زنگ سی‌شارپ – قسمت نوزدهم

نوشته‌ی مسعود درویشیان

لینک مستقیم این مطلب در وب‌تارگت

در قسمت هجدهم و هفدهم با آرایه‌های یک‌بعدی آشنا شدید و دانستید که ماهیت آن‌ها چیست و چگونه تعریف می‌شوند. البته نکته‌ای در مورد ماهیت اصلی آرایه‌ها مانده است که بعد از آشنایی با Object ها به آن پی می‌برید اما در این قسمت با آرایه‌های چند بعدی آشنا می‌شوید.

آرایه‌های چند بعدی

آن‌چنان که آرایه‌های یک‌بعدی در برنامه‌نویسی کاربرد دارند آرایه‌های چندبعدی رایج نیستند. یک آرایه چندبعدی آرایه‌ای است که ۲ یا بیشتر از ۲ بعد دارد و برای دسترسی به عناصر این آرایه‌ها باید از ۲ یا بیشتر از ۲ شاخص (index) استفاده کنیم.

آرایه دو بعدی

ساده‌ترین شکل آرایه‌های چندبعدی، آرایه‌های ۲ بعدی هستند. در آرایه‌های ۲ بعدی موقعیت هر عنصر با دو index مشخص می‌شود. اگر به آرایه‌های ۲ بعدی به چشم یک جدول نگاه کنیم، یک index مشخص‌کننده‌ی سطر و index دیگر مشخص‌کننده‌ی ستون است. برای ساختن یک آرایه ۲ بعدی با ابعاد ۴ × ۳ می‌نویسید:

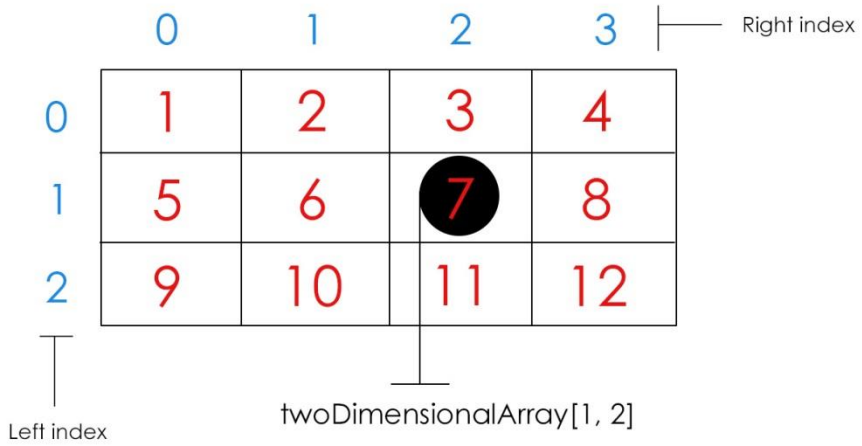
```
int[,] twoDimensionalArary = new int[3,4];
```

برای دسترسی به خانه‌های آرایه‌ی ۲ بعدی باید از طریق دو index که توسط کاما از هم جدا شده‌اند این‌کار را انجام دهید. به مثال زیر توجه کنید:

```
using System;
class Example
{
    static void Main()
    {
        int[,] twoDimensionalArray = new int[3,4];

        // First row
        twoDimensionalArray[0, 0] = 1;
        twoDimensionalArray[0, 1] = 2;
        twoDimensionalArray[0, 2] = 3;
        twoDimensionalArray[0, 3] = 4;

        // Second row
        twoDimensionalArray[1, 0] = 5;
        twoDimensionalArray[1, 1] = 6;
```

```
        twoDimensionalArray[1, 2] = 7;
        twoDimensionalArray[1, 3] = 8;


        // Third row
        twoDimensionalArray[2, 0] = 9;
        twoDimensionalArray[2, 1] = 10;
        twoDimensionalArray[2, 2] = 11;
        twoDimensionalArray[2, 3] = 12;

        // Displaying Array's data
        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < 4; j++)
            {
                Console.Write(twoDimensionalArray[i, j] + "\t");
            }
            Console.WriteLine();
        }
    }
}
```

تصویر مفهومی و گرافیکی این آرایه دو بعدی به شکل زیر است:



دقت کنید که چگونه در این برنامه خانه‌های آرایه‌ی دو بعدی را مقداردهی کرده‌ایم. پس از پر کردن خانه‌های آرایه، از طریق حلقه‌های تودرتو مقدار تک تک عناصر آرایه را نمایش داده‌ایم. همان‌طور که در شکل بالا می‌بینید، برای مقداردهی به خانه‌های آرایه ۲ بعدی می‌بایست دو index سطر و ستون را مشخص کنید.

