**Reporte**

Fecha: 06/02/19

**COMPILADORES**

PROFESOR: DAVID TORRES MARTINEZ

EQUIPO #2

GRUPO:502-A

**INTEGRANTES DEL EQUIPO**

* **ALDERETE AGUILAR RUBÉN**
* GARCIA MERCADO ANDRÉS
* **GIL LÓPEZ HUMBERTO**
* MARTINEZ HERNÁNDEZ KEVIN DAVID
* **PACHECO PACHECO ANA LAURA**

***Introducción***

La tarea que desarrolla un compilador resulta ser invisible ante los ojos del usuario, ya que la manera de establecer y mantener la comunicación entre la computadora y el usuario viene dada cuando el usuario escribe una orden, y sin más ni menos la computadora muestra la información que responde a la orden del usuario. La pregunta es, ¿cómo se ejecuta esta orden? El avance de la computación tuvo que ver con lograr que el lenguaje natural de los sistemas fuera entendido por las máquinas computacionales de una manera sencilla, pero fue un proceso complicado hasta llegar a lo que se conoce como compilador. Un compilador es definido como un programa que traduce del programa fuente a programa objeto. En este curso de compiladores se consideró la fase de análisis para el desarrollo del compilador, la cual consta del analizador léxico, sintáctico y semántico. A lo largo del curso se contemplaron diferentes algoritmos complementarios y necesarios para el desarrollo de cada uno de estos analizadores. En este sprint se entrega la fase de análisis del compilador, con su respectiva traducción, el lenguaje fuente es C y el lenguaje destino es Pascal.

***Desarrollo***

Para desarrollar la traducción de lenguajes, se necesitó corregir y definir muy bien la gramática, para después obtener las acciones semánticas de esta gramática y de esta manera se implementó en el lenguaje JAVA en el entorno de desarrollo integrado NETBEANS, también se realizó la conexión de los diferentes programas en una sola interfaz, dado que la interfaz se modificó se tuvo que realizar la conexión del proyecto nuevamente, por otra parte se realizó el desarrollo de la misma fase de análisis utilizando los programas Flex y Bison. Esta entrega requirió de una mayor organización, puesto que en esta ocasión se trató de un trabajo que exigió una coordinación más eficiente que en las entregas anteriores, esto a causa de que las modificaciones que se realizaron implicaron labores de diseño tanto de gramática como de las acciones semánticas y una versión final de todos los proyectos entregados durante el curso. A continuación, se enuncia el campo de colaboración de cada integrante del equipo.

El compañero perteneció al contingente número 2, el cual se encargó de modificar la gramática para agregar otras funcionalidades y corregir algunas que ya se hacían, para ello utilizó la última versión del proyecto (correspondiente al sprint #11) y de esta manera realizar pruebas de que lo que se estaba desarrollando, una vez que tuvo correcta la gramática, se pasó a obtener las acciones semánticas de esta, para esto se dedicó a buscar en internet la sintaxis del lenguaje Pascal y así lograr la traducción de C a Pascal. Estuvo al pendiente para nuevas modificaciones que el compañero Humberto necesitaba hacer para programación de las acciones semánticas, atendiendo en todo momento lo que se requiriera y sugiriendo diferentes maneras de implementación. Dado que esta entrega contempla todos los algoritmos de los diferentes analizadores, se tuvieron que hacer modificaciones en el analizador léxico que no funcionaba del todo bien, por lo que el compañero realizó la tabla de símbolos, la cual contempla una lista de los símbolos (identificadores) del código de entrada, además de mostrar el valor y la función en la que se encuentra el símbolo, al igual realizó la implementación de la tabla de errores, en la cual muestra el símbolo que no forma parte del lenguaje y el número de línea en la que se encuentra.

**Alderete Aguilar Rubén**

El compañero trabajó en el contingente número 1, este contingente fue el encargado de hacer la implementación de los analizadores léxico, sintáctico y semántico en los programas Flex y Bison. Por lo que su primera actividad fue instalar estos programas y checar los manuales para saber su funcionamiento. Específicamente realizó la implementación del analizador semántico en Bison, para lo cual necesitó de los analizadores desarrollados por los otros compañeros de contingente. Bison es un programa que tiene como entrada una especificación sintáctica. Bison es un generador de análisis de propósito general que convierte una descripción gramatical para un LALR la gramática libre de contexto en programa de C para analizar esa gramática. El analizador semántico da un significado coherente a lo que se ha hecho en el análisis sintáctico. Se encarga de que los tipos de las expresiones sean compatibles entre otras tareas, por lo que el compañero se concentró en realizar esta parte del compilador necesitando las acciones semánticas de la gramática definida, todo esto en el formato que ocupa Bison, creando un archivo donde se especificó el léxico, el cual es el mismo archivo que se ocupó para el analizador sintáctico, también se ocupó otro archivo en el que se especificaron las reglas de producción (gramática), muy similar al que se ocupó para el analizador sintáctico teniendo como extra la especificación de las acciones semánticas.

