МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Специализация Программирование интернет-приложений

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Разработка компилятора CHAW-2017»

Выполнил студент Чаевский Андрей Владимирович

(Ф.И.О.)

Руководитель проекта ст.пр. Наркевич Аделина Сергеевна

(учен. степень, звание, должность, подпись, Ф.И.О.)

Заведующий кафедрой к.т.н., доц. Пацей Н.В.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Консультанты ст.пр. Наркевич Аделина Сергеевна

(учен. степень, звание, должность, подпись, Ф.И.О.)

(учен. степень, звание, должность, подпись, Ф.И.О.)

Нормоконтролер ст.пр. Наркевич Аделина Сергеевна

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПЕУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий   
Кафедра программной инженерии

Утверждаю

Зав. кафедрой ИП

\_\_\_\_\_\_\_\_\_\_\_\_ Н.В. Пацей

подпись инициалы и фамилия

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2017г.

**ЗАДАНИЕ**

**к курсовому проектированию**

**по дисциплине** "Языки программирования"

Специальность: 1-40 01 01 Программное обеспечение информационных технологий

Студент: Чаевский Андрей Владимирович Группа: 4

(фамилия, имя, отчество)

**1. Тема проекта** Разработка компилятора CHAW-2017

утверждена приказом по университету от «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г. №

**2. Срок сдачи студентом законченного проекта:** \_\_\_ декабря 2017 г.

**3. Исходные данные к проекту:**

Разработка программы осуществляется на языке C++ (стандартизации International Standard ISO/IEC 14882:2014(E) Programming Language C++ 14) в среде разработки Microsoft Visual Studio Enterprise 2017. Операционная система под которой происходит разработка Windows 10 (64-bit). Типы данных: dig и chr. Функции стандартной библиотеки: dig strl(chr) – длина строки, dig power(dig, dig) – подстрока. Арифметические операции: +, -, \*. Оператор вывода в стандартный поток: out. Поддерживается условный оператор if … then … else… .

**4. Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов):**

Введение

1) Спецификация языка программирования

2) Структура транслятора

3) Разработка лексического анализатора

4) Разработка синтаксического анализатора

5) Разработка семантического анализатора

6) Вычисление выражений

7) Генерация кода

8) Контрольный пример

9) Тестирование транслятора (разработка и тестирование интерпретатора)

Приложения

Литература

**5. Перечень графического материала (с точным указанием обязательных чертежей)**

Дерево разбора синтаксического анализатора

**6. Консультанты по проекту с указанием относящихся к ним разделов проекта**

|  |  |
| --- | --- |
| Раздел | Консультант |
| Разработка синтаксического и семантического анализатора. | Наркевич А. С. |
| Генерация кода. Разработка тестовых примеров. | Наркевич А. С. |
| Оформление пояснительной записки к курсовому проекту. | Наркевич А. С. |

**7. Календарный план**

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование этапов курсового проекта | Срок выполнения этапов проекта | Примечание |
| 1 | Спецификация специализированного языка CHAW-2017 | 20.10.2017 |  |
| 2 | Разработка лексического анализатора | 10.11.2017 |  |
| 3 | Разработка синтаксического анализатора | 01.12.2017 |  |
| 4 | Разработка семантического анализатора | 01.12.2017 |  |
| 5 | Генерация кода | 08.12.2017 |  |
| 6 | Тестирование компилятора | 08.12.2017 |  |
| 7 | Оформление пояснительной записки к курсовому проекту | 08.12.2017 |  |
| 8 | Сдача проекта | 19.12.2017 |  |

**8. Дата выдачи задания 18.09.2017**

Руководитель Наркевич А. С.

(подпись)

Задание принял к исполнению Чаевский А.В.

(дата и подпись студента)

**Содержание**

[**Введение** 7](#_Toc501385915)

[**Глава 1. Спецификация языка программирования** 9](#_Toc501385916)

[**1.1. Характеристика языка программирования** 9](#_Toc501385917)

[**1.2.** **Алфавит языка** 9](#_Toc501385918)

[**1.3. Символы сепараторы** 10](#_Toc501385919)

[**1.4. Применяемые кодировки** 10](#_Toc501385920)

[**1.5. Типы данных** 10](#_Toc501385921)

[**1.6. Преобразование типов данных** 11](#_Toc501385922)

[**1.7. Идентификаторы** 11](#_Toc501385923)

[**1.8. Литералы** 11](#_Toc501385924)

[**1.9. Область видимости идентификаторов** 11](#_Toc501385925)

[**1.10. Инициализация данных** 11](#_Toc501385926)

[**1.11.** **Инструкции языка** 12](#_Toc501385927)

[**1.12. Операции языка** 12](#_Toc501385928)

[**1.13. Выражения и их вычисления** 12](#_Toc501385929)

[**1.14. Программные конструкции языка** 13](#_Toc501385930)

[**1.15.** **Область видимости** 13](#_Toc501385931)

[**1.16. Семантические проверки** 13](#_Toc501385932)

[**1.17.** **Распределение оперативной памяти на этапе выполнения** 13](#_Toc501385933)

[**1.18. Стандартная библиотека и её состав** 13](#_Toc501385934)

[**1.19.** **Ввод и вывод данных** 14](#_Toc501385935)

[**1.20. Точка входа** 14](#_Toc501385936)

[**1.21.** **Препроцессор** 14](#_Toc501385937)

[**1.22. Соглашения о вызовах** 14](#_Toc501385938)

[**1.23.** **Объектный код** 14](#_Toc501385939)

[**1.24. Классификация сообщений транслятора** 14](#_Toc501385940)

[**1.25.** **Контрольный пример** 15](#_Toc501385941)

[**Глава 2. Структура транслятора** 16](#_Toc501385942)

[**2.1 Компоненты транслятора, их назначение и принципы взаимодействия** 16](#_Toc501385943)

[**2.2 Перечень входных параметров транслятора** 17](#_Toc501385944)

[**2.3 Перечень протоколов, формируемых транслятором и их содержимое** 17](#_Toc501385945)

[**Глава 3. Разработка лексического анализатора** 18](#_Toc501385946)

[**3.1 Структура лексического анализатора** 18](#_Toc501385947)

[**3.2 Контроль входных символов** 18](#_Toc501385948)

[**3.3 Удаление избыточных символов** 19](#_Toc501385949)

[**3.4 Перечень ключевых слов, сепараторов, символов операций и соответствующих им лексем, регулярных выражений и конечных автоматов** 19](#_Toc501385950)

[**3.5 Основные структуры данных** 20](#_Toc501385951)

[**3.6 Принцип обработки ошибок** 20](#_Toc501385952)

[**3.7 Структура и перечень сообщений лексического анализатора** 20](#_Toc501385953)

[**3.8 Параметры лексического анализатора и режимы его работы** 21](#_Toc501385954)

[**3.9 Алгоритм лексического анализа** 21](#_Toc501385955)

[**3.10 Контрольный пример** 21](#_Toc501385956)

[**Глава 4. Разработка синтаксического анализатора** 22](#_Toc501385957)

[**4.1 Структура синтаксического анализатора** 22](#_Toc501385958)

[**4.2 Контекстно свободная грамматика, описывающая синтаксис языка** 22](#_Toc501385959)

[**4.3 Построение конечного магазинного автомата** 24](#_Toc501385960)

[**4.4 Основные структуры данных** 25](#_Toc501385961)

[**4.5 Описание алгоритма синтаксического разбора** 25](#_Toc501385962)

[**4.6 Структура и перечень сообщений синтаксического анализатора** 25](#_Toc501385963)

[**4.7 Параметры синтаксического анализатора и режимы его работы** 26](#_Toc501385964)

[**4.8 Принцип обработки ошибок** 26](#_Toc501385965)

[**4.9 Контрольный пример** 26](#_Toc501385966)

[**Глава 5. Разработка семантического анализатора** 27](#_Toc501385967)

[**5.1 Структура семантического анализатора** 27](#_Toc501385968)

[**5.2 Функции семантического анализатора** 27](#_Toc501385969)

[**5.3 Структура и перечень сообщений семантического анализатора** 27](#_Toc501385970)

[**5.4 Принцип обработки ошибок** 27](#_Toc501385971)

[**5.5 Контрольный пример** 27](#_Toc501385972)

[**Глава 6. Преобразование выражений** 27](#_Toc501385973)

[**6.1 Выражения, допускаемые языком** 28](#_Toc501385974)

[**6.2 Польская запись** 28](#_Toc501385975)

[**6.3 Программная реализация обработки выражений** 29](#_Toc501385976)

[**6.4 Контрольный пример** 29](#_Toc501385977)

[**Глава 7. Генерация кода** 30](#_Toc501385978)

[**7.1 Структура генератора кода** 30](#_Toc501385979)

[**7.2 Представление типов данных в оперативной памяти** 30](#_Toc501385980)

[**7.3 Алгоритм работы генератора кода** 31](#_Toc501385981)

[**Глава 8. Тестирование транслятора** 32](#_Toc501385982)

[**8.1 Тестирование фазы проверки на допустимость символов** 32](#_Toc501385983)

[**8.2** **Тестирование лексического анализатора** 32](#_Toc501385984)

[**8.3 Тестирование синтаксического анализатора** 32](#_Toc501385985)

[**8.4 Тестирование семантического анализатора** 32](#_Toc501385986)

[**Приложения** 33](#_Toc501385987)

[**Контрольный пример** 33](#_Toc501385988)

[**Приложение А** 34](#_Toc501385989)

[**Приложение В** 42](#_Toc501385990)

[**Приложение Г** 43](#_Toc501385991)

[**Приложение Е** 49](#_Toc501385992)

[**Литература** 50](#_Toc501385993)

# **Введение**

Целью курсового проекта является разработка компилятора для своего языка программирования: CHAW-2017. Этот язык программирования предназначен для выполнения простейших операций и арифметических действий над числами.

Компилятор – это программа, задачей которого является перевод программы, написанной на одном из языков программирования (в моём случае на CHAW-2017) в программу на язык ассемблера.

Компиляция состоит из двух частей: анализа и синтеза. Анализ – это разбиение исходной программы на составные части и создание ее промежуточного представления. Синтез – конструирование требуемой целевой программы из промежуточного представления. В данном курсовом проекте мой исходный код транслируется на язык ассемблера. Язык ассемблера – тип языка программирования низкого уровня, представляющий собой формат записи машинных команд, удобный для восприятия человеком.

