using System;

namespace EXAMPLE\_1

{

class Program

{

static void Main(string[] args)

{

#region 1. Invalid Example

// ❌ Invalid array creation (Uncommenting below line will cause Error CS1586)

// int[] intArray = new int[];

#endregion

#region 2. Array Declaration & Initialization

int[] intArray = new int[6]; // Declared with fixed size (default values will be 0)

string[] stringArray = { "Thillai", "Tamizh", "Shanmugam" }; // Declared and initialized directly

#endregion

#region 3. Assigning Values to Array

intArray[0] = 0;

intArray[1] = 1;

intArray[2] = 2;

// Assign values using a for loop (from index 3 to end)

for (int i = 3; i < intArray.Length; i++)

{

intArray[i] = i;

}

#endregion

#region 4. Accessing Values without Loop

Console.WriteLine($"stringArray[0] = {stringArray[0]}");

Console.WriteLine($"stringArray[1] = {stringArray[1]}");

#endregion

#region 5. Accessing Values using For Loop

for (int i = 0; i < intArray.Length; i++)

{

Console.WriteLine($"intArray[{i}] = {intArray[i]}");

}

#endregion

}

}

}

using System;

namespace ArrayFunctionsDemo

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("=== ARRAY FUNCTIONS DEMO ===\n");

#region 1. Length

Console.WriteLine("Length → Gets the total number of elements in the array.");

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

Console.WriteLine("Output: " + numbers.Length + "\n");

#endregion

#region 2. GetLength, GetLowerBound, GetUpperBound

Console.WriteLine("GetLength → Gets the number of elements in the specified dimension.");

Console.WriteLine("GetLowerBound → Gets the smallest valid index of a dimension.");

Console.WriteLine("GetUpperBound → Gets the largest valid index of a dimension.");

int[,] matrix = new int[3, 5]; // 3 rows, 5 cols

Console.WriteLine("Output: GetLength(0) = " + matrix.GetLength(0)); // rows

Console.WriteLine("Output: GetLength(1) = " + matrix.GetLength(1)); // cols

Console.WriteLine("Output: GetLowerBound(0) = " + matrix.GetLowerBound(0));

Console.WriteLine("Output: GetUpperBound(1) = " + matrix.GetUpperBound(1) + "\n");

#endregion

#region 3. IndexOf & LastIndexOf

Console.WriteLine("IndexOf → Finds the first occurrence of a value.");

Console.WriteLine("LastIndexOf → Finds the last occurrence of a value.");

int[] arr1 = { 10, 20, 30, 20, 40, 20 };

Console.WriteLine("Output: IndexOf(20) = " + Array.IndexOf(arr1, 20));

Console.WriteLine("Output: LastIndexOf(20) = " + Array.LastIndexOf(arr1, 20) + "\n");

#endregion

#region 4. Sort

Console.WriteLine("Sort → Sorts the elements of the array in ascending order.");

int[] arr2 = { 5, 3, 8, 1, 2 };

Array.Sort(arr2);

Console.WriteLine("Output: " + string.Join(", ", arr2) + "\n");

#endregion

#region 5. Reverse

Console.WriteLine("Reverse → Reverses the order of elements in the array.");

int[] arr3 = { 1, 2, 3, 4, 5 };

Array.Reverse(arr3);

Console.WriteLine("Output: " + string.Join(", ", arr3) + "\n");

#endregion

#region 6. Clear

Console.WriteLine("Clear → Sets a range of elements in the array to the default value (0, null, false).");

int[] arr4 = { 1, 2, 3, 4, 5 };

Array.Clear(arr4, 1, 2); // clear index 1 & 2

Console.WriteLine("Output: " + string.Join(", ", arr4) + "\n");

#endregion

#region 7. Resize

Console.WriteLine("Resize → Changes the size of the array (creates a new one internally).");

int[] arr5 = { 1, 2, 3 };

Array.Resize(ref arr5, 5);

Console.WriteLine("Output: " + string.Join(", ", arr5) + "\n");

#endregion

#region 8. Copy

Console.WriteLine("Copy → Copies elements from one array to another.");

int[] src = { 1, 2, 3 };

int[] dest = new int[3];

Array.Copy(src, dest, 3);

Console.WriteLine("Output: " + string.Join(", ", dest) + "\n");

#endregion

#region 9. Clone

Console.WriteLine("Clone → Creates a shallow copy of the array.");

int[] clone = (int[])src.Clone();

Console.WriteLine("Output: " + string.Join(", ", clone) + "\n");

#endregion

#region 10. Exists

Console.WriteLine("Exists → Checks if any element matches a condition (uses Predicate).");

int[] arr6 = { 1, 2, 3, 4, 5 };

bool exists = Array.Exists(arr6, n => n > 3);

