Revista Tecnocientífica URU
Universidad Rafael Urdaneta
Facultad de Ingeniería
No. 14 Enero - Junio 2018
Depósito Legal: PPI 201402ZU4464
ISSN: 2343-6360

Analizador sintáctico-semántico para la enseñanza de la programación de software

Alfredo Díaz[1] y Asdrúbal Granados[2]

[1]Ingeniero en Informática. Docente del Instituto Universitario Politécnico "Santiago Mariño", Extensión COL-Cabimas

[2]Ingeniero en Sistemas, Egresado del Instituto Universitario Politécnico "Santiago Mariño", Extensión COL-Cabimas.

Correo electrónico: asdrubalgranados@gmail.com y alfredojodp@hotmail.com

Recibido: 01-11-2016 Aceptado: 03-07-2018.

Resumen

La presente investigación, tiene como propósito, desarrollar un analizador sintáctico- semántico para la enseñanza de la programación de software para estudiantes de Ingeniería en Sistemas, con lo cual, se busca proveer una herramienta amistosa para el estudiante para que comprenda lo que ocurre en el proceso de compilación de un programa. La misma se sustenta en los aportes de Aho [1] Grune *et al*. [6] y Alfonseca *et al.* [2] .Se utilizó el entorno *NetBeans* y el lenguaje Java para su desarrollo. La investigación está sustentada en la modalidad documental de tipo tecnológica, tomando como unidad de análisis los softwares inter-p y pseint. Se obtuvo como resultado un analizador sintáctico-semántico con un preprocesador capaz de realizar la descomposición simbólica del código fuente. Se concluyó que se debe complementar la enseñanza de la programación con estrategias basadas en las TIC para que el estudiante comprenda los procesos que ocurren durante la compilación, con lo cual, se generará un aprendizaje más significativo.

Palabras Clave: Analizador léxico, sintáctico, semántico.

Syntactic-semantic analyzer for teaching software programming

Abstract

The purpose of this research is to develop a semantic-syntactic parser for teaching software programming for students of Systems Engineering, which is looking to provide a friendly tool for the student to understand what happens in the programs compiling process. It is based on contributions by Aho [1]Grune *et al.* [6] and Alfonseca *et al.* [2] . It was codified using the Java language and NetBeans environment. The research is classified as technological and documentary and the analysis units were the inter-p and pseint software. A syntactic-semantic analyzer was obtained and it is capable of performing the symbolic decomposition of the source code that is the previous step for the final compilation. It was concluded that it should complement the teaching strategies for the software-programming students to understand the processes that occur during compilation, in order to generate, a more meaningful learning.

Keywords: Semantic, syntactic, parser.

Introducción

La programación es una de las áreas fundamentales dentro de Ingeniería en Sistemas. De allí que las casas de estudio, han tratado de vincular dentro de sus diseños curriculares, asignaturas vinculadas al desarrollo de software con los distintos lenguajes de programación disponibles en el mercado. En ese sentido, los docentes, deben procurar el uso de herramientas y recursos que les permitan a los estudiantes, desarrollar sus capacidades en este ámbito, ya que le otorgará pertinencia al egresado dentro del campo laboral.

En ese sentido, de acuerdo a los postulados por Díaz y Serra [5] , aun cuando las instituciones de Educación Superior, ofertan carreras vinculadas a las TIC, la matrícula de estudiantes ha disminuido, ya que los mismos perciben con dificultad las tareas asociadas a la programación, lo cual, está relacionado

con las herramientas que complementan la enseñanza en el aula, ya que la documentación puede resultar extensa, ocasionando un desbalance entre la oferta y demanda de profesionales capacitados para el desarrollo de software en el país y esto contribuye al deterioro de la industria nacional del software.

De igual manera, las herramientas para la programación de software como Interfaces de Desarrollo Integrado (IDE) se componen básicamente de un compilador que se encarga de traducir el lenguaje formal de programación en un lenguaje máquina, sin evidenciar explícitamente lo que ocurre durante la conversión o traducción del programa fuente.

