**Документация на курсова работа**

**HTML CRAWLER**

***Информатика и софтуерни науки II-ри курс***

1. **Изисквания и условие**

HTML Crawler

Условие

Да се направи програма, извличаща съдържание от HTML файл. Програмата трябва да разполага с конзолен интерфейс. Програмата трябва да получава параметрите си от командния ред. Тя трябва да позволява търсене и промяна на части от HTML документа.

Примерите са за следния HTML:

<html>

    <body>

    <p>Text1</p>

    <p>Text2</p>

    <p id='p3'>Text3</p>

    <div>

    <div>Text4</div>

    <p>Text5</p>

    </div>

    <table>

    <tr>

    <td>11</td>

    </tr>

    <tr>

    <td>22</td>

    </tr>

    </table>

    <table id='table2'>

    <tr>

    <td>33</td>

    </tr>

    <tr>

    <td>44</td>

    </tr>

    </table>

<a href="http://https://www.w3schools.com">w3schools</a>

     <img src="img\_girl.bmp"/>

    </body>

</html>

Програмата трябва да разполага със следните функционалности:

1. Изграждане на дървовиден модел на документ.

Изграждане на дървовиден модел на подаден от потребителя документ. Да се вземе в предвид, че някои тагове може да нямат съответстващ затварящ таг (пр: <img>). При наличие на грешка в документа програмата трябва да съобщава за нея. Студентът трябва да реализира структурите за създаване на дървовидна структура, както и изграждането на дървовидния модел на документа.

2. Търсене по релативен път.

Търсене на части от модела по релативен път. Елементи на пътя за търсене:

    "//" - корен.

    Примери:

PRINT "//" -> извежда целия HTML документ

    "/" - определя следващо ниво.

    Примери:

PRINT "//html/body/p" -> извежда: "Text1", "Text2", "Text3"

    PRINT "//html/body/table/tr/td" -> извежда: "11", "22", "33", "44"

    "[x]" - x-ти пореден елемент от нивото.

    Примери:

PRINT "//html/body/p[2]" -> извежда: "Text2";

    "\*" - кой да е таг.

    Примери:

PRINT "//html/body/div/\*" -> извежда: "Text4", "Text5"

    PRINT "//html/body/div" -> извежда: "<div>Text4</div><p>Text5</p>"

    tag[@attribute='value'] - определя таг с атрибут със стойност 'value'.

    Примери:

PRINT "//html/body/p[@id='p3']" -> извежда: 'Text3'.

    PRINT "//html/body/table[@id='table2']/tr[2]/td" -> извежда: "44"

    При желание могат да се реализират и други части от езика XPath, което ще носи допълнителни точки. Работата на XPath може да се тества тук: http://xpather.com.

3. Промяна на възел.

Възможност при посочване на един или няколко възела (поддървета) те да се променят/подменят с новоподаден възел (поддърво) или текст. Ако дървовидният модел на документа се изменя, трябва да се изменя само където това е необходимо, без да се преизгражда цялото дърво наново.

     Примери:

    SET "//html/body/p" "AAA" -> променя съдържанието на трите <p> елемента от "Text1", "Text2", "Text3" на "AAA", "AAA", "AAA".

    SET "//html/body/div/div" "<b>Text4</b>" -> Променя съдържанието на посочения div от "Text4" на "<b>Text4</b>"

4. Копиране на възел.

Функцията не трябва да минава през парсване на HTML текст, а директно да копира вече изградената дървовидна структура. Да се помисли как копията могат да се представя ефективно в паметта.

Пример:

   COPY "//html/body/div/div" "//html/body/table[@id='table2']/tr[2]/td" -> копира съдържанието на div елемента в td елемента.

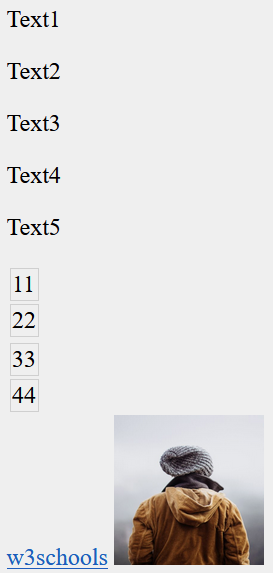
5. Запис във файл.

