МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Информационные системы и технологии

Специальность 1-40 01 01 «Программное обеспечение информационных технологий»

Специализация Программирование интернет-приложений

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Разработка транслятора KEY-2016»

Выполнил студент Курлович Евгений Юрьевич

(Ф.И.О.)

Руководитель проекта к.т.н., доц. Смелов В.В.

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

Заведующий кафедрой к.т.н., доц. Смелов В.В.

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

Консультанты к.т.н., доц. Смелов В.В.

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

Нормоконтролер к.т.н., доц. Смелов В.В.

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

Курсовой проект защищен с оценкой

Минск 2016

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра Информационные системы и технологии

«Утверждаю»

Заведующий кафедрой

(В. В. Смелов)

(подпись) (инициалы и фамилия)

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2016 г.

**ЗАДАНИЕ**

**НА КУРСОВОЙ ПРОЕКТ**

Студенту Курловичу Евгению Юрьевичу

(фамилия, имя, отчество)

**1. Тема проекта** Разработка транслятора KEY-2016

утверждена приказом по университету от «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2016 г. №

**2. Срок сдачи студентом законченного проекта:** \_\_ декабря 2016 г.

**3. Исходные данные к проекту:**

Разработка программы осуществляется на языке C++ (стандартизации International Standard ISO/IEC 14882:2014(E) Programming Language C++ 14) в среде разработки Visual Studio 2015 update 2. Операционная система под которой происходит разработка: Windows 10 (64-bit). Типы данных: number, word. Предусмотрены программные конструкции. Функции стандартной библиотеки: возведение числа в квадрат, извлечение квадратного корня из числа. Операции: арифметические (с выставлением приоритетов). Оператор вывода: out.

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

1) Спецификация языка программирования

2) Структура транслятора

3) Разработка лексического анализатора

4) Разработка синтаксического анализатора

5) Разработка семантического анализатора

6) Генерация кода

7) Разработка и тестирование транслятора

8) Преобразование выражений

8) Приложение

9) Литература

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

Дерево разбора синтаксического анализатора

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

|  |  |
| --- | --- |
| Раздел | Консультант |
| Разработка спецификации языка программирования и структуры транслятора | Смелов В.В |
| Разработка лексического анализатора | Смелов В.В |
| Разработка синтаксического и семантического анализатора. | Смелов В.В |
| Генерация кода. Разработка тестовых примеров. | Смелов В.В |
| Оформление пояснительной записки к курсовому проект. | Смелов В.В |

**7. Календарный план**

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование этапов курсового проекта | Срок выполнения этапов проекта | Примечание |
| 1 | Спецификация специализированного языка KEY-2016 |  |  |
| 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. Дата выдачи задания 03.09.2016**

**Руководитель**

(подпись)

**Подпись студента** Курлович Е.Ю.

Содержание

[Введение 7](#_Toc469859821)

[Глава 1. Спецификация языка программирования KEY-2016 8](#_Toc469859822)

[1.1. Характеристика языка программирования 8](#_Toc469859823)

[1.2. Алфавит языка 8](#_Toc469859824)

[1.3. Применяемые сепараторы 9](#_Toc469859825)

[1.4. Применяемые кодировки 9](#_Toc469859826)

[1.5. Типы данных 9](#_Toc469859827)

[1.6. Преобразование типов данных: 9](#_Toc469859828)

[1.7. Идентификаторы 9](#_Toc469859829)

[1.8. Литералы 9](#_Toc469859830)

[1.9. Объявление данных и область видимости 10](#_Toc469859831)

[1.10. Инициализация данных 10](#_Toc469859832)

[1.11. Инструкции языка 10](#_Toc469859833)

[1.12.Операции языка 11](#_Toc469859834)

[1.13. Выражения и их вычисления 11](#_Toc469859835)

[1.14. Программные конструкции языка 11](#_Toc469859836)

[1.15. Область видимости идентификаторов 12](#_Toc469859837)

[1.16. Семантические проверки 12](#_Toc469859838)

[1.17. Стандартная библиотека и её состав 12](#_Toc469859839)

[1.18. Вывод данных 13](#_Toc469859840)

[1.19. Точка входа 13](#_Toc469859841)

[1.20. Препроцессор 13](#_Toc469859842)

[1.21. Объектный код 13](#_Toc469859843)

[1.22. Контрольный пример 13](#_Toc469859844)

[Глава 2. Структура транслятора 14](#_Toc469859845)

[2.1. Компоненты транслятора их назначение и принципы взаимодействия 14](#_Toc469859846)

[2.2. Перечень входных параметров транслятора 15](#_Toc469859847)

[2.3. Перечень протоколов, формируемых транслятором и их содержимое 16](#_Toc469859848)

[Глава 3. Разработка лексического анализатора 17](#_Toc469859849)

[3.1. Структура лексического анализатора. 17](#_Toc469859850)

[3.2. Контроль входных символов 17](#_Toc469859851)

[3.3. Удаление избыточных символов 17](#_Toc469859852)

[3.4. Перечень ключевых слов, сепараторов, символов операций, соответствующих им лексем и конечных автоматов 18](#_Toc469859853)

[3.5. Основные структуры данных 19](#_Toc469859854)

[3.6. Принцип обработки ошибок и их перечень 19](#_Toc469859855)

[3.7. Алгоритм лексического анализатора 19](#_Toc469859856)

[3.8. Контрольный пример 19](#_Toc469859857)

[Глава 4. Разработка синтаксического анализатора 20](#_Toc469859858)

[4.1. Структура синтаксического анализатора 20](#_Toc469859859)

[4.2. Грамматика языка 20](#_Toc469859860)

