ANALIZADOR LÉXICO Y SINTÁCTICO

declare_list (tema P4) de PHP

Alumno: Boglioli, Alan
Legajo: 38507

INTRODUCCIÓN

El proyecto realizado se enfoca en ser un analizador léxico y sintáctico para la declaración declare_list (tema P4) del lenguaje PHP. declare_list se refiere a la lista, que se pasa como argumento, cuando se llama a la función "declare(...)":

*declare(**<declare_list>**) {*
    ...
}

declare_list, entonces, es el argumento de dicha función.

El ejecutable encargado de realizar dicho análisis se llama ***analizadorP4.exe***.

HERRAMIENTAS UTILIZADAS

- Flex (generador de analizador léxico)
- Bison (generador de analizador sintáctico)
- Compilador GCC
- Plataforma MinGW para la utilización de herramientas GNU en el sistema operativo Microsoft Windows.

ESPECIFICACIÓN BNF

La especificación en BNF de *declare_list* se encuentra en el archivo declare_list_bnf.txt el cual está dentro de la carpeta doc/ del directorio principal del proyecto.

CÓMO UTILIZAR EL ANALIZADOR

Al analizador se le pueden pasar las cadenas a ser reconocidas de dos formas distintas: a través de un archivo e ingresándolas por línea de comando.

El nombre del archivo se puede pasar como argumento de línea de comando en el momento que se llama al analizador:

analizadorP4.exe nombre_archivo.txt (en cmd.exe, siendo nombre_archivo.txt el archivo a analizar)

O también, se puede ejecutar desde consola sin argumentos. Al ejecutarse de esta forma, el programa mostrará un menú con opciones. Para usar algunas de las funciones descriptas por

las opciones se debe ingresar el número de opción. Por este método, también existe la opción de abrir un archivo.

Una vez que se hayan pasado las cadenas a analizar, el programa mostrará por pantalla la tabla de símbolos primero y luego el análisis sintáctico, el cual muestra las respectivas derivaciones de los no terminales (símbolos en minúscula encerrados por < y >).

INGRESAR CADENAS POR LÍNEA DE COMANDO

Al ser, declare_list, una lista que se pasa como argumento, las declaraciones son separadas por una "," (coma). Por ejemplo:

w=5, s=array(9), asd= saludo::hola ...

También, dichas declaraciones se pueden realizar en múltiples líneas, debido a que PHP es un lenguaje muy flexible que no considera lo saltos de línea como tokens, sino que los borra, al igual que los espacios en blanco.:

```
w=8,
arr=array(j => 5,
r=>98
,
www=65)
```

FUNCIONES PRINCIPALES DECLARADAS POR EL USUARIO

Para imprimir la tabla de símbolos, se creó la función *printSymbolTable()*, la cual recorre un arreglo que contiene los tokens y sus respectivos valores y los va imprimiendo.

Para imprimir el desarrollo del análisis sintáctico, derivando los símbolos no terminales que van apareciendo, se desarrollaron dos funciones. Una vez terminado el análisis sintáctico realizado por Bison, se llama a *makeTree()*, función que permite armar el árbol sintáctico y luego de esto a *printTree()*, la cual a su vez imprime todos los niveles del árbol gracias a la función *printTreeLevel()*.

La función más importante es *makeTree()* debido a que, para derivar desde el primer nivel de no terminales (declare_list) hacia abajo, es necesario hacer uso de un arreglo que almacene las estructuras reconocidas por Bison, y luego las imprima en forma inversa.

Para el árbol sintáctico se usarion dos estructuras, *s_treeLevel* y *s_token*. La estructura *s_token* tiene una variable llamada *level_ref*, usada en caso de que este token sea un no terminal, para referenciar el nivel donde se encuentra la cadena que deriva de dicho no terminal. Y así, cuando se imprima el árbol, se reemplazará la cadena derivada del no terminal en lugar de este.

OBSERVACIONES y CONCLUSIONES

Lo que más tiempo me llevó es lograr definir las funciones necesarias para lograr que se imprima el “árbol” sintáctico, con sus derivaciones, ya que Bison genera un analizador de abajo hacia arriba y de derecha a izquierda, por lo que debí invertir el orden de impresión.

Es un proyecto muy interesante. A pesar de usar GNU/Linux nunca había utilizado estas herramientas (Flex y Bison), ni las conocía. He descubierto que son muy importantes para facilitar el trabajo a las personas que se encargan de desarrollar nuevos lenguajes de programación.