ج۲ مجازی سیشارپ
(برنامه نویسی پیشرفته)
کاردانی نرم ۱

مدرس:

فاطمه دهقانی فیروزآبادی

تکرار

ساختارهای تکرار به شما اجازه می‌دهند که یک یا چند دستور کد را تا زمانی که یک شرط برقرار است تکرار کنید. بدون ساختارهای تکرار شما مجبورید همان تعداد کدها را بنویسید که بسیار خسته کننده است. مثلاً شما مجبورید 10 بار جمله "Hello World." را تایپ کنید مانند مثال زیر :

```
Console.WriteLine("Hello World.");
Console.WriteLine("Hello World.");
Console.WriteLine("Hello World.");
Console.WriteLine("Hello World.");
Console.WriteLine("Hello World.");
Console.WriteLine("Hello World.");
Console.WriteLine("Hello World.");
Console.WriteLine("Hello World.");
Console.WriteLine("Hello World.");
Console.WriteLine("Hello World.");
```

البته شما می‌توانید با کپی کردن این تعداد کد را راحت بنویسید ولی این کار در کل کیفیت کدنویسی را پایین می‌آورد. راه بهتر برای نوشتن کدهای بالا استفاده از حلقه‌ها است. حلقه‌ها در سی شارپ عبارتند از :

- while
- do while
- for
- foreach

درباره حلقه foreach، بعد از مبحث آرایه‌ها توضیح می‌دهیم.

حلقه While

ابتدایی‌ترین ساختار تکرار در سی شارپ حلقه While است. ابتدا یک شرط را مورد بررسی قرار می‌دهد و تا زمانیکه شرط برقرار باشد کدهای درون بلوک اجرا می‌شوند. ساختار حلقه While به صورت زیر است :

```
while(condition)
{
    code to loop;
}
```

می‌بینید که ساختار While مانند ساختار if بسیار ساده است. ابتدا یک شرط را که نتیجه آن یک مقدار بولی است می‌نویسیم اگر نتیجه درست یا true باشد سپس کدهای داخل بلوک While اجرا می‌شوند. اگر شرط غلط یا false باشد وقتی که برنامه به حلقه While برسد هیچکدام از کدها را اجرا نمی‌کند. برای متوقف شدن حلقه باید مقادیر داخل حلقه While اصلاح شوند.

اگر مقدار شمارنده یک بماند و آن را افزایش ندهیم و یا مقدار شرط هرگز false نشود یک حلقه بینهایت به وجود می‌آید. به این نکته توجه کنید که در شرط بالا به جای علامت < از => استفاده شده است. اگر از علامت < استفاده می‌کردیم کد ما 9 بار تکرار می‌شد چون مقدار اولیه 1 است و هنگامی که شرط به 10 برسد false می‌شود چون 10 < 10 نیست. اگر می‌خواهید یک حلقه بی نهایت ایجاد کنید که هیچگاه متوقف نشود باید یک شرط ایجاد کنید که همواره درست (true) باشد.

```
while(true)
{
   //code to loop
}
```

این تکنیک در برخی موارد کارایی دارد و آن زمانی است که شما بخواهید با استفاده از دستورات break و return که در آینده توضیح خواهیم داد از حلقه خارج شوید.

```
using System;

public class Program
{
    public static void Main()
    {
        int counter = 1;

        while (counter <= 10)
        {
            Console.WriteLine("Hello World!");
            counter++;
        }

    }
}
```

```
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

برنامه بالا 10 بار پیغام Hello World! را چاپ می‌کند. اگر از حلقه در مثال بالا استفاده نمی‌کردیم مجبور بودیم تمام 10 خط را تایپ کنیم. اجازه دهید که نگاهی به کدهای برنامه فوق بیندازیم. ابتدا در خط 7 یک متغیر تعریف و از آن به عنوان شمارنده حلقه استفاده شده است. سپس به آن مقدار 1 را اختصاص می‌دهیم چون اگر مقدار نداشته باشد نمی‌توان در شرط از آن استفاده کرد.

در خط 9 حلقه While را وارد می‌کنیم. در حلقه While ابتدا مقدار اولیه شمارنده با 10 مقایسه می‌شود که آیا از 10 کمتر است یا با آن برابر است. نتیجه هر بار مقایسه ورود به بدنه حلقه While و چاپ پیغام است. همانطور که مشاهده می‌کنید بعد از هر بار مقایسه مقدار شمارنده یک واحد اضافه می‌شود (خط 12). حلقه تا زمانی تکرار می‌شود که مقدار شمارنده از 10 کمتر باشد.

حلقه do while

حلقه do while یکی دیگر از ساختارهای تکرار است. این حلقه بسیار شبیه حلقه while است با این تفاوت که در این حلقه ابتدا کد اجرا می شودو سپس شرط مورد بررسی قرار می‌گیرد. ساختار حلقه do while به صورت زیر است :

```
do
{
   code to repeat;

} while (condition);
```

همانطور که مشاهده می‌کنید شرط در آخر ساختار قرار دارد. این بدین معنی است که کدهای داخل بدنه حداقل یکبار اجرا می‌شوند. برخلاف حلقه while که اگر شرط نادرست باشد دستورات داخل بدنه اجرا نمی‌شوند. برای اثبات این موضوع به کدهای زیر توجه کنید :

```
int number = 1;
do
{
    Console.WriteLine("Hello World!");
} while (number > 10);
```

```
Hello World!
```

با اجرای کد بالا، اول دستورات بلوک do اجرا می‌شوند و بعد مقدار number با عدد 10 مقایسه می‌شود. در نتیجه حتی اگر شرط نادرست باشد باز هم قسمت do حداقل یک بار اجرا می‌شوند.

