**Basic C# DataTypes:**

| **Name** | **System Class Name** | **Description** | **Bitsize** | **Example** |
| --- | --- | --- | --- | --- |
| int | Int32 | An integer value between -2,147,483,648 and 2,147,483,647 | 32-bits | 40,000 |
| long | Int64 | An integer between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807 | 64-bits | 3,000,000,000 |
| short | Int16 | An integer between -32,768 and 32,767 | 16-bits | 342 |
| float | Single | A floating point number between -3.402823e38 and 3.402823e38 | 32-bits | 3.14159 |
| double | Double | A floating point number between -1.79769313486232e308 and 1.79769313486232e308. Allows more precision than a float. | 64-bits | 3.1415926535897932 |
| char | Char | One unicode character | 16-bits | 'g' |
| bool | Boolean | A true or false value | 8-bits | true |
| string | String | An indexable sequence of characters | variable | "Hello World!" |
| object | Object | The base type that all classes inherit from. A generic way to treat every memory space. | variable | Pretty much everything! |

**Examples:**

**Explicit Casting:**

double DoubleValue = 3.14159265358;

int IntegerValue = (int)DoubleValue;

// IntegerValue == 3

**Explicit Casting:**

double floatNum = 7.76;

int num = floatNum;

//This will cause a compiler type-mismatch error!!

int num = (int)floatNum;

//Floating point to integer conversion truncates all numbers after the decimal

//As such num now evaluated to 7

int num = (int)"24";

// This, however, will not work as strings can not be type cast to int

**Explicit Casting:**

//Converting an integer to a string

int num = 7;

string numAsString = num.ToString(); //Evaluates as "7"

//Converting that string value to a double

double stringAsDouble = numAsString.ToDouble(); //Evaluates as 7.0

**Conditionals:**

//Declare a variable called rings that is of the Int Type

int rings = 5;

if (rings >= 5)

{

Console.WriteLine("You are welcome to join the party");

}

else

{

Console.WriteLine("Go win some more rings");

}

**Below is with an added extra condition, an *else if* statement.**

int rings = 5;

if (rings >= 5)

{

Console.WriteLine("You are welcome to join the party");

}

else if (rings > 2)

{

Console.WriteLine("Decent...but {0} rings aren't enough", rings);

}

else

{

Console.WriteLine("Go win some more rings");

}

**We can use Logical Operators in our Conditionals as well. Let's say we want to change the criteria for entering our NBA Legends party. Let's say you have to have at least 5 rings AND have the name Kobe to enter the party:**

int rings = 5;

//By placing the keyword const in front of a variable declaration

//you establish it as constant and immutable

const string name = "Kobe";

if (rings >= 5 && name == "Kobe")

{

Console.WriteLine("Welcome to the party {0}, congratulations on your {1} rings", name, rings);

}

**We can change our criteria and say that you have to have at least 5 rings or have at least 3 All-Star appearances.**

int rings = 5;

int numOfAllStarAppearances = 17;

if (rings >= 5 || numOfAllStarAppearances > 3)

{

Console.WriteLine("Welcome you are truly a legend");

}

**Or we can just let in everyone who is not crazy.**

bool crazy = true;

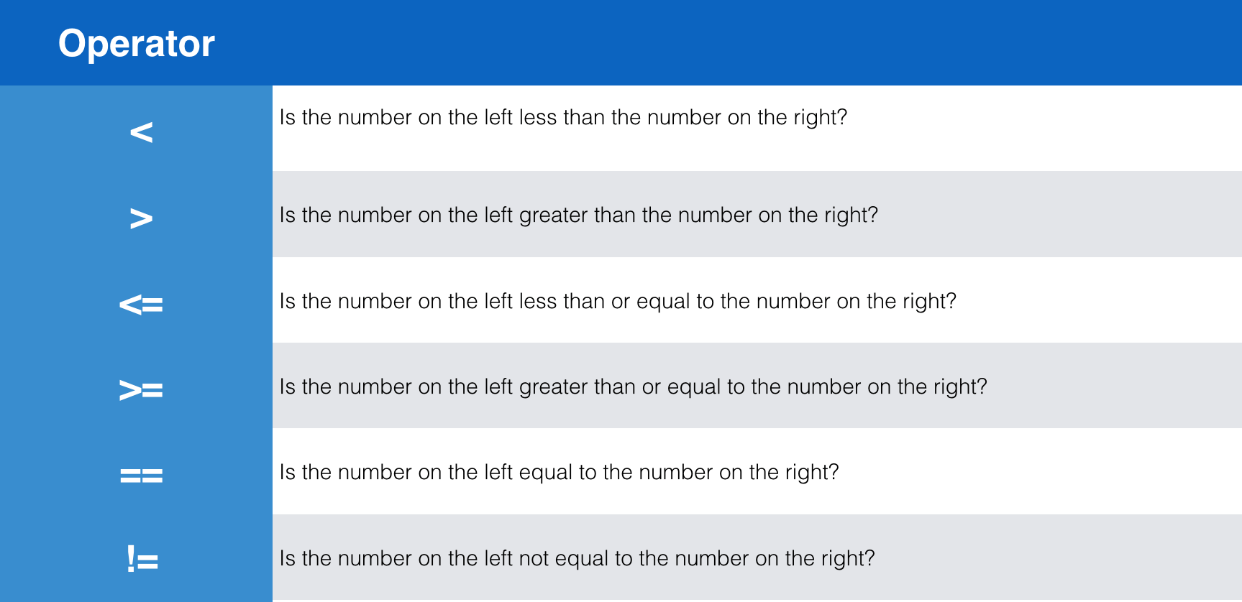
if (!crazy)

