|  |  |
| --- | --- |
| HÜ03 | Abgabetermin: 3.2.2023 23:59 |
| Name: Johannes Werner | Aufwand in h: 10-20 |
| Bezüglich den Aufwand, ab einen gewissen Punkt habe ich aufgehört zu zählen | Punkte: |

## Parallele Häufigkeitsanalyse 8(+1+1) Punkte

Erstellen Sie ein Programm zur Häufigkeitsanalyse von Buchstaben in Textdateien.

Implementieren Sie dazu eine grafische Benutzeroberfläche, in welcher der Benutzer ein Verzeichnis (verwenden Sie einen OpenFolderDialog) mit Textdateien auswählt. Jede Textdatei soll dabei einzeln analysiert werden. Sie erkennen, dass sich diese Aufgabenstellung sehr gut eignet, um diese parallel durchzuführen. Somit tun Sie dies auch!

Das Ergebnis der Häufigkeitsanalyse soll anschließend in eine Datei (für jede Datei eine Ergebnisdatei) in einem eigenen Verzeichnis gespeichert werden. Auch dieses Verzeichnis soll vom Benutzer ausgewählt werden können.

In Ihrer Benutzeroberfläche soll der Benutzer informiert werden, welche Dateien bereits fertig analysiert wurden (in einer Textbox).

Zusatzaufgabe 1: Erstellen Sie einen Fortschrittsbalken. Überlegen Sie, wie Sie den Fortschritt möglichst wahrheitsgetreu darstellen können.

Zusatzaufgabe 2: Begrenzen Sie die maximale Anzahl an gleichzeitigen Analysen. Benutzer spezifiziert Anzahl vor Start der Analyse.

Verwenden Sie zum Testen die zur Verfügung gestellten Textdateien.

# Solution idea

Firstly, this homework was no fun for me. I have rewritten the code at least three times, because of the parallel programming and my problems with thread save variables. For this reason, my estimated work time is that high. Nevertheless, I finished my programming approach, could implement nice features, did it on my own and could learn a lot. Thank you for that!

I started working on this project, by designing a first version of the GUI. During this step I already noticed that a lot of my ideas would not work. Next, I tried to write the code. BTW I got the code for the code analysis from a school project from last year. After a few hours of research, I perfected the file open dialogs. After that all the time went into research, bug-fixing and rewriting the async-code. Finally, everything worked and I could optimize and improve the whole project, for example with a partly resizable GUI.

# Code

Form1.cs

using System.Diagnostics.Metrics;

using System.Reflection;

using System.Windows.Forms;

namespace HUE03

{

public partial class Form1 : Form

{

int numberOfCurrentTasks = 0;

int numberOfFiles = 0;

int numberOfAnalyzedFiles = 0;

List<string> queue = new List<string>();

public Form1()

{

InitializeComponent();

}

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

{

OpenFileDialog dialog = new OpenFileDialog();

dialog.Title = "Select files to analyze";

dialog.InitialDirectory = "C:\\";

dialog.Filter = "txt files (\*.txt)|\*.txt";

dialog.Multiselect = true;

if (dialog.ShowDialog() == DialogResult.OK)

{

foreach(string file in dialog.FileNames)

{

this.numberOfFiles++;

if (this.numberOfCurrentTasks < this.nudParallelAnalyses.Value)

{

Interlocked.Increment(ref this.numberOfCurrentTasks);

Task.Run(() => AnalyzeFile(file));

}

else

this.queue.Add(file);

}

UpdateProgressBar();

}

}

public void AnalyzeFile(string file)

{

if (File.Exists(file))

{

string filename = Path.GetFileName(file);

string[] lines = File.ReadAllLines(file);

Dictionary<char, int> frequency = FrequencyAnalysis(lines, filename);

File.WriteAllText(this.tbOutputPath.Text + filename, "frequency analysis for the file '" + filename + "'\n\n");

foreach (KeyValuePair<char, int> character in frequency)

{

File.AppendAllText(this.tbOutputPath.Text + filename, character.Key + ": " + character.Value + "\n");

}

this.numberOfAnalyzedFiles++;

UpdateProgressBar();

Interlocked.Decrement(ref this.numberOfCurrentTasks);

if (this.queue.Count > 0 && this.numberOfCurrentTasks < this.nudParallelAnalyses.Value)

{

Interlocked.Increment(ref this.numberOfCurrentTasks);

string temp = this.queue.ElementAt(0);

this.queue.Remove(temp);

Task.Run(() => AnalyzeFile(temp));

}

}

else

{

this.numberOfFiles--;

UpdateProgressBar();

WriteInformation(String.Format("[ERROR] File with the path '{0}' could not be found!", file));

}

}

public Dictionary<char, int> FrequencyAnalysis(string[] lines, string filename)

{

WriteInformation(String.Format("[INFO] The frequency analysis for the file '{0}' got startet.", filename));

