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 MAG-2018»

Выполнил студент Марковский Артемий Геннадьевич

(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

Минск 2018

Минск 2018

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий   
Кафедра программной инженерии

Утверждаю

Заведующая кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_ Н.В. Пацей

подпись инициалы и фамилия

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2018г.

**ЗАДАНИЕ**

**к курсовому проектированию**

**по дисциплине** "Языки программирования"

Специальность: ПОИТ Группа: 4

Студент: Марковский Артемий Геннадьевич

(фамилия, имя, отчество)

**1. Тема проекта** Разработка компилятора MAG-2018

утверждена приказом по университету от «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г. №

**2. Срок сдачи студентом законченного проекта:** \_\_\_ декабря 2018 г.

**3. Исходные данные к проекту:**

Разработка программы осуществляется на языке C++ (стандартизации International Standard ISO/IEC ISO/IEC 14882:2017 Programming Language C++ 17) в среде разработки Visual Studio 2017 (15.8.4). Операционная система под которой происходит разработка Windows 10 (64-bit). Типы данных: numeral len(row) – длина строки, numeral pows(numeral, numeral) – возведение в степень. Арифметические операции: +, -, \*. Оператор вывода в стандартный поток: output.

**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) Тестирование транслятора (Разработка и тестирование интерпретатора)

Приложения

Литература

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

1) Граф

**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 проект. | Наркевич А. С. |

**7. Календарный план**

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование этапов курсового проекта | Срок выполнения этапов проекта | Примечание |
| 1 | Спецификация специализированного языка MAG-2018 |  |  |
| 2 | Разработка лексического анализатора |  |  |
| 3 | Разработка синтаксического анализатора |  |  |
| 4 | Разработка семантического анализатора |  |  |
| 5 | Генерация кода |  |  |
| 6 | Тестирование компилятора |  |  |
| 7 | Оформление пояснительной записки к курсовому проект |  |  |
| 8 | Сдача проекта |  |  |

**8. Дата выдачи задания 15.09.2018**

Руководитель Наркевич А.С.

(подпись)

Задание принял к исполнению Марковский А.Г.

(дата и подпись студента)

**Содержание**

[**Введение** 3](#_Toc501542835)

[**Глава 1. Спецификация языка программирования** 3](#_Toc501542836)

[**1.1.Характеристика языка программирования** 3](#_Toc501542837)

[**1.2.Алфавит языка** 3](#_Toc501542838)

[**1.3.Символы сепараторы** 3](#_Toc501542839)

[**1.4.Применяемые кодировки** 3](#_Toc501542840)

[**1.5.Типы данных** 3](#_Toc501542841)

[**1.6.Преобразование типов данных** 3](#_Toc501542842)

[**1.7.Идентификаторы** 3](#_Toc501542843)

[**1.8.Литералы** 3](#_Toc501542844)

[**1.9.Область видимости идентификаторов** 3](#_Toc501542845)

[**1.10.Инициализация данных** 3](#_Toc501542846)

[**1.11.Инструкции языка** 3](#_Toc501542847)

[**1.12.Операции языка** 3](#_Toc501542848)

[**1.13.Выражения и их вычисления** 3](#_Toc501542849)

[**1.14.Программные конструкции языка** 3](#_Toc501542850)

[**1.15.Область видимости** 3](#_Toc501542851)

[**1.16.Семантические проверки** 3](#_Toc501542852)

[**1.17.Распределение оперативной памяти на этапе выполнения** 3](#_Toc501542853)

[**1.18.Стандартная библиотека и её состав** 3](#_Toc501542854)

[**1.19.Ввод и вывод данных** 3](#_Toc501542855)

[**1.20.Точка входа** 3](#_Toc501542856)

[**1.21.Препроцессор** 3](#_Toc501542857)

[**1.22.Соглашения о вызовах** 3](#_Toc501542858)

[**1.23.Объектный код** 3](#_Toc501542859)

[**1.24.Классификация сообщений транслятора** 3](#_Toc501542860)

[**1.25.Контрольный пример** 3](#_Toc501542861)

[**Глава 2. Структура транслятора** 3](#_Toc501542862)

[**2.1 Компоненты транслятора, их назначение и принципы взаимодействия** 3](#_Toc501542863)

[**2.2 Перечень входных параметров транслятора** 3](#_Toc501542864)

[**2.3 Перечень протоколов, формируемых транслятором и их содержимое** 3](#_Toc501542865)

[**Глава 3. Разработка лексического анализатора** 3](#_Toc501542866)

[**3.1 Структура лексического анализатора** 3](#_Toc501542867)

[**3.2 Контроль входных символов** 3](#_Toc501542868)

[**3.3 Удаление избыточных символов** 3](#_Toc501542869)

[**3.4 Перечень ключевых слов, сепараторов, символов операций и соответствующих им лексем, регулярных выражений и конечных автоматов** 3](#_Toc501542870)

[**3.5 Основные структуры данных** 3](#_Toc501542871)

[**3.6 Принцип обработки ошибок** 3](#_Toc501542872)

[**3.7 Структура и перечень сообщений лексического анализатора** 3](#_Toc501542873)

[**3.8 Параметры лексического анализатора и режимы его работы** 3](#_Toc501542874)

[**3.9 Алгоритм лексического анализа** 3](#_Toc501542875)

[**3.10 Контрольный пример** 3](#_Toc501542876)

[**Глава 4. Разработка синтаксического анализатора** 3](#_Toc501542877)

[**4.1 Структура синтаксического анализатора** 3](#_Toc501542878)

[**4.2 Контекстно свободная грамматика, описывающая синтаксис языка** 3](#_Toc501542879)

[**4.3 Построение конечного магазинного автомата** 3](#_Toc501542880)

[**4.4 Основные структуры данных** 3](#_Toc501542881)

[**4.5 Описание алгоритма синтаксического разбора** 3](#_Toc501542882)

[**4.6 Структура и перечень сообщений синтаксического анализатора** 3](#_Toc501542883)

[**4.7 Параметры синтаксического анализатора и режимы его работы** 3](#_Toc501542884)

