**Lab Instructions and Solution for Module- 2:** **C# Basic and Statement Tutorials**

**Objectives**

After completing this module you will be able to learn:

* Data types and variables and type conversions
* CSharp statements
* Class, object, method, variable passing

**Task 1: Variables,Data Types and Type Conversions**

### Variables: A variable is referred to a memory location in which a value can be stored temporarily. Programmes work by manipulating data stored in memory. These storage areas come under the general heading of Variables. You can write C# code to read the value from the memory location or write something to it.

### Data types: Data Types in a programming language describes that what type of data a variable can hold . In C# every variable and object must have a declared type.

**Syntax : DataType VariableName**

**DataType : The type of data that the variable can hold**

**VariableName :The variable which you declare for hold the values.**

**Examples: int sum;**

**Here,int is the data type and sum is the variable name**

**int**

int variables are stored signed 32 bit integer values in the range of -2,147,483,648 to +2,147,483,647

**CSharp declaration : int sum;**

**CSharp Initialization : sum = 100;**

**CSharp default initialization value : 0**

**decimal**

The decimal keyword denotes a 128-bit data type.The approximate range and precision for the decimal type are -1.0 X 10-28 to 7.9 X 1028

**CSharp declaration : decimal val;**

**CSharp Initialization : val = 0.12M;**

**CSharp default initialization value : 0.0M**

**string**

The string type represents a string of Unicode characters. string variables are stored any number of alphabetic,numerical, and special characters .

**CSharp declaration : string str;**

**CSharp Initialization : str = "Hello World";**

**bool**

The bool keyword is an alias of System.Boolean. It is used to declare variables to store the Boolean values, true and false. In C# , there is no conversion between the bool type and other types.

**CSharp declaration : bool flag;**

**CSharp Initialization : flag = true;**

**CSharp default initialization value : false**

**float**

**CSharp declaration : float myFloat;**

**CSharp Initialization : myFloat = 0.50F;**

**double**

**CSharp declaration : double myDouble;**

**CSharp Initialization : myDouble = 0.009;**

Type Conversion:

In C# windows application every input works as a string type of text.If we work with string type then no need to convert. But, if we work with other data type then we need to convert string data type to other data type. If we see output, it also works with string type of text. So we need to again convert other datatype to string type when we see output.

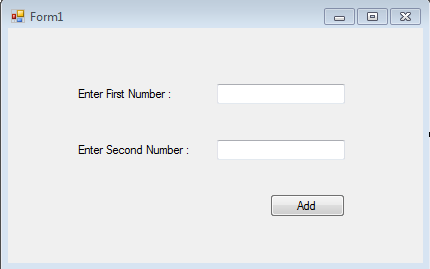
Examples for type conversion-1 :

If you want to convert a string that contains a number to an int/float/double data type, you can use Parse() method. Use ToString() method convert int/float/double to string for showing output.

We create a desktop application :

1. Create a desktop application and change it’s properties
2. Design the user interface like the following figure(Fig-1)

**1**



1. Double click on the Add.
2. Write down the code snippet in the block

private void addB\_Click(object sender, EventArgs e)

{

String firstN = firstNumberTB.Text;

String secondN = secondNumberTB.Text;

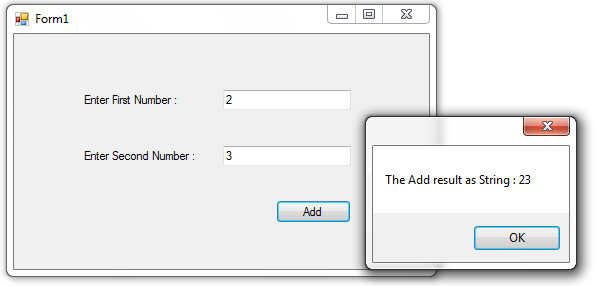
String result = firstN + secondN;

MessageBox.Show("The add result : "+result);

}

1. Build the solution and run it and You will get the following picture(Fig-2)

**2**



Here, We see the number are concatenated. To solve this we have to convert those input to number type (int, float, double etc.)

Examples for type conversion-2:

**C#** allows you to convert from a type to other allowable types by specifying the type to be converted using ‘type.parse(value)’. Such as-

* String num =”10”;

int number= int.parse(num); //convert string type to int type

* int x=10;

float y =float.parse(x); //convert int type to float type

* float x=1.44;

int y=int.parse(x); //convert float type to integer type

From this knowledge, we have to change a little in our code.

