1. Министерство науки и высшего образования Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт компьютерных наук и кибербезопасности

**К У Р С О В А Я Р А Б О Т А**

1. «Реализация грамматики HTML5»
2. по дисциплине «Формальный грамматики и теория компиляторов»
3. Выполнил
4. студент гр. 5151004/10101 Гуторова Л.С.

<*подпись*>

1. Пересунько А.А.

<*подпись*>

1. Резникова М.С.

<*подпись*>

1. Проверил
2. Старший преподаватель Грибков Н.А.

<*подпись*>

1. Санкт-Петербург
2. 2024

Содержание

[Содержание 2](#_Toc166625289)

[1 Цель 3](#_Toc166625290)

[2 Постановка задач 3](#_Toc166625291)

[3 Теоретические исследования 4](#_Toc166625292)

[4 Описание решения 7](#_Toc166625293)

[4.1 Созданные файлы и реализация makefile 7](#_Toc166625294)

[4.2 Лексический анализатор 7](#_Toc166625295)

[4.2.1 Функции файла 7](#_Toc166625296)

[4.2.2 Состояния 8](#_Toc166625297)

[4.2.3 Работа стэка 9](#_Toc166625298)

[4.3 Синтаксический анализатор 10](#_Toc166625299)

[5 Тестирование и результаты работы программы 10](#_Toc166625300)

[6 Выводы 14](#_Toc166625301)

[Список используемых источников 15](#_Toc166625302)

[ПРИЛОЖЕНИЕ А 16](#_Toc166625303)

[ПРИЛОЖЕНИЕ Б 28](#_Toc166625304)

[ПРИЛОЖЕНИЕ В 30](#_Toc166625305)

[ПРИЛОЖЕНИЕ Г 33](#_Toc166625306)

# Цель

Главной целью курсовой работы является разработка распознавателя современного формального языка (HTML5) с помощью генератора синтаксических анализаторов YACC, генератора лексических анализаторов Flex и языка C++. Разработанный анализатор должен предупреждать о наличии конструкций и элементов, относящихся к другим стандартам HTML, а также распознавать базовые синтаксические и лексические ошибки.

# Постановка задач

Для достижения поставленных целей необходимо:

* изучить основы HTML5: синтаксис, спецификацию, различия с предыдущими версиями;
* разработать программу для Flex, с целью создания лексического анализатора для грамматики;
* разработать грамматику для генератора синтаксических анализаторов YACC;
* протестировать полученный распознаватель.

# Теоретические исследования

HTML5 (HyperText Markup Language Version 5) представляет собой язык для организации и отображения контента в интернете. Это последняя версия HTML, завершенная в 2014 году. HTML5 разработан с целью улучшения поддержки мультимедийных технологий, сохраняя обратную совместимость, читаемость кода и простоту анализа для парсеров.

HTML5 внедряет ряд новых элементов и атрибутов, соответствующих современным требованиям к структуре веб-страниц.

HTML5 — это последняя версия языка разметки HTML, используемая для создания и структурирования веб-страниц. Основные особенности HTML5 включают в себя:

1. HTML5 внедряет ряд новых элементов, таких как <header>, <footer>, <nav>, <article>, <section> и другие, которые позволяют разработчикам более точно определять структуру веб-страницы.
2. HTML5 предоставляет встроенную поддержку аудио и видео без необходимости использования дополнительных плагинов, таких как Flash.
3. HTML5 включает в себя новые элементы для рисования, такие как <canvas>, что позволяет разработчикам создавать интерактивные графики и визуализации.
4. HTML5 предоставляет новые атрибуты и элементы для упрощения работы с формами, включая элемент <input> с различными типами (например, email, url, date) и возможность проверки данных на стороне клиента.
5. HTML5 предоставляет средства для локального хранения данных на стороне клиента, такие как localStorage и sessionStorage, что улучшает возможности веб-приложений для работы в оффлайн-режиме.
6. HTML5 включает в себя функции адаптивного дизайна и поддержку мобильных устройств, что делает его более удобным для создания мобильных приложений и веб-сайтов.

Некоторые основные элементы, которые используются в данном языке:

* Корневой элемент: элемент <html>.
* Метаданные документа: <head>, а также элементы, которые располагаются внутри него.
* Скрипты: скрипты позволяют добавлять интерактивности на веб-страницу, в эту группу входят элементы, управляющие скриптами.
* Структурные элементы: элементы, управляющие основными разделами веб-страницы, вроде <body>, <section>, <nav>, <article>, <aside> и др.
* Группирование контента: Элементы, обрамляющие текст, списки, изображения.
* Текст: Элементы, изменяющие вид текста, например, делающие его жирным или курсивным, а также выделяющие текст по смыслу — аббревиатура, цитата, переменная, код и т. д.
* Рецензирование: Элементы <ins> и <del> показывающие редактирования в документе.
* Внедряемый контент: Элементы, вставляемые на страницу в виде разных объектов – изображения, видео, аудио и др.
* Табличные данные: Элементы для создания и управления видом таблиц.
* Формы: Формы являются одним из важных элементов любого сайта и предназначены для обмена данными между пользователем и сервером. В эту группу входят элементы для создания формы и её полей.
* Интерактивные элементы: Специальные виджеты, с помощью которых пользователь может получать дополнительную информацию или управление.
* Ссылки: Элементы <a> и <area>.

Подобное группирование условно и может принимать другой вид, потому что одни и те же элементы могут принадлежать разным группам.

Для обозначения начала и конца элемента применяются теги. Внутри тегов могут быть атрибуты со своими значениями, расширяющими возможности тегов, а также содержимое.

Атрибуты тегов расширяют возможности самих тегов и позволяют гибко управлять различными настройками отображения элементов веб-страницы. Общее количество атрибутов достаточно велико, но их значения, как правило, можно сгруппировать по разным типам, например, задающих цвет, размер, адрес и др.

Структура языка HTML5:

* HTML документ начинается с элемента <!DOCTYPE>, который предназначен для указания типа текущего документа — DTD (document type definition, описание типа документа);
* HTML-документ состоит из дерева HTML-элементов и текста;
* элемент <html> определяет начало HTML-файла, внутри него хранится заголовок (<head>) и тело документа (<body>);
* элементы могут вкладываться друг в друга;
* отношения между множественными вложенными элементами подразделяются на родительские, дочерние и сестринские;
* теги не чувствительны к регистру;
* чаще всего каждый элемент обозначается в исходном документе открывающим и закрывающим тегом;
* открывающий тег элемента начинается с символа «<», далее идет имя элемента, его атрибуты и символ «>»;
* закрывающий тег элемента начинается с символов «</», далее идет имя элемента и символ «>»;
* атрибуты начинаются с имени атрибута, знака «=» и значения атрибута в одинарных или двойных кавычках;
* число атрибутов элемента неограниченно.

# Описание решения

## Созданные файлы и реализация makefile

В ходе работы был разработан распознаватель языка HTML5. В основе языка лежит лексический lex-файл lex.l (Приложение 1), а логику работы с лексикой и калькулятором содержит yacc-файл grammar.y (Приложение 2). Для запуска программы используется созданные MakeFile (Приложение 4).

