Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых»

(ВлГУ)

разработка компилятора подмножества

процедурно-ориентированного языка

Пояснительная записка

RU. 643.02068048.0001-01 81 01

На 18 листах

|  |  |  |
| --- | --- | --- |
| Руководитель |  | к.т.н. доцент кафедры ИЗИ Ю.М. Монахов |
|  |
| Исполнитель |  | студент гр. ИСБ-117 В.А. Максимов |

**Владимир 2020**

Оглавление

[ВВЕДЕНИЕ 5](#_Toc42703699)

[ПРОЕКТИРОВАНИЕ КОМПИЛЯТОРА 6](#_Toc42703700)

[1.1 ПРОЕКТИРОВАНИЕ СКАНЕРА ТОКЕНОВ 6](#_Toc42703701)

[1.2 Структура данных - Токен 13](#_Toc42703702)

[1.3 Проектирование парсера 14](#_Toc42703703)

[1.4 Проектирование транслятора 16](#_Toc42703705)

[2.1 Упаковка приложения 17](#_Toc42703706)

[2.5 Руководство пользователя 17](#_Toc42703707)

[3. Тесты 17](#_Toc42703708)

[ЗАКЛЮЧЕНИЕ 17](#_Toc42703709)

[Приложение 18](#_Toc42703710)

Изм.

Лист

№ докум.

Подпись

Дата

Лист

2

RU. 02068048.502900.01.01

Разраб.

Максимов В.А.

Провер.

Монахов Ю.М.

Реценз.

Н. Контр.

Монахов Ю.М.

Утверд.

Компилятор процедурно-ориентированного языка Golang

Лит.

Листов

29

ВлГУ

# ВВЕДЕНИЕ

Задачи курсовой работы заключаются в:

Описание программы, предназначенной для компиляции программ на некотором исходном языке программирования под целевую платформу. Исходным языком, предназначенным для компиляции является выбранное мной подмножество синтаксиса языка Golang. Язык, выбранный для разработки Python 3.8. Программа интерпртируется.

Требования к приложению:

Требования к входному языку:

1. Должны присутствовать операторные скобки.

2. Должна игнорироваться индентация программы.

3. Должны поддерживаться комментарии любой длины.

4. Входная программа должна представлять собой единый модуль, но также должна быть поддержка вызова функций.

Операторы:

1. Оператор присваивания.

2. Арифметика (\*, /, +, -, >, <, =).

3. Логические операторы (И, ИЛИ, НЕ).

4. Условный оператор (ЕСЛИ).

5. Операторы цикла (while, break, continue).

6. Базовый вывод (строковый литерал, переменная).

7. Типы (целочисленный 32 бита, с плавающей запятой 32 бита).

Требования к выходному языку:

1. В ассемблере.

Интерфейс приложения:

* На вход поступает файл с кодом программы на языке Golang
* В процессе работы компилятора выполняется этот код на целевой платформе
* На выход получаем этот файл, очищенный от комментариев и пустых блоков

# ПРОЕКТИРОВАНИЕ КОМПИЛЯТОРА

# 1.1 ПРОЕКТИРОВАНИЕ СКАНЕРА ТОКЕНОВ

**Лексер** – представляет собой модуль, предназначенный для считывания списка токенов программы.

Листинг кода лексера ниже:

# Generated from D:/Python/compiler/lexanalyz\LexerGo.g4 by ANTLR 4.8  
from antlr4 import \*  
from io import StringIO  
from typing.io import TextIO  
import sys  
  
  
  
