МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ

ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра программной инженерии

Специальность 6-05-0612-01 Программная инженерия

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Разработка компилятора RPV-2024»

Выполнил студент Редько Павел Валерьевич

(Ф.И.О.)

Руководитель проекта стаж.-преп. Некрасова А.П.

(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 стаж.-преп. Некрасова А.П.

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

Нормоконтролер стаж.-преп. Некрасова А.П.

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

Курсовой проект защищен с оценкой

Минск 2024

Содержание

Введение 5

1. Спецификация языка программирования 6

1.1 Характеристика языка программирования 6

1.2 Определение алфавита языка программирования 6

1.3 Применяемые сепараторы 6

1.4 Применяемые кодировки 6

1.5 Типы данных 6

1.6 Преобразование типов данных 6

1.7 Идентификаторы 6

1.8 Литералы 6

1.9 Объявление данных 6

1.10 Инициализация данных 6

1.11 Инструкции языка 6

1.12 Операции языка 6

1.13 Выражения и их вычисление 6

1.14 Конструкции языка 6

1.15 Область видимости идентификаторов 6

1.16 Семантические проверки 6

1.17 Распределение оперативной памяти на этапе выполнения 6

1.18 Стандартная библиотека и ее состав 6

1.19 Ввод и вывод данных 6

1.20 Точка входа 6

1.21 Препроцессор 7

1.22 Соглашения о вызовах 7

1.23 Объектный код 7

1.24 Классификация сообщений транслятора 7

1.25 Контрольный пример 7

2. Структура транслятора 8

2.1 Компоненты транслятора их назначение и принципы взаимодействия 8

2.2 Перечень входных параметров транслятора 8

2.3 Протоколы, формируемые транслятором 8

3. Разработка лексического анализатора 9

3.1 Структура лексического анализатора 9

3.2 Контроль входных символов 9

3.3 Удаление избыточных символов 9

3.4 Перечень ключевых слов 9

3.5 Перечень ключевых слов 9

3.6 Структура и перечень сообщений лексического анализатора 9

3.7 Принцип обработки ошибок 9

3.8 Параметры лексического анализатора 9

3.9 Алгоритм лексического анализа 9

3.10 Контрольный пример 9

4. Разработка синтаксического анализатора 10

4.1 Структура синтаксического анализатора 10

4.2 Контекстно-свободная грамматика, описывающая синтаксис языка 10

4.3 Построение конечного магазинного автомата 10

4.4 Основные структуры данных 10

4.5 Описание алгоритма синтаксического разбора 10

4.6 Структура и перечень сообщений синтаксического анализа тора 10

4.7 Параметры синтаксического анализатора и режимы его работы 10

4.8 Принцип обработки ошибок 10

4.9 Контрольный пример 10

5. Разработка семантического анализатора 11

5.1 Структура семантического анализатора 11

5.2 Функции семантического анализатора 11

5.3 Структура и перечень сообщений семантического анализатора 11

5.4 Принцип обработки ошибок 11

5.5 Контрольный пример 11

6. Вычисление выражений 12

6.1 Выражения, допускаемые языком 12

6.2 Польская запись и принцип ее построения 12

6.3 Программная реализация обработки выражений 12

6.4 Контрольный пример 12

7. Генерация кода 13

7.1 Структура генератора кода 13

7.2 Представление типов данных в оперативной памяти 13

7.3 Статическая библиотека 13

7.4 Особенности алгоритма генерации кода 13

7.5 Входные параметры, управляющие генерацией кода 13

7.6 Контрольный пример 13

8. Тестирование транслятора 14

8.1 Общие положения 14

8.2 Результаты тестирования 14

9. Разработка и тестирование интерпретатора 15

9.1 Структура и перечень сообщений времени выполнения 15

Заключение 16

Список использованных источников 17

Приложение A 18

Введение

Целью курсового проекта поставлена разработка собственного языка программирования и транслятора для него. Язык программирования носит название RPV-2024 – первые буквы ФИО и год разработки.

Языком разработки транслятора был выбран язык С++. Среда разработки – Visual Studio Community 2022. Код на языке RPV-2024 будет транслироваться в язык JavaScript.

Для достижения поставленной цели, задание на курсовой проект можно разбить на следующие подзадачи:

1. Написание спецификации языка RPV-2024. Содержит в себе: определение синтаксиса и семантики языка; описание ключевых слов, операторов, типов данных и т. д.
2. Разработка лексического анализатора. Для описания лексики языка программирования применяются регулярные грамматики, относящиеся к типу 3 иерархии Хомского.
3. Разработка синтаксического анализатора. Его назначением является распознавание синтаксических конструкций языка и формирование промежуточного кода.
4. Разработка семантического анализатора. Основные действия семантического анализатора: проверка семантических правил исходного языка; дополнение внутреннего представления программы операторами и действиями, неявно предусмотренными семантикой исходного языка.
5. Вычисление выражений.
6. Генерация кода. В данном разделе описывается процесс генерации кода.
7. Тестирование компилятора. В данном разделе описывается разработка контрольного примера, демонстрирующего правильную работу компилятора.

Каждая из подзадач описывается в соответствующих главах пояснительной записки. Там же представлено решение данных подзадач.

1. Спецификация языка программирования
   1. Характеристика языка программирования

Язык программирования RPV-2024 является строго типизированным, не объектно-ориентированным, компилируемым. Поддерживает 3 типа данных: целочисленный (number), символьный (char) и строковый (string). Стандартная библиотека содержит функцию лексикографического сравнения строк и функцию возведения числа в степень.

* 1. Определение алфавита языка программирования

Язык программирования SES-2023 использует стандартную таблицу символов Windows-1251, в которой определены латинские и русские символы, символы-разделители, символы математических операций и специальные символы.

* 1. Применяемые сепараторы

Символы сепараторы служат в качестве разделителей операций языка. Сепараторы, используемые в языке программирования RPV-2024, приведены в таблице 1.1.

Таблица 1.1 – Применяемые сепараторы

|  |  |
| --- | --- |
| Символы | Назначение |
| (…) | Ограничение блока параметров функции |
| {…} | Ограничение тела функции |
| ; | Разделитель программных конструкций |
| пробел | Служит для разделения программных конструкций. Допускается везде, кроме идентификаторов и ключевых слов |
| , | Разделитель параметров функции |

* 1. Применяемые кодировки

Для написания исходного кода программы на языке RPV-2024 используются латинские символы и символы разделители кодировки Windows-1251. Русские символы допускается использовать в строковых и символьных литералах.

* 1. Типы данных

Допускается использование фундаментальных типов данных. В языке RPV- 2024 есть 3 типа данных: целочисленный, символьный и строковый. Описание ти-пов данных, предусмотренных в данным языке представлено в таблице 1.2.

Таблица 1.2 – Типы данных

|  |  |
| --- | --- |
| Тип данных | Описание |
| number | Размер в байтах: 4.  Допустимый диапазоны значений: от −2147483648 до 2147483647.  По умолчанию инициализируется 0.  Применяемые операции:  + – сложение;  - – вычитание;  == – равенство;  != – не равенство;  > – больше;  < – меньше;  >= – больше или равно;  <= – меньше или равно; |
| string | По умолчанию инициализируется пустой строкой: “”.  Размер в байтах: один символ – один байт.  Допустимый диапазоны значений: от 0 до 255. |
| char | По умолчанию инициализируется пустым символом: ‘’.  Размер в байтах: 1 байт. |

* 1. Преобразование типов данных

В языке программирования RPV-2024 отсутствует допущения преобразова-ния типов данных.

* 1. Идентификаторы

Для именования функций, параметров и переменных используются идентифи-каторы. Не предусмотрены зарезервированные идентификаторы. Имя идентифика-торов не должно совпадать с ключевыми словами языка и с именами функций стан-дартной библиотеки.

Имя идентификатора составляется по следующим образом:

* состоит из символов латинского алфавита [a..z];
* допускается использование цифр (идентификатор не может начинаться с цифры).
  1. Литералы

Литерал — это элемент программы, который непосредственно представляет значение.

В языке RPV-2024 существует 3 типа литералов: целочисленный, символьный и строковый. Язык поддерживает представление целочисленных литералов в вось-меричной, двоичной и десятичной системе счисления.

Краткое описание литералов представлено в таблице 1.3.

