Informe sobre el proyecto HULK

Abraham Romero Imbert

Noviembre, 2023

Resumen

En este informe se estará explicando de manera más detallada el funcionamiento del Proyecto HULK. Se interiorizará la ejecución del mismo así como en las clases y métodos que intervienen en el proceso para dar una respuesta al usuario, una vez hecha una consulta.

1. Introducción

Es un intérprete del lenguaje de programación HULK.

Este proyecto, consiste en implementar un subconjunto de HULK, ya definido. HULK es un lenguaje mucho más grande. HULK es un lenguaje de programación imperativo, funcional, estática y fuertemente tipado. Casi todas las instrucciones en HULK son expresiones. En particular, este subconjunto de HULK se compone solamente de expresiones que pueden escribirse en una línea.

2. Funcionamiento

¿Cómo funciona?

Un intérprete de un lenguaje de programación es un programa que lee y ejecuta el código fuente escrito en ese lenguaje. El proceso de interpretación generalmente se realiza en tiempo real, es decir, a medida que el código se va leyendo.

El intérprete toma el código fuente y lo analiza para identificar su estructura y significado. Luego, traduce el código a instrucciones que la máquina puede entender y ejecutar. Durante este proceso, el intérprete también verifica la sintaxis del código para detectar posibles errores.

Una vez que el código ha sido interpretado, el intérprete lo ejecuta línea por línea, realizando las operaciones especificadas en cada instrucción. Esto puede incluir cálculos matemáticos, manipulación de datos, interacción con el sistema operativo o cualquier otra tarea que el programa esté diseñado para realizar.

En resumen, un intérprete de un lenguaje de programación funciona como un traductor que lee, analiza y ejecuta el código fuente escrito en ese lenguaje, permitiendo que los programas se ejecuten en un entorno informático.

2.1. Ruta de ejecución

La estructura de un intérprete consta de tres partes principales

Analizador léxico: Se encarga de leer el código fuente y dividirlo en tokens.

Analizador sintáctico: Verifica si la estructura del código es correcta y genera un árbol de sintaxis abstracta.

Ejecutor de código: Ejecuta el código.

2.2. Analizador léxico

Aquí particularmente utilicé Expresiones Regulares para identificar los tokens en la entrada del usuario.

El analizador léxico de un intérprete con expresiones regulares funciona de la siguiente manera:

1. Definición de tokens: Se definen las reglas para identificar los diferentes tipos de elementos del lenguaje, como identificadores, números, operadores, palabras clave, etc.

2. Expresiones regulares: Se utilizan expresiones regulares para definir patrones que representan los diferentes tokens. Por ejemplo, una expresión regular para identificar el signo de suma podría ser -".

3. Análisis del código fuente: El analizador léxico recorre el código fuente caracter por caracter, aplicando las expresiones regulares definidas para identificar los tokens.

4. Generación de tokens: Cuando se encuentra un patrón que coincide con una expresión regular, se genera un token correspondiente y se almacena en una lista para su posterior procesamiento por el analizador sintáctico.

5. Manejo de errores: El analizador léxico también debe ser capaz de manejar errores como caracteres no válidos o tokens mal formados, y reportarlos al usuario.

En resumen, el analizador léxico con expresiones regulares es responsable de identificar y generar los tokens que representan los elementos del lenguaje en el código fuente, utilizando patrones definidos con expresiones regulares. Estos tokens son luego utilizados por el analizador sintáctico para construir la estructura del programa y realizar su interpretación.

2.3. Analizador semántico

Un analizador semántico es una fase del proceso de compilación o interpretación de un programa. Su función es verificar si el programa es semánticamente consistente con la definición del lenguaje. Consta de varias partes en su análisis

Exploración del árbol de sintaxis abstracta: El analizador semántico explora el árbol de sintaxis abstracta (AST). El AST es una representación gráfica del código fuente que se obtiene después del análisis sintáctico. También se agregan las definiciones de funciones y variables dependiendo de su ámbito ya sea local o global.

Detección de errores semánticos: Durante la exploración, el analizador semántico detecta los errores semánticos. Los errores semánticos son aquellos que se producen cuando se transgrede el significado de elementos de expresión.

2.4. Ejecución del código

La ejecución del código se realiza a través del método Execute( ) de la clase Statement-Node que ejecuta y evalua las expresiones y devuelve su valor y ejecuta las sentencias.

2.5. Manejo de errores

El manejo de errores se hace a través de la estructura Try-Catch donde al devolverse cualquiera de los errores que definí en cada una de las partes se imprime la primera que se devuelve.

3. Conclusión

En este informe se ha explicado detalladamente el código implementado por mí para la ejecución del HULK. Se han detallado las clases y los métodos propios de cada una, así como la manera en que se ejecutan estos, o sea, cómo funcionan. En vista de la posibilidad de un posterior mejoramiento de la capacidad de procesamiento de este proyecto o la implementación de otras funcionalidades se expresa que los métodos anteriores pueden estar sujetos a cambios. En mi experiencia particular este como proyecto en la Facultad de Matemática y Computación en la Universidad de la Habana considero que fue un gran paso.