Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

Lenguaje Betes Ra!

Compiladores e Intérpretes

Primer Proyecto

Grupo 20

Profesora:
Aurelio Sanabria

Estudiantes:
Alberto Zumbado Abarca
Jonathan Quesada Salas
Pablo Alberto Muñoz Hidalgo
Jose Pablo Quirós Hidalgo
Luis Andres Rojas Murillo

Alajuela, 2022

Índice

1. **Motivación**

Con lo que respecta a describir la motivación del primer proyecto de Compiladores e Intérpretes desemboca con el desarrollo del lenguaje de programación llamado “Los Betes Ra!”, se centraliza en poder solventar la mayor cantidad de problemas de uso general por medio de una gramática bien estructurada estableciendo un inicio y un final claro en cada bloque de código que pueda llegar a surgir eso con lo que respecta a la temática seleccionada y el cómo se adaptó en cuanto a la gramática respecta. Las estructuras principales como pueden ser la repetición o bifurcación establecer un inicio y un final en el bloque de código y en cuanto a los términos seleccionados pueden destacar la forma en representar las expresiones matemáticas la cual corresponden y se asocian a una lógica matemática de las tablas de verdad con lo que respecta a las negaciones y adicionalmente la definición de respectivas variables en el programa las cuales pueden ser globales y locales.

2. Puntos Fuertes y Chistosos del Lenguaje

Con lo que respecta a las características que hacen único a nuestro lenguaje es que se va a basar en una formación lógica, podemos ejemplificar esto por medio del condicionante “and” y “or” que son representados por símbolos que se asemejan a una aplicación lógica básica, por lo que son fáciles de entender y no requiere conocimiento adicional. Y con lo que respecta las implicaciones se desarrolla de una manera intuitiva la cual se representa de la siguiente manera “=>” lo cual hace que de esta manera que el lenguaje sea más atractivo para los ojos del que lea el código de una forma más lógica y organizada, básicamente un lenguaje que se puede auto explicar de una mejor manera.

Adicionalmente teniendo en cuenta el punto anterior sobre la formulación lógica del lenguaje, también una característica que lo hace único es su tipado en español y el uso de símbolos combinados con letras para definir acciones o variables, véase de ejemplo las funciones, comentarios o asignaciones de variables. Por último una característica inmensamente diferenciadora y valiosa es no tener la necesidad de declarar el tipo de variable que retorna una función o el hecho de que retorne una

variable, lo podemos ver como que el lenguaje asume un void hasta que se retorna una variable o varias de un mismo tipo de esa función.

Unos puntos a recalcar que pueden entrar en la categoría de graciosos es que los retornos y los breaks en el lenguaje de “Los Betes Ra!” es que el retorno es la referencia a un meme muy famoso el cual es “messirve” el cual acotentese a una entrevista que tuvo Messi y este se puso unos lentes negros muy lindos.

Y por otra parte el break cuando se necesita hacer un alto en un recorrido de una función o procedimiento se declarará la palabra “siu” haciendo referencia a la icónica frase o expresión que hace Cristiano Ronaldo cuando mete un gol el cual la hace como celebración con un postura firme el cual llega a imponer respeto y admiración.

3. Gramáticas EBNF

Programa ::= (Función | Comentario | Condicional | Asignación | Ciclo | Invocar) +

Comentario ::= #C.*

Asignación::= TipoVariable TipoValor Identificador (<=>) (Operación | Valor);

Operación ::= Operar Valor (Operadores Valor)+

Operadores ::= (+ | - | * | ** | / | // | %)

Función ::= Func Identificador {Parámetro} -> (Instrucciones)+ <-

Invocar ::= Inv Identificador {Parámetro}

Condicional ::= Si {ExpCondicional} -> Instrucciones + <-

ExpCondicional ::= (Comparación ((^ | ~^) Comparación)*))

Comparación ::= ((Valor | Operación) Comparador (Valor | Operación)

Comparador ::= (< | > | <= | >= | ~== | ==)?