A screenshot of a computer program

Description automatically generated

Рисунок 1 – Команды для сборки лексического и синтаксического анализаторов

## Лексический анализатор

Главной задачей лексического анализатора является разбор лексем текста. В контексте анализа html-разметки в качестве основных лексем были выделены теги, атрибуты, состояния и комментарии – они составляют основной синтаксис HTML5. Помимо синтаксиса файл lex.l содержит функционал, связанный с обработкой синтаксиса выше уровня лексем, обработкой входного файла и вызовом вывода ошибок и предупреждений.

A screenshot of a computer program

Description automatically generated

Рисунок 2 – Содержание файла lex.l

Каждый тег был выделен в отдельный элемент. Элементы могут соединяться в другой элемент, который обрабатывается и отправляет синтаксическому анализатору конкретный токен.

Так как язык не чувствителен к регистру, то было принято решение об использовании такого типа конструкций:

tagHTML [hH][tT][mM][lL]

Для решения проблемы распознавания тегов, которые могут встречаться в других версиях языка HTML5, был создан список с регулярными выражениями старых версий. В случае таких тестов выводится предупреждение, но программа обрабатывает эти теги.

Атрибуты были собраны в группы и обрабатываются с помощью специальных состояний.

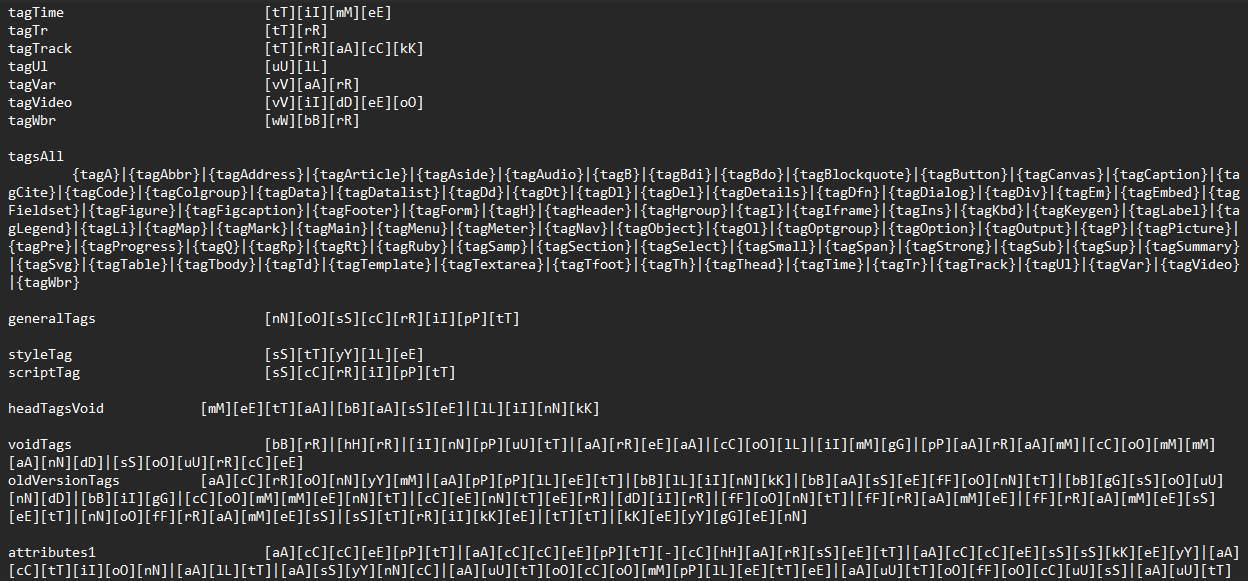


Рисунок 3 – Реализация лексем тэгов и атрибутов в lex.l

### Функции файла

#### Main

В функцию main передается аргумент с названием тестового файла. Далее обрабатывается этот аргумент. Открывается файл, с которого считываются токены. Если файл не найден, то программа выдаст ошибку. Далее происходит инициализация стека для корректной обработки закрывающих тегов, парсинг, деинициализация стека. После тестовый файл закрывается. В случае синтаксически верного тестового файла выводится сообщение об корректности программы.

#### Обработчики

В программе реализованы два обработчика: ошибок и предупреждений. В первом выводится ошибка и программа завершает работу. Во втором выводится сообщение с предупреждением. Этот обработчик используется для старых версий тегов, атрибутов, доктайпа.

### Состояния

Состояния используются для корректной обработки корректности конструкций языка.

#### Состояния, которые возвращают один токен

Есть состояния, которые лишь возвращают токен для синтаксического анализатора, а именно: \<{tagDOCTYPE}\> и закрывающие токены основных элементов программы.

#### Обработка старых или несуществующих элементов

В случае, если встречаются элементы, которые устарели или не существуют, то данные состояния выведут ошибки и либо вернут корректный токен(в случае устаревших конструкций), либо завершат работу программы(в случае несуществующих элементов).

#### Состояния тегов

Многие состояния тегов обрабатывают состояния атрибутов. Эти состояния будут описаны ниже. А далее возвращают токен с тегом. Есть два состояния, которые обрабатывают теги (без атрибутов и с атрибутами), которые заносят тег в стэк. Это необходимо для того, чтобы корректно обрабатывать теги, которые должны закрыться.

В случае закрывающегося тега вызывается функция проверки tagStackCheck().

#### Состояния атрибутов

Описание каждого блока состояния:

* {attributesALL}: если атрибут найден, то возвращается токен атрибута.
* {oldVersionAttr}: если атрибут старой версии, то выводит предупреждение и возвращает атрибут.
* {name}: если атрибут неизвестен, то выводит предупреждение и возвращает атрибут.
* {attributesValue}: возвращает значение атрибута.
* [\n]: увеличивает счетчик строк на 1.
* \/\>: Выводит предупреждение о нестандартном использовании самозакрывающегося тега и возвращает символ закрытия.
* [>]: Возвращает символ закрытия и сбрасывает состояние.
* {whitespace}: игнорирует пробелы и табуляцию.
* {attrWithoutValue}: игнорирует атрибуты без значения.
* .: Игнорирует любой другой символ.

#### Состояния для закрывающих тегов

Для основных тегов есть обработка закрывающих тегов. Например, для тега style: когда обнаруживается закрывающий тег </style>, код возвращает значение TAG\_STYLE\_CLOSE и сбрасывает состояние. Если встречается символ новой строки (\\n), то счетчик строк увеличивается на 1. Если встречается любой другой символ, то он игнорируется.

#### Обработка комментариев

Когда обнаруживается комментарий, то программа переходит в состояние commentTagStart. Оно возвращает токен открытого комментария, а далее программа переходит в состояние, ожидающее завершения комментария. В состоянии comment\_continue когда обнаруживается, что комментарий закрыт, то код возвращает значение TAG\_COMMENT\_END и сбрасывает состояние. Если встречается символ новой строки (\\n), то счетчик строк увеличивается на 1.

### Работа стэка

Был реализован код работы со стэком. Код содержит базовые функции для стэка: StackPush(), StackPop(), StackIsEmpty(), StackInit(), StackDeinit().

