Milestone 2 (Team 24)

1. Language Name:

L0 (Language Zero)

2. Language Design:

2.1. Language Description

Primitive Type:

This programming language includes two primitive types: boolean values and int numeric value.

Operation:

For boolean type data, the corresponding operators include “equal”, “larger than”, “no less than”, “less than”, no larger than”, “not equal”. For int numeric type data, the corresponding operators include “plus”, “minus”, “multiply”, “divide”.

This programming language includes three statements: assignment to associate a value with a variable, if-then-else statement to make decisions, and while statement for iterative execution.

2.2. Compiler Description

Parsing Technique:

The parser uses a bottom-up parsing technique. The parser will match the input with the right-hand sides of the grammar rules first. if the match process succeeds, the parser will replace the right-hand side with the nonterminal on the left side of the grammar rule.

Data Structure:

Lexical analyser: use linked lists to store tokens

Parser: use linked lists to generate a parser tree

2.3. Interpreter

We will design an interpreter analyzing the intermediate codes which follow three-address code. The interpreter will do plenty of calculations for the two operands on the right hand side and store the result on the left hand side operand. We may implement “jump” and other commands to make our interpreter design easier.

3. Grammar

program : statement_list ;

statement_list : statement statement_list | statement ;

statement : declaration | assignment | if_statement | while_statement | print ;

declaration : 'var' ID ';' ;

assignment : ID ':=' low_expression ';' ;

if_statement : 'if' '(' boolean_expression ')' 'correct' '{' statement_list '}' | 'if' '(' boolean_expression ')' 'correct' '{' statement_list '}' 'wrong' '{' statement_list '}' ;

while_statement : 'while' '(' boolean_expression ')' '{' statement_list '}' ;

print : 'print' low_expression ;

boolean_expression : low_expression '==' low_expression | low_expression '>' low_expression | low_expression '>=' low_expression | low_expression '<' low_expression | low_expression '<=' low_expression | low_expression '!=' low_expression | boolean_val ;

boolean_val : 'true' | 'false' ;

low_expression : high_expression '+' low_expression | high_expression '-' low_expression | high_expression ;

high_expression : item '*' high_expression | item '/' high_expression | item ;

item : ID | NUMBER ;

ID : [a-z|A-Z]+ ;

NUMBER : [0-9]+ ;

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

4. Development Language

We plan to use C or Java to implement the compiler. We will specify the language we want to use and more details in the future.