МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение  
высшего образования

**«Дальневосточный федеральный университет»**

**ШКОЛА ЕСТЕСТВЕННЫХ НАУК**

**Кафедра прикладной математики, механики, управления и программного обеспечения**

РАЗРАБОТКА ТРАНСЛЯТОРА, ПЕРЕВОДЯЩЕГО ПОДМНОЖЕСТВО ЯЗЫКА JAVA В ЭКВИВАЛЕНТНОЕ ПОДМНОЖЕСТВО ЯЗЫКА GO

**КУРСОВОЙ ПРОЕКТ**

по дисциплине «Теория вычислительных процессов и структур»

по образовательной программе подготовки бакалавров по направлению 02.03.03 Математическое обеспечение и администрирование информационных систем

Выполнили студенты гр. Б8117-02.03.03

\_\_\_\_\_\_\_\_\_\_\_ Дейко А.О.

\_\_\_\_\_\_\_\_\_\_\_ Проскурин Д.А.

\_\_\_\_\_\_\_\_\_\_\_ Кокорин И.В..

Руководитель: старший преподаватель кафедры ПММУиПО Лемза А.О.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

(подпись)

Защищён с оценкой \_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

(подпись) И.О. Фамилия

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2020 г.

г. Владивосток

2020 г.

[Введение 3](#_Toc513753911)

[1 Неформальная постановка задачи 3](#_Toc1757287326)

[2 Синтаксис входного языка 9](#_Toc2014738130)

[3 Контекстные условия 9](#_Toc155568555)

[4 Таблица соответствия 9](#_Toc2072079054)

[5 Проект лексического анализатор 9](#_Toc598089411)

[5.1 Таблица ключевых слов 9](#_Toc345534689)

[5.2 Типы лексем 9](#_Toc2049318156)

[5.3 Ошибки лексического анализатора 9](#_Toc109576416)

[5.4 Конечный автомат лексического анализатора 9](#_Toc206466828)

[6 Проект синтаксического анализатора 9](#_Toc234752776)

[Заключение 9](#_Toc2069738125)

[Список литературы 9](#_Toc323976636)

# Введение

Программист, написавший программу на каком-либо языке программирования, не сможет сразу запустить свою программу. Одним из способов сделать это является перевод её на язык машины, на которой эта программа будет запущена, иначе говоря, эту программу нужно скомпилировать.

Таким образом компилятор – это программа, которая считывает текст программы, написанной на одном языке (исходном), и транслирует (переводит) его в эквивалентный текст на другом языке (целевом).

Цель курсового проекта: разработать проект транслятора, который преобразует программу, содержащую подмножество языка Java в программу, содержащую подмножество языка Go генерируя эквивалентный исходный код. Язык задаётся множеством цепочек, принадлежащих ему;

Задачи:

* разработать грамматику подмножества языка Java;
* описать контекстные условия входного языка;
* описать соответствие конструкций входного и выходного языков;
* разработать проект лексического анализатора;
* разработать проект синтаксического анализатора.

# Неформальная постановка задачи

Разработать проект транслятора из подмножества языка «Java» в эквивалентное подмножество языка «Go».

Подмножество языка «Java» включает:

1. Типы данных:***byte ,short, int,long,float,double,char,boolean.***
2. Операторы:

***Арифметические операторы(***Таблица 1***).***

|  |  |  |
| --- | --- | --- |
| Оператор | Описание | Пример |
| + | Складывает значения по обе стороны от оператора | A + B даст 30 |
| - | Вычитает правый операнд из левого операнда | A - B даст -10 |
| \* | Умножает значения по обе стороны от оператора | A \* B даст 200 |
| / | Оператор деления делит левый операнд на правый операнд | B / A даст 2 |
| % | Делит левый операнд на правый операнд и возвращает остаток | B % A даст 0 |
| ++ | Инкремент - увеличивает значение операнда на 1 | B++ даст 21 |
| -- | Декремент - уменьшает значение операнда на 1 | B-- даст 19 |

