



آموزش ASP.NET Core2 - فصل صفر (آموزش سی شارپ)

مدرس: آرزو ابراهیمی

جلسات آموزشی سی شارپ (پیش نیاز)

جلسه یازدهم

جلسه نهم

جلسه ششم

جلسه اول

جلسه دوازدهم

جلسه دهم

جلسه هفتم

جلسه دوم

جلسه سیزدهم

جلسه هشتم

جلسه سوم

جلسه چهاردهم

جلسه پنجم

سی شارپ چیست ؟

سی شارپ (C#) یک زبان برنامه‌نویسی شی گراست ، که توسط شرکت مایکروسافت ساخته شده و ترکیبی از قابلیت‌های خوب C++ و JAVA است.

اگر با این دو زبان آشنایی دارید، این شانس را دارید که زبان C# را راحت یاد بگیرید.

این زبان به قدری راحت است که هم کسانی که قبلاً برنامه‌نویسی نکرده‌اند و هم دانش آموزان می‌توانند راحت آن را یاد بگیرند.

دات نت فریم ورک (.NET Framework) چیست؟

.NET Framework یک چارچوب است که توسط شرکت مایکروسافت برای توسعه انواع نرم افزارها علی الخصوص ویندوز طراحی شد.

.NET Framework همچنین میتواند برای توسعه نرم افزارهای تحت وب مورد استفاده قرار بگیرد.

تا کنون چندین نسخه از .NET Framework انتشار یافته که هر بار قابلیت‌های جدیدی به آن اضافه شده است.

دات نت فریم ورک (NET Framework) چیست؟

NET Framework شامل کتابخانه کلاس محیط کاری ((Framework Class Library (FCL)) که در بر گیرنده کلاس ها ، ساختارها ، داده های شمارشی و ... می باشد.

مهم ترین قسمت NET Framework زبان مشترک زمان اجرا ((Common Language Runtime(CLR)) است که محیطی را فراهم می آورد که برنامه ها در آن اجرا شوند.

تاریخچه زبان سی شارپ

در سال ۱۹۹۹م، شرکت (Sun Microsystems) اجازه استفاده از زبان برنامه‌نویسی جاوا را در اختیار شرکت میکروسافت قرار داد. تا در سیستم عامل خود از آن استفاده کند. جاوا در اصل به هیچ پلت فرم یا سیستم عاملی وابسته نبود، ولی میکروسافت برخی از مفاد قرار داد را زیر پا گذاشت و قابلیت مستقل از سیستم عامل بودن جاوا را از آن برداشت. شرکت Sun Microsystems پرونده‌ای علیه میکروسافت درست کرد و میکروسافت مجبور شد تا زبان شی گرای جدیدی با کامپایل جدید که به C++ شبیه بود را درست کند.

نسخه های مختلف زبان سی شارپ

نسخه سی شارپ	نسخه .NET Framework	نسخه Visual Studio	تاریخ ارائه
C# 1.0	.NET Framework 1.0	Visual Studio.NET 2002	January 2002
C# 1.1	.NET Framework 1.1	Visual Studio.NET 2003	April 2003
C# 2.0	.NET Framework 2.0	Visual Studio 2005	November 2005
C# 3.0	.NET Framework 3.0\3.5	Visual Studio 2008	November 2007
C# 4.0	.NET Framework 4.0	Visual Studio 2010	April 2010
C# 5.0	.NET Framework 4.5	Visual Studio 2012/2013	August 2012
C# 6.0	.NET Framework 4.6	Visual Studio 2015	July 2015
C# 7.0	.NET Framework 4.6.2	Visual Studio 2017	March 2017
C# 7.1	.NET Framework 4.6.2	Visual Studio 2017	August 2017
C# 7.2	.NET Framework 4.7.1	Visual Studio 2017	November 2017

نحوه تبدیل کدهای سی شارپ به یک برنامه اجرایی

- ▶ 1. برنامه نویس برنامه خود را با یک زبان دات نت مانند سی شارپ می نویسد.
- ▶ 2. کدهای سی شارپ به وسیله کامپایلر به کدهای معادل آن در زبان میانی تبدیل می شوند.
- ▶ 3. کدهای زبان میانی در یک فایل اسمبلی ذخیره می شوند.
- ▶ 4. وقتی برنامه اجرا می شود کامپایلر ((Just In Time (JIT)) کدهای زبان میانی را در لحظه به کدهایی که برای کامپیوتر قابل خواندن باشند تبدیل می کند.

نصب و آشنایی با Visual Studio 2017

Visual Studio 2017

ویژوال استودیو 2017 جدیدترین نسخه از محیط کدنویسی مایکروسافت است که در سه نسخه‌ی Enterprise ، Community و Professional عرضه شده است.

مقایسه نسخه ها از لحاظ فنی [مشاهده](#)

سخت افزارها و نرم افزارهای مورد نیاز

► Software:

Windows 10 version 1507 or higher

Windows Server 2016

Windows 8.1 (with Update 2919355)

Windows Server 2012 R2 (with Update 2919355)

Windows 7 SP1 (with latest Windows Updates)

► Hardware:

۱.۸ GHz or faster processor. Dual-core or better recommended

۲ GB of RAM; 4 GB of RAM recommended (2.5 GB minimum if running on a virtual machine)

Hard disk space: 1GB to 40GB, depending on features installed

Video card that supports a minimum display resolution of 720p (1280 by 720)

Visual Studio will work best at a resolution of WXGA (1366 by 768) or higher

ویژوال استودیو

▶ دانلود نرم افزار Visual Studio 2017 <https://www.visualstudio.com/downloads/>

▶ دانلود نرم افزار از طریق شاتلند [لینک دانلود](#)

پایان جلسه اول

دستور چاپ در سی شارپ

دستور چاپ در سی شارپ

```
System.Console.WriteLine ("دستوری که می خواهیم چاپ شود");
```

```
System.Console.Write ("دستوری که می خواهیم چاپ شود");
```

قواعد مهم در سی شارپ

- ▶ بعد از هر دستور در سی شارپ باید از علامت سیمی کالن (;) استفاده شود.
- ▶ در سی شارپ هر آکولاد باز { دارای یک آکولاد بسته } است.

توضیحات تک خطی ▶

```
// single line comment
```

توضیحات چند خطی ▶

```
/* multi line comment */
```















توضیحات XML ▶

```
/// <summary>  
/// This is XML comments  
/// </summary>
```

استفاده از IntelliSense

شاید یکی از ویژگی های مهم Visual Studio، IntelliSense می باشد. IntelliSense ما را قادر می سازد که به سرعت به کلاس ها و متدها و ... دسترسی پیدا کنیم.

استفاده از IntelliSense

آیکون	مربوط به	آیکون	مربوط به
	پارامترها و متغیرهای محلی ((Locals and Parameters))		ثابت (Constant)
	خاصیت (Property)		رویداد (Event)
	فیلد (Field)		متد (Method)
	رابط (Interface)		کلاس (Class)
	ساختار (Structure)		نوع شمارشی (Enum)
	کلمه کلیدی (Keyword)		کد کوتاه (Code Snippet)
	نماینده (Delegate)		فضای نام (Namespace)

کاراکترهای کنترلی

کاراکترهای کنترلی کاراکترهای ترکیبی هستند که با یک بک اسلش (\) شروع می‌شوند و به دنبال آنها یک حرف یا عدد می‌آید و یک رشته را با فرمت خاص نمایش می‌دهند. برای مثال برای ایجاد یک خط جدید و قرار دادن رشته در آن می‌توان از کاراکتر کنترلی \n استفاده کرد.

کاراکترهای کنترلی

عملکرد	کاراکتر کنترلی	عملکرد	کاراکتر کنترلی
Form Feed	\f	چاپ کوتیشن	\'
خط جدید	\n	چاپ دابل کوتیشن	\"
سر سطر رفتن	\r	چاپ بک اسلش	\\
حرکت به صورت افقی	\t	چاپ فضای خالی	\0
حرکت به صورت عمودی	\v	صدای بیپ	\a
چاپ کاراکتر یونیکد	\u	حرکت به عقب	\b

استفاده از علامت @

- استفاده از علامت @ برای نادیده گرفتن کاراکترهای کنترلی
- استفاده از علامت @ برای نگهداری از قالب بندی رشته‌ها

متغيرها

متغیرها

متغیر مکانی از حافظه است که شما می‌توانید مقادیری را در آن ذخیره کنید. می‌توان آن را به عنوان یک ظرف تصور کرد که داده‌های خود را در آن قرار داده‌اید. محتویات این ظرف می‌تواند پاک شود یا تغییر کند. هر متغیر دارای یک نام نیز هست. که از طریق آن می‌توان متغیر را از دیگر متغیرها تشخیص داد و به مقدار آن دسترسی پیدا کرد.

نام گذاری متغیرها

- ▶ 1. نام متغیر باید با یک از حروف الفبا (a-z or A-Z) شروع شود.
- ▶ 2. نمی‌تواند شامل کاراکترهای غیرمجاز مانند # , ? , ^ , \$. باشد.
- ▶ 3. نمی‌توان از کلمات رزرو شده در سی شارپ برای نام متغیر استفاده کرد.
- ▶ 4. نام متغیر نباید دارای فضای خالی (spaces) باشد.
- ▶ 5. اسامی متغیرها نسبت به بزرگی و کوچکی حروف حساس هستند.
- ▶ در سی شارپ دو حرف مانند a و A دو کاراکتر مختلف به حساب می‌آیند.

متغیر ها - نوع ساده

انواع ساده انواعی از داده‌ها هستند که شامل اعداد، کاراکترها و رشته‌ها و مقادیر بولی می‌باشند. به انواع ساده انواع اصلی نیز گفته می‌شود.

انواع ساده دارای مجموعه مشخصی از مقادیر هستند و محدوده خاصی از اعداد را در خود ذخیره می‌کنند.

