Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра интеллектуальных информационных технологий

**Отчет по лабораторной работе №1**

**по курсу «ЯПИС»**

на тему: **«**Языковые процессоры интеллектуальных систем и интеллектуализация CASE-технологий»

|  |  |
| --- | --- |
| Выполнил студент группы: | Гончаров Д.Р. |
| Проверила: | Баснина А.П. |

**МИНСК 2022**

1. Вариант

Лабораторная работа 15 варианта

1. Примеры кода (2-3 примера):

Пример 1:

mainauf {  
 Integer a=3  
 Integer b= 12  
 Float c = 9.23  
 (Integer) b  
 (Float) 4  
 while(a<b){  
 left -- a  
 if(a<b){  
 right ++ a  
 left -- b  
 right ++ b  
  
 while(a>b){  
 a= wuf func(b)  
 }  
  
 }else { right ++ b }  
 break  
 }  
 for(Integer i = 0; i<3; right ++ i){  
 left -- b  
 }  
  
 b= wuf func(a)  
}  
  
auf Integer func(Integer v){  
  
 Integer d = (2+4)  
  
 while(d<v){  
 left -- d  
 }  
 return (1+2);  
 }

Пример 2:

mainauf {  
 Integer a=3  
 Integer b= 12  
 (Integer) b  
 while(a<b){  
 left -- a  
 if(a<b){  
 right -- a  
 right -- b  
  
 while(a>b){  
 a= wuf func(b)  
 }  
  
 }else { left ++ b }  
 continue  
 }  
 for(Integer i = 0; i<3; right ++ i){  
 left -- b  
 }  
  
 b= wuf func(a)  
}  
  
auf Integer func(Integer v){  
  
 Integer d = (100-99)  
  
 while(d>v){  
 left -- d  
 }  
 return (5+0);  
 }

1. Описание грамматики

grammar AntlrTest;  
*/\*r : 'j' ID+ ;  
ID : [a-z]+ | [A-Z]+ ;  
WS : [ \t\r\n]+ -> skip ;  
digit : 'digit' NUM+;\*/*PT:'.';  
NUM : [0-9]+; //перечисление возможных цифр  
UNARY\_OPERATOR\_SIDE: 'left'|'right'; //перечисление возможных   
MULTIPLICATION : '\*'; //определение оператора умножения  
DIV: '/'; //определение оператора деления  
PLUS: '+'; //определение оператора сложения  
MINUS: '-'; //определение оператора вычитания  
LOGIC: '&&'|'||'; //определение логических операторов  
COMPARE:'!='|'=='|'<'|'>'|'>='|'<='; //определение операторов сравнения  
UNARY\_OPERATOR: '--'|'++'; //определение унарных операторов  
MAIN\_FUNCTION\_ID:'mainauf'; //определение функции main  
NAME : [a-z]+|[A-Z]+; //определение возможных символов в имен  
TYPE: 'Integer'|'Float'; //определение возможных типов  
dbl : NUM+ PT NUM+;  
  
parameter: ((TYPE NAME)|NUM|NAME)?;  
  
  
expression: //определение всяческих выражений  
 UNARY\_OPERATOR\_SIDE UNARY\_OPERATOR NAME #unaryOperator  
| TYPE (identifier|(NAME '=' expression)) #defineVariable  
| '('expression')' #parens  
| expression operation =(MULTIPLICATION|DIV) expression #mulDiv  
| expression operation =(PLUS|MINUS) expression #plusMinus  
| expression operation =COMPARE expression #compare  
| expression operation = LOGIC expression #logic  
| expression '^' expression #idk  
| '!' expression #not  
| NAME #name  
| (NUM|dbl) #num  
| '('TYPE')' (NUM|dbl|NAME)#typeConverter  
|identifier #id  
|NAME '=' function\_call#varEqlsFunc  
  
;  
  
identifier:NAME '=' NUM|dbl;  
  
WHITESPACE: [ \t\r\n]+ -> skip;  
function: 'auf' TYPE NAME '(' parameter (',' parameter)\* ')' '{' statement\* return\_Rule'}';  
mainFunction: MAIN\_FUNCTION\_ID'{' statement\* '}';  
function\_call: 'wuf' NAME '(' (parameter) (','parameter)\* ')';  
  
//определение правил  
return\_Rule: 'return' ((expression';')|function\_call';' ) ;  
else\_rule:('else' '{'statement+ '}');  
if\_Rule: 'if' '(' expression')' '{'(statement)\* '}' else\_rule?;  
while\_Rule:'while' '(' expression ')' '{'(statement)\* '}';  
for\_Rule:'for' '(' expression';'expression';'expression')' '{'statement\*'}' ;  
break\_Rule:'break';  
continue\_Rule:'continue';  
statement\_rules:(if\_Rule|while\_Rule|for\_Rule|break\_Rule|continue\_Rule|(expression)+|function\_call);  
  
statement:statement\_rules+;  
  
cool:(function+|mainFunction)\* ;

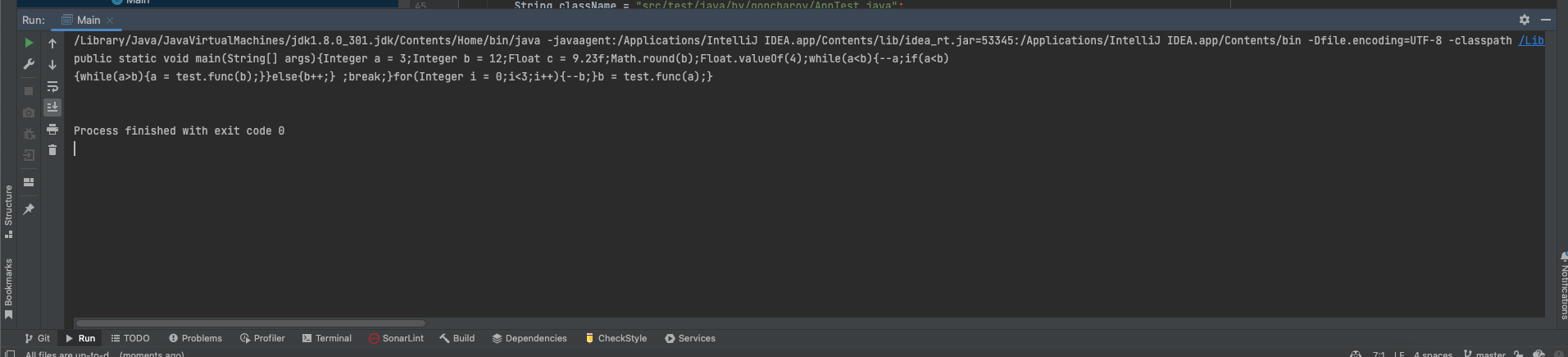
1. Описание дополнительных классов

Кроме классов для определения операторов, выражений и классов связанных с функциями были разработаны три класса: “Converter” (класс был создан для финального конвертирования в язык Java), “MyWalker” (класс был создан для проверки ошибок), “MyVisitor” (класс-посетитель, делает тоже самое что и слушатель, только он именно посещает узлы).

1. Перечень генерируемых ошибок

В моей лабораторной работе отслеживаются только два типа ошибок: синтаксические и ошибка существующего файла.

1. Пример генерации промежуточного кода



1. Пример работы компилятора

