ISTANBUL KÜLTÜR UNIVERSITY, DEPARTMANT OF COMPUTER ENGINEERING

CSE5001 WEB PROGRAMMING LAB MANUAL

Lab 04: C# Essentials

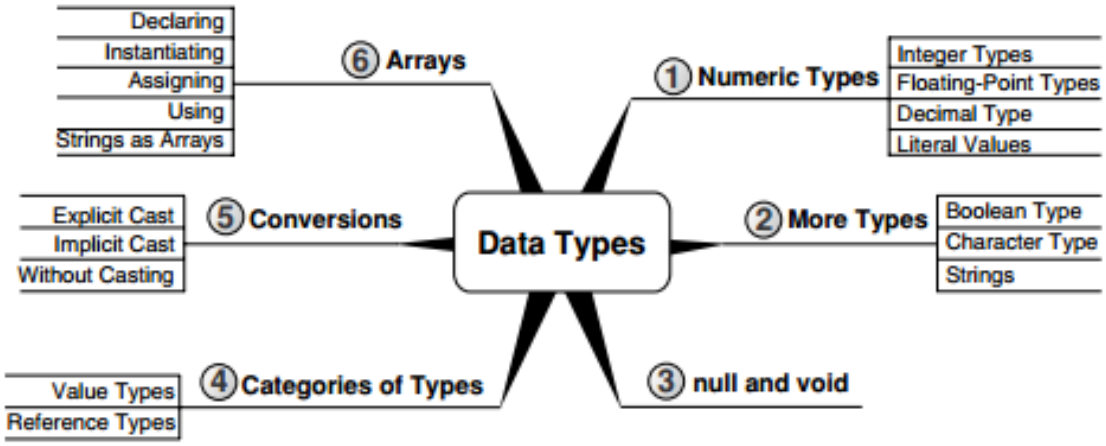
Objective: To Acquire a working knowledge of C# programming

- Declaring and initializing variables with a variety of data types
- Exploring operators,
- Controlling flow with conditional code and loops
- Creating custom classes

Summary: C# is an object-oriented language designed by Microsoft and used by systems engineers, desktop programmers, and mobile app developers. The goal of C# is to provide a simple, safe, modern, object-oriented, high performance, robust and durable language for .NET development.

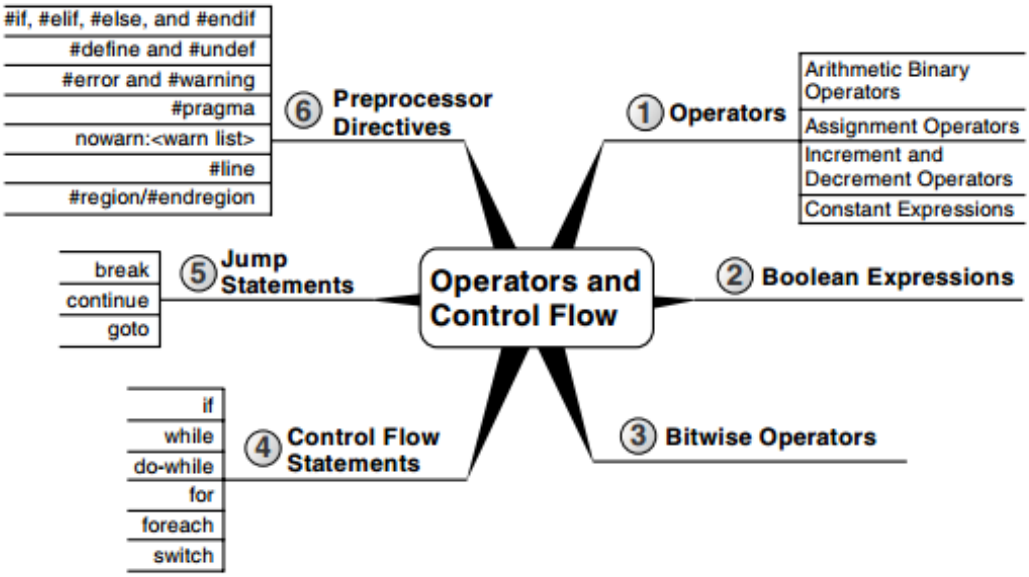
Data Types:

In C# thousands of types exist, and you can combine types to create new types. A few types in C#, however, are relatively simple and are considered the building blocks of all other types. These types are the **predefined types**. The C# language's predefined types include eight integer types, two binary floating-point types for scientific calculations and one decimal float for financial calculations, one Boolean type, and a character type.



Operators and Control Flow:

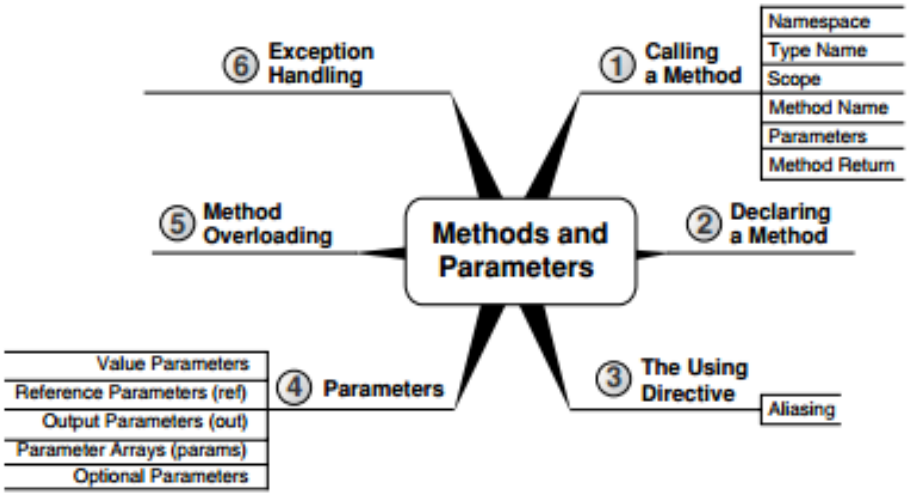
Operators provide syntax for performing different calculations or actions appropriate for the operands within the calculation. **Control flow** statements provide the means for conditional logic within a program or looping over a section of code multiple times.



Methods and Parameters:

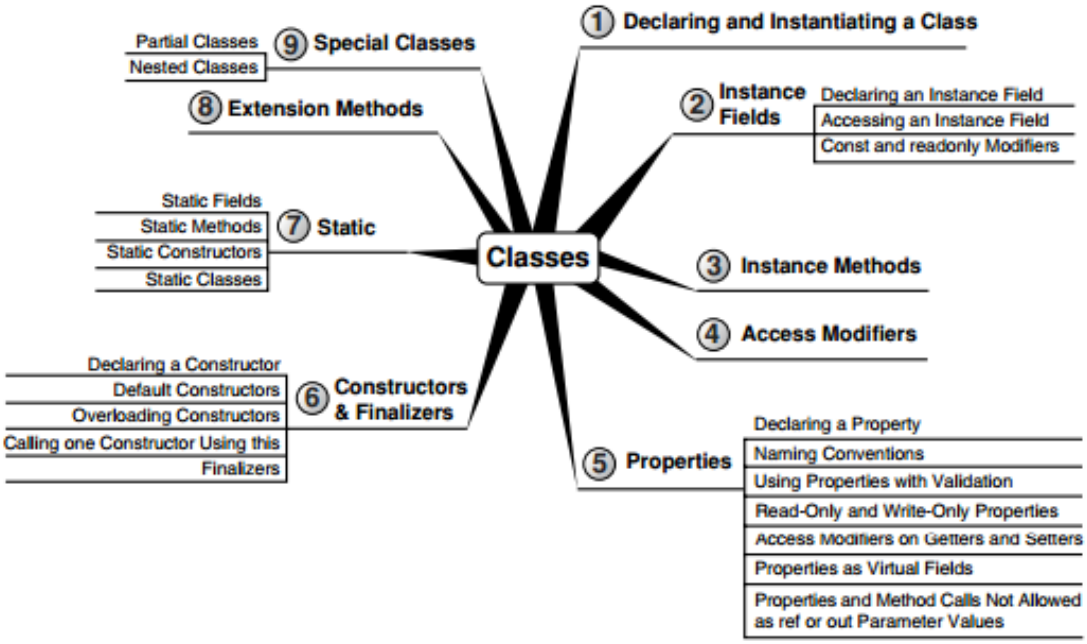
A **method** is a means of grouping together a sequence of statements to perform a particular action or compute a particular result. This provides greater structure and organization for the statements that comprise a program.