En ese orden de ideas, en el Instituto Universitario Politécnico "Santiago Mariño" extensión Costa Oriental, sede Cabimas, se pudo apreciar que una cantidad considerable de estudiantes de la carrera de ingeniería de sistemas se muestran reacios con el aprendizaje de la programación de software, posiblemente debido a que algunas de las herramientas que son proporcionadas en la institución tienen cierto grado de complejidad. Además de eso, también se percibió que presentan inconvenientes con el manejo del inglés, siendo este fundamental al momento de programar, ya que en la sintaxis de la mayoría de los lenguajes de programación modernos se utiliza este idioma, ocasionando desmotivación y falta de compromiso con el estudio por parte de los mismos.

Relacionado con el problema anterior, una descripción similar fue hecha por Díaz. y Serra [5] mediante un estudio realizado a 15 estudiantes de la asignatura Lenguaje de programación I, que pretendió determinar el nivel de competencias alcanzadas en la construcción de algoritmos, tomando como base un problema planteado, el diseño de diagramas de flujo a partir del algoritmo y su respectiva codificación en el lenguaje C++. En el mismo se concluyó que los estudiantes presentaron debilidades en los fundamentos básicos de programación, pudiéndose evidenciar problemas al momento de aplicar la sintaxis del lenguaje, estructurar las instrucciones de entrada/salida, utilizar los ciclos repetitivos y de decisión así como desconocimiento de procedimientos de cálculo.

Por consiguiente, resulta claro que de seguir presentándose la problemática descrita anteriormente, será muy complicado poder equilibrar la oferta de profesionales capacitados para el área de las TIC con la demanda del mercado laboral, lo cual ocasionará que exista un exceso de profesionales en áreas no relacionadas con las TIC, además que se retrasaría la modernización tecnológica del país y se correría el riesgo de entrar en obsolescencia tecnológica, ya que las herramientas y los conocimientos que hoy están a la vanguardia, mañana serán cosas simples porque la tecnología avanza demasiado rápido.

Para darle solución a esta problemática se desarrolló un analizador sintáctico-semántico sencillo, amistoso y que facilite la enseñanza de programación de software, la comprensión de lo que ocurre durante el proceso de compilación y la aplicación de los conocimientos aprendidos. Esta herramienta cuenta con los componentes básicos de un Entorno Integrado de Desarrollo, muestra de manera explicativa al usuario como encontrar los diferentes errores que puedan surgir al momento de escribir código, además de tener una interfaz gráfica que permite al estudiante poder aplicar estructuras de decisión, control y selección. También muestra cómo generalmente la computadora interpreta estas estructuras para generar una salida de código que resulte en un programa de computadora.

Se hizo énfasis que para el desarrollo de esta herramienta se ha diseñado un lenguaje de programación en español al cual se le pueden aplicar las técnicas para el desarrollo de un programa, el cual está basado en la sintaxis del lenguaje de programación C. El desarrollo del analizador léxico, sintáctico y semántico se comenzó primero con desarrollar la parte léxica que se encarga de cotejar los símbolos y las palabras en una tabla de datos que está dentro de una base de datos para facilitar su mantenimiento, para identificar palabras reservadas del lenguaje y comparar que lo que el estudiante escribe dentro del analizador corresponda al lenguaje creado para la herramienta.

Luego, se procedió a elaborar la parte sintáctica que verifica que existan palabras, números y expresiones bien formadas, además de dividir cada carácter o palabra en símbolos abstractos para compararlos con un árbol de decisión y verificar la sintaxis en general y su ubicación en las sentencias, y de esta manera comprobar la estructura y los bloques de código. Por último se elaboró la parte semántica que comprueba la estructura general del código, la lógica implementada en la mayoría de las sentencias,

la verificación de estructuras de decisión, secuencia, repetición y control y las oraciones escritas en el lenguaje de programación establecido y la funcionalidad del código en general.