[**4.8 Принцип обработки ошибок** 3](#_Toc501542885)

[**4.9 Контрольный пример** 3](#_Toc501542886)

[**Глава 5. Разработка семантического анализатора** 3](#_Toc501542887)

[**5.1 Структура семантического анализатора** 3](#_Toc501542888)

[**5.2 Функции семантического анализатора** 3](#_Toc501542889)

[**5.3 Структура и перечень сообщений семантического анализатора** 3](#_Toc501542890)

[**5.4 Принцип обработки ошибок** 3](#_Toc501542891)

[**5.5 Контрольный пример** 3](#_Toc501542892)

[**Глава 6. Преобразование выражений** 3](#_Toc501542893)

[**6.1 Выражения, допускаемые языком** 3](#_Toc501542894)

[**6.2 Польская запись** 3](#_Toc501542895)

[**6.3 Программная реализация обработки выражений** 3](#_Toc501542896)

[**6.4 Контрольный пример** 3](#_Toc501542897)

[**Глава 7. Генерация кода** 3](#_Toc501542898)

[**7.1 Структура генератора кода** 3](#_Toc501542899)

[**7.2 Представление типов данных в оперативной памяти** 3](#_Toc501542900)

[**7.3 Алгоритм работы генератора кода** 3](#_Toc501542901)

[**Глава 8. Тестирование транслятора** 3](#_Toc501542902)

[**8.1 Тестирование фазы проверки на допустимость символов** 3](#_Toc501542903)

[**8.2Тестирование лексического анализатора** 3](#_Toc501542904)

[**8.3 Тестирование синтаксического анализатора** 3](#_Toc501542905)

[**8.4 Тестирование семантического анализатора** 3](#_Toc501542906)

[**Заключение** 3](#_Toc501542907)

[**Приложение А** 3](#_Toc501542908)

[**Приложение В** 3](#_Toc501542909)

[**Приложение Г** 3](#_Toc501542910)

[**Приложение Д** 3](#_Toc501542911)

[**Приложение Е** 3](#_Toc501542912)

[**Приложение Ж** 3](#_Toc501542915)

[**Приложение З** 3](#_Toc501542917)

[**Литература** 3](#_Toc501542918)

# **Введение**

Целью курсового проекта является разработка компилятора для своего языка программирования: MAG-2018. MAG-2018 не содержит технически сложно реализуемых компонентов и конструкций, поскольку основное внимание при разработке уделяется компонентам, реализующим важнейшие функции любого компилятора, таких как лексический, синтаксический и семантический анализаторы, а также генератор ассемблерного кода.

Следовательно, основные задачи данной курсовой задачи можно представить в виде:

– разработка лексического и семантического анализатора;

– разработка синтаксический анализатор;

– генератор исходного кода на языке ассемблера.

Разработка данной программной системы подразумевает проход по каждому из этапов классического цикла разработки программного обеспечения, откуда следует ещё одна цель: развить, усовершенствовать и закрепить навыки проектирования, реализации и тестирования программных продуктов.

# **Глава 1. Спецификация языка программирования**

## **Характеристика языка программирования**

Язык MAG-2018 ­– это процедурный, универсальный, строготипизированный, компилируемый язык. Hе является объектно-ориентированным.­

* 1. **Алфавит языка**

Алфавит языка MAG-2018 основан на кодировке Windows-1251, представленной на рисунке 1.1.

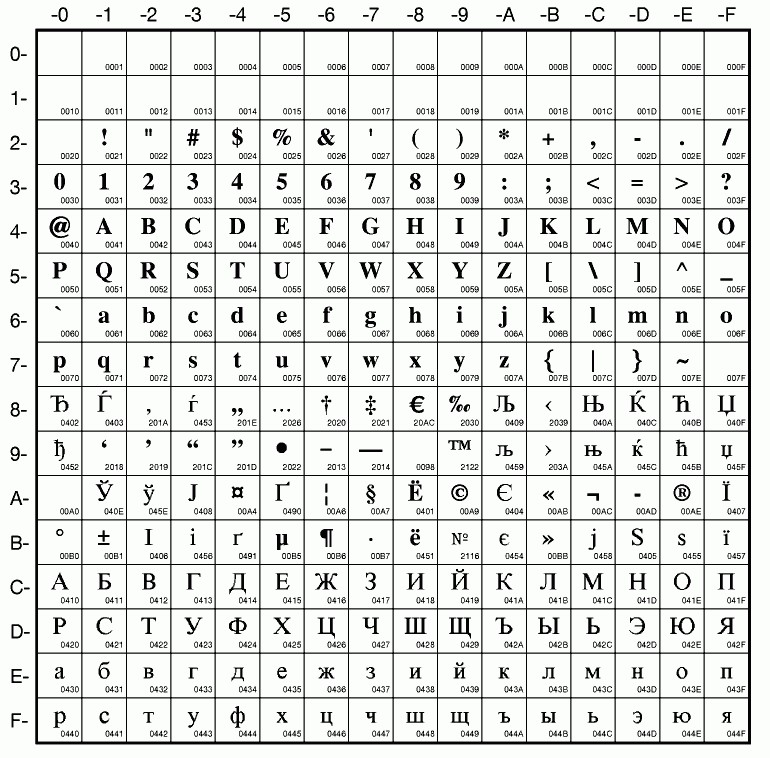


Рисунок 1.1 – Алфавит входных символов

Исходный код MAG-2018 может содержать символы латинского алфавита, цифры десятичной системы счисления от 0 до 9, символы-сепараторы, русские символы разрешены только в строковых литералах.

## **Символы сепараторы**

Символы, которые являются сепараторами представлены в таблице 1.1.

Таблица 1.1 – Сепараторы

|  |  |
| --- | --- |
| Символ-сепаратор | Применение |
| ; | Разделитель инструкций |
| {} | Программный блок функций |
| () | Параметры функций, приоритетность операций |
| = | Присвоение значения |
| Пробел | Разделитель цепочек языка. Может применяться везде кроме ключевых слов и идентификаторов |
| , | Разделитель параметров функции |
| +  -  \* | Арифметические операции |

## **Применяемые кодировки**

Для написания исходного кода на языке программирования MAG-2018 используется кодировка Windows-1251.

