МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Программной Инженерии

Специальность 6-05-0612-01 “Программная инженерия”

Специализация 6-05-0612-01 “Программная инженерия”

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Разработка компилятора KAA – 2024»

Выполнил студент Кулешов Артём Алексеевич

(Ф.И.О.)

Руководитель проекта Гончар Е. А.

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

Заведующий кафедрой Смелов В.В.

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

Курсовой проект защищен с оценкой

Минск 2024

Содержание

[**Введение** 6](#_Toc185207586)

[**1. Спецификация языка программирования** 7](#_Toc185207587)

[**1.1.** **Характеристика языка программирования** 7](#_Toc185207588)

[**1.2.** **Алфавит языка** 7](#_Toc185207589)

[**1.3.** **Символы сепараторы** 7](#_Toc185207590)

[**1.4.** **Применяемые кодировки** 7](#_Toc185207591)

[**1.5.** **Типы данных** 8](#_Toc185207592)

[**1.6.** **Преобразование типов данных** 9](#_Toc185207593)

[**1.7.** **Идентификаторы** 9](#_Toc185207594)

[**1.8.** **Литералы** 9](#_Toc185207595)

[**1.9.** **Область видимости идентификаторов** 10](#_Toc185207596)

[**1.10.** **Инициализация данных** 10](#_Toc185207597)

[**1.11.** **Инструкции языка** 10](#_Toc185207598)

[**1.12.** **Операции языка** 10](#_Toc185207599)

[**1.13.** **Выражения и их вычисления** 11](#_Toc185207600)

[**1.14.** **Программные конструкции языка** 11](#_Toc185207601)

[**1.15.** **Область видимости** 12](#_Toc185207602)

[**1.16.** **Семантические проверки** 12](#_Toc185207603)

[**1.17.** **Распределение оперативной памяти на этапе выполнения** 12](#_Toc185207604)

[**1.18.** **Стандартная библиотека и её состав** 12](#_Toc185207605)

[**1.19.** **Ввод и вывод данных** 13](#_Toc185207606)

[**1.20.** **Точка входа** 13](#_Toc185207607)

[**1.21.** **Препроцессор** 13](#_Toc185207608)

[**1.22.** **Соглашения о вызовах** 13](#_Toc185207609)

[**1.23.** **Объектный код** 13](#_Toc185207610)

[**1.24.** **Классификация сообщений транслятора** 13](#_Toc185207611)

[**1.25.** **Контрольный пример** 14](#_Toc185207612)

[**2. Структура транслятора** 15](#_Toc185207613)

[**2.1 Компоненты транслятора, их назначение и принципы взаимодействия** 15](#_Toc185207614)

[**2.2 Перечень входных параметров транслятора** 15](#_Toc185207615)

[**2.3 Перечень протоколов, формируемых транслятором и их содержимое** 16](#_Toc185207616)

[**3. Разработка лексического анализатора** 17](#_Toc185207617)

[**3.1 Структура лексического анализатора** 17](#_Toc185207618)

[**3.2 Контроль входных символов** 17](#_Toc185207619)

[**3.3 Удаление избыточных символов** 18](#_Toc185207620)

[**3.4 Перечень ключевых слов, сепараторов, символов операций и соответствующих им лексем, регулярных выражений и конечных автоматов** 18](#_Toc185207621)

[**3.5 Основные структуры данных** 19](#_Toc185207622)

[**3.6 Структура и перечень сообщений лексического анализатора** 20](#_Toc185207623)

[**3.7 Принцип обработки ошибок** 20](#_Toc185207624)

[**3.8 Параметры лексического анализатора и режимы его работы** 20](#_Toc185207625)

[**3.9 Алгоритм лексического анализа** 21](#_Toc185207626)

[**3.10 Контрольный пример** 21](#_Toc185207627)

[**4. Разработка синтаксического анализатора** 22](#_Toc185207628)

[**4.1 Структура синтаксического анализатора** 22](#_Toc185207629)

[**4.2 Контекстно свободная грамматика, описывающая синтаксис языка** 22](#_Toc185207630)

[**4.3 Построение конечного магазинного автомата** 23](#_Toc185207631)

[**4.4 Основные структуры данных** 23](#_Toc185207632)

[**4.5 Описание алгоритма синтаксического разбора** 24](#_Toc185207633)

[**4.6 Структура и перечень сообщений синтаксического анализатора** 24](#_Toc185207634)

[**4.7 Параметры синтаксического анализатора и режимы его работы** 24](#_Toc185207635)

[**4.8 Принцип обработки ошибок** 24](#_Toc185207636)

[**4.9 Контрольный пример** 25](#_Toc185207637)

[**5. Разработка семантического анализатора** 26](#_Toc185207638)

[**5.1 Структура семантического анализатора** 26](#_Toc185207639)

[**5.2 Функции семантического анализатора** 26](#_Toc185207640)

[**5.3 Структура и перечень сообщений семантического анализатора** 26](#_Toc185207641)

[**5.4 Принцип обработки ошибок** 26](#_Toc185207642)

[**5.5 Контрольный пример** 27](#_Toc185207643)

[**6. Преобразование выражений** 28](#_Toc185207644)

[**6.1 Выражения, допускаемые языком** 28](#_Toc185207645)

[**6.2 Польская запись** 28](#_Toc185207646)

[**6.3 Программная реализация обработки выражений** 29](#_Toc185207647)

[**6.4 Контрольный пример** 29](#_Toc185207648)

[**7. Генерация кода** 30](#_Toc185207649)

[**7.1 Структура генератора кода** 30](#_Toc185207650)

[**7.2 Представление типов данных в оперативной памяти** 30](#_Toc185207651)

[**7.3 Статическая библиотека** 31](#_Toc185207652)

[**8.1 Тестирование фазы проверки на допустимость символов** 32](#_Toc185207653)

[**8.2** **Тестирование лексического анализатора** 32](#_Toc185207654)

[**8.3 Тестирование синтаксического анализатора** 32](#_Toc185207655)

[**8.4 Тестирование семантического анализатора** 32](#_Toc185207656)

[**Заключение** 33](#_Toc185207657)

[**Приложение А** 34](#_Toc185207658)

[**Приложение Б** 38](#_Toc185207659)

[**Приложение Г** 48](#_Toc185207660)

[**Приложение Д** 50](#_Toc185207661)

[**Приложение Е** 52](#_Toc185207662)

[**Приложение Ж** 54](#_Toc185207663)

[**Приложение И** 55](#_Toc185207664)

# **Введение**

Задачей данного курсового проекта является разработка транслятора для своего языка программирования: KAA-2024.

Написание транслятора будет осуществляться на языке C++, при этом код на языке KAA-2024 будет транслироваться в язык ассемблера.

Транслятор KAA-2024 состоит из следующих частей:

– семантический анализатор;

– синтаксический анализатор;

– логический 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 будут приведены в соответствующих главах курсового проекта.

# **1. Спецификация языка программирования**

## **Характеристика языка программирования**

Язык программирования KAA-2024 – это универсальный язык высокого уровня. Он является процедурным, компилируемым, не объектно-ориентированным. Язык строго типизируемый, что говорит о невозможности преобразования типов, транслируемым языком программирования.

* 1. **Алфавит языка**

Совокупность символов, используемых в языке, называется алфавитом языка.

На этапе выполнения могут использоваться символы латинского алфавита, цифры десятичной системы счисления от 0 до 9, спецсимволы, а также непечатные символы пробела, табуляции и перевода строки.

