# **Глава 1. Спецификация языка программирования**

## **Характеристика языка программирования**

Язык программирования GMP-2019 предназначен для выполнения сравнения целых чисел и операций над строками.

Язык программирования GMP-2019 является процедурным, строго типизированным, не объектно-ориентированным, компилируемым.

* 1. **Алфавит языка**

Алфавит языка GMP-2019 основан на кодировке ASCII. Таблица кодировки ASCII представлена на рисунке 1.1.

Исходный код может содержать символы латинского алфавита, цифры десятичной системы счисления, символы пробела, табуляции, перевода строки, спецсимволы [], символы операторов: “< > = !” и символы сепараторов: , ; {}()

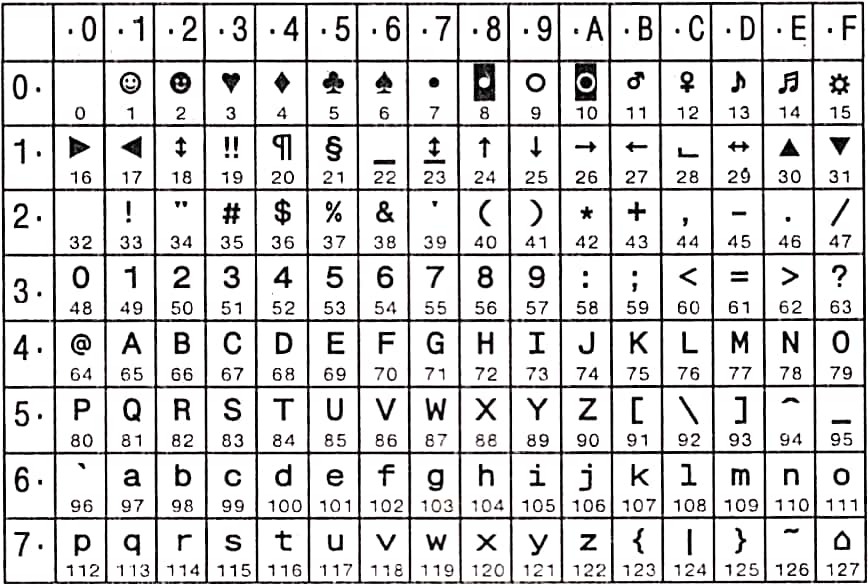


Рисунок 1.1 – Таблица кодировки ASCII

* 1. **Применяемые сепараторы**

Применяемые сепараторы в языке GMP-2019, приведены в таблице 1.1. Таблица 1.1 — Применяемые сепараторы

|  |  |
| --- | --- |
| Сепаратор | Назначение сепаратора |
| ; | Разделитель инструкций |
| { } | Программный блок |
| ( ) | Параметры |
| ‘ ’ | Служит для разделения. Допускается везде, кроме идентификаторов и ключевых слов |
| , | Разделитель параметров в функции |

* 1. **Применяемые кодировки**

Для написания исходного кода на языке программирования GMP-2019 используется кодировка ASCII, которая представлена на рисунке 1.1.

## **Типы данных**

В языке GMP-2019 есть 2 типа данных: целочисленный и строковый. Описание типов данных, предусмотренных в данным языке представлено в таблице 1.2.

Таблица 1.2 – Типы данных языка GMP-2019

|  |  |
| --- | --- |
| Тип данных | Описание типа данных |
| ushort | Фундаментальный тип данных. Предусмотрен для объявления беззнаковых целочисленных данных (2 байта).  Автоматически инициализируется нулевым значением.  Операции:  <= бинарная операция сравнения меньше либо равно;  >= бинарная операция сравнения больше либо равно;  == бинарная операция сравнения на равенство;  != бинарная операция сравнения на не равенство;  < бинарная операция сравнения меньше;  > бинарная операция сравнения больше; |
| line | Является строковым типом данных. Предназначен для работы с символами, каждый символ занимает 1 байт. Максимальное количество символов – 255, включая символ окончания строки.  Инициализация по умолчанию: нулевой символ. |

## **Преобразование типов данных**

Язык GMP-2019 строго типизированный, преобразование типов данных не поддерживается.