1. Double click on the Add.
2. Write down the code snippet in the block

private void addB\_Click(object sender, EventArgs e)

{

int firstN = int.Parse(firstNumberTB.Text);

int secondN = int.Parse(secondNumberTB.Text);

string result = firstN + secondN;

MessageBox.Show("The Add result as Number : " + result);

}

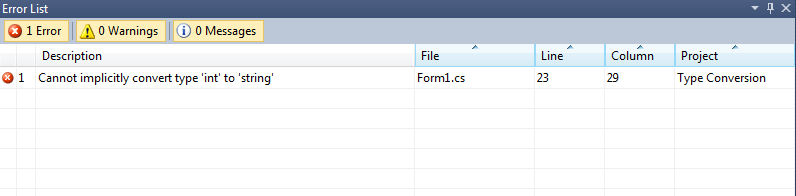
1. Build the solution and run it and it will create an error. You will get the following picture(Fig-3)

**3**



You should see a blue wiggly line under your MessageBox code. From Error List you can see the error description. We will see the following picture (Fig -4)

**4**



From the error description we see that, we have to convert type ‘int’ to ‘string’. Double click on the error we will reach on the error line. Change the type.

private void addB\_Click(object sender, EventArgs e)

{

int firstN = int.Parse(firstNumberTB.Text);

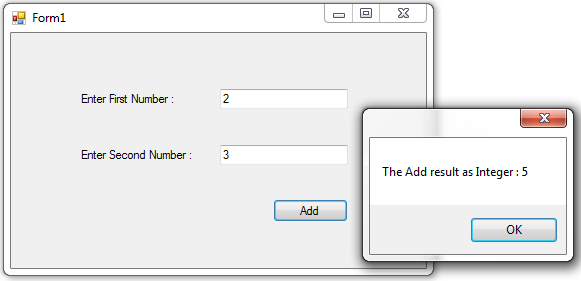
int secondN = int.Parse(secondNumberTB.Text);

int result = firstN + secondN;

MessageBox.Show("The Add result as Integer : " + result.ToString());

}

Build the solution and run it and it will create an error. You will get the following picture(Fig-5)

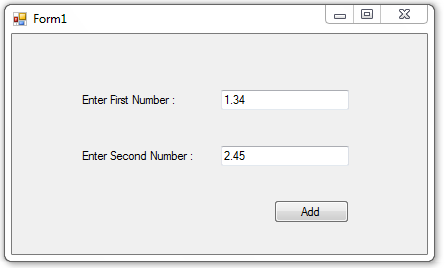


**5**

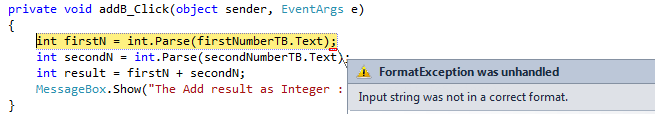
Examples for type conversion-3:

From the example, there should be caused error in some cases. See the picture below(Fig-6). What will be happen?

**6**



It will produce an exception like below.



To solve this, we have to change the type of the variable. It will look like this

private void addB\_Click(object sender, EventArgs e)

{

double firstN = double.Parse(firstNumberTB.Text);

double secondN = double.Parse(secondNumberTB.Text);

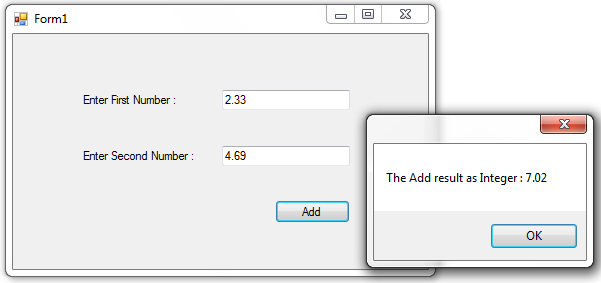
double result = firstN + secondN;

MessageBox.Show("The Add result as Integer : " + result.ToString());

}

Now, build the solution and run it and it will create an error. You will get the following picture(Fig-7)