## **Символы сепараторы**

Сепараторы необходимы для разделения операций языка. Сепараторы, используемые в языке программирования KAA-2024, приведены в таблице 1.1.

Таблица 1.1 – Сепараторы

|  |  |
| --- | --- |
| [] | используются для записи параметров, условия для условных блоков, возвращаемого значения, вывода |
| {} | определяют границы тела функции и условных блоков |
| () | увеличивают приоритет выражения |
| пробел | разделяет лексемы |
| , | разделяет параметры функции |
| < > ~ # $ ? | разделяют операнды в условном выражении |
| ; | разделяет инструкции |
| “ | определяет границы строковых литералов |
| ‘ | определяет границы символьных литералов |
| = | разделяет присваиваемое выражение и переменную |
| + – / \* % | разделяют операнды в арифметических выражениях |

## **Применяемые кодировки**

Для написания исходного кода на языке программирования KAA-2024 используется кодировка Windows-1251, представленной на рисунке 1.1.

Изображение выглядит как текст, снимок экрана, число, Прямоугольник

Автоматически созданное описание

Рисунок 1.1 – Алфавит входных символов

## **Типы данных**

В KAA-2024 используется четыре типа данных: целочисленные (integer), символьный(symbol), строковые (line), логический (bool). Пользовательские типы данных не поддерживаются. Допускается использование фундаментальных типов данных, определенных в таблице 1.2.

Таблица 1.2 – Типы данных языка KAA-2024

|  |  |
| --- | --- |
| Тип данных | Описание типа данных |
| integer | Фундаментальный тип данных. Предусмотрен для объявления целочисленных данных. Автоматически инициализируется нулевым значением. |
| symbol | Фундаментальный тип данных. Предусмотрен для объявления символов. Автоматическая инициализация нулевым значением. |

Продолжение таблицы 1.2

|  |  |
| --- | --- |
| line | Фундаментальный тип данных. Предусмотрен для объявления строк. Автоматическая инициализация нулевым значением. Максимальное количество символов в строке – 255. |
| bool­ | Фундаментальный тип данных. Предусмотрен для объявления логической переменной, которая имеет одно из двух значений: true, false. Автоматическая инициализация значением false. |

## **Преобразование типов данных**

В языке программирования KAA-2024 преобразование типов данных не поддерживается.

## **Идентификаторы**

Для именования функций, параметров и переменных используются идентификаторы. Идентификаторы, объявленные внутри функционального блока, получают префикс, идентичный имени функции, внутри которой они объявлены. Идентификаторы не должны совпадать с ключевыми словами. Не предусмотрены зарезервированные идентификаторы. Имя идентификатора составляется по следующим образом:

* состоит из символов латинского алфавита нижнего регистра [a..z].
* максимальная длина идентификатора равна 10 и не должна превышать это значение.

## **Литералы**

В языке существует 3 типа литералов: целого, символьного и строкового типов. Краткое описание литералов представлено в таблице 1.3.

Таблица 1.3 – Описание литералов

|  |  |
| --- | --- |
| Тип литерала | Описание |
| Литералы целого типа | Целочисленные литералы десятичного, двоичного и восьмеричного представления, автоматически инициализируются 0. |
| Символьные литералы | Символ, заключённые в “” (двойные кавычки), автоматически инициализируются NULL. |
| Строковые литералы | Символы, заключённые в “” (двойные кавычки), автоматически инициализируются пустой строкой, строковые переменные. Максимальное число символов в литерале . |

## **Область видимости идентификаторов**

Область видимости «сверху вниз» (по принципу С++). В языке KAA-2024 требуется обязательное объявление переменной перед её использованием. Все переменные должны находиться внутри программного блока языка. Имеется возможность объявления одинаковых переменных в разных блоках.

## **Инициализация данных**

Таблица 1.4 – Способы инициализации переменных

|  |  |
| --- | --- |
| Вид инициализации | Примечание |
| ann <тип данных> <идентификатор>; | Автоматическая инициализация: переменные типа integer инициализируются нулём, переменные типа symbol и line – пустой строкой, bool – true/false; |
| <идентификатор> = <значение>; | Присваивание переменной значения. |

## **Инструкции языка**

Все возможные инструкции языка программирования KAA-2024 представлены в общем виде в таблице 1.5.

Таблица 1.5 – Инструкции языка программирования KAA-2024

|  |  |
| --- | --- |
| Инструкция | Запись на языке KAA-2024 |
| Объявление переменной | ann <тип данных> <идентификатор>; |
| Присваивание | <идентификатор> = <значение>|<идентификатор>; |
| Объявление внешней функции | function <тип данных> <идентификатор> [<тип данных> <идентификатор>, …] {…} |
| Блок инструкций | {  …  } |
| Возврат из подпрограммы | give <идентификатор> / <литерал>; |
| Вывод данных | print [<идентификатор> / <литерал>]; |
| Условный оператор | if [<условие>]  { … }  otherwise  { … } |
| Оператор цикла | cycle [<условие>]  { … } |

## **Операции языка**

Язык программирования KAA-2024 может выполнять арифметические операции, представленные в таблице 1.6.

Таблица 1.6 – Приоритетности операций языка программирования KAA-2024

|  |  |
| --- | --- |
| Операция | Приоритетность операции |
| (  ) | 0 |
| , | 1 |
| \*  / (деление нацело)  % (деление с остатком) | 3 |
| +  - | 2 |

Максимальным значением приоритетности является “3”, минимальным “0” 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. Фигурные скобки используются для составления блоков кода функций, операторов. Квадратные – для передачи параметров функций.

## **Программные конструкции языка**

Ключевые программные конструкции языка программирования KAA-2024 представлены в таблице 1.7.

Таблица 1.7 – Программные конструкции языка KAA-2024

|  |  |
| --- | --- |
| Главная функция (точка входа в приложение) | main  { … } |
| Функция | function <тип данных> <идентификатор> [<тип данных> <идентификатор>, …]  { …  give <выражение>;  } |
| Процедура | procedure <идентификатор> [<тип данных> <идентификатор>, …]  { … } |

## **Область видимости**

В языке KAA-2024 переменные обязаны находится внутри программного блока функций. Все объявления и операции с переменными происходят внутри какого-либо блока. Каждая переменная или параметр функции получают префикс – название функции, внутри которой они находятся. 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 стеке. Таблица лексем и таблица идентификаторов сохраняются в структуры с выделенной под них динамической памятью, которая очищается по окончанию работы транслятора.

## **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 |
| combine | line | line a, line b | Функция производит конкатенацию строк a и b, возвращает строку |
| lexcomp | integer | line a, line b | Функция производит сравнение числовые коды символов в строках. Возвращает значение равное 0, если |

Продолжение таблицы 1.9

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | числовые коды строк равны, больше нуля, если числовой код a больше b, меньше нуля в противном случае. |
| measure | integer | line a | Функция вычисляет длину строки a |
| randomize | integer | integer min,  integer max | Функция возвращает случайно сгенерированное число в промежутке [min, max] |
| exponent | integer | integer a,  integer n | Функция возводит число a в степень n, возвращает результат |
| OutputStr | - | line a | Функция выводит в консоль строку a |
| OutputInt | - | integer a | Функция выводит в консоль число a |

## **Ввод и вывод данных**

В языке KAA-2024 не реализованы средства ввода данных. Вывод в стандартный поток вывода: speak [<идентификатор или литерал>];

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

## **Точка входа**

Точкой входа является функция main.