انواع ساده و محدود

نوع	دامنه
sbyte	اعداد صحیح بین 128- تا 127
byte	اعداد صحیح بین 0 تا 255
short	اعداد صحیح بین 32768- تا 32767
ushort	اعداد صحیح بین 0 تا 65535
int	اعداد صحیح بین 2147483648- تا 2147483647
uint	اعداد صحیح بین 0 تا 4294967295
long	اعداد صحیح بین 9223372036854775808- تا 922337203685477807
ulong	اعداد صحیح بین 0 تا 18446744073709551615

مقادیر اعشاری

نوع	دامنه تقریبی دقت	دقت
float	$\pm 1.5E-45$ to $\pm 3.4E38$	7 رقم
double	$\pm 5.0E-324$ to $\pm 1.7E308$	15 - 16 رقم
decimal	$(-7.9 \times 10^{28}) / (10^0 \text{ to } 28)$ to $(7.9 \times 10^{28}) / (10^0 \text{ to } 28)$	28 - 29 رقم معنادار

داده های غیر عددی

نوع	مقادیر مجاز
char	کاراکترهای یونیکد
bool	مقدار true و false
string	مجموعه ای کاراکترها

تعريف متغير

Datatype identifier;

Datatype identifier1, identifier2,..., identifierN;

محدوده متغیر

محدوده یک متغیر مشخص می‌کند که متغیر در کجای کد قابل دسترسی است. هنگامیکه برنامه به پایان متد Main() می‌رسد متغیرها از محدوده خارج و بدون استفاده می‌شوند. محدوده متغیرها انواعی دارد که در جلسات بعدی با آنها آشنا می‌شوید.

مقداردهی متغیرها

می‌توان فوراً بعد از تعریف متغیرها مقادیری را به آنها اختصاص داد. این عمل را مقداردهی می‌نامند.
در زیر نحوه مقدار دهی متغیرها نشان داده شده است :

```
datatype identifier = value ;
```

```
datatype identifier1 = value1, identifier2 = value2 , ... identifierN = valueN ;
```


جانگهدار (Placeholders)

اگر به مثال اسلاید دقت نگاه کنید رشته قالب بندی شده دارای عدد صفری است که در داخل دو آکولاد محصور شده است. البته عدد داخل دو آکولاد می‌تواند از صفر تا n باشد. به این اعداد جانگهدار می‌گویند. این اعداد بوسیله مقدار آرگومان بعد جایگزین می‌شوند.

```
Console.WriteLine("The values are {0}, {1}, {2}, and {3}.", value1, value2, value3, value4);
```

ثابت ها

ثابت‌ها، انواعی از متغیرها هستند که مقدار آنها در طول برنامه تغییر نمی‌کند. ثابت‌ها حتماً باید مقدار دهی اولیه شوند و اگر مقدار دهی آنها فراموش شود در برنامه خطا به وجود می‌آید. بعد از این که به ثابت‌ها مقدار اولیه اختصاص داده شد هرگز در زمان اجرای برنامه نمی‌توان آن را تغییر داد.

```
const datatype identifier = initial_value ;
```


تبدیل ضمنی

تبدیل ضمنی متغیرها یک نوع تبدیل است که به طور خودکار توسط کامپایلر انجام می‌شود. یک متغیر از یک نوع داده می‌تواند به طور ضمنی به یک نوع دیگر تبدیل شود به شرطی که مقدار آن از مقدار داده‌ای که می‌خواهد به آن تبدیل شود کمتر باشد.

تبدیلاتی که کامپایلر به صورت ضمنی انجام می دهد.

نوع منبع	تبدیل امن به نوع متغیر
byte	short, ushort, int, uint, long, ulong, float, double, decimal
sbyte	short, int, long, float, double, decimal
short	int, long, float, double, decimal
ushort	int, uint, long, ulong, float, double, decimal
int	long, float, double, decimal
uint	long, ulong, float, double, decimal
long	float, double, decimal
ulong	float, double, decimal
float	double
char	ushort, int, uint, long, ulong, float, double, decimal

تبدیل صریح

تبدیل صریح، نوعی تبدیل است که، برنامه را مجبور می‌کند که یک نوع داده را به نوعی دیگر تبدیل کند، اگر این نوع تبدیل از طریق تبدیل ضمنی انجام نشود. در هنگام استفاده از این تبدیل باید دقت کرد چون در این نوع تبدیل ممکن است مقادیر اصلاح یا حذف شوند. ما می‌توانیم این عملیات را با استفاده از Cast انجام دهیم. Cast فقط نام دیگر تبدیل صریح است و دستور آن به صورت زیر است :

```
datatypeA variableA = value;  
datatypeB variableB = (datatypeB)variableA;
```

تبدیل با استفاده از کلاس Convert

کلاس Convert یک کلاس استاتیک است که می‌توان از آن برای تبدیل مقادیر از نوعی به نوع دیگر استفاده کرد. این کلاس به نوبه خود دارای متدهایی برای تبدیل انواع داده به یکدیگر می‌باشد.

تبدیل با استفاده از کلاس Convert

دستور	نتیجه	دستور	نتیجه
Convert.ToBoolean(val)	Val به bool تبدیل می شود	Convert.ToInt64(val)	Val به long تبدیل می شود
Convert.ToByte(val)	Val به byte تبدیل می شود	Convert.ToSByte(val)	Val به sbyte تبدیل می شود
Convert.ToChar(val)	Val به char تبدیل می شود	Convert.ToSingle(val)	Val به float تبدیل می شود
Convert.ToDecimal(val)	Val به decimal تبدیل می شود	Convert.ToString(val)	Val به string تبدیل می شود
Convert.ToDouble(val)	Val به double تبدیل می شود	Convert.ToUInt16(val)	Val به ushort تبدیل می شود
Convert.ToInt16(val)	Val به short تبدیل می شود	Convert.ToUInt32(val)	Val به uint تبدیل می شود
Convert.ToInt32(val)	Val به int تبدیل می شود	Convert.ToUInt64(val)	Val به ulong تبدیل می شود

پایان جلسه دوم

عبارات و عملگرها

عبارات و عملگرها

عملگر: نمادهایی هستند که اعمال خاص انجام می‌دهند.

عملوند: مقادیری که عملگرها بر روی آنها عملی انجام می‌دهند.

$$X+Y$$

سی شارپ از عملگرهای ریاضی برای انجام محاسبات استفاده می‌کند.

عملگر	مثال	نتیجه
+	<code>var1 = var2 + var3;</code>	Var1 برابر است با حاصل جمع var2 و var3
-	<code>var1 = var2 - var3;</code>	Var1 برابر است با حاصل تفریق var2 و var3
*	<code>var1 = var2 * var3;</code>	Var1 برابر است با حاصلضرب var2 در var3
/	<code>var1 = var2 / var3;</code>	Var1 برابر است با حاصل تقسیم var2 بر var3
%	<code>var1 = var2 % var3;</code>	Var1 برابر است با باقیمانده تقسیم var2 و var3
+	<code>var1 = +var2;</code>	Var1 برابر است با مقدار var2
-	<code>var1 = -var2;</code>	Var1 برابر است با مقدار var2 ضربدر -1

عملگرهای ریاضی

عملگر	مثال	نتیجه
++	<code>var1 = ++var2;</code>	مقدار var1 برابر است با var2 بعلاوه 1
++	<code>var1 = var2++;</code>	مقدار var1 برابر است با var2 و به متغیر var2 یک واحد اضافه می‌شود.
--	<code>var1 = -- var2;</code>	مقدار var1 برابر است با var2 منهای 1
--	<code>var1 = var2 - -;</code>	مقدار var1 برابر است با var2 و از متغیر var2 یک واحد کم می‌شود.

عملگرهای تخصیصی

نوع دیگر از عملگرهای سی شارپ عملگرهای جایگزینی نام دارند. این عملگرها مقدار متغیر سمت راست خود را در متغیر سمت چپ قرار می‌دهند.

عملگر	نتیجه	مثال
=	<code>var1 = var2;</code>	مقدار <code>var1</code> برابر است با مقدار <code>var2</code>
+=	<code>var1 += var2;</code>	مقدار <code>var1</code> برابر است با حاصل جمع <code>var1</code> و <code>var2</code>
-=	<code>var1 -= var2;</code>	<code>var1</code> برابر است با حاصل تفریق <code>var1</code> و <code>var2</code>
*=	<code>var1 *= var2;</code>	مقدار <code>var1</code> برابر است با حاصل ضرب <code>var1</code> در <code>var2</code>
/=	<code>var1 /= var2;</code>	مقدار <code>var1</code> برابر است با حاصل تقسیم <code>var1</code> بر <code>var2</code>
%=	<code>var1 %= var2;</code>	مقدار <code>var1</code> برابر است با باقیمانده تقسیم <code>var1</code> بر <code>var2</code>

عملگرهای مقایسه ای

▶ از عملگرهای مقایسه‌ای برای مقایسه مقادیر استفاده می‌شود. نتیجه این مقادیر یک مقدار بولی (منطقی) است. این عملگرها اگر نتیجه مقایسه دو مقدار درست باشد مقدار true و اگر نتیجه مقایسه اشتباه باشد مقدار false را نشان می‌دهند.

عملگرهای مقایسه ای

عملگر	مثال	نتیجه
==	<code>var1 = var2 == var3;</code>	var1 در صورتی true است که مقدار var2 با مقدار var3 برابر باشد در غیر این صورت false است.
!=	<code>var1 = var2 != var3;</code>	var1 در صورتی true است که مقدار var2 با مقدار var3 برابر نباشد در غیر این صورت false است.
<	<code>var1 = var2 < var3;</code>	var1 در صورتی true است که مقدار var2 کوچکتر از var3 مقدار باشد در غیر این صورت false است.
>	<code>var1 = var2 > var3;</code>	var1 در صورتی true است که مقدار var2 بزرگتر از مقدار var3 باشد در غیر این صورت false است.
<=	<code>var1 = var2 <= var3;</code>	var1 در صورتی true است که مقدار var2 کوچکتر یا مساوی مقدار var3 باشد در غیر این صورت false است.
>=	<code>var1 = var2 >= var3;</code>	var1 در صورتی true است که مقدار var2 بزرگتر یا مساوی var3 مقدار باشد در غیر این صورت false است.

عملگرهای منطقی

▶ عملگرهای منطقی بر روی عبارات منطقی عمل می کنند و نتیجه آنها نیز یک مقدار بولی است.

