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 SAV-2018»

Выполнил студент Сойкель Александр Владимирович

(Ф.И.О.)

Руководитель проекта преп.ст. Рауба Алексей Александрович

(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.)

преп.ст.Рауба Алексей Александрович

(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

Содержание

[Введение 6](#_Toc501544228)

[Глава 1 Спецификация языка программирования SAV-2018 7](#_Toc501544229)

[1.1 Характеристика языка программирования 7](#_Toc501544230)

[1.2 Алфавит языка 7](#_Toc501544231)

[1.3 Применяемые сепараторы 8](#_Toc501544232)

[1.4 Применяемые кодировки 8](#_Toc501544233)

[1.5 Типы данных 8](#_Toc501544234)

[1.6 Преобразование типов данных 9](#_Toc501544235)

[1.7 Идентификаторы 9](#_Toc501544236)

[1.8 Литералы 9](#_Toc501544237)

[1.9 Объявление данных и область видимости 9](#_Toc501544238)

[1.10 Инициализация данных 9](#_Toc501544239)

[1.11 Инструкции языка 10](#_Toc501544240)

[1.12 Операции языка 10](#_Toc501544241)

[1.13 Выражения и их вычисления 11](#_Toc501544242)

[1.14 Программные конструкции языка 11](#_Toc501544243)

[1.15 Область видимости идентификаторов 11](#_Toc501544244)

[1.16 Семантические проверки 11](#_Toc501544245)

[1.17 Распределение оперативной памяти на этапе выполнения 12](#_Toc501544246)

[1.18 Стандартная библиотека и ее состав 12](#_Toc501544247)

[1.19 Ввод и вывод данных 12](#_Toc501544248)

[1.19 Точка входа 12](#_Toc501544249)

[1.20 Препроцессор 12](#_Toc501544250)

[1.21 Соглашения о вызовах 13](#_Toc501544251)

[1.22 Объектный код 13](#_Toc501544252)

[1.23 Классификация сообщений транслятора 13](#_Toc501544253)

[1.24 Контрольный пример 13](#_Toc501544254)

[Глава 2. Структура транслятора 14](#_Toc501544255)

[2.1 Компоненты транслятора их назначение и принципы взаимодействия 14](#_Toc501544256)

[2.2 Перечень входных параметров транслятора 14](#_Toc501544257)

[2.3 Перечень протоколов формируемых транслятором и их содержимое 15](#_Toc501544258)

[Глава 3. Разработка лексического анализатора 16](#_Toc501544259)

[3.1 Структура лексического анализатора 16](#_Toc501544260)

[3.2 Контроль входных символов 16](#_Toc501544261)

[3.3 Удаление избыточных символов 17](#_Toc501544262)

[3.4 Перечень ключевых слов, сепараторов, символов операций соответствующих им лексем, регулярных выражений и конечных автоматов 17](#_Toc501544263)

[3.5 Основные структуры данных 18](#_Toc501544264)

[3.6 Принцип обработки ошибок 18](#_Toc501544265)

[3.7 Структура и перечень сообщений лексического анализатора 18](#_Toc501544266)

[3.8 Параметры лексического анализатора и режимы его работы 18](#_Toc501544267)

[3.9 Алгоритм лексического анализа 19](#_Toc501544268)

[3.10 Контрольный пример 19](#_Toc501544269)

[Глава 4 Разработка синтаксического анализатора 20](#_Toc501544270)

[4.1 Структура синтаксического анализатора 20](#_Toc501544271)

[4.2 Контекстно-свободная грамматика, описывающая синтаксис языка 20](#_Toc501544272)

[4.3. Построение конечного магазинного автомата 21](#_Toc501544273)

[4.4 Основные структуры данных 21](#_Toc501544274)

[4.5 Описание алгоритма синтаксического разбора 22](#_Toc501544275)

[4.6 Структура и перечень сообщений синтаксического анализатора 22](#_Toc501544276)

[4.7 Параметры синтаксического анализатора и режимы его работы 22](#_Toc501544277)

[4.8 Принцип обработки ошибок 23](#_Toc501544278)

[4.9 Контрольный пример 23](#_Toc501544279)

[Глава 5 Разработка семантического анализатора 24](#_Toc501544280)

[5.1 Структура семантического анализатора 24](#_Toc501544281)

[5.2 Функции семантического анализатора 24](#_Toc501544282)

[5.3 Структура и перечень сообщений семантического анализатора 24](#_Toc501544283)

[5.4 Принцип обработки ошибок 24](#_Toc501544284)

[5.5 Контрольный пример 25](#_Toc501544285)

[Глава 6 Преобразование выражений 26](#_Toc501544286)

[6.1 Выражения, допускаемые языком 26](#_Toc501544287)

[6.2 Польская запись и принцип ее построения 26](#_Toc501544288)

[6.3 Программная реализация обработки выражений 27](#_Toc501544289)

[6.4 Контрольный пример 27](#_Toc501544290)

[Глава 7 Генерация кода 28](#_Toc501544291)

[7.1 Структура генератора кода 28](#_Toc501544292)

[7.2 Принцип построения объектного кода 28](#_Toc501544293)

[7.3 Статическая библиотека 29](#_Toc501544294)

[7.4 Входные параметры генератора кода 30](#_Toc501544295)

[7.5 Контрольный пример 30](#_Toc501544296)

[Глава 8. Тестирование транслятора 31](#_Toc501544297)

[8.1 Общие положения 31](#_Toc501544298)

[8.2 Тестирование проверки на допустимость символов 31](#_Toc501544299)

[8.3 Тестирование лексического анализатора 31](#_Toc501544300)

[8.4 Тестирование синтаксического анализатора 31](#_Toc501544301)

[8.5 Тестирование семантического анализатора 32](#_Toc501544302)

[Заключение 33](#_Toc501544303)

[Приложение А 34](#_Toc501544304)

[Приложение Б 35](#_Toc501544305)

[Приложение В 36](#_Toc501544306)

[Приложение Г 37](#_Toc501544307)

[Приложение Д 38](#_Toc501544308)

[Приложение Е 39](#_Toc501544309)

[Шаблоны генератора кода SAV-2018: 39](#_Toc501544310)

[Приложение Ж 41](#_Toc501544311)

[Приложение З 42](#_Toc501544312)

[Приложение И 44](#_Toc501544313)

[Список используемой литературы 46](#_Toc501544314)

# Введение

Компилятор — это программа, которая преобразует исходный текст программы, написанный на языке программирования высокого уровня, в программу на машинном языке, «понятную» компьютеру.

Классический компилятор состоит из следующих частей:

– Лексический анализатор;

– Синтаксический анализатор;

– Семантический анализатор;

– Генератор кода.

Все части компилятора взаимодействуют между собой, обрабатывая входной текст и строят из него эквивалентный текст на понятном языке программирования для компьютера.

Цель курсового проекта — разработка всех частей компилятора для собственного языка программирования. В процессе создания компилятора будут проведены многие исследования эффективности тех или иных алгоритмов, будут укрепленные и получены новые знания о структуре языков программирования, и в конечном итоге будет получен требуемый результат, а именно — рабочий проект на собственном языке программирования.

Курсовой проект разбит на 8 глав, в каждой из которых будет подробно описан процесс работы анализаторов и генератора. Также будет представлен новый язык программирования SAV-2018 и код рабочей программы, демонстрирующий все особенности языка. Также будут представлены различные коды языка с ошибочным завершением и подробным объяснением причин этого завершения.

В процессе работы будут проведены все основные этапы разработки программного обеспечения. На этапе анализа будут изучены алгоритмы построения компилятора, на этапе проектирования будет разработана грамматика языка SAV-2018, на этапе реализации будет написан рабочий код, а на этапе тестирования будут выявлены основные недочёты проделанной работы, на этапе сопровождения будет оформлена документация и исправлены все ошибки, найденные ранее.

