PROG1017

Memo

|  |  |
| --- | --- |
| To: | Cathy Burchill |
| From: | Kang |
| Date: | December 3, 2022 |
| Re: | Lab # 5 |

This create this program we need to gather all the knowledge we acquired during the course.

After going through the material random object wasn’t that difficult. An interesting part was to remember that the last number using the Next method is ignored, so we have to add 1 to the value.

Another interesting part was to use the function directly when using another method rather than storing in a variable to use after.

One part that took me a lot of time was to build the loop for code validation. When I first did I put the counter outside the event and did work, but others studens told me that no variables class levels was allowed. When I put the count inside the event but the value did not changed, and I spent a lot of time trying diferent loops ( bad data first, good data first ), but still did not worked. So I had to leave the count outside.

The hardest part was how to use the index to change to get the numeric up and down value.

CODE

namespace Lab5

{

//Kang Hyun Kwon due 06/dec

//create a program that validates the code first in order to run.

//user has 3 attempts to run or program will close.

//user has to choose groupbox Text - validate , swap , join, count characters

//user choose groupboxStats - choose a number from numeric updown and get ramdom values(seed values). Then, getting the sum , mean and odd numbers

public partial class Form1 : Form

{

//class level constant with name

const string PROGRAMMER = "Kang";

public Form1()

{

InitializeComponent();

}

//GetRandom function. Will return a variable the a random number between MINCODE and MAXCODE values

private int GetRandom(int min, int max)

{ //random with no seed value will give different numbers all the time is called

Random rand = new Random();

//max+1 because the Next method ignore the last number inputed

int randomNum = rand.Next(min, max+1);

return randomNum;

}

//when program runs

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

{

//Min and Max value for GetRandom

const int MINCODE = 100000;

const int MAXCODE = 200000;

//add PROGRAMMER string to the title of the form

this.Text += " "+PROGRAMMER;

grpChoose.Hide();

grpText.Hide();

grpStats.Hide();

txtCode.Focus();

lblCode.Text= Convert.ToString(GetRandom(MINCODE, MAXCODE));

}//end of form load event

//btbLoginEvent. //if user input wrong code and less than 3 times. Show a message and number of error entered

int count = 1;

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

{

const int ERRORSALLOWED = 3;

//allow only 2 mistakes. The third one will close the program.

if (lblCode.Text != txtCode.Text && count <3)

{

MessageBox.Show(count +" incorrect code(s) entered \nTry again - only "+ERRORSALLOWED+ " attempts allowed", PROGRAMMER);

count++;

txtCode.SelectAll();

}//if user insert the right code

else if (lblCode.Text == txtCode.Text)

{

grpLogin.Enabled=false;

grpChoose.Visible=true;

radText.Checked=true;

grpText.Visible=true;

SetupOption();

}

//if user commit mistakes 3 times, show a message and close the program

else

{

MessageBox.Show(count + " attempts to login \nAccount locked - Closing program", PROGRAMMER);

Close();

}

}//end of Login event

//clear txtbox and set acceptbutton to btnJoin and cancelbutton to btnReset

private void ResetTextGrp()

{

txtString1.Text="";

txtString2.Text="";

lblResults.Text="";

this.AcceptButton= btnJoin;

this.CancelButton= btnReset;

txtString1.Focus();

}// end of resetTextGrp function

//function ResetStatsGrp - clear all labels and lstbox. Acceptbutton is btnGenerate and cancelbutton is btnClear

private void ResetStatsGrp()

{

lblSum.Text="";

lblMean.Text="";

lblOdd.Text="";

lstBoxStats.Items.Clear();

this.AcceptButton= btnGenerate;

this.CancelButton= btnClear;

//Get minimum value for numeric updown

nudHowMany.Value = nudHowMany.Minimum;

nudHowMany.Focus();

}//end of ResetStatsGrp

//function SetupOption - will hide or show the groupbox corre

private void SetupOption()

{

if (radText.Checked)

{

grpText.Visible=true;

grpStats.Visible=false;

//call function ResetTextGrp

ResetTextGrp();

}

else

{

grpText.Visible=false;

grpStats.Visible=true;

