

1 Lecture 1: C# and Windows Forms Fundamentals

Course: Scientific Visual Programming

Platform: C# Windows Forms Application

Recommended duration: 90 minutes

1.1 Learning objectives

By the end of this lecture, students should be able to explain the structure of a C# Windows Forms project, create a form, distinguish controls from containers, declare variables, and write a simple event handler.

1.2 1. What is visual programming?

Visual programming combines a graphical design environment with source code. The programmer designs a user interface by placing controls on a form and then writes code that responds to user actions. In Windows Forms, a form is a desktop window, while controls such as buttons, labels, text boxes, and list boxes provide interaction.

The visual designer accelerates interface construction, but it does not replace programming concepts. A reliable application still requires clear names, appropriate data types, validation, and separation between user-interface logic and application logic.

1.3 2. Windows Forms project structure

A modern Windows Forms project usually contains a project file, a startup class, and one or more form classes. A form commonly has two important files: a code file such as `MainForm.cs` and a designer file such as `MainForm.Designer.cs`. The designer file contains generated initialization code. Students should normally modify controls through the designer or controlled code rather than editing generated sections manually.

A form class inherits from `Form`:

```
public partial class MainForm : Form
{
    public MainForm()
    {
        InitializeComponent();
    }
}
```

`InitializeComponent` creates controls, assigns properties, and connects event handlers. The constructor runs when the form is created.

1.4 3. Common controls and properties

A `Label` displays text. A `TextBox` accepts text input. A `Button` triggers an action. A `CheckBox` represents an independent yes/no choice. A `ComboBox` provides a drop-down selection. A `ListBox` displays a list of items. A `Panel`, `GroupBox`, or `TableLayoutPanel` organizes other controls.

Important properties include `Name`, `Text`, `Enabled`, `Visible`, `Size`, `Location`, `BackColor`, `ForeColor`, and `Dock`. Use descriptive names such as `txtStudentName`, `btnCalculate`, and `lblResult`. The `Name` property is used in code; the `Text` property is what the user sees.

1.5 4. Events and event handlers

An **event** is a notification that something happened. The Click event occurs when a button is activated. The Load event occurs when a form is loaded. The TextChanged event occurs when text changes. An event handler is a method that responds to an event.

```
private void btnHello_Click(object sender, EventArgs e)
{
    string name = txtName.Text.Trim();
    lblMessage.Text = string.IsNullOrEmpty(name)
        ? "Please enter your name."
        : $"Hello, {name}!";
}
```

The sender parameter identifies the object that raised the event. EventArgs carries event information; specialized events use types such as KeyEventArgs or MouseEventArgs.

1.6 5. Variables, types, and conversion

C# is statically typed. Common types include int, double, decimal, bool, char, and string. Text read from a TextBox is a string, so numeric input must be validated and converted before calculation.

```
if (decimal.TryParse(txtPrice.Text, out decimal price))
{
    decimal total = price * 1.15m;
    lblTotal.Text = total.ToString("C2");
}
else
{
    MessageBox.Show("Enter a valid price.");
}
```

Prefer TryParse to direct parsing when the input comes from a user. It prevents invalid input from producing an unhandled exception.

1.7 6. A first scientific calculator

A simple calculator form may contain two text boxes, an operation combo box, a calculate button, and a result label. The button handler reads the values, validates them, performs the selected operation, and updates the result. The user interface should show understandable messages for missing or invalid input. The calculation itself can be moved to a separate method:

```
private decimal Add(decimal first, decimal second)
{
    return first + second;
}
```

This makes the logic easier to test and keeps the event handler focused on interaction.

1.8 7. Good naming and layout practice

Keep a consistent naming convention. Prefixes such as `txt`, `btn`, `lbl`, and `cmb` can help beginners recognize control types, although teams may prefer descriptive names without prefixes. Set the tab order so that keyboard users can move through the form logically. Use layout containers instead of placing every control at a fixed coordinate when the interface must resize.

1.9 8. Practice activity

Create a form that calculates the area of a rectangle. Add text boxes for length and width, a button named `btnCalculate`, and a label named `lblArea`. Validate that both values are positive numbers. Display the result with two decimal places and show a clear message when input is invalid.

1.10 Summary

Windows Forms provides a visual way to build desktop interfaces, but the application still depends on core C# concepts. Controls expose properties and events. Event handlers should validate user input, call focused methods, and update the interface with clear feedback.

1.11 References