UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO.
FACULTAD DE INGENIERÍA.

Compiladores

Analizador Léxico

Torres Ruiz Mateo

Rodríguez Mendoza Angel Rafael

Fecha de entrega: 19 de Septiembre de 2017

Descripción del problema

Este proyecto consiste en la creación de un analizador léxico como primera fase para crear un compilador al final del curso. El analizador debe ser capaz de: reconocer las clases que definimos en el salón de clases, crear la tabla de símbolos, de cadenas y la tabla final de tokens, así como tener algún tipo de manejador de errores cuando no se reconocer una palabra como parte del lenguaje. El analizador recibirá el código fuente, el cuál puede contener errores y comentarios, y debe dar como resultado las tablas antes mencionadas.

Hay nueve clases definidas en el lenguaje:

- 0 Identificadores: los identificadores solo están compuestos por palabras en minúsculas. (a..z)(a..z)*
- 1 Palabras reservadas: son ENT, ESCRIBE, HAZ, LEE, MIENTRAS, REAL, SI, SINO. (ENT | ESCRIBE | HAZ | LEE | MIENTRAS | REAL | SI | SINO)
- 2 Operadores de asignación: el único operador de asignación es “=”. =
- 3 Operadores relacionales: solo son aceptados >, >=, <, <=, ==, !=. (> | >= | < | <= | == | !=)
- 4 Operadores aritméticos: son DIVIDE, MAS, MENOS, MULTIPLICA. (DIVIDE | MAS | MENOS | MULTIPLICA)
- 5 Símbolos especiales: solo son aceptados “,”, “;”, “[“, “]”, “(“, “)”. (, | ; | [|] | (|))
- 6 Constantes numéricas enteras. (1..9)(0..9)*
- 7 Constantes numéricas reales: siempre con un “.” (1..9)(0..9)*.(0..9)
- 8 Constante cadena: se guarda la cadena junto con las comillas que abren y las que cierran. ”.*”

Propuesta

Por consideraciones de escalabilidad y como reto personal, decidimos usar C++ como el lenguaje para desarrollar el proyecto, ya que es un lenguaje más flexible que C y que también cumple con las especificaciones ANSI C++. Se hizo uso de las estructuras enum para poder definir el autómata finito determinístico y de algoritmos simples que reconocen las cadenas, identificadores y constantes numéricas. También nos apoyamos mucho de la herramienta Git para controlar y mantener control sobre los cambios que hacíamos en el proyecto.

Desarrollo

Análisis

Decidimos dividir el trabajo en tres etapas principales: la definición de los tokens y sus clases, lo cual le correspondió a Mateo. Después se trabajó sobre el analizador propiamente, al cual nosotros le llamamos lexer. El lexer fue trabajado por ambos miembros del equipo. La última etapa consistió en crear un makefile para facilitar la compilación del programa, así como un pequeño script que comprueba que el

compilador instalado es compatible con g++11, ambas tareas fueron desarrolladas por Ángel.

Diseño e implementación

Los tokens fueron definidos de dos formas diferentes. Las palabras reservadas, operadores aritméticos, símbolos especiales y operadores relacionales fueron implementados con estructuras tipo enum, por lo tanto, solo guardamos el token y lo buscamos en las estructuras correspondientes. En caso de que se encontrará, agregamos el token a la tabla de de componentes léxicos. En caso de que no se encuentre, se imprime un mensaje de error.

Para la clase de identificadores, constantes numéricas enteras y reales y las constantes de cadenas, se diseñaron algoritmos que identifican a cada una de estas clases. El operador de asignación fue lo más sencillo ya que solo hay un solo elemento que pertenece a la clase. Todos estos métodos para reconocer los tokens, están definidos en el archivo lexer.cpp.

En cuanto a las tablas de símbolos y cadenas, en el mismo archivo lexer.cpp, cuando se reconoce una clase tipo identificador o de cadenas, se mandan a llamar las funciones que guardan en las tablas correspondientes. Para guardar, primero se usa la función find que es propia de las estructuras de dato tipo map. En caso de que no se encuentre, se agregan los nuevos elementos y se actualiza la posición de las tablas.

Al momento de hacer la impresión de las tablas de símbolos y de cadenas, los valores aparecen en desorden, esto es debido a la naturaleza de la estructura map. Sin embargo, ese orden no es de relevancia para la ejecución del programa.

Indicaciones

Antes de compilar y ejecutar el programa, es recomendable ubicarse en el directorio que contiene los archivos del proyecto y ejecutar el archivo “configure”. Dependiendo del sistema, puede que este archivo este listo para ejecutar o se le tengan que otorgar permisos. Para otorgarle permisos debe ejecutar:

$chmod u+x configure

Después solo es necesario ejecutarlo:

$./configure

Este script verifica que se encuentre gcc-4.7, ya que es la versión de gcc mínima que soporta g++11. En caso de que no se encuentre una versión mayor o igual a la 4.7, se instalará gcc-4.7 y g++-4.7, para lo cual se pedirá la contraseña para poder hacer instalaciones. En caso de que ya no se necesite instalar nada, el script mostrará un mensaje informando que todo está listo para compilar.

Para compilar el proyecto, solo hace falta ubicarse en el directorio que contiene los

códigos fuente y ejecutar:

$make

Este comando creará un archivo llamado anaLex, el cuál es el ejecutable. Para poder ejecutarlo solo se necesita ejecutar:

$./anaLex *archivoAAnalizar*

Conclusiones

Torres Ruiz Mateo

Este proyecto presentó un reto porque fue necesario integrar conocimientos de varias asignaturas. En general, siento que el aprendizaje que me dejó el investigar, diseñar e implementar el proyecto, fue muy bueno. También creo que el trabajo colaborativo es muy valioso y nos acostumbra a cómo es que se trabaja en el mundo profesional. En este proyecto intentamos utilizar tecnologías como Git para poder trabajar de forma organizada, lo cual facilitó mucho el desarrollo del trabajo. En general ha sido de mis proyectos favoritos y estoy ansioso de terminar el compilador completo.

Rodríguez Mendoza Ángel Rafael

Con este proyecto me pude dar cuenta de la relación e importancia que tienen las asignaturas que hemos cursado hasta el día de hoy, ya que es de los primeros proyectos en donde veo aplicaciones reales y muy relevantes de conceptos vistos en lenguajes formales y automatas y estructuras discretas. También creo que el conocer cómo funciona un compilador es parte esencial de nuestra formación, ya que es una de las tecnologías que han permitido tanto avances en la computación. El proyecto también me sirvió para refrescar mis conocimientos sobre algoritmos, estructuras de datos y un poco de sistemas operativos. En general ha sido uno de los proyectos más retadores y que más satisfacción me han provocado en la carrera.