COMPILADOR JYCEL en Java para Lenguaje Natural en Español (JYCELPROES)

COMPILER JYCEL Java for Natural Language Spanish (JYCELPROES)

Julián E. Jiménez Acuña
Fundación Universitaria Juan de Castellanos
Tunja (Colombia)
(57) 320 940 3776
julyman_jiac@hotmail.com

Leidy K. Cely Carreño
Fundación Universitaria Juan de Castellanos
Tunja (Colombia)
(57) 311 206 0640
kathycely99@hotmail.com

Leonardo Bernal Zamora [1]
Iván A. Delgado González [2]
Fundación Universitaria Juan de Castellanos
Tunja (Colombia)

ABSTRACT

En este artículo se pretende mostrar el desarrollo de un nuevo lenguaje de programación JYCEL (acrónimo de los apellidos de los autores) con sintaxis en español, teniendo como base los lenguajes naturales y las estructuras de sentencias de alto nivel (Java), al igual que el desarrollo de su compilador en java, este se compone de cuatro fases fundamentales: analizador léxico, analizador sintáctico desarrollados con ayuda de herramientas como JFlex y Cup respectivamente, analizador semántico para el manejo de variables y finalmente el generador de código el cual se basa en la traducción del nuevo lenguaje de programación a Java para su posterior compilación.

This article aims to show the development of a new language JYCEL programming (acronym for the surnames of the authors) syntax in Spanish, based on natural languages and structures of high level statements (Java), as developing your java compiler, this consists of four main phases: lexical analyzer, parser developed using tools like JFlex and Cup respectively, semantic analyzer for handling variables and finally the code generator which is based on translating the new programming language Java for later compilation.

CCS Concepts

• Software and its engineering---Software notations and tools---Language features---Control structures
• Software and its engineering---Software notations and tools---Compilers---Parsers

Keywords

Analizador Léxico; Analizador Semántico; Analizador Sintáctico; Compilador; Generador de Código; Java; Lenguaje Natural.

Code Generator; Compiler; Java; Lexical Analyzer; Natural Language; Parser; Semantic Analyzer.

Conference'10, Month 1–2, 2010, City, State, Country.

DOI: http://dx.doi.org/10.1145/12345.67890

1. INTRODUCCIÓN

Los primeros programadores crearon varios lenguajes de programación de forma que su sintaxis pudiera ser interpretada por una computadora, esto se logró gracias a la construcción de compiladores. El desarrollo de estos compiladores, abrió un nuevo campo hacia los lenguajes de programación, el uso e implementación de estos lenguajes de bajo nivel permitió un resurgimiento a nuevos lenguajes de medio y alto nivel.

El desarrollo del software se ha venido fortaleciendo, por lo que un estudiante que quiere comenzar sus estudios en Ingeniería de Sistemas y carreras afines, busca instituciones que lo puedan fortalecer profesionalmente. En el transcurso de su aprendizaje adopta habilidades de programación de forma intuitiva gracias a técnicas y herramientas que complementan su desarrollo analítico por medio de la práctica y la investigación, lo cual va mejorando de desempeño a medida que las aplica cotidianamente.

2. FUNDAMENTO TEÓRICO

2.1 Lenguaje natural

Se considera como lenguaje natural todo aquel que se asemejan al que usamos cotidianamente, una de las características principales es que surge de manera espontánea entre las personas. Se puede decir que, es el que está diseñado para poder realizar una interacción entre humanos y maquinas fácilmente.

2.2 Pseudocódigo

Es una forma de poder definir que se quiere realizar en código fuente, para esto se emplea el lenguaje natural con un poco de lenguaje de programación.

2.3 Lenguaje de programación

Es aquel lenguaje que permite indicar una secuencia de instrucciones para posteriormente ser procesadas por un ordenador.

---

[1] Leonardo Bernal Zamora; Fundación Universitaria Juan de Castellanos; Tunja (Colombia); (57) 316 428 9699; lbernalz@jdc.edu.co
[2] Iván Andrés Delgado González; Fundación Universitaria Juan de Castellanos, Tunja (Colombia); (57) 312 481 6150; idelgado@jdc.edu.co

2.4 Compilador

Es un tipo de traductor el cual requiere como entrada un lenguaje formal, para obtener finamente como salida un ejecutable. Las fases de desarrollo de un compilador se presentan en la figura 1.

