|  |
| --- |
| Tomasz Przybylski KC68744 |
| CMDF CX3005 Portfolio |
| Fibonacci  Caesar Cypher  Grade Calculator  Password Checker & Generator |

|  |
| --- |
| Tomasz Przybylski  25/04/2017 |

Contents

[Fibonacci Program 2](#_Toc480881162)

[Summary: 2](#_Toc480881163)

[Flowcharts: 2](#_Toc480881164)

[General Flowchart: 2](#_Toc480881165)

[Event Specific: 4](#_Toc480881166)

[Annotated Code: 5](#_Toc480881167)

[Variable List: 6](#_Toc480881168)

[Comments: 7](#_Toc480881169)

[Test Plan: 8](#_Toc480881170)

[Caesar Cypher 9](#_Toc480881171)

[Summary: 9](#_Toc480881172)

[Flowcharts: 9](#_Toc480881173)

[General Flowchart: 9](#_Toc480881174)

[Event Specific: 10](#_Toc480881175)

[Annotated Code: 11](#_Toc480881176)

[Variable List: 14](#_Toc480881177)

[Comments: 15](#_Toc480881178)

[Test Plan: 16](#_Toc480881179)

[Grade Calculator 17](#_Toc480881180)

[Summary: 17](#_Toc480881181)

[Flowcharts: 17](#_Toc480881182)

[General: 17](#_Toc480881183)

[Event Specific: 18](#_Toc480881184)

[Annotated Code: 19](#_Toc480881185)

[Display Form: 19](#_Toc480881186)

[Certificate Form: 22](#_Toc480881187)

[Variable List: 23](#_Toc480881188)

[Comments: 24](#_Toc480881189)

[Test Plan: 25](#_Toc480881190)

[Password Generator & Checker 26](#_Toc480881191)

[Summary: 26](#_Toc480881192)

[Flowcharts: 26](#_Toc480881193)

[Annotated Code: 27](#_Toc480881194)

[Form1 (Login): 27](#_Toc480881195)

[Form 2 (Generation): 30](#_Toc480881196)

[Variables List: 33](#_Toc480881197)

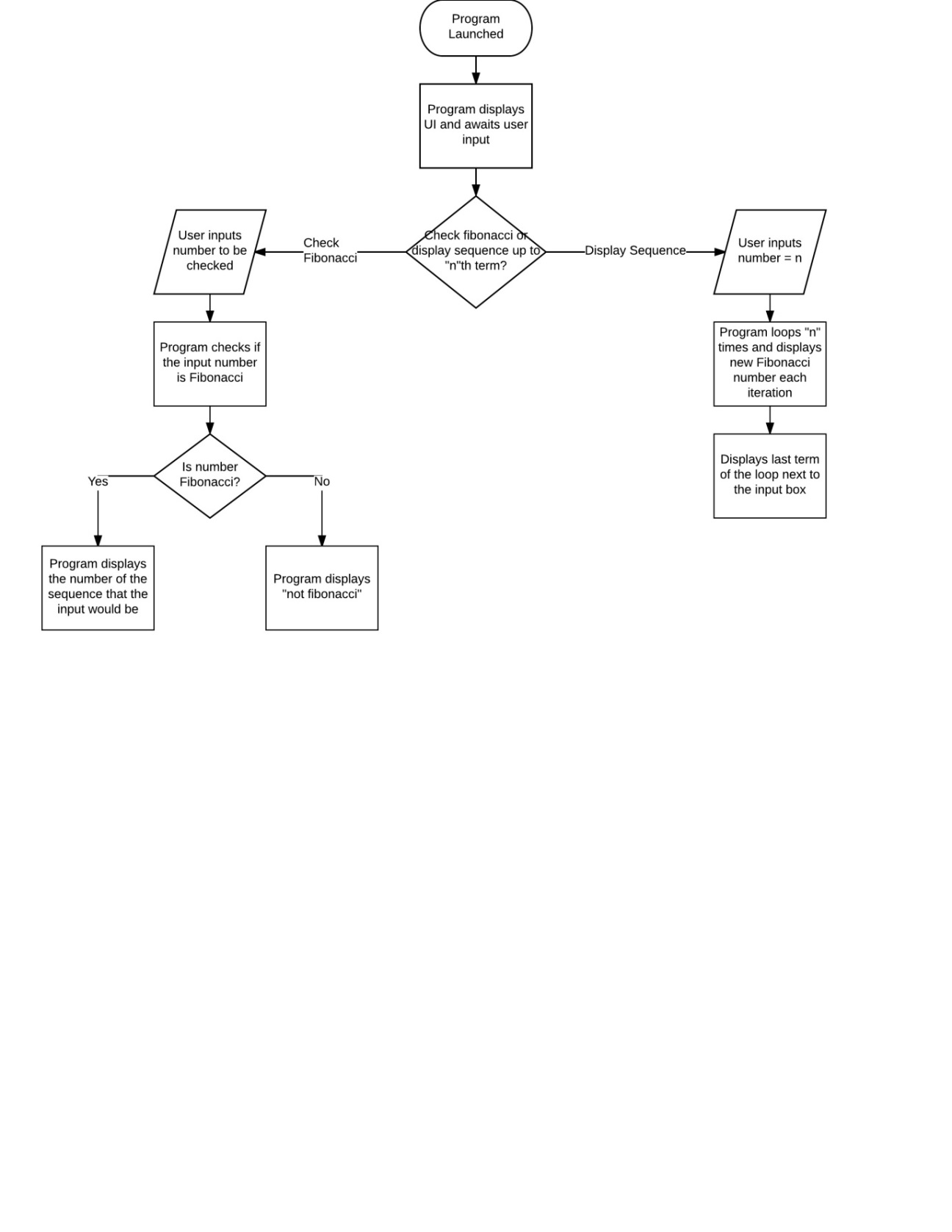
# Fibonacci Program

## Summary:

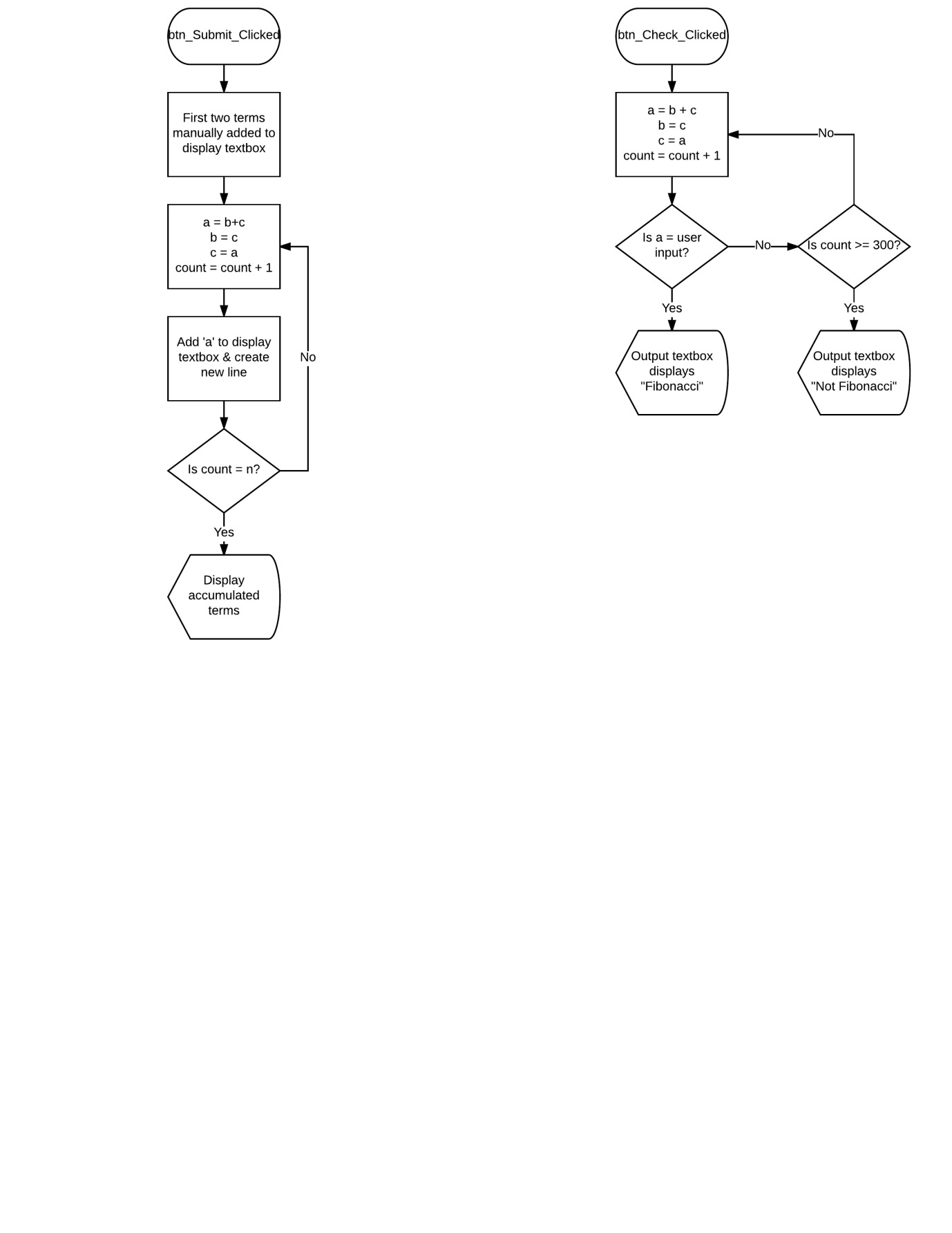
The program allows the user to look at the sequence of Fibonacci numbers up to the 91st Fibonacci number, and allows for users to input an integer and check if it is part of the Fibonacci sequence, and also which nth term in the sequence it is.

