 Cairo University

Faculty of Engineering

Computer Engineering Department

Fourth Year

**Compilers Project Document**

**Names: Sec: B.N:**

Amal AbdelNafei 1 6

Engy Samy 1 7

Rania Alaa 1

Mennah Rabie 2 17

**• Project Overview:**

**• Tools and Technologies used:**

In our project, we used Flex and Bison as lexer and parser, using c++.

**• A list of tokens and a description of each.**

|  |  |
| --- | --- |
| **Token** | **Description** |
| const | Keyword to be used when declaring a constant. |
| var | Keyword to be used when declaring a variable. |
| true | Boolean value. |
| false | Boolean value. |
| = | Assignment operator. |
| || | Comparing OR. |
| && | Comparing AND. |
| ! | Comparing NOT. |
| << | Bitwise shift left. |
| >> | Bitwise shift right. |
| | | Bitwise or. |
| & | Bitwise and. |
| ~ | Bitwise not. |
| >= | Greater than or equal. |
| > | Greater than. |
| <= | Smaller than or equal. |
| < | Smaller than. |
| != | If not equal. |
| == | If equal. |
| $$ | Starting a comment. |
| + | Addition. |
| - | Subtraction. |
| \* | Multiplication. |
| / | Division. |
| % | Mod. |
| \*\* | Power. |
| ^ | Xor. |
| If , else | Keywords to be used with if –else statements. |
| While | Keyword to be used with while loops. |
| for | Keyword to be used with for loops. |
| repeat , until | Keywords to be used with repeat-until loops. |
| switch, case, default, done | Keywords to be used with switch cases. |
| i\_[any alphanumeric characters] e.g. : (i\_x1) | Identifier for integer variables. |
| f\_[any alphanumeric characters] e.g. : (f\_x1) | Identifier for float variables. |
| b\_[any alphanumeric characters] e.g. : (b\_x1) | Identifier for Boolean variables. |
| s\_[any alphanumeric characters] e.g. : (s\_x1) | Identifier for string variables. |

**• A list of the language production rules.**

* stmt: stmt variable | stmt constant\_stmt | stmt assignment | stmt if\_else\_if\_else\_stmt | stmt for\_loop | stmt while\_loop | stmt repeat\_until\_loop | stmt switch\_case | variable | constant\_stmt | assignment | if\_else\_if\_else\_stmt | for\_loop | while\_loop | repeat\_until\_loop | switch\_case
* variable: VAR id1 ';' endls | VAR decl\_assign endls | VAR error ';'
* id1: INT\_ID | FLOAT\_ID | BOOL\_ID
* decl\_assign: ID EQUAL expr
* constant\_stmt: CONST constant endls
* constant: ID EQUAL value
* assignment: ID EQUAL expr ';' endls
* expr : str\_expr | bool\_expr | int\_expr | float\_expr
* int\_expr:   
  int\_expr '+' int\_expr | int\_expr '-' int\_expr | int\_expr '\*' int\_expr|int\_expr '/' int\_expr | int\_expr '%' int\_expr |int\_expr '&' int\_expr | int\_expr '|' int\_expr | int\_expr '^' int\_expr |'~' int\_expr| int\_expr BITWISE\_SHIFT\_LEFT int\_expr | int\_expr BITWISE\_SHIFT\_RIGHT int\_expr | '(' int\_expr ')' | INT |INT\_ID
* float\_expr:

float\_expr '+' float\_expr | float\_expr '-' float\_expr | float\_expr '\*' float\_expr| float\_expr '/' float\_expr | float\_expr '%' float\_expr | float\_expr POW float\_expr|

float\_expr '+' int\_expr| float\_expr '-' int\_expr | float\_expr '\*' int\_expr | float\_expr '/' int\_expr | float\_expr '%' int\_expr | float\_expr POW int\_expr |

int\_expr '+' float\_expr | int\_expr '-' float\_expr | int\_expr '\*' float\_expr | int\_expr '/' float\_expr| int\_expr '%' float\_expr | int\_expr POW float\_expr | '(' float\_expr ')' |FLOAT | FLOAT\_ID