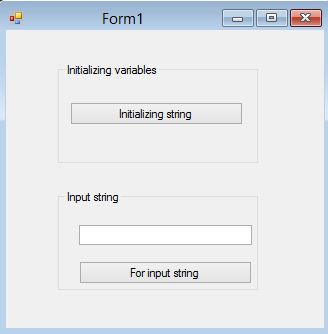
**7**



Examples for type conversion-4:

If we working with string type no need to convert. At first create a desktop application and design the user interface like this(Fig-8)

**8**



For initializing:

private void initial\_button\_Click(object sender, EventArgs e)

{

string x = "My String";

MessageBox.Show(x);

}

For input string:

private void input\_button\_Click(object sender, EventArgs e)

{

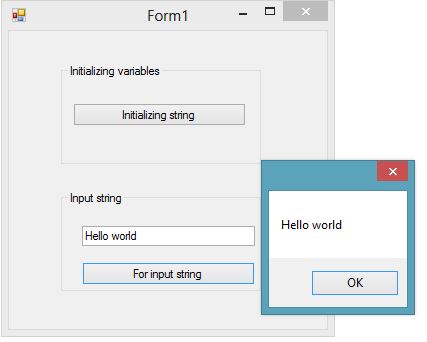
string x = textBox1.Text;

MessageBox.Show(x);

}

Build the solution and run it and You will get the following picture(Fig-9)

**9**



**Task 2: CSharp Statements**

**Practice-1: How to use C# if else statements**

The conditional statement **if.. else** in C# is using for check the condition and making decision based on that condition. The conditional statement examining the data using comparison operators as well as logical operators. **if** statement will evaluate the condition. If it is true, it will execute the statements that follow it. Otherwise, it will execute the statements in else block.

Syntax 1: if statement without else

**if (condition)**

**{**

**statement;**

**}**

Syntax 2 : if statement with else

**if (condition)**

**{**

**statement;**

**}**

**else**

**{**

**statement;**

**}**

If you want to check more than one conditions at the same time , you can use **else if** statement

Syntax 3:

**if (condition)**

**{**

**statement;**

**}**

**else if (condition)**

**{**

**statement;**

**}**

**else**

**{**

**statement;**

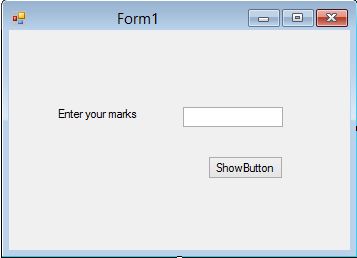
**}**

**Walkthrough 1: Working with conditional statements (if..else)**

**Steps:**

1. Create a desktop application
2. Design the user interface like the following figure(Fig-10)

**10**



1. Double click on the ShowButton
2. Write down the code snippet in the block

private void show\_button\_Click(object sender, EventArgs e)

{

float marks = float.Parse(marksTB.Text);

if (marks>=80)

{

MessageBox.Show("You got A+");

}

else if (marks >= 75)

{

MessageBox.Show("You got A");

}

else if (marks >= 70)

{

MessageBox.Show("You got A-");

}

else if (marks >= 65)

{

MessageBox.Show("You got B+");

}

else if (marks >=60)

{

MessageBox.Show("You got B");

}

else

{

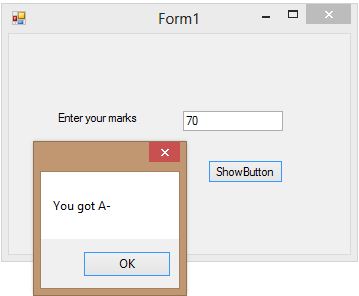
MessageBox.Show("Failed");

}

}

1. Build the solution and run it and you will get the following picture(Fig-11)

**11**



**Practice-2: How to use C# switch case statements**

The C# **switch** statement allows you to choose from many statements based on multiple selections by passing control to one of the case statements within its body. The switch statement executes the case corresponding to the value of the expression.

Syntax:

**switch (expression)**

**{**

**case expression:**

**//your code here**

**statement;**

**break;**

**default:**

**//your code here**

**statement;**

**break;**

**}**

expression : An integral or string type expression.

The break keyword lets the program stop when the desire value is met.

