Основное задание по курсу «Языки программирования».

Основной целью данного задания является изучение основ определения ЯП в терминах контекстно-свободных грамматик и проектирование интерпретаторов и компиляторов в рамках данного курса.

Для выполнения задания необходимо:

Разработать язык программирования, который должен реализовать следующие компоненты:

* Присваивание (оператор или операция), арифметические и логические операции с возможностью изменения приоритета – скобочные формы.
* Ветвление оператор if, включая вариант факультативного else.
* Цикл while.
* Поддержка целочисленного и логического типа данных Строковый тип — + 5 баллов.
* Вывод значений переменных и констант;
* Определение переменных с контролем их не- / пере- определения.
* Многострочные комментарии в стиле Си-подобных языков.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* строки комментариев \*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Рекомендуемый инструментария для реализации проекта –– ANTLR4.

Определение минимального языка, удовлетворяющим базовым требованиям находится в приложении (т. е. ниже по тексту).

Для получения зачета необходимо разработать интерпретатор и компилятор, преобразующий исходный текст программы в *трёхадресный код* (60 баллов); или

разработать интерпретатор и компилятор, преобразующий исходный текст программы в *ассемблер* (75 баллов). Ссылка на README в репозитории эмулятора ассемблера <https://github.com/asurkis/risc-emulator>)

Реализация оператора ветвления if elif else ***вместо*** базового if — + 10 баллов.

Реализация цикла for ***в дополнении*** к базовому while — +10 баллов.

Язык реализации — определяется студентом.

Допускается работа в коллективах по 2 студента.

Приложение.

Пример определения минимального ЯП в терминах ANTLR4.

grammar Minilang;

program: (stmt)\* EOF;

stmt: varDecl

| assignment

| print

| ifstmt

| blockstmt

| whilestmt

;

expr: '(' expr ')' #parenthesisExpr

| left=expr op=(ASTERISK | SLASH) right=expr #mulDivExpr

| left=expr op=(PLUS | MINUS) right=expr #plusMinusExpr

| left=expr compOperator right=expr #compExpr

| ID #idExp

| NUM #numExpr

;

// описания отдельных выражений и утверждений

varDecl: 'var' ID '=' expr ';' ;

assignment: ID '=' expr ';' ;

compOperator: op=(LESS | LESS\_OR\_EQUAL | EQUAL | NOT\_EQUAL | GREATER | GREATER\_OR\_EQUAL) ;

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

blockstmt: '{' (stmt)\* '}' ;

ifstmt: 'if' '(' expr ')' stmt elsestmt? ;

elsestmt: 'else' stmt ;

whilestmt: 'while' '(' expr ')' stmt ;

// список токенов

ID : [a-zA-Z\_] [a-zA-Z\_0-9]\* ;

NUM : [0-9]+ ;

ASTERISK : '\*' ;

SLASH : '/' ;

PLUS : '+' ;

MINUS : '-' ;

ASSIGN : '=' ;

EQUAL : '==' ;

NOT\_EQUAL : '!=' ;

LESS : '<' ;

LESS\_OR\_EQUAL : '<=' ;

GREATER : '>' ;

GREATER\_OR\_EQUAL : '>=' ;

SPACE : [ \r\n\t]+ -> skip;

LINE\_COMMENT : '//' ~[\n\r]\* -> skip;