## **Flowcharts:**

### **General Flowchart**:



### Event Specific:



## Annotated Code:

Public Class Form1

Private Sub btn\_Submit\_Click(sender As Object, e As EventArgs) Handles btn\_Submit.Click

txt\_DisplaySeq.Clear()

Dim Ln\_a As Long = 0

Dim Ln\_b As Long = 1

Dim Ln\_fibo As Long = 0 'Fibonacci number

Dim Int\_nth As Integer = 1 'N-th Fibonacci number

txt\_DisplaySeq.Text = "\*0th Number\* " & txt\_DisplaySeq.Text + Ln\_a.ToString & ControlChars.NewLine 'Displays the first two digits of the Fibonacci sequence (0, 1)

txt\_DisplaySeq.Text = txt\_DisplaySeq.Text + Ln\_b.ToString & ControlChars.NewLine

Try

Do

Ln\_fibo = Ln\_a + LN\_b 'Adds the two values of a and b'

Ln\_a = Ln\_b 'A takes the value of B, while B takes the value of a+b'

Ln\_b = Ln\_fibo

txt\_DisplaySeq.Text = txt\_DisplaySeq.Text + Ln\_fibo.ToString & ControlChars.NewLine 'Writes the fibonacci number to the text box and creates a new line'

Int\_nth = Int\_nth + 1

Loop Until Int\_nth = txt\_Enternth.Text

'Loops for nth number of times before stopping the loop

txt\_DisplayFibo.Text = Ln\_fibo.ToString 'writes final loop result to the text box

Catch ex As Exception

txt\_DisplaySeq.Text = "Invalid Term - Check if out of range"

txt\_DisplayFibo.Text = "Invalid Term"

End Try

End Sub

Private Sub btn\_Check\_Click(sender As Object, e As EventArgs) Handles btn\_Check.Click

Dim Dbl\_a As Double = 0

Dim Dbl\_b As Double = 1

Dim Dbl\_fibo As Double = 0

Dim Int\_nth As Integer = 1

Dim Dbl\_checkinput As Double

Dim Bln\_fibonacci As Boolean

Try

Dbl\_checkinput = txt\_CheckInput.Text

Catch ex As Exception

End Try

If Dbl\_checkinput < 1 Then

txt\_CheckInput.Text = "Invalid Input"

End If

Do

Dbl\_fibo = Dbl\_a + Dbl\_b

If Dbl\_fibo = Dbl\_checkinput Then 'checks if the input is a fibonacci number with each loop

Bln\_fibonacci = True 'if the two numbers match then the loop will end early

txt\_CheckOutput.Text = Int\_nth + 1 'If the input is a fibonacci number, this will display which fibonacci number it is.

End If

Dbl\_a = Dbl\_b

Dbl\_b = Dbl\_fibo

Int\_nth = Int\_nth + 1

Loop Until Int\_nth = 300 Or Bln\_fibonacci = True

'Loops until the number being checked turns out to be a Fibonacci or until Loop gets to 300th iteration

If Bln\_fibonacci = True Then

Else

txt\_CheckOutput.Text = "Not Fibonacci"

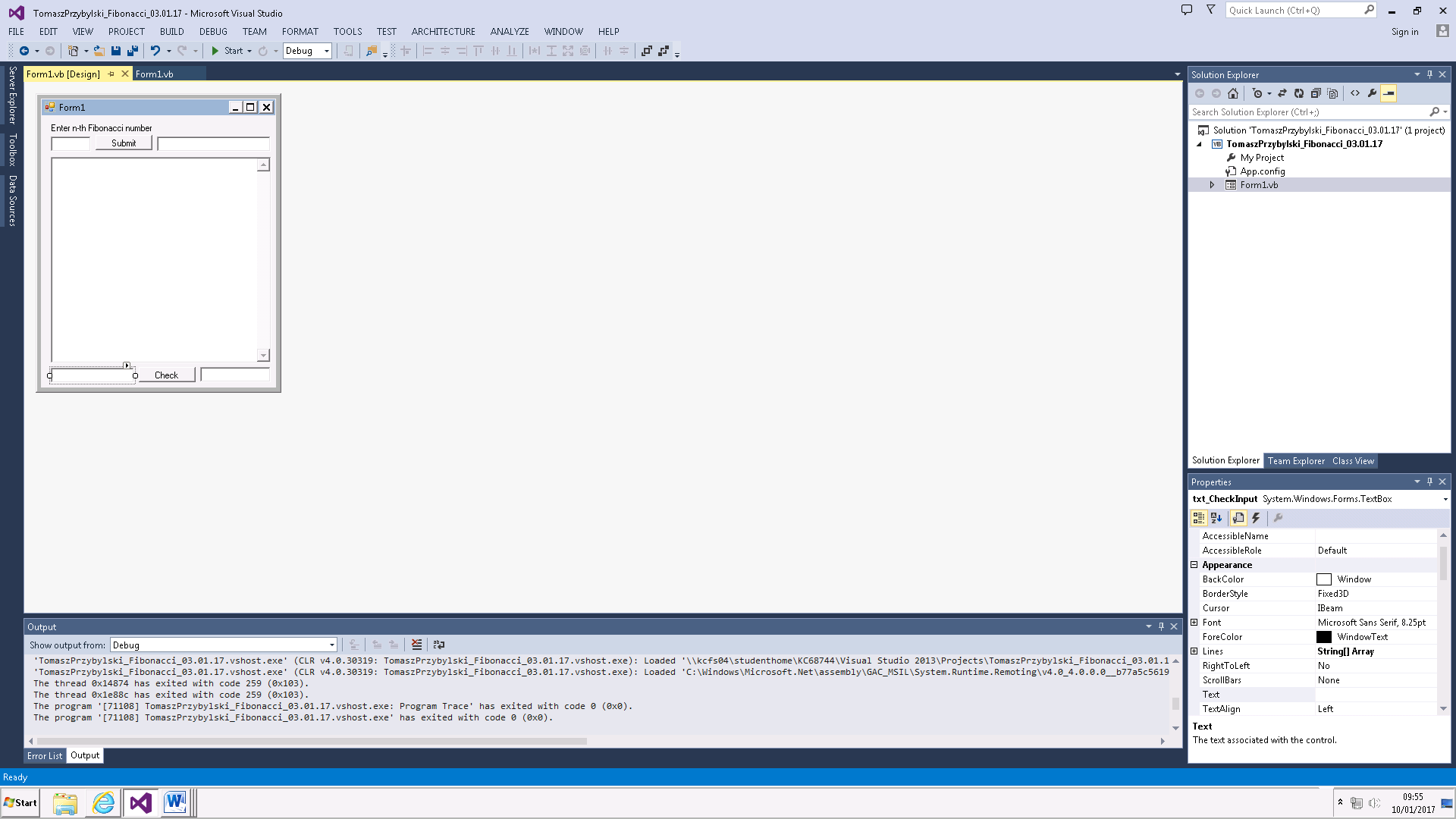
End If

End Sub

End Class

### Variable List:

|  |  |  |
| --- | --- | --- |
| Name of Variable | Type of Variable | Other Comments |
| Ln\_a | Long | Used in the sequence displaying. |
| Dbl\_a | Double | Used in the Fibonacci number checking |
| Ln\_b | Long | Used in the sequence displaying. |
| Dbl\_b | Double | Used in the Fibonacci number checking |
| Ln\_Fibo | Long | Used in the sequence displaying. |
| Dbl\_Fibo | Double | Used in the Fibonacci number checking |
| Int\_nth | Integer | Used as a count to stop the loop, either at “n”th term or once it reaches 300 for the Fibonacci check |
| Dbl\_Checkinput | Double | Holds the number to be checked |
| Bln\_Fibonacci | Boolean | Will reflect if the number is Fibonacci or not |



Input for checking if a number is a Fibonacci number

Displays “Not Fibonacci” if number is not a Fibonacci number, otherwise displays the term of the Fibonacci number.

For example: 13 would display 7, because it is the 7th Fibonacci term

Displays all the Fibonacci numbers up to the “N-th” term in sequence

The “N-th” Fibonacci number

Input Box for “N-th” term

## Comments:

* In my submit.click sub, the “Ln\_a”, “Ln\_b”, and “Ln\_fibo” variables are declared as long, whereas in the check.click sub, they are declared as double. This is because a “long” type variable will have less range but more precision than a “double” type variable. This means that my program can only display 91 terms, but it can display them all fully. This is preferred because the checking part of the program can only check precise numbers. If I used double variable types in my sequence generation, I could have achieved up to and beyond 1000 terms for the Fibonacci sequence, but they would begin to lose precision and be written in scientific notation.
* Although my sequence generator will display 91 terms, the Fibonacci checker can be used up to the 216th term. This is because of the previously mentioned “long” and “double” variable difference.

## Test Plan:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test | Description | Input | Expected Outcome | Actual Outcome | Other Comments |
| #1 | Input a positive integer into the “N-th term” textbox | 7 | Fibonacci Sequence displays until 7th term | As expected |  |
| #2 | Input a negative integer into the “N-th term” textbox | -13 | Invalid term displayed | As expected |  |
| #3 | Input a string into the “n-th term” textbox | ABCD | Invalid term displayed | As expected |  |
| #4.1  #4.2 | Input a very long number or string into the textbox | ABCDEFGHIJKLMNOPQRSTUVWXYZ  1234567891011121314151617181920 | Invalid term displayed | As expected |  |
| #5 | Input a string into the textbox for checking fibonacci | ABCD | Not Fibonacci displayed | As expected |  |
| #6 | Input a Fibonacci number into textbox for checking | 21 | 8 | As expected |  |
| #7 | Input a non-Fibonacci number into textbox for checking | 22 | Not Fibonacci | As expected |  |
| #8 | Input very long integer into textbox for checking | 999999999999999999999999999999999999 | Not Fibonacci | As expected |  |
| #9 | No input for “N-th term” textbox |  | Invalid term displayed | As expected |  |
| #10 | No input for the Fibonacci checker |  | Invalid term displayed | As expected – also displays “Not Fibonacci” |  |