[4.3 Построение конечного магазинного автомата 21](#_Toc469859861)

[4.4. Основные структуры данных 21](#_Toc469859863)

[4.5 Описание алгоритма синтаксического разбора 22](#_Toc469859864)

[4.6. Принцип обработки ошибок 22](#_Toc469859865)

[4.7. Контрольный пример 22](#_Toc469859866)

[Глава 5 Разработка семантического анализатора 23](#_Toc469859867)

[5.1. Структура семантического анализатора 23](#_Toc469859868)

[5.2. Функции семантического анализатора 23](#_Toc469859869)

[5.3. Принцип обработки ошибок 23](#_Toc469859870)

[5.4. Контрольный пример 23](#_Toc469859871)

[Глава 6. Разработка транслятора кода 24](#_Toc469859872)

[Глава 7 Преобразование выражений 26](#_Toc469859873)

[7.1 Выражения, допускаемые языком 26](#_Toc469859874)

[7.2 Польская запись и принцип её построения 26](#_Toc469859875)

[7.3. Примеры преобразования выражений 26](#_Toc469859876)

[Глава 8 Тестирование транслятора 28](#_Toc469859877)

[Заключение 29](#_Toc469859878)

[Список используемой литературы 30](#_Toc469859879)

[Приложение 1: Контрольный пример 31](#_Toc469859880)

[Приложение 2: Таблица входных символов 32](#_Toc469859881)

[Приложение 3: Графы переходов 33](#_Toc469859882)

[Приложение 4: Структуры таблицы лексем и таблицы идентификаторов 35](#_Toc469859883)

[Приложение 5: Семантические проверки 39](#_Toc469859884)

[Приложение 6: Грамматика 40](#_Toc469859885)

[Приложение 7: Таблица ошибок 41](#_Toc469859886)

[Приложение 8 Магазинный автомат 42](#_Toc469859887)

[Приложение 9 Результат работы синтаксического анализатора 43](#_Toc469859888)

# Введение

Перед нами поставлена задача разработки языка программирования. Разрабатываемое приложение состоит из нескольких подпограмм. Каждая подпрограмма выполняется последовательно. Название разрабатываемого языка программирования KEY-2016. Для данного языка была разработан свой синтаксис, а также семантика. Запуск приложения должен осуществляться посредством командной строки с указанием параметров запуска. Основной целью курсового проекта, является усвоение устройства языков программирования и принцип работы компиляторов. Написание транслятора будет осуществляться на языке C++. Для выполнения данной задачи были поставлены следующие задачи:

1. Разработка спецификации языка программирования

2. Разработка структуры транслятора

3. Разработка лексического анализатора

4. Разработка синтаксического анализатора

5. Разработка семантического анализатора

6. Разработка транслятора кода

7. Преобразование выражений

8. Тестирование транслятора

# 

# Глава 1. Спецификация языка программирования KEY-2016

## 1.1. Характеристика языка программирования

Язык программирования KEY-2016 является универсальным, строго типизированным, не объектно-ориентированным, интерпретируемым.

## 1.2. Алфавит языка

Алфавит KEY-2016 основан на кодировке Windows-1251 представленной на рис.1.



рис.1 Алфавит входных символов

Символы, используемые на этапе выполнения: [a…z], [A…Z], [0…9]. А также спецсимволы: { } ( ) ; \* + - / = , ’ [ ] пробел. Данные символы находятся в первой половине Windows-1251.

## 1.3. Применяемые сепараторы

В языке KEY-2016 используются сепараторы, представленные в табл. 1*.* Пробел допускается везде кроме идентификаторов и ключевых слов.

табл. 1. Сепараторы, применяемые в языке KEY-2016

|  |  |
| --- | --- |
| Сепаратор | Описание |
| ; - точка с запятой | Разделитель конструкций |
| {} - фигурные скобки | Программный блок |
| () - круглые скобки. | Приоритетность операций |
| [] – квадратные скобки | Программный блок |
| , - запятая | Разделитель параметров функции |

## 1.4. Применяемые кодировки

Применяется кодировка Windows - 1251. Которая предоставлена на рис.1.

## 1.5. Типы данных

В языке KEY-2016 используется 2 типа данных:

1) number– целые числа. Инициализация по умолчанию 0. Минимальное допустимое значение number = - 65536 , a максимальное значение number = 65536. В случае выхода за рамки дозволенных значений, значение будет урезаться до минимального или максимального соответственно.

2) word– строка. Инициализация по умолчанию пустой строкой. Максимальная допустимая длина строки = 255 символов.

## 1.6. Преобразование типов данных:

В языке KEY-2016 преобразование не поддерживается.

## 1.7. Идентификаторы

В языке KEY-2016 идентификаторы должны быть составлены только из символов нижнего регистра английского алфавита. Типы идентификаторов: имя переменной или функции, литерал, параметр, имя стандартной функции. Идентификатор составляется из букв английского алфавита от 1 до 10 символов, без пробелов. Максимальная длина идентификатора равна 10 символам, в противном случае урезается до указанной длины. Идентификатор не может совпадать с ключевыми словами. Максимальное количество идентификаторов 100.

## 1.8. Литералы

В языке KEY-2016 существует 2 типа литералов: литералы целого типа и строковые. Представлены в табл. 2

табл. 2 Литералы

|  |  |
| --- | --- |
| Тип | Описание |
| Литерал целого типа | Числовое значение. Может иметь знак (положительное или отрицательное). Может быть только rvalue. |
| Строковый литерал | Символы, заключённые в ' ' (одинарные кавычки). Могут быть только rvalue. Могут состоять только из символов верхнего и нижнего регистра английского алфавита. |

## 1.9. Объявление данных и область видимости

В языке KEY-2016 объявление данных начинается с ключевого слова create, указывается тип данных и имя идентификатора.