عملگر	نام	مثال
&&	منطقی AND	var1 = var2 && var3
	منطقی OR	var1 = var2 var3
!	منطقی NOT	var1 = !var1

عملگر منطقی AND(&&)

X&&Y	Y	X
true	true	true
false	false	true
false	true	false
false	false	false

عملگر منطقی (OR) II

X Y	Y	X
true	true	true
true	false	true
true	true	false
false	false	false

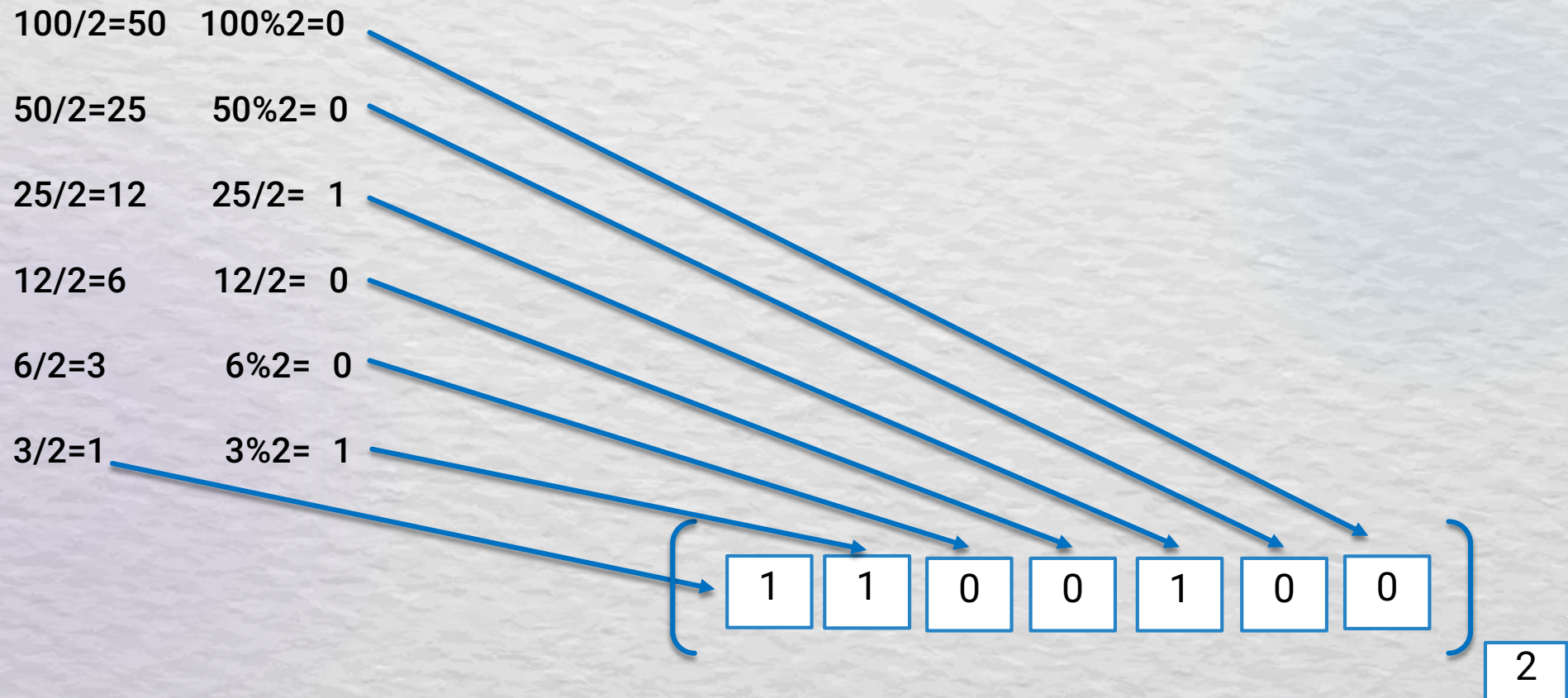
عملگر منطقی NOT(!)

!X	X
false	true
true	false

عملگرهای بیتی

عملگرهای بیتی به شما اجازه می‌دهند که شکل باینری انواع داده‌ها را دستکاری کنید.

تبدیل عدد 100 به مبنای 2



تبدیل عدد از مبنای 2 به مبنای 10

$(1100100)_2$

تبدیل به مبنای 10

$$(0 \cdot 2^0) + (0 \cdot 2^1) + (1 \cdot 2^2) + (0 \cdot 2^3) + (0 \cdot 2^4) + (1 \cdot 2^5) + (1 \cdot 2^6) = 0 + 0 + 4 + 0 + 0 + 32 + 64 = 100$$

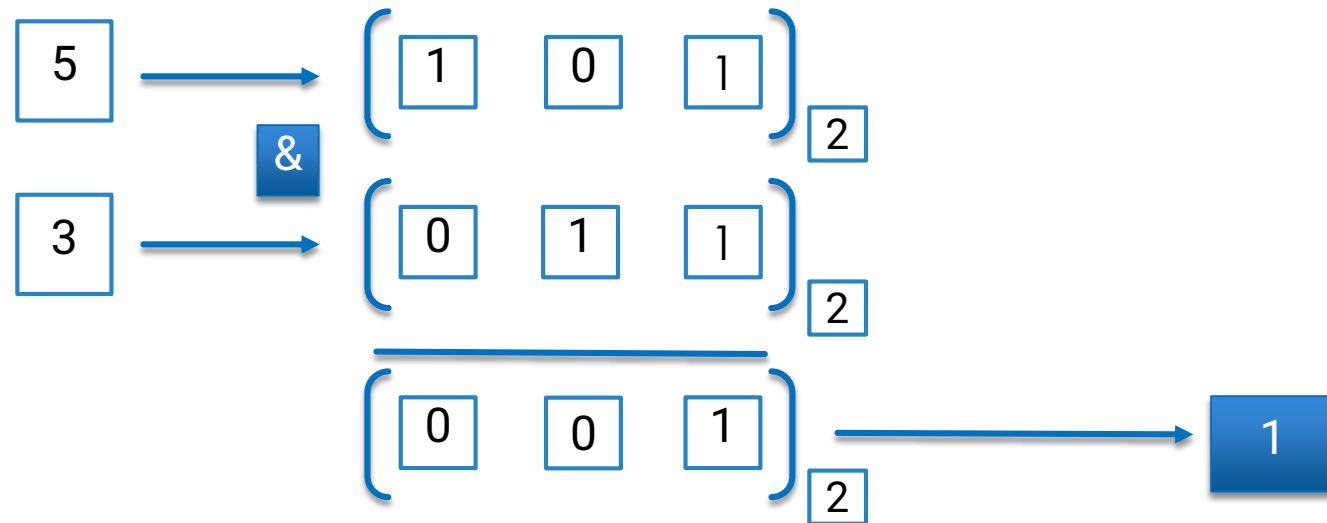
عملگرهای بیتی

عملگر	نام	مثال
&	بیتی AND	$x = y \& z;$
	بیتی OR	$x = y z;$
^	بیتی XOR	$x = y \wedge z;$
~	بیتی NOT	$x = \sim y;$
&=	بیتی AND Assignment	$x \&= y;$
=	بیتی OR Assignment	$x = y;$
^=	بیتی XOR Assignment	$x \wedge= y;$

عملگر بیتی AND (&)

X AND Y	Y	X
1	1	1
0	0	1
0	1	0
0	0	0

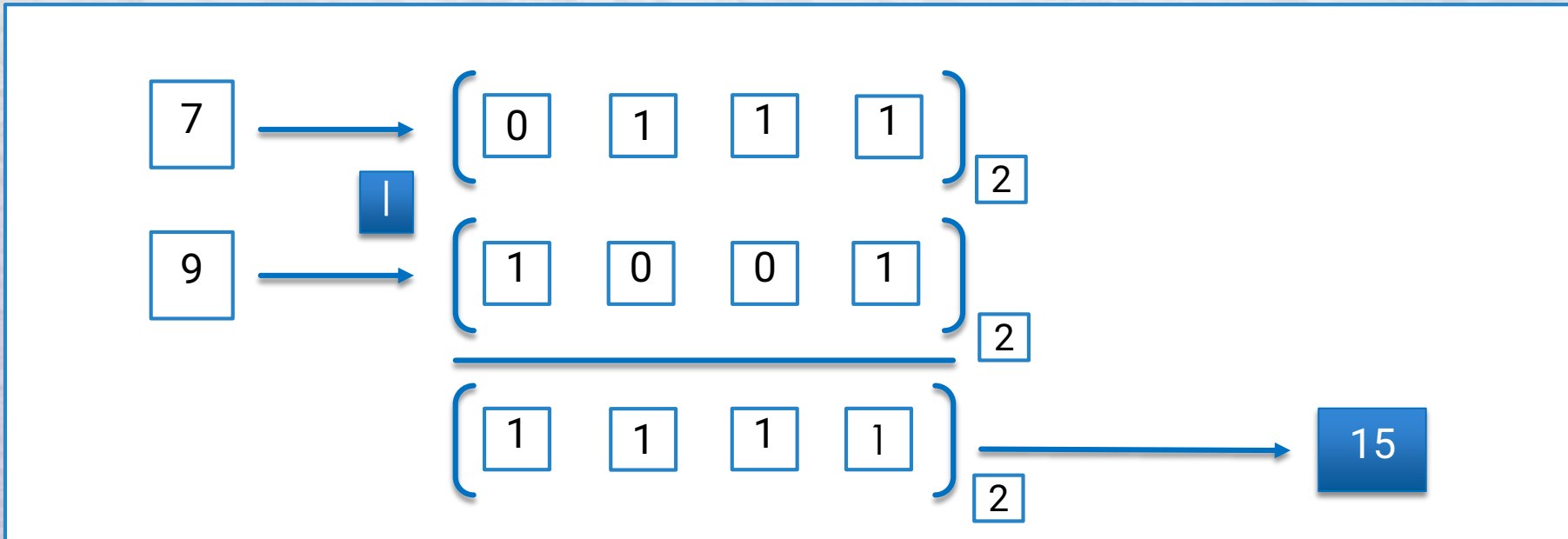
عملگر بیتی AND (&)



عملگر بیتی (OR)

