AST21105 Object-Oriented Programming & Design

Lab 9 – GUI Programming using C#

A. Submission Details

No submission is required for this lab.

Important: You are highly recommended to spend more time to practice the techniques that we introduced you during the lab at home. Since this is important for you to do a good job on the programming assignment for this course.

B. Objective

GUI-based programs are undoubtedly the most useful and important applications that we need nowadays. Programs with no Graphical User Interface still serve the needs of users, however, these programs are generally less user-friendly and therefore not easy to use compared to those with good GUIs. All the programs that we introduced you so far are all console-based (i.e. with text UI only) and thus they are not very attractive. In order to give you essential experience on building up GUIs, a full demo of procedure involved in doing so will be shown in a step-by-step manner.

The objectives of this lab are:

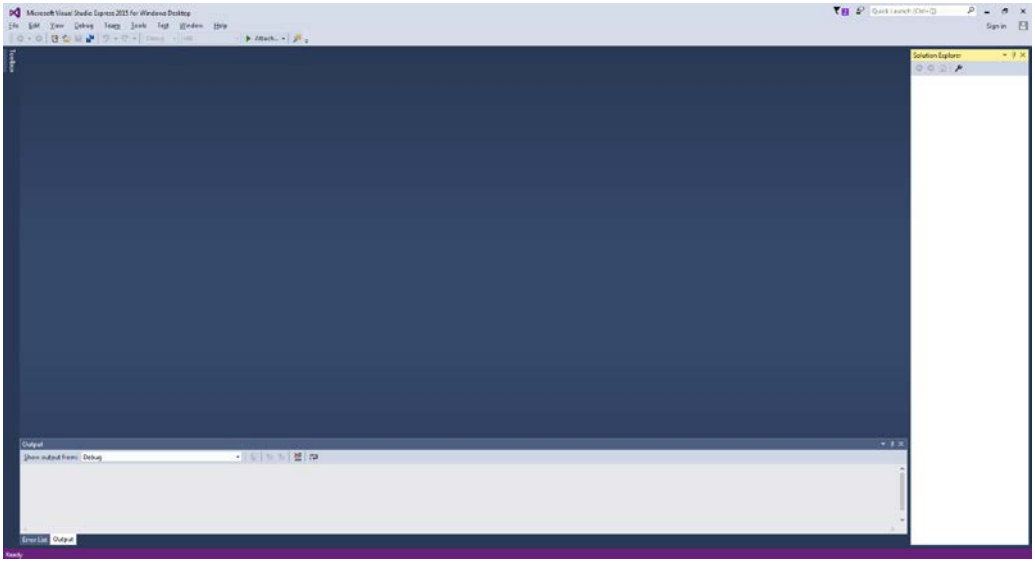
1. To help you get familiar with the procedure on creating Graphical User Interface using C#
2. Use C# GUI to call user-defined type's member functions constructed using C++.

C. Demonstration

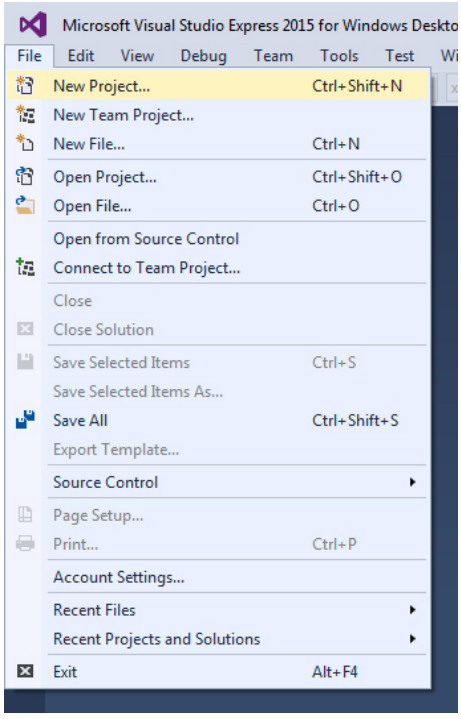
The purpose of this part of lab is to demonstrate you how to use Visual Studio development environment to write simple GUI applications. This lab focuses on giving you the idea of an event-driven architecture and helps you to understand how to create simple user interface using Visual Studio Form Designer and to get the concept of an event as a user action that triggers execution of a named code segment.