Когда встречается тег, то он с помощью функции StackPush() заносится в стэк. Когда встречается закрывающийся тег, то вызывается функция StackCheck(). Это функция достает верхнее значение из стека и проверяет, корректный ли закрывающий тег в коде. Если они имеют разное значение, то выводится не найденный тег, а также все теги, которые хранятся в стэке. Если одинаковое, то значение убирается из стэка, и программа продолжает работу.

A screenshot of a computer screen

Description automatically generated

Рисунок 4 – Пример реализации функций обработки в файле lex.l

## Синтаксический анализатор

Данный код проверяет корректность последовательности токенов. Файл грамматики также включает правила для работы с лексемами как токенами. Эта часть является основой программы, так как здесь происходит проверка правильности последовательностей токенов. Правила также включают вызов функции для вывода сообщения об ошибке при обнаружении несоответствия.

A screenshot of a computer program

Description automatically generated

Рисунок 5 – Содержание grammar.y, реализующего синтаксический анализатор

Для начала проверяется, есть ли корректный docktype. Далее идет метка html\_section. Есть еще вариант, что в начале написан комментарий.

В метке html\_section идет следующий вариант меток: html\_start head\_section body\_section TAG\_HTML\_CLOSE или в конце еще добавляется комментарий.

html\_start содержит проверку на токен тега HTML, а далее проверка на то, есть ли атрибуты у этого тега.

Head\_section содержит проверку на наличие head. В этой метке проверяется следующая структура: тег head, содержимое head, тег title, закрывающийся тег head. В содержимом head проверяется наличие следующих тегов: <noscript>, <meta>, <base>, <link>, <script> или комментарий.

В title\_section проверяется наличие открывающегося токена, наличия атрибутов и закрытия тега.

Похожая структура имеется в body\_section. А далее закрывается тег html.

# Тестирование и результаты работы программы

В качестве результата программа использует вывод текста в консоль: “Program is correct” при отсутствии ошибок html-кода или тип ошибки, строку и пояснения, если анализатор нашёл ошибки html-кода.



Рисунок 6 – Пример успешного результата работы анализатора

Были введены несколько типов ошибок, такие как синтаксические и семантические. Синтаксическая ошибка может появляться при некорректном написании или использовании несуществующего в лексике тегов. Семантической ошибкой будет являться любое нарушение логики построения html-разметки.

A screenshot of a computer program

Description automatically generated

Рисунок 7 – Примеры вывода о завершении программы с разными исходами

Для тестирования работы анализатора были составлены txt-файлы, содержащие html-синтаксис. Анализатор принимает в качестве входного файла любые файлы, которые могут быть представлены в текстовом формате, поэтому один и тот же файл с расширениями .c, .html или .txt будет одинаково проходить проверку анализатором независимо от расширения. Далее представлена таблица с кратким описанием содержания каждого тестового файла:

| **Описание / Папка с тестами** | **Название примера** | **Название файла** |
| --- | --- | --- |
| Тесты на проверку отдельных тегов изолированно / SingleTagTests | Документ | document.txt |
| Заголовок | heading.txt |
| Проверка работы шаблона со стилями | style.txt |
| Проверка работы шаблона со скриптом | script.txt |
| Проверка шаблона <base> | base.txt |
| Проверка шаблона <noscript> | noscript.txt |
| Форматирование ввода <kbd> | kbd.txt |
| Форматирование вывода <samp> | samp.txt |
| Форматирование кода <code> | code.txt |
| Форматирование строк <pre> | pre.txt |
| Форматирование переменных <var> | var.txt |
| Проверка конструкции a href | ahref.txt |
| Тесты, собирающие теги по группам / CustomTests | Проверка основных тегов: <DOCKTYPE!>, <head>, <title>, <h1>, <p>, <a href>, <img>, <button>, <list> | MainTags.txt |
| Проверка атрибутов: <title>, <href>, <width>, <height>, <alt>, <style>, <lang>, <scr> | Attributes.txt |
| Проверка атрибутов для <p>: <hr>, <br>, <pre> | ParagraphsAttributies.txt |
| Проверка атрибута style в разных вариациях: <body style>,<h1 style>, <p style> | StylesAttributes.txt |
| Проверка атрибутов форматирования текста: <b>, <strong>, <i>, <em>, <small>, <mark>, <del>, <ins>, <sub>, <sup> | TextFormating.txt |
| Проверка цитирования, аббревиатур, <address>, <cite>, <bdo> | QuotesFormating.txt |
| Проверка форматирования комментариев | Comments.txt |
| Проверка <a href>, <target>, внешние и внутренние ссылки, картинка и кнопка как ссылка, mailto | Links.txt |
| Базовый формат таблицы, таблица с границами, добавление заполнения ячеек, выравнивание заголовков, добавление интервалов между границами, ячейка охватывает много строк или столбцов, добавление подписи к таблице, особый стиль для отдельной таблицы | Tables.txt |
| Неупорядоченный список, упорядоченый список, описанный список | Lists.txt |
| Проверка элементов блочного уровня и встроенных элементов | DIV\_SPAN.txt |
| Использование атрибута class, использование атрибута class для встроенных элементов, определение стилей для класса, несколько классов, различные элементы с одним классом | Class.txt |
| Синтаксис фрейма и фрейм - целевая ссылка | Frame.txt |
| Проверка синтаксиса форм, элемент <input>, текстовые поля, элемент <label>, кнопки radio, кнопка submit, атрибут action, атрибут target, атрибут method, атрибут name | Forms.txt |
| Элемент <input>, элемент <select>, элемент <option>, атрибут size, атрибут multiple, элемент <textarea>, элемент <button>, элементы <fieldset> и <legend>, элемент <datalist>, элемент <output> | FormsTags.txt |
| Типы ввода форм: text, password, submit, reset, radio, checkbox, button, color, date, datetime-local, email, file, month, number, range, search, tel, time, url, week | FormTypes.txt |
| Тесты с проверкой примеров сайтов с функционалом / Sites | TutorialsTonight шаблон | test\_TutorialsTonight.txt |
| Шаблон простого сайта на HTML | test\_approved.txt |
| Шаблон страницы с функцией вывода на старницу времени обновления сервера | ServerUpdate.txt |
| Проверка шаблона без тега <head> | WithouthHEAD.txt |

Для удобного вызова тестов, помимо сборки основных файлов лексического и синтаксического анализаторов, в makefile были добавлены цели custom, single, sites и err\_tests lля вызова тестовых файлов группами относительно папок, в которых они находятся. Цель err\_tests вызывает тесты из папки TestsWithErrors/, созданной специально для наглядной демонстрации вывода ошибок.

A screenshot of a computer program

Description automatically generated

Рисунок 8 – Цели для вызова тестов в Makefile

# Выводы

В данной курсовой работе была поставлена цель разработки распознавателя современного формального языка HTML5. Для ее достижения были успешно решены следующие задачи:

* изучено теоретическое основание html5, его специфика и отличия от предыдущих версий;
* создан лексический анализатор с помощью генератора flex, который распознает токены в соответствии с грамматикой языка;
* разработана грамматика для генератора синтаксических анализаторов yacc, которая определяет правила формирования синтаксически правильных конструкций;
* осуществлено тестирование полученного распознавателя на различных примерах HTML5-кода.