## **Типы данных**

В языке MAG-2018 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 MAG-2018

|  |  |
| --- | --- |
| Тип | Пояснение |
| numeral | Является целочисленным типом данных (4 байта). Предназначен для арифметических операций над числами. По умолчанию инициализируется нулём. |
| row | Является строковым типом данных. Каждый символ занимает 1 байт. Максимальное число символов – 255. Последний, 225-й символ является признаком конца строки «\0». По умолчанию инициализируется пустой строкой. |

## **Преобразование типов данных**

В языке программирования MAG-2018 преобразование типов данных не поддерживается, т.е. язык является строготипизированным.

## **Идентификаторы**

Идентификаторы могут выступать в качестве имен функций, имен параметров, имен переменных. Зарезервированные идентификаторы не предусмотрены. Идентификаторы не должны совпадать с ключевыми словами. Для идентификаторов предусмотрено использование символов нижнего регистра латинского алфавита ([a…z]). Максимальная длина идентификатора – 5 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**

Все переменные должны находиться внутри программного блока языка. В языке MAG-2018 требуется обязательное объявление переменной перед её использованием. Инициализация переменной происходит после ее объявления. Имеется возможность объявления одинаковых переменных в разных блоках. Каждая переменная получает префикс – название функции, в которой она объявлена.

## **Инициализация данных**

Инициализация данных в языке программирования MAG-2018, представлена в таблице 1.4.

Таблица 1.4 – Инициализации переменных

|  |  |
| --- | --- |
| Инициализации | Примечание |
| tabula <тип данных> <идентификатор>; | Автоматическая инициализация при объявлении. Переменные типа numeral инициализируются нулём, переменные типа row – пустой строкой. |

## **Инструкции языка**

Все возможные инструкции языка программирования MAG-2018 представлены в общем виде в таблице 1.5.

Таблица 1.5 – Инструкции языка программирования MAG-2018

|  |  |
| --- | --- |
| Инструкция | Запись на языке MAG-2018 |
| Объявление переменной | tabula <тип данных> <идентификатор>; |
| Присваивание | <идентификатор> = <значение>/<идентификатор>; |
| Объявление функции | <тип данных> munus <идентификатор> (<тип данных> <идентификатор>, …) {…} |
| Блок инструкций | main {…} |
| Возврат из функции | reditum <идентификатор>/<литерал>; |
| Вывод данных | output (<идентификатор>/<литерал>); |
| Вызов функции | <идентификатор>(<идентификатор>, …);  <идентификатор>(<литерал>, …);  <идентификатор>(<идентификатор>,<литерал>, …); |

## **Операции языка**

В языке MAG-2018 предусмотрены следующие операции, представленные в таблице 1.6.

Таблица 1.6 – Операций языка программирования MAG-2018

|  |  |  |
| --- | --- | --- |
| Операция | Название | Приоритет |
| + | Сложение | 2 |
| - | Вычитание | 2 |
| \* | Умножение | 3 |

Выражение в скобках, обозначающих приоритет операций, имеет приоритет равный 0. Параметры функций имеют приоритет 4.

## **Выражения и их вычисления**

В языке MAG-2018 реализованы следующие правила при составлении и вычислении выражений:

– допускается использовать скобки для изменения приоритета;

– использование двух подряд идущих операторов не допускается;

– выражение может содержать вызов функции;

– каждое выражение обязано заканчиваться сепаратором (;)

– выражение просматриваются слева направо и выполняются в соответствии с приоритетом.

## **Программные конструкции языка**

Ключевые программные конструкции языка программирования MAG-2018 представлены в таблице 1.7.

Таблица 1.7 – Программные конструкции языка MAG-2018

|  |  |
| --- | --- |
| Конструкция | Синтаксис |
| Главная функция | main{ <программный блок> } |
| Функция | <тип> munus <идентификатор> (<тип> <идентификатор>, …)  {reditum <выражение/идентификатор>;} |

## **Область видимости**

В языке MAG-2018 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 | Все идентификаторы должны быть объявлены |
| 2 | При объявлении идентификатора должен указываться тип |
| 3 | Длина идентификатора не должна превышать 5 символов |
| 4 | Тип формальных параметров функции должен совпадать с типом фактических параметров, переданных в функцию |
| 5 | Количество фактических параметров, переданных в функцию должно соответствовать количеству формальных параметров вызываемой функции |
| 6 | Длина строчного литерала не должна превышать 255 символов без учёта кавычек |
| 7 | Применение арифметических операций допускается только с целочисленным типом numeral |
| 8 | Обязательное наличие главной функции main |
| 9 | Названия функций не должны повторяться |
| 10 | Тип возвращаемого значения и тип функции должны соответствовать |
| 11 | Наличие строго одной точки входа в программу |
| 12 | Нет повторяющихся объявлений идентификаторов в одной области видимости |

## **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 | Применение |
| pows(numeral a, numeral b) | Возведение a в степень b |
| len(row a) | Определение длины строки а |
| output | Вывод на консоль |

## **Ввод и вывод данных**

Для вывода предусмотрен оператор output (<идентификатор/литерал>).

Ввод данных не поддерживается языком программирования MAG-2018.

## **Точка входа**

В языке MAG-2018 точкой входа является главная функция main.

## **Препроцессор**

Препроцессор в языке программирования MAG-2018 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**

Исходный текст на языке MAG-2018 транслируется на язык ассемблера.

## **Классификация сообщений транслятора**

В случае возникновения ошибки в коде программы на языке MAG-2018 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 |
| 0-99 | Системные ошибки |
| 100-109 | Ошибки параметров |
| 110-119 | Ошибки открытия и чтения файлов |
| 120-199 | Ошибки лексического анализа |
| 200-299 | Ошибки синтаксического анализа |
| 300-399 | Ошибки семантического анализа |

## **Контрольный пример**

Контрольный пример представлен в Приложении А.