Procedure for creating Graphical User Interface (GUI) using Visual C#:

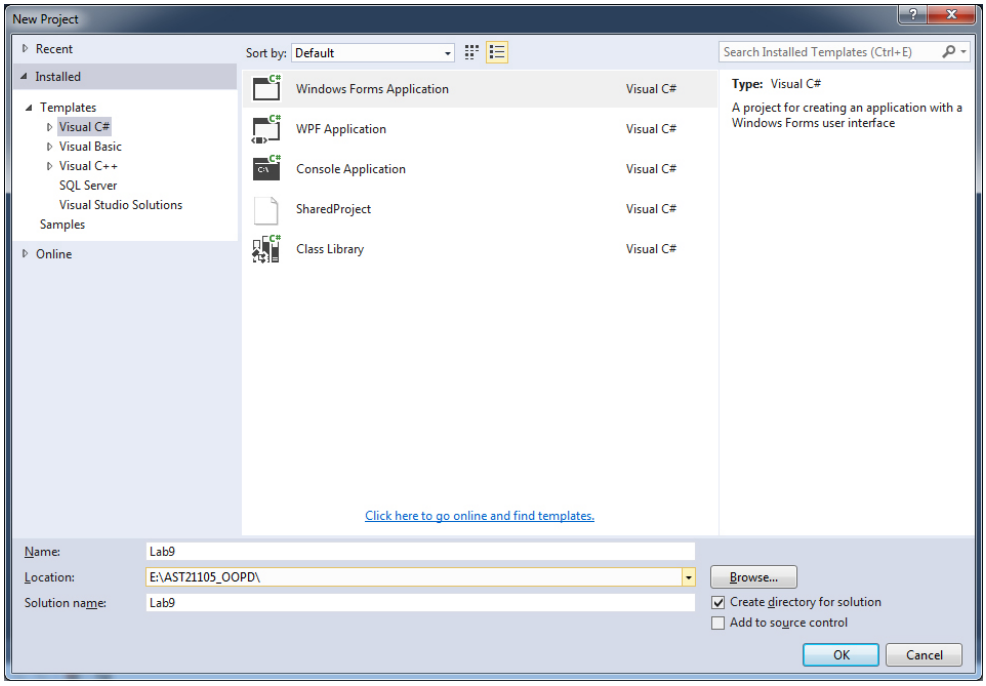
1. Start up Visual Studio and you should see something as follows:



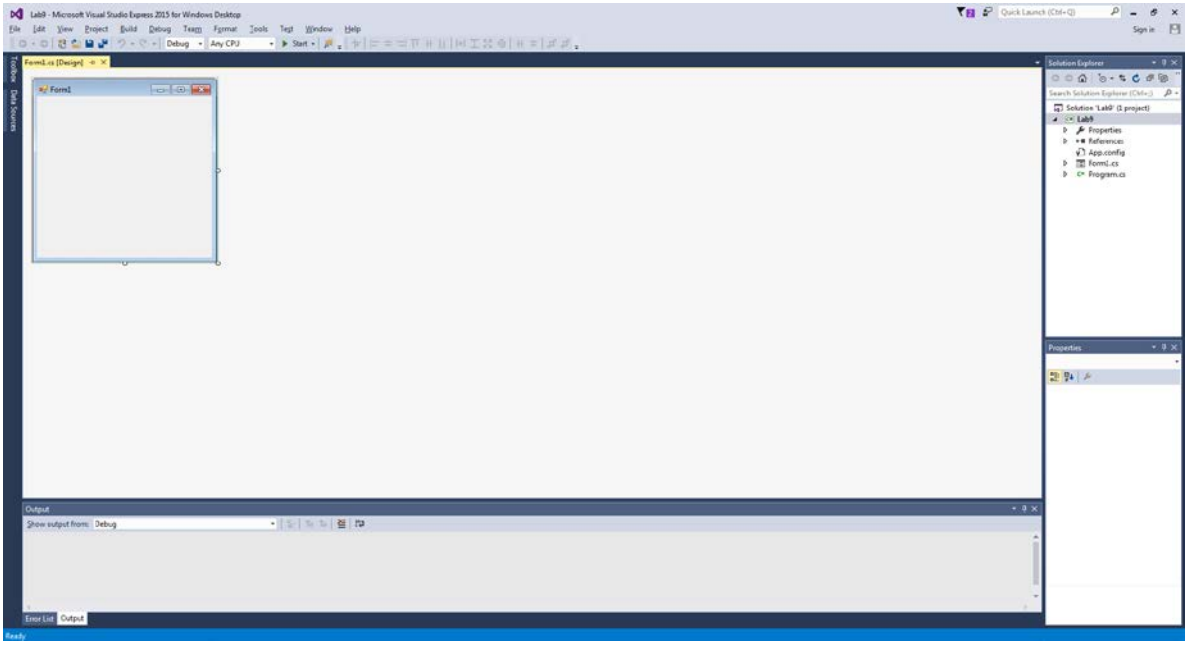
2. Create a new project by clicking on the “File” > “New Project”.