Мой компилятор состоит из следующих составных частей:

– лексический и семантический анализаторы;

– синтаксический анализатор;

– генератор исходного кода на языке ассемблера.

Исходя из цели курсового проекта, были определены следующие задачи:

– разбработка спецификации языка программирования;

– разбратка структуры транслятора;

– разработка лексического и семантического анализаторов;

– разработка синтаксического анализатора;

– преобразование выражений;

– генерация кода на язык ассемблера;

– тестирование транслятора.

Решения каждой из поставленных задач буду приведены в соответствующих главах курсового проекта, а именно:

В первой главе работы определена спецификация языка программирования, т.е. описан синтаксис и семантика языка.

Во второй главе работы представлена структура транслятора, т.е. перечислены компоненты транслятора, их назначение и принципы взаимодействия, перечень входных параметров, перечень протоколов, формируемых транслятором и их содержимое.

В третьей главе работы показана разработка лексического анализатора, порождающего таблицы лексем и идентификаторов.

В четвертой главе работы рассказывается о синтаксическом анализаторе, который выполняет синтаксический разбор текста с распечаткой протокола разбора и дерева разбора на основе таблицы лексем.

В пятой главе описан семантический анализатор, показана его работа (распечатка выданных сообщений в трёх примерах на разных этапах).

В шестой главе решены вопросы преобразования выражений, допускаемых языком и приведена часть протокола для контрольного примера, отображающая результаты преобразования выражений в польский формат.

В седьмой главе представлена генерация кода, где из промежуточного представления порождается код на целевом языке.

В восьмой главе описывается тестирование транслятора.

# **Глава 1. Спецификация языка программирования**

## **Характеристика языка программирования**

Язык CHAW-2017 ­– это процедурный, универсальный, строготипизированный, компилируемый язык. Hе является объектно-ориентированным.

* 1. **Алфавит языка**

Алфавит языка CHAW-2017 основан на кодировке Windows-1251, представленной на рисунке 1.1.



Рисунок 1.1 – Алфавит входных символов

Исходный код CHAW-2017 может содержать символы латинского алфавита, цифры десятичной системы счисления от 0 до 9, русские символы разрешены только в строковых литералах.

## **Символы сепараторы**

Символы, которые являются сепараторами представлены в таблице 1.1.

Таблица 1.1 – Сепараторы

|  |  |
| --- | --- |
| Сепаратор | Назначение |
| ;  :  ‘ ‘ (пробел)  ,  =  +  -  \* | разделение инструкций |
| {  } | программный блок |
| (  ) | параметры/приоритетность операций (в выражениях) |

## **Применяемые кодировки**

Для написания исходного кода на языке программирования CHAW-2017 используется кодировка Windows-1251.

## **Типы данных**

В языке CHAW-2017 есть 2 типа данных: целочисленный и строковый. Описание типов данных, предусмотренных в данным языке представлено в таблице 1.2.

Таблица 1.2 – Типы данных языка CHAW-2017

|  |  |
| --- | --- |
| Тип данных | Описание типа данных |
| dig | Фундаментальный тип данных. Предусмотрен для объявления целочисленных положительных данных (4 байта).  Автоматически инициализируется нулевым значением.  Возможные операции:  *арифметические*  + – бинарный, суммирование;  - – бинарный, вычитание;  \* – бинарный, умножение;  = – присваивание значения; |
| chr | Фундаментальный тип данных. Предусмотрен для объявления строк. (1 символ – 1 байт). Автоматическая инициализация строкой нулевой длины. Максимальное количество символов в строке – 255. |

## **Преобразование типов данных**

В языке программирования CHAW-2017 преобразование типов данных не поддерживается, т.е. язык является строготипизированным.

## **Идентификаторы**

В имени идентификатора допускаются символы латинского алфавита нижнего регистра. Максимальная длина имени - 10 символов.

## **Литералы**

В языке существует 2 типа литералов: целого и символьного типов. Краткое описание литералов представлено в таблице 1.3.

Таблица 1.3 – Описание литералов

|  |  |
| --- | --- |
| Тип литерала | Описание |
| Литералы целого типа | Целочисленные неотрицательные литералы, инициализируются 0. Литералы только rvalue. |
| Строковые литералы | Символы, заключённые в “” (двойные кавычки), инициализируются пустой строкой, строковые переменные. Только rvalue. |

## **Область видимости идентификаторов**

Область видимости «сверху вниз» (по принципу С++). В языке CHAW-2017 требуется обязательное объявление переменной перед её использованием. Все переменные должны находиться внутри программного блока языка. Имеется возможность объявления одинаковых переменных в разных блоках. Каждая переменная получает префикс – название функции, в которой она объявлена.

## **Инициализация данных**

Таблица 1.4 – Способы инициализации переменных

|  |  |
| --- | --- |
| Вид инициализации | Примечание |
| var <тип данных> <идентификатор>; | Автоматическая инициализация: переменные типа dig инициализируются нулём, переменные типа chr – пустой строкой. |
| <идентификатор> = <значение>; | Присваивание переменной значения. |

## **Инструкции языка**

Все возможные инструкции языка программирования CHAW-2017 представлены в общем виде в таблице 1.5.

Таблица 1.5 – Инструкции языка программирования CHAW-2017

|  |  |
| --- | --- |
| Инструкция | Запись на языке CHAW-2017 |
| Объявление переменной | var <идентификатор> : <тип данных>; |
| Присваивание | <идентификатор> = <значение>/<идентификатор>; |
| Объявление внешней функции | func <идентификатор> (<идентификатор>:<тип данных>, …) : <тип данных> {…} |
| Блок инструкций | body  {  …  } |
| Возврат из подпрограммы | ret <идентификатор> / <литерал>; |
| Вывод данных | out (<идентификатор> / <литерал>); |

## **Операции языка**

Язык программирования CHAW-2017 может выполнять арифметические операции, представленные в таблице 1.6.

Таблица 1.6 – Приоритетности операций языка программирования CHAW‑2017

|  |  |
| --- | --- |
| Операция | Приоритетность операции |
| (  ) | 0 или 4 |
| , | 1 |
| \* | 2 |
| +  - | 3 |

Максимальным значением приоритетности является “0”, минимальным “4” соответственно.

Также языком поддерживается операция сравнения операндов. Данная операция не имеет приоритета.

## **Выражения и их вычисления**

Круглые скобки в выражении используются для изменения приоритета операций. Также не допускается запись двух подряд идущих арифметических операций. Выражение может содержать вызов функции.

## **Программные конструкции языка**

Ключевые программные конструкции языка программирования CHAW-2017 представлены в таблице 1.7.

Таблица 1.7 – Программные конструкции языка CHAW-2017

|  |  |
| --- | --- |
| Главная функция (точка входа в приложение) | body  {  } |
| Функция | func <идентификатор> (<идентификатор> : <тип>, …) : <тип>  {  …  ret <выражение>;  } |

## **Область видимости**

В языке CHAW-2017 переменные обязаны находится внутри программного блока функций (по принципу С++). Объявление глобальных переменных не предусмотрено. Объявление пользовательских областей видимости не предусмотрено.

## **Семантические проверки**

Таблица с перечнем семантических проверок, предусмотренных языком, приведена в таблице 1.8.

Таблица 1.8 – Семантические проверки

|  |  |
| --- | --- |
| Номер | Правило |
| 1 | Идентификаторы функций не должны повторяться |
| 2 | Операнды в операторах ветвления и выхода из функции должны быть целочисленного типа |
| 3 | Тип данных передаваемых значений в функцию должен совпадать с типом параметров при её объявлении |
| 4 | Тип данных передаваемых значений в функцию стандартной библиотеки должен соответствовать заявленному. |
| 5 | Идентификатор должен быть объявлен до его использования. |
| 6 | Операнды в арифметическом выражении не могут быть разных типов |

## **Распределение оперативной памяти на этапе выполнения**

Все переменные размещаются в куче.

## **Стандартная библиотека и её состав**

Функции стандартной библиотеки с описанием представлены в таблице 1.9. Стандартная библиотека написана на языке программирования C++.

Таблица 1.9 – Состав стандартной библиотеки

|  |  |  |  |
| --- | --- | --- | --- |
| Имя функции | Возвращаемое значение | Принимаемые параметры | Описание |
| pow | dig | dig x – число  dig n – степень | Функция возводит число x в степень n |
| strl | dig | chr x - строка | Функция вычисляет длину строки x |
| outd | 0 | dig x - число | Функция выводит на консоль число x |
| outc | 0 | chr x - строка | Функция выводит на консоль строку x |

## **Ввод и вывод данных**

Ввод данных не поддерживается языком программирования CHAW-2017.

out (<идентификатор или литерал>); – вывод в стандартный поток вывода.

В зависимости от типа параметра определяется функция: outd или outc, которые входят в состав стандартной библиотеки и описаны в таблице 1.9.

## **Точка входа**

Точкой входа является функция body.

## **Препроцессор**

Препроцессор в языке программирования CHAW-2017 не предусмотрен.

## **Соглашения о вызовах**

В языке вызов функций происходит по соглашению о вызовах stdcall. Особенности stdcall:

– все параметры функции передаются через стек;

– память высвобождает вызываемый код;

– занесение в стек параметров идёт справа налево.

## **Объектный код**

CHAW-2017 транслируется в язык ассемблера.

## **Классификация сообщений транслятора**

В случае возникновения ошибки в коде программы на языке CHAW-2017 и выявления её транслятором в текущий файл протокола выводится сообщение. Их классификация сообщений приведена в таблице 1.10.

Таблица 1.10. Классификация сообщений транслятора

|  |  |
| --- | --- |
| Интервал | Описание ошибок |
| 0-9 | Системные ошибки |
| 10-19 | Ошибки параметров |
| 20-29 | Ошибки открытия и чтения файлов |
| 30-49 | Ошибки лексического анализа |
| 50-79 | Ошибки синтаксического анализа |
| 80-99 | Ошибки семантического анализа |

## **Контрольный пример**

Контрольный пример представлен в главе Приложения.