Пример: create number a, create word b;

Область видимости: сверху вниз, параметры внутри функции, объявления внутри функции видны только внутри функции, объявления переменных за пределами функций и главной функции не предусмотрены.

## 1.10. Инициализация данных

Инициализация переменной происходит после её объявления. Инициализация переменной в момент объявления запрещена.

Например: create word a ; a = ‘слово’; create number b; b = 5;

## 1.11. Инструкции языка

В языке KEY-2016 предусмотрены следующие инструкции. Представлены в табл. 3.

табл. 3 Инструкции языка KEY-2016

|  |  |
| --- | --- |
| Инициализация переменной | Имя переменной = значение; |
| Создание внешней функции | function идентификатор (тип данных идентификатор, …)  Область видимости сверху вниз (по принципу С++). Все переменные являются локальными. Предусмотрены только функции типа дынных number. |
| Главная функция | start ()  [  …  end;  ] |
| Возврат из подпрограммы | return идентификатор или литерал; |
| Вывод данных | out(идентификатор); |

## 1.12.Операции языка

В языке KEY-2016 предусмотрены следующие операции с данными. Приоритетность операций определяется с помощью (). Операции представлены в табл. 4.

табл. 4 Операции языка KEY-2016

|  |  |
| --- | --- |
| Операция | Описание |
| + | бинарный, арифметическое суммирование, (number + number); |
| - | бинарный, арифметическое вычитание, (number - number); |
| \* | бинарный, арифметическое умножение, (number \* number); |
| / | бинарный, арифметическое целочисленное деление, (number / number); |

## 1.13. Выражения и их вычисления

В языке KEY-2016 выражения записываются в строчку. Круглые скобки используются для изменения приоритета.

## 1.14. Программные конструкции языка

В языке программирования KEY-2016 предусмотрена одна главная функция и внешние функции. Программные конструкции представлены в табл. 5.

табл.5 Программные конструкции KEY-2016

|  |  |
| --- | --- |
| Внешняя функция | function идентификатор (тип данных идентификатор, … )  {  …  return идентификатор / литерал;  }  Область видимости сверху вниз (по принципу С++). Все переменные являются локальными. |
| Главная функция | start  [  …  end;  ]  Область видимости сверху вниз. Все переменные являются локальными. |

## 1.15. Область видимости идентификаторов

Сверху вниз, параметры внутри функции, объявления внутри функции видны только внутри функции, объявления за пределами функций и главной функции запрещены.

## 1.16. Семантические проверки

В языке программирования KEY-2016 выполняются семантические проверки. Список ошибок представлен в Приложении 6.

Семантически проверки языка KEY-2016:

1. Наличие главной функции start;
2. Проверка на попытку переопределения;
3. Проверка на несоответствие типов;
4. Проверка на наличие второй главной функции start;
5. Проверка параметров функций стандартной библиотеки;
6. Использование стандартных функций без подключения библиотеки;
7. Проверка на соответствие параметров в функции;
8. Проверка литералов

## 1.17. Стандартная библиотека и её состав

В KEY-2016 присутствует стандартная библиотека MATHEMATIC. Подключение стандартной библиотеки обязательно перед использованием функций. Пример: math; Библиотека представлена в табл. 6.

табл. 6 Стандартная библиотека

|  |  |
| --- | --- |
| Функция | Описание |
| step(идентификатор или литерал); | Возводит идентификатор в квадрат. Применима только для идентификаторов типа переменная, тип данных number, числовых литералов. |
| koren (идентификатор или литерал); | Получение квадратного корня. Применима только для идентификаторов типа переменная, тип данных number, числовых литералов. |

## 1.18. Вывод данных

В языке KEY-2016 вывод происходит с помощью оператора out.

Например: out(идентификатор), out(‘текст’);

## 1.19. Точка входа

В языке KEY-2016 точкой входа является функция start. Наличие данной функции обязательно.

## 1.20. Препроцессор

Директивы препроцессора не предусмотрены.

## 1.21. Объектный код

Язык программирования KEY-2016 транслируется в JavaScript.

## 1.22. Контрольный пример

Контрольный пример для тестирования приложения представлен в приложении 1: Контрольный пример.

# 

# Глава 2. Структура транслятора

## 2.1. Компоненты транслятора их назначение и принципы взаимодействия

Транслятор языка KEY-2016 разделён на отдельные части, которые взаимодействуют между собой и выполняют отведённые им функции. Алгоритм выполнения и описание каждой части приложения представлено в табл. 7.

табл. 7 Структура транслятора

|  |  |
| --- | --- |
| Наименование подпрограммы | Предназначение |
| Лексический анализатор | Обрабатывает входной файл исходного кода, проверяя его на разрешённые, запрещённые и игнорируемые символы. Преобразуя исходный код в более простой, с помощью замены длинных слов на лексемы, состоящие из одного символа, что упрощает последующую работу с кодом. Каждая лексема несёт в себе многочисленную информацию: имена идентификаторов, тип данных, тип переменной и так далее. После выполнения данной подпрограммы на выходе мы получаем таблицу лексем и таблицу идентификаторов. |
| Синтаксический анализатор | С помощью синтаксического анализатора проверяется правильность написанных конструкций по заданной грамматике. |
| Семантический анализатор | Реализован как отдельная часть. Нужен для того, чтобы проверить все правила, которые невозможно проверить на этапах лексического и синтаксического анализаторов |
| Транслятор кода | Связан с работой лексического анализатора. Посредством полученных данных: таблицы лексем и идентификаторов, код на языке KEY-2016 транслируется в код на языке JavaScript. |

Каждый этап программы имеет взаимосвязь, которую мы можем наблюдать на рис. 2.