def serializedATN():  
 with StringIO() as buf:  
 buf.write("\3\u608b\ua72a\u8133\ub9ed\u417c\u3be7\u7786\u5964\2M")  
 buf.write("\u0249\b\1\4\2\t\2\4\3\t\3\4\4\t\4\4\5\t\5\4\6\t\6\4\7")  
 buf.write("\t\7\4\b\t\b\4\t\t\t\4\n\t\n\4\13\t\13\4\f\t\f\4\r\t\r")  
 buf.write("\4\16\t\16\4\17\t\17\4\20\t\20\4\21\t\21\4\22\t\22\4\23")  
 buf.write("\t\23\4\24\t\24\4\25\t\25\4\26\t\26\4\27\t\27\4\30\t\30")  
 buf.write("\4\31\t\31\4\32\t\32\4\33\t\33\4\34\t\34\4\35\t\35\4\36")  
 buf.write("\t\36\4\37\t\37\4 \t \4!\t!\4\"\t\"\4#\t#\4$\t$\4%\t%")  
 buf.write("\4&\t&\4\'\t\'\4(\t(\4)\t)\4\*\t\*\4+\t+\4,\t,\4-\t-\4.")  
 buf.write("\t.\4/\t/\4\60\t\60\4\61\t\61\4\62\t\62\4\63\t\63\4\64")  
 buf.write("\t\64\4\65\t\65\4\66\t\66\4\67\t\67\48\t8\49\t9\4:\t:")  
 buf.write("\4;\t;\4<\t<\4=\t=\4>\t>\4?\t?\4@\t@\4A\tA\4B\tB\4C\t")  
 buf.write("C\4D\tD\4E\tE\4F\tF\4G\tG\4H\tH\4I\tI\4J\tJ\4K\tK\4L\t")  
 buf.write("L\4M\tM\4N\tN\4O\tO\4P\tP\4Q\tQ\4R\tR\4S\tS\4T\tT\3\2")  
 buf.write("\3\2\3\2\3\2\3\2\3\2\3\3\3\3\3\3\3\3\3\3\3\3\3\3\3\3\3")  
 buf.write("\4\3\4\3\4\3\4\3\4\3\5\3\5\3\5\3\5\3\5\3\5\3\5\3\5\3\5")  
 buf.write("\3\5\3\6\3\6\3\6\3\6\3\6\3\6\3\6\3\7\3\7\3\7\3\7\3\7\3")  
 buf.write("\b\3\b\3\b\3\b\3\b\3\b\3\t\3\t\3\t\3\n\3\n\3\n\3\n\3\13")  
 buf.write("\3\13\3\13\3\13\3\13\3\13\3\13\3\f\3\f\3\f\3\f\3\f\3\r")  
 buf.write("\3\r\3\r\3\r\3\r\3\16\3\16\3\16\3\16\3\16\3\17\3\17\3")  
 buf.write("\17\3\17\3\17\3\17\3\17\3\17\3\20\3\20\3\20\3\20\3\20")  
 buf.write("\3\20\3\20\3\21\3\21\3\21\3\21\3\21\3\21\3\22\3\22\3\22")  
 buf.write("\3\22\3\22\3\22\3\22\3\22\3\22\3\22\3\22\3\22\3\23\3\23")  
 buf.write("\3\23\3\24\3\24\3\24\3\24\3\24\3\24\3\25\3\25\3\25\3\25")  
 buf.write("\3\25\3\26\3\26\3\26\3\26\3\26\3\26\3\26\3\26\3\26\3\27")  
 buf.write("\3\27\3\27\3\27\3\30\3\30\3\30\3\30\3\30\3\30\3\30\3\31")  
 buf.write("\3\31\3\31\3\31\3\31\3\31\3\31\3\32\3\32\3\32\3\32\3\33")  
 buf.write("\3\33\3\33\3\33\3\34\3\34\3\34\7\34\u014b\n\34\f\34\16")  
 buf.write("\34\u014e\13\34\3\35\3\35\3\36\3\36\3\37\3\37\3 \3 \3")  
 buf.write("!\3!\3\"\3\"\3#\3#\3$\3$\3%\3%\3&\3&\3\'\3\'\3(\3(\3(")  
 buf.write("\3)\3)\3)\3\*\3\*\3\*\3+\3+\3+\3+\3,\3,\3,\3-\3-\3-\3.\3")  
 buf.write(".\3.\3/\3/\3/\3\60\3\60\3\61\3\61\3\61\3\62\3\62\3\63")  
 buf.write("\3\63\3\63\3\64\3\64\3\65\3\65\3\66\3\66\3\67\3\67\3\67")  
 buf.write("\38\38\38\39\39\39\3:\3:\3;\3;\3<\3<\3=\3=\3>\3>\3?\3")  
 buf.write("?\3@\3@\3@\3A\3A\7A\u01a9\nA\fA\16A\u01ac\13A\3B\3B\7")  
 buf.write("B\u01b0\nB\fB\16B\u01b3\13B\3C\3C\3C\6C\u01b8\nC\rC\16")  
 buf.write("C\u01b9\3D\3D\3D\5D\u01bf\nD\3D\5D\u01c2\nD\3D\5D\u01c5")  
 buf.write("\nD\3D\3D\3D\5D\u01ca\nD\5D\u01cc\nD\3E\3E\5E\u01d0\n")  
 buf.write("E\3E\3E\3F\3F\3F\5F\u01d7\nF\3F\3F\3G\3G\7G\u01dd\nG\f")  
 buf.write("G\16G\u01e0\13G\3G\3G\3H\3H\3H\7H\u01e7\nH\fH\16H\u01ea")  
 buf.write("\13H\3H\3H\3I\6I\u01ef\nI\rI\16I\u01f0\3I\3I\3J\3J\3J")  
 buf.write("\3J\7J\u01f9\nJ\fJ\16J\u01fc\13J\3J\3J\3J\3J\3J\3K\6K")  
 buf.write("\u0204\nK\rK\16K\u0205\3K\3K\3L\3L\3L\3L\7L\u020e\nL\f")  
 buf.write("L\16L\u0211\13L\3L\3L\3M\3M\3M\3M\3M\3M\3M\3M\3M\3M\3")  
 buf.write("M\3M\3M\3M\3M\3M\3M\3M\3M\3M\3M\3M\3M\3M\3M\3M\5M\u022f")  
 buf.write("\nM\3N\6N\u0232\nN\rN\16N\u0233\3O\3O\3P\3P\3Q\3Q\5Q\u023c")  
 buf.write("\nQ\3Q\3Q\3R\3R\5R\u0242\nR\3S\5S\u0245\nS\3T\5T\u0248")  
 buf.write("\nT\3\u01fa\2U\3\3\5\4\7\5\t\6\13\7\r\b\17\t\21\n\23\13")  
 buf.write("\25\f\27\r\31\16\33\17\35\20\37\21!\22#\23%\24\'\25)\26")  
 buf.write("+\27-\30/\31\61\32\63\33\65\34\67\359\36;\37= ?!A\"C#")  
 buf.write("E$G%I&K\'M(O)Q\*S+U,W-Y.[/]\60\_\61a\62c\63e\64g\65i\66")  
 buf.write("k\67m8o9q:s;u<w=y>{?}@\177A\u0081B\u0083C\u0085D\u0087")  
 buf.write("E\u0089F\u008bG\u008dH\u008fI\u0091J\u0093K\u0095L\u0097")  
 buf.write("M\u0099\2\u009b\2\u009d\2\u009f\2\u00a1\2\u00a3\2\u00a5")  
 buf.write("\2\u00a7\2\3\2\21\3\2\63;\3\2\62;\4\2ZZzz\4\2\f\f^^\3")  
 buf.write("\2bb\4\2$$^^\4\2\13\13\"\"\4\2\f\f\17\17\13\2$$))^^cd")  
 buf.write("hhppttvvxx\3\2\629\5\2\62;CHch\4\2GGgg\4\2--//\26\2\62")  
 buf.write(";\u0662\u066b\u06f2\u06fb\u0968\u0971\u09e8\u09f1\u0a68")  
 buf.write("\u0a71\u0ae8\u0af1\u0b68\u0b71\u0be9\u0bf1\u0c68\u0c71")  
 buf.write("\u0ce8\u0cf1\u0d68\u0d71\u0e52\u0e5b\u0ed2\u0edb\u0f22")  
 buf.write("\u0f2b\u1042\u104b\u136b\u1373\u17e2\u17eb\u1812\u181b")  
 buf.write("\uff12\uff1b\u0104\2C\\c|\u00ac\u00ac\u00b7\u00b7\u00bc")  
 buf.write("\u00bc\u00c2\u00d8\u00da\u00f8\u00fa\u0221\u0224\u0235")  
 buf.write("\u0252\u02af\u02b2\u02ba\u02bd\u02c3\u02d2\u02d3\u02e2")  
 buf.write("\u02e6\u02f0\u02f0\u037c\u037c\u0388\u0388\u038a\u038c")  
 buf.write("\u038e\u038e\u0390\u03a3\u03a5\u03d0\u03d2\u03d9\u03dc")  
 buf.write("\u03f5\u0402\u0483\u048e\u04c6\u04c9\u04ca\u04cd\u04ce")  
 buf.write("\u04d2\u04f7\u04fa\u04fb\u0533\u0558\u055b\u055b\u0563")  
 buf.write("\u0589\u05d2\u05ec\u05f2\u05f4\u0623\u063c\u0642\u064c")  
 buf.write("\u0673\u06d5\u06d7\u06d7\u06e7\u06e8\u06fc\u06fe\u0712")  
 buf.write("\u0712\u0714\u072e\u0782\u07a7\u0907\u093b\u093f\u093f")  
 buf.write("\u0952\u0952\u095a\u0963\u0987\u098e\u0991\u0992\u0995")  
 buf.write("\u09aa\u09ac\u09b2\u09b4\u09b4\u09b8\u09bb\u09de\u09df")  
 buf.write("\u09e1\u09e3\u09f2\u09f3\u0a07\u0a0c\u0a11\u0a12\u0a15")  
 buf.write("\u0a2a\u0a2c\u0a32\u0a34\u0a35\u0a37\u0a38\u0a3a\u0a3b")  
 buf.write("\u0a5b\u0a5e\u0a60\u0a60\u0a74\u0a76\u0a87\u0a8d\u0a8f")  
 buf.write("\u0a8f\u0a91\u0a93\u0a95\u0aaa\u0aac\u0ab2\u0ab4\u0ab5")  
 buf.write("\u0ab7\u0abb\u0abf\u0abf\u0ad2\u0ad2\u0ae2\u0ae2\u0b07")  
 buf.write("\u0b0e\u0b11\u0b12\u0b15\u0b2a\u0b2c\u0b32\u0b34\u0b35")  
 buf.write("\u0b38\u0b3b\u0b3f\u0b3f\u0b5e\u0b5f\u0b61\u0b63\u0b87")  
