PrimerPaso

Daniel Berrocal Murillo, 2013114667
Josué Salas Barrantes, 2013114529
Jeferson Gomez Espinoza, 2013022493

(Dated: 16 June 2015)

The next document is going to be about compilers and interpreters, basically about a new project, that is called "PrimerPaso" which is oriented to children to stimulate them to get interested about what programming is about and how powerful it is. Which is based on the popular programming language called Java.

I. ¿QUÉ ES EL LENGUAJE PRIMERPASO?

Primer paso, es un lenguaje que está principalmente orientado para la población de niños de Alajuela, con el cual se espera poder tener un mayor interés a lo que es el estudio de las ciencias de la computación, la programación, es decir, se interesen por estudiar algo relacionado con la tecnología y la programación de diferentes aplicaciones. Este lenguaje, permitirá realizar programas básicos de computación, principalmente desarrollados por niños, esto les permitirá tener una perspectiva diferente de la lógica y la manera en que están siendo desarrolladas las aplicaciones, esto les podrá dar una motivación extra que los oriente a seguir por la guía del desarrollo, o lo que son las ciencias de la computación. PrimerPaso es un lenguaje que será escrito en Groovy, que es un lenguaje de alto nivel, que es opcionalmente tipado, y está orientado a la plataforma de Java. Este nuevo lenguaje que se desarrollará tiene su sintaxis completamente en español.

II. ¿CUÁLES SON SUS CARACTERÍSTICAS RINCIPALES?

Este lenguaje de programación, consta de algunas características que hacen resaltar sobre los demás, como por ejemplo:

1. Lenguaje en español: PrimerPaso está completamente escrito en español, esto debido a que está orientado a los niños de Alajuela, y el idioma que se habla es el Español, por esa razón e decidió hacerlo en este idioma, es decir, para facilitar la comprensión de los términos.

2. Permite realizar programas sencillos, con funcionales simples, esto para evitar generar controversia con algunos términos o agregarle complejidad al mismo, de manera que los niños no lo comprendan, que esa es lo que se desea evitar al 100%.

3. Cuenta con al menos 30 diferentes instrucciones, que son implementadas en el programa, y pueden ser utilizadas para el desarrollo de aplicación.

III. INFORMACIÓN TÉCNICA DE PRIMERPASO

A. Palabras reservadas

Especificación de cada palabra reservada:

- **por mientras (condicional){}** : se ejecuta lo que esté dentro del bloque de código que está dentro de él. Es la misma idea de un while en Java.
- **para cada (valor entero, condicional, índice){}** : Ciclo que se ejecuta mientras se cumpla la condició.
- **hacer { } mientras (condición):** Es igual que un mientras pero se asegura que entre la primera vez al ciclo aunque no se cumpla la condición.
- **retorna(valor):** Esta palabra reservada lo que realiza es devolver un tipo de dato (ya sea, hilera, entero, letra).
- **si(condicional){}**: Es una condición que si se cumple se realiza el bloque de código que está dentro de él, es decir, dentro de los "{}".
- **sino(condicional){}**: Es una condición que puede o no ir después del condicional "si" lo que realiza es valorar el condicional, y si se cumple se realiza el bloque de código dentro de él.
- **sino(condicional){}**: Es una condición que puede o no ir después del condicional "si" lo que realiza es valorar el condicional, y si se cumple se realiza el bloque de código dentro de él.
- **entonces{}**: Entra cuando no se cumple ninguna condición anterior (si, sino).
- **imprimir(valor o variable o hilera)**: imprime en consola lo que está dentro de los paréntesis.
- **leer:** Lee la entrada del usuario por medio del teclado
- **nada:** Valor de retorno igual a nada, es decir, no retorna ningún tipo de dato. Ejemplo: **nada** imprimeNombre(hilera nombre){
  imprimir(nombre);}

- **principal:** función principal, que es la que se va a ejecutar cuando se corra el programa.
- **salir:** Es para poder salir de un ciclo aunque se cumpla la condición.
- **hilera.largo():** Se le aplica a una variable de tipo hilera y devuelve como resultado el largo de la variable.
- **arreglo.cantidad():** Se le aplica a un arreglo y devuelve como resultado la cantidad de elementos que posee el arreglo.
- **hilera(numero):** Se le aplica a una variable de tipo entero y devuelve como resultado la misma variable pero de tipo hilera.
- **numero(hilera):** Se le aplica a una variable de tipo hilera y devuelve como resultado la misma variable pero de tipo entero.
- **hilera.mayuscula():** Devuelve una hilera igual a la original pero con todos los caracteres en mayúsculas.
- **minimo(entero1, entero2):** Devuelve el menor de los argumentos, deben ser de tipo entero.
- **maximo(entero1, entero2):** Devuelve el mayor de los argumentos, deben ser de tipo entero.
- **hilera.caracterEn(entero posición):** Devuelve el carácter que se encuentra en la posición "posicion" de la hilera. potencia(entero base, entero exponente): Devuelve el valor del primer argumento elevado a la potencia del segundo argumento, los argumentos deben ser de tipo entero.
- **arreglo.ordenarAsc():** Devuelve un arreglo ordenado ascendentemente
- **arreglo.ordenarDesc():** Devuelve un arreglo ordenado descendentemente
- **arreglo.mayor():** Devuelve el entero mayor del arreglo.
- **arreglo.menor():** Devuelve el entero menor del arreglo. arreglo.dividir(entero inicio, entero final): Devuelve un arreglo con los enteros que se encuentran entre las posiciones dadas.
- **hilera.cambiar(entero posicion, letra nueva):**Cambia la letra de la posicion dada por la nueva letra.
- **hilera.remplazar(letra vieja, letra nueva):** Reemplaza todas las letras de una hilera por otra letra dada.
- **hilera.contiene(hilera subhilera):** Retorna verdadero si una hilera contiene una subhilera dada o de lo contrario retorna falso.
- **hilera.esVacia():** Retorna verdadero si una hilera es de largo 0 de lo contrario retorna falso.
- **compararHileras(hilera hilera1, hilera hilera2):** Compara dos hileras y retorna verdadero si son iguales o falso en caso de no que no lo sean.
- **compararLetras(letra letra1, letra letra2):**Compara dos letras y retorna verdadero si son iguales o falso en caso de no que no lo sean.

