Definición general

Esta segunda etapa del proyecto conocida formalmente como parser o Análisis Sintáctico es, si se quiere la más importante del proyecto, pues de la gramática depende que el compilador se desempeñe de la mejor forma posible. De ahí que este proyecto se desarrolle con cuidado y lo mejor posible.
Para esta etapa del proyecto se debe entregar un programa que reciba un código fuente escrito en C y realice el análisis léxico y sintáctico correspondiente. Para esto se debe utilizar la definición del lenguaje aceptado por el Scanner junto con la gramática. Por lo tanto el programa debe hacerse en Java utilizando Jflex y Cup
Al finalizar el parseo el programa deberá desplegarle al usuario el resultado del Análisis léxico y sintáctico que se efectúo. Se espera que despliegue

1. Listado de errores léxicos encontrados: El programa debe desplegar una lista de todos los errores léxicos que se encontraron en el código fuente. Debe desplegar la línea en la que se encontró el error. Es importante que el programa pueda recuperarse del error y no desplegar los errores en cascada ni terminar de hacer el scaneo al encontrar el primer error.

2. Listado de errores sintácticos encontrados: El programa debe desplegar una lista de todos los errores sintácticos que se encontraron en el código fuente. Debe desplegar la línea en la que se encontró el error. Además, el mensaje de error debe ser lo más específico posible, con el fin de que el programador pueda llegar al error y corregirlo de forma eficiente. Es importante que el programa debe recuperarse del error y evitar no desplegar errores en cascada ni terminar de hacer el parseo al encontrar el primer error.

Descripción Detallada

Se les sugiere tomar en cuenta los siguientes aspectos para asegurar la completítud del programa.

- Las palabras reservadas que se deben incluir en la gramática son las siguientes: (Estas se utilizarán de la forma que se usan en C)

  b*reak, case, char, const, continue, default, do, else, for, if, int, long, return, short, switch, void, while*

- Se agregarán dos funciones: read y write. La primera puede o no tener un parámetro y la segunda siempre tendrá al menos un parámetro. El parámetro de la primera debe ser un identificador, y el del write puede ser cualquier expresión.

- Se podrá crear variables de los siguientes tipos: char, int, long, short. Así como declarar constantes. Se debe permitir una lista de variables de un tipo.

- La estructura del programa es la siguiente. Las globales podrían no venir.

  Globales o Constantes (pudiera no tener)

  Funciones (debe tener al menos una función)

- La estructura de las funciones es la siguiente

Función

```
Tipo f (tipo x, tipo y, ….  ) //Pudiera no tener parametros
{
        [ declaraciones de variables o constantes (pudiera no tener)

        [ cuerpo
}
```

Tipo puede ser int, char, long, short para las variables. Las funciones además de estos tipos, también pueden ser void.

Los operadores que se deben tomar en cuenta son los siguientes:

Aritméticos:	"++"	"--“	"="	"+"	"-"	"*”	"/"	"%"	"("	")"
	"+="	"-="	“*=”	"/="						
Booleanos:	"=="	">="	">"	<=	"<"	"!="	"||"	"&&"	"!"	

- En el cuerpo de una función puede venir expresiones, o estructuras de control, instrucciones como read, write, break, continue
- Las estructuras de control son: if, if-else, for, while, switch.
- Las expresiones pueden ser asignaciones, expresiones aritméticas o booleanas o literales.
- En el caso de las estructuras de control if, for y while en su expresión no deben validar que sea booleana, puede ser cualquier tipo de expresión incluso una asignación. En el caso de switch debe validar que la expresión es numérica.
- Se deben implementar buenos mensajes de error
- Recuerden que esta es una etapa muy importante del proyecto del curso, por eso entre más detallado y funcional esté, más facilidad van a tener en el resto del proyecto.

Documentación

Se espera que sea un documento donde especifique lo siguiente:

- Portada, índice, introducción
- Estrategia de Solución: Una explicación de como lograron solucionar el proyecto. Puede incluir consideraciones de diseño, lógica o cualquier aspecto que consideren importante para llegar a la solución.
- Análisis de Resultados: Deberá elaborar un listado de todas y cada una de las actividades y tareas que deben cubrirse a nivel funcional, para cada una de ellas debe aportar el porcentaje de realización y en caso de no ser el 100% debe justificarse.
- Lecciones aprendidas: Debe prepararse un listado de las lecciones aprendidas producto del desarrollo de la tarea programada. Las lecciones aprendidas pueden ser de carácter personal y/o técnico que involucre aspectos que han logrado un aprendizaje en temas de investigación, desarrollo de habilidades técnicas y habilidades blandas como trabajo en equipo, comunicación, forma de expresar ideas, entre otros.
- Casos de pruebas: se espera que definan claramente cada prueba, cuáles son los resultados esperados y cuáles fueron los resultados obtenidos. No es necesario que sean grandes pero deben evaluar la funcionalidad completa del programa.
- Manual de usuario: especificar como compilar y correr su tarea.
- Bitácora de trabajo durante las semanas de trabajo, incluyendo verificaciones realizadas (si existieran) de consultas realizadas con el profesor o asistente.

- Bibliografía y fuentes digitales utilizadas

Aspectos Administrativos

- Los grupos deben permanecer iguales a la primera etapa del proyecto.
- El trabajo se debe de entregar el día 4 de noviembre de 2022. Deben enviarlo subiéndolo al TEC-DIGITAL en el espacio respectivo.
- Recuerde que oficialmente no se recibirán trabajos con entrega tardía.
- Referencia:
    - http://www.cs.princeton.edu/~appel/modern/java/CUP/manual.html#intro
    - http://www2.cs.tum.edu/projects/cup/