Цель работы заключалась в написании программы, распознающей входной файл на языке html5, и выводящей информацию об ошибках. В ходе работы были реализованы распознающая грамматика и лексический анализатор.

1. Теоретические сведения

1.1 Средства анализа

Реализация компилятора для настоящего языка программирования всегда является весьма трудоемкой задачей, и поэтому давно предпринимались попытки автоматизировать этот процесс. Чаще всего объектом приложения таких усилий служили лучше всего изученные (и наиболее простые) части компилятора - сканер и парсер. В данной лабораторной работе будет использован YACC - для построения парсера, а также будет написан лексический анализатор.

Лексический анализатор – читает входной поток и передает лексемы (со значениями, если требуется) процедуре разбора. Лексический анализатор - это целочисленная функция с именем yylex. Функция возвращает номер лексемы, характеризующий вид прочитанной лексемы. Если этой лексеме соответствует значение, его надо присвоить внешней переменной yylval.

Вместо самостоятельной реализации функции yylex можно воспользоваться лексическим анализатором FLEX. Но в данной курсовой работе, как уже было сказано ранее, функция yylex будет написана самостоятельно.

YACC (синтаксический анализатор) – программа, которая строит восходящий LALR(1) распознаватель для данного языка. На вход программа принимает файл со спецификациями, на выходе – программа на языке C.

В результате построения грамматики могут возникнуть конфликты типа сдвиг-свертка или свертка-свертка. YACC по умолчанию использует два метода разрешения неоднозначностей:

1. Если приоритеты альтернативных действий различны, то выполняется действие с большим приоритетом

2. Если приоритеты действий одинаковы, то в случае их левой ассоциативности выбирается свертка, в случае правой ассоциативности - сдвиг. Если они не ассоциативны – возникает ошибка;

3. Если приоритета нет, то:

при конфликте сдвиг-свертка по умолчанию выполняется сдвиг;

при конфликте свертка-свертка по умолчанию выбирается правило, встретившееся в yacc-спецификации первым.

1.2 Язык HTML

HTML (Hyper Text Markup Language «язык гипертекстовой разметки») – стандартизированный язык разметки документов во Всемирной паутине. Изначально язык HTML создавался как язык для обмена научной и технической документацией. Сейчас язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

На данный момент существует множество версий данного языка. Последняя 5-ая версия была рекомендована к применению с 1 ноября 2016 года. В данной курсовой работе будет произведен анализ языка версии 5.

Цель разработки HTML5 – улучшение уровня поддержки мультимедиа-технологий с одновременным сохранением обратной совместимости, удобочитаемости кода для человека и простоты анализа для парсеров. В HTML5 реализовано множество новых синтаксических особенностей, например, для упрощения создания и управления графическими и мультимедийными объектами в сети без необходимости использования сторонних API и плагинов.

Код HTML состоит из следующих элементов:

1. Теги (одиночные, парные)

Парный тег состоит из двух частей: открывающей и закрывающей. Между этими двумя частями, пишется весь остальной код страницы, которая будет отображаться на экране. Тег <html> сообщает браузеру, что это html-документ, и является основным (родительским) тегом для всех остальных элементов.

2. Атрибуты тегов

Атрибут — это дополнительная команда. Пишется он в открывающей части тега.

3. Значения атрибутов.

Программа, написанная на языке HTML, обязательно имеет определенную структуру:

4. Любой код разметки начинается с <!DOCTYPE html>, этот элемент говорит браузеру, на каком языке разметки и его версии написан документ. В HTML5 нет делений, элемент doctype один единственный и при его наличии браузер работает в стандартном режиме.

5. Тег <html> является контейнером, который заключает в себе все содержимое веб-страницы, включая теги <head>и <body>.

6. Тег <head> предназначен для хранения других элементов, цель которых — помочь браузеру в работе с данными. Также внутри контейнера <head> находятся метатеги, которые используются для хранения информации предназначенной для браузеров и поисковых систем.

Например, механизмы поисковых систем обращаются к метатегам для получения описания сайта, ключевых слов и других данных.

Содержимое тега <head> не отображается напрямую на веб-странице, за исключением тега <title>устанавливающего заголовок окна веб-страницы.

Структура документа HTML:

```
<!DOCTYPE html>
<html>
    <head>
        <title></title>
    </head>
    <body>
    </body>
</html>
```

2. Результаты работы

2.1 Распознающая грамматика

Построение грамматики было выполнено снизу-вверх. Изначально были определены токены (терминалы). Были выделены основные элементы HTML кода: теги, атрибуты. Так как тег может быть с атрибутами или без, одиночный или парный, а закрывающий тег вовсе не содержит атрибутов, были определены 5 терминалов:

Single_Tag_No_Attribute – одиночный тег без атрибутов;

Pair_Tag_No_Attribute_Start – открывающий тег без атрибутов;

Single_Tag_With_Attribute – одиночный тег с атрибутами;

Pair_Tag_Start_With_Attribute – открывающий тег с атрибутами;

Tag_End – закрывающий тег.