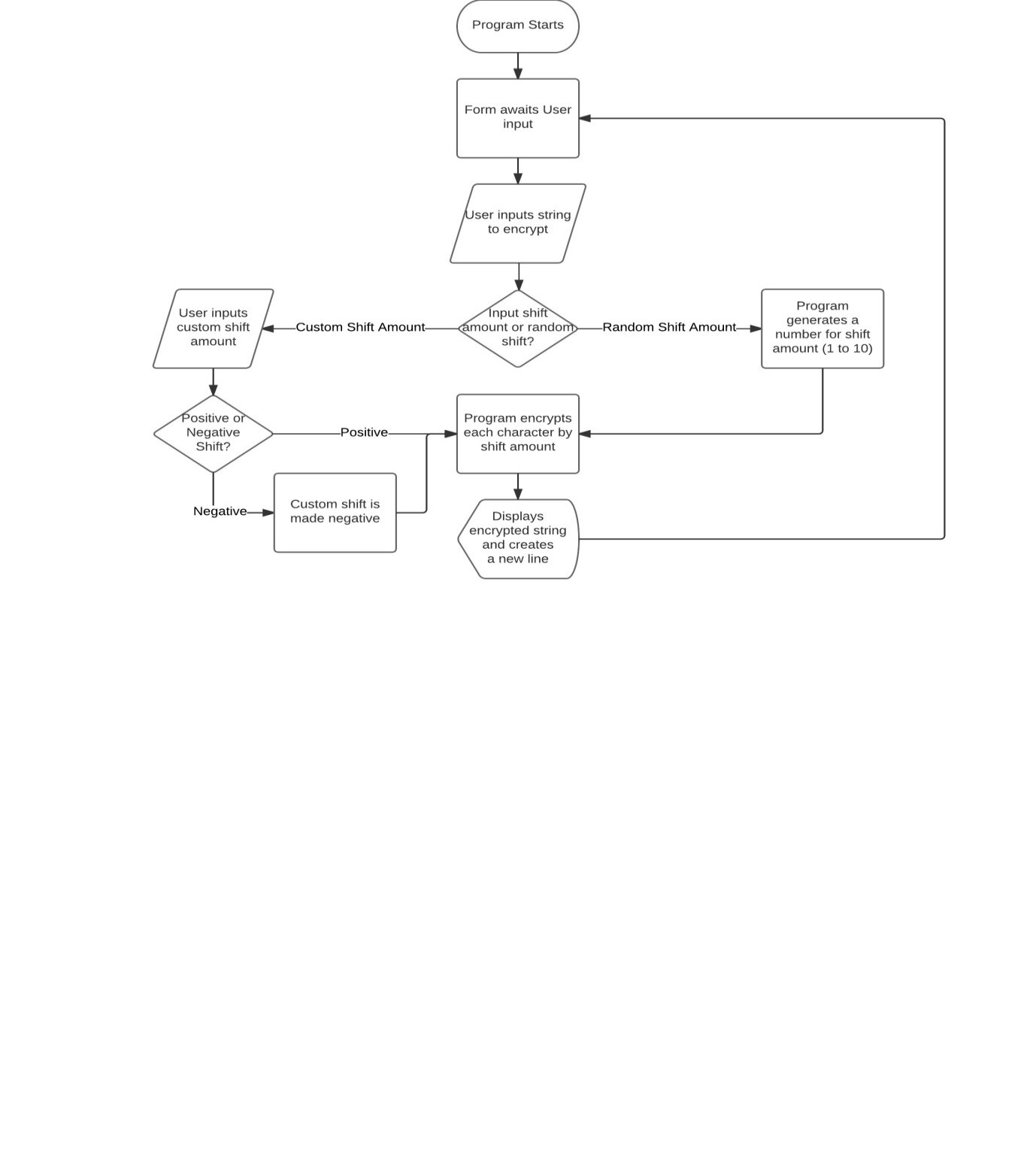
# Caesar Cypher

## Summary:

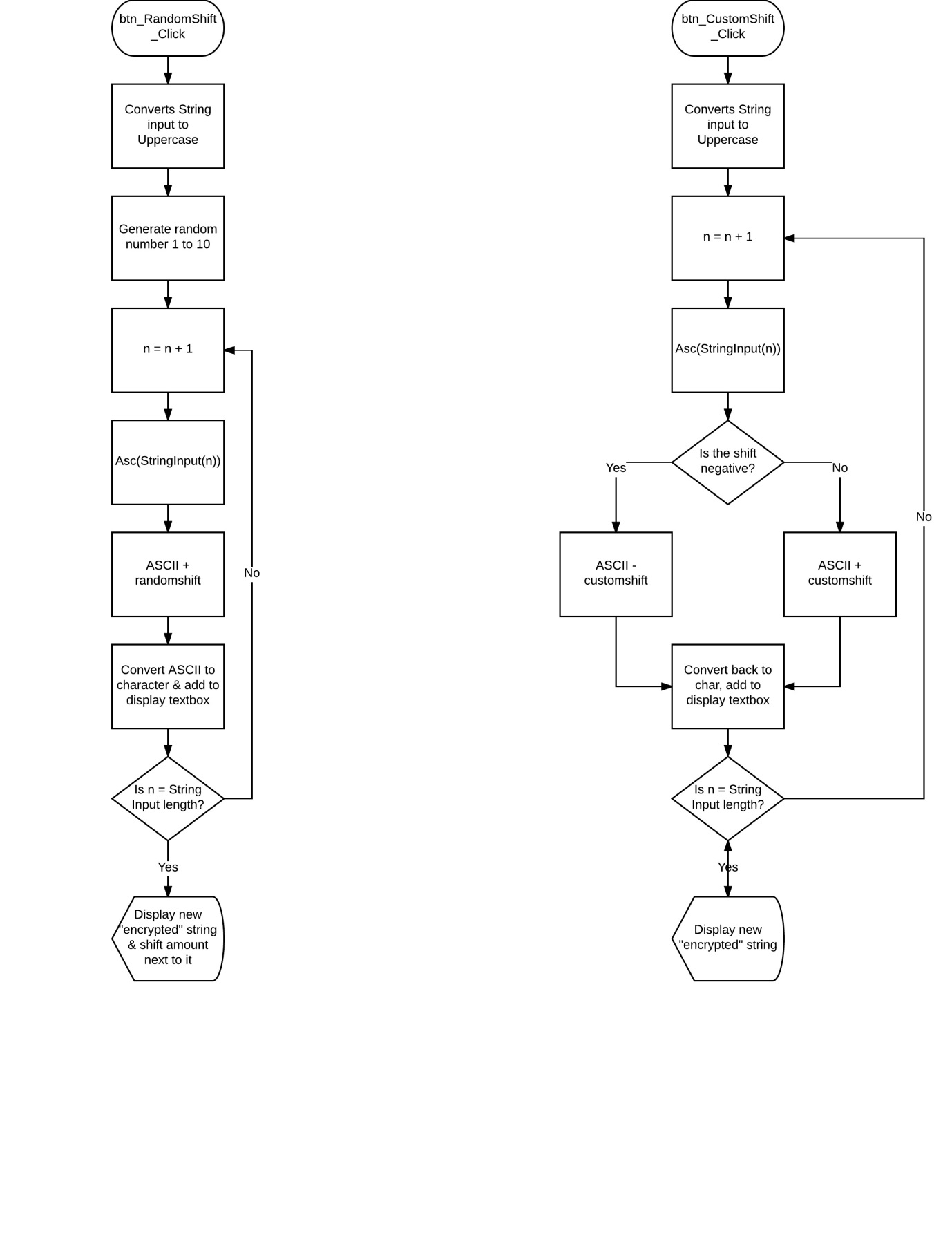
This program allows the user to input a series of strings, and then either have the sequence encrypted via a shift (up or down) of their choice, or to have a random shift of 1-10 applied. The program then displays the encrypted string and if a random shift was chosen, it will also display the shift amount, needed to decrypt the message.

## Flowcharts:

### General Flowchart:



### Event Specific:



## Annotated Code:

Public Class frm\_Main

Dim Str\_cypherinput As String 'the string input is assigned here

Public Int\_shiftamount As Integer 'the integer input for the shift

Public Str\_cypheroutput As String 'the string after it has been shifted

Dim Int\_Inputlength As Integer 'the length of the input string

Dim IntArr\_ShiftArr(1) As Integer 'an array storing the ASCII of the shifted character

Dim StrArr\_PostArr(1) As String 'an array storing the shifted character

Dim Int\_ran As Integer 'takes the value of the randomly generated number

Dim Rdm\_Rd As New Random 'used to generate a random number

Private Sub btn\_RandomShift\_Click(sender As Object, e As EventArgs) Handles btn\_RandomShift.Click

Str\_cypherinput = UCase(txt\_InputWord.Text) 'converts string to uppercase

Int\_Inputlength = txt\_InputWord.TextLength - 1 'corrects the string length so that calculations are correct

Int\_ran = Rdm\_Rd.Next(1, 11) 'generates a random integer between inclusive 1 and exclusive 11 (1-10)

Int\_shiftamount = Int\_ran

ReDim IntArr\_ShiftArr(Int\_Inputlength) 're-declares the IntArr\_ShiftArr to the size of the string length

ReDim StrArr\_PostArr(Int\_Inputlength) 're-declares the StrArr\_PostArr to the size of the string length

If txt\_InputWord.Text = "" Then

MsgBox("No String Input")

GoTo escape1

End If

For i = 0 To Int\_Inputlength

If Str\_cypherinput(i) <> " " Then

IntArr\_ShiftArr(i) = Asc(Str\_cypherinput(i)) + Int\_shiftamount 'converts the current character into ASCII and adds the shift amount onto the ASCII value

If IntArr\_ShiftArr(i) > 90 Then

IntArr\_ShiftArr(i) = IntArr\_ShiftArr(i) - 26 'if the ASCII value is above the uppercase Z value, that particular ASCII value is moved back down to uppercase A, creating a loop like a physical caesar cypher would create

End If

StrArr\_PostArr(i) = Convert.ToChar(IntArr\_ShiftArr(i))

ElseIf Str\_cypherinput(i) = " " Then 'spaces are kept in the same place

StrArr\_PostArr(i) = " "

End If

Next

For i = 0 To Int\_Inputlength

Str\_cypheroutput = Str\_cypheroutput + StrArr\_PostArr(i) 'this re-creates the string with the new encrypted characters

Next

txt\_Display.Text = txt\_Display.Text + Str\_cypheroutput & " Shift Amount: " & Int\_shiftamount.ToString & ControlChars.NewLine 'displays the encrypted (or decrypted) string in the textbox on the bottom, and creates a new line

Str\_cypheroutput = ""

escape1:

End Sub

Private Sub btn\_positive\_Click(sender As Object, e As EventArgs) Handles btn\_Positive.Click

'Majority of this code is the same as previous, with only some of the variables changed and lacking the integer generation

Str\_cypherinput = UCase(txt\_InputWord.Text)

Int\_Inputlength = txt\_InputWord.TextLength - 1

Try 'a try catch block, in case there is a problem with the shift input

Int\_shiftamount = txt\_CustomShift.Text

Catch ex As Exception

MessageBox.Show("Invalid Shift Input")

End Try

If Int\_shiftamount >= 26 Or Int\_shiftamount < 1 Then

MessageBox.Show("Please input a number between 1 and 25")

GoTo escape1

End If

ReDim IntArr\_ShiftArr(Int\_Inputlength)

ReDim StrArr\_PostArr(Int\_Inputlength)

For i = 0 To Int\_Inputlength

If Str\_cypherinput(i) <> " " Then

IntArr\_ShiftArr(i) = Asc(Str\_cypherinput(i)) + Int\_shiftamount

If IntArr\_ShiftArr(i) > 90 Then

IntArr\_ShiftArr(i) = IntArr\_ShiftArr(i) - 26 'this accounts for the "Positive" shift

End If

StrArr\_PostArr(i) = Convert.ToChar(IntArr\_ShiftArr(i))

ElseIf Str\_cypherinput(i) = " " Then

StrArr\_PostArr(i) = " "

End If

Next

For i = 0 To Int\_Inputlength

Str\_cypheroutput = Str\_cypheroutput + StrArr\_PostArr(i)

Next

txt\_Display.Text = txt\_Display.Text + Str\_cypheroutput & ControlChars.NewLine

Str\_cypheroutput = ""

escape1:

End Sub

Private Sub txt\_customshift\_KeyPress(ByVal sender As System.Object, ByVal e As System.Windows.Forms.KeyPressEventArgs) Handles txt\_CustomShift.KeyPress

If Not Char.IsDigit(e.KeyChar) And Not Char.IsControl(e.KeyChar) Then

e.Handled = True 'disables the user from inputting any non-integer characters

End If

End Sub

Private Sub btn\_Clear\_Click(sender As Object, e As EventArgs) Handles btn\_Clear.Click

txt\_Display.Clear() 'clears the display

End Sub

Private Sub btn\_Negative\_Click(sender As Object, e As EventArgs) Handles btn\_Negative.Click