**García Mercado Andrés**

**Gil López Humberto**

Humberto en esta entrega se dedicó principalmente en la codificación de las acciones semánticas, al igual que corregir en la programación algunas cosas de la gramática que teníamos mal en el sprint anterior, cabe mencionar que él también se encargó de modificar la nueva interfaz y así de la misma manera conectarla al proyecto para que se mostraran correctamente los datos a la hora de ejecutar el programa. Otra de las cosas que realizo en esta entrega fue proponer algunos casos de prueba para verificar que el flujo de los algoritmos implementados fuese el correcto.

**López Morales Valentín**

El compañero Valentín en este sprint se encargó de realizar el analizador léxico usando la herramienta Flex, por lo cual tuvo que investigar el funcionamiento de este. Lo que nos comento es que Flex recorre una entrada estándar hasta que encuentre una concordancia y ya después permite ejecutar la acción, que en este caso fue un código en C. También nos comentó que para usar Flex con Bison, normalmente se especifica la opción –d de Bison para que genera el fichero ‘y.tab.h’ que contiene las definiciones de todos los ‘tokens’ que aparecen en la fuente Bison. Cabe mencionar que igual realizo los ejemplos que se encuentran en la página y se basó en los manuales que se encuentran en la misma para así poder entender más rápido como utilizar Flex.

**Martínez Hernández Kevin David**

Kevin fue parte del contingente número 1, él se encargó de realizar el analizador sintáctico del compilador, usando Bison para desarrollarlo, de igual forma su primera actividad fue instalar este programa y estudiar los manuales proporcionados por el profesor en su sitio web, además de realizar los ejemplos mostrado en las diapositivas para comprender mejor su funcionamiento. El analizador sintáctico consistió en capturar en un archivo léxico, es decir, todos los tokens de la gramática y se ocupó otro documento en el que se especificó el funcionamiento de la gramática, qué hacer cuando se encontrara a ciertas reglas, al igual que se realizó una función que trata los errores, esta función se llama error y otra que activa y hace trabajar a todo el analizador sintáctico, esta función se llama yyparser. Todo esto especificado en el mismo archivo excepto el léxico mencionado en primera instancia.

La compañera se encargó de conectar todas las partes de los diferentes analizadores en una nueva interfaz ya antes realizada, y tratar las interfaces de salida para que el manejo de la interfaz fuera más intuitivo y que los diferentes algoritmos de cada analizador mostraran únicamente el resultado que les correspondiera. También se encargó de realizar el reporte de trabajo, al igual que el checklist dependiendo de lo que se contempló en la gramática generada por los otros compañeros y de lo que logró programarse de las acciones semánticas. La parte del analizador léxico considera el algoritmo de Thompson, el AFD, el analizador léxico con tu tira de tokens, tabla de símbolos y tabla de errores. Para el analizador sintáctico se contempló el algoritmo de primeros y siguientes, la colección canónica, tabla de análisis sintáctico y finalmente el analizador sintáctico-semántico. Las acciones semánticas fueron las encargadas de realizar la traducción a Pascal. Todo esto es lo que puede realizarse a través de la nueva interfaz.

**Pacheco Pacheco Ana Laura**

Para la traducción se consideró la sintaxis básica de Pascal, se presenta una tabla con lo utilizado para cada caso en C a Pascal.

|  |  |
| --- | --- |
| Acción | Pascal |
| Declaración de apuntadores | type  ptrnum = ^tipo; |
| Declaración de variables | Var  ListaIdentificadores1 : TipoDato1; |
| Declaración de arreglos | Var  nombre: ARRAY[rango] OF tipo; |
| Declaración de arreglo de apuntadores | No se contempló |
| Inicialización de variables | begin  numero1 := 5;  End. |
| Inicialización de arreglos | No se contempló |
| Operaciones aritméticas | Var  Suma:integer  Begin  Suma:=12+8;  End. |
| If/else | if <expresión\_lógica> then  begin   <bloque\_de\_instrucciones\_1>  end;    else   begin   <bloque\_de\_instrucciones\_2>    end; |
| while | while <expresión\_lógica> do   begin          <bloque\_de\_instrucciones>  end; |
| For | for <variable> := <valor\_inicial> to <valor\_final> do  begin    <bloque\_de\_instrucciones>    end; |
| Switch | case <expresión> of   <lista\_de\_constantes\_1> : begin   <bloque\_de\_instrucciones\_1>   end;   <lista\_de\_constantes\_2> : begin   <bloque\_de\_instrucciones\_2>   end;   ...  else   begin    <bloque\_de\_instrucciones\_n+1>   end   end; |

Conclusión

La entrega fue iniciada desde el día indicado en el documento de asignación de tareas, comenzando con la gramática y las acciones semánticas, concluyendo con la traducción y la generación de los analizadores en Flex y Bison, el desarrollo de los trabajos contemplados para la entrega fue difícil, ya que hubo pequeños detalles en los cambios realizados, el tiempo no nos era suficiente para corregir esos errores que tal vez eran resueltos pero generaba otro error, finalmente se pudo concluir el trabajo con detalles mínimos pero que esperamos no afecten demasiado el objetivo principal del compilador integrado.