рис. 2 Структура транслятора

## 2.2. Перечень входных параметров транслятора

Для указания запуска параметров запуска транслятора следует указывать параметры командной строки, которые представлены и описаны в табл. 8

табл. 8 Параметры транслятора

|  |  |  |
| --- | --- | --- |
| Параметр | Правило  указания | Описание параметра |
| -in: | -in:[полный путь к файлу] | Предназначен для определения местонахождения файла с исходным кодом, для проверки и разбора его на лексическом анализаторе.  Указание данного параметра обязательно. |
| -log: | -log:[полный путь к файлу] | Устанавливается местонахождение и имя файла, содержащего информацию о ходе работы транслятора.  Указание данного параметра необязательно. |

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

Содержание файла c перечнем протоколов, формируемых транслятором представлено в табл. 9

табл.9 Протоколы транслятора

|  |  |
| --- | --- |
| **Тип информации** | **Описание**  **информации** |
| Дата и время трансляции | Выводит дату и время. |
| Параметры командой строки | Выводит информацию об указанных параметрах командной строки. |
| Полная таблица лексем | Выводит таблицу лексем с информацией к каждой лексеме. |
| Упрощённая таблица лексем | Выводит только лексемы без дополнительной информации. |
| Таблица идентификаторов | Выводит таблицу идентификаторов с дополнительной информацией. |
| Трассировочная информация синтаксического анализа | Выводит полную информацию о разборе таблицы лексем на синтаксическом анализаторе. |
| Правила разбора | Выводит правила, по которым осуществился разбор исходного кода. |
| Преобразованная упрощённая таблица лексем | Выводит упрощённую таблицу лексем, в которой преобразованы математические выражения. |

# Глава 3. Разработка лексического анализатора

## 3.1. Структура лексического анализатора.

Схема работы лексического анализатора представлена на рис. 3.

Лексический

анализатор

рис. 3 Схема работы лексического анализатора.

Входными данными для лексического анализатора является текст программы на языке KEY-2016. В результате работы лексического анализатора получим таблицу лексем и таблицу идентификаторов.

## 3.2. Контроль входных символов

Таблица входных символов представлена на рис. 1. Реализация таблицы на языке C++ представлена в приложении 2: Таблица входных символов*.* Обозначения символов в таблице представлено в табл. 10.

табл. 10 Контроль входных символов

|  |  |
| --- | --- |
| **Символы** | **Значение в таблице символов** |
| () \* + - = , / \ % | SEP |
| Символ новой строки | NS |
| Запрещённый | F |
| Разрешённый | T |
| Игнорируемый | I |
| Пробел | PR |

## 3.3. Удаление избыточных символов

При считывании из файла с исходным кодом в случае если считанный символ оказался пробелом, тогда мы проверяем предыдущий символ, если предыдущий символ являлся пробелом, значит текущий пробел мы просто игнорируем и не заносим в массив.

В результате работы данного алгоритма между соседними словами будет максимум 1 пробел.

## 3.4. Перечень ключевых слов, сепараторов, символов операций, соответствующих им лексем и конечных автоматов

Ключевые слова KEY-2016: create, number, word, function, return, math, end, koren, step, out, start. Сепараторы KEY-2016 представлены в табл. 1. Графы переходов представлены в Приложении 3: Графы переходов.

Для каждой фразы также соответствует автомат, по которому происходит разбор выражения. Автомат является – детерминированным, то есть имеет конечное состояние. Проверка происходит следующим образом: на каждый автомат в массиве подаётся фраза и с помощью графа переходов происходит разбор, если разбор выполнен, то происходит заполнение таблицы лексем и при необходимости таблицы идентификаторов.

Каждой фразе соответствует отдельный символ, называемый лексемой. Соответствие фраз с такими лексемами представлен в табл. 11.

табл. 11 Лексемы

|  |  |  |
| --- | --- | --- |
| **Фрагмент** | **Лексема** | **Примечание** |
| number  word | n | Тип данных |
| идентификатор | i | (переменная, функция или параметр) |
| литералы | l | данные |
| function | f | функция |
| return | r | Возврат значения из функции |
| start | s | Главная функция |
| out | o | Вывод |
| koren | k | Вычисление квадратного корня |
| step | b | Возведение в квадрат |
| ; | ; | Сепаратор |
| { | { | Начало тела внешней функции |
| } | } | Конец тела внешней функции |
| ( | ( | Начало перечисления параметров функции |
| ) | ) | Конец перечисления параметров функции |
| = | = | Оператор присваивания |
| +  -  \*  / | +  -  \*  / | Арифметические операторы |
| , | , | Сепаратор параметров |
| [ | [ | Начало тела главной функции |
| ] | ] | Конец тела главной функции |
| end | e | Завершение программы |
| create | c | Создание переменной |

## 3.5. Основные структуры данных

Код С++ с описанием структур таблиц лексем и идентификаторов, приведён в приложении 4: Структуры таблицы лексем и таблицы идентификаторов.

## 3.6. Принцип обработки ошибок и их перечень

При обнаружении ошибки во время работы транслятора, вызывается функция получения ошибки, в которую передается, в зависимости от места возникновения ошибки, следующая информация: код ошибки, номер строки в коде, номер позиции в строке или только код ошибки.