Далее были определены нетерминалы, такие как:

singletag – одиночный тег (с атрибутами или без);

startpairtag – открывающий тег (с атрибутами или без);

endpairtag – закрывающий тег.

Singletag может представлять из себя, как тег без атрибутов (Single_Tag_No_Attribute), так и тег с атрибутами (Single_Tag_With_Attribute). Startpairtag, аналогично singletag’у, может быть либо тегом без атрибутов (Pair_Tag_No_Attribute_Start), либо тегом с атрибутами (Pair_Tag_Start_With_Attribute). Закрывающий тег для удобства восприятия был вынесен в отдельное правило endpairtag.

Так как весь HTML документ может состоять из парных тегов, между которыми может быть какой-либо блок (текст, скрипт и т.п.), или из одиночных, было добавлено следующее правило:

stat: singletag | startpairtag list endpairtag,

где list – входной файл; stat – строка во входном файле;

Также в грамматике происходит проверка каждого атрибута на принадлежность его к определенному тегу. Для этого была реализована функция TestAttribute: сначала ищется тег в массиве тегов и атрибутов (в цикле сверяется имя тега с каждым элементом первого столбца), далее в цикле сравнивается исходный атрибут с каждым атрибутом из строки текущего тега.

Грамматика:

```
list:
      | list stat

      | list '\n';
stat:
      singletag

      | startpairtag list endpairtag ;

singletag:

      Single_Tag_No_Attribute

      | Single_Tag_With_Attribute ;

startpairtag:

      Pair_Tag_No_Attribute_Start

      | Pair_Tag_Start_With_Attribute ;

endpairtag: Tag_End;
```

2.2 Лексический анализатор

Работа анализатора заключается в следующем:

Из входного потока считывается по одному символу, который передается в функцию Parse(). Функция Parse() проверяет, являются ли считанный символ и следующий за ним критерием начала комментария, если да, то управление передается в функцию comment(), которая обрабатывает комментарии (пропускает их). Если нет, то далее проверяется, являются ли считанный символ и следующий за ним началом тега, и если являются, то в массив str[] записываются все символы, пока не встретим критерий окончания тега ‘>’. Если же функция Parse() не определила ни комментарий, ни тег, то управление передается обратно в yylex().

Следующим шагом является обработка уже самой считанной строки (тег с атрибутами или без). Для этого управление передается в функцию ParseTagAndAttribute, в которой происходит поиск имени тега, имени атрибутов, пропуск знаков ‘\n’ и пробелов между атрибутами, а также пропуск значений атрибутов.

Теперь строка разбита на имя тега и имена атрибутов (если таковые имеются). Далее имя тега и имена атрибутов передаются в грамматику, в соответствии со значением тега (одиночный тег без атрибутов, одиночный тег с атрибутами, открывающий тег без атрибутов, открывающий тег с атрибутами, закрывающий тег).

Так как в HTML, как и в любом другом языке, есть множество нюансов, которые нужны учитывать (например, такие как: незакрытый тег, недопустимое имя тега или атрибута, неверное расположение тега), были реализованы следующие проверки:

1. Проверка на не существующий тег.
2. Проверка на не существующий атрибут (описано выше, проверяется в грамматике).
3. Проверка на незакрытый тег.
4. Проверка основных тегов.

Для реализации 1 и 2 пунктов был создан двумерный массив, в котором первый столбец – это сам тег, все последующие – атрибуты, разрешенные для данного тега. Массив был создан по стандарту HTML5 (ссылка на источник приведена в конце документа).

В функции ParseTagAndAttribute, после определения имени тега, происходит его проверка (в цикле сверяется имя тега с каждым элементом первого столбца из массива тегов и атрибутов). Если тег найден в списке тегов, то управление передается обратно в функцию yylex(), если же не найден, то вывод ошибки “неверный тег” с номером строки.

Для реализации 3 был создан линейный список:

```
typedef struct list
{
     char tag[25];
     struct list *ptr;
} lst;
lst *head = NULL;
```

Изначально все открывающие теги подряд записываются в список, как только встречается закрывающий тег, из списка в обратном порядке извлекаются теги и каждый закрывающий тег сверяется с соответствующим ему открывающим. Если имена тегов не совпадают, то выводится информация об ошибке незакрытого тега с номером строки.

Так как у HTML документа есть определенная структура, которая была приведена ранее, необходимо выполнять проверки, чтобы эта структура не была нарушена. Для этого, как только

встречаются теги html, head, title или body, устанавливается флаг. Далее каждый раз, когда мы встретим снова один из этих тегов, проверяется, не установлены ли определенные флаги, и если установлены, то происходит вывод об ошибке.