## **Препроцессор**

Препроцессор в языке программирования KAA-2024 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**

KAA-2024 транслируется в язык ассемблера.

## **Классификация сообщений транслятора**

В случае возникновения ошибки в коде программы на языке KAA-2024 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 |
| 0-110 | Системные ошибки, ошибки параметров |
| 200-220 | Ошибки лексического анализа |
| 600-620 | Ошибки синтаксического анализа |
| 300-400 | Ошибки семантического анализа |

## **Контрольный пример**

Контрольный пример представлен в Приложении А.

# **2. Структура транслятора**

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

Транслятор преобразует программу, написанную на языке KAA-2024 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.

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

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.

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 KAA-2024 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 KAA-2024

|  |  |  |
| --- | --- | --- |
| Входной параметр | Описание | Значение по умолчанию |
| -in:<имя\_файла> | Входной файл с расширением .txt, в котором содержится исходный код на KAA-2024 | Не предусмотрено |
| -log:<имя\_файла> | Файл для записи полного протокола работы транслятора | <имя\_файла>.log |
| -out:<имя\_файла> | Файл для записи результата работы транслятора | <имя\_файла>.out.asm |
| -tokens | Ключ для вывода промежуточного представления кода | По умолчанию отсутствует |
| -lex | Ключ для вывода таблицы лексем в консоль | По умолчанию отсутствует |
| -id | Ключ для вывода трассировки синтаксического анализа в файл | По умолчанию отсутствует |

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

Таблица с перечнем протоколов, формируемых транслятором языка

KAA-2024 и их назначением представлена в таблице 2.2

Таблица 2.2 – Протоколы, формируемые транслятором языка KAA-2024

|  |  |
| --- | --- |
| Формируемый протокол | Описание протокола |
| Файл журнала с параметром <log> | Содержит информацию о времени выполнения приложения; входных параметрах в приложение; код на языке KAA-2024 с сепараторами и без избыточных пробелов, табуляций и переходов на новую строку; таблицу идентификаторов; таблицу лексем; промежуточное представление кода; трассировку синтаксического анализа; дерево разбора, время выполнения разбора; промежуточное представление кода после приведения его к польской нотации. |
| Выходной файл c параметром <out> | Содержит сгенерированный код на языке Ассемблера. |

# **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 KAA-2024. На выходе формируется таблица лексем и таблица идентификаторов. Структура лексического анализатора представлена на рисунке 3.1.



Рисунок 3.1 – Структура лексического анализатора KAA-2024

## **3.2 Контроль входных символов**

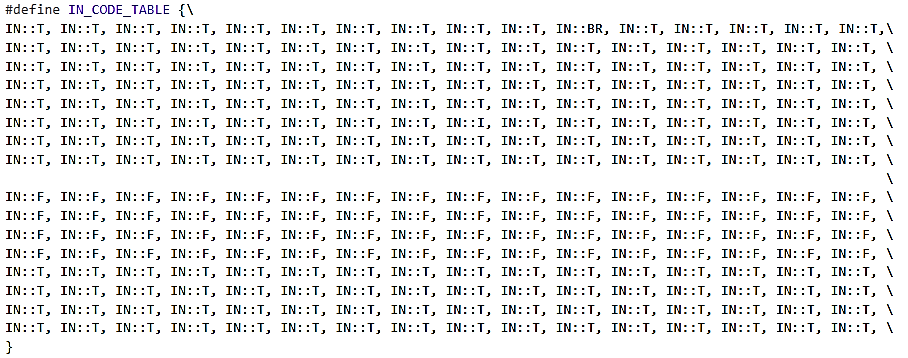
Таблица для контроля входных символов представлена на рисунке 3.2

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, I – игнорируемый символ.

## **3.3 Удаление избыточных символов**

Избыточными символами являются символы табуляции и пробелы.

Избыточные символы удаляются на этапе разбиения исходного кода на лексемы.

Описание алгоритма удаления избыточных символов:

1. Посимвольно считываем исходный код, занесенный в структуру In.

2. Встреча пробела или знака табуляции вне пределов строкового литерала является своего рода встречей символа-сепаратора.

3. В отличие от других символов-сепараторов не записываем в таблицу 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 | ann | c |
| integer, symbol, line, bool | t |
| main | m |
| function | f |
| procedure | p |
| give | r |
| print | s |
| cycle | u |
| if | w |
| otherwise | ! |
| Иное | Идентификатор | i |
| Литерал | l |
| Функции стандартной библиотеки | combine | k |
| lexcomp | s |
| measure | d |
| exponent | e |
| breakl | b |
| randomize | z |

Продолжение таблицы 3.1

|  |  |  |
| --- | --- | --- |
| Сепараторы | ; | ; |
| , | , |
| { | { |
| } | } |
| ( | ( |
| ) | ) |
| [ | [ |
| ] | ] |
| Операторы | + | + |
| - | - |
| \* | \* |
| / | / |
| % | % |
| Логические (~ # > <) | g |
| = | = |

Пример реализации таблицы лексем представлен в Приложении Б.

Также в Приложении B находятся конечные автоматы, соответствующие лексемам языка KAA-2024.

## **3.5 Основные структуры данных**

Описание основных структур данных, используемых для хранения таблиц идентификаторов, представлено на рис. 3.3.

|  |
| --- |
|  |

Рисунок 3.3. — Структуры таблиц идентификаторов KAA-2024

Описание основных структур данных, используемых для хранения таблиц лексем, представлено на рис. 3.4.

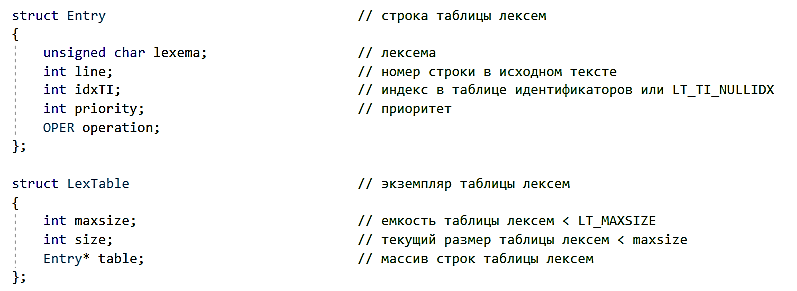


Рисунок 3.4 — Структуры таблиц лексем KAA-2024

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

Перечень сообщений лексического анализатора представлен на рисунке 3.5.

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 3.5 – Перечень ошибок лексического анализатора

## **3.7 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.

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

Входным параметром лексического анализатора является исходный текст программы, написанный на языке KAA-2024, а также файл протокола.

## **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 main.

Граф конечного автомата для этой лексемы представлен на рисунке 3.6. S0 – начальное состояние, S4 – конечное состояние автомата. В виде кода представлен на рисунке 3.7.

|  |
| --- |
| n  m  a  i |

Рисунок 3.6 — Граф переходов для цепочки “main”

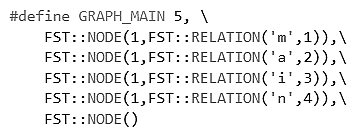


Рисунок 3.7 — Граф переходов для цепочки “main”

## **3.10 Контрольный пример**

Результат работы лексического анализатора – вывод в протокол таблицы лексем и идентификаторов – представлен в приложении Б.

