

Tecnológico de Costa Rica

Análisis Sintáctico Compilador MiniC#

Curso:

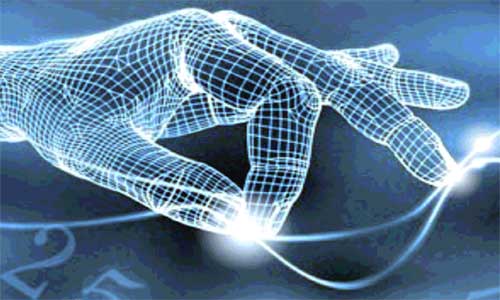
Compiladores e Interpretes

Estudiantes:

* Josué Hidalgo
* Frarlon Rodriguez
* Ricardo Venegas

Profesor:

Oscar Víquez Acuña



2012

Análisis del lenguaje:

*(Análisis de la gramática, del código que puedo crear o no puedo crear con ella)*

La gramática me permite crear programas creados por una sola clase, cuya clase está compuesta por declaraciones de constantes, variables o clases internas, y un cuerpo de la clase formado por declaraciones de métodos; las clases declaradas internamente pueden ser implementadas solamente para declarar variables dentro de ellas, utilizadas como registro de información. Existen comentarios anidados, de manera tal que se permita abrir un comentario dentro de otro, además que pueda iniciar en una línea y terminar en otra o en la misma en que empieza el comentario. Están las clases léxicas “*letter*”, cualquier letra minúscula o mayúscula, “*digit*”, un número entre cero y nueve, e “*id*” como una letra o una letra seguida de más letras o dígitos. Los tipos de datos son hechos como tipos básicos int, char, float y bool, además de poder ser un *id*.

Soluciones e implementación:

(Cuestiones técnicas de la implementación del proyecto)

En el proceso de la realización del Scanner se implementó la herramienta “*C#Lex.exe*”,para de esta manera poder generarlo por medio de un archivo *lex* para nuestra gramática. Para la creación del scanner se estudió la gramática y se tomaron las palabras terminales, y los patrones de estas mismas, se creó una clase token en donde se almaceno variables para identificar los tokens. Se analizó la estructura de los archivos lex para representar patrones y demás atributos del scanner.

El “*parser”* fue implementado mediante la implementación del algoritmo TOP-DOWN de descenso recursivo, analizando la gramática, y las posibilidades de cada no terminal, cada expresión no terminal de la gramática se represente como un método del parser para analizar las opciones que se pueden tomar.

Para la implementación de la estructura del árbol de sintaxis abstracta (*AST* – *Abstract Syntax Tree*) se ha utilizado la estructura de datos orientado a objetos, especificando las clases según sus implementaciones en la gramática y fracciones, y al mismo tiempo abstrayendo características universales.

Resultados obtenidos:

(Cuadro explicativo de lo realizado y lo no realizado – énfasis en esto último)

|  |  |
| --- | --- |
| Scanner | implementado |
| Parser | implementado con problemas |
| AST | implementado con problemas |
|  |  |

Conclusiones:

Del trabajo

Bibliografía:

Bibliografía. (Documentos con autor)