 buf.write("\u0b8c\u0b90\u0b92\u0b94\u0b97\u0b9b\u0b9c\u0b9e\u0b9e")  
 buf.write("\u0ba0\u0ba1\u0ba5\u0ba6\u0baa\u0bac\u0bb0\u0bb7\u0bb9")  
 buf.write("\u0bbb\u0c07\u0c0e\u0c10\u0c12\u0c14\u0c2a\u0c2c\u0c35")  
 buf.write("\u0c37\u0c3b\u0c62\u0c63\u0c87\u0c8e\u0c90\u0c92\u0c94")  
 buf.write("\u0caa\u0cac\u0cb5\u0cb7\u0cbb\u0ce0\u0ce0\u0ce2\u0ce3")  
 buf.write("\u0d07\u0d0e\u0d10\u0d12\u0d14\u0d2a\u0d2c\u0d3b\u0d62")  
 buf.write("\u0d63\u0d87\u0d98\u0d9c\u0db3\u0db5\u0dbd\u0dbf\u0dbf")  
 buf.write("\u0dc2\u0dc8\u0e03\u0e32\u0e34\u0e35\u0e42\u0e48\u0e83")  
 buf.write("\u0e84\u0e86\u0e86\u0e89\u0e8a\u0e8c\u0e8c\u0e8f\u0e8f")  
 buf.write("\u0e96\u0e99\u0e9b\u0ea1\u0ea3\u0ea5\u0ea7\u0ea7\u0ea9")  
 buf.write("\u0ea9\u0eac\u0ead\u0eaf\u0eb2\u0eb4\u0eb5\u0ebf\u0ec6")  
 buf.write("\u0ec8\u0ec8\u0ede\u0edf\u0f02\u0f02\u0f42\u0f6c\u0f8a")  
 buf.write("\u0f8d\u1002\u1023\u1025\u1029\u102b\u102c\u1052\u1057")  
 buf.write("\u10a2\u10c7\u10d2\u10f8\u1102\u115b\u1161\u11a4\u11aa")  
 buf.write("\u11fb\u1202\u1208\u120a\u1248\u124a\u124a\u124c\u124f")  
 buf.write("\u1252\u1258\u125a\u125a\u125c\u125f\u1262\u1288\u128a")  
 buf.write("\u128a\u128c\u128f\u1292\u12b0\u12b2\u12b2\u12b4\u12b7")  
 buf.write("\u12ba\u12c0\u12c2\u12c2\u12c4\u12c7\u12ca\u12d0\u12d2")  
 buf.write("\u12d8\u12da\u12f0\u12f2\u1310\u1312\u1312\u1314\u1317")  
 buf.write("\u131a\u1320\u1322\u1348\u134a\u135c\u13a2\u13f6\u1403")  
 buf.write("\u1678\u1683\u169c\u16a2\u16ec\u1782\u17b5\u1822\u1879")  
 buf.write("\u1882\u18aa\u1e02\u1e9d\u1ea2\u1efb\u1f02\u1f17\u1f1a")  
 buf.write("\u1f1f\u1f22\u1f47\u1f4a\u1f4f\u1f52\u1f59\u1f5b\u1f5b")  
 buf.write("\u1f5d\u1f5d\u1f5f\u1f5f\u1f61\u1f7f\u1f82\u1fb6\u1fb8")  
 buf.write("\u1fbe\u1fc0\u1fc0\u1fc4\u1fc6\u1fc8\u1fce\u1fd2\u1fd5")  
 buf.write("\u1fd8\u1fdd\u1fe2\u1fee\u1ff4\u1ff6\u1ff8\u1ffe\u2081")  
 buf.write("\u2081\u2104\u2104\u2109\u2109\u210c\u2115\u2117\u2117")  
 buf.write("\u211b\u211f\u2126\u2126\u2128\u2128\u212a\u212a\u212c")  
 buf.write("\u212f\u2131\u2133\u2135\u213b\u2162\u2185\u3007\u3009")  
 buf.write("\u3023\u302b\u3033\u3037\u303a\u303c\u3043\u3096\u309f")  
 buf.write("\u30a0\u30a3\u30fc\u30fe\u3100\u3107\u312e\u3133\u3190")  
 buf.write("\u31a2\u31b9\u3402\u3402\u4db7\u4db7\u4e02\u4e02\u9fa7")  
 buf.write("\u9fa7\ua002\ua48e\uac02\uac02\ud7a5\ud7a5\uf902\ufa2f")  
 buf.write("\ufb02\ufb08\ufb15\ufb19\ufb1f\ufb1f\ufb21\ufb2a\ufb2c")  
 buf.write("\ufb38\ufb3a\ufb3e\ufb40\ufb40\ufb42\ufb43\ufb45\ufb46")  
 buf.write("\ufb48\ufbb3\ufbd5\ufd3f\ufd52\ufd91\ufd94\ufdc9\ufdf2")  
 buf.write("\ufdfd\ufe72\ufe74\ufe76\ufe76\ufe78\ufefe\uff23\uff3c")  
 buf.write("\uff43\uff5c\uff68\uffc0\uffc4\uffc9\uffcc\uffd1\uffd4")  
 buf.write("\uffd9\uffdc\uffde\2\u025a\2\3\3\2\2\2\2\5\3\2\2\2\2\7")  
 buf.write("\3\2\2\2\2\t\3\2\2\2\2\13\3\2\2\2\2\r\3\2\2\2\2\17\3\2")  
 buf.write("\2\2\2\21\3\2\2\2\2\23\3\2\2\2\2\25\3\2\2\2\2\27\3\2\2")  
 buf.write("\2\2\31\3\2\2\2\2\33\3\2\2\2\2\35\3\2\2\2\2\37\3\2\2\2")  
 buf.write("\2!\3\2\2\2\2#\3\2\2\2\2%\3\2\2\2\2\'\3\2\2\2\2)\3\2\2")  
 buf.write("\2\2+\3\2\2\2\2-\3\2\2\2\2/\3\2\2\2\2\61\3\2\2\2\2\63")  
 buf.write("\3\2\2\2\2\65\3\2\2\2\2\67\3\2\2\2\29\3\2\2\2\2;\3\2\2")  
 buf.write("\2\2=\3\2\2\2\2?\3\2\2\2\2A\3\2\2\2\2C\3\2\2\2\2E\3\2")  
 buf.write("\2\2\2G\3\2\2\2\2I\3\2\2\2\2K\3\2\2\2\2M\3\2\2\2\2O\3")  
 buf.write("\2\2\2\2Q\3\2\2\2\2S\3\2\2\2\2U\3\2\2\2\2W\3\2\2\2\2Y")  
 buf.write("\3\2\2\2\2[\3\2\2\2\2]\3\2\2\2\2\_\3\2\2\2\2a\3\2\2\2\2")  
 buf.write("c\3\2\2\2\2e\3\2\2\2\2g\3\2\2\2\2i\3\2\2\2\2k\3\2\2\2")  
 buf.write("\2m\3\2\2\2\2o\3\2\2\2\2q\3\2\2\2\2s\3\2\2\2\2u\3\2\2")  
 buf.write("\2\2w\3\2\2\2\2y\3\2\2\2\2{\3\2\2\2\2}\3\2\2\2\2\177\3")  
 buf.write("\2\2\2\2\u0081\3\2\2\2\2\u0083\3\2\2\2\2\u0085\3\2\2\2")  
 buf.write("\2\u0087\3\2\2\2\2\u0089\3\2\2\2\2\u008b\3\2\2\2\2\u008d")  
 buf.write("\3\2\2\2\2\u008f\3\2\2\2\2\u0091\3\2\2\2\2\u0093\3\2\2")  
 buf.write("\2\2\u0095\3\2\2\2\2\u0097\3\2\2\2\3\u00a9\3\2\2\2\5\u00af")  
 buf.write("\3\2\2\2\7\u00b7\3\2\2\2\t\u00bc\3\2\2\2\13\u00c6\3\2")  
 buf.write("\2\2\r\u00cd\3\2\2\2\17\u00d2\3\2\2\2\21\u00d8\3\2\2\2")  
 buf.write("\23\u00db\3\2\2\2\25\u00df\3\2\2\2\27\u00e6\3\2\2\2\31")  
 buf.write("\u00eb\3\2\2\2\33\u00f0\3\2\2\2\35\u00f5\3\2\2\2\37\u00fd")  
 buf.write("\3\2\2\2!\u0104\3\2\2\2#\u010a\3\2\2\2%\u0116\3\2\2\2")  
 buf.write("\'\u0119\3\2\2\2)\u011f\3\2\2\2+\u0124\3\2\2\2-\u012d")  
 buf.write("\3\2\2\2/\u0131\3\2\2\2\61\u0138\3\2\2\2\63\u013f\3\2")  
 buf.write("\2\2\65\u0143\3\2\2\2\67\u0147\3\2\2\29\u014f\3\2\2\2")  
 buf.write(";\u0151\3\2\2\2=\u0153\3\2\2\2?\u0155\3\2\2\2A\u0157\3")  
 buf.write("\2\2\2C\u0159\3\2\2\2E\u015b\3\2\2\2G\u015d\3\2\2\2I\u015f")  
 buf.write("\3\2\2\2K\u0161\3\2\2\2M\u0163\3\2\2\2O\u0165\3\2\2\2")  
 buf.write("Q\u0168\3\2\2\2S\u016b\3\2\2\2U\u016e\3\2\2\2W\u0172\3")  
 buf.write("\2\2\2Y\u0175\3\2\2\2[\u0178\3\2\2\2]\u017b\3\2\2\2\_\u017e")  
 buf.write("\3\2\2\2a\u0180\3\2\2\2c\u0183\3\2\2\2e\u0185\3\2\2\2")  
 buf.write("g\u0188\3\2\2\2i\u018a\3\2\2\2k\u018c\3\2\2\2m\u018e\3")  
 buf.write("\2\2\2o\u0191\3\2\2\2q\u0194\3\2\2\2s\u0197\3\2\2\2u\u0199")  
 buf.write("\3\2\2\2w\u019b\3\2\2\2y\u019d\3\2\2\2{\u019f\3\2\2\2")  
 buf.write("}\u01a1\3\2\2\2\177\u01a3\3\2\2\2\u0081\u01a6\3\2\2\2")  
 buf.write("\u0083\u01ad\3\2\2\2\u0085\u01b4\3\2\2\2\u0087\u01cb\3")  
 buf.write("\2\2\2\u0089\u01cf\3\2\2\2\u008b\u01d3\3\2\2\2\u008d\u01da")  
 buf.write("\3\2\2\2\u008f\u01e3\3\2\2\2\u0091\u01ee\3\2\2\2\u0093")  
 buf.write("\u01f4\3\2\2\2\u0095\u0203\3\2\2\2\u0097\u0209\3\2\2\2")  
 buf.write("\u0099\u0214\3\2\2\2\u009b\u0231\3\2\2\2\u009d\u0235\3")  
 buf.write("\2\2\2\u009f\u0237\3\2\2\2\u00a1\u0239\3\2\2\2\u00a3\u0241")  