В результате проделанной работы был создан распознаватель HTML5, который выполняет следующие функции:

* проверяет синтаксическую корректность HTML5-документов, выдавая предупреждения о несоответствиях стандарту;
* обнаруживает базовые лексические ошибки, такие как некорректные имена тегов и атрибутов;
* различает конструкции и элементы, относящиеся к более старым версиям html, предупреждая о их использовании.

Созданный распознаватель может быть использован для проверки HTML5-кода в процессе разработки веб-приложений, что позволит повысить его качество и соответствие действующим стандартам.

Список используемых источников

1. DIVE INTO HTML5 by Mark Pilgrim URL: <http://diveintohtml5.info>
2. HTML5 // htmlbook.ru URL: <https://htmlbook.ru/html5>
3. HTML Примеры // SchoolsW3 URL: <https://www.schoolsw3.com/html/html_examples.php>
4. HTML 4.01 Strict DTD // W3 URL: <https://www.w3.org/TR/html4/strict.dtd>
5. Мартин Браун Создание анализаторов текста при помощи lex и yacc // URL: [http://www.larionov.mytomsk.ru/translate/lex\_yacc.pdf - 2012](http://www.larionov.mytomsk.ru/translate/lex_yacc.pdf%20-%202012)
6. Информация о программах lex и yacc // IBM URL: https://www.ibm.com/docs/ru/aix/7.2?topic=concepts-lex-yacc-program-information (дата обращения: 04.03,2024).
7. Примеры программ с использованием lex и yacc // IBM URL: https://www.ibm.com/docs/ru/aix/7.1?topic=information-example-program-lex-yacc-programs (дата обращения: 05.03,2024).

ПРИЛОЖЕНИЕ А

Листинг программы lex.l

/\* Declaration section\*/

%{

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <errno.h>

#include "grammar.tab.h"

extern FILE\* yyin;

int line = 1;

int line\_body=1;

int flag\_warning=1;

#ifndef unix

#define fileno \_fileno

#endif

%}

%option stack yylineno

%x attr\_exp

%x title\_attr\_exp

%x title\_text\_exp

%x comment\_continue

%x style\_section

%x style\_attr\_exp

%x script\_section

%x script\_attr\_exp

whitespace [ \t\r\n]

int [0-9]+

name [a-zA-Z\_]+[:a-zA-Z\_0-9\.\-]\*

str [a-zA-Z\_]+[ :a-zA-Z\_0-9\.]\*

tagDOCTYPE \![dD][oO][cC][tT][yY][pP][eE][ ]+[hH][tT][mM][lL]

tagHTML [hH][tT][mM][lL]

tagHEAD [hH][eE][aA][dD]

tagBODY [bB][oO][dD][yY]

tagTITLE [tT][iI][tT][lL][eE]

tagA [aA]

tagAbbr [aA][bB][bB][rR]

tagAddress [aA][dD][dD][rR][eE][sS][sS]

tagArticle [aA][rR][tT][iI][cC][lL][eE]

tagAside [aA][sS][iI][dD][eE]

tagAudio [aA][uU][dD][iI][oO]

tagB [bB]

tagBdi [bB][dD][iI]

tagBdo [bB][dD][oO]

tagBlockquote [bB][lL][oO][cC][kK][qQ][uU][oO][tT][eE]

tagButton [bB][uU][tT][tT][oO][nN]

tagCanvas [cC][aA][nN][vV][aA][sS]

tagCaption [cC][aA][pP][tT][iI][oO][nN]

tagCite [cC][iI][tT][eE]

tagCode [cC][oO][dD][eE]

tagColgroup [cC][oO][lL][gG][rR][oO][uU][pP]

tagData [dD][aA][tT][aA]

tagDatalist [dD][aA][tT][aA][lL][iI][sS][tT]

tagDd [dD][dD]

tagDl [dD][lL]

tagDt [dD][tT]

tagDel [dD][eE][lL]

tagDetails [dD][eE][tT][aA][iI][lL][sS]

tagDfn [dD][fF][nN]

tagDialog [dD][iI][aA][lL][oO][gG]

tagDiv [dD][iI][vV]

tagEm [eE][mM]

tagEmbed [eE][mM][bB][eE][dD]

tagFieldset [fF][iI][eE][lL][dD][sS][eE][tT]

tagFigcaption [fF][iI][g][cC][aA][pP][tT][iI][oO][nN]

tagFigure [fF][iI][gG][uU][rR][eE]

tagFooter [fF][oO][oO][tT][eE][rR]

tagForm [fF][oO][rR][mM]

tagH [hH]1|[hH]2|[hH]3|[hH]4|[hH]5|[hH]6

tagHeader [hH][eE][aA][dD][eE][rR]

tagHgroup [hH][gG][rR][oO][uU][pP]

tagI [iI]

tagIframe [iI][fF][rR][aA][mM][eE]

tagIns [iI][nN][sS]

tagKbd [kK][bB][dD]

tagKeygen [kK][eE][yY][gG][eE][nN]

tagLabel [lL][aA][bB][eE][lL]

tagLegend [lL][eE][gG][eE][nN]

tagLi [lL][iI]

tagMap [mM][aA][pP]

tagMain [mM][aA][iI][nN]

tagMark [mM][aA][rR][kK]

tagMenu [mM][eE][nN][uU]

tagMeter [mM][eE][tT][eE][rR]

tagNav [nN][aA][vV]

tagObject [oO][bB][jJ][eE][cC][tT]

tagOl [oO][lL]

tagOptgroup [oO][pP][tT][gG][rR][oO][uU][pP]

tagOption [oO][pP][tT][iI][oO][nN]

tagOutput [oO][uU][tT][pP][uU][tT]

tagP [pP]

tagPicture [pP][iI][cC][tT][uU][rR][eE]

tagPre [pP][rR][eE]

tagProgress [pP][rR][oO][gG][rR][eE][sS][sS]

tagQ [qQ]

tagRp [rR][pP]

tagRt [rR][tT]

tagRuby [rR][uU][bB][yY]

tagSamp [sS][aA][mM][pP]

tagSection [sS][eE][cC][tT][iI][oO][nN]

tagSelect [sS][eE][lL][eE][cC][tT]

tagSmall [sS][mM][aA][lL][lL]

tagSpan [sS][pP][aA][nN]

tagStrong [sS][tT][rR][oO][nN][gG]

tagSub [sS][uU][bB]

tagSup [sS][uU][pP]

tagSummary [sS][uU][mM][mM][aA][rR][yY]

tagSvg [sS][vV][gG]

tagTable [tT][aA][bB][lL][eE]

tagTbody [tT][bB][oO][dD][yY]

tagTd [tT][dD]

tagTemplate [tT][eE][mM][pP][lL][aA][tT][eE]

tagTextarea [tT][eE][xX][tT][aA][rR][eE][aA]

tagTfoot [tT][fF][oO][oO][tT]

tagTh [tT][hH]

tagThead [tT][hH][eE][aA][dD]

tagTime [tT][iI][mM][eE]

tagTr [tT][rR]

