INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Nombre: Cruz López Adrián

Grupo: 3CM15

Asignatura: Compiladores

Profesor: Roberto Tecla Parra

Proyecto Final: Mini Logo
Manual Técnico

Fecha: 17/12/2021

Índice

Instrucciones básicas

Instrucción FORWARD [longitud]

Declaración que hará el usuario utilizando la palabra "**FORWARD**", indicando el parámetro dentro de los corchetes "**[ ]**", y se deberá de colocar un número entero el cual indicará la longitud que el puntero de Mini Logo que se desee avanzar dependiendo de la posición y ángulos en la que se encuentre el puntero, el cual para indicar el fin de esta instrucción, deberá agregarse "**;**".

Sintaxis

FORWARD [longitud];

Instrucción TURN [ángulo]

Declaración que hará el usuario utilizando la palabra reservada "**TURN**", indicando el parámetro a través de los corchetes "**[ ]**", se deberá de colocar un número entero que indicará un número comprendido entre 0 y 360, esto para que el puntero de Mini Logo gire en la dirección deseada, el cual, para indicar el fin de esta instrucción, deberá agregarse "**;**".

Sintaxis

TURN [ángulo];

Instrucción COLOR [n1, n2, n3]

Declaración que hará el usuario utilizando la palabra "**COLOR**", indicando el parámetro dentro de los corchetes "**[ ]**", y se deberán introducir 3 números enteros entre 0 y 255 que corresponden a valores RGB, que indicarán el color del trazado del puntero de Mini Logo. Para indicar el fin de esta instrucción, deberá agregarse "**;**".

Sintaxis

COLOR [n1, n2, n3];

Funciones y procedimientos

Función

Es un bloque de código que realiza alguna operación. Una función puede definir estricta u opcionalmente los parámetros de entrada que permiten pasar argumentos a una función. Las funciones son útiles para encapsular las operaciones comunes en un solo bloque reutilizable, idealmente con un nombre que describa claramente lo que hace la función.

De esta forma, se enlista a la función con la sintaxis que sigue, la cual se constituye con la construcción de la función con la palabra reservada "**function**" seguida de paréntesis "**( )**", y que encierran al cuerpo de la sentencia entre llaves "**{ }**". Además, de manera opcional se puede regresar un valor utilizando la palabra "**return**".

Sintaxis

```
function nombredefuncion () {
instrucciones;
[return valor_parametro;]
}
```

Es importante mencionar que se pueden hacer uso de los argumentos a través de la posición de estos mismos, con el operador "**$n**", en donde "**n**" indica la posición para el argumento en la función. No es necesario indicar los parámetros a utilizar, sin embargo, se podrán nombrar dentro de la función.

```
nombredefuncion (parámetro1, parámetro2, ..., parámetroN);
```

Procedimiento

Son un conjunto de instrucciones que se ejecutan sin retornar ningún valor. De esta forma, la construcción de la función se realiza con la palabra "**procedure**" seguida del nombre de este y posteriormente "**( )**", los cuales encierran al cuerpo de la sentencia entre los caracteres "**{ }**".