 buf.write("\3\2\2\2\u00a5\u0244\3\2\2\2\u00a7\u0247\3\2\2\2\u00a9")  
 buf.write("\u00aa\7d\2\2\u00aa\u00ab\7t\2\2\u00ab\u00ac\7g\2\2\u00ac")  
 buf.write("\u00ad\7c\2\2\u00ad\u00ae\7m\2\2\u00ae\4\3\2\2\2\u00af")  
 buf.write("\u00b0\7f\2\2\u00b0\u00b1\7g\2\2\u00b1\u00b2\7h\2\2\u00b2")  
 buf.write("\u00b3\7c\2\2\u00b3\u00b4\7w\2\2\u00b4\u00b5\7n\2\2\u00b5")  
 buf.write("\u00b6\7v\2\2\u00b6\6\3\2\2\2\u00b7\u00b8\7h\2\2\u00b8")  
 buf.write("\u00b9\7w\2\2\u00b9\u00ba\7p\2\2\u00ba\u00bb\7e\2\2\u00bb")  
 buf.write("\b\3\2\2\2\u00bc\u00bd\7k\2\2\u00bd\u00be\7p\2\2\u00be")  
 buf.write("\u00bf\7v\2\2\u00bf\u00c0\7g\2\2\u00c0\u00c1\7t\2\2\u00c1")  
 buf.write("\u00c2\7h\2\2\u00c2\u00c3\7c\2\2\u00c3\u00c4\7e\2\2\u00c4")  
 buf.write("\u00c5\7g\2\2\u00c5\n\3\2\2\2\u00c6\u00c7\7u\2\2\u00c7")  
 buf.write("\u00c8\7g\2\2\u00c8\u00c9\7n\2\2\u00c9\u00ca\7g\2\2\u00ca")  
 buf.write("\u00cb\7e\2\2\u00cb\u00cc\7v\2\2\u00cc\f\3\2\2\2\u00cd")  
 buf.write("\u00ce\7e\2\2\u00ce\u00cf\7c\2\2\u00cf\u00d0\7u\2\2\u00d0")  
 buf.write("\u00d1\7g\2\2\u00d1\16\3\2\2\2\u00d2\u00d3\7f\2\2\u00d3")  
 buf.write("\u00d4\7g\2\2\u00d4\u00d5\7h\2\2\u00d5\u00d6\7g\2\2\u00d6")  
 buf.write("\u00d7\7t\2\2\u00d7\20\3\2\2\2\u00d8\u00d9\7i\2\2\u00d9")  
 buf.write("\u00da\7q\2\2\u00da\22\3\2\2\2\u00db\u00dc\7o\2\2\u00dc")  
 buf.write("\u00dd\7c\2\2\u00dd\u00de\7r\2\2\u00de\24\3\2\2\2\u00df")  
 buf.write("\u00e0\7u\2\2\u00e0\u00e1\7v\2\2\u00e1\u00e2\7t\2\2\u00e2")  
 buf.write("\u00e3\7w\2\2\u00e3\u00e4\7e\2\2\u00e4\u00e5\7v\2\2\u00e5")  
 buf.write("\26\3\2\2\2\u00e6\u00e7\7e\2\2\u00e7\u00e8\7j\2\2\u00e8")  
 buf.write("\u00e9\7c\2\2\u00e9\u00ea\7p\2\2\u00ea\30\3\2\2\2\u00eb")  
 buf.write("\u00ec\7g\2\2\u00ec\u00ed\7n\2\2\u00ed\u00ee\7u\2\2\u00ee")  
 buf.write("\u00ef\7g\2\2\u00ef\32\3\2\2\2\u00f0\u00f1\7i\2\2\u00f1")  
 buf.write("\u00f2\7q\2\2\u00f2\u00f3\7v\2\2\u00f3\u00f4\7q\2\2\u00f4")  
 buf.write("\34\3\2\2\2\u00f5\u00f6\7r\2\2\u00f6\u00f7\7c\2\2\u00f7")  
 buf.write("\u00f8\7e\2\2\u00f8\u00f9\7m\2\2\u00f9\u00fa\7c\2\2\u00fa")  
 buf.write("\u00fb\7i\2\2\u00fb\u00fc\7g\2\2\u00fc\36\3\2\2\2\u00fd")  
 buf.write("\u00fe\7u\2\2\u00fe\u00ff\7y\2\2\u00ff\u0100\7k\2\2\u0100")  
 buf.write("\u0101\7v\2\2\u0101\u0102\7e\2\2\u0102\u0103\7j\2\2\u0103")  
 buf.write(" \3\2\2\2\u0104\u0105\7e\2\2\u0105\u0106\7q\2\2\u0106")  
 buf.write("\u0107\7p\2\2\u0107\u0108\7u\2\2\u0108\u0109\7v\2\2\u0109")  
 buf.write("\"\3\2\2\2\u010a\u010b\7h\2\2\u010b\u010c\7c\2\2\u010c")  
 buf.write("\u010d\7n\2\2\u010d\u010e\7n\2\2\u010e\u010f\7v\2\2\u010f")  
 buf.write("\u0110\7j\2\2\u0110\u0111\7t\2\2\u0111\u0112\7q\2\2\u0112")  
 buf.write("\u0113\7w\2\2\u0113\u0114\7i\2\2\u0114\u0115\7j\2\2\u0115")  
 buf.write("$\3\2\2\2\u0116\u0117\7k\2\2\u0117\u0118\7h\2\2\u0118")  
 buf.write("&\3\2\2\2\u0119\u011a\7t\2\2\u011a\u011b\7c\2\2\u011b")  
 buf.write("\u011c\7p\2\2\u011c\u011d\7i\2\2\u011d\u011e\7g\2\2\u011e")  
 buf.write("(\3\2\2\2\u011f\u0120\7v\2\2\u0120\u0121\7{\2\2\u0121")  
 buf.write("\u0122\7r\2\2\u0122\u0123\7g\2\2\u0123\*\3\2\2\2\u0124")  
 buf.write("\u0125\7e\2\2\u0125\u0126\7q\2\2\u0126\u0127\7p\2\2\u0127")  
 buf.write("\u0128\7v\2\2\u0128\u0129\7k\2\2\u0129\u012a\7p\2\2\u012a")  
 buf.write("\u012b\7w\2\2\u012b\u012c\7g\2\2\u012c,\3\2\2\2\u012d")  
 buf.write("\u012e\7h\2\2\u012e\u012f\7q\2\2\u012f\u0130\7t\2\2\u0130")  
 buf.write(".\3\2\2\2\u0131\u0132\7k\2\2\u0132\u0133\7o\2\2\u0133")  
 buf.write("\u0134\7r\2\2\u0134\u0135\7q\2\2\u0135\u0136\7t\2\2\u0136")  
 buf.write("\u0137\7v\2\2\u0137\60\3\2\2\2\u0138\u0139\7t\2\2\u0139")  
 buf.write("\u013a\7g\2\2\u013a\u013b\7v\2\2\u013b\u013c\7w\2\2\u013c")  
 buf.write("\u013d\7t\2\2\u013d\u013e\7p\2\2\u013e\62\3\2\2\2\u013f")  
 buf.write("\u0140\7x\2\2\u0140\u0141\7c\2\2\u0141\u0142\7t\2\2\u0142")  
 buf.write("\64\3\2\2\2\u0143\u0144\7p\2\2\u0144\u0145\7k\2\2\u0145")  
 buf.write("\u0146\7n\2\2\u0146\66\3\2\2\2\u0147\u014c\5\u00a3R\2")  
 buf.write("\u0148\u014b\5\u00a3R\2\u0149\u014b\5\u00a5S\2\u014a\u0148")  
 buf.write("\3\2\2\2\u014a\u0149\3\2\2\2\u014b\u014e\3\2\2\2\u014c")  
 buf.write("\u014a\3\2\2\2\u014c\u014d\3\2\2\2\u014d8\3\2\2\2\u014e")  
 buf.write("\u014c\3\2\2\2\u014f\u0150\7\*\2\2\u0150:\3\2\2\2\u0151")  
 buf.write("\u0152\7+\2\2\u0152<\3\2\2\2\u0153\u0154\7}\2\2\u0154")  
 buf.write(">\3\2\2\2\u0155\u0156\7\177\2\2\u0156@\3\2\2\2\u0157\u0158")  
 buf.write("\7]\2\2\u0158B\3\2\2\2\u0159\u015a\7\_\2\2\u015aD\3\2\2")  
 buf.write("\2\u015b\u015c\7?\2\2\u015cF\3\2\2\2\u015d\u015e\7.\2")  
 buf.write("\2\u015eH\3\2\2\2\u015f\u0160\7=\2\2\u0160J\3\2\2\2\u0161")  
 buf.write("\u0162\7<\2\2\u0162L\3\2\2\2\u0163\u0164\7\60\2\2\u0164")  
 buf.write("N\3\2\2\2\u0165\u0166\7-\2\2\u0166\u0167\7-\2\2\u0167")  
 buf.write("P\3\2\2\2\u0168\u0169\7/\2\2\u0169\u016a\7/\2\2\u016a")  
 buf.write("R\3\2\2\2\u016b\u016c\7<\2\2\u016c\u016d\7?\2\2\u016d")  
 buf.write("T\3\2\2\2\u016e\u016f\7\60\2\2\u016f\u0170\7\60\2\2\u0170")  
 buf.write("\u0171\7\60\2\2\u0171V\3\2\2\2\u0172\u0173\7~\2\2\u0173")  
 buf.write("\u0174\7~\2\2\u0174X\3\2\2\2\u0175\u0176\7(\2\2\u0176")  
 buf.write("\u0177\7(\2\2\u0177Z\3\2\2\2\u0178\u0179\7?\2\2\u0179")  
 buf.write("\u017a\7?\2\2\u017a\\\3\2\2\2\u017b\u017c\7#\2\2\u017c")  
 buf.write("\u017d\7?\2\2\u017d^\3\2\2\2\u017e\u017f\7>\2\2\u017f")  
 buf.write("`\3\2\2\2\u0180\u0181\7>\2\2\u0181\u0182\7?\2\2\u0182")  
 buf.write("b\3\2\2\2\u0183\u0184\7@\2\2\u0184d\3\2\2\2\u0185\u0186")  
 buf.write("\7@\2\2\u0186\u0187\7?\2\2\u0187f\3\2\2\2\u0188\u0189")  
 buf.write("\7~\2\2\u0189h\3\2\2\2\u018a\u018b\7\61\2\2\u018bj\3\2")  
 buf.write("\2\2\u018c\u018d\7\'\2\2\u018dl\3\2\2\2\u018e\u018f\7")  
 buf.write(">\2\2\u018f\u0190\7>\2\2\u0190n\3\2\2\2\u0191\u0192\7")  