tagTrack [tT][rR][aA][cC][kK]

tagUl [uU][lL]

tagVar [vV][aA][rR]

tagVideo [vV][iI][dD][eE][oO]

tagWbr [wW][bB][rR]

tagsAll {tagA}|{tagAbbr}|{tagAddress}|{tagArticle}|{tagAside}|{tagAudio}|{tagB}|{tagBdi}|{tagBdo}|{tagBlockquote}|{tagButton}|{tagCanvas}|{tagCaption}|{tagCite}|{tagCode}|{tagColgroup}|{tagData}|{tagDatalist}|{tagDd}|{tagDt}|{tagDl}|{tagDel}|{tagDetails}|{tagDfn}|{tagDialog}|{tagDiv}|{tagEm}|{tagEmbed}|{tagFieldset}|{tagFigure}|{tagFigcaption}|{tagFooter}|{tagForm}|{tagH}|{tagHeader}|{tagHgroup}|{tagI}|{tagIframe}|{tagIns}|{tagKbd}|{tagKeygen}|{tagLabel}|{tagLegend}|{tagLi}|{tagMap}|{tagMark}|{tagMain}|{tagMenu}|{tagMeter}|{tagNav}|{tagObject}|{tagOl}|{tagOptgroup}|{tagOption}|{tagOutput}|{tagP}|{tagPicture}|{tagPre}|{tagProgress}|{tagQ}|{tagRp}|{tagRt}|{tagRuby}|{tagSamp}|{tagSection}|{tagSelect}|{tagSmall}|{tagSpan}|{tagStrong}|{tagSub}|{tagSup}|{tagSummary}|{tagSvg}|{tagTable}|{tagTbody}|{tagTd}|{tagTemplate}|{tagTextarea}|{tagTfoot}|{tagTh}|{tagThead}|{tagTime}|{tagTr}|{tagTrack}|{tagUl}|{tagVar}|{tagVideo}|{tagWbr}

generalTags [nN][oO][sS][cC][rR][iI][pP][tT]

styleTag [sS][tT][yY][lL][eE]

scriptTag [sS][cC][rR][iI][pP][tT]

headTagsVoid [mM][eE][tT][aA]|[bB][aA][sS][eE]|[lL][iI][nN][kK]

voidTags [bB][rR]|[hH][rR]|[iI][nN][pP][uU][tT]|[aA][rR][eE][aA]|[cC][oO][lL]|[iI][mM][gG]|[pP][aA][rR][aA][mM]|[cC][oO][mM][mM][aA][nN][dD]|[sS][oO][uU][rR][cC][eE]

oldVersionTags [aA][cC][rR][oO][nN][yY][mM]|[aA][pP][pP][lL][eE][tT]|[bB][lL][iI][nN][kK]|[bB][aA][sS][eE][fF][oO][nN][tT]|[bB][gG][sS][oO][uU][nN][dD]|[bB][iI][gG]|[cC][oO][mM][mM][eE][nN][tT]|[cC][eE][nN][tT][eE][rR]|[dD][iI][rR]|[fF][oO][nN][tT]|[fF][rR][aA][mM][eE]|[fF][rR][aA][mM][eE][sS][eE][tT]|[nN][oO][fF][rR][aA][mM][eE][sS]|[sS][tT][rR][iI][kK][eE]|[tT][tT]|[kK][eE][yY][gG][eE][nN]

attributes1 [aA][cC][cC][eE][pP][tT]|[aA][cC][cC][eE][pP][tT][-][cC][hH][aA][rR][sS][eE][tT]|[aA][cC][cC][eE][sS][sS][kK][eE][yY]|[aA][cC][tT][iI][oO][nN]|[aA][lL][tT]|[aA][sS][yY][nN][cC]|[aA][uU][tT][oO][cC][oO][mM][pP][lL][eE][tT][eE]|[aA][uU][tT][oO][fF][oO][cC][uU][sS]|[aA][uU][tT][oO][pP][lL][aA][yY]|[cC][hH][aA][rR][sS][eE][tT]|[cC][hH][eE][cC][kK][eE][dD]|[cC][iI][tT][eE]|[cC][lL][aA][sS][sS]|[cC][oO][lL][sS]|[cC][oO][lL][sS][pP][aA][nN]|[cC][oO][nN][tT][eE][nN][tT]

attributes2 [cC][oO][nN][tT][eE][nN][tT][eE][dD][iI][tT][aA][bB][lL][eE]|[cC][oO][nN][tT][rR][oO][lL][sS]|[cC][oO][oO][rR][dD][sS]|[dD][aA][tT][aA]|[dD][aA][tT][eE][tT][iI][mM][eE]|[dD][eE][fF][aA][uU][lL][tT]|[dD][eE][fF][eE][rR]|[dD][iI][rR]|[dD][iI][rR][nN][aA][mM][eE]|[dD][iI][sS][aA][bB][lL][eE][dD]|[dD][oO][wW][nN][lL][oO][aA][dD]|[dD][rR][aA][gG][gG][aA][bB][lL][eE]|[eE][nN][cC][tT][yY][pP][eE]|[fF][oO][rR]|[fF][oO][rR][mM]|[fF][oO][rR][mM][aA][cC][tT][iI][oO][nN]|[hH][eE][aA][dD][eE][rR][sS]

attributes3 [hH][eE][iI][gG][hH][tT]|[hH][iI][dD][dD][eE][nN]|[hH][iI][gG][hH]|[hH][rR][eE][fF]|[hH][rR][eE][fF][lL][aA][nN][gG]|[hH][tT][tT][pP][-][eE][qQ][uU][iI][vV]|[iI][dD]|[iI][sS][mM][aA][pP]|[kK][iI][nN][dD]|[lL][aA][bB][eE][lL]|[lL][aA][nN][gG]|[lL][iI][sS][tT]|[lL][oO][oO][pP]|[lL][oO][wW]|[mM][aA][xX]|[mM][aA][xX][lL][eE][nN][gG][tT][hH]|[mM][eE][dD][iI][aA]|[mM][eE][tT][hH][oO][dD]|[mM][iI][nN]|[mM][uU][lL][tT][iI][pP][lL][eE]|[mM][uU][tT][eE][dD]|[nN][aA][mM][eE]|[nN][oO][vV][aA][lL][iI][dD][aA][tT][eE]

attributes4 [rR][eE][aA][dD][oO][nN][lL][yY]|[rR][eE][lL]|[rR][eE][qQ][uU][iI][rR][eE][dD]|[rR][eE][vV][eE][rR][sS][eE][dD]|[rR][oO][wW][sS]|[rR][oO][wW][sS][pP][aA][nN]|[sS][aA][nN][dD][bB][oO][xX]|[sS][cC][oO][pP][eE]|[sS][eE][lL][eE][cC][tT][eE][dD]|[sS][hH][aA][pP][eE]|[sS][iI][zZ][eE]|[sS][iI][zZ][eE][sS]|[sS][pP][aA][nN]|[sS][pP][eE][lL][lL][cC][hH][eE][cC][kK]|[sS][rR][cC]|[sS][rR][cC][dD][oO][cC]|[sS][rR][cC][lL][aA][nN][gG]|[sS][rR][cC][sS][eE][tT]|[sS][tT][aA][rR][tT]|[sS][tT][eE][pP]