Sintaxis

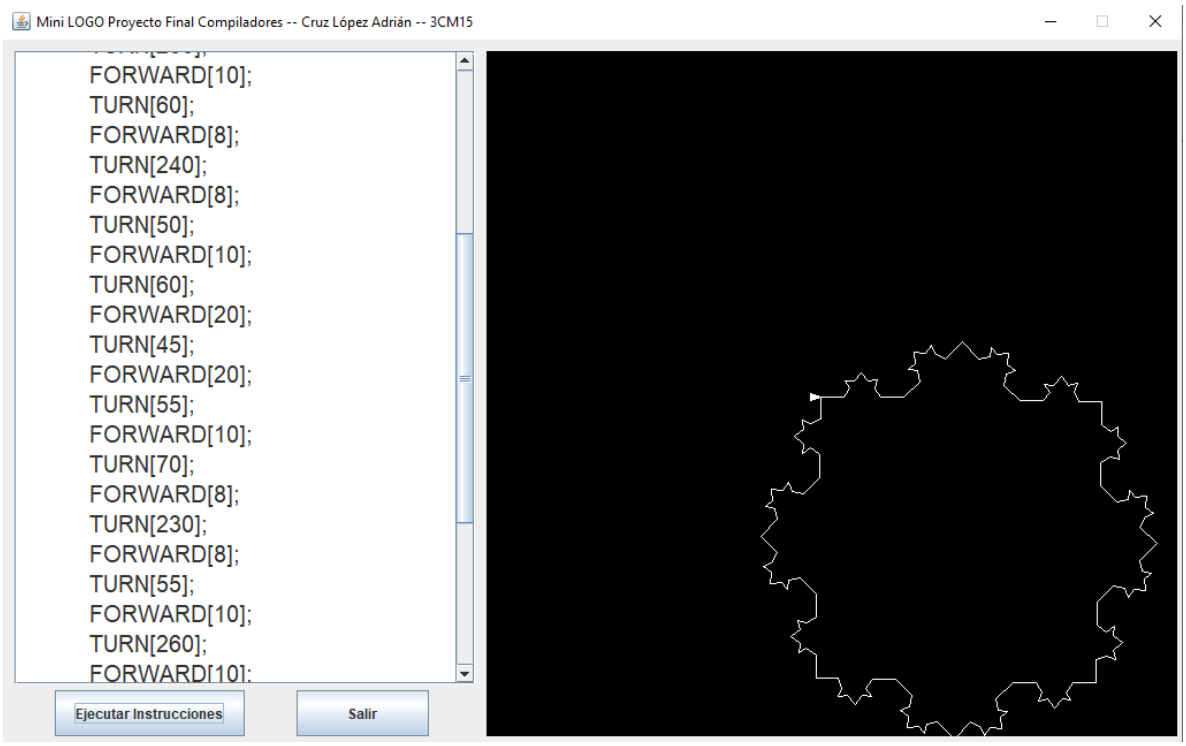
```
procedure nombredeprocedimiento () {
instrucciones;
}
```

Es importante mencionar que se pueden hacer uso de los argumentos a través de la posición de estos mismos, con el operador "**$n**", en donde "**n**" indica la posición para el argumento en la función. No es necesario indicar los parámetros a utilizar, sin embargo, se podrán nombrar dentro de la función.

```
nombredefuncion (parámetro1, parámetro2, ..., parámetroN);
```

