LAB NO.: 1 **Date:**

BASIC VC# PROGRAMS - I

Objectives:

- To use different tools using visual studio and code them in C#.

Introduction:

Description of the tools used:

Text Box: A Text Box control is used to display, or accept as input, a single line of text. This control has additional functionality that is not found in the standard Windows text box control, including multiline editing and password character masking. A text box object is used to display text on a form or to get user input while a C# program is running. In a text box, a user can type data or paste it into the control from the clipboard. For displaying a text in a Text Box control, you can code like this

textBox1.Text = "Hello Manipal!";

You can also collect the input value from a Text Box control to a variable like this way

string var;

var = textBox1.Text;

Label: Labels are one of the most frequently used C# control. We can use the Label control to display text in a set location on the page. Label controls can also be used to add descriptive text to a Form to provide the user with helpful information. The Label class is defined in the System. Windows.Forms namespace.

Add a Label control to the form - Click Label in the Toolbox and drag it over the forms Designer and drop it in the desired location. If you want to change the display text of the Label, you have to set a new text to the Text property of Label.

label1.Text = "This is my first Label";

In addition to displaying text, the Label control can also display an image using the Image property, or a combination of the Image Index and Image List properties.

label1.Image = Image.FromFile("C:\\testimage.jpg");

Button: A button is a control, which is an interactive component that enables users to communicate with an application. The Button class inherits directly from the Button Base class. A Button can be clicked by using the mouse, ENTER key, or SPACEBAR if

the button has focus. When you want to change display text of the Button, you can change the Text property of the button.

button1.Text = "Click Here";

Similarly if you want to load an Image to a Button control, you can code like this

button1.Image = Image.FromFile("C:\\testimage.jpg");

Radio Button: A radio button or option button enables the user to select a single option from a group of choices when paired with other Radio Button controls. When a user clicks on a radio button, it becomes checked, and all other radio buttons with same group become unchecked. The Radio Button control can display text, an image, or both. Use the Checked property to get or set the state of a Radio Button.

radioButton1.Checked = true;

The radio button and the check box are used for different functions. Use a radio button when you want the user to choose only one option. When you want the user to choose all appropriate options, use a check box. Like check boxes, radio buttons support a Checked property that indicates whether the radio button is selected.

Check Box: Check Boxes allow the user to make multiple selections from a number of options. Check Box is used, to give the user, an option, such as true/false or yes/no. You can click a check box to select it and click it again to deselect it. The Check Box control can display an image or text or both. Usually Check Box comes with a caption, which you can set in the Text property.

checkBox1.Text = "Net-informations.com";

You can use the Check Box control Three State property to direct the control to return the Checked, Unchecked, and Indeterminate values. You need to set the check box's Three State property to True to indicate that you want it to support three states.

checkBox1.ThreeState = true;

The radio button and the check box are used for different functions. Use a radio button when you want the user to choose only one option. When you want the user to choose all appropriate options, use a check box.

List Box: The List Box control enables you to display a list of items to the user that the user can select by clicking. In addition to display and selection functionality, the List Box also provides features that enable you to efficiently add items to the List Box and to find text within the items of the list. You can use the Add or Insert method to add items to a list box. The Add method adds new items at the end of an unsorted list box.

listBox1.Items.Add("Sunday");

If you want to retrieve a single selected item to a variable, you can code like this:

string var; var = listBox1.Text;

The Selection Mode property determines how many items in the list can be selected at a time. A List Box control can provide single or multiple selections using the Selection Mode property. If you change the selection mode property to multiple select, then you will retrieve a collection of items from ListBox1.SelectedItems property.

listBox1.SelectionMode = SelectionMode.MultiSimple;

Combo Box: A Combo Box displays a text box combined with a List Box, which enables the user to select items from the list or enter a new value. The user can type a value in the text field or click the button to display a drop down list. You can add individual objects with the Add method. You can delete items with the Remove method or clear the entire list with the Clear method.

To add the items into the drop down list:

```
comboBox1.Items.Add("Sunday");
comboBox1.Items.Add("Monday");
comboBox1.Items.Add("Tuesday");
```

Picture Box: The Windows Forms Picture Box control is used to display images in bitmap, GIF, icon, or JPEG formats**.** You can set the Image property to the Image you want to display, either at design time or at run time. You can programmatically change the image displayed in a picture box, which is particularly useful when you use a single form to display different pieces of information.

pictureBox1.Image = Image.FromFile("c:\\testImage.jpg");

The Size Mode property, which is set to values in the Picture Box Size Mode enumeration, controls the clipping and positioning of the image in the display area.

pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;

There are five different Picture Box Size Mode available under Picture Box control.

Auto Size - Sizes the picture box to the image.

Center Image - Centers the image in the picture box.

Normal - Places the upper-left corner of the image at upper left in the picture box.

Stretch Image - Allows you to stretch the image in code.

The Picture Box is not a selectable control, which means that it cannot receive input focus. The following C# program shows how to load a picture from a file and display it in stretch mode.

Progress Bar: A progress bar is a control that an application can use to indicate the progress of a lengthy operation such as calculating a complex result, downloading a large file from the web etc. Progress Bar controls are used whenever an operation takes more than a short period of time. The Maximum and Minimum properties define the range of values to represent the progress of a task.

Minimum : Sets the lower value for the range of valid values for progress.
Maximum : Sets the upper value for the range of valid values for progress.
Value : This property obtains or sets the current level of progress.

By default, Minimum and Maximum are set to 0 and 100. As the task proceeds, the Progress Bar fills in from the left to the right to delay the program briefly so that you can view changes in the progress bar clearly.