# 1 Спецификация языка программирования SAV-2018

## 1.1 Характеристика языка программирования

SAV-2018 язык компилируемый, то есть исходный код преобразуется компилятором в машинный код и записывает в файл с особым расширением, для последующей идентификации этого файла; процедурный, то есть предоставляет возможность определять каждый шаг в процессе решения задач; универсальный и строготипизированный язык программирования. Язык SAV-2018 не является объектно-ориентированным языком.

## 1.2 Алфавит языка

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



Рисунок 1.1 – Алфавит входных символов

Исходный код SAV-2018 может содержать символы латинского алфавита большого и малого регистров, цифры десятичной системы счисления от 0 до 9, русские символы разрешены только в строковых литералах.

## 1.3 Применяемые сепараторы

Символы-сепараторы служат для разделения цепочек языка на уровне первичной обработки исходного текста программы. Символы-сепараторы приведены в таблице 1.1.

Таблица 1.1 – Символы-сепараторы

|  |  |  |
| --- | --- | --- |
| Символ-  Сепаратор | Название | Применение |
| **;** | Точка с запятой | Разделитель инструкций |
| **{}** | Квадратные скобки | Программный блок функции |
| **()** | Круглые скобки | Параметры функций, приоритетность операций |
| ‘ ‘ | Пробел | Разделитель цепочек языка. Может применяться везде кроме ключевых слов и идентификаторов |
| , | Запятая | Разделитель параметров функции |

## 1.4 Применяемые кодировки

В языке SAV-2018 применяется кодировка Windows-1251.

## 1.5 Типы данных

В языке SAV-2018 предусмотрены типы данных, приведенные в таблице 1.2.

Таблица 1.2 – Типы данных

|  |  |
| --- | --- |
| Тип | Пояснение |
| int | Целочисленный тип данных (4 байта). Автоматическая инициализация нулём.  Поддерживаемые операции:   * + (бинарный) – оператор сложения; * - (бинарный) – оператор вычитания; * \* (бинарный) – оператор произведения; * / (бинарный) – оператор деления. * = (бинарный) – оператор присваивания. |

Продолжение таблицы 1.2

|  |  |
| --- | --- |
| str | Строковый тип данных. Максимальное число символов – 255. Каждый символ занимает 1 байт. При объявлении инициализируется пустой строкой.  Поддерживаемые операции:   * = (бинарный) – оператор присваивания. |

## 1.6 Преобразование типов данных

Преобразование типов данных в языке SAV-2018 не предусматривается.

## 1.7 Идентификаторы

Идентификаторы в языке SAV-2018 состоят из символов нижнего регистра латинского алфавита. Максимальная длина идентификатора — 5 символов. При нахождении идентификатора большей длины он обрезается до 5 символов. Зарезервированные идентификаторы предусмотрены для стандартной библиотеки языка.

## 1.8 Литералы

В языке SAV-2018 предусмотрены литералы, представленные в таблице 1.3.

Таблица 1.3 – Литералы

|  |  |
| --- | --- |
| Литерал | Пояснение |
| Целочисленные литералы | Целые неотрицательные числа. Могут быть только rvalue. |
| Строковые литералы | Набор символов, заключённых в двойные кавычки (“). Могут состоять из всех разрешённых языком символов. Могут быть только rvalue. |

## 1.9 Объявление данных и область видимости

В языке SAV-2018 требуется обязательное объявление переменной перед её использованием. Имеется возможность инициализировать переменные как во время объявления, так и в последствии. Все переменные должны находиться внутри программного блока языка. Имеется возможность объявления одинаковых переменных в разных блоках. В случае объявления глобальной переменной будет выдана соответствующая ошибка. Каждая переменная получает префикс – название функции, в которой она объявлена.

## 1.10 Инициализация данных

В языке SAV-2018 предусмотрены следующие виды инициализации, представленные в таблице 1.4.

Таблица 1.4 – Инициализация данных

|  |  |
| --- | --- |
| Вид инициализации | Примечание |
| var<тип данных><идентификатор>; | Автоматическая инициализация: переменные типа int инициализируются нулём, переменные типа srt — пустой строкой |
| var<тип данных><идентификатор> = <значение>; | Инициализация при объявлении переменной |

## 1.11 Инструкции языка

В языке SAV-2018 предусмотрены инструкции, представленные в таблице 1.5.

Таблица 1.5 – Инструкции языка

|  |  |
| --- | --- |
| Инструкция | Синтаксис |
| Объявление переменной | var <тип данных><идентификатор>; |
| Объявление функции | <тип данных>function<идентификатор> (<тип данных><идентификатор>, …) […] |
| Присваивание | **<**идентификатор> = <выражение>; |
| Возврат из функции | return <выражение>; |
| Ввод данных | write: <идентификатор>; |
| Вывод данных | print: <выражение>; |
| Подключение стандартной библиотеки | lib.connect |

## 1.12 Операции языка

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

Таблица 1.6 – Операции языка

|  |  |
| --- | --- |
| Тип оператора | Операторы |
| Арифметические | () – приоритет  **+ –** сложение  - – вычитание |

Продолжение таблицы 1.6

|  |  |
| --- | --- |
|  | \* – умножение  / – деление  = – присваивание |
| Строковые | = – присваивание |

## 1.13 Выражения и их вычисления

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

– Допускается использовать скобки для составления приоритета;

– Выражение может содержать вызов функции;

– Каждое выражение обязано заканчиваться разделителем инструкций;

– Использование двух подряд идущих операторов не допускается;

– Выражение просматриваются справа налево и выполняются в соответствии с приоритетом.

## 1.14 Программные конструкции языка

В языке SAV-2018 предусмотрены программные конструкции, представленные в таблице 1.7.

Таблица 1.7 – Конструкции языка

|  |  |
| --- | --- |
| Конструкция | Синтаксис |
| Главная функция | main(){ ... } |
| Функция | <тип>function<идентификатор> (<тип><идентификатор>, …)  {  return <выражение>;  } |

## 1.15 Область видимости идентификаторов

В языке SAV-2018 переменные обязаны находится внутри программного блока функций (по принципу С++). Объявление глобальных переменных не предусмотрено. Объявление пользовательских областей видимости не предусмотрено.

## 1.16 Семантические проверки

В языке SAV-2018 предусмотрены семантические правила, представленные в таблице 1.8.

Таблица 1.8 – Семантические правила

|  |  |
| --- | --- |
| № | Правило |
| 1 | Все идентификаторы должны быть объявлены до использования. |
| 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 int. |
| 8 | Объявление глобального идентификатора не допускается. |

## 1.17 Распределение оперативной памяти на этапе выполнения

В языке SAV-2018 все литералы расположены в сегменте констант.

## 1.18 Стандартная библиотека и ее состав

В языке SAV-2018 предусмотрена стандартная библиотека, функции которой описаны в таблице 1.9.

Таблица 1.9 – Функции стандартной библиотеки

|  |  |
| --- | --- |
| Функция | Применение |
| str function tocollab (str a, str b) | Объединение строк a и b. |
| int function tolength (string a) | Определение длины строки. |

Стандартная библиотека написана на языке JavaScript и подключается при указании инструкции <lib.connect> в начале файла с исходным кодом.

## 1.19 Ввод и вывод данных

В языке SAV-2018 предусмотрен оператор ввода — write: <идентификатор>, и оператор вывода — print: <идентификатор> или print: <литерал>.

## 1.20 Точка входа

В языке SAV-2018 точкой входа является главная функция main(). Выполнение программы начинается с первой инструкции главной функции.

## 1.21 Препроцессор

В языке SAV-2018 препроцессор не предусмотрен.

## 1.22 Соглашения о вызовах

В языке SAV-2018 предусмотрено соглашение о вызовах \_stdcall. \_stdcall предусматривает передачу параметров через стек справа налево, память высвобождает вызываемый код.

## 1.23 Объектный код

Язык SAV-2018 транслируется в язык JavaScript, а затем – в объектный код.

## 1.24 Классификация сообщений транслятора

В языке SAV-2018 предусмотрена классификация сообщений транслятора, представленная в таблице 1.11.

