مروری اجمالی به دستور زبان سی شارپ (C# Syntax):

```
using System;


namespace HelloWorld
{
 class Program
 {
  static void Main(string[] args)
  {
   Console.WriteLine("Hello World!");
  }
 }
}
```

خروجی:

```
Hello World!
```

توضیح و تشریح قطعه کد بالا:

خط 1: using System به این معنی است که این فضای نام را به برنامه خود اضافه کرده‌ایم و می توانیم از کلاس های فضای نامSystem در برنامه خود استفاده کنیم.

خط 2: یک خط خالی است. C# فضای خالی را نادیده می گیرد. گاهی اوقات برای خوانایی بهتر کد خوب است که یک یا چند سطر را با زدن دکمه Enter خالی بگذارید.

خط 3: namespace HelloWorld ایجاد یک فضای نام به نام HelloWorld است. فضای نام برای سازماندهی کد استفاده می شود و یک ظرف نگهدارنده و محفظه برای کلاس‌ها و سایر فضاهای نام است.

خط 4: پرانتزهای مجعد {} شروع و پایان یک بلوک کد را نشان می دهد.

خط 5: کلاس محفظه‌ای برای داده‌ها و متدها است که قسمت اصلی کدنویسی در این بخش قرار می‌گیرد. هر خط کدی که در سی شارپ اجرا می شود باید داخل یک کلاس باشد. در مثال ما کلاس را Program نامگذاری کردیم.

اگر متوجه نشدید که using System، namespace و class چطور کار می کنند، نگران نباشید. فقط به آن‌ها به عنوان چیزی که تقریبا همیشه در برنامه‌های سی شارپ شما ظاهر می شود فکر کنید. در درس‌های بعدی بیشتر در مورد آنها خواهید آموخت.

خط 7: یکی دیگر از مواردی که همیشه در یک برنامه سی شارپ ظاهر می شود، متد Main است. هر کدی که در براکت های مجعد آن {} باشد اجرا خواهد شد. لازم نیست در حال حاضر کلمات کلیدی قبل و بعد ازMain را درک کنید. در طول این دوره ذره ذره با آنها آشنا خواهید شد.

خط 9: Console کلاسی از فضای نام System است که دارای متد ()WriteLine است که برای چاپ متن استفاده می شود. در مثال بالا "Hello World!" در خروجی کد چاپ می‌شود.

نکته: تمام موارد فضاهایی نام، کلاس‌ها ومتدها در سی شارپ با استفاده از قاعده PascalCase نامگذاری می‌شوند؛ در این قاعده حرف اول تمام کلمات بصورت بزرگ نوشته می‌شود.

نکته: اگر using System را در خط اول حذف کنید، بایستی آن را به روش زیر در ابتدای دسترسی به کلاس Console بنویسید:

```
System.Console.WriteLine()
```

نکته: هر دستور سی شارپ با یک semicolon یا همان نقطه ویرگول (;) خاتمه می یابد. بایستی توجه داشته باشید که خطوط کدی مانند تعریف فضای نام، کلاس و متد که دارای بلوک می‌باشند و بلوک آن‌ها با {} مشخص شده، یک دستور سی شارپ به حساب نمی‌آیند و در نتیجه در انتهای آن‌ها نقطه ویرگول (;) نمیگذاریم.

همانطور که در خط 9 کد بالا مشاهده کردید، برای خروجی مقادیر یا چاپ متن در سی شارپ، می توانید از متد ()WriteLine استفاده کنید؛ می توانید هر تعداد متد ()WriteLine را که می خواهید به کد خود اضافه کنید. توجه داشته باشید که برای هر بار استفاده از متد یک خط جدید در خروجی چاپ می کند:

```
using System;

namespace HelloWorld
{
 class Program
 {
  static void Main(string[] args)
  {
   Console.WriteLine("Hello World!");
   Console.WriteLine("I am Learning C#");
   Console.WriteLine("It is awesome!");
  }
 }
```

```
}
```

خروجی:

```
Hello World!
I am Learning C#
It is awesome!
```

کامنت‌گذاری در سی شارپ

از کامنت‌ها می توان برای توضیح کد سی شارپ و خوانایی بیشتر آن استفاده کرد. همچنین می توان از آن برای جلوگیری از اجرا در مرحه توسعه و تست کد استفاده کرد.