# **Глава 2. Структура транслятора**

## **2.1 Компоненты транслятора, их назначение и принципы взаимодействия**

Транслятор преобразует программу, написанную на языке CHAW-2017 в программу на языке ассемблера. Компонентами транслятора являются лексический, синтаксический и семантический анализаторы, а также генератор кода на язык ассемблера. Принцип их взаимодействия представлен на рисунке 2.1.



Рисунок 2.1 – Структура транслятора

Лексический анализ – первая фаза трансляции. Назначением лексического анализатора является нахождение ошибок лексики языка и формирование таблицы лексем и таблицы идентификаторов. Подробнее описан в 3 главе.

Семантический анализ в свою очередь является проверкой исходной программы на семантическую согласованность с определением языка, т.е. проверяет правильность текста исходной программы с точки зрения семантики. Подробное описание представлено в 5 главе.

В моём трансляторе этапы лексического и семантического анализа являются единым целым и выполняются одновременно.

Синтаксический анализ – это основная часть транслятора, предназначенная для распознавания синтаксических конструкций и формирования промежуточного кода. Входным параметром для синтаксического анализа является таблица лексем. Синтаксический анализатор распознаёт синтаксические конструкции, выявляет синтаксические ошибки при их наличии и формирует дерево разбора. Подробнее рассмотрен в главе 4.

Генератор кода – этап транслятора, выполняющий генерацию ассемблерного кода на основе полученных данных на предыдущих этапах трансляции. Генератор кода принимает на вход таблицы идентификаторов и лексем и транслирует код на языке CHAW-2017, прошедший все предыдущие этапы, в код на языке Ассемблера. Более полно описан в главе 7.

## **2.2 Перечень входных параметров транслятора**

Входные параметры представлены в таблице 2.1.

Таблица 2.1 Входные параметры транслятора языка CHAW-2017

|  |  |  |
| --- | --- | --- |
| Входной параметр | Описание | Значение по умолчанию |
| -in:<имя\_файла> | Входной файл с расширением .txt, в котором содержится исходный код на CHAW-2017 | Не предусмотрено |
| -log:<имя\_файла> | Файл для записи результата проверки входного файла на допустимость символов | <имя\_файла>.log |
| -out:<имя\_файла> | Файл для записи результата работы транслятора | <имя\_файла>.asm |
| -lex:<имя\_файла> | Файл для записи результата работы лексического и семантического анализа. | <имя\_файла>.lex |
| -sin:<имя\_файла> | Файл для записи результата работы синтаксического анализа. | <имя\_файла>.sin |
| -ntr | Ключ для вывода трассировки синтаксического анализа в файл | Ключа нет, т.е. трассировка не записывается |

## **2.3 Перечень протоколов, формируемых транслятором и их содержимое**

Таблица с перечнем протоколов, формируемых транслятором языка

CHAW‑2017 и их назначением представлена в таблице 2.2

Таблица 2.2 – Протоколы, формируемые транслятором языка CHAW-2017

|  |  |
| --- | --- |
| Формируемый протокол | Описание протокола |
| Файл журнала с параметром <log> | Содержит информацию о входных параметрах в приложение и о этапе проверки символов на допустимость. |
| Выходной файл c параметром <out> | Содержит сгенерированный код на языке Ассемблера. |
| lex:<имя\_файла> | Результат работы лексического и семантического анализа. Содержит таблицы лексем и идентификаторов. |
| sin:<имя\_файла> | Результат работы синтаксического анализа. Содержит правила разбора, трассировку, а также преобразованные после польской записи таблицы лексем и идентификаторов. |

# **Глава 3. Разработка лексического анализатора**

## **3.1 Структура лексического анализатора**

Лексический анализатор – часть транслятора, выполняющая лексический анализ. Лексический анализатор принимает обработанный и разбитый на отдельные компоненты исходный код на языке CHAW-2017. На выходе формируется таблица лексем и таблица идентификаторов. Структура лексического анализатора представлена на рисунке 3.1



Рисунок 3.1 Структура лексического анализатора CHAW-2017

## **3.2 Контроль входных символов**

Таблица для контроля входных символов представлена на рисунке 3.2

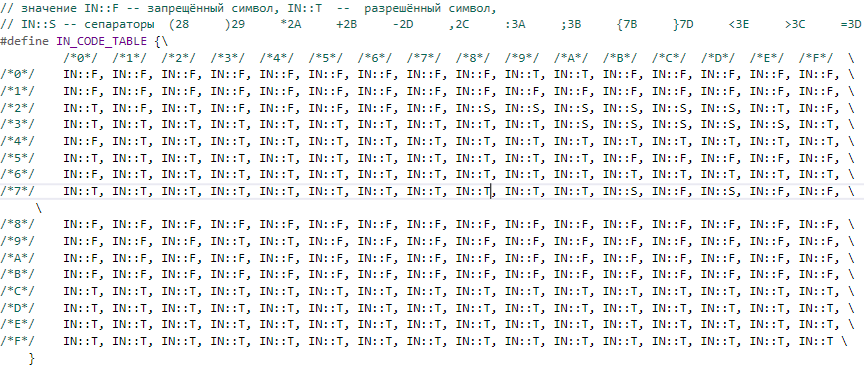


Рисунок 3.2. Таблица контроля входных символов

Принцип работы таблицы заключается в соответствии значения каждому элементу в шестнадцатеричной системе счисления значению в таблице ASCII.

Описание значения символов: T – разрешённый символ, F – запрещённый символ, S – сепаратор.

## **3.3 Удаление избыточных символов**

Избыточными символами являются символы табуляции и пробелы.

Избыточные символы удаляются на этапе разбиения исходного кода на токены.

Описание алгоритма удаления избыточных символов:

1. Посимвольно считываем файл с исходным кодом программы.

2. Встреча пробела или знака табуляции является своего рода встречей символа-сепаратора.

3. В отличие от других символов-сепараторов не записываем в очередь лексем эти символы, т.е. игнорируем.

## **3.4 Перечень ключевых слов, сепараторов, символов операций и соответствующих им лексем, регулярных выражений и конечных автоматов**

Лексемы – это символы, соответствующие ключевым словам, символам операций и сепараторам, необходимые для упрощения дальнейшей обработки исходного кода программы. Данное соответствие описано в таблице 3.1.

Таблица 3.1 Соответствие ключевых слов, символов операций и сепараторов с лексемами

|  |  |  |
| --- | --- | --- |
| Тип цепочки | Цепочка | Лексема |
| Ключевые слова | var | v |
| dig | t |
| chr | t |
| func | f |
| out | o |
| power | p |
| strl | s |
| ret | r |
| body | b |
| if | q |
| then | 1 |
| else | 2 |
| Иное | Идентификатор | i |
| Целочисленный литерал | d |
| Строковый литерал | c |

Продолжение таблицы 3.1

|  |  |  |
| --- | --- | --- |
| Сепараторы | ; | ; |
| , | , |
| : | : |
| { | { |
| } | } |
| ( | ( |
| ) | ) |
| Операторы | + | a |
| - | a |
| \* | a |
| > | > |
| < | < |
| = | = |

Пример реализации таблицы лексем представлен в приложении А.

Также в приложении А находятся конечные автоматы, соответствующие лексемам языка CHAW-2017.

## **3.5 Основные структуры данных**

Основные структуры таблиц лексем и идентификаторов данных языка CHAW-2017, используемых для хранения, представлены в приложении А. В таблице лексем содержится лексема, её номер, полученный при разборе, номер строки в исходном коде и приоритет. В таблице идентификаторов содержится имя идентификатора, номер в таблице лексем, тип данных, смысловой тип идентификатора и его значение.

## **3.6 Принцип обработки ошибок**

При возникновении ошибки типа предупреждение транслятор продолжает свою работу, а предупреждения записываются в специальную структуру с номером ошибки и диагностическим сообщением.

Когда возникает критическая ошибка – работа транслятора прекращается.

В случаях, если на этапе было найдено менее 5 предупреждений, они будут выведены в файл после окончания этапа трансляции.

В случае, если была найдена критическая ошибка или в структуре находятся 5 предупреждений – работа транслятора прекращается и диагностическое сообщение будет выведено в файл.

## **3.7 Структура и перечень сообщений лексического анализатора**

Перечень сообщений лексического анализатора представлен на рисунке 3.3.

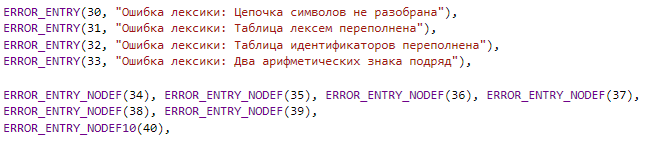


Рисунок 3.3 – Перечень ошибок лексического анализатора

## **3.8 Параметры лексического анализатора и режимы его работы**

Входным параметром лексического анализа является очередь, состоящая из структур, полями которых являются лексема и номер её строки в исходном файле, полученные на этапе проверки исходного кода на допустимость символов.

## **3.9 Алгоритм лексического анализа**

Лексический анализ выполняется программой (входящей в состав транслятора), называемой лексическим анализатором. Цель лексического анализа — выделение и классификация лексем в тексте исходной программы. Лексический анализатор производит распознаёт и разбирает цепочки исходного текста программы. Это основывается на работе конечных автоматов, которую можно представить в виде графов. Регулярные выражения — аналитический или формульный способ задания регулярных языков. Они состоят из констант и операторов, которые определяют множества строк и множество операций над ними. Любое регулярное выражение можно представить в виде графа.

Пример. Регулярное выражение для ключевого слова body: ‘body’.

Граф конечного автомата для этой лексемы представлен на рисунке 3.4. S0 – начальное состояние, S4 – конечное состояние автомата.

b

d

y

o

Рисунок 3.4 – Граф переходов для цепочки ‘body’

## **3.10 Контрольный пример**

Результат работы лексического анализатора – таблицы лексем и идентификаторов – представлен в приложении А.

# **Глава 4. Разработка синтаксического анализатора**

## **4.1 Структура синтаксического анализатора**

Синтаксический анализ – это фаза трансляции, выполняемая после лексического анализа и предназначенная для распознавания синтаксических конструкций. Входом для синтаксического анализа является таблица лексем и таблица идентификаторов, полученные после фазы лексического анализа. Выходом – дерево разбора. Структура синтаксического анализатора представлена на рисунке 4.1.



