

Dokumentacja Programu Anty Plagiator 3000

Mateusz Adamek, Arkadiusz Datka, Filip Gawel, Szymon Greń

29 stycznia 2020



**Politechnika
Śląska**

Wydział Matematyki Stosowanej
Politechnika Śląska
Rok akademicki 2019/2020

Spis treści

1	Część pierwsza	3
1.1	Opis programu	3
1.2	Instrukcja obsługi	3
2	Część druga	12
2.1	Część techniczna	12
2.2	Opis działania	13
2.3	Pseudokod Odległości Levenshteina	14
3	Implementacja	15
3.1	Uproszczony schemat blokowy programu	15
4	Kod Programu	16
4.1	Program.cs - kod	16
4.2	Program.cs - wyjaśnienia	16
4.3	Form1.cs - kod	16
4.4	Form1.cs - wyjaśnienia	18
4.5	IView.cs - kod	18
4.6	IView.cs - wyjaśnienia	19
4.7	Presenter.cs - kod	19
4.8	Presenter.cs - wyjaśnienia	20
4.9	Model.cs - kod	20
4.10	Model.cs - wyjaśnienia	24
4.11	Algorytm.cs - kod	25
4.12	Algorytm.cs - wyjaśnienia	27
4.13	WZORY.cs - kod	28
4.14	WZORY.cs - wyjaśnienia	32
4.15	Optymalizacja.cs - kod	32
4.16	Optymalizacja.cs - wyjaśnienia	34

1 Część pierwsza

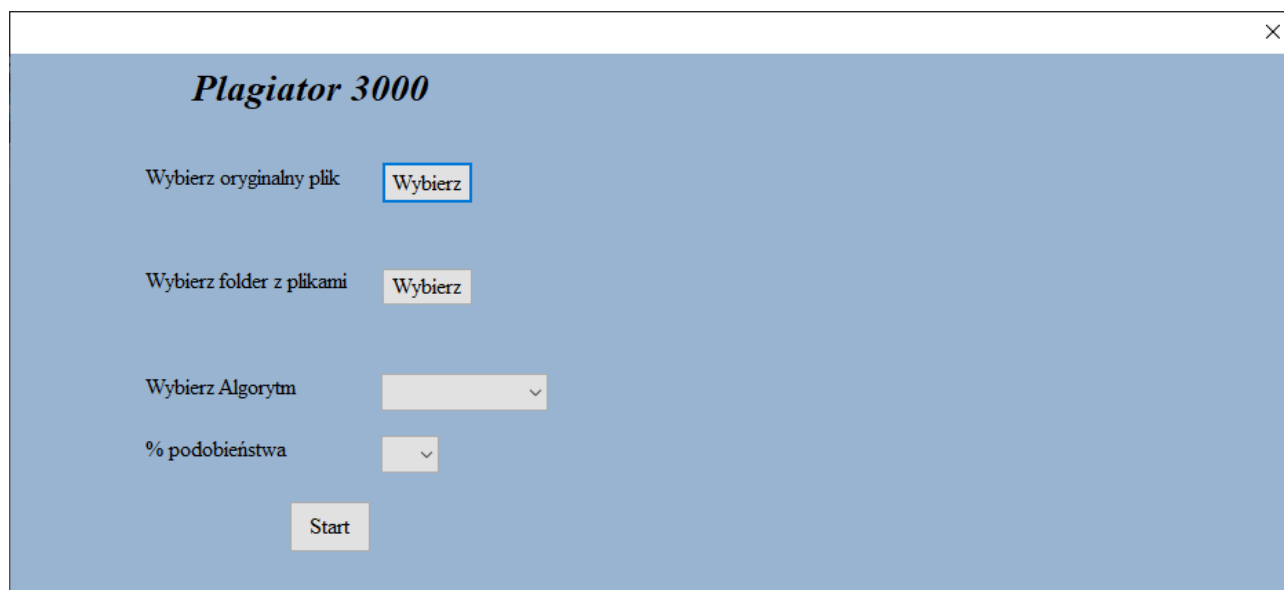
1.1 Opis programu

Program jest realizacją projektu z przedmiotu Inżynieria oprogramowania, polegającym na sprawdzeniu czy w danych dokumentach nie znajdują się splagiatowane wzory matematyczne. Program realizuje to zadanie poprzez wczytanie wybranego pliku .tex oraz folderu z innymi plikami .tex. Wynikiem jest raport mówiący czy plik (lub pliki) zostały splagiatowane.

Projekt został opracowany w języku C# oraz korzystaliśmy z LaTeXa.

1.2 Instrukcja obsługi

Aby uruchomić program, należy uruchomić plik .exe oraz opcjonalnie środowisko programistyczne (Visual Studio). Po uruchomieniu programu wyświetli nam się odpowiednie okno:

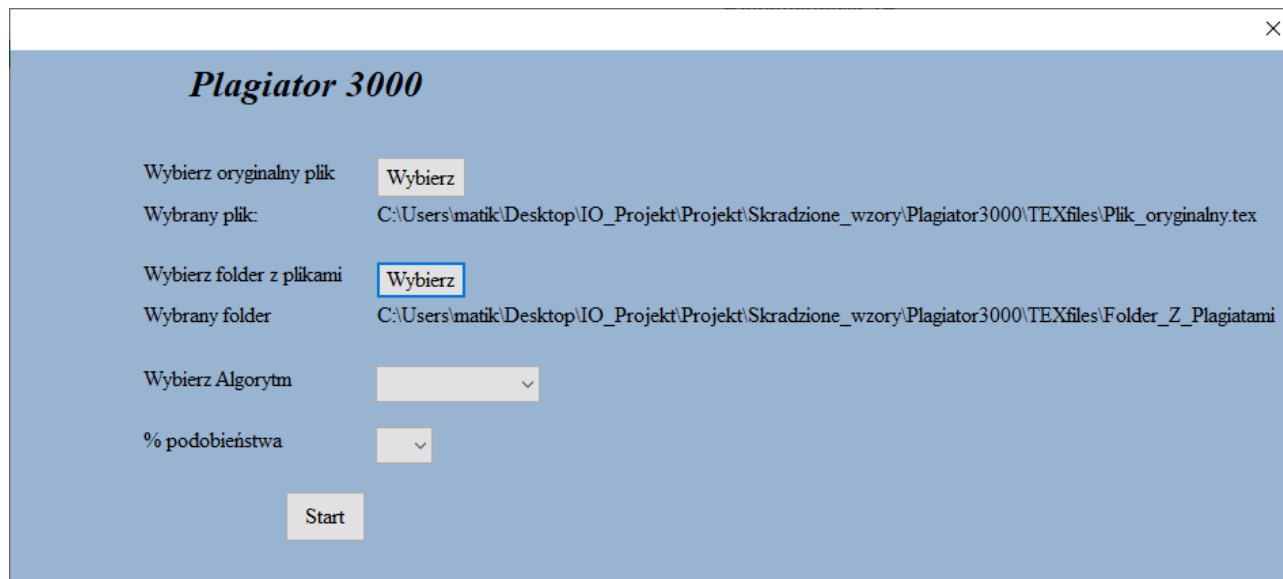


Rysunek 1: Wygląd programu po jego uruchomieniu

Następnie musimy wybrać kolejno:

- Plik podejrzany o plagiat lub oryginał
- Folder z plikami oryginalnymi lub podejrzanymi o plagiat
- Odpowiedni algorytm
- Procent podobieństwa od kiedy uznajemy plik lub pliki za plagiat

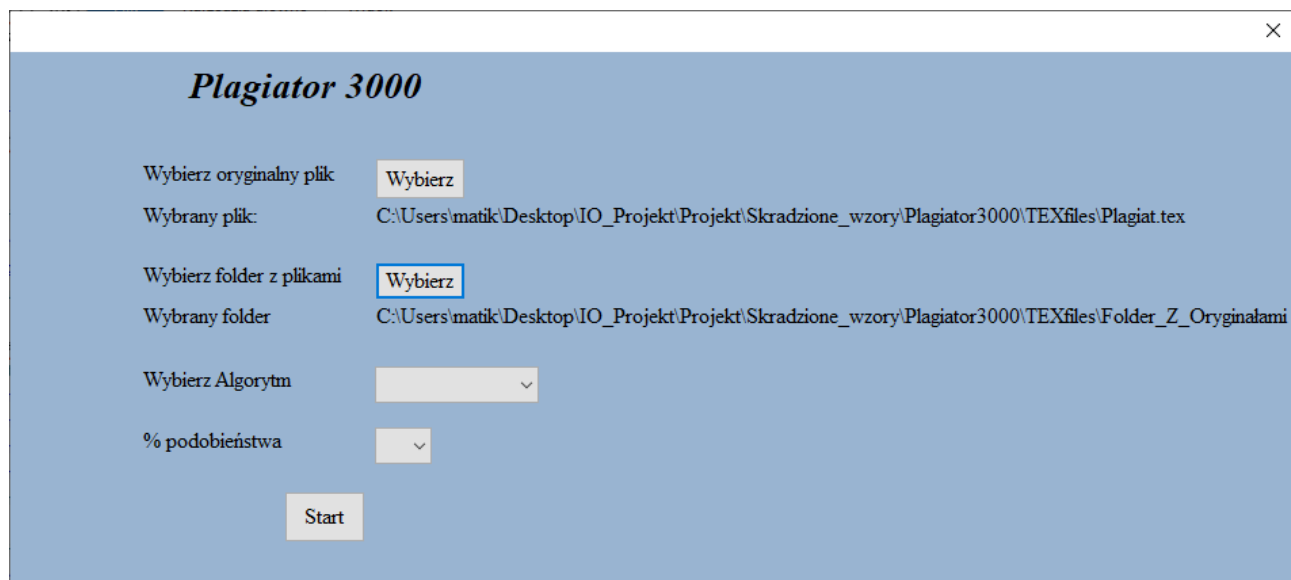
Możemy więc wczytać plik oryginalny i folder z plagiatami, ale również całkowicie odwrotnie. Plik podejrzany o plagiat oraz folder z plikami oryginalnymi (lub jednym plikiem):



