Universidad de La Habana

Facultad de Matemática y Computación

HULK

Proyecto de Programación

John García Muñoz C-111
Curso 2023

¿Qué es HULK?

1. •Descripción

-HULK (Havana University Language for Kompilers) es un lenguaje de programacion interpretado, implementado sobre C#.

2. •Funcionalidad

- El compilador de Hulk funciona a través de un parsing recursivo descendente, es decir, generar un árbol con las distintas operaciones a realizar por el intérprete. El compilador ejecuta un análisis descendente recursivo por la izquierda, de manera que gracias dicha recursividad, ejecuta de manera ordenada todas las instrucciones de una línea, siempre que estas respeten las reglas sintácticas y semánticas del lenguaje.

¿Qué admite Hulk?

Hulk permite al usuario crear y operar tres tipos de datos: string, números y bools, y operar con estos con las operaciones binarias de suma(+), resta(-), multiplicación(*), división(/) y concatenación(@). Aparte dispone de funciones aritméticas reservadas: cos(value), sin(value), log(argument, base), sqrt(value), pow(value, exponent) y rand();

Además, es posible implementar nuevas funciones, como sigue:

function nombreDeFunción(x, y,...) => (expresión);

También es posible declarar nuevas variables, tal como sigue:

let nombreDeVariable = (expresión) in (expresión);

Sin embargo, se ha de tener en cuenta que las variables solo existen en el scope de la expresión que las declara, luego las variables declaradas por un let solo se pueden usar dentro de la expresión del in correspondiente.

Es posible también el uso de condicionales, mediante el tradicional if-else, cuyo funcionamiento varía un poco para adaptarse a ser usado en una sola línea, siendo además,

de obligatorio uso el else:

if (condición) (instrucción) else (instrucción);

Sobre la Compilación

3. •Analizador Léxico

Es la primera parte del programa, donde luego de recibir la entrada (la línea de texto del usuario), la clase Analize contiene, entre otros métodos auxiliares, la función NextToken(), que se encarga de devolver un token correspondiente a cada caracter o sucesión de caracteres de dicha entrada.

4. •Analizador Sintáctico

El analizador sintáctico se encuentra en la clase Syntax, y su primera función es aplicar iterativamente el método NextToken() de Analize para generar la lista de

tokens a analizar. Luego, comienza la fase de parsing. Se llama a la función CheckFullExpression(), que luego llama a CheckExpression(), para luego empezar a revisar los tokens, de la forma antes comentada. Para esto se ha creado una clase Expression, de la que heredan multitud de clases que se asocian a los distintos tokens. Una vez se identifica el tipo de un token, se chequean las posibles expresiones generales que se pueden formar con él. Por ejemplo, cuando se identifica un número, se revisa que sea parte o no de una expresión binaria más compleja, como una suma. En caso de que así sea, se revisan las posibles operaciones que implican al otro sumando. De esta forma el árbol quedaría como una expresión suma, que contiene un operador(+) y dos nodos que son expresiones número. La clase contiene la lista errorList, que cuando se detecta un error sintáctico(por ejemplo: una expresión sin sentido como "2 + !;"), se añade a dicha lista un string con una descripción del error, para ser mostrada al usuario al final(de pasar esto, el código no se evalúa).

5. Analizador Semántico

Es aquí cuando el árbol sintáctico está listo y no contiene errores de sintaxis, que se procede a evaluar el código. Para ello se emplea la clase Evaluator, que tiene como método principal EvaluateExpression(). Este método se

encarga de verificar de qué tipo es la expresión retornada por el análisis sintáctico, y posteriormente todos los demás métodos van *avanzando por las ramas del árbol sintáctico*, que es una expresión de expresiones. Al igual que con el parser, aquí se dispone de una lista de errores errorList, que ejecuta la misma función, en caso de que se encuentre con un error semántico, como podría ser que al sumar dos variables que resulten ser un bool y un número.

Ejecutando Hulk

6. Ejecución

-Este proyecto está desarrollado para la versión objetivo de .NET Core 6.0. Para ejecutarlo debe ir a la ruta en la que está ubicada el proyecto y ejecutar en la terminal de Linux:

“‘bash
make dev “‘

- Si está en Windows, debe poder hacer lo mismo desde la terminal del WSL (Windows Subsystem for Linux), en caso contrario puede ejecutar:

“‘bash
dotnet watch run –project HULK “‘