**User Documentation**

**Description of ALGOL-W Language:**

ALGOL-W is like Algol programming language with few modifications.

**Structure:**

ALGOL-W program start with BEGIN and end with END and ‘.’ which is end of ALGOL-W program marker.

Syntax like,

BEGIN

ALGOL-W code

END.

If ‘.’ appear anywhere inside code, program terminated.

We can’t use ‘.’ anywhere inside code except as string literals and in comment.

For define scope inside ALGOL-W, new scope start with BEGIN and end with END; .

Syntax like,

BEGIN

BEGIN

Scope 1

END;

ALGOL-W code

BEGIN

Scope 2

BEGIN

Scope 3

END;

END;

END.

**DataTypes:**

1. STRING datatype can can be used as comparison statement and in WRITE and WRITELN statement as String literal.
2. LOGICAL datatype is used for condition return value check.

It has TRUE and FALSE value which is consider as string literal and in MIPS it’s value is 1 for TRUE and 0 for FALSE.

It is used as error checking in IF and WHILE statement whether condition return LOGICAL value or not.

1. INTEGER datatype is used to store integer value.

We can assigned integer value, perform computation like arithmetic and logical or relational operations.

We can take user input of INTEGER type and print its value.

**Operators:**

We can perform arithmetic operation, relational operation and some Logical operation only on INTEGER data type.

To do arithmetic operation,we have +, -, \*, /, Remainder, Divide operators are available.

To do relational operation, we have <, >, =, != operators which only hold LOGICAL value.

To do logical operation, we have AND, OR operator.

Also unary operator ! Is available.

Syntax like,

variable1 operator variable2

expression operator expression

like

a + b, a> b, A AND b

N := N + 1;

SUM := SUM + (a \* b);

Unary operator like, ! Variable1 or ! (expression)

like ! a

Operator Precedence:

( ) has highest priority.

! first level

\*, /, AND second level

+, -, OR, <, > , = ,!= third level

**Conditional Statement:**

For Conditional Statement, IF..THEN.. Statement is included in ALGOL-W.

Only IF Part is present and only one statement can be consider as THEN part.

Syntax is like

IF Condition THEN Statement

Example:

IF PRICE > MAX

THEN MAX := PRICE;

**Loop Statement:**

For Branching Statement, WHILE Loop Statement is in ALGOL-W.

Block code is start with BEGIN and end with END;

Syntax is like

WHILE Condition Do

BEGIN

List of statements separated by ;

END;

Example:

WHILE PRICE > 0 DO

BEGIN

N := N + 1;

SUM := SUM + PRICE;

READON(PRICE); COMMENT readon not included in tiny Algol-W;

END;

we can nested if statement inside while loop like,

WHILE PRICE > 0 DO

BEGIN

N := N + 1;

SUM := SUM + PRICE;

IF PRICE > MAX

THEN MAX := PRICE;

READON(PRICE); COMMENT readon not included in tiny Algol-W;

END;

**Literals:**

String quoted with ”...” is consider as literal.

String literals must be single line and no more than 256 characters.

We can use “ inside string literal but it must be precede by another “.

Like,

“...””….”

Integer constant value in assignment statement, expression, condition statement is consider as literals.

Also, LOGICAL value TRUE, FALSE is consider as literals.

For example, integer such as 123,string "abc", FALSE, TRUE

**IO Statements:**

READ, WRITE and WRITELN is work as input/output statement in program.

READ can take integer or string datatype variable as its argument to take user input.

READ can not take literals as its argument

WRITE and WRITELN is print its argument.

Integer variable, literals as argument for WRITE and WRITELN.

Difference between WRITE and WRITELN is WRITELN statement print newline at end of its execution and WRITE statement don’t.

It is single line statement.

Syntax Like,

1. READ(argument);

argument is only one integer or string variable

INTEGER A;

READ(A);

1. WRITE(argument);

WRITELN(argument);

argument is only one integer or string variable or one literal.

INTEGER A;

A := 10;

WRITE(“Value of A = “);

WRITELN(A);

**Variable Declaration:**

Datatype followed by variable name which is case sensitive and ‘;’.

Only one variable declared in statement.

Syntax like,

INTEGER A;

STRING s;

LOGICAL l;

INTEGER A, b; is illegal syntax

**Assignment Statement:**

Assignment statement start with variable name followed by assignment token ‘:=’ which is followed by literals, expression or variable.

Syntax like,

A := 10;

A := “String literal”;

A := a + 10;

A := a AND b;

Statement end with ‘;’.

we can assign value to integer and logical datatype.

But variable must be declared before it is assign or used.

**Comment Statement:**

We can write comment anywhere in code.

Comment is remove in scanner phase.

Comment is start with COMMENT and end with ‘;’.

Comment is single line comment.

Syntax like,

COMMENT comment ;

where comment is any characters,digits, punctuations(except ‘;’)

**How To use Compiler:**

There are 5 main class file:

1. BbAlgolParser
2. BbAlgolScanner
3. BbAlgolSymbolTable
4. FileHandler
5. AssemblyCodeGen

Give your own code text file as input in BbAlgolScanner class.

Compile and Run BbAlgolParser file in which public static void main is reside.

It produce list of MIPS assembly code.

Use any MIPS simulator to show output.

Here,

BbAlgolScanner take text file as user input and convert it into list of tokens with its name and token number.

It remove comments and whitespaces.

It also show error message like comment don’t end with ;, : don’t followed by =, string is more than 256 character, string don’t end with “ at end of line etc. errors based on DFA.

BbAlgolScanner is implemented based on DFA-Deterministic Finite Automata.

BbAlgolSymbolTable store variable as scope vise with its identifier, scope number, datatype and offset value.

FileHandler take input text file name from BbAlgolScanner and and convert it into character array.

In which CharacterReader class method getChar() give next character every time it called.

AssemblyCodeGen include code for list of MIPS instruction related to particular ALGOL-W code instructions.

BbAlgolParser use getToken() method of BbAlgolScanner and generate parse tree and related MIPS instructions.

It is implemented based on BNF-Backus Normal Form and MIPS instruction is implemented based on SDT- Syntax Directed tree