Etapa Uno: Analizador Léxico (Scanner)

PROF. KIRSTEIN GÄTJENS S.

"El coraje no siempre ruge.
A veces es la voz tranquila al final del día que dice:
'Lo intentaré de nuevo mañana'"
Sabiduría Popular

Instrucciones generales:

- Esta segunda etapa sirve para construir una biblioteca que haga el trabajo de análisis léxico del compilador.
- Hay que darle mantenimiento a todas las partes de la documentación. Hay que unificar las partes de documentación de la etapa cero y agregar las partes adicionales de esta etapa.
- Esta entrega es para el jueves 10 de abril a la medianoche al correo de kirstein.evaluaciones@gmail.com Será a este correo porque esta etapa se espera que la revise el asistente.
- Deben trabajar con su pareja de trabajo oficial. Daremos unos días a partir de hoy para considerar divorcios y nuevos matrimonios. No sean alcahuetas, es aceptable entregar el trabajo solos si la otra persona no trabajó absolutamente nada.
- Se debe entregar un compreso con todo lo solicitado ordenado por directorios en la raíz del compreso. Un directorio para pruebas, uno para los programas de ensamblador y otro para la documentación en PDF. Además un directorio con los fuentes del programa. Se espera un ejecutable que se pueda probar sin recompilar en la raíz del archivo compreso.
- Es importante que le den mantenimiento a la documentación. Voy a asignar un porcentaje de la nota de cada etapa para motivarlos a que lo hagan.
- La sección de pruebas debe crecer sustancialmente para no tener solo las pruebas generales de la etapa cero sino agregar un conjunto pruebas especializadas en las revisiones léxicas. Determinar todos los casos límite para cada familia de tokens y tener pruebas que los comprueben.
- El tamaño de las pruebas debe contemplar el tamaño del buffer usado para procesar el archivo. Debe poder usarse varios buffers en el proceso de algunas pruebas. Pueden usar un buffer más pequeño para eso (unos 128 bytes). Habrá una prueba especial que llamaremos la prueba gordita. Esta prueba debe ser un archivo de al menos 65kb. Pueden recurrir al copy paste para lograrla.

- Los Algoritmos de conversión de todos los tipos deben hacerse para esta etapa. Se les da un ejemplo de otros semestres, pero deben decidir los algoritmos de este lenguaje. Pueden reusar muchos de ese ejemplo, pero tomen una decisión pensada.

- En la documentación debe haber una sección dedicada a explicar los algoritmos de conversión. Se espera que las explicaciones incluyan ejemplos.

- En la documentación debe dedicarse una sección al autómata. Pueden hacer los dibujos por partes para no tener una "araña" poco legible. Eso si todas las partes deben estar conectadas al estado inicial

- Se debe hacer como parte de la documentación el listado de errores. Los errores deben numerarse.

- La gramática se debe mantener en la documentación y si en el transcurso del proyecto se nota que hay algo que arreglar se debe arreglar de una vez.

- Como parte de esta segunda etapa, se les solicita que programen en ensamblador de TASM todos los algoritmos de conversión. No se permiten usar las directivas que comienzan con punto como .stack o .data o .code o .etc. igual que hacíamos en Arquitectura. Deben recordar que se usa la pila para enviar y recibir los parámetros.

- Deben hacer seis programas en ASM, uno para cada tipo básico. Esto es las conversiones de todos los tipos a ese tipo en particular. Para poder probarlo lo que haremos será pedirle al usuario una entrada de cada tipo, convertirlo y desplegar la respuesta. Recuerden que las rutinas de E/S ya las tienen programadas.

- Cada programa en ensamblador se debe llamar con el nombre del tipo básico recortado a 8 letras en inglés.

- Todo programa de ensamblador que escriban (o lleguen a generar) debe tener al inicio entre comentarios una portada corta que lo identifique. Algo así como un miniacerca de.

- Subject: Etapa-Uno-Apellido1-Nombre1-Apellido2-Nombre2.RAR

  Donde los nombres cortos se ordenan alfabéticamente por apellido.

- El PDF se debe llamar con el mismo nombre que el rar.

- Los nombres de las carpetas internas deben ser:

- Pruebas

- ASM

- Documentación

- Gramática

- El pdf principal va afuera y no encarpetado.
- Solo uno de los dos estudiantes debe enviar la tarea.
- Si se envía más de una copia de la tarea recuerde que se toma en cuenta solo la que llegó de última que haya llegado a tiempo.
- Esta etapa va a valer por un valor de 18 resúmenes exactos.
- Como es usual, cualquier duda que NO involucre la solución de la tarea o parte de ella debe realizarse como un una consulta al grupo de Whatsapp, en caso contrario al correo del profesor, recuerde que no permitimos mensajes privados por W.
- La biblioteca a desarrollar para el scanner debe programarse en el lenguaje de alto nivel seleccionado para el proyecto (opciones: C, Java, Python, similares). Sean sensatos y escojan un lenguaje que dominen. No tomen este proyecto como una oportunidad para aprender un lenguaje nuevo.
- Las cuatro rutinas que deben programar de forma obligatoria son: Inicializar-scanner, Finalizar-scanner, DemeToken y TomeToken. Es posible que programen múltiples rutinas internas entre las cuales tenemos a demecaractere y tomecarcater.
- Es obligatorio para el proyecto realizar la biblioteca del scanner con un autómata modificado a como se explicó en clases. No pueden usar software de generación automática ni hacer el scanner alambrado.
- Una sección nueva de la documentación es la que documenta el autómata. Se espera que se explique de forma gráfica aunque separándolo en pequeñas secciones para que los diagramas tengan sentido.
- El listado de elementos del lenguaje debe crearse para esta etapa y ser una parte de la documentación. Se les da un ejemplo del listado de palabras reservadas del lenguaje viejo en Cabécar.
- Con el fin de poder probar la biblioteca del scanner se necesita escribir una pequeña aplicación. En este caso será un generador de muros de ladrillos (lo escogió la mayoría sin Chavarría).
- El muro de ladrillos es un archivo .HTML que despliega ladrillos de colores donde cada ladrillo es el lexema de un token de un programa que se escanea o un comentario o un error.

- El programa se espera que sea de línea de comandos y que reciba solo el nombre del archivo fuente a compilar. Bueno, en este caso a amurallar.

- Luego del muro de ladrillos se espera un despliegue de las estadísticas de las familias de los tokens, el número de líneas del programa, el número de caracteres del archivo de entrada, número de comentarios (de línea y de bloque) y cantidad de errores detectados. Si una estadística está en cero, no se debe desplegar.

- Es obligatorio utilizar recuperación de error y no detener el compilador por pánico. Por lo anterior debe haber una sección en la documentación donde expliquen en qué partes del scanner se necesitó hacer recuperación de error para evitar errores en cascada.

- Como parte de la documentación se deben incluir todas las partes de la documentación de la etapa cero ya corregidas y actualizadas.

¡¡Suerte!!