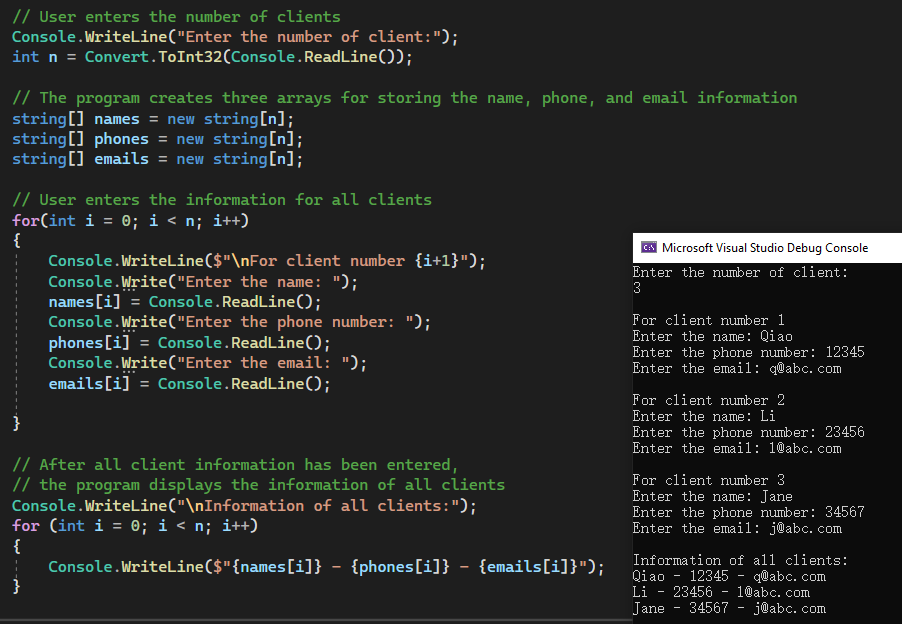
**Class activity 3**



**Class activity 4**

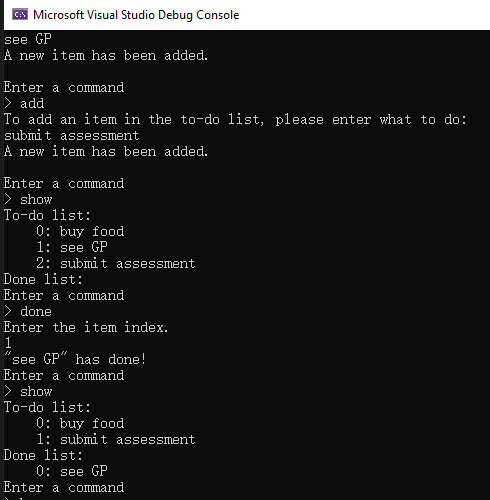


**Class activity 5**



**Class activity 6**

|  |
| --- |
| static void Main(string[] args)  {  bool exit = false;  List<string> todo = new List<string>();  List<string> done = new List<string>();  while (!exit)  {  Console.WriteLine("Enter a command");  Console.Write("> ");  string command = Console.ReadLine();  switch (command)  {  case "add":  Console.WriteLine("To add an item in the to-do list, please enter what to do:");  string newItem = Console.ReadLine();  todo.Add(newItem);  Console.WriteLine("A new item has been added.\n");  break;  case "done":  Console.WriteLine("Enter the item index.");  int index = Convert.ToInt32(Console.ReadLine());  string itemToMove = todo[index];  todo.RemoveAt(index);  done.Add(itemToMove);  Console.WriteLine($"\"{itemToMove}\" has done!");  break;  case "show":  Console.WriteLine("To-do list:");  int i = 0;  foreach(string item in todo)  {  Console.WriteLine($" {i}: {item}");  i++;  }  Console.WriteLine("Done list:");  i = 0;  foreach (string item in done)  {  Console.WriteLine($" {i}: {item}");  i++;  }  break;  case "exit":  exit = true;  break;  default:  Console.WriteLine("invalid input! try again:");  break;  }  }  Console.WriteLine("Byebye");  } |



End of this tutorial