Рисунок 4.1 – Структура синтаксического анализатора

## **4.2 Контекстно свободная грамматика, описывающая синтаксис языка**

В синтаксическом анализаторе транслятора языка CHAW-2017 используется контекстно-свободная грамматика , где

T – множество терминальных символов (было описано в разделе 1.2 данной пояснительной записки),

N – множество нетерминальных символов (первый столбец таблицы 4.1),

P – множество правил языка (второй столбец таблицы 4.1),

S – начальный символ грамматики, являющийся нетерминалом.

Эта грамматика имеет нормальную форму Грейбах, т.к. она не леворекурсивная (не содержит леворекурсивных правил) и правила  имеют вид:

1. , где ; (или , или )
2. , где — начальный символ, при этом если такое правило существует, то нетерминал  не встречается в правой части правил.

Грамматика языка CHAW-2017 представлена в приложении Б.

TS – терминальные символы, которыми являются сепараторы, знаки арифметических операций и некоторые строчные буквы.

NS – нетерминальные символы, представленные несколькими заглавными буквами латинского алфавита.

Таблица 4.1 – Перечень правил, составляющих грамматику языка и описание нетерминальных символов CHAW-2017

|  |  |  |
| --- | --- | --- |
| Нетерминал | Цепочки правил | Описание |
| A | fi(F):t{N}A  b{N} | Порождает правила, описывающее общую структуру программы |
| N | vF;N  vF;  rR;  i=E;N  i=E;  o(O);  o(O);N  qQ1{N}2{N}N  qQ1{N}N | Порождает правила, описывающие инструкции языка |
| E | i  c  d  iM  dM  p(P)  p(P)M  s(S)  s(S)M  i(I)  i(I)M  (E)  (E)M | Порождает правила, описывающие выражения |
| F | i,F  i:t  i:t,F | Порождает правила, описывающие параметры локальной функции при её объявлении |
| O | i  c  d | Порождает правила, описывающие параметры стандартной функции out |
| P | i,d  d,i | Порождает правила, описывающие параметры стандартной функции power |
| S | i  c | Порождает правила, описывающие параметры стандартной функции strl |

Продолжение таблицы 4.1

|  |  |  |
| --- | --- | --- |
| I | i  i,I  d  d,I  c  c,I | Порождает правила, описывающие параметры локальной функции при её вызове |
| M | aE  aEM | Порождает правила, описывающие арифметические действия |
| Q | (R<R)  (R>R) | Порождает правила, описывающие операнды оператора ветвления |
| R | i  d | Порождает правила, описывающие параметр оператора выхода из функции |

## **4.3 Построение конечного магазинного автомата**

Конечный автомат с магазинной памятью представляет собой семерку, описание которой представлено в таблице 4.2. Структура данного автомата показана в приложении В.

Таблица 4.2 – Описание компонентов магазинного автомата

|  |  |  |
| --- | --- | --- |
| Компонента | Определение | Описание |
|  | Множество состояний автомата | Состояние автомата представляет из себя структуру, содержащую позицию на входной ленте, номера текущего правила и цепочки и стек автомата |
|  | Алфавит входных символов | Алфавит является множеством терминальных и нетерминальных символов, описание которых содержится в разделе 1.2 и в таблице 4.1. |
|  | Алфавит специальных магазинных символов | Алфавит магазинных символов содержит стартовый символ и маркер дна стека |
|  | Функция переходов автомата | Функция представляет из себя множество правил грамматики, описанных в таблице 4.1. |
|  | Начальное состояние автомата | Состояние, которое приобретает автомат в начале своей работы. Представляется в виде стартового правила грамматики (нетерминальный символ А) |
|  | Начальное состояние магазина автомата | Символ маркера дна стека ($) |

Продолжение таблицы 4.2

|  |  |  |
| --- | --- | --- |
|  | Множество конечных состояний | Конечные состояние заставляют автомат прекратить свою работу. Конечным состоянием является пустой магазин автомата и совпадение позиции на входной ленте автомата с размером ленты |

## **4.4 Основные структуры данных**

Основные структуры данных синтаксического анализатора включают в себя структуру магазинного автомата и структуру грамматики Грейбах, описывающей правила языка CHAW-2017. Данные структуры представлены в приложении В.

## **4.5 Описание алгоритма синтаксического разбора**

Принцип работы автомата следующий:

1. В магазин записывается стартовый символ;
2. На основе полученных ранее таблиц формируется входная лента;
3. Запускается автомат;
4. Выбирается цепочка, соответствующая нетерминальному символу, записывается в магазин в обратном порядке;
5. Если терминалы в стеке и в ленте совпадают, то данный терминал удаляется из ленты и стека. Иначе возвращаемся в предыдущее сохраненное состояние и выбираем другую цепочку нетерминала;
6. Если в магазине встретился нетерминал, переходим к пункту 4;
7. Если наш символ достиг дна стека, и лента в этот момент пуста, то синтаксический анализ выполнен успешно. Иначе генерируется исключение.

## **4.6 Структура и перечень сообщений синтаксического анализатора**

Перечень сообщений синтаксического анализатора представлен на рисунке 4.1.

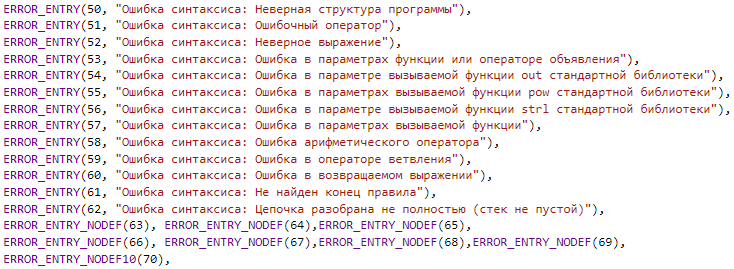


Рисунок 4.1 – Перечень сообщений синтаксического анализатора

## **4.7 Параметры синтаксического анализатора и режимы его работы**

Входным параметром синтаксического анализатора является таблица лексем, полученная на этапе лексического анализа, а также правила контекстно-свободной грамматики в форме Грейбах.

Выходными параметрами являются трассировка прохода таблицы лексем (при наличии разрешающего ключа) и правила разбора, которые записываются в файл протокола данного этапа обработки.

## **4.8 Принцип обработки ошибок**

Обработка ошибок происходит следующим образом:

1. Синтаксический анализатор перебирает все правила и цепочки правила грамматики для нахождения подходящего соответствия с конструкцией, представленной в таблице лексем.
2. Если невозможно подобрать подходящую цепочку, то генерируется соответствующая ошибка.
3. Все ошибки записываются в общую структуру ошибок.
4. В случае нахождения ошибки, после всей процедуры трассировки в протокол будет выведено диагностическое сообщение.

## **4.9 Контрольный пример**

Пример разбора синтаксическим анализатором исходного кода на языке CHAW-2017 представлен в приложении Г. Дерево разбора исходного кода также представлено в приложении Г.

# **Глава 5. Разработка семантического анализатора**

## **5.1 Структура семантического анализатора**

Семантический анализ происходит при выполнении фазы лексического анализа и реализуется в виде отдельных проверок текущих ситуаций в конкретных случаях: установки флага или нахождении в особом месте программы (оператор выхода из функции, оператор ветвления, вызов функции стандартной библиотеки).

## **5.2 Функции семантического анализатора**

Семантический анализатор выполняет проверку на основные правила языка (семантики языка), которые описаны в разделе 1.16.

## **5.3 Структура и перечень сообщений семантического анализатора**

Сообщения, формируемые семантическим анализатором, представлены на рисунке 5.1.

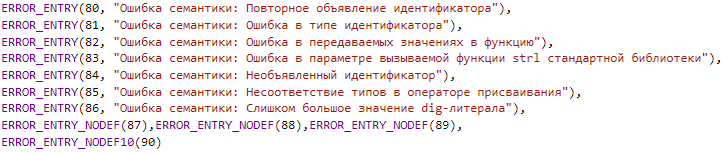


Рисунок 5.1 – Перечень сообщений семантического анализатора

## **5.4 Принцип обработки ошибок**

Принцип обработки ошибок идентичен принципу обработки ошибок на этапе лексического анализа (раздел 3.6).

## **5.5 Контрольный пример**

Результат работы контрольного примера расположен в приложении А, где показан результат лексического анализатора, т.к. представленные таблицы лексем и идентификаторов проходят лексическую и семантическую проверки одновременно.

**Глава 6. Преобразование выражений**

## **6.1 Выражения, допускаемые языком**

В языке CHAW-2017 допускаются выражения, применимые к целочисленным типам данных. В выражениях поддерживаются арифметические операции, такие как +, -, \* и (), и вызовы функций как операнды арифметических выражений.

Приоритет операций представлен в таблице 6.1.

Таблица 6.1 – Приоритет операций в языке CHAW-2017

|  |  |
| --- | --- |
| Приоритет | Операция |
| 0 | ( |
| 0 | ) |
| 1 | , |
| 2 | + |
| 2 | - |
| 3 | \* |
| 3 | / |
| 4 | ( – скобка параметров функции |
| 4 | ) – скобка параметров функции |

## **6.2 Польская запись**

Выражения в языке CHAW-2017 преобразовываются к обратной польской записи.

Польская запись – это альтернативный способ записи арифметических выражений, преимущество которого состоит в отсутствии скобок.

Обратная польская запись – это форма записи математических и логических выражений, в которой операнды расположены перед знаками операций. 

Алгоритм построения:

– исходная строка: выражение;

– результирующая строка: польская запись;

– стек: пустой;

– результирующая строка: польская запись;

– исходная строка просматривается слева направо;

– операнды переносятся в результирующую строку в порядке их следования;

– операция записывается в стек, если стек пуст или в вершине стека лежит отрывающая скобка;

– операция выталкивает все операции с большим или равным приоритетом в результирующую строку;

– запятая не помещается в стек, если в стеке операции, то все выбираются в строку;

– отрывающая скобка помещается в стек;

– закрывающая скобка выталкивает все операции до открывающей скобки, после чего обе скобки уничтожаются;

– закрывающая скобка с приоритетом, равным 4, выталкивает все до открывающей с таким же приоритетом и генерирует @ – специальный символ, в которого записывается информация о вызываемой функции, а в поле приоритета для данной лексемы записывается число параметров вызываемой функции;

– по концу разбора исходной строки все операции, оставшиеся в стеке, выталкиваются в результирующую строку.

