UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERIA
ESCUELA DE SISTEMAS

Lenguajes Formales y de Programación 1
Escuela de vacaciones de junio 2022

Proyecto #2: Simple C v2.0

La empresa ha quedado satisfecha con el funcionamiento de Simple C V1.0, por lo cual le extiende la solicitud de una segunda versión con nuevas funcionalidades e implementada con herramientas de renombre como Lex y Yacc.

La empresa no requiere de un lenguaje funcional, se solicita un analizador léxico capaz de reportar si un código fuente pertenece o no a dicho lenguaje y un analizador sintáctico que pueda reportar errores sintácticos en un código fuente.

Importante: su aplicación será encargada de analizar la sintaxis del código fuente, en ningún momento tendrá en cuenta el resultado del código fuente.

Aplicación

Se solicita una aplicación de línea de comandos. Al ejecutar su aplicación esta solicitará los siguientes parámetros:

- Archivo del código fuente (.sc)
- Nombre de archivo de reporte (.html)

La aplicación se encargará de analizar el archivo de código fuente y de escribir los reportes con el nombre dado.

Gramática

La gramática del lenguaje debe ser documentada. La gramática debe contener:

- Alfabeto del lenguaje
  - Símbolos terminales
  - Símbolos no terminales
- Sintaxis del lenguaje
  - Precedencia de operadores
  - Producciones

Librería PLY

Se debe utilizar la librería PLY para la implementación del analizador léxico y sintáctico.

Analizador léxico

La implementación del analizador léxico se realizará con la herramienta Lex, brindada por la librería PLY. El analizador léxico se encargará de reportar la lista de tokens del código fuente y los errores léxicos encontrados.

Analizador sintáctico

La implementación del analizador sintáctico se realizará con la herramienta Yacc, brindada por la librería PLY. El analizador sintáctico se encargará de validar la sintaxis del código fuente, reportar el árbol sintáctico abstracto y los errores sintácticos encontrados.

Lenguaje

Resumidamente, las funciones de este lenguaje serán:

- El lenguaje será case insensitive
- Declaración y asignación de variables
- Operaciones aritméticas, lógicas y relacionales
- Estructuras condicionales
- Estructuras iterativas
- Sentencias de control de flujo
- Funciones del lenguaje
- Creación de funciones y métodos con y sin parámetros
- Retorno de funciones y métodos
- Llamada de funciones y métodos

La sintaxis detallada se presenta a continuación.

Comentarios

El lenguaje soporte comentarios de una línea y multilínea, de la siguiente manera:

```
// comentario de una línea

/*

        Comentario
        De varias
        Líneas

*/
```

Tipos de datos

El lenguaje contará con los siguientes tipos de datos:

Tipo de dato	Ejemplo
Int	67
Double	39.87

String	“Hola mundo”
Char	‘a’
Boolean	true

Identificadores

Los identificadores del lenguaje deben tener la siguiente estructura:

(**[UNA LETRA O GUIÓN BAJO]**)(**[CERO O MÁS LETRAS, GUIONES BAJOS O NÚMEROS]**)

Operaciones

El lenguaje cuenta con varias operaciones aritméticas, lógicas y relacionales.

Operación	Operador	Ejemplo
Suma	+	7 + 8
Resta	-	5 - 90
Multiplicación	*	17 * 0
División	/	4 / 3
Resto	%	91 % 2
Igualación	==	4 == 5
Diferenciación	!=	7 != 7
Mayor	>	8 > 0
Mayor igual	>=	-9 >= -90
Menor	<	87 < 16
Menor igual	<=	77 <= 4
AND	&&	true && true
OR	||	true || false
NOT	!	! false

Queda fuera del alcance de la aplicación validar los operandos de las operaciones.

Instrucciones

Las instrucciones pueden estar tanto dentro como fuera de funciones y métodos. Las instrucciones del lenguaje son:

- Declaración y asignación de variables
- Estructuras condicionales
- Estructuras iterativas
- Sentencias de control de flujo

- Declaración de métodos y funciones
- Retorno de métodos y funciones
- Llamada de métodos y funciones

Declaración y asignación de variables

Podemos asignar datos a variables de la siguiente manera:

```
// declaración

[TIPO_DATO] [ID] = [DATO] ;

// asignación

[ID] = [DATO] ;
```

Estructuras condicionales

El lenguaje contará con las siguientes estructuras condicionales:

```
// sentencia if

if ( [OPERACION] ) {

[INSTRUCCIONES]

}

// sentencia if else

if ( [OPERACION] ) {

[INSTRUCCIONES]

} else {

[INSTRUCCIONES]

}
```

Estructuras iterativas

El lenguaje contará con las siguientes estructuras iterativas:

```
// sentencia while

while ( [OPERACION] ) {

[INSTRUCCIONES]

}

// sentencia do-while

do {

[INSTRUCCIONES]

} while ( [OPERACION] ) ;
```

Sentencias de control de flujo

El lenguaje contará con las siguientes sentencias de control de flujo:

```
// sentencia break

break ;

// sentencia continue

continue ;
```