3. Create a Visual C# “Window Forms Application” and name it as Lab9. Then click “OK”.



After the application project is created, the following should be shown in screen.



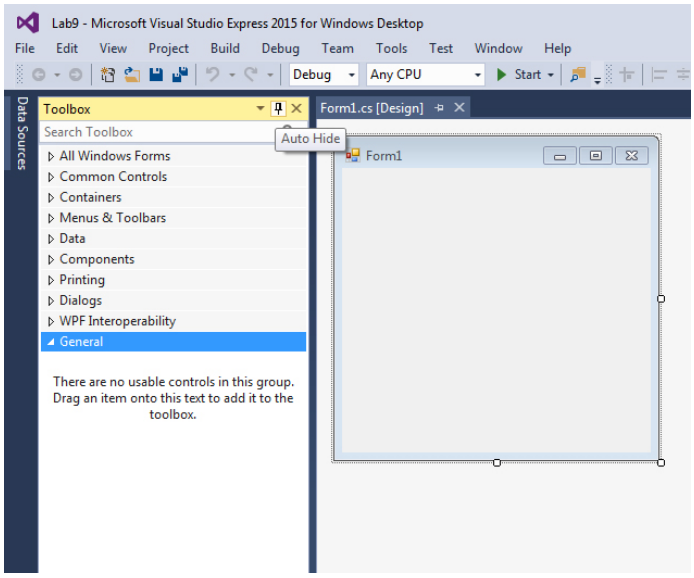
There may be some differences in configuration (such as placement of the tool bars and boxes). A new window application will start with an ***empty form called Form1***. This form represents the user interface that will be shown when the application is first run. You can interact with it to change its appearance.

4. Adjust the size of the Window Form according to your needs.

5. In order for an application to be useful, it will need to have ***widgets*** for the user to interact with. Widgets are items like ***buttons, scrollbars icons and text entry boxes***.

   To add widgets to the application, the "Toolbox" window must be visible. If there is "Toolbox" item on the left of the display hovering the mouse pointer over, it should make the toolbox visible.

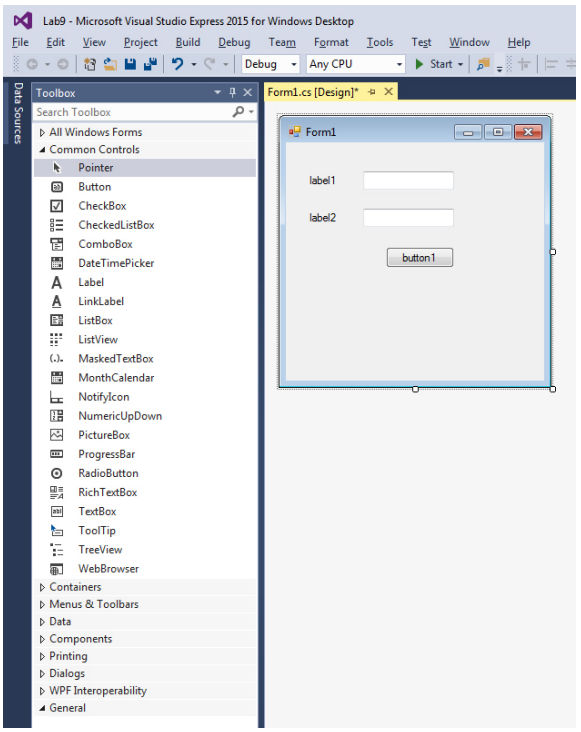
   (Note: You can then prevent it from auto-hiding by ***clicking on the "pin" icon***. If you cannot find the toolbox at all, you can show it by selecting "Toolbox" from the "View" menu.)



6. The toolbox has a number of categories inside it. All of the widgets used in this lab are under the "Common Controls" category. Click the "+" icon next to the category to view its content.

   Suppose we would like to build up a simple UI as follows:

   Click on the following items one by one to add appropriate controls to the form.



   (a) Select "Label" from the "Common Controls" and click on the form to add a label.

   (The label will be created with name "label1" and with default properties. You can change all the properties using the "Properties" window.)

   (b) Select "Label" again to add another label to the form (default name is "label2").

   (c) Select "Textbox" widget and add a textbox to the form (default name is "text1").

   (d) Select "Textbox" again to add another textbox to the form (default name is "text2").