Methods can receive data via arguments that are supplied for their **parameters**. Parameters are variables used for passing data from the caller (the code containing the method call) to the invoked method (Write(), WriteLine(), GetFiles(), CountLines(), and so on).



Classes:

Classes, which are the templates for objects themselves.



Procedure 01: Hello World and Data types

1. Open Visual Studio for Web.
2. In the **File** menu, click **New Web Site**.
   The **New Web Site** dialog box is displayed.
3. Under **Installed**, click **Visual C#** and then select **ASP.NET Empty Web Site**.
4. In the **Web location** box, select **File System**, and then enter the name of the folder where you want to keep the pages for your website.
   For example, type the folder name **C:\Lab04_CSharpEssentials**. Click **OK**.
5. In **Solution Explorer**, right-click the root of your Web site, and then click **Add New Item**.
6. Select **Web Form**, name the file **DataTypes.aspx**, and then click **Add**.
7. Right click in the page itself and choose **View Code**
8. Add the following code block to your Page_Load()

```
string myText;
myText = "Hello Wold"; //variable assignment
Response.Write(myText);//The Write method writes a specified string to the current
HTTP output
```

9. Output: Hello World
10. Add the following code block and run again

```
int number = 5;
Response.Write("<br>" + number);
```

11. Converting a string to an int

```
string firstName ="john";
string ageText ="7";
int age;
age = int.Parse(ageText);
Response.Write("<br> Age converted to int "+ age);
```

12. Format numbers to a string variable

```
double dnumber = 1.256988745;
Response.Write(dnumber.ToString("#,##0.00")); //display two digit after decimal
point
```

13. Round-Trip Formatting
    - To more accurately identify the string representation of the double value it is possible to convert it using a format string and the round-trip format specifier, R (or r).

```
double d1 = 1.618033988749895;
Response.Write("<br> Original number:" + d1);
string s = string.Format("{0:R}", d1);
Response.Write("<br> Number to string (R):" + s);
double d2 = double.Parse(s);
Response.Write("<br> String to number:" + d2);
```

Procedure 02: Operators

*(open a new blank web page with the name **Operators.aspx**)*

8. Add the following codes to your Page_Load() and run it

```
int total = 7 + 3;
int difference = 7-3;
int multiplication = 7 * 3;
float division = 7 / 3;

Response.Write("<br> Plus Result(7+3):" + total);
Response.Write("<br> Minus Result(7-3): " + difference);
Response.Write("<br> Multiplication Result(7*3): " + multiplication);
Response.Write("<br> Division Result(7/3): " + division);
```

9. Output :
   Plus Result(7+3):10
   Minus Result(7-3): 4
   Multiplication Result(7*3): 21
   Division Result(7/3): 2

10. Increment and Decrement Operators

```
int spaceCount = 8, lines = 5;

spaceCount = spaceCount + 1;
spaceCount += 1;
spaceCount++;
```

```
Response.Write("<br> spaceCount:" + spaceCount);

lines = lines - 1;
lines -= 1;
lines--;
Response.Write("<br> lines:" + lines);
```