attributes5 [tT][aA][bB][iI][nN][dD][eE][xX]|[tT][aA][rR][gG][eE][tT]|[tT][rR][aA][nN][sS][lL][aA][tT][eE]|[tT][yY][pP][eE]|[uU][sS][eE][mM][aA][pP]|[vV][aA][lL][uU][eE]|[wW][iI][dD][tT][hH]|[wW][rR][aA][pP]|[pP][rR][oO][pP][eE][rR][tT][yY]|[sS][tT][yY][lL][eE]|[rR][oO][lL][eE]|[tT][iI][tT][lL][eE]

events1 [oO][nN][aA][bB][oO][rR][tT]|[oO][nN][aA][fF][tT][eE][rR][pP][rR][iI][nN][tT]|[oO][nN][bB][eE][fF][oO][rR][eE][pP][rR][iI][nN][tT]|[oO][nN][bB][eE][fF][oO][rR][eE][uU][nN][lL][oO][aA][dD]|[oO][nN][bB][lL][uU][rR]|[oO][nN][cC][aA][nN][pP][lL][aA][yY]|[oO][nN][cC][aA][nN][pP][lL][aA][yY][tT][hH][rR][oO][uU][gG][hH]|[oO][nN][cC][hH][aA][nN][gG][eE]|[oO][nN][cC][lL][iI][cC][kK]|[oO][nN][cC][oO][nN][tT][eE][xX][tT][mM][eE][nN][uU]|[oO][nN][cC][oO][pP][yY]

events2 [oO][nN][cC][uU][eE][cC][hH][aA][nN][gG][eE]|[oO][nN][cC][uU][tT]|[oO][nN][dD][bB][lL][cC][lL][iI][cC][kK]|[oO][nN][dD][rR][aA][gG]|[oO][nN][dD][rR][aA][gG][eE][nN][dD]|[oO][nN][dD][rR][aA][gG][eE][nN][tT][eE][rR]|[oO][nN][dD][rR][aA][gG][lL][eE][aA][vV][eE]|[oO][nN][dD][rR][aA][gG][oO][vV][eE][rR]|[oO][nN][dD][rR][aA][gG][sS][tT][aA][rR][tT]|[oO][nN][dD][rR][oO][pP]|[oO][nN][dD][uU][rR][aA][tT][iI][oO][nN][cC][hH][aA][nN][gG][eE]|[oO][nN][eE][mM][pP][tT][iI][eE][dD]|[oO][nN][eE][nN][dD][eE][dD]

events3 [oO][nN][eE][rR][rR][oO][rR]|[oO][nN][fF][oO][cC][uU][sS]|[oO][nN][hH][aA][sS][hH][cC][hH][aA][nN][gG][eE]|[oO][nN][iI][nN][pP][uU][tT]|[oO][nN][iI][nN][vV][aA][lL][iI][dD]|[oO][nN][kK][eE][yY][dD][oO][wW][nN]|[oO][nN][kK][eE][yY][pP][rR][eE][sS][sS]|[oO][nN][kK][eE][yY][uU][pP]|[oO][nN][lL][oO][aA][dD]|[oO][nN][lL][oO][aA][dD][eE][dD][dD][aA][tT][aA]|[oO][nN][lL][oO][aA][dD][eE][dD][mM][eE][tT][aA][dD][aA][tT][aA]|[oO][nN][lL][oO][aA][dD][sS][tT][aA][rR][tT]

events4 [oO][nN][mM][oO][uU][sS][eE][dD][oO][wW][nN]|[oO][nN][mM][oO][uU][sS][eE][mM][oO][vV][eE]|[oO][nN][mM][oO][uU][sS][eE][oO][uU][tT]|[oO][nN][mM][oO][uU][sS][eE][oO][vV][eE][rR]|[oO][nN][mM][oO][uU][sS][eE][uU][pP]|[oO][nN][mM][oO][uU][sS][eE][wW][hH][eE][eE][lL]|[oO][nN][oO][fF][fF][lL][iI][nN][eE]|[oO][nN][oO][nN][lL][iI][nN][eE]|[oO][nN][pP][aA][gG][eE][hH][iI][dD][eE]|[oO][nN][pP][aA][gG][eE][sS][hH][oO][wW]|[oO][nN][pP][aA][sS][tT][eE]|[oO][nN][pP][aA][uU][sS][eE]

events5 [oO][nN][pP][lL][aA][yY]|[oO][nN][pP][lL][aA][yY][iI][nN][gG]|[oO][nN][pP][oO][pP][sS][tT][aA][tT][eE]|[oO][nN][pP][rR][oO][gG][rR][eE][sS][sS]|[oO][nN][rR][aA][tT][eE][cC][hH][aA][nN][gG][eE]|[oO][nN][rR][eE][sS][eE][tT]|[oO][nN][rR][eE][sS][iI][zZ][eE]|[oO][nN][sS][cC][rR][oO][lL][lL]|[oO][nN][sS][eE][aA][rR][cC][hH]|[oO][nN][sS][eE][eE][kK][eE][dD]|[oO][nN][sS][eE][eE][kK][iI][nN][gG]|[oO][nN][sS][eE][lL][eE][cC][tT]|[oO][nN][sS][tT][aA][lL][lL][eE][dD]|[oO][nN][sS][tT][oO][rR][aA][gG][eE]

events6 [oO][nN][sS][uU][bB][mM][iI][tT]|[oO][nN][sS][uU][sS][pP][eE][nN][dD]|[oO][nN][tT][iI][mM][eE][uU][pP][dD][aA][tT][eE]|[oO][nN][tT][oO][gG][gG][lL][eE]|[oO][nN][uU][nN][lL][oO][aA][dD]|[oO][nN][vV][oO][lL][uU][mM][eE][cC][hH][aA][nN][gG][eE]|[oO][nN][wW][aA][iI][tT][iI][nN][gG]|[oO][nN][wW][hH][eE][eE][lL]|[oO][pP][eE][nN]|[oO][pP][tT][iI][mM][uU][mM]|[pP][aA][tT][tT][eE][rR][nN]|[pP][lL][aA][cC][eE][hH][oO][lL][dD][eE][rR]|[pP][oO][sS][tT][eE][rR]|[pP][rR][eE][lL][oO][aA][dD]

attributesALL {attributes1}|{attributes2}|{attributes3}|{attributes4}|{attributes5}|{events1}|{events2}|{events3}|{events4}|{events5}|{events6}

oldVersionAttr [aA][lL][iI][gG][nN]|[bB][gG][cC][oO][lL][oO][rR]|[bB][oO][rR][dD][eE][rR]|[cC][oO][lL][oO][rR]

attrWithoutValue [iI][tT][eE][mM][sS][cC][oO][pP][eE]|[aA][sS][yY][nN][cC]|[dD][eE][fF][eE][rR]

attributesValue \"[^<"]\*\")|("'"[^<']\*"'"

commentTagStart "\<\!\-\-"

commentTagEnd "\-\-\>"

cdataTag "\<!\[CDATA"

xmlTAG "<\?"[xX][mM][lL]

tagDOCTYPEold \![dD][oO][cC][tT][yY][pP][eE][ ]+[hH][tT][mM][lL][ ]+[pP][uU][bB][lL][iI][cC]

