**C# Jagged Arrays**

Назъбеният масив е масив от масиви, така че масивите от членове могат да бъдат с различни размери. С други думи, дължината на всеки индекс на масив може да се различава. Елементите на Jagged Array са референтни типове и са инициализирани на нула по подразбиране. Jagged Array също може да се смесва с многомерни масиви. Тук броят на редовете ще бъде фиксиран към момента на деклариране, но можете да променяте броя на колоните.

**Декларация**

В назъбените масиви потребителят трябва да предостави само броя на редовете. Ако потребителят също ще предостави броя на колоните, тогава този масив вече няма да бъде назъбен масив.

**Синтаксис:**

data\_type[][] name\_of\_array = new data\_type[rows][]

**Пример:**

int[][] jagged\_arr = new int[4][]

В горния пример е деклариран едномерен масив, който има 4 елемента (реда), всеки от които е 1-D масив от цели числа.

**Инициализация**

Елементите на Jagged Array трябва да бъдат инициализирани преди използването му. Можете отделно да инициализирате всеки елемент от масива. Има много начини за инициализиране на елемента на масива Jagged.

**Пример 1**: Предоставяне на размера на всеки елемент от масива поотделно. Тук всеки от елементите е 1-D масив от цели числа, където:

• Първият ред или елемент е масив от 2 цели числа.

• Вторият ред или елемент е масив от 4 цели числа.

• Третият ред или елемент е масив от 6 цели числа.

• Четвъртият ред или елемент е масив от 7 цели числа:

jagged\_arr[0] = new int[2];

jagged\_arr[1] = new int[4];

jagged\_arr[2] = new int[6];

jagged\_arr[3] = new int[7];

**Пример 2:** Когато размерът на масива не е необходим, тогава неговите елементи могат да бъдат инициализирани с директни стойности, както следва:

jagged\_arr[0] = new int[] {1, 2, 3, 4};

jagged\_arr[1] = new int[] {11, 34, 67};

jagged\_arr[2] = new int[] {89, 23};

jagged\_arr[3] = new int[] {0, 45, 78, 53, 99};

**Деклариране и инициализация**

**Пример 1:** Using the Direct Method

int[][] jagged\_arr = new int[][]

{

new int[] {1, 2, 3, 4},

new int[] {11, 34, 67},

new int[] {89, 23},

new int[] {0, 45, 78, 53, 99}

};

**Пример 2:** Използване на метода на късата ръка. Няма инициализация по подразбиране за елементите, така че потребителят не може да пропусне новия оператор от инициализацията на елементите.

int[][] jagged\_arr =

{

new int[] {1, 2, 3, 4},

new int[] {11, 34, 67},

new int[] {89, 23},

new int[] {0, 45, 78, 53, 99}

};

**Достъп до елементите**

За достъп до елементите на масива Jagged потребителят трябва да посочи реда и колоната с името на масива.

**Пример:**// Достъп и присвояване на 99 на третия елемент ([2]) от втория масив ([1])jagged\_arr[1][2] = 99;

// Достъп и присвояване на 47 на първия елемент ([0]) от четвъртия масив ([3]):

jagged\_arr[3][0] = 47;

**Примерна програма:**

// C# програма, която илюстрира децларирането и инициализацията на Jagged Arrays

using System;

class GFG {

// Main Method

public static void Main()

{

// Declare the Jagged Array of four elements:

int[][] jagged\_arr = new int[4][];

// Initialize the elements

jagged\_arr[0] = new int[] {1, 2, 3, 4};

jagged\_arr[1] = new int[] {11, 34, 67};

jagged\_arr[2] = new int[] {89, 23};

jagged\_arr[3] = new int[] {0, 45, 78, 53, 99};

// Display the array elements:

for (int n = 0; n < jagged\_arr.Length; n++) {

// Print the row number

System.Console.Write("Row({0}): ", n);

for (int k = 0; k < jagged\_arr[n].Length; k++) {

// Print the elements in the row

System.Console.Write("{0} ", jagged\_arr[n][k]);

}

System.Console.WriteLine();

}

}

}

**Резултат от изпълнението:**

Row(0): 1 2 3 4

Row(1): 11 34 67

Row(2): 89 23

Row(3): 0 45 78 53 99

**Назъбени масиви с многомерни масиви**

Възможно е да се смесват назъбени и многоизмерни масиви. По-долу са декларацията и инициализацията на 1-D назъбен масив, който съдържа четири двуизмерни елемента на масив с различни размери.

**Пример:**int[][, ] jagged\_arr1 = new int[4][, ]

{

new int[, ] { {1, 3}, {5, 7} },

new int[, ] { {0, 2}, {4, 6}, {8, 10} },

new int[, ] { {7, 8}, {3, 1}, {0, 6} },

new int[, ] { {11, 22}, {99, 88}, {0, 9} }

};

Потребителят може да получи достъп до отделните елементи, както е показано в примера по-долу, който показва стойността на елемента [2, 0] от втория масив (т.е. стойността е 8)

**Пример:** System.Console.Write("{0}", jagged\_arr1[1][1, 0]);

**Примерна програма**

// C# програма, която илюстрира смесването на 1-D Jagged Array с 4 двумерни масива

using System;

namespace geeksforgeeks {

class GFG {

// Main Method

public static void Main()

{

// Declaration and Initialization of

// Jagged array with 4 2-D arrays

int[][, ] jagged\_arr1 = new int[4][, ] {new int[, ] {{1, 3}, {5, 7}},

new int[, ] {{0, 2}, {4, 6}, {8, 10}},

new int[, ] {{7, 8}, {3, 1}, {0, 6}},

new int[, ] {{11, 22}, {99, 88}, {0, 9}}};

// Display the array elements:

// Length method returns the number of

// arrays contained in the jagged array

for (int i = 0; i < jagged\_arr1.Length; i++)

{

int x = 0;

// GetLength method takes integer x which

// specifies the dimension of the array

for (int j = 0; j < jagged\_arr1[i].GetLength(x); j++)

{

// Rank is used to determine the total

// dimensions of an array

for (int k = 0; k < jagged\_arr1[j].Rank; k++)

Console.Write("Jagged\_Array[" + i + "][" + j + ", " + k + "]: "

+ jagged\_arr1[i][j, k] + " ");

Console.WriteLine();

}

x++;

Console.WriteLine();

}

}

}

}

**Изход:**

Jagged\_Array[0][0, 0]: 1 Jagged\_Array[0][0, 1]: 3

Jagged\_Array[0][1, 0]: 5 Jagged\_Array[0][1, 1]: 7

Jagged\_Array[1][0, 0]: 0 Jagged\_Array[1][0, 1]: 2

Jagged\_Array[1][1, 0]: 4 Jagged\_Array[1][1, 1]: 6

Jagged\_Array[1][2, 0]: 8 Jagged\_Array[1][2, 1]: 10

Jagged\_Array[2][0, 0]: 7 Jagged\_Array[2][0, 1]: 8

Jagged\_Array[2][1, 0]: 3 Jagged\_Array[2][1, 1]: 1

Jagged\_Array[2][2, 0]: 0 Jagged\_Array[2][2, 1]: 6

Jagged\_Array[3][0, 0]: 11 Jagged\_Array[3][0, 1]: 22

Jagged\_Array[3][1, 0]: 99 Jagged\_Array[3][1, 1]: 88

Jagged\_Array[3][2, 0]: 0 Jagged\_Array[3][2, 1]: 9