## 3.7. Алгоритм лексического анализатора

После разбиения текста из файла с исходным кодом на слова, для каждого слова подбирается автомат способный его разобрать, в случае если такой автомат существует, тогда цепочка будет разобрана, иначе ошибка. Далее лексический анализатор анализирует лексему, соответствующую данному слову и выполняет действия описанные для данной лексемы. Лексический анализатор продолжает работать пока не будет разобрано последнее слово.

## 3.8. Контрольный пример

В результате работы лексического анализатора мы получим таблицу лексем, таблицу идентификаторов и информацию об ошибках. Текст таблицы лексем и таблицы идентификаторов представлен в Приложении 4: Структуры таблицы лексем и таблицы идентификаторов.

# 

# 

# Глава 4. Разработка синтаксического анализатора

## 4.1. Структура синтаксического анализатора

Структура работы синтаксического анализатора и входные-выходные данные представлены на рис. 4.

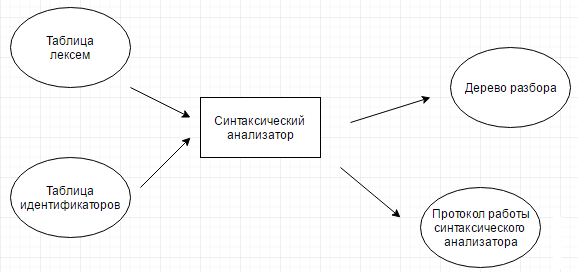


рис.4 Синтаксический анализатор

## 4.2. Грамматика языка

Грамматика для синтаксического разбора языка KEY-2016 представлена

в таблице 12. Для описания языка, разбираемого синтаксического анализатором, применяют грамматики типа 2 – контекстно-свободные грамматики.

Грамматика типа 2:  - контекстно-свободная грамматика, где T – множество терминалов, N – множество нетерминалов, P – правила перехода, S – стартовый символ. Описание терминалов представлено в табл. 11.

Правила имеют вид: , где , ,  - словарь грамматики .

табл. 12 Правила перехода нетерминальных символов

|  |  |
| --- | --- |
| Нетерминал | Правила |
| S – структура программы | fi(F){N}S | s[N] | m;S |
| N – конструкции внутри функции | cni;N | i=E;N | oE;N | i=E;N | m;N| i=E; | rE; | e; |
| F – параметры функции | ni | ni,F |
| E - выражения | i | l | | (E) | iM |i(W) | k(i) | k(l) | b(i) | b(l) | iM | (E)M |
| W – параметры функции при вызове | i,W | l,W | l | i |
| M - знаки | +E | +EM |-E |-EM | \*E | \*EM | /E | /EM |

## 4.3 Построение конечного магазинного автомата

Конечный автомат с магазинной памятью можно представить в виде семёрки , где М – автомат, Q – множество состояний, V – алфавит входных символов, Z – алфавит магазина,  - функция переходов,  – начальное состояние автомата,  – начальное состояние магазина, F – множество конечных состояний. Автомат с магазинной памятью представлен на рис. 5



рис. 5 Автомат с магазинной памятью.

Принцип работы автомата следующий:

1. в магазин записывается стартовый символ;
2. на основе полученных раннее 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 с 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 вызывается ошибка.

Структура магазинного автомата представлена в виде структуры с++ (описание автомата находится в зоне комментариев) в приложении 7: Магазинный автомат.

## 4.4. Основные структуры данных

Код с++ с описанием структур данных описывающих грамматику представлен в Приложении 6: Грамматика*.*

## 4.5 Описание алгоритма синтаксического разбора

Алгоритм синтаксического разбора основан на работе автомата с магазинной памятью. Работа данного автомата описана в пункте 4.3.

## 4.6. Принцип обработки ошибок

При возникновении ошибок синтаксический анализатор пытается откатиться назад, если это возможно и повторить шаг по следующей цепочке правил, в случае невозможности вызывается специализированная ошибка для правила, до которого максимально добрался анализатор. Ошибки выдаваемы синтаксическим анализатором описаны в приложении 8: Таблица ошибок.

## 4.7. Контрольный пример

Протокол разбора и дерево разбора представлены в приложении 9: Результат работы синтаксического анализатора.

## 

# 

# Глава 5 Разработка семантического анализатора

## 5.1. Структура семантического анализатора

Проверка на ошибки в исходном коде производится как одновременно с работой синтаксического и лексического анализаторов, так и в отдельной части транслятора. Семантические правилаязыка KEY-2016 представлены в гл. 1. п. 1.16.

## 5.2. Функции семантического анализатора

Семантический анализатор выполняет проверку на основные правила языка (семантики языка), которые представлены в гл. 1. п. 1.16*.* Таблица ошибок представлена в виде фрагмента кода с++ в приложении 8: Таблица ошибок.

## 5.3. Принцип обработки ошибок

В случае возникновения ошибок, вызываем функцию получения ошибки, которая принимает обязательным параметром код ошибки в таблице сообщений. Затем производится вывод ошибки в поток лог-файла, в случае отсутствия данного потока, ошибка выводится в консоль. Структура данных с++ в которой хранится информация об ошибках представлена в приложении 8: Таблица ошибок*.*

## 5.4. Контрольный пример

Контрольные примеры, предназначенные для проверки обработки всех возможных ошибок, представлены в виде текстовых файлов с исходным кодом на языке KEY-2016, которые прилагаются на переносном носителе CD-R вместе с файлами решения под среду разработки MS Visual Studio 2015. Также несколько примеров которые демонстрируют работу обработчика ошибок представлены в приложении 5: Семантические проверки.