X OR Y	Y	X
1	1	1
1	0	1
1	1	0
0	0	0

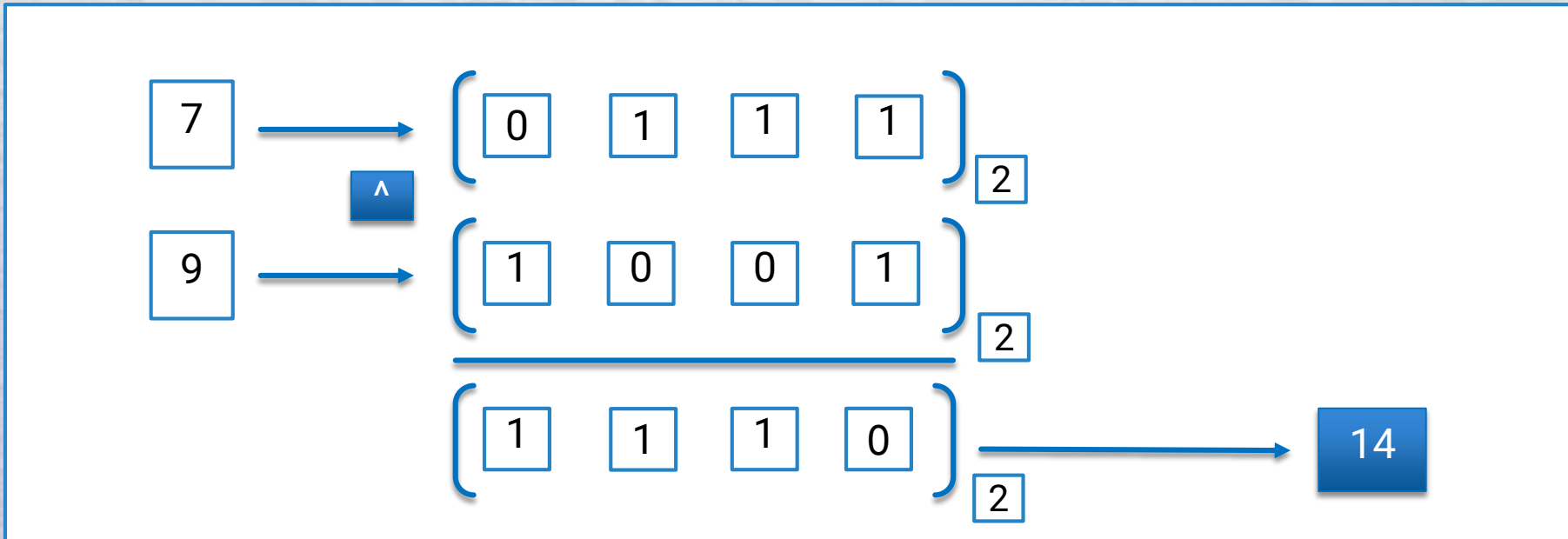
عملگر بیتی (OR)



عملگر بیٹی XOR(^)

X XOR Y	Y	X
0	1	1
1	0	1
1	1	0
0	0	0

عملگر بیتی XOR(^)



عملگر بیتی NOT(~)

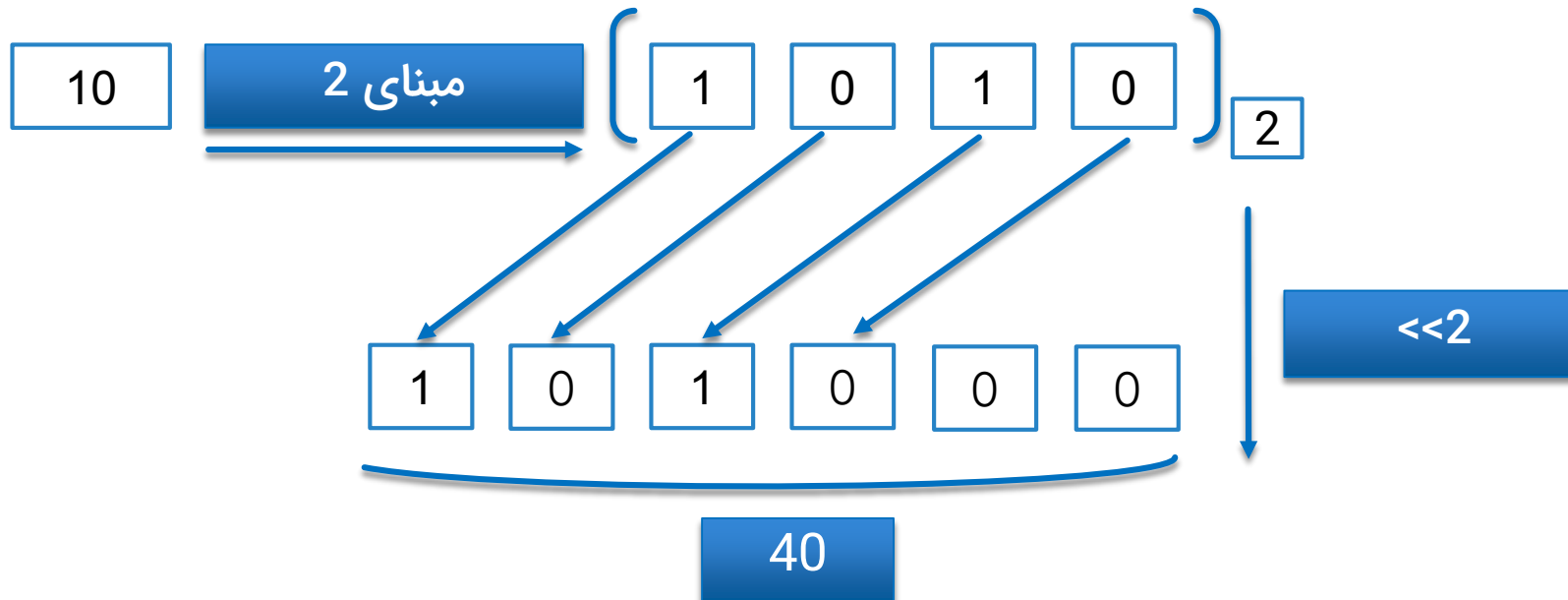
NOT X	X
0	1
1	0

عملگر بیتی تغییر مکان (shift)

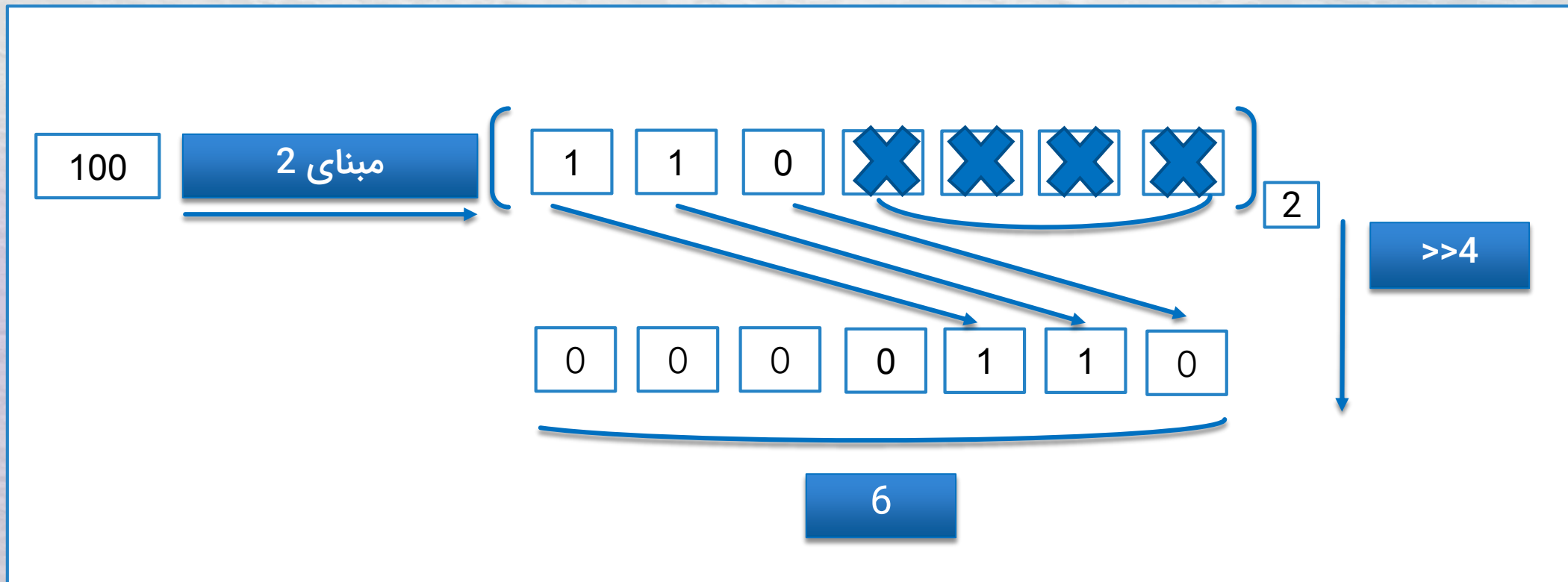
این نوع عملگرها به شما اجازه می‌دهند که بیت‌ها را به سمت چپ یا راست جا به جا کنید. دو نوع عملگر بیتی تغییر مکان وجود دارد که هر کدام دو عملوند قبول می‌کنند.

عملگر	نام	مثال
>>	تغییر مکان به سمت چپ	Binary $x = y \ll 2;$
<<	تغییر مکان به سمت راست	Binary $x = y \gg 2;$

عملگر تغییر مکان به سمت چپ



عملگر تغییر مکان به سمت راست



تقدم عملگرها

تقدم عملگرها مشخص می‌کند که در محاسباتی که بیش از دو عملوند دارند ابتدا کدام عملگر اثرش را اعمال کند. عملگرها در سی شارپ در محاسبات دارای حق تقدم هستند.

به عنوان مثال :

```
number = 1 + 2 * 3 / 1;
```


عملگر	تقدم
++, -- (used as prefixes)	بالا ترین
+, - (unary)	
*, /, %	
+, -	
<<, >>	
<, >, <=, >=	
==, !=	
&	
^	
&&	
=, *=, /=, %=, +=, -=	
++, - (used as suffixes)	پایین ترین



پایان جلسه سوم

ساختارهای تصمیم

دستور if

دستور if ساده ترین دستور شرطی هست که به برنامه می گوید اگر شرطی برقرار است کد معینی را انجام بده.

```
if (condition)
{
    code to execute.
}
```


دستور if...else

زمانی که شما بخواهید اگر شرط خاصی برقرار شد یک دستور و اگر برقرار نبود دستور دیگر اجرا شود باید از دستور if...else استفاده کنید.

```
if (condition)
{
    code to execute if condition is true ;
}
else
{
    code to execute if condition is false ;
}
```

عملگر شرطی

عملگر شرطی (: ?) در سی شارپ مانند دستور شرطی if...else عمل می کند.

<condition> ? <result is true> : <result is false>

دستور if چندگانہ

▶ اگر بخواهیم چند شرط را بررسی کنیم.

```
if (condition)
{
    code to execute ;
}
else if (condition)
{
    code to execute ;
}
else
{
    code to execute ;
}
```

```
if (condition)
{
    if (condition)
    {
        code to execute ;
    }
    else if (condition)
    {
        code to execute ;
    }
}

else
{
    if (condition)
    {
        code to execute ;
    }
}
```



عملگرهای منطقی

عملگرهای منطقی به شما کمک می کنند که چندین شرط را با هم ترکیب کنید.

```
switch (testVar)
{
    case compareVal1:
        code to execute if testVar== compareVal1;
        break;
    case compareVal2:
        code to execute if testVar== compareVal2;
        break;
    .
    .
    case compareValN:
        code to execute if testVar== compareValN;
        break;
}
```

پایان جلسه پنجم

ساختارهای تکرار

While

ابتدا یک شرط را مورد بررسی قرار می دهد و تا زمانی که شرط برقرار باشد. کدهای درون بلوک اجرا می شوند.

