

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение  
высшего образования

**«Дальневосточный федеральный университет»**

**ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ (ШКОЛА)**

**Департамент программной инженерии и искусственного интеллекта**

РАЗРАБОТКА ТРАНСЛЯТОРА, ПЕРЕВОДЯЩЕГО ПОДМНОЖЕСТВО ЯЗЫКА PERL В ЭКВИВАЛЕНТНОЕ ПОДМНОЖЕСТВО ЯЗЫКА C++

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ**

по дисциплине «Теория языков программирования и компиляторы»

по образовательной программе подготовки бакалавров по направлению 09.03.04 Программная инженерия

Выполнили студенты

гр. Б9120-09.03.04прогин

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Югалдина Ю.К.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Черепанова И. А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Селютин А. В.

Руководитель:

ассистент департамента ПИиИИ

Симаков В.К.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

(подпись)

г. Владивосток

2023 г

**Оглавление**

[Введение 4](#_Toc137091147)

[1 Неформальная постановка задачи 5](#_Toc137091148)

[2 Синтаксис входного языка 5](#_Toc137091149)

[3 Контекстные условия языка PERL 5](#_Toc137091150)

[4 Контекстные условия языка C++ 5](#_Toc137091151)

[5 Таблица соответствия языков 5](#_Toc137091152)

[6 Проект лексического анализатора 6](#_Toc137091153)

[6.1 Модель данных 6](#_Toc137091154)

[6.2 Таблица ключевых слов 6](#_Toc137091155)

[6.3 Таблица зарезервированных имен 6](#_Toc137091156)

[6.4 Типы лексем 6](#_Toc137091157)

[6.5 Типы ошибок 6](#_Toc137091158)

[6.6 Конечный автомат лексического анализатора 7](#_Toc137091159)

[7 Проект семантического анализатора 8](#_Toc137091160)

[7.1 Модель данных 8](#_Toc137091161)

[7.2 Описание алгоритма X семантического анализа 8](#_Toc137091162)

[7.3 Пример дерева разбора 8](#_Toc137091163)

[7.4 Типы ошибок 8](#_Toc137091164)

[8 Проект семантического анализатора 9](#_Toc137091165)

[8.1 Модель данных (если есть) 9](#_Toc137091166)

[8.2 Методы проверки нарушений контекстных условия 9](#_Toc137091167)

[8.3 Типы ошибок 9](#_Toc137091168)

[9 Проект генератора кода 10](#_Toc137091169)

[9.1 Модель данных (если есть) 10](#_Toc137091170)

[9.2 Метод кодогенерации 10](#_Toc137091171)

[9.3 Примеры кодогенерации 10](#_Toc137091172)

[10 Тестирование 11](#_Toc137091173)

[10.1 Лексический анализатор 11](#_Toc137091174)

[10.2 Синтаксический анализатор 11](#_Toc137091175)

[10.3 Семантический анализатор 11](#_Toc137091176)

[10.4 Генератор кода 11](#_Toc137091177)

[Заключение 12](#_Toc137091178)

# Введение

Транслятор – это программа ЭВМ, предназначенная для автоматического перевода описания алгоритма с одного языка программирования на другой, в частности на машинный язык.

Язык программирования — формальная знаковая система, предназначенная для записи компьютерных программ. Язык программирования определяет набор лексических, синтаксических и семантических правил, задающих внешний вид программы и действия, которые выполнит исполнитель (компьютер) под её управлением.

Для написания транслятора нужно было выбрать входной и выходной язык программирования. В качестве входного языка был выбран язык программирования Perl. Для выходного – C++.

**Цель курсового проекта:** разработать проект транслятора, который преобразует программу, содержащую подмножество языка PERL, в программу, содержащую подмножество языка C++ генерируя эквивалентный исходный код.

**Задачи:**

1. описать грамматику подмножества входного языка;
2. описать контекстные условия входного и выходного языков;
3. описать соответствие конструкций входного и выходного языков;
4. разработать проект лексического анализатора;
5. разработать проект синтаксического анализатора;
6. разработать проект семантического анализатора;
7. разработать проект генератора кода выходного языка;
8. реализовать транслятор входного языка в выходной;
9. выполнить тестирование разработанного транслятора.