El desarrollo del analizador también contiene una interfaz gráfica que permite al usuario interactuar de manera simple con la herramienta, además de visualizar los errores que puedan producirse y mostrar al usuario como localizarlos mediante ayudas visuales. Esta interfaz gráfica incluso permite al estudiante formar bloques de código de manera gráfica, es decir, se agregaron componentes que permiten establecer condiciones mediante parámetros a los cuales, luego de hacer clic, se hace la estructuración del bloque de código al cual se les aplica el análisis léxico, sintáctico y semántico.

A los efectos de dar cumplimiento a esta investigación, se planteó como objetivo general: Desarrollar un analizador sintáctico-semántico para la enseñanza de la programación de software, para lo cual, fueron formulados los siguientes objetivos específicos: a) Describir los elementos que conforman el análisis sintáctico y semántico, b) Diseñar los prototipos del analizador sintáctico-semántico para la enseñanza de la programación de software y c) Probar el funcionamiento del analizador sintáctico-semántico para la enseñanza de la programación de software

Fundamentos Teóricos

Analizador Sintáctico-Semántico

Aho [1] indica que es un programa informático cuyo propósito es traducir un programa escrito en un lenguaje de programación a otro de tipo intermedio, previa verificación de la escritura y construcción de expresiones, variables, palabras y números presentes en el código fuente. Una función importante es reportar cualquier error en el programa fuente que se detecte durante el proceso de traducción. Si el programa destino es un programa ejecutable en lenguaje máquina, entonces el usuario puede ejecutarlo para procesar las entradas y producir salidas. Por otro lado, Grune *et al.* [6] refieren que es un programa que toma como entrada un texto escrito en cierto lenguaje y produce como salida el texto de un programa en otro lenguaje, a la vez que preserva el significado del texto original.

En concordancia con lo planteado por los autores, un analizador sintáctico-semántico es un programa informático que recibe un conjunto de caracteres que pertenecen a un alfabeto o un lenguaje y que mediante reglas, patrones o estructuras, previamente definidas, interpreta una secuencia lógica de caracteres con el propósito de traducirlas a un lenguaje para realizar una tarea específica.

Analizador Léxico

Aho [1] indica que un analizador léxico, es un escáner cuyo propósito es leer el flujo de caracteres que componen el programa fuente y los agrupa en secuencias significativas, conocidas como lexemas. Para cada lexema, el analizador produce como salida un token de la forma nombre-valor. En el token, el primer componente es un símbolo abstracto que se utiliza durante el análisis sintáctico, y el segundo componente apunta a una entrada en la tabla de símbolos para el mismo.

En ese mismo orden de ideas, Alfonseca *et al.* [2] exponen que es un autómata finito determinista que reconoce el lenguaje generado por las expresiones regulares correspondientes a las unidades sintácticas del lenguaje fuente. Este se encarga de dividir el programa fuente en un conjunto de unidades sintácticas, las cuales son una secuencia de caracteres con cohesión lógica. Con base en lo expresado por los autores, un analizador léxico tiene como función principal leer los caracteres de la entrada del programa fuente, agruparlos en lexemas y producir como salida una secuencia de tokens o unidades sintácticas para cada lexema en el programa original.

Analizador Sintáctico

Para Aho [1] el análisis sintáctico utiliza los primeros componentes de los tokens producidos por el analizador de léxico para crear una representación intermedia en forma de árbol que describa la estructura gramatical del flujo de los mismos. El analizador sintáctico obtiene una cadena de tokens del

analizador léxico y verifica que la cadena de nombres pueda generarse mediante la gramática para el lenguaje fuente.

Así mismo, Alfonseca *et al.,* [2] aseguran que es el encargado de la interpretacion y compilacion, el cual tiene como objetivo realizar el resto del análisis, iniciado previamente por el analizador léxico, para comprobar que la sintaxis de la instrucción en cuestion es correcta. Por otro lado, el analizador sintáctico reporta cualquier error sintáctico en forma inteligible y se recupera de los errores que ocurren con frecuencia para seguir procesando el resto del programa. El analizador sintáctico construye un árbol de análisis sintáctico y lo pasa al resto del compilador para que lo siga procesando. Este proceso se realiza considerando como símbolos terminales las unidades sintácticas devueltas por el analizador léxico.