- ✓ کامنت‌های تک خطی با دو اسلش رو به جلو (//) شروع می شود.
- ✓ کامنت‌های چند خطی با /* شروع می شود و با */ پایان می یابد.

```
using System;

namespace HelloWorld
{
 class Program
 {
  static void Main(string[] args)
  {
   // این یک کامنت تک خطی است

   /* این یک کامنت چند خطی استاین یک کامنت چند خطی استاین یک کامنت چند خطی استاین یک کامنت چند خطی استاین یک کامنت چند خطی استاین یک کامنت چند خطی استاین یک کامنت چند خطی استاین یک کامنت چند خطی استاین یک کامنت چند خطی است */
   Console.WriteLine("Hello World!");
   Console.WriteLine("I am Learning C#");
   Console.WriteLine("It is awesome!");
  }
 }
}
```

خروجی:

```
Hello World!
I am Learning C#
It is awesome!
```

متغیرها در سی شارپ

متغیرها ظرف‌هایی برای ذخیره مقادیر داده‌ها هستند.

در سی شارپ انواع مختلفی از متغیرها (تعریف شده با کلمات کلیدی مختلف) وجود دارد، به عنوان مثال:

- ✓ int یا همان integer به معنای اعداد صحیح، اعداد کامل مانند 123 یا 23- را ذخیره می کند.

- ✓ double اعداد اعشاری مانند 9.5 را ذخیره می کند.
- ✓ char کاراکترهای تکی مانند ’a‘ یا ’B‘ را ذخیره می کند. مقادیر Char با تک کوتیشن (‘) احاطه شده‌اند.
- ✓ string به معنای رشته، متنی مانند "Hello World" را ذخیره می کند. مقادیر رشته با دابل کوتیشن (") احاطه شده‌اند.
- ✓ bool مقادیر را با دو حالت true و false ذخیره می کند.

ایجاد متغیرها

برای ایجاد یک متغیر، باید نوع آن را مشخص کرده و یک مقدار به آن اختصاص دهید:

```
type variableName = value;
نوع      نام متغیر      = مقدار;
```

مثلا:

```
String firstName = “Ali”;
```

به قسمت سمت چپ علامت مساوی در ایجاد متغیر، اعلان متغیر و به قسمت سمت راست مساوی مقداردهی گویند.

می‌توانید ابتدا در یک خط متغیر را اعلان کنید و سپس در خط‌های پایین‌تر عملیات مقداردهی را انجام دهید:

```
String firstName;

firstName = “Ali”;
```

- ✓ در واقع در قسمت اعلان متغیر 2 کار اساسی انجام می‌دهیم: 1- تعیین نوع متغیر، 2- اختصاص یک نام منحصر به فرد به متغیر که حکم ID دارد.
- ✓ در قسمت مقداردهی نیز متناسب با نوع متغیر، یک داده را درون متغیر ذخیره می‌کنیم.
- ✓ از علامت مساوی نیز برای تخصیص مقادیر به متغیر استفاده می شود.

برای ایجاد متغیری که باید یک متن را ذخیره کند، به مثال زیر توجه کنید.

یک متغیر به نام firstName و از نوع رشته یا همان string ایجاد کرده و مقدار "Ali" را به آن اختصاص می‌دهیم:

```
string name = "Ali";
```

برای ایجاد متغیری که باید یک عدد را ذخیره کند، به مثال زیر توجه کنید.

```
int myNum = 15;
```

همانطور که گفته شد، می توانید یک متغیر را بدون تخصیص مقدار اعلان کنید و بعداً آن را مقداردهی کنید:

```
int myNum;

myNum = 15;
```

توجه داشته باشید که اگر مقدار جدیدی را به یک متغیر موجود اختصاص دهید، مقدار قبلی را بازنویسی (Over-Write) می کند:

```
using System;


namespace MyApplication
{
  class Program
  {
    static void Main(string[] args)
    {
      int myNum = 15;
      Console.WriteLine("old value for myNum is: " + myNum);
      myNum = 20;
      Console.WriteLine("new value for myNum is: " +myNum);
    }
  }
}
```

خروجی:

```
old value for myNum is: 15
new value for myNum is: 20
```

مروری سریع بر نحوه ایجاد متغیرهای دیگر انواع داده در سی شارپ:

```
double myDoubleNum = 5.1234567890D;
```

```
float myNum = 5.75F;
char myLetter = 'D';
bool myBool = true;
long myNum = 15000000000L;
```