# **Глава 2. Структура транслятора**

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

В состав компилятора входят три основных компонента: лексический анализатор, семантический анализатор, синтаксический анализатор и генератор машинного кода. Процесс компиляции текста программы в объектный код разбивается на несколько независимых фаз, которые реализуются соответствующими блоками компилятора. Общая схема работы компилятора приведена на рисунке 2.1



Рисунок 2.1 – Структура транслятора

Лексический анализатор – часть транслятора, выполняющая лексический анализ. Основной задачей лексического анализатора является формирование таблицы лексем и таблицы идентификаторов, необходимых для дальнейших этапов работы транслятора, а также обнаружение определённых ошибок. Результат работы лексического анализа фиксируется в файле-протоколе. На вход лексического анализатора поступает файл с исходным кодом программы. По окончанию данного этапа получаем таблицу лексем и таблицу идентификаторов.

Семантический анализатор – часть транслятора, выполняющая семантический анализ, то есть исходный код проверяется на наличие ошибок, которые невозможно отследить при помощи регулярной и контекстно-свободной грамматики. Входными данными являются таблица лексем и идентификаторов. Код семантического анализатора реализуется в лексическом анализаторе.

Синтаксический анализатор – часть транслятора, выполняющая синтаксический анализ. Проверяется правильность расположения идентификаторов и ключевых слов в исходном коде. Для того, чтобы провести данную операцию на вход поступает таблица лексем и идентификаторов. На выходе получаем дерево разбора.

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 MAG-2018, прошедший все предыдущие этапы, в код на языке Ассемблера.

## **2.2 Перечень входных параметров транслятора**

Входные параметры представлены в таблице 2.1.

Таблица 2.1 ­– Входные параметры транслятора языка MAG-2018

|  |  |  |
| --- | --- | --- |
| Входной параметр | Описание | Значение по умолчанию |
| -in:<имя\_файла> | Входной файл с расширением .txt, в котором содержится исходный код. Является обязательным параметром | 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>.log |
| -syn:<имя\_файла> | Файл для записи результата работы синтаксического анализа. | <имя\_файла>.txt |

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

Таблица с перечнем протоколов, формируемых в процессе выполнения программы, и их назначением представлена в таблице 2.2

Таблица 2.3 ­– Классификация сообщений транслятора

|  |  |
| --- | --- |
| Формируемый протокол | Описание протокола |
| Файл с параметром -log: | Содержит информацию о входных параметрах |
| Файл с параметром -out: | Содержит результат перевода исходного кода на язык ассемблера. |
| Файл с параметром -lex: | Содержит таблицы лексем и идентификаторов. |
| Файл с параметром -syn: | Содержит правила разбора, а также преобразованные после польской записи таблицы лексем и идентификаторов. |

# **Глава 3. Разработка лексического анализатора**

## **3.1 Структура лексического анализатора**

Для начала работы лексического анализатора, на его вход подаётся исходный текст, который рассматривается как несвязный поток символов программы. Во время работы лексического анализа первым этапом код разбивается на слова, для более простого оперирования им. Затем каждое слово подаётся на детерминированный автомат, на котором при разборе определяется нужный граф для определения лексемы. В результате удачного разбора каждой цепочки слов на выходе работы лексического анализатора получаем заполненную таблицу лексем и идентификаторов. Общая схема работы лексического анализатора представлена на рисунке 3.1.



Рисунок 3.1 ­– Структура лексического анализатора MAG-2018

## **3.2 Контроль входных символов**

Таблица для контроля входных символов представлена на рисунке 3.2

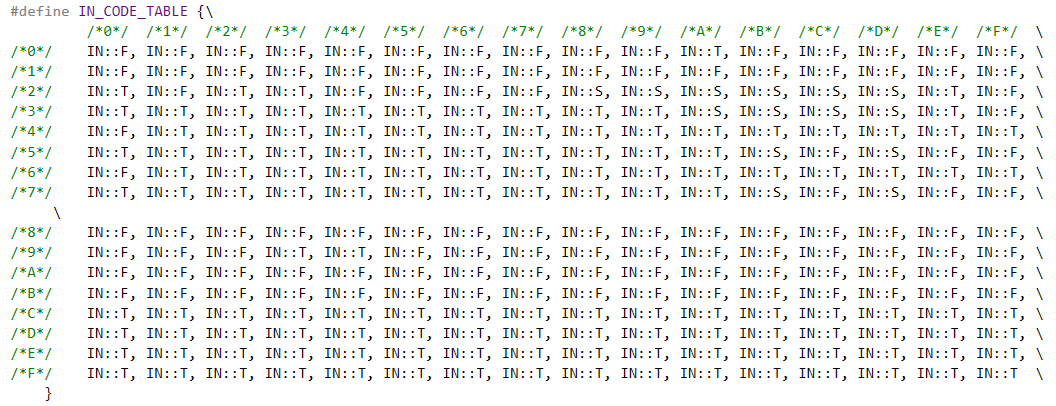


Рисунок 3.2­ – Таблица контроля входных символов

Таблица содержит типы входных символов. Каждый элемент массива соответствует номеру символа в таблице Windows-1251. Символы, представленные на рисунке 3.2, представлены в таблице 3.1.

Каждый считанный из файла с исходным кодом символ проверяется на допустимость, в случае, если символ помечен значением T либо S – записывается в цепочку, F – срабатывает обработчик исключений и генерируется ошибка

Таблица 3.1 ­– ­­­­ Входные символы

|  |  |
| --- | --- |
| Символ | Значение в таблице входных символов |
| T | Разрешенный |
| F | Запрещенный |
| S | Сепаратор |

## **3.3 Удаление избыточных символов**

Удаление избыточных символов является неотъемлемой частью лексического анализа. Избыточные символы удаляются на этапе формирования слов, которые поступят на вход лексическому анализатору. Все незначащие пробелы и символы табуляции не участвуют в формировании слов, если только они не внутри строкового литерала.

Описание алгоритма удаления избыточных символов:

– посимвольно считываем файл с исходным кодом программы;

– если встречаем кавычку, то символы пробелов внутри кавычек не удаляются.

