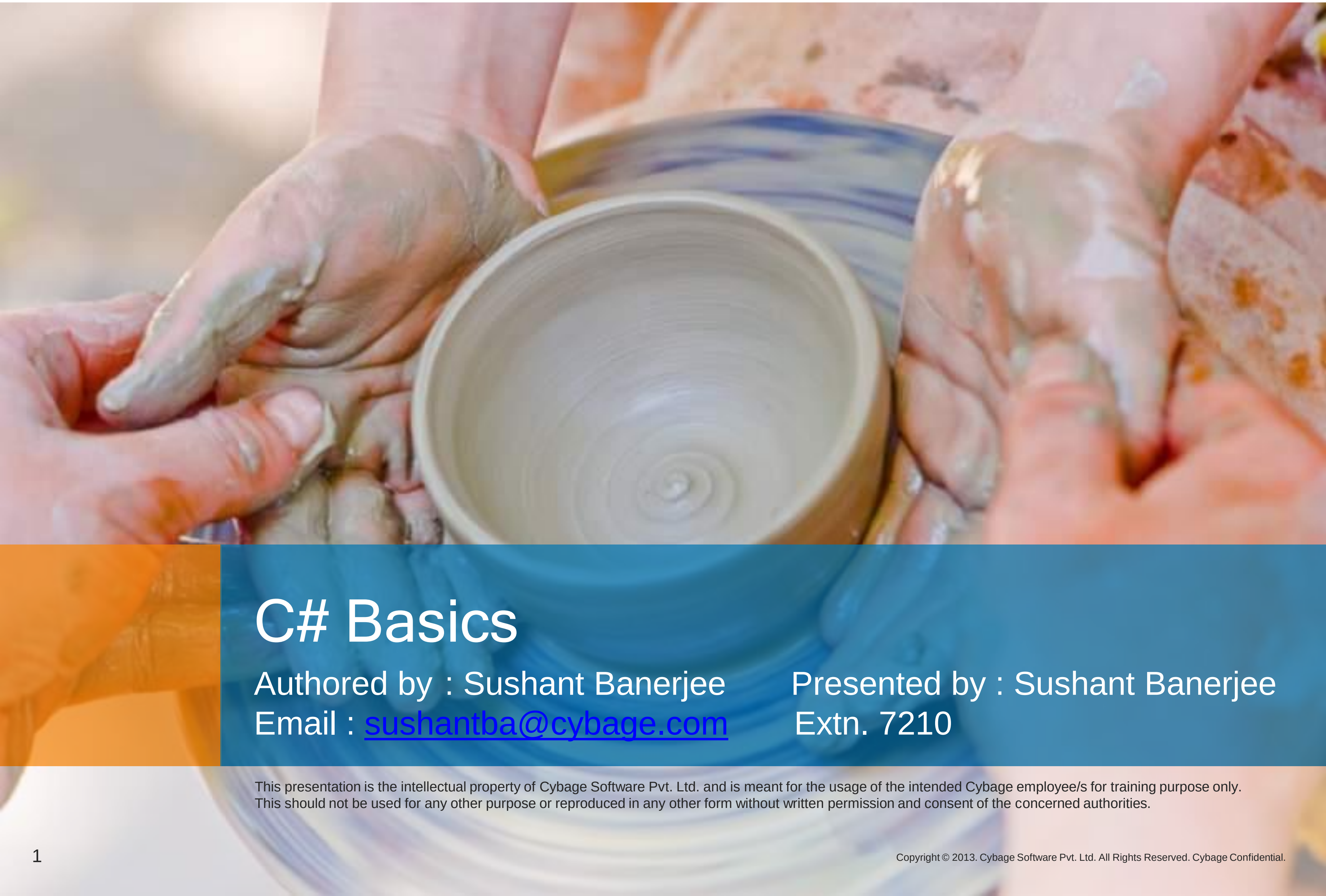


C# Basics

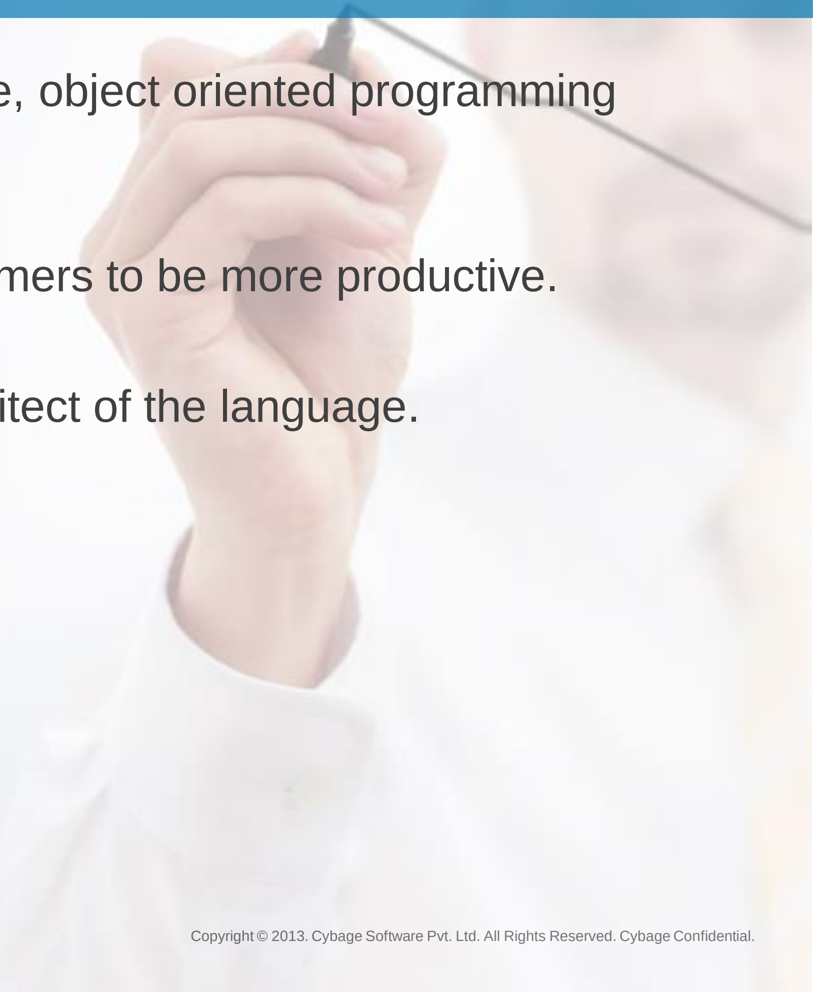
Authored by : Sushant Banerjee
Email : sushantba@cybage.com

Presented by : Sushant Banerjee
Extn. 7210

Agenda

- Introduction to C#
- C# Language Fundamentals
- Type in C#
- Type Conversion
- Operators and Expressions
- Statements
- String
- Array
- Date and Time

What is C#



- C# is a general purpose, type-safe, object oriented programming language.
- The goal of C# is to help programmers to be more productive.
- Anders Hejlsberg is the chief architect of the language.
- C# is case sensitive.

Evolution of C#

Version	Date	.NET Framework	Visual Studio
1.0	Jan 2002	1.0	2002
1.2	April 2003	1.1	2003
2.0	Nov 2005	2.0	2005
3.0	Nov 2007	3.0 and 3.5	2008
4.0	April 2010	4.0	2010
5.0	Aug 2012	4.5	2012