انواع داده های سی شارپ

همانطور که در مطالب قبلی در مورد متغیرها توضیح داده شد، یک متغیر در سی شارپ باید یک نوع داده مشخص داشته باشد، در واقع نوع داده حجم متغیر و نوع مقادیر متغیر را مشخص می کند. استفاده از نوع داده صحیح برای متغیر مربوطه مهم است. برای جلوگیری از خطا، برای صرفه جویی در زمان و حافظه، همچنین کد شما را قابل نگهداری و خواناتر می کند. انواع داده سی شارپ به شرح زیر هستند:

- ✓ int که مخفف کلمه integer و به معنای اعداد صحیح است. اعداد صحیح را از 2,147,483,648- تا 2,147,483,647 ذخیره می کند:

```
int myNum = 100000;
```

- ✓ long دیگر نوع داده برای نگهداری اعداد صحیح است که اعداد صحیح را از 9,223,372,036,854,775,808- تا 9,223,372,036,854,775,807 ذخیره می کند. این مورد زمانی استفاده می شود که int برای ذخیره مقدار کافی نباشد. توجه داشته باشید که باید مقدار را با "L" پایان دهید:

```
long myNum = 15000000000L;
```

- ✓ float اعداد اعشاری را ذخیره می کند. برای ذخیره اعداد تا 6 رقم اعشار کافی است.
- ✓ double اعداد اعشاری را ذخیره می کند. برای ذخیره اعداد تا 15 رقم اعشار کافی است. توجه داشته باشید که باید مقدار را با "F" برای float و "D" برای double پایان دهید:

```
float myNum1 = 5.75F;
double myNum2 = 19.912345678909D;
```

- ✓ bool مقادیر true یا false را ذخیره می کند:

```
bool isCSharpFun = true;
bool isFishTasty = false;
```

- ✓ char یک کاراکتر/حرف را که با تک کوتیشن (‘) احاطه شده است ذخیره می کند:

```
char myGrade = 'B';
```

- ✓ string دنباله‌ای از کاراکترها را که با دبل کوتیشن (“) احاطه شده‌اند ذخیره می کند:

```
string greeting = "Hello World";
```

ثابت ها (Constants)

اگر نمی خواهید دیگران (یا خودتان) مقادیر موجود درون یک متغیر را بازنویسی کنند، می توانید کلمه کلیدی const را جلوی نوع متغیر اضافه کنید. این کلمه کلیدی متغیر را به عنوان یک متغیر ثابت یا "constant" اعلان کرده که به معنای غیرقابل تغییر و فقط خواندنی (read-only) است:

```
using System;
```

```
namespace MyApplication
{
 class Program
 {
  static void Main(string[] args)
  {
   const int myNum = 15;
   myNum = 20;
   Console.WriteLine(myNum);
  }
 }
}
```

خروجی:

```
prog.cs(10,7): error CS0131: The left-hand side of an assignment must be a variable, a property or an indexer
Compilation failed: 1 error(s), 0 warnings
```

کلمه کلیدی const زمانی مفید است که می خواهید یک متغیر همیشه مقدار یکسانی را ذخیره کند تا دیگران (یا خودتان) کد شما را خراب نکنند. مثالی که اغلب به عنوان ثابت از آن یاد می شود، عدد PI (3.14) است.

توجه: نمی توانید یک متغیر ثابت را بدون مقداردهی اعلان کنید. اگر این کار را انجام دهید، خطایی رخ می دهد: یک متغیر const باید یک مقدار داشته باشد.

نمایش متغیرها

همانطور که در مطالب قبلی ارائه شد، متد WriteLine() از کلاس Console و از فضای نام System اغلب برای نمایش مقادیر متغیر در پنجره کنسول استفاده می شود. برای ترکیب متن و متغیر، از کاراکتر + استفاده کنید:

```
using System;

namespace MyApplication
{
 class Program
 {
```

```
    static void Main(string[] args)
    {
      string name = "John";
      Console.WriteLine("Hello " + name);
    }
  }
}
```

خروجی:

Hello John

همچنین می توانید از کاراکتر + برای اضافه کردن یک متغیر به متغیر دیگر استفاده کنید:

```
string firstName = "John ";

string lastName = "Doe";

string fullName = firstName + lastName;

Console.WriteLine(fullName);
```