```
int number = 1;
while (number > 10)
{
    Console.WriteLine("Hello World!");
}
```

اما در کد بالا چون اول مقدار number ابتدا مورد مقایسه قرار می‌گیرد، اگر شرط درست نباشد دیگر کدی اجرا نمی‌شود. یکی از موارد برتری استفاده از حلقه do while نسبت به حلقه while زمانی است که شما بخواهید اطلاعاتی از کاربر دریافت کنید. در دو کد زیر، یک عملیات یکسان توسط دو حلقه while و do while پیاده سازی شده است :

```
//while version

Console.WriteLine("Enter a number greater than 10: ");
number = Convert.ToInt32(Console.ReadLine());

while(number < 10)
{
   Console.WriteLine("Enter a number greater than 10: ");
   number = Convert.ToInt32(Console.ReadLine());
}
```

```
//do while version

do
{
   Console.WriteLine("Enter a number greater than 10: ");
   number = Convert.ToInt32(Console.ReadLine());
} while (number < 10);
```

حلقه for

یکی دیگر از ساختارهای تکرار حلقه for است. این حلقه عملی شبیه به حلقه while انجام می‌دهد و فقط دارای چند خصوصیت اضافی است. ساختار حلقه for به صورت زیر است :

```
for(initialization; condition; operation)
{
   code to repeat;
}
```

مقدار دهی اولیه (initialization) اولین مقداری است که به شمارنده حلقه می‌دهیم. شمارنده فقط در داخل حلقه for قابل دسترسی است.

شرط (condition) در اینجا مقدار شمارنده را با یک مقدار دیگر مقایسه می‌کند و تعیین می‌کند که حلقه ادامه یابد یا نه.

عملگر (operation) که مقدار اولیه متغیر را کاهش یا افزایش می‌دهد.

در زیر یک مثال از حلقه for آمده است:

```
using System;

public class Program
{
    public static void Main()
    {
        for(int i = 1; i <= 10; i++)
        {
            Console.WriteLine("Number " + i);
        }
    }
}
```

در زیر یک مثال از حلقه for آمده است:

```
using System;

public class Program
{
    public static void Main()
    {
        for(int i = 1; i <= 10; i++)
        {
            Console.WriteLine("Number " + i);
        }
    }
}
```

```
Number 1
Number 2
Number 3
Number 4
Number 5
Number 6
Number 7
Number 8
Number 9
Number 10
```

برنامه بالا اعداد 1 تا 10 را با استفاده از حلقه for می‌شمارد. ابتدا یک متغیر به عنوان شمارنده تعریف می‌کنیم و آن را با مقدار 1 مقدار دهی اولیه می‌کنیم. سپس با استفاده از شرط آن را با مقدار 10 مقایسه می‌کنیم که آیا کمتر است یا مساوی؟ توجه کنید که قسمت سوم حلقه (i++) فوراً اجرا نمی‌شود. کد اجرا می‌شود و ابتدا رشته Number و سپس مقدار جاری i یعنی 1 را چاپ می‌کند. آنگاه یک واحد به مقدار i اضافه شده و مقدار i برابر 2 می‌شود و بار دیگر i با عدد 10 مقایسه می‌شود و این حلقه تا زمانی که مقدار شرط true شود ادامه می‌یابد

حال اگر بخواهید معکوس برنامه بالا را پیاده سازی کنید یعنی اعداد از

بزرگ به کوچک چاپ شوند باید به صورت زیر عمل کنید :

```
for (int i = 10; i > 0; i--)
{
   //code omitted
}
```

کد بالا اعداد را از 10 به 1 چاپ می‌کند (از بزرگ به کوچک). مقدار اولیه شمارنده را 10 می‌دهیم و با استفاده از عملگر کاهش (–) برنامه‌ای که شمارش معکوس را انجام می‌دهد ایجاد می‌کنیم. می‌توان قسمت شرط و عملگر را به صورت‌های دیگر نیز تغییر داد. به عنوان مثال می‌توان از عملگرهای منطقی در قسمت شرط و از عملگرهای تخصیصی در قسمت عملگر افزایش یا کاهش استفاده کرد. همچنین می‌توانید از چندین متغیر در ساختار حلقه for استفاده کنید.

```
for (int i = 1, y = 20; i < 10 && y >= 2; i++, y -= 2)
{
   //some code here
}
```

به این نکته توجه کنید که اگر از چندین متغیر شمارنده یا عملگر در حلقه for استفاده می‌کنید باید آنها را با استفاده از کاما از هم جدا کنید.

حلقه های تو در تو (Nested Loops)

سی شارپ به شما اجازه می‌دهد که از حلقه‌ها به صورت تو در تو استفاده کنید. اگر یک حلقه در داخل حلقه دیگر قرار بگیرد، به آن حلقه تو در تو گفته می‌شود. در این نوع حلقه‌ها، به ازای اجرای یک بار حلقه بیرونی، حلقه داخلی به طور کامل اجرا می‌شود. در زیر نحوه ایجاد حلقه تو در تو آمده است :

```
for (init; condition; increment)
{
   for (init; condition; increment)
   {
      //statement(s);
   }
   //statement(s);
}
```

```
while(condition)
{
   while(condition)
   {
      //statement(s);
   }
   //statement(s);
}
```

```
do
{
   //statement(s);
   do
   {
      //statement(s);
   }
   while(condition);

}
while(condition);
```

نکته‌ای که در مورد حلقه‌های تو در تو وجود دارد این است که، می‌توان از یک نوع حلقه در داخل نوع دیگر استفاده کرد. مثلاً می‌توان از حلقه for در داخل حلقه while استفاده نمود. در مثال زیر نحوه استفاده از این حلقه‌ها ذکر شده است. فرض کنید که می‌خواهید یک مستطیل با 3 سطر و 5 ستون ایجاد کنید :

```
1  using System;
2
3  namespace NestedLoopsDemo
4  {
5      class Program
6      {
7          static void Main(string[] args)
8          {
9              for (int i = 1; i <= 4; i++)
10             {
11                 for (int j = 1; j <= 5; j++)
12                 {
13                     Console.Write(" * ");
14                 }
15                 Console.Write("\n");
16             }
17         }
18     }
19 }
```

