C Sharp/Math

Lesson Goals

This lesson will focus on basic math functionality in C# as well as user input via the console. At the end of this lesson you should be able to:

- Understand how to use basic math operations.
- Explain the order in which math operations are completed.
- Use the console to read in user input.
- Understand how to access the more powerful functionality of the **math** namespace.

Basic Math

Until now we've used hardcoded values to output information to the screen. Now we'll use math on those values to force the computer to perform some mathematical grunt workA computer uses the same laws for math as we do. The C# language follows the rule of **PEDMAS** (or **BEDMAS** depending on your high school teacher). PEDMAS stands for **P**arenthesis, **E**xponents, **D**ivision, **M**ultiplication, **A**ddition and **S**ubtraction. The operations on the left are executed before the operations to the right.

```
Example: 3 + (4 + 5) * 2
First step (Parenthesis): 3 + 9 * 2
Second step (Multiplication): 3 + 18
Last step (Addition): 21
```

Following the law of PEDMAS ensures that every computer (and human) will get a consistent answer. With that in mind, it is now possible to unleash mathematical operations on the computer

Type	Syntax	Example
Multiplication	var1 * var2	x = 2 * 3; // x = 6
Division	var1 / var2	x = 12 / 4; // x = 3
Addition	var1 + var2	x = 2 + 1; // x = 3
Subtraction	var1 - var2	x = 3 - 6; // x = -3
Parenthesis	(...)	x = (2 + 4) / 3; // x = 6 / 3 = 2
Modulus (Remainder)	var1 % var2	x = 3 % 2; // x = 1

Code Example 1

```
using System;

namespace MathExampleOne
{
    class Program
    {
        static void Main(string[] args)
        {
            double pi = 3.14159;
            double radius = 2.5;
```

```
            double area = pi * radius * radius;
            Console.WriteLine("The area of a circle with radius " +
radius + " is " + area);
            Console.WriteLine("Hit any key to end...");
            Console.ReadKey();
        }
    }
}
```

Output

```
The area of a circle with radius 2.5 is 19.6349375
Hit any key to end...
```

Remarks

```
double pi = 3.14159;
double radius = 2.5;
```

Here we've assigned a value for the constant PiPi is required for finding the area of a circle. We've also assigned a constant value for the radius of the circle.

```
double area = pi * radius * radius;
```

Here is where the computer gets to work number crunching. The area of a circle is given as A = pi * r * r. The result of the calculation will be stored in a variable named **area**.

```
Console.WriteLine("The area of a circle with radius " + radius + " is "
+ area);
```

We now output the result of the calculations to the consoleAs a bonus, we've also output the radius on the circle.

```
Console.WriteLine("Hit any key to end...");
Console.ReadKey();
```

These lines of code pause the execution of the program. Without these lines the program would terminate very quickly, and we wouldn't be able to view the program outputConsole.ReadyKey() waits until the user presses a key

User Input

A programmer can request the user to input a value in the console. The easiest method to read from the console is to read the console input until the user hits the **enter** button. A line must be stored in a C# type called a **string**. A string contains a list of **char**s. This string must then be converted to the **type** that was requested by the program. The best way to learn is by example and by doing, so here are some examples of reading a string and then converting that string to another data type.

```
string var = Console.ReadLine();                    // stores the
users input into a string
double radius = double.Parse(var);                  // converts the
string into a double using the Parse method
```

```
double radius = double.Parse(Console.ReadLine());   // same as the
previous method
```

```
int value = int.Parse(Console.ReadLine());          // converts to an
integer instead of a double
```

```
float value = float.Parse(Console.ReadLine());      // converts to a
float
```

Each C# data type has a method to parse a string, and convert that string into the corresponding data type. The problem however is that the user might make a mistake an enter a string that **cannot** be parsed. An example would be if the program requested an integer value, but the user entered a floating point value. The program will then throw an error that the **Input string was not in a correct format**. For now we will assume that the user will always enter the correct values, and the error that was generated will be handled in a later lesson on try, catch() and finally.

Code Example 2

```
using System;

namespace VariablesExampleTwo
{
    class Program
    {
        static void Main(string[] args)
        {
            double pi = 3.14159;
            Console.Write("Please enter a value for the radius: ");
            double radius = double.Parse(Console.ReadLine());
            double area = pi * radius * radius;
            Console.WriteLine("The area of a circle with radius " +
radius + " is " + area);
            Console.WriteLine("Hit any key to end...");
            Console.ReadKey();
        }
    }
}
```

Output

```
Please enter a value for the radius: 2.5
The area of a circle with radius 2.5 is 19.6349375
Hit any key to end...
```

Remarks

```
Console.Write("Please enter a value for the radius: ");
double radius = double.Parse(Console.ReadLine());
```

First we prompt the user for a value. This ensures that the user realizes that the program is waiting for input. Then we read a **double** into the variable named **radius**. This radius is used to compute the area of the circle. Note that this program will not deal with any errors that could be generated by the Parse() method.

Math Namespace

C# has many more math functions stored in the Math namespace. Exploration of the entire Math namespace is beyond the scope of this lesson, but many online references are available. We're going to modify the circle program to use the Math namespace where appropriate.

Code Example 3

```
using System;

namespace VariablesExampleThree
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Please enter a value for the radius: ");
            double radius = double.Parse(Console.ReadLine());

            double area = Math.PI * Math.Pow(radius, 2);
            Console.WriteLine("The area of a circle with radius " +
radius + " is " + area);
            Console.WriteLine("Hit any key to end...");
            Console.ReadKey();
        }
    }
}
```

Output

```
Please enter a value for the radius: 2.5
The area of a circle with radius 2.5 is 19.6349540849362
Hit any key to end...
```

Remarks

```
double area = Math.PI * Math.Pow(radius, 2);
```

Notice that there is no longer a line to store a value of pi. The Math namespace already contains a definition of PI (which is much more accurate than the previous version). The Math namespace also has a method for computing powers. radius^2 is represented quite a bit easier with the Math namespace. It would also become very unwieldy to write something like radius^20 with the method that we were using in the previous examples.

Summary

- Math operations are computed in the order of PEDMAS.
- User input from the console is first read as a string, and then converted to the correct data type.
- The conversion method (Parse) can throw an error if the string is incorrectly formatted.
- The Math namespace contains many methods that make complex mathematical operations much easier to perform.

Practice Exercises

- Use user input and the Math namespace to generate the sin and cos of a user assigned value.
- Use user input and the Math namespace to find the maximum value of two user assigned values.

Where To Go Next

Topics in C#		
Beginners	**Intermediate**	**Advanced**
▪ Lesson 1: Introduction to .NET and Introduction to Mono ▪ Lesson 2: C# Compilers ▪ Lesson 3: Introduction to C# ▪ Lesson 4: First Program ▪ Lesson 5: Variables and Casting ▪ Lesson 6: Math and User Input ▪ Lesson 7: Conditions ▪ Lesson 8: Loops ▪ Lesson 9: Functions ▪ Lesson 10: Exceptions and Exception Handling	▪ Lesson 1: Lists, Stacks, Queues ▪ Lesson 2: Structures and Enumerations ▪ Lesson 3: Delegates ▪ Lesson 4: Generics ▪ Lesson 5: Partial and Static Classes	▪ Lesson 1: Lambda Expressions ▪ Lesson 2: Singleton Pattern
Part of the School of Computer Science		