**СОДЕРЖАНИЕ**

[**ЦЕЛЬ РАБОТЫ** 3](#_Toc151463599)

[**ЗАДАЧИ** 4](#_Toc151463600)

[**1.** **ХОД РАБОТЫ** 5](#_Toc151463601)

[**1.1.** **Теоретические сведения** 5](#_Toc151463602)

[**1.2.** **Разработка лексического анализатора** 7](#_Toc151463603)

[**1.3.** **Разработка синтаксического анализатора** 10](#_Toc151463604)

[**1.4.** **Конфликты грамматики** 12](#_Toc151463605)

[**1.5.** **Работоспособность парсера** 14](#_Toc151463606)

[**ВЫВОД** 16](#_Toc151463607)

**ЦЕЛЬ РАБОТЫ**

Разработать средство проверки корректности разметки веб-документов в формате XML с помощью генератора лексических анализаторов FLEX и генератора синтаксических анализаторов Bison.

# **ЗАДАЧИ**

Для достижения поставленных целей необходимо:

1. Изучить особенности языка XML и его основные сущности;
2. Реализовать лексический анализатор с помощью средств Flex/Lex, который распознает лексемы и передает в анализатор;
3. Разработать грамматику для генератора синтаксических анализаторов Bison;
4. Выполнить тестирование полученного валидатора;
   * + 1. **ХОД РАБОТЫ**
   1. **Теоретические сведения**

XML (eXtensible Markup Language – расширяемый язык разметки) – это формальный язык разметки, который позволяет описывать структурированные данные в формате, который может быть прочитан как человеком, так и компьютером. Он широко используется в Интернете для обмена данных и представления документов. Язык называется расширяемым, поскольку он не фиксирует разметку, используемую в документах: разработчик волен создать разметку в соответствии с потребностями к конкретной области, будучи ограниченным лишь синтаксическими правилами языка.

XML документы имеют физическую и логическую структуру. С физической точки зрения документ состоит из одной или нескольких **сущностей**, каждая из которых может отсылать на другую сущность. Каждый XML документ имеет ровно одну сущность, называемую сущностью документа, которая служит стартовой точкой для XML процессора. Все сущности имеют содержание - символы (исключение составляет сущность документа), а также идентифицируются по имени. Символы разделены на два типа: **символы разметки** (начальные тэги, конечные тэги, тэги пустых элементов, ссылки на сущности, ссылки на символы, комментарии, разделители секций CDATA, объявление типов документов, инструкции обработки, декларации XML, декларации текста) и **символы данных** (все остальные части документа). С логической точки зрения, документ строится из деклараций, элементов, комментариев, ссылок на символ и инструкций обработки. Все они размечаются в документе явным образом. Логические и физические структуры должны иметь корректную вложенность.

XML использует теги, атрибуты и значения для описания данных. Теги – это имена, заключенные в угловые скобки (например, <имя>). Атрибуты – это дополнительная информация, которая может быть привязана к тегам (например, <имя атрибута="значение">). Значения – это данные, расположенные между открывающим и закрывающим тегами (например, <имя>значение</имя>). В связи с физической структурой XML наличие символов («<», «>», «'», «”», «&») в поле “значение” вызывает проблемы у синтаксического анализатора с пониманием результирующих данных. Решение состоит в том, чтобы экранировать управляющие символы, чтобы анализатор мог правильно интерпретировать их как данные и не путать с разметкой:

* ‘<’ - &lt;
* ‘>’ - &gt;
* ‘&’ - &amp;
* ‘ - &apos;
* “ - &quot;

Одна из особенностей XML заключается в возможности представления документа в виде древовидной структуры, которая начинается с "корневого" элемента и разветвляется на "дочерние" элементы. Любой XML документ должен содержать корневой элемент.

Как итог, можно выделить следующие основные правила синтаксиса XML:

1. Все элементы XML файла должны иметь закрывающий тэг;
2. Тэги регистрозависимы;
3. XML элементы должны соблюдать корректную вложенность;
4. Если есть XML пролог, то это обязательно должна быть первая строчка;
5. Значения XML атрибутов должны заключаться в кавычки;
6. Наличие особых символов (<, >, &, ’, “), имеющих специальные обозначения;
   1. **Разработка лексического анализатора**

Был выделен ряд неизменяемых единиц языка (терминальных символов), которые в дальнейшем при парсинге входного файла будет возвращать функция yylex().