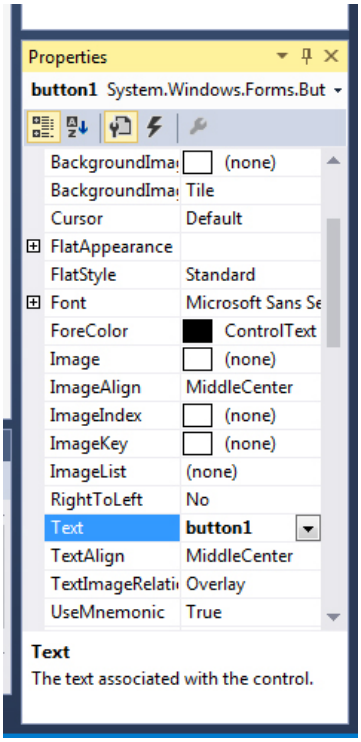
   (e) Select "Button" to add a button to the form (default name is button1).

   Note: Every widget that is part of the user interface has properties. These properties can be changed during the design of the application and many of them can be changed at runtime by the program code. One property that ***cannot be changed at runtime*** is the ***"(Name)"*** property. This is the name by which the widget is internally known. This name is not seen by the user, but will be used by you to refer to the widget in your program code.

The properties of a widget can be modified using the "***Properties***" window near the bottom right of the Visual Studio UI (user interface).

If the properties window is not shown, then you can make it visible by selecting the "Properties Window" item on the view menu. There are lots of properties so you will have to scroll to find them. It may help to resize the properties window to be large by dragging its top edge.

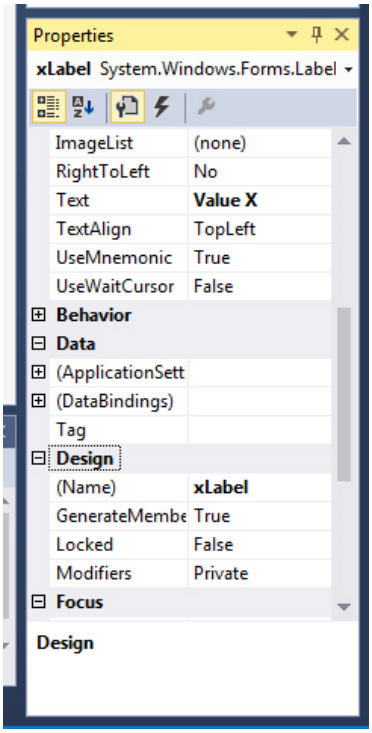
7. Click on the **button** that you created and you should be able to see the following properties window.



   When modifying properties, the widget that is currently selected in the form layout editor (the one gets highlighted) is the widget that you're modifying the properties of.

   Change the properties as shown below in the property list.
   a) Scroll to find the (Name) property, change the button to be named "calculateButton".
   b) Scroll to find the (Text) property, change the text of button to "Calculate".

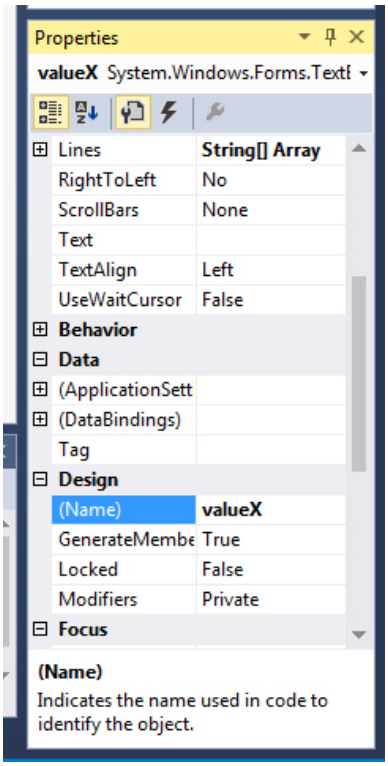
8. Click on the first label that you created earlier and you should be able to see the following properties window.



   For the first label, change the properties as shown below in the property list.
   a) Scroll to find the (Name) property, change the first label to be named "xLabel".
   b) Scroll to find the (Text) property, change the text of label to "Value X".

   For the second label, change the properties as shown below in the property list.
   a) Scroll to find the (Name) property, change the second label to be named "yLabel".
   b) Scroll to find the (Text) property, change the text of label to "Value Y".