Таблица 1.3 – Описание литералов

|  |  |
| --- | --- |
| Литералы | Характеристика |
| Символьные | Символ алфавита языка заключенный в кавычки ‘’ |
| Целочисленные в двоичной форме | Последовательность цифр 0 и 1 с обязательным началом числа с b и необязательным началом с n(для отрицательных значений): b101 |
| Целочисленные в восьмеричной форме | Последовательность цифр 0…7 с обязательным началом числа с o и необязательным началом с n(для отрицательных значений): o23 |
| Целочисленные в десятичной форме | Последовательность цифр 0…9 и необязательным началом с n(для отрицательных значений) : 24242, n23 |
| Строковые | Символы алфавита языка заключенный в кавычки “” |

* 1. Объявление данных

Для объявления переменной используется ключевое слово define, после кото-рого необходимо указать тип данных и имя идентификатора. Конструкция для объ-явления переменных:

define <тип данных> <идентификатор>;

<идентификатор>=<литерал>|<идентификатор>;

* 1. Инициализация данных

При объявлении переменной допускается инициализация данных. Объектами-инициализаторами могут быть идентификаторы, литералы, выражения и вызов функции. По умолчанию переменные типа number инициализируются нулём, пере-менные типа char – пустым символом, а переменные типа string – пустой стро-кой.

* 1. Инструкции языка

В языке программирования RPV-2024 применяются инструкции, представ-ленные в таблице 1.4.

Таблица 1.4 – Инструкции языка

|  |  |
| --- | --- |
| Инструкция | Описание |
| Объявление переменной | define <тип данных><идентификатор>;  Пример: define string str1; |
| Операция присваивания | <идентификатор>=<литерал>|<идентификатор>;  Пример: num=7; |

Таблица 1.4(продолжение)

|  |  |
| --- | --- |
| Вывод данных | print(<идентификатор>|<литерал>);  Пример: print**(“**good morning”); print(x); |
| Вызов функции | <идентификатор>(<список параметров>);  Пример: sum(a, b); |
| Возврат значения функции | return <идентификатор>|<литерал>;  Пример: return result; |

* 1. Операции языка

Операции, которые можно использовать в языке RPV-2024, представлены в таблице 1.5.

Таблица 1.5 – Операции языка и их приоритеты

|  |  |  |  |
| --- | --- | --- | --- |
| Оператор | Характеристика | Приоритет | Применение |
| == | равенство | 3 | к типу number |
| != | не равенство | 3 | к типу number |
| > | больше | 3 | к типу number |
| < | меньше | 3 | к типу number |
| >= | больше или равно | 3 | к типу number |
| <= | меньше или равно | 3 | к типу number |
| + | сложение | 4 | к типу number |
| - | вычитание | 4 | к типу number |

* 1. Выражения и их вычисление

Круглые скобки в выражении используются для изменения приоритета опе-раций. Не допускается запись двух подряд идущих арифметических операций. Вы-ражение может содержать вызов функции.

* 1. Конструкции языка

Ключевые конструкции языка в программировании представляют собой ос-новные строительные блоки для создания программ.

Ключевые программные конструкции языка программирования RPV-2024 представлены в таблице 1.6.

Таблица 1.6 – Программные конструкции языка RPV-2024

|  |  |
| --- | --- |
| Конструкция | Представление в языке |
| Главная функция | main{  …  } |
| Функция | <тип данных> function <идентификатор> (<тип данных> <идентификатор>, …) {  …  return <идентификатор>|<литерал>;  }; |
| Условный оператор | if(<условие>) {  …  } |
| Цикл | cycle(<тип данных> <идентификатор>|<идентификатор>; <условие>; <выражение>) {  …  } |

* 1. Область видимости идентификаторов

Все переменные и параметры локальны и могут объявляться только в теле функции. Идентификаторы, объявленные в одной функции, недоступны любой дру-гой. Объявление глобальных переменных не предусмотрено.

* 1. Семантические проверки

Перечень семантических проверок, предусмотренных языком, приведен в таб-лице 1.7.

Таблица 1.7 – Перечень семантических проверок

|  |  |
| --- | --- |
| Номер | Правило |
| 1 | Идентификаторы функций не должны повторяться |
| 2 | Операнды в операторах ветвления и выхода из функции должны быть целочисленного типа |
| 3 | Тип данных передаваемых значений в функцию должен совпадать с типом параметров при её объявлении |
| 4 | Идентификатор должен быть объявлен до его использования. |
| 5 | Операнды в арифметическом выражении не могут быть различных типов |
| 6 | Тип возвращаемого функцией значения должен совпадать с типом функции |
| 7 | Не допустить выход за пределы диапазона значений у типов данных |

* 1. Распределение оперативной памяти на этапе выполнения

В языке RPV-2024 все переменные хранятся в куче.

* 1. Стандартная библиотека и ее состав

В языке RPV-2024 присутствует стандартная библиотека, которая подключа-ется автоматически при трансляции исходного кода в язык JavaScript. Содержимое библиотеки и описание функций представлено в таблице 1.8.

Таблица 1.8 – Стандартная библиотека для языка RPV-2024

|  |  |
| --- | --- |
| Функция | Характеристика |
| function lex\_compare(s1, s2) | Функция, выполняющая лексикографическое сравнение строк. Возвращает 1, если сравнение успешно иначе -1 |
| аunction power(number, to\_power) | Функция, выполняющая возведение числа в заданную степень. Возвращает полученное число. |

* 1. Ввод и вывод данных

Ввод данных не поддерживается. Вывод данных осуществляется с помощью ключевого слова print. В качестве аргумента принимаются литералы, идентификаторы и выражения: print(“something”); print(a); print(a + b);

* 1. Точка входа

В языке RPV-2024 точкой входа в языке является функция main. Точка входа не может отсутствовать или быть переопределена.

* 1. Препроцессор

В языке RPV-2024 не предусмотрена реализация препроцессора.

* 1. Соглашения о вызовах
  2. Объектный код

Исходный код языка RPV-2024 транслируется в язык JavaScript.

* 1. Классификация сообщений транслятора

Было определено 4 типа ошибок: ошибки лексического анализа, ошибки син-таксического анализа, ошибки семантического анализа и какие-либо системные ошибки.

Таблица 1.9 – Классификация сообщений транслятора

|  |  |
| --- | --- |
| Интервал | Описание ошибок |
| 0-1 | Системные ошибки |
| 100-112 | Ошибки параметров |
| 113-119 | Ошибки лексического анализа |
| 600-608 | Ошибки синтаксического анализа |
| 120-130 | Ошибки семантического анализа |

* 1. Контрольный пример

Контрольный пример на языке RPV-2024 приведен в приложении А.

1. Структура транслятора
   1. Компоненты транслятора их назначение и принципы взаимодействия

В языке RPV-2024 исходный код транслируется в код на языке JavaScript. Однако перед этим код проходит несколько этапов, в ходе которых формируются таблицы идентификаторов и лексем, необходимые для последующей генерации кода.

Таблицы создаются на этапе лексического анализа. Одной из задач этой фазы является максимальное упрощение кода путём замены его на набор лексем и удаления лишних символов. В данной реализации лексический анализатор частично выполняет функции семантического анализатора, проверяя в некоторых случаях корректность написанного кода.

Синтаксический анализатор — это этап транслятора, который проверяет исходный код на соответствие правилам грамматики. Входными данными для синтаксического анализа являются таблица лексем и таблица идентификаторов. Результатом его работы является дерево разбора.

Семантический анализатор — это часть транслятора, которая выполняет семантический анализ, то есть проверяет исходный код на наличие ошибок, которые невозможно выявить с помощью регулярной и контекстно-свободной грамматики. Входными данными для этого этапа также служат таблица лексем и таблица идентификаторов.

