**EXPERIMENT NO: 04**

|  |
| --- |
| **Title:** Implementation of Multidimensional & Jagged array. |
| **Aim:** Study of Arrays and Jagged array. |

**Theory:**

**Arrays:**

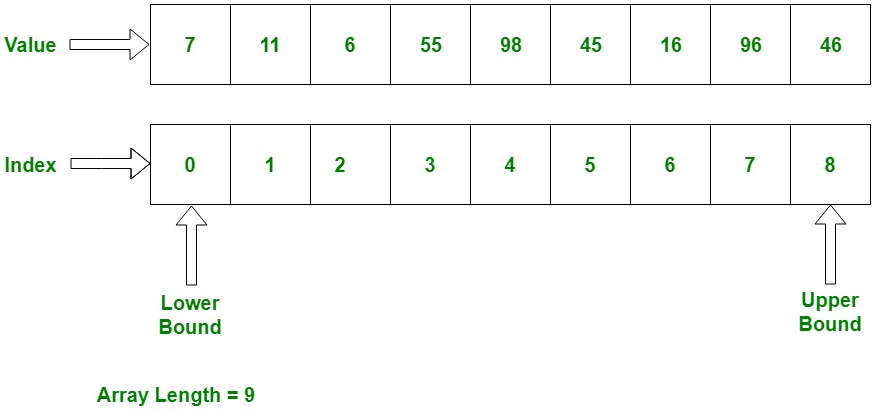
An array is a group of like-typed variables that are referred to by a common name. And each data item is called an element of the array. The data types of the elements may be any valid data type like char, int, float, etc. and the elements are stored in a contiguous location. Length of the array specifies the number of elements present in the array. In C# the allocation of memory for the arrays is done dynamically. The variables in the array are ordered and each has an index beginning from 0.

**Important Points Arrays in C#:**

* In C#, all arrays are dynamically allocated.
* Since arrays are objects in C#, we can find their length using member length. This is different from C/C++ where we find length using sizeof operator.
* A C# array variable can also be declared like other variables with [] after the data type.
* The variables in the array are ordered and each has an index beginning from 0.
* C# array is an object of base type System.Array.
* Default values of numeric array and reference type elements are set to be respectively zero and null.
* A jagged array elements are reference types and are initialized to null.
* Array elements can be of any type, including an array type.
* Array types are reference types which are derived from the abstract base type Array. These types implement IEnumerable and for it, they use foreach iteration on all arrays in C#.

The array has can contain primitive data types as well as objects of a class depending on the definition of an array. Whenever use primitives data types, the actual values have to be stored in contiguous memory locations. In the case of objects of a class, the actual objects are stored in the heap segment.

The following figure shows how array stores values sequentially:



**Explanation:**

The index is starting from 0, which stores value. We can also store a fixed number of values in an array. Array index is to be increased by 1 in sequence whenever it’s not reach the array size.

**Array Declaration Syntax:**

< Data Type > [ ] < Name\_Array >

Here,

**< Data Type > :** It define the element type of the array.

**[ ]:** It define the size of the array.

**< Name\_Array >:** It is the Name of array.

**Example :**

int[] x; // can store int values

string[] s; // can store string values

double[] d; // can store double values

Student[] stud1; // can store instances of Student class which is custom class

Note : Only Declaration of an array doesn’t allocate memory to the array. For that array must be initialized.

**Array Initialization:**

As said earlier, an array is a reference type so the new keyword used to create an instance of the array. We can assign initialize individual array elements, with the help of the index.

**Syntax:**

type [ ] < Name\_Array > = new < datatype > [size];

Here, type specifies the type of data being allocated, size specifies the number of elements in the array, and Name\_Array is the name of an array variable. And new will allocate memory to an array according to its size.

**Examples:** To Show Different ways for the Array Declaration and Initialization

**Example 1:**

// defining array with size 5. But not assigns values

int[] intArray1 = new int[5];

The above statement declares & initializes int type array that can store five int values. The array size is specified in square brackets ([]).

**Example 2:**

// defining array with size 5 and assigning values at the same time

int[] intArray2 = new int[5]{1, 2, 3, 4, 5};

The above statement is the same as, but it assigns values to each index in {}.

**Example 3:**

// defining array with 5 elements which indicates the size of an array

int[] intArray3 = {1, 2, 3, 4, 5};

In the above statement, the value of the array is directly initialized without taking its size. So, array size will automatically be the number of values which is directly taken.

**Initialization of an Array after Declaration:**

Arrays can be initialized after the declaration. It is not necessary to declare and initialize at the same time using the new keyword. However, Initializing an Array after the declaration, it must be initialized with the new keyword. It can’t be initialized by only assigning values.

**Example:**

// Declaration of the array

String [] str1, str2;

// Initialization of array

str1 = new string[5]{ “Element 1”, “Element 2”, “Element 3”, “Element 4”, “Element 5” };

str2 = new string[5]{ “Element 1”, “Element 2”, “Element 3”, “Element 4”, “Element 5” };

**Accessing Array Elements:**

At the time of initialization, we can assign the value. But, we can also assign the value of the array using its index randomly after the declaration and initialization. We can access an array value through indexing, placed index of the element within square brackets with the array name

**Example:**

//declares & initializes int type array

int[] intArray = new int[5];

// assign the value 10 in array on its index 0

intArray[0] = 10;

// assign the value 30 in array on its index 2

intArray[2] = 30;

// assign the value 20 in array on its index 1

intArray[1] = 20;

// assign the value 50 in array on its index 4

intArray[4] = 50;

// assign the value 40 in array on its index 3

intArray[3] = 40;

