Multi_Array

January 2, 2025

[2]:
```scala
// Initialize a 2D array
val matrix = Array(
  Array(10, 20, 30),
  Array(40, 50, 60),
  Array(70, 80, 90)
)

// Print the matrix dimensions
println(s"Rows: ${matrix.length}, Columns: ${matrix(0).length}")
matrix.foreach(row => println(row.mkString(", ")))
```

```
Rows: 3, Columns: 3
10, 20, 30
40, 50, 60
70, 80, 90
```

```
matrix = Array(Array(10, 20, 30), Array(40, 50, 60), Array(70, 80, 90))
```

[2]:
```
Array(Array(10, 20, 30), Array(40, 50, 60), Array(70, 80, 90))
```

[3]:
```scala
// Access specific elements
println(matrix(0)(1)) // Access the second element of the first row
println(matrix(2)(2)) // Access the last element
```

```
20
90
```

[4]:
```scala
// Define a 2D array with predefined dimensions
val grid: Array[Array[Int]] = Array.ofDim[Int](2, 3)

// Fill the array
for (i <- grid.indices; j <- grid(i).indices) {
  grid(i)(j) = i * j
}

// Print the filled grid
grid.foreach(row => println(row.mkString(", ")))
```

```
0, 0, 0
0, 1, 2

grid = Array(Array(0, 0, 0), Array(0, 1, 2))
```

[4]:
```
Array(Array(0, 0, 0), Array(0, 1, 2))
```

[5]:
```
// Initialize a 3D array
val cube = Array.ofDim[Int](2, 2, 3)

// Fill the 3D array
for (i <- 0 until 2; j <- 0 until 2; k <- 0 until 3) {
  cube(i)(j)(k) = i + j - k
}

// Print the 3D array
for (layer <- cube) {
  layer.foreach(row => println(row.mkString(", ")))
  println() // Add space between layers
}
```

```
0, -1, -2
1, 0, -1

1, 0, -1
2, 1, 0


cube = Array(Array(Array(0, -1, -2), Array(1, 0, -1)), Array(Array(1, 0, -1),␣
 ↪Array(2, 1, 0)))
```

[5]:
```
Array(Array(Array(0, -1, -2), Array(1, 0, -1)), Array(Array(1, 0, -1), Array(2,
1, 0)))
```

[6]:
```
// Dynamic 2D array with size 4x4
val size = 4
val dynamicGrid = Array.ofDim[Int](size, size)

// Fill the grid based on custom logic (e.g., diagonal elements as 1)
for (i <- 0 until size; j <- 0 until size) {
  dynamicGrid(i)(j) = if (i == j) 1 else 0
}

// Print the dynamic grid
dynamicGrid.foreach(row => println(row.mkString(", ")))
```

```
1, 0, 0, 0
0, 1, 0, 0
```

```
0, 0, 1, 0
0, 0, 0, 1
```

```
size = 4
dynamicGrid = Array(Array(1, 0, 0, 0), Array(0, 1, 0, 0), Array(0, 0, 1, 0),␣
 ↪Array(0, 0, 0, 1))
```

[6]:
```
Array(Array(1, 0, 0, 0), Array(0, 1, 0, 0), Array(0, 0, 1, 0), Array(0, 0, 0,
1))
```

[7]:
```scala
// Initialize a multi-dimensional array
val rows = 3
val cols = 4
val multiDimArray = Array.ofDim[Int](rows, cols)

// Fill the array with incremental values
var value = 1
for (i <- 0 until rows; j <- 0 until cols) {
  multiDimArray(i)(j) = value
  value += 1
}

// Print the multi-dimensional array
multiDimArray.foreach(row => println(row.mkString(", ")))
```

```
1, 2, 3, 4
5, 6, 7, 8
9, 10, 11, 12
```

```
rows = 3
cols = 4
multiDimArray = Array(Array(1, 2, 3, 4), Array(5, 6, 7, 8), Array(9, 10, 11, 12))
value = 13
```

[7]:
```
13
```

[ ]:
```
```