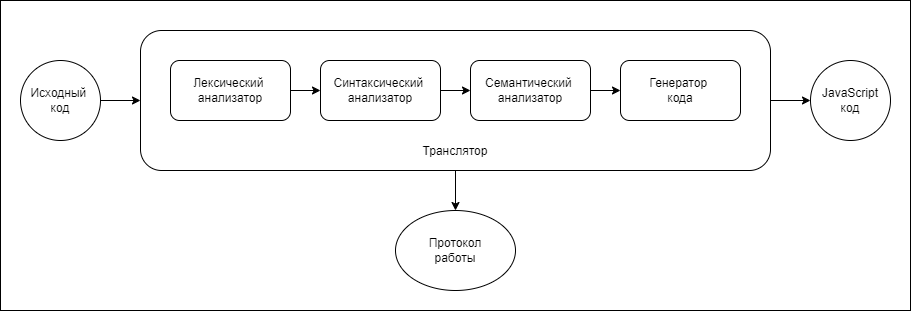
Генератор кода — это этап транслятора, который преобразует промежуточное представление кода в код на языке JavaScript. На вход генератора подаются таблица лексем и таблица идентификаторов. В результате работы генератора создаётся файл с расширением .js, содержащий представление исходного кода на языке JavaScript.

Рисунок 2.1 – Структура транслятора

* 1. Перечень входных параметров транслятора

В таблице 2.1 представлены входные параметры, которые могут быть использованы для управления работой транслятора.

Таблица 2.1 – Входные параметры транслятора языка SAV-2024

|  |  |  |
| --- | --- | --- |
| Параметр | Тип | Характеристика |
| -in: | Обязательный | Указывает на файл с исходным кодом. Исходный код содержится в файле с расширением \*.txt |
| -out: | Не обязательный | Указывает имя выходного файла. Если не указан явно, то имя протокола формируется конкатенацией имени файла исходного кода и постфикса «.out» |
| -log: | Не обязательный | Указывает имя протокола. Если не указан явно, то имя протокола формируется конкатенацией имени файла исходного кода и постфикса «.log» |

* 1. Протоколы, формируемые транслятором

Таблица с перечнем протоколов, формируемых транслятором языка RPV-2024 и их назначением представлена в таблице 2.2.

Таблица 2.2 – Протоколы, формируемые транслятором языка RPV-2024

|  |  |
| --- | --- |
| Протокол | Характеристика |
| Файл для вывода протокола работы программы log\_file.txt | Содержит возможные сообщения об ошибках |
| Выходной файл out\_file.txt | Содержит обработанный код на языке RPV-2024 |
| Lexems.txt | Содержит таблицу лексем |
| Identificators.txt | Содержит таблицу идентификаторов |
| program.js | Содержит сгенерированный код на языке JavaScript |

1. Разработка лексического анализатора
   1. Структура лексического анализатора

Лексический анализатор – это фаза транслятора выполняющая лексический анализ исходного кода. Схема работы лексического анализатора представлена на рисунке 3.1.

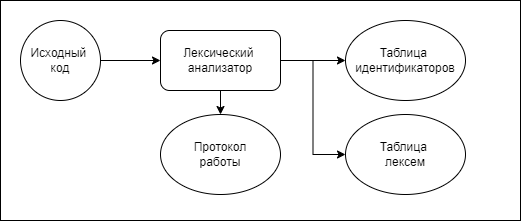


Рисунок 3.1 Структура лексического анализатора RPV-2024

Входные данные для лексического анализатора является обработанный и раз-битый на отдельные компоненты исходный код на языке RPV-2024. Выходные дан-ные: таблица лексем и таблица идентификаторов. Кроме того, формируется протокол работы анализатора.

* 1. Контроль входных символов

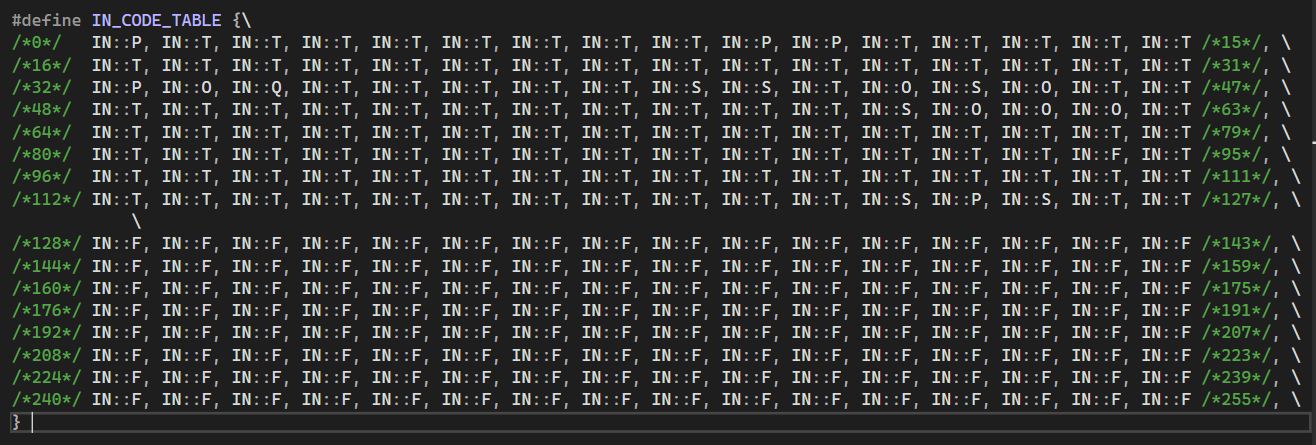
Таблица для контроля входных символов представлена на рисунке 3.2

Рисунок 3.2. Таблица контроля входных символов

Принцип работы таблицы заключается в соответствии значения каждому эле-менту в шестнадцатеричной системе счисления значению в таблице Windows- 1251.

Описание значения символов: T – разрешённый символ, F – запрещённый символ, S – сепаратор, P – пробелы, табуляция и переход на новую строку, O – операторы, Q – двойная кавычка.

* 1. Удаление избыточных символов

Избыточными символами являются символы табуляции и пробелы. Избыточ-ные символы удаляются на этапе разбиения исходного кода на токены. Все символы пробела и табуляции, которые не влияют на смысл исходного кода удаляются.

* 1. Перечень ключевых слов

Лексемы – это символы, соответствующие ключевым словам, символам опе-раций и сепараторам, необходимые для упрощения дальнейшей обработки исход-ного кода программы. Данное соответствие описано в таблице 3.1.

Таблица 3.1 – Соответствие ключевых слов, символов операций и сепараторов с лексемами

|  |  |  |  |
| --- | --- | --- | --- |
| Тип цепочки | Цепочка | | Лексема |
| Ключевые слова | define | | d |
| number | | t |
| string | | t |
| char | | t |
| function | | f |
| return | | t |
| print | | p |
| main | | m |
| if | | e |
| cycle | | c |
| Иное | Идентификатор | | i |
| Литерал | | l |
| Сепараторы | ; | ; | |
| , | , | |
| { | { | |
| } | } | |
| ( | ( | |
| ) | ) | |
| Операторы | + | v | |
| - | v | |
| > | v | |
| < | v | |
| <= | v | |
| >= | v | |
| == | v | |
| != | v | |
| = | = | |

* 1. Основные структуры данных

Основные структуры таблиц лексем и идентификаторов данных языка RPV- 2024, используемых для хранения, представлены в приложении Б. В таблице лек-сем содержится лексемы исходного кода. В таблице идентификаторов содержится имя идентификатора, номер в таблице лексем, тип данных, смысловой тип иденти-фикатора и его значение.

* 1. Структура и перечень сообщений лексического анализатора

Перечень сообщений лексического анализатора представлен в таблице 3.2. Также он генерирует ошибки и семантического анализа.

Таблица 3.2 – Перечень сообщений лексического анализатора

|  |  |
| --- | --- |
| Код | Сообщение |
| 113 | Ошибка на уровне лексического анализатора: элемент не распознан |
| 114 | Ошибка на уровне лексического анализатора: ошибка при создании файла таблицы идентификаторов |
| 115 | Ошибка на уровне лексического анализатора: ошибка при создании файла таблицы лексем |
| 116 | Ошибка на уровне лексического анализатора: ошибка при создании лексической таблицы (превышен максимальный размер) |
| 117 | Ошибка на уровне лексического анализатора: ошибка при создании таблицы идентификаторов (превышен максимальный размер) |
| 118 | Ошибка на уровне лексического анализатора: ошибка при получении строки лексической таблицы (нет элемента) |
| 119 | Ошибка на уровне лексического анализатора: ошибка при получении строки таблицы идентификаторов (нет элемента) |

* 1. Принцип обработки ошибок

Если возникает ошибка, то работа транслятора прекращается и диагностиче-ское сообщение будет выведено в log файл.