The screenshot shows the 'Plagiator 3000' application window. It has a blue background and a title bar with a close button. The window contains the following elements:

- Title:** *Plagiator 3000*
- Wybierz oryginalny plik:** A button labeled 'Wybierz' is next to the label.
- Wybrany plik:** The text 'C:\Users\matik\Desktop\IO_Projekt\Projekt\Skradzone_wzory\Plagiator3000\TEXfiles\Plik_oryginalny.tex' is displayed.
- Wybierz folder z plikami:** A button labeled 'Wybierz' is next to the label.
- Wybrany folder:** The text 'C:\Users\matik\Desktop\IO_Projekt\Projekt\Skradzone_wzory\Plagiator3000\TEXfiles\Folder_Z_Plagiatami' is displayed.
- Wybierz Algorytm:** A dropdown menu with a downward arrow.
- % podobieństwa:** A dropdown menu with a downward arrow.
- Start:** A button at the bottom center.

Rysunek 2: Plik oryginalny oraz folder z plagiatami

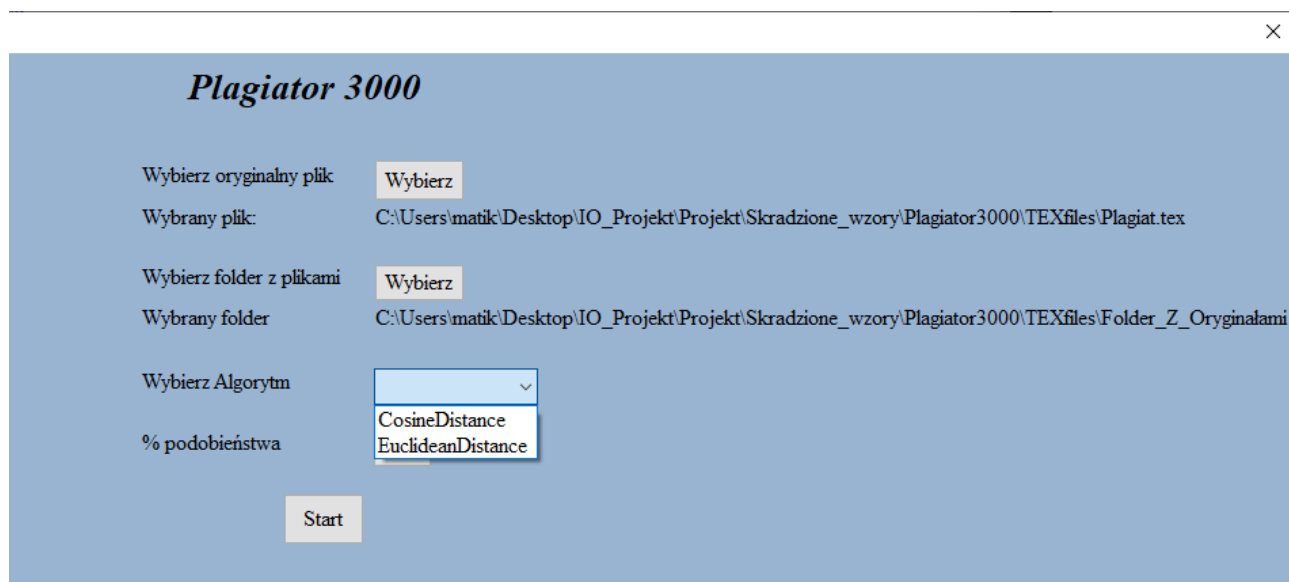


The screenshot shows the 'Plagiator 3000' application window with the same layout as Figure 2, but with different file and folder paths:

- Wybierz oryginalny plik:** A button labeled 'Wybierz' is next to the label.
- Wybrany plik:** The text 'C:\Users\matik\Desktop\IO_Projekt\Projekt\Skradzone_wzory\Plagiator3000\TEXfiles\Plagiat.tex' is displayed.
- Wybierz folder z plikami:** A button labeled 'Wybierz' is next to the label.
- Wybrany folder:** The text 'C:\Users\matik\Desktop\IO_Projekt\Projekt\Skradzone_wzory\Plagiator3000\TEXfiles\Folder_Z_Oryginałami' is displayed.
- Wybierz Algorytm:** A dropdown menu with a downward arrow.
- % podobieństwa:** A dropdown menu with a downward arrow.
- Start:** A button at the bottom center.

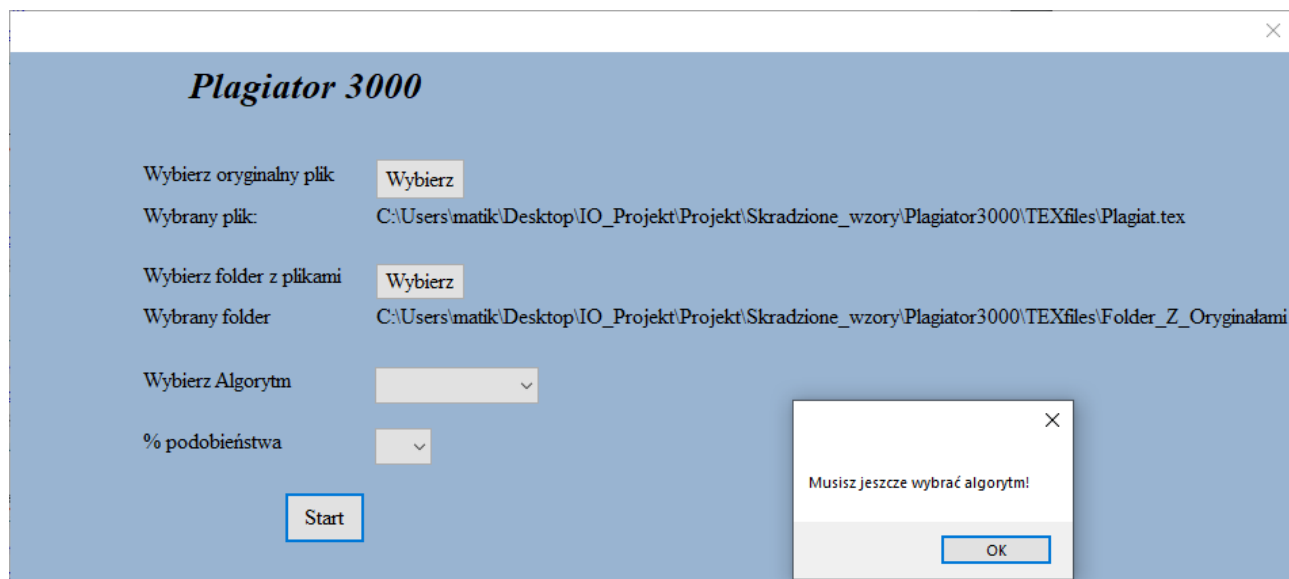
Rysunek 3: Plik podejrzany oraz folder z oryginałami

Wybór algorytmu oraz odpowiedniego procentu podobieństwa znajduje się w rozwijanych listach.



