Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра интеллектуальных информационных технологий

**Лабораторная работа №1 по курсу «Языковые процессоры интеллектуальных систем» на тему:**

**«**Язык, описывающий математические вычисления**»**

Выполнил Шершень К.А.

студент группы

121702:

Проверил: Соколович М.Г.

**МИНСК 2024**

1. **Требования к разрабатываемому языку:**
   1. Встроенные типы не менее трех
   2. Возможность инициализация переменных всех типов при объявлении: <тип>? <имя\_переменной> = <выражение>
      1. Инициализирующее выражение может быть константным
   3. Встроенные операции не менее 10 штук
   4. Встроенные функции
      1. log, ln, sin, cos, tan, asin, acos, atan
   5. Использование сложных выражений (составных и со скобками)
   6. Блочный оператор
   7. Управляющие структуры
      1. Условный оператор (if-then-else)
      2. Оператор цикла с итерациями (for)
   8. Пользовательские подпрограммы
      1. Передача и возврат параметров
   9. Задание локальной и глобальной области видимости для имен переменных
   10. Объявление переменных
       1. Не явное
   11. Преобразование типов
       1. Явное, например, a = (int) b
   12. Оператор присваивания
       1. Многоцелевой, например, a, b = c, d
   13. Структуры, ограничивающие область видимости
       1. Подпрограммы и блочные операторы
   14. Маркер блочного оператора
       1. Не явный, например как в python
   15. Условные операторы
       1. Двух вариантный оператор if-then-else
   16. Перегрузка подпрограмм
       1. Присутствует
   17. Передача параметров в подпрограмму
       1. По значению и результату
   18. Допустимое место объявления подпрограмм:
       1. В любом месте программы, также и внутри другой подпрограммы.
   19. Встроенные типы: int, float
   20. Операции: переопределить +, -, \*, \ и т.д. для встроенных типов
2. **Вариант целевого кода:**
   1. Исполняемый файл (.exe), для генерации целевого кода использовать LLVM (http://llvm.org)
3. **Грамматика языка:**
   1. Лексер:

lexer grammar MathLangLexer;

PLUS: '+';

MINUS: '-';

MULT: '\*';

DIV: '/';

MOD: '%';

POW: '^';

ASSIGN: '=';

LPAREN: '(';

RPAREN: ')';

LBRACE: '{';

RBRACE: '}';

COLON: ':';

COMMA: ',';

SEMI: ';';

FUNC: 'func';

RETURN: 'return';

GT: '>';

LT: '<';

EQ: '==';

NEQ: '!=';

GTE: '>=';

LTE: '<=';

IF: 'if';

ELSE: 'else';

SWITCH: 'switch';

CASE: 'case';

DEFAULT: 'default';

FOR: 'for';

WHILE: 'while';

ID: [a-zA-Z\_][a-zA-Z0-9\_]\*;

INT\_LITERAL: [0-9]+;

FLOAT\_LITERAL: [0-9]+'.'[0-9]+;

WS: [ \t]+ -> skip;

NEWLINE: [\r\n]+ -> skip;

* 1. Парсер:

parser grammar *MathLangParser*;

options {

tokenVocab = MathLangLexer;

}

// Стартовое правило (программа может содержать функции и выражения)

program: (function\_definition | statement)+;

// Правило для подпрограмм

function\_definition

: *FUNC* *ID* *LPAREN* parameters? *RPAREN* *LBRACE* block *RBRACE*

;

// Параметры функции

parameters

: *ID* (*COMMA* *ID*)\*

;

block

: statement\*

;

// Выражение

expr

: expr *PLUS* expr

| expr *MINUS* expr

| expr *MULT* expr

| expr *DIV* expr

| expr *MOD* expr

| expr *POW* expr

| expr *GT* expr

| expr *LT* expr

| expr *EQ* expr

| expr *NEQ* expr

| expr *GTE* expr

| expr *LTE* expr

| *LPAREN* expr *RPAREN*

| function\_call

| if\_statement

| switch\_statement

| for\_statement

| while\_statement

| *INT\_LITERAL*

| *FLOAT\_LITERAL*

| *ID*

;

// Правило для условных операторов if-else

if\_statement

: *IF* *LPAREN* expr *RPAREN* *LBRACE* block *RBRACE* (*ELSE* *LBRACE* block *RBRACE*)?

