Get User Input

```
Console.WriteLine("Enter your name:");
string name = Console.ReadLine();
Console.WriteLine("Welcome, " + name);
```

A string is a sequence of characters enclosed in a a pair of single or double quotation mark.

- In C# we can read a text line from the console using **Console.ReadLine()** and we can convert the text to a integer number using **int.Parse**(text).
- The Console's **ReadLine** method waits for the user to type a string of characters at the keyboard and press the Enter key.

Another ReadLine() Example : Adding Integers

```
int num1, num2;
num1 = int.Parse(Console.ReadLine());
num2 = int.Parse(Console.ReadLine());

int sum = num1 + num2;
Console.WriteLine(sum);
```

integers are positive or negative whole numbers that don't use a decimal place.

To read an integer value, first use ReadLine() function to obtain the data as string, then convert it to an integer using the int.Parse() function.

Modifying the above program

```
Console.Write("Enter the first number: ");
string num1 = Console.ReadLine();
Console.Write("Enter the second number: ");
string num2 = Console.ReadLine();
int num1_as_int = int.Parse(num1);
int num2_as_int = int.Parse(num2);
int sum = num1_as_int + num2_as_int;
Console.WriteLine("The sum of two numbers is: " + sum);
```

*****

Naming Your Variables

- The name can contain letters, digits, and the underscore character (_).
- The first character of the name must be a letter. The underscore is also a **legal** first character.
- C# is case sensitive; thus, the names **count** and **Count** refer to two different variables.

Variable Name	*Legality*
`Percent`	Legal
`y2x5__w7h3`	Legal
`yearly_cost`	Legal
`_2010_tax`	Legal, but not advised
`checking#account`	Illegal; contains the illegal character #
`double`	Illegal; is a keyword
`9byte`	Illegal; first character is a digit

Declaring a Variable

- A variable declaration tells the compiler the name of the variable and the type of information that the variable will be used to store. If your program attempts to use a variable that hasn't been declared, the compiler generates an error message.

Format	Example
typename varname;	int my_number; int count, number, start;

Assigning Values to Your Variables

Format	Example
varname = value;	my_variable = 5;

- You can assign a value to a variable any time after it has been declared. You can even do this at the same time you declare a variable:

int my_variable = 5;

Exercise:

Fill in the missing parts to get user input, stored in the variable **userName**:

```
Console.WriteLine("Enter username:");
        userName = Console.      ;
Console.WriteLine("Username is: " + userName);
```

C# Data Types

```
int myNum = 5;               // Integer (whole number)
double myDoubleNum = 5.99D;  // Floating point number
char myLetter = 'D';         // Character
bool myBool = true;          // Boolean
string myText = "Hello";     // String
```

Data Type	Size	Description
int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits
char	2 bytes	Stores a single character/letter, surrounded by single quotes
bool	1 bit	Stores true or false values
string	2 bytes per character	Stores a sequence of characters, surrounded by double quotes

Exercise:

Add the correct data type for the following variables:

```
        myNum = 9;
        myDoubleNum = 8.99;
        myLetter = 'A';
        myBoolean = false;
        myText = "Hello World";
```

C# Type Casting

Casting, is the process of converting the value of a single data type (such as an integer) into another data type. This conversion is done either automatically or manually.

```
int myInt = 9;
double myDouble = myInt;       // Automatic casting: int to double

Console.WriteLine(myInt);      // Outputs 9
Console.WriteLine(myDouble);   // Outputs 9

double myDouble = 9.78;
int myInt = (int) myDouble;    // Manual casting: double to int

Console.WriteLine(myDouble);   // Outputs 9.78
Console.WriteLine(myInt);      // Outputs 9
```

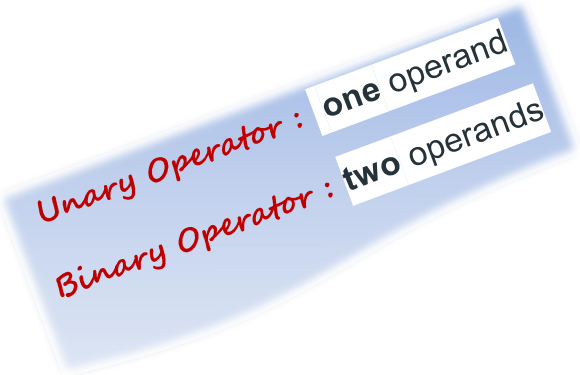
It is also possible to convert data types explicitly by using built-in methods

```
int myInt = 10;
double myDouble = 5.25;
bool myBool = true;

Console.WriteLine(Convert.ToString(myInt));    // convert int to string
Console.WriteLine(Convert.ToDouble(myInt));    // convert int to double
Console.WriteLine(Convert.ToInt32(myDouble));  // convert double to int
Console.WriteLine(Convert.ToString(myBool));   // convert bool to string
```