9. Click on the first textbox that you created and you should be able to see the following properties window.



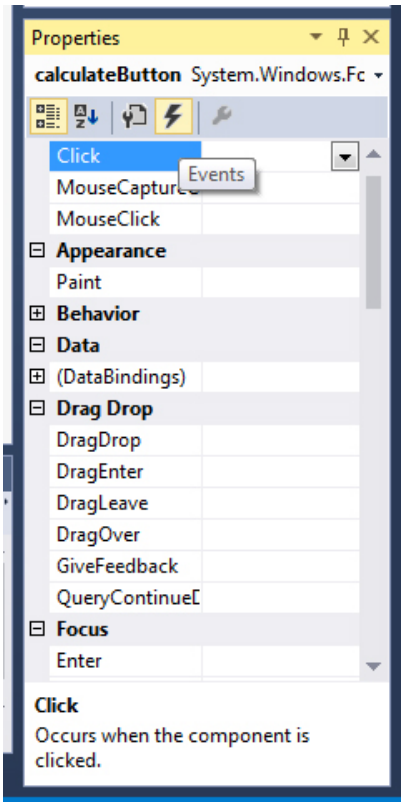
For the first textbox, change the properties as shown below in the property list.

- Scroll to find the (Name) property, change the first textbox to be named "valueX".

For the second label, change the properties as shown below in the property list.

- Scroll to find the (Name) property, change the second textbox to be named "valueY".

10. The basis of event-drive GUI programming is that you write ***small pieces of code*** that are executed in response to event caused by the user. When an event occurs on a widget, you can invoke (call) ***a function / method (an event handler)*** to perform a specific task.



In addition to its list of properties, each widget in the user interface has a number of events that can be called. These events are invoked in response to user action.

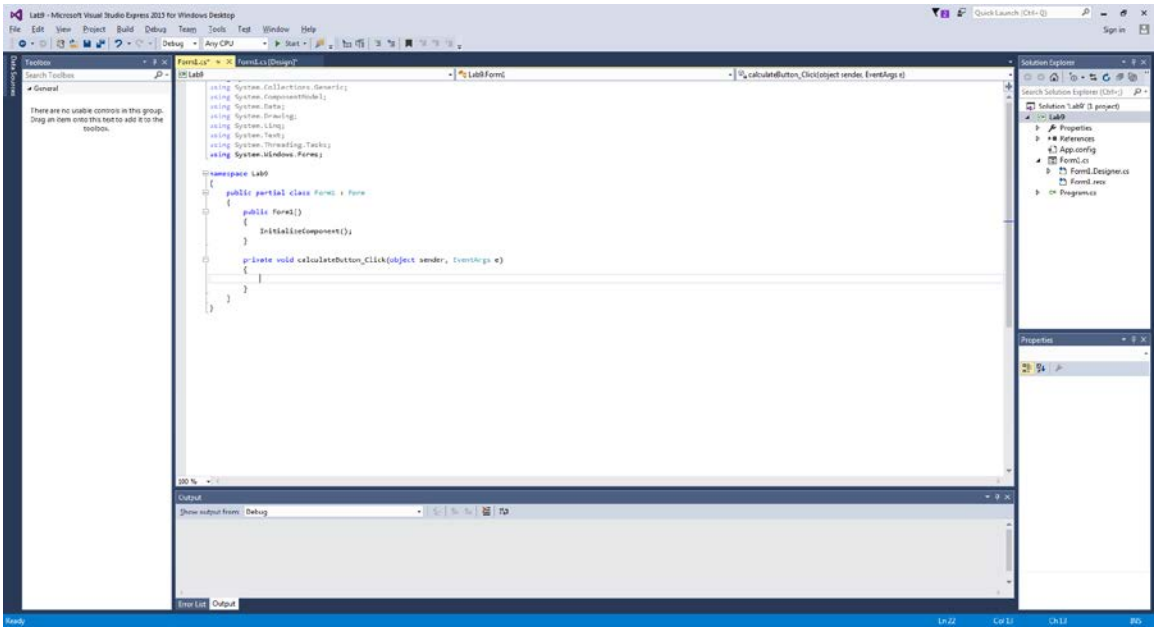
For example, if the user clicks on a button, the "click" event occurs. The list of events available for each widget is available in the properties window. Near the top of the properties window, there will be a lightning bolt icon. If you click this icon, the properties window will switch from displaying the properties of a widget to the events that widget is capable of generating.

To create an event handler (i.e. a function to handle the event) for a widget, first select the widget from the form designer, then double click on the event in the event list.

For example, to create a "Click" event handler for the button that show a box saying "The button is clicked", we do the following:

a) Select the button
b) Find the ***"Click" event*** from the event list of button and ***double click on it***.

At this point, the user interface will change. A new view will open – the code view. This contains the code of your program expressed as a series of event handlers.



A new event handler will have just been created called “calculateButton_Click”. The name of the event handler reflects the widget on which the event has occurred (calculateButton) and the name of the event (Click).