# **4. Разработка синтаксического анализатора**

## **4.1 Структура синтаксического анализатора**

Синтаксический анализ – это вторая 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 KAA-2024 используется контекстно-свободная грамматика , где

1. T – множество терминальных символов.
2. N – множество нетерминальных символов (первый столбец таблицы 4.1).
3. P – множество правил языка (второй столбец таблицы 4.1).
4. 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.

Правила языка KAA-2024 представлена в Приложении Г.

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

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

Перечень правил, составляющих грамматику языка и описание нетерминальных символов KAA-2024 показаны в приложении Г.

## **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 Д.

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 ($) |
|  | 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 KAA-2024. Данные структуры представлены в Приложении Д.

## **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**

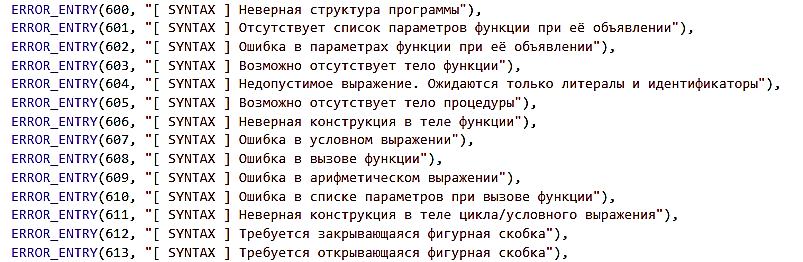
Перечень сообщений синтаксического анализатора представлен на рисунке 4.1.

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, поток вывода протокола, а также правила контекстно-свободной грамматики в форме Грейбах.

Выходными параметрами являются трассировка прохода таблицы лексем и правила разбора, которые записываются в файл протокола.

## **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 предоставлен в Приложении Е в виде фрагмента трассировки и дерева разбора исходного кода.

# **5. Разработка семантического анализатора**

## **5.1 Структура семантического анализатора**

Семантический анализ в трансляторе языка KAA-2024 выделен в отдельную фазу, 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 и т.д.).

## **5.2 Функции семантического анализатора**

За семантический анализ отвечает функция Analyze. Ее входными параметрами является таблица лексем и поток вывода в протокол.

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.

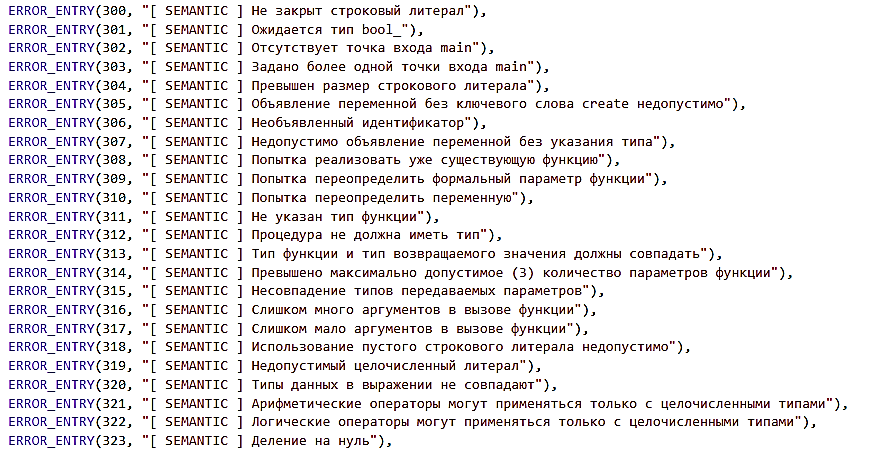


Рисунок 5.1 – Перечень сообщений семантического анализатора

## **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 Б, где показан результат лексического анализатора, т.к. представленные таблицы лексем и идентификаторов проходят лексическую и семантическую проверки одновременно.

Таблица 5.3 – Тестирование функций

|  |  |
| --- | --- |
| Исходный код с ошибкой | Генерируемое сообщение об ошибке |
| {  ann integer a = 5;  print [a];  } | ERROR CODE 302: [ SEMANTIC ] Отсутствует точка входа main  Строка -1 позиция -1 |
| main  {  ann integer a = 5;  print [a];  print ["Bye!];  } | ERROR CODE 300: [ SEMANTIC ] Не закрыт строковый литерал  Строка -1 позиция -1 |
| main  {  ann a = 5;  print [a];  print ["Bye!"];  } | ERROR CODE 307: [ SEMANTIC ] Недопустимо объявление переменной без указания типа  Строка 2 позиция 0 |

# **6. Преобразование выражений**

## **6.1 Выражения, допускаемые языком**

В языке KAA-2024 допускаются выражения, применимые к целочисленным типам данных. В выражениях поддерживаются арифметические операции, такие как +, -, \*, /, %(остаток от деления) и (), и вызовы функций как операнды арифметических выражений.

Приоритет операций представлен в таблице 6.1.

Таблица 6.1 – Приоритет операций в языке KAA-2024

|  |  |
| --- | --- |
| Приоритет | Операция |
| 0 | ( ) |
| 1 | , |
| 2 | + - |
| 3 | \* / % |
| 4 | [ ] – скобки параметров функции |

## **6.2 Польская запись**

Выражения в языке KAA-2024 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 выражений, в которой операторы расположены после своих операндов. Выражение в обратной польской нотации читается слева направо: операция выполняется над двумя операндами, непосредственно стоящими перед знаком этой операции.

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 |
| (l+l)\*l/l |  |  |
| l+l)\*l/l |  |  |
| +l)\*l/l | l |  |
| l)\*l/l | l | + |
| )\*l/l | ll | + |
| \*l/l | ll+ |  |
| l/l | ll+ | \* |
| /l | ll+l | \* |
| l | ll+l\* | / |
|  | ll+l\*l | / |
|  | ll+l\*l/ |  |

## **6.3 Программная реализация обработки выражений**

Программная реализация алгоритма преобразования выражений в обратный польский формат основана функциях Poliz и StartPoliz. Функция StartPoliz принимает как параметр адрес таблицы лексем и содержит цикл, в ходе которого перебираются все лексемы исходного кода. Если последовательность лексем соответствует началу выражения, вызывается функция Poliz, где и проводится точечное преобразование выражений к польской нотации.

## **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 Ж приведен измененное представление промежуточного кода, отображающее результаты преобразования выражений в польский формат.

# **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 KAA-2024 размещены в сегменте данных(.data). Литералы – в сегменте констант (.const). Соответствия между типами данных идентификаторов на языке KAA-2024 и на языке ассемблера приведены в таблице 7.1. Сгенерированный код предоставлен в Приложении З.

Таблица 7.1 – Соответствия типов идентификаторов языка KAA-2024 и языка Ассемблера

|  |  |  |
| --- | --- | --- |
| Тип идентификатора на языке KAA-2024 | Тип идентификатора на языке ассемблера | Пояснение |
| integer | SDWORD | Хранит целочисленный тип данных со знаком. |
| symbol | DWORD | Хранит целочисленный тип данных без знака, размером 1 байт. |
| line | DWORD | Каждый символ строки хранится размером в 1 байт. |
| bool | SDWORD | Хранит целочисленный тип данных без знака. |

## **7.3 Статическая библиотека**

Функции из стандартной библиотеки содержатся в проекте LIB, в свойствах которого указан тип конфигурации «статическая библиотека». Подключение библиотеки происходит с помощью includelib на этапе генерации кода путем вывода в поток out. Таким же образом c помощью оператора EXTRN объявляются названия функций из библиотеки. Оператор EXTRN выполняет две функции. Во-первых, он сообщает ассемблеру, что указанное символическое имя является внешним для текущего ассемблирования. Вторая функция оператора EXTRN состоит в том, что он указывает ассемблеру тип соответствующего символического имени. Так как ассемблирование является очень формальной процедурой, то ассемблер должен знать, что представляет из себя каждый символ.

