**STRUCTURI DE DATE AVANSATE**

**-PROIECT-**

**NUME STUDENT: ISPAS ANDREI-COSMIN**

**NUME PROIECT: pattern\_match\_website**

**1.Descriere proiect:**

Proiectul este de tip aplicatie web. Ideea aplicatiei este de a cauta numarul total de aparitii ale unui subsir intr-un sir si pozitiile la care acestea apar.

De exemplu: Fie dat sirul: “asdfasdb” si substringu “asd”, acesta apare in sir de doua ori, incepand cu pozitiile 0 respectiv 4.

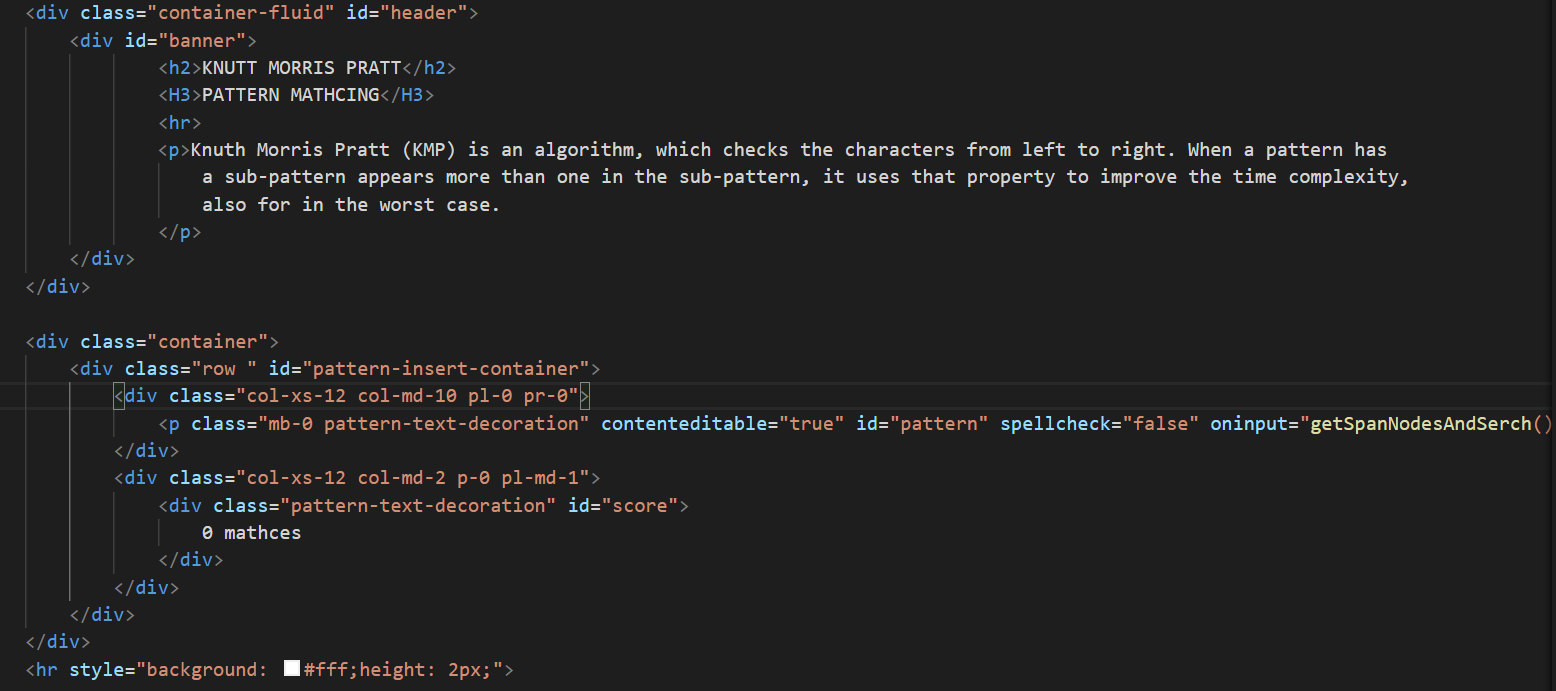
Un algoritm naiv ar face cautare in O(n\*m), n fiind lungimea stringului iar m fiind lungimea substringului. Cautarea devine foarte lenta atunci cand cautam in texte foarte mari. Din aceasta cauza am ales algoritmul “Knuth Morris Pratt”.