%%

\<{tagDOCTYPE}\> {

return TAG\_DOCKTYPE;

}

\<{tagDOCTYPEold} {

HandlerWarning("Old version of the docktype for HTML5");

return TAG\_DOCKTYPE;

}

{xmlTAG} {

HandlerError("XML declaration");

}

{cdataTag} {

HandlerError("CDATA declaration(XML)");

}

\<{tagHTML} {

BEGIN(attr\_exp);

return TAG\_HTML;

}

\<{tagHEAD} {

BEGIN(attr\_exp);

return TAG\_HEAD;

}

\<{tagBODY} {

line\_body=line;

BEGIN(attr\_exp);

return TAG\_BODY;

}

\<{tagTITLE} {

BEGIN(title\_attr\_exp);

return TAG\_TITLE;

}

\<{tagsAll} {

StackPush(yytext);

BEGIN(attr\_exp);

return COMMON\_TAG;

}

\<\/{tagHTML}{whitespace}\*\> {

return TAG\_HTML\_CLOSE;

}

\<\/{tagHEAD}{whitespace}\*\> {

return TAG\_HEAD\_CLOSE;

}

\<\/{tagBODY}{whitespace}\*\> {

return TAG\_BODY\_CLOSE;

}

\<\/{tagTITLE}{whitespace}\*\> {

return TAG\_TITLE\_CLOSE;

}

\<{generalTags} {

BEGIN(attr\_exp);

return GENERAL\_TAG;

}

\<\/{generalTags}{whitespace}\*\> {

return GENERAL\_TAG\_CLOSE;

}

\<{oldVersionTags} {

HandlerWarning("Old version of the tag for HTML5");

StackPush(yytext);

BEGIN(attr\_exp);

return COMMON\_TAG;

}

\<{voidTags} {

BEGIN(attr\_exp);

return COMMON\_VOID\_TAG;

}

\<{headTagsVoid} {

BEGIN(attr\_exp);

return HEAD\_SECTION\_VOID\_TAG;

}

\<{styleTag} {

BEGIN(style\_attr\_exp);

return TAG\_STYLE;

}

\<{scriptTag} {

BEGIN(script\_attr\_exp);

return TAG\_SCRIPT;

}

\<\/{tagsAll}{whitespace}\*\> {

StackCheck(yytext);

return COMMON\_TAG\_CLOSE;

}

\<\/{voidTags}{whitespace}\*\> {

HandlerWarning("Non-HTML5 construction");

}

\<{name} {

HandlerError("Unknown tag");

}

\<\/{name}{whitespace}\*\> {

HandlerError("Unknown end tag");

}

<attr\_exp>{

{attributesALL}{whitespace}\*[=] {

return ATTRIBUTE;

}

{oldVersionAttr}{whitespace}\*[=] {

HandlerWarning("Old version of the attribute for HTML5");

return ATTRIBUTE;

}

{name}{whitespace}\*[=] {

HandlerWarning("Unknown attribute");

return ATTRIBUTE;

}

{whitespace}\*{attributesValue} {

return VALUE;

}

[\n] {

line++;

}

\/\> {

HandlerWarning("Non-HTML5 construction - selfclosing void tag");

BEGIN(0);

return CLOSING\_MORE\_SIGN;

}

[>] {

BEGIN(0);

return CLOSING\_MORE\_SIGN;

}

{whitespace} ;

{attrWithoutValue}{whitespace}\* ;

. ;

}

<title\_attr\_exp>{

{attributesALL}{whitespace}\*[=] {

return ATTRIBUTE;

}

{oldVersionAttr}{whitespace}\*[=] {

HandlerWarning("Old version of the attribute for HTML5");

return ATTRIBUTE;

}

{name}{whitespace}\*[=] {

HandlerWarning("Unknown attribute");

return ATTRIBUTE;

}

{whitespace}\*{attributesValue} {

return VALUE;

}

[>] {

BEGIN(title\_text\_exp);

return CLOSING\_MORE\_SIGN;

}

[\n] {

line++;

}

{whitespace} ;

}

<title\_text\_exp>{

\<\/{tagTITLE}{whitespace}\*\> {

return TAG\_TITLE\_CLOSE;

}

[^ \t\r\n] {

BEGIN(0);

return TITLE\_TEXT;

}

. ;

}

<style\_attr\_exp>{

{attributesALL}{whitespace}\*[=] {

return ATTRIBUTE;

}

{oldVersionAttr}{whitespace}\*[=] {

HandlerWarning("Old version of the attribute for HTML5");

return ATTRIBUTE;

}

{name}{whitespace}\*[=] {

HandlerWarning("Unknown attribute");

return ATTRIBUTE;

}

{whitespace}\*{attributesValue} {

return VALUE;

}

[>] {

BEGIN(style\_section);

return CLOSING\_MORE\_SIGN;

}

[\n] {

line++;

}

{whitespace} ;

}

<style\_section>{

\<\/{styleTag}{whitespace}\*\> {

BEGIN(0);

return TAG\_STYLE\_CLOSE;

}

[\n] {

line++;

}

. ;

}

<script\_attr\_exp>{

{attributesALL}{whitespace}\*[=] {

return ATTRIBUTE;

}

{oldVersionAttr}{whitespace}\*[=] {

HandlerWarning("Old version of the attribute for HTML5");

return ATTRIBUTE;

}

{name}{whitespace}\*[=] {

HandlerWarning("Unknown attribute");

return ATTRIBUTE;

}

{whitespace}\*{attributesValue} {

return VALUE;

}

[>] {

BEGIN(script\_section);

return CLOSING\_MORE\_SIGN;

}

[\n] {

line++;

}

{whitespace} ;

{attrWithoutValue}{whitespace}\* ;

}

<script\_section>{

\<\/{scriptTag}{whitespace}\*\> {

BEGIN(0);

return TAG\_SCRIPT\_CLOSE;

}

[\n] {

line++;

}

. ;

}

{commentTagStart} {

BEGIN(comment\_continue);

return TAG\_COMMENT\_START;

}

<comment\_continue>{

{commentTagEnd} {

BEGIN(0);

return TAG\_COMMENT\_END;

}

[\n] {line++;}

. ;

}

{str} ;

[\n] {line++;}

[\r] ;

. ;

%%

void HandlerWarning(const char\* msg){

printf("WARNING: %s\n, line: %d\n", msg, line);

flag\_warning=0;

}

void HandlerError(const char\* msg){

printf("ERROR: %s\n, line: %d\n", msg, line);

exit(-1);

}

int yywrap(){}

void yyerror(char \*msg){

fprintf(stderr,"ERROR: %s\n", msg);

}

int main(int argc, char\*argv[]){

errno=0;

if (argc < 2){

HandlerError("Incorrect number of command arguments");

}

yyin = fopen(argv[1], "r");

if (yyin == NULL) {

HandlerError("The test file was not found");

}

StackInit();

yyparse();