11. Increment Comparing the Prefix and Postfix Increment Operators

```
int x = 123;
x++;
x++;
Response.Write("<br> Postfix x" + x);
++x;
++x;
Response.Write("<br> Prefix x" + x);
```

12. Assigning the Result of a Relational Operator to a bool Variable

```
bool result = 60 > 6;
Response.Write("<br> 60 > 6 : " + result);
```

Procedure 03: Control Flow Statements

*(open a new blank web page with the name **ControlFlow.aspx**)*

8. Add the following if/else Statement to your Page_Load() and run it

```
int input = 3; //assign different number and try again

if (input <= 0)
       Response.Write("<br> Input is less than or equal to 0.");
else if (input == 1)
       Response.Write("<br> Input is equal to 1.");
else Response.Write("<br> Input is is greater than 1.");
```

13. Using the OR Operator in if statement

```
int hourOfTheDay = 25;

if ((hourOfTheDay > 24) || (hourOfTheDay < 1))
    Response.Write("<br> (OR Operator) The time you entered is invalid.");
```

14. Using the AND Operator in if statement

```
if ((hourOfTheDay > 24) &&  (hourOfTheDay < 1))
    Response.Write("<br> (AND Operator) bigger than 1 and less than 24");
```

15. Using for loop

```
int count = 5;
for (int i = 1; i <= count; i++)
{
       Response.Write("<br>(For loop)" + i.ToString());
}
```

Procedure 04: Methods and Parameters

*(open a new blank web page with the name **MethodParameters.aspx**)*

8. Write a method to display Full Name (return void)

```
protected void Page_Load(object sender, EventArgs e)
{
    string firstName ="Fatma";
    string lastName ="Patlar Akbulut";

    DisplayFullName(firstName, lastName);
}
/* fname  and lname are Parameters of DisplayFullName Method*/
public void DisplayFullName(string fname, string lname)
{
    Response.Write(fname + " " + lname);
}
```

9. Output: Fatma Patlar Akbulut

10. Write a method to return welcome message (return string)

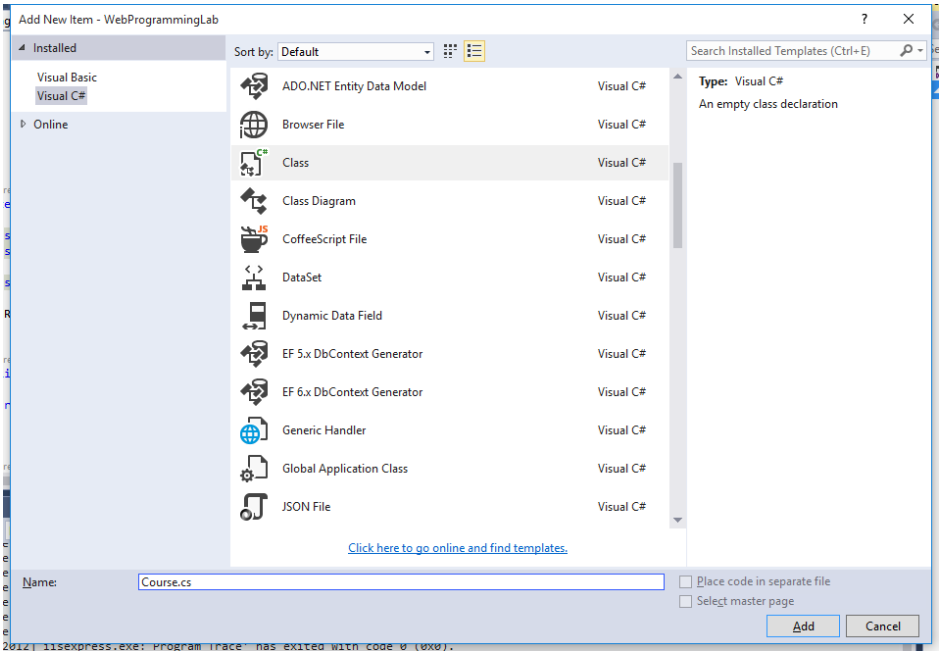
```
protected void Page_Load(object sender, EventArgs e)
{
    string firstName ="Fatma";
    string lastName ="Patlar Akbulut";

    string welcomeMessage="Welcome";

    Response.Write(GetGreeting(firstName, welcomeMessage));
}
public string GetGreeting(string fname, string message)
{
   return message + " " +fname;
}
```