Analizador Semántico

Aho [1] establece que el analizador semántico utiliza el árbol sintáctico y la información en la tabla de símbolos para comprobar la consistencia semántica del programa fuente con la definición del lenguaje. También recopila información sobre el tipo y la guarda, ya sea en el árbol sintáctico o en la tabla de símbolos, para usarla más tarde durante la generación de código intermedio. Así mismo, Alfonseca *et al.* [2] establece que el analizador semantico es la parte del compilador que se se comprueba la corrección semántica del programa, el cual recibe como entrada, el árbol del análisis del programa, una vez realizado el análisis léxico y sintáctico.

Como complemento a lo indicado por los autores, se describe el análisis semántico como un proceso mediante el cual se añade al árbol de derivación una serie de anotaciones, que permiten determinar la corrección semántica del programa y preparar la generación de código, en la cual un analizador semántico se compone de un conjunto de rutinas independientes, llamadas por los analizadores léxico y sintáctico. Este análisis utiliza como entrada el árbol sintáctico generado en el paso previo para comprobar restricciones de tipo y otras limitaciones semánticas y preparar la generación de código. Las rutinas semánticas suelen hacer uso de una pila que contiene la información semántica asociada a los operandos en forma de registros semánticos.

Enseñanza de la programación de Software

Según Arias [3] la enseñanza es el proceso de transmisión de una serie de conocimientos, técnicas, normas, y habilidades que está basado en diversos métodos con el apoyo de una serie de materiales, en el cual el docente transmite sus conocimientos al o a los alumnos a través de diversos medios, técnicas, y herramientas de apoyo; siendo él, la fuente del conocimiento, y el alumno un simple receptor ilimitado del mismo. La enseñanza es una actividad realizada conjuntamente mediante la interacción de cuatro elementos: uno o varios profesores, docentes o facilitadores, uno o varios alumnos o discentes, el objeto de conocimiento, y el entorno educativo o mundo educativo donde se ponen en contacto a profesores y alumnos a través de una serie de instituciones u organizaciones destinadas para tal fin.

Tomando lo planteado por los autores, se define la enseñanza de la programación de software como el proceso de transmisión de técnicas, conocimientos y habilidades mediante métodos y estrategias que permiten diseñar, codificar, mantener y depurar el código fuente de programas computacionales a través de una serie de herramientas destinadas para tal fin bajo un esquema previo.

Metodología

Tipo y Diseño de la Investigación

La presente investigación, se enmarca dentro del tipo tecnológica ya que se orienta al diseño de una herramienta de apoyo en el campo de la ingeniería en sistemas. En tal sentido, Carmona [4] expresa que estos estudios implican actividades que, a través de la aplicación del método científico, buscan descubrir nuevos conocimientos para posteriormente buscar aplicaciones prácticas en pro del diseño o mejoramiento de un producto, proceso o equipo.

De acuerdo con lo planteado anteriormente, la presente investigación es de tipo tecnológica ya que pretende utilizar los conocimientos adquiridos, a través de las técnicas de recolección de datos, con aplicaciones prácticas, a fin de proponer un analizador sintáctico-semántico en función de mejorar la enseñanza de la programación de software.

La investigación según su diseño, se clasifica como No Experimental, Transeccional, Según Hernández, Fernández y Baptista [7], en este tipo de diseños se recolectan datos en un solo momento; en un tiempo único. Su propósito es describir variables y su incidencia o interrelación en un momento dado. Por otra parte, no se manipulan deliberadamente las variables. Con base en lo anterior, la presente investigación corresponde al diseño antes mencionado, ya que no se realizan experimentos o se preparan situaciones especiales, donde se establezca una manipulación de las variables, y la recolección de datos, se realiza en un solo momento durante el transcurso del estudio.

