Trabajo Práctico Especial

1. Introducción

Los compiladores son elementos clave de la informática y los que permiten convertir prosa digital en acción. Hay cientos, miles y probablemente en cada clase de Lenguajes y Compiladores del mundo se contruya uno. No vamos a hacer menos entonces...

1.1. Objetivo

El objetivo principal es el desarrollo completo de un lenguaje y su compilador, construyendo los dos componentes principales de un compilador: el analizador léxico y el analizador sintáctico. Para ello, tienen que elaborar primero la idea del lenguaje, crear la gramática e implementar el compilador que genere un programa ejecutable para alguna plataforma. El programa tiene que estar desarrollado en C, utilizando Lex como Scanner (analizador léxico) y Yacc como Parser (analizador sintáctico).

2. Primera Parte: Gramática

La primera parte es la creación de la definición de la gramática que produce el lenguaje que el Compilador aceptará y transformará.

El mismo debe contener obligatoriamente y como mínimo:

- Tipos de datos: numérico y cadenas.
- Constantes, Delimitadores.
- Operadores aritméticos, relacionales, lógicos, de asignación.
- Bloque Condicional (if).
- Bloque Do-While (repetición hasta condición).

Ejemplo de un lenguaje posible:

```
int x;
int y;
main() {
  x = 6;
  y = 1;
  while (x>0) {
   y=y*x;
   x=x-1;
   }
  puts("El factorial de 6 es: ");
  puts(y);
  puts("\n");
  puts("El valor de x+1 es:");
  puts(x+1);
}
```

(el anterior es un ejemplo, pueden definir ustedes las palabras reservadas que deseen, y la estructura que les parezca más conveniente).

La primera parte deberán entregarla para **la clase del 28 de Octubre del 2017**, conteniendo

- Proyecto Completo: código fuente, librerías adicionales, Makefile de Compilación.
- Gramática en un archivo BNF (Backus Nair Form)
- Scanner en LEX (archivo .l) que procese de entrada un programa en en lenguaje creado, e identifique los lexemas y genere como salida una palabra del lenguaje.
- Un README.md explicando cómo compilar y ejecutar el programa.

3. Segunda Parte: Yet Another Compiler Compiler

Los compiladores de compiladores, son herramientas que permiten en base a una definición de un autómata finito y una gramática en BNF generar un parser que pueda aceptar la palabra del lenguaje, detectar posibles errores sintácticos, detectar errores semánticos y traducir el programa para generar un binario ejecutable para alguna plataforma.

La implementación deberá realizarse con Yacc, complementado lo realizado en la primera parte con Lex.

El compilador debe:

- Tener un mecanismo para indicar cuál es el punto de entrada (i.e. el *main*).
- Proveer un mecanismo de entrada de datos y de salida.
- Estar programado en C.
- Generar código ejecutable (en el lenguaje que quieran, preferentemente C o assembler).

4. Material a entregar

Para la Segunda Entrega, la entrega final, deberán presentar:

- Proyecto Completo: código fuente, librerías adicionales, Makefile de Compilación.
- Los ejecutables del compilador. Pueden incluir herramientas y librerías adicionales que ustedes utilicen.
- Un README.md explicando cómo compilar y ejecutar el programa.
- Cinco(5) programas de ejemplo.
- Un informe que contenga, en este orden:
  - Carátula
  - Índice
  - Idea subyacente y objetivo del lenguaje.
  - Consideraciones realizadas (no previstas en el enunciado).
  - Descripción del desarrollo del TP.
  - Descripción de la gramática.
  - Dificultades encontradas en el desarrollo del TP.
  - Futuras extensiones, con una breve descripción de la complejidad de cada una.
  - Referencias.

Adicionales deseables que se evaluarán positivamente:

- Optimizaciones.

- Benchmarking de los tiempos de ejecución de alguno de los programas de ejemplo (e.g. test de primalidad) generados por el Compilador vs. otros lenguajes.
- Agregados extra al lenguaje como manejo de concurrencia para el procesamiento de los mensajes, matrices y vectores, tipos de datos de punto flotante, manejo de listas y colas, etc

Importante: No hay ningún inconveniente en utilizar librerías públicas, soluciones similares públicas, soluciones de foros, etc., pero es necesario aclarar, y enumerar cada una de ellas en la sección Referencias. No se aceptan bloques de código públicos implementados verbatim sin ningún tipo de análisis. Tampoco implementaciones que resuelven problemas que no están detallados (e.g. implementa un garbage collector sin explicar cómo). Tampoco hay inconveniente en que interactuen con otros grupos. *La diferencia entre plagio y ciencia es una referencia.*

5. Sugerencias

- Los terminales del lenguaje pueden ser más generales como IDENTIFICADOR. Utilizando LEX pueden después asociar exactamente qué conjunto de caracteres de entrada permiten formar un identificador válido del lenguaje.
- Pueden usar una tabla de símbolos donde almacenar a los objetos del programa.

6. Grupos

- Harambe
- Sudo!!
- Juan Lambvuschini
- ЯО
- `');DROP TABLES;`
- Z80

7. Fecha de entrega

El material a entregar puede ser o bien enviado por email a `rramele@gmail.com` en una carpeta ZIP rotulada con el nombre del grupo, o bien en un Branch o Tag de Git, Bitbucket u otro SCM similar. En el segundo caso, es requerido que envíen un email detallando el nombre del repositorio y el nombre del TAG que corresponde a la entrega.

Si el proyecto no compila, la entrega se considerará inválida.

El TP se debe entregar antes del día 24/11/2015 23:59 GMT -3.

8. Criterio de Corrección

- A. Informe: secciones, prosa, errores ortográficos, claridad en la exposición.
- B. Cumplimiento de consignas.
- C. Implementación del TP: investigación, creatividad, innovación, practicidad , definición y exposición de la idea
- subyacente, alineamiento con los temas vistos en la materia.

- D. Calidad en la presentación: armado del proyecto, documentación del código, estructura del código, scripts de automatización, test de regresión, etc.
- E. $+\alpha$: Puntos extras agregados más allá de los requerimientos del enunciado e incluso del trabajo mismo.

Cada item es puntuado del 1-10 y la nota del TP surge del promedio de todos los puntos.

9. Material de consulta

Se sugiere leer los documentos de las páginas:

1. Çompiladores, Principios, Técnicas y Herramientas",Aho, Sethi, Ullman, Addison Wesley.(El Libro del Dragón), capítulos 1,2 ,3 y 5,6,7.
2. Engineering a Compiler, Appeal, Ginsbur
3. Stevens.Advanced Programming in the UNIX Environment.1992
4. `http://peter.michaux.ca/articles/assembly-hello-world-for-os-x`
5. Free BSD Developer Handbook `http://www.freebsd.org/doc/en/books/developers-handbook/book.html#X86-SYSTEM-CALLS`
6. `http://web.cecs.pdx.edu/~apt/cs322/CompilingOO.pdf`
7. `http://www.doesnotunderstand.org/public/DSLRob2015`
8. `http://www.springer.com/gp/book/9781461446989`
9. `flex.sourceforge.net/manual/`