Laboratory Practical Report

of

**Visual Programming with C#**

**(ICT ED 465)**

Submitted To

**TRIBHUVAN UNIVERSITY**

In Partial Fulfillment of the Requirements of the course

**B.Ed. ICTE 6th Semester**

Submitted By

Sanam Tamang

Symbol No.: 76214020

T.U. Regd. No.: 9-2-214-54-2019

Under the guidance of

**Er. Santosh Dahal**

Lecturer

Sukuna Multiple Campus

**SUKUNA MULTIPLE CAMPUS**

Sundarharaincha-12, Morang, Nepal

2080

**CERTIFICATE**

This is to certify that the Laboratory Practical Report

of

**Visual Programming with C#**

**(ICT ED 465)**

In Partial Fulfillment of the Requirements of the course

**B.Ed. ICTE 6th Semester**

Submitted By

Sanam Tamang

Symbol No.: 76214020

T.U. Regd. No.: 9-2-214-54-2019

is a bonafide record of experiments carried out by him/her under by guidance.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Er. Santosh Dahal

Lecturer

Sukuna Multiple Campus

Sundarharaincha-12, Morang

(Internal Examiner)

Submitted for the Final Examination on: 2080/08/20

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Lecturer

(External Examiner

**Table of Contents**

[**Program.cs** 1](#_Toc152542042)

[**Explanation of program.cs** 1](#_Toc152542043)

[**Form1.cs** 2](#_Toc152542044)

[**Explanation of form1.cs** 9](#_Toc152542045)

[**Output:** 12](#_Toc152542046)

# **Program.cs**

using System;

using System.Windows.Forms;

namespace Calculator

{

static class Program

{

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

## **Explanation of program.cs**

1. **using System;:** This line indicates that the code is using the System namespace. Namespaces in C# organize classes, methods, and other types into logical groups. The System namespace contains fundamental types and basic functionality provided by the .NET Framework.
2. **using System.Windows.Forms;:** This line imports the System.Windows.Forms namespace. It allows you to use Windows Forms controls and features in your application. Windows Forms provides a graphical user interface (GUI) for creating desktop applications.
3. **namespace Calculator:** This defines a namespace named Calculator. Namespaces help prevent naming conflicts and organize related code. All the classes and methods within this file belong to the Calculator namespace.
4. **static class Program:** This declares a static class named Program. A static class cannot be instantiated, and its members (such as methods) can be accessed directly without creating an instance of the class.
5. **static void Main():** This is the entry point of the application. When you run the program, execution starts from this method. It initializes the application, sets visual styles, and runs the main form (Form1).
6. **Application.EnableVisualStyles();:** This line enables visual styles for the application. Visual styles include colors, fonts, and other visual elements that form an operating system theme. It ensures that your application looks consistent with the user’s system settings.
7. **Application.SetCompatibleTextRenderingDefault(false);:** This line sets a property that handles compatibility with old frameworks. If your application is not updated from a .NET Framework 1.0 or 1.1 application, leave it at the default value (false).
8. **Application.Run(new Form1());:** This runs the main form of your application (Form1). The Application.Run method starts the application message loop, which handles events and keeps the application responsive.

# **Form1.cs**

using System;

using System.Drawing;

using System.Windows.Forms;

namespace Calculator

{

public partial class Form1 : Form

{

double totalSum = 0;

string inputvalue;

bool operationCondition = false;

int counter = 0;

double lastInputNumber;

double beforeLastInputNumber;

string number;

bool a = true;

///

int num\_of\_input\_operation = 0;

double firstNum;

double total;// used for sum of first "x (inputOperation) y = total

public Form1()

{

InitializeComponent();

}

Point lastPoint;

//1. x (input operation) y = z for the first time -> onother operation without equals// calculate the current sum without "Btn\_Equals\_OnClick" method

private double Calculation(string inputvalue, double firstNum, double secondNum)

{

double sum = 0;

switch (inputvalue)

{

case "+":

sum = firstNum + secondNum;

break;

case "-":

sum = firstNum - secondNum;

break;

case "x":

sum = firstNum \* secondNum;

break;

case "/":

sum = firstNum / secondNum;

break;

case "√":

textBox\_OutPutValue.Text = Math.Sqrt(totalSum).ToString();

operationCondition = false;

break;

default:

Lbl\_1.Text = "Invalid argument!";

break;

}

return sum;

}

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

{

a = true;

if (textBox\_OutPutValue.Text == "0" || operationCondition)

{

textBox\_OutPutValue.Clear();

}

operationCondition = false;

if (((Button)sender).Text == ".")