//call function ResetStatsGrp

ResetStatsGrp();

}

}

//event will call SetupOption function

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

{

SetupOption();

}

//event will call SetupOption function

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

{

SetupOption();

}

//btnReset event will call ResetTextGrp function

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

{

ResetTextGrp();

}

//btnClear event will call ResetStatsGrp function

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

{

ResetStatsGrp();

}

//Function swap in GrpText. It will get value by reference

private void Swap(ref string txt1, ref string txt2)

{

//create a temporary variable only to store the first value and be sent to the other variable

string temp = txt1;

txt1 = txt2;

txt2= temp;

}//end of Swap function

//function CheckInput- if txtString1 or txtString2 is empty will return false. Otherwise, true.

private bool CheckInput()

{

bool isValid = true;

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

{

isValid=false;

}

return isValid;

}//end of CheckInput function

//Swap checkbox event

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

{

//if txtbox is filled with data

if (CheckInput())

{

//store txtbox values in variables

string txt1 = txtString1.Text;

string txt2 = txtString2.Text;

//it will only swap the txtboxes if checkbox is checked

if (chkSwap.Checked)

{

txtString1.Text= txt2;

txtString2.Text=txt1;

}

//call the function Swap sending value by reference

Swap(ref txt1, ref txt2);

//Show a message when swapped

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

}

}//end of Swap checkbox event

//btn Join event

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

{

//call function CheckInput

if (CheckInput()){

//show info and and result

lblResults.Text="First string = " + txtString1.Text + "\nSecond string= " +txtString2.Text +"\nJoined = " +

txtString1.Text + "-->" + txtString2.Text;

}

}

//btnAnalyze event

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

{ //call function CheclInput

if (CheckInput())

{

//call method Length that return an integer with the number of characters in the txtbox

lblResults.Text="First String = "+ txtString1.Text+ "\nCharacters = " + txtString2.Text.Length +

"\nSecond string = " + txtString2.Text + "\nCharacters = " +txtString2.Text.Length;

}

}//end of btnAnalyze event

//btnGenerate event

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

{

//constants set for MIN and MAX value

const int MIN = 1000;

const int MAX = 2000;

//random number using seed value- always generates the same sequence

Random rand = new Random(733);

//clear the lstBox

lstBoxStats.Items.Clear();

//Get the numbers chosen by the user on numeric up and down

int howMany = Convert.ToInt32(nudHowMany.Value);

//for loop to generate x random numbers chosen from numeric up and down

for (int i = 0; i < howMany; i++)

{

int randomNum = rand.Next(MIN, MAX-1);

lstBoxStats.Items.Add(randomNum);

}//

//call the function AddList to get the sum of the numbers created

AddList();

lblSum.Text= Convert.ToString(AddList());

//create a var mean which will find the mean by getting the sum from function AddList and

//dividing by the quantity of number is listbox.

double mean = AddList()/lstBoxStats.Items.Count;

//converting the variable to String

lblMean.Text= mean.ToString("n2");

//Insert the number of Odd numbers by calling the function CountOdd

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

}//end of btnGenerate event

//function AddList to get the Sum

private int AddList()

{ //count initialized by 0 because index starts at 0

int count = 0;

int sum = 0;

//while will loop while true

while (count <lstBoxStats.Items.Count)

{ //value will be added while statement is true getting values by their Index number ( starts at 0)

sum += Convert.ToInt32(lstBoxStats.Items[count]);

//count +1 to get the next velue from list box

count++;

}

//return the total sum os the list

return sum;

}//end of function Addlist

//Function CountOdd

private int CountOdd()

{

//initialized by 0 because index starts at 0

int count=0;

int oddNum=0;

do

{

//if will each value by their index(starts at 0), and get odd number by %2 =1.

if(Convert.ToInt32(lstBoxStats.Items[count])%2 == 1)

{ //Add +1 if true

oddNum++;

}

//add counter +1

count++;

//return loop while count < list Items number.

} while (count < lstBoxStats.Items.Count);//end of CountOdd function

return oddNum;

}//end of CountOdd function

}

}//end of program