* str\_expr: STRING |STR\_ID
* if\_else\_if\_else\_stmt: if\_stmt1 else\_if\_stmt
* if\_stmt1: IF start\_while\_if bool\_expr end\_while\_if '{' stmt '}'
* if\_stmt: IF start\_while\_if bool\_expr end\_while\_if '{' stmt '}'
* else\_if\_stmt: ELSE if\_stmt else\_if\_stmt | ELSE '{' stmt '}' | epsilon
* while\_loop: WHILE ‘(‘ bool\_expr ‘)’ '{' endls stmt '}' ';' endls
* for\_loop: FOR '(' for\_assignment ',' bool\_expr ')' '{' endls stmt '}' '(' for\_assignment ')' ';' endls
* repeat\_until\_loop: REPEAT '{' endls stmt '}' UNTIL '(' bool\_expr ')' ';' endls
* bool\_expr: LOGIC\_NOT bool\_expr| bool\_expr LOGIC\_AND bool\_expr | bool\_expr LOGIC\_OR bool\_expr | boolean | BOOL\_ID | bool\_term | '(' bool\_expr ')'
* bool\_term: compare\_opd EQ compare\_opd | compare\_opd NOT\_EQ compare\_opd | compare\_opd GR compare\_opd | compare\_opd GR\_EQ compare\_opd | compare\_opd SM compare\_opd | compare\_opd SM\_EQ compare\_opd
* compare\_opd: INT | INT\_| FLOAT| FLOAT\_ID | STRING | STR\_ID
* switch\_case: SWITCH '(' id ')' '{' case\_stmts defaultt '}'
* case\_stmts: case\_stmt case\_stmts | case\_stmt
* case\_stmt: CASE value ':' stmt DONE ‘;’
* defaultt: DEFAULT ':' stmt

**• A list of the quadruples and a short description of each**

|  |  |
| --- | --- |
| **Quadruple** | **Description** |
| STO, i\_x, , i\_y, | i\_y = i\_x |
| ADD,i\_x,i\_y,i\_z, | i\_z=i\_x+i\_y |
| SUB,i\_x,i\_y,i\_z, | i\_z=i\_x-i\_y |
| MUL, i\_x,i\_y,i\_z, | i\_z=i\_x\*i\_y |
| DIV, i\_x,i\_y,i\_z, | i\_z=i\_x/i\_y |
| AND, i\_x,i\_y,i\_z, | i\_z=i\_x&i\_y |
| OR, i\_x,i\_y,i\_z, | i\_z=i\_x|i\_y |
| XOR, i\_x,i\_y,i\_z, | i\_z=i\_x^i\_y |
| NOT, i\_x, ,i\_z, | i\_z=~i\_x |
| SHFTL, i\_x,i\_y,i\_z, | i\_z=i\_x<<i\_y |
| SHFTR, i\_x,i\_y,i\_z, | i\_z=i\_>>i\_y |
| EQ,b\_x,b\_y,b\_z | b\_z= b\_x==b\_y |
| NOT\_EQ,b\_x,b\_y,b\_z | b\_z= b\_x!=b\_y |
| GR,b\_x,b\_y,b\_z | b\_z= b\_x>b\_y |
| GR\_EQ,b\_x,b\_y,b\_z | b\_z= b\_x>=b\_y |
| SM,b\_x,b\_y,b\_z | b\_z= b\_x<b\_y |
| SM\_EQ,b\_x,b\_y,b\_z | b\_z= b\_x<=b\_y |
| JMP, L1, , | Unconditional jump to label L1 |
| JFALSE, L1, t1, | If t1=false then jump to L1 |
| JTRUE, L1, t1, | If t1=true then jump to L1 |

* **Semantic checks:**

1. Multiple declaration of the same variable.
2. Variables used before being declared.
3. Variables used before being initialized.
4. View unused variables.
5. Type checking.
6. Constant not to be assigned again.