* 1. Параметры лексического анализатора

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

* 1. Алгоритм лексического анализа

Лексический анализ выполняется программой (входящей в состав транслято-ра), называемой лексическим анализатором. Цель лексического анализа — выделе-ние и классификация лексем в тексте исходной программы. Лексический анализатор производит распознаёт и разбирает цепочки исходного текста программы. Это ос-новывается на работе конечных автоматов, которую можно представить в виде графов. Регулярные выражения — аналитический или формульный способ задания регулярных языков. Они состоят из констант и операторов, которые определяют множества строк и множество операций над ними. Любое регулярное выражение можно представить в виде графа. И ещё она производит проверку на использование переменной перед объявлением и проверку на повторное объявление переменных.

* 1. Контрольный пример

Результат работы лексического анализатора (таблицы лексем и идентифика-торов) представлен в приложении Б.

1. Разработка синтаксического анализатора
   1. Структура синтаксического анализатора

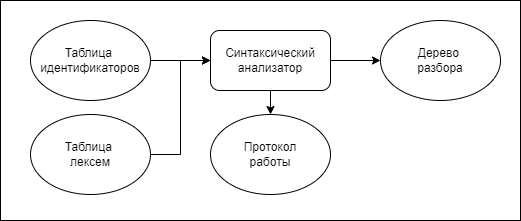
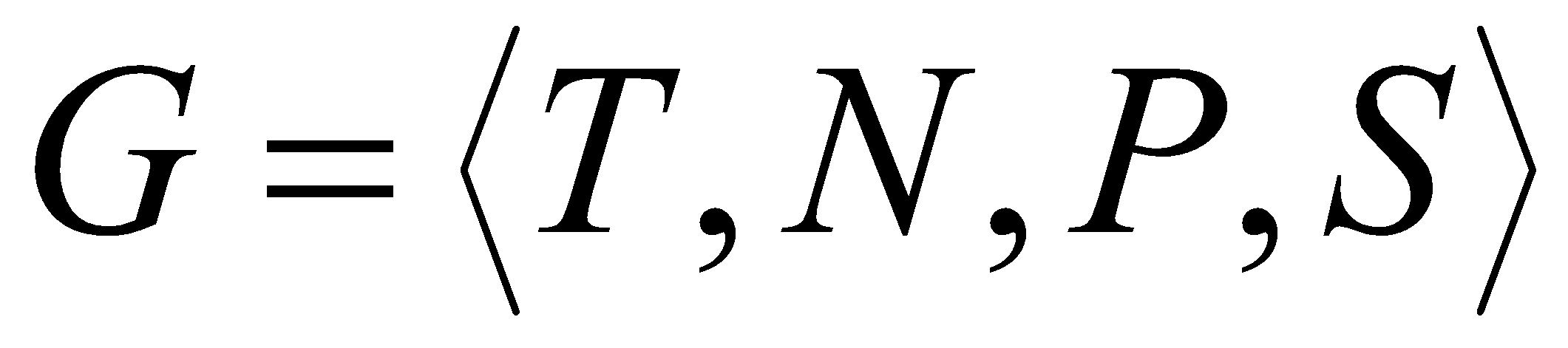
Синтаксический анализ – это фаза трансляции, выполняемая после лексиче-ского анализа и предназначенная для распознавания синтаксических конструкций.

Рисунок 4.1 – Структура синтаксического анализатора

Входные данные для синтаксического анализа является таблица лексем и таб-лица идентификаторов, полученные после фазы лексического анализа. Выходные данные: дерево разбора.

* 1. Контекстно-свободная грамматика, описывающая синтаксис языка

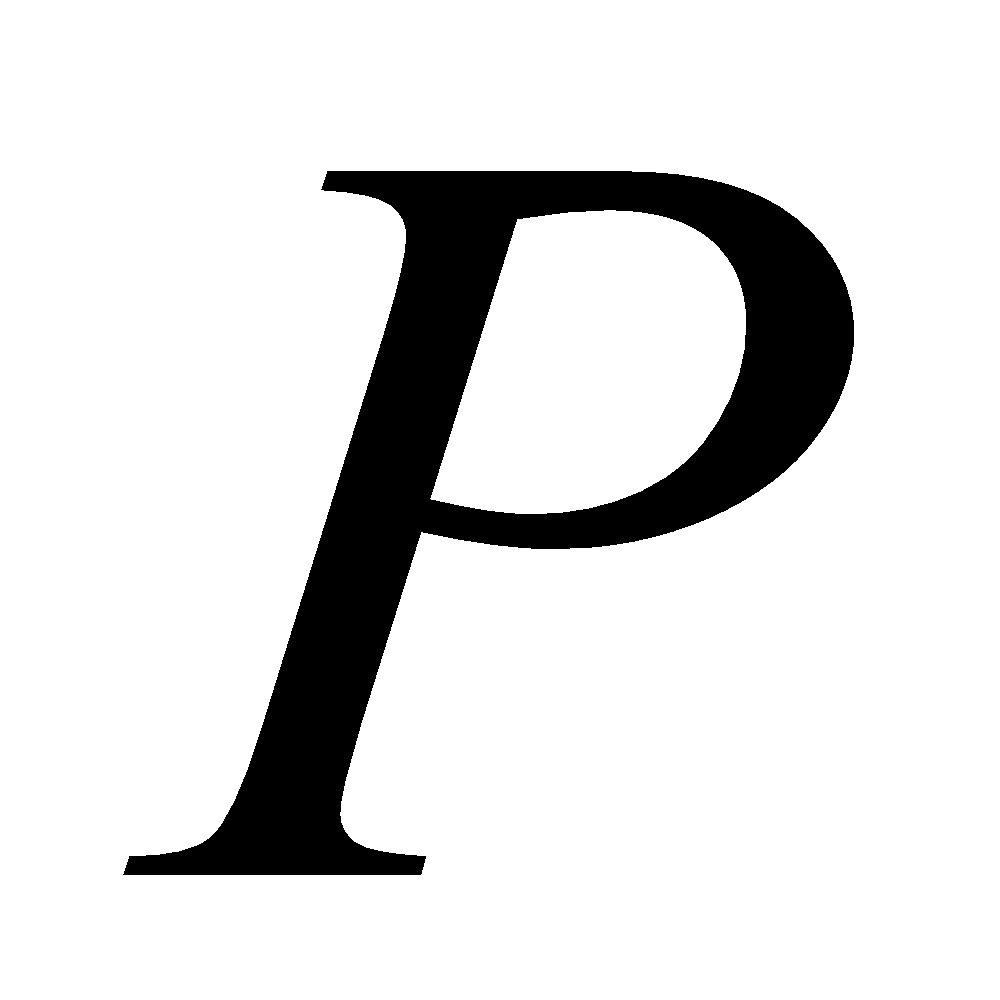
В синтаксическом анализаторе транслятора языка SAV-2024 используется контекстно-свободная грамматика , где