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


do...while

این حلقه بسیار شبیه حلقه while است با این تفاوت که در این حلقه ابتدا کد اجرا می شود و سپس شرط مورد بررسی قرار میگیرد.

```
do
{
    code to repeat ;
}
while ( condition );
```


for

حلقه for عملی شبیه حلقه while انجام می دهد و فقط دارای چند خصوصیت اضافی است.

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

حلقه های تو در تو

```
for ( init ; condition ; increment)
{
    for ( init ; condition ; increment)
    {
    }
}
```

خارج شدن از حلقه با استفاده از break

پایان جلسه ششم

آرایه ها

آرایه ها

آرایه مجموعه ای از عناصر همنوع است. هر آرایه دارای نامی است که مانند متغیرهای معمولی نامگذاری میشود. برای دسترسی به عناصر آرایه از متغیری بنام اندیس استفاده میشود. به همین دلیل به آرایه متغیر اندیس دار نیز گفته میشود.

تعريف آرایه

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



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

مقداردهی آرایه

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

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

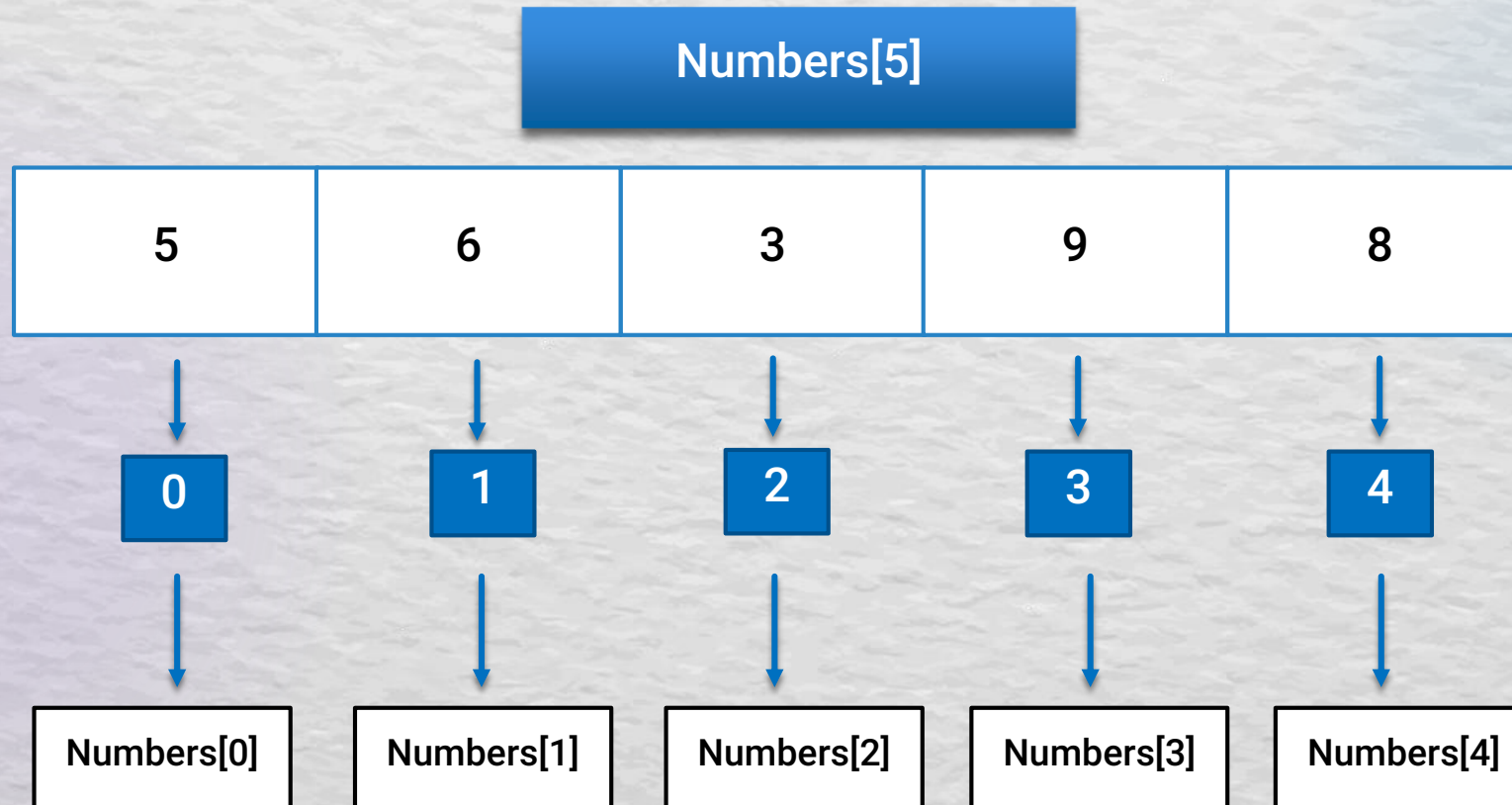
```
datatype[ ] arrayName = new datatype [ ]{val1,val2,...,valn} ;
```

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

```
datatype[ ] arrayName = {val1,val2,...,valn} ;
```

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

دسترسی به مقادیر آرایه



دستیابی به مقادیر آرایه با حلقه for

```
for ( init ; condition ; increment )  
{  
}
```

حلقه foreach برای پیمایش آرایه

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

تعریف آرایه های چندبعدی

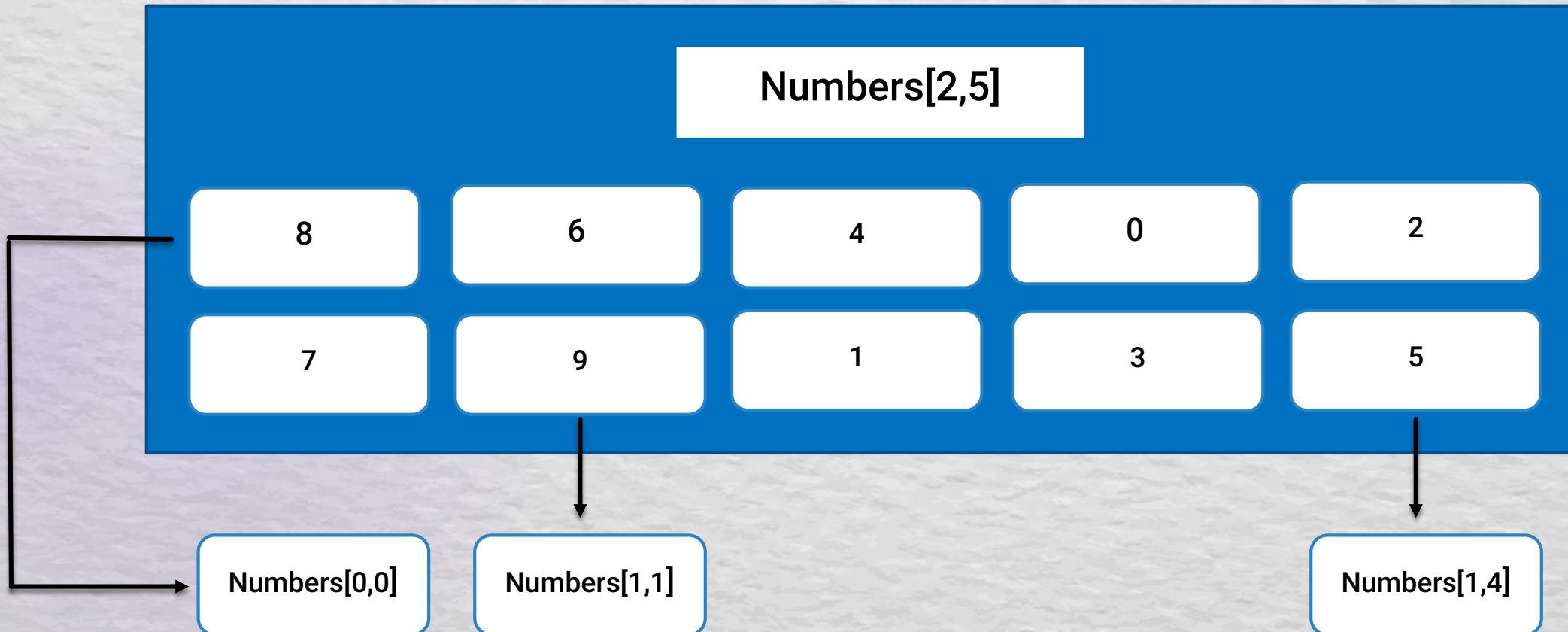
```
datatype[ , ] arrayName = new datatype [ lengthX , lengthY ];
```

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


آرایه های چندبعدی

Numbers[3,5]				
Numbers[0,0]	Numbers[0,1]	Numbers[0,2]	Numbers[0,3]	Numbers[0,4]
Numbers[1,0]	Numbers[1,1]	Numbers[1,2]	Numbers[1,3]	Numbers[1,4]
Numbers[2,0]	Numbers[2,1]	Numbers[2,2]	Numbers[2,3]	Numbers[2,4]

