Lenguaje Hack en Español

Instituto Tecnológico de Costa Rica



Escuela de Ingeniería en Computación Compiladores e Intérpretes - II Semestre 2016 Sede Cartago Jose Antonio Ruiz Jara - 2014096391 Ricardo Andrey Sánchez Delgado - 2014010852 Profesor: Esteban Arias Méndez Miércoles 30 de noviembre de 2016

Abstract—Since most programming languages are written in English, as part of a project, we decide to translate some Hack language instructions to Spanish, looking for an easier and different experience to those beginners in programming with low English level or without English language skills. As result of the project, we build a mini-interpreter for Hack language, with instructions translated to the Spanish language. This interpreter is capable of executing basic flow programs from a source code file.

Keywords—hack, interprete, java, lexico, sintáctico, español.

I. Introducción

El actual proyecto consiste en la aplicación de las bases aprendidas a lo largo del curso, acerca del funcionamiento y comportamiento de los compiladores y los intérpretes. a pesar de que ambos están diseñados con el mismo fin, cada uno de ellos realiza sus tareas de una forma diferente a la que la realiza otro. Esa diferencia difícilmente se hace notar por el usuario, porque la tarea de ambos es poder transformar instrucciones de alto nivel a un lenguaje que la máquina pueda ejecutar.

El proyecto está planteado de forma tal que pueden realizarse un scanner y un parser funcional que sea capaz de ejecutar instrucciones y tener manejo de errores, Tomando como base un lenguaje de programación existente para modificar su gramática y traducir sus instrucciones al español.



Fig. 1 Logo del lenguaje de programación Hack

Para este caso específico se utilizará como base el lenguaje de programación Hack, cuyo logo se aprecia en Fig. 1, el cual será interpretado mediante herramientas tipo Lex&Yacc (las cuales se mencionarán más adelante). Se decidió utilizar este lenguaje por la similitud que tiene con el lenguaje PHP en su sintaxis, y porque sería interesante observar cómo sería un lenguaje orientado al desarrollo web principalmente, con sus instrucciones escritas en español.

II. DESCRIPCIÓN DEL LENGUAJE

Hack es un lenguaje de código abierto bajo licencia BSD, desarrollado por Facebook. Es un lenguaje multiplataforma y es utilizado como una alternativa al lenguaje de programación PHP. Uno de sus principales usos es su implementación en la máquina virtual HipHop(HHVM) y gran parte del código de facebook está escrito en Hack.

La única gran modificación que se le realizó al lenguaje, en la realización de este proyecto, es la traducción de sus instrucciones al español, pues características como tipado dinámico y que es un lenguaje interpretado, se han tratado de mantener en la medida de lo posible.

III. INFORMACIÓN TÉCNICA DEL LENGUAJE

El lenguaje Hack en su implementación original es un lenguaje interpretado, de tipado dinámico y estático. En la implementación realizada se mantiene que el lenguaje sea interpretado y que tenga tipado dinámico.

El alfabeto utilizado en nuestra implementación de hack es la siguiente: $[0-9a-zA-Z><_?():,\$=./+-*]$

Los caracteres del alfabeto utilizados como delimitadores son los siguientes: "," para separar valores como parámetros de funciones ";" para indicar el fin de una instrucción

A. Palabras reservadas

"<?hh" palabra para identificar el inicio de un programa Hack, debe estar al inicio de el código fuente

"funcion"- palabra para indicar el inicio de una funcion

"sinretornar"-para indicar que una funcion no retorna datos

"como"-palabra equivalente a 'as' en otros lenguajes

"booleano"-utilizado para indicar valores verdaderos o falsos

"flotante"-utilizado para indicar valores de punto flotante

"entero" -utilizado para indicar valores de numeros enteros

"numero" - para indicar valores que puden ser flotantes o enteros

"recurso"- palabra para indicar que se necesita un recurso externo

"cadena"-utilizado para indicar valores string

"este"-utilizado en clases, equivalente a 'this'