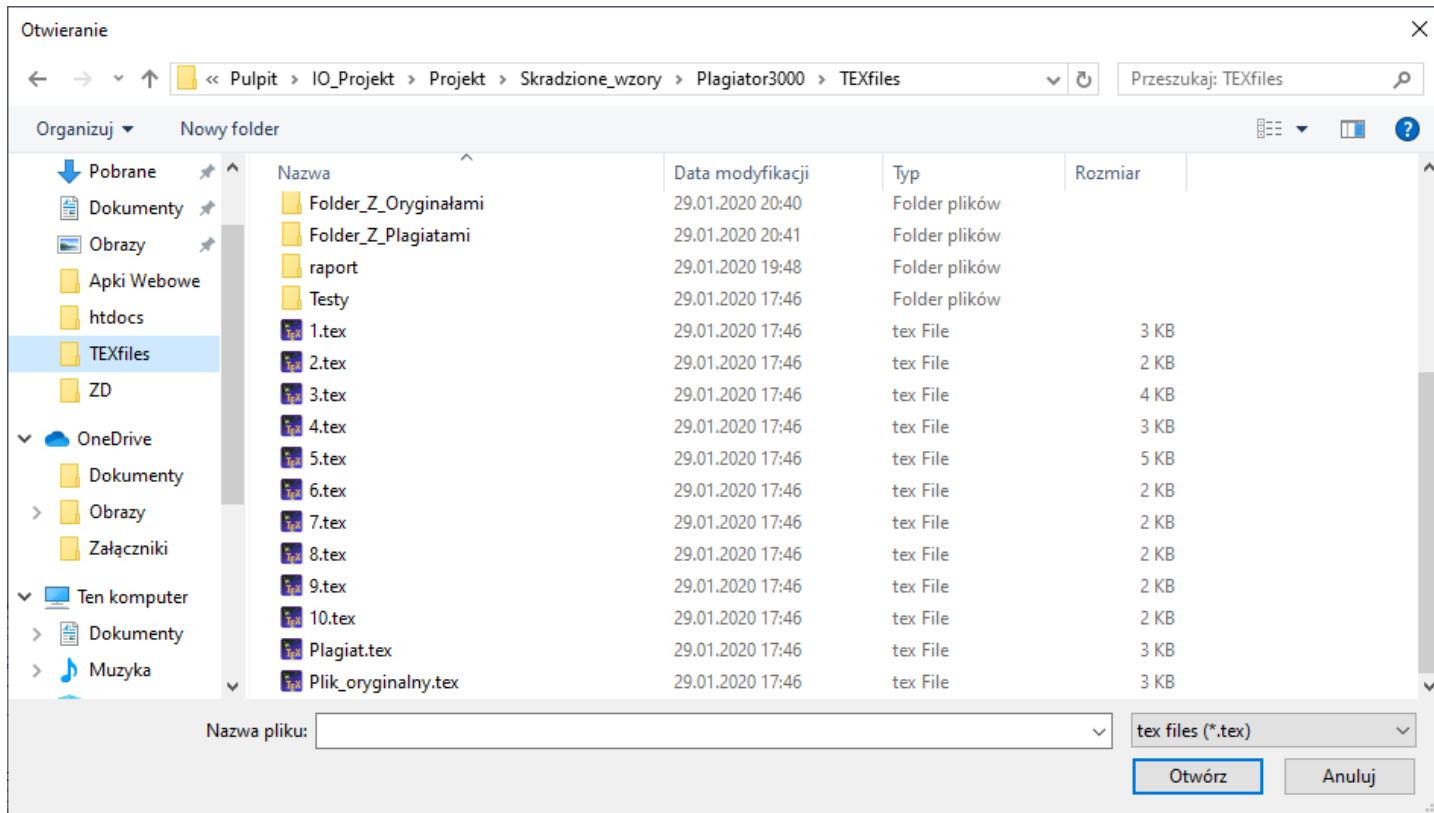
Rysunek 4: Wybór algorytmu

Jeśli nie wybierzemy którejkolwiek opcji zostanie wyświetlony komunikat:



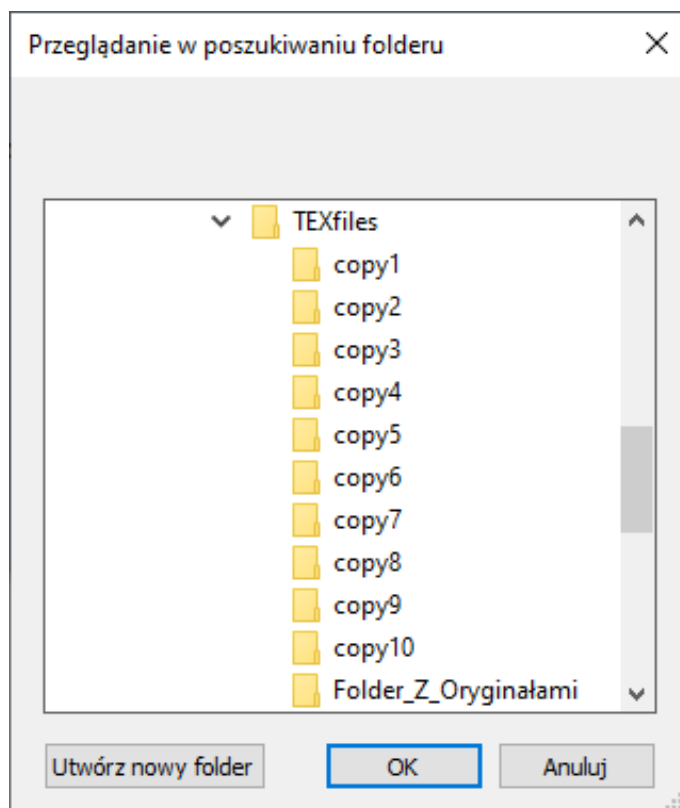
Rysunek 5: Okno informacyjne

Okno do wyboru pliku wygląda w ten sposób (obrazek poniżej). Wyszukujemy jedynie plików .tex, jednak nie jest dla użytkownika zabroniona zmiana na inny rodzaj rozszerzenia. Jednak by program działał poprawnie dokument musi mieć składnię jak pliki TeX.



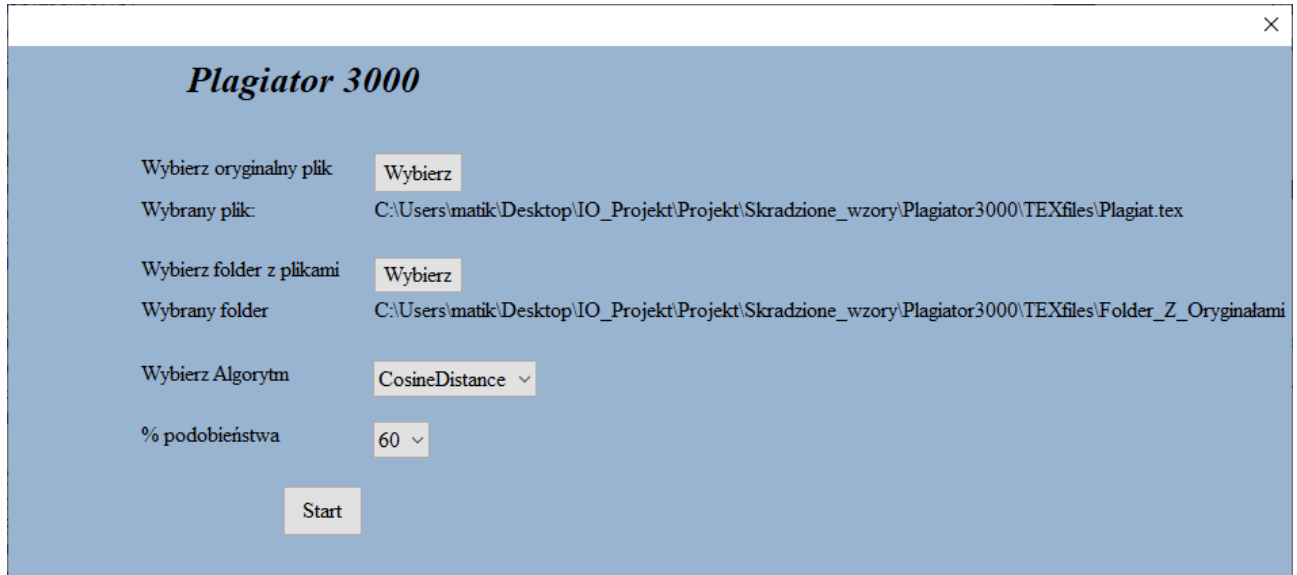
Rysunek 6: Wybieranie pliku

Okno do wyboru folderu jest proste i nie pozwala na zmianę rozszerzeń. To znaczy po wyborze odpowiedniego folderu wyszukiwane są tylko pliki z rozszerzeniem .tex.



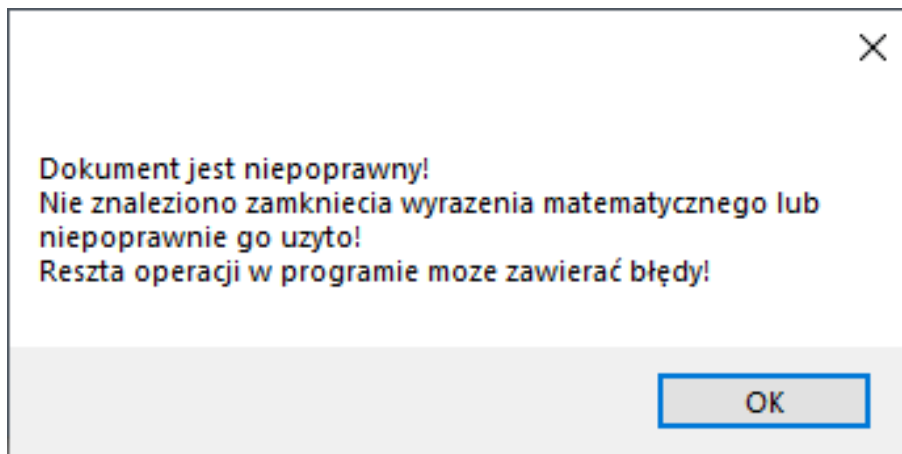
Rysunek 7: Wybieranie folderu

Po wyborze wszystkich elementów uruchamiamy program i następnie czekamy na wygenerowanie raportów.