Таблица 1.11 – Сообщения транслятора

|  |  |
| --- | --- |
| Префикс сообщения | Принадлежность |
| 0 – 10 | Ошибки, связанные с получением входных параметров (критическая ошибка) |
| 100-101: | Ошибки, полученные на этапе лексического анализа |
| 600 – 699: | Ошибки, полученные на этапе синтаксического анализа |
| 102 – 199: | Ошибки, полученные на этапе семантического анализа |

## 1.25 Контрольный пример

Контрольный пример языка SAV-2018 представлен в приложении А.

# 2. Структура транслятора

## 2.1 Компоненты транслятора их назначение и принципы взаимодействия

Лексический анализатор считывает последовательно все слова (токены, лексемы) на вход исходный код программы на языке SAV-2018, удаляет избыточные пробелы, разделяя весь код на отдельные цепочки и на основе конечных автоматов, построенных для всех допустимых цепочек, строит таблицу лексем и таблицу идентификаторов. Все сообщения об ошибках, полученных на этапе лексического анализа передаются в протоколе работы компилятора.

Синтаксический анализатор принимает на вход полученную таблицу лексем и путем перебора всех возможных синтаксический конструкций языка SAV-2018 формирует дерево разбора. В случае неправильного разбора выводит ошибки в протокол работы синтаксического анализатора.

Следом за синтаксическим анализом следует этап семантического анализа. Семантический анализатор на основе синтаксических правил языка SAV-2018 проверяет полученные данные с предыдущих этапов и выдаёт ошибки в случае их обнаружения.

Генератор кода принимает таблицы лексем и идентификаторов, дерево разбора и путём подстановки определённых шаблонов под каждое правило из дерева разбора переводит исходный код на языке SAV-2018 на язык JavaScript.



Рисунок 2.1 – Структура транслятора

## 2.2 Перечень входных параметров транслятора

Язык SAV-2018 предусматривает входные параметры, описанные в таблице 2.1.

Таблица 2.1 – Входные параметры транслятора

|  |  |
| --- | --- |
| Параметр | Назначение |
| -in:<имя> | Обязательный параметр, указывающий путь к файлу исходного кода на языке SAV-2018. |
| -out:<имя> | Необязательный параметр, указывающий путь к файлу выходного кода на языке . В случае его отсутствия путь к файлу строится с помощью пути, полученного при передаче параметра –in:. |
| -log:<имя> | Необязательный параметр, указывающий путь к файлу с протоколом работы транслятора. В случае его отсутствия путь к файлу строится с помощью пути, полученного при передаче параметра –in: с постфиксом .log |
| **-**LT | Необязательный параметр. Выводит в файл выходного протокола таблицу лексем. |
| -IT | Необязательный параметр. Выводит в файл выходного протокола таблицу идентификаторов. |
| -PN | Необязательный параметр. Выполняет перевод выражений в таблице лексем в польскую запись. |
| -SA | Необязательный параметр. Создает в каталоге проекта SAV-2018 файл in.txtSint.txt. Выполняет синтаксический анализ исходной программы и выводит в созданный файл результат работы синтаксического анализатора транслятора. |
| -CG | Необязательный параметр. Создает в каталоге проекта SAV-2018 файл in.txt.html. Выполняет генерацию исходного кода на языке SAV-2018 в код программы на языке JavaScript . |

## 2.3 Перечень протоколов формируемых транслятором и их содержимое

Транслятор на языке SAV-2018 формирует протоколы, представленные в таблице 2.2.

Таблица 2.2 – Протоколы транслятора

|  |  |
| --- | --- |
| Протокол | Содержимое |
| in.txt.log | Содержит информацию о входных данных, таблице лексем, таблице идентификаторов и ошибках, найденных на этапах лексического анализа и семантического анализа. |
| in.txtSint.txt | Содержит информацию о синтаксическом анализаторе и ошибках, найденных в процессе синтаксического анализа. |
| in.txt.html | Содержит код исходной программы, сгенерированный в язык. |

# 3. Разработка лексического анализатора

## 3.1 Структура лексического анализатора

Входными данными для лексического анализатора является исходный код программы, написанной на языке SAV-2018.

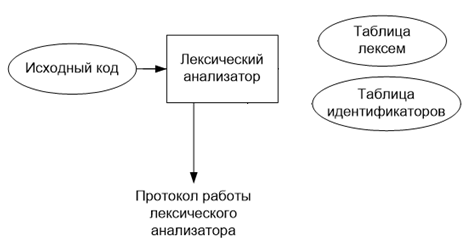
****

Рисунок 3.1 – Структура лексического анализатора

## 3.2 Контроль входных символов

Таблица, контролирующая входные символы, представлена на рисунке 3.2.

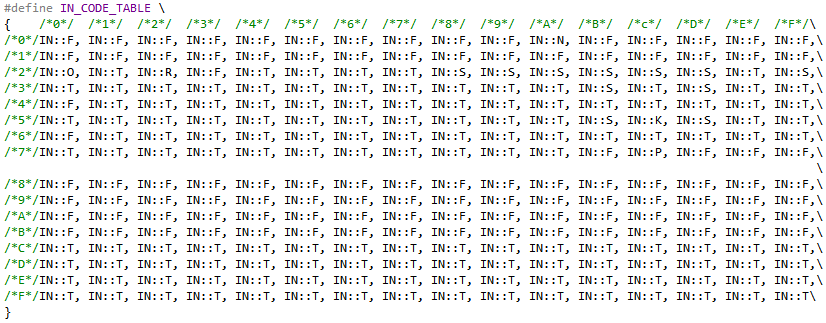


Рисунок 3.2 – Контроль входных символов

Таблица содержит типы входных символов, каждый элемент массива соответствует номеру символа в таблице Windows-1251. При проверке исходного кода идет сравнение типа входного символа и типа, соответствующего его индексу в этом массиве. В зависимости от назначения символы делятся на группы:

– F – запрещенные в языке символы;

– Т – разрешенные символы;

– K – символ начала однострочного комментария;

– P – символ, обозначающий вертикальную черту;

– S – символы-сепараторы;

– R – символ двойных кавычек;

– N – символ перевода строки;

– О – символ пробела.

## 3.3 Удаление избыточных символов

Алгоритм удаления избыточных символов в языке SAV-2018 описан ниже:

1. Посимвольное считывание исходного кода на языке SAV-2018.
2. Встреча двойной кавычки ставит флаг, который не позволяет удалять избыточные символы. Повторная встреча двойных кавычек снимает данный флаг.
3. При нахождении пробела просматривается следующий символ. Если следующий символ является пробелом или сепаратором, то текущий символ игнорируется.
4. При нахождении вертикальной черты просматривается следующий символ. Если следующий символ является вертикальной чертой, текущий символ игнорируется.
5. При нахождении обратной косой черты просматривается следующий символ до нахождения символа вертикальной черты. Все найденные в этом промежутке символы игнорируются. После этого проверка продолжается с символа, следующего за символом вертикальной черты.
6. В конце каждой итерации посимвольного считывания текущий символ сохраняется в буфер.

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

В языке SAV-2018 предусмотрены ключевые слова, сепараторы, символы операций и соответствующие им лексемы, приведённые в таблице 3.1.

Таблица 3.1 – Ключевые слова, сепараторы, операторы и их описание

|  |  |  |
| --- | --- | --- |
| Тип цепочки | Цепочка | Лексема |
| Ключевые слова | var | D |
| int | t |
| str | t |
| function | F |
| write; | W |
| return | R |
| main | M |
| Сепараторы | ; | ; |
| , | , |
| { | { |

Продолжение таблицы 3.1

|  |  |  |
| --- | --- | --- |
|  | } | } |
| ( | ( |
| ) | ) |
| Операторы | + | V |
| - | V |
| \* | V |
| / | V |
| = | = |
| Остальное | Идентификатор | I |
| Целочисленный литерал | L |
| Строковый литерал | L |

## 3.5 Основные структуры данных

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

## 3.6 Принцип обработки ошибок

При нахождении ошибки в процессе работы лексического анализатора языка SAV-2018 заполняется определённая структура, содержащая идентификатор ошибки, номер строки, в которой найдена ошибка и позиция на этой строке, и в последствии выводится в протокол работы лексического и семантического анализаторов. Таблица идентификаторов при этом выведена не будет. Также не будет выполнена генерация исходного кода. Ошибки синтаксического анализатора записываются в файл с результатом работы синтаксического анализатора.