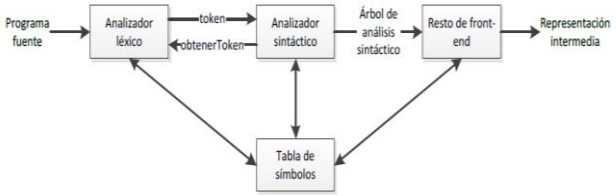

Figura 1. Fases de desarrollo de un compilador

2.4.1 Analizador léxico

Un analizador léxico o de escaneo, se encarga de leer el flujo de caracteres de un programa fuente para agruparlos en secuencias significativas (lexemas), y de esta forma generar una secuencia de tokens para cada lexema del programa fuente.

2.4.2 Analizador sintáctico

El parser o analizador sintáctico hace uso de los tokens generados por el analizador léxico, para crear un árbol de derivación que describa la estructura gramatical de los tokens.

2.4.3 Analizador semántico

El análisis semántico dota de un significado coherente a lo que hemos hecho en el análisis sintáctico. El chequeo semántico se encarga de que los tipos que intervienen en las expresiones sean compatibles o que los parámetros reales de una función sean coherentes con los parámetros formales.

2.5 Metacompilador

Es un compilador de compiladores, tiene como función principal aceptar como entrada un lenguaje de programación y producir un compilador de este lenguaje.

2.5.1 JFlex

Se define como un generador de analizadores lexicográficos, el cual está desarrollado en java y genera código java. JFlex hace uso de un archivo plano el cual debe contar con un formato específico para ser compilado posteriormente.

2.5.2 CUP

La función de CUP es la de generar analizadores sintácticos en java, es muy similar al uso de JFlex ya que requiere un archivo plano, el cual debe tener la siguiente estructura:

- Descripciones de paquete e Importaciones: En esta parte del archivo se indica el paquete al que pertenece el archivo y se importan las clases necesarias.
- Código de usuario: Permite introducir código java el cual aparecerá tal cual, en el archivo generado.
- Lista de símbolos: En esta sección del archivo se indican los terminales y no terminales de la gramática.
- Precedencia: Se indicara la precedencia de los terminales solo si es necesario, con el fin de evitar la ambigüedad.
- Gramática: Enunciación de la gramática.

3. METODOLOGÍA

El desarrollo del compilador JYCEL y el lenguaje de programación JYCELPROES (acrónimo de los apellidos de los autores y programación en español) surgieron con la idea de crear una herramienta de ayuda para la enseñanza de programación básica en los primeros semestres de Ingeniería de Sistemas, siendo el lenguaje totalmente en español, con uso de estructuras de control básicas y como funcionalidad adicional, permite visualizar la traducción de programa fuente al lenguaje de programación Java.

3.1 Lenguaje programación JYCELPROES

Como ya se mencionó anteriormente JYCELPROES es un lenguaje de programación en español, pero adicionalmente cuenta con las siguientes características:

- Uso de estructuras de control básicas.
- Uso de palabras simples para comenzar y terminar los programas (INICIAR - FINALIZAR).
- Diferencia las palabras reservadas o propias del lenguaje en mayúsculas del resto de la estructura del código fuente, que debe ser escrito en minúsculas.
- Usa una estructura sencilla para el desarrollo de programas (Figura 2).

