LOGO

Yanzhao Li

college of science

University of Arizona

Tucson, AZ 85719

yanzhaoli@email.arizona.edu

Yunxiao Hu

college of science

University of Arizona

Tucson, AZ 85719

yunxiaohu@email.arizona.edu

April 28, 2020

1 Abstract

The goal of this guide is to do a simple introduction to the Logo language. To let the audience know the basic concepts of writing programs in Logo programming language and what features are in this language.

2 Introduction

The Logo programming language is an educational programming language. It was designed in 1967 by three people. This programming language is always used to teach children. This programming language does not have complex types, nor complex sub-programs. it can easily to draw the things by code. We make a game as final project. It let user to control a machine to pick up the apples on the screen. If the machine hit the wall, it will be broken, and the game will finish. It will count the score for user. Interesting fact about the loge programming language is it has a turtle on the screen and the user can use code let the turtle go forward or turn around. It will leave a trail of turtle on the screen.

3 History

Logo is an educational programming language, designed in 1967 by Wally Feurzeig, Seymour Papert, and Cynthia Solomon. Logo is not an acronym: the name was coined by Feurzeig while he was at Bolt, Beranek and Newman, and derives from the Greek logos, meaning word or thought. The goal was to create a mathematical land where children

could play with words and sentences (WIKI). In the whole world, there still are many people learning and using this language although sometimes some people may not notice it.

4 Control Structures

```
A. If statement
Example code
to if_else
make "x false
ifelse :x [ print [ this won't be printed. ] ] [ print [ this will be
printed.] ]
make "y true
if :y [ print [ this will be printed.] ]
end
```

In Logo, to declare a variable we need to use "make" and '"', like: make "x false. This means declare a variable x = false. ifelse means if condition is true then executes the first block, else execute the second block. Example: ifelse :x [print [this won't be printed.]] [print [this will be printed.]] First we declare x = false, so, in this part of code, program can only execute the second block: [print [this will be printed.]] What's more in Logo, when we want to use a variable we declare, we need to add a '"' before the variable name, like '"x'.

```
B. For loop
Example code:
to for_loop
for [i 1 100] [fd :i * 10 rt :i]
end
```

Logo for loop the control condition is for [i 1 100] which means for (i=1; i<=100; i++) in java. The part which we need to notice [i 1 100] includes 100.

```
C. While loop
Example code:
to my_while
```

```
make "number sum random 99 1
while [:number<90] [
print [ this  will be  printed.]
make "number sum random 99 1 ]
end
```

While loop control condition is: while [:number<90]. This means program executes the print [this will be printed.] make "number sum random 99 1] until the variable number larger or equal 90.

D. Repeat

Example code:

```
to star
  repeat 5 [ fd 100 rt 144 ]
end
```

In logo we have an very interesting control structure: repeat which can simple control the program to execute a block how many times. In the example code, program can execute 5 times [fd 100 rt 144].

E. Forever

Example code:

```
to myforever
forever [print "hi]
end
```

In Loge, another interesting control structure: forever which can let program execute a block infinite time. To stop it, we can set a count variable like repcount then set a if statement to control it.

F: STOP

Example code:

```
to mystop
repeat 20 [MAKE "x REPCOUNT ifelse :x>10 [STOP] [print repcount]]
end
```

Logo gives us an instruction which can stop and jump out of the block. Just as same as "break" in java or C. In example code, program executes the block 20 times, however, in the block we have a ifelse statement. When x > 10, program will execute [STOP] block which can stop the program and jump out of the block.

5 Data Types

There are three types in Logo. Word, number, list, arry.

Word :

Word is an ordered collection of characters, delimited by spaces, square brackets, parentheses, braces and arithmetic operators. When we want to Logo read the word that itself, we need to quote mart in front of it. Like: "Hello If a word includes a a space, we should use the following idiomatic form: print "|Hello, World!|

Number :

Numbers are special kinds of words, that happen to contain only digits, the decimal point and the character e. The legal number can be: 1 1.9 0.01 0.05 .7 4. 1.1e1 0.2e2 .6e3. What's more, numbers are always evaluated as themselves, that is their value after evaluation is what it was before evaluation. On the other hand, we can quote numbers, if we want.

List :

List are ordered collections of elements. Lists elements can be words, numbers, other lists or, as in the third example, arrays. For example: [a b c 1], [a [b 1] d], [a {b 1} d]. What's more, if the list only contains words and numbers, we call it as the sentence.

Array :

As lists, arrays are ordered collections of elements. Differently from lists, arrays have a fixed length that should be declared beforehand, while lists can grow and shrink at will. Array elements can be words, numbers, other arrays or lists. And Braces delimit an array. For example, {a b c 1}, {a [b 1] d}, {a {b 1} d}.

6 Subprograms

In Logo, if we want to write a function we need to use the structure: "to function-name parameters .....the body..... end" for example in p4.program.lgo, we have a function named DrawWall.

```
to DrawWall :X :Y
    MoveToGrid.LowerLeft :X :Y
    setfloodcolor 0
    bitblock :GridSize :GridSize
end
```

In this function DrawWall, we can see there are two parameters X, Y then execute some code goto the end and end of the function. What's more, if we want to return a value to another function, we need to use a command named: output. To use the output, there is an example code:

```
to area :radius
output product 3.14159 power :radius 2
end
```

From the code above, we can know the function named "area", and the parameter is radius. The output return the value of area of a circle. The output we can treat it almost like return in java or C.

Recursion is always useful and interesting. Fortunately, Logo can allow us to use recursion, but how to write a recursion function in Logo, here is an example:

```
to factorial :n
if equalp :n 0 [output 1]
output product :n factorial difference :n 1
end
```

This function can compute the factory of an integer. In Logo we know we can use if statement, thus, the base case in this function is when the parameter n equals to zero. So, we use "if equalp :n 0 [output 1]", when n equals to zero return value 1. Otherwise, do the recursion part "output product :n factorial difference :n 1", call function factorial and parameter is "n-1", then do the multiplication. In java like: return n*factorial(n-1);

7 Summary

To the summary, this guide does a simple introduction about the Logo programming language. It can let the audience know how to write a simple Logo program. It discusses the control structures, data types and some simple examples to

show how to use the Logo programming language. What's more, it also does a little introduction about the history of the Logo programming language.

8 References

[4] [5] [2] [3] [1]

References

[1] Learn fun with logo simply easy learning, https://www.tutorialspoint.com/logo/index.htm.

[2] Allen B. Downey and Gay Guido. How to think like a computer scientist. 2003.

[3] Bob DuCharme. Logo for kids: An introduction.

[4] Brian Harvey. Berkeley logo 6.0.

[5] Roy D Pea. Logo programming and problem solving. 1987.