Опираясь на спецификацию пятой редакции к XML, были выделены следующие регулярные выражения для описания общих синтаксических конструкций и некоторых параметров языка:

Таблица 1. Ключевые конструкции и параметры

| **Название** | **Конструкция** |
| --- | --- |
| Один пробельный символ | [\n\r\t ] |
| Один и более пробельных символов | {S1}+ |
| Ссылка на символ | "&#"[0-9]+";"|"&#x"[0-9a-fA-F]+";" |
| Равенство | {S}?"="{S}? |
| Имя атрибута | ([\_:]|[\x41-\x5A]|[\x61-\x7A]|[\xC0-\xD6]|[\xD8-\xF6]|[\xF8-\xFF])(([\x41-\x5A]|[\x61-\x7A]|[\xC0-\xD6]|[\xD8-\xF6]|[\xF8-\xFF])|[0-9.\-\_:])\* |
| Символьные данные | [^<]\* |
| Значение атрибута | \"[^"]\*\"|\'[^']\*\' |
| Объявления типа документа | {DT\_IdentSE}({S})?(\[({DT\_ItemSE})\*]({S})?)? |
| Имя типа документа и возможные внешние идентификаторы | {S}{Name}({S}({Name}|{QuoteSE}))\* |
| Сканирование элементов в теле объявлений типов документов | \<(!(--{Until2Hyphens}>|[^-]{MarkupDeclCE})|\?{Name}({PI\_Tail}))|%{Name};|{S} |
| Одинарные или двойные кавычки | \"[^"]\*\"|'[^']\*' |
| Инструкция обработки (PI) | \?>|{S1}{UntilQMs}([^>?]{UntilQMs})\*> |
| Последовательность вопросительных знаков | [^?]\*\?+ |
| Выражение до дефиса | [^-]\*- |
| Тело комментария | {UntilHyphen}([^-]{UntilHyphen})\*- |
| Внутренний наборы деклараций | ([^\]"'><]+|{QuoteSE})\*> |

С помощью анализатора flex и регулярных выражений были заданы следующие терминалы:

1. "<?"[Xx][Mm][Ll] – начало декларации XML (STARTXMLDECL);
2. "<!DOCTYPE" – начало декларация типа документа (START\_DOCKTYPE);
3. {S}"version"{Eq25} – параметр версии XML (VERSIONINFO);
4. {S}"encoding"{Eq25} – декларация кодировки (ENCODING);
5. {S}"standalone"{Eq25} – декларация автономного документа (STANDALONE);
6. {S}?"?>" – конец декларации XML (ENDXMLDECL);
7. \"[^\"]+\" – терминальная строка (TERSTRING);
8. [\-][\-][>] – конец комментария (PICOMMENT);
9. "<?"[Xx][Mm][Ll]{S}({S}|([^?]|("?"[^>])))\*"?>"|"<?"[Xx][Mm][Ll]"?>" – неверная инструкция обработки (PIERROR);
10. "<?"({S}|([^?]|("?"[^>])))\*"?>" – верная инструкция обработки (PI);
11. "<" – символ начала тега (OPEN\_TAG);
12. ">" – символ конца тега (CLOSE\_TAG);
13. "/" – косая черта (SLASH);
14. {Eq25} – равенство (Eq);
15. {Name} – имя атрибута (ID);
16. {attvalue} – значение атрибута (ATTVALUE);
17. \> - конец декларации DOCTYPE (END\_DOCKTYPE);

Кроме того, был применен механизм начальных условий для обработки комментариев, декларации типа документа, декларации XML, «тела» XML документа, а также объявления элементов.

Изначально лексический анализатор находится в состоянии <INITIAL>, в котором он обязательно должен встретить первым параметром начало декларации XML и перейти в состояние <STXMLDECL> для проверки тела корневого тега, иначе документ считается недопустимым. Далее происходит переход в состояние <XMLDECL\_END>, где обрабатывается «тело» XML файла, а также декларация типа документа (DOCKTYPE). Затем он может перейти в состояние <COMMENT> и начать проверку комментария на наличие двойного тире, либо в состояние <IN\_TAG> для проверки элементов (пустых тегов или тегов с атрибутами) на наличие особых тегов и их корректной вложенности.

Лексический анализатор проверяет наличие особых символов (<, >, &, ’, “) в поле “значение” атрибута и предупреждает о необходимости экранирования данных символов.