# 1 Неформальная постановка задачи

Тут нужно написать про то, что нужно было сделать и что нужно сделать коротко. Говорим какое подмножество входного языка мы выбрали.

Подмножество языка PERL включает в себя

1. типы данных:

* Скаляр – простые переменные им предшествует знак $. (В Perl нет определения типов данных как в Си. В языке Perl они могут быть числом или строкой символов)
* Массив (предшествует знак @)

2. операторы:

* Арифметические: +, -, \*, /, \*\* (возведение в степень).
* Логические: ! (отрицание), && (и), || (или).
* Сравнения: <, >, <=, >=, it (возвращает true, если левый аргумент меньше по строке, чем правый аргумент), gt (возвращает истину, если левый аргумент по строке больше, чем правый аргумент), le ( возвращает true, если левый аргумент меньше или равен правому аргументу), ge (возвращает true, если левый аргумент по строке больше или равен правому аргументу).
* Равенства: ==, !=, <=> (возвращает -1, 0 или 1 в зависимости от того, является ли левый аргумент численно меньшим, равным или большим, чем правый аргумент)

3. операторы циклов: while;

4. операторы ветвления: if-elsif-else;

5. стандартные функции: print, abs, sqrt;

6. объявление функций: sub;

7. ключевые слова: while, return, sub.

# 2 Синтаксис входного языка

<PROGRAM> ::= <INSTRUCT> | <INSTRUCTIONS>

<INSTRUCT> ::= <ROOT> | <NESTED>

<ROOT> ::= <FUNCTION > <SEMICOLON>

<FUNCTION > ::= SUB <NAME\_FUNCT> ( <FORMAL\_PARAMS>) LPR < BLOCK > RPR

<BLOCK>::= <NESTED\_STMT BLOCK> | <NESTED\_STMT>

<NESTED\_STMT> ::= <COMPOUND\_STMT><SEMICOLON> | <SIMPLE\_STMT> <SEMICOLON>

<COMPOUND\_STMT> ::= <IF\_STMT> | <FOR\_STMT> | < WHILE\_STMT >

<IF\_STMT> ::= <IF expression LPR < BLOCK > RPR | IF expression LPR < BLOCK > RPR ELSE LPR < BLOCK > RPR>

<WHILE\_STMT> ::= <WHILE expression LPR < BLOCK > RPR >

<FOR\_STMT> ::= FOR ( id <SEMICOLON> expression SEMICOLON> increment) LPR < BLOCK > RPR

<SIMPLE\_STMT> ::= <ASSIGNMENT> | <EXPRESSION> | <RETURN\_STMT>

<ASSIGNMENT>::= <ID ASSIGN expression >

<RETURN\_STMT> ::= < RETURN expression>

< EXPRESSION > ::= < expression >

<SEMICOLON> ::= ";"

# 3 Контекстные условия языка PERL

Контекстные условия о правилах описания идентификаторов в программах:

* Регистр символов в идентификаторе имеет значения;
* Идентификатором считается любая последовательность латинских букв или цифр, начинающаяся с буквы. Буквой считается также символ подчеркивания "\_";
* Некоторые слова являются *ключевыми*, служат для оформления конструкций языка и не могут использоваться как идентификаторы, ряд слов является *контекстно ключевыми*(они являются ключевыми только в некотором контексте);
* Идентификатор не может переноситься со строки на строку.
* Идентификаторы могут быть любого типа, такие как переменные, функции, модули, классы и т.д. В Perl тип идентификатора определяется контекстом, в котором он используется.

Контекстные условия о правилах использования идентификаторов в своей области действия:

* Идентификатор не может быть использован при его объявлении (кроме рекурсивного метода);
* Использующее вхождение идентификаторов может быть в любой конструкции.

Контекстные условия, определяющие правила соответствия видов величин:

* Когда переменная используется в скалярном контексте, Perl ожидает, что переменная будет содержать единственное скалярное значение, такое как число, строка или ссылка. Любой другой вид данных (список, хэш и т. д.) будет приведен к скаляру, например, длина списка или хэша будет возвращена.