**8. Тестирование транслятора**

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

В языке KAA-2024 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 |
| Fu№ction integer aver [integer n, integer m] | ERROR CODE 200: [ LEXICAL ] Недопустимый символ в исходном файле (-in)  Строка 1 позиция 4 |

## **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 |
| function integerfhgff\_ aver [intneger n, integer m] | ERROR CODE 205: [ LEXICAL ] Неизвестная последовательность символов  Строка 0 позиция 9 |

## **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 |
| main  {  ann integer hey = 5;  write [hey]; breakl; | ERROR CODE 612: [ SYNTAX ] Требуется закрывающаяся фигурная скобка  Строка -1 позиция -1 |

## **8.4 Тестирование семантического анализатора**

Итоги тестирования семантического анализатора приведены в пункте 5.5.

# **Заключение**

В ходе выполнения курсовой работы был разработан транслятор и генератор кода для языка программирования KAA-2024 со всеми необходимыми компонентами. Таким образом, были выполнены основные задачи данной курсовой работы:

1. Сформулирована спецификация языка KAA-2024;
2. Разработаны конечные автоматы и важные алгоритмы на их основе для

эффективной работы лексического анализатора;

1. Осуществлена программная реализация лексического анализатора,

распознающего допустимые цепочки спроектированного языка;

1. Разработана контекстно-свободная, приведённая к нормальной форме

Грейбах, грамматика для описания синтаксически верных конструкций языка;

1. Осуществлена программная реализация синтаксического анализатора;
2. Разработан семантический анализатор, осуществляющий проверку

используемых инструкций на соответствие логическим правилам;

1. Разработан транслятор кода на язык ассемблера;
2. Проведено тестирование всех вышеперечисленных компонентов.

Окончательная версия языка KAA-2024 включает:

1. 4 типа данных;
2. Поддержка оператора вывода строки;
3. Наличие 5 арифметических операторов для вычисления выражений
4. Наличие 6 логических операторов для использования в условиях цикла и

условной конструкции

1. Поддержка функций; Операторов цикла и условия;
2. Наличие библиотеки стандартных функций языка
3. Структурированная и классифицированная система для обработки ошибок

пользователя.

Проделанная работа позволила получить необходимое представление о структурах и процессах, использующихся при построении трансляторов, а также основные различия и преимущества тех или иных средств трансляции.

# **Приложение А**

function integer funcf [integer a, integer b]

{

ann integer result = 0;

result = (a + b)\* 2 / 2;

result = result \* 2;

give [result];

}

function integer funcs []

{

ann integer noparm = 3;

print[noparm];

give[noparm];

}

procedure cyclen [integer n]

{

ann integer i = 0;

print ["Counting to length: "];

print [n]; stop;

cycle [i < n]

{

i = i + 1;

print [i];

print [" "];

}

stop;

}

main

{

ann integer testfir = funcf[1,5];

print [testfir];

ann integer testsec = funcs[];

print [testsec];

ann line start = combine ["KPO Course Project ", "KAA2024"];

print [start];

ann integer binary = 1b10;

print [binary];

ann integer oct = 0b10;

print [oct];

ann integer diff = binary - oct;

print ["binary - oct = "];

print [diff];

ann bool boolean = true;

print ["Bool: "];

print [boolean];

ann symbol char = "A";

print ["Symbol: "];

print [char];

ann integer exp = 2 + exponent [2, 6];

print ["exp[2, 6] = "];

print [exp];

if[exp ~ 64]

{

print["Pow 2, 6 equal 64"]; stop;

}

otherwise

{

print["Pow 2, 6 not equal 64"]; stop;

}

ann integer length = measure ["Artsiom"];

print ["Length: "];

print [length];

ann integer randd = randomize [1, 52];

print ["Random: "];

print [randd];

ann integer art = 256\*160;

print ["Lexcomp: "];

print [art];

ann integer z = 7;

cyclen[z];

if[z $ 10]

{

print ["If <= 7"]; stop;

}

otherwise

{

print ["If > 7"]; stop;