## 3.7 Структура и перечень сообщений лексического анализатора

В языке SAV-2018 предусмотрена ошибка, найденная в процессе работы лексического анализатора. Она представлена в таблице 3.2.

Таблица 3.2 – Сообщения лексического анализатора

|  |  |
| --- | --- |
| Код | Сообщение |
| 100 | Фрагмент исходного кода не распознан. |
| 101 | Недопустимый символ. |

## 3.8 Параметры лексического анализатора и режимы его работы

В языке SAV-2018 допускаются параметры лексического анализатора, приведённые в таблице 3.3.

Таблица 3.3 – Параметры лексического анализатора

|  |  |
| --- | --- |
| Параметр | Назначение |
| -LT | Необязательный параметр. Отвечает за вывод таблицы лексем в протокол работы компилятора лексического и семантического анализаторов. |
| -IT | Необязательный параметр. Отвечает за вывод таблицы идентификаторов в протокол работы лексического и семантического анализаторов. |

## 3.9 Алгоритм лексического анализа

Лексический анализ в языке SAV-2018 основан на работе конечных автоматов для всех разрешённых цепочек. При удачном разборе цепочки лексема, соответствующая автомату, заносится в таблицу лексем. Если лексема соответствует идентификатору, литералу или оператору, то цепочка заносится в таблицу идентификаторов. Все найденные ошибки выводятся в протокол работы транслятора. Пример работы конечного автомата для цепочки main представлен на рисунке 3.3.



Рисунок 3.3 – Граф переходов для цепочки main

S0 соответствует начальному состоянию автомата, Sn – конечному состоянию автомата. Цепочка считается прочтённой, если автомат пришёл в конечное состояние и на своей ленте не имеет больше символов.

## 3.10 Контрольный пример

Результатом работы лексического анализатора являются таблицы лексем и идентификаторов. Содержимое таблиц на основе контрольного примера из п. 1.23 представлено в приложении В.

# 4 Разработка синтаксического анализатора

## 4.1 Структура синтаксического анализатора

В языке SAV-2018 синтаксический анализатор принимает на свой вход таблицу лексем и таблицу идентификаторов. На основе полученных данных идёт перебор всех возможных синтаксических конструкций языка и в случае успеха —построение дерева разбора. Структура синтаксического анализатора представлена на рисунке 4.1.

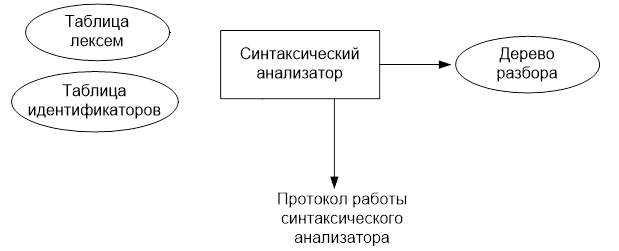


Рисунок 4.1 – Структура синтаксического анализатора

## 4.2 Контекстно-свободная грамматика, описывающая синтаксис языка

Список нетерминальных символов, используемых при синтаксическом анализе языка SAV-2018 представлен в таблице 4.1

Таблица 4.1 – Описание нетерминальных символов SAV-2018

|  |  |
| --- | --- |
| Нетерминал | Описание |
| S | Порождает стартовые правила, описывающее общую конструкцию программы (функции) |
| N | Порождает правила, описывающие инструкции языка |
| E | Порождает правила, описывающие выражения |
| F | Порождает правила, описывающие формальные параметры функции |
| M | Порождает правила, описывающие арифметические действия |

Продолжение таблицы 4.1

|  |  |
| --- | --- |
| W | Порождает правила, описывающие фактические параметры функции |

Перечень правил, описывающих и формирующих контекстно-свободную грамматику языка SAV-2018, представлен в таблице 4.2.

Таблица 4.2 – Перечень правил, составляющих грамматику языка SAV-2018

|  |  |
| --- | --- |
| Нетерминал | Цепочки правил |
| S | m[N] | m[N]; | tfi()[NrE;]S | tfi()[N]S | tfi(F)[NrE;]S | tfi()[NrE;] | tfi()[N] | tfi(F)[NrE;] |
| F | ti | ti,F |
| W | i | l | i,w | l,w |
| N | dti; | dti=E; | rE; | i=E; | i(); | wE; | pE; | pN; | dti;N | dti=E;N | rE;N | i=E;N | i();N | wE;N | pE;N | pN;N |
| E | i | l | (E) | i(W) | iM | lM | (E)M | i(W)M |
| M | vE | vEM |

## 4.3 Построение конечного магазинного автомата

Конечный автомат с магазинной памятью представляет собой семерку,. Подробное описание компонентов магазинного автомата представлено в таблице 4.3.

Таблица 4.3 – Описание компонент магазинного автомата

|  |  |
| --- | --- |
| Компонента | Определение |
|  | Множество состояний автомата |
|  | Алфавит входных символов |
|  | Алфавит специальных магазинных символов |
|  | Функция переходов автомата |
|  | Начальное состояние автомата |
|  | Начальное состояние магазина автомата |
|  | Множество конечных состояний |

## 4.4 Основные структуры данных

Основные структуры данных синтаксического анализатора представляются в виде структуры магазинного конечного автомата, выполняющего разбор исходной ленты, и структуры грамматики Грейбах, описывающей синтаксические правила языка SAV-2018. Структура данного автомата показана в приложении Г.

## 4.5 Описание алгоритма синтаксического разбора

Алгоритм синтаксического анализа языка SAV-2018 представлен следующим образом:

1. Просмотр стека магазинного автомата и таблицы лексем.

2. В случае нахождения на вершине стека нетерминального символа подбирается такое правило перехода из него в цепочку, чтобы первая лексема правила перехода совпадала с символом, лежащим в текущий момент на ленте и сохранение текущего состояния.

3. В случае провала – возврат в предыдущее состояние и повтор пункта 2.

4. Если на ленте не осталось символов и стек стал пустым, то информация о дереве разбора выводится в протокол синтаксического анализатора.

5.В случае выявления ошибок – занесение их в протокол синтаксического анализатора.

## 4.6 Структура и перечень сообщений синтаксического анализатора

В языке SAV-2018 предусмотрены сообщения синтаксического анализатора, представленные в таблице 4.4.

Таблица 4.4 – Сообщения синтаксического анализатора

|  |  |
| --- | --- |
| Код | Сообщение |
| 600 | Неверная структура программы |
| 601 | Неверное составление функции |
| 602 | Неверное выражение |
| 603 | Ошибка, связанная со знаками |
| 604 | Неверные формальные параметры функции |
| 605 | Неверные фактические параметры функции |

## 4.7 Параметры синтаксического анализатора и режимы его работы

В языке SAV-2018 предусмотрен параметр синтаксического разбора, представленный в таблице 4.5.

Таблица 4.5 – Параметры синтаксического анализатора

|  |  |
| --- | --- |
| Параметр | Описание |
| -SA | Создает файл in.txtSint.txt. Выполняет синтаксический анализ исходной программы и выводит в созданный файл результат работы синтаксического анализатора транслятора. |

## 4.8 Принцип обработки ошибок

Синтаксический анализатор перебирает все возможные правила и цепочки правила грамматики в целях поиска подходящего соответствия. Если ни одна из цепочек правила не подошла для рассматриваемой конструкции, то генерируется ошибка в соответствии с таблицей 4.2. Ошибка выводится в протокол работы синтаксического анализатора.

## 4.9 Контрольный пример

Результатом работы синтаксического анализатора является дерево разбора, содержимое которого представлено в приложении Д. Фрагменты процесса разбора представлены в приложении З.

# 5 Разработка семантического анализатора

## 5.1 Структура семантического анализатора

В компиляторе языка SAV-2018 предусмотрен семантический анализатор. Он является встроенным в лексический анализатор, и выполняет поиск ошибок на этапе лексического анализа.