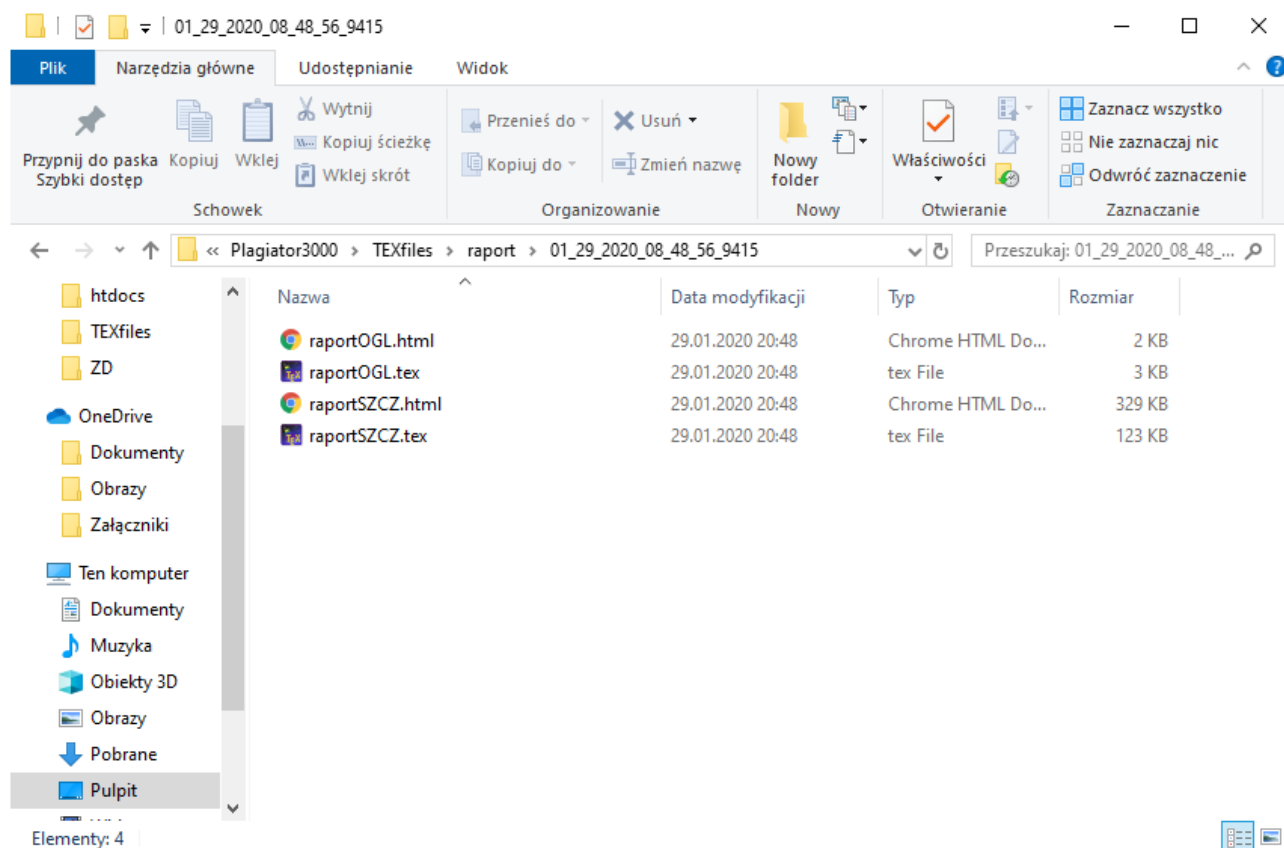
Rysunek 8: Poprawny wybór wszystkich opcji

Jeśli wczytaliśmy dokumenty z błędami program postara się wyciągnąć podobieństwo plików jednak zostaniemy o tym poinformowani, że wyniki mogą być niepoprawne.



Rysunek 9: Okno informujące o złym/złych plikach

Po poprawnym wypełnieniu danych w programie powstają zaś po dwa raporty dla rozszerzeń .tex oraz .html. Raport ogólny zawiera informacje o podobieństwie plików, zaś raport szczegółowy uwzględnia podobieństwo każdego wzoru.



Rysunek 10: Folder z wygenerowanymi raportami

Raport ogólny:

Raport ogólny porównania plików

29 stycznia 2020

Plik bazowy :

C:\Users\matik\Desktop\IO_Projekt\Projekt\Skradzione_wzory\Plagiator3000\TEXfiles\Plagiat.tex

Plik :

C:\Users\matik\Desktop\IO_Projekt\Projekt\Skradzione_wzory\Plagiator3000\TEXfiles\Folder_Z_Orygi

Stopień podobieństwa:

59,2902766473735%

Plik :

C:\Users\matik\Desktop\IO_Projekt\Projekt\Skradzione_wzory\Plagiator3000\TEXfiles\Folder_Z_Orygi

Stopień podobieństwa:

62,2815523746277%

Plik :

C:\Users\matik\Desktop\IO_Projekt\Projekt\Skradzione_wzory\Plagiator3000\TEXfiles\Folder_Z_Orygi

Stopień podobieństwa:

65,3122275101067%

Plik :

C:\Users\matik\Desktop\IO_Projekt\Projekt\Skradzione_wzory\Plagiator3000\TEXfiles\Folder_Z_Orygi

Stopień podobieństwa:

55,3268388469528%

Plik :

C:\Users\matik\Desktop\IO_Projekt\Projekt\Skradzione_wzory\Plagiator3000\TEXfiles\Folder_Z_Orygi

Stopień podobieństwa:

58,6048079465028%

Rysunek 11: Wygląd raportu ogólnego

Raport szczegółowy:

Raport szczegółowy porównania plików

29 stycznia 2020

Plik bazowy :

C:\Users\matik\Desktop\IO_Projekt\Projekt\Skradzone_wzory\Plagiator3000\TEXfiles\Plagiat.tex

Plik :

C:\Users\matik\Desktop\IO_Projekt\Projekt\Skradzone_wzory\Plagiator3000\TEXfiles\Folder_Z_Oryg

Lista podobnych wzorów:

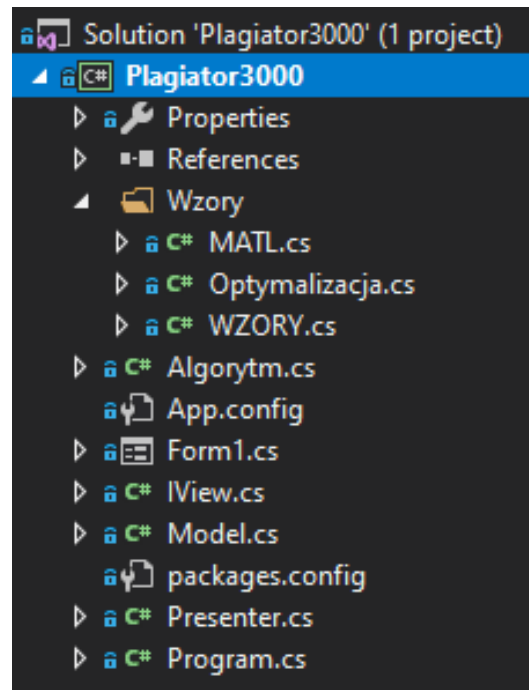
Wzór	Jest podobny do wzoru oryginalnego	Procent podobieństwa
$x^2 = 4$	$x^2 = 4$	100
$h = \frac{a\sqrt{3}}{2}$	$h = \frac{a\sqrt{3}}{2}$	100
$\lim (a_n - b_n) = a - b$	$\lim (a_n - b_n) = a - b$	100
x^{2+a}	x^{2+a}	100
$F(s) = \{Lf\}(s) = \int_0^{\epsilon fty} e^{-st} f(t) dt$	$F(s) = \{Lf\}(s) = \int_0^{\epsilon fty} e^{-st} f(t) dt$	100
$\lim (a_n + b_n) = a + b$	$\lim (a_n + b_n) = a + b$	100
$F(s) = \{Lf\}(s) = \int_0^{\epsilon fty} e^{-st} f(t) dt$	$F(s) = \{Lf\}(s) = \int_0^{\epsilon fty} e^{-st} f(t) dt$	100
$\lim (a_n + b_n) = a + b$	$\lim (a_n + b_n) = a + b$	100
$h = \frac{a\sqrt{3}}{2}$	$h = \frac{a\sqrt{3}}{2}$	100
$F(s) = \{Lf\}(s) = \int_0^{\epsilon fty} e^{-st} f(t) dt$	$F(s) = \{Lf\}(s) = \int_0^{\epsilon fty} e^{-st} f(t) dt$	100
$\lim (a_n + b_n) = a + b$	$\lim (a_n + b_n) = a + b$	100
$h = \frac{a\sqrt{3}}{2}$	$h = \frac{a\sqrt{3}}{2}$	100
$\lim (a_n + b_n) = a + b$	$\lim (a_n - b_n) = a - b$	95,9166304662544
$\lim (a_n - b_n) = a - b$	$\lim (a_n + b_n) = a + b$	95,9166304662544
$\lim (a_n + b_n) = a + b$	$\lim (a_n - b_n) = a - b$	95,9166304662544
$\lim (a_n + b_n) = a + b$	$\lim (a_n - b_n) = a - b$	95,9166304662544
$\sqrt[3]{8} = 8^{\frac{1}{3}} = 2$	$h = \frac{a\sqrt{3}}{2}$	92,8442061738191
$\sqrt[3]{8} = 8^{\frac{1}{3}} = 2$	$h = \frac{a\sqrt{3}}{2}$	92,8442061738191
$a^{\frac{m}{n}} = \sqrt[n]{a^m}$	$h = \frac{a\sqrt{3}}{2}$	91,8337558167546
$a^{\frac{m}{n}} = \sqrt[n]{a^m}$	$h = \frac{a\sqrt{3}}{2}$	91,8337558167546

Rysunek 12: Wygląd raportu szczegółowego

2 Część druga

2.1 Część techniczna

Struktura projektu podzielona jest na odrębne pliki, które realizują określone funkcje i działania. Program zaprojektowany jest we wzorcu MVP w celu odseparowania warstwy logiki od warstwy prezentacji.



Rysunek 13: Pliki programu

1. IView interfejs ten określa co musi zawierać nasz widok.

- string file – wyświetla ścieżkę wczytanego pliku
- string direct – wyświetla ścieżkę wczytanego folderu
- string alg – pokazuje wybrany algorytm
- string err - pokazuje wybór procentowy
- event Action Add_File – zdarzenie, które wywołuje się w momencie kliknięcia na przycisk „do wyboru pliku”.
- event Action Add_Direc – zdarzenie, które wywołuje się w momencie kliknięcia na przycisk „do wyboru folderu”.
- event Action Start_Prog – zdarzenie, które wywołuje się w momencie kliknięcia na przycisk „Startu programu”.

