

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 KOTLIN В ЭКВИВАЛЕНТНОЕ ПОДМНОЖЕСТВО ЯЗЫКА C#

**КУРСОВОЙ ПРОЕКТ**

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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Воронова Д.В.,

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Исаков А.В.,

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Мельникова Е.А.

Руководитель:

ассистент департамента ПИиИИ

Симаков В.К.

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

(подпись)

г. Владивосток

2023 г

**Оглавление**

[Введение 4](#_Toc129079426)

[1 Неформальная постановка задачи 5](#_Toc129079427)

[2 Синтаксис входного языка 5](#_Toc129079428)

[3 Контекстные условия языка X\_LANG 5](#_Toc129079429)

[4 Контекстные условия языка Y\_LANG 5](#_Toc129079430)

[5 Таблица соответствия языков 5](#_Toc129079431)

[6 Проект лексического анализатора 6](#_Toc129079432)

[6.1 Модель данных 6](#_Toc129079433)

[6.2 Таблица ключевых слов 6](#_Toc129079434)

[6.3 Таблица зарезервированных имен 6](#_Toc129079435)

[6.4 Типы лексем 6](#_Toc129079436)

[6.5 Типы ошибок 6](#_Toc129079437)

[6.6 Конечный автомат лексического анализатора 7](#_Toc129079438)

[7 Проект семантического анализатора 8](#_Toc129079439)

[7.1 Модель данных 8](#_Toc129079440)

[7.2 Описание алгоритма X семантического анализа 8](#_Toc129079441)

[7.3 Пример дерева разбора 8](#_Toc129079442)

[7.4 Типы ошибок 8](#_Toc129079443)

[8 Проект семантического анализатора 9](#_Toc129079444)

[8.1 Модель данных (если есть) 9](#_Toc129079445)

[8.2 Методы проверки нарушений контекстных условия 9](#_Toc129079446)

[8.3 Типы ошибок 9](#_Toc129079447)

[9 Проект генератора кода 10](#_Toc129079448)

[9.1 Модель данных (если есть) 10](#_Toc129079449)

[9.2 Метод кодогенерации 10](#_Toc129079450)

[9.3 Примеры кодогенерации 10](#_Toc129079451)

[10 Тестирование 11](#_Toc129079452)

[10.1 Лексический анализатор 11](#_Toc129079453)

[10.2 Синтаксический анализатор 11](#_Toc129079454)

[10.3 Семантический анализатор 11](#_Toc129079455)

[10.4 Генератор кода 11](#_Toc129079456)

[Заключение 12](#_Toc129079457)

# Введение

**Транслятор** — программа или техническое средство, преобразовывающее программу, представленную на одном из языков программирования, в программу, написанную на другом языке.

Язык программирования — это формальная знаковая система, на которой пишут компьютерную программу.

В данном курсовом проекте в качестве входного языка был выбран язык Kotlin, в качестве выходного – C#.

**Цель курсового проекта:** разработать проект транслятора, который преобразует программу, содержащую подмножество языка X\_LANG, в программу, содержащую подмножество языка Y\_LANG 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

В ходе данного курсового проекта необходимо разработать транслятор, который будет переводить подмножество языка Kotlin в подмножество языка C#. Для этого нужно проанализировать и описать входной и выходной язык, разработать лексический, синтаксический и семантический анализаторы, а также генератор кода выходного языка. На основе этих шагов нужно реализовать и протестировать транслятор.

Все переменные в языке Kotlin являются объектами в том смысле, что можно вызывать функции-члены и свойства любых переменных.

Некоторые типы могут иметь специальное внутреннее представление – например, числа, символы и логические значения могут быть представлены как примитивные значения во время выполнения, – но для пользователя они выглядят как обычные классы.