// Accessing array elements using index

intArray[0]; //returns 10

intArray[2]; //returns 30

**Implementation: Accessing Array elements using different loops**

// C# program to illustrate creating an array of integers, puts some values in the array,and prints //each value to standard output.

using System;

namespace geeksforgeeks {

class GFG {

// Main Method

public static void Main()

{

// declares an Array of integers.

int[] intArray;

// allocating memory for 5 integers.

intArray = new int[5];

// initialize the first elements of the array

intArray[0] = 10;

// initialize the second elements of the array

intArray[1] = 20;

// so on...

intArray[2] = 30;

intArray[3] = 40;

intArray[4] = 50;

// accessing the element using for loop

Console.Write("For loop :");

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

Console.Write(" " + intArray[i]);

Console.WriteLine("");

Console.Write("For-each loop :");

// using for-each loop

foreach(int i in intArray)

Console.Write(" " + i);

Console.WriteLine("");

Console.Write("while loop :");

// using while loop

int j = 0;

while (j < intArray.Length) {

Console.Write(" " + intArray[j]);

j++;

}

Console.WriteLine("");

Console.Write("Do-while loop :");

// using do-while loop

int k = 0;

do

{

Console.Write(" " + intArray[k]);

k++;

} while (k < intArray.Length);

}

}

}

**One Dimensional Array:**

In this array contains only one row for storing the values. All values of this array are stored contiguously starting from 0 to the array size. For example, declaring a single-dimensional array of 5 integers:

int[] arrayint = new int[5];

The above array contains the elements from arrayint[0] to arrayint[4]. Here, the new operator has to create the array and also initialize its element by their default values. Above example, all elements are initialized by zero, because it is the int type.**Example:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace One\_Dimensional\_Array

{

class Program

{

static void Main(string[] args)

{

//Declaring single dimensional array

string[] Books = new string[5];

Books[0] = "C#";

Books[1] = "Java";

Books[2] = "VB.NET";

Books[3] = "C++";

Books[4] = "C";

Console.WriteLine("All the element of Books array is:\n\n");

int i = 0;

//Formatting Output

Console.Write("\t1\t2\t3\t4\t5\n\n\t");

for (i = 0; i < 5; i++)

{

Console.Write("{0}\t", Books[i]);

}

Console.ReadLine();

}

}

}

**Multi-Dimensional Array in C#:-**

The multi-dimensional array in C# is such type of array that contains more than one row to store data on it. The multi-dimensional array is also known as a rectangular array in c sharp because it has the same length of each row. It can be a two-dimensional array or three-dimensional array or more. It contains more than one comma (,) within single rectangular brackets (“[ , , ,]”). To storing and accessing the elements from a multidimensional array, you need to use a nested loop in the program.

## **Example 1: C# program to declare, input, and print a two-dimensional array**

using System;

namespace arrayEx {

class Program {

static void Main(string[] args) {

int i = 0;

int j = 0;

int[, ] arr;

// Declaration

arr = new int[2, 3];

// Input array elements

Console.Write("Enter Elements : \n");

for (i = 0; i < 2; i++) {

for (j = 0; j < 3; j++) {

Console.Write("\tElement[" + i + "," + j + "]: ");

arr[i, j] = Convert.ToInt32(Console.ReadLine());

}

}

// Print array Elements

Console.Write("\n\nElements are: \n");

for (i = 0; i < 2; i++) {

for (j = 0; j < 3; j++) {

Console.Write(arr[i, j] + " ");

}

Console.WriteLine();

}

}

}

}

### **Output**

Enter Elements :

Element[0,0]: 10

Element[0,1]: 20

Element[0,2]: 30

Element[1,0]: 40

Element[1,1]: 50

Element[1,2]: 60

Elements are:

10 20 30

40 50 60

Press any key to continue . . .

**Jagged Arrays:**

An array whose elements are arrays is known as Jagged arrays it means “array of arrays“. The jagged array elements may be of different dimensions and sizes. Every row has a different number of elements in it. Sometimes, a jagged array can also be known as an "array of arrays".

## **Declaration of a Jagged Array**

The following syntax shows the declaration of a jagged array:

<data\_type>[][] variable = new <data\_type> [row\_size][];

The following code shows the declaration of a jagged array with different sizes of arrays:

int[][] jarr = new int[2][];

jarr[0] = new int [4];

jarr[1] = new int [6];

## **Initialization of a Jagged Array**

The following code shows the initialization of a jagged array:

int[][] jarr = new int[][] {new int[] {1, 2, 3}, new int[] {4,5, 6, 7}};

**Example:**

using System;

namespace arrayEx {

class Program {

static void Main(string[] args) {

int i = 0;

int j = 0;

int[][] jarr = new int[][] {new int[] {1,2,3}, new int[] {4,5,6,7}};

Console.Write("\n\nElements are: \n");

for (i = 0; i < jarr.GetLength(0); i++) {

for (j = 0; j < jarr[i].Length; j++) {

Console.Write(jarr[i][j] + " ");

}

Console.WriteLine();

}

}

}

}

**Output**

Elements are:

1 2 3

4 5 6 7

Press any key to continue . . .

**Problem Statement:**

Employee Qualifications Management System

You are assigned to develop a program to manage the qualifications of employees. The program should allow the user to input the number of employees, their names, and their qualifications.

1. The program should perform the following tasks:
2. Prompt the user to enter the number of employees.
3. For each employee, prompt the user to enter the name and the number of qualifications.
4. For each qualification, prompt the user to enter the qualification details.

1. Display the entered qualifications for each employee.

Your program should use a jagged array to store employee names and their qualifications. Each employee may have a different number of qualifications.

**Conclusion:**

**Sample Questions:**

1) What is Array?

2) Explain difference between array and jagged array?