– встреча символа табуляции внутри кавычек генерирует исключение;

– при встрече символа пробела увеличиваем счётчик игнорируемых символов на единицу и переходим к следующему символу;

– символ табуляции так же не записывается в результирующую очередь;

– увеличиваем счётчик линий на единицу и обнуляем позицию строке.

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

Ключевые слова - это последовательности символов алфавита языка, имеющие специальное назначение, зарезервированные для обозначения типов переменных, класса хранения, элементов операторов. В языке MAG-2018 предусмотрены ключевые слова, сепараторы, символы операций и соответствующие им лексемы, приведённые в таблице 3.2.

Таблица 3.2 ­– Ключевые слова, сепараторы, операторы

|  |  |  |
| --- | --- | --- |
| Тип цепочки | Цепочка | Лексема |
| Ключевые слова | tabula | v |
| numeral | x |
| row | y |
| munus | f |
| output | o |
| reditum | r |
| main | m |
| pow | p |
| len | n |
| Сепараторы | ; | ; |
| , | , |
| { | { |
| } | } |
| ( | ( |
| ) | ) |
| " | " |
| = | = |
| Операторы | + | a |
| - | a |
| \* | a |
| / | a |
| Остальное | Идентификатор | i |
| Тип идентификатора | t |
| Целочисленный литерал | x |
| Строковый литерал | y |

## **3.5 Основные структуры данных**

Продолжение таблицы 3.2

Описание основных структур данных, используемых для хранения таблиц лексем и идентификаторов, представлено в приложении Б.

## **3.6 Принцип обработки ошибок**

В компиляторе MAG-2018, при столкновении с исключением, сразу же генерируется ошибка и прекращается работа компилятора.

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

Перечень сообщений, формируемых лексическим анализатором в ходе своей работы, представлен в таблице 3.3.

Таблица 3.3 ­– ­­­­ Перечень сообщений лексического анализатора

|  |  |
| --- | --- |
| Код ошибки | Сообщение |
| 120 | Цепочка символов не разобрана |
| 121 | Таблица лексем переполнена |
| 122 | Таблица идентификаторов переполнена |
| 123 | Дублирование идентификаторов функции |
| 124 | Дублирование арифметических операций |

## **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. Структуры формируются на этапе проверки исходного кода на допустимость символов.

## **3.9 Алгоритм лексического анализа**

Лексический анализ в языке MAG-2018 основан на работе конечных автоматов. Взятое с вершины очереди слово проверяется с помощью детерминированного конечного автомата. При удачном разборе цепочки лексема и информация о ней заносится в таблицу лексем. Если лексема соответствует идентификатору, литералу или оператору, то заносится в таблицу идентификаторов.

Работу конечного автомата можно представить в виде графа. Граф конечного автомата для цепочки row представлен на рисунке 3.4. S0 – начальное состояние, S3 – конечное состояние автомата. Цепочка считается разобранной, если автомат перешёл в конечное состояние и входная лента не имеет больше символов.

r

w

o

Рисунок 3.4 – Граф переходов для цепочки row

## **3.10 Контрольный пример**

Результат работы лексического анализатора представлен в Приложении В.

# **Глава 4. Разработка синтаксического анализатора**

## **4.1 Структура синтаксического анализатора**

Второй этап компиляции *–* это синтаксический разбор, называемый также грамматическим разбором, на нем проверяется правильность следования операторов. Входом для синтаксического анализа является таблица лексем (токенов) и таблица идентификаторов. 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 MAG-2018 используется контекстно-свободная грамматика , где

*–* T – множество терминальных символов;

*–* N – множество нетерминальных символов;

*–* P – множество правил языка;

*–* S – начальный символ грамматики, являющийся нетерминалом.

Контекстно-свободная грамматика  имеет нормальную форму Грейбах, если 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.

Перечень правил, составляющих грамматику языка MAG-2018 представлен в таблице 4.1.

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

|  |  |  |
| --- | --- | --- |
| Нетерминал | Цепочки правил | Описание |
| S | tfi(F){N}S  m{N} | Порождает правила, описывающее общую структуру программы |
| N | vti;N  rE;  i=E;N  o(i);N  o(y);N  o(x);N  o(i);  o(y);  o(x); | Порождает правила, описывающие инструкции языка |
| E | i  y  x  (E)  iM  xM  i(W)  iM  vM  (E)M  i(W)M  p(i,x)  p(x,i)  p(x,x)  p(i,i)  n(y)  n(i) | Порождает правила, описывающие выражения и параметры функций стандартной библиотеки |
| F | ti  ti,F | Порождает правила, описывающие параметры локальной функции при её объявлении |
| W | i  y  x  i,W  y,W  x,W | Порождает правила, описывающие параметры при вызове локальной функции |
| M | aE  aEM | Порождает правила, описывающие  арифметические операции |

## **4.3 Построение конечного магазинного автомата**

Конечный автомат с магазинной памятью представляет собой семерку, описание которой представлено в таблице 4.2.

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. |
|  | 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 S) |
|  | Начальное состояние магазина автомата | Символ маркера дна стека ($) |
|  | 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 MAG-2018. Данные структуры представлены в приложении Г.

## **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 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.

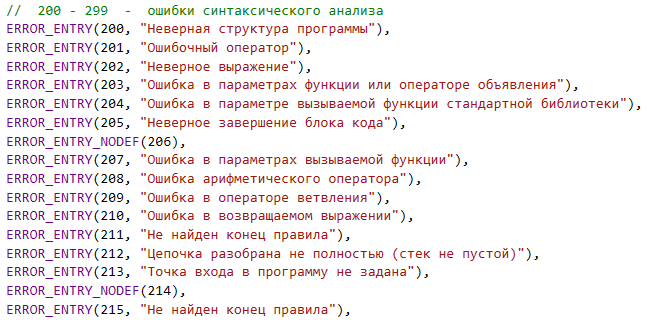


Рисунок 4.1 – Перечень сообщений синтаксического анализатора

## **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 является 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 MAG-2018 представлен в приложении Д. Дерево разбора исходного кода также представлено в приложении Д.