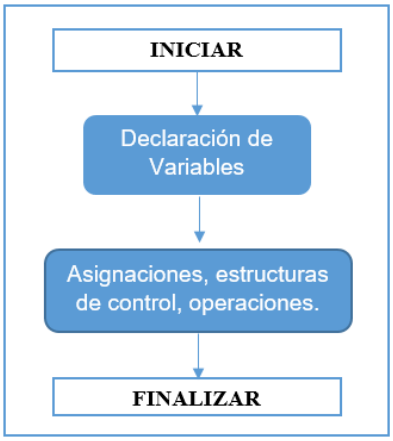

Figura 2. Estructura de un programa en lenguaje JYCELPROES

3.1.1 Declaración del alfabeto

El alfabeto de JYCELPREOES fue declarado teniendo en cuenta que sería un lenguaje en español y además contaría con algunas estructuras de un lenguaje de programación convencional como se describe a continuación:

- Letras: Contiene todas las letras del alfabeto entre minúsculas y mayúsculas.
- Dígitos: Conformados por números enteros desde un mínimo de -2.147.483.648 y un máximo 2.147.483.647
- Símbolos: Son permitidos los siguientes símbolos + - * / , : ( ) = < > # % ! –. Teniendo en cuenta estos ítems, el alfabeto sigma del JYCELPROES está conformado así:
- ∑= { a,…,z, A,…,Z, -2.147.483.648,…, 2.147.483.647, +, -, *, /, ,, :, (,), =, <, >, #, %, !, - }

3.1.2 Diagramas de transición

Es una gráfica que muestra las acciones que sigue un analizador lexicográfico al momento que encuentra un token, estos se ilustran con flechas etiquetadas con los caracteres que va asociar y que

conectan a círculos que se llaman estados, que denotan la transición de un estado a otro por medio de una flecha etiquetada. Una vez finalizadas las transiciones, es decir, al llegar a un estado de aceptación se representa con un círculo dentro de otro y éste devuelve el valor del token como lo muestra la Figura 3.

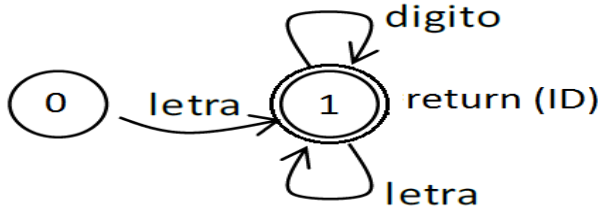

Figura 3. Diagrama de transición ID

3.1.3 Gramática

La gramática del lenguaje JYCELPROES esa conformada por terminales, no-terminales, producciones y un símbolo inicial de la gramática, el cual como su nombre lo indica será el encargado de dar inicio, lo que es fundamental para el análisis sintáctico del compilador JYCEL.

La gramática de libre contexto es expresada en notación de Backus-Naur (BNF), ya que otorgan variedad de ventajas significativas para un analizador sintáctico en la fase de desarrollo de un compilador.

Esto se podrá entender mejor, teniendo en cuenta que los terminales compondrán la parte derecha de una gramática, y los no-terminales serán la parte izquierda como se muestra en la siguiente Figura 4.

```
iniciar ::= INICIAR x FINALIZAR;
x ::= d b;
b ::= i b
    | i;
i ::= asig
    |mientras
    |si
    |para
    |mostrar;
asig ::= ID ASIG p COMA;
mientras ::= MIENTRAS ABRE_P p CIERRA_P PUNTOS b FINM;
si ::= SI ABRE_P p CIERRA_P PUNTOS b ENTONCES b FINS
    |SI  ABRE_P p CIERRA_P PUNTOS b FINS;
para ::= PARA  ABRE_P asig p COMA s CIERRA_P PUNTOS b FINP;
mostrar ::= MOSTRAR ABRE_P n CIERRA_P COMA;
```

Figura 4. Ejemplo Gramática

Por lo tanto, se describirá el conjunto de terminales, no-terminales, producciones y símbolo inicial de la gramática que ayudará a comprender mejor la estructura del lenguaje y su énfasis sintáctico, así:

3.1.3.1 Conjunto de terminales

La gramática se conforma por los siguientes terminales, escritos en mayúscula, Figura 5.