بصورت زیر نیز قابل نوشتن است، اما کد بالا تمیزتر است زیرا عملیات اتصال دو رشته متن به یکدیگر را در یک خط جداگانه انجام داده و نتیجه را درون یک متغیر دیگر ریخته است اما در کد زیر این عملیات بصورت مستقیم درون آرگومان متد WriteLine() انجام شده و این نحوه کدزنی از نظم و خوانایی کمتری برخوردار است که هنگام زیاد شدن کدها و بزرگ شدن برنامه اذیت کننده خواهد بود:

```
string firstName = "John ";

string lastName = "Doe";

Console.WriteLine(firstName + lastName;);
```

برای مقادیر عددی، کاراکتر + به عنوان یک عملگر ریاضی کار می کند (توجه کنید که در اینجا از متغیرهای int (integer) استفاده می کنیم:

```
int x = 5;

int y = 6;

Console.WriteLine(x + y);
```

خروجی:

11

ایجاد چند متغیر در یک خط

برای ایجاد بیش از یک متغیر از یک نوع در یک خط، از یک لیست جدا شده با کاما (,) استفاده کنید:

```
int x = 5, y = 6, z = 50;
Console.WriteLine(x + y + z);
```

خروجی:

```
61
```

همچنین می توانید یک مقدار را به چندین متغیر در یک خط اختصاص دهید:

```
int x, y, z;
x = y = z = 50;
Console.WriteLine(x + y + z);
```

خروجی:

```
150
```

قوانین نامگذاری متغیرها در سی شارپ

همه متغیرهای C# باید با نام‌های منحصر به فرد مشخص شوند.

این اسامی منحصر به فرد را شناسه یا ID می‌نامند.

شناسه ها می توانند نام های کوتاه مانند x و y یا نام های توصیفی تر مانند (age، firstName، totalVolume) باشند.

توجه: برای ایجاد کد قابل فهم و قابل نگهداری توصیه می شود از نام های توصیفی استفاده کنید:

```
// نامگذاری زیر خوب و قابل فهمه
string firstName = "Ali";
```

```
// بصورت زیر هم میشه اما این قابل فهم نیست که علی چیه؟ اسم هست یا فامیلی؟ اسم شخص هست یا اسم پدر شخص؟ و بسیاری ابهامات دیگر در پروژه‌های بزرگ و واقعی
string f = "Ali";
```

قوانین کلی برای نامگذاری متغیرها عبارتند از:

- ✓ نام ها می توانند شامل حروف انگلیسی (A-z)، اعداد انگلیسی (0-9) و کاراکتر زیرخط (_) باشند.
- ✓ نام ها باید با یک حرف شروع شوند.
- ✓ نام ها باید با حروف کوچک شروع شوند و نمی توانند دارای فضای خالی باشند.
- ✓ نام ها به حروف بزرگ و کوچک حساس هستند "myVar" و "myvar" متغیرهای متفاوتی هستند.
- ✓ کلمات رزرو شده از جمله کلمات کلیدی C# مانند int یا double نمی توانند به عنوان نام استفاده شوند.

تبدیل نوع متغیر Casting در سی شارپ

در سی شارپ دو نوع تبدیل وجود دارد:

تبدیل ضمنی (به‌طور خودکار): char -> int -> long -> float -> double

```
int myInt = 9;
double myDouble = myInt;
```

تبدیل صریح (دستی): double -> float -> long -> int -> char

```
double myDouble = 9.78;
int myInt = (int) myDouble;
```

تبدیل نوع با استفاده از متدهای سی شارپ

همچنین با استفاده از متدهای داخلی سی شارپ، مانند Convert.ToBoolean، Convert.ToDouble، Convert.ToString، Convert.ToInt32 (int) و Convert.ToInt64 (long) می‌توان انواع داده‌ها را به طور صریح تبدیل کرد:

```
using System;
namespace MyApplication
{
 class Program
 {
  static void Main(string[] args)
  {
   int myInt = 10;
   double myDouble = 5.25;
   bool myBool = true;
   Console.WriteLine(Convert.ToString(myInt));
   Console.WriteLine(Convert.ToDouble(myInt));
   Console.WriteLine(Convert.ToInt32(myDouble));
```

```
    Console.WriteLine(Convert.ToString(myBool));
    }
  }
}
```

خروجی:

```
10
10
5
True
```

قبلاً آموخته اید که Console.WriteLine() برای خروجی (چاپ) مقادیر استفاده می شود. اکنون از Console.ReadLine() برای دریافت ورودی کاربر استفاده می کنیم.

```
string userName = Console.ReadLine();
```

این متد هر چیزی که تایپ کنیم به عنوان یک داده رشته‌ای (string) ذخیره می‌کند.