## 5.2 Функции семантического анализатора

В языке SAV-2018 предусмотрены следующие правила, которые использованные в процессе работы семантического анализа:

– Все идентификаторы должны быть объявлены до использования.

– При объявлении идентификатора должен указываться тип.

– Длина идентификатора не должна превышать 5 символов.

– Тип формальных параметров функции должен совпадать с типом фактических параметров, переданных в функцию.

– Количество фактических параметров, переданных в функцию должно соответствовать количеству формальных параметров вызываемой функции.

– Длина строчного литерала не должна превышать 255 символов без учёта кавычек.

– Применение арифметических операций допускается только с целочисленным типом int.

– Объявление глобального идентификатора не допускается.

## 5.3 Структура и перечень сообщений семантического анализатора

В языке SAV-2018 предусмотрены сообщения семантического анализатора, представленные в таблице 5.2.

Таблица 5.1 – Сообщения семантического анализатора

|  |  |
| --- | --- |
| Код | Сообщение |
| 101 | Недопустимый символ в файле с исходным кодом. |
| 102 | Допустимая длина идентификатора превышена. |
| 103 | Не объявленный идентификатор. |
| 104 | Нельзя присвоить значение функции. |
| 105 | Допустимая длина строкового литерала превышена. |

## 5.4 Принцип обработки ошибок

В языке SAV-2018 все ошибки семантического анализатора записываются в протокол работы лексического и семантического анализаторов.

## 5.5 Контрольный пример

Контрольный пример основывается на тестировании функций семантического анализатора при наличии соответствующих ошибок в исходном коде. Контрольный пример разработан таким образом, что в нём отсутствуют какие-либо семантические ошибки. Примеры вывода семантических ошибок описаны в пункте 8.5.

# 6 Преобразование выражений

## 6.1 Выражения, допускаемые языком

В языке SAV-2018 допускаются вычисления выражений с целочисленными типами данных, а также с функциями, возвращающими целочисленный результат, при помощи специальных операторов. Каждый оператор имеет свой приоритет, что показано в таблице 6.1.

Таблица 6.1 – Приоритеты операций

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

## 6.2 Польская запись и принцип ее построения

Польская запись удобна для вычисления выражений и как промежуточная форма представления выражений в трансляторе. Принцип обратной польской записи может быть применен не только к выражениям, но и операторам языков программирования.

Выражение читается слева направо. Операция выполняется над двумя операндами, непосредственно стоящими перед знаком этой операции. Результат операции заменяет в выражении последовательность её операндов и знак операции. Результатом вычисления выражения становится результат последней вычисленной операции.

Алгоритм построения:

– исходная строка: выражение;

– результирующая строка: польская запись;

– стек: пустой;

– 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;

– квадратная закрывающая скобка выталкивает все до открывающей и генерирует @n (индекс n указывает число операндов, разделенных запятыми);

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

## 6.3 Программная реализация обработки выражений

|  |  |  |
| --- | --- | --- |
| Исходная строка | Результирующая строка | Стек |
|  |  |  |
|  |  | ( |
|  |  | ( |
|  |  | (+ |
|  |  | (+ |
|  |  |  |
|  |  | \* |
|  |  | \*( |
|  |  | \*( |
|  |  | \*(– |
|  |  | \*(– |
|  |  |  |
|  |  |  |

## 6.4 Контрольный пример

Контрольный пример реализации польской записи на основе контрольного примера из главы 1 и представлен в приложении И.

# 7 Генерация кода

## 7.1 Структура генератора кода

В языке SAV-2018 генерация кода является заключительным этапом транслятора. Генератор принимает на вход таблицы лексем и идентификаторов, а также дерево разбора, полученные в результате лексического и синтаксического анализов. В соответствии с деревом разбора при помощи заготовленных шаблонов строится выходной файл на языке JavaScript, который будет являться результатом работы транслятора в целом. Структура генератора кода SAV-2018 представлена на рисунке 7.1.

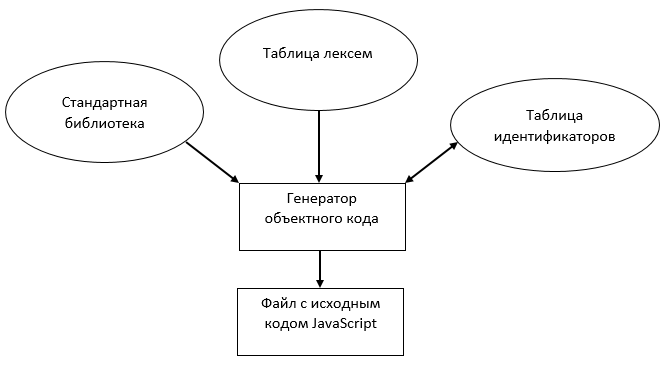


Рисунок 7.1 – Структура генератора кода

## 7.2Принцип построения объектного кода

Принцип построения объектного кода генератором кода описан ниже:

1. В результате работы лексического анализатора происходит проверка на наличие параметра подключения статической библиотеки в файле с исходным кодом на языке SAV-2018. При нахождении параметра подключается файл, содержащий функции стандартной библиотеки. В выходной файл записывается заголовок HTML-документа, а также строка, подключающая файл стандартной библиотеки. Переходим к п.3. Иначе – переходим к п.2.

2. В выходной файл записывается заголовок HTML-документа.

3. Берем первую лексему из таблицы лексем. Пока не достигнут конец таблицы лексем, проверяем текущую лексему. В результате проверки записываем соответствующее преобразование лексемы в файл с объектным кодом. Затем переходим к проверке следующей лексемы:

3.1 Если лексема “m”, формируем функцию main.

3.2 Если лексема является скобкой, символом равенства, запятой, оператором, записываем аналогичный символ.

3.3 Если лексема является левой/правой квадратной скобкой, записываем левую/правую фигурную скобку соответственно.

3.4 Если лексема “r”, записываем оператор возврата значения.

3.5 Если лексема является знаком операции, записываем соответствующую операцию из таблицы лексем.

3.6 Если лексема является точкой с запятой, записываем закрывающую круглую скобку. Затем записываем точку с запятой.

3.7 Если лексема “f”, записываем оператор определения функции.

3.8 Если лексема является идентификатором, выполняем проверку с помощью таблиц лексем и идентификаторов:

3.8.1 Если предшествующая лексема “t”, записываем оператор определения переменной. Иначе – переход к п.3.8.2.

3.8.2 Записываем идентификатор.

3.9 Если лексема является литераломвыполняем проверку с помощью таблицы идентификаторов:

3.10 Если литерал численного типа, записываем его. Иначе – переход к п.3.9.2.

3.11 Если литерал строкового типа, заключаем его в одинарные кавычки. Записываем его вместе с кавычками.

3.12 Если лексема определяет оператор ввода, записываем следующую за ним лексему, равно, функцию ввода сообщения языка JavaScript.

3.13 Если лексема определяет оператор вывода, записываем функцию вывода языка JavaScript.

4. Записываем вызов функции main.

5. Закрываем “тело” HTML-документа.

## 7.3 Статическая библиотека

В языке SAV-2018 предусмотрена статическая библиотека. Статическая библиотека содержит функции, написанные на языке JavaScript. Подключение библиотеки осуществляется параметром “lib.connect” в первой строке файла с исходным кодом на языке SAV-2018. Функции стандартной библиотеки описаны в таблице 7.1.

Таблица 7.1 – Функции статической библиотеки

|  |  |
| --- | --- |
| Функция | Применение |
| str function tocollab (str a, str b) | Объединение строк a и b. |
| int function tolength (string a) | Определение длины строки. |

## 7.4 Входные параметры генератора кода

В языке SAV-2018 предусмотрены входные параметры генератора кода, который представлен в таблице 7.5.

Таблица 7.5 – Параметры генератора кода

|  |  |
| --- | --- |
| Параметр | Назначение |
| -CG | Необязательный параметр. Создает в каталоге проекта SAV-2018 файл in.txt.html. Выполняет генерацию исходного кода на языке SAV-2018 в код программы на языке JavaScript. |

## 7.5 Контрольный пример

Результат генерации JavaScript кода на основе контрольного примера из приложения А приведён в приложении Ж. Результат работы контрольного примера приведён в приложении И.