StackDeinit();

fclose(yyin);

if(flag\_warning) printf("Program is correct.\n");

return 0;

}

ПРИЛОЖЕНИЕ Б

Листинг программы grammar.y

%{

#define YYERROR\_VERBOSE 1

#include <stdio.h>

extern int line;

%}

%token CLOSING\_MORE\_SIGN

%token TAG\_DOCKTYPE TAG\_COMMENT\_START TAG\_COMMENT\_END

%token TAG\_HTML TAG\_HTML\_CLOSE

%token TAG\_HEAD TAG\_HEAD\_CLOSE

%token GENERAL\_TAG GENERAL\_TAG\_CLOSE HEAD\_SECTION\_VOID\_TAG

%token TAG\_BODY TAG\_BODY\_CLOSE

%token COMMON\_TAG COMMON\_TAG\_CLOSE COMMON\_VOID\_TAG

%token TAG\_TITLE TAG\_TITLE\_CLOSE TITLE\_TEXT

%token TAG\_STYLE TAG\_STYLE\_CLOSE TAG\_SCRIPT TAG\_SCRIPT\_CLOSE

%token ATTRIBUTE VALUE

%%

html\_doc: TAG\_DOCKTYPE html\_section

| comments TAG\_DOCKTYPE html\_section

| error { yyerrok; printf("in line %d.\n", line); exit(-1);}

comments: TAG\_COMMENT\_START TAG\_COMMENT\_END

| comments TAG\_COMMENT\_START TAG\_COMMENT\_END

html\_section: html\_start head\_section body\_section TAG\_HTML\_CLOSE

| html\_start head\_section body\_section TAG\_HTML\_CLOSE comments

html\_start: TAG\_HTML payload

| comments TAG\_HTML payload

head\_section: head\_section\_start head\_section\_content title\_section head\_section\_content TAG\_HEAD\_CLOSE

head\_section\_start: TAG\_HEAD payload

| comments TAG\_HEAD payload

head\_section\_content:

| head\_section\_content GENERAL\_TAG payload head\_section\_content GENERAL\_TAG\_CLOSE

| head\_section\_content HEAD\_SECTION\_VOID\_TAG payload

| head\_section\_content TAG\_STYLE payload TAG\_STYLE\_CLOSE

| head\_section\_content TAG\_SCRIPT payload TAG\_SCRIPT\_CLOSE

| head\_section\_content TAG\_COMMENT\_START TAG\_COMMENT\_END

title\_section: TAG\_TITLE payload TITLE\_TEXT TAG\_TITLE\_CLOSE

body\_section: body\_section\_start body\_section\_content TAG\_BODY\_CLOSE

body\_section\_start: TAG\_BODY payload

| comments TAG\_BODY payload

body\_section\_content:

| body\_section\_content COMMON\_TAG payload body\_section\_content COMMON\_TAG\_CLOSE

| body\_section\_content GENERAL\_TAG payload body\_section\_content GENERAL\_TAG\_CLOSE

| body\_section\_content COMMON\_VOID\_TAG payload

| body\_section\_content TAG\_SCRIPT payload TAG\_SCRIPT\_CLOSE

| body\_section\_content TAG\_COMMENT\_START TAG\_COMMENT\_END

payload: CLOSING\_MORE\_SIGN

| atr\_sequence CLOSING\_MORE\_SIGN

atr\_sequence: ATTRIBUTE VALUE

| atr\_sequence ATTRIBUTE VALUE

ПРИЛОЖЕНИЕ В

Листинг программы stack.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "grammar.tab.h"

int line;

int line\_body;

typedef struct stack {

char tagName[20];

int line;

struct stack\* link;

} tagStack;

tagStack\* top;

tagStack\* root;

void StackPush(char\* value) {

tagStack\* tmp, \* tmp2;

tmp = (tagStack\*)malloc(sizeof(tagStack));

tmp2 = (tagStack\*)malloc(sizeof(tagStack));

for (int i = 0; i < strlen(value); i++)

tmp->tagName[i] = value[i];

tmp->tagName[strlen(value)] = '\0';

tmp2 = root;

while (tmp2->link != NULL)

tmp2 = tmp2->link;

tmp->link = top;

top = tmp;

tmp->line=line;

}

void StackPop() {

tagStack\* tmp;

if (top == NULL) {

printf("\nStack Underflow");

return;

}

tmp = top;

top = top->link;

tmp->link = NULL;

free(tmp);

}

int StackIsEmpty() {

return (top == NULL);

}

char\* StackTop() {

if (!StackIsEmpty())

return top->tagName;

else exit(1);

}

void StackPrint() {

tagStack\* tmp;

if (top == NULL) {

printf("\nStack Underflow");

exit(1);

}

tmp = top;

while (tmp != NULL) {

for(int i=1;i<20&&tmp->tagName[i]!='\0';i++) printf("%c", tmp->tagName[i]);

printf(", line: %d; ", tmp->line);

tmp = tmp->link;

}

}

void StackDeinit() {

if(!StackIsEmpty()){

printf("The following closing tags were not found due to program termination:\nbody, line: %d; ", line\_body);

StackPrint();

printf("\n");

}

while (!StackIsEmpty()) StackPop();

}

void StackInit() {

tagStack\* tmproot;

tmproot = (tagStack\*)malloc(sizeof(tagStack));

tmproot->link = NULL;

root = tmproot;

top = NULL;

}

void StackCheck(char\* data) {

if (StackIsEmpty()) {

printf("ERROR: there were no opening tags.\n");

exit(-1);

}

int i;

char newData[20];

newData[0] = '<';

for (i = 2; i < strlen(data); i++)

newData[i - 1] = data[i];

newData[i - 2] = '\0';

if (strcmp(newData, StackTop())) {

printf("ERROR: incorrect nesting of tags on line %d.\n", line);

printf("Expected: %s>; received: %s>.\n", StackTop(), newData);

StackDeinit();

exit(-1);

}

StackPop();

}

ПРИЛОЖЕНИЕ Г

Листинг Makefile

# Определение переменных

CT\_DIR := ./tests/CustomTests

CT\_FILES := $(wildcard $(CT\_DIR)/\*.txt)

TE\_DIR := ./tests/TestsWithErrors

TE\_FILES := $(wildcard $(TE\_DIR)/\*.txt)

S\_DIR := ./tests/Sites

S\_FILES := $(wildcard $(S\_DIR)/\*.txt)

STT\_DIR := ./tests/SingleTagTests

STT\_FILES := $(wildcard $(STT\_DIR)/\*.txt)

# Цель по умолчанию

all:

bison -d grammar.y --verbose

flex lex.l stack.cpp

cc lex.yy.c grammar.tab.c -o out

clean:

rm grammar.tab.c grammar.tab.h lex.yy.c grammar.output out

custom: $(CT\_FILES)

$(foreach file, $^, ./out $(file);)

single: $(STT\_FILES)

$(foreach file, $^, ./out $(file);)

sites: $(S\_FILES)

$(foreach file, $^, ./out $(file);)

err\_tests: $(TE\_FILES)

$(foreach file, $^, ./out $(file);)

.PHONY: all clean custom single sites err\_tests