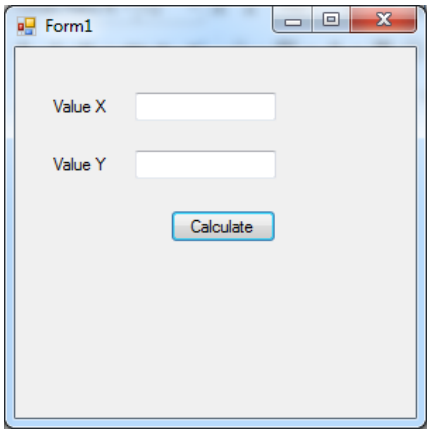
The event handler has a body that begins with { and ends with }. Between these brackets is where you put statements of program code that will be executed when the given event occurs.

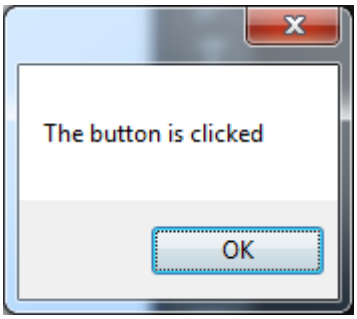
11. Insert program code to the event handler as follows:

```
private void calculateButton_Click(object sender, EventArgs e)
{
    MessageBox.Show("The button is clicked");
}
```

12. Execute the program by clicking on the green “play” button in the UI. If you click this now, then the program should compile and run. If everything works out then you should see your application on the screen.

You can click the “Calculate” button to run your code to show up the message box.





13. Close the program and return to the development environment.

When you created the event handler, Visual Studio automatically switched you to your program. You can get back to the user interface design view by clicking the “Form1.cs [Design]” tab at the top of the code area.

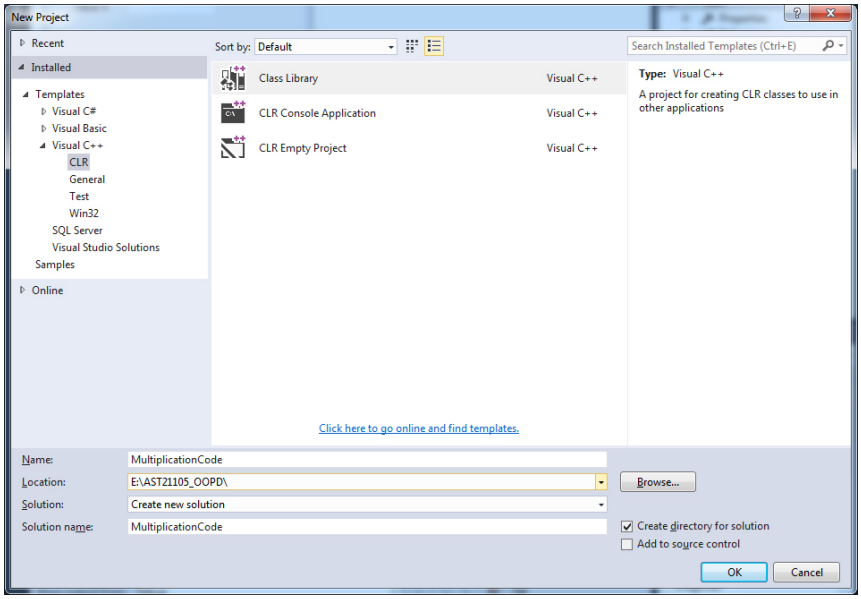
Make sure that the form designer is visible again (i.e. click the “[Design]” tab).

14. Save the project by choosing “Save All” from the file menu.

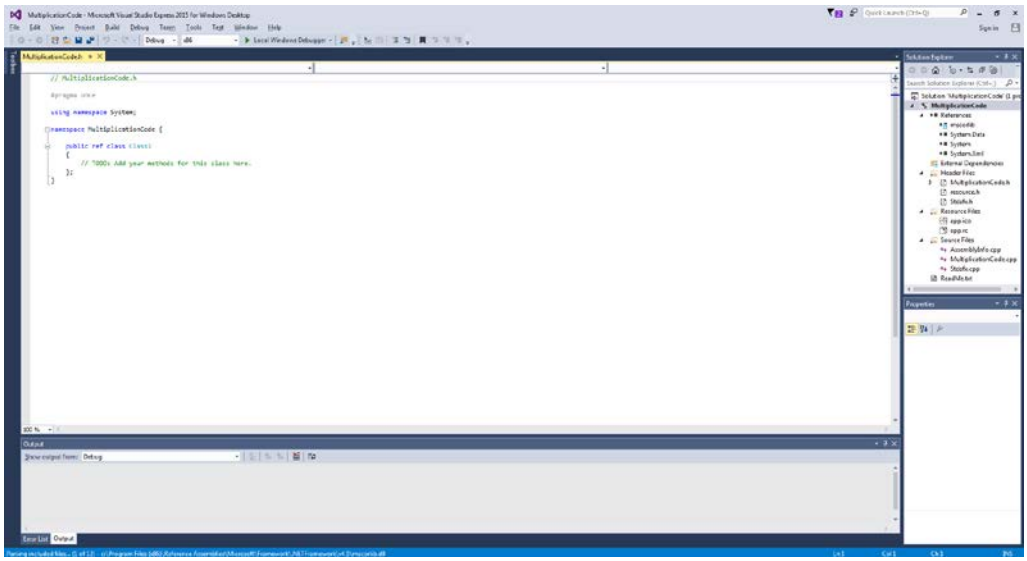
Procedure for creating a user-defined type using C++ and compile it into DLL:

1. Create a new project clicking on the “File” > “New Project”.

2. Create a Visual C++ “CLR (Class Library)” project and name it as “MultiplicationCode”. Then click “OK”.



After the project is created, the following screen should be shown.



3. Change the name of predefined class. The default name is “Class1”. Now, change it to “MathsOperations”.

4. Add the following member function declaration to MultiplicationCode.h.

```
public ref class MathsOperations
{
    public:
        int multiply(int x, int y);
};
```

5. Add the following member function to the class.

```
using namespace MultiplicationCode;

int MathsOperations::multiply(int x, int y)
{
    return x * y;
}
```

6. Compile the code into DLL by clicking “Build” > “Build Solution”.

Procedure for linking C++ code to C# Graphical User Interface:

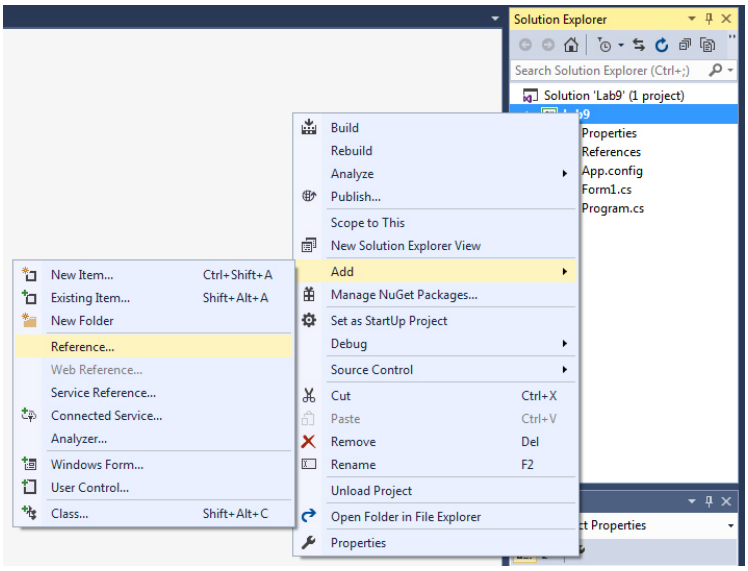
In order to allow the Graphical User Interface created using C# to call the member function defined using C++, the following must be done.

1. Open the C# project. This could be simply done by clicking on the Lab9.sln file.

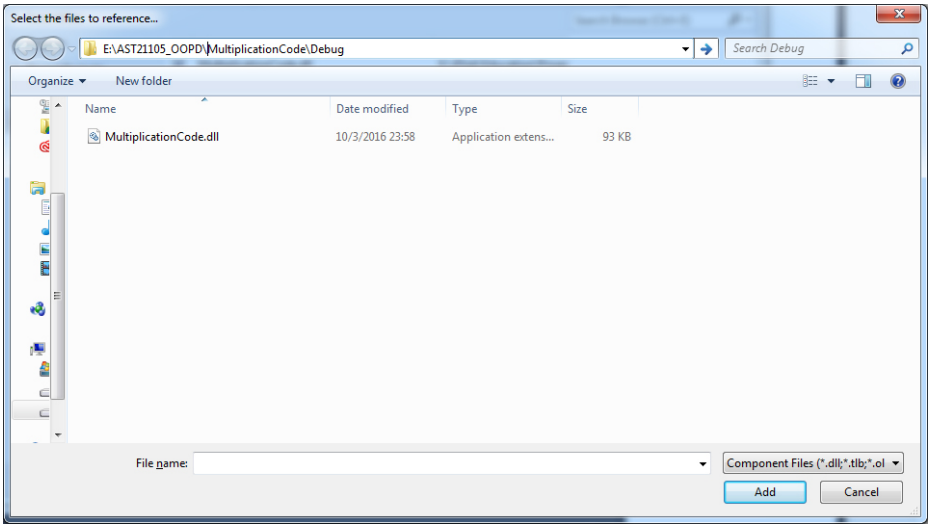
2. Add a reference to the DLL that was created from building the C++ project.