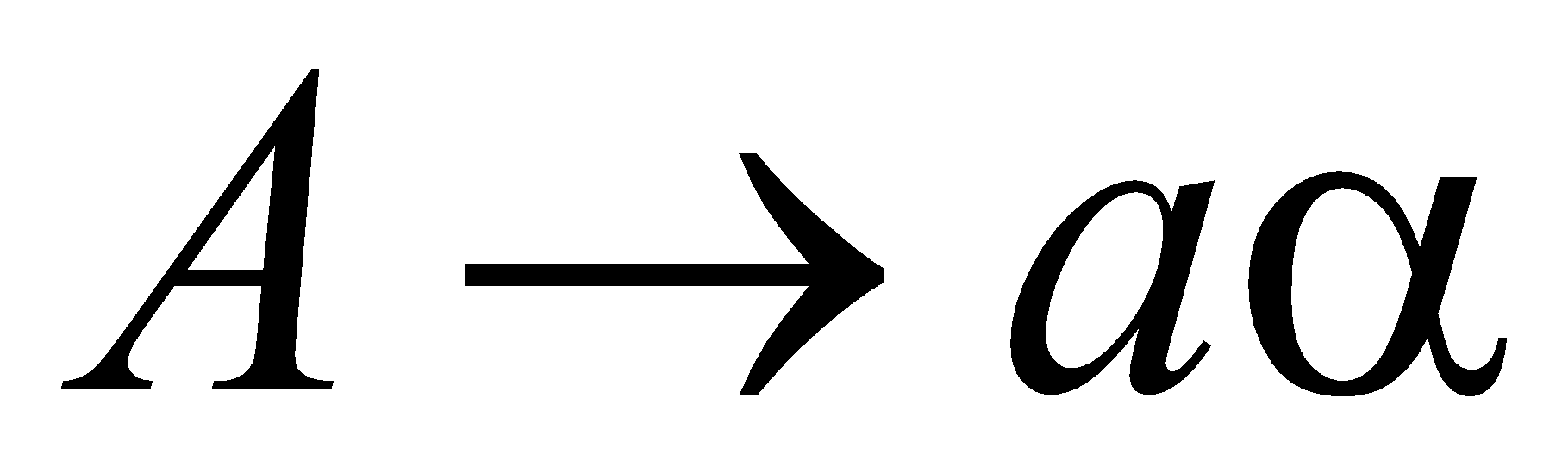
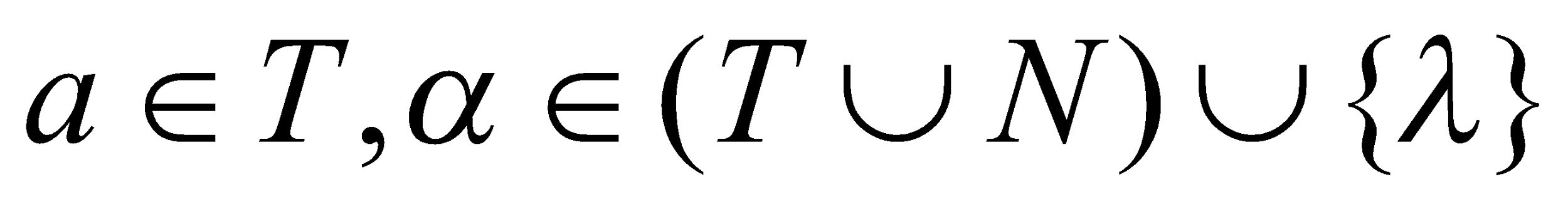
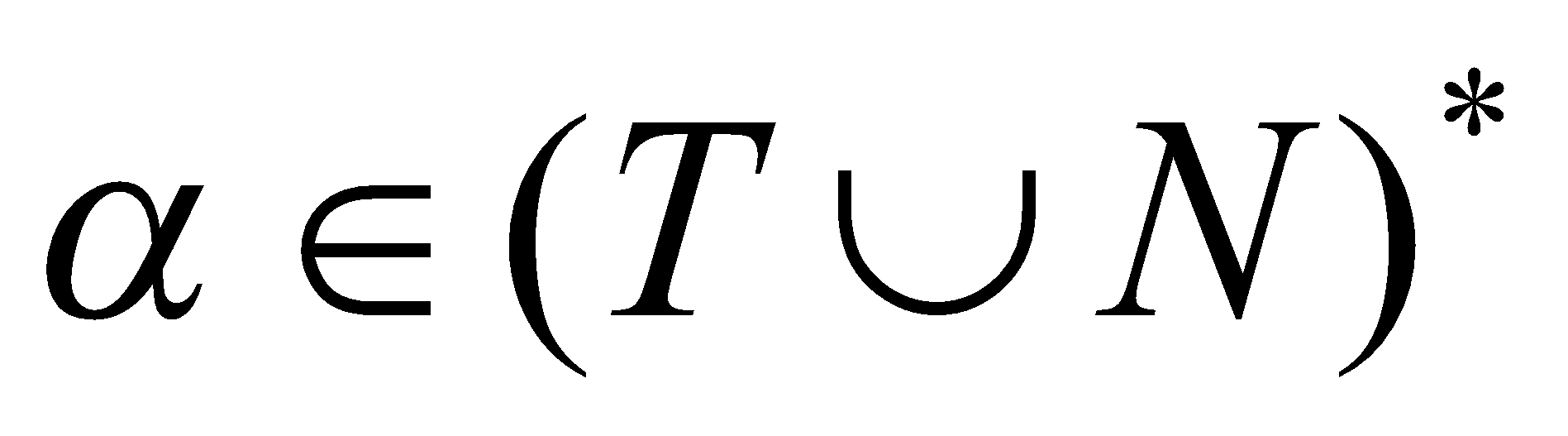
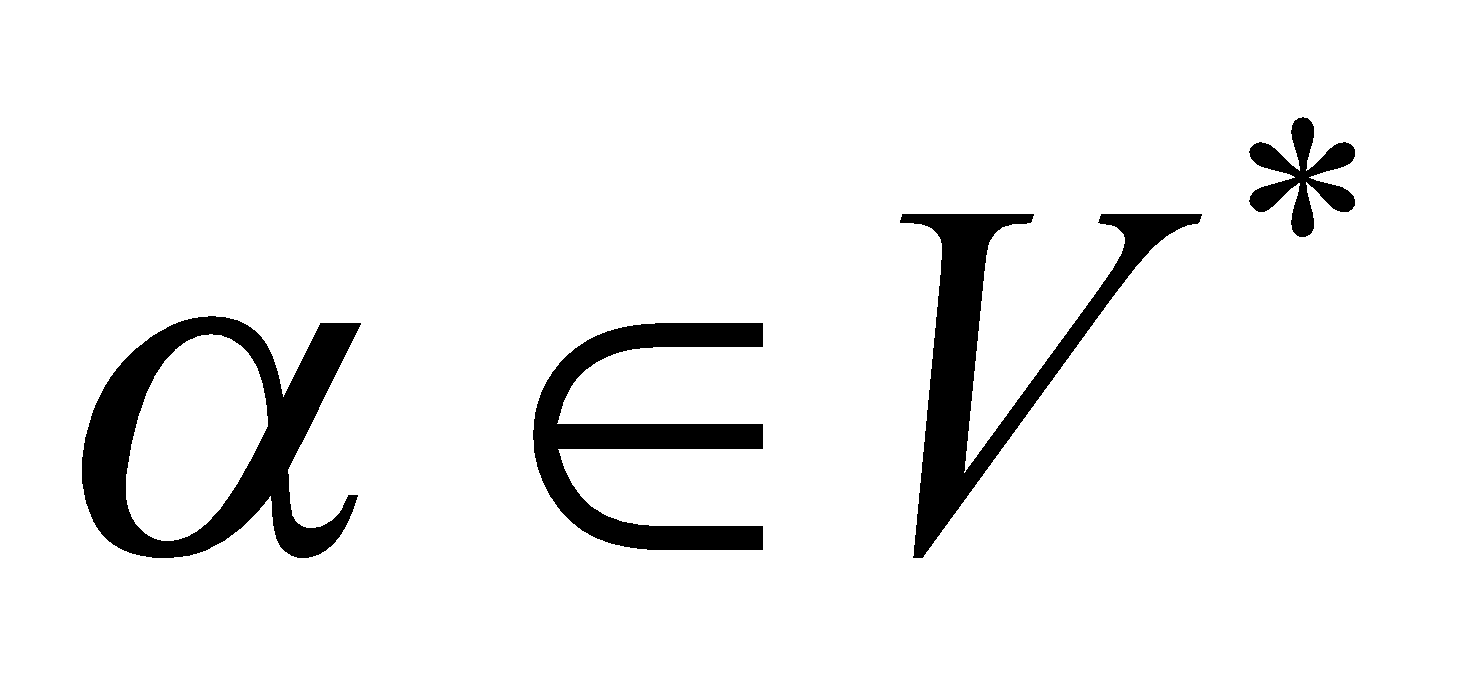
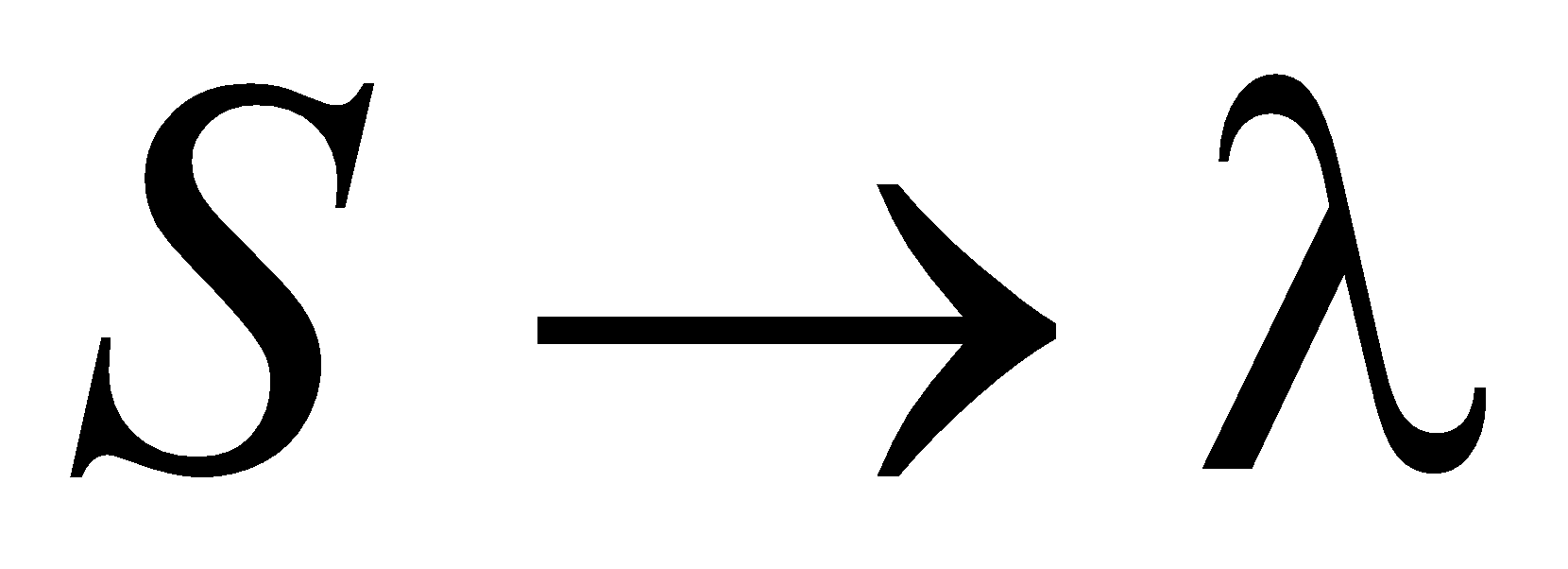