Таблица 6.2 – Пример преобразования выражения в обратную польскую запись

|  |  |  |
| --- | --- | --- |
| Исходная строка | Результирующая строка | Стек |
| q\*2 - s(i) |  |  |
| \*2 - s(i) | q |  |
| 2 - s(i) | q | \* |
| - s(i) | q2 | \* |
| s(i) | q2\* | - |
| (i) | q2\* | - |
| i) | q2\* | - |
| ) | q2\*i | - |
|  | q2\*i@1- |  |

## **6.3 Программная реализация обработки выражений**

Программная реализация алгоритма преобразования выражений к польской записи представлена в приложении Д.

## **6.4 Контрольный пример**

Пример преобразования выражения к польской записи представлен в таблице 6.4. Преобразование выражений в формат польской записи необходимо для построения более простых алгоритмов их вычисления.

В приложении Д приведены изменённые таблицы лексем и идентификаторов, отображающие результаты преобразования выражений в польский формат.

# **Глава 7. Генерация кода**

## **7.1 Структура генератора кода**

Генерация объектного кода — это перевод компилятором внутреннего представления исходной программы в цепочку символов выходного языка. На вход генератора подаются таблицы лексем и идентификаторов, на основе которых генерируется файл с ассемблерным кодом.



Рисунок 7.1 Структура генератора кода

## **7.2 Представление типов данных в оперативной памяти**

Элементы таблицы идентификаторов расположены в разных сегментах языка ассемблера – .data и .const. Идентификаторы языка CHAW-2017 размещены в сегменте данных(.data). Литералы – в сегменте констант (.const). Соответствия между типами данных идентификаторов на языке CHAW-2017 и на языке ассемблера приведены в таблице 7.1.

Таблица 7.1 – Соответствия типов идентификаторов языка CHAW-2017 и языка Ассемблера

|  |  |  |
| --- | --- | --- |
| Тип идентификатора на языке CHAW-2017 | Тип идентификатора на языке ассемблера | Пояснение |
| dig | SDWORD | Хранит целочисленный тип данных со знаком. |
| chr | DWORD | Хранит указатель на начало строки. |
| L(0-9) | BYTE  DWORD | Литералы: символьные,  целочисленные |

## **7.3 Алгоритм работы генератора кода**

Преобразования происходят по принципу «встретил – описал, что встретил», т.е. встретив определённую лексему и зная, в каком месте программы находится сейчас лексема, программа генерирует код на языке Ассемблера.

На рисунке 7.2 представлен пример описания лексемы на языке Ассемблера.

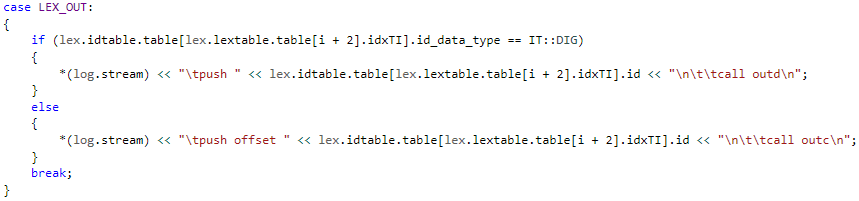


Рисунок 7.2 – Алгоритм для лексемы ‘o’

Генерируемый код записывается в файл, который был параметром ‑out. Сгенерированный код можно посмотреть в приложении Е.

# **Глава 8. Тестирование транслятора**

## **8.1 Тестирование фазы проверки на допустимость символов**

В языке CHAW-2017 не разрешается использовать запрещённые входным алфавитом символы. Результат использования запрещённого символа показан в таблице 8.1.

Таблица 8.1 – Тестирование фазы проверки на допустимость символов

|  |  |
| --- | --- |
| Исходный код | Диагностическое сообщение |
| func funa(], b : dig) : dig | Ошибка 23: Ошибка проверки входного файла: Недопустимый символ в исходном файле, строка 1, позиция 8 |

## **Тестирование лексического анализатора**

На этапе лексического анализа могут возникнуть ошибки, описанные в пункте 3.7. Результаты тестирования лексического анализатора показаны в таблице 8.2.

Таблица 8.2 – Тестирование лексического анализатора

|  |  |
| --- | --- |
| Исходный код | Диагностическое сообщение |
| Func funa(a, b : dig) : dig | Ошибка 30: Ошибка лексики: Цепочка символов не разобрана, строка 1, позиция -1 |

## **8.3 Тестирование синтаксического анализатора**

На этапе синтаксического анализа могут возникнуть ошибки, описанные в пункте 4.6. Результаты тестирования синтаксического анализатора показаны в таблице 8.3.

Таблица 8.3 – Тестирование синтаксического анализатора

|  |  |
| --- | --- |
| Исходный код | Диагностическое сообщение |
| var c, d : dig | Ошибка 53: Ошибка синтаксиса: Ошибка в параметрах функции или операторе объявления, строка 3, позиция -1 |

## **8.4 Тестирование семантического анализатора**

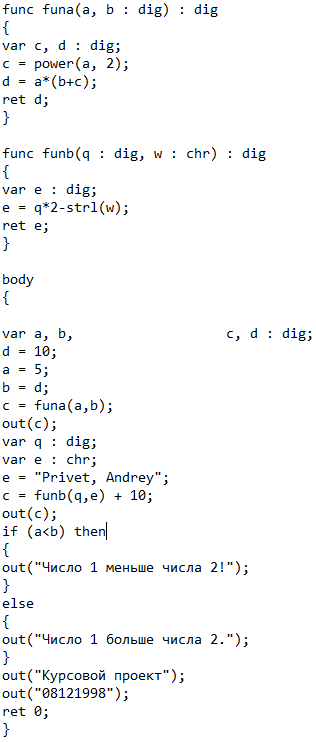
Итоги тестирования семантического анализатора приведены в таблице 8.4.

Таблица 8.4 – Тестирование семантического анализатора

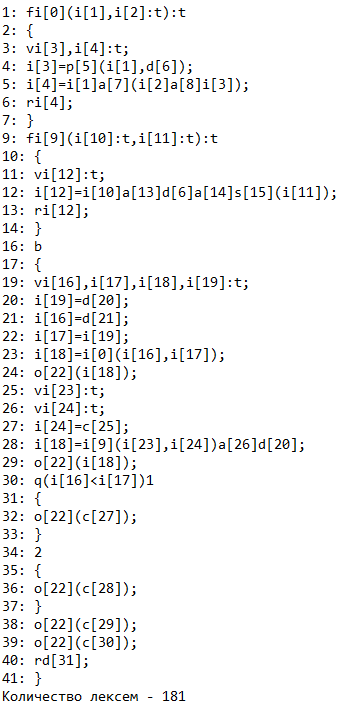
|  |  |
| --- | --- |
| Исходный код | Диагностическое сообщение |
| func funa(a, b : dig) : dig  {  var c, d : dig;  q = power(a, 2); | Ошибка 84: Ошибка семантики: Необъявленный идентификатор, строка 4, позиция -1 |

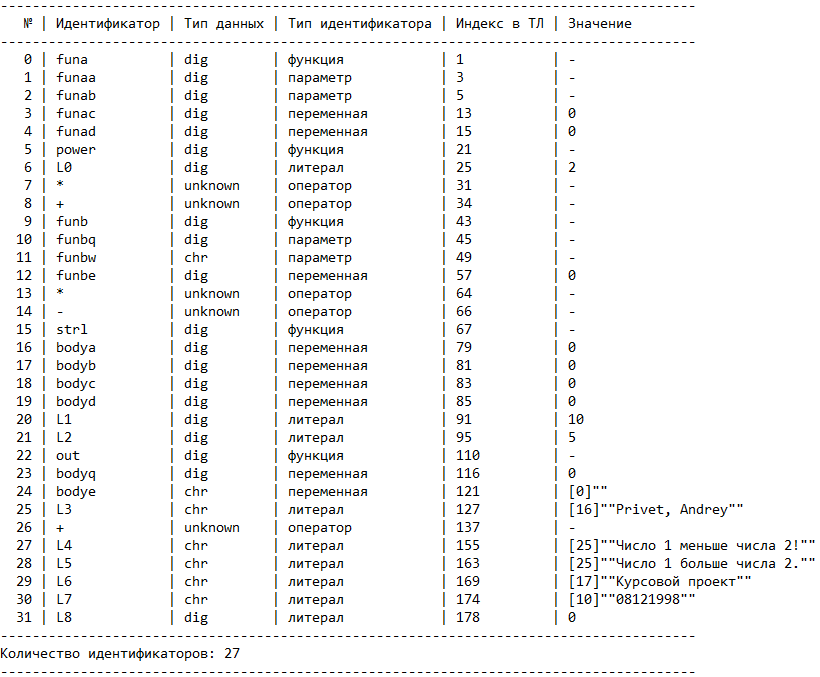
# **Приложения**

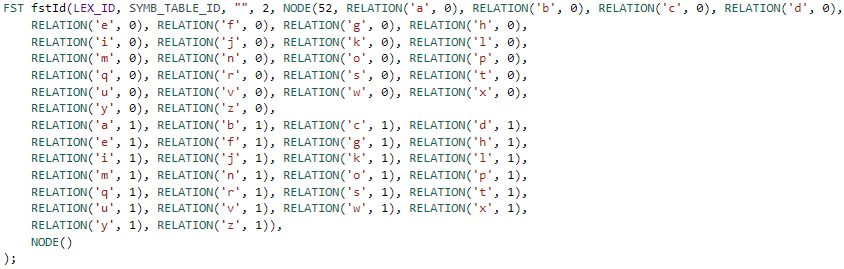
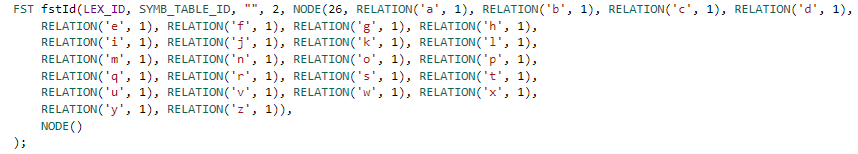
## **Контрольный пример**