Таблица -Арифметические операторы

***Операторы сравнения(Таблица 2).***

|  |  |  |
| --- | --- | --- |
| Оператор | Описание | Пример |
| == | Проверяет, равны или нет значения двух операндов, если да, то условие становится истинным | (A == B) — не верны |
| != | Проверяет, равны или нет значения двух операндов, если значения не равны, то условие становится истинным | (A != B) — значение истинна |
| > | Проверяет, является ли значение левого операнда больше, чем значение правого операнда, если да, то условие становится истинным | (A > B) — не верны |
| < | Проверяет, является ли значение левого операнда меньше, чем значение правого операнда, если да, то условие становится истинным | (A < B) — значение истинна |
| >= | Проверяет, является ли значение левого операнда больше или равно значению правого операнда, если да, то условие становится истинным | (A >= B) — значение не верны |
| <= | Проверяет, если значение левого операнда меньше или равно значению правого операнда, если да, то условие становится истинным | (A <= B) — значение истинна |

Таблица -Операторы сравнения

***Побитовые операторы(Таблица 3).***

|  |  |  |
| --- | --- | --- |
| Оператор | Описание | Пример |
| & (побитовое и) | Бинарный оператор AND копирует бит в результат, если он существует в обоих операндах. | (A & B) даст 12, который является 0000 1100 |
| | (побитовое или) | Бинарный оператор OR копирует бит, если он существует в любом из операндов. | (A | B) даст 61 который равен 0011 1101 |
| ^ (побитовое логическое или) | Бинарный оператор XOR копирует бит, если он установлен в одном операнде, но не в обоих. | (A ^ B) даст 49, которая является 0011 0001 |
| ~ (побитовое дополнение) | Бинарный оператор дополнения и имеет эффект «отражения» бит. | (~ A) даст -61, которая является формой дополнением 1100 0011 в двоичной записи |
| << (сдвиг влево) | Бинарный оператор сдвига влево. Значение левых операндов перемещается влево на количество бит, заданных правым операндом. | A << 2 даст 240, который 1111 0000 |
| >> (сдвиг вправо) | Бинарный оператор сдвига вправо. Значение правых операндов перемещается вправо на количество бит, заданных левых операндом. | A >> 2 даст 15, который является 1111 |
| >>> (нулевой сдвиг вправо) | Нулевой оператор сдвига вправо. Значение левых операндов перемещается вправо на количество бит, заданных правым операндом, а сдвинутые значения заполняются нулями. | A >>> 2 даст 15, который является 0000 1111 |

Таблица -Побитовые операторы

***Логическое операторы(Таблица 4).***

|  |  |  |
| --- | --- | --- |
| Оператор | Описание | Пример |
| && | Называется логический оператор «И». Если оба операнда являются не равны нулю, то условие становится истинным | (A && B) — значение false |
| || | Называется логический оператор «ИЛИ». Если любой из двух операндов не равен нулю, то условие становится истинным | (A || B) — значение true |
| ! | Называется логический оператор «НЕ». Использование меняет логическое состояние своего операнда. Если условие имеет значение true, то оператор логического «НЕ» будет делать false | !(A && B) — значение true |

Таблица -Логические операторы

***Операторы присваивания(Таблица 5).***

|  |  |  |
| --- | --- | --- |
| Оператор | Описание | Пример |
| = | Простой оператор присваивания, присваивает значения из правой стороны операндов к левому операнду | C = A + B, присвоит значение A + B в C |
| += | Оператор присваивания «Добавления», он присваивает левому операнду значения правого | C += A, эквивалентно C = C + A |
| -= | Оператор присваивания «Вычитания», он вычитает из правого операнда левый операнд | C -= A, эквивалентно C = C - A |
| \*= | Оператор присваивания «Умножение», он умножает правый операнд на левый операнд | C \* = A эквивалентно C = C \* A |
| /= | Оператор присваивания «Деление», он делит левый операнд на правый операнд | C /= A эквивалентно C = C / A |
| %= | Оператор присваивания «Модуль», он принимает модуль, с помощью двух операндов и присваивает его результат левому операнду | C %= A, эквивалентно C = C % A |