Запис на зареден (и променен) модел в HTML файл.

6\*. Визуализация.

Визуализация на HTML документа чрез графичен интерфейс на Windows (GDI). Изрисуват се само елементите за таблица (ръбовете на таблицата са видими, с дебелина 1px), <img> (само за bmp) и <a> (подчертан текст в син цвят), както и текст. Не е нужна поддръжката на стилово форматиране.

Пример:



Фиг. Визуализация на посочения по-горе HTML документ. BMP файлът се намира в папката, в която е и HTML документът.

Реализация и точки

Всички функции за обработка на текст трябва да се реализират от студента (не е разрешено използването на функциите string.Split, string.IndexOf, Regex, функциите на LINQ и т.н.). Всички помощни структури и типове трябва да се реализират от студента, в това число стекове, свързани списъци, хеш таблици, дървета и т.н.

Студентът трябва да реализира програмата в следната задължителна последователност:

1. Изграждане на дървовиден модел на документ.

В HTML документа големината на шрифта не трябва да е от значение. За изграждане на дървовиден модел на документа, трябва да се използват само разработени от студента типове.

Макс. брой точки за реализация: 20;

1. Търсене по релативен път.

Реализирането на паралелно търсене не е задължително, но ще донесе допълнителни точки.

Макс. брой точки за реализация: 15;

1. Промяна на възел.

Макс. брой точки за реализация: 10;

1. Копиране на възел.  
     
   Макс. брой точки за реализация: 10;

1. Запис във файл.

Макс. брой точки за реализация: 5;

1. \* - Тази точка е незадължителна. При реализиране на всички точки, вкл. незадължителната, студентът ще бъде освободен от изпит с отлична оценка.
2. **Реализация**

Първата стъпка която се предприе беше изборът на **‘Windows Forms’** за поле на разработка, с цел изпълнение и на визуалната част на проекта в последствие.

По условие се изисква целият ‘html’ код да се преобразува в дървовидна форма. За целта първо беше изграден класът **‘HTreeNode’** . Това е обект , който притежава свойствата **‘Tag’**, **’ValueText’, ‘\_props’, ‘\_children’, ‘IsCoppied’** , метод **‘AddChild’** и два конструктора. Като **‘Tag’** отговаря за специфичния ‘html’ таг, **’ValueText’** ни дава информация дали въпросният клон е просто едно листо, **‘\_props’** e свързан списък отговарящ за атрибутите на ‘html’ таг, **‘IsCoppied’** ни дава информация дали клонът е бил ‘слабо‘ копиран и **‘\_children’** държи в свързан списък децата на текущият обект/клон.

Втората стъпка беше изграждането на класа ‘HLinkedList’- това е свързаният списък в който се съхраняват ‘децата’ на всеки клон. Като по-специфичното е, че наследява IEnumarable<HtreeNode> и използва класът ‘Node’, който върши работа на „пойнтер“ в самия лист. Интерфейсът позволява за по-лесно обхождане на дървото от потребителя. Този свързан списък позволява свободното добавяне и премахване на връзки.

Третата предприета стъпка беше изграждането на класа ‘NLinkedList’, който също е свързан списък, но този път не наследява ‘IEnumarable’ и служи само за съхранение на ‘Атрибутите’ на всеки таг.

Четвъртата стъпка беше създаването на класа ‘HashTable’, чрез който се реализира „Хешираща Таблица“, на която още при зареждането в програмата се подават два файла от които да се напълнят позициите ѝ с валидни html тагове. Това се прави с цел намаляване времето за валидиране на даден таг при изграждането на дървото, тъй като за потребителят е важно html документът да бъде без грешки.

Петата стъпка беше изграждането на ‘utility’ класа ‘TextManipualation’, в който са реализирани методи за обработка на обекти от тип ‘string’.

Шестата стъпка беше съставянето на класа ‘LinkStack’, който използвайки класа ‘NLinkedList’ симулира стек. По този получаваме динамично изграден стек като структура от данни.

В седма стъпка беше изграден и вторият ‘utility’ клас - ‘HParser’, в който са и самите алгоритмите за преобразуване на html документ в дървовидна структура. В часност метода ‘HtmlParser’ отговаря за тази цел.