2. Presenter aktualizuje widok oraz komunikuje się z modelem.

- Presenter(IView view, Model model) – konstruktor nasłuchuje zdarzeń i w odpowiednim momencie wykonuje konkretną funkcję.
- void OriginalFile() – funkcja umożliwia wybranie pliku.
- void Plag_Direc() – funkcja umożliwia wybranie folderu:
- void Operation() – funkcja odwołująca się do modelu z obliczeniami:

3. Model odpowiada za logikę aplikacji.

- `string Load_Orig_Latex()` - Funkcja wczytująca ścieżkę pliku
- `string Load_Plagiat_Direc()` - Funkcja wczytująca ścieżkę do folderu
- `SameOrNot(string alg, string err)` - Główna funkcja która odpowiada za wyodrębnienie wzorów oraz obliczenie prawdopodobieństwa.
- `List<String> sciezki(string Path)` - Funkcja przechowująca ścieżki
- `string konwersjaSlova(string slowo)` - Funkcja konwertująca
- `konwersjaNajlepszegoSlovaNaSwiecie(string slowo)` - Funkcja konwertująca
- `raport(List<double> tablica_wynikow, List<string[]> tablica_wynikow_wzory, string err)` - Funkcja odpowiedzialna za raporty.

2.2 Opis działania

Na samym początku program rozdziela wzory matematyczne od pozostałego tekstu. Korzysta ze składni LaTeX-a, gdzie wzory matematyczne można zapisać na kilka sposobów:

- Między znakami `$`
- Między znakami `$$`
- Między znakami `\(oraz\)`
- Między znakami `\[oraz\]`
- Między `begin` oraz `end{math}`
- Między `begin` oraz `end{displaymath}`
- Między `begin` oraz `end{equation}`

Po wyodrębnieniu wzorów z pliku oraz plików z folderu są one porównywane przez wybrany algorytm:

- Cosine distance
- Euclidean distance
- Levenshtein distance

Po porównaniu wzorów składany jest raport ogólny z porównaniem plików i raport szczegółowy wraz z porównaniem wzorów. Pliki oraz wzory które przekroczyły próg procentowy wyznaczony przez użytkownika zaznaczane są na czerwono.

2.3 Pseudokod Odległości Levenshteina

```
int LevenshteinDistance(char s[1..m], char t[1..n])

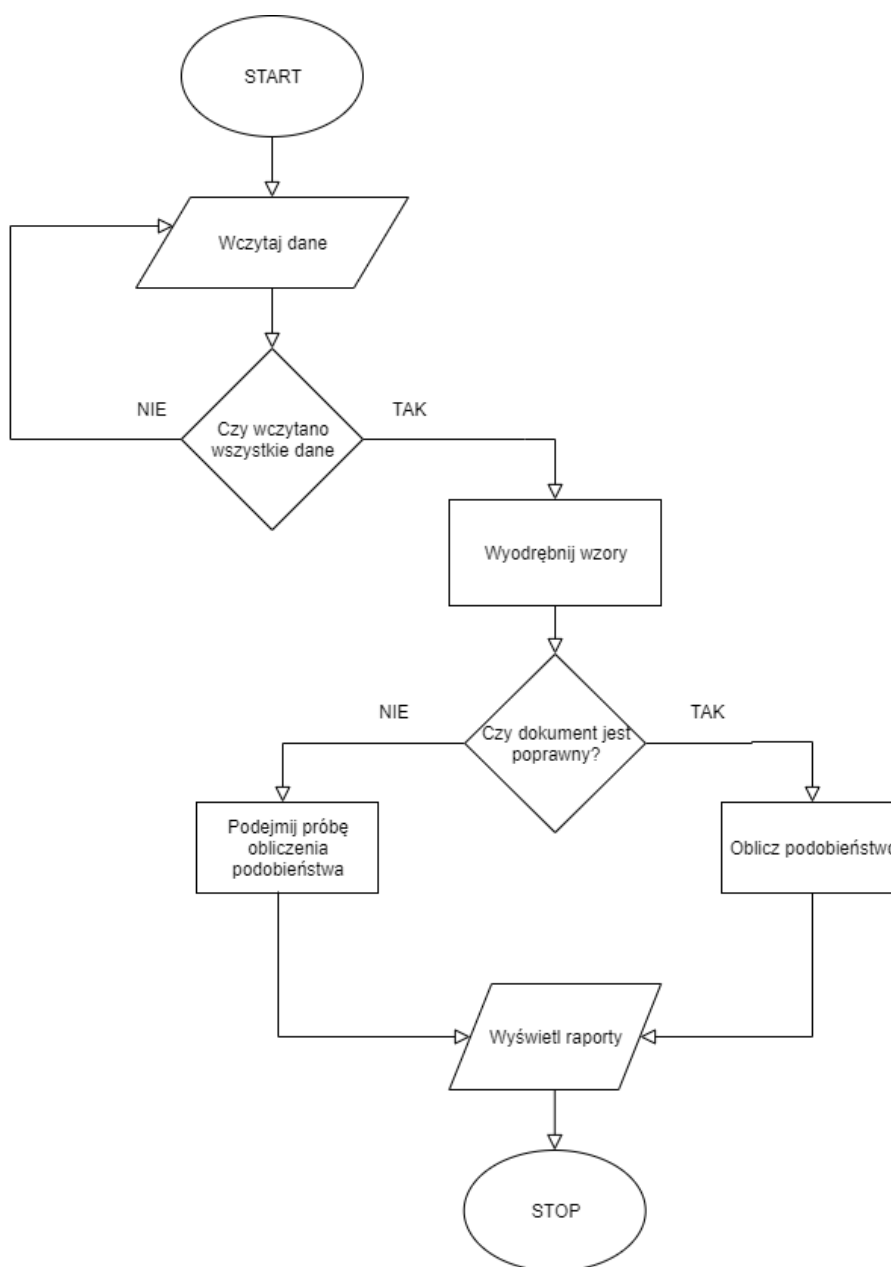
    declare int d[0..m, 0..n]
    for i from 0 to m
        d[i, 0] := i
    for j from 1 to n
        d[0, j] := j

    for i from 1 to m
        for j from 1 to n
            if s[i] = t[j] then cost := 0
                else cost := 1
            d[i, j] := minimum(d[i-1, j] + 1,
                               d[i, j-1] + 1,
                               d[i-1, j-1] + cost)

    return d[m, n]
)
```

3 Implementacja

3.1 Uproszczony schemat blokowy programu



Rysunek 14: Uproszczony schemat

4 Kod Programu

4.1 Program.cs - kod

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Plagiator3000
{
    static class Program
    {
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Model model = new Model();
            IView view = new Form1();
            Presenter presenter = new Presenter(view, model);

            Application.Run((Form)view);
        }
    }
}
```

4.2 Program.cs - wyjaśnienia

Z tego miejsca nasz program zaczyna działać. Jedynie co musieliśmy tu zmienić to wskazać, że korzystamy z MVP. Przekazujemy więc Model, Presenter oraz IView oraz odpowiednio uruchamiamy.

4.3 Form1.cs - kod

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Plagiator3000
{
    public partial class Form1 : Form, IView
    {
        public Form1()
        {
            InitializeComponent();
        }
        public string file
        {

```



```

        set
        {
            label5.Text = value;
        }
    }
    public string direct
    {
        set
        {
            label7.Text = value;
        }
    }
    public string alg
    {
        get
        {
            return comboBox1.Text;
        }
    }
    public string err
    {
        get
        {
            return comboBox2.Text;
        }
    }
}
public event Action Add_File;
public event Action Add_Direc;
public event Action Start_Prog;

public void Message(string s)
{
    MessageBox.Show(s);
}
private void button1_Click(object sender, EventArgs e)
{
    label4.Visible = true;
    Add_File.Invoke();
}
private void button2_Click(object sender, EventArgs e)
{
    label6.Visible = true;
    Add_Direc.Invoke();
}
private void button3_Click(object sender, EventArgs e)
{
    if (label5.Text == "" && label7.Text == "")
    {
        MessageBox.Show("Musisz uzupełnić wszystkie pola!");
    }
    else if (label5.Text == "")
    {
        MessageBox.Show("Musisz jeszcze wybrać oryginalny plik!");
    }
}

```

```

        else if (label7.Text == "")
        {
            MessageBox.Show(" Musisz jeszcze wybrac folder z plikami!");
        }
        else if (comboBox1.Text == "")
        {
            MessageBox.Show(" Musisz jeszcze wybrac algorytm!");
        }
        else if (comboBox2.Text == "")
        {
            MessageBox.Show(" Musisz jeszcze wybrac od ilu prcent testowny
plik jest plagiatem!");
        }
        else
        {
            Start_Prog.Invoke();
        }
    }
}
}

```

4.4 Form1.cs - wyjaśnienia

Form1 ma zaimplementowany interfejs. Ustawione są tu "get-y" oraz "set-y" do odpowiednich komponentów. Również eventy są przydzielone do odpowiednich wyzwalaczy (w naszym przypadku do przycisków). Także w odrębie przycisku "Start" sprawdzana jest poprawność wprowadzonych danych (to jest czy są uzupełnione wszystkie elementy).

