Technical Documentation:

* Language is case sensitive (BEGIN != begin != Begin)
* Keywords are reserved (may not appear as identifier names).
* Types are integer, string, and Logical.

Integers are strings of digits,start with 1-9 and followed by 0 or more 0-9.

The only integer that begins with 0 is the integer 0.

Strings are composed of ASCII characters (no special characters like \n).

* List of keywords (written in upper case)

BEGIN END COMMENT

IF THEN REM

WHILE DO DIV

READ WRITE WRITELN

INTEGER STRING LOGICAL

TRUE FALSE

OR AND

* Lexemes in ALGOL-W: (# below is used to denote zero or more occurrences)

identifier::=letter #(letter | digit | underscore)

letter: "a" | "A" .... "z" | "Z"

digit::="0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9",

punctuator::= "(" | ")" | ";" | “:=” | "."

infix:= "\*" | "+" | “-“ | “/” | "<" | ">" | "=" | "!=" |AND|OR|DIV|REM

unary operators ∷= !

non\_zero\_digit::="1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

Boolean values ::= TRUE | FALSE

Token numbers:

1 identifier

2 any literal (integer such as 123,string "abc", FALSE, TRUE)

3 types (keywords 'INTEGER' 'STRING' 'LOGICAL')

4 addition operators (+ - OR)

5 multiplication operators (\* / DIV REM AND)

6 relational operators = , !=, <, >

7 BEGIN 8 END

9 IF 10 THEN

11 WHILE 12 DO

3 INTEGER 3 STRING 3 LOGICAL

2 TRUE 2 FALSE 5 DIV 5 REM

4 OR 5 AND

13 READ 13 WRITE 13 WRITELN (IO\_operations)

14 ( 15 ) 16 ; (not part of a comment)

17 ! Boolean not 18 . (only at the end of the program)

BNF For ALGOL-W:

**BNF with 17 nonterminals**

1.program : blockst ‘.’

2.blockst : BEGINTOK stats ENDTOK

3.stats : statmt ';' stats | <empty>

4.decl : BASICTYPETOK IDTOK

5.statmt : decl | ifstat | assstat | blockst | loopst | iostat | <empty>

6.assstat : idref ASTOK expression

7.ifstat : IFTOK expression THENTOK statmt

8.loopst : WHILETOK expression DOTOK stat

9.iostat : READTOK ( idref ) | WRITETOK (expression)

10.expression : term expprime

11.expprime : ADDOPTOK term expprime | <empty>

12.term : relfactor termprime

13.termprime : MULOPTOK relfactor termprime | <empty>

14.relfactor : factor factorprime

15.factorprime : RELOPTOK factor | <empty>

16.factor : NOTTOK factor | idref | LITTOK | '(' expression ')'

17. idref : IDTOK

SOME **SEMANTICS**

OR was combined with ADDOPs, AND combined with MULOPS to reduce the levels of precedence and shorten the grammar. This deviates from true Algol.

There is no unary – or +.

**Strings:**

strings are not allowed in any assignment statements, and there are no operations on strings.

The only place a string expression can be used is in a WRITE or WRITELN.

Strings are not allowed in assignment statements.

**Assignment compatible:**

Two types are assignment compatible if and only if they are the same type.

You cannot assign LOGICAL to INTEGER or vice versa.

**Expressions:**

The operators AND, OR, and NOT(!) operate only on type LOGICAL.

The operators DIV, REM operate only on INTEGER.

The operators +, -, \* can be used on INTEGER and REAL

The operator / can be used on INTEGER or REAL but always returns a REAL result (8/2=4.0)

AND, OR are evaluated by short-circuit evaluation

List of Known Bugs:

Scanner:

* COMMENT is not followed by ‘;’.
* ‘:’ is not followed by ‘=’.
* String length is more than 256 character.
* String is not end by ‘”’.

Parser:

* Error in 'blockst' routine. Program Doesn't start with BEGIN Token.
* Error in 'blockst' routine. Missing END Token.
* Error in 'stats' routine. Missing ';' at the end of statement.
* Error in 'decl' routine. TYPE Token doesn't followed by IDENTIFIER.
* Error in 'ifstat' routine. IF is not followed by THEN Token.
* Error in 'loopst' routine. WHILE is not followed by DO Token.
* Error in 'iostat' routine of READTOK. '(' Token is missing.
* Error in 'iostat' routineof READTOK. ')' Token is missing.
* Error in 'factor' routine of WRITETOK. '(' Token is missing.
* Error in 'factor' routine of WRITETOK. ')' Token is missing.
* Error in 'idref' routine. Not IDENTIFIER Token.

Symbol Table:

* Error: Declared variable before used it.
* Error: Variable is already declared in the scope.

Future Suggestions:

* IF Statement can include more statements in THEN part.
* ELSE part of IF Statement can be more useful in ALGOL-W.
* STRING can do assignment and perform concatenation operation with +.
* Variable declaration of same data type can be done on single line and separated by commas.
* Variable declaration and assignment statement can be done in single line.
* If FOR Loop can be in ALGOL-W, then it is more efficient way to work on it.
* In WRITELN and WRITE, String literal and print variable value can be done in one statement.
* Can introduce some mostly used escape sequence like ‘\n’, ‘\t’ etc.

Unimplemented Instructions:

* No MIPS code generation for read string value from user.
* No MIPS code generation to handle LOGICAL datatype.
* No Error checking and bug identification in code generation part.