**Module 2: Arrays & Strings in C#**

**Learning Objectives**

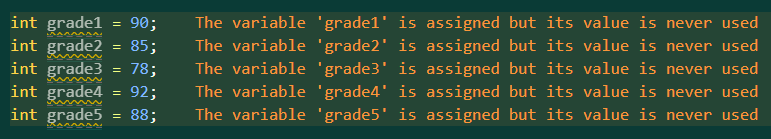
**By the end of this module, you should be able to:**

* **Explain what arrays and strings are in C#.**
* **Differentiate between 1D, 2D, and jagged arrays.**
* **Perform traversal, insertion, deletion, and searching on arrays.**
* **Understand how strings behave like character arrays.**
* **Apply arrays and strings to real-world problems such as managing student grades.**

**Introduction to Arrays**

**Why Arrays?**

**Imagine you want to store the grades of 5 students. Without arrays, you’d write:**

****

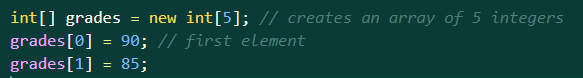
**This is inefficient and hard to manage.**

**An array is a collection of items stored under one name, where each item is identified by an index number.**

* **Arrays are fixed size. Once you declare the number of elements, it cannot be changed.**
* **Arrays are zero-indexed: the first element is at index 0.**

**Declaring an Array**

**datatype[] arrayName = new datatype[size];**

**Example:**

**One-Dimensional (1D) Arrays**

**A 1D array is a simple list of values stored in a straight line (like lockers in a hallway).**

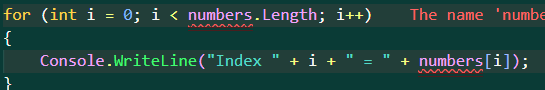
**Example:**

**int[] numbers = {10, 20, 30, 40, 50};**

**Visual Representation**

| **Index** | **0** | **1** | **2** | **3** | **4** |
| --- | --- | --- | --- | --- | --- |
| **Value** | **10** | **20** | **30** | **40** | **50** |

**Traversing a 1D Array**

****

**Learning Point:**

* **.Length gives the number of elements.**
* **Arrays let us store multiple values without creating separate variables.**

**Two-Dimensional (2D) Arrays**

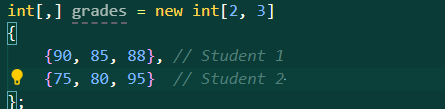
**A 2D array is like a table with rows and columns.**

* **The first index = row number.**
* **The second index = column number.**

**Syntax:**

**datatype[,] arrayName = new datatype[rows, columns];**

**Example:**

****

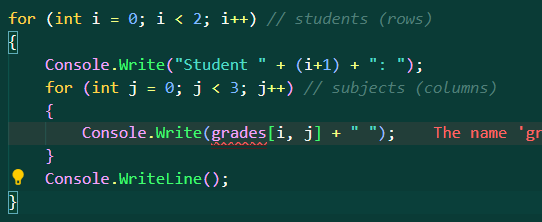
**Visual Representation**

**This stores grades for 2 students across 3 subjects:**

| **Student ↓ / Subject →** | **Subject 1** | **Subject 2** | **Subject 3** |
| --- | --- | --- | --- |
| **Student 1** | **90** | **85** | **88** |
| **Student 2** | **75** | **80** | **95** |

* **grades[0,0] = 90 → Student 1, Subject 1**
* **grades[0,1] = 85 → Student 1, Subject 2**
* **grades[1,2] = 95 → Student 2, Subject 3**

**Traversing a 2D Array**

****

**Output:**

**Student 1: 90 85 88**

**Student 2: 75 80 95**

**Learning Point:  
Use 2D arrays for tabular data like student marks, seating charts, or multiplication tables.**

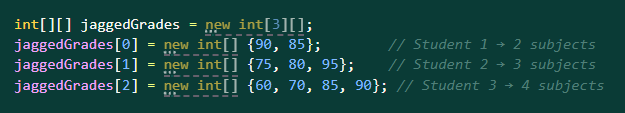
**Jagged Arrays**

**A jagged array is an array of arrays, where each row can have a different number of elements.**

**Syntax:**

**datatype[][] arrayName = new datatype[rows][];**

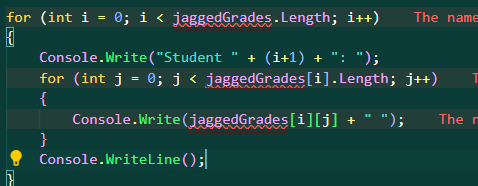
**Example:**

****

**Visual Representation**

| **Student** | **Grades** |
| --- | --- |
| **1** | **90, 85** |
| **2** | **75, 80, 95** |
| **3** | **60, 70, 85, 90** |

**Traversing a Jagged Array**

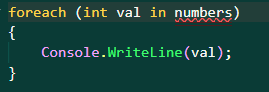
****

**Learning Point:  
Jagged arrays are flexible — good when each row of data is of different length (e.g., students with different number of subjects).**

**Operations on Arrays**

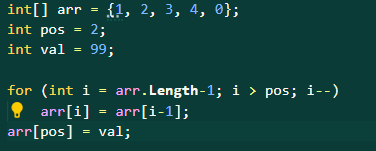
1. **Traversal**

**Visiting every element.**

****

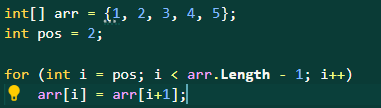
**(b) Insertion**

**To insert into an array, shift elements to make space.**

****

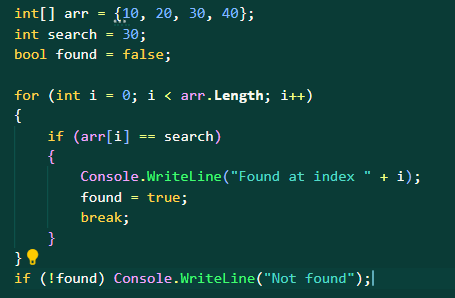
**(c) Deletion**

**shift elements left to overwrite the deleted value.**

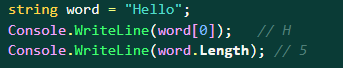
****

**(d) Searching**

**Find an element by checking each index.**

****

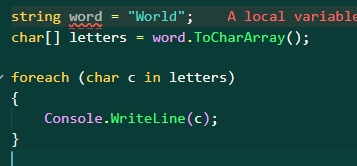
**Learning Point:  
Arrays are fixed size → you must shift elements when inserting or deleting.**

****

**Strings in C#**

**A string is a sequence of characters.  
In C#, strings are objects, but they behave like character arrays.**

**Example:**

****

**Strings as Character Arrays**

**Learning Point:**

* **Strings are immutable → you can’t directly change a character.**
* **To modify, convert into a char[], make changes, then rebuild the string.**

**Mini Project: Student Grades Manager**

**Goal: A program that allows input of grades, displays them, calculates average, and searches for a grade.**

**Learning Point:  
This ties everything together:**

* **Array stores grades.**
* **Traversal displays grades.**
* **Calculation uses loops.**
* **Searching demonstrates finding a grade.**