Str\_cypherinput = UCase(txt\_InputWord.Text) 'identical to the positive shift sub, just moves the ASCII down instead of up

Int\_Inputlength = txt\_InputWord.TextLength - 1

Try

Int\_shiftamount = txt\_CustomShift.Text

Catch ex As Exception

MessageBox.Show("Invalid Shift Input")

End Try

If Int\_shiftamount >= 26 Or Int\_shiftamount < 1 Then

MessageBox.Show("Please input a number between 1 and 25")

GoTo escape1

End If

ReDim IntArr\_ShiftArr(Int\_Inputlength)

ReDim StrArr\_PostArr(Int\_Inputlength)

For i = 0 To Int\_Inputlength

If Str\_cypherinput(i) <> " " Then

IntArr\_ShiftArr(i) = Asc(Str\_cypherinput(i)) - Int\_shiftamount

If IntArr\_ShiftArr(i) < 65 Then

IntArr\_ShiftArr(i) = IntArr\_ShiftArr(i) + 26 'this accounts for the "Negative" shift

End If

StrArr\_PostArr(i) = Convert.ToChar(IntArr\_ShiftArr(i))

ElseIf Str\_cypherinput(i) = " " Then

StrArr\_PostArr(i) = " "

End If

Next

For i = 0 To Int\_Inputlength

Str\_cypheroutput = Str\_cypheroutput + StrArr\_PostArr(i)

Next

txt\_Display.Text = txt\_Display.Text + Str\_cypheroutput & ControlChars.NewLine

Str\_cypheroutput = ""

escape1:

End Sub

Private Sub txt\_InputWord\_KeyPress(ByVal sender As System.Object, ByVal e As System.Windows.Forms.KeyPressEventArgs) Handles txt\_InputWord.KeyPress

If Not Char.IsLetter(e.KeyChar) And Not Char.IsControl(e.KeyChar) And Not Char.IsSeparator(e.KeyChar) Then

e.Handled = True 'disables input of numbers of symbols (apart from Spacebar) into the "Input text" textbox

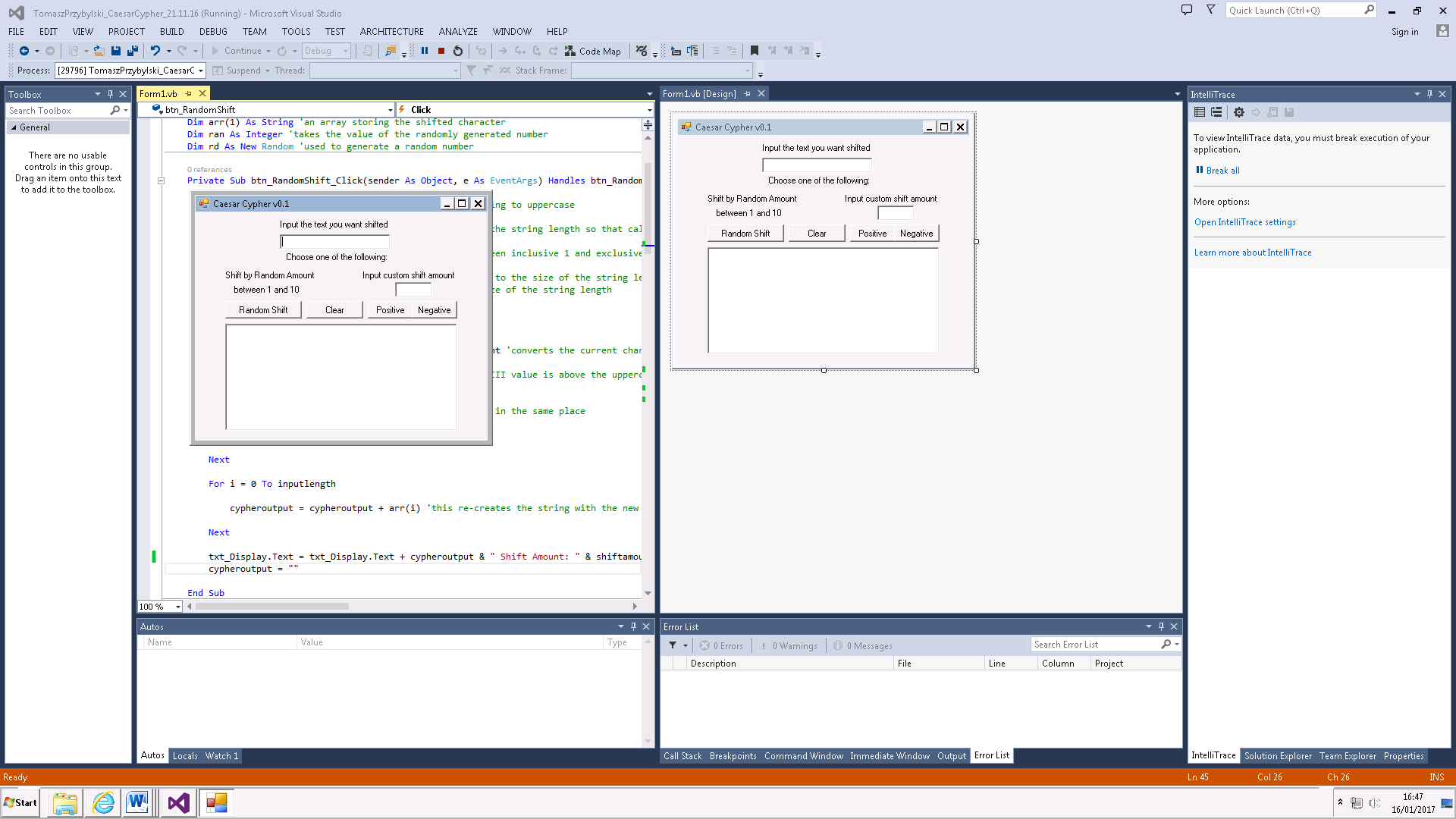
End If

End Sub

End Class

### Variable List:

|  |  |  |
| --- | --- | --- |
| Variable Name | Variable Type | Other Comments |
| Str\_cypherinput | String | Holds the string to be encrypted |
| Int\_shiftamount | Integer | Holds the amount of places to shift the Caesar cypher |
| Str\_cypheroutput | String | Holds the string after its been encrypted |
| Int\_inputlength | Integer | Holds the length of the string (used in the array) |
| IntArr\_ShiftArr | Integer (array) | Holds the ASCII of each letter after being shifted |
| StrArr\_PostArr | String (array) | Holds each of the next encrypted letters |
| Int\_ran | Integer | Holds the randomly generated integer from Rdm\_Rd |
| Rdm\_Rd | New Random | Randomly generates an integer between 1 and 10 inclusive |



Output textbox. Will display the encrypted string

Shifts input text by a random amount between 1 and 10

Clears output textbox

Negative shift

Positive shift

Input for shift amount

Input for string

## Comments:

* I made two separate arrays for this program. Although the program could be created without using these and by re-creating the string during the loop, this method minimises the chances of errors or mix-ups during the code and makes it much friendlier to understand by looking at it.
* The clear button is only included for user-friendliness. It does not affect how the program executes
* Originally my program just shifted the input text either randomly or by a custom amount, but I decided to split that up to Positive and Negative shift so that the program could be also used to decrypt anything as long as the shift amount is known. This also doubles as a debug tool to check if the input text was being shifted correctly one way or another.
* I also decided to limit the shifts to letters like a real Caesar Cypher would, and in the same way that a Caesar Cypher loops with Z shifted by 1 equalling A, I made the code stay within the bounds of the ASCII letters (with the exception of a spacebar)
* I disabled any number or symbol input for the string.

## Test Plan:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test | Description | Input | Expected Outcome | Actual Outcome | Other Comments |
| #1 | Input 11 Character string into textbox & shift by +11 | Hello Hello | SPWWZ SPWWZ | As expected |  |
| #2 | Input 11 Character string into textbox & shift by -11 | Hello Hello | WTAAD WTAAD | As expected |  |
| #3 | Input 20 character string into textbox & shift by +- 13 | ABCDEFGHIJKLMNOPQRST | NOPQRSTUVWXYZABCDEFG | As expected |  |
| #4 | Input numbers & symbols into textbox | 1234+-{} | Inputs not allowed to be keyed in | As expected |  |
| #5 | Shift amount set to -3 | -3 | Invalid amount message displayed | As expected |  |
| #6 | Shift amount set to 40 | 40 | Invalid amount message displayed | As expected |  |
| #7 | Random Shift used with string in textbox | Help | Shifted by a random amount | As expected |  |
| #8 | No input for string & shift attempted |  | Invalid input message | As expected |  |
| #9 | Input for string present, no shift input | ABC | Invalid shift input & error message displayed | As expected |  |
| #10 | No input for string & random shift attempted |  | No string message displayed | As expected |  |