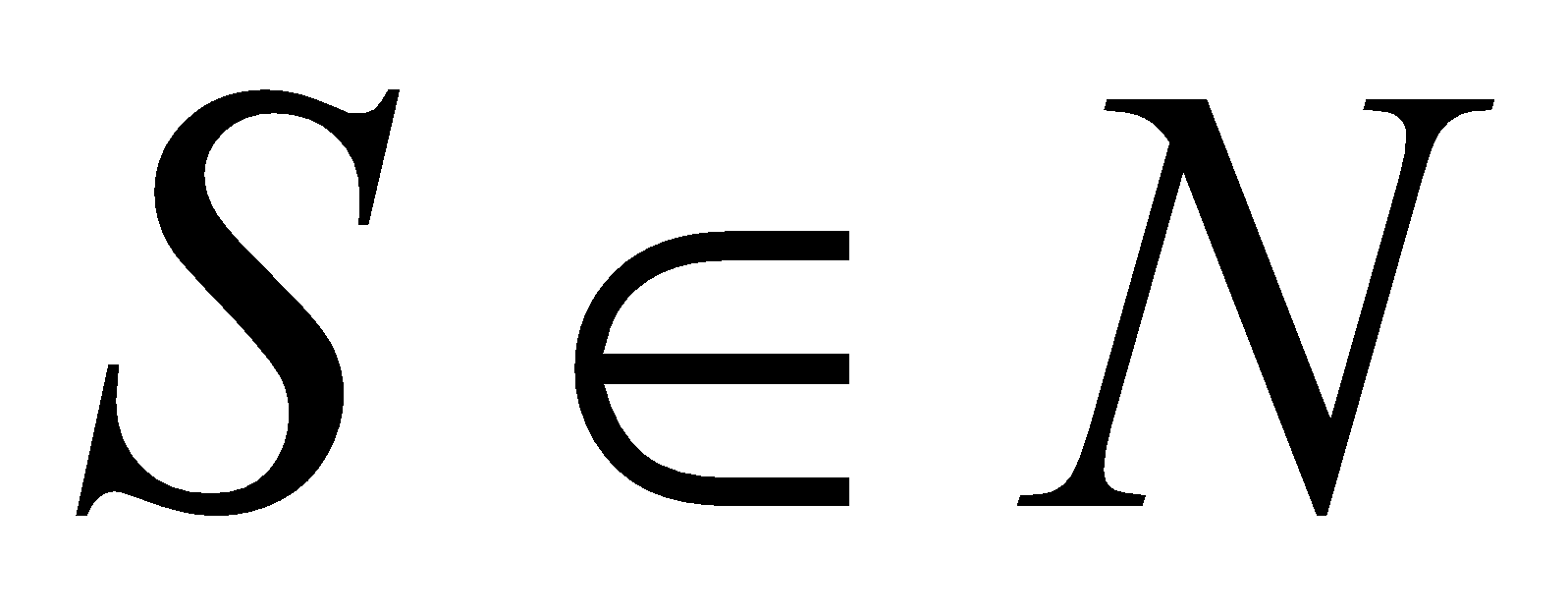
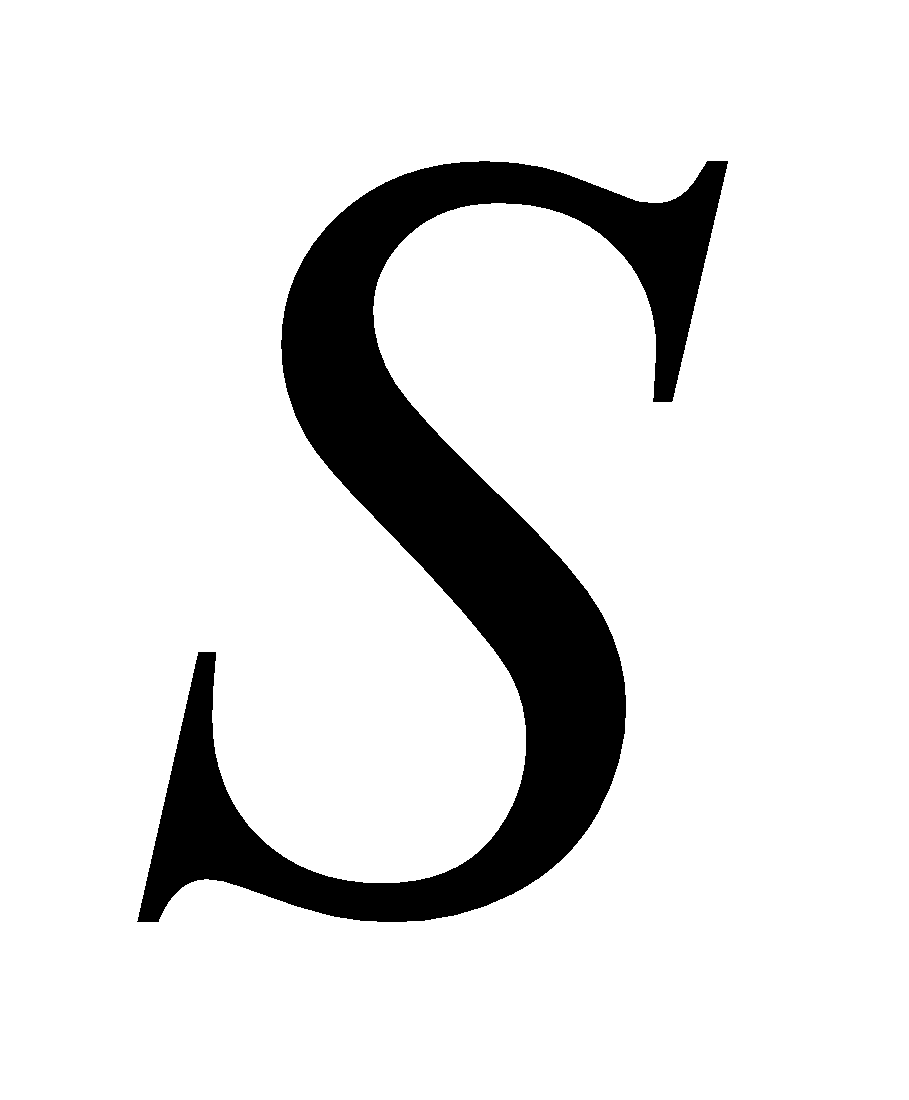
T – множество терминальных символов (было описано в разделе 1.2 данной пояснительной записки),