```
 *  *  *  *  *
 *  *  *  *  *
 *  *  *  *  *
 *  *  *  *  *
```

در کد بالا به ازای یک بار اجرای حلقه for اول (خط 9)، حلقه for دوم (11-14) به طور کامل اجرا می‌شود. یعنی وقتی مقدار i برابر عدد 1 می‌شود، علامت * توسط حلقه دوم 5 بار چاپ می‌شود، وقتی i برابر 2 می‌شود، دوباره علامت * پنج بار چاپ می‌شود و ... . در کل منظور از دو حلقه for این است که در 4 سطر علامت * در 5 ستون چاپ شود یا 4 سطر ایجاد شود و در هر سطر 5 بار علامت * چاپ شود. خط 15 هم برای ایجاد خط جدید است. یعنی وقتی حلقه داخلی به طور کامل اجرا شد، یک خط جدید ایجاد می‌شود و علامت‌های * در خطوط جدید چاپ می‌شوند. البته به جای این خط می‌توان Console.WriteLine(); را هم نوشت.

خارج شدن از حلقه با استفاده از break و continue

گاهی اوقات با وجود درست بودن شرط می‌خواهیم حلقه متوقف شود. سؤال اینجاست که چطور این کار را انجام دهید؟ با استفاده از کلمه کلیدی break حلقه را متوقف کرده و با استفاده از کلمه کلیدی continue می‌توان بخشی از حلقه را رد کرد و به مرحله بعد رفت. برنامه زیر نحوه استفاده از continue و break را نشان می‌دهد :

```
using System;

public class Program
{
    public static void Main()
    {
        Console.WriteLine("Demonstrating the use of break.n");

        for (int x = 1; x < 10; x++)
        {
            if (x == 5)
                break;

            Console.WriteLine("Number " + x);
        }

        Console.WriteLine("nDemonstrating the use of continue.n");

        for (int x = 1; x < 10; x++)
        {
            if (x == 5)
                continue;

            Console.WriteLine("Number " + x);
        }
    }
}
```

```
Demonstrating the use of break.

Number 1
Number 2
Number 3
Number 4

Demonstrating the use of continue.

Number 1
Number 2
Number 3
Number 4
Number 6
Number 7
Number 8
Number 9
```

در این برنامه از حلقه for برای نشان دادن کاربرد دو کلمه کلیدی فوق استفاده شده است اگر به جای for از حلقه‌های while و do...while استفاده می‌شد نتیجه یکسانی به دست می‌آمد. همانطور که در شرط برنامه (خط 11) آمده است، وقتی که مقدار x به عدد 5 برسد، سپس دستور break اجرا (خط 12) و حلقه بلافاصله متوقف می‌شود، حتی اگر شرط x < 10 برقرار باشد. از طرف دیگر در خط 22 حلقه for فقط برای یک تکرار خاص متوقف شده و سپس ادامه می‌یابد. (وقتی مقدار x برابر 5 شود حلقه از 5 رد شده و مقدار 5 را چاپ نمی‌کند و بقیه مقادیر چاپ می‌شوند.

آرایه ها

آرایه نوعی متغیر است که لیستی از آدرسهای مجموعه‌ای از داده‌های هم نوع را در خود ذخیره می‌کند. تعریف چندین متغیر از یک نوع برای هدفی یکسان بسیار خسته کننده است. مثلاً اگر بخواهید صد متغیر از نوع اعداد صحیح تعریف کرده و از آنها استفاده کنید. مطمئناً تعریف این همه متغیر بسیار کسالت آور و خسته کننده است. اما با استفاده از آرایه می‌توان همه آنها را در یک خط تعریف کرد. در زیر راهی ساده برای تعریف یک آرایه نشان داده شده است :

```
datatype[] arrayName = new datatype[length];
```

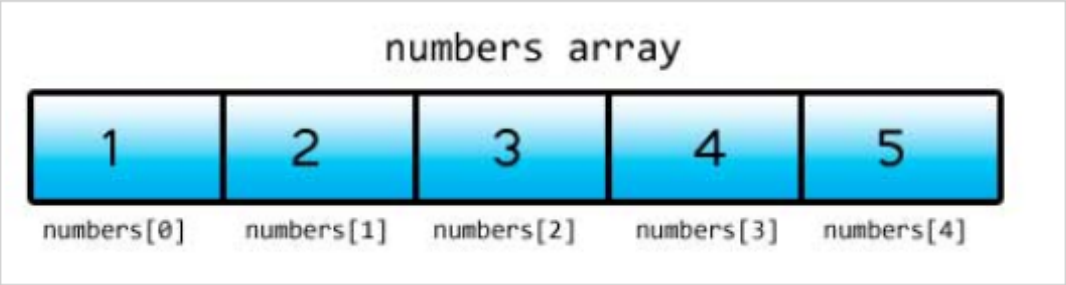
Datatype نوع داده‌هایی را نشان می‌دهد که آرایه در خود ذخیره می‌کند. کروشه که بعد از نوع داده قرار می‌گیرد و نشان دهنده استفاده از آرایه است.arrayName که نام آرایه را نشان می‌دهد. هنگام نامگذاری آرایه بهتر است که نام آرایه نشان دهنده نوع آرایه باشد. به عنوان مثال برای نامگذاری آرایه‌ای که اعداد را در خود ذخیره می‌کند از کلمه number استفاده کنید. طول آرایه که به کامپایلر می‌گوید شما قصد دارید چه تعداد داده یا مقدار را در آرایه ذخیره کنید. از کلمه کلیدی new هم برای اختصاص فضای حافظه به اندازه طول آرایه استفاده می‌شود. برای تعریف یک آرایه که 5 مقدار از نوع اعداد صحیح در خود ذخیره می‌کند باید به صورت زیر عمل کنیم :

```
int[] numbers = new int[5];
```

در این مثال 5 آدرس از فضای حافظه کامپیوتر شما برای ذخیره 5 مقدار رزرو می‌شود. حال چطور مقادیرمان را در هر یک از این آدرسها ذخیره کنیم؟ برای دسترسی و اصلاح مقادیر آرایه از اندیس یا مکان آنها استفاده می‌شود.