}

}

# **Приложение Б**

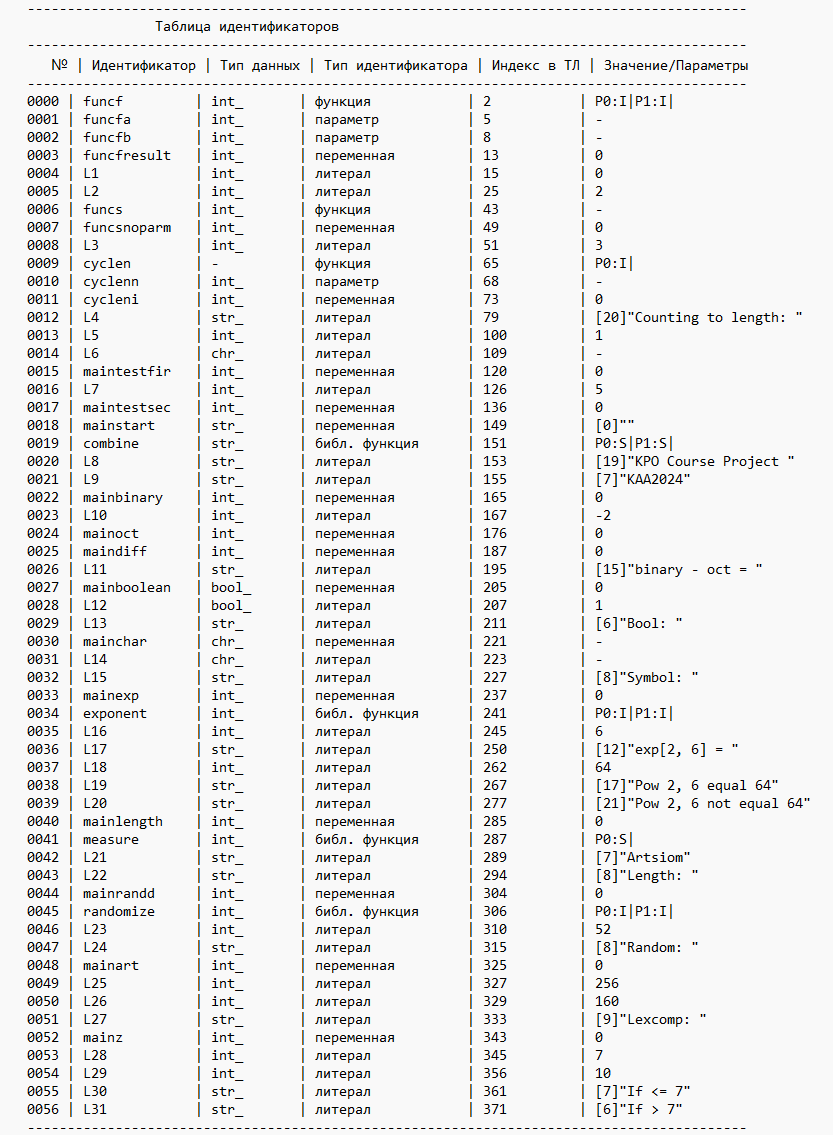


Рисунок Б.1 – Таблица идентификаторов для контрольного примера

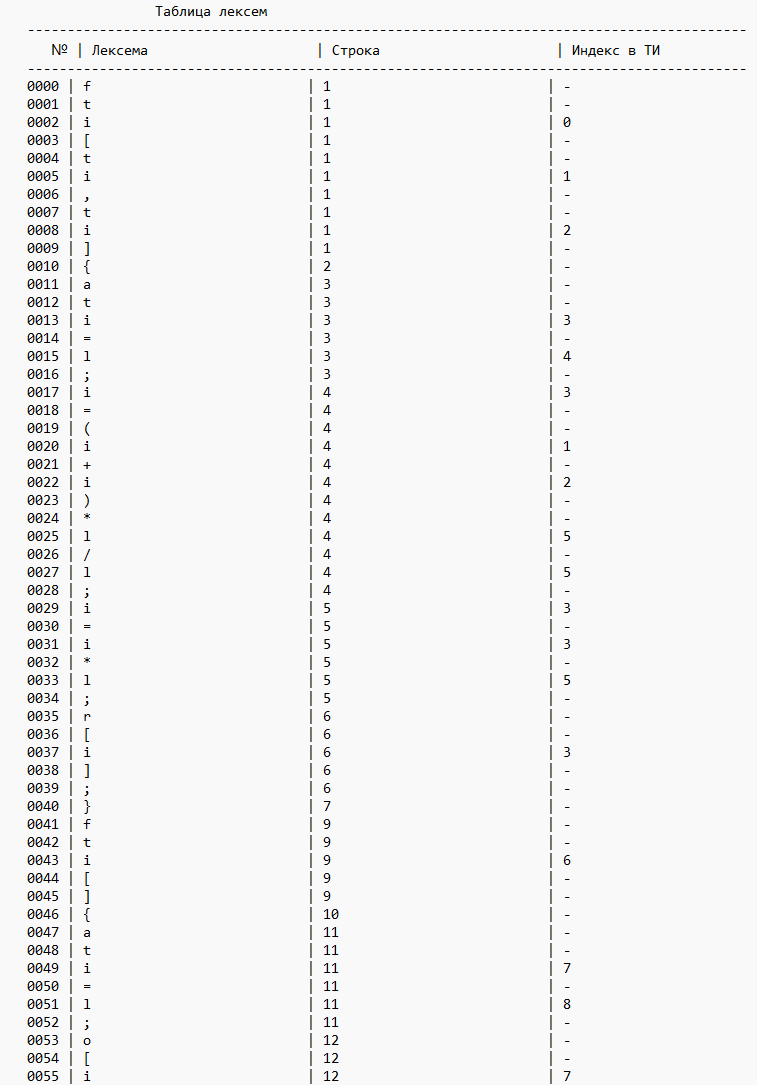


Рисунок Б. 2 – Начало таблицы лексем для контрольного примера

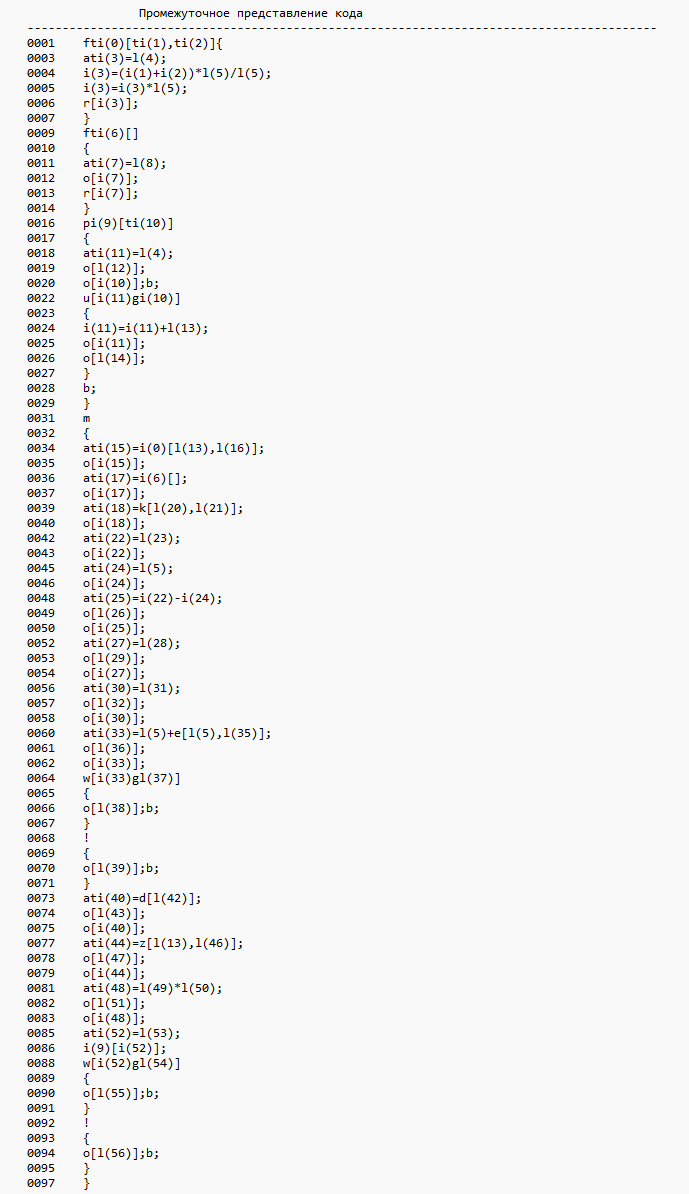


Рисунок Б.3 – Промежуточное представление кода на языке KAA-2024

**Приложение В**

**Изображение выглядит как снимок экрана, текст

AI-generated content may be incorrect.**

Рисунок В.1 – Начало файла с регулярными выражениями для языка KAA-2024

Изображение выглядит как снимок экрана, Красочность, искусство

AI-generated content may be incorrect.

Рисунок В.2 – Продолжение файла с регулярными выражениями для языка KAA-2024

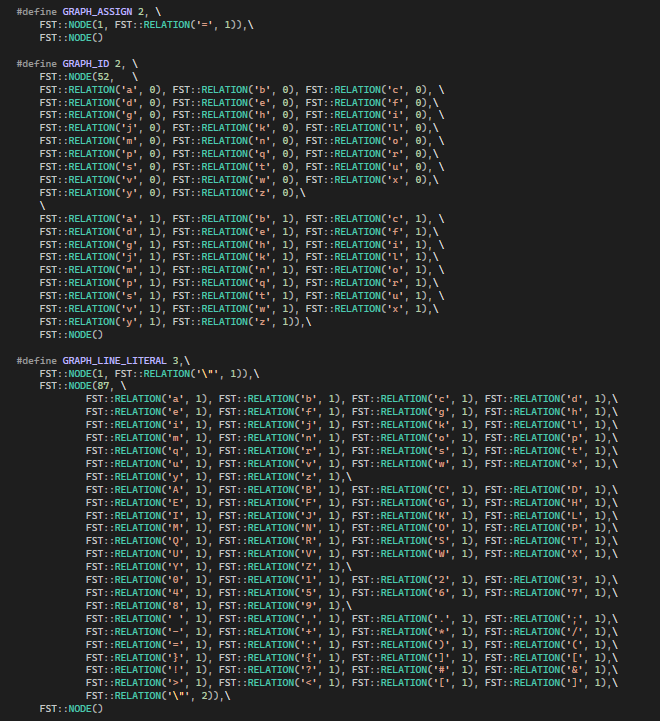


Рисунок В.3 – Продолжение файла с регулярными выражениями для языка KAA-2024

Изображение выглядит как снимок экрана, текст, дизайн

AI-generated content may be incorrect.

