Proiect

STRUCTURI DE DATE AVANSATE

IDENTIFICAREA MOTIVELOR LITERARE IN POEZII

(IMPLEMENTAREA ALGORITMULUI KNUTH-MORRIS-PRATT)

OPREA ANDREEA-TABITA

Informatica Romana, Anul II

Facultatea de Matematica si Informatica, Universitatea de Vest Timisoara

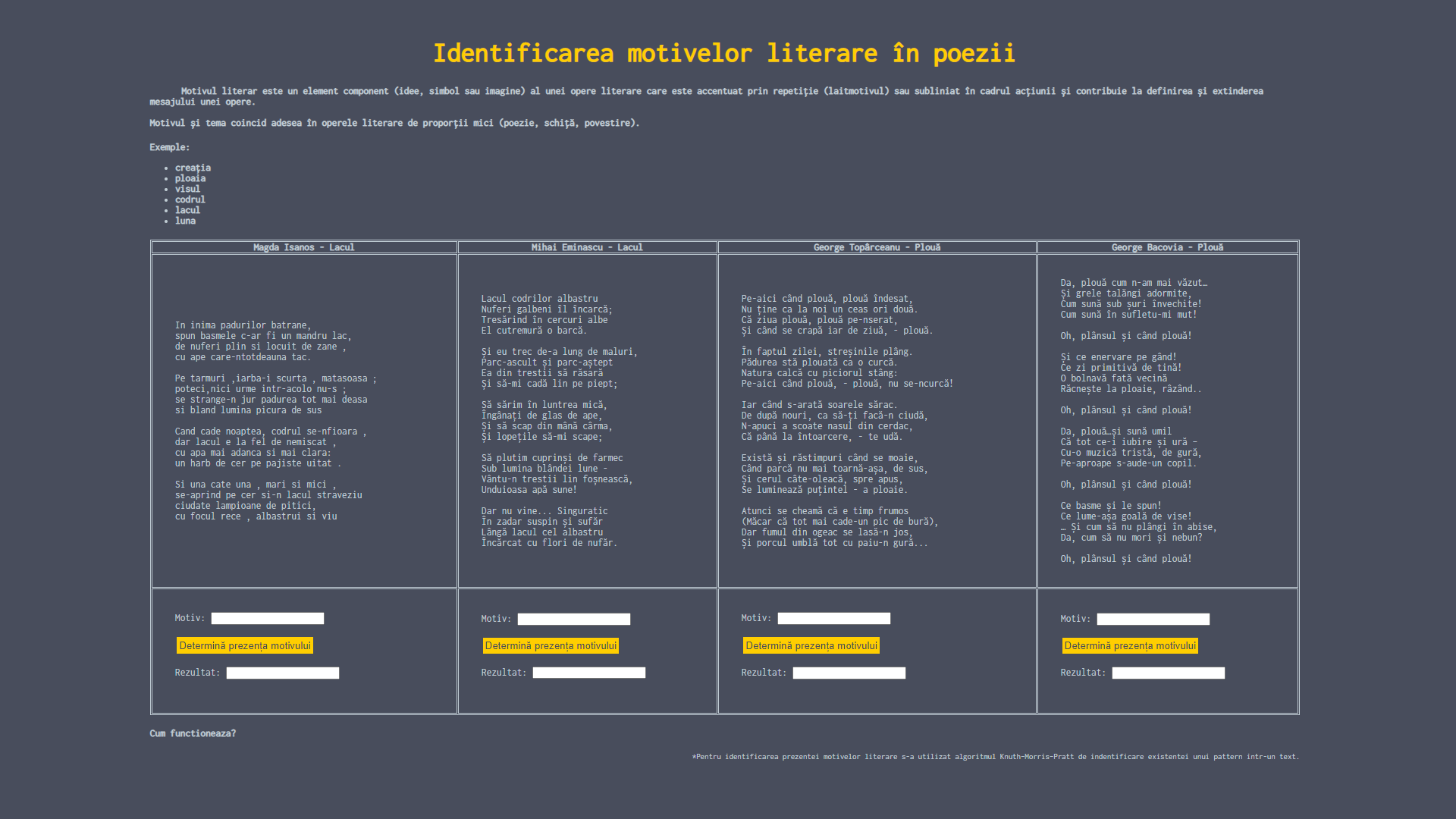
Cuprins

1. Descriere proiect

2. Utilitate

3. Limbaje utilizate

4. Algoritm implementat



# Descriere proiect

* Proiectul consta intr-un single-page website avand ca tematica identificarea motivelor literare in textele lirice date.
* Proiecul presupune implementarea algoritmului KNUTH-MORRIS-PRATT de identificare a unui subsir intr-un sir.

style.css

Index.html

<h1>

<p>

<table>

<script>

<form>

<span>

KMP.js

# 2. utilitate

* Proiectul poate fi utilizat in cadrul orelor de Limba si literatura romana din cadrul invatamantului preuniversitar.
* Poeziile pot fi adaptate planificarii didactice.
* Poate fi util si pentru texte non-lirice sau texte de mari dimensiuni.

# Limbaje utilizate

* HTML pentru a realiza scheletul paginii.
* CSS (style.css) pentru a modifica stilul paginii.
* JAVASCRIPT pentru a implementa formularul si algoritmul.