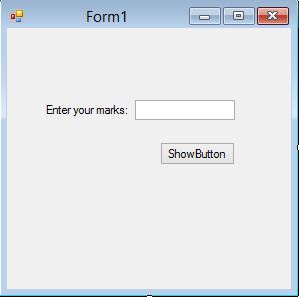
The default keyword lets the program execute the statements that follow it when the desire value is not met.

**Walkthrough 2: Working with switch case statements**

**Steps:**

1. Create a desktop application
2. Design the user interface like the following figure(Fig-12)

**12**



1. Double click on the ShowButton
2. Write down the code snippet in the block

private void show\_button\_Click(object sender, EventArgs e)

{

float marks = float.Parse(marksTB.Text);

int value =(int)marks/10; //Convert float to integer

switch (value)

{

case 3:

MessageBox.Show("You have failed");

break;

case 4:

MessageBox.Show("You have got B");

break;

case 5:

MessageBox.Show("You have got B+");

break;

case 6:

MessageBox.Show("You have got A-");

break;

case 7:

MessageBox.Show("You have got A");

break;

case 8:

MessageBox.Show("You have got A+");

break;

default:

MessageBox.Show("Out of range !!");

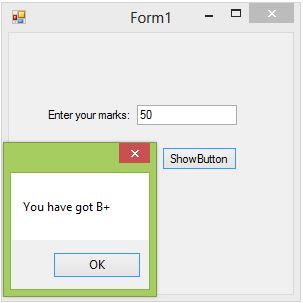
break;

}

}

1. Build the solution and run it and you will get the following picture(Fig-13)

**13**



**Practice-3: How to use C# Iteration Statements**

C# provides several looping mechanisms, which enable you to execute a block of code repeatedly until a certain condition is met. In each case, a statement is executed until a Boolean expression returns true. By using these looping mechanisms, you can avoid typing the same line of code over and over.

1. **For loop:**

For loop is the statements within the code block of a for loop will execute a series of statements as long as a specific condition remains true. A for loop must have 3 basic parameters. It must have a loop initializer value, a loop ending value and also an increment/decrement operator. The code in **for loop** will be executed from the start value until the condition is met. The increment will be added continuously to the start value until the condition is met.

**Syntax:**

for(initialization;condition;increment/decrement)

{

code statement;

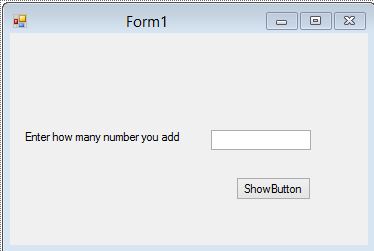
}

**Walkthrough 3: Working with for loop**

**Steps:**

1. Create a desktop application
2. Design the user interface like the following figure(Fig-14)

**14**



1. Double click on the ShowButton
2. Write down the code snippet in the block

private void show\_button\_Click(object sender, EventArgs e)

{

int n = int.Parse(addTB.Text);

int i,sum=0;

for (i = 0; i <= n; i++)

{

sum = sum + i;

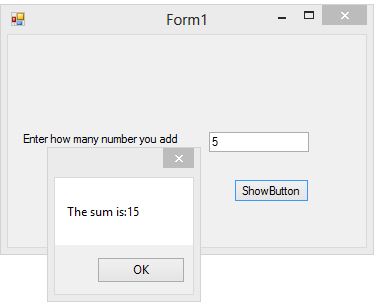
}

MessageBox.Show("The sum is:"+sum);

}

1. Build the solution and run it and you will get the following picture(Fig-15)

**15**



1. **While loop:**

The **while loop** executes code repeatedly until a condition is met. Without a false condition, the statements in while loop are executed infinitely. Therefore, make use you provide a stop point (false condition) where it should stop.

Syntax:

while(condition)

{

statement;

}

Examples:

int count = 1;

while (count < = 4)

{

MessageBox.Show("The value of i is : " + count);

count = count + 1;

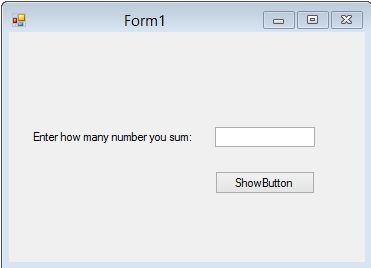
}

**Walkthrough 4: Working with while loop**

**Steps:**

1. Create a desktop application
2. Design the user interface like the following figure(Fig-16)

**16**



1. Double click on the ShowButton
2. Write down the code snippet in the block

private void show\_button\_Click(object sender, EventArgs e)

