PROG1017

Memo

|  |  |
| --- | --- |
| To: | Cathy Burchill |
| From: | Abiodun Folayan |
| Date: | December 6, 2022 |
| Re: | Lab 5 |

Problem encountered during this Lab is highlighted below:

**Problem**: I initially had problem creating functions that return value.

**Solution**: I was able to get this right after numbers of repeated practices, I was able to resolved this after watching Cathy’s recorded video and checked through the PDF practice as well. It took me approximately 45minutes cumulative to get right

**Problem**: I also had a challenge around creating correct while loop for Addlist

**Solution**: I was able to get this right after I engaged Cathy about the problem. This pointed me to which materials to check precisely

**Problem**: Finally, I had a little challenge around creating SwapData function.

**Solution**: I consulted materials on creating swap Data

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Diagnostics;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.Rebar;

namespace Lab5

{

public partial class Form1 : Form

{

//declare class-level constant string

const string PROGRAMMER = "ABIODUN";

public Form1()

{

InitializeComponent();

}

//Lab 5, by Folayan Abiodun

//Due date: 06-12-2022

//This program focuses majorly on functions creations, random number generations which send and return values.

//it also validates user inputs and calculate calculated value using loops

/\*Name: ResetTextGrp

\* Sent: None

\* Returned: None

\* This function resets the Text group\*/

//Create function ResetTextGroup

private void ResetTextGrp()

{

this.AcceptButton = btnJoin;

txtString1.Text = "";

txtString2.Text = "";

chkSwap.Checked = false;

lblResults.Text = "";

txtString1.Focus();

}

/\*Name: ResetStatsGrp

\* Sent: None

\* Returned: None

\* This function resets the Stats group\*/

//Create function ResetStatsGroup

private void ResetStatsGrp()

{

this.AcceptButton = btnGenerate;

lblSum.Text = "";

lblMean.Text = "";

lblOdd.Text = "";

lstNumbers.Items.Clear();

}

/\*Name: SelectTextRad

\* Sent: None

\* Returned: None

\* This function selects text radio button, hide group text and group stats\*/

//Create function SelectTextRad

private void SelectTextRad()

{

this.AcceptButton = btnGenerate;

radStats.Checked = true;

grpText.Hide();

grpStats.Show();

}

/\*Name: SelectStatsRad

\* Sent: None

\* Returned: None

\* This function unselects stats radio button, hide group text, grp stats and focus the cursor\*/

//Create function SelectStatsRad

private void SelectStatsRad()

{

radStats.Checked = false;

grpStats.Hide();

grpText.Show();

txtString1.Focus();

this.AcceptButton = btnJoin;

}

/\*Name: Setupoption

\* Sent: None

\* Returned: None

\* This function shows and hide appropriate groups based on radbutton selected\*/

//Create function SetupOption

//call selectStatsRad function when radiobutton text is selected

//otherwise selectTextRad function when other radbutton is selected

private void SetupOption()

{

if (radText.Checked)

{

SelectStatsRad();

ResetTextGrp();

}

else

{

SelectTextRad();

ResetStatsGrp();

}

}

/\*Name: Form Load

\* Sent: None

\* Returned: None

\* This function updates form name, hide group, focus cursor and generate random numbers\*/

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

{

//cancatenate form name when form loads

this.Text += "\t " + PROGRAMMER;

//Hide group

grpChoose.Hide();

grpStats.Hide();

grpText.Hide();

//txtbox code to accept the cursor

txtCode.Focus();

//Get generated Random number by calling GetRandom function and write to Authentication code label

int AuthenticationCode = GetRandom(min, max);

lblCode.Text = AuthenticationCode.ToString();

}

/\*Name: GetRandom

\* Sent: integers min and max

\* Returned: int

\* This function generates random numbers based on constant min and max int\*/

//Declare constant integers

//Create function GetRandom, send two integers and return an int

//Declare local constant integers

const int min = 100000, max = 200000;

private int GetRandom(int min, int max)

{

//create random object RandNum, and assign no random value to rand

Random Rand = new Random();

int GetRandom = Rand.Next(min, max);

return GetRandom;

}

/\*Name: btnReset\_Click

\* Sent: none

\* Returned: none

\* This function reset Text grp by calling ResetTextGrp function\*/

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

{

//call function ResetTextGrp

ResetTextGrp();

}

/\*Name: btnClear\_Click

\* Sent: none

\* Returned: none

\* This function Clear stats grp by calling ResetStatsGrp function\*/

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

{

//call function ResetStatsGroup

ResetStatsGrp();

}

/\*Name: radStats\_CheckedChanged

\* Sent: none

\* Returned: none

\* This function radStats checked by calling SetupOption function\*/

private void radStats\_CheckedChanged(object sender, EventArgs e)

{

//call function SetupOption

SetupOption();

}

private void radText\_CheckedChanged(object sender, EventArgs e)

{

//call function SetupOption

SetupOption();

}

//declare local constants

const int MIN = 1000, MAX = 5001, SEED = 733;

private void nudHowMany\_ValueChanged(object sender, EventArgs e)

{

//create random object with seed value of 733

Random rand = new Random(SEED);

//clear the listbox

lstNumbers.Items.Clear();