Interfaz gráfica

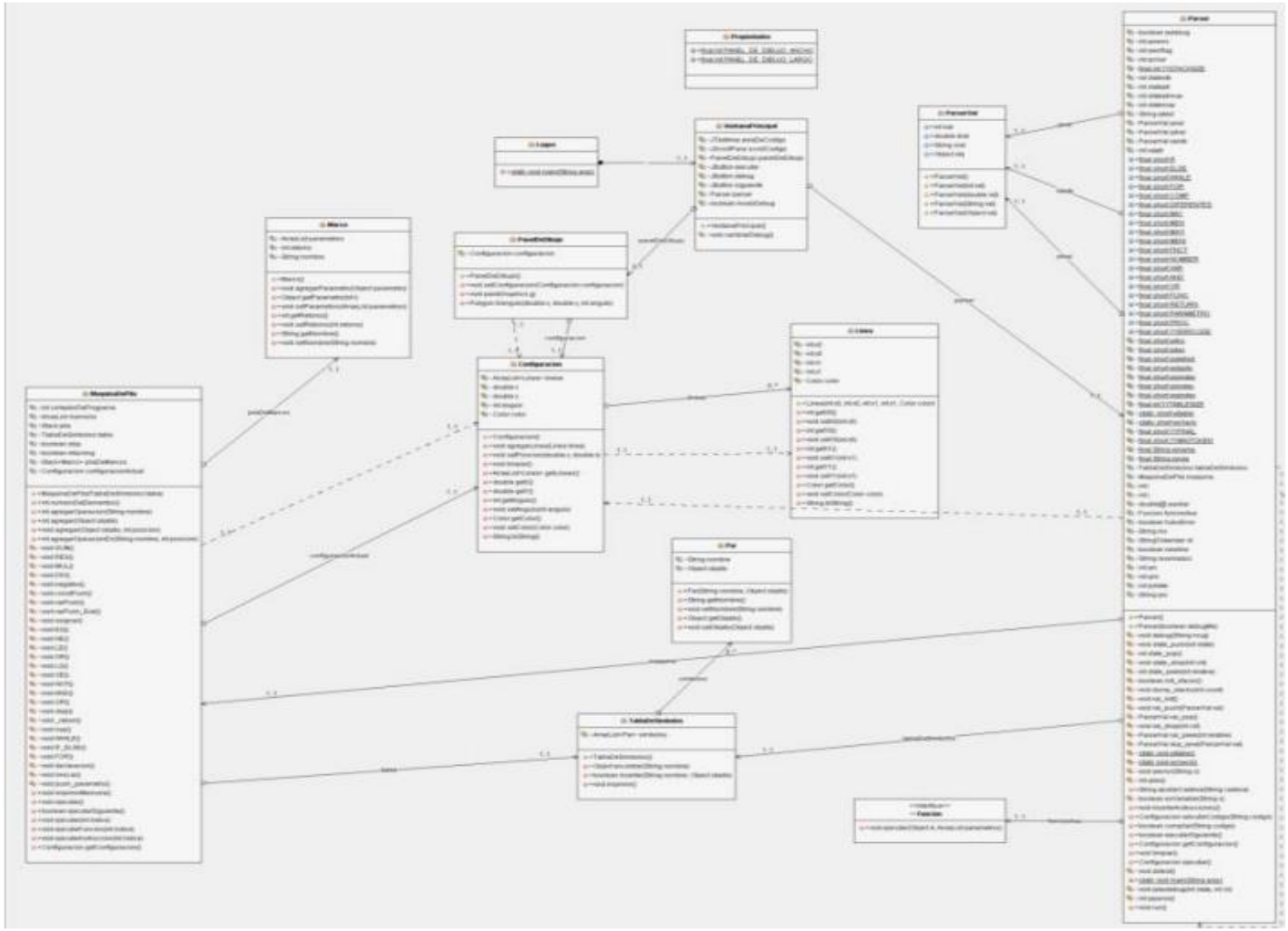
Es importante llevar a cabo la descripción de los elementos que dispone el usuario para hacer uso de la herramienta grafica para hacer la declaración de las instrucciones que desee el usuario, así como la visualización de las instrucciones.



Cuenta con:

1. Un área de texto, donde se ingresan las instrucciones a ejecutar.
2. Un botón “Ejecutar Instrucciones” que permite dibujar las acciones ingresadas.
3. Un botón “Salir” que terminará el programa y cerrará la ventana.
4. Un scroll que se mostrará cuando el código ingresado sea extenso.
5. Un panel de dibujo donde se visualiza la figura.

