МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Домашняя работа №4

Выполнил:

Круглов Егор Ильич, P3324

Преподаватель:

Лаздин Артур Вячеславович

Санкт-Петербург

2025

Выполнение:

Код также выложен на github: https://github.com/KruglovEgor/Compilers/tree/main/lab4

Грамматика в формате g4:

```
grammar MyLang;

program: statement+ EOF;

statement:
    assignment
    | ifStatement
    | whileLoop
    | printStmt
    | inputStmt;

assignment: ID '=' expr ';';

ifStatement: 'if' '(' expr ')' block ('else' block)?;

whileLoop: 'while' '(' expr ')' block;

printStmt: 'print' '(' expr ')' ';';

inputStmt: 'input' '(' ID ')' ';';

block: '{' statement* '}' | statement;

expr:
    expr op=('*' | '/' | '%') expr    # MulDivMod
    | expr op=('+' | '-') expr        # AddSub
    | expr op=('>' | '<' | '>=' | '<=' | '==' | '!=') expr # Compare
    | NUMBER                          # Number
    | STRING                          # String
    | ID                              # Variable
    | '(' expr ')'                    # Parens
    | '-' expr                        # UnaryMinus
    | '!' expr                        # LogicalNot
    ;

// Лексер
ID: [a-zA-Z_][a-zA-Z0-9_]*;
NUMBER: [0-9]+;
STRING: '"' .*? '"';

// Пропускаем пробелы и переносы строк
WS: [ \t\n\r]+ -> skip;

// Комментарии
COMMENT: '//' ~[\r\n]* -> skip;
```

Интерпретатор:

```
from antlr4 import *
from MyLangLexer import MyLangLexer
from MyLangParser import MyLangParser
from MyLangVisitor import MyLangVisitor


class InterpreterError(Exception):
    pass
```

```
class MyInterpreter(MyLangVisitor):
    def __init__(self):
        self.vars = {}

    def visitProgram(self, ctx: MyLangParser.ProgramContext):
        for stmt in ctx.statement():
            self.visit(stmt)

    def visitAssignment(self, ctx: MyLangParser.AssignmentContext):
        var_name = ctx.ID().getText()
        value = self.visit(ctx.expr())
        self.vars[var_name] = value
        return value

    def visitPrintStmt(self, ctx: MyLangParser.PrintStmtContext):
        value = self.visit(ctx.expr())
        print(value)

    def visitInputStmt(self, ctx: MyLangParser.InputStmtContext):
        var_name = ctx.ID().getText()
        try:
            value = int(input())
            self.vars[var_name] = value
        except ValueError:
            raise InterpreterError("Ожидалось целое число")

    def visitIfStatement(self, ctx: MyLangParser.IfStatementContext):
        if self._is_true(self.visit(ctx.expr())):
            self.visit(ctx.block(0))
        elif ctx.block(1):
            self.visit(ctx.block(1))

    def visitWhileLoop(self, ctx: MyLangParser.WhileLoopContext):
        while self._is_true(self.visit(ctx.expr())):
            self.visit(ctx.block())

    def visitMulDivMod(self, ctx: MyLangParser.MulDivModContext):
        left = self.visit(ctx.expr(0))
        right = self.visit(ctx.expr(1))
        op = ctx.op.text

        if isinstance(left, str) or isinstance(right, str):
            raise InterpreterError("Нельзя выполнять арифметические операции со
строками")

        if op == '*':
            return left * right
        elif op == '/':
            if right == 0:
                raise InterpreterError("Деление на ноль")
            return left // right  # Целочисленное деление
        elif op == '%':
            if right == 0:
                raise InterpreterError("Деление по модулю на ноль")
            return left % right

    def visitAddSub(self, ctx: MyLangParser.AddSubContext):
        left = self.visit(ctx.expr(0))
        right = self.visit(ctx.expr(1))
        op = ctx.op.text

        if isinstance(left, str) or isinstance(right, str):
            return str(left) + str(right)

        return left + right if op == '+' else left - right

    def visitCompare(self, ctx: MyLangParser.CompareContext):
```

```
        left = self.visit(ctx.expr(0))
        right = self.visit(ctx.expr(1))
        op = ctx.op.text

        if type(left) != type(right):
            raise InterpreterError(f"Нельзя сравнивать {type(left)} и {type(right)}")

        return 1 if {
            '>': left > right,
            '<': left < right,
            '>=': left >= right,
            '<=': left <= right,
            '==': left == right,
            '!=': left != right
        }[op] else 0

    def visitNumber(self, ctx: MyLangParser.NumberContext):
        return int(ctx.NUMBER().getText())

    def visitString(self, ctx: MyLangParser.StringContext):
        return ctx.STRING().getText()[1:-1]

    def visitVariable(self, ctx: MyLangParser.VariableContext):
        var_name = ctx.ID().getText()
        if var_name not in self.vars:
            raise InterpreterError(f"Неинициализированная переменная '{var_name}'")
        return self.vars[var_name]

    def visitParens(self, ctx: MyLangParser.ParensContext):
        return self.visit(ctx.expr())

    def visitUnaryMinus(self, ctx: MyLangParser.UnaryMinusContext):
        return -self.visit(ctx.expr())

    def visitLogicalNot(self, ctx: MyLangParser.LogicalNotContext):
        value = self.visit(ctx.expr())
        return 1 if value == 0 else 0

    def _is_true(self, value):
        if isinstance(value, str):
            return len(value) > 0
        return value != 0


def run_interpreter(input_stream):
    lexer = MyLangLexer(input_stream)
    stream = CommonTokenStream(lexer)
    parser = MyLangParser(stream)
    tree = parser.program()

    interpreter = MyInterpreter()
    try:
        interpreter.visit(tree)
    except InterpreterError as e:
        print(f"Ошибка выполнения: {e}")


if __name__ == '__main__':
    import sys

    if len(sys.argv) != 2:
        print("Использование: python my_interpreter.py <файл>")
        exit(1)

    input_stream = FileStream(sys.argv[1], encoding='utf-8')
    run_interpreter(input_stream)
```