4.5 IView.cs - kod

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Plagiator3000
{
    interface IView
    {
        string file { set; }
        string direct { set; }
        string alg { get; }
        string err { get; }

        void Message(string s);

        event Action Add_File;
        event Action Add_Direc;
        event Action Start_Prog;
    }
}

```

4.6 IView.cs - wyjaśnienia

Jest to nasz Interfejs i "widok" w wzorcu MVP. Ustalamy w nim eventy oraz zmienne które będzie przekazywał.

4.7 Presenter.cs - kod

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Plagiator3000
{
    class Presenter
    {
        IView view;
        Model model;

        public Presenter(IView view, Model model)
        {
            this.view = view;
            this.model = model;
            this.view.Add_File += OriginalFile;
            this.view.Add_Direc += Plag_Direc;
            this.view.Start_Prog += Operation;
        }
        public void OriginalFile()
        {
            try
            {
                view.file = model.Load_Orig_Latex();
            }
            catch(Exception e)
            {
                view.Message("Bład ładowania oryginalnych plików: " + e.Message);
            }
        }
        public void Plag_Direc()
        {
            try
            {
                view.direct = model.Load_Plagiat_Direc();
            }
            catch (Exception e)
            {
                view.Message("Bład ładowania plagiatów: " + e.Message);
            }
        }
        public void Operation()
        {
            try
            {
```

```

        model.SameOrNot(view.alg, view.err);
    }
    catch (Exception e)
    {
        view.Message("Bład algorytmu: " + e.Message);
    }
}
}
}

```

4.8 Presenter.cs - wyjaśnienia

Jest to nasz Presenter we wzorcu MVP. "Nasłuchuje" widoku czy zachodzi w nim jakaś zmiana - jeśli tak to przekazuje dane dalej do modelu. Model po ich "obróbcie" zwraca wyniki. Presenter zaś pobiera wyniki od modelu i przekazuje widokowi do wyświetlenia.

4.9 Model.cs - kod

- Kod odpowiedzialny za wczytywanie:

```

public string Load_Orig_Latex()
{
    using (OpenFileDialog openFileDialog = new OpenFileDialog())
    {
        openFileDialog.InitialDirectory = "c:\\";
        openFileDialog.Filter = "tex files (*.tex)|*.tex|All files (*.*)|*.*";
        openFileDialog.RestoreDirectory = true;

        if (openFileDialog.ShowDialog() == DialogResult.OK)
        {
            path = openFileDialog.FileName;
        }
    }
    return path;
}

public string Load_Plagiat_Direc()
{
    using (FolderBrowserDialog win = new FolderBrowserDialog())
    {
        if (win.ShowDialog() == DialogResult.OK)
        {
            path_dir = win.SelectedPath;
        }
    }
    return path_dir;
}

```

- Kod odpowiedzialny za przeprowadzanie operacji

```

public void SameOrNot(string alg, string err)
{
    var files = Directory.EnumerateFiles(path_dir, "*.*",
        SearchOption.AllDirectories).Where(s => s.EndsWith(".tex"));
    string text = File.ReadAllText(path);
}

```

```

Console.WriteLine(text);
string [] tab_oryg = WZORY.Orig_Latex_Operation_Wzory(path);

string wzor_oryg, wzor_plag;
double sameornot;
foreach (string fileName in files)
{
    string File_Latex = fileName;

    string [] tab_plag = WZORY.Orig_Latex_Operation_Wzory(
        File_Latex);

    for (int i = 0; i < tab_oryg.Length; i++)
    {
        wzor_oryg = tab_oryg[i];
        for (int j = 0; j < tab_plag.Length; j++)
        {
            wzor_plag = tab_plag[j];
            if (alg == "CosineDistance")
            {
                double cosDist = Algorytm.CosineDistance(
                    wzor_oryg, wzor_plag);
                sameornot = Algorytm.ToPercent(alg, cosDist);
            }
            else
            {
                double euclDist = Algorytm.EuclideanDistance(
                    wzor_oryg, wzor_plag);
                sameornot = Algorytm.ToPercent(alg, euclDist);
            }
            listwithalgo.Add(sameornot);
            main_list.Add(new string [] {fileName, wzor_plag,
                wzor_oryg, sameornot.ToString() });
            sum += sameornot;
            iter++;
        }
    }
    main_proc = sum / iter;
    listmaintex.Add(main_proc);
    main_proc = 0;
    sum = 0;
    iter = 0;
}
raport(listmaintex, main_list, err);
}

```

- Kod odpowiedzialny za raportowanie

```

public void raport(List<double> tablica_wynikow, List<string[]>
    tablica_wynikow_wzory, string err)
{
    List<string> sciezki_test = sciezki(path_dir);
    string body_tex = "";
    string prebody_tex = "";
    string body_tex2 = "";

```

```

string prebody_tex2 = "";
string body_html3 = "";
string body_html4 = "";
tablica_wynikow_wzory = sotr(tablica_wynikow_wzory);

prebody_tex += "\\begin{flushleft}\\n" + "Plik bazowy : " +
    konwersjaSlowa(path) + "\\n\\end{flushleft}\\n\\hrule\\n";
for (int i = 0; i < sciezki_test.Count; i++)
{
    if (tablica_wynikow[i] < int.Parse(err))
    {
        body_tex += "\\begin{flushleft}\\n" + "Plik : " +
            konwersjaSlowa(sciezki_test[i]) + "\\\\\\n{\\huge
            Stopie podobie stwa: " + tablica_wynikow[i] +
            "\\%} \\\\ \\n" + "\\n\\end{flushleft}\\n\\hrule\\n";
    }
    else
    {
        body_tex += "\\begin{flushleft}\\n" + "Plik : \\
            textcolor{Red}{";
        body_tex += konwersjaSlowa(sciezki_test[i]);
        body_tex += "}\\\\\\n{\\huge Stopie podobie stwa: \\
            textcolor{Red}{";
        body_tex += tablica_wynikow[i] + "}\\%} \\\\ \\n" + "\\n
            \\end{flushleft}\\n\\hrule\\n";
    }
}
var raportTEX = "\\documentclass{article}\\n\\usepackage{polski
}\\n\\usepackage[utf8]{inputenc}\\n\\usepackage{ragged2e}\\n
\\usepackage[dvipsnames]{xcolor}\\n\\begin{document}\\n\\
title{\\huge\\bfseries Raport og lny por wnanie plik w
}\\n\\date{\\today}\\n\\maketitle\\n" + prebody_tex +
body_tex + "\\end{document}";

prebody_tex2 += "\\begin{flushleft}\\n" + "Plik bazowy : " +
    konwersjaSlowa(path) + "\\n\\end{flushleft}\\n\\hrule\\n";
for (int i = 0; i < sciezki_test.Count; i++)
{
    string lista_wzorow2 = "\\begin{longtable}{|c|c|c|} \\n \\
        hline \\n Wz r & Jest podobny do wzoru oryginalnego &
        Procent podobie stwa \\\\ \\hline \\n";
    for (int j = 0; j < tablica_wynikow_wzory.Count; j++)
    {
        lista_wzorow2 += "$" +
            konwersjaNajlepszegoSlowaNaSwiecie(
                tablica_wynikow_wzory[j][1]) + "$ & $" +
            konwersjaNajlepszegoSlowaNaSwiecie(
                tablica_wynikow_wzory[j][2]) + "$ & $" +
            tablica_wynikow_wzory[j][3] + "$ \\\\ \\hline \\n";
    }
    lista_wzorow2 += "\\end{longtable} \\n";
    body_tex2 += "\\begin{flushleft}\\n" + "Plik : " +
        konwersjaSlowa(sciezki_test[i]) + "\\\\\\nLista
        podobnych wzor w: \\\\ \\n" + lista_wzorow2 + "\\n\\end

```

```

        {flushleft}\n\\hrule\n";
    }
    var raportTEX2 = "\\documentclass{article}\\n\\usepackage{
    polski}\\n\\usepackage[utf8]{inputenc}\\n\\usepackage{
    ragged2e}\\n\\usepackage{longtable}\\n\\begin{document}\\n\\
    title{\\huge\\bfseries Raport szczegĆowy por wnania
    plik w }\\n\\date{\\today}\\n\\maketitle\\n" + prebody_tex2
    + body_tex2 + "\\end{document}";

    for (int i = 0; i < sciezki_test.Count; i++)
    {
        if (tablica_wynikow[i] < int.Parse(err))//nie zaznacza na
            czerwono
        {
            body_html3 += "<h3>Plik: " + sciezki_test[i] + " </h3>
            \\n <h2>Stopie podobie stwa: </h2> \\n <h3> " +
            tablica_wynikow[i] + " </h3> \\n <hr> \\n";
        }
        else//zaznacza na czerwono
        {
            body_html3 += "<h3><font color=\\\"red\\\">Plik: " +
            sciezki_test[i] + " </font></h3> \\n <h2>Stopie
            podobie stwa: </h2> \\n <h3><font color=\\\"red\\\"> "
            + tablica_wynikow[i] + " </font></h3> \\n <hr> \\n
            ";
        }
    }
    var raportTEX3 = "<!DOCTYPE html> \\n <html lang=\\\"pl\\\"> \\n <
    head> \\n <meta charset=\\\"utf-8\\\"> \\n <script src=\\\"https
    ://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.1/MathJax.js
    ?config=TeX-AMS-MMLHTMLorMML\\\"></script> \\n <title>Raport
    </title> \\n </head> \\n <body> \\n <header> \\n <h1>Raport
    og lny por wnania plik w </h1> \\n </header> \\n <h2>Plik
    bazowy: " + path + "</h2> \\n <hr> \\n " + body_html3 + " </
    body> \\n </html>";

    for (int i = 0; i < sciezki_test.Count; i++)
    {
        body_html4 += "<h3>Plik: " + sciezki_test[i] + " </h3> \\n
        <hr> \\n <table style=\\\"width:100% \\\"> \\n <tr> \\n <th>
        Wz r</th> \\n <th>Jest podobny do wzoru oryginalnego
        </th> \\n <th>Procent podobie stwa</th> \\n </tr> \\n";
        for (int j = 0; j < tablica_wynikow_wzory.Count; j++)
        {
            body_html4 += "<tr> \\n <td><script type=\\\"math/tex;
            mode=display\\\"> " +
            konwersjaNajlepszegoSlowaNaSwiecie(
            tablica_wynikow_wzory[j][1]) + " </script></td> \\n
            <td><script type=\\\"math/tex; mode=display\\\"> " +
            konwersjaNajlepszegoSlowaNaSwiecie(
            tablica_wynikow_wzory[j][2]) + " </script></td> \\n
            <td><script type=\\\"math/tex; mode=display\\\"> " +
            konwersjaNajlepszegoSlowaNaSwiecie(
            tablica_wynikow_wzory[j][3]) + " </script></td> \\n