Unidad de Análisis

Según Hernández, Fernández y Baptista [7] puede definirse como unidad de análisis al grupo de personas, contextos, eventos, sucesos o comunidades, sobre la cual se habrán de recolectar datos. En virtud de la revisión documental realizada para entender el análisis léxico, sintáctico y semántico realizado al código fuente en virtud de diseñar el producto final, se tomaron como unidades de análisis los *softwares inter-p y p-seint*, utilizados para la escritura y ejecución de algoritmos en idioma español, previo análisis del programa escrito. Igualmente, se ha considerado la bibliografía proporcionada por Aho [1] sobre fundamentos de compiladores.

Resultados

Primeramente, se realizó un diagnóstico en un grupo sobre las estrategias utilizadas en la enseñanza de la programación en estudiantes de Ingeniería en Sistemas del Instituto Universitario Politécnico "Santiago Mariño", Extensión Costa Oriental del Lago – Cabimas. Se pudo apreciar el uso de recursos y estrategias tradicionales (expositivas) con uso de algunas herramientas basadas en las TIC; pero que se limitaban a la emisión de un mensaje de error cuando la estructura sintáctica de la instrucción dada en determinado lenguaje presentaba inconsistencias o estaba escrita incorrectamente.

Una vez realizada la revisión teórica y establecida la metodología que sustenta la investigación, se procedió al diseño y construcción de cada uno de los módulos del analizador sintáctico-semántico. A tales efectos, se procedió de la siguiente manera:

Para el analizador léxico, se construyó un preprocesador sencillo, el cual, serviría para la escritura de cada una de las instrucciones del código fuente. En ese sentido, se diseñó un lenguaje tipo pseudocódigo en español con algunas instrucciones típicas de otros lenguajes cuyo origen fuese anglosajón. Igualmente, se consideraron algunas funciones típicas como apertura, cierre y almacenamiento de archivos fuente.

Con base en las consideraciones anteriores, se procedió a crear una rutina que permitiese leer detalladamente cada uno de los caracteres tanto imprimibles como no imprimibles del código fuente introducido por el usuario en el preprocesador para luego, compararlo con la tabla de símbolos previamente elaborada con soporte a los caracteres ASCII estándar. De encontrar algún carácter no reconocido, se emite un mensaje de error evitando que el proceso continúe; sin embargo, de ocurrir lo contrario, se genera una representación simbólica resultante de la concatenación de cada uno de los símbolos representados en el código fuente.

El paso siguiente, es generar el análisis sintáctico, para lo cual, se crearon rutinas que permitiesen determinar la existencia de números, palabras, variables, constantes y expresiones bien formadas. En ese sentido, se tomaron como premisa las condiciones típicas de los lenguajes de programación para la nomenclatura de identificadores y demás elementos del lenguaje; a saber: una variable no puede empezar con números, ni tener caracteres especiales o ser una palabra reservada; al tiempo que los números sólo pueden soportar dígitos del 0 al 9 y las expresiones deben estar debidamente balanceadas, con operandos y operadores consistentes.

A la luz de lo anteriormente expuesto, se obtiene una representación intermedia a manera de árbol sintáctico que es el insumo principal del análisis semántico, el cual, consiste en la revisión de las estructuras propias del lenguaje contra la tabla de símbolos. En cuanto a este aspecto, se crearon funciones y procedimientos que permitieron determinar si las instrucciones propias del lenguaje están bien escritas y con las reglas definidas por la sintaxis. Se creó una rutina para cada estructura lingüística, que permitió verificar la correcta construcción de las mismas.