ESPECIAL	**PUNTOS**	**DIF**
VARIABLE	**ASIG**	**MAI**
MIENTRAS	**ABRE_P**	**MEI**
FINM	**CIERRA_P**	**MENOR**
SI	**COMA**	**MAYOR**
ENTONCES	**IGUAL**	**NOT**
FINS	**SUMA**	**AND**
PARA	**MULT**	**OR**
FINP	**RESTA**	**INICIAR**
MOSTRAR	**MOD**	**FINALIZAR**
ABS	**DIV**	

Figura 5. Conjunto de Terminales

3.1.3.2 Conjunto de no-terminales

Estos serán escritos en minúscula, conforman la parte izquierda de la gramática (Figura 6).

iniciar	**mientras**	**m**
x	**para**	**p**
b	**mostrar**	**f**
i	**d**	**n**
asig	**s**	
si	**l**	

Figura 6. Conjunto de No-Terminales

3.1.3.3 Producciones

Las producciones son declaradas en notación BNF, estas producciones nos ayudan a evaluar el orden de la gramática, es decir, la organización de cada una de las producciones principalmente las estructuras de control formando así un árbol de derivación.

3.2 Compilador JYCEL

La función del compilador JYCEL es procesar el lenguaje de programación JYCELPROES, de tal manera que reporte errores del lenguaje si se presentan y luego compilar el código fuente.

El compilador JYCEL fue desarrollado teniendo en cuenta cuatro fases principales de todo compilador; análisis lexicográfico, análisis sintáctico, análisis semántico y generador de código. Para el desarrollo de algunas de las fases se utilizaron herramientas como los metacompiladores (son una gran ventaja a la hora de desarrollar un compilador, ya que son generadores de analizadores a partir de ciertas reglas declaradas en archivos de configuración).

3.2.1 Analizador léxico

El analizador léxico fue desarrollado con ayuda del metacompilador JFlex ya que cuenta con una gran ventaja, tiene la capacidad de crear un archivo .Java a partir de las reglas léxicas del lenguaje JYCELPROES (Figura 7).

Figura 7. Funcionamiento de JFlex

El archivo .flex que contiene las reglas léxicas deben cumplir con un formato específico según (Fuentes, 2009, 7), allí se resume en 3 ítems: código usuario (permite introducir código java al archivo

que será generado), opciones y declaraciones (permite configurar el analizador léxico con ayuda de una serie de directivas especificadas por JFlex) y finalmente las reglas léxicas (permiten con ayuda de código Java, indicar las acciones que se ejecutaran cuando los símbolos analizados corresponden a alguna expresión regular y posteriormente generar los tokens).

Una vez generado el analizador léxico (.java) se procede a compilarlo y así realizar el análisis léxico al lenguaje JYCELPROES, como resultado del análisis léxico obtenemos los tokens del lenguaje (Figura 8).

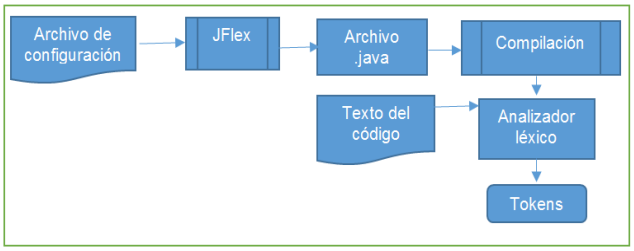

Figura 8. Operación de JFlex

3.2.2 Analizador sintáctico

El desarrollo del analizador sintáctico se llevó acabo con el metacompilador Cup e interactuando con JFlex, ya que el analizador sintáctico generado por Cup toma los tokens del analizador léxico y verifica su orden correspondiente con ayuda de un árbol sintáctico (Figura 9).