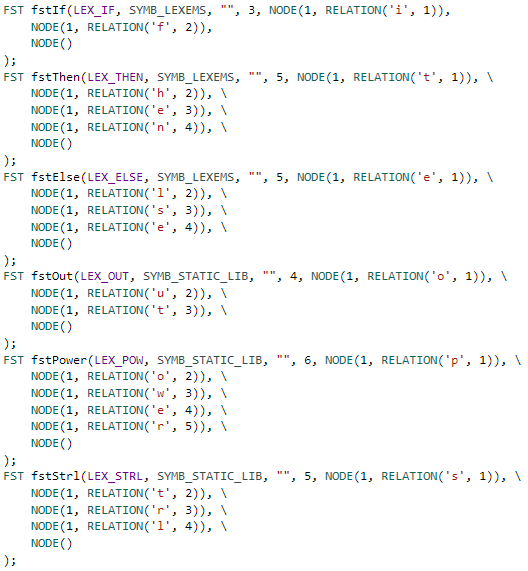
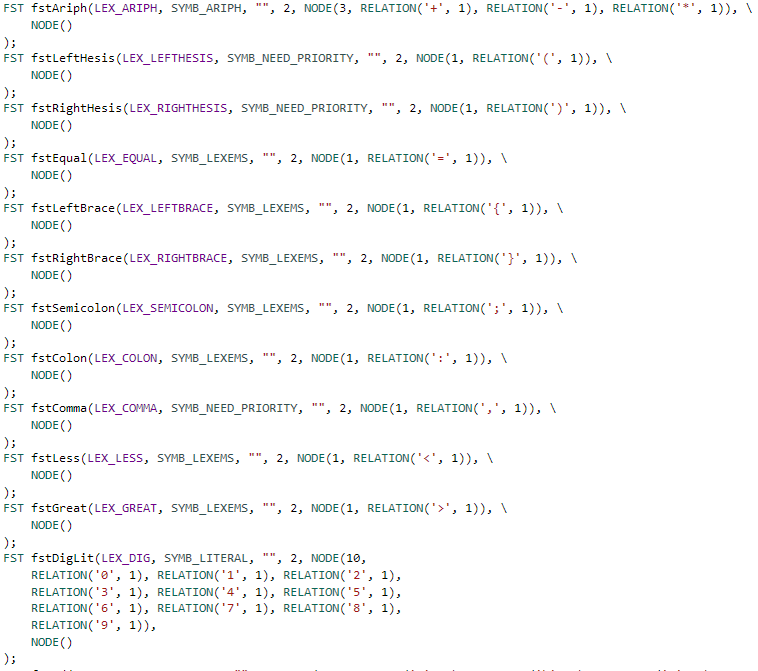