# **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.

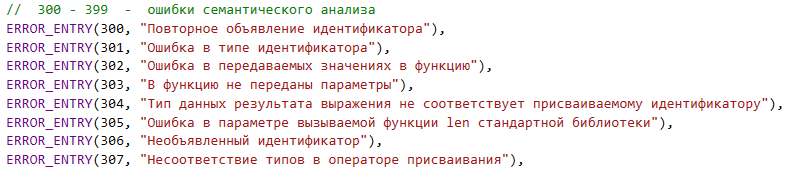


Рисунок 5.1 – Перечень сообщений семантического анализатора

## **5.4 Принцип обработки ошибок**

Принцип обработки ошибок идентичен принципу обработки ошибок на этапе лексического анализа.

## **5.5 Контрольный пример**

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

## **Глава 6. Преобразование выражений**

## **6.1 Выражения, допускаемые языком**

В языке MAG-2018 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 MAG-2018

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

## **6.2 Польская запись**

Выражения в языке MAG-2018 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.

## **6.3 Программная реализация обработки выражений**

Программная реализация алгоритма преобразования выражений к польской записи представлена в Приложении E.

## **6.4 Контрольный пример**

Контрольный пример представлен в Приложении Ж.

## **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.

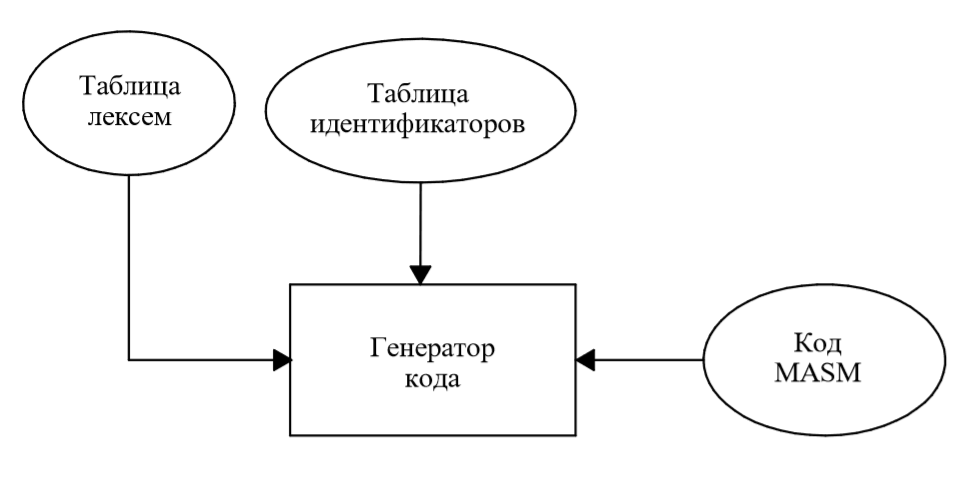


Рисунок 7.1 *–* Структура генератора кода

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

Элементы таблицы идентификаторов расположены в разных сегментах языка ассемблера – .data и .const. Идентификаторы языка MAG-2018 размещены в сегменте данных(.data). Литералы – в сегменте констант (.const). Соответствия между типами данных идентификаторов на языке MAG-2018 и на языке ассемблера приведены в таблице 7.1.

Таблица 7.1 – Соответствия типов идентификаторов языка MAG-2018 и языка Ассемблера

|  |  |  |
| --- | --- | --- |
| Тип идентификатора на языке MAG-2018 | Тип идентификатора на языке ассемблера | Пояснение |
| numeral | SDWORD | Хранит целочисленный тип данных со знаком. |
| row | DWORD | Хранит указатель на начало строки. |
| L(0-7) | BYTE  DWORD | Литералы: символьные,  целочисленные |

## **7.3 Алгоритм работы генератора кода**

Преобразования происходят по принципу, 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.

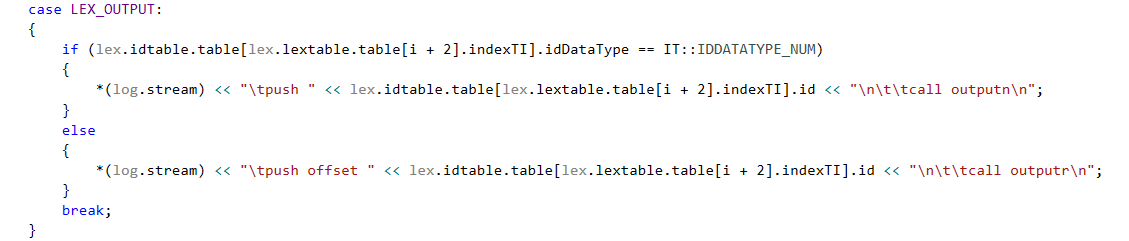


Рисунок 7.2 – Алгоритм для лексемы ‘o’

Сгенерированный код можно посмотреть в приложении З.

# **Глава 8. Тестирование транслятора**

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

В языке MAG-2018 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 |
| numeral munus afunc [numeral a, numeral b) | Ошибка 111: Ошибка проверки входного файла. Недопустимый символ в исходном файле (-in) строка 0, позиция 21 |

## **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 |
| Ta3bula numeral c; | Ошибка 120: Цепочка символов не разобрана строка 3 |

## **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 |
| numeral munus afunc (muneralll a, numeral b) | Ошибка 203: Ошибка в параметрах функции или операторе объявления строка 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 |
| tabula numeral c;  tabula numeral c; | Ошибка 300: Повторное объявление идентификатора строка 4 |

## **Заключение**

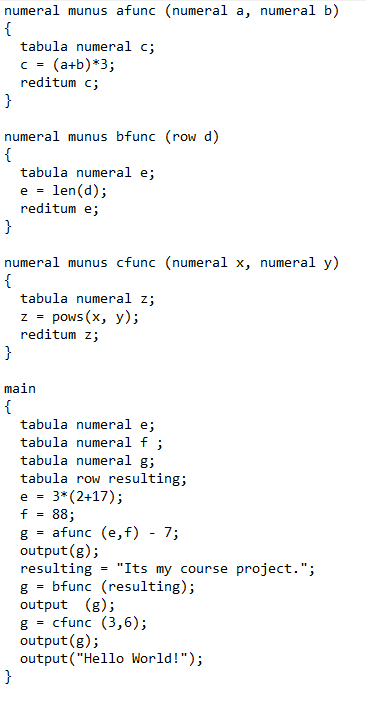
В ходе выполнения курсовой работы был разработан транслятор для языка программирования MAG-2018. Таким образом, были выполнены основные задачи данной курсовой работы:

* Сформулирована спецификация языка MAG-2018;
* Разработаны конечные автоматы и алгоритмы для реализация лексического анализатора;
* Разработана контекстно-свободная, приведённая к нормальной форме Грейбах, грамматика для описания синтаксически верных конструкций языка;
* Разработан семантический анализатор, осуществляющий проверку смысла используемых инструкций;
* Разработан транслятор с языка программирования MAG-2018 на язык Aссемблер;
* Проведено тестирование всех вышеперечисленных компонентов.

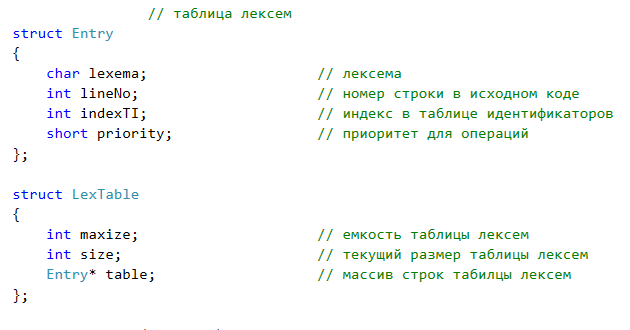
Окончательная версия языка MAG-2018 включает:

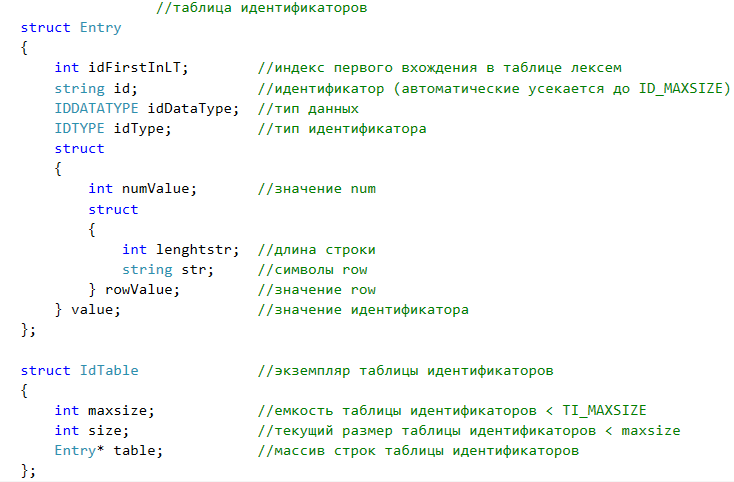
1. 2 типа данных;
2. Поддержка операции вывода;
3. Возможность вызова функций стандартной библиотеки;
4. Наличие 3 арифметических операторов для вычисления выражений;
5. Структурированная система для обработки ошибок пользователя.