Таким образом, подмножество языка Kotlin включает в себя такие типы данных, как Integer, Float, Double, Boolean, character, string, array. String делится на Escaped String и Raw String. Integer включает в себя Byte, Short, Int, Long. Кроме того, Kotlin предоставляет еще unsigned integer типы данных, а именно UByte, UShort, UInt, ULong. Чтобы объявить переменную, тип данных которой не будет меняться, нужно использовать оператор val, и var в противном случае. Для приведения одного типа данных к другому используется оператор as.

Массивы в языке создаются с помощью функции arrayOf() или синонимов этой функции. Узнать длину массива можно при помощи свойства size, а входящие в массив индексы при помощи indices. Перебор можно сделать с использованием цикла for или forEach.

Списки объявляются с помощью функции listOf(). С помощью функций add() и remove() в них можно добавлять или удалять элементы. Доступ к отдельному элементу списка можно получить по номеру индекса через квадратные скобки. Также можно использовать метод get() с указанием номера индекса.

В подмножестве языка Kotlin содержатся логические операции дизъюнкции, конъюнкции, отрицания и стандартные арифметические операции сложения, вычитания, умножения, деления и деления по модулю. Из унарных операций есть логическое отрицание, унарный плюс и минус. Поддерживаются также операции инкремента и декремента.

В языке Kotlin есть такие виды циклов, как for, while и do-while. Язык поддерживает традиционные операторы break и continue для выхода из них.

В подмножестве языка Kotlin есть два оператора ветвления: if для одиночного ветвления и when – для множественного. Оба ветвления могут иметь как блочную, так и строчную структуру.

Функции в Kotlin объявляются ключевым словом fun. Стандартными функциями являются различные математические функции, например abs(), ceil(), floor(), max(), min(), round(), sqrt(), cbrt(), exp(). log(), pow(), sign() и тригонометрические функции. Также к стандартным относятся функции для работы с вводом и выводом, то есть print(), println(), readLine().

# 2 Синтаксис входного языка

Тут нужно написать про какие обозначения используются ниже, например для переноса строки или табуляции. Описываем грамматику входного языка (<PROGRAM> ::= <INSTRUCT> | <INSTRUCTIONS> и тд и тп.). Можно добавлять подразделы для описания некоторых конструкций, если вам так удобнее.

# 3 Контекстные условия языка X\_LANG

Тут нужно написать про контекстные условия входного языка, мы это писали в первом семестре для одной из задач.

# 4 Контекстные условия языка Y\_LANG

Аналогично

# 5 Таблица соответствия языков

Тут нужно написать про то, о чем этот раздел. Этот раздел о том, как соответствуют отдельные лексемы одного языка другом, как соответствует структура одного языка другому (это нужно для того, чтобы показать что и куда транслятор переводит). Надо описать это наиболее общим способом. Ниже есть пример (кусочек).

Изображение выглядит как стол

Автоматически созданное описание

# 6 Проект лексического анализатора

Тут нужно написать что делает лексер, что такое лексический анализ, о разделах ниже и тд.

## 6.1 Модель данных

Какие классы/структуры использовали, почему. Диаграммы классов можно вставить, если есть. Описать как представлены лексемы и токены в памяти

## 6.2 Таблица ключевых слов

Что такое ключевые слова, чем они отличаются от остальных лексем и тд. Надо написать, что ниже есть таблица с ключевыми словами :) Ниже есть пример такой таблицы.

Изображение выглядит как стол

Автоматически созданное описание

## 6.3 Таблица зарезервированных имен

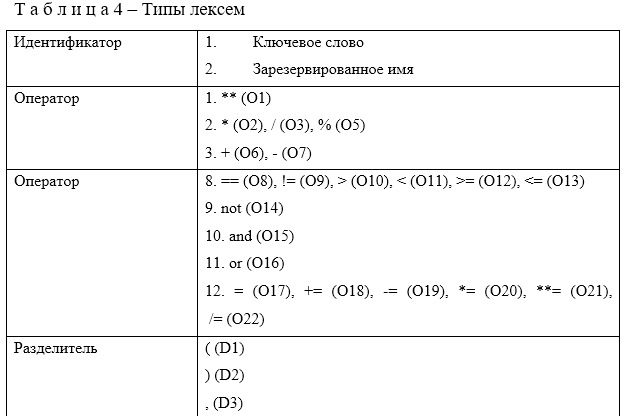
Что, зачем, в чем отличие от ключевых слов и других лексем. Ниже пример таблицы