Identificador ::= [a-z A-Z][A-Z a-z 0-9 _]*

Parámetro ::= TipoValor Identificador (, TipoValor Identificador)*

Valor ::= (Identificador | Literal)

Ciclo ::= Rep {Comparación} -> Instrucciones + | Rompe <-

Instrucciones ::= (Ciclo | Condicional | Retorna | Asignación | OperarVariable | Invocar)

OperarVariable::= Identificador (<=>) (Operación | Valor);

Literal::= (Texto | Entero | Flotante | Lectura | Nulo)

Texto::= “ ([^"]*) “

Entero ::= -?[0-9]+

Flotante ::= -?([0-9]*[.])[0-9]+

Nulo ::= Nulo

TipoVariable ::= (GLOB | CONS | LOC)

Retorna ::= messirve Valor;

Rompe ::= siu;

Lectura ::= Identificador[Entero]

TipoValor::= (Texto | Entero | Flotante)

4. Ejemplos de Código

4.1 Ejemplo 1 (Factorial con condicional):

```
#GLOB #Entero resultado <=> 0;
#F factorialCondicional{ #Entero numero } => $
        #R ((numero - 1)  > 0) => $
                #LOC #Entero i <=> numero - 1;
                resultado <=> (( numero * i ) + resultado);
                numero <=> numero - 1;  $
        messirve resultado;
$
```

4.2 Ejemplo 2 (Área de un círculo):

```
#GLOB #Flotante pi <=> 3.14;

#F areaCirculo{#Flotante radio} => $

        #LOC #Flotante area <=> 0;

        #LOC #Flotante operacion <=> pi * radio ** 2;

        #LOC #Flotante area <=> operacion;

        messirve area;

$
```

4.2 Ejemplo 2 (Comparación de textos):

```
#GLOB #Entero exito <=> 1;
#GLOB #Entero fracaso <=> 0;
#F comparadorTexto {#Texto texto1, #Texto texto2} => $

        #C Se validan que los textos no sean nulos #C

        #LOC #Entero recorrerTexto1 <=> 0;
        #R (texto1[recorrerTexto1]  ~== Nulo)$
                recorrerTexto1 <=> recorrerTexto1  + 1;
        $

        #LOC #Entero recorrerTexto2<=> 0;
        #R (texto2[recorrerTexto2]  ~== Nulo)$
                recorrerTexto2 <=> recorrerTexto2 + 1;
        $

        #C Se comparan los textos sin ser nulos #C

        Si (texto1 == texto2) => $
                messirve exito;
        $
        messirve fracaso;
$
```

4.3 Ejemplo 3 (Suma de dígitos hasta 0):

```
#F sumaDigitos{#Entero numero} => $
```

```
        #LOC #Entero suma <=> 0;

        #LOC #Entero digito <=> 0;

        #R (numero ~== 0) $

                digito <=> numero % 10;

                suma +<=> digito;

                numero <=> numero // 10;

        $

        messirve suma;

$
```

4.4 Ejemplo #4 (Ocurrencias del dígito en un número):

```
#F frecuencia {#Entero numero, #Entero digito} => $

        #LOC #Entero cantidad <=> 0;
        #LOC #Entero ultimo_digito <=> 0;

        #R (numero ~== 0) $
                ultimo_digito <=> numero % 10;

                Si (ultimo_digito == digito) => $

                        cantidad = cantidad + 1;
                $
                numero <=> numero // 10;
        $
        messirve cantidad ;
$
```

4.5 Ejemplo #5 (Palíndromo):

```
#F palindromo {#Texto palabra1, #Texto palabra2} => $

        #LOC #Entero i <=> 0;
        #R (palabra1[i]  ~== Nulo)$
                i <=> i + 1;
                $
        #LOC #Entero j <=> 0;
        #R (palabra2[j]  ~== Nulo)$
                j <=> j + 1;
                $
```

```
        #LOC #Entero longitud1 <=> i;
        #LOC #Entero longitud2 <=> j;

        Si ( longitud1 ~== longitud2 ) => $
                messirve Nulo;
                $
        #LOC #Entero contador <=> 0;
        #LOC #Entero contadorDesendente <=> longitud2 - 1;

        #R ( contador < longitud1 )$
                Si ( palabra1[contador] ~== longitud2[contadorDesendente] )=> $
                messirve Nulo;
                $
                contador <=> contador + 1;
                contadorDesendente <=> contadorDesendente - 1;
                $
        siu;
        $
$
```