;

// Правило для switch-case

switch\_statement

: *SWITCH* *LPAREN* expr *RPAREN* *LBRACE* case\_block *RBRACE*

;

case\_block

: (*CASE* expr *COLON* block)\* (*DEFAULT* *COLON* block)? # CaseBlock

;

update

: *ID* (*PLUS* *EQ* expr | *MINUS* *EQ* expr | *PLUS* *PLUS* | *MINUS* *MINUS*)?

;

// Правило для цикла for

for\_statement

: *FOR* *LPAREN* assignment expr *SEMI* update *RPAREN* *LBRACE* block *RBRACE*

;

// Правило для цикла while

while\_statement

: *WHILE* *LPAREN* expr *RPAREN* *LBRACE* block *RBRACE*

;

// Вызов функции

function\_call

: *ID* *LPAREN* argument\_list? *RPAREN*

;

// Аргументы функции

argument\_list

: expr (*COMMA* expr)\*

;

// Оператор присваивания

assignment

: *ID* *ASSIGN* expr *SEMI*

;

return\_statement

: *RETURN* expr *SEMI*

;

// Операторы в блоке

statement

: assignment

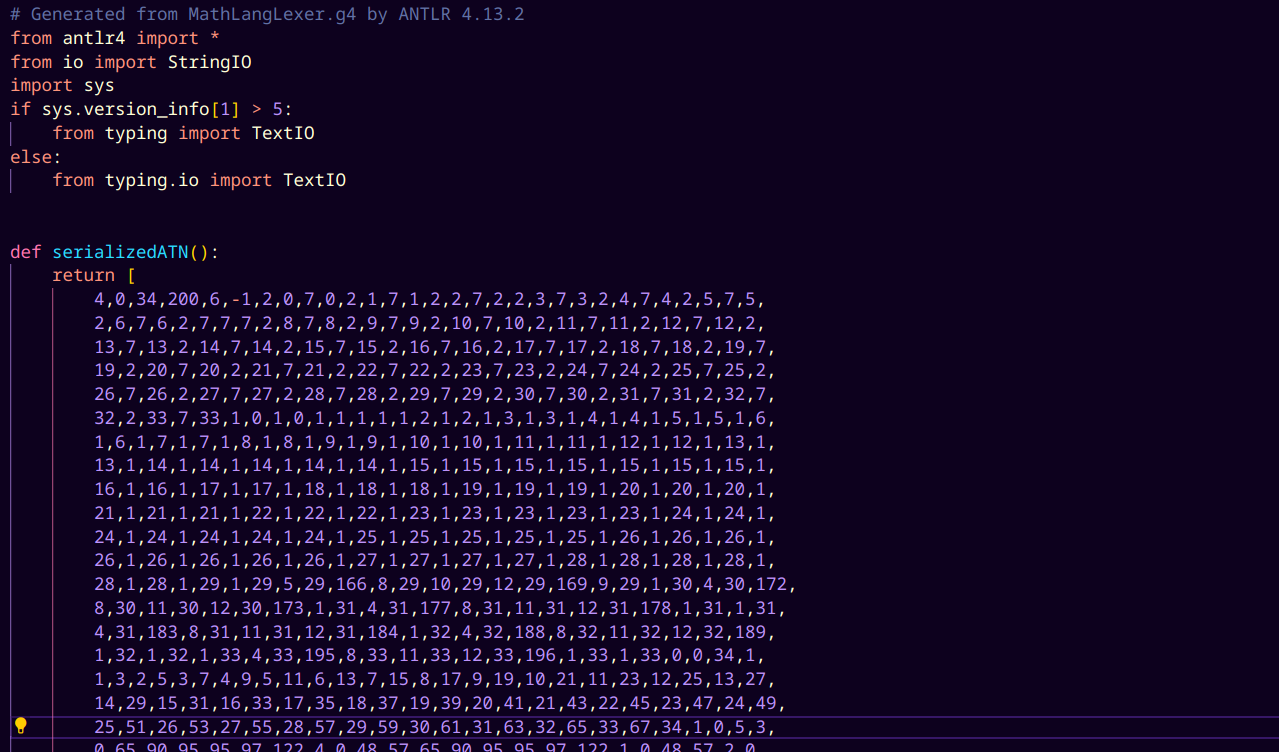
| expr *SEMI*

| return\_statement

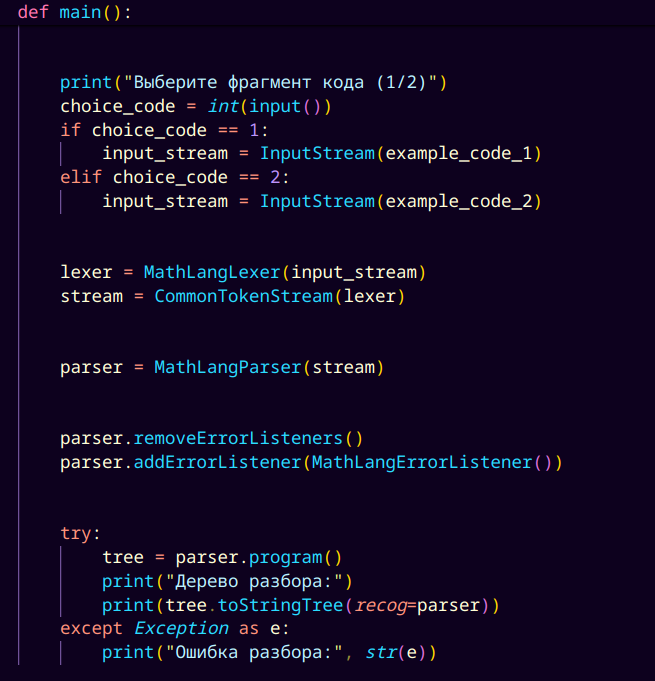
;

1. **Синтаксический анализатор:**

Синтаксический анализатор разработан на языке программирования Python:

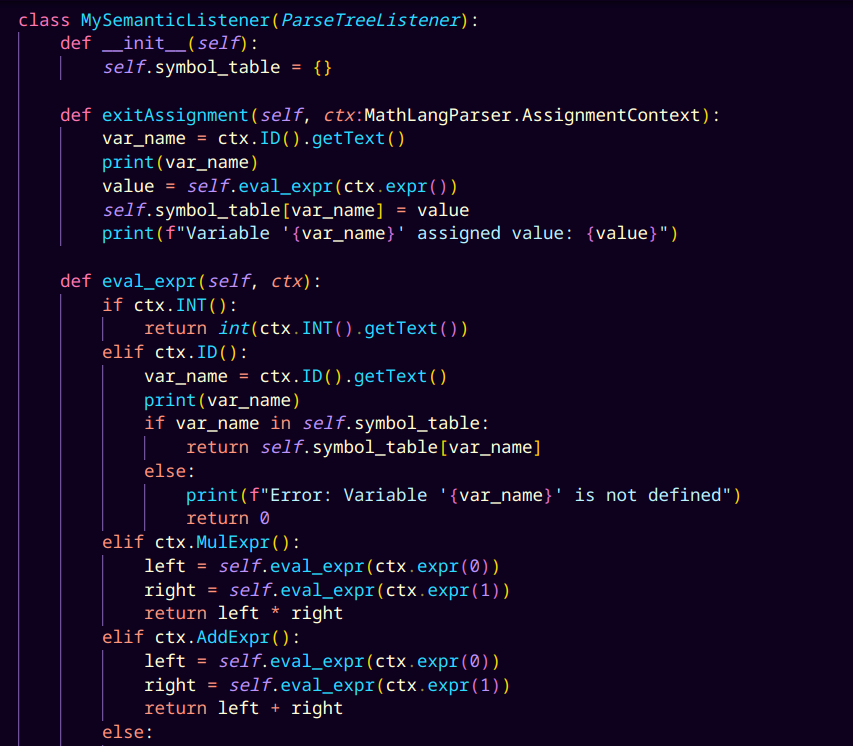


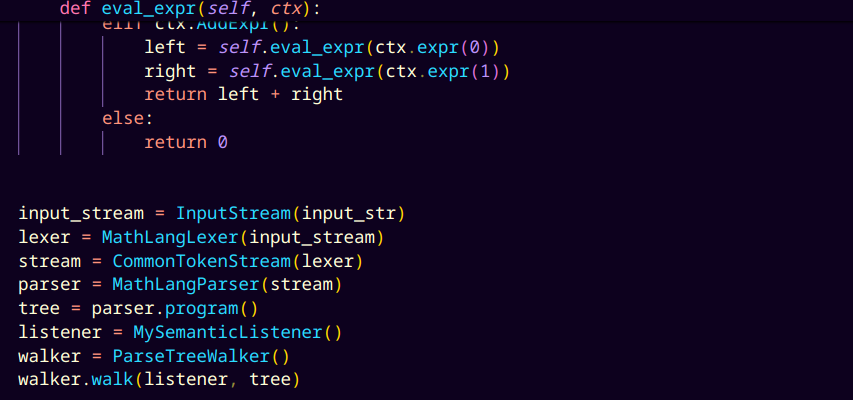
****



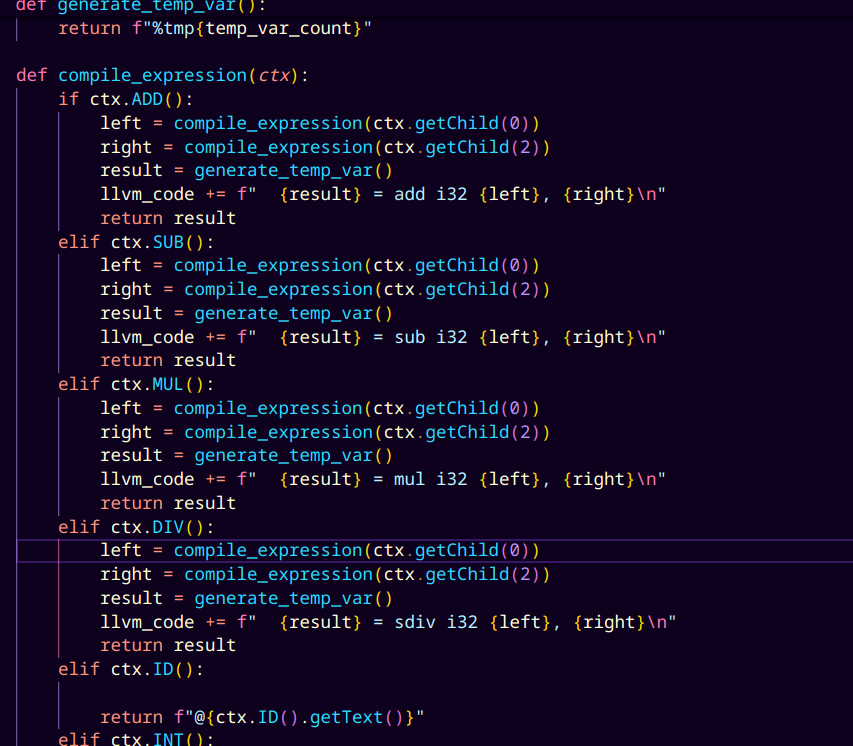