Dictionary<char, int> frequency = new Dictionary<char, int>();

for(int i = 0; i < lines.Length - 1; i++)

{

foreach(char character in lines[i])

{

if (frequency.ContainsKey(character))

frequency[character]++;

else

frequency[character] = 1;

}

}

WriteInformation(String.Format("[INFO] The frequency analysis for the file '{0}' is finished.", filename));

return frequency;

}

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

{

FolderBrowserDialog dialog = new FolderBrowserDialog();

dialog.Description = "Select a folder to save the results";

dialog.UseDescriptionForTitle = true;

dialog.InitialDirectory = "C:\\";

if(dialog.ShowDialog() == DialogResult.OK)

{

this.tbOutputPath.Text = dialog.SelectedPath + "\\";

this.btnLoadFiles.Enabled = true;

}

else

{

this.btnLoadFiles.Enabled = false;

this.tbOutputPath.Text = "Could not load any folder! Try again...";

}

}

private void WriteInformation(string message)

{

var threadParameters = new System.Threading.ThreadStart(delegate { WriteInformationThreadSave(message + Environment.NewLine);}) ;

var thread = new System.Threading.Thread(threadParameters);

thread.Start();

}

private void WriteInformationThreadSave(string message)

{

if (InvokeRequired)

{

this.Invoke(new Action<string>(WriteInformationThreadSave), new object[] { message });

}

else

{

this.tbInformation.Text = message + this.tbInformation.Text;

}

}

private void UpdateProgressBar()

{

if (InvokeRequired)

{

this.Invoke(new Action(UpdateProgressBar));

}

else

{

this.lbProgressBar.Text = this.numberOfAnalyzedFiles + " / " + this.numberOfFiles + " Files got analyzed";

this.pb.Value = int.Parse(Math.Round((double)this.numberOfAnalyzedFiles / this.numberOfFiles \* 100).ToString());

