TRABAJO PRÁCTICO

TEORÍA DE LENGUAJES, AUTOMATAS Y COMPILADORES

Martín Victory
Florencia Cavallin
Segundo Fariña

Objetivo

El objetivo del trabajo, como anuncia la consigna, fue desarrollar un compilador para un lenguaje creado por nosotros mismos. Pensamos en crear un lenguaje que no fuera necesariamente útil para el desarrollo de algo en particular, sino que con un objetivo más creativo e innovador.

Este lenguaje se basa en las acciones de un personaje, más específicamente, en un 'stickman'. El personaje principal es único y todas las funciones que se pueden crear en este lenguaje se centran en generar posibles nuevas actividades que pueda realizar.

Consideraciones

Además de las especificaciones dadas por la cátedra, decidimos seguir el consejo dado en clase de crear un árbol de sintaxis y así poder parciar de forma correcta, eficiente y más rápido el código de nuestro nuevo lenguaje.

En 'syntaxTree.h' se podrán encontrar las definiciones de todas las estructuras utilizadas para generar el árbol. Cada nodo contiene los nodos de sus respectivas derivaciones a realizar y en caso de poder elegir entre muchas derivaciones, se almacenó la derivación que se llevo a cabo y un nodo void que más tarde es castigado una vez realizada la derivación correspondiente. Todos los nodos constan de una runCode, la cual es una función que devuelve un puntero a void. Esta función es distinta para todos los nodos, que representan todos los símbolos no terminales juntos a todas sus producciones posibles, ya que para cada derivaciones habrá que ejecutar una función particular. Las funciones para cada producción se encuentran en el archivo 'nodeFunctions.c'. Por último, con respecto al árbol sintáctico, en el archivo 'syntaxTree.c' se encuentran todas las funciones de creación y agregación para cada nodo.

Desarrollo

YACC

El archivo yacc es yacc.y. En este definimos con un %union todos los tipos de nodos de nuestro árbol sintáctico. También agregamos todos los tokens posibles para el lenguaje y con %type unimos todos los símbolos no terminales de nuestro lenguaje con sus respectivos nodos.
Decidimos agregar precedencias para los operadores lógicos y de asignación con el objetivo de eliminar la ambigüedad. Definimos start con nuestro símbolo no terminal Start y creamos la gramática. En cada una de las posibles producciones creamos los nodos para poder construir el árbol sintáctico. Usamos /* empty */ para las producciones lambda.

LEX

El archivo lex es lexl.l. En este archivo definimos las expresiones regulares de los tokens de nuestro lenguaje. Por ejemplo: el nombre de los archivos a incluir, el nombre de las variables y las palabras reservadas, entre otras. De esta forma podemos agregar restricciones a la escritura de nuestros tokens.

‘declaraciones.c’

Como su nombre lo indica, en este archivo guardamos las declaraciones de todas las variables del código. Consta de estructuras que definen los atributos de una variable, en el caso de un integre: nombre y valor, en el caso de una función: tipo de retorno, nombre y parámetros. Las posibles variables que son guardadas, es decir los tipo de variables que se pueden declarar en este lenguaje son: integers, strings, boolean y funciones.

‘lib.c’

En este archivo contenemos al encargado de manejar lo que se ve en pantalla. Desde los comando posibles que puede ingresar en usuario, hasta la representación gráfica de las actividades que realiza el ‘stickman’.

‘lib.stickLib’ ‘other.stickLib’

Ambos archivos son librerías estándares de ‘stickman’ para el código pero el objetivo de este lenguaje es que el usuario aumente las posibles acciones del personaje con nuevas librerías.

TestCodes

Este directorio contiene los programas ejemplo pedido por la consigna.

Compiled

Compiled es un directorio donde se guarda la traducción del lenguaje ‘stickman’ a C para el código generado.

MakeFile

En el makefile se compila primero el yacc.y, luego el archivo lex.l. Se borran los archivos lex.yy.c, y.tab.c y y.tab.h para hacer un clean en el directorio y volver a generarlos. Finalmente se compila con gcc los archivos: cstickman.c, y.tab.c, lex.yy.c syntaxTree/*.c y helpers/*.c bajo el nombre de cstickman y se redirecciona la salida al directorio compiled, como fue aclarado anteriormente.

Descripción de la gramatica

La descripción de la gramatica se puede encontrar en el README.md