آرایه‌های چند بعدی

سی‌شارپ به شما اجازه می‌دهد آرایه‌هایی با ابعاد ۲ یا بیشتر از ۲ بعد بسازید:

```
type[ , ... , ] multiDimensionalArray = new type[size1, size2, ... , sizeN];
```

برای مثال در زیر یک آرایه ۳ بعدی با ابعاد ۱۰ × ۴ × ۳ می‌سازیم:

```
int[, ,] threeDimensionalArray = new int[3, 4, 10];
```

برای این‌که به خانه‌ای در موقعیت ۲،۴،۱ این آرایه مقدار ۱۰۰ را اختصاص دهیم به‌این صورت عمل می‌کنیم:

```
threeDimensionalArray[2, 4, 1] = 100;
```

به مثال زیر توجه کنید:

```
using System;
class Example
{
    static void Main()
    {
        int[, ,] threeDimensionalArray = new int[3, 3, 3];

        int result = 1;
        int sum;

        for (int x = 0; x < 3; x++)
            for (int y = 0; y < 3; y++)
                for (int z = 0; z < 0; z++)
                    threeDimensionalArray[x, y, z] = result++;

        sum = threeDimensionalArray[0, 0, 0] +
              threeDimensionalArray[1, 1, 1] +
              threeDimensionalArray[2, 2, 2];

        Console.WriteLine("Summation of first diagonal is " + sum);
    }
}
```

در این مثال از طریق ۳ حلقه‌ی تودرتو خانه‌های آرایه را پر کرده‌ایم.

مقداردهی به آرایه‌های ۲ بعدی می‌تواند از الگوی زیر نیز انجام گیرد:

```
type[,] array_name = {
                        { val, val, val, ..., val },
                        { val, val, val, ..., val },
                        .
                        .
                        .
                        { val, val, val, ..., val }
                    };
```

در این‌جا هر بلاک داخلی معین‌کننده‌ی یک سطر است. درون هر سطر، مقدار اول در موقعیت و مکان اول قرار می‌گیرد. مقدار دوم در مکان دوم والی آخر. توجه کنید که بلاک‌های داخلی توسط کاما از هم جدا می‌شوند و در نهایت سمی‌کالن بعد از براکت بسته شده‌ی آخر قرار می‌گیرد.

به مثال زیر توجه کنید:

```
using System;
class Example
{
    static void Main()
    {
        int[,] myTwoDimArray = {
                                    { 1, 0, 1, 1 },
                                    { 0, 0, 1, 0 },
                                    { 1, 1, 1, 0 },
                                    { 0, 1, 0, 1 }
                                };

        for (int i = 0; i < 4; i++)
        {
            for (int j = 0; j < 4; j++)
            {
                Console.Write(myTwoDimArray[i, j] + "\t");
            }
            Console.WriteLine();
        }
    }
}
```

آرایه‌های دنده‌دار (Jagged Array)

در مثال‌های قبل، آرایه‌های ۲ بعدی ساخته شده، آرایه‌ی مستطیلی نیز نامیده می‌شوند. آرایه‌های ۲ بعدی مستطیل شکل را در نظر بگیرید، در این آرایه‌ها طول هر سطر در تمام آرایه یکسان است و به همین دلیل شکل مستطیل هستند. سی‌شارپ به شما اجازه می‌دهد نوع مخصوصی از آرایه‌ی ۲ بعدی بسازید که Jagged Array یا آرایه‌ی دنده‌دار نامیده می‌شود. یک Jagged Array آرایه‌ای از آرایه‌هاست که در آن طول هر آرایه می‌تواند متفاوت باشد.