# 4 Контекстные условия языка C++

Контекстные условия о правилах описания идентификаторов в программах:

* Все используемые в программах идентификаторы должны быть описаны до их использования в программе;
* Каждый из идентификаторов, используемых в программе, может быть описан несколько раз только в том случае, если все различные описания этого идентификатора имеют разные области действия (блок программы, модулю программы, подпрограммы, описание в формальных параметрах процедур и функций и т.д.);
* Регистр символов в идентификаторе имеет значения;
* Идентификатором считается любая последовательность латинских букв или цифр, начинающаяся с буквы. Буквой считается также символ подчеркивания "\_";
* Некоторые слова являются *ключевыми*, служат для оформления конструкций языка и не могут использоваться как идентификаторы, ряд слов является *контекстно ключевыми*(они являются ключевыми только в некотором контексте);
* Идентификатор не может переноситься со строки на строку.

Контекстные условия о правилах использования идентификаторов в своей области действия:

* Идентификатор не может быть использован при его объявлении (кроме рекурсивного метода);
* Использующее вхождение идентификаторов может быть в любой конструкции.

Контекстные условия, определяющие правила соответствия видов величин:

* Количество объявленных в методе или функции параметров должно совпадать с количеством переданных;
* Неявные преобразования при использовании разных типов запрещены;
* Использование и описание массивов (и других типов) должно соответствовать выделенной им области памяти.

# 5 Таблица соответствия языков

В таблице 1 приведено сравнение языков Perl и C++.

Таблица 1 – Cсравнительная характеристика языков Perl и C++

|  |  |
| --- | --- |
| **Perl** | **C++** |
| 1. Структура программы | 1. Структура программы |
| 2. Циклы  2.1 for (<модификатор>; <условие>; <модификация>){  <блок>  }  2.2 while (<условие>) {  <блок>  }  end;  2.3 do {  <блок>} until(<условие>) | 2. Циклы  2.1 for (<модификатор>; <условие>; <модификация>){  <блок>  }  2.2 while (<условие>){  <блок>  }  2.3 do{  <блок>  }while (<условие>); |
| 3. Ветвления  3.1 if (<условие>){  <блок>  }else{  <блок>  }  3.2  switch (<модификатор>){  case <значение>{  <блок>}  …  case <значение>{  <блок>}  else {  <блок>  } | 3. Ветвления  3.1 if (<условие>){  <блок>  }else{  <блок>  }  3.2  switch (<модификатор>){  case <значение>:  <блок>  break;  …  case <значение>:  <блок>  break;  default:  <блок>  break;  } |

# 6 Проект лексического анализатора

Лексический анализ – процесс получения из текста лексем, т. е. слов. В свою очередь, лексер – это подпрограмма, которая позволяет провести лексический анализ. Для этого лексер должен уметь выполнять следующие задачи:

1. Выделять слова (лексемы) из текста;

2. Отделять лексемы от разделительных символов (пробелы, табуляция и т. п.). Следует также отметить:

• Не всегда пробелы (и прочие символы) являются разделительными символами;

• Иногда между лексемами нет разделителей;

• "Лишние" разделяющие символы удаляются;

3. Удалять комментарии.

## 6.1 Модель данных

Были реализованы классы Token и Lexer.

Класс Token – класс, определяющий токен – пару вида «лексема-тип данных».

Класс Lexer – класс, позволяющий разбить текст на токены.

## 6.2 Таблица ключевых слов

|  |  |
| --- | --- |
| if | if |
| else | else |
| while | while |
| for | for |
| do | do |
| in | in |