{

if (!textBox\_OutPutValue.Text.Contains("."))

{

textBox\_OutPutValue.Text += ((Button)sender).Text;

}

}

else

{

if (counter != 0)

{

textBox\_OutPutValue.Text = ((Button)sender).Text;

counter = 0;

}

else textBox\_OutPutValue.Text += ((Button)sender).Text;

}

lastInputNumber = double.Parse(textBox\_OutPutValue.Text);

number = ((Button)sender).Text;

///1.

if (num\_of\_input\_operation == 0)

{

firstNum = double.Parse(textBox\_OutPutValue.Text);

}

if (num\_of\_input\_operation >= 1)// the condition is ">=1" in order to calculate the following operation/operations without the method "Btn\_Equals\_OnClick" !

{

total = Calculation(inputvalue, firstNum, lastInputNumber);

}

}

private void Btn\_Clear(object sender, EventArgs e)

{

textBox\_OutPutValue.Font = new Font("Nirmala UI", 24, FontStyle.Bold);

Lbl\_1.Font = new Font("Nirmala UI", 12, FontStyle.Bold);

textBox\_OutPutValue.Text = "0";

Lbl\_1.Text = null;

counter = 0;

a = true;

inputvalue = null;

beforeLastInputNumber = 0;///

num\_of\_input\_operation = 0;///

total = 0;

//

number = null;

operationCondition = false;

firstNum = 0;

}

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

{

a = true;

counter = 0;

inputvalue = ((Button)sender).Text;//input operation

totalSum = double.Parse(textBox\_OutPutValue.Text);// current value

operationCondition = true;

//1. check for "x + y \* z must give t."

if (num\_of\_input\_operation >= 1)

{

textBox\_OutPutValue.Text = total.ToString();

firstNum = total;

totalSum = total;// stores the current sum for "Btn\_Equals\_OnClick" method !

}

if (((Button)sender).Text != "√")

{

Lbl\_1.Text = $"{textBox\_OutPutValue.Text} {inputvalue} ";

}

else // √

{

Lbl\_1.Text = $"{inputvalue}({totalSum.ToString()})";

}

num\_of\_input\_operation++;

beforeLastInputNumber = double.Parse(textBox\_OutPutValue.Text);// for example: x + y... , beforeLastInputNumber = x

if (textBox\_OutPutValue.Text.Length >= 14)

{

textBox\_OutPutValue.Font = new Font("Nirmala UI", 17, FontStyle.Bold);

Lbl\_1.Font = new Font("Nirmala UI", 9, FontStyle.Bold);

}

}

private void Btn\_Equals\_OnClick(object sender, EventArgs e)

{

if ((textBox\_OutPutValue.Text != "0" && a && (!operationCondition || (operationCondition && inputvalue == "√") || (operationCondition && num\_of\_input\_operation > 0))) || (textBox\_OutPutValue.Text == "0" && a && !operationCondition /\*&& counter != 0\*/))

{

if (operationCondition)

{

if (num\_of\_input\_operation > 0)

{

Lbl\_1.Text += $"{textBox\_OutPutValue.Text} =";

operationCondition = false;

}

else Lbl\_1.Text += " =";

}

else

{

if (counter == 0)

{

Lbl\_1.Text += $"{textBox\_OutPutValue.Text} =";

}

else

{

if (inputvalue == "√")

{

Lbl\_1.Text = $"{inputvalue}({totalSum.ToString()}) =";

}

else

{

Lbl\_1.Text = $"{textBox\_OutPutValue.Text} {inputvalue} {lastInputNumber} =";

}

}

}

switch (inputvalue)

{

case "+":

if (counter == 0)