```
numbers[0] = 1;
numbers[1] = 2;
numbers[2] = 3;
numbers[3] = 4;
numbers[4] = 5;
```

اندیس یک آرایه از صفر شروع شده و به یک واحد کمتر از طول آرایه ختم می‌شود. به عنوان مثال شما یک آرایه 5 عضوی دارید، اندیس آرایه از 0 تا 4 می‌باشد چون طول آرایه 5 است پس 1-5 برابر است با 4. این بدان معناست که اندیس 0 نشان دهنده اولین عضو آرایه است و اندیس 1 نشان دهنده دومین عضو و الی آخر. برای درک بهتر مثال بالا به شکل زیر توجه کنید :



به هر یک از اجزاء آرایه و اندیسهای داخل کروشه توجه کنید. کسانی که تازه شروع به برنامه نویسی کرده‌اند معمولاً در گذاشتن اندیس دچار اشتباه می‌شوند و مثلاً ممکن است در مثال بالا اندیسها را از 1 شروع کنند. اگر بخواهید به یکی از اجزائ آرایه با استفاده از اندیسی دسترسی پیدا کنید که در محدوده اندیسهای آرایه شما نباشد با پیغام خطای IndexOutOfRangeException مواجه می‌شوید و بدین معنی است که شما آدرسی را می‌خواهید که وجود ندارد. یکی دیگر از راه‌های تعریف سریع و مقدار دهی یک آرایه به صورت زیر است :

```
datatype[] arrayName = new datatype[length] { val1, val2, ... valN };
```

در این روش شما می‌توانید فوراً بعد از تعریف اندازه آرایه مقادیر را در داخل آکولاد قرار دهید. به یاد داشته باشید که هر کدام از مقادیر را با استفاده از کاما از هم جدا کنید. همچنین تعداد مقادیر داخل آکولاد باید با اندازه آرایه تعریف شده برابر باشد. به مثال زیر توجه کنید :

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

این مثال با مثال قبل هیچ تفاوتی ندارد و تعداد خطهای کدنویسی را کاهش می‌دهد. شما می‌توانید با استفاده از اندیس به مقدار هر یک از اجزاء آرایه

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

این مثال با مثال قبل هیچ تفاوتی ندارد و تعداد خطهای کدنویسی را کاهش می‌دهد. شما می‌توانید با استفاده از اندیس به مقدار هر یک از اجزاء آرایه دسترسی یابید و آنها را به دلخواه تغییر دهید. تعداد اجزاء آرایه در مثال بالا 5 است و ما 5 مقدار را در آن قرار می‌دهیم. اگر تعداد مقادیری که در آرایه قرار می‌دهیم کمتر یا بیشتر از طول آرایه باشد با خطا مواجه می‌شویم. یکی دیگر از راه‌های تعریف آرایه در زیر آمده است. شما می‌توانید هر تعداد عنصر را که خواستید در آرایه قرار دهید بدون اینکه اندازه آرایه را مشخص کنید. به عنوان مثال :

```
int[] numbers = new int[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
```

در این مثال ما 10 مقدار را به آرایه اختصاص داده‌ایم. نکته اینجاست که طول آرایه را تعریف نکرده‌ایم. در این حالت کامپایلر بعد از شمردن تعداد مقادیر داخل آکولاد طول آرایه را تشخیص می‌دهد. به یاد داشته باشید که اگر برای آرایه طولی در نظر نگیرید باید برای آن مقدار تعریف کنید در عیر این صورت با خطا مواجه می‌شوید :

```
int[] numbers = new int[]; //not allowed
```

یک راه بسیار ساده‌تر برای تعریف آرایه به صورت زیر است :

```
int[] numbers = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
```

به سادگی و بدون احتیاج به کلمه کلیدی new می‌توان مقادیر را در داخل آکولاد قرار داد. کامپایلر به صورت اتوماتیک با شمارش مقادیر طول آرایه را تشخیص می‌دهد.

دستیابی به مقادیر آرایه با استفاده از حلقه for

در زیر مثالی در مورد استفاده از آرایه‌ها آمده است. در این برنامه 5 مقدار از کاربر گرفته شده و میانگین آنها حساب می‌شود:

```
using System;

public class Program
{
    public static void Main()
    {
        int[] numbers = new int[5];
        int total = 0;
        double average;

        for (int i = 0; i < numbers.Length; i++)
        {
            Console.Write("Enter a number: ");
            numbers[i] = Convert.ToInt32(Console.ReadLine());
        }

        for (int i = 0; i < numbers.Length; i++)
        {
            total += numbers[i];
        }

        average = total / (double)numbers.Length;

        Console.WriteLine("Average = {0}", average);
    }
}
```

```
Enter a number: 90
Enter a number: 85
Enter a number: 80
Enter a number: 87
Enter a number: 92
Average = 86
```

حلقه foreach

حلقه foreach یکی دیگر از ساختارهای تکرار در سی شارپ می‌باشد که مخصوصاً برای آرایه‌ها، لیست‌ها و مجموعه‌ها طراحی شده است. حلقه foreach با هر بار گردش در بین اجزاء، مقادیر هر یک از آنها را در داخل یک متغیر موقتی قرار می‌دهد و شما می‌توانید بواسطه این متغیر به مقادیر دسترسی پیدا کنید. در زیر نحوه استفاده از حلقه foreach آمده است :

```
foreach (datatype temporaryVar in array)
{
    code to execute;
}
```

temporaryVar متغیری است که مقادیر اجزای آرایه را در خود نگهداری می‌کند.temporaryVar باید دارای نوع باشد تا بتواند مقادیر آرایه را در خود ذخیره کند. به عنوان مثال آرایه شما دارای اعدادی از نوع صحیح باشد باید نوع متغیر موقتی از نوع اعداد صحیح باشد یا هر نوع دیگری که بتواند اعداد صحیح را در خود ذخیره کند مانند double یا long. سپس کلمه کلیدی in و بعد از آن نام آرایه را می‌نویسیم. در زیر نحوه استفاده از حلقه foreach آمده است :

```
1   using System;
2
3   public class Program
4   {
5       public static void Main()
6       {
7           int[] numbers = { 1, 2, 3, 4, 5 };
8
9           foreach (int n in numbers)
10          {
11              Console.WriteLine("Number {0}", n);
12          }
13      }
14  }
```

