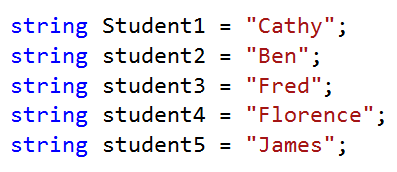
# Chapter 5 – Arrays

## What is an array?

An array is a type of data structure. Arrays are more complex than individual variables as they can store multiple values in indexed locations.

## The need for arrays in programming

There is frequently a need when designing programs to store multiple related values; for example, a teacher wishes to store the names of the students in a Computer Science class. Without the availability of arrays, the students’ names would have to be stored in individual variables (as below), which can get quite cumbersome when you have a lot of values to be stored.



The most efficient method of storing the names is in a **string array**.

## How is an array structured?

A one-dimensional array is structured into indices with each index storing a separate value. These values can be accessed and changed using the index number. It is common for indices to be numbered starting with 0.

So, our student name example would be structured like the table below:

**Array StudentNames**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Index [0] | Index [1] | Index [2] | Index [3] | Index [4] |
| “Cathy” | “Ben” | “Fred” | “Florence” | “James” |

If Cathy leaves the class and is replaced by Sarah, then the appropriate index would be accessed like this:

**StudentNames[0] = “Sarah”**

This would **overwrite** the value in index [0] of the student names array.

If Cathy leaves the class and is not replaced, then the value in index [0] would be replaced with an empty string like below:

**StudentNames[0] = “”**



Written Task – Accessing Arrays

An array was created to store the results of a test. Answer the questions below.

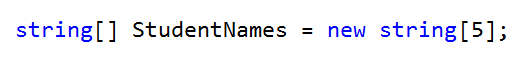
**Array TestMarks**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| [0] | [1] | [2] | [3] | [4] | [5] | [6] |
| 66 | 34 | 56 | 13 | 45 | 72 | 38 |

1. What is the value contained in index [6]?
2. How many indices does this array have?
3. What is the index number for the value 45?
4. After a test remark, the value in index [2] is changed to 61. Write a statement to change the value.

## Declaring an array in C#

Arrays are declared like so:



1. **Array datatype** – in this example, the datatype of the index values will be strings, but any of the other datatypes (mentioned in Chapter 2) could be used. In C# it is not possible to store different datatypes together in the same array, although it is possible to create a record of different datatypes and then create an array of records (see further on in this chapter).
2. **Array identifier** – this is the name we give to the array.
3. **Size** – **new string [5]** creates a new empty string array of five indices: ([0], [1], [2], [3], [4]).

## Initialising an array in C#

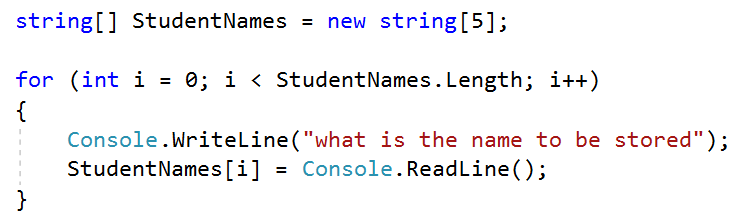
Like a variable, an array can be declared and initialised at the same time. See example below:



The values are placed in curly brackets separated by commas at the end of the declaration statement for the array. In this case there is no need to state the number of indices to be created as it will create an index for each stated value.

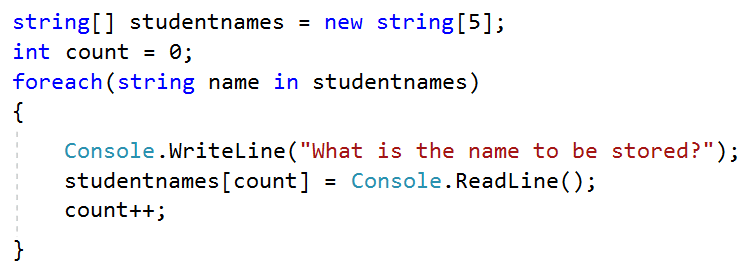
Alternatively, values can be passed into the array either all at once using a loop the same size as the array (example 1), a for each loop (example 2) or individually (example 3).