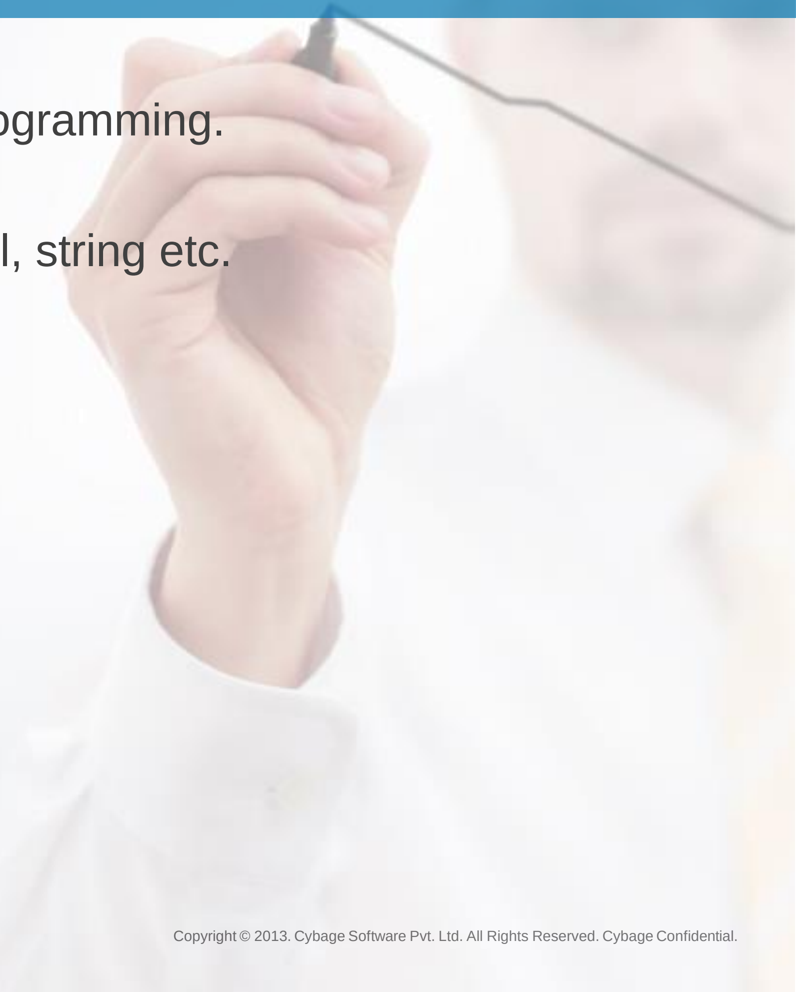
Writing C# Programs

- Using Simple Text Editor – Notepad
- Basic Structure of a C# program
- Code Compilation Process in C#
- Command line arguments
- Using Visual Studio IDE