{

int n = int.Parse(sumTB.Text);

int i=1,sum=0;

while ( i <= n)

{

sum = sum + i;

i++;

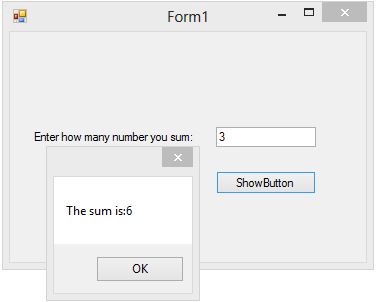
}

MessageBox.Show("The sum is:"+sum);

}

1. Build the solution and run it and you will get the following picture(Fig-17)

**17**



1. **Do-While loop**

Do while loop first executes the whole code block one time and then checks the boolean expression. If the expression is false, this loop quit execution and go to the next statements. Otherwise, it runs until it finds the boolean expression false. The difference between **do..while** and while is that do..while evaluates its expression at the bottom of the loop instead of the top. Therefore, the statements within the do block are always executed at least once.

Example:

int count = 4;

do{

MessageBox.Show(" Loop Executed ");

count++;

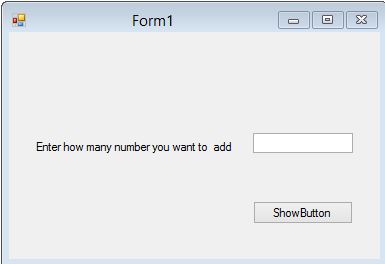
}while (count < =4);

**Walkthrough 5: Working with do-while loop**

**Steps:**

1. Create a desktop application
2. Design the user interface like the following figure(Fig-18)

**18**



1. Double click on the ShowButton
2. Write down the code snippet in the block

private void button1\_Click(object sender, EventArgs e)

{

int n = int.Parse(sumTB.Text);

int i = 1, sum = 0;

do

{

sum = sum + i;

i++;

}

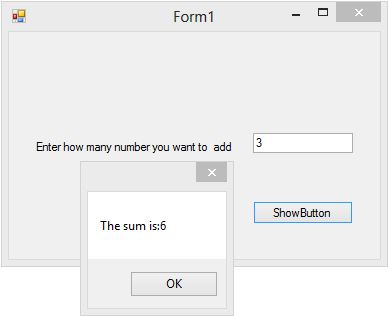
while (i <= n);

MessageBox.Show("The sum is:" + sum);

}

1. Build the solution and run it and you will get the following picture(Fig-19)

**19**

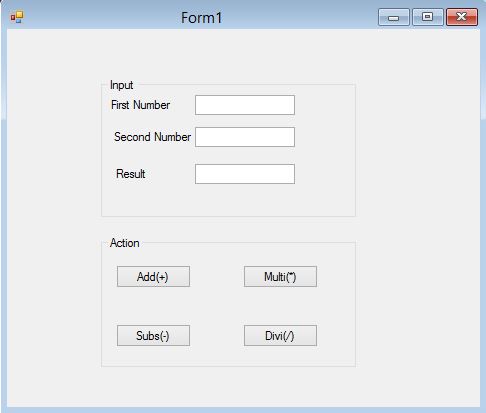


**Task 3: Working with class,object and method**

The first Class that you'll create is a very simple one, and won't be that much use in the real world. But it will show you how Classes are structured, and how Objects are created from Classes.Here we build simple calculator using class,object and method.We will get a concept how to use class,object and method.

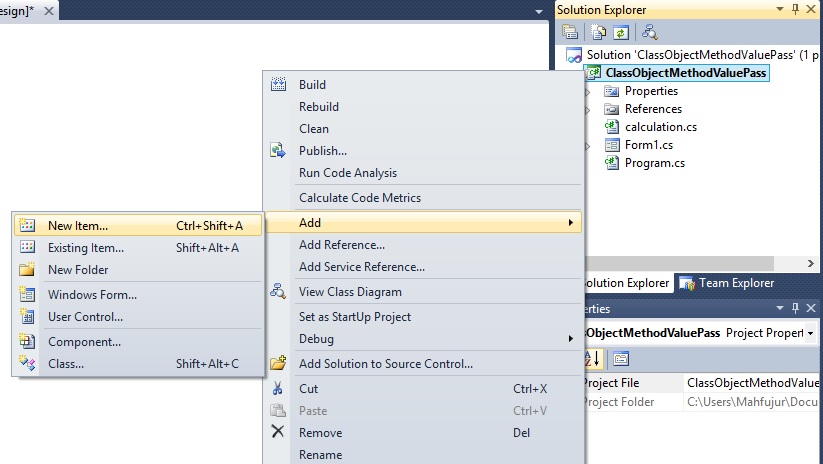
So start a new project, and design the user interface like this(Fig-20)