B. Tipos de datos

- **entero :** Es un valor numérico, que va de -1024 a 1024, sólo números positivos, esto porque es orientado a niños de escuela y no hay necesidad de manejar números tan grandes.
- **hilera:** Son palabras que se encuentran entre comillas dobles. Ejemplo: "Esto es un string".
- **bool:** Puede ser verdadero o falso.
- **verdadero:** valor de verdad, es decir, le da un valor de verdad a una variable de tipo bool.
- **falso:**valor de verdad, es decir, le da un valor de falso a una variable de tipo bool.
- **arreglo:**Es una colección finita de elementos. Podrá ser de tipo entero. Se define de la siguiente manera: entero[] = 1,2,3,4,5,6,7

C. Notas importantes

- Al final de cada línea dentro del bloque de código debe ir un punto y coma (";").
- Al definir una variable se debe agregar el punto y coma al final (";"). Ejm:
  entero variable = 4;

D. Herramientas utlizadas

Para el desarrollo de la aplicación se utlizaron diferentes herramientas, las cuales mencionamos a continucación:

1. ANTLR4

Sus siglas en inglés son: ANother Tool for Language Recognition, las cuales vienen a traducirse como, otra herramienta para el reconocimiento del lenguaje, es un "parser" generador para la lectura, procesamiento, ejecución, o la traducción de texto estructurado o archivos binario. Es muy utilizada para construir lenguajes, herramientas y frameworks. ANTLR4 es capaz de generar un analizador que puede construir y recorrer árboles de análisis, solamente con una gramática. La versión que se utilizará será la versión 4.5, que está disponible en su página web: (`http://www.antlr.org/download.html`). ANTLR4 nos facilitó el desarrollo del compilador, ya que contiene una gran referencia bibliográfica que fue creada por el desarrollador del mismo. La cuál explica el proceso de cómo desarrollar un compilador poco a poco.

2. Groovy

El lenguaje de programación que se utilizó para el desarrollo del compilador será Groovy, que es un lenguaje dinámico, puede ser tipado o no, es un lenguaje con una sintaxis concisa, familiar y fácil de aprender. Una característica importante por la cual nos ayudó a elegirlo, es que se integra fácilmente con cualquier programa de java. La versión que utilizaremos será la versión más reciente, que es la versión 2.4 de Groovy, la cual se encuentra disponible en (http://groovy-lang.org/download.html).

IV. REFERENCIAS BIBLIOGRÁFICAS

louden Louden,K, *Construcción de Compiladores: Principios y Práctica*, University of Illinois. The LLVM Compiler Infrastructure. Disponible en: `http://llvm.org/`Apache2, *A multi-faceted language for the Java platform.*, Disponible en: `http://groovy-lang.org/` Parr, T. *ANTLR*, Disponible en: `http://www.antlr.org/index.html` Oracle *Java SE at a Glance*,Disponible en: `http://www.oracle.com/technetwork/java/javase/overview/index.html` Wikipedia, La enciclopedia libre. *Lenguaje de programación.*, Disponible en: `http://es.wikipedia.org/w/index.php?title=Lenguaje_de_programaci%C3%B3n&oldid=81098364` Wikipedia, La enciclopedia libre. *Notación de Backus-Naur*, University of Illinois. The LLVM Compiler Infrastructure. Louden,K, *Construcción de Compiladores: Principios y Práctica*, Disponible en: `http://es.wikipedia.org/w/index.php?title=Notaci%C3%B3n_de_Backus-Naur&oldid=79075064`

V. COMENTARIOS Y OBSERVACIONES

El desarrollo de este compilador, ha sido un proceso muy tedioso, y un poco complicado, como recomendación es de suma importancia que pasen revisando constantemente la documentación de la herramienta utilizada (ANTLR4), el link donde se encuentra esta herramienta, está disponible en la bibliografía.

El generador del lexer nos permitió analizar si estaba recibiendo los símbolos del lenguaje que luego serían analizados para proceder a ser "parseados" por el parser. Que generaría el árbol que luego sería recorrido para la generación del código.

Una de las principales características que debe tener todo lenguaje antes de comenzar a crear el lexer, parser y todo el proceso de la generación del código, es que la gramática debe de estar bien definida, no debe haber ambigüedad, no se deben generar ciclos, como se dijo anteriormente se resume en tener una gramática bien desarrollada y definida.

VI. TRABAJO FUTURO

La idea del compilador a futuro, es poder agregarle más funcionalidad que permitan realizar programas más complejos y robustos, en este momento, solo posee 30 instrucciones definidas dentro del lenguaje, que están siendo implementadas, aún no están terminadas. Se espera completarlo al 100% el compilador y utilizarlo para cumplir la meta principal., que era mostrarle el lenguaje a los niños y enseñarles como realizar programas basados en PrimerPaso.