General Syntax

- Identifiers
- Keywords
- Literals
- Punctuators
- Operators
- Comments

C# Types

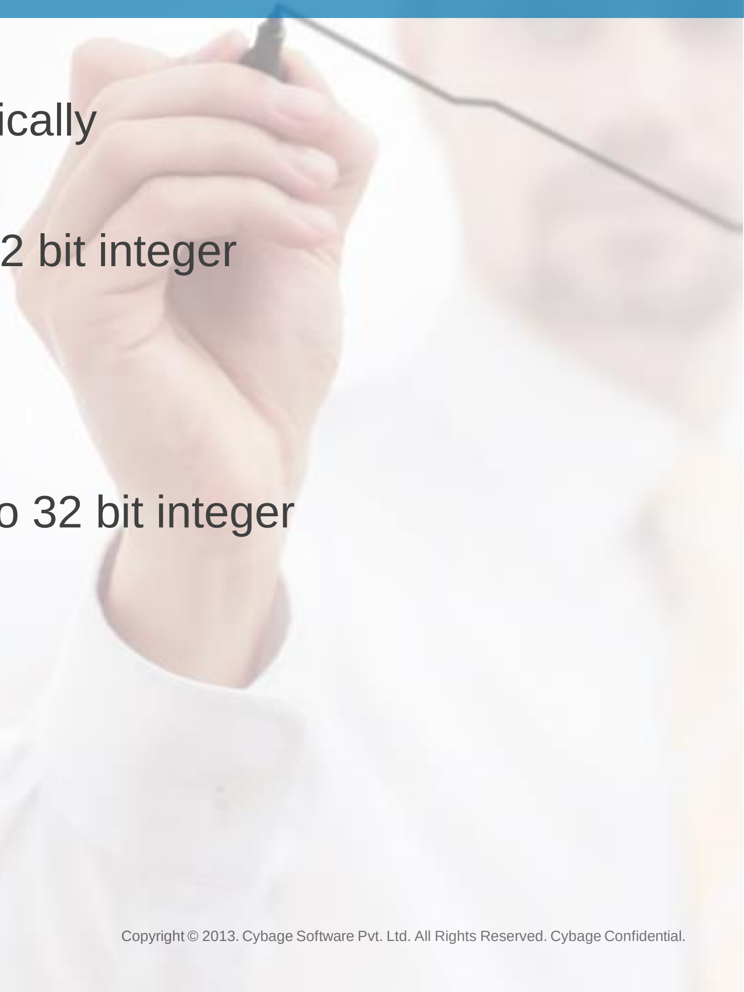


- Types are building blocks of C# programming.
- Pre-defined Types such as int, bool, string etc.
- Custom Types such as class
- Value Types and Reference Types
- Default values of types in C#
- Declaring variable and constrants

Type Conversion

- Implicit Conversion happens automatically
  - short s = 10; //16 bit integer
  - int i = s; //implicit conversion to 32 bit integer

- Explicit Conversion requires casting
  - long l = 1234; //64 bit integer
  - int i = (int)l;//Explicit conversion to 32 bit integer



Predefined Types in C#

- Value Types
  - Numeric
    - Signed Integer (sbyte, short, int, long)
    - Unsigned Integer (byte, ushort, uint, ulong)
    - Real Number (float, double, decimal)
  - Logical (bool)
  - Character (char)
- Reference Types
  - String (string)
  - Object (object)

Numeric Types

- Integral Signed

C# Type	System Type	Suffix	Size	Range
sbyte	SByte		8 bits	-128 to 127
short	Int16		16 bits	-32,768 to 32,767
int	Int32		32 bits	-2,147,483,648 to 2,147,483,647
long	Int64	L	64 bits	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807

Numeric Types (Contd…)

- Integral Unsigned

C# Type	System Type	Suffix	Size	Range
byte	Byte		8 bits	0 to 255
ushort	UInt16		16 bits	0 to 65,535
uint	UInt32	U	32 bits	0 to 4,294,967,295
ulong	UInt64	UL	64 bits	0 to 18,446,744,073,709,551,615

Numeric Types (Contd…)

- Real Numbers

C# Type	System Type	Suffix	Size	Approximate Range	Precision
float	Single	F	32 bits	±1.5e−45 to ±3.4e38	7 digits
double	Double	D	64	±5.0e−324 to ±1.7e308	15 – 16 digits
decimal	Decimal	M	128 bits	±1.0 × 10−28 to ±7.9 × 1028	28 – 29 digits

Other Types

C# Type	System Type	Size	Range
Bool	Boolean	8 bits	True or False
char	Char	Unicode 16 bits	U+0000 to U+FFFF
string	String	2 GB	0 to 2 Billion characters

Implicitly Typed Local Variable (C# 3.0)

- The **var** keyword is used to declare variables and the type of the variables are inferred by the compiler.
- These variables must be declared and initialize in one step.

```
var x = 5;
var s = "some text";
var f = 12.5F;
```

- Once initialized type can not be changed

```
var x = 5;
x = "abc";//compile time error since x is of type int
```