## **Идентификаторы**

Общее количество идентификаторов ограничено максимальным размером таблицы идентификаторов. Идентификаторы должны начинаться только с символов латинского алфавита, могут содержать цифры. Максимальная длина идентификатора равна 8 символам. Идентификаторы, объявленные внутри функционального блока, получают префикс, идентичный имени функции, внутри которой они объявлены. Префикс занимает 8 дополнительных символов. В случае превышения заданной длины, идентификаторы усекаются до длины, равной 16 символов (8 символов на имя идентификатора, 8 символов на префикс). Данные правила действуют для всех типов идентификаторов.

<буква> ::= a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z

<цифра> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<идентификатор> ::= <буква>{ (<цифра> |<буква> ) }

## **Литералы**

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

Таблица 1.3 – Описание литералов

|  |  |
| --- | --- |
| Тип литерала | Описание |
| Литералы целого типа | Целочисленные беззнаковые литералы, десятичное и шестнадцатеричное представления. Литералы только rvalue. |
| Строковые литералы | Состоит из символов латинского алфавита, заключенных в "…" (двойные кавычки). Только rvalue. |

<цифра> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<шестнадцатеричное число> ::= <цифра> | a | b | c | d | e | f

<целочисленный литерал> ::= (0x<шестнадцатеричное число><шестнадцатеричное число><шестнадцатеричное число><шестнадцатеричное число>)|({<цифра>})

## **Объявление данных и область видимости**

В языке GMP-2019 объявление данных начинается с ключевого слова var, указывается тип данных и имя идентификатора. Требуется обязательное объявление переменной перед её использованием.

Примеры: var ushort a, var line b;

Все переменные в языке GMP-2019 имеют область видимости, а именно префикс — название функции, в которой они находятся, что разрешает использование в различных функциях переменных с одинаковым именем. Параметры функции видны только внутри неё. Переменные, объявленные в одной функции, недоступны в другой.

## **Инициализация данных**

В момент объявления переменных в языке GMP-2019 происходит автоматическая инициализация в зависимости от типа данных. Инициализация другими значениями в момент объявления не допускается. Виды инициализации представлены в таблице 1.4.

Таблица 1.4 — Способы инициализации переменных

|  |  |
| --- | --- |
| Вид инициализации | Примечание |
| <идентификатор> = <значение>; | Присваивание переменной значения. |

## **Инструкции языка**

Инструкции языка GMP-2019 представлена в таблице 1.5.

Таблица 1.5 – Инструкции языка

|  |  |
| --- | --- |
| Инструкция | Форма записи |
| Объявление переменной | var <тип данных> <идентификатор> |
| Объявление функции | <тип данных> function <идентификатор> (<тип данных> <идентификатор>, …)  {  < программный блок>  return <идентификатор>|<литерал>.  } |
| Вызов функции | <идентификатор> (<идентификатор>, …) |
| Присвоение значения | <идентификатор> = <значение>; |
| Печать данных | print <литерал>|<идентификатор> |
| Функции стандартной библиотеки  (применяются при инициализации и в выражениях) | compare(line, line) — лексикографическое сравнение строк  pow(ushort, ushort) — возводит первый операнд типа ushort в степень, равную значению второго операнда типа ushort. (перенести) |
| Возвращаемое значение | return <литерал>|<идентификатор> |

## **Операции языка**

Операции сравнения, используются в условной конструкции, которые можно использовать в языке GMP-2019, представлены в таблице 1.6.

Таблица 1.6 — Операции языка

|  |  |
| --- | --- |
| Инструкция | Форма записи |
| Операции сравнения языка | > — бинарное больше  < — бинарное меньше  >= — бинарное больше либо равно  <= — бинарное меньше либо равно  == — бинарное равно  != — бинарное не равно |

## **Выражения и их вычисление**

В языке присутствуют выражения сравнения, использующиеся в условной конструкции

## **Программные конструкции языка**

Основные программные конструкции языка GMP-2019 представлены в таблице 1.7.