# Algoritmul knuth-morris-pratt

**// Dacă modelul este găsit, acesta returnează indexul de la începutul primei potriviri din „text”. În caz contrar, se returnează -1.**

function **kmpSearch**(pattern, text) {

if (pattern.length == 0)

return 0; **// daca pattern-ul este null sau este un semn de punctuatie, atunci se va returna 0, adica se va afisa ca motivul nu exista**

if(pattern==" ")

return 0;

if(pattern==".")

return 0;

if(pattern=="?")

return 0;

if(pattern==",")

return 0;

if(pattern==";")

return 0;

if(pattern.length == 1)//daca lungimea patternului este o litera atunci se returneaza 0

return 0;

//**Calculam longest suffix-prefix (LSP)**

var lsp = [0]; **// cazul de baza**

for (var i = 1; i < pattern.length; i++) {

var j = lsp[i - 1]; **// presupunem ca extindem LSP-ul anterior**

while (j > 0 && pattern.charAt(i) != pattern.charAt(j))

j = lsp[j - 1];

if (pattern.charAt(i) == pattern.charAt(j))

j++;

lsp.push(j);

}

**// parcurgem textul**

var j = 0; **// numarul de caractere care se potivesc in pattern**

for (var i = 0; i < text.length; i++) {

while (j > 0 && text.charAt(i) != pattern.charAt(j))

j = lsp[j - 1]; // revenim la pattern

if (text.charAt(i) == pattern.charAt(j)) {

j++; **//Următorul caracter corespunde, poziția e incrementata**

if (j == pattern.length)