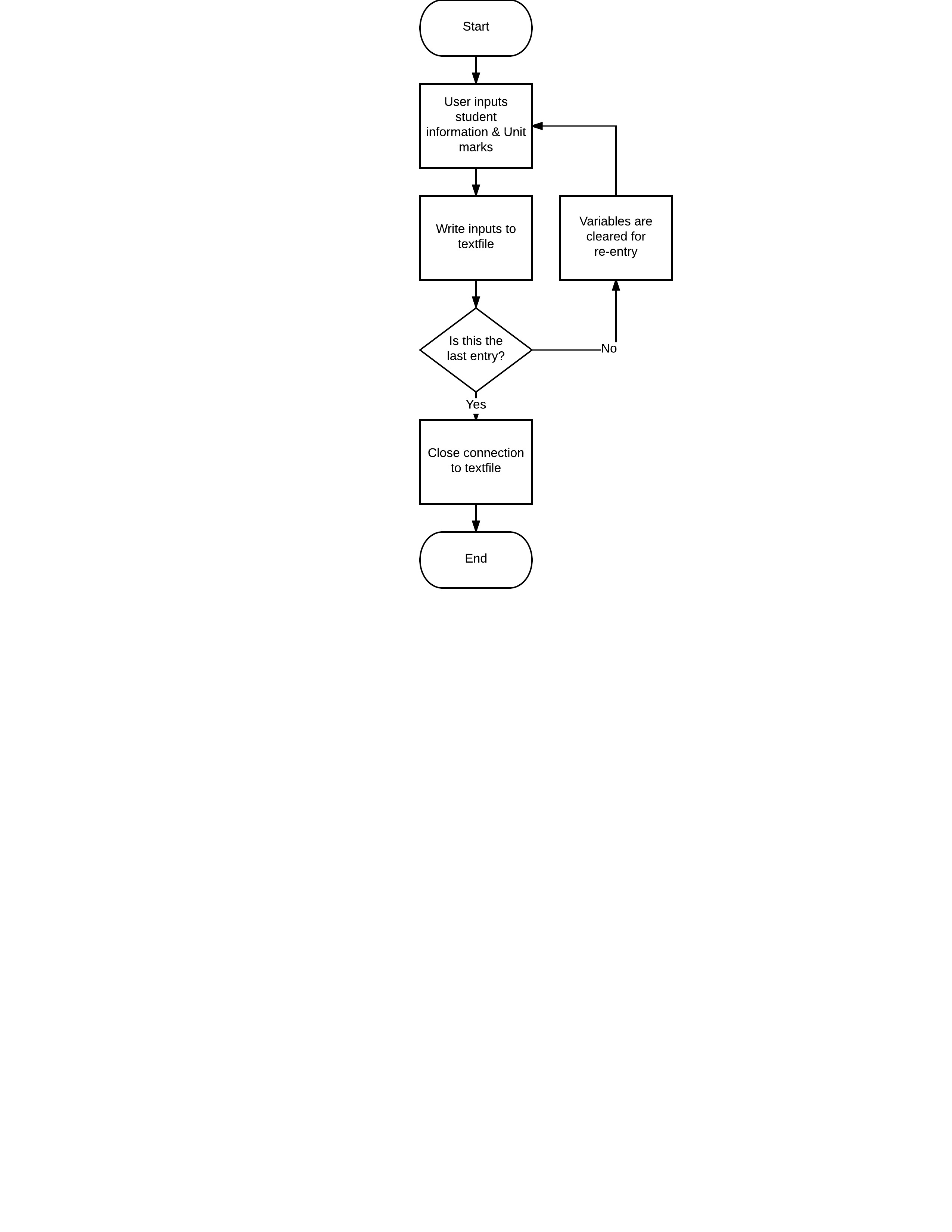
# Grade Calculator

## Summary:

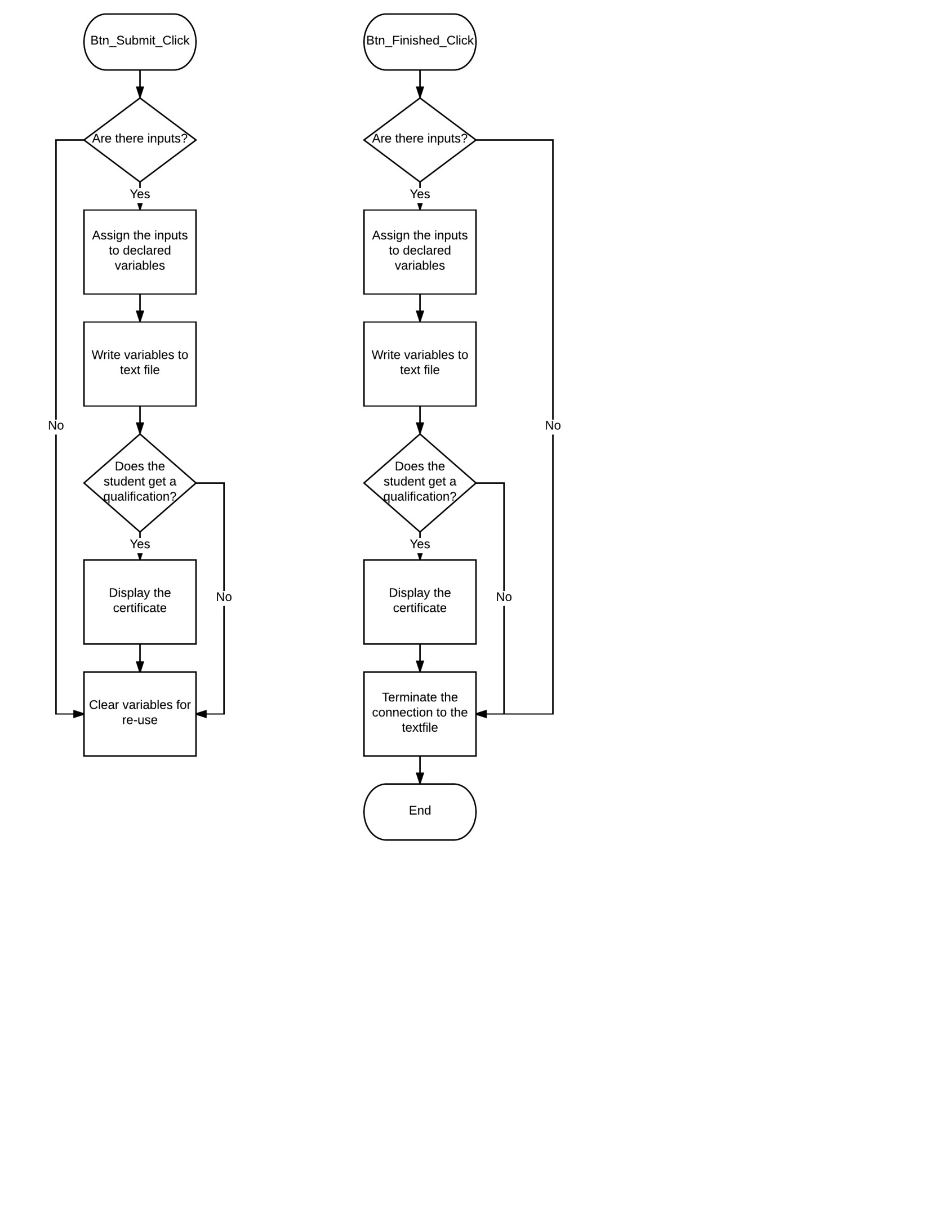
The Grade calculator enables the user to input a student’s details (name, DOB, ID number) and the results of their Unit 1, 2 and 3 assessments, and generate a certificate if the results are a Pass or above overall.

## Flowcharts:

### General:



### Event Specific:



## Annotated Code:

### Display Form:

Imports System.IO

Public Class Frm\_Scores

Dim Int\_u1 As Integer 'Variables for storing the Unit marks.

Dim Int\_u2 As Integer

Dim int\_u3 As Integer

Structure studentinfo

Dim str\_name As String

Dim str\_DOB As String

Dim str\_number As String

Dim int\_U1Marks As Integer

Dim int\_U2Marks As Integer

Dim int\_U3Marks As Integer

Dim int\_TMarks As Integer

End Structure

Public stinfo(1) As studentinfo

Private Sub btn\_Submit\_Click(sender As Object, e As EventArgs) Handles btn\_Submit.Click 'This sub writes the information to file.

If txt\_EnterName.Text = "" Or txt\_EnterStudentNumber.Text = "" Then

MsgBox("Please input valid information. Try again")

GoTo 1

End If

Dim str\_file\_name As String = "N:\SavingTestFolder\SavingTestFile1.txt"

Dim str\_txtfile = My.Computer.FileSystem.OpenTextFileWriter(str\_file\_name, True)

stinfo(1).str\_name = txt\_EnterName.Text

stinfo(1).str\_DOB = txt\_DOBDay.Text & "/" & txt\_DOBMon.Text & "/" & txt\_DOBYear.Text

stinfo(1).str\_number = txt\_EnterStudentNumber.Text

str\_txtfile.WriteLine(stinfo(1).str\_name) 'str\_txtfile.writeline is used to write a line to the textfile specified in 'txtfile'

str\_txtfile.WriteLine(stinfo(1).str\_DOB)

str\_txtfile.WriteLine(stinfo(1).str\_number)

str\_txtfile.WriteLine("Unit 1: " & stinfo(1).int\_U1Marks \* 10)

str\_txtfile.WriteLine("Unit 2: " & stinfo(1).int\_U2Marks \* 10)

str\_txtfile.WriteLine("Unit 3: " & stinfo(1).int\_U3Marks \* 10)

str\_txtfile.WriteLine("Total: " & stinfo(1).int\_TMarks \* 10)

If stinfo(1).int\_TMarks >= 23 Then

cert.Show()

str\_txtfile.WriteLine("Certificate achieved")

ElseIf stinfo(1).int\_TMarks < 23 Or stinfo(1).int\_U1Marks < 7 Or stinfo(1).int\_U2Marks < 7 Or stinfo(1).int\_U3Marks < 7 Then

str\_txtfile.WriteLine("Qualification Failed")

MsgBox("This student has not achieved the boundaries for certification")

End If

str\_txtfile.WriteLine("--------------------------------------------------------------")

str\_txtfile.Close()

1:

lbl\_U2Grade.Text = "" 'This section of the code clears all the textboxes to make it easier to use again without restarting the program. Allows the user to input the next set of data immediately after the first.

lbl\_U1Grade.Text = ""

lbl\_U3Grade.Text = ""

txt\_U2Marks.Clear()

txt\_U1Marks.Clear()

txt\_U3Marks.Clear()

txt\_EnterName.Clear()

txt\_DOBDay.Clear()

txt\_EnterStudentNumber.Clear()

txt\_DOBMon.Clear()

txt\_DOBYear.Clear()

End Sub

Private Sub Frm\_Scores\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

frm\_StudentDetails.Hide()

End Sub

Private Sub txt\_U1Marks\_TextChanged(sender As Object, e As EventArgs) Handles txt\_U1Marks.TextChanged

Try

stinfo(1).int\_U1Marks = txt\_U1Marks.Text

Int\_u1 = txt\_U1Marks.Text

Catch ex As Exception

End Try

If stinfo(1).int\_U1Marks >= 9 And stinfo(1).int\_U1Marks <= 10 Then 'If states to determine the mark for the specific unit

lbl\_U1Grade.Text = "D"

ElseIf stinfo(1).int\_U1Marks = 8 Then

lbl\_U1Grade.Text = "M"

ElseIf stinfo(1).int\_U1Marks = 7 Then

lbl\_U1Grade.Text = "P"

ElseIf stinfo(1).int\_U1Marks > 10 Or stinfo(1).int\_U1Marks < 0 Then 'Validation to make sure extreme data cannot be inputted into the program and cause further errors

lbl\_U1Grade.Text = "Invalid Marks"

Else

lbl\_U1Grade.Text = "BP"

End If

stinfo(1).int\_TMarks = Int\_u1 + Int\_u2 + int\_u3

If Int\_u1 < 7 Or Int\_u2 < 7 Or int\_u3 < 7 Then 'If statement to determine if all units were passed

stinfo(1).int\_TMarks = 0

End If

If stinfo(1).int\_TMarks > 25 Then

lbl\_OverallGrade.Text = "D"

ElseIf stinfo(1).int\_TMarks > 23 And stinfo(1).int\_TMarks <= 25 Then

lbl\_OverallGrade.Text = "M"

ElseIf stinfo(1).int\_TMarks >= 21 And stinfo(1).int\_TMarks <= 23 Then

lbl\_OverallGrade.Text = "P"

ElseIf stinfo(1).int\_TMarks > 27 Or stinfo(1).int\_TMarks < 0 Then

lbl\_OverallGrade.Text = "Invalid Marks"

Else

lbl\_OverallGrade.Text = "BP"

End If

End Sub 'The next two subs contain identical code, just for checking Unit 2 marks and Unit 3 marks. As such, they will not be annotated as the annotations above also apply to them.