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.

Правила языка SAV-2024 представлена в приложении Б.

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

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

Таблица 4.1 – Перечень правил, составляющих грамматику языка и описание не-терминальных символов SAV-2024

|  |  |  |
| --- | --- | --- |
| Нетерминал | Цепочки правил | Описание |
| S | tfi(F){NrE;}S  tfi(F){rE;}S  m{N} | Порождает правила, описывающее общую структуру программы |
| N | dti;N  dti;  dti=E;N  dti=E;  i=E;N  i=E;  tfi(F){NrE;}N  tfi(F){rE;}N  tfi(F){NrE;}  tfi(F){rE;}  tfi(){NrE;}N  tfi(){rE;}N  tfi(){NrE;}  tfi(F){rE;}  e(L){N}N  e(L){N}  c(C){N}N  e(C){N}  p(E);  p(E);N | Порождает правила, описывающие инструкции языка |
| E | i  l  (E)  i()  i(W)  iM  lM  (E)M  i()M  i(W)M | Порождает правила, описывающие выражения |
| M | vE  vEM | Порождает правила, описывающие арифметические действия |
| F | ti  ti,F | Порождает правила, описывающие параметры шаблона функции |
| W | i  l  i,W  l,W | Порождает правила, описывающие параметры функции |

Таблица 4.1(продолжение)

|  |  |  |
| --- | --- | --- |
| C | dti;L;i=iM  i;L;i=iM | Порождает правила, описывающие цикл. |
| L | ivi  lvl  lvi  ivl  i  l | Порождает правила, описывающие условную конструкцию. |

* 1. 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 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 |
|  | Функция переходов автомата | Функция представляет из себя множество правил грамматики, описанных в таблице 4.1. |
|  | Начальное состояние автомата | Состояние, которое приобретает автомат в начале своей работы. Представляется в виде стартового правила грамматики (нетерминальный символ S) |
|  | Начальное состояние магазина автомата | Символ маркера дна стека ($) |
|  | Множество конечных состояний | Конечные состояние заставляют автомат прекратить свою работу. Конечным состоянием является пустой магазин автомата и совпадение позиции на входной ленте автомата с размером ленты |

* 1. Основные структуры данных

Основные структуры данных синтаксического анализатора включают в себя структуру магазинного автомата и структуру грамматики Грейбах, описывающей правила языка RPV-2024. Данные структуры представлены в приложении В.

* 1. Описание алгоритма синтаксического разбора

Принцип работы автомата следующий:

1. В магазин записывается стартовый символ;

2. На основе полученных ранее таблиц формируется входная лента;

3. Запускается автомат;

4. Выбирается цепочка, соответствующая нетерминальному символу, за-писывается в магазин в обратном порядке;

5. Если терминалы в стеке и в ленте совпадают, то данный терминал уда-ляется 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 син-таксический анализ выполнен успешно. После 3 исключений синтаксический анали-затор завершает работу и генерирует последнее исключение.

* 1. Структура и перечень сообщений синтаксического анализа тора

Перечень сообщений синтаксического анализатора представлен в таблице 4.3.

Таблица 4.3 – Перечень сообщений синтаксического анализатора

|  |  |
| --- | --- |
| Код | Сообщение |
| 600 | Ошибка на уровне синтаксического анализатора: неверная структура программы |
| 601 | Ошибка на уровне синтаксического анализатора: ошибочный оператор |
| 602 | Ошибка на уровне синтаксического анализатора: ошибка в выражении |
| 603 | Ошибка на уровне синтаксического анализатора: ошибка в операторах выражений |
| 604 | Ошибка на уровне синтаксического анализатора: ошибка в параметрах функции |
| 605 | Ошибка на уровне синтаксического анализатора: ошибка в параметрах вызываемой функции |
| 606 | Ошибка на уровне синтаксического анализатора: значением данного оператора может быть только переменная или литерал |
| 607 | Ошибка на уровне синтаксического анализатора: ошибка в условии условного оператора |

* 1. 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 и правила разбора, которые выводятся в консоль.

* 1. Принцип обработки ошибок

Обработка ошибок происходит следующим образом:

1. Синтаксический анализатор перебирает все правила и цепочки правила грамматики для нахождения подходящего соответствия с конструкцией, представ-ленной в таблице лексем.

2. Если невозможно подобрать подходящую цепочку, то генерируется со-ответствующая ошибка.

3. Все ошибки записываются в общую структуру ошибок.

4. В случае нахождения ошибки, после всей процедуры трассировки в протокол будет выведено диагностическое сообщение.

* 1. Контрольный пример

Пример разбора синтаксическим анализатором исходного кода на языке RPV-2024 представлен в приложении Г.

1. Разработка семантического анализатора
   1. Структура семантического анализатора

Семантический анализ происходит после выполнения фазы синтаксического анализа и реализуется в виде отдельных проверок текущих ситуаций в конкретных случаях и заполнении таблицы идентификаторов.

* 1. Функции семантического анализатора

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

* 1. Структура и перечень сообщений семантического анализатора

Сообщения, формируемые семантическим анализатором, представлены в таблице 5.1.

Таблица 5.1 – Перечень сообщений семантического анализатора

|  |  |
| --- | --- |
| Код | Сообщение |
| 120 | Ошибка на уровне семантического анализатора: невозможно применить оператор к данному типу операнда |
| 121 | Ошибка на уровне семантического анализатора: тип возвращаемого значения не соответствует типу функции |
| 122 | Ошибка на уровне семантического анализатора: использована необъявленная переменная |
| 123 | Ошибка на уровне семантического анализатора: повторное объявление имени |
| 124 | Ошибка на уровне семантического анализатора: функция должна возвращать значение |
| 125 | Ошибка на уровне семантического анализатора: ошибка в условии условного оператора |
| 126 | Ошибка на уровне семантического анализатора: длина строкового литерала превышает допустимое значение |
| 127 | Ошибка на уровне семантического анализатора: несоответствие типов в выражении |
| 128 | Ошибка на уровне семантического анализатора: несовпадение фактических и формальных параметров функции |
| 129 | Ошибка на уровне семантического анализатора: выход за предел целочисленного литерала |

* 1. Принцип обработки ошибок

Принцип обработки ошибок идентичен принципу обработки ошибок на этапе лексического анализа (раздел 3.7).

* 1. Контрольный пример

Демонстрации ошибок, диагностируемых семантическим анализатором на разных этапах трансляции приведена в разделе 8.2

1. Вычисление выражений
   1. Выражения, допускаемые языком

В языке RPV-2024 допускаются выражения, применимые к целочисленным типам данных. В выражениях поддерживаются арифметические операции, такие как +, -, операторы сравнения такие как >, <, >=, <=, == и !=, и вызовы функций как операнды выражений.

Приоритет операций представлен в таблице 6.1.

Таблица 6.1 – Приоритет операций в языке RPV-2024

|  |  |
| --- | --- |
| Приоритет | Операция |
| -3 | ; |
| -2 | , |
| -1 | ) |
| 1 | ( |
| 3 | >= |
| 3 | <= |
| 3 | != |
| 3 | == |
| 3 | > |
| 3 | < |
| 4 | + |
| 4 | - |

* 1. Польская запись и принцип ее построения

Выражения в языке SAV-2024 преобразовываются к обратной польской запи-си.

Польская запись – это альтернативный способ записи арифметических выра-жений, преимущество которого состоит в отсутствии скобок.

Обратная польская запись – это форма записи математических и логических выражений, в которой операнды расположены перед знаками операций.

Алгоритм построения:

* исходная строка: выражение;
* результирующая строка: польская запись;
* стек: пустой;
* результирующая строка: польская запись;
* исходная строка просматривается слева направо;
* операнды переносятся в результирующую строку в порядке их следования;
* операция записывается в стек, если стек пуст или в вершине стека лежит от-рывающая скобка;
* операция выталкивает все операции с большим или равным приоритетом в результирующую строку;
* запятая не помещается в стек, если в стеке операции, то все выбираются в строку;
* отрывающая скобка помещается в стек;
* закрывающая скобка выталкивает все операции до открывающей скобки, после чего обе скобки уничтожаются;
* закрывающая скобка, если поднят флаг функции, выталкивает все до откры-вающей скобки в обратном порядке и добавляет идентификатор функции в конце;
* по концу разбора исходной строки все операции, оставшиеся в стеке, вытал-киваются в результирующую строку.