```

```

        </tr> \n";
    }
    body_html4 += "</table> \n";
}
var raportTEX4 = "<!DOCTYPE html> \n <html lang=\"pl\"> \n <
head> \n <meta charset=\"utf-8\"> \n <script src=\"https
://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.1/MathJax.js
?config=TeX-AMS-MMLHTMLorMML\"></script> \n <title>Raport
</title> \n <style> \n table, th, td { \n border: 1px
solid black; \n } \n </style> \n </head> \n <body> \n <
header> \n <h1>Raport szczeg  owy por  wnania plik w </
h1> \n </header> \n <h2>Plik bazowy: " + path + "</h2> \n"
+ body_html4 + " </body> \n </html>";

DateTime dt = DateTime.Now;
string PATH = Path.GetDirectoryName(path) + "\\raport\\" + dt.
ToString("MM_dd_yyyy_hh_mm_ss.ffff");
string PATHtex = PATH + "\\raportOGL.tex";
string PATHtex2 = PATH + "\\raportSZCZ.tex";
string PATHtex3 = PATH + "\\raportOGL.html";
string PATHtex4 = PATH + "\\raportSZCZ.html";
bool exists = System.IO.Directory.Exists(PATH);
if (exists)
{
    System.IO.File.WriteAllText(PATHtex, raportTEX);
    System.IO.File.WriteAllText(PATHtex2, raportTEX2);
    System.IO.File.WriteAllText(PATHtex3, raportTEX3);
    System.IO.File.WriteAllText(PATHtex4, raportTEX4);
    Process.Start("chrome.exe", PATHtex3);
    Process.Start("chrome.exe", PATHtex4);
}
else if (!exists)
{
    System.IO.Directory.CreateDirectory(PATH);
    System.IO.File.WriteAllText(PATHtex, raportTEX);
    System.IO.File.WriteAllText(PATHtex2, raportTEX2);
    System.IO.File.WriteAllText(PATHtex3, raportTEX3);
    System.IO.File.WriteAllText(PATHtex4, raportTEX4);
    Process.Start("chrome.exe", PATHtex3);
    Process.Start("chrome.exe", PATHtex4);
}

sotr(tablica_wynikow_wzory);
}

```

4.10 Model.cs - wyja nienia

[breaklines]Kod odpowiedzialny za wczytywanie:

Funkcje tutaj odpowiedzialne s  za otworenie okien dialogowych, do wyboru pliku oraz folderu. Nast pnie przekazuj   cie zki (do kterych zostaly zapisane  cie zki z wybranego pliku oraz folderu) dalej. Kod odpowiedzialny za przeprowadzanie operacji G wna funkcja programu odpowiedzialna za wczytanie plik w, wyodr bienie wzor w, obliczenia podobie stwa oraz stworzenia raportu. Korzystaj c

z klasy WZORY wczytuje wszystkie wzory z plików, następnie korzystając z klasy Algorytm zwraca podobieństwa wzorów. Na końcu wywołuje funkcję raport w celu stworzenia raportów z obliczonych danych. Kod odpowiedzialny za raportowanie Funkcja ta jest odpowiedzialna za raportowanie. Ogólny w którym znajduje się podsumowanie całych plików i ich średniego podobieństwa, oraz raport szczegółowy w którym znajdują się wszystkie porównania wraz z wzorami i ich podobieństwem. Łącznie są tworzone 4 raporty. Po jednym ogólnym oraz szczegółowym w rozszerzeniu .tex oraz po jednym w rozszerzeniu .html.

4.11 Algorytm.cs - kod

- Cosine Distance

```
public static double CosineDistance(string wzorOrig, string wzorCopy)
{
    ReturnExceptionIfNullOrEmpty(wzorOrig, wzorCopy);

    double licznik = 0;
    double mianownik;
    double mianOrig = 0; //pierwiastek(ai ^2)
    double mianCopy = 0; //pierwiastek(bi ^2)
    double cosDistance;

    char[] lettersOrig = PatternToLetters(wzorOrig);
    char[] lettersCopy = PatternToLetters(wzorCopy);

    Dictionary<char, int> frqLtrsOrig = CountFrequentlyOfLetters(
        lettersOrig);
    Dictionary<char, int> frqLtrsCopy = CountFrequentlyOfLetters(
        lettersCopy);

    frqLtrsCopy = DeleteOtherCharInCopy(frqLtrsOrig, frqLtrsCopy);

    foreach (var c in frqLtrsOrig)
    {
        mianOrig += Math.Pow(c.Value, 2);
    }
    mianOrig = Math.Sqrt(mianOrig);

    foreach (var c in frqLtrsCopy)
    {
        mianCopy += Math.Pow(c.Value, 2);
    }
    mianCopy = Math.Sqrt(mianCopy);

    mianownik = mianOrig * mianCopy;

    for (int i = 0; i < frqLtrsOrig.Count; i++)
    {
        licznik += frqLtrsOrig.ElementAt(i).Value * frqLtrsCopy.
            ElementAt(i).Value;
    }

    //jesli mianownik jest rowny 0 oznacza, e zadna litera sie
    nie powtorzyła w plagiacie
```

```

        if (mianownik == 0)
        {
            return 1.0;
        }

        double cosSimilarity = licznik / mianownik;
        cosDistance = 1 - cosSimilarity;
        return cosDistance;
    }

```

- Euclidean Distance

```

public static double EuclideanDistance(string wzorOrig, string
wzorCopy)
{
    ReturnExceptionIfNullOrEmpty(wzorOrig, wzorCopy);

    char[] lettersOrig = PatternToLetters(wzorOrig);
    char[] lettersCopy = PatternToLetters(wzorCopy);

    Dictionary<char, int> frqLtrsOrig = CountFrequentlyOfLetters(
        lettersOrig);
    Dictionary<char, int> frqLtrsCopy = CountFrequentlyOfLetters(
        lettersCopy);

    frqLtrsCopy = DeleteOtherCharInCopy(frqLtrsOrig, frqLtrsCopy);

    double euclideanDistance;
    double roznica;
    double suma = 0;

    for (int i = 0; i < frqLtrsOrig.Count; i++)
    {
        roznica = frqLtrsOrig.ElementAt(i).Value - frqLtrsCopy.
            ElementAt(i).Value;
        roznica *= roznica;
        suma += roznica;
    }
    euclideanDistance = Math.Sqrt(suma);
    return euclideanDistance;
}
private static void ReturnExceptionIfNullOrEmpty(string o, string
c)
{
    if (String.IsNullOrEmpty(o))
        throw new Exception("wzor oryginalny jest pusty!");

    if (String.IsNullOrEmpty(c))
        throw new Exception("wzor plagiat jest pusty!");
}
public static double ToPercent(string algorytm, double lb)
{
    double result = 0;
    if (algorytm == "CosineDistance")
    {

```

```

        result = (double) (1 - lb);
    }
    else
    {
        double pom = (double) (maxOfEuclidean - lb);
        result = Scale(pom, 0.0, maxOfEuclidean, 0.0, 1.0);
    }
    return result * 100; //percent
}

```

4.12 Algorytm.cs - wyjaśnienia

Plik Algorytm.cs jest klasą statyczną. Posiada 9 metod trzy publiczne i sześć prywatnych. Dwie z publicznych metod to algorytmy: cosine distance oraz Euclidean distance, trzecia metoda konwertuje wynik na procentowe podobieństwo.

CosineDistance(string wzorOryginalny, string wzorSkopiowany)

- Przetwarza wzór, i tworzy listy dla oryginału i kopii. Typ listy: Dictionary < char, int > Następnie oblicza ile razy powtórzył się dany znak we wzorze.
- znak, który występuje w kopii, a nie występuje w oryginale jest usuwany
- znak, który występuje w oryginale, a nie występuje w kopii dodawany jest do listy wzoru skopiowanego i przyjmuje wartość 0
- listy Dictionary < char, int > wzorOryg oraz Dictionary < char, int > wzorCopy są wykorzystywane w algorytmie "cosine distance",
- zwraca wartość, w przedziale [0; 1] im mniejsza liczba tym wzór oryginalny jest bardziej podobny skopiowanego

EuclideanDistance(string wzorOryginalny, string wzorSkopiowany)

- Przetwarza wzór, i tworzy listy dla oryginału i kopii. Typ listy: Dictionary < char, int > Następnie oblicza ile razy powtórzył się dany znak we wzorze.
- znak, który występuje w kopii, a nie występuje w oryginale jest usuwany
- znak, który występuje w oryginale, a nie występuje w kopii dodawany jest do listy wzoru skopiowanego i przyjmuje wartość 0
- listy Dictionary < char, int > wzorOryg oraz Dictionary < char, int > wzorCopy są wykorzystywane w algorytmie "euclidean distance",
- zwraca wartość, w przedziale [0; ∞] im mniejsza liczba tym wzór oryginalny jest bardziej podobny skopiowanego

ToPercent(string algorytm, double value)

- W zależności od zmiennej algorytm, która może przyjmować wartości: "cosine distance" lub "euclidean distance" zwracany jest wynik podobieństwa wyrażony w procentach.
- Wynik jest odpowiednio mapowany w zakresie od 0 do 1.