Тесты:

Нахождение числа Фибонначи:

```
print("Введите номер числа Фибоначчи:");
input(i);

if (i < 0){
    print("Число должно быть неотрицательным");
}
else{
    a = 0;
    b = 1;
    c = 0;

    if (i == 1){
        c = 1;
    }

    while (i - 1 > 0){
        c = a + b;
        a = b;
        b = c;
        i = i - 1;
    }
    print("Ваше число Фибоначчи:");
    print(c);
}
```

```
(venv) PS E:\GitHub\Compilers\lab4> python my_interpreter.py tests/fibonachi.mylang
Введите номер числа Фибоначчи:
0
Ваше число Фибоначчи:
0
(venv) PS E:\GitHub\Compilers\lab4> python my_interpreter.py tests/fibonachi.mylang
Введите номер числа Фибоначчи:
1
Ваше число Фибоначчи:
1
(venv) PS E:\GitHub\Compilers\lab4> python my_interpreter.py tests/fibonachi.mylang
Введите номер числа Фибоначчи:
8
Ваше число Фибоначчи:
21
(venv) PS E:\GitHub\Compilers\lab4> python my_interpreter.py tests/fibonachi.mylang
Введите номер числа Фибоначчи:
-1
Число должно быть неотрицательным
(venv) PS E:\GitHub\Compilers\lab4>
```

Нахождение НОД:

```
print("Нахождение НОД");
print("Введите первое число:");
input(a);
print("Введите второе число:");
input(b);

while (a * b != 0){
    if (a > b){
        a = a % b;
    }
    else{
        b = b % a;
    }
}

print("Получившийся НОД:");
print(a + b);
```

```
Нахождение НОД
Введите первое число:
63
Введите второе число:
49
Получившийся НОД:
7
(venv) PS E:\GitHub\Compilers\lab4> python my_interpreter.py tests/nod.mylang
Нахождение НОД
Введите первое число:
13
Введите второе число:
7
Получившийся НОД:
1
(venv) PS E:\GitHub\Compilers\lab4> python my_interpreter.py tests/nod.mylang
Нахождение НОД
Введите первое число:
50
Введите второе число:
125
Получившийся НОД:
25
```

Сложение чисел и строк:

```
a = 1;
b = 10;

c = "У нас ";
```

```
d = "получилось ";

print(a + b);
print(c + d);
print(c + d + a + b);
print(c + d + (a + b));
```

```
(venv) PS E:\GitHub\Compilers\lab4> python my_interpreter.py tests/sum.mylang
11
У нас получилось
У нас получилось 110
У нас получилось 11
```