Boxing and Unboxing

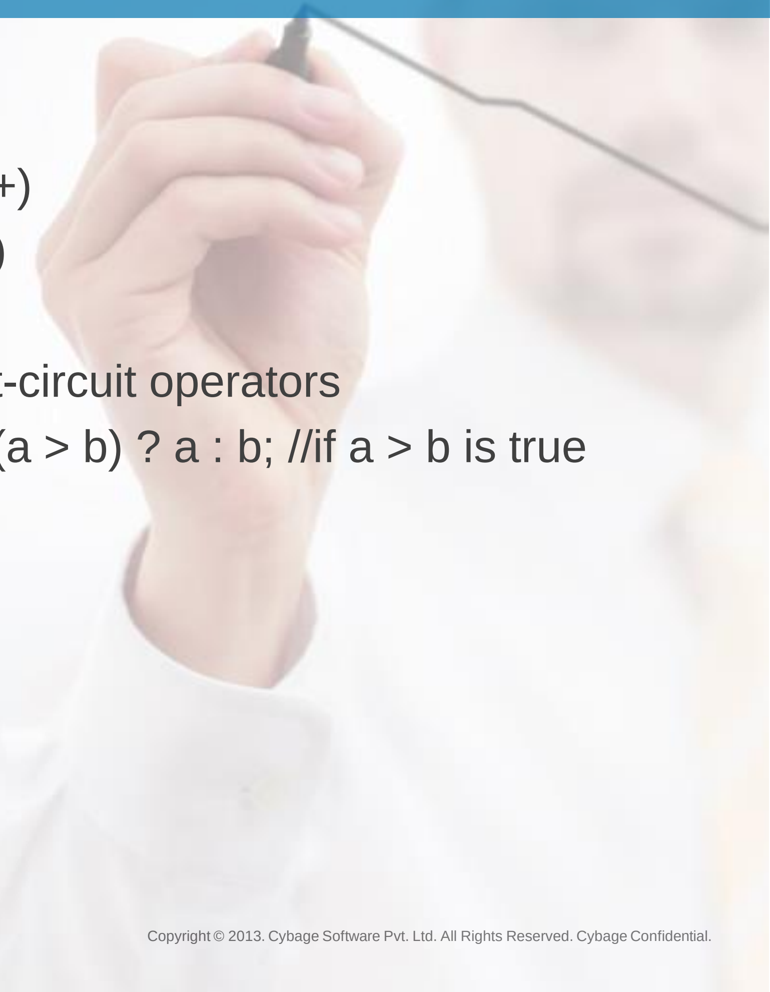
- Converting a value type to a reference type is known as boxing and the reverse is unboxing.

```
int x = 25;

object o = x;//implicit boxing
object o2 = (object)x;//explicit boxing

x = o;//implicit unboxing will produce compile time error
x = (int)o;//explicit unboxing will compile successfully
```

Operators and Expressions



- Arithmetic Operators (+ - * / %)
- Increment – Pre (++1) and Post (1++)
- Decrement – Pre (--1) and Post (1--)
- Comparison (==, <=, >=, >, <, !=)
- Conditional && , || also called short-circuit operators
- Conditional or ternary operator x = (a > b) ? a : b; //if a > b is true returns a otherwise returns b

Statements

- Declaration Statements – variables, local, fields
- Expression Statements – assignment, method call or object instantiation statements
- Selection Statements – if-else, switch
- Iteration Statements – while, do-while, for, foreach
- Jump Statements- break, continue, goto, return, throw

