Variables and Data Types

- Declaration in C# syntax

  <type> <name>;

- All variables must be declared before you can use them.
- Include an initial values as follows:

  <type> <name> = <value>;

- Declaring a variable reserves a space in memory for the variable and allows you to reference it using the name.

Variables – Simple Types

- Numbers
- Boolean (true or false)
- Differ from complex types in that they cannot have attributes.
- Numeric types – there are many because of the mechanism of storing numbers in the computer as a series of 0s and 1s.

Variables – Simple Types

- Smallest unit of computer memory is a bit.
  - can be either one or zero – like a switch, it is either on or off.
  - for integer values you take a number of bits to represent your number in binary format.
- A variable storing N bits allows you to represent any number between 0 and $(2^N - 1)$.
- Any number larger is too big to store in this variable.

Variables – Simple Types

- Computer memory is organized, not by bits but by groups of 8 bits, called a byte.
- Signed numbers are handled by reserving one bit for the sign (note – there are different ways of representing negative numbers in memory – just note that there is one bit unavailable as part of the value)
- C# has predefined standard types in the .NET framework to represent integers.

Variables – Simple Types

Type	Alias For	Allowed Values
sbyte	System.Sbyte	Integer between -128 and 127
byte	System.Byte	Integer between 0 and 255
short	System.Int16	Integer between -32768 and 32767
ushort	System.UInt16	Integer between 0 and 65535
int	System.Int32	Integer between -2147483648 and 2147483647
uint	System.UInt32	Integer between 0 and 4294967295
long	System.Int64	Integer between -9223372036854775808 and 9223372036854775807
ulong	System.UInt64	Integer between 0 and 18446744073709551615

Variables – Simple Types

- Floating Point numbers. Numbers that are not whole and have a decimal part.
- $+/-m \times 2^e$ for float and double
- $+/-m \times 10^e$ for decimal

Type	Alias for	Min M	Max M	Min E	Max E	Approx. Min Val	Approx. Max Val
float	System.Single	0	2^{24}	-149	104	1.5×10^{-45}	3.4×10^{38}
double	System.Double	0	2^{53}	-1075	970	5.0×10^{-324}	1.7×10^{308}
decimal	System.Decimal	0	2^{96}	-28	0	1.0×10^{-28}	7.9×10^{28}

Variables – Simple Types

- Characters / Strings
- char is similar to ushort
- Strings can have any length – the amount of memory used just increases.

Type	Alias for	Allowed values
char	System.Char	Single Unicode character, stored and an integer between 0 and 65535
bool	System.Boolean	Boolean value, true or false
string	System.String	A sequence of characters

Review of some syntax rules

- C# code is made up of statements, each terminated with a semi-colon.
- White space (blanks, tab characters, new-line characters) are ignored by the compiler.
- Indenting makes your code more readable and understandable.
- C# is a block-structured language (statements are part of a block of code. Blocks are delimited by { } – known as curly braces.

Review of some syntax rules

Example – showing blocks and indentation:

```
{
    <some line of code>;
    {
        <another line>;
        <another line 2>;
    }
    <some other line>;
}
```

Review of some syntax rules

Comments:

// single line comment

/* multi-line

comment */

/// comments in XML format that can be used to generate documentation

Review of some syntax rules

- C# is case sensitive, meaning that it interprets uppercase and lowercase letters as different.

- WriteLine, writeline, WRITELINE, Writeline would all refer to a different method or variable.

C# Console Application

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text
namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
            // output text to the screen.
            Console.WriteLine("Hello World");
            Console.ReadLine();
        }
    }
}
```

Notes on literal values

- Double quotations to enclose strings. To actually assign a double quote in a string, you need to "escape" it by using a \ as follows:

  my string = "He said, \"Hi\"";

  This would print out as: He said, "Hi"

- \ is used for special characters: \' \" \\

  all stand for the escaped (second) character.

Notes on literal values

Escape Sequence	Character
\0	Null
\a	Alert (beep)
\b	Backspace
\f	Form feed
\n	New Line
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab

Notes on literal values

Numeric literals can use a suffix to specifically indicate the type.

Types	Category	Suffix	Example
int, unint, long, ulong	integer	none	100
uint, ulong	integer	u or U	100U
long, ulong	integer	l or L	100L
ulong	integer	any comb of u and l	100UL
float	real	f or F	1.5F
double	real	none, d of D	1.5
decimal	real	m or M	1.5M
bool	Boolean	none	true or false

Expressions

- C# Operators allow you to manipulate variables
- Combine operators with variables and literals (operands) to create expressions

- Types of operators include mathematical, assignment and logical operators

Expressions

- Categories of operators as follows:
  - Unary – Act on a single operand (a few)
  - Binary – Act on two operands (most)
  - Ternary – Act on three operands (only one)

Mathematical operators

Operator	Category	Example	Result
+	Binary	var1 = var2 + var3;	var1 is assigned the value that is the sum of var2 and var3
-	Binary	var1 = var2 – var3;	var1 is assigned the value that is the result of var3 subtracted from var2
*	Binary	var1 = var2 * var3;	var1 is assigned the value that is the product of var2 and var3
/	Binary	var1 = var2 \ var3;	var1 is assigned the value that is the result of dividing var2 by var3
%	Binary	var1 = var2 % var3;	var1 is assigned the remainder resulting when var2 is divided by var3
+	Unary	var1 = +var2;	var1 is assigned the value of var2
-	Unary	var1 = -var2;	var1 is assigned the value of var2 multiplied by -1

Mathematical operators on strings – concatenation

Operator	Category	Example	Result
+	Binary	var1 = var2 + var3;	var1 is assigned the value that is the concatenation of the two strings stored in var2 and var3.

Example:

var2 = "This is one string"
var3 = " and this is another string"

var1 = var2 + var3 results in "This is one string and this is another string"

Increment and decrement operator (add or subtract 1)

Operator	Category	Example	Result
++	Unary	var1 = ++var2;	var1 is assigned the value of var2 + 1 and var2 is incremented by 1.
- -	Unary	var1 = - - var2;	var1 is assigned the value var2 – 1 and var2 is decremented by 1.
++	Unary	var1 = var2++;	var1 is assigned the value of var2 var2 is then incremented by 1.
- -	Unary	var1 = var2- -;	var1 is assigned the value of var2 var2 is then decremented

Convert

- Allows you to explicitly convert between types: integers to doubles, strings to integers, strings to doubles and so forth.

- Useful for mathematical calculations and for taking string input and storing it in numeric variables

Assignment operators = is what we have seen so far

Operator	Category	Example	Result
=	Binary	var1 = var2 ;	var1 is assigned the value of var2
+=	Binary	var1 += var2 ;	var1 is assigned the sum of var1 and var2
- =	Binary	var1 -= var2 ;	var1 is assigned the value of var2 subtracted from var1
*=	Binary	var1 *= var2;;	var1 is assigned the product of var1 and var2
/*	Binary	var1 /= var2;	var1 is assigned the value that is the result of dividing var1 by var2
%=	Binary	var1 %= var2;	var1 is assigned the remainder when var1 is divided by var1

Operator Precedence

Order in sequence, except as controlled by parentheses.

Order highest to lowest

- ++, - - (used as prefixes), +, - (unary)
- *, /, %
- +, - (binary)
- =, *=, /=, %=, +=, -=
- ++, - - (used as suffixes)