11. Output : Welcome Fatma

Procedure 05: Declaring and Instantiating a Class

*(open a new blank web page with the name **Class.aspx**)*

7. In **Solution Explorer**, right-click and choose **Add**, and then click **Class**.
   The **Add New Item** dialog box is displayed.
8. Type **Course.cs** in the **Name** box and then click **Add** to create the class.



A class named Course is added to your class library.

9. In the code editor, replace the existing code with the following code to define the **Course** class

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

/// <summary>
/// Summary description for Course
/// </summary>
public class Course
{
    /* Fields*/
    private int _courseCode;
    private string _courseName;

    /*constructor*/
    public Course()
    {
        CourseCode = 0;
        CourseName = "undefined";
    }
    /*properties*/
    public int CourseCode
    {
        get { return _courseCode; }
        set { _courseCode = value; }
    }
    public string CourseName
    {
        get { return _courseName; }
        set { _courseName = value; }
    }

    /*method*/
```

```
    public string GetCourseType()
    {
        if (_courseCode < 5000)
            return "Technical Elective";
        else
            return "Core";
    }
}
```

11. In the **File** menu, click **Save All** to save the project. Open your **Class.aspx** switch to code behind and write the following code block on Page_Load()

```
Course myCourse = new Course();  //Instance of Course Class
myCourse.CourseName = " Web Programming ";
myCourse.CourseCode = 5001;
Response.Write("Course Name: " + myCourse.CourseName + // Accessing class fields
"<br/> Course Code: " + myCourse.CourseCode.ToString() +  // Accessing class
fields
"<br/> Course Type: " + myCourse.GetCourseType()); //method calling
```

12. Press CTRL+F5 to start the solution.

Now you can start your lab assignment!

Take Home Exercises

1. Write a program to accept **two integers** and check whether they are **equal** or **not**.
2. Write a program that takes two money type to perform **Money Conversion** each other.
3. Write a program to generate the **Temperature Conversion**.
4. Write a function to generate the **Factorial** operation of a given number.
5. Write a function that takes **five numbers** as input to calculate and display the **average**.
6. Write a program to display individual characters of string in **reverse order**.
7. Write a program to **count total number** of alphabets, digits and special characters in a string.
8. Write a program **count** total number of **vowel** in a string.
9. Write a program that find the **number of times** a **substring** appears in a given string.
10. Write a program to sort a string **array** in **ascending** order.
11. Write a program to find **sum** of all elements of **array**.

USEFUL INFORMATION

Escape Characters

Escape Sequence	Character Name	Unicode Encoding
\'	Single quote	\u0027
\"	Double quote	\u0022
\\	Backslash	\u005C
\0	Null	\u0000
\a	Alert (system beep)	\u0007
\b	Backspace	\u0008
\f	Form feed	\u000C
\n	Line feed (sometimes referred to as a newline)	\u000A
\r	Carriage return	\u000D
\t	Horizontal tab	\u0009
\v	Vertical tab	\u000B
\uxxxx	Unicode character in hex	\u0029
\x[n][n][n]n	Unicode character in hex (first three placeholders are options); variable-length version of \uxxxx	\u3A