**200**



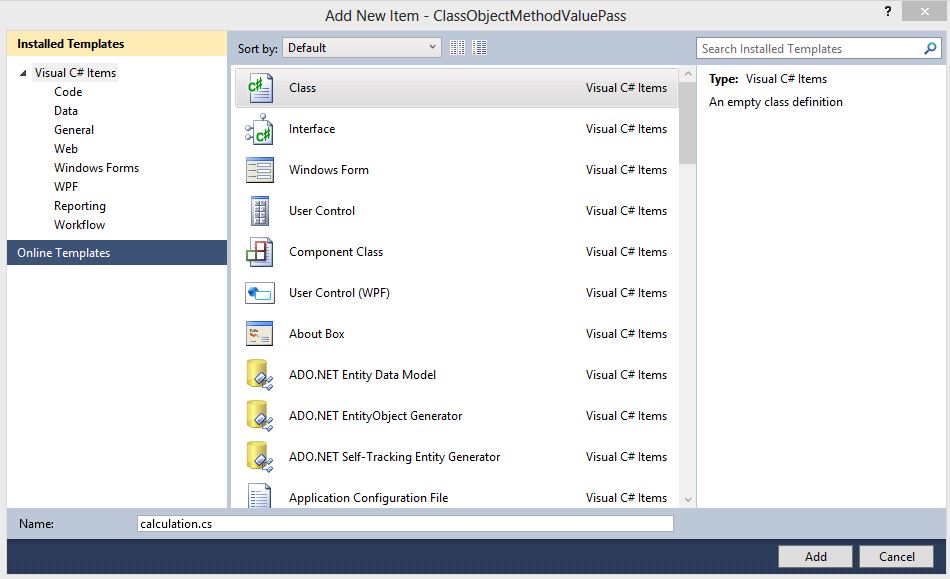
To add a new class to your programme,right click the project name at solution explorer.(Fig-21)

**210**



Then click the New Item and show the following figure(Fig-22)

**220**



Tips: Class name convension

After creating the class name we write the following code for only add.

class calculation

{

public double add(double firstnumber, double secondnumber)

{

double answer = firstnumber + secondnumber;

return answer;

}

}

**Tips:** Here calculation is a class name. add() is a method for adding.

Here we work with only Add Buttton(Fig-20).Double Click on the Add Button.write the following code:

private void add\_button\_Click(object sender, EventArgs e)

{

calculation obj = new calculation();

double result = obj.add(double.Parse(firstTB.Text), double.Parse(secondTB.Text));

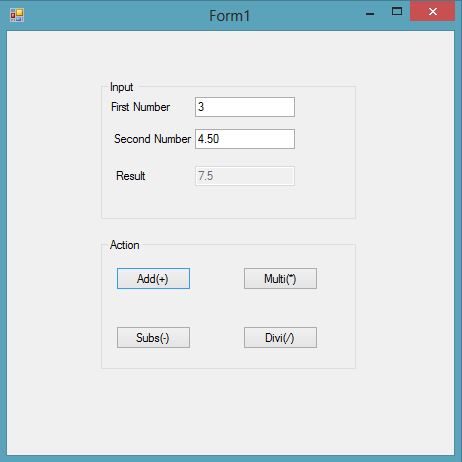
resultTB.Text = result.ToString();

}

**Tips:** Here calculaton is a class name and obj is a object of calculation class.Here we call a method add() using object obj which create from calculation class and passing parameter to the add() method.

Now we build the project and run the we get the following figure(Fig-23)

**230**



Exercise for you : You complete this project for other Button that means Multi(\*),Subs(-),Div(/). Create method for multiplication,substraction and division.Call this method using object.