4.13 WZORY.cs - kod

- Wczytywanie i pętla

```
for (int i = 0; i < new_text.Length; i++)
{
    try
    {
        if (new_text[i] == @"\begin{math}")
        {
            while (new_text[i + 1] != @"\end{math}")
            {
                mat[l] = new_text[i + 1];
                pom[lpom] = new_text[i + 1];
                l++;
                i++;
                count++;
                lpom++;
            }
            i++;
            if (count == 1)
            {
                wzory[lwzor] = mat[l - 1];
                lwzor++;
            }
            else if (count > 1)
            {
                join_char = String.Join("", pom);
                wzory[lwzor] = join_char;
                lwzor++;
            }
            count = 0;
            lpom = 0;
        }
        else if (new_text[i] == @"\begin{displaymath}")
        {
            while (new_text[i + 1] != @"\end{displaymath}")
            {
                mat[l] = new_text[i + 1];
                pom[lpom] = new_text[i + 1];
                l++;
                i++;
                count++;
                lpom++;
            }
            i++;
            if (count == 1)
            {
                wzory[lwzor] = mat[l - 1];
                lwzor++;
            }
            else if (count > 1)
            {
                join_char = String.Join("", pom);
                wzory[lwzor] = join_char;
```

```

        lwzor++;
    }
    count = 0;
    lpom = 0;
}
else if (new_text[i] == @"\begin{equation}")
{
    while (new_text[i + 1] != @"\end{equation}")
    {
        mat[l] = new_text[i + 1];
        pom[lpom] = new_text[i + 1];
        l++;
        i++;
        count++;
        lpom++;
    }
    i++;
    if (count == 1)
    {
        wzory[lwzor] = mat[l - 1];
        lwzor++;
    }
    else if (count > 1)
    {
        join_char = String.Join("", pom);
        wzory[lwzor] = join_char;
        lwzor++;
    }
    count = 0;
    lpom = 0;
}
}
catch { MessageBox.Show("Dokument jest niepoprawny!\nNie
    znaleziono zamknienia wyrażenia matematycznego lub
    niepoprawnie go użyto!\nReszta operacji w programie
    może zawiera błędy!"); }
}

```

- Wczytywanie 2 pętla

```

try
{
    for (int i = 0; i < text.Length; i++)
    {
        if ((text[i].ToString() == @"$".ToString()) && (text[i
            + 1].ToString() == @"$".ToString()))
        {
            i += 2;
            while ((text[i].ToString() != @"$".ToString()) &&
                (text[i + 1].ToString() != @"$".ToString()))
            {
                znaki[j] = text[i].ToString();
                j++;
                i++;
            }
        }
    }
}

```

```

        znaki[j] = text[i].ToString();
        i++;
        j = 0;
        join_char = String.Join("", znaki);
        Array.Clear(znaki, 0, znaki.Length);
        mat[1] = join_char;
        wzory[lwzor] = mat[1];
        lwzor++;
        l++;
    }
    else if ((text[i].ToString() == @"$".ToString()) && (
        text[i + 1].ToString() != @"$".ToString()) && (
        text[i - 1].ToString() != @"$".ToString()))
    {
        while (text[i + 1].ToString() != @"$".ToString())
        {
            znaki[j] = text[i + 1].ToString();
            j++;
            i++;
        }
        i++;
        j = 0;
        join_char = String.Join("", znaki);
        Array.Clear(znaki, 0, znaki.Length);
        mat[1] = join_char;
        wzory[lwzor] = mat[1];
        lwzor++;
        l++;
    }
    else if ((Equals(text[i].ToString(), @"\").ToString())
        && (Equals(text[i + 1].ToString(), @"(").ToString()
        ())))
    {
        i += 2;
        while (!spr)
        {
            if (text[i].ToString() == @"\")
            {
                if (text[i + 1].ToString() == @")".ToString())
                {
                    spr = true;
                    i++;
                }
            }
            if (spr)
            { }
            else
            {
                znaki[j] = text[i].ToString();
                j++;
                i++;
            }
        }
    }
}

```

```

        i++;
        j = 0;
        join_char = String.Join("", znaki);
        Array.Clear(znaki, 0, znaki.Length);
        mat[1] = join_char;
        wzory[lwzor] = mat[1];
        lwzor++;
        l++;
        spr = false;
    }
    else if ((Equals(text[i].ToString(), @"\".ToString())
        && (Equals(text[i + 1].ToString(), @"\".ToString()
        ())))
    {
        i += 2;

        while (!spr)
        {
            if (text[i].ToString() == @"\"")
            {
                if (text[i + 1].ToString() == @"]".ToString()
                ())
                {
                    spr = true;
                    i++;
                }
            }
            if (spr)
            {}
            else
            {
                znaki[j] = text[i].ToString();
                j++;
                i++;
            }
        }
        i++;
        j = 0;
        join_char = String.Join("", znaki);
        Array.Clear(znaki, 0, znaki.Length);
        mat[1] = join_char;
        wzory[lwzor] = mat[1];
        lwzor++;
        l++;
        spr = false;
    }
}

}
catch { MessageBox.Show("Dokument jest niepoprawny!\nNie
    znaleziono zamknienia wyrażenia matematycznego lub
    niepoprawnie go użyto!\nReszta operacji w programie może
    zawierać błędy!"); }

```

4.14 WZORY.cs - wyjaśnienia

- Wczytywanie 1 pętla
Pierwsza pętla korzysta z tablicy `new_text[]` w której znajdują się całe linie z wczytanego dokumentu. Pobiera kolejne wartości tablicy aż nie natrafi na jeden z trzech początków zaczynających tekst matematyczny od `begin`. Wtedy dopóki nie natrafi na `end` zapisuje do tablicy na wzory wszystkie napotkane elementy.
- Wczytywanie 2 pętla
Druga pętla korzysta z tablicy `text[]` w której każdy element to pojedynczy znak. Pętla sprawdza każdy element aż nie natrafi na znak lub znaki zaczynające tekst matematyczny. Wtedy zapisuje każdy element do pomocniczej tablicy, aż nie napotka zamknięcia. Po zamknięciu łączy wszystkie elementy(char) w jeden(string) i zapisuje go do tablicy z wzorami matematycznymi.

4.15 Optymalizacja.cs - kod

```
public static string [] Optymalizacje(string text_load)
{
    string [] text_opt = new string [text_load.Length + 20];

    int k = 0, n = 0;
    for (int i = 0; i < text_load.Length; i++)
    {
        if ((text_load[i].ToString() == @"\".ToString()) && (text_load
            [i + 1].ToString() == @"e".ToString()) && (text_load[i +
            5].ToString() == @"m".ToString()) && (text_load[i + 9].
            ToString() == @"}").ToString()))
        {
            for (int m = 0; m < 11; m++)
            {
                if (n == 10)
                {
                    text_opt[k] = "\n".ToString();
                    k++;
                    text_opt[k] = text_load[i].ToString();
                    i++;
                    k++;
                }
                else
                {
                    text_opt[k] = text_load[i].ToString();
                    k++;
                    i++;
                    n++;
                }
            }
            n = 0;
        }
        else if ((text_load[i].ToString() == @"\".ToString()) && (
            text_load[i + 1].ToString() == @"e".ToString()) && (
            text_load[i + 5].ToString() == @"d".ToString()) && (
            text_load[i + 9].ToString() == @"l".ToString()) && (
            text_load[i + 16].ToString() == @"}").ToString()))
        {
            for (int m = 0; m < 18; m++)
```



```

{
    if (n == 17)
    {
        text_opt[k] = "\n".ToString();
        k++;
        text_opt[k] = text_load[i].ToString();
        i++;
        k++;
    }
    else
    {
        text_opt[k] = text_load[i].ToString();
        k++;
        i++;
        n++;
    }
}
n = 0;
}
else if ((text_load[i].ToString() == @"\".ToString()) && (
    text_load[i + 1].ToString() == @"e".ToString()) && (
    text_load[i + 5].ToString() == @"e".ToString()) && (
    text_load[i + 9].ToString() == @"t".ToString()) && (
    text_load[i + 13].ToString() == @"}").ToString()))
{
    for (int m = 0; m < 15; m++)
    {
        if (n == 14)
        {
            text_opt[k] = "\n".ToString();
            k++;
            text_opt[k] = text_load[i].ToString();
            i++;
            k++;
        }
        else
        {
            text_opt[k] = text_load[i].ToString();
            k++;
            i++;
            n++;
        }
    }
    n = 0;
}
else
{
    text_opt[k] = text_load[i].ToString();
    k++;
}
}
k = 0;
return text_opt;
}

```

4.16 Optymalizacja.cs - wyjaśnienia

Kod tutaj odpowiedzialny jest za optymalizację wczytanego tekstu przez wyciąganiem z niego wzorów. Tutaj stosowane są zabezpieczenia polegające na oddzielenie fragmentów tekstu z begin oraz end do osobnych linii. W ten sposób pętla pierwsza może działać poprawnie i wczytywać tekst po lini, bez obawy, że pewne wzory mogą zostać pominięte.