Algoritmul KMP cauta de la stanga la dreapta in text aparitiile unui substring in timp de complexitate O(n). Complexitatea se datoreaza tabloului de sufixe si prefixe. Inainte de cautarea propriuzisa se genereaza un tablou cu sufixele si prefixele patternului pe care il cautam. Acesta permite mutearea index-ului trecand peste pozitiile la care nu ar putea fi o potrivire de caractere intre pattern si textul in care se cauta.

**2. Prezentarea tehnologiilor folosite**

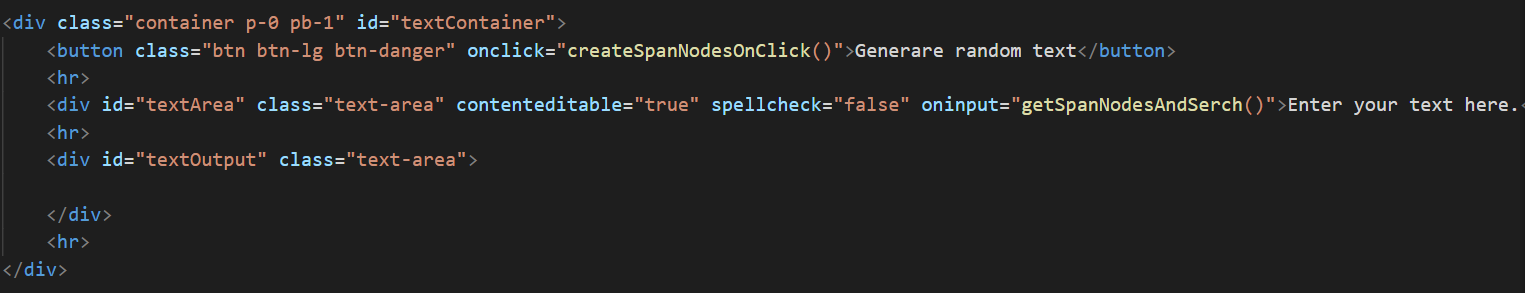
Proiectul este de tip aplicatie web. Scheletul proiectului este reprezentat de partea HTML, stilizarea se face prin CSS iar interactiunea dintre DOM si algoritm o reprezinta JavaScript. Pentru a face aplicatia responsive am folosit libraria Bootstrap 4.

**3.Prezentarea codului sursa**

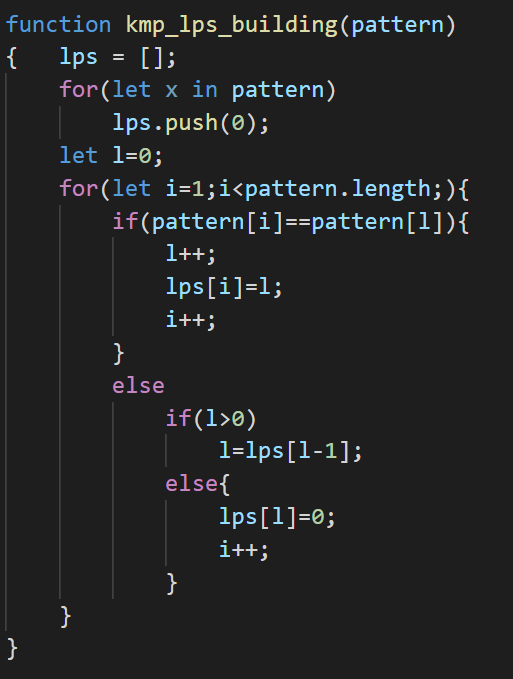


Prima parte este header-ul aplicatiei, cu un banner in care este o mica descriere a algoritmului Knut Morris Pratt.

Cea de-a doua parte avand id=”pattern-insert-container” este componenta in care user-ul insereaza patternul pe care vrea sa il caute. Inserarea se face intr-un element de tip paragraf avand proprietatea “contenteditable=true”, urmata de un element de tip division care reprezinta numarul de match-uri.