 buf.write("@\2\2\u0192\u0193\7@\2\2\u0193p\3\2\2\2\u0194\u0195\7")  
 buf.write("(\2\2\u0195\u0196\7`\2\2\u0196r\3\2\2\2\u0197\u0198\7")  
 buf.write("#\2\2\u0198t\3\2\2\2\u0199\u019a\7-\2\2\u019av\3\2\2\2")  
 buf.write("\u019b\u019c\7/\2\2\u019cx\3\2\2\2\u019d\u019e\7`\2\2")  
 buf.write("\u019ez\3\2\2\2\u019f\u01a0\7,\2\2\u01a0|\3\2\2\2\u01a1")  
 buf.write("\u01a2\7(\2\2\u01a2~\3\2\2\2\u01a3\u01a4\7>\2\2\u01a4")  
 buf.write("\u01a5\7/\2\2\u01a5\u0080\3\2\2\2\u01a6\u01aa\t\2\2\2")  
 buf.write("\u01a7\u01a9\t\3\2\2\u01a8\u01a7\3\2\2\2\u01a9\u01ac\3")  
 buf.write("\2\2\2\u01aa\u01a8\3\2\2\2\u01aa\u01ab\3\2\2\2\u01ab\u0082")  
 buf.write("\3\2\2\2\u01ac\u01aa\3\2\2\2\u01ad\u01b1\7\62\2\2\u01ae")  
 buf.write("\u01b0\5\u009dO\2\u01af\u01ae\3\2\2\2\u01b0\u01b3\3\2")  
 buf.write("\2\2\u01b1\u01af\3\2\2\2\u01b1\u01b2\3\2\2\2\u01b2\u0084")  
 buf.write("\3\2\2\2\u01b3\u01b1\3\2\2\2\u01b4\u01b5\7\62\2\2\u01b5")  
 buf.write("\u01b7\t\4\2\2\u01b6\u01b8\5\u009fP\2\u01b7\u01b6\3\2")  
 buf.write("\2\2\u01b8\u01b9\3\2\2\2\u01b9\u01b7\3\2\2\2\u01b9\u01ba")  
 buf.write("\3\2\2\2\u01ba\u0086\3\2\2\2\u01bb\u01c4\5\u009bN\2\u01bc")  
 buf.write("\u01be\7\60\2\2\u01bd\u01bf\5\u009bN\2\u01be\u01bd\3\2")  
 buf.write("\2\2\u01be\u01bf\3\2\2\2\u01bf\u01c1\3\2\2\2\u01c0\u01c2")  
 buf.write("\5\u00a1Q\2\u01c1\u01c0\3\2\2\2\u01c1\u01c2\3\2\2\2\u01c2")  
 buf.write("\u01c5\3\2\2\2\u01c3\u01c5\5\u00a1Q\2\u01c4\u01bc\3\2")  
 buf.write("\2\2\u01c4\u01c3\3\2\2\2\u01c5\u01cc\3\2\2\2\u01c6\u01c7")  
 buf.write("\7\60\2\2\u01c7\u01c9\5\u009bN\2\u01c8\u01ca\5\u00a1Q")  
 buf.write("\2\u01c9\u01c8\3\2\2\2\u01c9\u01ca\3\2\2\2\u01ca\u01cc")  
 buf.write("\3\2\2\2\u01cb\u01bb\3\2\2\2\u01cb\u01c6\3\2\2\2\u01cc")  
 buf.write("\u0088\3\2\2\2\u01cd\u01d0\5\u009bN\2\u01ce\u01d0\5\u0087")  
 buf.write("D\2\u01cf\u01cd\3\2\2\2\u01cf\u01ce\3\2\2\2\u01d0\u01d1")  
 buf.write("\3\2\2\2\u01d1\u01d2\7k\2\2\u01d2\u008a\3\2\2\2\u01d3")  
 buf.write("\u01d6\7)\2\2\u01d4\u01d7\n\5\2\2\u01d5\u01d7\5\u0099")  
 buf.write("M\2\u01d6\u01d4\3\2\2\2\u01d6\u01d5\3\2\2\2\u01d7\u01d8")  
 buf.write("\3\2\2\2\u01d8\u01d9\7)\2\2\u01d9\u008c\3\2\2\2\u01da")  
 buf.write("\u01de\7b\2\2\u01db\u01dd\n\6\2\2\u01dc\u01db\3\2\2\2")  
 buf.write("\u01dd\u01e0\3\2\2\2\u01de\u01dc\3\2\2\2\u01de\u01df\3")  
 buf.write("\2\2\2\u01df\u01e1\3\2\2\2\u01e0\u01de\3\2\2\2\u01e1\u01e2")  
 buf.write("\7b\2\2\u01e2\u008e\3\2\2\2\u01e3\u01e8\7$\2\2\u01e4\u01e7")  
 buf.write("\n\7\2\2\u01e5\u01e7\5\u0099M\2\u01e6\u01e4\3\2\2\2\u01e6")  
 buf.write("\u01e5\3\2\2\2\u01e7\u01ea\3\2\2\2\u01e8\u01e6\3\2\2\2")  
 buf.write("\u01e8\u01e9\3\2\2\2\u01e9\u01eb\3\2\2\2\u01ea\u01e8\3")  
 buf.write("\2\2\2\u01eb\u01ec\7$\2\2\u01ec\u0090\3\2\2\2\u01ed\u01ef")  
 buf.write("\t\b\2\2\u01ee\u01ed\3\2\2\2\u01ef\u01f0\3\2\2\2\u01f0")  
 buf.write("\u01ee\3\2\2\2\u01f0\u01f1\3\2\2\2\u01f1\u01f2\3\2\2\2")  
 buf.write("\u01f2\u01f3\bI\2\2\u01f3\u0092\3\2\2\2\u01f4\u01f5\7")  
 buf.write("\61\2\2\u01f5\u01f6\7,\2\2\u01f6\u01fa\3\2\2\2\u01f7\u01f9")  
 buf.write("\13\2\2\2\u01f8\u01f7\3\2\2\2\u01f9\u01fc\3\2\2\2\u01fa")  
 buf.write("\u01fb\3\2\2\2\u01fa\u01f8\3\2\2\2\u01fb\u01fd\3\2\2\2")  
 buf.write("\u01fc\u01fa\3\2\2\2\u01fd\u01fe\7,\2\2\u01fe\u01ff\7")  
 buf.write("\61\2\2\u01ff\u0200\3\2\2\2\u0200\u0201\bJ\2\2\u0201\u0094")  
 buf.write("\3\2\2\2\u0202\u0204\t\t\2\2\u0203\u0202\3\2\2\2\u0204")  
 buf.write("\u0205\3\2\2\2\u0205\u0203\3\2\2\2\u0205\u0206\3\2\2\2")  
 buf.write("\u0206\u0207\3\2\2\2\u0207\u0208\bK\2\2\u0208\u0096\3")  
 buf.write("\2\2\2\u0209\u020a\7\61\2\2\u020a\u020b\7\61\2\2\u020b")  
 buf.write("\u020f\3\2\2\2\u020c\u020e\n\t\2\2\u020d\u020c\3\2\2\2")  
 buf.write("\u020e\u0211\3\2\2\2\u020f\u020d\3\2\2\2\u020f\u0210\3")  
 buf.write("\2\2\2\u0210\u0212\3\2\2\2\u0211\u020f\3\2\2\2\u0212\u0213")  
 buf.write("\bL\2\2\u0213\u0098\3\2\2\2\u0214\u022e\7^\2\2\u0215\u0216")  
 buf.write("\7w\2\2\u0216\u0217\5\u009fP\2\u0217\u0218\5\u009fP\2")  
 buf.write("\u0218\u0219\5\u009fP\2\u0219\u021a\5\u009fP\2\u021a\u022f")  
 buf.write("\3\2\2\2\u021b\u021c\7W\2\2\u021c\u021d\5\u009fP\2\u021d")  
 buf.write("\u021e\5\u009fP\2\u021e\u021f\5\u009fP\2\u021f\u0220\5")  
 buf.write("\u009fP\2\u0220\u0221\5\u009fP\2\u0221\u0222\5\u009fP")  
 buf.write("\2\u0222\u0223\5\u009fP\2\u0223\u0224\5\u009fP\2\u0224")  
 buf.write("\u022f\3\2\2\2\u0225\u022f\t\n\2\2\u0226\u0227\5\u009d")  
 buf.write("O\2\u0227\u0228\5\u009dO\2\u0228\u0229\5\u009dO\2\u0229")  
 buf.write("\u022f\3\2\2\2\u022a\u022b\7z\2\2\u022b\u022c\5\u009f")  
 buf.write("P\2\u022c\u022d\5\u009fP\2\u022d\u022f\3\2\2\2\u022e\u0215")  
 buf.write("\3\2\2\2\u022e\u021b\3\2\2\2\u022e\u0225\3\2\2\2\u022e")  
 buf.write("\u0226\3\2\2\2\u022e\u022a\3\2\2\2\u022f\u009a\3\2\2\2")  
 buf.write("\u0230\u0232\t\3\2\2\u0231\u0230\3\2\2\2\u0232\u0233\3")  
 buf.write("\2\2\2\u0233\u0231\3\2\2\2\u0233\u0234\3\2\2\2\u0234\u009c")  
 buf.write("\3\2\2\2\u0235\u0236\t\13\2\2\u0236\u009e\3\2\2\2\u0237")  
 buf.write("\u0238\t\f\2\2\u0238\u00a0\3\2\2\2\u0239\u023b\t\r\2\2")  
 buf.write("\u023a\u023c\t\16\2\2\u023b\u023a\3\2\2\2\u023b\u023c")  
 buf.write("\3\2\2\2\u023c\u023d\3\2\2\2\u023d\u023e\5\u009bN\2\u023e")  
 buf.write("\u00a2\3\2\2\2\u023f\u0242\5\u00a7T\2\u0240\u0242\7a\2")  
 buf.write("\2\u0241\u023f\3\2\2\2\u0241\u0240\3\2\2\2\u0242\u00a4")  
 buf.write("\3\2\2\2\u0243\u0245\t\17\2\2\u0244\u0243\3\2\2\2\u0245")  
 buf.write("\u00a6\3\2\2\2\u0246\u0248\t\20\2\2\u0247\u0246\3\2\2")  
 buf.write("\2\u0248\u00a8\3\2\2\2\34\2\u014a\u014c\u01aa\u01b1\u01b9")  
 buf.write("\u01be\u01c1\u01c4\u01c9\u01cb\u01cf\u01d6\u01de\u01e6")  
 buf.write("\u01e8\u01f0\u01fa\u0205\u020f\u022e\u0233\u023b\u0241")  
 buf.write("\u0244\u0247\3\2\3\2")  
 return buf.getvalue()  
  