# 8. Тестирование транслятора

## 8.1 Общие положения

Для тестирования программ в языке SAV-2018 будут использоваться заведомо исходные тексты. В случае нахождения ошибок вся информация будет содержаться в соответственных протоколах работы транслятора.

# 8.2 Тестирование проверки на допустимость символов

В языке SAV-2018 не разрешается использовать запрещённые входным алфавитом символы. Результат использования запрещённого символа показан в таблице 8.1.

Таблица 8.1 – Тестирование проверки на допустимость символов

|  |  |
| --- | --- |
| Исходный код | Диагностическое сообщение |
| main  {  ~  } | Ошибка 101: Недопустимый символ в файле с исходным кодом, строка:3, позиция 1 |

# 8.3 Тестирование лексического анализатора

На этапе лексического анализа в языке SAV-2018 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 |
| Main  {  Var int a!b;  } | Ошибка 100: Фрагмент исходного кода не распознан, cтрока: 3 |

# 8.4 Тестирование синтаксического анализатора

На этапе синтаксического анализа в языке SAV-2018 могут возникнуть ошибки, описанные в пункте 4.6. Результаты тестирования лексического анализатора показаны в таблице 8.3.

Таблица 8.3 - Тестирование синтаксического анализатора

|  |  |
| --- | --- |
| Исходный код | Диагностическое сообщение |
| Main  } | 600: Неверная структура программы, строка: 1 |
| int function () | 601: Неверное составление функции, строка: 1 |

Продолжение таблицы 8.3

|  |  |
| --- | --- |
| int function ()  {  b = 1 ++ 3;  } | 602: Неверное выражение, строка: 3 |
| intfunctionsum()  {  b = 1 3;  } | 603: Ошибка, связанная со знаками, строка: 1 |
| int function sum (int int a)  {  } | 604: Неверные формальные параметры функции, строка: 1 |
| sum(a b c); | 605: Неверные фактические параметры функции, строка: 1 |

# 8.5 Тестирование семантического анализатора

Семантический анализ в языке SAV-2018 содержит проверки по семантическим правилам, описанным в пункте 1.16. Итоги тестирования семантического анализатора приведены в таблице 8.4.

Таблица 8.4 – Тестирование семантического анализатора

|  |  |
| --- | --- |
| Исходный код | Диагностическое сообщение |
| Main  {  ~  } | Ошибка 101: Недопустимый символ в файле с исходным кодом, cтрока: 3, позиция 1 |
| main  {  var int a12345678;  } | Ошибка 102: Допустимая длина идентификатора превышена, строка: 3 |
| main  {  abc;  } | Ошибка 103: Не объявленный идентификатор, строка 3 |
| int function sum()  {  }  main()  {  sum = 5;  } | Ошибка 104: Нельзя присвоить значение функции, строка 6 |
| var str a = “aaaaaaa…aaaaa” | Ошибка 105: Допустимая длина строкового литерала превышена, строка 1 |

# Заключение

В ходе курсового проекта был разработан новый язык программирования SAV-2018, а также компилятор для данного языка.

Язык программирования SAV-2018 содержит:

– 2 типа данных: целочисленный int и строковый str;

– 5 арифметических операторов;

– 8 ключевых слов;

– Стандартную библиотеку, написанную на языкаJavaScipt, содержащую 2 функции;

– Оператор вывода в диалоговое окно: print: <выражение>;

– Оператор ввода: write: <выражение>.

Транслятор SAV-2018 содержит:

– Лексический анализатор;

– Синтаксический анализатор;

– Семантический анализатор;

– Генератор кода.

Общее число протоколов, выводимых в ходе трансляции – 3.

Общее количество ошибок, обрабатываемых транслятором – 18.

Общее количество автоматов, применяемых при лексическом анализе – 22.

Общее число шаблонов, применяемых при генерации кода – 19.

Общее количество строк, написанных в ходе данной работы – около 2700.

# Приложение А

Контрольный пример языка SAV-2018:

lib.connect

str function out()

{

var str q = "this function";

print: q;

print: "is working";

}

int function sum (int a, int b)

{

var int c;

c = (a+b)\*(a-b);

returnb;

}

\однострочный комментарий

main

{

var int f = tolenght("sav");

var int a = 6;

print: tocollab("this","thing");

var int prim = sum(a, 2);

print: prim;

var str test;

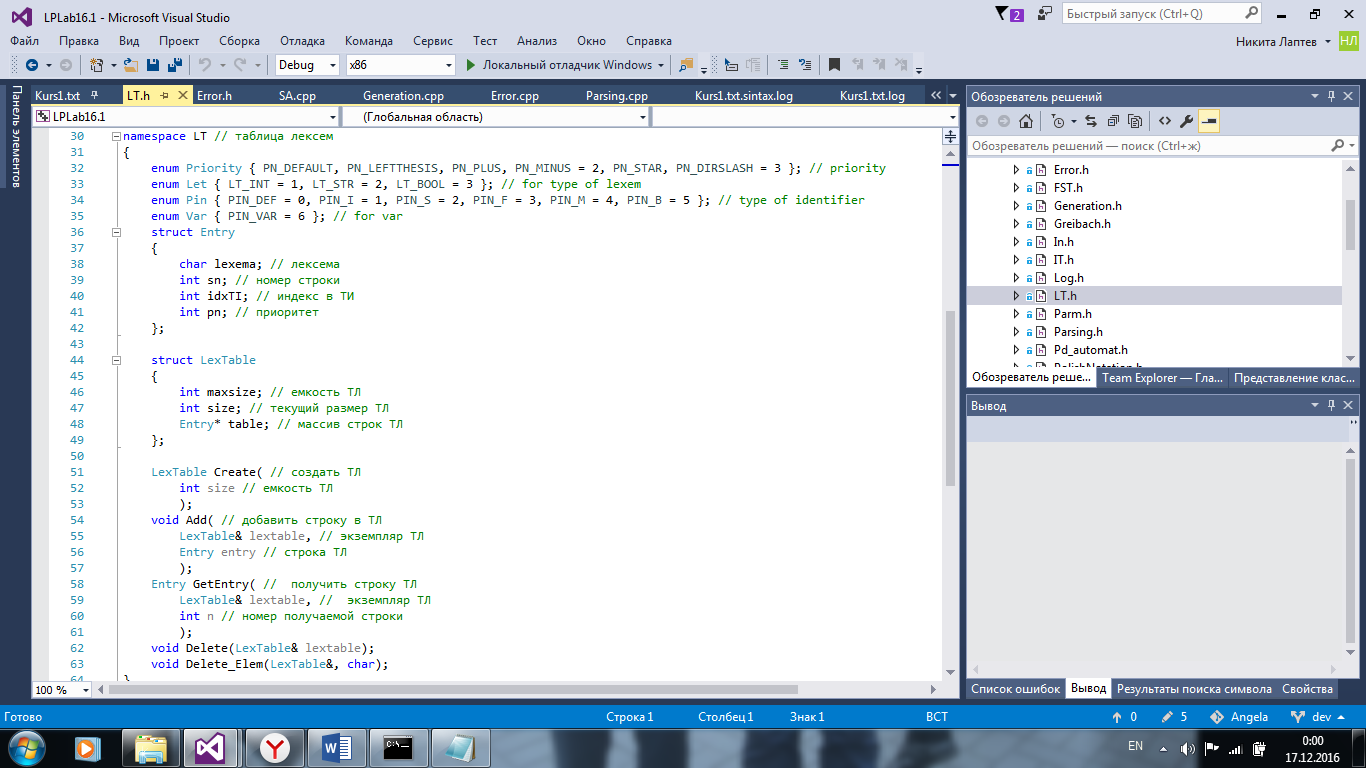
write: test;

print: test;