```
Number 1
Number 2
Number 3
Number 4
Number 5
```

در برنامه آرایه‌ای با 5 جزء تعریف شده و مقادیر 1 تا 5 در آنها قرار داده شده است (خط 7). در خط ۹ حلقه foreach شروع می‌شود. ما یک متغیر موقتی تعریف کرده‌ایم که اعداد آرایه را در خود ذخیره می‌کند. در هر بار تکرار از حلقه foreach متغیر موقتی n ، مقادیر عددی را از آرایه استخراج می‌کند. حلقه foreach مقادیر اولین تا آخرین جزء آرایه را در اختیار ما قرار می‌دهد.

حلقه foreach برای دریافت هر یک از مقادیر آرایه کاربرد دارد. بعد از گرفتن مقدار یکی از اجزای آرایه، مقدار متغیر موقتی را چاپ می‌کنیم. حلقه foreach یک ضعف دارد و آن اینست که این حلقه ما را قادر می‌سازد که به داده‌ها دسترسی یابیم و یا آنها را بخوانیم ولی اجازه اصلاح اجزاء آرایه را نمی‌دهد. برای

درک این مطلب در مثال زیر سعی شده است که مقدار هر یک از اجزا آرایه افزایش یابد :

```
int[] numbers = { 1, 2, 3 };

foreach(int number in numbers)
{
   number++;
}
```

اگر برنامه را اجرا کنید با خطا مواجه می‌شوید. برای اصلاح هر یک از اجزا آرایه می‌توان از حلقه for استفاده کرد.

```
int[] numbers = { 1, 2, 3 };

for (int i = 0; i < number.Length; i++)
{
   numbers[i]++;
}
```

این حلقه یه ضعف دیگر نیز دارد و آن این است که ما نمی‌توانیم با استفاده از حلقه foreach به اندیس عناصر آرایه دست یابیم، مثلاً کد زیر که با حلقه for نوشته شده است را نمی‌توانیم با حلقه foreach پیاده سازی کنیم:

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

for (int i = 0; i < number.Length; i++)
{
    Console.WriteLine(number[i]);
}
```

البته این ضعف نیز به نوعی ناشی از غیر قابل ویرایش بودن عناصر آرایه در حلقه foreach است.

آرایه های چند بعدی

آرایه‌های چند بعدی آرایه‌هایی هستند که برای دسترسی به هر یک از عناصر آنها باید از چندین اندیس استفاده کنیم. یک آرایه چند بعدی را می‌توان مانند یک جدول با تعدای ستون و ردیف تصور کنید. با افزایش اندیسها اندازه ابعاد آرایه نیز افزایش می‌یابد و آرایه‌های چند بعدی با بیش از دو اندیس به وجود می‌آیند. نحوه ایجاد یک آرایه با دو بعد به صورت زیر است :

```
datatype[,] arrayName = new datatype[lengthX, lengthY];
```

و یک آرایه سه بعدی به صورت زیر ایجاد می‌شود :

```
datatype[, ,] arrayName = new datatype[lengthX, lengthY, lengthZ];
```

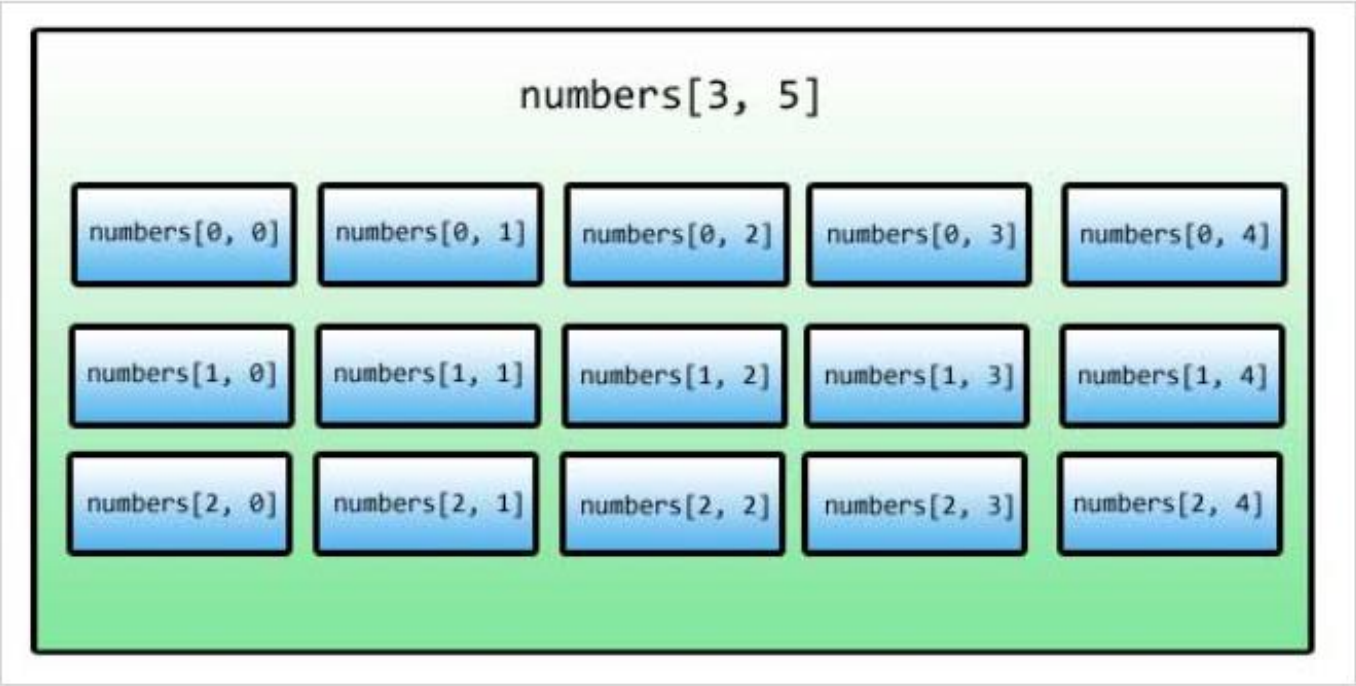
می‌توان یک آرایه با تعداد زیادی بعد ایجاد کرد به شرطی که هر بعد دارای طول مشخصی باشد. به دلیل اینکه آرایه‌های سه بعدی یا آرایه‌های با بیشتر از دو بعد بسیار کمتر مورد استفاده قرار می‌گیرند اجازه بدهید که در این درس بر روی آرایه‌های دو بعدی تمرکز کنیم. در تعریف این نوع آرایه ابتدا نوع آرایه یعنی اینکه آرایه چه نوعی از انواع داده را در خود ذخیره می‌کند را مشخص می‌کنیم. سپس یک جفت کروشه و در داخل کروشه‌ها یک کاما قرار می‌دهیم. به تعداد کاماهایی که در داخل کروشه می‌گذارید توجه کنید. اگر آرایه ما دو بعدی است باید 1 کاما و اگر سه بعدی است باید 2 کاما قرار دهیم. سپس یک نام برای آرایه انتخاب کرده و بعد تعریف آنرا با گذاشتن کلمه new ، نوع داده و طول آن کامل می‌کنیم. در یک آرایه دو بعدی برای دسترسی به هر یک از عناصر به دو مقدار نیاز داریم یکی مقدار X و دیگری مقدار Y که مقدار x نشان دهنده ردیف و مقدار Y نشان دهنده ستون آرایه است البته اگر ما آرایه دو بعدی را به صورت جدول در نظر بگیریم. یک آرایه سه بعدی را می‌توان به صورت یک مکعب تصور کرد که دارای سه بعد است و x طول، Y عرض و z ارتفاع آن است.