اپراتورها (Operators) در سی شارپ

عملگرها برای انجام عملیات بر روی متغیرها و مقادیر استفاده می شوند.

عملگرهای حسابی- عملگرهای حسابی برای انجام عملیات ریاضی رایج استفاده می شوند:

```
int x = 5;

int y = 3;

Console.WriteLine(x + y);// جمع
```

```
int x = 5;
int y = 3;
Console.WriteLine(x - y);// تفریق
```

```
int x = 5;

int y = 3;

Console.WriteLine(x * y);// ضرب
```

```
int x = 12;

int y = 3;

Console.WriteLine(x / y);// تقسیم
```

```
int x = 5;

int y = 2;

Console.WriteLine(x % y);// باقیمانده تقسیم
```

عملگرهای تخصیصی- عملگرهای انتساب برای تخصیص مقادیر به متغیرها استفاده می شوند:

```
int x = 5;// = اختصاص مقدار 5 به متغیر با عملگر

x += 3;//          x = x + 3
```

مورد بالا برای تفریق، ضرب، تقسیم و باقیمانده نیز قابل انجام است.

عملگرهای مقایسه- عملگرهای مقایسه برای مقایسه دو مقدار (یا متغیر) استفاده می شوند. این در برنامه نویسی مهم است، زیرا به ما در یافتن پاسخ و تصمیم گیری کمک می کند.

مقدار برگشتی یک مقایسه true یا false است. این مقادیر به عنوان مقادیر Boolean شناخته می شوند و در فصل Booleans و If..Else بیشتر با آنها آشنا خواهید شد.

```
int x = 5;

int y = 3;

Console.WriteLine(x == y);// مقایسه برابری و مساوی بودن

Console.WriteLine(x != y);// مقایسه نابرابر بودن

Console.WriteLine(x > y);// مقایسه بزرگتر بودن

Console.WriteLine(x < y);// مقایسه کوچکتر بودن

Console.WriteLine(x >= y);// مقایسه بزرگتر یا مساوی بودن

Console.WriteLine(x <= y);// مقایسه کوچکتر یا مساوی بودن
```

عملگرهای منطقی- همانند عملگرهای مقایسه، می توانید مقادیر True یا False را با عملگرهای منطقی نیز تست کنید. عملگرهای منطقی برای تعیین منطق بین متغیرها یا مقادیر استفاده می شوند:

```
int x = 5;

Console.WriteLine(x > 3 && x < 10);// عملگر (وَ) برای ارزیابی برقراری دو عبارت

Console.WriteLine(x > 3 || x < 4);// عملگر (یا) برای ارزیابی برقراری حداقل یکی از دو عبارت

Console.WriteLine(!(x > 3 && x < 10));// عملگر (نبودن) برای ارزیابی برقرار نبودن یک عبارت
```

در توضیحات بالا برقرار بودن به معنای درستی و true بودن و برقرار نبودن به معنای نادرستی و false بودن عبارت است.

رشته‌ها در سی شارپ

یک رشته در سی شارپ در واقع یک شی است که حاوی خواص و متدهایی است که می تواند عملیات خاصی را روی رشته ها انجام دهد. به عنوان مثال، طول یک رشته را می توان با ویژگی Length پیدا کرد:

```
using System;

namespace MyApplication

{

 class Program

 {

  static void Main(string[] args)

  {

   string txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

   Console.WriteLine("The length of the txt string is: " + txt.Length);

  }

 }

}
```

خروجی:

```
The length of the txt string is: 26
```

متدهای رشته ای زیادی وجود دارد، به عنوان مثال ToUpper() و ToLower()، که یک کپی از رشته تبدیل شده به حروف بزرگ یا کوچک را برمی گرداند.