* 1. **Разработка синтаксического анализатора**

Здесь определяются структуры, глобальные переменные функции, нетерминалы и сама грамматика. Грамматика разработана на основе спецификации Extensible Markup Language (XML) 1.0 (Fifth Edition) (W3C Recommendation).

Таблица 2 – Правила, описывающие структуру языка XML

| **Правило** | **Описание** |
| --- | --- |
|  | Описывает документ XML |
|  | Описывает элемент «тела» XML |
|  | Описывает начальный тэг |
|  | Описывает тэг пустого элемента |
|  | Описывает ноль или больше атрибутов |
|  | Описывает определение атрибута |
|  | Описывает конечный тэг |
|  | Описывает пролог |
|  | Описывает содержимое элементов |
|  | Описывает декларацию XML |
|  | Описывает декларацию типа документа |
|  | Описывает ноль или больше необязательных пробелов, комментариев и PI |
|  | Описывает комментарий, PI |

Для разработанного парсера характерны проверки на следующего рода ошибки:

* использование строки “?xml” в PI;
* пустая декларацию типа документа;
* неверная кодировка, версия XML и определение автономного документа;
* отсутствие декларации XML в начале файла;
* отличие открывающего и закрывающего тега;
* неправильная структура XML (документ начинается не с обязательного пролога);
* пустой файл XML;
  1. **Конфликты грамматики**

В финальной версии грамматики присутствует 1 конфликт shift/reduce. При сборке Perser.tab.c был добавлен флаг –v, который генерирует файл состояний y.output. Благодаря этому можно детально узнать о всех конфликтах грамматики (см. Рисунок 1).

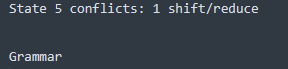


Рисунок 1 - Конфликты финальной версии программы.

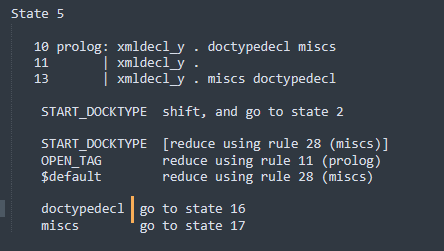


Рисунок 2 - Состояние 5, в котором был найден конфликт

Конфликт в данном состоянии возникает из-за наличия двух правил, по которым анализатор при получении токена START\_DOCKTYPE может произвести свертку по правилу 28 (пустой misc) или сдвинуть на анализ нетерминального символа docktypedecl. Т.е. при отсутствии комментария или PI между декларацией XML и декларацией типа документа анализатор выберет произвести сдвиг в состояние 2. Решением данного конфликта является либо запрет miscs быть пустым, либо явное указание наличия одного misc между декларацией XML и декларацией типа документа.

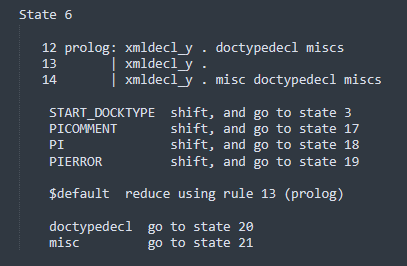


Рисунок 3 - Решение конфликат сдвиг/свертка

* 1. **Работоспособность парсера**

При реализации валидатора в качестве основы, на которую можно опираться при разработке, была выбрана спецификация XML 1.0 пятой редакции. Приведем различные примеры выводимых сообщений (см. Рисунки 4-10):

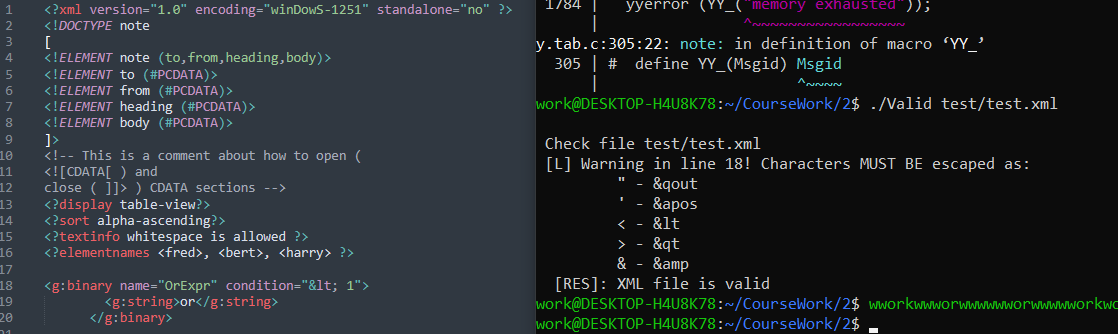
****

Рисунок 4 - В названии атрибута особый символ.