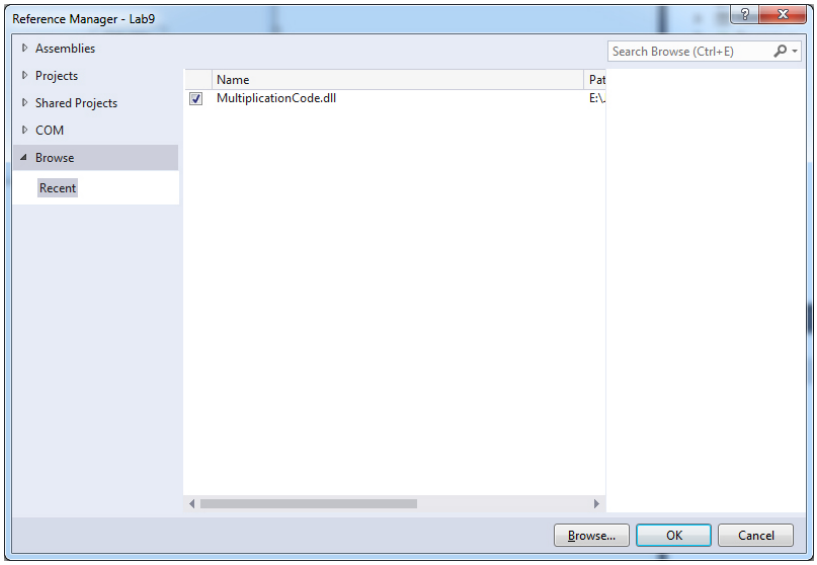
   Perform

   a) Right click on the ***Lab9 item*** at the solution explorer and click “***Add Reference***”.



b) Click on “Browse” button and browse the MultiplicationCode.dll file. Click “Add” and then “OK”.





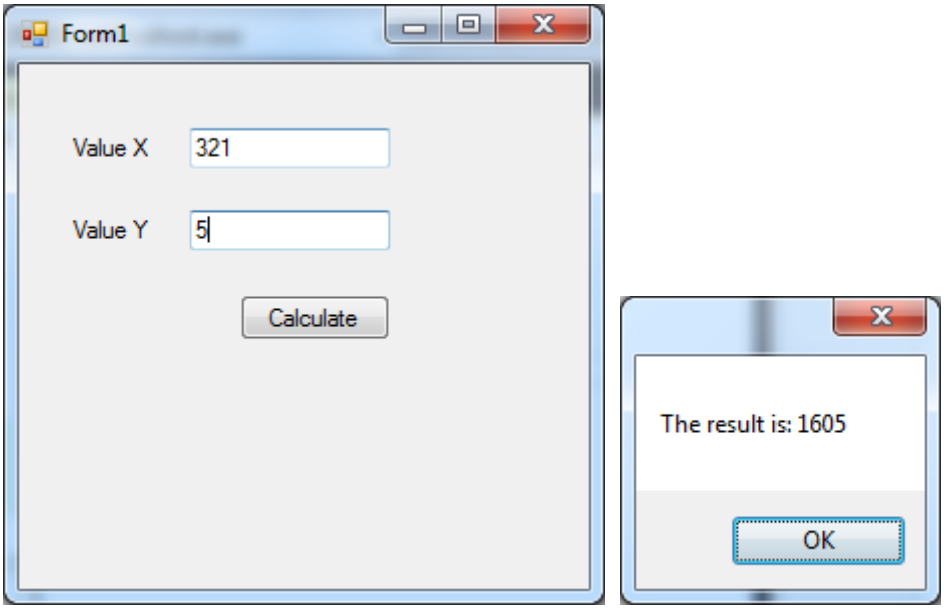
3. Add the line of code below at the very beginning of the file Form1.cs.

```
using MultiplicationCode;
```

4. Change the “calculateButton_Click” code as follows:

```
private void calculateButton_Click(object sender, EventArgs e)
{
    // Create an object of type MathsOperations defined in C++
    MathsOperations mOper = new MathsOperations();
    // Obtain the value in textbox "valueX" and
    // change from string to int
    int x = Int32.Parse(valueX.Text);
    // Obtain the value in textbox "valueY" and
    // change from string to int
    int y = Int32.Parse(valueY.Text);
    // Call the member function "multiply" and obtain the result
    int result = mOper.multiply(x, y);
    // Show the result by a popped up window
    MessageBox.Show("The result is: " + result.ToString());
}
```

5. Run the program and test out the result.



-End-