Таблица 1.7 — Основные конструкции языка

|  |  |
| --- | --- |
| Конструкция | Реализация |
| Главная функция  (точка входа) | main  {  <программный блок>  } |
| Функции | <тип данных> function <идентификатор>  (<тип данных> <идентификатор>, …)  {  <программный блок>  return <идентификатор>|<литерал>.  } |
| Условный оператор | if(<литерал>|<идентификатор><логический оператор><литерал>|<идентификатор>)  {  <программный блок>  }  else  {  <программный блок>  } |

## **Область видимости идентификаторов**

Область видимости «сверху вниз». В GMP-2019 требуется обязательное объявление переменной перед её использованием. Все переменные должны находиться внутри программного блока языка. Имеется возможность объявления одинаковых переменных в разных функциях. Каждая переменная получает префикс – название функции, в которой она объявлена

## **Семантические проверки**

В языке программирования GMP-2019 выполняются следующие семантические проверки:

1. Наличие функции main – точки входа в программу;

2. Единственность точки входа;

3. Переопределение идентификаторов;

4. Использование идентификаторов без их объявления;

5. Проверка соответствия типа функции и возвращаемого параметра;

6. Правильность передаваемых в функцию параметров: количество, типы;

7. Правильность строковых выражений;

8. Превышение размера строковых и числовых литералов;

9. Правильность составленного условия цикла/условного оператора.

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

Транслированный код использует две области памяти. В сегмент констант заносятся все литералы. В сегмент данных заносятся переменные и параметры функций. Локальная область видимости в исходном коде определяется за счет использования префиксов.

## **Стандартная библиотека и её состав**

В языке GMP-2019 предусмотрена стандартная библиотека. Функции, входящие в состав библиотеки, описаны в табл. 1.7. Стандартная библиотека подключается автоматически на этапе генерации кода.

Таблица 1.7 - Функции стандартной библиотеки языка GMP-2019

|  |  |
| --- | --- |
| Функция | Описание |
| compare(line a, line b) | Данная функция целочисленного типа, принимает два строковых параметра. Сравнивает строки и возвращает 1, если равны, 0, если a меньше b, 2, если a больше b |
| pow(ushort a, ushort b) | Данная функция целочисленного типа принимает два целочисленных параметра. Функция возводит число a в степень b и возвращает результат. |

## **Ввод и вывод данных**

Ввод данных языком программирования GMP-2019 не поддерживается.

Для вывода данных используется функция print(<имя идентификатора>|<литерал>). Пример: print(a);

## **Точка** **входа**

Функция точки входа в языке программирования GMP-2019 представлена в таблице 1.10.

Таблица 1.10 — Точка входа

|  |  |
| --- | --- |
| Конструкция | Реализация |
| Главная функция  (точка входа) | main  {  <программный блок>  } |

## **Препроцессор**

Препроцессор, принимающий и выдающий некоторые данные на вход транслятору, в языке GMP-2019 отсутствует.

## **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**

GMP-2019 транслируется в язык ассемблера.

## **Классификация сообщений транслятора**

Генерируемые транслятором сообщения определяют степень его информативности, то есть сообщения транслятора должны давать максимально полную информацию о допущенной пользователем ошибке при написании программы. Сообщения транслятора приведены в таблице 1.10, а также в приложении А.

Таблица 1.10 Классификация ошибок(диапазон)

|  |  |
| --- | --- |
| Префикс ошибки | Описание ошибки |
| LEX: ### | ### - код ошибки. Сообщение, генерируемое на этапе лексического анализа. Диапазон: 200-299 |
| SYN: ### | ### - код ошибки. Сообщение, генерируемое на этапе синтаксического анализа. Диапазон: 600-699 |
| SEM: ### | ### - код ошибки. Сообщение, генерируемое на этапе семантического анализа. Диапазон: 500-599 |
| SYS: ### | ### - код ошибки. Сообщение, генерируемое при критической ошибке. Диапазон: 0-99 |

## **Контрольный пример**

ushort function min(ushort a, ushort b)

{

var ushort result;

if (b < a) {

result = b;

} else {

result = a;

}

return result;

}

line function ismore(ushort a, ushort b)

{

var line result;

if (a > b) {

result = "First argument more than second";

} else {

result = "Second argument more than first";

}

return result;

}

main

{

[var line s234;]

var line str1;

var line str2;

var ushort a;

var ushort b;

var ushort c;

var ushort result;

b = 0x0011;

c = 18;

str1 = "Mikhail";

str2 = str1;

result = compare(str1, str2);

if(result == 1) {

a = min(b, c);

} else {

print("strings are not equal");

}

if(a >= b) {

b = pow(c, 2);

print(b);

}

str1 = ismore(b, 300);

print(str1);

return 0;

}

# **Глава 2. Структура транслятора**

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

Транслятор преобразует программу, написанную на языке GMP-2019 в программу на языке ассемблера. Компонентами транслятора являются лексический, синтаксический и семантический анализаторы, а также генератор кода на язык ассемблера. Принцип их взаимодействия представлен на рисунке 2.1.