Рисунок В.4 – Продолжение файла с регулярными выражениями для языка KAA-2024

Изображение выглядит как текст, снимок экрана

AI-generated content may be incorrect.

Рисунок В.5 – Продолжение файла с регулярными выражениями для языка KAA-2024

Изображение выглядит как снимок экрана, текст, шаблон, дизайн

AI-generated content may be incorrect.

Рисунок В.6 – Продолжение файла с регулярными выражениями для языка KAA-2024



Рисунок В.7 – Конец файла с регулярными выражениями для языка KAA-2024

# **Приложение Г**

Изображение выглядит как текст, снимок экрана, программное обеспечение

AI-generated content may be incorrect.

Рисунок Г.1 – Начало файла с правилами для языка KAA-2024

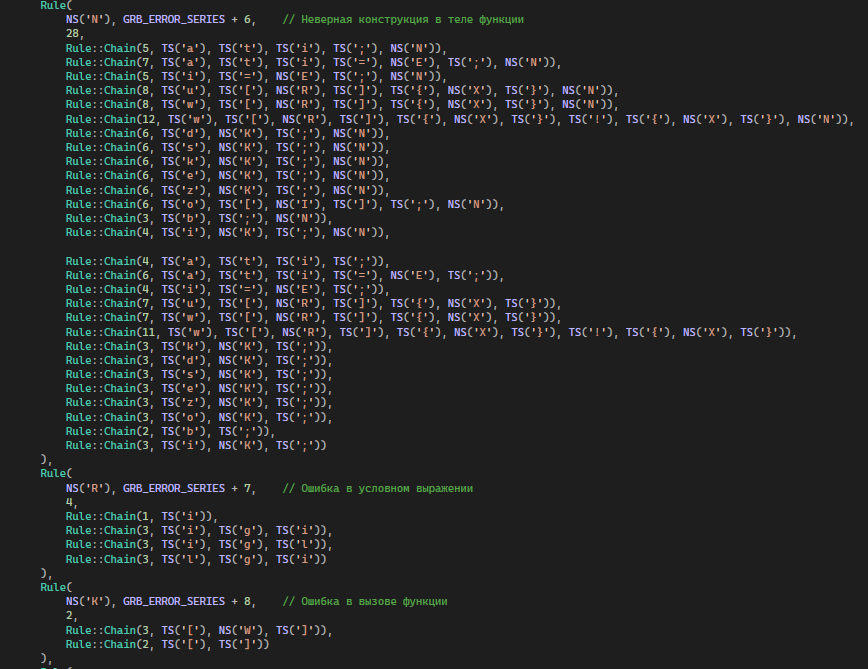


Рисунок Г.2 – Продолжение файла с правилами для языка KAA-2024

Изображение выглядит как текст, снимок экрана

AI-generated content may be incorrect.

Рисунок Г.3 – Конец файла с правилами для языка KAA-2024

# **Приложение Д**

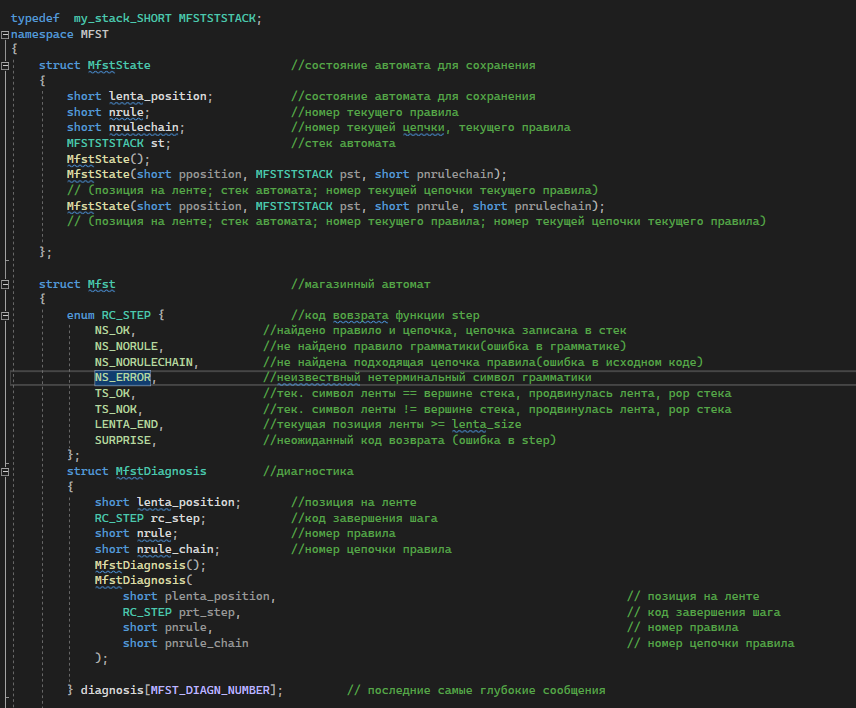


Рисунок Д.1 – Начало заголовочного файла с описанием конечного автомата с магазинной памятью

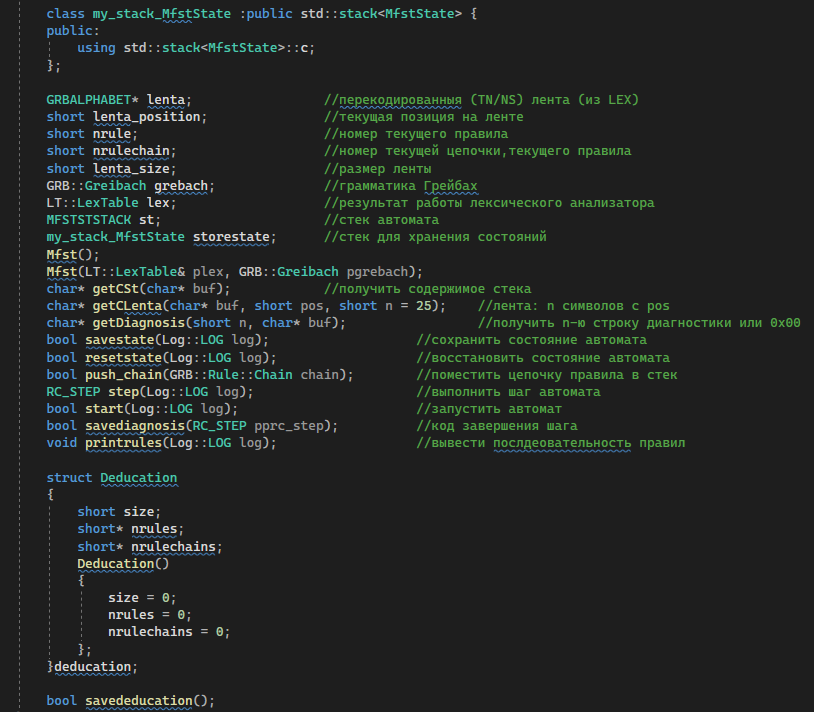


Рисунок Д.2 – Конец заголовочного файла с описанием конечного автомата с магазинной памятью