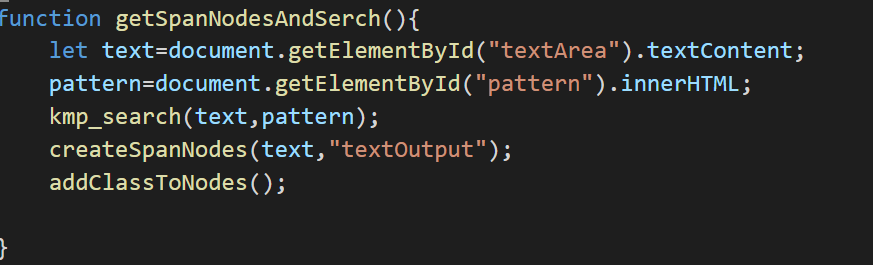


Cel de-al treilea container este partea in care se insereaza textul de catre utilizator in div cu id=”textArea” avand “contenteditable=true” urmat de un div cu id=”textOutput” in care va fi afisat textul .

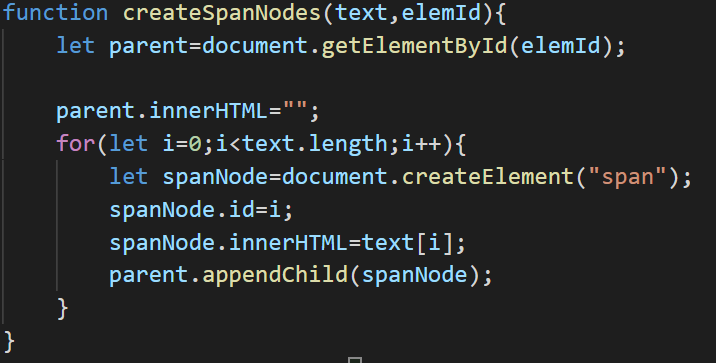




Prima functie, kmp\_lps\_building construieste tabloul de sufixe si prefixe luate din stringul primit ca si parametru si il salveaza in tabloul “lps”. Cea de-a doua functie, kmp\_search efectueaza cautarea propriuzisa si salveaza pozitiile la care a fost gasit pattern-ul in tabloul “results”.



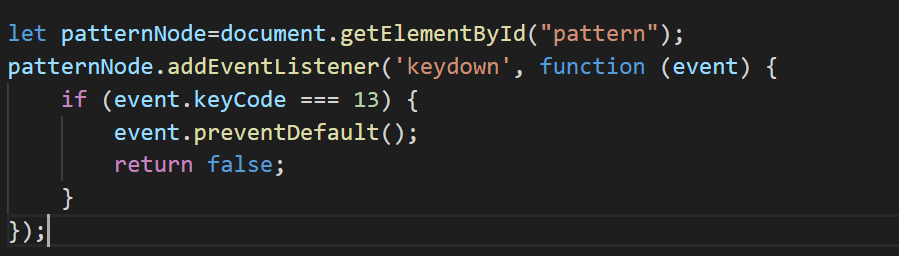
Atunci cand utilizatorul schimba pattern-ul sau textul in care cauta, se apeleaza functia “getSpanNodesAndSearch” care ia textul si patternul pe care utilizatorul le-a introdus, aplica functia de cautare si apleaza celelalte doua functii.



Create span nodes preia textul pe care utilizatorul l-a introdus si pentru fiecare caracter in parte din string creaza un nod DOM de tipul <span> pe care il introduce in tagul cu id =”textOutput”. Transforma fiecare caracter intr-un span deoarece atunci cand un match are loc fiecarui span i se va aplica o clasa CSS pentru a vedea unde a fost gasit patternul.



Atunci cand se creaza fiecare element de tip span, i se atribuie un id=pozitia la care a fost gasit in string. Astfel, adaugarea clasei ia fiecare pozitie salvata in tabloul “results” si incepand de la pozitia respectiva atribuie clasa elementelor pana la pozitie+lungimea patternului.



Functia ii atribuie un event elementului in care userul tasteaza pattern-ul pentru a dezactiva actiunea default atunci cand apasa tasta “enter”;