Trabajo fin de grado sobre la construcción de un entorno de aprendizaje para la programación.

Ezequiel Santamaría Navarro

November 16, 2015

Contents

Abstract

Abstracto

Parte 1

Introducción

Introducción

Parte 2

Lenguaje zl

2.1 Introducción

El lenguaje está enfocado a cumplir las siguientes propiedades:

- Enfocado a parecerse sintáticamente al pseudocódigo en castellano, parecido al enseñado en la asignatura IP.
- Diferenciando los tipos de datos, sin hacer implícitas sus conversiones ni sus operaciones.
- Insensible a mayúsculas y minúsculas, y aceptando tildes y eñes en los nombres.
- Con un único tipo de datos para representar números enteros y decimales (similar a Javascript).

2.2 Definición formal del lenguaje

La gramática del lenguaje en formato BNF es:

```
; Insensible a mayúsculas y minúsculas
; Notación BNF
; <_> significa espacio en blanco (cualquier tipo de espacio en blanco), o un comentario
; Algunos de los espacios son opcionales, allá donde no haya ambiguedad:
; por ejemplo: a.b o a<-b


; Reglas gramaticales
; nombreSimple expresión regular ([A-Za-záéíóúÁÉÍÓÚñÑ][A-Za-záéíóúÁÉÍÓÚñÑ0-9]*)
; numero expresión regular ((?:[0-1]+(?:\|2))|(?:[0-9A-Fa-f]+(?:\|16))|(?:[0-9]+(?:\|10)?))
; texto expresión regular \"([^"\\]|\\.|\\\n)*\"
; letra expresión regular \'([^"\\]|\\.|\\\n)*\'
```

```
; comentario expresión regular \/\/.*

; Nota, para las expresiones del mismo nivel,
; el arbol que se construye se reordena para evaluar primero a la izquierda.

<programa> ::= [<configuraciones> <_>] <subrutina>+

<configuraciones> ::= "configuracion" <_> (<configuracion> <_>)* "fin"

<configuracion> ::= "importar" <_> <nombre>
    | "configurar" <_> <nombre> <_> "<-" <_> <expresion>

<subrutina>  ::= <subrutinaCabecera> <_> <subrutinaCuerpo>

<subrutinaCabecera>  ::= "subrutina" (<_> <modificador>)* [<_> <nombre>]

<subrutinaCuerpo>  ::= <datos> <_> <algoritmo> <_> "fin"

<modificador> ::= "externa"
    | "es"
    | "rápida"
    | "rapida"

<nombre> ::= <nombreSimple> (<_> "." <_> <nombre>)*

<datos> ::= "datos" <_> (<declaracion> <_>)+

<algoritmo> ::= "algoritmo" <_> (<sentencia> <_>)+

<declaracion> ::= <nombre> <_> "es" <_> <nombre> (<_> "de" <_> <declaracionModificador>)*

<sentencia> ::= <asignacion>
    | <ecuacion>
    | <llamada>
    | <repetir>
    | <sicondicional>
    | <mientras>

<declaracionModificador> ::= "entrada"
    | "salida"

<asignacion> ::= <nombre> <_> "<-" <_> <expresion>

<ecuacion> ::= <expresion> <_> "=" <_> <expresion>

<llamada> ::= <nombre> <_> "->" <_> <nombre> <_> "->" <_> <nombre>
    | <nombre> <_> "->" <_> <nombre>
    | <nombre> <_> "[" <_> (<llamadaAsignacion> <_>)+ "]"

<repetir> ::= "repetir" <_> <expresion> <_> "veces" <_> (<sentencia> <_>)+ "fin"

<mientras> ::= "mientras" <_> <expresion> <_> "hacer" (<sentencia> <_>)+ "fin"

<sicondicional> ::= "si" <_> <expresion> <_> "hacer" <_> (<sentencia> <_>)+ "fin"
    | "si" <_> <expresion> <_> "hacer" <_> (<sentencia> <_>)+ <sinocondicional>
    | "si" <_> <expresion> <_> "hacer" <_> (<sentencia> <_>)+ <sino>
```

```
<sinocondicional> ::= "o" <_> "si" <_> <expresion> <_> "hacer" <_> (<sentencia> <_>)+ "fin"
    | "o" <_> "si" <expresion> <_> "hacer" <_> (<sentencia> <_>)+ <sinocondicional>
    | "o" <_> "si" <expresion> <_> "hacer" <_> (<sentencia> <_>)+ <sino>

<sino> ::= "si" <_> "no" <_> "hacer" <_> (<sentencia> <_>)+ "fin"

<llamadaAsignacion> ::= <expresion> <_> "->" <_> <nombre>
    | <nombre> <_> "<-" <_> <expresion>

<expresion> ::= <expresionTercera> <_> <operadorBinarioCuarto> <_> <expresion>
    | <expresionTercera>

<expresionTercera> ::= <expresionSegunda> <_> <operadorBinarioTercero> <_> <expresionTercera>
    | <expresionSegunda>

<expresionSegunda> ::= <expresionPrimera> <_> <operadorBinarioSegundo> <_> <expresionSegunda>
    | <expresionPrimera>

<expresionPrimera> ::= <operadorUnario> <_> <expresion>
    | <evaluacion> <_> <operadorBinarioPrimero> <_> <expresionPrimera>
    | <evaluacion>

<evaluacion> ::= <numero>
    | <texto>
    | <letra>
    | "verdadero"
    | "falso"
    | <nombre>
    | "(" <_> <expresion> <_> ")"

<operadorUnario> ::= "-"
    | "+"
    | "no"

<operadorBinarioCuarto> ::= "y"
    | "o"

<operadorBinarioTercero> ::= "<"
    | ">"
    | "<="
    | ">="
    | "="
    | "<>"

<operadorBinarioSegundo> ::= "+"
    | "-"

<operadorBinarioPrimero> ::= "*"
    | "/"
```

Cabe destacar de la gramática las 4 expresiones y los cuatro grupos de operadores, para poder definir correctamente el orden en el cuál los operadores se reducen.

Por otro lado, los número en el lenguaje permiten indicar base 2, 10 o 16 de la siguiente manera:

- 51966 como número en base 10
- 51966|10 como el mismo número
- $CAFE$|16 o $cafe$|16 como el mismo número en base hexadecimal.
- 1100101011111110|2 como el número en base binaria.