Private Sub txt\_U2Marks\_TextChanged(sender As Object, e As EventArgs) Handles txt\_U2Marks.TextChanged

Try

stinfo(1).int\_U2Marks = txt\_U2Marks.Text

Int\_u2 = txt\_U2Marks.Text

Catch ex As Exception

End Try

If stinfo(1).int\_U2Marks >= 9 And stinfo(1).int\_U2Marks <= 10 Then

lbl\_U2Grade.Text = "D"

ElseIf stinfo(1).int\_U2Marks = 8 Then

lbl\_U2Grade.Text = "M"

ElseIf stinfo(1).int\_U2Marks = 7 Then

lbl\_U2Grade.Text = "P"

ElseIf stinfo(1).int\_U2Marks > 10 Or stinfo(1).int\_U2Marks < 0 Then

lbl\_U2Grade.Text = "Invalid Marks"

Else

lbl\_U2Grade.Text = "BP"

End If

stinfo(1).int\_TMarks = Int\_u1 + Int\_u2 + int\_u3

If Int\_u1 < 7 Or Int\_u2 < 7 Or int\_u3 < 7 Then

stinfo(1).int\_TMarks = 0

End If

If stinfo(1).int\_TMarks > 25 Then

lbl\_OverallGrade.Text = "D"

ElseIf stinfo(1).int\_TMarks > 23 And stinfo(1).int\_TMarks <= 25 Then

lbl\_OverallGrade.Text = "M"

ElseIf stinfo(1).int\_TMarks >= 21 And stinfo(1).int\_TMarks <= 23 Then

lbl\_OverallGrade.Text = "P"

ElseIf stinfo(1).int\_TMarks > 27 Or stinfo(1).int\_TMarks < 0 Then

lbl\_OverallGrade.Text = "Invalid Marks"

Else

lbl\_OverallGrade.Text = "BP"

End If

End Sub

Private Sub txt\_U3Marks\_TextChanged(sender As Object, e As EventArgs) Handles txt\_U3Marks.TextChanged

Try

stinfo(1).int\_U3Marks = txt\_U3Marks.Text

int\_u3 = txt\_U3Marks.Text

Catch ex As Exception

End Try

If stinfo(1).int\_U3Marks >= 9 And stinfo(1).int\_U3Marks <= 10 Then

lbl\_U3Grade.Text = "D"

ElseIf stinfo(1).int\_U3Marks = 8 Then

lbl\_U3Grade.Text = "M"

ElseIf stinfo(1).int\_U3Marks = 7 Then

lbl\_U3Grade.Text = "P"

ElseIf stinfo(1).int\_U3Marks > 10 Or stinfo(1).int\_U3Marks < 0 Then

lbl\_U3Grade.Text = "Invalid Marks"

Else

lbl\_U3Grade.Text = "BP"

End If

stinfo(1).int\_TMarks = Int\_u1 + Int\_u2 + int\_u3

If Int\_u1 < 7 Or Int\_u2 < 7 Or int\_u3 < 7 Then

stinfo(1).int\_TMarks = 0

End If

If stinfo(1).int\_TMarks > 25 Then

lbl\_OverallGrade.Text = "D"

ElseIf stinfo(1).int\_TMarks > 23 And stinfo(1).int\_TMarks <= 25 Then

lbl\_OverallGrade.Text = "M"

ElseIf stinfo(1).int\_TMarks >= 21 And stinfo(1).int\_TMarks <= 23 Then

lbl\_OverallGrade.Text = "P"

ElseIf stinfo(1).int\_TMarks > 27 Or stinfo(1).int\_TMarks < 0 Then

lbl\_OverallGrade.Text = "Invalid Marks"

Else

lbl\_OverallGrade.Text = "BP"

End If

End Sub

Private Sub btn\_Finished\_Click(sender As Object, e As EventArgs) Handles btn\_Finished.Click

'This sub does the same as the btn\_Submit\_Click sub, only this is used when dealing with the last set of data.

Dim str\_file\_name As String = "N:\SavingTestFolder\SavingTestFile1.txt"

Dim str\_txtfile = My.Computer.FileSystem.OpenTextFileWriter(str\_file\_name, True)

'Clicking "Finished" on the form with no input will result in the closing of the textfile connection and the termination of the program

If txt\_EnterName.Text = "" And txt\_EnterStudentNumber.Text = "" Then

str\_txtfile.Close()

End

End If

'Clicking "Finished" on the form with input will result in one last write to textfile, followed by the closing of the connection and termination of program.

stinfo(1).str\_name = txt\_EnterName.Text

stinfo(1).str\_DOB = txt\_DOBDay.Text & "/" & txt\_DOBMon.Text & "/" & txt\_DOBYear.Text

stinfo(1).str\_number = txt\_EnterStudentNumber.Text

str\_txtfile.WriteLine(stinfo(1).str\_name) 'Writes data to file, same as in the btn\_Submit\_click sub.

str\_txtfile.WriteLine(stinfo(1).str\_DOB)

str\_txtfile.WriteLine(stinfo(1).str\_number)

str\_txtfile.WriteLine("Unit 1: " & stinfo(1).int\_U1Marks \* 10)

str\_txtfile.WriteLine("Unit 2: " & stinfo(1).int\_U2Marks \* 10)

str\_txtfile.WriteLine("Unit 3: " & stinfo(1).int\_U3Marks \* 10)

str\_txtfile.WriteLine("Total:" & stinfo(1).int\_TMarks \* 10)

If stinfo(1).int\_TMarks >= 23 Then

cert.Show()

str\_txtfile.WriteLine("Certificate achieved")

Else

str\_txtfile.WriteLine("Qualification Failed")

MsgBox("This student has not achieved the boundaries for certification")

End If

str\_txtfile.WriteLine("--------------------------------------------------------------")

str\_txtfile.Close() 'Closes the connection to the textfile and terminates the program

End

End Sub

End Class

### Certificate Form:

Public Class cert

Private Sub cert\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

'Displays Student Information entered earlier in a certificate format if the conditions were met earlier.

lbl\_DisplayDOB.Text = Frm\_Scores.txt\_DOBDay.Text & "/" & Frm\_Scores.txt\_DOBMon.Text & "/" & Frm\_Scores.txt\_DOBYear.Text

lbl\_DisplayIDNumber.Text = Frm\_Scores.txt\_EnterStudentNumber.Text

lbl\_DisplayName.Text = Frm\_Scores.txt\_EnterName.Text

lbl\_Unit1.Text = Frm\_Scores.txt\_U1Marks.Text & " \* 10 Credits = " & Frm\_Scores.stinfo(1).int\_U1Marks \* 10 & " | Grade : " & Frm\_Scores.lbl\_U1Grade.Text

lbl\_Unit2.Text = Frm\_Scores.txt\_U2Marks.Text & " \* 10 Credits = " & Frm\_Scores.stinfo(1).int\_U2Marks \* 10 & " | Grade : " & Frm\_Scores.lbl\_U2Grade.Text

lbl\_Unit3.Text = Frm\_Scores.txt\_U3Marks.Text & " \* 10 Credits = " & Frm\_Scores.stinfo(1).int\_U3Marks \* 10 & " | Grade : " & Frm\_Scores.lbl\_U3Grade.Text

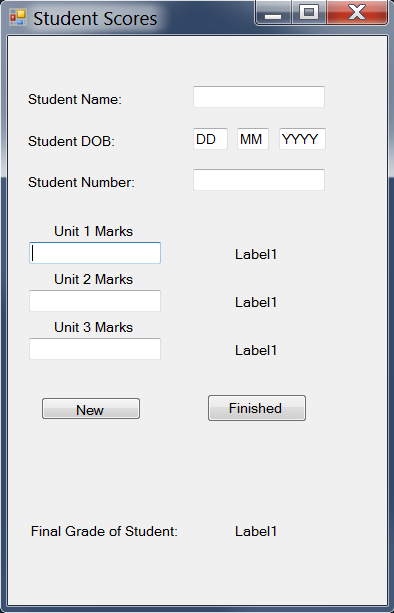
lbl\_DisplayOverall.Text = Frm\_Scores.lbl\_OverallGrade.Text

End Sub

End Class

### Variable List:

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Variable Type** | **Other Comments** |
| Int\_u1 | Integer | Holds the Unit 1 score, used in calculating grades |
| Int\_u2 | Integer | Holds the Unit 2 score, used in calculating grades |
| Int\_u3 | Integer | Holds the Unit 3 score, used in calculating grades |
| Stinfo | Studentinfo (Structure) | Creates a data structure to hold a number of variables together. |
| Str\_name | String | Part of the studentinfo structure |
| Str\_DOB | String | Part of the studentinfo structure |
| Str\_number | String | Part of the studentinfo |
| Int\_U1Marks | Integer | Holds the Unit 1 scores for displaying. |
| Int\_U2Marks | Integer | Holds the Unit 2 scores for displaying |
| Int\_U3Marks | Integer | Holds the Unit 3 scores for displaying |
| Int\_TMarks | Integer | Holds the total scores for displaying |
| Str\_file\_name | String | Holds the file path for the text file |
| Str\_txtfile | My.Computer.FileSystem.OpenTextFileWriter | Enables writing to text file |



Displays final grade. If this is BP, the student does not receive a certificate. If the Grade is P or above when “New” or “Finished” is pressed, a certificate will appear

“Unit 3” Input for marks, followed by a label that changes to reflect the grade depending on the scores

“Unit 2” Input for marks, followed by a label that changes to reflect the grade depending on the scores

“Unit 1” Input for marks, followed by a label that changes to reflect the grade depending on the scores

Button used to write data to file and terminate the connection

Button used to write the data to file and refresh the form for the next data entry

Student’s KC number

Student’s Date of Birth: Day/Month/Year

Student’s full name

## Comments:

* The program currently runs an extra “Form1.vb” that isn’t being used. This is because I changed a big chunk of code so that it was more user friendly and fit the project description better, and the Form was left over. It currently does nothing, and so shouldn’t affect the ability of the program.
* The program works by letting the user input information about the student and creating the certificate (if applicable) and then clearing to be used again if the user desires to create another entry. If not, the program will just write the data and then exit. This means that data doesn’t need to be stored for a long period of time, and multiple entries don’t need to be stored and written at once. Instead, the entries are written after they are submitted by the user, so only one set of data has to be held by the program at any one time.