## **Приложение А**



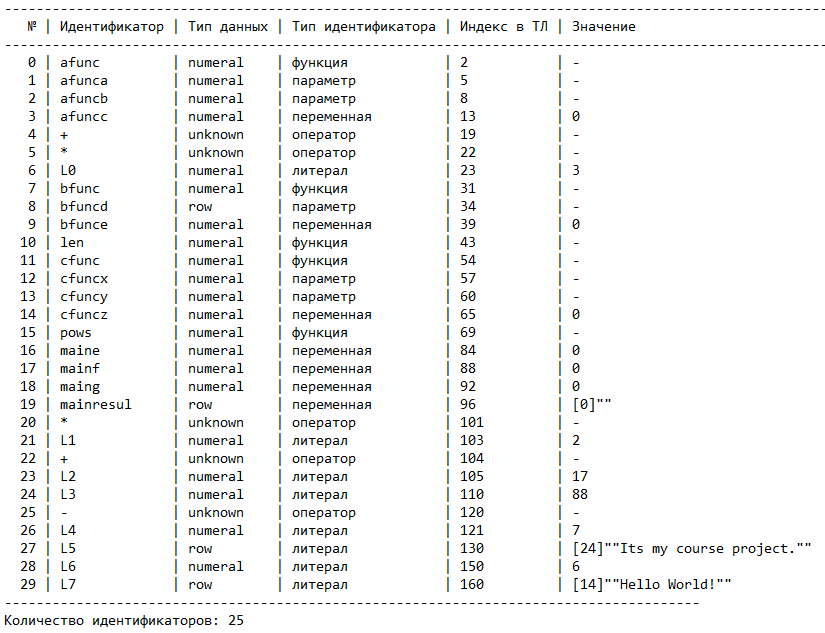
**Приложение Б**

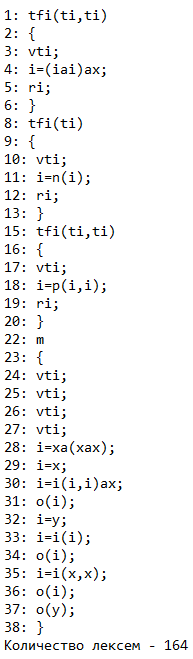




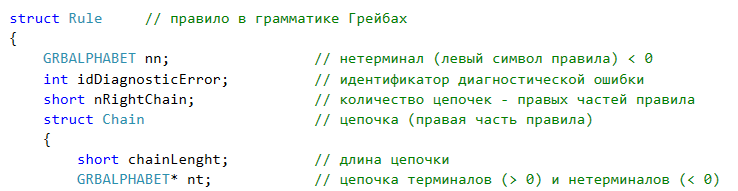
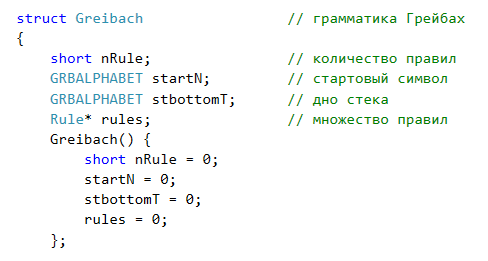
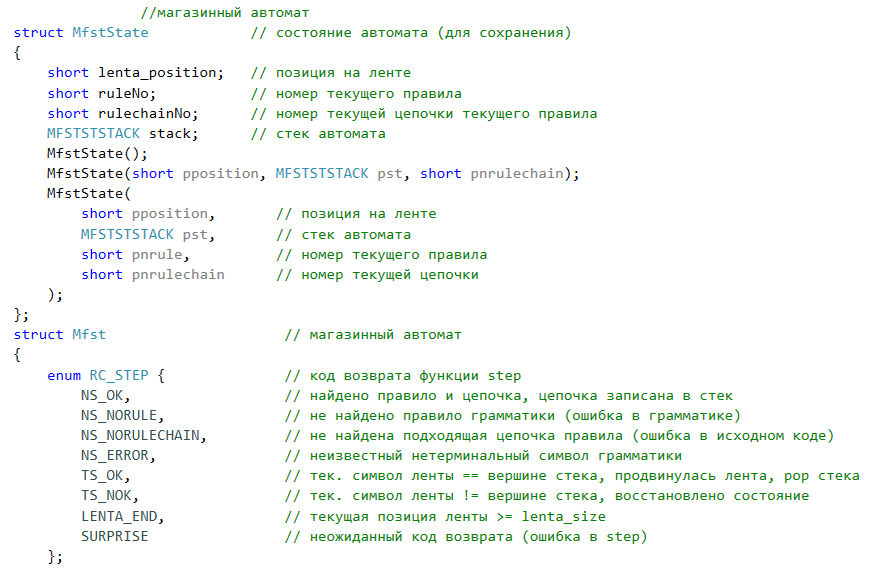


## **Приложение В**



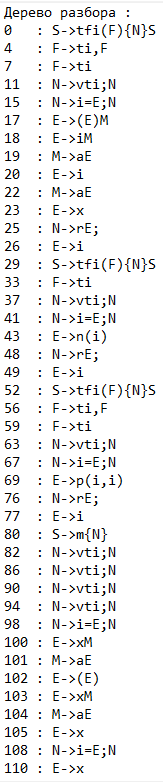
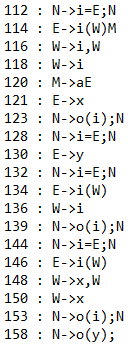


## **Приложение Г**

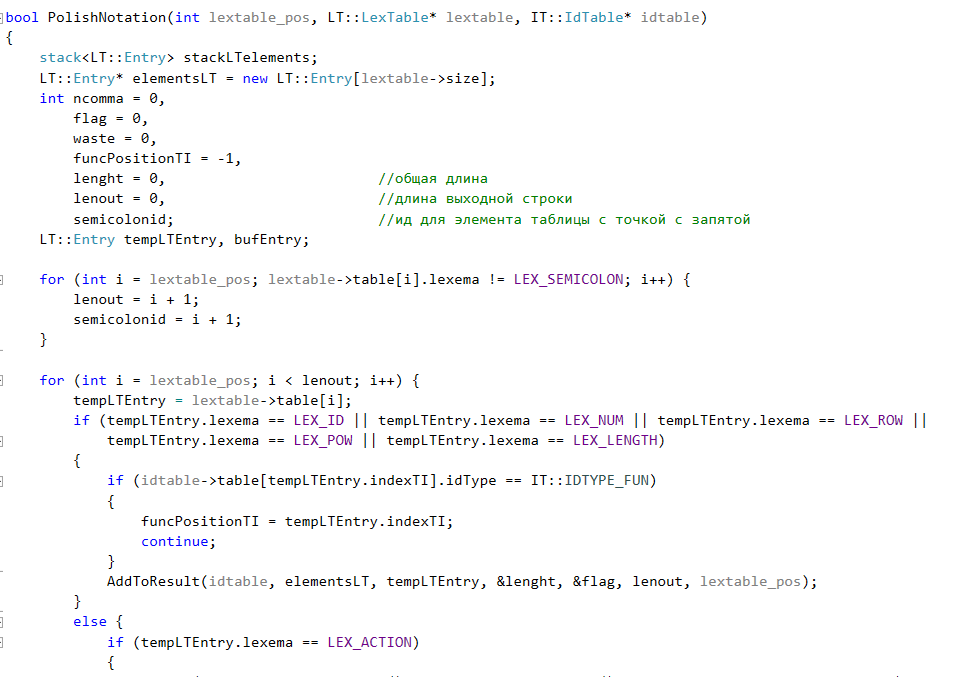


## **Приложение Д**

## 



## **Приложение Е**



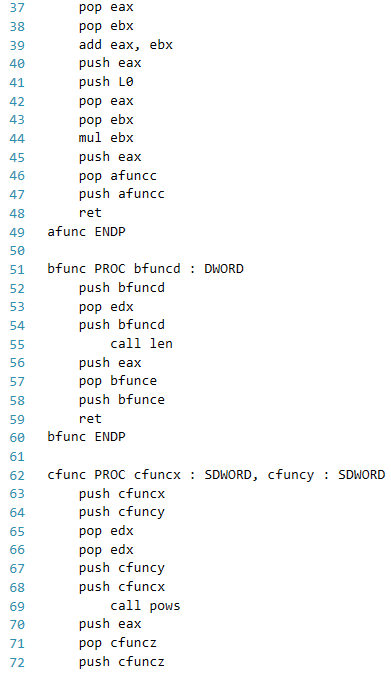
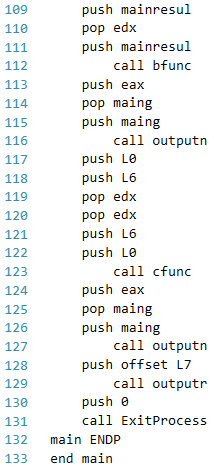
## 

## 

## **Приложение Ж**

## 

## **Приложение З**



## **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.