نکته مهم: در شروع یادگیری برنامه نویسی بهتر است تمرکز فرد روی یادگیری اصول و مبانی برنامه نویسی باشد و به متدهای داخلی یک زبان برنامه نویسی بیش از حد توجه نکند. به طور قطع در صورتی که یک برنامه نویس متدهای داخلی زبان مورد تخصص خود را به خوبی به خاطر داشته باشد یک مزیت است اما ابتدا بایستی مبانی برنامه نویسی را عمیقا یاد گرفت، سپس در ادامه مسیر فرصت آشنایی و کار با متدهای داخلی زبان نیز فراهم خواهد شد.

الحاق رشته (String Concatenation)

عملگر + را می توان بین رشته ها برای ترکیب آنها استفاده کرد. به این حالت الحاق می گویند:

```
string firstName = "John ";
string lastName = "Doe";
string name = firstName + lastName;
Console.WriteLine(name);
```

خروجی:

John Doe

همچنین می توانید از متد string.Concat() برای به هم پیوستن دو رشته استفاده کنید:

```
string firstName = "John ";
string lastName = "Doe";
string name = string.Concat(firstName, lastName);
Console.WriteLine(name);
```

خروجی:

John Doe

اگر دو عدد را اضافه کنید، نتیجه یک عدد خواهد بود:

```
int x = 10;
int y = 20;
int z = x + y;
Console.WriteLine(z);
```

خروجی:

30

اگر دو عدد را به صورت رشته اضافه کنید، نتیجه یک الحاق رشته خواهد بود:

```
string x = "10";
string y = "20";
string z = x + y;
```

```
Console.WriteLine(z);
```

خروجی:

```
1020
```

یکی دیگر از گزینه های الحاق رشته، درون یابی رشته ای است که مقادیر متغیرها را به جای نگهدارنده های یک رشته جایگزین می کند. توجه داشته باشید که لازم نیست نگران فضاها باشید، مانند الحاق:

```
string firstName = "John";

string lastName = "Doe";

string name = $"My full name is: {firstName} {lastName}";

Console.WriteLine(name);
```

خروجی:

```
My full name is: John Doe
```

شما می توانید با مراجعه به شماره ایندکس (index) یک کاراکتر در یک رشته داخل [] به کاراکترهای یک رشته دسترسی داشته باشید.

```
string myString = "Hello";

Console.WriteLine(myString[0]);
```

خروجی:

```
H
```

توجه: ایندکس‌های رشته با 0 شروع می شوند: [0] اولین کاراکتر است. [1] کاراکتر دوم و ... است.

همچنین می توانید با استفاده از متد IndexOf() موقعیت ایندکس یک کاراکتر خاص را در یک رشته پیدا کنید:

```
string myString = "Hello";

Console.WriteLine(myString.IndexOf("e"));
```

خروجی:

```
1
```

روش مفید دیگر Substring() است که کاراکترها را از یک رشته استخراج می کند، این متد در آرگومان اول ایندکس کاراکتری را می‌گیرد که برش را از آن شروع می‌کند و در آرگومان دوم گام برش را مشخص می‌کنیم. در صورتی که آرگومان دوم را خالی بگذاریم، گام برش تا انتهای رشته خواهد بود:

```
string name = "John Doe";

string lastName = name.Substring(5, 3);

Console.WriteLine(lastName);
```

خروجی:

Doe

از آنجایی که رشته ها باید در داخل دابل کوتیشن نوشته شوند، سی شارپ این رشته را اشتباه متوجه می شود و یک خطا ایجاد می کند:

```
string txt = "We are the so-called "Vikings" from the north.";
```

راه حل برای جلوگیری از این مشکل، استفاده از کاراکتر بک اسلش (\ ) است .کاراکتر بک اسلش (\) کاراکترهای خاص را به کاراکترهای رشته ای تبدیل می کند:

کاراکتر فرار	نتیجه
\'	'
\"	"
\\	\
\n	New Line
\t	Tab
\b	Backspace

بولین‌ها در سی شارپ

اغلب، در برنامه نویسی، به یک نوع داده نیاز دارید که فقط می تواند یکی از دو مقدار را داشته باشد، مانند:

- YES / NO
- ON / OFF
- TRUE / FALSE

برای این کار، سی شارپ دارای یک نوع داده bool است که می تواند مقادیر true یا false را بگیرد.