### Example 1 – populating an array with user input using a loop

In this example, the for loop is set to iterate one less than the length of the array (assuming the first array element has the value of 0). The length of the array can be found by accessing the **.Length** property of the array object.

### Example 2 – populating an array with a FOR EACH Loop

A **for each** loop is a special type of loop that can be used specifically for looping through all the different elements of an array.



### Example 3 – populating array indices separately

If rather than populating an array all at once we want to just access a single element, then we use an assignment statement. In this example, the fourth element of the array is being assigned the value of ‘Mary’.



## Accessing an array in C#

Once an array has been declared and initialised, the array can be accessed for assignment purposes or for the checking of conditions in iteration and selection statements using the three methods above.

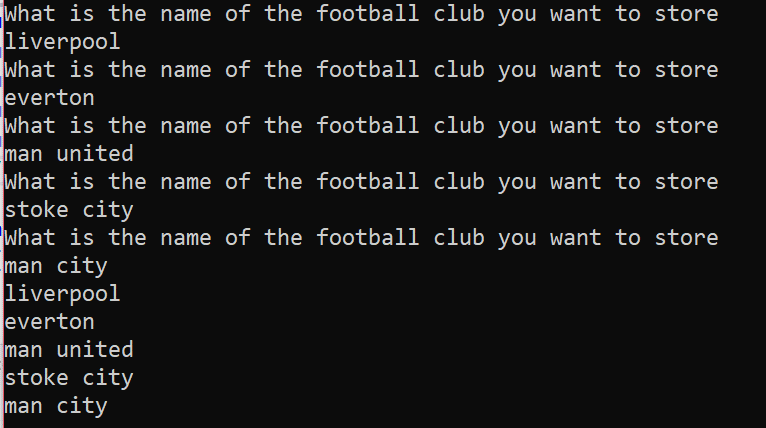


**{ }**

TIP – use a for loop or a for each loop if you want to access each element in turn.

Coding Task: *Program 5.1*

Create a program that inputs and stores football club names in an array of five elements and then   
outputs the contents of the array to the screen.



## Searching and sorting an array in C#

There are two operations which are commonly performed on arrays: searching and sorting.

A simple method of finding out whether a value is in the array is to use a linear search where every value is accessed in turn to see if whether is the value being sought. A more efficient search can be done using the **.BinarySearch** method (see below) on an already sorted list.

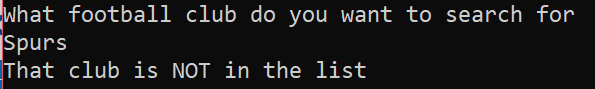


**{ }**

Arrays can be easily sorted using the **.Sort** method (see below).

Coding Task: *Program 5.2*

Adapt program 5.1 so that the program asks for a particular football club and then checks the list   
to see whether that club is in the list or not. Use a linear search first (check every element against the one you are searching for) and then try using a binary search. Remember that with a binary search you must sort the list first.



## Useful array methods and properties

The array object has a number of useful methods and properties that allow for certain operations to be done on the array contents.

|  |  |  |
| --- | --- | --- |
| **Method/property** | **Used for** | **Syntax** |
| **.Length** | Finding the number of elements in the array | **Clubs.Length** |
| **.Sort()** | Sorting the elements into order | **Array.Sort(Clubs)** |
| **.Reverse()** | Reverses the original order of the elements | **Array.Reverse(Clubs)** |
| **.BinarySearch()** | An efficient search to be used on a sorted list. Finds the specified element and returns the index value of the element. | **Array.BinarySearch(Clubs, clubname)** |



Written Task

Find two other methods that can be used on an **array**.

## Two-dimensional arrays

A two-dimensional (2D) array is used to create a table of data in rows and columns with the same data type. An example of this is an array created to show the state of a noughts and crosses game board.

|  |  |  |  |
| --- | --- | --- | --- |
| Array index | [0] | [1] | [2] |
| [0] | **X** | **O** | **X** |
| [1] | **X** | **X** | **X** |
| [2] | **O** | **X** | **O** |

A 2D array would be **declared** like so:



**Initialising a 2D array** 

## Accessing each element in a 2D array

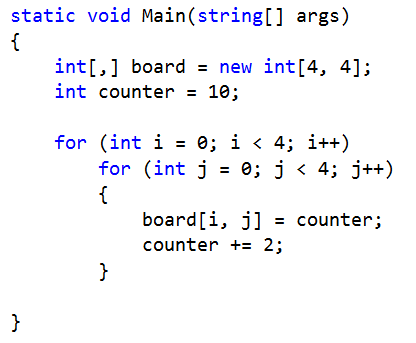
This code is populating a 2D integer array with a set of numbers starting at 10 and going up by two for each new number.