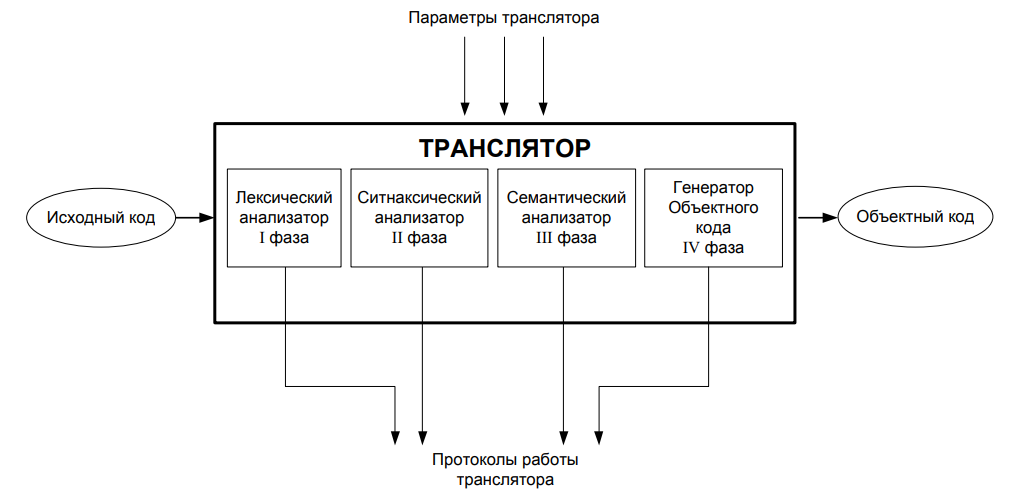


Рисунок 2.1 – Структура транслятора

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

Синтаксический анализатор – часть компилятора, выполняющая синтаксический анализ, то есть проверку исходного кода на соответствие правилам грамматики. Входной информацией для синтаксического анализа является таблица лексем и таблица идентификаторов. Выходной информацией является 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. Подробное описание представлено в 5 главе.

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 GMP-2019, прошедший все предыдущие этапы, в код на языке Ассемблера. Более полно описан в главе 7.

## **2.2 Перечень входных параметров транслятора**

Входные параметры представлены в таблице 2.1.

Таблица 2.1 Входные параметры транслятора языка GMP-2019

|  |  |  |
| --- | --- | --- |
| Входной параметр | Описание параметра | Значение по умолчанию |
| -in:<путь к in-файлу> | Файл с исходным кодом на языке GMP-2019, имеющий расширение .txt | Не предусмотрено |
| -log:<путь к log-файлу> | Файл журнала для вывода протоколов работы программы. | Значение по умолчанию:  <имя in-файла>.log |
| -out:<путь к out-файлу> | Выходной файл – результат работы транслятора. Содержит исходный код на языке асемблера. | Значение по умолчанию:  <имя in-файла>.asm |
| -notrace TODO |  |  | |

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

Таблица с перечнем протоколов, формируемых транслятором языка

GMP-2019 и их назначением представлена в таблице 2.2

Таблица 2.2 Протоколы, формируемые транслятором языка GMP-2019

|  |  |
| --- | --- |
| Формируемый протокол | Описание выходного протокола |
| Файл журнала, заданный параметром "-log:" | Файл с протоколом работы транслятора языка программирования GMP-2019. Содержит таблицу лексем и таблицу идентификаторов |
| Выходной файл, заданный параметром "-out:" | Результат работы программы – файл, содержащий исходный код на языке ассемблера. |

# **Глава 3. Разработка лексического анализатора**

## **3.1 Структура лексического анализатора**

Лексический анализатор – часть транслятора, выполняющая лексический анализ. Лексический анализатор принимает обработанный и разбитый на отдельные компоненты исходный код на языке GMP-2019. На выходе формируется таблица лексем и таблица идентификаторов. Структура лексического анализатора представлена на рисунке 3.1