## Test Plan:

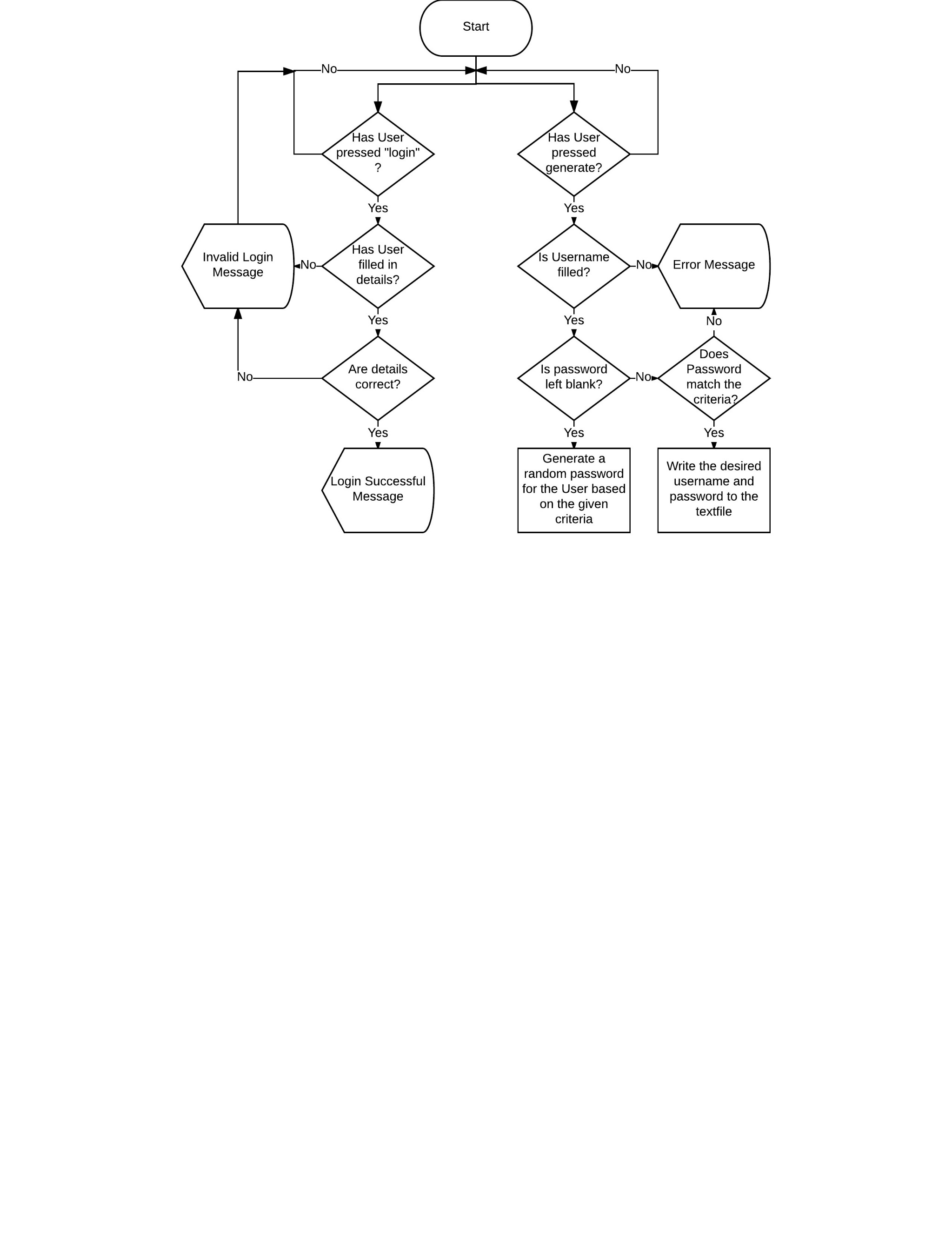
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test | Description | Input | Expected Outcome | Actual Outcome | Other Comments |
| #1 | Pressing “Finished” with no input |  | Program closes | As expected |  |
| #2 | Pressing “New” with no input |  | “ Invalid Inputs” message displayed, nothing written to file | As expected |  |
| #3 | Invalid mark input | 90 | Displays invalid marks & BP for final grade | Displays invalid marks but does not register as BP | The way to fix this is to force the corresponding marks to format to 0 if they are invalid (>10 or <0) |
| #4 | Invalid mark input | -3 | Displays invalid marks & BP for final grade | As expected |  |
| #5 | Invalid mark input | SSS | Displays invalid marks & BP for final grade | Displays BP for unit and final grade | Can easily be fixed with an extra if statement to distinguish string entries from integers |
| #6 | Missing Student Name or Student Number |  | Displays “Invalid Inputs” message and asks to correct | As expected |  |

# Password Generator & Checker

## Summary:

The Password program will perform a number of tasks. First of all, it serves as validation. Any username and password entered by the user will be checked against a word document to see if the two can be matched. Both entries, the username and the password have to be correct for the program to return a message saying that the details were found. Secondly, the program allows the user to create a new data entry. Anything (within the parameters i.e. 8 characters, 3 or more of the following: one upper case, one lower case, one special, one integer) contained in the username and password fields will be written to the file, ready to be validated against. Last but not least, if the password is left blank and generate is clicked, the program will generate a password based on the criteria above and write it to file.

## Flowcharts:



## Annotated Code:

### Form1 (Login):

Imports System.IO

Public Class PasswordEnter

Structure loaded

'this structure deals with the loaded username and password. Used in the validation

Dim lpass As String

Dim luser As String

End Structure

Dim loadedinfo(1) As loaded

Dim str\_passinput As String

Dim int\_passleng As Integer 'length of password

Dim int\_passlc As Integer = 0 'amount of lowercase characters in password

Dim int\_passuc As Integer = 0 'amount of uppercase characters in password

Dim int\_passin As Integer = 0 'amount of integer characters in password

Dim int\_passsp As Integer = 0 'amount of special characters in password

Dim str\_temp As String

Dim int\_count As Integer = 0 'Used in loops

Dim str\_userinput As String

Public file\_name As String = "N:\SavingTestFolder\PasswordFile1.txt" 'File path and name of the textfile the reading and writing will be taking place.

Dim bln\_usercorrect As Boolean 'Used in the validation. Returns true if the entered username matches the stored username

Dim bln\_passcorrect As Boolean 'Used in the validation. Returns true if the entered password matches the stored password

Dim filelength As Integer 'Used in the reading of the file to work out the length of the file.

Private Sub btn\_Submit\_Click(sender As Object, e As EventArgs) Handles btn\_Submit.Click

'Submit\_Click is the sub responsible for the Login button. This sub reads from the file and checks the entered information against the stored information and returns the results.

int\_count = 0

If System.IO.File.Exists(file\_name) = True Then 'Making sure the file is present.

Dim objcounter As New System.IO.StreamReader(file\_name) 'Reads the file to work out how large the file is.

Do While objcounter.Peek() <> -1

filelength = filelength + 1

objcounter.ReadLine()

Loop

objcounter.Close()

Dim objreader As New System.IO.StreamReader(file\_name)

ReDim loadedinfo(filelength) 'Re-declared the size of (loadedinfo) so that the entirety of the file can be stored in the structure, without needing a pre-set size.

Do While objreader.Peek() <> -1 'Reads the contents of the file, two lines at a time. The first line will always be the username, the second the password.

int\_count = int\_count + 1

loadedinfo(int\_count).luser = objreader.ReadLine

loadedinfo(int\_count).lpass = objreader.ReadLine

Loop

objreader.Close()

ElseIf System.IO.File.Exists(file\_name) = False Then 'If file is not present, file is created at file\_name

System.IO.File.Create(file\_name)

Dim objcounter As New System.IO.StreamReader(file\_name)

Do While objcounter.Peek() <> -1

filelength = filelength + 1

objcounter.ReadLine()

Loop

objcounter.Close()

Dim objreader As New System.IO.StreamReader(file\_name)

ReDim loadedinfo(filelength)

Do While objreader.Peek() <> -1

int\_count = int\_count + 1

loadedinfo(int\_count).luser = objreader.ReadLine

loadedinfo(int\_count).lpass = objreader.ReadLine

Loop

objreader.Close()

End If

str\_userinput = txt\_UnameEn.Text

For i = 1 To int\_count

If loadedinfo(i).luser = str\_userinput Then 'each entry in the structure is checked against the input and if the two match, bln\_usercorrect changes to True.

bln\_usercorrect = True

MsgBox("User Correct")

i = int\_count

Else : bln\_usercorrect = False

End If

Next

str\_passinput = txt\_PasswordEn.Text

If bln\_usercorrect = True Then 'If the entered username matched one of the stored ones, the passwords are then checked the same way. If the usernames were not the same, password checking is unnecessary.

For i = 1 To int\_count

If loadedinfo(i).lpass = str\_passinput Then 'Stored passwords are checked against the entered password and if two match, bln\_passcorrect is made True.

bln\_passcorrect = True

MsgBox("Pass correct")

i = int\_count

Else : bln\_passcorrect = False

End If

Next

End If

If bln\_usercorrect = True And bln\_passcorrect = True Then 'If both validations are successful, this message is displayed

MsgBox("Login Successful")

End If

If bln\_usercorrect = False Or bln\_passcorrect = False Then 'If either validation fails, this message is displayed

MsgBox("Details not found. Please check the inputs for mistakes, or alternatively please create the account using 'Generate'")

End If

txt\_PasswordEn.Clear()

txt\_PasswordEn.Refresh()

End Sub

Private Sub PasswordEnter\_Load(sender As Object, e As EventArgs) Handles MyBase.Load 'A help message available when the program starts and when "Help" is clicked

MsgBox("If you would like to login to your account, please input the correct username and password details and click on Login. Otherwise, please input the desired Username and Password and click generate, and the account will be created. If you would like to have a password generated for you, please leave the password field completely blank.")

End Sub

Private Sub btn\_help\_Click(sender As Object, e As EventArgs) Handles btn\_help.Click

MsgBox("If you would like to login to your account, please input the correct username and password details and click on Login. Otherwise, please input the desired Username and Password and click generate, and the account will be created. If you would like to have a password generated for you, please leave the password field completely blank.")

End Sub

Private Sub btn\_Create\_Click(sender As Object, e As EventArgs) Handles btn\_Create.Click

'This sub handles the "Generate" button and the writing to file of User details and random generation of a password.

str\_temp = ""

If txt\_PasswordEn.Text = "" Or txt\_PasswordEn.Text.ToLower = "generate" Then 'Unless the password field is blank or has the word generate in it, the program will atstr\_tempt to write the string in the password enter field to the textfile.