یک مثال از آرایه دو بعدی در زیر آمده است :

یک مثال از آرایه دو بعدی در زیر آمده است :

```
int[,] numbers = new int[3, 5];
```

کد بالا به کامپایلر می‌گوید که فضای کافی به عناصر آرایه اختصاص بده (در این مثال 15 خانه). در شکل زیر مکان هر عنصر در یک آرایه دو بعدی نشان داده شده است.



مقدار 3 را به x اختصاص می‌دهیم چون 3 سطر و مقدار 5 را به Y چون 5 ستون داریم اختصاص می‌دهیم. چطور یک آرایه چند بعدی را مقدار دهی کنیم؟ چند راه برای مقدار دهی به آرایه‌ها وجود دارد.

```
datatype[,] arrayName = new datatype[x, y] { { r0c0, r0c1, ... r0cX },
                                             { r1c0, r1c1, ... r1cX },
                                                      .
                                                      .
                                                      .
                                             { rYc0, rYc1, ... rYcX } };
```

برای راحتی کار می‌توان از نوشتن قسمت new datatype[,] صرف نظر کرد.

```
datatype[,] arrayName = { { r0c0, r0c1, ... r0cX },
                          { r1c0, r1c1, ... r1cX },
                                   .
                                   .
                                   .
                          { rYc0, rYc1, ... rYcX } };
```

به عنوان مثال :

```
int[,] numbers = { { 1, 2, 3, 4, 5 },
                   { 6, 7, 8, 9, 10 },
                   { 11, 12, 13, 14, 15 } };
```

و یا می‌توان مقدار دهی به عناصر را به صورت دستی انجام داد مانند :

```
array[0, 0] = value;
array[0, 1] = value;
array[0, 2] = value;
array[1, 0] = value;
array[1, 1] = value;
array[1, 2] = value;
array[2, 0] = value;
array[2, 1] = value;
array[2, 2] = value;
```

گردش در میان عناصر آرایه‌های چند بعدی

گردش در میان عناصر آرایه‌های چند بعدی نیاز به کمی دقت دارد. یکی از راههای آسان استفاده از حلقه foreach و یا حلقه for تو در تو است. اجازه دهید ابتدا از حلقه foreach استفاده کنیم.

```
using System;

public class Program
{
    public static void Main()
    {
        int[,] numbers = { { 1, 2, 3, 4, 5 },
                           { 6, 7, 8, 9, 10 },
                           { 11, 12, 13, 14, 15 }
                         };

        foreach (int number in numbers)
        {
            Console.Write(number + " ");
        }
    }
}
```

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
```

مشاهده کردید که گردش در میان مقادیر عناصر یک آرایه چند بعدی چقدر راحت است. به وسیله حلقه foreach نمی‌توانیم انتهای ردیفها را مشخص کنیم.

برنامه زیر نشان می‌دهد که چطور از حلقه for برای خواندن همه مقادیر آرایه و تعیین انتهای ردیف‌ها استفاده کنید.

```
using System;

public class Program
{
    public static void Main()
    {
        int[,] numbers = { { 1, 2, 3, 4, 5 },
                           { 6, 7, 8, 9, 10 },
                           { 11, 12, 13, 14, 15 }
                         };

        for (int row = 0; row < numbers.GetLength(0); row++)
        {
            for (int col = 0; col < numbers.GetLength(1); col++)
            {
                Console.Write(numbers[row, col] + " ");
            }