{

lastInputNumber = double.Parse(textBox\_OutPutValue.Text);//1

textBox\_OutPutValue.Text = (totalSum + double.Parse(textBox\_OutPutValue.Text)).ToString();

}

else

{

textBox\_OutPutValue.Text = (lastInputNumber + double.Parse(textBox\_OutPutValue.Text)).ToString();

}

a = true;

break;

case "-":

if (counter == 0)

{

lastInputNumber = double.Parse(textBox\_OutPutValue.Text);//1

textBox\_OutPutValue.Text = (totalSum - double.Parse(textBox\_OutPutValue.Text)).ToString();

}

else

{

textBox\_OutPutValue.Text = (double.Parse(textBox\_OutPutValue.Text) - lastInputNumber).ToString();

}

a = true;

break;

case "x":

if (counter == 0)

{

lastInputNumber = double.Parse(textBox\_OutPutValue.Text);//1

textBox\_OutPutValue.Text = (totalSum \* double.Parse(textBox\_OutPutValue.Text)).ToString();

}

else

{

textBox\_OutPutValue.Text = (double.Parse(textBox\_OutPutValue.Text) \* lastInputNumber).ToString();

}

a = true;

break;

case "/":

if (counter == 0)

{

if (!operationCondition && num\_of\_input\_operation > 0)

{

lastInputNumber = double.Parse(textBox\_OutPutValue.Text);

textBox\_OutPutValue.Text = (totalSum / double.Parse(textBox\_OutPutValue.Text)).ToString();

Console.WriteLine("123456789Hello World");

}

}

else

{

textBox\_OutPutValue.Text = (double.Parse(textBox\_OutPutValue.Text) / lastInputNumber).ToString();

}

a = true;

break;

case "√":

textBox\_OutPutValue.Text = Math.Sqrt(totalSum).ToString();

operationCondition = false;

break;

default:

Lbl\_1.Text = "Invalid argument!";

break;

}

counter++;

}

else

{

if (inputvalue != "√")

{

if (number == null || inputvalue == null)//looks for case: 0 -> =

{

Lbl\_1.Text = "0 =";

}

else

{

Lbl\_1.Text = $"{totalSum.ToString()} {inputvalue} 0 =";

}

}

else // √

{

Lbl\_1.Text = $"{inputvalue}({totalSum.ToString()}) =";

}

textBox\_OutPutValue.Text = "0";

if (counter != 0) a = false;

}

if (textBox\_OutPutValue.Text.Length >= 14)

{

textBox\_OutPutValue.Font = new Font("Nirmala UI", 17, FontStyle.Bold);

Lbl\_1.Font = new Font("Nirmala UI", 9, FontStyle.Bold);

}

num\_of\_input\_operation = 0;

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void panel1\_Paint(object sender, PaintEventArgs e)

{

}

private void panel1\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

private void panel1\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void label1\_Click(object sender, EventArgs e)

{

}

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

{

Application.Exit();

}

private void button2\_Click(object sender, EventArgs e)

{

WindowState = FormWindowState.Minimized;

}

private void label1\_MouseDown(object sender, MouseEventArgs e)

