Presentacion Hulk Interpreter

Mauricio Sunde Jimenez C111

Universidad de La Habana

2023

Resumen

HULK es un lenguaje de programación imperativo, funcional, estática y fuertemente tipado. Casi todas las instrucciones en HULK son expresiones. En particular, el subconjunto de HULK que se implemento se compone solamente de expresiones que pueden escribirse en una línea.

Que es Hulk?

HULK(Havana University Language for Kompilers) es un lenguaje de programacion imperativo, funcional, estatico y fuertemente tipado. Casi todas las instrucciones en HULK son expresiones. En particular, el subconjunto de HULK que se ha implementado se compone solamente de expresiones que pueden escribirse en una linea. El interprete esta desarrollado con tecnologia .NET Core 7.0 y en lenguaje CSharp.

Explicacion General

El primer paso en el intérprete es analizar el texto de entrada. El lexer toma el texto sin formato como una serie de caracteres. Despues los examina y los distribuye en una lista de tokens es decir palabras reconocidas por el lenguaje
En primer lugar, necesitamos identificar todos los tipos de tokens que se utilizan en la gramática del idioma. Algunos de estos tipos son:
Los Operadores como *,+,=,-,/,@,Modulo. Caracteres como el punto (.), la coma (,), el punto y coma (;), los parentesis (,). Los Operadores de comparacion >=,==,<=,!=,=,>,<,. Los literales como los numeros, strings, booleanos, o identificadores. Las keyWords del lenguaje como AND,OR,False,True,Function,If,Else,Let,In.
Luego de tener analizado totalmente nuestro input pasamos al parser donde creamos el Arbol de sintaxis abstracta. En este proyecto se usara la tecnica de Parsing descendente recursivo.
El parsing descendente recursivo es un método utilizado en el análisis sintáctico de un lenguaje de programación.

Consiste en construir un árbol sintáctico a partir de una cadena de entrada, aplicando reglas de producción de manera recursiva desde la raíz hacia las hojas del árbol.
El proceso de parsing (análisis sintáctico) se basa en una gramática formal que describe la estructura sintáctica del lenguaje. Esta gramática se representa mediante reglas de producción en forma de BNF (Backus-Naur Form) o alguna otra notación similar.
El análisis sintáctico descendente recursivo se inicia con un símbolo inicial de la gramática y recorre la cadena de entrada de izquierda a derecha. A medida que avanza, va aplicando las reglas de producción correspondientes para construir el árbol sintáctico. El proceso es recursivo porque cada regla de producción puede invocar a otras reglas de producción, de manera que se van explorando todas las posibles combinaciones de derivaciones hasta llegar a las hojas del árbol.
En general, cada no terminal de la gramática se asocia a una función o método en el código del parser, y la llamada a dicha función representa la aplicación de una regla de producción. Estas funciones se llaman recursivamente unas a otras para construir el árbol sintáctico.
Luego cuando nuestro AST esta construido pasamos a evaluar

Prioridad de Operaciones y biblioteca de funciones

1

La prioridad de las operaciones esta implementada de la siguiente manera en TokensPrecedence. 1. !, -, + (como operadores unarios maxima prioridad) 2. $\wedge$ 3. *, /, % 4. +, -, @ 5. $<=, >=, >, <, ==, != 6.$ & 7. |

2

Se han implementado en una biblioteca estatica las siguientes funciones por defecto sin(value) retorna el seno del valor cos(value) retorna el cos del valor sqrt(value) retorna la raiz cuadrada de un numero.
print(expresion) retorna la expresion que se le pasa por argumento.
log(value, base) retorna el logaritmo de un numero en una base especifica.

3

Se han implementado ademas variables globales como lo son. E retorna el Math.E de la biblioteca de Csharp. PI retorna el valor de Math.PI de la biblioteca de Csharp.

Muchas gracias!!!