# Глава 6. Разработка транслятора кода

Транслятор KEY-2016 выполняет трансляцию кода на языке KEY-2016 в код на языке JavaScript на основе таблицы лексем и таблицы идентификаторов. Схематично трансляция кода показана на рис.5*.* Транслятор кода начинает свою работу только в том случае если код на языке KEY-2016 прошёл предыдущие компоненты транслятора без ошибок.



рис. 5 Структура транслятора кода

На вход подаются по очереди лексемы, и в зависимости от поданной лексемы языка KEY-2016, в текстовый файл Generation.html записывается соответствующее ей слово. После разбора последней лексемы будет получен файл с кодом JavaScript. После открытия этого файла, откроется окно браузера и произойдёт выполнение программы. В табл. 9 представлены лексемы и соответствующие им слова для записи в файл.

табл.9 Трансляция JavaScript

|  |  |
| --- | --- |
| Лексема | Слово |
| { | { |
| c | var |
| ; | ; переход на новую строку |
| = | = |
| n | Запись не производится |
| пробел | Запись не производится |
| m | document.write(“Included MATHEMATIC library)” |
| o | document.write |
| ( | ( |
| ) | ) |
| e | document.write(“Programm is End”) |
| } | } |
| . | Переход на новую строку |
| + | + |
| - | - |
| \* | \* |
| / | / |
| f | function |
| , | , |
| r | return |
| k | koren |
| b | step |

В случае если встречается лексема идентификатора, его имя и значение будет взято из таблицы идентификаторов.

# 

# Глава 7 Преобразование выражений

## 7.1 Выражения, допускаемые языком

В языке KEY-2016 допускаются вычисления выражений исключительно с целочисленными типами данных. Вычисление выражений в языке KEY-2016 происходит без преобразования в польскую запись. Приоритет операций представлен в табл. 10.

табл. 10 Приоритеты операций

|  |  |
| --- | --- |
| Приоритет | Операция |
| 1 | ( |
| 1 | ) |
| 2 | + |
| 2 | - |
| 3 | \* |
| 3 | / |

## 7.2 Польская запись и принцип её построения

Польская запись - это альтернативный способ записи арифметических выражений, преимущество которого состоит в отсутствии скобок. Существует два типа польской записи: прямая и обратная, также известные как префиксная и постфиксная. Отличие их от классического, инфиксного способа заключается в том, что знаки операций пишутся не между, а, соответственно, до или после аргументов.

Суть алгоритма заключается в следующем. Просматривается исходная строка символов S слева направо, операнды переписываются в выходную строку В, а знаки операций заносятся в стек, который первоначально пуст, на основе следующих правил:

1. если в строке S встретился операнд, то помещаем его в строку В;
2. если в S встретилась открывающая скобка, то помещаем ее в стек;
3. если в S встретилась закрывающая скобка, то выталкиваем из стека в строку В все операции до открывающей скобки, саму отрывающую скобку также извлекаем из стека; обе скобки (открывающая и закрывающая) игнорируются;
4. если в S встретилась операция Х, то выталкиваем из стека все операции, приоритет которых не ниже Х, после чего операцию Х записываем в стек;
5. при достижении конца строки S, если стек не пуст, переписываем его элементы в выходную строку В.

## 7.3. Примеры преобразования выражений

Примеры преобразования выражений к польской записи представлены в табл. 11.

табл. 11. Примеры выражений в польской записи

|  |  |
| --- | --- |
| Выражение | Выражение в польской записи |
| 2\*(3+7) | 237+\* |
| 2+3 | 23+ |
| x+x\*(3+x) | xx3x++\* |

# 

# Глава 8 Тестирование транслятора

Тестирование транслятора KEY-2016 будет выполняться посредством компиляции исходного кода в проекте Visual Studio 2015 и запуска, полученного файла с исходным кодом на языке JavaScript. Во время тестирования транслятора файл с исходным кодом KEY-2016 будет изменяться для проверки эффективности и работоспособности транслятора в различных ситуациях.

Файлы решения под среду разработки MS Visual Studio 2015, а также файлы с исходным кодом на языке KEY-2016 прилагаются на переносном носителе CD-R.

# 

# Заключение

В данном курсовом проекте были выполнены поставленные минимальные требования. В ходе работы было изучено много нового, а также закреплены знания, которые были получены ранее. Также стоит отметить что данный курсовой проект позволил совместить закрепление знаний сразу по двум языкам программирования, таких как C++ и JavaScript. При написании приложения были усвоены такие понятия как синтаксический, лексический и семантический анализатор и многие другие.

В итоге был получен примитивный язык программирования KEY-2016, который не имеет сложных конструкций, которые реализованы на сегодняшний день во многих других языках программирования. Финальная версия транслятора состоит из 2600 строк кода.

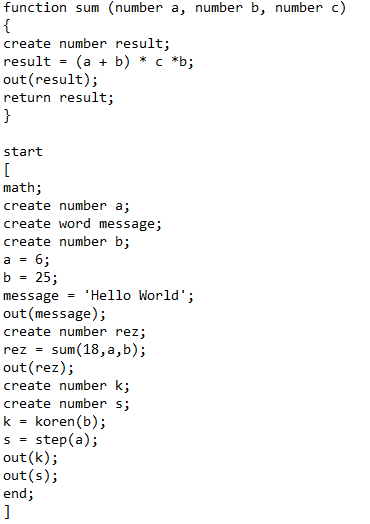
Окончательная версия языка KEY-2016 включает:

1. 2 типа данных;
2. Поддержка оператора вывода;
3. Возможность подключения и вызова функций стандартной библиотеки;
4. Наличие 4 арифметических операторов для вычисления выражений;
5. Возможность вызова функции в выражении