یک عبارت بولی با مقایسه مقادیر/متغیرها، مقدار بولی را برمی‌گرداند True یا False:
این برای ایجاد منطق و یافتن پاسخ مفید است.
به عنوان مثال، می توانید از یک عملگر مقایسه، مانند عملگر بزرگتر از (>) استفاده کنید تا بفهمید که آیا یک عبارت (یا یک متغیر) درست است یا خیر:

```
int x = 10;
int y = 9;
Console.WriteLine(x > y);
```

خروجی:

True

یک رویکرد بهتر، این است که یک مثال را در یک عبارت if...else بپیچیم تا بتوانیم بسته به نتیجه اقدامات مختلفی را انجام دهیم:

```
using System;
namespace MyApplication
{
 class Program
 {
  static void Main(string[] args)
  {
   int myAge = 25;
   int votingAge = 18;

   if (myAge >= votingAge)
   {
```

```
      Console.WriteLine("Old enough to vote!");
    }
    else
    {
      Console.WriteLine("Not old enough to vote.");
    }
  }
 }
}
```

خروجی:

Old enough to vote!

شرط‌ها در سی شارپ

سی شارپ دارای عبارات شرطی زیر است:

اگر شرط مشخص شده درست است، از if برای تعیین یک بلوک کد برای اجرا استفاده کنید.

اگر همان شرط نادرست باشد، از else برای تعیین یک بلوک کد برای اجرا استفاده کنید.

اگر شرط اول نادرست است، از else if برای تعیین یک شرط جدید برای ارزیابی استفاده کنید.

```
if (شرط)
{
 کدی که در صورت برقراری شرط باید اجرا بشه //
}
```

توجه داشته باشید که if با حروف کوچک است. حروف بزرگ If یا IF خطا ایجاد می کنند.

در مثال زیر، دو مقدار را آزمایش می کنیم تا بفهمیم 20 بزرگتر از 18 است یا خیر. اگر شرط True است، متنی را چاپ کنید:

```
if (20 > 18)
{
  Console.WriteLine("20 is greater than 18");
}
```

خروجی:

20 is greater than 18

یا بصورت زیر:

```
int x = 20;
int y = 18;
if (x > y)
{
  Console.WriteLine("x is greater than y");
}
```

خروجی:

x is greater than y

از عبارت else استفاده کنید تا یک بلوک از کد را مشخص کنید تا در صورت False بودن شرط اجرا شود:

```
int time = 20;
if (time < 18)
```

```
{
  Console.WriteLine("روز بخیر");
}
else
{
  Console.WriteLine("عصر بخیر");
}
```

خروجی:

عصر بخیر

اگر شرط اول False باشد از دستور else if برای تعیین یک شرط جدید استفاده کنید:

```
int time = 15;
if (time < 10)
{
  Console.WriteLine("Good morning.");
}
else if (time < 20)
{
  Console.WriteLine("Good day.");
}
else
{
  Console.WriteLine("Good evening.");
}
```

خروجی:

Good day.

در مثال بالا، ساعت (15) بزرگتر از 10 است، بنابراین شرط اول False است. شرط بعدی در عبارت else if اما True است زیرا 15 از 20 کمتر است، بنابراین کد درون بلوک آن اجرا شده و روی صفحه نمایش "Good day." چاپ می شود.

با این حال، اگر ساعت 22 بود، برنامه ما (Good evening.) را چاپ می کرد و اگر ساعت 9 بود برنامه (Good morning.) را چاپ می‌کرد.

حلقه‌ها در سی شارپ

حلقه‌ها می‌توانند یک بلوک کد را تا زمانی که یک شرط تعیین شده برقرار (true) است، اجرا کنند.

حلقه‌ها در برنامه نویسی بسیار مفید هستند زیرا وقتی که به تکرار یک قطعه کد در دفعات متعدد نیاز داریم، با انجام این کار در زمان صرفه جویی می‌کنند، خطاهای کد زنی برای انجام کد تکراری در دفعات مکرر را کاهش داده و کد را خواناتر می کنند.