برای ساخت یک Jagged Array از فرم کلی زیر استفاده می‌کنیم:

```
type[][] array-name = new type[size][ ];
```

در این‌جا size مشخص‌کننده‌ی تعداد سطرهای آرایه است و مقدار هر سطر باید به‌صورت جداگانه اختصاص داده شود. با این روش می‌توانید برای هر سطر طول متفاوتی داشته باشید.

```
int[][] jaggedArray = new int[3][];

jaggedArray[0] = new int[4];
```

```
jaggedArray[1] = new int[3];
jaggedArray[2] = new int[5];
```

در این مثال ابتدا یک Jagged Array تعریف کرده‌ایم که ۳ سطر دارد و سطرهای آن به‌ترتیب طولی برابر با ۴، ۳ و ۵ دارند. در تصویر زیر شکل مفهومی این آرایه و آدرس خانه‌های مختلف آن را می‌بینید:

برای پر کردن خانه‌های آرایه به‌صورت زیر عمل می‌کنیم:

```
using System;
class Example
{
    static void Main()
    {
        int[][] jaggedArray = new int[3][];

        jaggedArray[0] = new int[4];
        jaggedArray[1] = new int[3];
        jaggedArray[2] = new int[5];

        int i;

        // Store values in first row (first array)
        for (i = 0; i < 4; i++)
        {
            jaggedArray[0][i] = i;
        }

        // Store values in second row (second array)
        for (i = 0; i < 3; i++)
        {
            jaggedArray[1][i] = i;
        }

        // Store values in third row (third array)
        for (i = 0; i < 5; i++)
        {
            jaggedArray[2][i] = i;
        }
    }
}
```

Jagged Array در همه‌ی برنامه‌ها استفاده نمی‌شود اما در برخی موارد می‌تواند موثر واقع شود. برای مثال فرض کنید قصد دارید قیمت‌های بلیط قطار را در هر ایستگاه برای ۵ مسیر متفاوت ذخیره کنید. تصور کنید یکی از مسیرها ۱۰ ایستگاه

دارد و بقیه‌ی مسیرها کمتر از ۱۰ ایستگاه دارند. می‌بایست هریک از این مسیرها و قیمت ایستگاه‌ها توسط یک سطر از آرایه‌ی چندبعدی نمایش داده شود که در این صورت شما ۲ انتخاب دارید:

- می‌توانید یک آرایه مستطیلی (۲ بعدی) بسازید و اندازه هر سطر را برابر با ۱۰ قرار دهید که در این صورت خانه‌های آرایه شما در بعضی جاها خالی می‌ماند. زیرا یک مسیر ۱۰ ایستگاه دارد، مسیر دیگر ۳ ایستگاه، ۵ ایستگاه و ...
- انتخاب دوم می‌تواند Jagged Array باشد که به شما اجازه می‌دهد برای هر سطر به تعداد دل‌خواهی ستون داشته باشید.

به کد زیر که به این منظور نوشته شده است دقت کنید:

```
double[][] tickets = {
                        new double[] {5.50, 6.75, 7.95, 9.00, 12.00,
                                      13.00, 14.50, 17.00, 19.00, 20.25},
                        new double[] {5.00, 6.00},
                        new double[] {7.50, 9.00, 9.95, 12.00, 13.00, 14.00},
                        new double[] {3.50, 6.45, 9.95, 10.00, 12.75},
                        new double[] {15.00, 16.00}
                    };
```

به نحوه‌ی تعریف و مقداردهی Jagged Array در مثال بالا دقت کنید.

از آن‌جا که Jagged Array آرایه‌ای از آرایه‌هاست، محدودیتی از بابت این‌که تنها آرایه‌های ۱ بعدی مجاز باشند وجود ندارد و شما می‌توانید آرایه‌ای از آرایه‌های ۲ بعدی بسازید:

```
int[][,] twoDimJaggedArray = new int[3][,];

twoDimJaggedArray[0] = new int[4,2];
twoDimJaggedArray[1] = new int[3, 2];
twoDimJaggedArray[2] = new int[5,3];
```

همان‌طور که می‌بینید، twoDimJaggedArray[0] شامل یک آرایه ۲ بعدی ۲ × ۴ است. برای مقداردهی به یکی از خانه‌های این آرایه ۲ بعدی باید نوشت:

```
twoDimJaggedArray[0][2, 1] = 5;
```

Implicitly Typed Variable

همان‌طور که پیش از این ذکر شد، در سی‌شارپ متغیرها قبل از این‌که مورد استفاده قرار گیرند باید اعلام شوند و برای این‌کار به‌طور معمول ابتدا نوع متغیر را می‌نویسید، سپس نام متغیر را پس از آن قرار می‌دهید:

```
int x;
```

از زمانی‌که C# 3.0 منتشر شد، قابلیتی به‌وجود آمد که می‌توان به کامپایلر اجازه داد تا خودش نوع متغیر را با توجه به مقداری که به متغیر داده می‌شود، تشخیص دهد. به این عمل Implicitly Typed Variable می‌گویند.