{

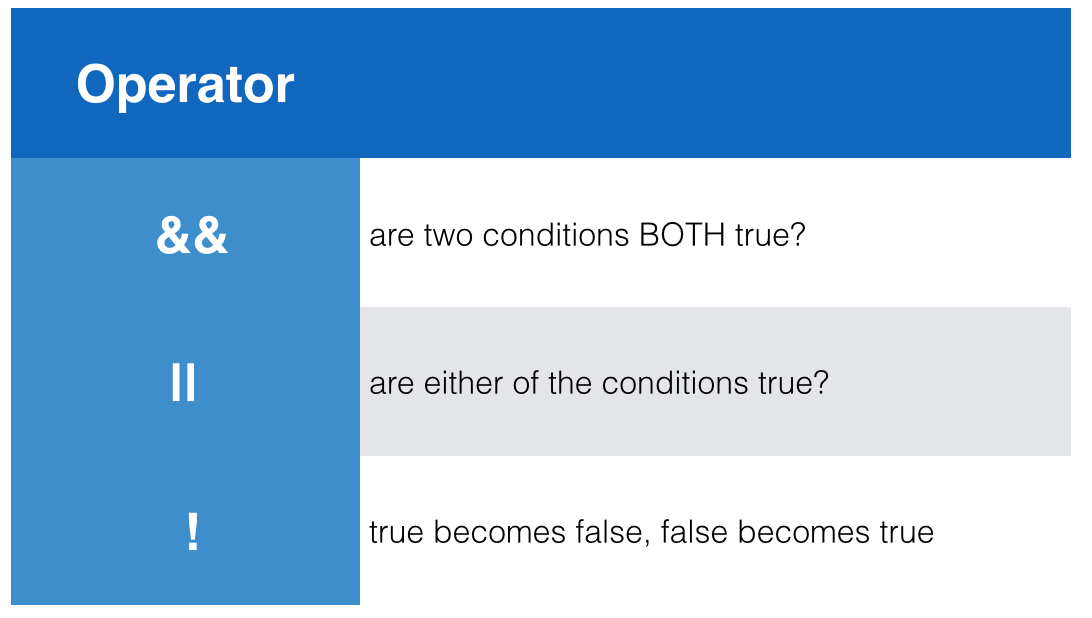
Console.WriteLine("Let's party!");

}

**Comparison Operators**



**Logical Operators**



**Loops**

**For Loops Examples:**

**A for loop performs a set of code in a specific range, sequence, or collection. For now, we will focus on using the "C styled" for loop with a range. Go ahead and run the following code below, in a new project, and see what the output is in the Console when you run.**

// loop from 1 to 5 including 5

for (int i = 1; i <= 5; i++)

{

Console.WriteLine(i);

}

// loop from 1 to 5 excluding 5

for (int i = 1; i < 5; i++)

{

Console.WriteLine(i);

}

**You can just as easily use variables to create a range as well!**

int start = 0;

int end = 5;

// loop from start to end including end

for (int i = start; i <= end; i++)

{

Console.WriteLine(i);

}

// loop from start to end excluding end

for (int i = start; i < end; i++)

{

Console.WriteLine(i);

**.Next Methods**

|  |  |
| --- | --- |
| **Method** | **Description** |
| .Next() | A 32-bit signed integer that is greater than or equal to 0 and less than maxValue |
| .Next(Int32) | Similar to .Next() but the provided parameter represents the maxValue |
| .Next(Int32, Int32) | Similar to .Next() but the provided parameters represent the minValue and maxValue for the range of number to randomize between |
| .NextDouble() | Returns a random floating-point number that is greater than or equal to 0.0, and less than 1.0. |