Mateusz Byczkowski

Dokumentacja wstępna z przedmiotu TKOM

Projekt: Oczyszczanie plików z rozszerzeniem .html z formatowań i stylów

- wybrany język programowania: c++

1. Funkcjonalność: Program będzie przetwarzał plik z rozszerzeniem .html, realizując następujące funkcje:
- usuwanie następujących tagów:
    - address
    - b
    - br
    - code
    - del
    - em
    - hr
    - i
    - ins
    - kbd
    - link
    - mark
    - samp
    - small
    - strong
    - sub
    - sup
    - u
    - var
- usuwanie fragmentów kodu css oraz javascript:
    - <style>...</style>
    - <script>...</script>
- spośród atrybutów pozostawiamy tylko:
    - charset
    - class
    - content
    - href
    - http-equiv
    - id
    - lang
    - name
    - source
    - src
    - type

- wymagania niefunkcjonalne:
Program obsługuję standard HTML5, oraz możliwie jak najlepiej poprzednie wersje, (brak obsługi XHTML).

2. Sposób uruchomienia wejścia/wyjścia. Po uruchomieniu programu mamy do wyboru 4 opcje(na początku wpisujemy nazwę pliku wejściowego):
1 – czytanie tokenów
2 – wyświetlenie document object model
3 – pokazanie zbioru (unikalnych)zagniezdżeń
4 – generowanie pliku wyjściowego o nazwie: [nazwa pliku wejściowego]clean.txt

3.
Tokeny:

Nazwa	Wartość
Open_doc	<!DOCTYPE html
Open	<
Open_end	</
Close	>
Name	^[a-zA-Z]([a-zA-Z1-6:-<])*
Assignment	=
Value	”[set of ascii_char]” | '[set of ascii_char]' | [^#(0-9)*] | [^(1-9)(0-9)*]
Text	[set of ascii_char] (zapytanie o text)
Close_empty	/>
Comment	^<!-- [set of ascii_char]? -->

*Po otrzymaniu tagu Close/Close_empty/Comment podejmujemy próbę wczytania tekstu (do pierwszego napotkanego tokena Open – funkcja getText)
Składnia

<doc_opener>:= <Open_doc> <Close>;
np. <!DOCTYPE html>

<opener>:= <Open> <Name> (<Name> <Assignment> <Value>)*;
np. <html, <table, <h1, <table border=”1” style=”backgroung-color:red;”

<element_end>:=<Open_end> <Name> <Close>;
np. </html>, </table>, </h1>

<element>:= <opener> ((<Close> (<content>)* (<element_end>)?) | <Close_empty>);
np. <h1 style=”color:red”> Tytuł </h1>

<content>:=<element> | <Text> | <Comment>

Produkcja startowa: <data>:= <doc_opener> <Comment>* <element>*

Wykorzystywane struktury:

```
enum Type
{
	OPEN_DOC,
	OPEN,
	OPEN_END
	CLOSE,
	NAME,
	ASSIGNEMNT,
	VALUE,
	TEXT,
	CLOSE_EMPTY,
	COMMENT
};

class Token
{
	public:
	Type type;
	String value;
	Int lineNr;
	Int colNr;
}

enum NodeType {
	SINGLE_TAG,
	DOUBLE_TAG,
	EMPTY_TAG,
	TEXT_NODE,
	COMMENT_NODE
};

class Node
{
	string name;
	NodeType type;
	list <shared_ptr<Node>> parents;
	list <pair<string, string>> attributes;
	list <shared_ptr<Node>> childElements;
	int nestedLevel;
}
```

Realizacja:

Moduły:

a) **Analizator leksykalny**

- Moduł będzie realizował funkcję get_token

- Pobiera po kolei znaki z pliku HTML, tworząc z nich token zgodny z gramatyką i zwróci go jako wynik (przekazać do parsera)

- Podczas swojej pracy program sprawdza czy Name w <opener> nie równa się „style” lub „script”. Wówczas analizator leksykalny przechodzi w tryb szukania wzorca w tekście (odpowiednio: „</style>” lub „</script>”). Wszystko pomiędzy znacznikami <style> </style> i <script> </script> traktujemy jako potomek odpowiedniego Node(Name = style/script) i dodajemy do drzewa jako Node (typ = TEXT).

b) **Analizator składniowy**

- Parser RD

- Rozpoczyna działanie od utworzenia Node(root).