  
class LexerGo(Lexer):  
  
 atn = ATNDeserializer().deserialize(serializedATN())  
  
 decisionsToDFA = [ DFA(ds, i) for i, ds in enumerate(atn.decisionToState) ]  
  
 BREAK = 1  
 DEFAULT = 2  
 FUNC = 3  
 INTERFACE = 4  
 SELECT = 5  
 CASE = 6  
 DEFER = 7  
 GO = 8  
 MAP = 9  
 STRUCT = 10  
 CHAN = 11  
 ELSE = 12  
 GOTO = 13  
 PACKAGE = 14  
 SWITCH = 15  
 CONST = 16  
 FALLTHROUGH = 17  
 IF = 18  
 RANGE = 19  
 TYPE = 20  
 CONTINUE = 21  
 FOR = 22  
 IMPORT = 23  
 RETURN = 24  
 VAR = 25  
 NIL\_LIT = 26  
 IDENTIFIER = 27  
 L\_PAREN = 28  
 R\_PAREN = 29  
 L\_CURLY = 30  
 R\_CURLY = 31  
 L\_BRACKET = 32  
 R\_BRACKET = 33  
 ASSIGN = 34  
 COMMA = 35  
 SEMI = 36  
 COLON = 37  
 DOT = 38  
 PLUS\_PLUS = 39  
 MINUS\_MINUS = 40  
 DECLARE\_ASSIGN = 41  
 ELLIPSIS = 42  
 LOGICAL\_OR = 43  
 LOGICAL\_AND = 44  
 EQUALS = 45  
 NOT\_EQUALS = 46  
 LESS = 47  
 LESS\_OR\_EQUALS = 48  
 GREATER = 49  
 GREATER\_OR\_EQUALS = 50  
 OR = 51  
 DIV = 52  
 MOD = 53  
 LSHIFT = 54  
 RSHIFT = 55  
 BIT\_CLEAR = 56  
 EXCLAMATION = 57  
 PLUS = 58  
 MINUS = 59  
 CARET = 60  
 STAR = 61  
 AMPERSAND = 62  
 RECEIVE = 63  
 DECIMAL\_LIT = 64  
 OCTAL\_LIT = 65  
 HEX\_LIT = 66  
 FLOAT\_LIT = 67  
 IMAGINARY\_LIT = 68  
 RUNE\_LIT = 69  
 RAW\_STRING\_LIT = 70  
 INTERPRETED\_STRING\_LIT = 71  
 WS = 72  
 COMMENT = 73  
 TERMINATOR = 74  
 LINE\_COMMENT = 75  
  