//run for loop filling listbox with random integers

//rand.Next(): numbers between 1000 and 5001

for (int count = 0; count < nudHowMany.Value; count++)

{

int randomNumber = rand.Next(MIN, MAX);

lstNumbers.Items.Add(randomNumber.ToString("d4"));

}

}

/\*Name: Addlist function

\* Sent: none

\* Returned: int

\* This function calculate sum and return sumup\*/

private int Addlist()

{ //create a variable

int sumup = 0;

int i = 0;

while (i < lstNumbers.Items.Count)

{

sumup += Convert.ToInt32(lstNumbers.Items[i]);

i++;

}

return sumup;

}

/\*Name: CountOdd function

\* Sent: none

\* Returned: int

\* This function counts odd numbers and return OddNum\*/

private int CountOdd()

{ //create a variables

int OddNum = 0;

int i = 0;

do

{

if (Convert.ToInt32(lstNumbers.Items[i]) % 2 != 0)

OddNum++;

i++;

} while (i < lstNumbers.Items.Count);

lblOdd.Text = OddNum.ToString();

return OddNum;

}

//generate and display into labels, the sum, mean and odd numbers

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

{

//create random object with seed value of 733

Random random = new Random(SEED);

//create loop for the generated random numbers

for (int i = 0; i < nudHowMany.Value; i++) {

lstNumbers.Items.Add(random.Next(MIN, MAX));

}

//generate the sum by calling Addlist function

//call Addlist function

Addlist();

//Write sum to the label

lblSum.Text = Addlist().ToString("n0");

//generate the odd numbers by calling CountOdd function

//call CountOdd function

CountOdd();

//write odd numbers into the label

lblOdd.Text = CountOdd().ToString();

//create event to calculate mean

double mean = Addlist() / (double) lstNumbers.Items.Count;

//display mean into the label

lblMean.Text = mean.ToString("n");

}

int count = 0;

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

{

//Declared local constant strings and integer

const string FIRSTATTEMPT = "1 incorrect code entered" + "\n" + "Try again" + "\t- " + "only 3 attempts allowed";

const string SECONDATTEMPT = "2 incorrect code(s) entered" + "\n" + "Try again" + "\t- " + "only 3 attempts allowed";

const string THIRDATTEMPT = "3 incorrect code(s) entered" + "\n" + "Try again" + "\t- " + "only 3 attempts allowed";

const string LASTATTEMPT = "3 attempts to login" + "\n" + "Account locked" + "-" + "Closing Program";

//apply the group if txtcode is the same with display lblcode

//go thrugh the else chain if the condition is not met

if (lblCode.Text == txtCode.Text)

{

grpChoose.Show();

grpLogin.Enabled = false;

radStats.Checked = false;

radText.Checked = true;

btnJoin.Focus();

}

else

{

count++;

if (count == 1)

{

MessageBox.Show(FIRSTATTEMPT, PROGRAMMER);

txtCode.SelectAll();

}

else if (count == 2)

{

MessageBox.Show(SECONDATTEMPT, PROGRAMMER);

txtCode.SelectAll();

}

else if (count == 3)

{

MessageBox.Show(THIRDATTEMPT, PROGRAMMER);

txtCode.SelectAll();

}

else

{

MessageBox.Show(LASTATTEMPT, PROGRAMMER);

this.Close();

}

}

}

//Join the following strings and display into same label

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

{

//write to label

lblResults.Text = "";

//put top string from listbox in label

lblResults.Text = "First string\t " + ""+ "=\t " + "" + txtString1.Text ;

//put bottom string in label

lblResults.Text += "\nSecond string\t " + "" + "=\t " + "" + txtString2.Text;

//put third string in label

lblResults.Text += "\nJoined\t " + "=\t " + txtString1.Text + "\t -->\t " + txtString2.Text;

}

//check bool input - return true if the textbox is populated, else return false

private bool Checkinput()

{

if ((txtString1.Text != "") && (txtString2.Text != ""))

return true;

else

return false;

}

//swap the textboxes by calling swapstring function when the checkboxswap is checked, else, checkboxswap is false

private void chkSwap\_CheckedChanged(object sender, EventArgs e)

{

if (chkSwap.Checked)

{

string string1 = txtString1.Text;

string string2 = txtString2.Text;

Swap(ref string1, ref string2);

txtString1.Text = string1;

txtString2.Text = string2;

lblResults.Text = "Strings have been swapped!";

}

else

{

chkSwap.Checked = false;

}

}

//swap the textboxes when the checkboxswap is checked

private void Swap(ref string string1, ref string string2)

{

string temp = string1;

string1 = string2;

string2 = temp;

}

//Analyze the following strings and display into same label

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

{

//write to label

lblResults.Text = "";

//put first string in label

lblResults.Text = "First string\t " + "=\t " + txtString1.Text;

//put first text length in label

lblResults.Text += "\n\t Characters\t " + "=\t " + txtString1.TextLength;

//put second string in label

lblResults.Text += "\nSecond String\t " + "=\t " + txtString2.Text;

//put second text length in label

lblResults.Text += "\n\t Characters\t " + "=\t " + txtString2.TextLength;

}

}

}