## Introduction

The programming language is similar to that of C but does not support complex operations that can be done in C.The manual contains the syntax and semantics and semantic checks of the programming language.

**Data Types**

i)int,
ii)unsignedint
iii)char
iv)bool
v)1D and 2D arrays.

## Lexical Considerations

keywords are case sensitive.The keywords and id must be separated by space.

## MACRO SYNTAX

Program : DeclarationBlock  EOF

DeclarationBlock: DeclarationBlock Declaration | Declaration

Declaration : VariableDecl | FunctionDecl

VariableDecl : Type VarDeclList

VarDeclList : VarDeclList’,’VarDeclSingle | VarDeclSingle

VarDeclSingle :  { ID | ID’[‘Expr’]’ | ID’[‘Expr’]’’[‘Expr’]’ | ID’=’Expr}

FunctionDecl : {Type | ‘void’} ID ‘({‘ParamList | ε}’)’ Block

ParamList : ParamList ‘,’ ParamSingle | ParamSingle

ParamSingle : Type  {ID | ID’[‘’]’ | ID’[‘’]’[‘’]’}

Block : ‘{‘Statement$^{*}$’}’

Statement : Locationid ‘=’ Expr

```
                    | FunctionCall
                    | IfElseBlock
                    | ForBlock
                    | WhileBlock
                    | ReturnStmt
                    | ‘break’
                    | ‘continue’
                    |VariableDecl
ifElseBlock : ‘if’ ( Expr ) Block | ‘if’ (  ) Block ‘else’ Block

FunctionCall : ID ( InputList | ε)

Type -> int | unsignedint | bool | char | float

InputList : InputList , Expr | Expr

ForBlock : ‘for’ ( Expr ; Expr ;Expr) Block

WhileBlock : ‘while’ (Expr) Block

ReturnStmt : ‘return’ {Expr | ε}



Expr : Expr BinaryOp Expr
          | UnaryOp Expr
          | (Expr)
          | Constant
          | FunctionCall
          | LocationId
          | Expr ‘?’ Statement ‘:’ Statement

Constant : INTLITERAL | FLOATLITERAL | CHARLITERAL | BoolConstant | null

BoolConstant -> true | false

BinaryOp : ArthOps | RelOps | Boolops

UnaryOp : ‘-’ | ‘!’

ArthOps : ‘/’  |  ‘*’  |  ‘+’  |  ‘-’  | ’%’ |

RelOps : ‘==’ | ‘!=’ |  ‘>=’  |  ‘<=’ | ‘<’ | ‘>’

BoolOps : ‘and’ | ‘or’ | ‘xor’
```

## MICRO SYNTAX

INTLITERAL : [-]?[0-9][0-9]*
FLOATLITERAL : [-]?[0-9][0-9]*[.][0-9][0-9]*
CHARLITERAL : ['][a-zA-Z0-9_][a-zA-Z0-9_]*[']
ID : [a-zA-Z][a-zA-Z0-9_]*
COMMENT : '//' ~[\r\n]*->skip
NS : [ \t\n]+ -> skip

## Semantics

### Control Statements

#### IfElseBlock

First, the Expr is evaluated. If the result is true, the block immediately after the if condition is executed. Otherwise, the Expr in else is executed, if it exists. If the result if false the block after the else is executed if it exists, otherwise there will be no execution.

#### WhileBlock

The Expr is evaluated.If the result is true, then the following block is executed repeatedly until the Expr becomes false. Once it becomes false execution goes on to the next step

#### ReturnStmt

Returns the expr next to return from the function to the caller

#### break

Breaks while or for loop whose scope it lies in.

#### TernaryStmt

If the Expr evaluates to true then the statement immediately after ? executes else the statement after : executes.

#### ForBlock

The loop index variable declares an integer variable whose scope is limited to the body of the loop Here first expression is the initialisation The second expression is the variable change after each iteration of the Block The third expression is the condition which should be true for the block to be executed, so loop ends as soon as this expr computes to false.

### Scope rules

1)The global scope consists of names of fields and methods introduced in the Program class declaration.

2) The method scope consists of names of variables and formal parameters introduced in a method declaration. Additional local scopes exist within each block in the code; these can come after if or for statements, or inserted anywhere a statement is legal.

3)An identifier introduced in a method scope can shadow an identifier from the global scope.

**Functions**

(i) passing argument values from the caller to the callee
(ii) executing the body of the callee
(iii) returning to the caller along with a result if return type is not void

A method that returns a result may be called as part of an expression, in which case the result of the call is the result of evaluating the expression in the return statement when this statement is reached

**Location**

There are three types of locations.
1)variable
2) 1-D array element.Array element can be accessed by ID[i] where i is the index
3)2-D array element.Array element can be accessed by ID[i][j] where i,j are the indices

## Semantic checks

1) There cannot be more than one variable with the same name in the same scope.

2) An identifier cannot be used before it is declared.

3) The Expr in the size in an array declaration must be integer and greater than 0 and indices used during access must be valid indices.

4) The number and types of arguments in a method call must be the same as the number and types of the method parameters.

5) The expression in a return statement must have the same type as the declared result type of the method definition and there should be no expr with return if it is void.

6) Variable names should be different from keywords

7) The Expr in an if statement must evaluate to type boolean.

8) The variables on both sides of = must have the same type.

10) All break statements must be  within a for or while loop

## Evaluation of operators

Precedence b/w arithmetic operators is taken care of and arithmetic operators have higher precedence than relational operators.