CODE-X

LANGUAGES AND COMPILERS

Project Report

Presented To:

- Dr. Mona
- Eng. Andrew

BY:

- Karim Yasser Ahmed
  - ID: 1152092
- Khaled Sameh Mohamed
  - ID: 1153073
- Mina Ashraf Louis
  - ID: 1152055
- Amr Khaled Zaky
  - ID: 1152003

PROJECT DESCRIPTION:

This project is a design and implementation of a C like programming language using Lex and Bison. The Language handles the declarations, mathematical and logical expressions, assignments, loops, switch statements and block structures. There is also an implemented simple GUI for out IDE.

TOOLS USED:

1. Flex (Lex) for lexical analysis
2. Bison (Yacc) parser generator
3. C & C# Language
4. Visual Studio Code
5. Calculator tutorial
6. Unity (C#) implementing the GUI

TOKENS:

- **Data Types Values:**

  CHAR_VALUE, STRING_VALUE, INT_VALUE, FLOAT_VALUE

- **Data Types:**

  INT CHAR FLOAT STRING BOOLEAN DOUBLE

- **Logical Operators**:

  NOT, AND_AND, OR_OR, EQUAL_EQUAL, NOT_EQUAL, GREATER_THAN, GREATER_THAN_EQUAL, SMALLER_THAN, SMALLER_THAN_EQUAL

- **Mathematical Operators**:

  PLUS_EQUAL, MINUS_EQUAL, MULTIPLY_EQUAL, DIVIDE_EQUAL, PLUS_PLUS, MINUS_MINUS, PLUS, MINUS, MULTIPLY, DIVIDE, REMAINDER

- **Language Words**:

  IDENTIFIER, VOID, MAIN, SEMI_COLON, TWO_DOTS, EQUAL, IF, DO, WHILE, FOR, SWITCH, ELSE, BREAK, TRUE, FALSE, CASE, DEFAULT

- **Braces:**

  OPENED_BRACKET, CLOSED_BRACKET, OPENED_BRACE, CLOSED_BRACE

GRAMMAR:

Main Structure

program : main

;

main : VOID MAIN OPENED_BRACKET CLOSED_BRACKET OPENED_BRACE body CLOSED_BRACE

;

body : whilestmt body

| ifstmt body

| dowhilestmt body

| forstmt body

| switchstmt body

| declaration body

| assignment body

| mathassignment SEMI_COLON body

| doublesign SEMI_COLON body

| COMMENT body

|

;

Identifier Declaration

declaration : datatype IDENTIFIER declarationstmt SEMI_COLON

;

declarationstmt : EQUAL mathexpr SEMI_COLON

|

;

Assignment

```
assignment      :       IDENTIFIER EQUAL mathexpr   SEMI_COLON
                        ;
```

Expressions (Logical & Mathematical)

```
logicalexpr :
                        mathexpr
                        | logicalexpr OR_OR logicalexpr
                        | logicalexpr AND_AND logicalexpr
                        | logicalexpr NOT_EQUAL logicalexpr
                        | logicalexpr EQUAL_EQUAL logicalexpr
                        | logicalexpr GREATER_THAN logicalexpr
                        | logicalexpr GREATER_THAN_EQUAL logicalexpr
                        | logicalexpr SMALLER_THAN logicalexpr
                        | logicalexpr SMALLER_THAN_EQUAL logicalexpr
                        |       OPENED_BRACKET logicalexpr CLOSED_BRACKET
                        ;
mathexpr :
                                IDENTIFIER
                                | val
                                | mathexpr  MINUS  mathexpr
                                | mathexpr PLUS mathexpr
                                | mathexpr DIVIDE  mathexpr
                                | mathexpr MULTIPLY mathexpr
                                | mathexpr REMAINDER mathexpr
                                | OPENED_BRACKET mathexpr CLOSED_BRACKET
                                | doublesign
                        ;
```

doublesign : PLUS_PLUS IDENTIFIER

| MINUS_MINUS IDENTIFIER

| IDENTIFIER PLUS_PLUS

| IDENTIFIER MINUS_MINUS

;

Mathematical Assignment

mathassignment :

IDENTIFIER MINUS_EQUAL mathexpr

| IDENTIFIER PLUS_EQUAL mathexpr

| IDENTIFIER DIVIDE_EQUAL mathexpr

| IDENTIFIER MULTIPLY_EQUAL mathexpr

;

IF Conditions

ifstmt: IF OPENED_BRACKET logicalexpr CLOSED_BRACKET OPENED_BRACE body CLOSED_BRACE elsestmt

;

elsestmt: ELSE OPENED_BRACE body CLOSED_BRACE

|

;

While Loop

whilestmt : WHILE OPENED_BRACKET logicalexpr CLOSED_BRACKET OPENED_BRACE body CLOSED_BRACE

;

For Loop

forstmt : FOR OPENED_BRACKET a b mathexpr CLOSED_BRACKET OPENED_BRACE body CLOSED_BRACE

;

a : declaration

| assignment

;

b : logicalexpr SEMI_COLON

;

Do While Statement

dowhilestmt : DO OPENED_BRACE body CLOSED_BRACE WHILE OPENED_BRACKET logicalexpr CLOSED_BRACKET SEMI_COLON

;

Switch Case Statement

switchstmt : SWITCH OPENED_BRACKET IDENTIFIER CLOSED_BRACKET OPENED_BRACE casestmt CLOSED_BRACE

;

casestmt : CASE val TWO_DOTS body BREAK SEMI_COLON casestmt

| defaultstmt

;

defaultstmt : DEFAULT TWO_DOTS body BREAK SEMI_COLON

;

Terminals Handling

```
val :   STRING_VALUE
      | CHAR_VALUE
      | TRUE
      | FALSE
      | INT_VALUE
      | FLOAT_VALUE
      ;
```

Data Type

```
datatype :   INT
           | FLOAT
           | STRING
           | CHAR
           | BOOLEAN
           | DOUBLE
           ;
```

QUADRUPLES:

Quadruple	
LD R1, X	Load X in R1
ADD R1, R2	R1 = R1 + R2
SUB R1, R2	R1 = R1 - R2
MUL R1, R2	R1 = R1 * R2
DIV R1, R2	R1 = R1 / R2
REM R1, R2	R1 = R1 % R2
STD R1, X **INC R1**	X++
STD R1, X **DEC R1**	X--
INC R1 **STD R1, X**	++X
DEC R1 **STD R1, X**	--X
MOV R1, constant **STD R1, X**	X = constant
ADD R1, R2 **ST R1, X**	R1 += R2
SUB R1, R2 **ST R1, X**	R1 -= R2
MUL R1, R2 **ST R1, X**	R1 *= R2
DIV R1, R2 **STD R1, X**	R1 /= R2
CMPG R1, R2	1 => R1>R2 0 => else
CMPGE R1, R2	1 => R1>=R2 0 => else

CMPL R1, R2	1 => R1<R2 0 => else
CMPLE R1, R2	1 => R1>=R2 0 => else
CMPE R1, R2	1 => R1=R2 0 => else
CMPNE R1, R2	1 => R1!=R2 0 => else
JMP L0	Jump to L0
JZ L0	If comparison = 0 => Jump to L0
JNZ L0	If comparison = 1 => Jump to L0