Manipulating Variable Values with Operators

- Operators can be broken into a number of categories:
  - The basic assignment operator
  - Mathematical/arithmetic operators
  - Relational operators
  - The Logical Operators



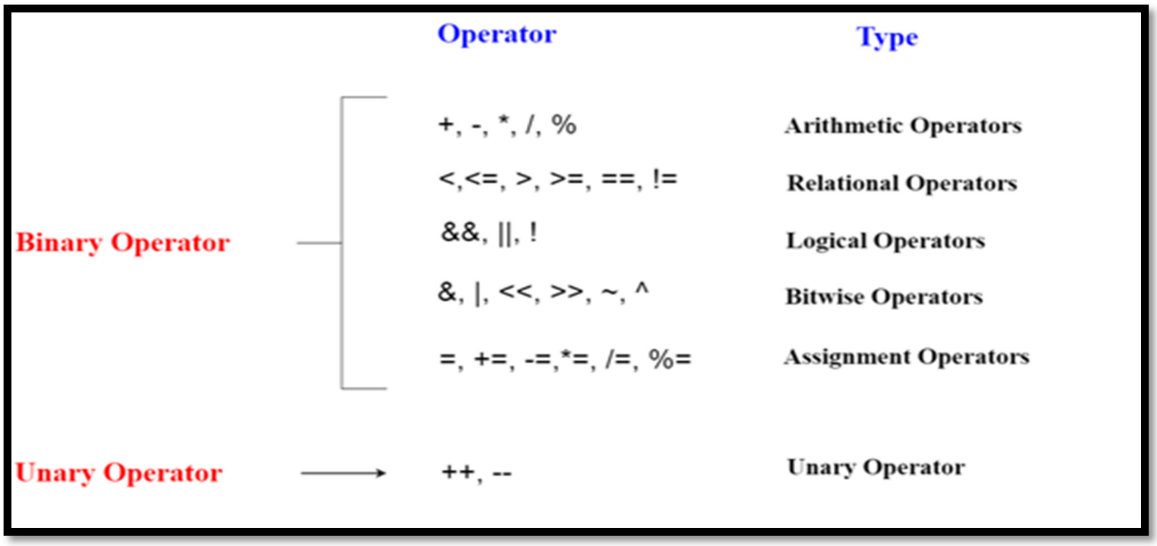
Assume variable A holds 10 and variable B holds 20, then

1. Arithmetic Operators		
Operator	**Description**	**Example**
+	Adds two operands	A + B = 30
-	Subtracts second operand from the first	A - B = -10
*	Multiplies both operands	A * B = 200
/	Divides numerator by de-numerator	B / A = 2
%	Modulus Operator and remainder of after an integer division	B % A = 0
++	Increment operator increases integer value by one	A++ = 11
--	Decrement operator decreases integer value by one	A-- = 9

2. Relational Operators		
Operator	**Description**	**Example**
==	Checks if the values of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

Assume variable A holds Boolean value true and variable B holds Boolean value false, then

3. Logical Operators		
Operator	**Description**	**Example**
&&	Called Logical AND operator. If both the operands are non zero then condition becomes true.	(A && B) is false.
||	Called Logical OR Operator. If any of the two operands is non zero then condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is true



******

Examples

- **Printing Text and Numbers**

```
string firstName = Console.ReadLine();
string lastName = Console.ReadLine();
int age = int.Parse(Console.ReadLine());
string town = Console.ReadLine();
Console.WriteLine("You are {0} {1}, a {2}-years old person from {3}.",
    firstName, lastName, age, town);
```

- **A simple example of C# program that converts from foots to meters**

```
Console.Write("Foots = ");
double foots = double.Parse(Console.ReadLine());
double meters = foots * 0.3048;
Console.Write("Meters = ");
Console.WriteLine(meters);
```

- **Calculate the square area by the given length of the side:**

```
Console.Write("a = ");
int a = int.Parse(Console.ReadLine());
int area = a * a;
Console.Write("Square area = ");
Console.WriteLine(area);
```

******

Exercises

- Write a program that reads a person's name and prints Hello, <name>!, where <name> is the name entered earlier.
- Write a program that reads a number from the console and converts the number from inches to centimeters. (one inch = 2.54 centimeters).
- Write a program that reads three numbers from the console b1, b2 and h and calculates the area of a trapezoid. The formula for trapezoid area is (b1 + b2) * h / 2.
- Write a program that reads from the console a number r and calculates and prints the area and perimeter of a circle with radius r.

  For the calculations you may use the following formulas:

  - **Area = Math.PI * r * r.**
  - **Perimeter = 2 * Math.PI * r.**