C# DateTimePicker Control: The Date Time Picker control allows you to display and collect date and time from the user with a specified format. The Date Time Picker control has two parts, a label that displays the selected date and a popup calendar that allows users to select a new date. The most important property of the Date Time Picker is the Value property, which holds the selected date and time.

```
dateTimePicker1.Value = DateTime.Today;
```

The Value property contains the current date and time the control is set to. You can use the Text property or the appropriate member of Value to get the date and time value.

```
DateTime iDate;
iDate = dateTimePicker1.Value;
```

The control can display one of several styles, depending on its property values. The values can be displayed in four formats, which are set by the Format property: Long, Short, Time, or Custom.

```
dateTimePicker1.Format = DateTimePickerFormat.Short;
```

Message Box: Displays a message window, also known as a dialog box, which presents a message to the user. It is a modal window, blocking other actions in the application until the user closes it. A Message Box can contain text, buttons, and symbols that inform and instruct the user.

Tree view: The Tree View control contains a hierarchy of Tree View Item controls. It provides a way to display information in a hierarchical structure by using collapsible nodes. The top level in a tree view are root nodes that can be expanded or collapsed if the nodes have child nodes. You can explicitly define the Tree View content or a data source can provide the content. The user can expand the Tree Node by clicking the plus sign (+) button, if one is displayed next to the Tree Node, or you can expand the Tree Node by calling the TreeNode. Expand method. You can also navigate through tree views with various properties: First Node, Last Node, Next Node, Prev Node, Next Visible Node, Prev Visible Node.

The full path method of tree view control provides the path from root node to the selected node.

treeView1.SelectedNode.FullPath.ToString();

Tree nodes can optionally display check boxes. To display the check boxes, set the Check Boxes property of the Tree View to true.

treeView1.CheckBoxes = true;

Example: Design a simple calculator using C#.

Open Visual studio. Click on **File->New->Project**. Select **windows form Application.** Change the **name of the project** in the **Name field** below and choose the **path** where the project has to be stored. Then Click on **OK** which will redirect to the form with a name **Form1** which acts as a panel where the user interface will be created. To do so, Tool Box is required which is displayed to the left side. (If it is not visible then go to **View->ToolBox.**)

From the **ToolBox** select **Button** control and drop it in the Form. Then in the **Properties window** (which is on the bottom right) go to **Text field** and change value- **textBox1** currently - to **1**. Text field in the properties window depicts the display name for the tool control.

Change the **Name** field of the button in the properties window to **cmd1**. This name is the name taken into the code. Similarly drag and drop the buttons for the rest of the numbers and the operators. Rename the text field and the Name field of the respective buttons according to the requirements. Now, drag and drop the **text field** from the tool box. Now we need to code the buttons so that we can make it usable. For that we will require few variable which are to be declared as given below:

public partial class Form1 : Form

```
{
string input = string.Empty; //to read the input when clicked
string Op1 = string.Empty; //First operand
string Op2 = string.Empty; //Second operand
char Operator; //Operator
double res = 0.0; //Final result
public Form1()
{
InitializeComponent();
}
```

Double click on button1 which will redirect you to **Form1.cs** from **Form1.cs[Design].** As you can see it has created a function with name **cmd1_Click** (cmd1 is the name which you had given in the name field of the properties window). The default event ofthe button is Click, that is why the word Click is attached withcmd1, which means **"On the event Click on the button"** the executable statements under cmd1_Click function should be executed. Type the following code under the created function

```
private void cmd1_Click(object sender, Event Args e)
{
this.textBox1.Text = string.Empty;
input = input + "1";
this.textBox1.Text += input;
}
```

Similarly, code for button 2 to button 9 by changing the values of input variable asshown in the code below:

```
private void cmd2_Click(object sender, EventArgs e)
{
this.textBox1.Text = string.Empty;
input += "2";
this.textBox1.Text += input;
}
```

Now it is time to code the operators. Similar to the previous step, double click on the operator button, maybe'+', which will be redirected to the function in Form1.cs. Type the following code under it.

```
private void add_Click(object sender, EventArgs e)
{
Op1 = input;
Operator = '+';
input = string.Empty;
}
```

When the operator button is clicked, it means that until then whatever the numbers are being pressed should be considered as the first operand. That's why the statement,**Op1=input.** And then input variable is cleared so that it starts reading the second operand. Similarly code for the rest of the operators.

Once the numbers and operators are coded, we need result. Double click on the equal to button and type the following code.

```
private void Ans_Click_Click(object sender, EventArgs e)
{
Op2 = input;
double num1, num2;
double.TryParse(Op1, out num1);
double.TryParse(Op2, out num2);
if (Operator == '+')
{
res = num1 + num2;
this.textBox1.Text = res.ToString();
}
else if (Operator == '-')
{
res = num1 - num2;
textBox1.Text = res.ToString();
}
else if (Operator == '*')
{
res = num1 * num2;
textBox1.Text = res.ToString();
}
```

```
else if (Operator == ’/’)
{
if (num2 != 0)
{
res = num1 / num2; textBox1.Text =
res.ToString();
}
else
{
textBox1.Text = "DIV/Zero!";
}
}
 input = string.Empty;
}
```

Lab exercises:

1. Design a scientific calculator using C#. Have at least 4 different kinds of scientific functions.
2. Develop a simple form to enter necessary details for online registration of students. Also, display the message (confirm or not) along with details entered on submit. Perform necessary validation. Use Text Box, Radio button, Combo box, Check box, Calendar, Label, Button, and Message Box.