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)

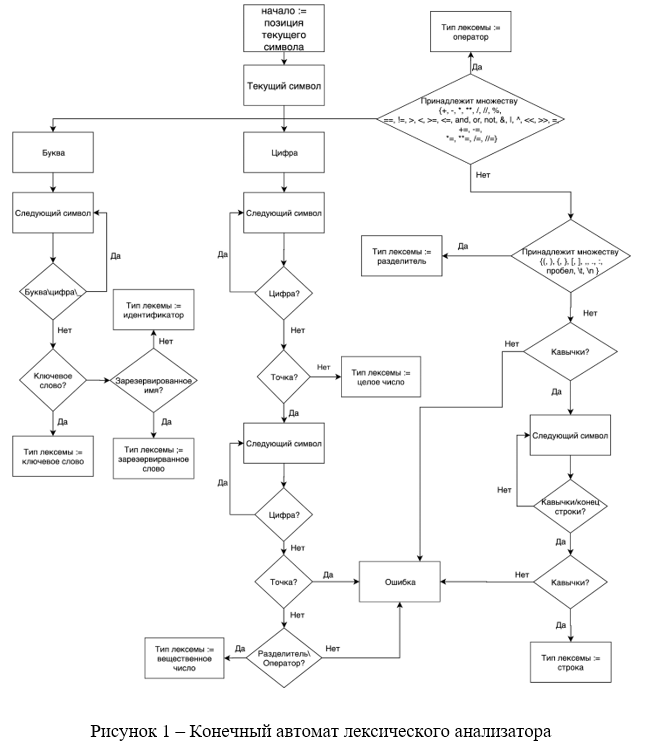
****

## 6.5 Типы ошибок

Все ошибки, которые может выдать лексер с примерами кода и сообщениями об ошибке

## 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.

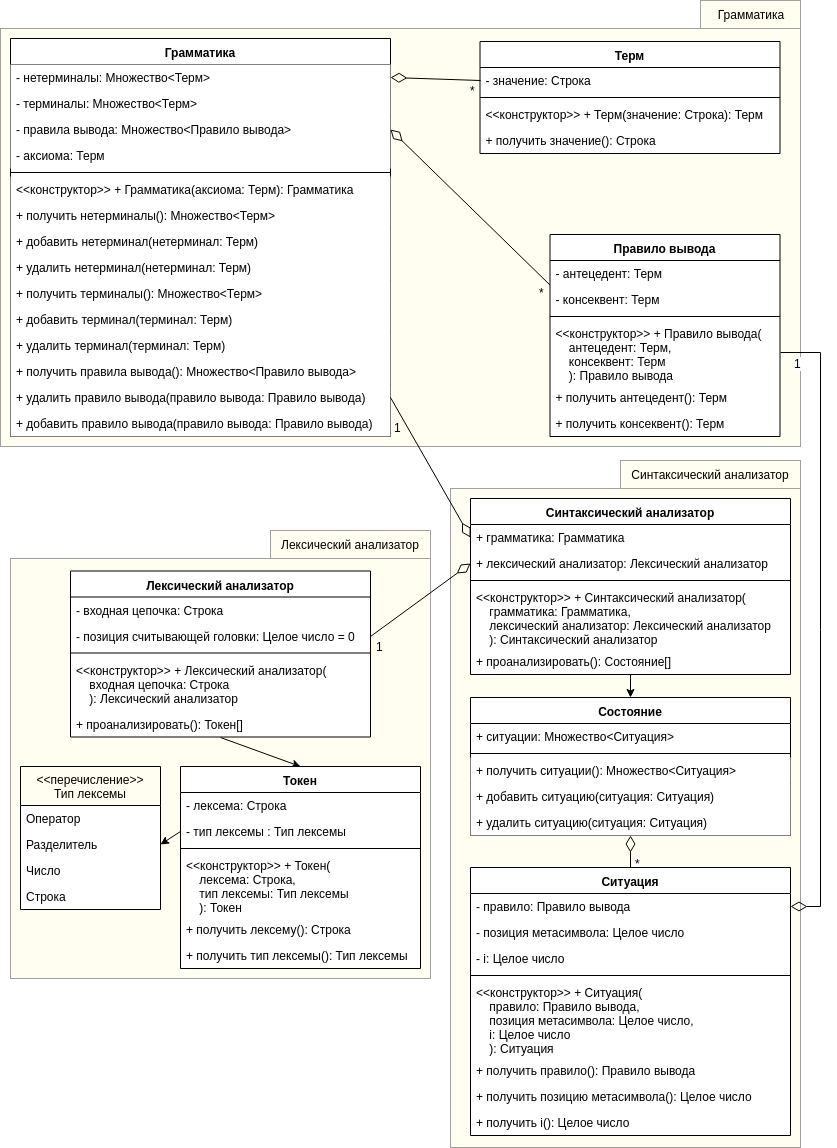


# 7 Проект семантического анализатора

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

Аналогично разделу 6

# 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

Аналогично 6

# 9 Проект генератора кода

## 9.1 Модель данных (если есть)

## 9.2 Метод кодогенерации

## 9.3 Примеры кодогенерации

# 10 Тестирование

## 10.1 Лексический анализатор

## 10.2 Синтаксический анализатор

## 10.3 Семантический анализатор

## 10.4 Генератор кода

someFn()

someFn(a, b)

someFn(a)

someFn(1+3)

someFn(x\*1)

someFn(a || b)

someFn(!a)

someFn(!a, a||b, c&&d)

someFn(!a, -a)

a=b

a = b&&c

a = b && c ||d

a = -d

a = b+7

a = (x/a)\*1 - 1

a = !b

a = (a || b)

if (a<b){

a=b

b=a

func()} else{}

if (a<b){}

if (a<b){} else{}

var p = myVar && myVar

var p = myVar || myVar

var p = myVar && !myVar

var p = !myVar && myVar

var x = !!b

var x = a && !!b