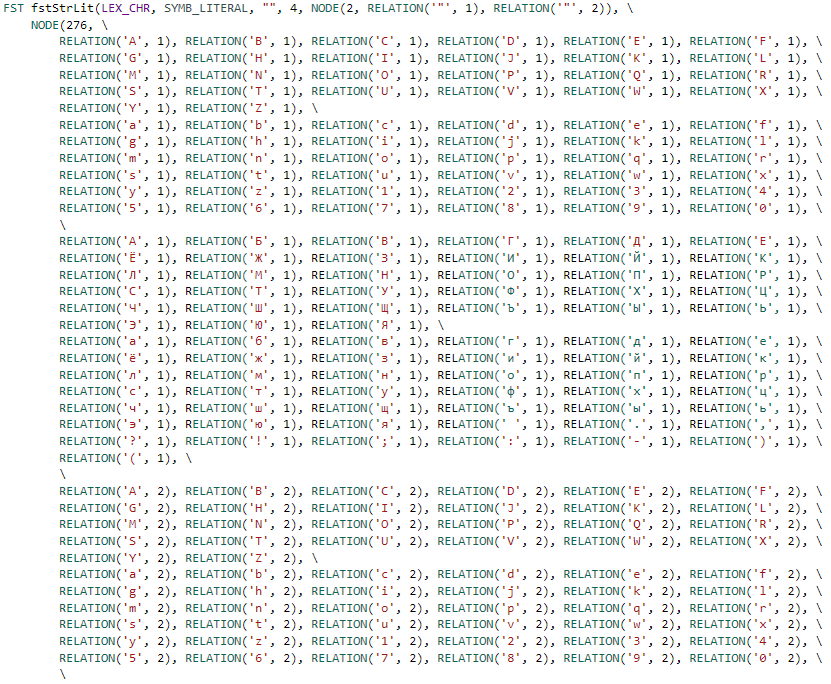
## **Приложение А**

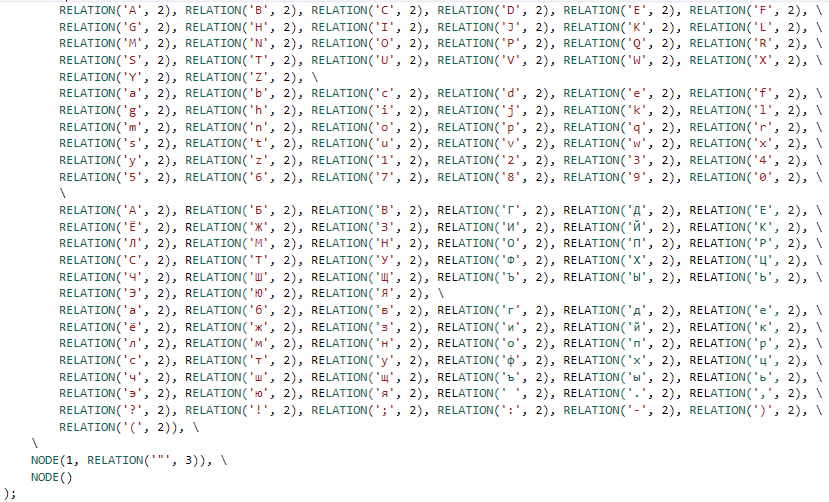


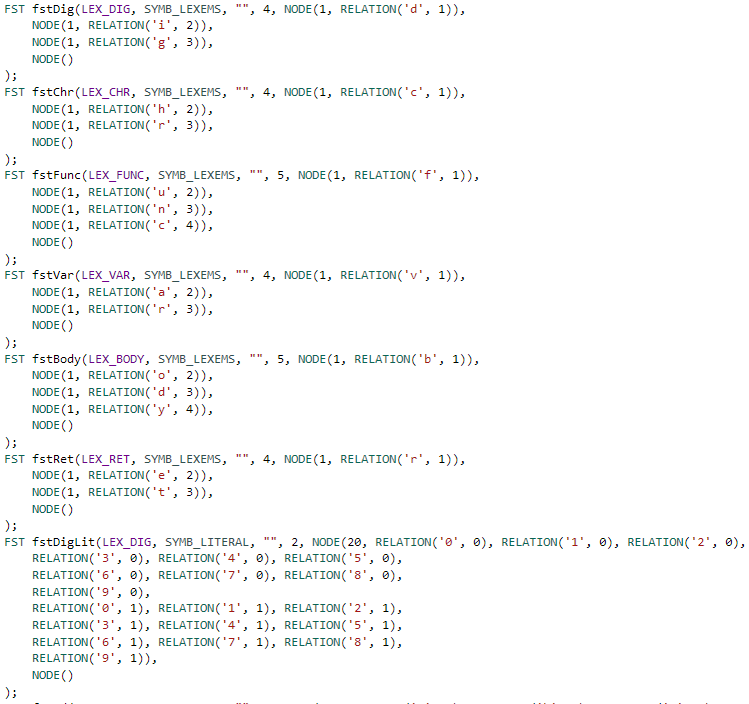


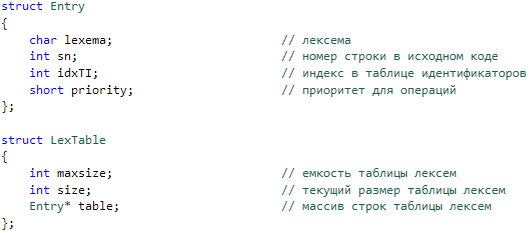


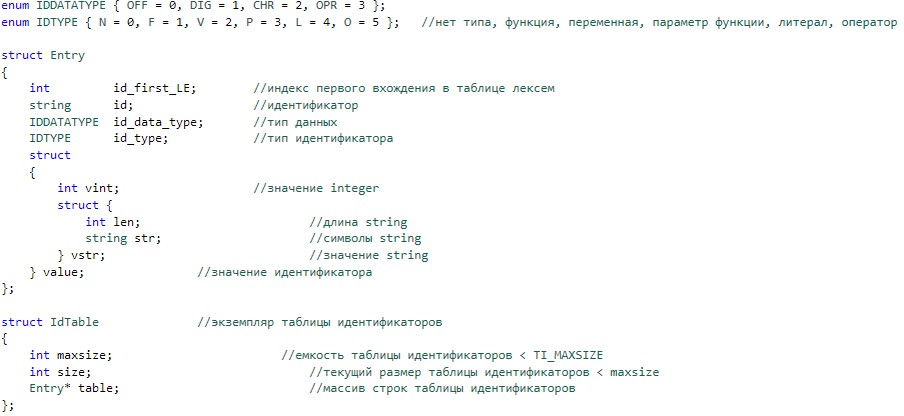




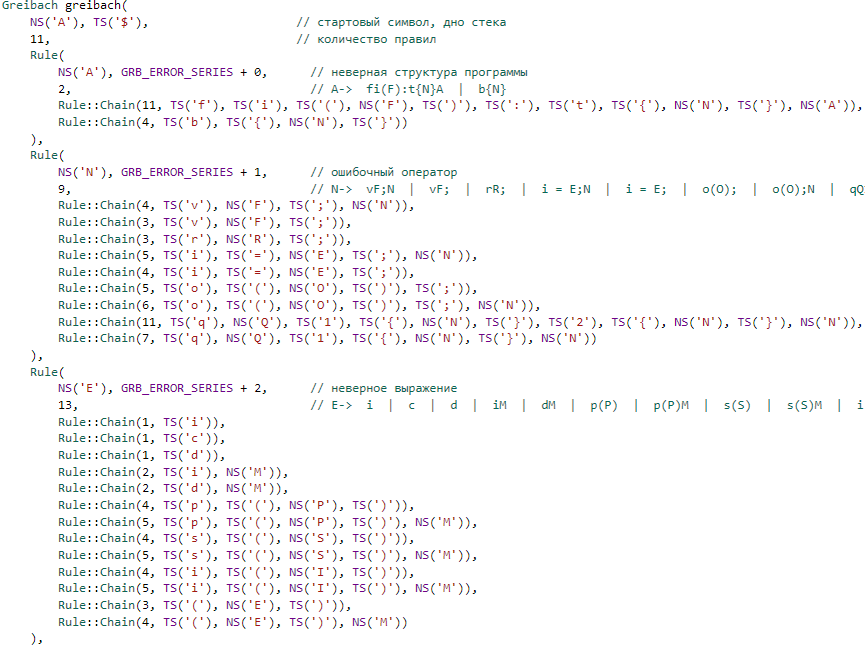


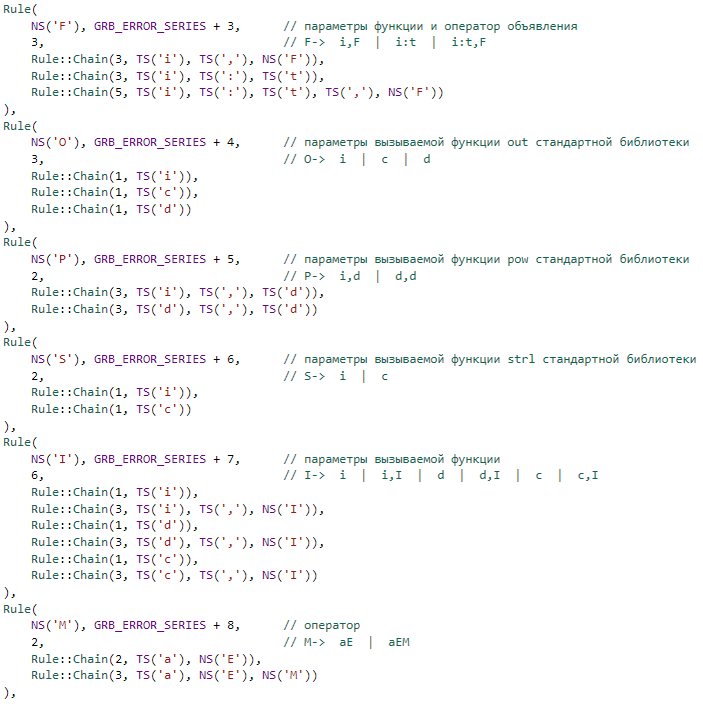


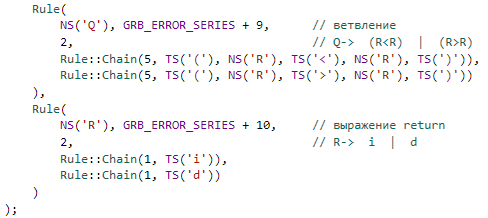




**Приложение Б**

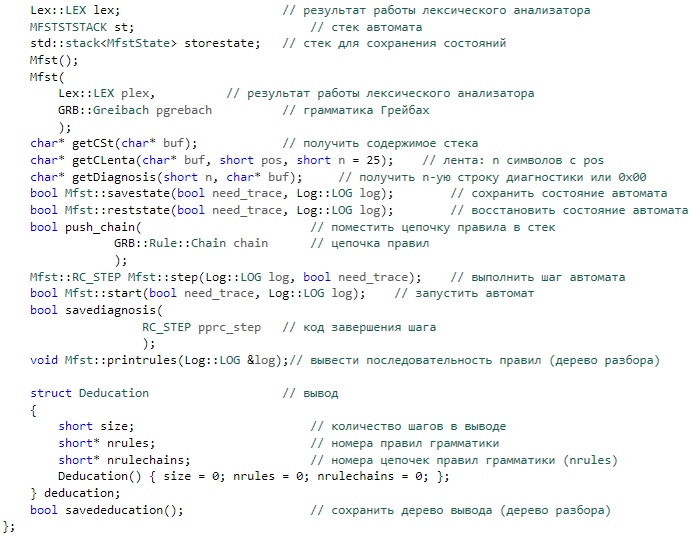






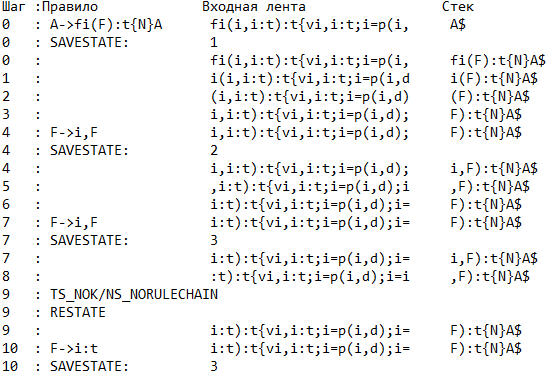
## **Приложение В**



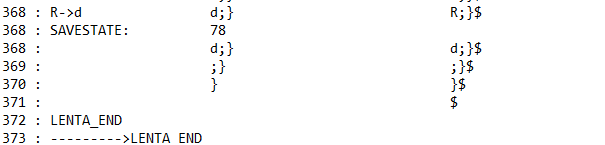


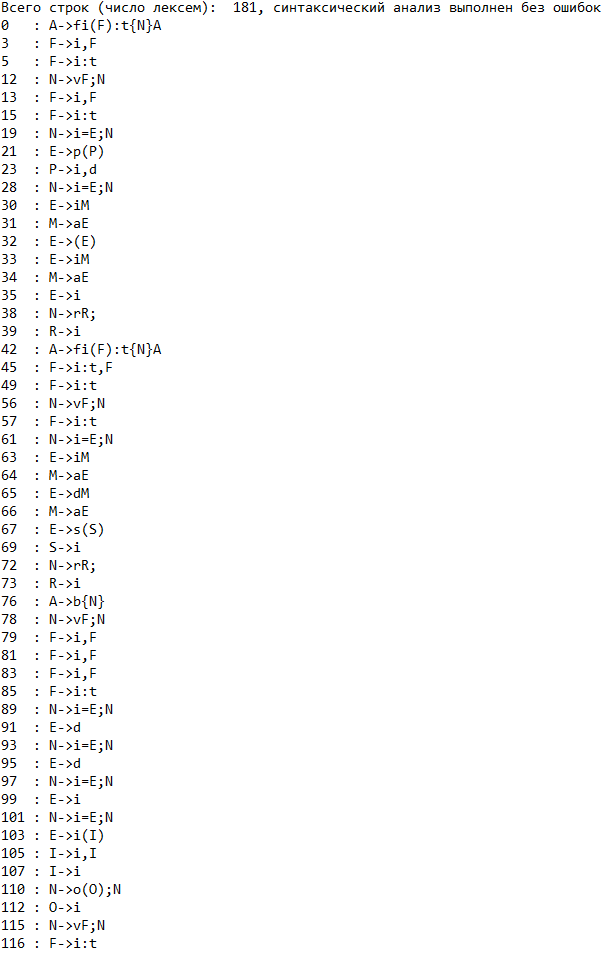
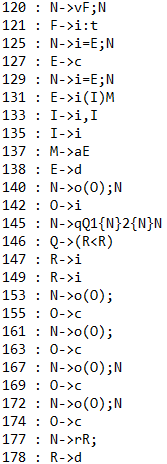
## **Приложение Г**