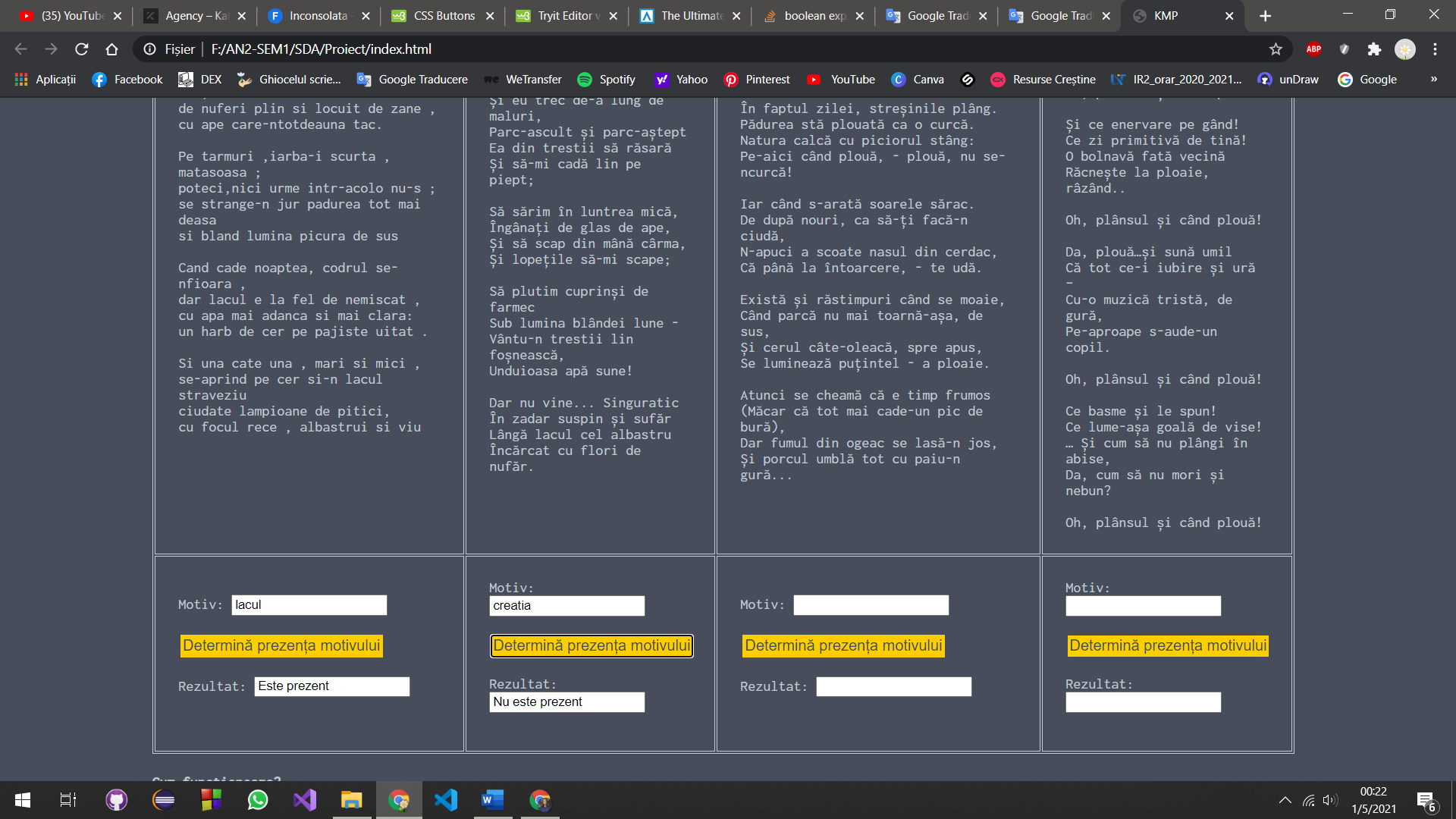
return i - (j - 1);

}

}

return -1; **// nu s-a gasit patternul in text**

}



Algoritmul a fost implementat in cadrul unui formular in care utilizatorul introduce patternul in casuta numta :”Motiv”. Valoarea introdusa esste analizata si procesata. Functia ce implementeaza algoritmul Knuth-Morris-Pratt este apelata prin apasarea butonului, iar rezultatul introdus in casuta corespunzatoare.

Important:

* Calculul tabelului LSP este independent de șirul de text de căutat. Deci, dacă același model este utilizat pe mai multe texte, tabelul poate fi precomputat și reutilizat. Algoritmul KMP necesită întotdeauna Ɵ (m) spațiu suplimentar pentru tabelul LSP.
* Complexitatea de timp a cautarii este O(m+n) indiferent de marimea alfabetului.