

# Syntax of Kachua Language

CS639A: Program Analysis, Verification and Testing (PAVT)

The assignments in this course will be based on a new (fun) program analysis framework called **Kachua**, that we have developed for this course. We have designed a very simple and easy to understand programming language, the Kachua language, which inspired from the Logo programming language.

A Kachua program is a collection of statements and the program is saved in a file with `.tl` extension. The grammar of Kachua language can be found at `KachuaCore/turtparse/tlang.g4`. You are encouraged to take a look at the grammar file to better understand the syntax of Kachua language. However, we have sufficient examples in this document to help you understand the syntax of Kachua language.

## 1 VARIABLE NAMING CONVENTION

A variable has the naming convention shown by the regular expression: `:[a-zA-Z\_][a-zA-Z0-9]*`. As you can see the variable name always starts with a colon (:) symbol. For example, `:vara10` is a valid variable name, however `_vara` is not.

## 2 STATEMENTS

Kachua programming language support the following types of statements. In Kachua language a block of statements is enclosed by square brackets unlike the curly braces in C.

- (1) Assignment statement
- (2) If statement
- (3) If-else statement
- (4) Repeat statement
- (5) Move statement
- (6) Pen statement
- (7) Goto statement

### 2.1 Assignment Statement

In an assignment statement you can only assign arithmetic expression or a constant value to a variable.

*`<variable> := <expression>`*

We show some examples below,

- `:move = 20`, assigns a constant value 20 to variable `:move`
- `:x = :y + :z`, evaluates `:x + :y` and assigns it to variable `:x`

### 2.2 If Statement

*`if (<condition>)[<statement(s)>]`*

```

1  if (:x > 10) [
2      :x = :x - 1
3  ]

```

List. 1. if :x is greater than 10 then execute (:x = :x -1) else do nothing.

### 2.3 If-else Statement

*if (<condition>)[<statement(s)>] else [<statement(s)>]*

```

1  if (:y1 > 42) [
2      :y2 = :z - 20
3
4  ] else [
5      :y2 = :w - 20
6      :y1 = :x1 + 20
7  ]

```

List. 2. if :y1 is greater than 42 execute the then branch else execute the else branch.

### 2.4 Repeat Statement

*repeat <value>[<statement(s)>]*

You can implement loop using `repeat` keyword in Kachua language. The *<value>* above can be either a variable name or a constant, any other expression is not supported.

```

1  repeat 10 [
2      :vara = :vara + 1
3  ]

```

List. 3. The body of the repeat statement is executed 10 times.

### 2.5 Move Statements

*<move\_command> = <expression>*

A move command is either of forward, back, up, down and is used to move the kachua sprite on the canvas. Some examples are shown below.

```

1  :varb = 200
2  forward :varb

```

List. 4. The kachua sprite moves 200 steps forward.

```
1 right 90
2 backward 200
3 left 10
```

List. 5. The kachua sprite rotates right by an angle 90 degree then moves 200 steps forward and finally rotates left by 10 degree.

## 2.6 Pen Statements

*⟨pen\_command⟩*

The pen statement here (*⟨pen\_command⟩*) takes no argument. There are two types of pen statements penup and pendown. pendown tells the interpreter to draw on the canvas when the kachua sprite moves next. penup tells the interpreter not to write anything on canvas on movement of the kachua sprite.

```
1 pendown
2 forward 80
3 penup
4 forward 20
```

List. 6. For the first forward statement, kachua sprite will move 80 steps forward and while moving it will draw a line. For the second forward statement, the kachua sprite will move 20 steps forward but this time it will not draw anything since the penup was called just before the forward statement.

## 2.7 Goto Statement

*goto(⟨x – coordinate⟩, ⟨y – coordinate⟩)*

The goto command moves the kachua sprite from the current position to the given x and y coordinates by following the shortest path.

```
1 pendown
2 goto (100, 100)
```

List. 7. This will draw a diagonal line in the direction of the kachua sprite's movement from the origin to the coordinate (100, 100) in the canvas.