            //Go to the next line
            Console.WriteLine();
        }
    }
}
```

```
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
```

همانطور که در مثال بالا نشان داده شده است با استفاده از یک حلقه for نمی‌توان به مقادیر دسترسی یافت بلکه به یک حلقه for تو در تو نیاز داریم، زیرا آرایه دو بعدی به صورت یک جدول شامل سطر و ستون است، پس لازم است که از یک حلقه for برای گردش در میان سطرها و از حلقه for دیگر برای گردش در میان ستون‌های این جدول (آرایه) استفاده کنیم.

اولین حلقه for (خط 12) برای گردش در میان ردیف‌های آرایه به کار می‌رود. این حلقه تا زمانی ادامه می‌یابد که مقدار ردیف کمتر از طول اولین بعد باشد (زیرا اندیس ابعاد آرایه از صفر شروع می‌شود. در مثال بالا مقدار اولین بعد برابر 3 است). در این مثال از متد GetLength() کلاس Array استفاده کرده‌ایم.

این متد طول آرایه را در یک بعد خاص نشان می‌دهد و دارای یک پارامتر است که همان بعد آرایه می‌باشد. به عنوان مثال برای به دست آوردن طول اولین بعد آرایه مقدار صفر را به این متد ارسال می‌کنیم چون اندیس ابعاد آرایه از صفر شروع می‌شود.

در داخل اولین حلقه for حلقه for دیگری تعریف شده است (خط 14). در این حلقه یک شمارنده برای شمارش تعداد ستونهای (col) هر ردیف تعریف شده است و در شرط داخل آن بار دیگر از متد GetLength() استفاده شده است، ولی این بار مقدار 1 را به آن ارسال می‌کنیم تا طول بعد دوم آرایه را به دست آوریم. پس به عنوان مثال وقتی که مقدار ردیف (row) صفر باشد، حلقه دوم از [0, 0] تا [0, 4] اجرا می‌شود. سپس مقدار هر عنصر از آرایه را با استفاده از حلقه نشان می‌دهیم، اگر مقدار ردیف(row) برابر 0 و مقدار ستون (col) برابر 0 باشد مقدار عنصری که در ستون 1 و ردیف 1 (numbers[0, 0]) قرار دارد نشان داده خواهد شد که در مثال بالا عدد 1 است.

بعد از اینکه دومین حلقه تکرار به پایان رسید، فوراً دستورات بعد از آن اجرا خواهند شد، که در اینجا دستور Console.WriteLine() که به برنامه اطلاع می‌دهد که به خط بعد برود. سپس حلقه با اضافه کردن یک واحد به مقدار row این فرایند را دوباره تکرار می‌کند. سپس دومین حلقه for اجرا شده و مقادیر دومین ردیف نمایش داده می‌شود. این فرایند تا زمانی اجرا می‌شود که مقدار row کمتر از طول اولین بعد باشد. حال بیایید آنچه را از قبل یاد گرفته‌ایم در یک برنامه به کار بریم. این برنامه نمره چهار درس مربوط به سه دانش آموز را از ما می‌گیرد و معدل سه دانش آموز را حساب می‌کند.

این برنامه نمره چهار درس مربوط به سه دانش آموز را از ما می‌گیرد و معدل سه دانش آموز را حساب می‌کند.

```
using System;

public class Program
{
    public static void Main()
    {
        double[,] studentGrades = new double[3, 4];
        double total;

        for (int student = 0; student < studentGrades.GetLength(0); student++)
        {
            total = 0;

            Console.WriteLine("Enter grades for Student {0}", student + 1);

            for (int grade = 0; grade < studentGrades.GetLength(1); grade++)
            {
                Console.Write("Enter Grade #{0}: ", grade + 1);
                studentGrades[student, grade] = Convert.ToDouble(Console.ReadLine());
                total += studentGrades[student, grade];
            }

            Console.WriteLine("Average is {0:F2}",
                              (total / studentGrades.GetLength(1)));
            Console.WriteLine();
        }

    }
}
```

```
Enter grades for Student 1
Enter Grade #1: 92
Enter Grade #2: 87
Enter Grade #3: 89
Enter Grade #4: 95
Average is 90.75

Enter grades for Student 2
Enter Grade #1: 85
Enter Grade #2: 85
Enter Grade #3: 86
Enter Grade #4: 87
Average is 85.75

Enter grades for Student 3
Enter Grade #1: 90
Enter Grade #2: 90
Enter Grade #3: 90
Enter Grade #4: 90
Average is 90.00
```

در برنامه بالا یک آرایه چند بعدی از نوع double تعریف شده است (خط 7). همچنین یک متغیر به نام total تعریف می‌کنیم که جمع نمرات وارد شده برای دانش آموز در آن قرار می‌گیرد. حال وارد حلقه for تو در تو می‌شویم (خط 10). در اولین حلقه for یک متغیر به نام student تعریف کرده‌ایم که مقادیر اولین بعد آرایه (که همان تعداد دانش آموزان است) در آن قرار می‌گیرد. از متد ()GetLength هم برای تشخیص تعداد دانش آموزان استفاده شده است. وارد بدنه حلقه for می‌شویم. در خط 12 مقدار متغیر total را برابر صفر قرار می‌دهیم. سپس برنامه یک پیغام را نشان می‌دهد و از شما می‌خواهد که نمرات دانش آموز را وارد کنید (student + 1). عدد 1 را به student اضافه کرده‌ایم تا به جای نمایش Student 0، با Student 1 شروع شود، تا طبیعی‌تر به نظر برسد. سپس به دومین حلقه for در خط 16 می‌رسیم. وظیفه این حلقه گردش در میان دومین بعد که همان نمرات دانش آموز است می‌باشد. برنامه چهار نمره مربوط به دانش آموز را می‌گیرد. هر وقت که برنامه یک نمره را از کاربر دریافت می‌کند، نمره به متغیر total اضافه می‌شود.

وقتی همه نمره‌ها وارد شدند، متغیر total هم جمع همه نمرات را نشان می‌دهد. در خطوط 24-23 معدل دانش آموز نشان داده می‌شود. به فرمت {0:F2} توجه کنید. این فرمت معدل را تا دو رقم اعشار نشان می‌دهد. معدل از تقسیم کردن total (جمع) بر تعداد نمرات به دست می‌آید. از متد GetLength(1) هم برای به دست آوردن تعداد نمرات استفاده می‌شود.

آرایه دندانه دار

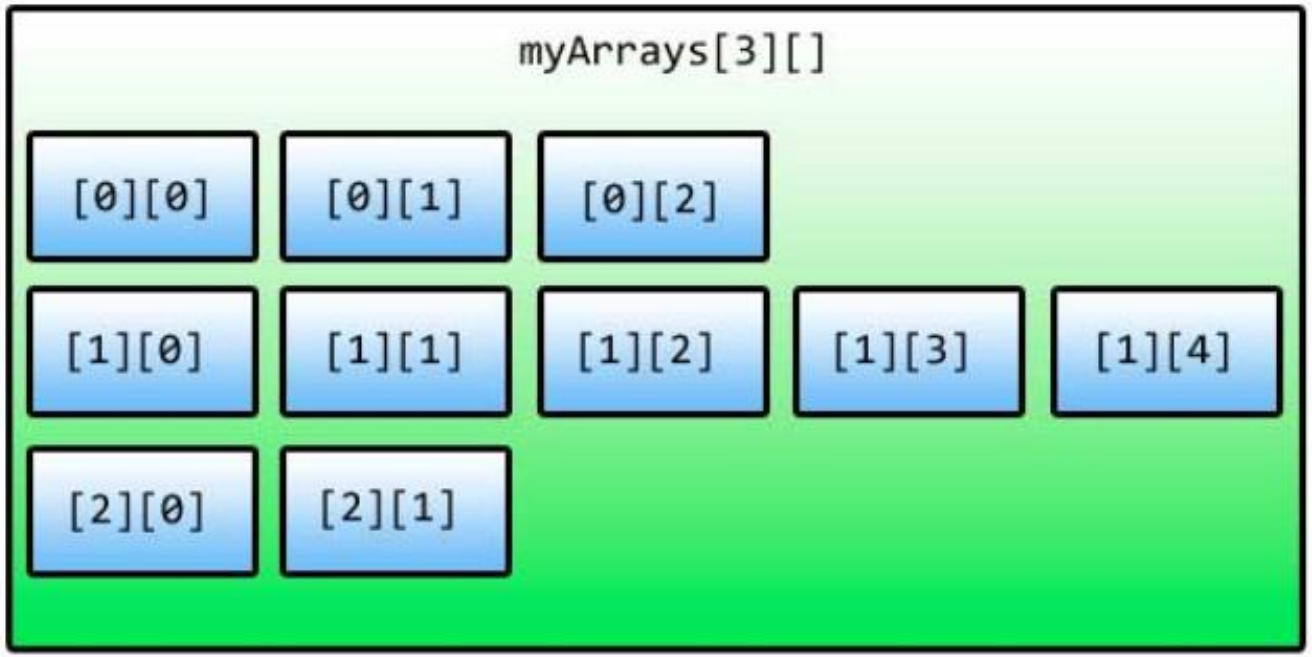
آرایه دندانه دار یا jagged array آرایه‌ای چند بعدی است که دارای سطرهای با طول متغیر می‌باشد. نمونه ساده‌ای از آرایه‌های چند بعدی، آرایه‌های مستطیلی است که تعداد ستون‌های سطرهای آنها برابر است. اما آرایه‌های دندانه دار دارای سطرهایی با طول متفاوت می‌باشند. بنابر این آرایه‌های دندانه دار را می‌توان آرایه‌ای از آرایه‌ها فرض کرد. دستور نوشتن این نوع آرایه‌ها به صورت زیر است :