\Uxxxxxxxx	Unicode escape sequence for creating surrogate pairs	\uD840DC01 (𠀁)

Fundamental Numeric Types

Type	Size	Range (Inclusive)	BCL Name	Signed	Literal Suffix
sbyte	8 bits	–128 to 127	System.SByte	Yes	
byte	8 bits	0 to 255	System.Byte	No	
short	16 bits	–32,768 to 32,767	System.Int16	Yes	
ushort	16 bits	0 to 65,535	System.UInt16	No	
int	32 bits	–2,147,483,648 to 2,147,483,647	System.Int32	Yes	
uint	32 bits	0 to 4,294,967,295	System.UInt32	No	U or u
long	64 bits	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	System.Int64	Yes	L or l
ulong	64 bits	0 to 18,446,744,073,709,551,615	System.UInt64	No	UL or ul

Floating-Point Types

Type	Size	Range (Inclusive)	BCL Name	Significant Digits	Literal Suffix
`float`	32 bits	$\pm 1.5 \times 10^{-45}$ to $\pm 3.4 \times 10^{38}$	`System.Single`	7	F or f
`double`	64 bits	$\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$	`System.Double`	15–16	⇆

Decimal Type

Type	Size	Range (Inclusive)	BCL Name	Significant Digits	Literal Suffix
`decimal`	128 bits	1.0×10^{-28} to approximately 7.9×10^{28}	`System.Decimal`	28–29	M or m

string Static Methods

Statement	Example

`static string string.Format(`
`  string format,`
`  ...)`
	`string text, firstName, lastName;`
	`//...`
	`text = string.Format("Your full name is {0} {1}.",`
	`  firstName, lastName);`
	`// Display`
	`// "Your full name is <firstName> <lastName>."`
	`System.Console.WriteLine(text);`

`static string string.Concat(`
`  string str0,`
`  string str1)`
	`string text, firstName, lastName;`
	`//...`
	`text = string.Concat(firstName, lastName);`
	`// Display "<firstName><lastName>", notice`
	`// that there is no space between names.`
	`System.Console.WriteLine(text);`

`static int string.Compare(`
`  string strA,`
`  string strB)`
`static int string.Compare(`
`  string strA,`
`  string strB)`
`  string ignoreCase)`
	`string option;`
	`//...`
	`// String comparison in which case matters.`
	`int result = string.Compare(option, "/help");`
	
	`// Display:`
	`//  0 if equal`
	`//  negative if option < /help`
	`//  positive if option > /help`
	`System.Console.WriteLine(result);`

	`string option;`
	`//...`
	`// Case-insensitive string comparison`
	`int result = string.Compare(`
	`  option, "/Help", true);`
	
	`// Display:`
	`//  0 if equal`
	`//  < 0 if option < /help`
	`//  > 0 if option > /help`
	`System.Console.WriteLine(result);`

string Methods

Statement	Example

`bool StartsWith(`
`  string value)`
`bool EndsWith(`
`  string value)`
	`string lastName`
	`//...`
	`bool isPhd = lastName.EndsWith("Ph.D.");`
	`bool isDr = lastName.StartsWith("Dr.");`

`string ToLower()`
`string ToUpper()`
	`string severity = "warning";`
	`// Display the severity in uppercase`
	`System.Console.WriteLine(severity.ToUpper());`

`string Trim()`
`string Trim(...)`
`string TrimEnd()`
`string TrimStart()`
	`// Remove any whitespace at the start or end.`
	`username = username.Trim();`

`string Replace(`
`  string oldValue,`
`  string newValue)`
	`string filename;`
	`//...`
	`// Remove ?'s from the string`
	`filename = filename.Replace("?", "");;`

The Relational and Equality Operators

Operator	Description	Example
<	Less than	`input<9;`
>	Greater than	`input>9;`
<=	Less than or equal to	`input<=9;`
>=	Greater than or equal to	`input>=9;`
==	Equality operator	`input==9;`
!=	Inequality operator	`input!=9;`

References

1. Michaelis, Mark, and Eric Lippert. Essential C# 6.0. Addison-Wesley Professional, 2015.