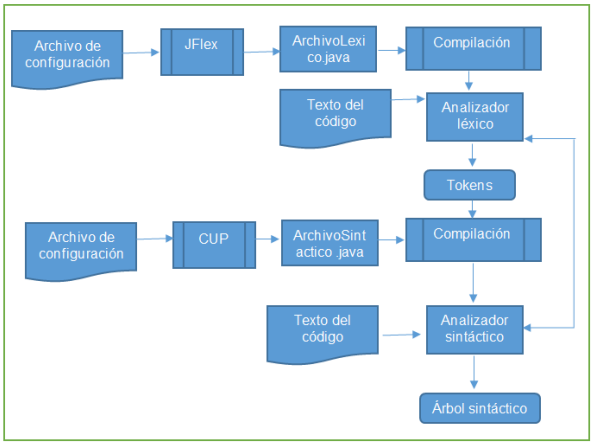

Figura 9. Integración JFlex y Cup

Cup al igual que JFlex requiere de un archivo plano (archivo de configuración) para generar el analizador léxico con extensión .java, este archivo también tiene un formato específico contenido por: descripciones del paquete e importaciones, código de usuario, lista de símbolos, precedencia (en el caso del lenguaje JYCELPROES no aplica, ya que el lenguaje no cuenta con ambigüedad) y la gramática. La sección donde se denota la gramática de JYCELPROES se realizó con base en la notación Backus-Naur, para iniciar la gramática se hace uso del terminal "iniciar" (Figura 10).

```
start with iniciar;
iniciar ::= INICIAR x FINALIZAR;
x ::= d b;
b ::= i b
```

Figura 10. Gramática BNF JYCELPROES

3.2.3 Analizador semántico

Como ya se mencionó anteriormente el analizador semántico se encarga de validar aspectos que el analizador sintáctico no está capacitado para hacer, algunos de estos, es el manejo de las variables y los valores que se les asignan; para ello se crearon varias tablas las cuales almacenan las variables declaradas y sus valores asignados en el lenguaje JYCELPROES.

3.2.3.1 Variables declaradas

Se recorrerá el árbol sintáctico con el fin de identificar las variables declaradas y junto con un método que devuelve el nombre de las variables, asegurar que no se repitan y posteriormente almacenarlas en una tabla.

3.2.3.2 Verificador de la tabla

Verifica y comparan las variables almacenadas en la tabla con las variables que se usan en el desarrollo del código, para esto es necesario recorrer por segunda vez el árbol sintáctico con el fin de reportar aquellas variables que no han sido declaradas y las que no han sido inicializadas para su uso en las estructuras de control. El analizador semántico se puede resumir en la Figura 11, donde se evidencia los dos procesos que recorren el árbol junto con su función.

```
Procedimiento creador tabla ()
    Inicio recorrido
        .........
        Tabla = variables
        ..........
Fin

Procedimiento verificador de tabla ()
   Inicio recorrido
        ..........
      Comparar tabla y variables existentes
        ............
Fin
```

Figura 11. Analizador semántico

3.2.4 Generador de código

La función del generador de código es traducir el lenguaje de programación JYCELPROES al lenguaje de programación que se desee en este caso, Java. Al igual que los analizadores léxico, sintáctico y semántico, requiere hacer un recorrido al código para traducir cada una de sus sentencias.

El resultado de esta traducción es almacenada en un archivo con extensión .java, para posteriormente ser ejecutado por el compilador de java el cual obtendrá el resultado que se especificó en el lenguaje de programación JYCELPROES y se enviara directamente al compilador JYCEL.

4. RESULTADOS Y DISCUSIÓN

4.1 Compilador JYCEL, análisis y pruebas

El compilador JYCEL hace uso de una interfaz sencilla y cómoda para el estudiante, lo que permitirá adaptarse fácilmente al entorno y programar de manera simple. A continuación se visualizará ésta interfaz, cómo el estudiante podrá interactuar con el compilador y hacer uso eficiente del lenguaje JYCELPROES.

La interfaz gráfica cuenta con un módulo de inserción del lenguaje JYCELPROES, tres módulos de visualización y un módulo de herramientas, visualizado en la figura 12.