## 6.3 Таблица зарезервированных имен

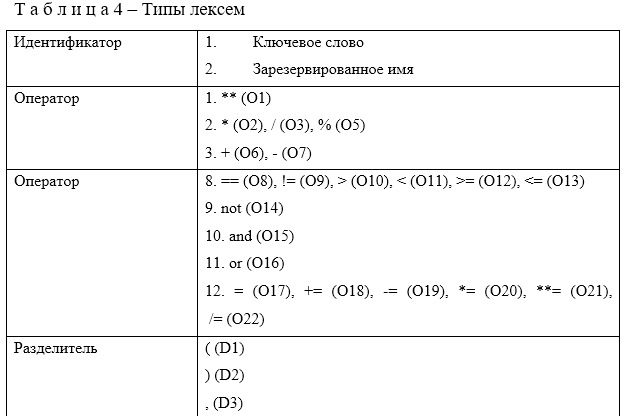
Что, зачем, в чем отличие от ключевых слов и других лексем. Ниже пример таблицы

Изображение выглядит как стол

Автоматически созданное описание

## 6.4 Типы лексем

Аналогично (что, зачем). Ниже пример таблицы (кусочек) (название токенов в первом столбце вы даете сами, а справа - представители токена)

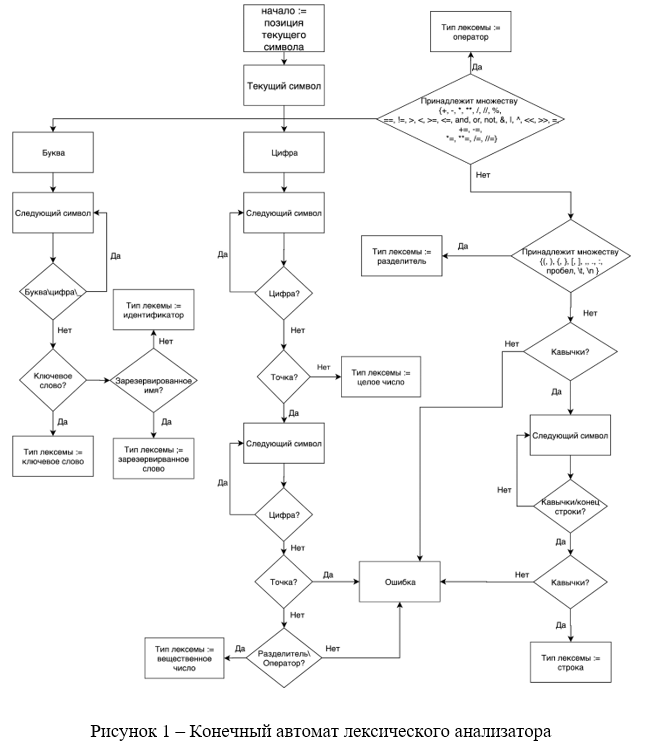
****

## 6.5 Типы ошибок

|  |  |  |
| --- | --- | --- |
| Тип ошибки | Входные данные | Выходные данные |
| Неверный оператор |  |  |
| Неверное объявление переменной  (нет закрывающей кавычки) |  |  |

## 6.6 Конечный автомат лексического анализатора

Аналогично что зачем и диаграмму автомата, можно в виде блок-схемы как ниже, лучше в виде диаграммы конечного автомата – она будет короче. Еще нужно привести пример его работы на примере какой-нибудь (демонстрирующей возможности языка) программы на входном языке. Можно привести пример вывода своего лексера.