Console.WriteLine("Output: " + exists + "\n");

#endregion

#region 11. Find

Console.WriteLine("Find → Returns the first element that matches a condition.");

int first = Array.Find(arr6, n => n > 3);

Console.WriteLine("Output: " + first + "\n");

#endregion

#region 12. FindAll

Console.WriteLine("FindAll → Returns all elements that match a condition.");

int[] all = Array.FindAll(arr6, n => n > 2);

Console.WriteLine("Output: " + string.Join(", ", all) + "\n");

#endregion

#region 13. FindIndex

Console.WriteLine("FindIndex → Returns the index of the first element that matches a condition.");

int idx = Array.FindIndex(arr6, n => n % 2 == 0);

Console.WriteLine("Output: " + idx + "\n");

#endregion

#region 14. TrueForAll

Console.WriteLine("TrueForAll → Checks if all elements match a condition.");

bool allPositive = Array.TrueForAll(arr6, n => n > 0);

Console.WriteLine("Output: " + allPositive + "\n");

#endregion

#region 15. ForEach

Console.WriteLine("ForEach → Performs an action on each element of the array.");

Console.Write("Output: ");

Array.ForEach(arr6, n => Console.Write(n + " "));

Console.WriteLine("\n");

#endregion

}

}

}

/\*

\* ✅ Summary:

Addition, Subtraction, Division → element-wise ✔️

Multiplication → dot-product rule ✔️

\*/

using System;

namespace EXAMPLE\_3

{

class Program

{

static void Main(string[] args)

{

// Matrix A (2x2)

int[,] A = { { 1, 2 }, { 3, 4 } };

// Matrix B (2x2)

int[,] B = { { 1, 5 }, { 5, 10 } };

Console.WriteLine("=== Matrix A ===");

PrintMatrix(A);

Console.WriteLine("=== Matrix B ===");

PrintMatrix(B);

// 1. Matrix Addition

Console.WriteLine("=== Matrix Addition (A + B) ===");

int[,] add = AddMatrices(A, B);

PrintMatrix(add);

// 2. Matrix Multiplication

Console.WriteLine("=== Matrix Multiplication (A x B) ===");

int[,] mul = MultiplyMatrices(A, B);

PrintMatrix(mul);

// 3. Matrix Division (Element-wise)

Console.WriteLine("=== Matrix Division (A ÷ B) ===");

double[,] div = DivideMatrices(A, B);

PrintMatrix(div);

}

// 🔹 Function: Print int matrix

static void PrintMatrix(int[,] matrix)

{

for (int i = 0; i < matrix.GetLength(0); i++)

{

for (int j = 0; j < matrix.GetLength(1); j++)

{

Console.Write(matrix[i, j] + "\t");

}

Console.WriteLine();

}

Console.WriteLine();

}

// 🔹 Function: Print double matrix

static void PrintMatrix(double[,] matrix)

{

for (int i = 0; i < matrix.GetLength(0); i++)

{

for (int j = 0; j < matrix.GetLength(1); j++)

{

Console.Write(matrix[i, j] + "\t");

}

Console.WriteLine();

}

Console.WriteLine();

}

// 🔹 Matrix Addition

static int[,] AddMatrices(int[,] A, int[,] B)

{

int rows = A.GetLength(0);

int cols = A.GetLength(1);

int[,] result = new int[rows, cols];

for (int i = 0; i < rows; i++)

for (int j = 0; j < cols; j++)

result[i, j] = A[i, j] + B[i, j];

return result;

}

// 🔹 Matrix Multiplication

static int[,] MultiplyMatrices(int[,] A, int[,] B)

{

int rowsA = A.GetLength(0);

int colsA = A.GetLength(1);

int rowsB = B.GetLength(0);

int colsB = B.GetLength(1);

if (colsA != rowsB)

throw new Exception("Matrix multiplication not possible: columns of A != rows of B");

int[,] result = new int[rowsA, colsB];

for (int i = 0; i < rowsA; i++)

{

for (int j = 0; j < colsB; j++)

{

int sum = 0;

for (int k = 0; k < colsA; k++)

sum += A[i, k] \* B[k, j];

result[i, j] = sum;

}

}

return result;

}

// 🔹 Matrix Division (Element-wise)

static double[,] DivideMatrices(int[,] A, int[,] B)

{

int rows = A.GetLength(0);

int cols = A.GetLength(1);

double[,] result = new double[rows, cols];

for (int i = 0; i < rows; i++)

{

for (int j = 0; j < cols; j++)

{

if (B[i, j] == 0)

throw new DivideByZeroException($"Division by zero at ({i},{j})");

result[i, j] = (double)A[i, j] / B[i, j];

}

}

return result;

}

}

}