Project Overview

This project is a simple programming language implemented using Lex and Yacc compiler generating packages. The structure of a source code should be as follows:

program:

//Constant and Variable declarations

begin:

//Program statements

end

Implemented parts:

Lexical Analysis (Accepted Tokens)  
Parser (Grammar rules)  
Syntax Analysis  
Semantic Analysis  
Quadruples generation

Tools and Technologies used

Lex Package: FLEX 2.6.0  
Yacc Package: GNU Bison 3.0.4

Tokens

|  |  |
| --- | --- |
| Token | Description |
| PROGRAM | “program” indicates the beginning of declaration statements. |
| S | “begin” which indicates the beginning of the program statements. |
| END | “end” indicates the end of the program |
| IDENTIFIER | Names of variables and constants.  Contains letters, numbers or underscores.  Can’t start with number. |
| CONST | To define a constant. |
| INT | Int type. |
| FLOAT | Float type. |
| BOOL | Bool type. |
| INUM | Integer number |
| FNUM | Float number |
| TRUE | Bool true value |
| FALSE | Bool false value |
| IF | Beginning of if statement |
| THEN | Used in if statement after the condition |
| ELSE | Used in if statement as alternative path |
| ENDIF | Indicates the end of if statement |
| WHILE | Beginning of while statement |
| DO | Beginning of do while statement |
| SWITCH | Beginning of switch statement |
| CASE | Case in switch statement |
| DEFAULT | Default in switch statement |
| CONTINUE | Continue used in loops |
| BREAK | Break used in loops |
| AND | Logical and |
| OR | Logical or |
| NOT | Logical not |
| PLUS | + |
| MINUS | - |
| MUL | \* |
| DIV | / |
| LT | < |
| GT | > |
| EQ | == |
| LTE | <= |
| GTE | >= |
| NE | != |
| COLON | : |
| SEMICOLON | ; |
| ( |  |
| ) |  |
| { |  |
| } |  |
| = |  |

Language Production Rules

program: PROGRAM declarations statements END

declarations: declarations declaration SEMICOLON | declarations err\_stmt

declaration: const\_dec | var\_dec

const\_dec: CONST type IDENTIFIER = inum

| CONST type IDENTIFIER = fnum

| CONST type IDENTIFIER = bval

Var\_dec: type IDENTIFIER

err\_stmt: error SEMICOLON| error )

type: INT|FLOAT|BOOL

bval: TRUE|FALSE

statements: statements statement SEMICOLON | statements err\_stmt

statement: assignment

| if\_stmt

| while\_stmt

|do\_while\_stmt

| for\_stmt

| switch\_stmt

| BREAK

| CONTINUE

Assignment: IDENTIFIER = expr | IDENTIFIER = bval

Fnum: FNUM | MINUS FNUM

Inum: INUM | MINUS INUM

Number: fnum|inum

If\_: IF (expr) then statements

If\_else: ENDIF| ELSE statements ENDIF

If\_stmt: if\_ if\_else

While\_stmt: WHILE (expr){statements}

Do\_while\_stmt: DO{statements}WHILE(expr);

S\_stmt: cases default | cases

Case: CASE inum COLON statements

Cases: | cases case

Switch\_stmt: SWITCH (IDENTIFIER){s\_stmt}

For\_stmt: FOR(IDENTIFIER=expr COLON expr COLON number)

Expr: fnum

|inum

|IDENTIFIER

|expr PLUS expr

| expr MINUS expr

| expr MUL expr

| expr DIV expr

| expr GT expr

| expr GTE expr

| expr LT expr

| expr LTE expr

| expr NE expr

| expr EQ expr

| expr EQ TRUE

| expr EQ FALSE

| expr NE TRUE

| expr NE FALSE

| expr AND expr

| expr OR expr

| NOT expr

| (expr)

Quadruples

|  |  |
| --- | --- |
| Quadruple | Description |
| Mov t1,t2 | t2=t1 |
| Add t1,t2,t3 | t3=t1+t2 |
| Sub t1,t2,t3 | t3=t1-t2 |
| Mul t1,t2,t3 | t3=t1\*t2 |
| Div t1,t2,t3 | t3=t1/t2 |
| And t1,t2,t3 | t3=t1 and t2 |
| Or t1,t2,t3 | t3=t1 or t2 |
| Not t1,t2 | t2= not t1 |
| GT t1,t2,t3 | t3= ( t1>t2 ) |
| LT t1,t2,t3 | t3= ( t1<t2 ) |
| EQ t1,t2,t3 | t3= ( t1==t2 ) |
| GE t1,t2,t3 | t3= ( t1>=t2 ) |
| LE t1,t2,t3 | t3= ( t1<=t2 ) |
| NE t1,t2,t3 | t3= ( t1!=t2 ) |
| goto L0 | Unconditional jump to label L0 |
| If not t1 goto L0 | Conditional jump to label L0  ( jump to L0 if t1 == false) |
| If t1 goto L0 | Conditional jump to label L0  ( jump to L0 if t1 == true) |
| L0: | Beginning of label L0 |