یک Implicitly Typed Variable با کلمه‌کلیدی var مشخص شده و حتماً بایستی مقداردهی شود. کامپایلر با توجه به مقدار داده شده به متغیر، نوع آن را تشخیص می‌دهد:

```
var e = 2.6080;
```

به دلیل این‌که متغیر e به‌صورت floating-point مقداردهی شده است (که به‌صورت پیش‌فرض double هستند) نوع متغیر e نیز به double تبدیل می‌شود.

اگر e به صورت زیر تعریف شود:

```
var e = 2.6080F;
```

آن‌گاه e از جنس float خواهد بود.

```
var name = "Roxy"; // has the same meaning as string name = "Roxy";
var age = 30; // has the same meaning as int age = 30;
```

Implicitly Typed Variable برای این‌که جای تعریف متغیر به صورت معمول (Explicitly typed variable) را بگیرد نیامده است. بلکه برای برخی موارد خاص در استفاده از LINQ است که در مقالات آینده با آن آشنا خواهید شد. بنابراین برای تعریف متغیر در حالت کلی از روش معمول استفاده کنید چراکه باعث خواناتر شدن و فهم بهتر کد می‌شود.

Implicitle Typed Array

بااستفاده از همان مکانیسم قبلی می‌توانید یک Implicitly typed Array بسازید. یک Implicitly typed Array با استفاده از کلمه‌کلیدی var ساخته می‌شود اما بعد از var از براکت [ ] استفاده نمی‌شود. علاوه‌بر این آرایه باید مستقیماً مقدار دهی

شود چراکه نوع و جنس آرایه با توجه به مقادیر درون آن مشخص می‌شود. همه‌ی مقادیر آرایه باید از یک نوع یا از نوع‌های سازگار باهم باشند.

این خط کد آرایه‌ای از جنس int با طول ۵ می‌سازد:

```
var vals = new[] { 1, 2, 3, 4, 5 };
```

ساخت Implicitly typed Array دو بعدی:

```
var vals = new[,] { { 1.1, 2.2 }, { 3.3, 4.4 }, { 5.5, 6.6 } };
```

در این‌جا آرایه‌ای ۳ × ۲ ساخته شده است. شما همچنین می‌توانید Implicitly typed Jagged Array داشته باشید:

```
var jaggedArray = new[] {
       new[] { 1, 2, 3, 4 },
       new[] { 9, 8, 7 },
       new[] { 11, 12, 13, 14 ,15 }
};
```

توجه کنید در مثال بالا چگونه به دو طریق از new[] استفاده شده است. ابتدا از آن برای ساخت آرایه‌ای از آرایه‌ها استفاده شده و سپس در هر سطر به‌صورت جداگانه آرایه‌ای را با توجه به مقادیر آن ساخته است. همان‌طور که ذکر شد از Implicitly typed Array/Variable بیشتر در کوئری‌های بر پایه LINQ مورد استفاده قرار می‌گیرد و نباید از آن‌ها در حالت معمول استفاده کرد.

حلقه‌ی Foreach

در سی‌شارپ حلقه‌ای به اسم foreach وجود دارد که این حلقه تک تک عناصر یک Collection را به‌ترتیب از ابتدا تا انتها بررسی می‌کند. یک Collection شامل گروهی از اشیاء (Objects) است. سی‌شارپ چندین نوع Collection دارد که یکی از آن‌ها آرایه است.

شکل کلی دستور foreach به‌صورت زیر است:

```
foreach (type item in collection) statement;
```

در این‌جا type item مشخص کننده‌ی اسم و نوع یک متغیر است که این متغیر، عنصر بعدی Collection (در این‌جا کالکشن، آرایه است) را در هربار که حلقه تکرار می‌شود دریافت می‌کند. این متغیر در اصطلاح iteration variable نامیده

می‌شود. بنابراین type باید با جنس آرایه یکسان (یا سازگار) باشد. type همچنین میتواند var باشد که در این مورد نیز کامپایلر با توجه به جنس آرایه، نوع را تشخیص می‌دهد اما بهتر است به‌طور معمول، نوع متغیر را خودتان مشخص کنید.