Funciones y métodos

Las funciones y métodos se declararán de la siguiente manera:

```
// método

void [ID] ( [PARAMETROS] ) {

[INSTRUCCIONES]

}


// función

[TIPO_DATO] [ID] ( [PARAMETROS] ) {

[INSTRUCCIONES]

}
```

// parámetros

(**[PARAMETRO]** ,) *

// parámetro

[TIPO_DATO] [ID]

Retorno de funciones y métodos

Las funciones pueden retornar de la siguiente manera:

// método

return ;

// función

return **[DATO] ;**

Llamada a funciones o métodos

Las funciones y métodos se llamarán de la siguiente manera:

// llamada

[**ID]** (**[ARGUMENTOS]**)

// argumentos

(**[ARGUMENTO]** ,) *

// argumento

[DATO]

Reportes

Los reportes serán archivos HTML a su creatividad.

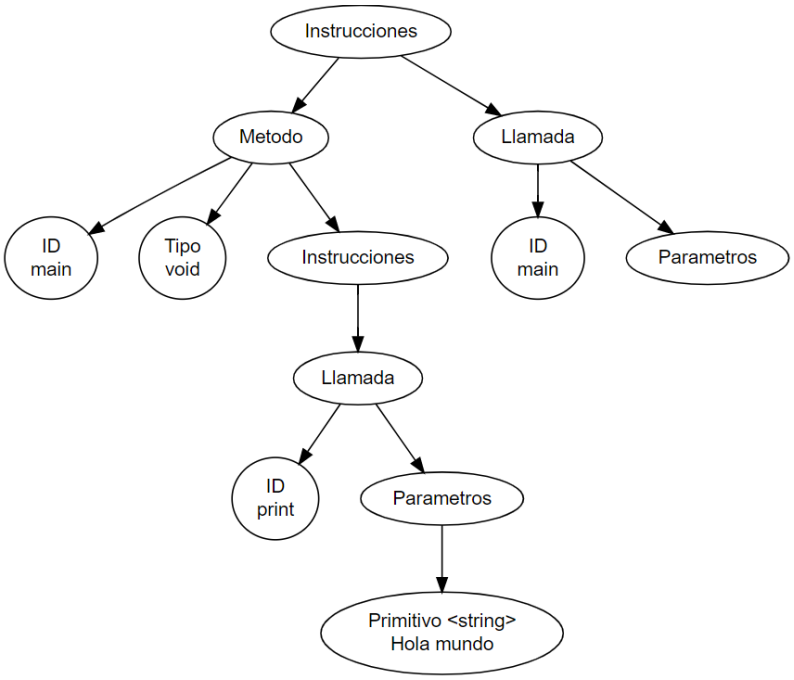
Reporte de tokens

Se deberá guardar como [nombre]-tokens.html, donde “nombre” es el nombre provisto en la línea de comandos. Este reporte contendrá los lexemas con su token correspondiente, así como el patrón de dicho token. Ejemplo:

Línea	Columna	Lexema	Token	Patrón
3	1	55	dato_int	-?[0-9]+

Reporte de AST

Se deberá guardar como [nombre]-ast.html, donde “nombre” es el nombre provisto en la línea de comandos. Se deberá generar un grafo con el AST obtenido del análisis sintáctico. Se recomienda el uso del lenguaje graphviz para este reporte. Este reporte solo será generado si no existen errores sintácticos.



Reporte de errores

Se deberá guardar como [nombre]-errores.html, donde “nombre” es el nombre provisto en la línea de comandos.

Línea	Columna	Tipo	Mensaje
3	1	Léxico	No se reconoce el lexema 3main
16	8	Sintáctico	Se esperaba “;” pero se obtuvo “hola_mundo”

Manuales

Se requiere la creación de un manual de usuario y un manual técnico, ambos en formato Markdown dentro del repositorio.

El manual de usuario deberá guiar a un usuario en el uso de la aplicación de línea de comandos. El manual técnico deberá contener una explicación técnica del desarrollo de su aplicación.

Consideraciones

- La aplicación deberá ser desarrollada en Python con uso exclusivo de la herramienta PLY.
- La aplicación debe ser amigable con el usuario.
- Para la calificación se utilizarán archivos de código fuente escritos por su auxiliar, se recomienda la creación de archivos de código fuente propios para probar todas las funcionalidades.
- Se deberá trabajar en un repositorio privado en GitHub e invitar al auxiliar de colaborador: pabloc54.
- El nombre del repositorio debe ser: LFP-VJ-**carnet**-P2
- Copias totales o parciales serán detectadas y reportadas con la escuela para su sanción correspondiente.

Entrega

- La fecha de entrega es: Miercoles 29 de junio de 2022 a las 23:59.
- La entrega es individual.
- Se deberá entregar el enlace al repositorio en Uedi.
- Se calificará únicamente lo subido al repositorio hasta la fecha de entrega.
- El proyecto será calificado en su computador, tener preparado todo al momento de la calificación pues no se darán nuevos horarios.
- Se tendrá un tiempo de calificación de 15 min por alumno.
- Entregas tarde tendrán una nota de cero puntos.