1. **Семантический анализатор:**

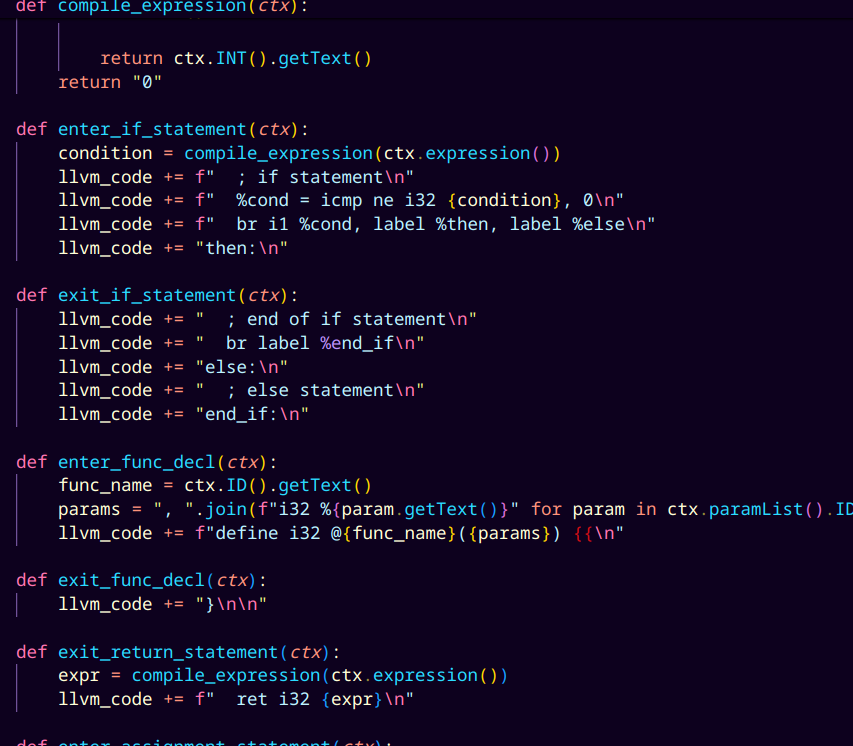
Семантический анализатор разработан на языке программирования Python:

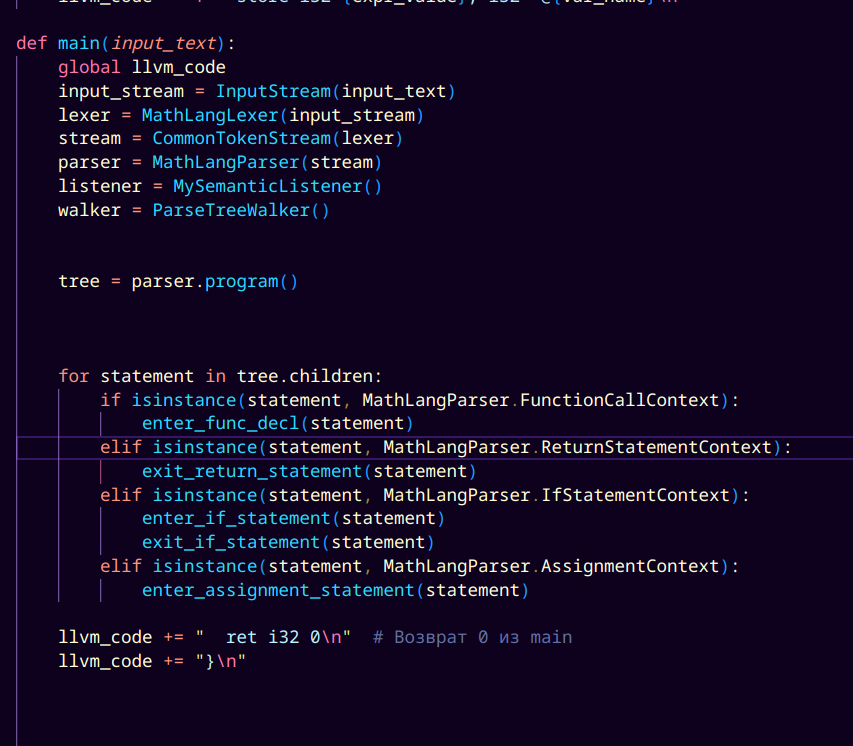




1. **Транслятор исходного кода с помощью LLVM:**

****

****

****

1. **Пример кода:**

//неявное обьявление переменных и многоцелевой оператор присваивания

int\_var,date := 10,2025

float\_var := 2.5

//неявное преобразование типов

sum\_var = int\_var + float\_var

//структура,ограничивающая область видимости и маркер блочного оператора

func cycle\_function(variable int,cycle\_number int) cycle\_varible int:

for i := 0, i <= cycle,i++:

variable += i

return cycle\_variable

// двух вариантный оператор

if data\_int < month\_int:

return 0

elif data\_int > month\_int:

return 1

else data\_int = month\_int:

return 2

// многовариантный switch case

switch i:

case 1:

-//-

case 2:

-//-

//перегрузка оператора присутствует

// перегрузка оператора == для сравнения int и float

if int\_var == float\_var:

return 0

// передача параметров в подпрограмму по ссылке

func\_input = link\_input(&var)

//различные операции языка

num1 := 2

num2 := 8

num\_sum := num\_1 + num\_2

num\_sub := num\_1 - num\_2

num\_mul := num\_1 \* num\_2

num\_div := num\_1 \ num\_2

num\_div2 := num\_1 % num\_2

num\_pow := num\_1 ^ num\_2

//функции языка

log\_var = log(2,3) //передаем float число и основание,возвращает логарифм числа по основанию

ln\_var = ln(2) //передаем float ,возвращает натуральный логарифм

sin\_var = sin(60) //передаем float угол ,возвращает синус угла

cos\_var = cos(30) //передаем float угол ,возвращает косинус угла

tan\_var = tan(45) //передаем float угол ,возвращает тангенс угла

asin\_var = asin(1) //передаем float синус ,возвращает угол для синуса

acos\_var = acos(0) //передаем float косинус ,возвращает угол для косинуса

atan\_var = atan(0.5) //передаем float тангенс,возвращает угол для тангенса

1. **Вывод:**

В ходе данной лабораторной работы были получены знания и навыки разработки грамматики языка, лексеров и парсеров. Также были разработаны синтаксический и семантический анализаторы, которые помогают выявить ошибки в коде перед компиляцией или же транслированием. Также был разработан транслятор исходного языка с использованием LLVM.