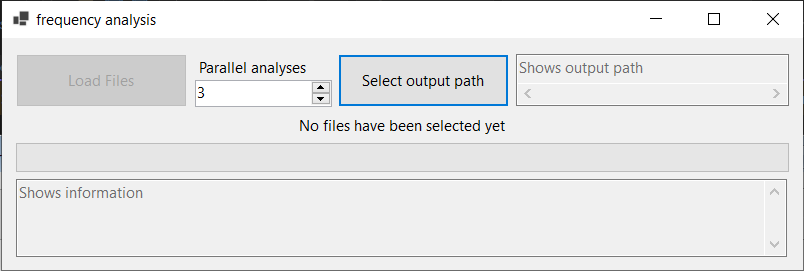
}

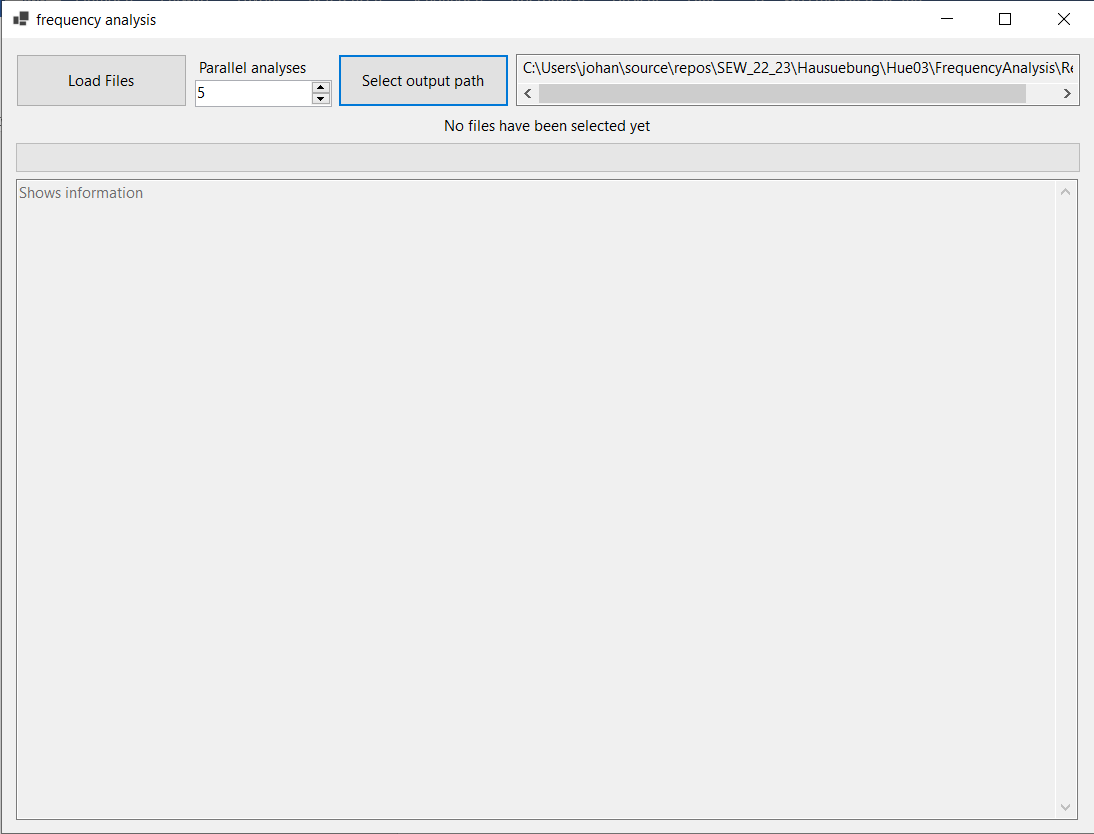
}

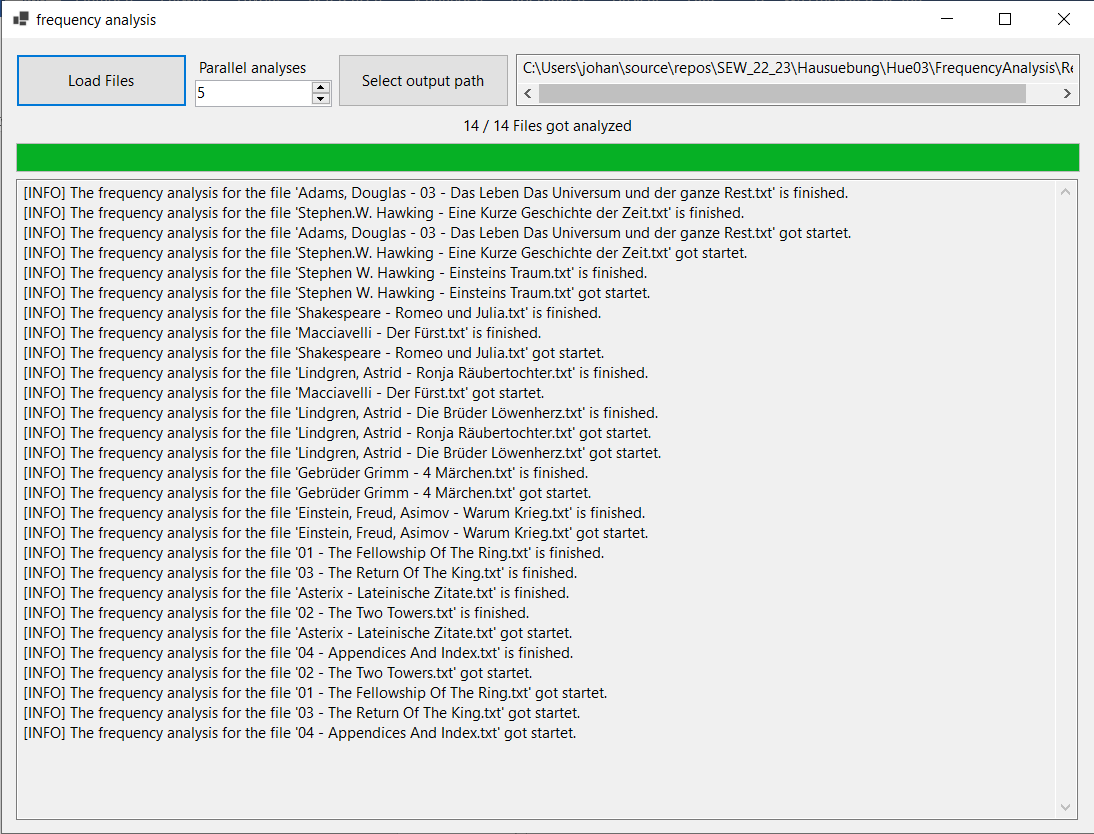
}

}

# Result







On example for a result file.

frequency analysis for the file 'Asterix - Lateinische Zitate.txt'

L: 4

i: 134

s: 139

t: 114

e: 331

: 328

l: 73

a: 102

n: 219

c: 66

h: 99

r: 114

Z: 7

u: 70

d: 116

B: 36

o: 44

m: 31

A: 7

S: 19

T: 5

E: 6

R: 4

I: 52

X: 32

-: 72

ä: 12

w: 24

Ü: 1

b: 22

z: 20

g: 61

f: 31

,: 22

p: 9

V: 21

k: 19

:: 30

W: 22

D: 11

ü: 21

!: 8

G: 15

ö: 12

(: 6

.: 12

): 6

v: 10

’: 3

F: 8

=: 1

C: 4

P: 1

: 3

H: 6

?: 2

O: 3

N: 2

K: 7

M: 3

…: 2

–: 1

j: 1

U: 1