Рисунок 3.1 Структура лексического анализатора GMP-2019

## **3.2 Контроль входных символов**

Таблица для контроля входных символов представлена на рисунке 3.2

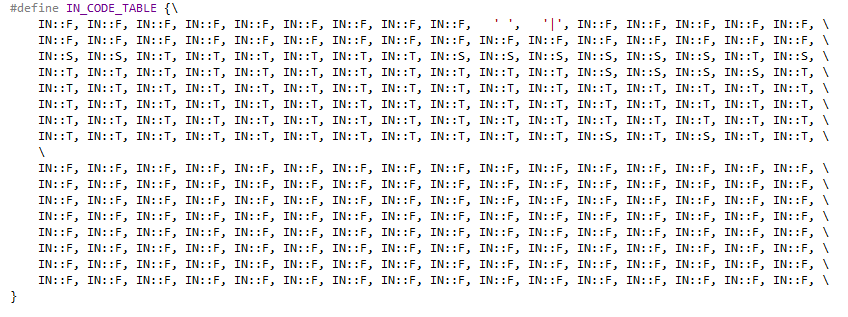


Рисунок 3.2. Таблица контроля входных символов

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

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

## **3.3 Удаление избыточных символов**

Избыточными символами являются символы табуляции и пробелы. Табуляция заменяется на пробел. Пробелы удаляются на этапе разбиения исходного кода на лексемы.

Описание алгоритма удаления избыточных символов:

1. Посимвольно считываем файл с исходным кодом программы.

2. Пробелы выступаю в роли сепаратора и пропускаются.

## **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 | Слова | Лексема |
| Ключевые слова | var | d |
| ushort | t |
| line | t |
| function | f |
| print | p |
| pow | b |
| compare | a |
| return | r |
| main | m |
| if | c |
| else | e |
|  |  |
| Иное | Идентификатор | i |
| Целочисленный литерал | l |
| Строковый литерал | l |

Продолжение таблицы 3.1

|  |  |  |
| --- | --- | --- |
| Сепараторы | ; | ; |
| , | , |
|  |  |
| { | { |
| } | } |
| ( | ( |
| ) | ) |
| Операторы | = | = |
| Логические операторы | o |

TODO Пример реализации таблицы лексем представлен в приложении А.

Также в приложении А находятся конечные автоматы, соответствующие лексемам языка GMP-2019.

## **3.5 Основные структуры данных**

Основные структуры таблиц лексем и идентификаторов данных языка GMP-2019, используемых для хранения, представлены в приложении TODO А. В таблице лексем содержится лексема, её номер, полученный при разборе, номер строки в исходном коде. В таблице идентификаторов содержится имя идентификатора, номер в таблице лексем, тип данных, смысловой тип идентификатора и его значение.

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

Перечень сообщений лексического анализатора представлен на рисунке 3.3.

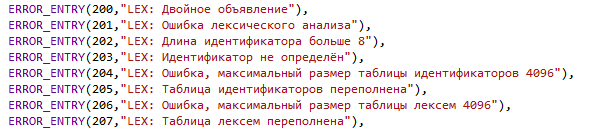


Рисунок 3.3 – Перечень ошибок лексического анализатора

## **3.7 Принцип обработки ошибок**

При возникновении ошибки транслятор завершает свою работу. Для обработки ошибок лексический анализатор использует таблицу с сообщениями. Структура сообщений содержит информацию о номере сообщения, номер строки и позицию, где было вызвано сообщение в исходном коде, информацию об ошибке.

## **3.8 Параметры лексического анализатора**

Входным параметром лексического анализа является структура, полученная после чтения входного файла на этапе проверки исходного кода на допустимость символов.

## **3.9 Алгоритм лексического анализа**

* проверяет входной поток символов программы на исходном языке на допустимость, удаляет лишние пробелы и добавляет сепаратор для вычисления номера строки для каждой лексемы;
* для выделенной части входного потока выполняется функция распознавания лексемы;
* при успешном распознавании информация о выделенной лексеме заносится в таблицу лексем и таблицу идентификаторов, и алгоритм возвращается к первому этапу;
* формирует протокол работы;
* при неуспешном распознавании выдается сообщение об ошибке.

Распознавание цепочек основывается на работе конечных автоматов. Работу конечного автомата можно проиллюстрировать с помощью графа переходов.

Пример. Регулярное выражение для ключевого слова main.