 channelNames = [ u"DEFAULT\_TOKEN\_CHANNEL", u"HIDDEN" ]  
  
 modeNames = [ "DEFAULT\_MODE" ]  
  
 literalNames = [ "<INVALID>",  
 "'break'", "'default'", "'func'", "'interface'", "'select'",   
 "'case'", "'defer'", "'go'", "'map'", "'struct'", "'chan'",   
 "'else'", "'goto'", "'package'", "'switch'", "'const'", "'fallthrough'",   
 "'if'", "'range'", "'type'", "'continue'", "'for'", "'import'",   
 "'return'", "'var'", "'nil'", "'('", "')'", "'{'", "'}'", "'['",   
 "']'", "'='", "','", "';'", "':'", "'.'", "'++'", "'--'", "':='",   
 "'...'", "'||'", "'&&'", "'=='", "'!='", "'<'", "'<='", "'>'",   
 "'>='", "'|'", "'/'", "'%'", "'<<'", "'>>'", "'&^'", "'!'",   
 "'+'", "'-'", "'^'", "'\*'", "'&'", "'<-'" ]  
  
 symbolicNames = [ "<INVALID>",  
 "BREAK", "DEFAULT", "FUNC", "INTERFACE", "SELECT", "CASE", "DEFER",   
 "GO", "MAP", "STRUCT", "CHAN", "ELSE", "GOTO", "PACKAGE", "SWITCH",   
 "CONST", "FALLTHROUGH", "IF", "RANGE", "TYPE", "CONTINUE", "FOR",   
 "IMPORT", "RETURN", "VAR", "NIL\_LIT", "IDENTIFIER", "L\_PAREN",   
 "R\_PAREN", "L\_CURLY", "R\_CURLY", "L\_BRACKET", "R\_BRACKET", "ASSIGN",   
 "COMMA", "SEMI", "COLON", "DOT", "PLUS\_PLUS", "MINUS\_MINUS",   
 "DECLARE\_ASSIGN", "ELLIPSIS", "LOGICAL\_OR", "LOGICAL\_AND", "EQUALS",   
 "NOT\_EQUALS", "LESS", "LESS\_OR\_EQUALS", "GREATER", "GREATER\_OR\_EQUALS",   
 "OR", "DIV", "MOD", "LSHIFT", "RSHIFT", "BIT\_CLEAR", "EXCLAMATION",   
 "PLUS", "MINUS", "CARET", "STAR", "AMPERSAND", "RECEIVE", "DECIMAL\_LIT",   
 "OCTAL\_LIT", "HEX\_LIT", "FLOAT\_LIT", "IMAGINARY\_LIT", "RUNE\_LIT",   
 "RAW\_STRING\_LIT", "INTERPRETED\_STRING\_LIT", "WS", "COMMENT",   
 "TERMINATOR", "LINE\_COMMENT" ]  
 ruleNames = [ "BREAK", "DEFAULT", "FUNC", "INTERFACE", "SELECT", "CASE",   
 "DEFER", "GO", "MAP", "STRUCT", "CHAN", "ELSE", "GOTO",   
 "PACKAGE", "SWITCH", "CONST", "FALLTHROUGH", "IF", "RANGE",   
 "TYPE", "CONTINUE", "FOR", "IMPORT", "RETURN", "VAR",   
 "NIL\_LIT", "IDENTIFIER", "L\_PAREN", "R\_PAREN", "L\_CURLY",   
 "R\_CURLY", "L\_BRACKET", "R\_BRACKET", "ASSIGN", "COMMA",   
 "SEMI", "COLON", "DOT", "PLUS\_PLUS", "MINUS\_MINUS", "DECLARE\_ASSIGN",   
 "ELLIPSIS", "LOGICAL\_OR", "LOGICAL\_AND", "EQUALS", "NOT\_EQUALS",   
 "LESS", "LESS\_OR\_EQUALS", "GREATER", "GREATER\_OR\_EQUALS",   
 "OR", "DIV", "MOD", "LSHIFT", "RSHIFT", "BIT\_CLEAR", "EXCLAMATION",   
 "PLUS", "MINUS", "CARET", "STAR", "AMPERSAND", "RECEIVE",   
 "DECIMAL\_LIT", "OCTAL\_LIT", "HEX\_LIT", "FLOAT\_LIT", "IMAGINARY\_LIT",   
 "RUNE\_LIT", "RAW\_STRING\_LIT", "INTERPRETED\_STRING\_LIT",   
 "WS", "COMMENT", "TERMINATOR", "LINE\_COMMENT", "ESCAPED\_VALUE",   
 "DECIMALS", "OCTAL\_DIGIT", "HEX\_DIGIT", "EXPONENT", "LETTER",   
 "UNICODE\_DIGIT", "UNICODE\_LETTER" ]  
  
 grammarFileName = "LexerGo.g4"

# 1.2 Структура данных - Токен

Эта структура данных используется парсером и лексером. Представляет собой пару значений – Тип токена и значение токена. Может возвращать оба значения, а также переопределять или устанавливать собственный тип.