آرایه های چندبعدی



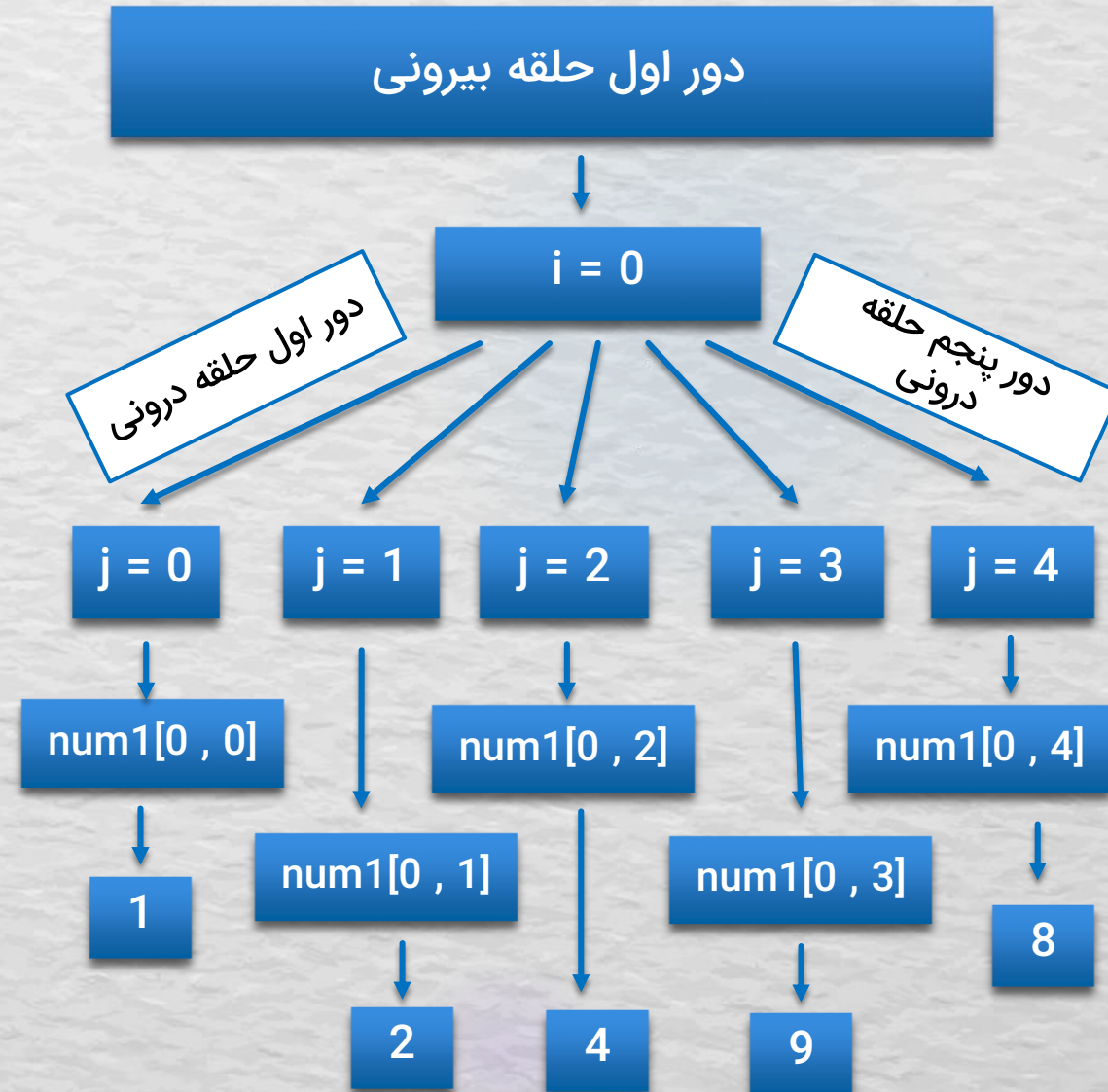
مقداردهی آرایه دو بعدی

```
datatype[ , ] arrayName = new datatype [lengthX, lengthY] { {r0 c0 , r0 c1 ,..., r0 cY} ,  
                                                                {r1 c0 , r1 c1 ,..., r1 cY} ,  
                                                                .  
                                                                .  
                                                                {rX c0 , rX c1 ,..., rX cY } } ;
```


دسترسی به مقادیر آرایه دو بعدی با حلقه for تو در تو

```
int[,] num1 = { { 1, 2, 4, 9, 8 }, { 0, 8, 6, 4, 3 }, { 1, 6, 5, 4, 2 } };

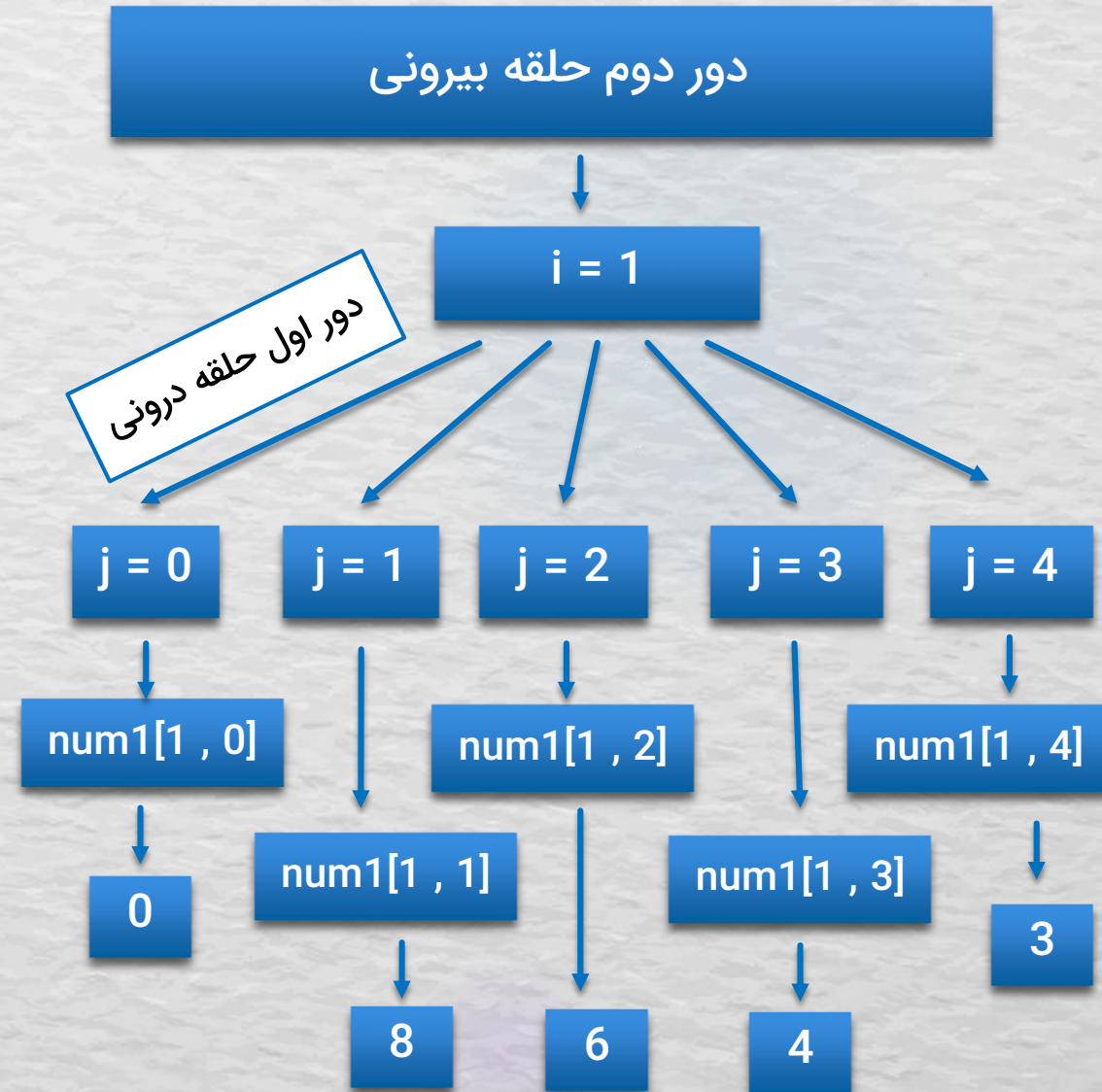
for (int i = 0; i < 3; i++)
{
    for (int j = 0; j < 5; j++)
    {
        Console.WriteLine(num1[i, j] + " ");
    }
}
```



دسترسی به مقادیر آرایه دو بعدی با حلقه for تو در تو

```
int[,] num1 = { { 1, 2, 4, 9, 8 }, { 0, 8, 6, 4, 3 }, { 1, 6, 5, 4, 2 } };

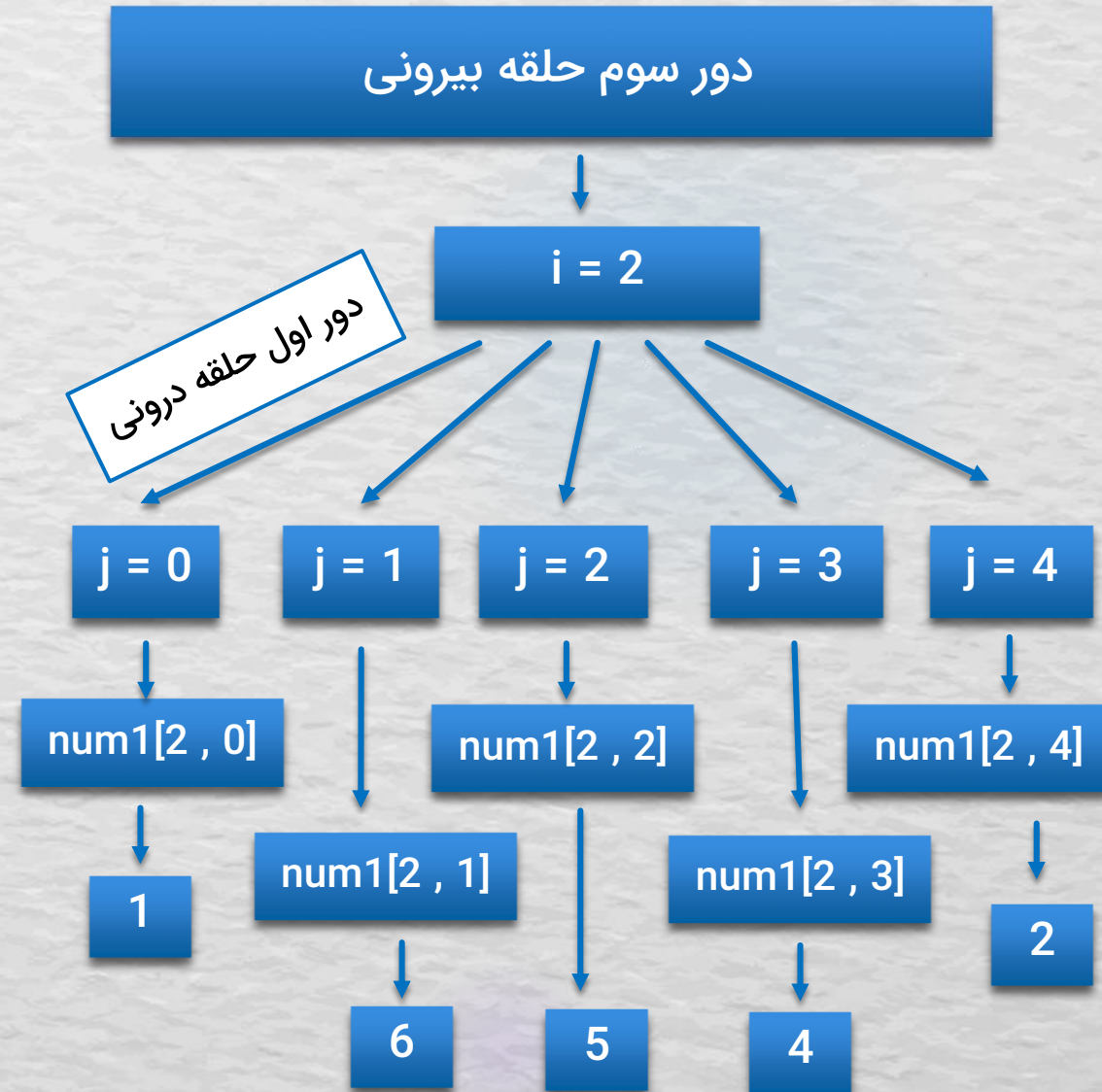
for (int i = 0; i < 3; i++)
{
    for (int j = 0; j < 5; j++)
    {
        Console.WriteLine(num1[i, j] + " ");
    }
}
```