The outer **for** loop (i) is looping through the **rows** in the 2D array. The inner loop (j) is looping through the **columns** in each row.

Each element in the array is accessed by the assignment statement **board[i, j] =**

When both loops are completed the array will look like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | [0] | [1] | [2] | [3] |
| [0] | 10 | 12 | 14 | 16 |
| [1] | 18 | 20 | 22 | 24 |
| [2] | 26 | 28 | 30 | 32 |
| [3] | 34 | 36 | 38 | 40 |





Written Task: *Accessing 2D Arrays*

An array was created to store the results of an exam of two papers. Answer the questions below.

**Array TestMarks**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Student [0] | Student [1] | Student [2] | Student [3] | Student [4] | Student [5] |
| Paper [0] | 34 | 56 | 13 | 45 | 72 | 38 |
| Paper [1] | 66 | 91 | 56 | 78 | 75 | 49 |

1. What is the value contained in TestMarks [1,5]?
2. How many values can be stored in this array?
3. What is the index reference for the value 45?
4. After a test remark, the value in index [1,1] is changed to 94. Write a statement to change the value.



**{ }**

Coding Task: *Program 5.3 – Outputting a 2D Array to a Grid*

Investigate how to output the 2D array below, showing the state of a game board to the console   
window so that it outputs a grid with the array elements in each section of the grid.

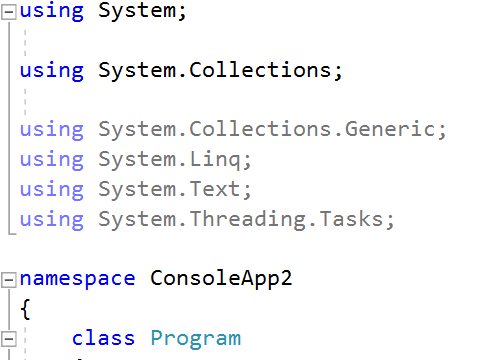
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **[0]** | **[1]** | **[2]** | **[3]** | **[4]** | **[5]** |
| **[0]** | **“Patrol”** |  |  |  |  |  |
| **[1]** |  |  |  | **“Ship”** |  |  |
| **[2]** |  |  |  |  |  | **“Carrier”** |
| **[3]** |  | **“Destroyer”** |  |  |  | **“Carrier”** |
| **[4]** |  | **“Destroyer”** |  |  |  | **“Carrier”** |
| **[5]** |  |  |  |  |  |  |

## Dynamic arrays

In C# when we want a dynamic array (an array that grows automatically when new elements are added) we can use an object called an **ArrayList**.

### Declaring an ArrayList

Firstly **System.Collections** needs to be added to the collections above the namespace.



Then a new **ArrayList** can be declared in your program.



There are a number of useful methods and properties that can be used on the list:

|  |  |  |
| --- | --- | --- |
| **Method/property** | **Used for** | **Syntax** |
| **.Count** | Counting up the number of elements in a list. This allows access to the Count property of the ArrayList object. | **numstudents = students.Count** |
| **.Add()** | Adding an element to the end of a list | **students.Add(“Mary”)** |
| **.Remove()** | Removing a specified element from the list | **students.Remove(“Mary”)** |
| **.RemoveAt()** | Removing an element from a specified index in the list | **students.RemoveAt(0)** |
| **.Sort()** | Sorts the list in order | **students.Sort()** |
| **.Reverse()** | Reverses the order of the list | **students.Reverse()** |
| **.Contains()** | Checks whether a specified value is in the list | **students.Contains(item)** |



Written Task

Find two other methods that can be used on an **ArrayList**.



**{ }**

Coding Task: *Program 5.4 – Charity Lottery*

Create a program that will help a charity to run its own lottery.