{

}

}

}

## **Explanation of form1.cs**

1. **Namespace and Class**: The code is part of a Windows Forms application named “Calculator.” It defines a class Form1 that inherits from Form.
2. **Fields and Variables**:
   * totalSum: A double variable to keep track of the total sum.
   * inputvalue: A string to store the current operation symbol (+, -, x, /, √).
   * operationCondition: A boolean flag indicating whether an operation is in progress.
   * Other variables like counter, lastInputNumber, etc., are used for various calculations.
3. **Constructor (**public Form1()**)**:
   * Initializes the form and sets up event handlers.
   * Calls InitializeComponent() to create UI controls.
4. **Method (**Calculation**)**:
   * Performs arithmetic calculations based on the input operation.
   * Handles addition, subtraction, multiplication, division, and square root.
   * Updates the output text box (textBox\_OutPutValue) with the result.
5. **Method**Btn\_Clear(object sender, EventArgs e):
   * This method is called when the “Clear” button is clicked.
   * It resets various variables and UI elements to their initial states.
   * Key actions:
     1. If the output text box (textBox\_OutPutValue) displays “0” or an operation is in progress (operationCondition), the text box is cleared.
     2. If the clicked button is “.”, it appends a decimal point to the text box (if not already present).
     3. Otherwise, it updates the text box with the clicked button’s value.
     4. The lastInputNumber is updated with the parsed value from the text box.
     5. If this is the first input operation (num\_of\_input\_operation == 0), the firstNum is set.
     6. If there have been one or more input operations (num\_of\_input\_operation >= 1), the total is calculated using the Calculation method.
6. **Method**Btn\_Click(object sender, EventArgs e):
   * This method handles button clicks for numeric and operation buttons.
   * Key actions:
     1. Resets the flag a to true.
     2. Clears the text box if it displays “0” or an operation is in progress.
     3. Appends the clicked button’s value to the text box (except for the decimal point).
     4. Updates lastInputNumber with the parsed value from the text box.
     5. Sets the number variable to the clicked button’s value.
     6. If this is the first input operation (num\_of\_input\_operation == 0), the firstNum is set.
     7. If there have been one or more input operations (num\_of\_input\_operation >= 1), the total is calculated using the Calculation method.
7. **Method**Btn\_Equals\_OnClick(object sender, EventArgs e) (not shown in the snippet):
   * This method is called when the “=” button is clicked.
   * It calculates the final result based on the accumulated input operations.
8. **Other Variables and Initialization**:
   * Various other variables (counter, beforeLastInputNumber, etc.) are used for tracking and calculations.
   * The form’s fonts are set, and initial values are assigned to variables.
9. **Method**Btn\_Equals\_OnClick(object sender, EventArgs e):
   * This method is called when the “=” button is clicked.
   * It handles the calculation and display of the result.
   * Key actions:
     1. Checks various conditions to determine whether to proceed with the calculation.
     2. Updates the label (Lbl\_1) to display the current expression being evaluated.
     3. Based on the input operation (inputvalue), performs the appropriate arithmetic operation.
     4. Updates the output text box (textBox\_OutPutValue) with the result.
     5. Resets the flag a to true.

Here’s a breakdown of the logic within the method:

* If the text box value is not “0” and certain conditions are met (explained in the code), proceed with the calculation.
* If an operation is in progress (operationCondition), update the label with the current expression.
* Otherwise, handle different cases based on the input operation:
  + If it’s addition (+), calculate the sum.
  + If it’s the square root (√), display the square root of the total sum.
  + Otherwise, perform the operation using the last input number and the current text box value.
* Finally, set the flag a to true.

1. **Method**panel1\_Paint(object sender, PaintEventArgs e):
   * This method is called when the panel (panel1) needs to be repainted.
   * It is typically used for custom drawing or graphics, but in this snippet, it’s empty.
2. **Methods for Mouse Interaction**:
   * panel1\_MouseDown(object sender, MouseEventArgs e):
     1. Called when the mouse button is pressed down over the panel.
     2. Records the current mouse position (e.X and e.Y) as the lastPoint.
   * panel1\_MouseMove(object sender, MouseEventArgs e):
     1. Called when the mouse moves over the panel.
     2. If the left mouse button is held down (e.Button == MouseButtons.Left), it adjusts the form’s position based on the difference between the current mouse position and the lastPoint.
     3. This code allows the user to drag the form by clicking and dragging within the panel.
3. **Other Methods**:
   * label1\_Click(object sender, EventArgs e):
     1. Empty method; called when the label (label1) is clicked.
   * button1\_Click(object sender, EventArgs e):
     1. Called when the “Exit” button (button1) is clicked.
     2. Exits the application.
   * button2\_Click(object sender, EventArgs e):
     1. Called when the “Minimize” button (button2) is clicked.
     2. Minimizes the form.
   * label1\_MouseDown(object sender, MouseEventArgs e):
     1. Empty method; called when the label (label1) is clicked.
4. **Closing Remarks**:
   * The code snippet includes event handlers related to mouse interaction (dragging the form) and buttons for exiting or minimizing the application.
   * The actual functionality of these methods would depend on the overall design and behavior of the calculator application.

# **Output:**