دسترسی به مقادیر آرایه دو بعدی با حلقه for تو در تو

```
int[,] num1 = { { 1, 2, 4, 9, 8 }, { 0, 8, 6, 4, 3 }, { 1, 6, 5, 4, 2 } };

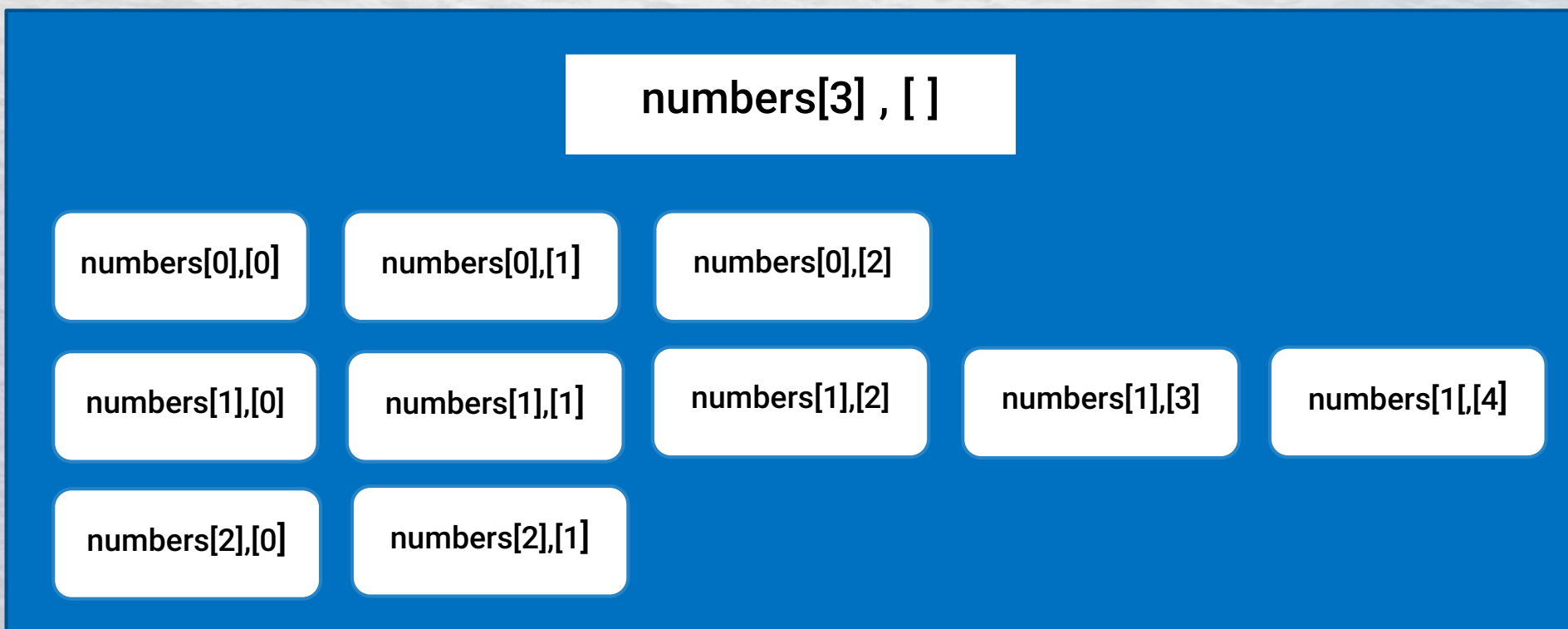
for (int i = 0; i < 3; i++)
{
    for (int j = 0; j < 5; j++)
    {
        Console.WriteLine(num1[i, j] + " ");
    }
}
```



آرایه های دندانه دار

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

آرایه های دندانه دار



numbers[3],[]

numbers[0],[0]

numbers[0],[1]

numbers[0],[2]

numbers[1],[0]

numbers[1],[1]

numbers[1],[2]

numbers[1],[3]

numbers[1],[4]

numbers[2],[0]

numbers[2],[1]

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

```
numbers[0]=new int [3];
```

```
numbers[1]=new int [5];
```

```
numbers[2]=new int [2];
```

تعريف آرایه مثال بالا

numbers[3],[]

1

2

3

5

4

3

2

1

11

22

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

مقداردهی آرایه مثال بالا

تمرین

با استفاده از حلقه for و foreach اعضای آرایه دندان‌دار را در خروجی چاپ کنید .

پایان جلسه هفتم

متدها (توابع)

متدها

متدها به شما اجازه می دهند که یک رفتار و یا وظیفه را تعریف کنید و مجموعه ای از کدها هستند که در هر جای برنامه می توان از آنها استفاده نمود.

```
returnType  MethodName  
{  
    Code to execute;  
}
```


مقدار بازگشتی متد

متدها می توانند مقدار بازگشتی از هر نوعی داشته باشند.

```
returnType MethodName  
{  
    return value;  
}
```


پارامترها و آرگومان ها

▶ پارامترها داده های خامی هستند که متد آنها را پردازش می کند. و سپس اطلاعاتی که به دنبال آن هستید در اختیار شما قرار می دهد.

▶ آرگومان ها مقادیری هستند که به پارامترها اختصاص داده می شوند.

پارامترها و آرگومان ها

```
static int CalculateSum (int number1 , int number2)
{
    Code to execute;
}
```



پارامترهای متد

```
public static void Main ()
{
    int num1=2,int num2=5;
    Console.WriteLine("Sum = {0}", CalculateSum (int num1 , int num2));
}
```



آرگومان های متد

نامیدن آرگومان ها

یکی دیگر از راه های ارسال آرگومان ها استفاده از نام آنهاست. استفاده از نام آرگومان ها خوانایی برنامه را بالا می برد.

```
MethodToCall (paramName1:value , paramName1:value , ... , paramNameN:value) ;
```


ارسال آرگومان ها به روش ارجاع

آرگومان ها را می توان به روش ارجاع ارسال کرد این بدان معناست که شما آدرس متغیر را ارسال می کنید نه مقدار آن را.

```
returnType  MethodName (ref datatype param1)
{
    Code to execute;
}
```

```
MethodName (ref argument) ;
```

پارامترهای out

پارامترهای out پارامترهایی هستند که متغیرهایی که مقداردهی اولیه نشدند را قبول می کنند.

مثال

```
void MyMethod (out int number)
{
    Code to execute;
}
```

ارسال آرایه به عنوان آرگومان

می توان آرایه ها را به عنوان آرگومان به متد ارسال کرد.

مثال

```
void MyMethod (int[ ] myArray)
{
    Code to execute;
}
```


کلمه کلیدی params

کلمه کلیدی params امکان ارسال تعداد دلخواه پارامترهای هم نوع و ذخیره آنها در یک آرایه ساده را فراهم می آورد.

```
void MyMethod (params int[ ] numbers)
{
    Code to execute;
}
```

```
public static void Main ()
{
    MyMethod (1,2,3) ;
    MyMethod (1,2,3,4) ;
}
```

محدوده متغیر

متغیرها در سی شارپ دارای محدوده هستند محدوده یک متغیر مشخص می کند که در کجای برنامه می توان از متغیر استفاده کرد و یا متغیر قابل دسترسی است .

پارامترهای اختیاری

پارامترهای اختیاری همانطور که اسمشان پیداست اختیاری هستند . و می توان به آنها آرگومان ارسال کرد یا نه .

```
static void PrintMessage (string message="welcome to c#")  
{  
    Console.WriteLine (message);  
}
```


سربارگذاری متدها

سربارگذاری متدها به شما اجازه می دهد که چندین متد با نام یکسان تعریف کنید که دارای امضا و تعداد پارامترهای مختلف هستند.

```
void MyMethod (int x , double y , string z)
```

امضای متد بالا

```
MyMethod (int , double , string)
```

متدهای بازگشتی

بازگشت فرایندی است که در آن متد مدام خود را فراخوانی می کند تا زمانی که به یک مقدار مورد نظر برسد.


```
static long Factorial (int number)
{
    if (number==1)
        return 1;
    else
        return number * Factorial(number-1);
}
```

Factorial (5)

return 5 * Factorial (4) = 120

return 4 * Factorial (3) = 24

return 3 * Factorial (2) = 6

return 2 * Factorial (1) = 2

1

نماینده ها (Delegates)

یک Delegate نوعی اشاره‌گر به توابع در سی شارپ است که می‌تواند ارجاعی را به یک یا چند تابع بخصوص داشته باشد. امضای یک Delegate باید با امضای متدی که به آن اشاره می‌کنید یکی باشد.

```
delegate returnType DelegateName (datatype param1 , datatype param2 , datatype param3) ;
```

ساختارها

ساختارهای یا struct انواعی از داده ها هستند که توسط کاربر تعریف می شوند و می توانند داری فیلد و متد باشند.

```
struct StructName  
{  
    member1 ;  
    member2 ;  
    member3 ;  
    ...  
    member4 ;  
}
```


پایان جلسه هشتم

برنامه نویسی شی گرا (OOP)

کلاس به شما اجازه می دهد یک نوع داده ای که توسط کاربر تعریف می شود و شامل فیلدها و خواص (properties) و متدها است را ایجاد کنید .

```
class ClassName
{
    field1 ;
    field2 ;
    ...
    fieldN ;

    method1;
    method2;
    ...
    methodN;
}
```

ایجاد نمونه از کلاس

```
ClassName  ObjectName = new ClassName ();
```


سازنده ها (Constructor)

سازنده ها متدهای خاصی هستند که وجود آنها برای ساخت اشیا لازم است . آنها به شما اجازه می دهند که مقادیری را به هر یک از مقادیر داده ای (فیلدها) اختصاص دهید. سازنده ها باید هم نام کلاس باشند.

```
ClassName ()  
{  
    code to execute;  
}
```

مخرب (Destructor)

مخرب ها نقطه مقابل سازنده ها هستند . مخرب ها متدهایی هستند که هنگام تخریب یک شی فراخوانی می شوند .

```
~ClassName ()  
{  
    code to execute;  
}
```


فیلدهای فقط خواندنی

از کلمه کلیدی readonly برای متغیرها استفاده میشود و اجازه تغییر مقادیر آن ها را نمی دهد.

```
readonly dataType Name = value ;
```


سطح دسترسی (Scope)

سطح دسترسی مشخص می کند که متدها و فیلدهای یک کلاس در کجای برنامه قابل دسترسی هستند.