هنگامی‌که حلقه‌ی foreach شروع به‌کار می‌کند، اولین عنصر آرایه به متغیر item اختصاص داده می‌شود و در هربار تکرار حلقه، عنصر بعدی آرایه گرفته شده و به item اختصاص داده می‌شود و تکرار حلقه تا زمانی‌که عنصری در آرایه وجود داشته باشد ادامه می‌یابد.

به مثال زیر توجه کنید:

```
using System;
class Example
{
    static void Main()
    {
        int[] anArray = { 2, 33, 56, 42, 13, 15, 79 };

        foreach (int loopvar in anArray)
        {
            Console.Write(loopvar + " ");
        }
        Console.WriteLine();
    }
}
```

همان‌طور که می‌بینید، بعد از اجرای این برنامه تمام خانه‌های آرایه در خروجی نمایش داده می‌شود. زمانی‌که حلقه شروع به‌کار می‌کند، عنصر اول آرایه در متغیر loopvar ذخیره می‌شود و در چرخش بعدی حلقه، عنصر بعدی آرایه به متغیر loopvar اختصاص می‌یابد و این روند تا زمانی‌که خانه‌های آرایه به اتمام برسد ادامه دارد.

حل تمرین شماره ۱۲: در این تمرین قصد داشتیم دفترچه تلفنی بسازیم که قابلیت افزودن، حذف ، جستجو و ویرایش کردن را داشته باشد. با توجه به توضیحاتی که از قسمت اول تاکنون داده شده است، حل این تمرین قابل فهم بوده و حتماً توانسته‌اید مساله را حل کنید اما با وجود این، در صورت نامفهوم بودن هر قسمت از این تمرین می‌توانید سوالات خود را مطرح کنید:

```
using System;
class SimplePhonebook
{
    static void Main()
    {
        // Setting Cursor Size
        Console.CursorSize = 100;

        int arraySize = 5;
        string[] names = new string[arraySize];
```

```
        string[] numbers = new string[arraySize];

        // A string array for storing main menu strings
        string[] menu = {
                            "╔═══════════ Phonebook Software ═══════════╗",
                            "║                                          ╒",
                            "│ 1. Add Contact                           │",
                            "│ 2. Search by Name and Number             │",
                            "│ 3. View All                              │",
                            "│ 4. Exit                                  │",
                            "╒                                          ╒",
                            "╚═══════════════════════════════════════════╝"
                        };

        while (true)
        {
            Console.Clear();
            // Setting console font color
            Console.ForegroundColor = ConsoleColor.Cyan;

            // Displaying main menu
            for (int i = 0; i < menu.Length; i++)
            {
                Console.Write(menu[i] + "\n");
            }
            Console.WriteLine("╔═══════════════════════════════════════╗");
            Console.WriteLine("│ You have to pick out a number         │");
            Console.WriteLine("╚═══════════════════════════════════════╝");
            Console.WriteLine("-----------------------------------------");
            Console.WriteLine("Enter your choice: ");
            Console.Write("> ");

            // Getting user entry for a service
            string userChoice = Console.ReadLine();
            switch (userChoice)
            {
                    // Adding a contact
                case "1":
                    Console.Clear();
                    Console.WriteLine("Enter Name: ");
                    Console.Write("> ");
                    string name = Console.ReadLine();
                    Console.WriteLine("Enter Number: ");
                    Console.Write("> ");
                    string number = Console.ReadLine();

                    // testing for empty part of arrays
                    for (int i = 0; i < arraySize; i++)
                    {
                        if (names[i] == null && numbers[i] == null)
                        {
                            numbers[i] = number;
                            names[i] = name;
                            Console.ForegroundColor = ConsoleColor.Green;
                            Console.Write("> Your contact added successfully!");
                            break;
                        }
                    }
                    break;

                    // Searching by contact name or number
                case "2":
```

```
                Console.Clear();
                Console.WriteLine("Enter a Name or a Number: ");
                Console.Write("> ");

                bool found = false;
                string inputForSearch = Console.ReadLine(); ;

                for (int i = 0; i < arraySize; i++)
                {
                    if (inputForSearch == names[i] || inputForSearch == numbers[i])
                    {
                        found = true;

                        Console.WriteLine();
                        Console.ForegroundColor = ConsoleColor.Green;
                        Console.WriteLine("> Founded!");
                        Console.WriteLine();
                        Console.ForegroundColor = ConsoleColor.Cyan;
                        Console.WriteLine("│  Name: " + names[i]);
                        Console.WriteLine("│  Number: " + numbers[i]);
                        Console.WriteLine();
                        Console.WriteLine("1. Edit      2. Delete      3. Cancel");