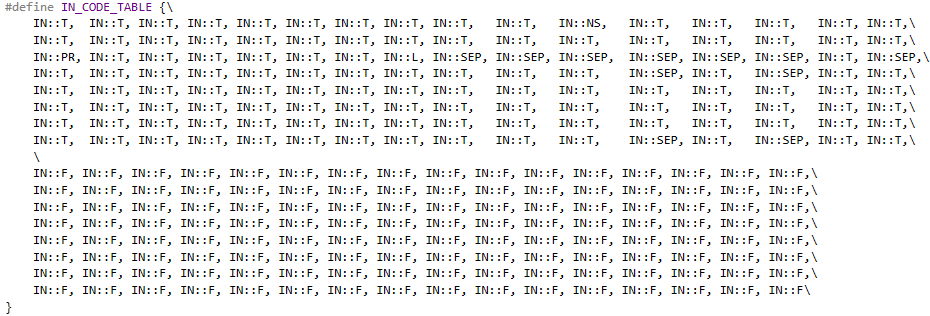
# Список используемой литературы

1. Смелов В.В. Курс лекций по предмету языки программирования – 2016
2. Р. Лафоре Программирование с++ - 2004
3. Cody Lindley Введение в JavaScript - 2008

# Приложение 1: Контрольный пример

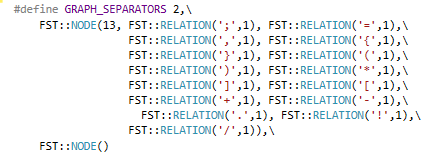
****

# Приложение 2: Таблица входных символов



# Приложение 3: Графы переходов

Граф сепараторов



S1

S0

=

S1

S0

;

S1

S0

,

S1

S0

{

S1

S0

}

S1

S0

(

S1

S0

)

S1

S0

[

S1

S0

Ъ]]

S1

S0

+

S1

S0

-

S1

S0

\*

S1

S0

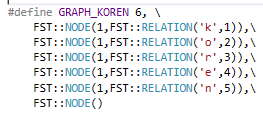
=

S1

S0

.

Граф функции стандартной библиотеки



S5

S4

S0

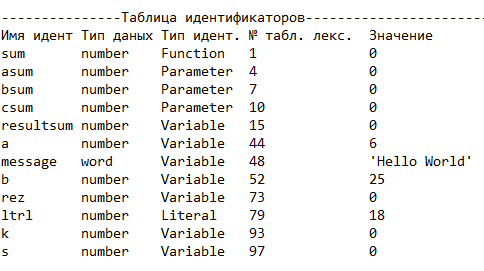
S3

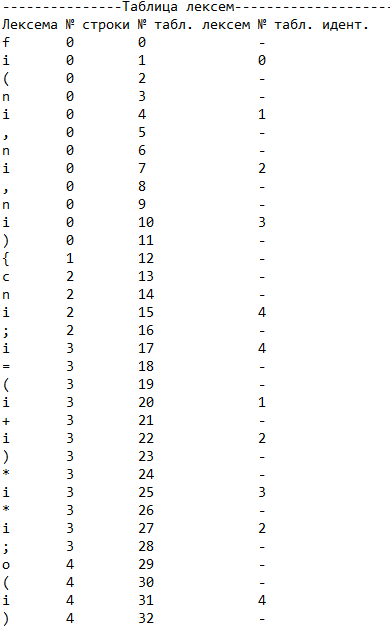
S2

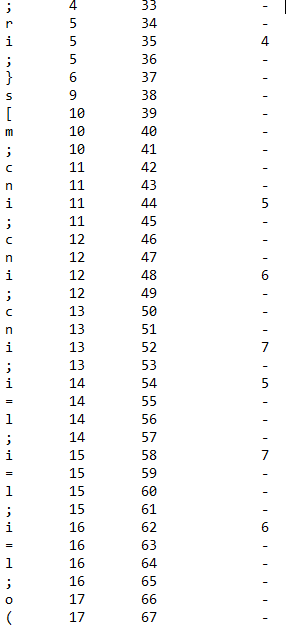
S1

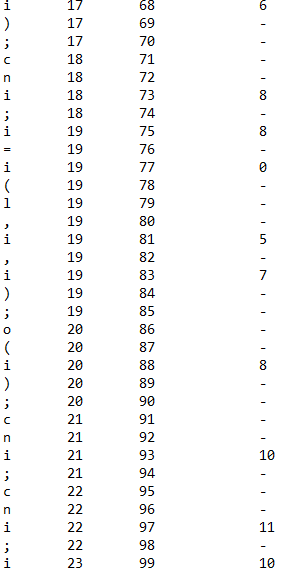
k o r e n

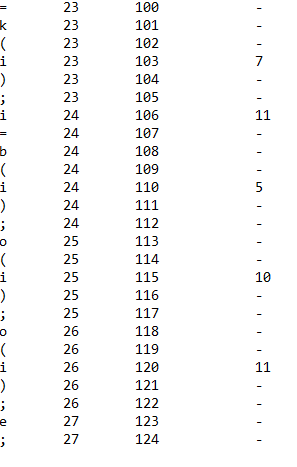
# Приложение 4: Структуры таблицы лексем и таблицы идентификаторов