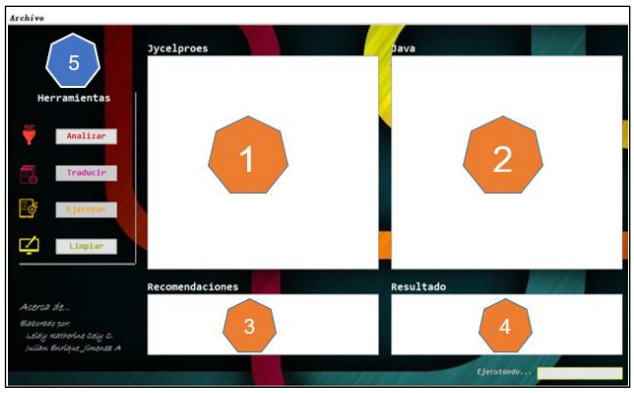

Figura 12. Interfaz Compilador JYCEL

(1) El módulo "JYCELPROES", permite digitar o cargar el código que se esté desarrollando en este lenguaje de programación.
(2) El campo Java es un módulo de visualización que mostrará la correspondiente traducción del lenguaje JYCELPROES a Java.
(3) El campo de recomendaciones es un módulo de visualización que muestra las recomendaciones de los resultados de los analizadores si son necesarios.
(4) El campo resultado es un módulo de visualización que muestra el resultado final de la compilación de JYCELPROES.
(5) El Modulo de herramientas consta de 5 botones que tienen las siguientes funcionalidades:
    a. El botón Analizar, analiza la sintaxis ingresada y revisa errores lexicográficos y sintácticos.
    b. El botón Traducir, realiza la traducción de la sintaxis ingresada sin errores de compilación.
    c. El botón Ejecutar, muestra el resultado de la sintaxis ingresada.
    d. El botón Limpiar, limpia los campos de la interfaz.

En el compilador JYCEL se realizaron las siguientes pruebas de compilación que se verán evidenciadas en las siguientes figuras 13 y 14, de tal forma que se demuestre el funcionamiento del compilador JYCEL haciendo uso del lenguaje de programación JYCELPROES y se muestre como corrige problemas que lleguen a presentarse.

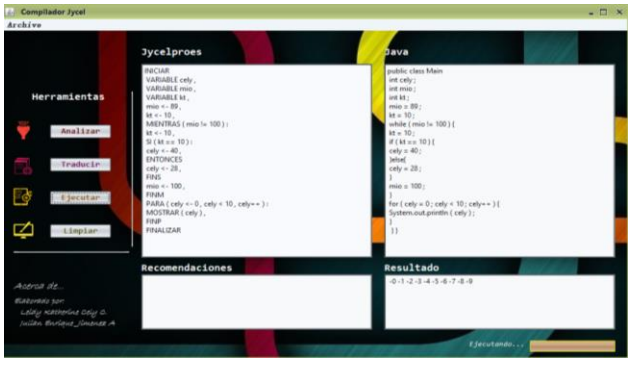

Figura 13. Compilador JYCEL

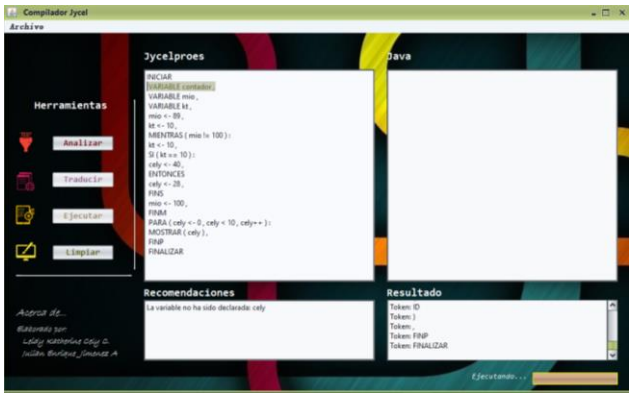

Figura 14. Error en el compilador JYCEL

Como se logra observar en las figuras anteriores, al hacer la debida inserción del código JYCELPROES, aseguramos el correcto funcionamiento del compilador, teniendo en cuenta las palabras simples para iniciar y finalizar un programa, así como el espacio que debe tener el inicio de cada sentencia.

En la figura 14 al hacer un cambio en la estructura del código donde no se inicializa una variable, en este caso la variable "cely", y hacer uso de ella en un ciclo PARA, muestra un mensaje de error en el módulo de visualización Recomendaciones, con la no declaración de la variable al iniciar el programa, posteriormente se hace la inhabilitación de los botones de herramientas, hasta no hacer la debida corrección y análisis del mismo.