                        // editing, deleting or canceling
                        while (true)
                        {
                            Console.Write("> ");
                            string inputForEdit = Console.ReadLine();
                            if (inputForEdit == "1")
                            {
                                Console.WriteLine();
                                Console.WriteLine("Enter new Name: ");
                                Console.Write("> ");
                                names[i] = Console.ReadLine();
                                Console.WriteLine("Enter new Number: ");
                                Console.Write("> ");
                                numbers[i] = Console.ReadLine();
                                Console.ForegroundColor = ConsoleColor.Green;
                                Console.WriteLine("> the contact " +
                                    " changed successfully!");
                                break;
                            }
                            else if (inputForEdit == "2")
                            {
                                Console.WriteLine();
                                Console.WriteLine("You sure you wanna " +
                                    " delete this contact? (enter yes or no)");
                                while (true)
                                {
                                    Console.Write("> ");
                                    inputForEdit = Console.ReadLine();
                                    if (inputForEdit == "yes")
                                    {
                                        names[i] = numbers[i] = null;
                                        Console.ForegroundColor = ConsoleColor.Green;
                                        Console.WriteLine("> The contact " +
                                            " deletet successfully!");
                                        break;
                                    }
                                    else if (inputForEdit == "no")
                                    {
                                        break;
```

```
                                }
                                else
                                {
                                    Console.ForegroundColor = ConsoleColor.Red;
                                    Console.WriteLine("> Invalid Input!" +
                                        " Enter yes or no");
                                    Console.ForegroundColor = ConsoleColor.Cyan;
                                    continue;
                                }
                            }
                            break;
                        }
                        else if (inputForEdit == "3")
                        {
                            break;
                        }
                        else
                        {
                            Console.WriteLine();
                            Console.ForegroundColor = ConsoleColor.Red;
                            Console.WriteLine("> Invalid Input!");
                            Console.WriteLine("> You have to enter" +
                                " a number between 1 and 3");
                            Console.WriteLine();
                            Console.ForegroundColor = ConsoleColor.Cyan;
                            continue;
                        }
                    }
                }
            }

            if (!found)
            {
                Console.ForegroundColor = ConsoleColor.Red;
                Console.WriteLine("> Not Found!");
            }
            break;

            // Showing all contacts
        case "3":
            Console.Clear();
            for (int i = 0; i < arraySize; i++)
            {
                if (names[i] == null && numbers[i] == null)
                {
                    Console.WriteLine("contact {0} is empty!", i + 1);
                    Console.WriteLine("----------------------");
                }
                else
                {
                    Console.WriteLine("Name: " + names[i]);
                    Console.WriteLine("Number: " + numbers[i]);
                    Console.WriteLine("----------------------");
                }
            }
            break;

            // Exit from the program
        case "4":
            Console.ForegroundColor = ConsoleColor.Yellow;
            Console.WriteLine();
            Console.WriteLine("Thank you for using our software! Bye");
```

```
                    Environment.Exit(0);
                    break;
                default:
                    Console.ForegroundColor = ConsoleColor.Red;
                    Console.Write("> Fail, Invalid Input!");
                    break;
            }
            Console.ReadLine();
        }
    }
}
```

مسلم است که از خیلی حالات مختلف در این برنامه صرف نظر شده است چراکه باعث پیچیده‌تر و نامفهوم‌تر شدن برنامه می‌شود.

کلیه حقوق مادی و معنوی برای وب‌سایت وب‌تارگت محفوظ است.
استفاده از این مطلب در سایر وب‌سایت‌ها و نشریات چاپی تنها با ذکر و درج لینک منبع مجاز است.