En cuanto a la retroalimentación que se arroja al usuario una vez culminados todos los procesos de análisis, se utilizan cuadros de diálogo descriptivos con mensajes de error, advertencia e información. El lenguaje utilizado fue Java con bibliotecas gráficas GTK+ y el Entorno de Desarrollo Eclipse, todo esto con el propósito de proporcionar un ambiente amigable y adecuado a los sistemas operativos Windows y Linux. Para la interfaz de usuario, se utilizaron elementos como menús, botones de comando y cuadros de texto para la comunicación usuario-aplicación. Así mismo, se definió una disposición de elementos que favoreciera la utilización del software, tal como se muestra en la Figura 1:

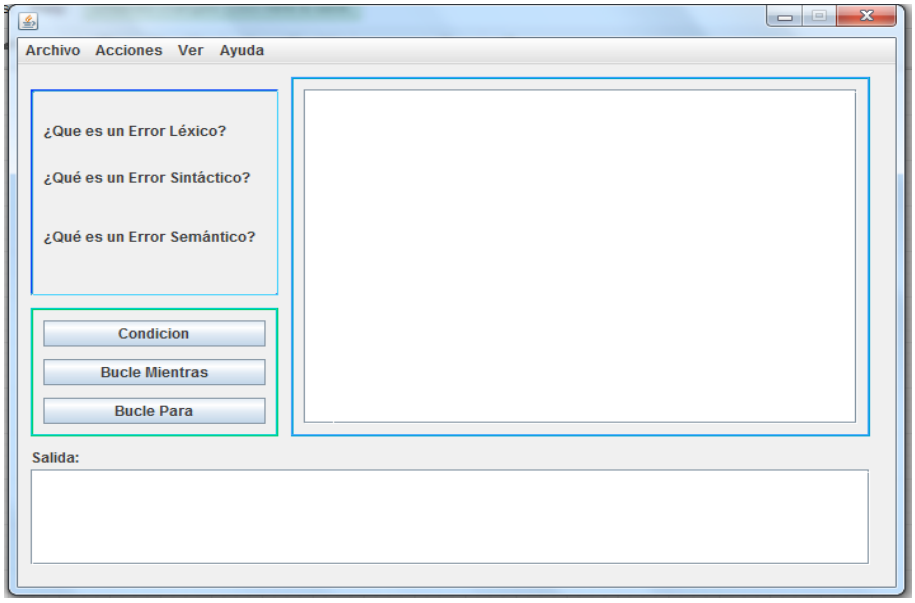

Figura 1. Interfaz de usuario del Analizador Sintáctico-Semántico.

Es de hacer notar la presencia de botones de comando que brindan la posibilidad de insertar estructuras del lenguajes como condiciones y bucles con propósitos didácticos, es decir, ayudar al estudiante a entender la lógica funcional de las mismas y una ventana de salida donde se pueden observar los resultados obtenidos una vez realizados los análisis léxico, sintáctico y semántico del código fuente.

Discusión

Con base en los resultados obtenidos, se procedió a realizar la sustentación de los mismos con los postulados teóricos. En ese sentido, se obtuvo un analizador léxico que es capaz de comparar los caracteres del código fuente con la tabla de símbolos del lenguaje. En tal sentido, corresponde con las ideas de Aho [1] quien indica que un analizador léxico, lee los caracteres que componen el código fuente y se generan representaciones o tókens que son la entrada del análisis sintáctico, previa comparación con el alfabeto del lenguaje.

Seguidamente, el análisis sintáctico revisa la correcta formación de palabras, variables, números y expresiones, generando como salida intermedia con el respectivo árbol sintáctico. Esto presenta similutud con los postulados de Aho [1] quien asegura que éste utiliza los primeros componentes de los tokens producidos por el analizador léxico para crear una estructura en forma de árbol que describa la gramática del flujo de los mismos.

En el mismo orden de ideas, el análisis semántico toma como entrada el árbol sintáctico para verificar la congruencia de las estructuras como expresiones u operaciones y verifica la localización de las palabras de acuerdo a la instrucción analizada. En función de lo anterior, Aho [1] establece que se utiliza el árbol sintáctico y la tabla de símbolos para comprobar la consistencia interna del programa con las sentencias del lenguaje.