```
datatype[][] arrayName;
```

ابتدا datatype که نوع آرایه است و سپس چهار کروشه باز و بسته و بعد از آن نام آرایه را می‌نویسیم. مقداردهی به این آرایه‌ها کمی گیج کننده است. به مثال زیر توجه کنید :

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

myArrays[0] = new int[3];
myArrays[1] = new int[5];
myArrays[2] = new int[2];
```



ابتدا با استفاده از کلمه کلیدی new سطرهای آرایه را مشخص می‌کنیم. بعد از کلمه کلیدی new نوع آرایه و سپس در اولین کروشه باز و بسته تعداد سطرها را می‌نویسیم. سپس تعداد ستون‌های هر سطر را با استفاده از اندیس هر سطر به صورت بالا مشخص می‌کنیم. سپس بعد از تعریف ستون‌ها به صورت زیر می‌توان با استفاده از آکولاد مقادیر هر سطر را مشخص کرد :

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

myArrays[0] = new int[3] { 1, 2, 3 };
myArrays[1] = new int[5] { 5, 4, 3, 2, 1 };
myArrays[2] = new int[2] { 11, 22 };
```

یک روش بهتر برای مقدار دهی آرایه‌های دندانه دار به صورت زیر است :

```
int[][] myArrays = new int[3][] { new int[3] { 1, 2, 3 },
                                  new int[5] { 5, 4, 3, 2, 1 },
                                  new int[2] { 11, 22 } };
```

می‌توان طول سطرها را هم مشخص نکرد :

```
int[][] myArrays = new int[][] { new int[] { 1, 2, 3 },
                                 new int[] { 5, 4, 3, 2, 1 },
                                 new int[] { 11, 22 } };
```

کد بالا را به صورت سادهتر زیر هم می‌توان نوشت :

```
int[][] myArrays = { new int[] { 1, 2, 3 },
                     new int[] { 5, 4, 3, 2, 1 },
                     new int[] { 11, 22 } };
```

برای دسترسی به مقدار عناصر یک آرایه دندانه دار باید اندیس سطر و ستون آن را در اختیار داشته باشیم :

```
array[row][column]
Console.WriteLine(myArrays[1][2]);
```

نمی‌توان از حلقه foreach برای دسترسی به عناصر آرایه دندانه دار استفاده کرد :

```
foreach(int array in myArrays)
{
    Console.WriteLine(array);
}
```

اگر از حلقه foreach استفاده کنیم با خطا مواجه می‌شویم چون عناصر این نوع آرایه‌ها، آرایه هستند نه عدد یا رشته یا... . برای حل این مشکل باید نوع متغیر موقتی (array) را تغییر داده و از حلقه foreach دیگری برای دسترسی به مقادیر استفاده کرد.

```
foreach(int[] array in myArrays)
{
   foreach(int number in array)
   {
      Console.WriteLine(number);
   }
}
```

همچنین می‌توان از یک حلقه for تو در تو به صورت زیر استفاده کرد :

```
for (int row = 0; row < myArrays.Length; row++)
{
   for (int col = 0; col < myArrays[row].Length; col++)
   {
      Console.WriteLine(myArrays[row][col]);
   }
}
```

در اولین حلقه از خاصیت Length برای به دست آوردن تعداد سطرها (که همان آرایه‌های یک بعدی هستند) و در دومین حلقه از خاصیت Length برای به دست آوردن عناصر سطر جاری استفاده می‌شود.

تمرین

- برنامه ای بنویسید که دو مرتبه پنج عدد از کاربر بگیرد و آنها را در دو آرایه قرار دهد و اعداد متناظر با هم را در آرایه ها جمع کند و در آرایه جدید قرار دهد