Players can either input their own numbers (five numbers between 1 and 60) or ask the program to generate a lucky-dip set of numbers (five numbers between 1 and 60). If a customer’s set of numbers contains a multiple of 10 (10, 20, 30, 40, 50, or 60) then they can have an additional raffle number (this number should be randomly generated between 1 and 60).

Once a player’s numbers have been input or generated, their line of numbers should be sorted into order and output to the player.

## Structs and arrays of records

### What is a record?

A record is a collection of data items about an object, a person or a thing.

For example, a teacher holds details about her students and their exam marks. Each item is stored in a column or a **field** and all the items for a particular student are stored in one **row or record**.

This is one record.

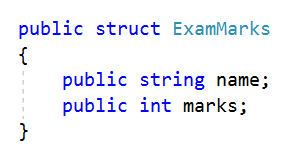
|  |  |
| --- | --- |
| **name** | **marks** |
| **Lucy** | **23** |
| **Ben** | **45** |
| **David** | **62** |
| **Louise** | **49** |

We could consider storing this information in a 2D array, but this is not possible in C# because we cannot store multiple data types together in the same array and here we would need string for the name and integer for the marks.

### What is a structure?

In C# we refer to a record as a structure (using the keyword **struct**). Structures allow us to mix data types and then we can store multiple records in an array (of records).

### Declaring a structure for a single record



This declares the individual values and data types for a single record. This is then known as a user-defined data type.

### Declaring a single record object



This object will hold a single record.

### Assigning values to a single record object



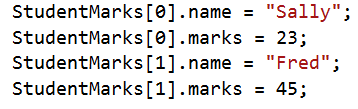
Each value is accessed separately using objectname.value

### Declaring an array of records

This declares an array of data type ExamMarks to hold two records.



### Assigning/accessing values in an array of records





**{ }**

Each record is accessed using its index number after the array identifier.

Coding Task: *Program 5.5 – Superheroes Cards*

A toy company is creating an electronic card game for young children that will feature superheroes.

**The following values need to be stored:**

* Superhero name (string)
* Lifespan (integer)
* Skill (string)
* Strength (integer)
* Speed (real)

**Your program should:**

* Store the names of at least four superheroes
* Allow for the input of the superheroes
* Output the input superheroes from the array to the console window

## Chapter 5 – Consolidation Tasks

Program 5.6

Load the partially built **Program 5.6.txt** into a new console mode application.

At the moment the program contains an empty board array and a procedure that outputs the contents of the board array to the console window.

You are to add more functionality to the Fruits program by writing code that will:

* Create a **struct** for the fruit record

|  |  |  |
| --- | --- | --- |
| **Struct FRUIT** | **Data Type** | **Example Value** |
| FRUITIDENTIFIER | Char | “A” |
| FRUITNAME | String | “Apple” |
| FRUITVALUE | Integer | 10 (randomly generated) |

* Create an **array of seven fruits**
* **Randomly generate coordinates for the fruit identifier** (char) to be placed on the game board
* Write code that will enable a player to **inspect a fruit** by entering the coordinates of the fruit on the game board

The console window below shows the required output.



**Evidence required:**

1. **An annotated code listing** for the required extensions to the program

2. **Screenshots of testing** carried out that prove the required extensions work

### Extension challenge

Add further code that will total up the fruit values and output a score to the console window.

Written Tasks

**Accessing arrays**

1. 38
2. 7 (0 to 6)
3. 4
4. TestMarks[2] = 61;

**Array methods**

Array.Find(**array**, **predicate**)

Looks for the **predicate** in the given **array**

Array.Resize(ref **array**, **newSize**)

Resizes the given **array** to the size **newSize**

**Accessing 2D arrays**

1. 49
2. 12 (2 papers \* 6 students)
3. TestMarks[0,3]
4. TestMarks[1,1] = 94

**ArrayList methods**

**arrayList**.Insert(**index**, **value**)

Inserts the object (number or string or such) **value** into the given **index** of the named **arrayList**

**arrayList**.Clear()

Clears the given **arrayList** of all of its values

Consolidation Tasks

**Program 5.6**

Text

Description automatically generatedText

Description automatically generated

Text

Description automatically generated