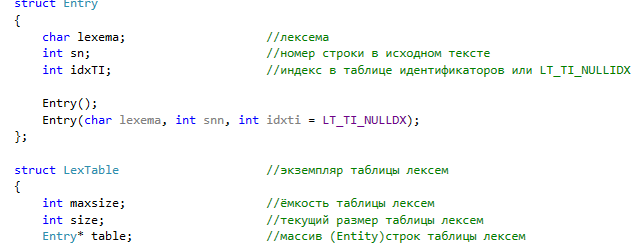


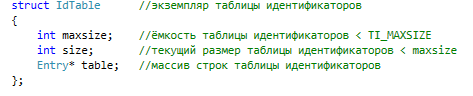


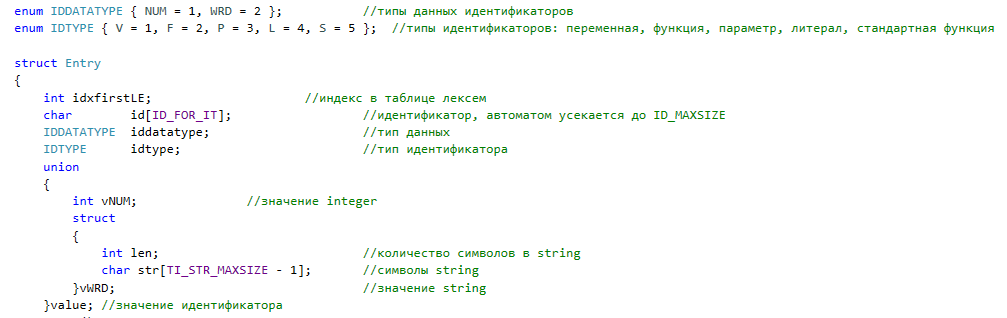






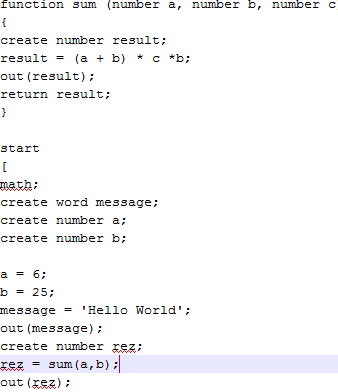






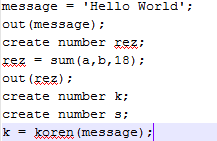
# Приложение 5: Семантические проверки

1.)

****



2.)



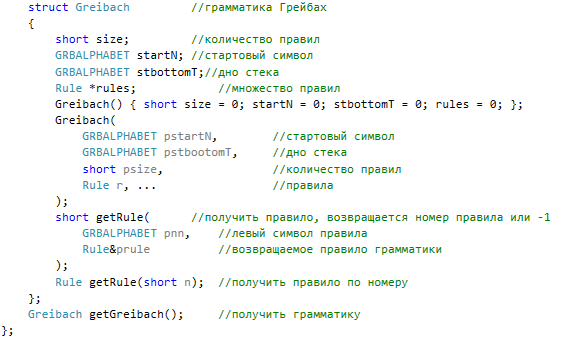
****

3.)

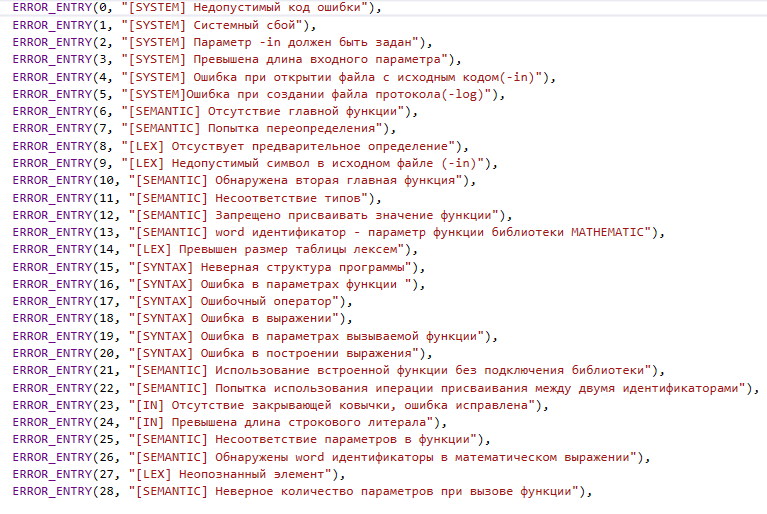


****

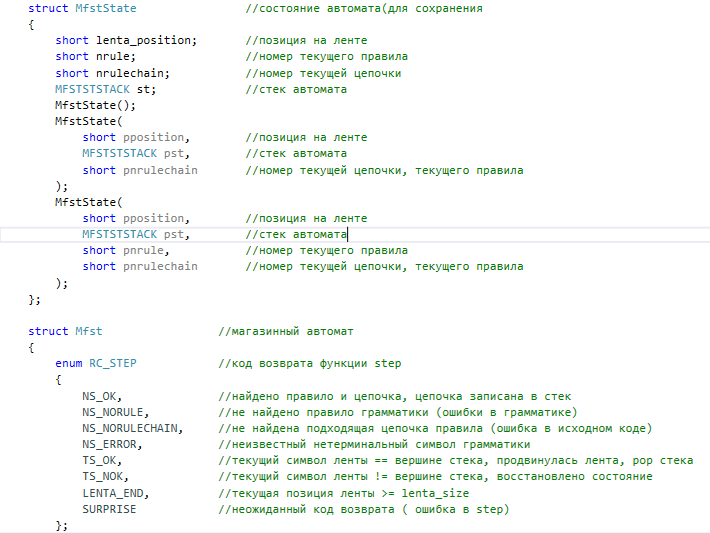
# Приложение 6: Грамматика

****

# Приложение 7: Таблица ошибок



# Приложение 8 Магазинный автомат



# Приложение 9 Результат работы синтаксического анализатора