My.Forms.Form2.Show()

Else

int\_passleng = txt\_PasswordEn.TextLength

If int\_passleng < 8 Then 'Checks string length

MsgBox("Password too short. 8 characters or more only")

GoTo 1

End If

int\_count = 0

Do

str\_temp = txt\_PasswordEn.Text(int\_count)

If 64 < Asc(str\_temp) And Asc(str\_temp) < 91 Then 'Checks string's Uppercase count

int\_passuc = int\_passuc + 1

ElseIf 96 < Asc(str\_temp) And Asc(str\_temp) < 123 Then 'Checks string's Lowercase count

int\_passlc = int\_passlc + 1

ElseIf 31 < Asc(str\_temp) And Asc(str\_temp) < 48 Or 57 < Asc(str\_temp) And Asc(str\_temp) < 65 Or 90 < Asc(str\_temp) And Asc(str\_temp) < 97 Or 122 < Asc(str\_temp) And Asc(str\_temp) < 127 Then 'Checks string's special count

int\_passsp = int\_passsp + 1

ElseIf 47 < Asc(str\_temp) And Asc(str\_temp) < 58 Then 'Checks string's Integer count

int\_passin = int\_passin + 1

End If

int\_count = int\_count + 1

Loop Until int\_count = int\_passleng

MessageBox.Show("Uppercase: " & int\_passuc & Environment.NewLine & "Lowercase: " & int\_passlc & Environment.NewLine & "Numbers: " & int\_passin & Environment.NewLine & "Specials: " & int\_passsp)

If int\_passuc > 0 Then

int\_passuc = 1

Else

int\_passuc = 0

End If

If int\_passlc > 0 Then

int\_passlc = 1

Else

int\_passlc = 0

End If

If int\_passsp > 0 Then

int\_passsp = 1

Else

int\_passsp = 0

End If

If int\_passin > 0 Then

int\_passin = 1

Else

int\_passin = 0

End If

If int\_passuc + int\_passlc + int\_passsp + int\_passin < 3 Then 'At least 3 of the character categories have to be present. The above code checks this.

MessageBox.Show("Please use characters from at least 3 of these categories: " & Environment.NewLine & "Uppercase" & Environment.NewLine & "Lowercase" & Environment.NewLine & "Numbers" & Environment.NewLine & "Special Characters such as @")

Else

If System.IO.File.Exists(file\_name) = True Then

'If the validation of the new password is successful,

Dim objwriter As New System.IO.StreamWriter(file\_name, True)

MsgBox(txt\_PasswordEn.Text)

MsgBox(txt\_UnameEn.Text)

objwriter.WriteLine(txt\_UnameEn.Text)

objwriter.WriteLine(txt\_PasswordEn.Text)

objwriter.Close()

End If

End If

End If

1:

txt\_PasswordEn.Clear()

txt\_PasswordEn.Refresh()

End Sub

End Class

### Form 2 (Generation):

Public Class Form2

Private Sub btn\_Generate\_Click(sender As Object, e As EventArgs) Handles btn\_Generate.Click

'This form mainly deals with random generation of the password.

If PasswordEnter.txt\_UnameEn.Text = "" Then

MsgBox("Username was left blank. Please restart the program and enter a Username before going into the generate password screen")

End If

Dim str\_firstname As String

Dim bln\_fname As Boolean

Dim str\_surname As String

Dim int\_count As Integer

Dim bln\_sname As Boolean

Dim str\_keyword As String

Dim bln\_kword As Boolean

Dim int\_passlen As Integer

Dim rnd As New Random

Dim rnd\_int As Integer

Dim rnd\_ucase As Integer

Dim rnd\_sp As Integer

Dim rnd\_lcase As Integer

Dim str\_passgen As String = ""

Dim temp As Integer

Dim tem As String = ""

str\_firstname = txt\_firstname.Text

If str\_firstname <> "" Then

bln\_fname = True

End If

str\_surname = txt\_surname.Text

If str\_surname <> "" Then

bln\_sname = True

End If

str\_keyword = txt\_optional.Text

If str\_keyword <> "" Then

bln\_kword = True

End If

int\_passlen = rnd.Next(8, 25)

int\_count = 0

Do

Do

If int\_count Mod 2 = 0 Then

rnd\_int = rnd.Next(0, 2)

End If

rnd\_lcase = rnd.Next(0, 2)

rnd\_ucase = rnd.Next(0, 2)

If int\_count Mod 4 = 0 Then

rnd\_sp = rnd.Next(0, 2)

End If

Loop Until rnd\_int + rnd\_lcase + rnd\_ucase + rnd\_sp >= 3

If rnd\_int = 1 Then 'Random Integer character generation

int\_count = int\_count + 1

If bln\_fname = True Then

rnd\_int = rnd.Next(0, Len(str\_firstname))

rnd\_int = Asc(str\_firstname(rnd\_int))

rnd\_int = rnd\_int + rnd.Next(0, 3)

rnd\_int = rnd\_int Mod 10

temp = temp + rnd\_int

End If

If bln\_sname = True Then

rnd\_int = rnd.Next(0, Len(str\_surname))

rnd\_int = Asc(str\_surname(rnd\_int))

rnd\_int = rnd\_int + rnd.Next(0, 3)

rnd\_int = rnd\_int Mod 10

temp = temp + rnd\_int

End If

If bln\_kword = True Then

rnd\_int = rnd.Next(0, Len(str\_keyword))

rnd\_int = Asc(str\_keyword(rnd\_int))

rnd\_int = rnd\_int + rnd.Next(0, 3)

rnd\_int = rnd\_int Mod 10

temp = temp + rnd\_int

End If

If bln\_fname = False And bln\_kword = False And bln\_sname = False Then

temp = rnd.Next(0, 9)

End If

temp = temp Mod 10

str\_passgen = str\_passgen + temp.ToString

End If

If rnd\_sp = 1 Then 'random special character generation

int\_count = int\_count + 1

Select Case rnd.Next(0, 3)

Case 0

rnd\_sp = rnd.Next(0, 14)

rnd\_sp = 33 + rnd\_sp

tem = Convert.ToChar(rnd\_sp)

Case 1

rnd\_sp = rnd.Next(0, 7)

rnd\_sp = 58 + rnd\_sp

tem = Convert.ToChar(rnd\_sp)

Case 2

rnd\_sp = rnd.Next(0, 6)

rnd\_sp = 91 + rnd\_sp

tem = Convert.ToChar(rnd\_sp)

End Select

str\_passgen = str\_passgen + tem

End If

If rnd\_lcase = 1 Then 'random lowercase character generation

int\_count = int\_count + 1

rnd\_lcase = rnd.Next(0, 26)

rnd\_lcase = rnd\_lcase + 97

str\_passgen = str\_passgen + Convert.ToChar(rnd\_lcase)

End If

If rnd\_ucase = 1 Then 'random uppercase character generation

int\_count = int\_count + 1

rnd\_ucase = rnd.Next(0, 26)

rnd\_ucase = rnd\_ucase + 65

tem = rnd\_ucase.ToString

str\_passgen = str\_passgen + Convert.ToChar(rnd\_ucase)

End If

Loop Until int\_count >= int\_passlen

If System.IO.File.Exists(PasswordEnter.file\_name) = True Then 'Once the length of the password has been met, the username and password are written to file and displayed for the user.

Dim objwriter As New System.IO.StreamWriter(PasswordEnter.file\_name, True)

objwriter.WriteLine(PasswordEnter.txt\_UnameEn.Text)

objwriter.WriteLine(str\_passgen)

objwriter.Close()

MsgBox("Password Generated! " & Environment.NewLine & "Username is: " & PasswordEnter.txt\_UnameEn.Text & Environment.NewLine & "Generated Password is: " & str\_passgen)

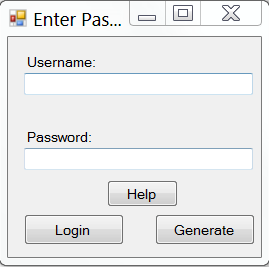
End If

End Sub

### End Class

### Variables List:

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Variable Type** | **Other Comments** |
| Loadedinfo | Loaded (structure) | Handles the data read from the text file |
| Str\_Passinput | String | The entered password |
| Int\_Passleng | Integer | The length of the entered password |
| Int\_Passlc | Integer | The Lowercase character count of the entered password |
| Int\_Passuc | Integer | The Uppercase character count of the entered password |
| Passin | Integer | The Integer character count of the entered password |
| Passsp | Integer | The Special character count of the entered password |
| temp | string |  |
| Int\_count | Integer | Used to limit loops to a necessary length |
| Str\_userinput | String | Handles the username input |
| File\_name | String | Handles the filepath and name of the textfile |
| Bln\_usercorrect | Boolean | Boolean value used for checking if the Username Validation was successful |
| Bln\_passcorrect | Boolean | Boolean value used for checking if the Password validation was successful |
| Filelength | Integer | Handles counting the length of the file, used in creating an array structure |
| Objcounter | System.Io.StreamReader | Enables reading of the file & its contents |
| Objreader | System.IO.StreamReader | Enabled reading of the file & its contents |
| Str\_firstname | String | Handles the input for first name |
| Bln\_fname | Boolean | Boolean value dependent on if the first name is present |
| Str\_surname | String | Handles the input for surname |
| Bln\_sname | Boolean | Boolean value dependent on if the surname is present |
| Str\_keyword | String | Handles the input for an optional keyword |
| Bln\_kword | Boolean | Boolean value dependent on if the keyword is present |
| Int\_count | Integer | Used in loops |
| Int\_passlen | Integer | Length of password |
| Rnd | Random | Generates a random number between (x, y) |
| Rnd\_int | Integer | Handles the randomly generated integers |
| Rnd\_ucase | Integer | Handles the randomly generated uppercase |
| Rnd\_lcase | Integer | Handles the randomly generated lowercase |
| Rnd\_sp | Integer | Handles the randomly generated special |
| Str\_passgen | String | Holds the randomly generated password |
| Temp | Integer | Temporary value, used in calculations |
| Tem | String | Temporary value, used in putting the password together |

**

Generates the password if it has been left blank and writes the user details to file

Triggers the validation against stored data

Password entered by the User

Username entered by the User

## Comments:

-The password generated by the program will be completely random, as neither the length nor the amount of each type of characters is pre-determined, only there must be at least 3 kinds of them.

- The complexity of the random generation is quite high. It could be a lot lower, but that would cause the password to be more predictable.

## Test Plan:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test | Description | Input | Expected Outcome | Actual Outcome | Other Comments |
| #1 | No inputs, login button pressed |  | Invalid login message appears | As expected |  |
| #2 | No inputs, generate button pressed |  | Error message telling user to restart the program with Username appears | As expected |  |
| #3 | Partially right login details, login pressed | Username correct, password incorrect | Invalid login message | As expected |  |
| #4 | Partially right login details, login pressed | Username incorrect, password correct | Invalid login message | As expected |  |
| #5 | Invalid login details, login pressed | Username and password incorrect | Invalid login message | As expected |  |
| #6 | Valid login details, login pressed | Username and password correct | Login successful message | As expected |  |
| #7 | Username present, no password, generate pressed | Username, no password | Form 2 opens up, ready for password generation | As expected |  |
| #8 | Username and password present, generate pressed | Password is less than 8 character | Password too short message | As expected |  |
| #9 | Username and password present, generate pressed | Password only contains upper and lowercase | Not enough categories message displayed | As expected |  |
| #9 | Username and password present, generate pressed | Password fits parameters | Details written to file, success message displayed | As expected |  |
| #10 | Form 2 – No keywords, firstname or surname used |  | Randomly generates password & displays it for user, then writes to file. | As expected |  |
| #11 | Form 2- Keywords, firstname and surname present |  | Randomly generates password (affected by keyword, firstname and surname) & displays for user, then writes to file. | As expected |  |
| #12 | Login pressed, file doesn’t exist |  | File created, program continues to run as normal | As expected |  |