How to create Typing Test.

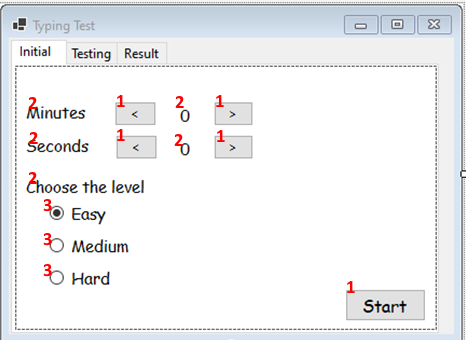
The steps of creating the project «Typing Test», WinForms Framework, C#.

You can get this project from GitHub <https://github.com/LenaPesochek/project/tree/main/TypingTest>

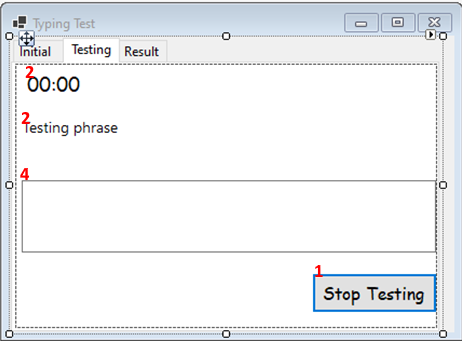
1. You begin with creating a TabControl and placing the necessary elements on the form.

The TabControl consists of three items:

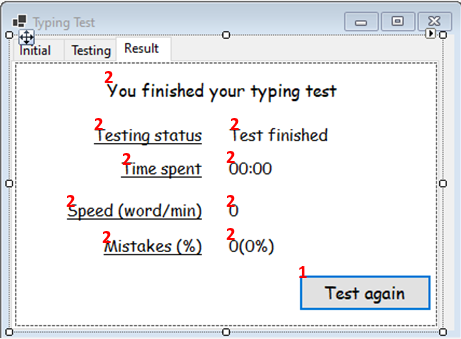
1. Input data



1. Testing



1. Results output



To add the elements to the form you should drag them from the toolbar. The elements used are buttons(1), labels(2), radio buttons(3) and textbox(4).

1. Time setting and switching between the tabs
   1. To set time for testing you should use event “button\_click”. For example, to increase the number of minutes you should write the next method:

private void MinuteIncreasingButton\_Click(object sender, EventArgs e) {

var minuteCount = int.Parse(MinuteLabel.Text);

minuteCount++;

MinuteLabel.Text = minuteCount.ToString();

}

Using the same logic write the methods for the other SetTimeButtons.

* 1. To set the necessary Tab active write the method “SelectTabByName”:

public void SelectTabByName(string name) {

TabControl.SelectTab(name);

}

1. Declaration of the public class “Testing” which implements the next functions:

* Getting a test phrase
* Timer control
* Input data analyzing
  1. The phrases you get for the typing test need to be stored in a separate file. Every line in the file contains difficulty and a phrase according to the mask. For example:

[Easy]Hello, world!

All phrases: <https://github.com/LenaPesochek/project/blob/main/TypingTest/TypingTest/Content/phrases.txt>

* 1. In order to get a test phrase according to the chosen difficulty you should write the next method:

string GetTestPhrase() {

var difficultyInPhrase = "[" + Difficulty.ToString() + "]";

var currentDifficultyPhrases = File.ReadLines(phrasesPath).Where(x => x.Contains(difficultyInPhrase));

var phrase = currentDifficultyPhrases.Skip(new Random().Next(0, currentDifficultyPhrases.Count() - 1)).First();

phrase = phrase.Replace(difficultyInPhrase, "");

return phrase;

}

* 1. Timer control:
     1. Creating a field Timer timer = new Timer();
     2. In a Testing class constructor we add an event handler and set an interval in which the event is raised

public Testing(Form form) {

…;

timer.Tick += new EventHandler(TimerTick);

timer.Interval = 1000;

}

* 1. Then you create the method “start” that runs the timer and displays the test phrase.

public void Start(int startValue, CurrentDifficulty difficulty) {

timerCurrentValue = startValue;

timerStartValue = startValue;

Difficulty = difficulty;

timer.Start();

TestingForm.SetTestingPhraseLabelText(TestPhrase);

}

* 1. To stop testing you should write the method “Stop” that stops the timer. This method will be called if the input phrase matches the test phrase, when pressing the “Stop testing” button, when time is over. Parameter “forced”: Boolean (true, if the method is called by pressing the “Stop testing” button, otherwise false)

public void Stop() {

timer.Stop();

TestingForm.SelectTabByName("ResultTab");

}

* 1. The method “CompareTestPhrase”.

To compare an input phrase to the test phrase you need to transform these phrases to arrays by the method “String.Split”.

public void CompareTestPhrase(string inputPhrase) {

InputPhrase = inputPhrase;

if(InputPhrase == "") {

MistakesCount = 0;

CorrectWordsCount = 0;

return;

}

var inputPhraseArray = GetSplitInputPhrase();

if(InputPhrase == TestPhrase) {

MistakesCount = 0;

CorrectWordsCount = inputPhraseArray.Length;

Stop(false);

return;

}

var testPhraseArray = GetSplitTestPhrase();

var mistakesCount = 0;

for(var i=0; i<inputPhraseArray.Length; i++) {

if(inputPhraseArray[i] != testPhraseArray[i])

mistakesCount++;

}

MistakesCount = mistakesCount;

CorrectWordsCount = inputPhraseArray.Length – MistakesCount;

}

Every time this method is called the value of the property “MistakesCount” will be overwritten. The “MistakesCount” represents a number of entered words which have a mistake or are unfinished.

1. Counting statistics:
   1. A number of mistakes is counted in the method “CompareTestPhrase”
   2. Percentage of mistakes = (the number of mistakes) / (a number of entered words) <https://github.com/LenaPesochek/project/blob/main/TypingTest/TypingTest/Testing.cs#L24>
   3. Time spent on testing = (time for testing) – (time left when the timer stopped) <https://github.com/LenaPesochek/project/blob/main/TypingTest/TypingTest/Testing.cs#L74>
   4. Speed of printing = (a number of correct entered words) / (time spent on testing) <https://github.com/LenaPesochek/project/blob/main/TypingTest/TypingTest/Testing.cs#L81>
   5. Setting of testing status: “Test is finished”, “Test is stopped”, “Time is over” <https://github.com/LenaPesochek/project/blob/main/TypingTest/TypingTest/Testing.cs#L109>
2. To display the results of the typing test you need to write a public method in the class “Form”:

public void FillResultTab(string testStatus, string testTime, string speed, int mistakes, int mistakesPercent) {

TestResultLabel.Text = testStatus;

TestTimeLabel.Text = testTime;

SpeedLabel.Text = speed;

MistakesCountLabel.Text = string.Format("{0} ({1}%)", mistakes, mistakesPercent);

}

This method is called in the method “Stop” in the class “Testing” <https://github.com/LenaPesochek/project/blob/main/TypingTest/TypingTest/Testing.cs#L67>

To start the typing test again it’s necessary to set default values to the form controls properties “text” and “checked” by writing a method “ClearTestingForm” (<https://github.com/LenaPesochek/project/blob/main/TypingTest/TypingTest/Form.cs#L102>) and to set “InitialTab” active:

private void TestingAgainButton\_Click(object sender, EventArgs e) {

SelectTabByName("InitialTab");

ClearTestingForm();

}