String Type

- Verbatim String
- String Concatenation
- StringBuilder class

String Manipulation Methods

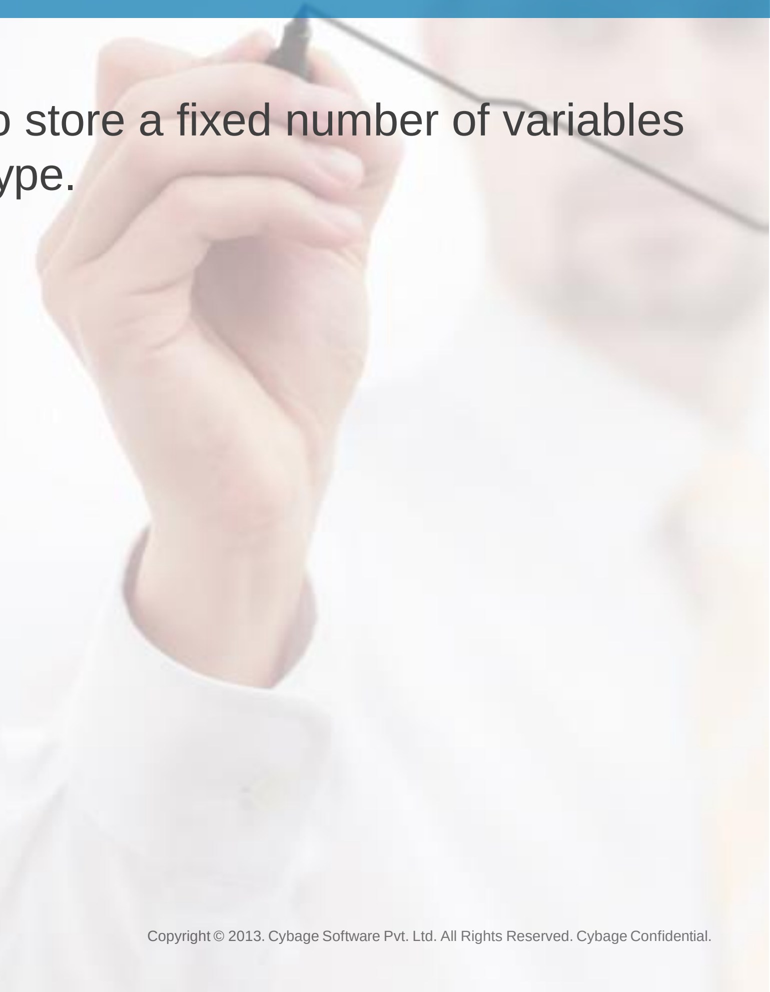
Method	Description
Contains, StartsWith, EndsWith	To search specific word in a string
IndexOf	Returns index of specific character or string in a string value
Substring	Extracts part of the string
Insert, Remove, Replace	To insert or remove characters
TrimStart, TrimEnd, Trim	To remove whitespace characters
ToUpper, ToLower	Returns uppercase or lowercase string
Split and Join	Split a sentence into array of words and join does opposite

String Formatting

```
string mystring = string.Format("Make : {0}, Model : {1}", "BMW", 7243);
//display default currency format of your system
string mystring = string.Format("{0:C}", 5000);
//display large number in readable format adding , in required place
string mystring = string.Format("{0:N}", 123456789);
//display percentage % of a number
string mystring = string.Format("{0:P}", .123);
//display phone number in specified format (if you remove last #, numbers
  will move right to left)
string mystring = string.Format("Phone Number : {0:(###) ###-####}",
  1234567890);
Console.WriteLine(mystring);
```

Array

- An array data structure allows you to store a fixed number of variables (also called elements) of the same type.
- Basically you can create
  - Single Dimensional arrays
  - Multidimensional arrays
  - Jagged arrays



Single Dimensional Array

```
// Declare a single-dimensional array
int[] array1 = new int[5];

// Declare and set array element values
int[] array2 = new int[] { 1, 3, 5, 7, 9 };

// Alternative syntax
int[] array3 = { 1, 2, 3, 4, 5, 6 };
```

Two Dimensional Array

```
// Declare a two dimensional array [row, column]
int[,] multiDimensionalArray1 = new int[2, 3];

// Declare and set array element values
int[,] multiDimensionalArray2 = { { 1, 2, 3 }, { 4, 5, 6 } };
```

Jagged Array

- A jagged array is an array whose elements are arrays.
- The elements of a jagged array can be of different dimensions and sizes.
- A jagged array is sometimes called an "array of arrays."
- The following is a declaration of a single-dimensional array that has three elements, each of which is a single-dimensional array of integers:

```
int[][] jaggedArray = new int[3][];
```

- Before using a jaggedArray, its elements must be initialized.

```
jaggedArray[0] = new int[5];
jaggedArray[1] = new int[10];
jaggedArray[2] = new int[15];
```

Working With DateTime

- Creating DateTime Object
- Using DateTime Methods
- Using DateTime Properties
- Using TimeSpan

Bibliography, Important Links

- C# 5.0 in a Nutshell – Published by O'Reilly
- www.msdn.com – Library
- http://en.wikipedia.org

Any Questions?

Thank you!