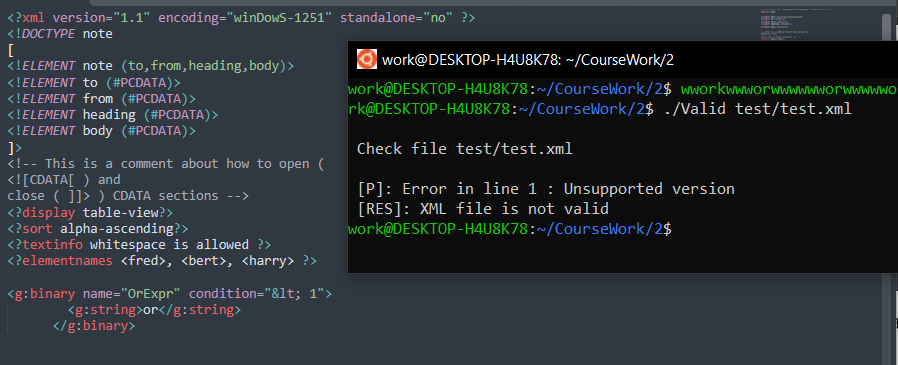


Рисунок 5 - Ошибка в версии XML

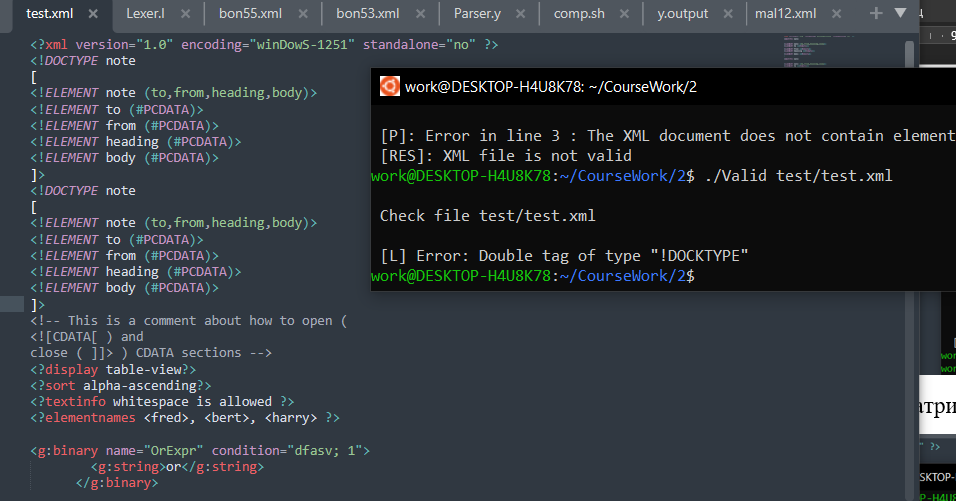


Рисунок 6 - Двойное объявление Docktype

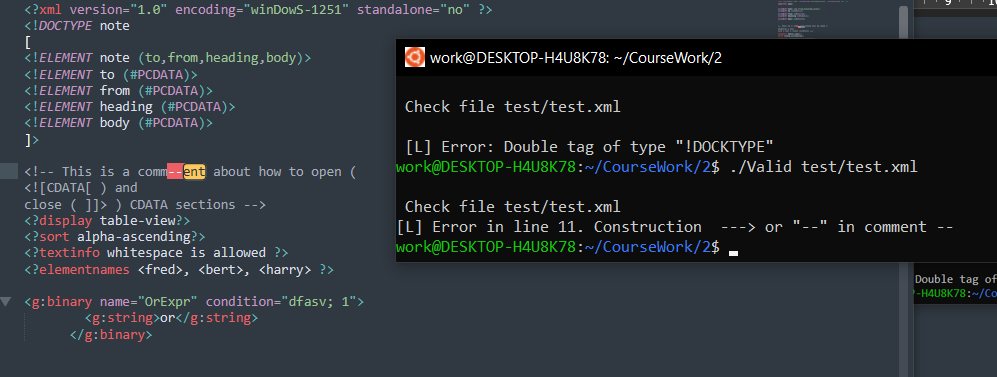


Рисунок 7 - Двойное тире в комментарии

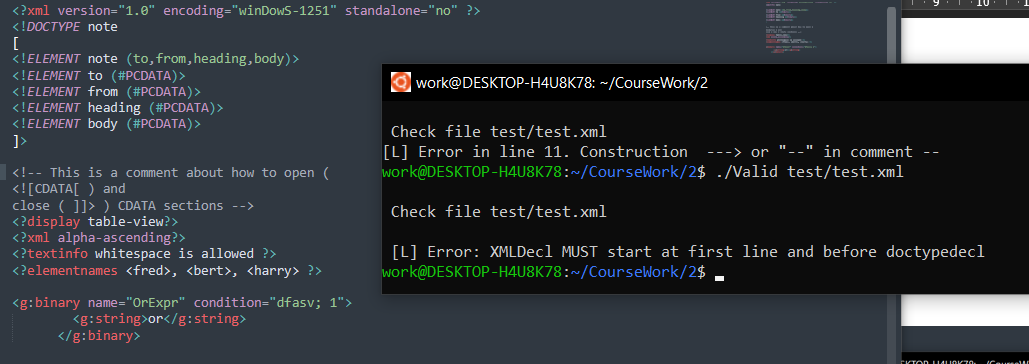


Рисунок 8 - PI начинается с <?xml

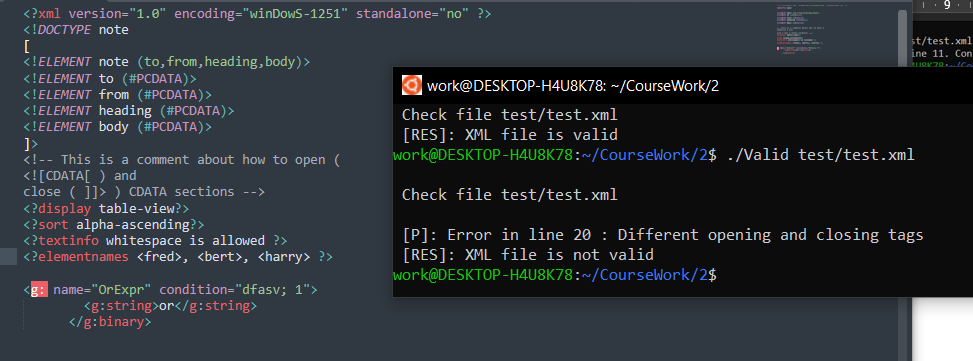


Рисунок 9 - Разный открывающий и закрывающий теги

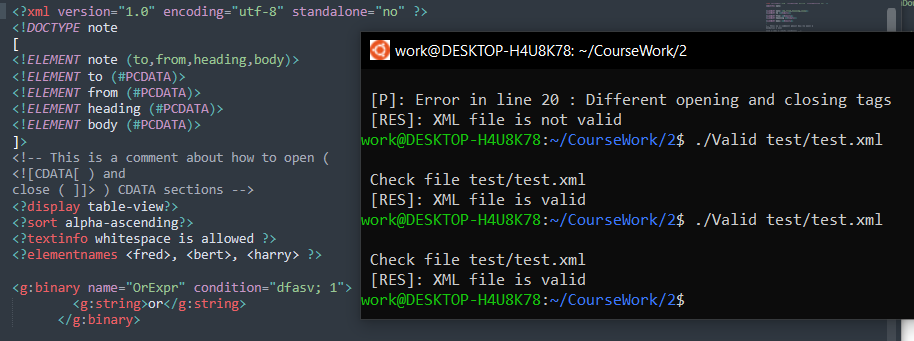


Рисунок 10 - Пример валидного XML файла.

# **ВЫВОД**

Во время выполнения курсовой работы была подробно изучена спецификация языка XML, а также были получены навыки работы со средствами разработки парсеров языков разметки, а также построения грамматик, которые позволят распознать принадлежность входного кода к существующему языку. В конечном итоге был получен инструмент для валидации XML файлов общей структуры, проверяющий и выдающий соответствующие сообщения о большом количестве ошибок.

При разработке удалось избежать конфликтов типа reduce/reduce, так как большинство подобных ошибок возникало из-за некорректной обработки комментариев и инструкций процессора. Для одного конфликта shift/reduce было составлено альтернативное правило, описывающее ту же самую конструкцию без конфликта.

Существующую реализацию также можно дополнить обработкой идентификации языка, реализацией обработки условных секций <!ENTITY, а также реализовать полную поддержку DTD, но для этого необходимо дополнительно парсить и анализировать файлы .dtd.