MileStone1 Documentation (Team 8)

Shubhangi Gupta : 1219474243
Arjun Borkhatariya : 1220063169
Paresh Pandit : 1220832314
Nitish Tripathi : 1219500269

Name of the Language : Impetus

Context Free Grammar (BNF)

<Program> ::= <Block>.
<Block> ::= { <Declarations> ; <Commands>; <Print> }

<Declarations> ::= <Declarations> ; <Declarations>
<Declarations> ::= <Datatype> <Identifier> = <Number>
<Declarations> ::= <Datatype> <Identifier> = “ <String> ”
<Declarations> ::= <Datatype> <Identifier> = <Identifier>
<Declarations> ::= const <Identifier> = <Number>
<Declarations> ::= ∈

<Commands> ::= <Commands> ; <Commands>
<Commands> ::= if <Boolean> <Block> else <Block>
<Commands> ::= for (<Expression> ; <Expression> ; <Expression>) <Block>
<Commands> ::= for <Identifier> in range (<Number> , <Number>) <Block>
<Commands> ::= while(<Boolean>) <Block>
<Commands> ::= <Identifier> = <Expression>
<Commands> ::= ∈

<Print> ::= print(<Identifier>) ; print(<Identifier>)
<Print> ::= ∈

<Expression> ::= <Expression> <Operator> <Expression>
<Expression> ::= <Identifier>
<Expression> ::= <Number>
<Expression> ::= <Boolean> ? <Expression> : <Expression>

<Operator> ::= =
<Operator> ::= *

<Operator> ::= /
<Operator> ::= +
<Operator> ::= -

<Boolean> ::= true
<Boolean> ::= false
<Boolean> ::= <Expression> <Comparator> <Expression>
<Boolean> ::= not <Boolean>

<Comparator> ::= ==
<Comparator> ::= >
<Comparator> ::= <
<Comparator> ::= >=
<Comparator> ::= <=
<Comparator> ::= !=
<Comparator> ::= &&
<Comparator> ::= ||

<Datatype> ::= int | string | float

<Identifier> ::= atom()
<String> ::= [a-zA-Z] [a-zA-Z0-9]

<Number> ::= Digit | <Number> <digit> | <Number> . <Number>
Digit ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

∈ ::= []

Design of Language :

Program : Program can have a block consisting of declarations and commands.

Block : Block can have multiple declarations and commands.

Syntax for block :
```
 block
 {
   // declarations
   // commands
```

```
            //print statement
          }
```

Declarations: The language supports declaration of constant and variables.

declaration support three data types: int, string and float

```
    for eg. int x = 8
    y = "string"
    float = 5.5
```

Constants can be declared using the **'const'** keyword followed by identifier and assigned value.

```
            for eg.:- const x = 8
```

Constraint : Constants can have only numeric values.

Commands: Commands in this language can be multiple conditional constructs such as 'for', 'while' or 'if else'. Initialization is also a part of commands.

Control Structures: Language supports if-else control statements

Syntax:

```
if(Conditional expression)
 {
  //Declaration
  //Commands

 }
else
{
   //Declarations
   //Commands

}
```

Loop Structures : Language supports - for loop, while loop and for i in range loops.

Syntax of while loop :-

```
while(Condition expression)
{
 //block
}
```

Syntax of for loop :-

```
for(Condition expression)
{
 //block

}
```

Syntax of for i in range loop:-

```
for (identifier in range (number , number) )
{
 //block
}
```

Ternary operator :
Identifier = Condition ? (Expression evaluated if condition is true) : (Expression evaluated if condition is false)

Expressions : arithmetic expressions can be evaluated using arithmetic operators.
Ex : x=2+3 or y=6*3.

Arithmetic operators : Arithmetic operations supported are :
Addition (+) , Subtraction (-), Multiplication (*), Division (/)

Conditional and logical operators : Language supports the following comparisons :
== , <=, >=, >, < , != , &&, ||

Print Statement : The print statement in the language is **‘print’.**
ex: print(identifier)

Choice of tools for Language:

lexical analyser: using Lex

Parser: using Prolog

Interpreter: Initially we planned to develop an interpreter in Java but we are also considering Prolog now.

GIT Repository :

https://github.com/CompilerDesignTeam8/SER502-Spring2021-Team8.git