Informe escrito de Proyecto de Programación II: Intérprete para HULK

Carlos Daniel Largacha Leal

Grupo C112

Primer año de Lic. en Ciencia de la Computación
Facultad de Matemática y Computación
Universidad de La Habana
Curso 2023-2024

Contents

1 Introducción

HULK es un lenguaje de programación imperativo, funcional, estático y fuertemente tipado. Casi todas las instrucciones en HULK son expresiones. En el presente proyecto se implementará un intérprete de un subconjunto, de dicho lenguaje de programación, compuesto solamente de expresiones que pueden escribirse en una única línea. Cada tipo de expresión del HULK es tratada en clases dedicadas a cada expresión, por lo que abordaremos toda su funcionalidad a la par que explicamos el funcionamiento del intérprete.

2 Clase Main

Esta clase es la encargada de recibir la línea de código introducido por el usuario y de enviarlos a la clase Tokenizer, además de imprimir en pantalla el resultado de procesar la expresión introducida

3 Clase Tokenizer

Como su nombre lo indica esta clase es la encargada de separar en tokens la línea de código para facilitar su procesamiento en la clase Parser, en el método tokenizer se van guardando todos los tokens en una lista, también se encarga de revizar si todos los paréntesis están balanceados y de revizar que los tokens de tipo string tengan sus comillas de cierre.

4 Clase Parser

Esta clase contien las listas donde se almacenan las funciones declaradas y las variables con sus respectivos valores. El método Begin Parser recive la lista de tokens y los analiza para detectar que tipo de expresión tiene que procesar el interprete, en este método se captan la mayoría de los errores, así como se dectecta si un token es invalido al no cumplir con ninguna de las características que busca el método.

5 Clase Let In Parser

Se encarga de procesar las expresiones de tipo Let in, en el método AssignVariable se chequea que la expresión sea vádila y posteriormente se guardan los nombres y los valores de las variables asignadas con el let, al termira de procesar la expresión let in las variables y su valor asignado se eliminan.

Ejemplo:
Let x = 4 in x + 10;
14

6 Clase Print Parser

Confirma que una epresión de tipo print sea válida, envia a la clase Tokenizer el string encerrado en los parénteis.
Ejemplo:
print(”El sentido de la vida”);
El sentido de la vida

7 Clase Boolean Tokenizer

Clase encarga de procesar las expresiones booleanas

8 Clase If Else Parser

Esta clase se encarga de confirmar que las expresiones If else sean válidas y de devolder el resultado a corde si la condición es verdadera o falsa.
Ejemplo:
If (7 != 3) print(7) else print(3);
7

9 Clase Function Parser y Function Arguments

Son las encargadas de procesar las expresiones de tipo function, estas confirman la validez de la expresion y procesen a almacenar las funciones con sus respectiavas variables y argumentos para posteriormente ser usadas en otras líneas de código.
Ejemplo:
function fib(x) = if(x ¡= 1) 1 else fib(x - 1) + fib(x - 2);
function declared

10 Calse Check Math Expresion

Confirma la validez de una expresión aritmética para ser enviada a la clase Arithmetic

11 Clase Arithmetic

Procesa las expresiones aritméticas introducidas en el programa empleando el algoritmo shunting yard