var x = a && !(a > 5)

var x = !!!b

var y = (a < 7) && b

var y = (a && b)

var y = !true && false

var y = true || (false && false) && my

var y = true || false || my

var y = true || false || !my

var y = (5 - 4) > 1

when(a) {

1 -> {}}

when(a) {1 -> {}

2 -> {}

}

when(a) {1 -> {}

2 -> {a=b}

}

when(a) {1 -> {}

2 -> {a=b}

3 -> {

a = a+b

}

}

for(a in b) {}

a = 1+2

a = 1+2-3-6-4-2

a = (b+c - 0)

a = 1+(2+3)

for(a in 1..2){

if (a < b) {a = a+b}

a = 7+1

a = 3+1+2

}

for(a in b){

if (a < b) {a = a+b}

a = 7+1

a = 3+1+2

continue

break

}

for(a in 1..2){

if (a < b) {

a = a+b

}

continue

if (a < b) {

a = a+b

}

}

public int double (int x) {

return 2\*x;

}

if (a < b) {

a=b

} else {}

if (a < b) { a=b}

a=b

var a = 1

b = a ||b

b = a&&(!b || (c && !d))

b = 4.3

while (a <99) {

a = a+1

}

while (a <99)

{

a = a+1

}

return x+44

return x

return (a || b)

# Заключение

Даже эмбрион знает, как писать заключение :)

Список литературы

<https://kotlinlang.org/docs/basic-types.html>

https://translated.turbopages.org/proxy\_u/en-ru.ru.1629bc8b-64525281-b29c5175-74722d776562/https/kotlinlang.org/api/latest/jvm/stdlib/kotlin/