سی شارپ دارای سه نوع حلقه است:

- ✓ حلقه while
- ✓ حلقه for
- ✓ حلقه foreach

حلقه while یک بلوک کد را تا زمانی که یک شرط مشخص True باشد، تکرار می‌کند:

```
using System;
namespace MyApplication
{
 class Program
 {
  static void Main(string[] args)
  {
   int i = 0;
   while (i < 5)
   {
    Console.WriteLine(i);
    i++;
   }
  }
 }
}
```

خروجی:

```
0
1
2
```

3
4

توجه: فراموش نکنید که متغیر مورد استفاده در شرط (متغیر i) را افزایش دهید، در غیر این صورت حلقه هرگز تمام نمی شود! (آخرین خط کد بدنه بلوک ;++i، که معادل i = i + 1 است.)

حلقه do/while نوعی از حلقه while است. این حلقه قبل از بررسی اینکه آیا شرط درست است یا خیر، یک بار بلوک کد را اجرا می کند، سپس تا زمانی که شرط درست باشد، حلقه را تکرار می کند:

```
int i = 0;
do
{
  Console.WriteLine(i);
  i++;
}
while (i < 5);
```

خروجی:

0
1
2
3
4

وقتی دقیقاً می دانید چند بار می خواهید از طریق یک بلوک کد حلقه بزنید، به جای حلقه while از حلقه for استفاده کنید، شاید در ظاهر این جمله اندکی شما را دچار ابهام کند، زیرا در ادامه خواهید دید که شرط اتمام حلقه for دقیقا مشابه حلقه while است و در هر دو انتهای تکرار حلقه مشخص است. معنی دقیقتر این جمله که در مستندات سی شارپ به آن اشاره شده است از دیدگاه شخصی بنده به شرح زیر است:

درون بدنه بلوک هر دو حلقه for و حلقه while می‌توان متغیر شمارنده حلقه را کم و زیاد کرد، مانند عملی که در حلقه while کردیم (i++)، در ادامه در حلقه for در مثال زیر خواهید دید که این عمل درون بدنه دستور حلقه for انجام می‌شود؛ در حالی که می‌توان این کار را مانند حلقه while درون بدنه بلوک حلقه for نیز انجام داد. اما به دو دلیل این کار خوب نیست: 1- در تکرارهای بالا و کدهای پیچیده‌تر که متغیر شمارنده درون بلوک حلقه for چندین بار کم و زیاد شده است شاهد باگ‌هایی بوده‌ایم که تعجب‌اور هستند. 2- خوانایی کد را کمتر می‌کند.

بنابراین بهتر است این قاعده را رعایت کنیم که در حلقه هایی که قرار است شمارشگر حلقه را بیش از یکبار، مورد تغییر قرار دهیم، بهتر است از حلقه while استفاده کرد. در غیر این صورت جهت خوانایی بیشر کد از حلقه for استفاده می‌کنیم:

```
using System;
namespace MyApplication
```

```
{
 class Program
 {
  static void Main(string[] args)
  {
   for (int i = 0; i < 5; i++)
   {
    Console.WriteLine(i);
   }
  }
 }
}
```

خروجی:

```
0
1
2
3
4
```

توضیح مثال بالا:

دستور اول، در بخش تعریف حلقه for یک متغیر را قبل از شروع حلقه تنظیم می کند.(int i = 0)

دستور دوم، شرط اجرای حلقه را تعریف می کند که i باید کمتر از 5 باشد. اگر شرط درست باشد، حلقه دوباره شروع می شود، اگر نادرست باشد، حلقه به پایان می رسد (i < 5).

دستور سوم، هر بار که بلوک کد در حلقه اجرا می شود مقدار i را یک واحد افزایش می دهد (i++).

حلقه های تو در تو

همچنین امکان قرار دادن یک حلقه در داخل یک حلقه دیگر وجود دارد. به این حلقه تودرتو می گویند.

"حلقه داخلی" یک بار برای هر تکرار "حلقه بیرونی" به صورت کامل اجرا می شود:

```
// حلقه بیرونی با 2 تکرار

for (int i = 1; i <= 2; i ++)

{

  Console.WriteLine("Outer: " + i);


  // حلقه داخلی با 3 تکرار که ضرب در 2 تکرار حلقه بیرونی شده و در مجموع 6 تکرار از آن خواهیم داشت.

  for (int j = 1; j <= 3; j++)

  {

    Console.WriteLine("  Inner: " + j);

  }

}
```

خروجی:

```
Outer: 1
  Inner: 1
  Inner: 2
  Inner: 3
Outer: 2
  Inner: 1
  Inner: 2
  Inner: 3
```