- Porozumiewa się z analizatorem leksykalnym (Scanner), który przesyła kolejne tokeny, na ich podstawie tworzy drzewo składające się z obiektów Node

- W przypadku otrzymania tokena typu <Text> - w Node cała zawartość zapisywana jest w polu name, wówczas Type = TEXT_NODE

- Na początku generowania węzła type = SINGLE_TAG, dopiero w momencie znalezienia tagu domykającego zmieniamy type na DOUBLE_TAG

*Jeżeli po skonstruowaniu <opener> i otrzyamniu tokenu CLOSE otrzymamy token OPEN/COMMENT/TEXT to tworzymy kolejny węzeł i zapamiętujemy wskazanie na niego w list <shared_ptr<Node>> childElements;

c) **Generator pliku wyjściowego**

- przekształca powstałą strukturę na dokument HTML'owy, przechodząc po drzewie i sprawdzając dla każdego węzła typ:
  - TAG – wypisujemy najpierw jego nazwę[1)] i atrybuty[2)], a następnie zawartość jego dzieci, a na końcu jego nazwę w znacznikach zamykających
  - SINGLE_TAG – tak jak w TAG, tylko bez znaczników zamykających
  - TEXT_NODE – wypisujemy jego zawartość (name)

Uwagi:

[1)] Nie przepisujemy znaczników, których nazwa (pole name) jest wymieniona w punkcie 1 (<b>, <i>,<link>…, jeśli <script> lub <style> - nie przepisujemy zawartości danego węzła ani jego dziecka)

[2)] Z atrybutów – zostawiamy tylko te wymienione w punkcie 1

d) **Moduł pokazujący rodzaje zagnieżdżeń**

- Bada drzewo drzewo dokumentu szukając unikalnych ścieżek. Wypisuje na wyjście unikalne zagnieżdżenia, tak aby na ich podstawie można było napisać własne style/formatowania.

4. Obsługa sytuacji wyjątkowych:

Błędy w składni html – konstrukcje niepoprawne językowo oznaczają rzucenie komunikatu o błędzie i przerwanie działania programu.

5. Przykłady testowe

a)

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
    color: blue;
    font-family: verdana;
    font-size: 300%;

}
p  {
    color: red;
    font-family: courier;
    font-size: 160%;
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

po usunięciu

b)

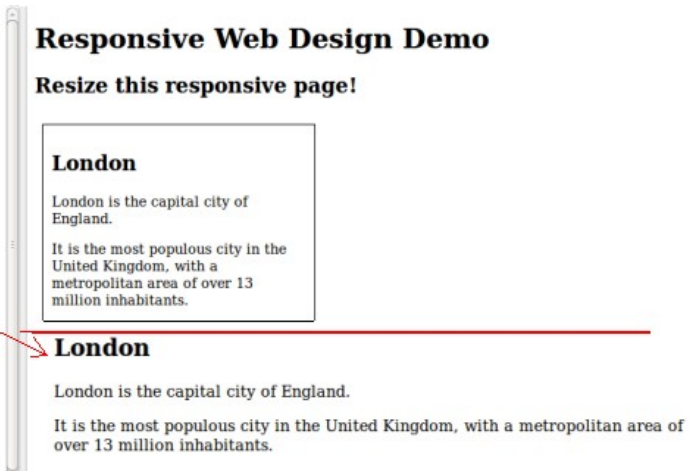
```
<!DOCTYPE html>
<html lang="en-us">
<head>
<style>
.city {
   float: left;
   margin: 10px;
   padding: 10px;
   width: 300px;
   height: 300px;
   border: 1px solid black;
}
</style>
</head>
<body>

<h1>Responsive Web Design Demo</h1>
<h2>Resize this responsive page!</h2>

<div class="city">
  <h2>London</h2>
  <p>London is the capital city of England.</p>
  <p>It is the most populous city in the United Kingdom,
with a metropolitan area of over 13 million inhabitants.</p>
</div>

</body>
</html>
```

po usunięciu

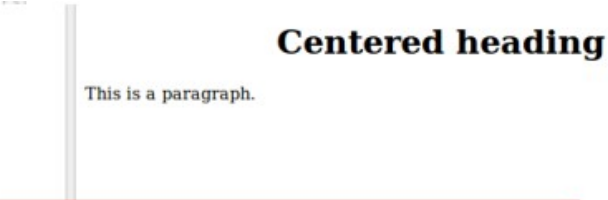


c)

```
<!DOCTYPE html>
<html>
<body>

<h1 style="text-align:center;">Centered heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```




```
<!DOCTYPE html>
<html>
<body>

<h1>Centered heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

d) usuwanie formatowań:

```
<!DOCTYPE html>
<html>
<body>

<h2>HTML <mark>Marked</mark> Formatting</h2>
<ins> Lores depsum itum </ins>
<br>
<em> Lores depsum itum </em>
<br>
<b> Lores depsum itum </b>
</body>
</html>
```

HTML Marked Formatting

Lores depsum itum
Lores depsum itum
Lores depsum itum

```
<!DOCTYPE html>
<html>
<body>

<h2>HTML Marked Formatting</h2>
Lores depsum itum
<br>
Lores depsum itum
<br>
Lores depsum itum
</body>
</html>
```

HTML Marked Formatting

Lores depsum itum
Lores depsum itum
Lores depsum itum