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

i

m

n

a

Рисунок 3.4 – Граф переходов для цепочки ‘body’

## **3.10 Контрольный пример**

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

# **Глава 4. Разработка синтаксического анализатора**

## **4.1 Структура синтаксического анализатора**

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



Рисунок 4.1 – Структура синтаксического анализатора

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

В синтаксическом анализаторе транслятора языка GMP-2019 используется контекстно-свободная грамматика , где

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

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

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

S – начальный символ грамматики, являющийся нетерминалом.

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

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

Грамматика языка GMP-2019 представлена в приложении Б.

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

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

Таблица 4.1 – Перечень правил, составляющих грамматику языка и описание нетерминальных символов GMP-2019

|  |  |  |
| --- | --- | --- |
| Нетерминал | Цепочки правил | Описание |
| S | S→m{NrE;};| tfi(F){NrE;};S | Порождает правила, описывающее общую структуру программы |
| N | N→dti;|i=E;|dti;N| i=E;N|p(l);N| p(i);N|p(l);| p(i);|  c(C){N}e{N}N| c(C){N}e{N}|  c(C){N}| c(C){N}N | Порождает правила, описывающие инструкции языка |
| E | E→i|l|(E)|i(W)| b(W)| a(W)| | Порождает правила, описывающие выражения |
| F | F→ti|ti,F | Порождает правила, описывающие параметры локальной функции при её объявлении |
| W | W→i|l|i,W|l,W | Порождает правила, описывающие параметры вызываемой функции |
| C | С→ioi|iol|loi|lol | Порождает правила, описывающие условия оператора if |

## **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 TODO В.

Таблица 4.2 – Описание компонентов магазинного автомата

|  |  |  |
| --- | --- | --- |
| Компонента | Определение | Описание |
|  | Множество состояний автомата | Состояние автомата представляет из себя структуру, содержащую позицию на входной ленте, номера текущего правила и цепочки и стек автомата |
|  | Алфавит входных символов | Алфавит является множеством терминальных и нетерминальных символов, описание которых содержится в разделе 1.2 и в таблице 4.1. |
|  | Алфавит специальных магазинных символов | Алфавит магазинных символов содержит стартовый символ и маркер дна стека |

Продолжение таблицы 4.2

|  |  |  |
| --- | --- | --- |
|  | 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 GMP-2019. Данные структуры представлены в приложении TODO В.

## **4.5 Описание алгоритма синтаксического разбора**

Принцип работы автомата:

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

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

Перечень сообщений синтаксического анализатора представлен на рисунке 4.1.

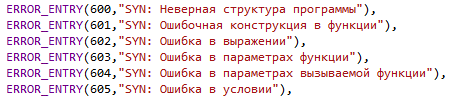


Рисунок 4.1 – Перечень сообщений синтаксического анализатора

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

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

Выходными параметрами являются трассировка прохода таблицы лексем (TODO при наличии разрешающего ключа) и правила разбора, которые выводятся на консоль.

## **4.8 Принцип обработки ошибок**

Синтаксический анализатор выполняет разбор исходной последовательности лексем до тех пор, пока не дойдёт до конца цепочки лексем или не найдёт ошибку. Тогда анализ останавливается и выводится сообщение об ошибке (если она найдена). Если в процессе анализа находятся более трёх ошибок, то анализ останавливается. После всей процедуры трассировки в протокол будет выведено диагностическое сообщение.

## **4.9 Контрольный пример**

Пример разбора синтаксическим анализатором исходного кода на языке GMP-2019 представлен в приложении TODO Г. Дерево разбора исходного кода также представлено в приложении Г.

# **Глава 5. Разработка семантического анализатора**

## **5.1 Структура семантического анализатора**

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

## **5.2 Функции семантического анализатора**

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

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

Сообщения, формируемые семантическим анализатором, представлены на рисунке 5.1.

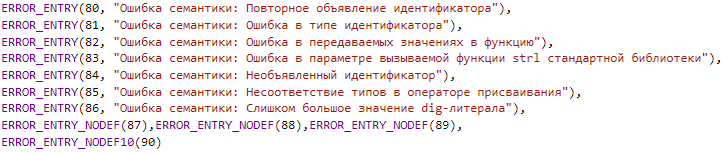


Рисунок 5.1 – Перечень сообщений семантического анализатора

## **5.4 Принцип обработки ошибок**

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

## **5.5 Контрольный пример**

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