A la luz de todo lo anterior, se puede establecer que el producto obtenido corresponde con los aspectos teóricos tomados como base de la presente investigación, ya que primeramente, se realiza la revisión de cada carácter del código fuente de acuerdo a la tabla de símbolos del lenguaje, se verifica la consistencia de cada token, formación de palabras, variables, números, expresiones y elementos sintácticos en general con el propósito de generar el árbol sintáctico y posteriormente la representación intermedia de los elementos semánticos, necesarios para proceder con las fases subsiguientes de la compilación.

De esta manera, el estudiante puede visualizar paso a paso lo que ocurre cuando está frente a un compilador, el cual, recibe como entrada un programa escrito en cualquier lenguaje y luego, se obtiene el código objeto en binario. Cabe destacar que esto último no es realizado por el software propuesto, pero proporciona una ilustración sobre lo que ocurre en las fases iniciales, las cuales, están al alcance del personal experimentado y especializado en compilación de programas.

Conclusiones

En relación al diagnóstico sobre el cual se sustentó la investigación, se pudo apreciar que los procesos de enseñanza de la programación de software, se apoyan en estrategias expositivas con nivel intermedio de integración con las TIC, por lo cual, se deben complementar con herramientas basadas en el análisis léxico, sintáctico y semántico, lo cual ayudará al estudiante en la comprensión de las estructuras de los lenguajes de programación, así como también, a entender lo que ocurre durante las fases de conversión del código fuente a la representación intermedia, previa a la compilación.

Al mismo tiempo, resulta pertinente introducir al estudiante en el estudio del inglés técnico para así facilitar el aprendizaje de los distintos lenguajes. Igualmente, es conveniente complementar la herramienta diseñada con un conjunto de algoritmos a modo de ejemplo, para que pueda apreciarse la descomposición simbólica que surge a partir del análisis léxico, sintáctico y semántico del código fuente de un programa.

Con referencia a los elementos que conforman el analizador sintáctico y semántico, se concluye que los en primera instancia, se trabaja con los tokens y tablas de símbolos. Igualmente, entran en juego las palabras bien formadas, variables, números, expresiones y sentencias propias del lenguaje de programación. Ahora bien, una vez superados estos procesos, el software presentó rutinas destinadas a la validación de instrucciones de acuerdo a reglas de sintaxis, que a su vez le daban sentido o significancia al código fuente.

Por último, se construyó un analizador sintáctico-semántico dirigido a estudiantes de programación con elementos gramaticales propios de los lenguajes utilizados en el ámbito de la informática. Igualmente, se creó un preprocesador con componentes propios del entorno gráfico, cuyo propósito es facilitar la escritura y administración de archivos fuente para ayudar a ilustrar las fases del proceso de compilación de cualquier programa. Esto último, reviste importancia, ya que presenta un resultado que usualmente no se percibe en los compiladores comerciales, ya que se puede visualizar la representación intermedia (árbol sintáctico) al realizar la descomposición del código en expresiones menos complejas.

Referencias Bibliográficas

[1] Aho, A., Compiladores: Principios, Técnicas y Herramientas (2ed.), Pearson Educación, México D.F. (2008).

[2] Alfonseca M., De La Cruz M., Ortega A., y Pulido E., Compiladores e Interpretes: teoría y práctica, Pearson Educación S.A, Madrid, (2006).

[3] Arias D., Enseñanza y Aprendizaje de las Ciencias Sociales: Una propuesta didáctica, Cooperativa Editorial Magisterio, Bogotá, (2005).

[4] Carmona Y., Investigación Tecnológica, Universidad de las Alas Peruanas, Facultad de Ingeniería. Lima, Perú (2011).

[5] Diaz A., y Serra L. Competencias Específicas en los estudiantes de Programación del Instituto Universitario Politécnico Santiago Mariño. Ethos Venezolana Vol. 7, No. 1, (2015) 46-57.

[6] Grune D., Bal H., Jacobs C., y Lagendoen K., Diseño de Compiladores Modernos, Mcgrawl-Hill, México D.F. ,(2007).

[7] Hernández R., Fernández C., y Baptista P., Metodología de la Investigación (5ed.). Mcgrawl-Hill, México D.F. (2010).