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

# Цель

Главной целью данной 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 С MAKEFILE)

## Лексический анализатор

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.

### 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.

## Синтаксический анализатор

# Тестирование и результаты работы программы

# Выводы

Список используемых источников

1. <http://diveintohtml5.info/>
2. https://htmlbook.ru/html5

ПРИЛОЖЕНИЕ А

Листинг программы «Алгоритмы для разложения числа на множители»