Таблица 6.2 – Пример преобразования выражения в обратную польскую запись

|  |  |  |
| --- | --- | --- |
| Исходная строка | Результирующая строка | Стек |
| (a+b - c) - f(i) |  |  |
| +b-c) - f(i) | a | ( |
| b-c) - f(i) | a | (+ |
| -c) - f(i) | ab | (+ |
| c)-f(i) | ab | (+- |
| )-f(i) | ab+- |  |
| - f(i) | ab+- |  |
| f(i) | ab+-c\* | - |
|  | ab+-i@- |  |

* 1. Программная реализация обработки выражений

Программная реализация алгоритма преобразования выражений к польской записи представлена в приложении Д.

* 1. Контрольный пример

Пример преобразования выражения к польской записи представлен в таблице 6.2. Преобразование выражений в формат польской записи необходимо для по-строения более простых алгоритмов их вычисления.

1. Генерация кода
   1. Структура генератора кода

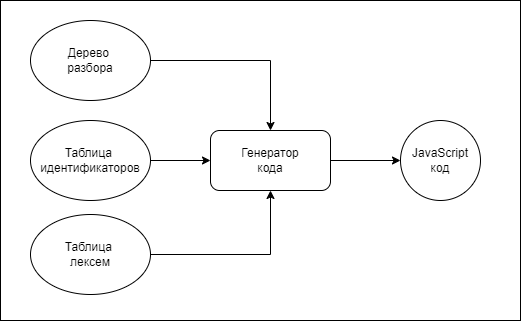
Заключительным этапом трансляции языка RPV-2024 является генерация ко-да. Генератор принимает на вход таблицу лексем, таблицу идентификаторов и де-рево разбора. На выходе получается файл с исходным кодом на JavaScript, который будет являться результатом работы транслятора. В случае возникновения ошибок генерация кода не будет осуществляться. Структура генератора кода представлена на рисунке 7.1.

Рисунок 7.1 – Структура генератора кода

* 1. Представление типов данных в оперативной памяти
  2. Статическая библиотека

В языке RPV-2024 предусмотрена статическая библиотека. Статическая библиотека содержит функции, написанные на языке JavaScript и автоматически подключается на начальном этапе генерации кода. Все функции статической библиотеки описаны в разделе 1.18.

* 1. Особенности алгоритма генерации кода

В языке RPV-2024 генерация кода строится на основе таблиц лексем и идентификаторов.

* 1. Входные параметры, управляющие генерацией кода

На вход генератору кода поступают таблицы лексем и идентификаторов ис-ходного кода программы на языке RPV-2024. Результат работы генератора кода выводятся в файл с расширением .js.

* 1. Контрольный пример

Результат генерации ассемблерного кода на основе контрольного примера из приложения А приведен в приложении Е.

1. Тестирование транслятора
   1. Общие положения

При возникновении ошибки на каком-либо этапе трансляции, она обрабаты-вается в главном файле программ: ошибка выводится на консоль и записывается в протокол работы.

* 1. Результаты тестирования

В таблице 8.1 приведены ошибки, генерируемые в процессе считывания входного файла, а также в процессе лексического, синтаксического и семантическо-го анализов.

Таблица 8.1 – Результаты тестирования транслятора

|  |  |
| --- | --- |
| Исходный код | Генерируемая ошибка |
| define number a = 0;  cycle (a2; a<=4; a=a+1) {  print(a);  } | Ошибка 113: Ошибка на уровне лексического анализатора: элемент не распознан, строка 8, позиция 8 |
| main {  k = 8;  …  } | Ошибка 122: Ошибка на уровне семантического анализатора: использована необъявленная переменная, строка 4, позиция 4 |
| define number b;  b = 11;  define number b; | Ошибка 123: Ошибка на уровне семантического анализатора: повторное объявление имени, строка 13 позиция 16 |
| define 4fg; | Ошибка 113: Ошибка на уровне лексического анализатора: элемент не распознан, строка 15, позиция 8 |
| … | … |

Заключение

В ходе проделанной работы был разработан транслятор для языка програм-мирования RPV-2024 на языке C++.

В пояснительной записке описана реализация поставленных в рамках курсо-вого проекта ряда задач:

* реализованы восемь операци` для работы с целочисленными данными;
* реализованы три типа данных;
* поддерживается оператор вывода;
* присутствует подключаемая стандартная библиотека;
* реализована условная конструкция;
* реализован оператор цикла;
* генерация выполнена в язык JavaScript;

Таким образом, разработанный транслятор соответствует требованиям зада-ния, обладает достаточной функциональностью для обработки программ на целе-вом языке, а также демонстрирует эффективность и удобство использования.

Список использованных источников

1. Страуструп, Б. Язык программирования C++ / Б. Страуструп. – 4-е изд. – М.: Бином, 2016. – 1376 с.

Приложение A

**Контрольный пример**

|  |
| --- |
| number function sum(number one, number two) {  return one + two;  }  main {  define number a = 0;  cycle (a; a<=4; a=a+1) {  print(a);  }    define number b;  b = 11;  if(b >= 3) {  print("number b >= 3");  }  if(b <= 4) {  print("number b <= 4");  }  if(b == 11) {  print("number b == 11");  }  print(sum(a, b));  define number b3;  b3 = power(b, 3);  print(b3);  define number nc = nb10010;  define number nd = no3471;  define number c = b10010;  define number d = o3471;  define string s1;  s1 = "ffss";  define string s2;  s2 = "ffssss";  define number for\_s;  for\_s = lex\_compare(s1, s2);  print(for\_s);  define char symb;  symb = 'd';  print(symb);  } |

Листинг 1 – Контрольный пример

**ПРИЛОЖЕНИЕ Б**

**Результат работы лексического анализатора и правила грамматики языка**

|  |
| --- |
| Таблица лексем  1. tfi(ti,ti){  2. rivi;  3. }  5. {  6. dti=l;  7. c(i;ivl;i=ivl){  8. p(i);  9. }  11. ti;  12. i=l;  13. e(ivl){  14. p(l);  15. }  16. e(ivl){  17. p(l);  18. }  19. e(ivl){  20. p(l);  21. }  23. (i(i,i));  25. ti;  26. i=i(i,l);  27. p(i);  29. ti=l;  30. dti=l;  31. dti=l;  32. dti=l;  34. ti;  35. i=l;  36. dti;  37. i=l;  38. dti;  39. i=i(i,i);  40. p(i);  42. ti;  43. i=l;  44. p(i);  45. } |

Листинг 1 – Таблица лексем

|  |
| --- |
| Таблица идентификаторов  Тип |Имя Тип данных Впервые в тл Значение  Функция |lex\_compare |NUM |-1  Функция |power |NUM |-1  Функция |sum |NUM |2  Параметр |one |NUM |5  Параметр |two |NUM |8  Переменная |a |NUM |21  Литерал |0 |NUM |23 |0  Литерал |4 |NUM |31 |4  Литерал |1 |NUM |37 |1  Переменная |b |NUM |48  Литерал |11 |NUM |52 |11  Литерал |3 |NUM |58 |3  Литерал |"number b >= 3" |STR |63 |"number b >= 3"  Литерал |"number b <= 4" |STR |76 |"number b <= 4"  Литерал |"number b == 11" |STR |89 |"number b == 11"  Переменная |b3 |NUM |105  Переменная |nc |NUM |123  Литерал |nb10010 |NUM |125 |-18  Переменная |nd |NUM |129  Литерал |no3471 |NUM |131 |-1849  Переменная |c |NUM |135  Литерал |b10010 |NUM |137 |18  Переменная |d |NUM |141  Литерал |o3471 |NUM |143 |1849  Переменная |s1 |STR |147  Литерал |"ffss" |STR |151 |"ffss"  Переменная |s2 |STR |155  Литерал |"ffssss" |STR |159 |"ffssss"  Переменная |for\_s |NUM |163  Переменная |symb |CHAR |181  Литерал |'d' |CHAR |185 |'d' |