Листинг токенов

BREAK : 'break';  
DEFAULT : 'default';  
FUNC : 'func';  
INTERFACE : 'interface';  
SELECT : 'select';  
CASE : 'case';  
DEFER : 'defer';  
GO : 'go';  
MAP : 'map';  
STRUCT : 'struct';  
CHAN : 'chan';  
ELSE : 'else';  
GOTO : 'goto';  
PACKAGE : 'package';  
SWITCH : 'switch';  
CONST : 'const';  
FALLTHROUGH : 'fallthrough';  
IF : 'if';  
RANGE : 'range';  
TYPE : 'type';  
CONTINUE : 'continue';  
FOR : 'for';  
IMPORT : 'import';  
RETURN : 'return';  
VAR : 'var';  
  
NIL\_LIT : 'nil';  
  
IDENTIFIER : LETTER (LETTER | UNICODE\_DIGIT)\*;  
  
*// Punctuation*L\_PAREN : '(';  
R\_PAREN : ')';  
L\_CURLY : '{';  
R\_CURLY : '}';  
L\_BRACKET : '[';  
R\_BRACKET : ']';  
ASSIGN : '=';  
COMMA : ',';  
SEMI : ';';  
COLON : ':';  
DOT : '.';  
PLUS\_PLUS : '++';  
MINUS\_MINUS : '--';  
DECLARE\_ASSIGN : ':=';  
ELLIPSIS : '...';  
  
LOGICAL\_OR : '||';  
LOGICAL\_AND : '&&';  
  
EQUALS : '==';  
NOT\_EQUALS : '!=';  
LESS : '<';  
LESS\_OR\_EQUALS : '<=';  
GREATER : '>';  
GREATER\_OR\_EQUALS : '>=';  
  
OR : '|';  
DIV : '/';  
MOD : '%';  
LSHIFT : '<<';  
RSHIFT : '>>';  
BIT\_CLEAR : '&^';  
  
EXCLAMATION : '!';  
  
PLUS : '+';  
MINUS : '-';  
CARET : '^';  
STAR : '\*';  
AMPERSAND : '&';  
RECEIVE : '<-';  
  
DECIMAL\_LIT : [1-9] [0-9]\*;  
OCTAL\_LIT : '0' OCTAL\_DIGIT\*;  
HEX\_LIT : '0' [xX] HEX\_DIGIT+;  
  
FLOAT\_LIT : DECIMALS ('.' DECIMALS? EXPONENT? | EXPONENT)  
 | '.' DECIMALS EXPONENT?  
 ;  
  
IMAGINARY\_LIT : (DECIMALS | FLOAT\_LIT) 'i';  
  
RUNE\_LIT : '\'' (~[\n\\] | ESCAPED\_VALUE) '\'';  
  
RAW\_STRING\_LIT : '`' ~'`'\* '`';  
INTERPRETED\_STRING\_LIT : '"' (~["\\] | ESCAPED\_VALUE)\* '"';  
  
WS : [ \t]+ -> channel(HIDDEN);  
COMMENT : '/\*' .\*? '\*/' -> channel(HIDDEN);  
TERMINATOR : [\r\n]+ -> channel(HIDDEN);  
LINE\_COMMENT : '//' ~[\r\n]\* -> channel(HIDDEN);

# 1.3 Проектирование парсера

Вторая фаза компилятора – это синтаксический анализ. На вход синтаксического анализатора подаются строки из первичного дерева. Данный компонент принимает список токенов на входе и создает на выходе дерево разбора. Как было описано выше, здесь используется библиотека и плагин antlr4. При помощи двух компонентов парсера и лексера создаём работающий компилятор, интерпретирующий язык. В Листинге 3 приведен код парсера. Парсер – модуль программы, предназначенный для чтения списка токенов программы и генерации списка инструкций - байтового представления кодов предусмотренных операций (имеющих тип

Метод parse начинает работу с правила sourceFile(), которое в свою очередь передает работу остальным правилам.

def sourceFile(self):  
  
 localctx = ParserGo.SourceFileContext(self, self.\_ctx, self.state)  
 self.enterRule(localctx, 0, self.RULE\_sourceFile)  
 self.\_la = 0 # Token type  
 try:  
 self.enterOuterAlt(localctx, 1)  
 self.state = 198  
 self.packageClause()  
 self.state = 199  
 self.eos()  
 self.state = 205  
 self.\_errHandler.sync(self)  
 \_la = self.\_input.LA(1)  
 while \_la==ParserGo.IMPORT:  
 self.state = 200  
 self.importDecl()  
 self.state = 201  
 self.eos()  
 self.state = 207  
 self.\_errHandler.sync(self)  
 \_la = self.\_input.LA(1)  
  
 self.state = 217  
 self.\_errHandler.sync(self)  
 \_la = self.\_input.LA(1)  
 while (((\_la) & ~0x3f) == 0 and ((1 << \_la) & ((1 << ParserGo.FUNC) | (1 << ParserGo.CONST) | (1 << ParserGo.TYPE) | (1 << ParserGo.VAR))) != 0):  
 self.state = 211  
 self.\_errHandler.sync(self)  
 la\_ = self.\_interp.adaptivePredict(self.\_input,1,self.\_ctx)  
 if la\_ == 1:  
 self.state = 208  
 self.functionDecl()  
 pass  
  
 elif la\_ == 2:  
 self.state = 209  
 self.methodDecl()  
 pass  
  
 elif la\_ == 3:  
 self.state = 210  
 self.declaration()  
 pass  
  
  
 self.state = 213  
 self.eos()  
 self.state = 219  
 self.\_errHandler.sync(self)  
 \_la = self.\_input.LA(1)  
  
 except RecognitionException as re:  
 localctx.exception = re  
 self.\_errHandler.reportError(self, re)  
 self.\_errHandler.recover(self, re)  
 finally:  
 self.exitRule()  
 return localctx

}

# После получения дерева разбора переводим его в АСТ с помощью доработанной библиотеки antlr-ast.

def generatedAst(text):  
  
 ParseTree = ast.parse(text, start="sourceFile")  
  
 old\_tree = ast.process\_tree(ParseTree)  
  
 return old\_tree

# 1.4 Проектирование транслятора

При проектировании транслятора использовалась библиотека llvmlite, которая представляет возможность для создания промежуточного кода ir-llvm.

Частичный исходный код инструкций:

class GeneratedIR():  
 reg = 1  
  
 header\_text = ""  
 main\_text = ""  
 buffer = ""  
 str\_i = 0  
 main\_reg = 1  
 br = 0  
 stack\_pop = True  
  
 br\_stack = []  
 br\_loop = []  
  
 def String\_generate(self):  
 self.main\_text += self.buffer  
 self.formatMainText()  
 text = ""  
 text += "\n"  
 text += self.header\_text  
 text += "define i32 @main() nounwind {\n"  
 text += str(self.main\_text)  
 text += " ret i32 0\n"  
 text += "}\n"  
 file = open("../lexanalyz/output.ll", 'w')  
 if file:  
 file.write(text)  
 else:  
 print("String\_generate ERROR!")  
 return text  
  
 def function\_start(self, id):  
 self.main\_text += self.buffer  
 main\_reg = self.reg  
 self.buffer = "define main @ " + id + " () nounwind {\n"  
 self.reg = 1  
  
 def function\_end(self):  
 self.buffer += "ret void\n"  
 self.formatself()  
 self.buffer += "}\n\n"  
 self.header\_text += str(self.buffer)  
 self.buffer = ""  
 self.reg = self.main\_reg  
  
 def call(self, id):  
 self.buffer += "call void @" + id + "()\n"

Также в этом модуле происходит обьединение всех используемых инструкций в один файл \*.ll

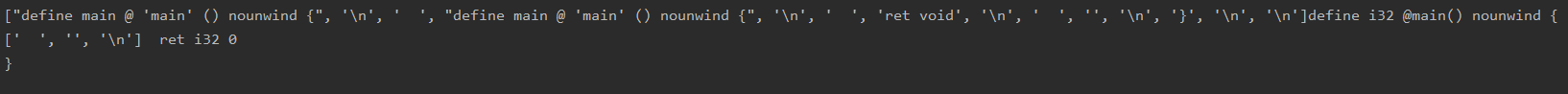
# 2.1 Упаковка приложения

Приложение находится в папке lexanalyz/ в виде нескольких файлов python-проекта, с подкаталогом gen, где лежат файлы парсера и лексера, и подкаталогом library с библиотекой antlr-ast.

# 2.5 Руководство пользователя

Для запуска компилятора необходимо заполнить файл test.go тестовой программой. Далее запустить файл Main.py. После этого будет создан файл output.ll, содержащий в себе промежуточный ir-llvm код.

# 3. Тесты

****

**Рисунок 2. Тест объявления функции и присвоения**

# ЗАКЛЮЧЕНИЕ

В данной программе был реализован лексер и парсер подмножества языка Golang. В результате работы над транслятором возникли сложности с получением нижних узлов АСТ. В связи с чем было принято считывать токены вручную, что поставило очень сложную задачу, которая не была до конца реализована.

# Приложение

**Github:** https://github.com/kilmani/compiler