Таблица -Операторы присваивания

1. Операторы циклов: цикл for, цикл do, цикл while.
2. Операторы ветвления: if else, тернарный условный оператор (условное выражение ? выражени1: выражение2).
3. Стандартные функции:

double min( a, b ) – возвращает минимум из a и b

double log( a ) – возвращает натуральный логарифм аргумента a

double pow( a, b ) – возвращает значение a, возведенное в степень b

double sqrt( a ) – возвращает квадратный корень из a

double time() – возвращает текущее значение модельного времени (в единицах модельного времени)

Date date() – возвращает текущую модельную дату (Date является стандартным Java классом)

int getMinute() – возвращает текущую минуту в часе текущей модельной даты

double minute()– возвращает интервал времени, соответствующий одной минуте, в выбранных в данной модели единицах модельного времени.

# Синтаксис входного языка

# Контекстные условия

Контекстные условия о правилах описания идентификаторов в программах:

1. Все используемые в программах идентификаторы должны быть описаны до их использования в программе.
2. каждый из идентификаторов, используемых в программе, должен быть описан один раз (для каждого идентификатора обычно определяется его область действия, и единственное описание должно относиться к этой области действия – блоку программы, модулю программы, подпрограммы, описание в формальных параметрах процедур и функций и т.д.).

Контекстные условия о правилах использования идентификаторов в своей области действия:

Контекстные условия, определяющие правила соответствия видов величин, входящих в синтаксические конструкции программ:

Контекстные условия, задающие различные количественные ограничения:

# Таблица соответствия

# Проект лексического анализатор

# Список ключевых слов

Ключевые(зарезервированные) слова:

abstract — абстрактный метод, абстрактный класс;

assert — отладка программы;

boolean — булев тип;

break — оператор для выхода из цикла или оператора switch;

byte — целочисленный тип;

case — переключатель оператора switch;

catch — оператор обработки исключений;

char — символьный тип;

class — класс;

const — константа;

continue — прекращение итерации цикла;

default — ветвь оператора switch, метод по умолчанию;

do — оператор цикла do/while;

double — тип числа с плавающей точкой;

else — оператор принятия решений;

enum — перечисление;

extends — родитель класса;

final — класс, который нельзя расширить, метод, который нельзя переопределить или завершённый член данных;

finally — оператор обработки исключений;

float — тип числа с плавающей точкой;

for — тип цикла;

goto;

if — оператор принятия решений;

implements — интерфейсы, реализуемые классом;

import — импорт пакета;

instaceof — является ли объект экземпляром класса;

int — целочисленный тип;

interface — интерфейс;

long — целочисленный тип;

native — метод с кодом, написанным на другом языке;

new — выделение памяти для нового массива или объекта;

package — пакет классов;

private — модификатор доступа;

protected — модификатор доступа;

public — модификатор доступа;

return — выход из метода и возвращение значения;

short — целочисленный тип;

static — переменная или метод, общий для всех экземпляров класса;

strictfp — строгие правила для вычислений с плавающей точкой;

super — объект или конструктор суперкласса;

switch — оператор управления выполнения программы;

synchronized — доступ к участку кода только одному потоку;

this — неявный аргумент метода или конструктора класса;

throw — оператор обработки исключений;

throws — оператор обработки исключений;

transient — данные, которые не должны быть постоянными;

try — оператор обработки исключений;

void — метод не возвращает значений;

volatile — поле доступно нескольким потокам;

while — тип цикла.

Зарезервированные логические значения: true, false, null. Не могут быть использованы в качестве переменных и имен других объектов.

# Типы лексем

# Ошибки лексического анализатора

В данной проекте лексический анализатор распознает следующие синтаксические ошибки:

1. Пропуск знака пунктуации;
2. Несогласованность скобок;
3. Неправильное формирование оператора;
4. Неверное образование имен переменных;
5. Неверное написание служебных слов;
6. Отсутствие условий окончания цикла;
7. Отсутствие описания массива и т.п.

# Конечный автомат лексического анализатора

# Проект синтаксического анализатора

# Заключение

# Список литературы