Diagrama de clases



Gramática

Aquí se define la especificación necesaria para el reconocimiento de instrucciones y declaraciones; una constante o variable; una asignación u operaciones aritméticas. También para la especificación de operadores relacionales y operadores lógicos. Además del reconocimiento del return de alguna función y la estructura principal para declarar un procedimiento y su respectiva lista de argumentos. También es posible observar las especificaciones como el orden que compone a la lista de argumentos separados por comas. La especificación de las sentencias de control e iterativas de cada una de ellas, indicando las partes que conforman su estructura y la delimitación de su condición.

```
    exp: VAR {
            $$ = new ParserVal(maquina.agregarOperacion("varPush_Eval"));
            maquina.agregar($1.sval);
        }
        |'-' exp {$$ = new
ParserVal(maquina.agregarOperacion("negativo"));}
        |NUMBER {
            $$ = new ParserVal(maquina.agregarOperacion("constPush"));
            maquina.agregar($1.dval);
        }
        | VAR '=' exp {
            $$ = new ParserVal($3.ival);
            maquina.agregarOperacion("varPush");
            maquina.agregar($1.sval);
            maquina.agregarOperacion("asignar");
            maquina.agregarOperacion("varPush_Eval");
            maquina.agregar($1.sval);
        }
        | exp '*' exp {
            $$ = new ParserVal($1.ival);
            maquina.agregarOperacion("MUL");
        }
        | exp '+' exp {
            $$ = new ParserVal($1.ival);
            maquina.agregarOperacion("SUM");
        }
        | exp '-' exp {
            $$ = new ParserVal($1.ival);
            maquina.agregarOperacion("RES");
        }
        |'(' exp ')' {$$ = new ParserVal($2.ival);}
        | exp COMP exp {
            maquina.agregarOperacion("EQ");
            $$ = $1;
        }
        | exp DIFERENTES exp {
            maquina.agregarOperacion("NE");
```

```
            $$ = $1;
        }
        | exp MEN exp {
            maquina.agregarOperacion("LE");
            $$ = $1;
        }
        | exp MENI exp {
            maquina.agregarOperacion("LQ");
            $$ = $1;
        }
        | exp MAY exp {
            maquina.agregarOperacion("GR");
            $$ = $1;
        }
        | exp MAYI exp {
            maquina.agregarOperacion("GE");
            $$ = $1;
        }
        | exp AND exp {
            maquina.agregarOperacion("AND");
            $$ = $1;
        }
        | exp OR exp {
            maquina.agregarOperacion("OR");
            $$ = $1;
        }
        | '!' exp {
            maquina.agregarOperacion("NOT");
            $$ = $2;
        }
        | RETURN exp { $$ = $2; maquina.agregarOperacion("_return"); }
        | PARAMETRO {
            $$ = new
ParserVal(maquina.agregarOperacion("push_parametro"));
            maquina.agregar((int)$1.ival);
        }
        //Instruccion necesaria para el acomodo de la lista de argumentos
        |nombreProc '(' arglist ')' {
            $$ = new
ParserVal(maquina.agregarOperacionEn("invocar",($1.ival)));
            maquina.agregar(null);
        }
        ;

    arglist:
        |exp {$$ = $1; maquina.agregar("Limite");}
        |arglist ',' exp {$$ = $1; maquina.agregar("Limite");}
        ;

    nop: /*NoOp*/ {$$ = new ParserVal(maquina.agregarOperacion("nop"));};
    stmt: if '(' exp stop ')' '{' linea stop '}' ELSE '{' linea stop'}' {
            $$ = $1;
            maquina.agregar($7.ival, $1.ival + 1);
```

```
            maquina.agregar($12.ival, $1.ival + 2);
            maquina.agregar(maquina.numeroDeElementos() - 1, $1.ival +
3);
        }
        | if '(' exp stop ')' '{' linea stop '}' nop stop {
            $$ = $1;
            maquina.agregar($7.ival, $1.ival + 1);
            maquina.agregar($10.ival, $1.ival + 2);
            maquina.agregar(maquina.numeroDeElementos() - 1, $1.ival +
3);
        }
        | while '(' exp stop ')' '{' linea stop '}' stop {
            $$ = $1;
            maquina.agregar($7.ival, $1.ival + 1);
            maquina.agregar($10.ival, $1.ival + 2);
        }
        | for '(' instrucciones stop ';' exp stop ';' instrucciones stop
')' '{' linea stop '}' stop {
            $$ = $1;
            maquina.agregar($6.ival, $1.ival + 1);
            maquina.agregar($9.ival, $1.ival + 2);
            maquina.agregar($13.ival, $1.ival + 3);
            maquina.agregar($16.ival, $1.ival + 4);
        }
        | funcion nombreProc '(' ')' '{' linea null '}'
        | procedimiento nombreProc '(' ')' '{' linea null '}'
        | instruccion '[' arglist ']' ';' {
            $$ = new ParserVal($1.ival);
            maquina.agregar(null);
        }
        ;
```