Начало разбора

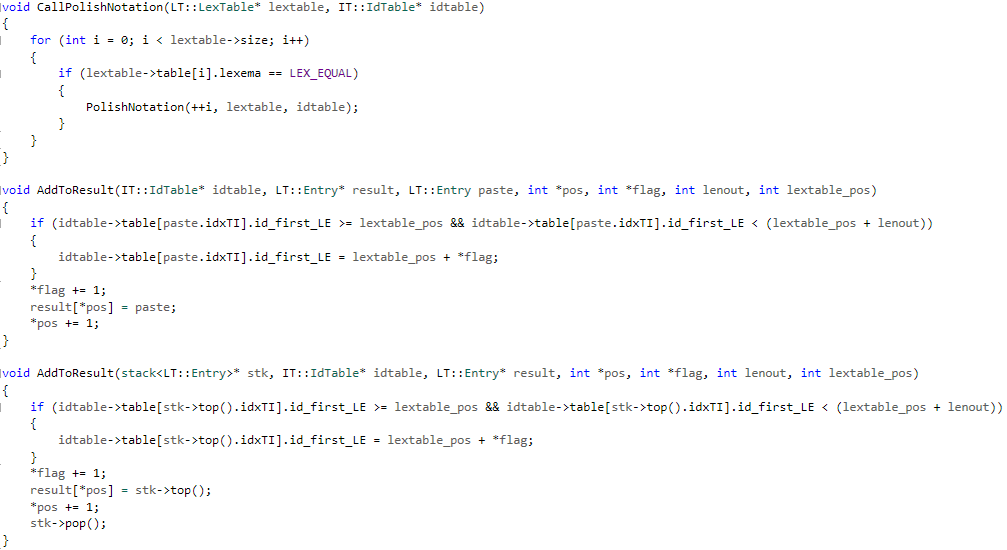


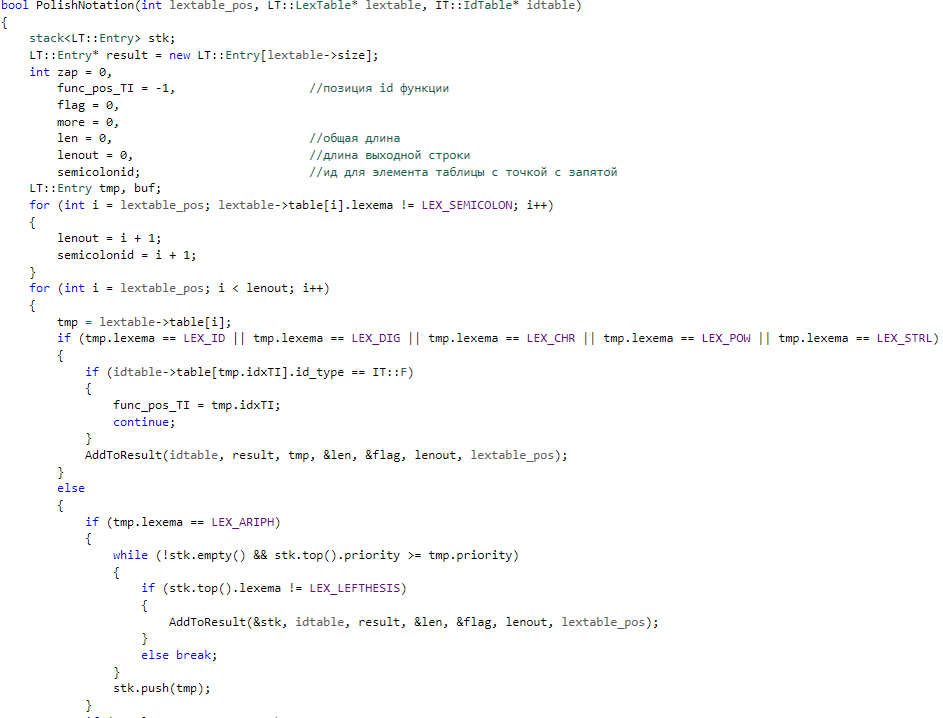
Конец разбора

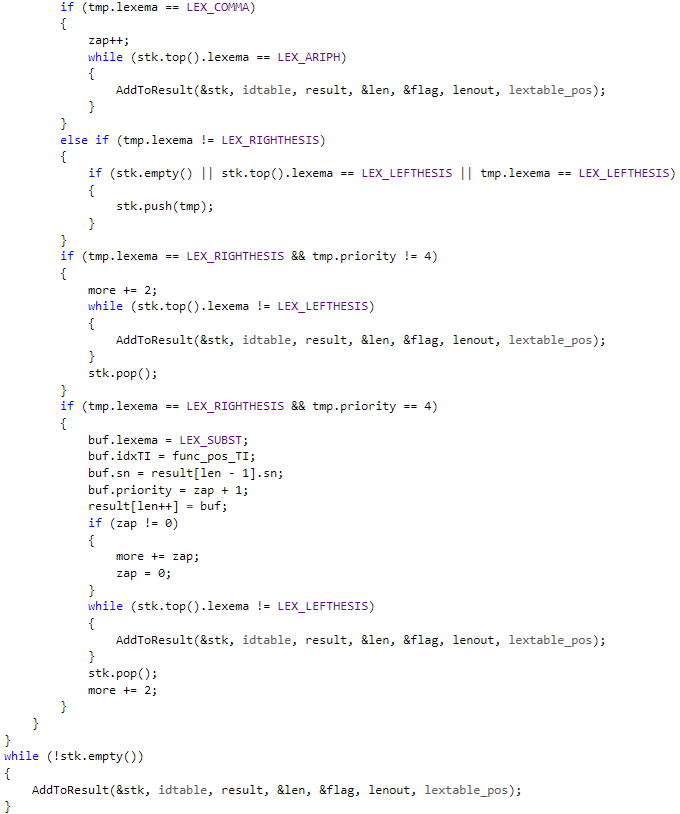


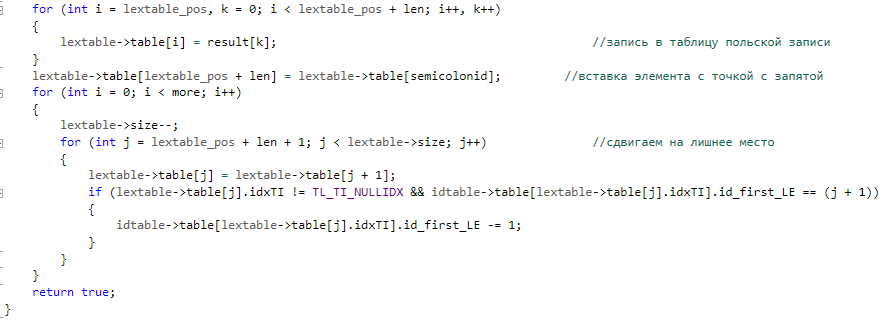
Правила разбора

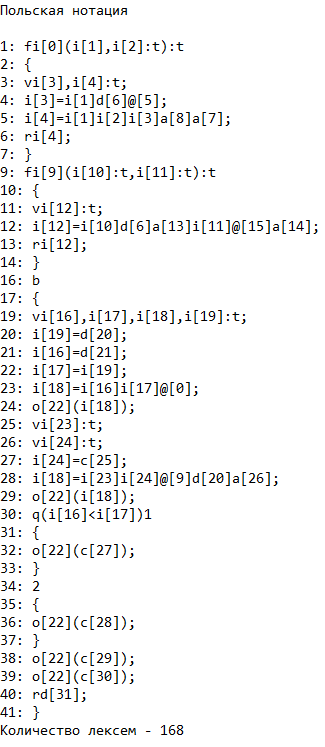
**Приложение Д**

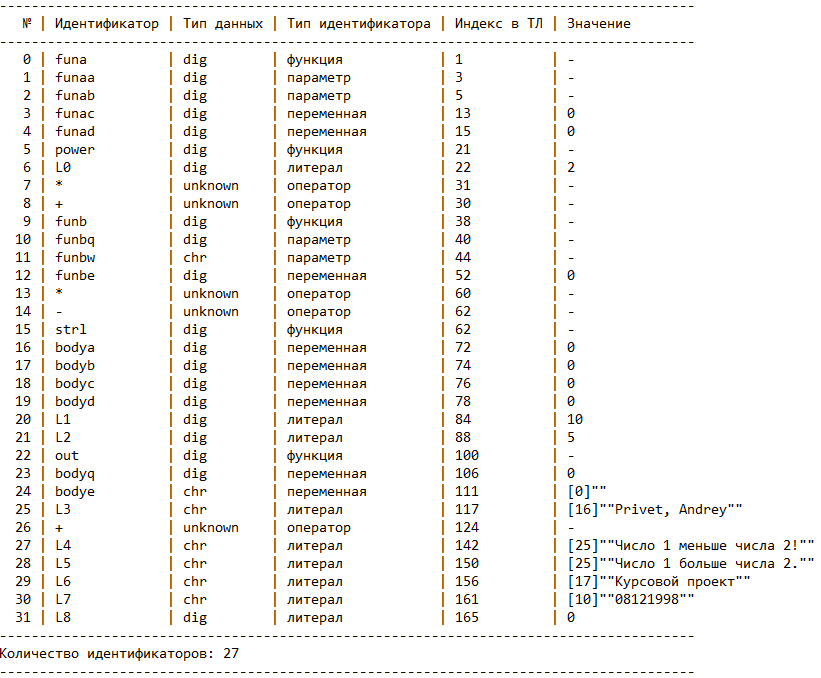




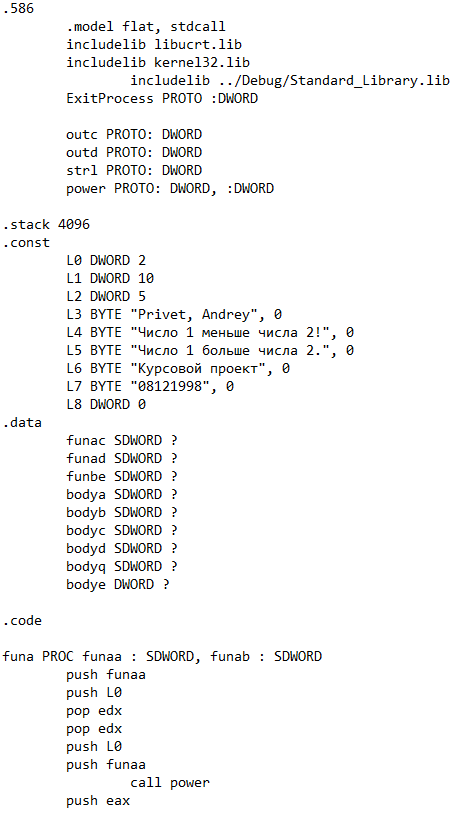
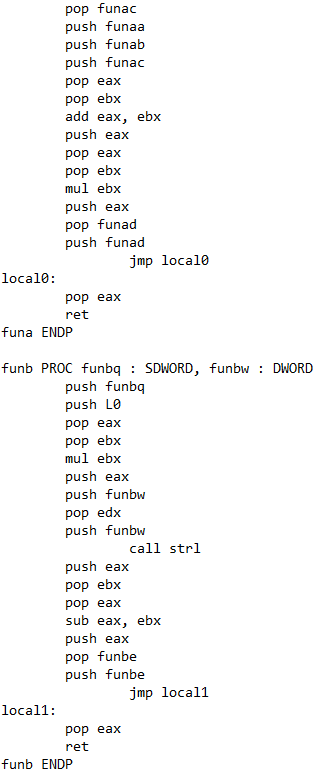


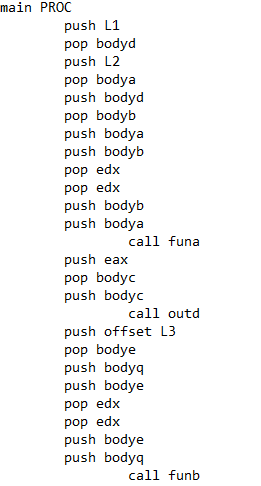
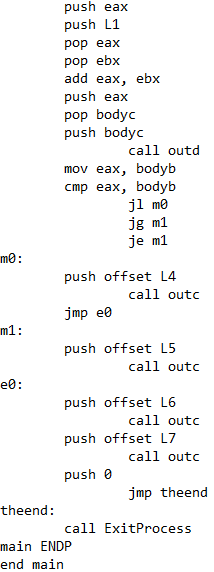






## **Приложение Е**





# **Литература**

1. Ахо А. Компиляторы: принципы, технологии и инструменты / А. Ахо, Р. Сети, Дж. Ульман. – M.: Вильямс, 2003. – 768с.
2. Ирвин К. Р. Язык ассемблера для процессоров Intel / К. Р. Ирвин. – M.: Вильямс, 2005. – 912с.