5. CONCLUSIONES

- El análisis sobre los lenguajes naturales como se muestra en el estado del arte, permitió estructurar el lenguaje de programación denominado JYCELPROES, un lenguaje de programación en español que permite el manejo de operaciones aritméticas, lógicas y estructuras de control básicas.
- La construcción de las principales fases de que comprende un compilador (léxico, sintáctico y semántico), fueron desarrollados exitosamente mediante la implementación de metacompiladores como JFLEX y CUP, los cuales permitieron la generación de los analizadores léxicos y sintácticos y su posterior integración con el analizador semántico.
- Aplicar los respectivos fundamentos matemáticos ejercidos sobre la gramática garantizó un óptimo desarrollo del compilador, de forma que la transformación de la fase lexicográfica a la fase sintáctica, no ejerciera ningún error en las gramáticas como en sus diagramas de transición hasta en sus reglas de producción.
- Uno de los aportes más significativos que se presentan como resultado del proyecto, es la traducción del lenguaje de programación JYCELPROES a Java, lo que permitirá a los estudiantes realizar un proceso más ordenado y estructurado para desarrollar sus competencias en la lógica de programación, tanto en lenguajes de medio y alto nivel.
- La interfaz del compilador JYCEL se presenta de forma amigable y de fácil manejo con los campos respectivos de interacción, para que el estudiante pueda ejercer libremente su programación, contando con ayudas y recomendaciones que lo orientan de una manera más cómoda y sencilla.

6. REF ERENCES

[1] Aguilar, H. T. (2014). Apuntes Compiladores. Toluca

[2] Canales, Isaac Andrés y Ruiz, Michel. Desarrollo de un compilador para pseudocódigo en lenguaje español. Tesis de ingeniero en informática. México, D.F.: Instituto Politécnico Nacional. Unidad profesional interdisciplinaria de ingeniería y ciencias sociales y administrativas, 2011

[3] Celada, Luis y Gil, Carlos. Desarrollo de un compilador de una representación basada en reglas a código bytes de java. España: Madrid: Universidad Complutense de Madrid. Facultad de informática, 2008

[4] CCM. (Septiembre de 2015). DOI= http://es.ccm.net/contents/304-lenguajes-de-programacion

[5] Cueva Lovello, Juan Manuel. Conceptos básicos de procesadores de lenguaje. España: SERVITEC, 1998. ProQuest ebrary. Web. 4 October 2015

[6] Eumed. revisado 2015-octubre-02. DOI= http://www.eumed.net/tesis-doctorales/2014/jlcv/software.htm

[7] Galvez Rojas, S., & Mora Mata, M. A. (2005). Compiladores: Traductores y compiladores con Lex/Yacc, Jflex/Cup y JavaCC. Malaga, España: Universidad de Malaga

[8] Jiménez Millán, José Antonio. Compiladores y procesadores de lenguajes. España: Servicio de Publicaciones de la Universidad de Cádiz, 2009. ProQuest ebrary. Web. 4 October 2015

[9] Montero, P. R. (26 de Enero de 2015). Linux Hispano. DOI= http://www.linuxhispano.net/2015/01/26/latino-nuevo-lenguaje-de-programacion-con-sintaxis-en-espanol/

[10] Pressman, Roger S. Ingeniería del software: un enfoque práctico. Séptima edición. México DF: McGraw-Hill. 2010. Pg. 40

[11] Razo, J. O. (2014). Introducción a la Programación

[12] Sommerville, Ian. Ingeniería de software. Séptima edición. Pearson Editores. México. 2005

[13] Vásquez, A. C. (2009). Procesamiento de lenguaje natural. DOI= http://revistasinvestigacion.unmsm.edu.pe/index.php/sistem/article/view/5923

[14] Vega, Rafael Anibal. Compilador de pseudocódigo como herramienta para el aprendizaje en la construcción de algoritmos. Tesis de Ingeniero de Sistemas. Colombia, Barranquilla: Universidad del Norte. Programa de ingeniería de sistemas, 2008

[15] _____, The design and Analysis of Computer Algorithms, Addison-Wesley Publishing Company 1974, ISBN 0-201-00029-6