# **Приложение Е**

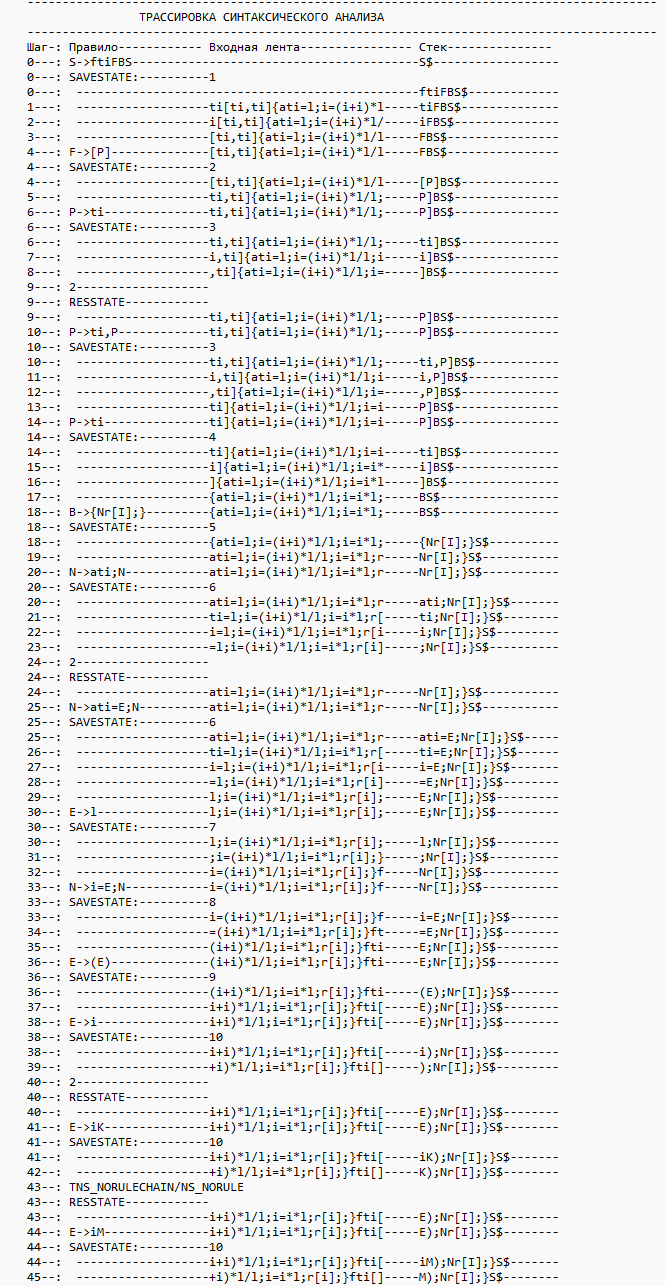


Рисунок Е.1 – Начало трассировки синтаксического анализа для контрольного примера на языке KAA-2024

Изображение выглядит как текст, снимок экрана

AI-generated content may be incorrect.

Рисунок Е.2 – Начало дерева разбора после трассировки синтаксического анализа для контрольного примера на языке KAA-2024

# **Приложение Ж**

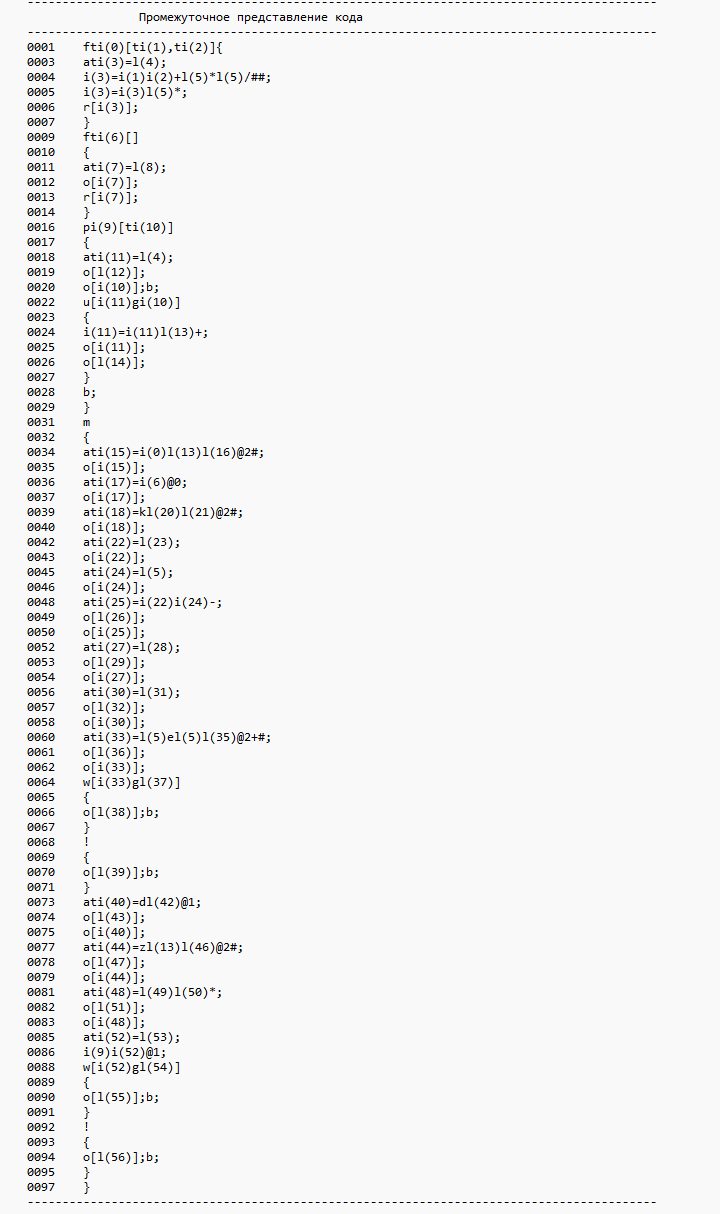


Рисунок Ж.1 – Промежуточное представление кода на языке KAA-2024 после польской нотации

# **Приложение И**

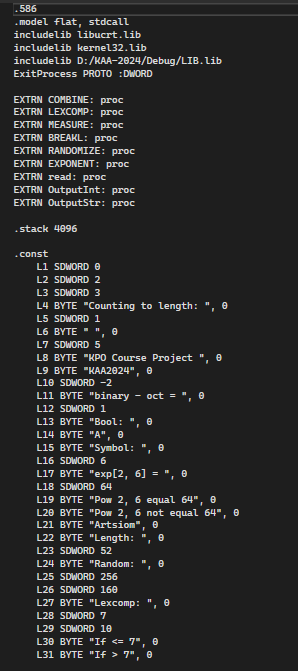


Рисунок И.1 – Начало сгенерированного кода на языке ассемблера

Изображение выглядит как текст, снимок экрана, дизайн

AI-generated content may be incorrect.

Рисунок И.2 – Продолжение сгенерированного кода на языке ассемблера

Изображение выглядит как текст, снимок экрана, Шрифт

AI-generated content may be incorrect.

Рисунок И.3 – Продолжение сгенерированного кода на языке ассемблера

Изображение выглядит как текст, снимок экрана, дизайн

AI-generated content may be incorrect.

Рисунок И.4 – Продолжение сгенерированного кода на языке ассемблера

Изображение выглядит как снимок экрана, текст, дизайн

AI-generated content may be incorrect.

Рисунок И.5 – Продолжение сгенерированного кода на языке ассемблера

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

AI-generated content may be incorrect.

Рисунок И.6 – Конец сгенерированного кода на языке ассемблера