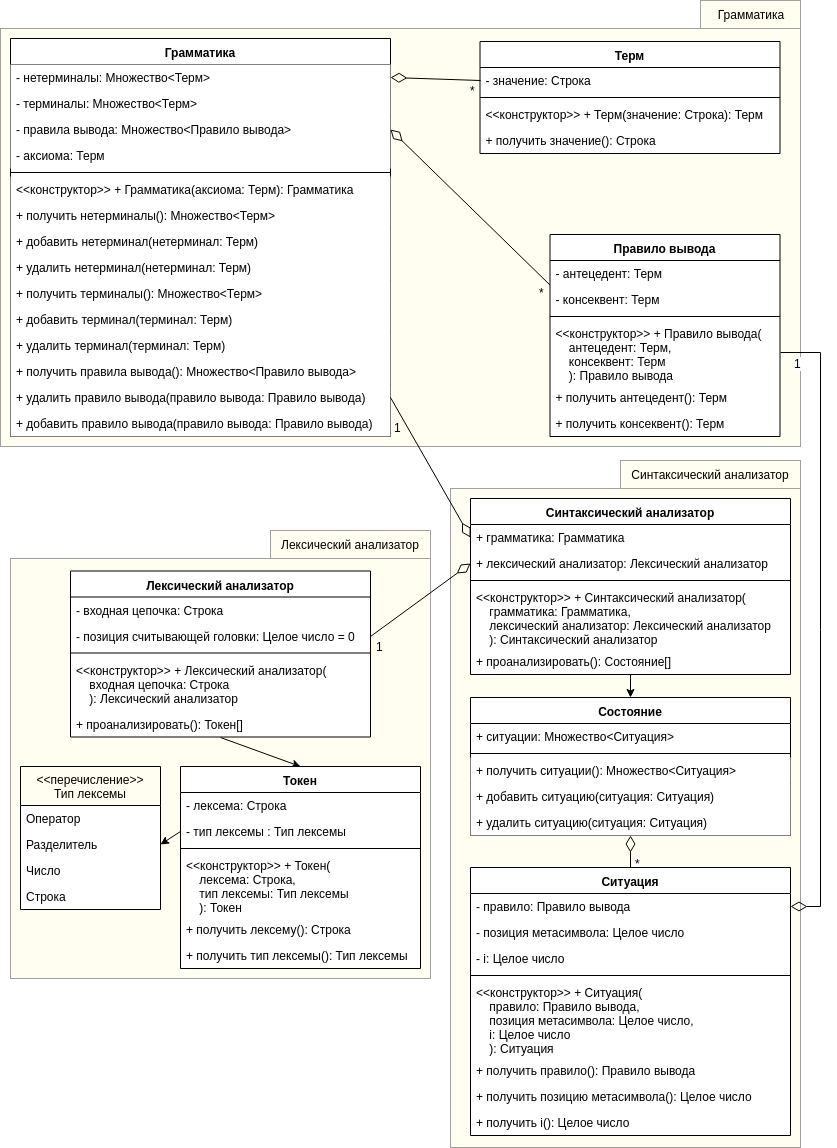


# 7 Проект семантического анализатора

Что зачем. Какой алгоритм синтаксического анализа выбран

## 7.1 Модель данных

Средствами чего реализовывали, какой язык для этого выбрали (для написания синтаксического анализатора), привести диаграммы моделей данных. Показать как моделируется дерево разбора. Ниже пример диаграммы.



## 7.2 Описание алгоритма X семантического анализа

Тут понятно. Просто распишите как в лекциях

## 7.3 Пример дерева разбора

Тут постройте по алгоритму дерево разбора – можете вставить вывод вашего парсера (семантического анализатора)

## 7.4 Типы ошибок

|  |  |  |
| --- | --- | --- |
| Тип ошибки | Входные данные | Выходные данные |
| Нету «точки с запятой» в конце выражения | $a = "lll" |  |
| Неправильное присваивание значения переменной | Изображение выглядит как Шрифт, текст, снимок экрана, часы  Автоматически созданное описание |  |

# 8 Проект семантического анализатора

Аналогично 6

## 8.1 Модель данных (если есть)

Аналогично 6

## 8.2 Методы проверки нарушений контекстных условия

Как проверяются ошибки типов и тд (контекстных условий)

## 8.3 Типы ошибок

|  |  |  |
| --- | --- | --- |
| Тип ошибки | Входные данные | Выходные данные |
| Не объявлена перемена «р», которая используется в выражении |  |  |
| Нету индикатора объявления у переменной, раньше не используемой |  |  |

# 9 Проект генератора кода

## 9.1 Модель данных (если есть)

## 9.2 Метод кодогенерации

## 9.3 Примеры кодогенерации

|  |  |
| --- | --- |
| Входные данные | Выходные данные (Ожидаемый результат) |
| $a = 10; | void main(){  int a = 10;  } |

# 10 Тестирование

## 10.1 Лексический анализатор

## 10.2 Синтаксический анализатор

## 10.3 Семантический анализатор

## 10.4 Генератор кода

# Заключение

Даже эмбрион знает, как писать заключение :)