سطح دسترسی public : این سطح دسترسی زمانی مورد استفاده قرار می گیرد که بخواهید به یک متد یا فیلد در خارج از کلاس و یا حتی پروژه دسترسی یابید .

سطح دسترسی private : زمانی که از این سطح دسترسی برای یک فیلد یا متد استفاده می شود دیگر آن متد یا فیلد در خارج از کلاس و پروژه قابل دسترسی نیستند.

خواص

خواص یا property استاندارد در سی شارپ برای دسترسی به اعضای داده ای (فیلدها) با سطح دسترسی private است. هر property دارای دو بخش است : بلوک set و بلوک get

```
get
{
    return name;
}

set
{
    name=value;
}
```

فضای نام

فضای نام راهی برای دسته بندی کدهای برنامه است . هر چیز در دات نت حداقل در یک فضای نام قرار دارد .

```
namespace MyNamespace  
{  
    ...  
}
```


ساختارها در برابر کلاس ها

ساختارها انواع مقداری هستند مانند : int ، double ، string . وقتی یک مقدار از ساختار را در در یک متغیر کپی می کنید ، در اصل خود مقدار را کپی کرده اید نه آدرس یا مرجع آن را . اما کلاس ها انواع مرجع هستند.

کتابخانه کلاس

کتابخانه کلاس مجموعه ای از کلاس ها و کدهاست که می تواند کامپایل شود و در نرم افزارهای دیگر برای استفاده مجدد به کار رود.

وراثت

وراثت به یک کلاس اجازه می دهد که خصوصیات یا متدهایی را از کلاس دیگر به ارث برد . وراثت مثل رابطه پدر و پسر می ماند ، به طوریکه فرزند خصوصیتی از قبیل رفتار و قیافه را از پدر خود به ارث برده باشد .

- ❖ کلاس پایه یا کلاس والد کلاسی است که بقیه کلاس ها از آن ارث می برند.
- ❖ کلاس مشتق یا کلاس فرزند کلاسی است که از کلاس پایه ارث بری می کند.

```
class Child : Parent
{
    ...
}
```


سطح دسترسی protect

سطح دسترسی protect اجازه می دهد که اعضای کلاس ، فقط در کلاس های مشتق شده از کلاس پایه قابل دسترسی باشند.

سطوح دسترسی

protected	private	public	قابل دسترسی در
true	true	true	داخل کلاس
false	false	true	خارج کلاس
true	false	true	کلاس مشتق

اعضای static

اگر بخواهیم عضو داده ای (فیلد) یا خاصیتی ایجاد کنیم که در همه نمونه های کلاس قابل دسترسی باشد از کلمه کلیدی static استفاده می کنیم .

متدهای مجازی

متدهای مجازی ، متدهایی از کلاس پایه هستند که می توان در کلاس مشتق آنها را `override` کرده و به صورت دلخواه پیاده سازی نمود.

رابط ها (interfaces)

رابط ها شبیه به کلاس ها هستند ، اما فقط شامل تعارفی برای متدها و خواص (property) می باشند. رابط ها را می توان به عنوان پلاگین های کلاس ها در نظر گرفت.

```
interface ISample
{
    ...
}
```


کلاس های انتزاعی (Abstract Class)

کلاس های انتزاعی (Abstract) کلاس هایی هستند که کلاس پایه سایر کلاس ها هستند. این نوع کلاس ها می توانند مانند کلاس های عادی دارای سازنده باشند. شما نمی توانید برای کلاس های انتزاعی نمونه ایجاد کنید چون هدف اصلی از به کار بردن کلاس های انتزاعی استفاده از آن ها به عنوان کلاس پایه برای کلاس های مشتق است.

```
public abstract class Base
{
    ...
}
```


کلاس های مهر و موم شده (sealed)

کلاس مهر و موم شده کلاسی است که دیگر کلاس ها نمی توانند از آن ارث بری کنند. چون قابلیت ارث بری ندارد .

ایجاد آرایه ای از کلاس ها

ساخت آرایه ای از کلاس ها تقریبا شبیه به ایجاد آرایه ای از انواع داده ای مانند `int` است .

پایان جلسه نهم

مجموعه ها

کلاس ArrayList

کلاس ArrayList به شما اجازه ی ذخیره مقادیر انواع مختلف داده و امکان حذف و اضافه کردن عناصر آرایه را در هر لحظه می دهد.

کلکسیون عمومی (Generic Collections)

برای ایجاد یک کلکسیون عمومی از کلاس `List<T>` مربوط به فضای نام `System.Collections.Generic` استفاده می شود.

انواع Enumerator و Enumerable

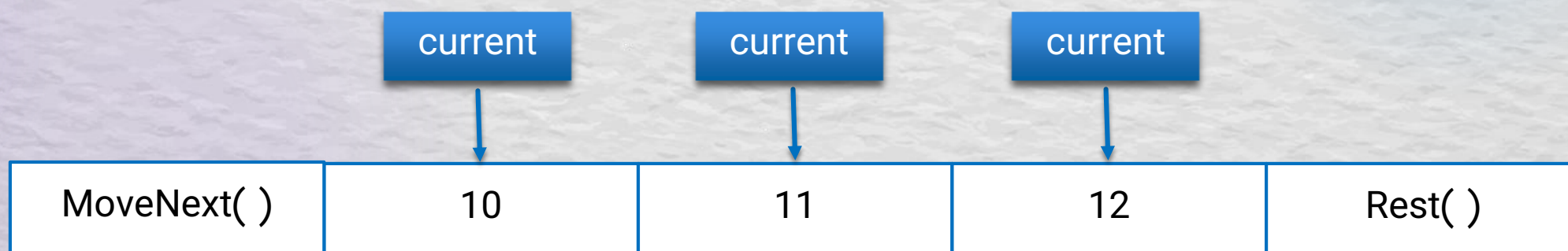
رابط های IEnumerable و IEnumerator

تمامی کلاس هایی که به نحوی شامل یک Collection هستند این دو رابط را پیاده سازی می کنند. وجود IEnumerable که توسط کلاس ها پیاده سازی می شود به کلاس این امکان را می دهد که به صورت ضمنی و توکار بشود شیء را پیمایش کرد.

رابط IEnumerator در سطح پایین تری از یک IEnumerable قرار دارد. با استفاده از این رابط می توان در هر جای بدنه متد اشیایی را برگشت بدهیم.

رابط Enumerator

یک شمارنده‌ی Enumerator رابط Enumerator را پیاده سازی می‌کند که دارای دو متد MoveNext() و Rest() و یک خاصیت به نام Current می باشد.



رابط IEnumerable

یک کلاس قابل شمارش (Enumerable) کلاسی هست که رابط IEnumerable را پیاده سازی می کند. رابط IEnumerable یک عضو دارد و آن عضو هم متد GetEnumerator() می باشد. و پارامتر برگشتی این متد از نوع همان رابط IEnumerator هست.

پیمایشگر Iterator

Iterator بلوک کدی هست که شامل همه مقادیری است که در یک حلقه foreach مورد استفاده قرار می گیرند .

پایان جلسہ دہم

مباحث پیشرفته برنامه نویسی C#

متدهای بی نام (Anonymous Methods)

متدهای بی نام متدهایی هستند که در واقع تعریف نمی شوند . بنابراین فقط برای یکبار مورد استفاده قرار می گیرند. این متدها هدفی برای delegate ها هستند.

```
delegate (parameters)
{
    // code for anonymous method.
}
```


عبارات لامبدا (Lambda expression)

عبارات لامبدا ساده شده دستور زبان متدهای بی نام هستند.

Expression-Bodied Members

Expression-Bodied Members یکی از ویژگی های C# 6.0 بوده که به شما اجازه استفاده از لامبدا برای کدنویسی راحت تر متدها و خاصیت های یک کلاس را می دهد.

متدهای توسعه یافته

همه متدها وابسته به کلاسی هستند که در آن تعریف می شوند. اما ویژگی توسعه متدها به شما اجازه می دهد متدی ایجاد کنید که علاوه بر کلاسی که در آن تعریف شده است ، به کلاس های دیگر وابسته باشد.

انواع بی نام

در سی شارپ می توان انواع بی نامی را تعریف کرد که یک روش عالی برای تعریف انواع موقتی جهت ذخیره انواع داده هاست .

عملگر ?? (Null Coalescing)

عملگر Null Coalescing یک عملگر باینری است که برای تشخیص مقدار دو عملوند به کار می رود. کاربرد اصلی این عملگر در قرار دادن یک مقدار nullable در یک مقدار non-nullable با یک دستور ساده است.

```
var result = operand1 ?? operand2
```

برنامه نویسی ناهمگام (Asynchronous)

منظور از برنامه نویسی ناهمگام همان برنامه نویسی به صورت موازی است . با استفاده از این شیوه برنامه نویسی کامپایلر می تواند چندین تابع یا متد را به صورت موازی و بدون مسدود کردن توابع دیگر اجرا کند.

❖ **Task**: Task ها برای مدیریت بهتر کارها و زمانبندی های خاص برای آن ها به کار گرفته می شوند.

❖ **Process**: وقتی کاربر برنامه ای را اجرا می کند مقداری از حافظه و منابع به آن برنامه تخصیص داده می شود. یکی از وظایف سیستم عامل تفکیک حافظه و منابع برای هر یک از برنامه های در حال اجراست که این جداسازی توسط Process انجام می شود.

❖ **Thread**: هر Process می تواند شامل چندین Thread باشد و هر Thread وظیفه انجام عملیات خاصی را بر عهده دارد.

پایان جلسه یازدهم