"nada"-equivalente a 'void' para indicar que no tiene valor almacenado

"arreglo"-para indicar el tipo de datos arreglo o lista

"pordefecto"-se utiliza cuando se va a establecer un valor predeterminado

"caso"-se utiliza en los switch("cambiar") para denotar las diferentes opciones que se pueden ejecutar

"y"-operador booleano equivalente a 'and'

"o"-operador booleano equivalente a 'or'

"verdadero"-valor booleano para denotar que un estado correcto/verdadero

"falso"-valor booleano para denotar que un estado incorrecto/falso

"si"- instrucción 'if' para ejecutar una sentencia cuando el valor condicional es verdadero

"sinosi"-instrucción 'elseif' para evaluar otro valor cuando un if u otro elseif no se cumple

"sino"- instrucción 'else' para ejecutar cuando no se cumple el valor condicional del if al que pertenece

"mientras"-equivalente a un 'while', se ejecuta mientras el valor condicional dado sea verdadero

"para"- equivalente a un 'for' donde se ejecuta un grupo de instrucciones hasta que el valor condicional sea falso

"continuar" - instrucción para salir de un ciclo y continuar con la siguiente ejecución del mismo

"detener"- equivalente a un 'break' para detener y salir de un ciclo

"retornar"- retorna un valor desde una función

"cambiar"- realiza la misma tarea que un 'switch' típico

"hacer" - se utiliza antes de un while para asegurarse que esa parte del código se ejecuta al menos una vez

"imprimir"- imprime el valor de una expresión

IV. HERRAMIENTAS UTILIZADAS

Las herramientas utilizadas para la realización del proyecto son las siguientes:

- -Java como lenguaje de programación
- -JFLEX como herramienta tipo lex
- -CUP como herramienta tipo Yacc
- -Eclipse Neon como IDE
- -GIT como herramienta de control de versiones
- -Ubuntu 16.04 como sistema operativo

V. COMENTARIOS

A nivel de programas que el interprete ejecuta, se debe establecer un tipo de retorno para las funciones, pero ese tipo no es validado.

Se modificó la típica estructura del switch la cual básicamente cada uno de sus case tiene 2 puntos como delimitador para indicar las sentencias que en caso de que se cumpla la condición se deben de ejecutar, esto se cambió por llaves: ""

Se cambiaron los operadores binarios ||y&&| por 'o' y 'y'.

VI. OBSERVACIONES

Dependiendo del lenguaje que se va a implementar, se debe tomar muy en cuenta el tipo de uso que se la va a dar y la forma de como se espera que trabaje, para elegir si para su ejecución debe ser compilado o interpretado.

Al tener gramáticas tan extensas puede llegar a ser un poco complejo trabajar con ellas

Se debe tratar de filtrar lo más que se pueda en el analizador léxico, todo lo que se envía al analizador sintáctico, pues de esa forma se pueden evitar problemas con algunas palabras reservadas y valores ingresados en el código que no necesariamente son palabras reservadas pero contienen caracteres en el mismo orden que estas, lo que podría llevar a que el analizador sintáctico lo interprete de manera incorrecta.

VII. TRABAJO FUTURO

Como parte del trabajo pendiente para futuras modificaciones, estaría mejorar los mensajes de errores sintácticos y semánticos, pues actualmente se muestra el tipo de error encontrado pero no se muestra la linea y columna en donde el error se encuentra. Además podría mejorarse o implementarse la recursividad, aunque el return funcione a nivel de función, la recursividad no fue incluida dentro de la gramática. Tambien sería importante definir el alcance para las variables, pues como se ha mencionado anteriormente todas las variables son globales una vez declaradas.

VIII. Conclusión

Como resultado del proyecto se obtuvo un interprete capaz de ejecutar instrucciones de control básicas como for,while,if,else,switch,print,break,continue,case, todos en su equivalente en español. También es capaz de declaración de variables de tipo entero, booleano y string, y de las operaciones operaciones básicas de cada uno de estos tipos. Las

declaraciones y llamadas a funciones son una parte importante implementada en el interprete, ambas con sus respectivas declaracion de parámetros.

REFERENCES

[1] Louden, K. C (2004), Construcción de compiladores: principios y práctica, Thomson.