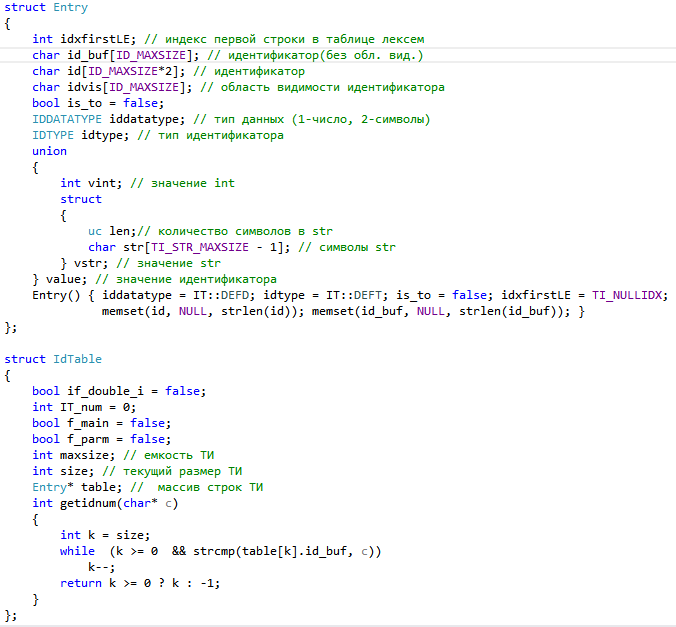
out();

}

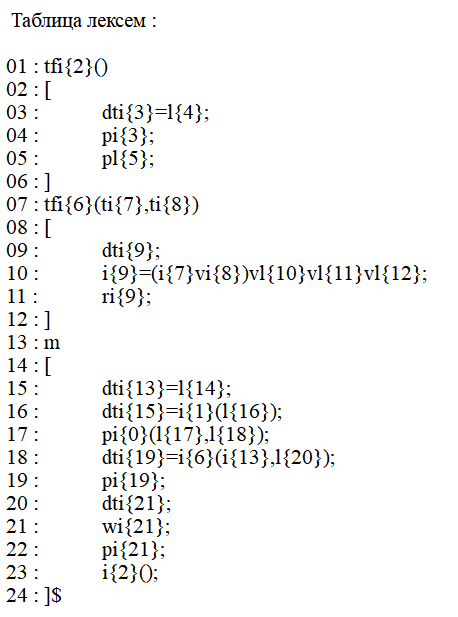
# Приложение Б

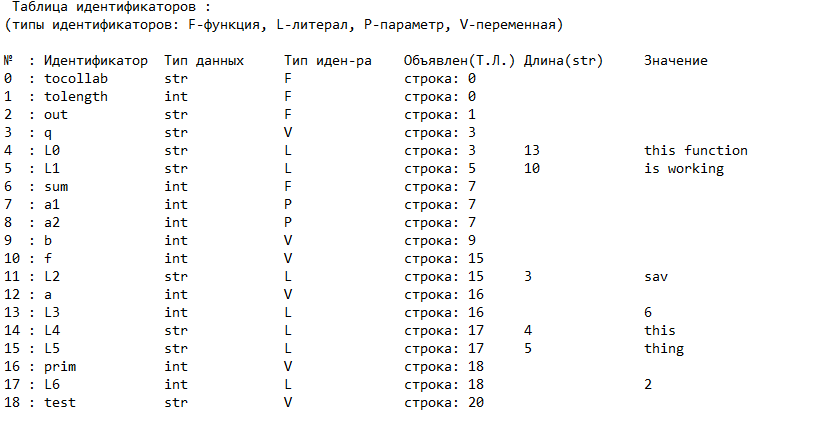
Структура данных для таблицы лексем и её элементов в языке SAV-2018:  


Структура данных для таблицы идентификаторов и её элементов в языке SAV-2018:

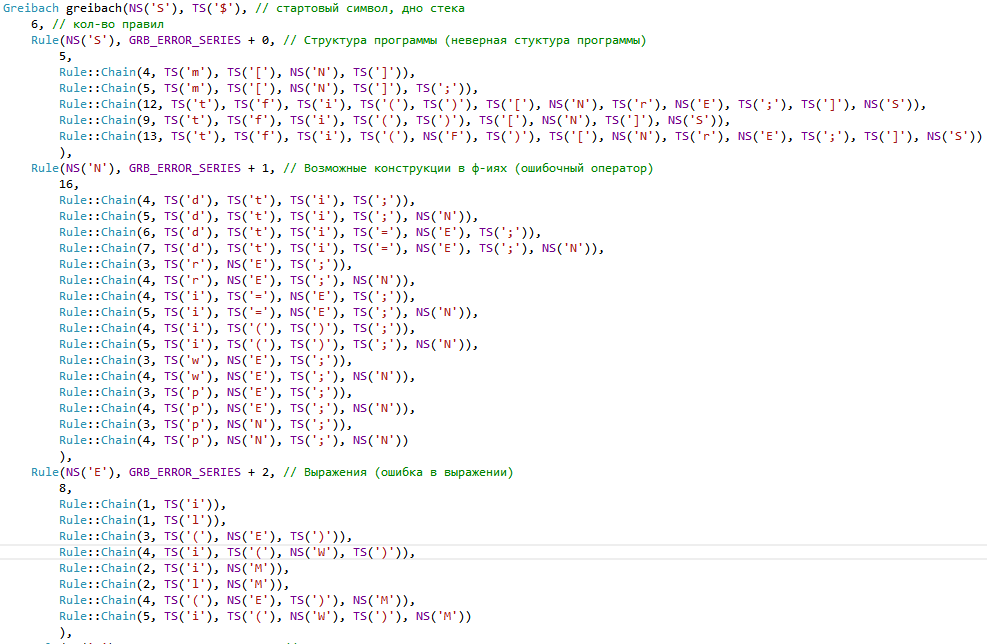


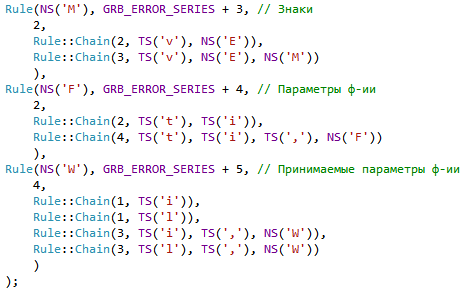
Приложение В  
Таблицы лексем и идентификаторов для контрольного примера языка SAV-2018:





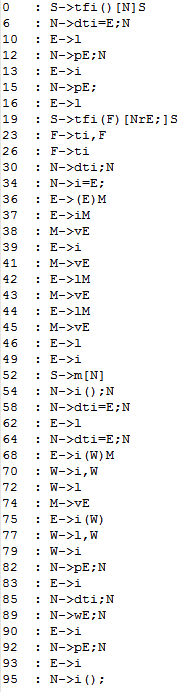
Приложение Г

Структура автомата с магазинной памятью, построенного на основе грамматики Грейбах для языка SAV-2018:  




# Приложение Д

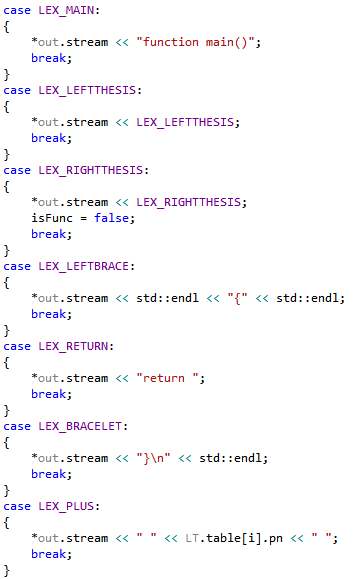
Дерево разбора на основе контрольного примера языка SAV-2018:

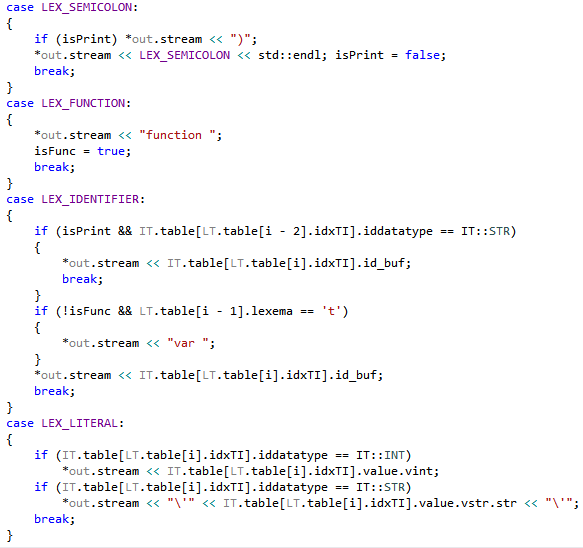


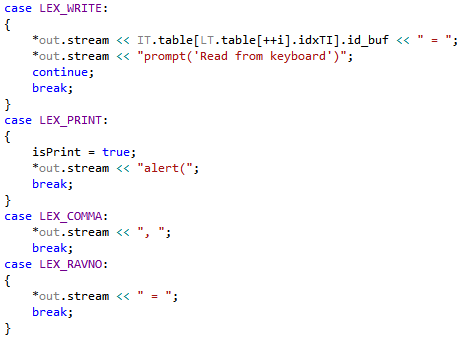
# **Приложение Е**

Шаблоны генератора кода SAV-2018:









Приложение Ж

Результат генерации кода на язык JavaScript для контрольного примера языка SAV-2018:

<!DOCTYPE HTML>

<html>

<head>

<meta charset = "win-1251">

<script src="lib.js"></script>

</head>

<body>

<script>

function out()

{

var q = 'this function';

alert(q);

alert('is working');

}

function sum(a1, a2)

{

var b;

b = (a1 + a2) \* (a1 - a2);

return b;

}

function main()

{

var f = tolength('sav');

var a = 6;

alert(tocollab('this', 'thing'));

var prim = sum(a, 2);

alert(prim);

var test;

test = prompt('Read from keyboard');

alert(test);

out();

}

main();

</script>

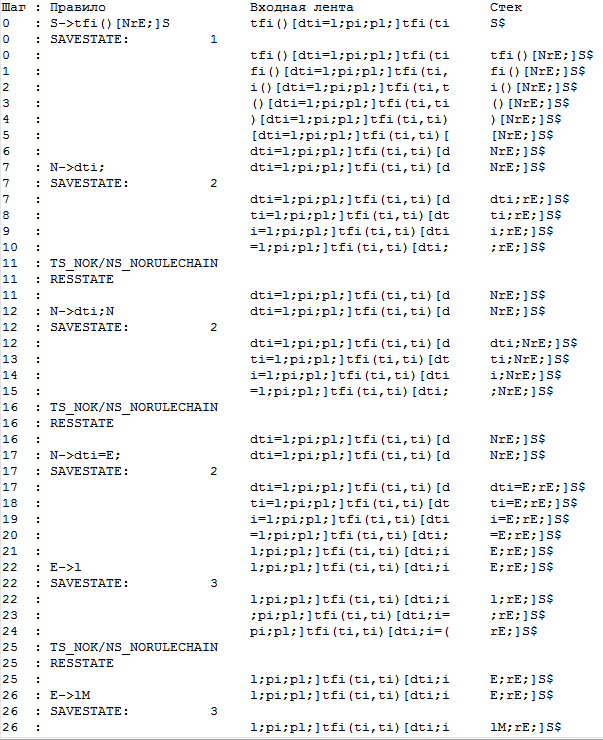
</body>

</html>

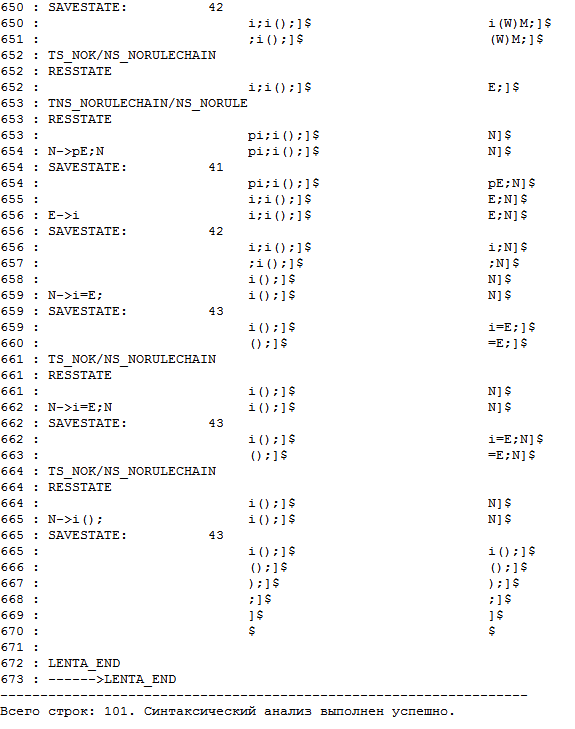
# Приложение З

Фрагменты процесса синтаксического разбора контрольного примера языка SAV-2018:

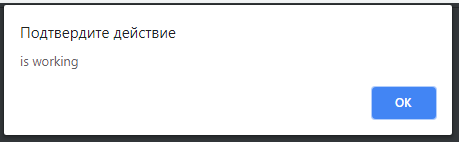
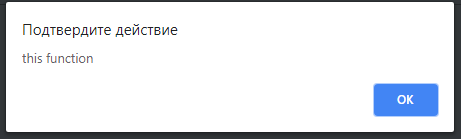
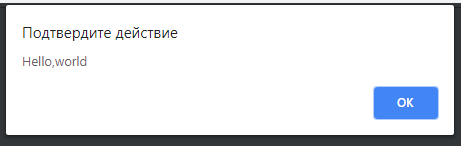
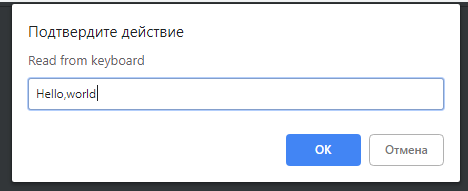
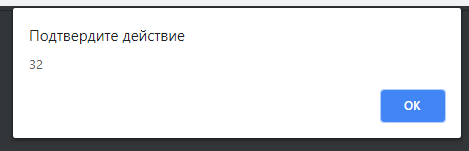
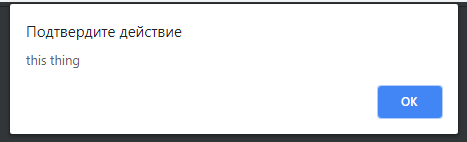
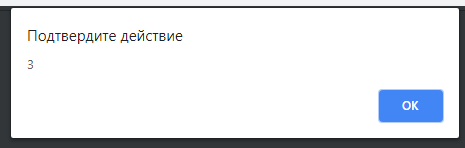
* Начало разбора



* Конец разбора



Приложение И

Результат работы контрольного примера:  


# Список используемой литературы

1. Ахо, А. Компиляторы: принципы, технологии и инструменты / А. Ахо, Р. Сети, Дж. Ульман. – M.: Вильямс, 2003. – 768с.

2. Смелов, В.В. Курс лекций по предмету языки программирования – 2016

3. Ахо, А. Теория синтаксического анализа, перевода и компиляции /А. Ахо, Дж. Ульман. – Москва : Мир, 1998. – Т. 2 : Компиляция. - 487 с.

4. Герберт, Ш. Справочник программиста по C/C++ / Шилдт Герберт. - 3-е изд. – Москва : Вильямс, 2003. - 429 с.

5. Прата, С. Язык программирования С++. Лекции и упражнения / С.Прата. – М., 2006 — 1104 c.

6. Страуструп, Б. Принципы и практика использования C++ / Б. Страуструп – 2009 – 1238 с.

7. \_\_stdcall [Электронный ресурс] – Режим доступа: <https://msdn.microsoft.com/en-us/library/zxk0tw93.aspx> - Дата доступа: 14.12.2017.

8. Windows 1251 [Электронный ресурс] – Режим доступа: <https://en.wikipedia.org/wiki/Windows-1251> - Дата доступа: 09.12.2017.