# Compiladores 2023-2
# Facultad de Ciencias, UNAM
# Proyecto Final

Lourdes del Carmen González Huesca  Juan Alfonso Garduño Solís
Braulio Aaron Santiago Carrillo

12 de Mayo de 2022
**Fecha de Entrega: 12 de Junio**

## Introducción.

Este es el proyecto final y vale el $20\,\%$ de la calificación final de la materia. Consiste en una serie de extensiones al compilador obtenido de las prácticas, puedes elegir libremente (a excepción de una) cuáles quieres implementar e incluso si haces ejercicios adicionales puedes sacar más de 10 (la calificación excedente se agregará a la suma de las calificaciones de las prácticas antes de hacer el promedio, haciendo todos los ejercicios del proyecto puedes obtener hasta ***+7 puntos*** en las prácticas, **siempre y cuando estén correctamente implementados**).

## Ejercicios.

El único proceso obligatorio es uno que traduzca un árbol de sintaxis abstracta obtenido a partir la aplicación de todos los procesos definidos en las prácticas durante el semestre a código Java. El código que produce dicho árbol se obtendrá de un archivo arbitrario *example.jly* con código en `Jelly` para y el resultado se escribirá en un archivo `example.java`.
Este proceso vale ***7 puntos*** y para juntar el resto de tu calificación puedes elegir entre los siguientes ejercicios:

1.- ***(1 punto)*** Extiende el lenguaje para que admita el ciclo `for`.
Recuerda que el `for` es azúcar sintáctica del ciclo `while`.

3.- ***(2 punto)*** Permite la definición y uso de variables globales.
Como en otros lenguajes, un archivo de Jelly podría tener la declaración de variables globales, escritas en el nivel superior y fuera de cualquier definición de método. Por ejemplo:

```
lon:int=90
i : int = -2
truent:bool=false
calificaciones:int[]
```

El código anterior describe las variables enteras y booleanas tienen que ser inicializadas, sin embargo los arreglos globales pueden ser declarados pero no inicializados.

4.- ***(2 puntos)*** Escribe un script para ejecutar el código compilado al lenguaje objetivo.
Para este ejercicio, la finalidad es que seas capaz de que a través de la terminal y una sola instrucción se ejecute todo el proceso de compilación.

5.- ***(2 puntos)*** Extiende el lenguaje desde la gramática para que nuestro lenguaje implemente cadenas, asi mismo permite al lenguaje imprimir con `println(e)`.
El primer paso es considerar la siguiente regla adicional en la definición de tipos:

$$\tau ::= \texttt{string}$$

Y para usar `println` considera las reglas estáticas:

$$\frac{\Gamma \vdash e : \texttt{int}}{\Gamma \vdash \texttt{println}(e) : \texttt{unit}} \qquad \frac{\Gamma \vdash e : \texttt{bool}}{\Gamma \vdash \texttt{println}(e) : \texttt{unit}} \qquad \frac{\Gamma \vdash e : \texttt{string}}{\Gamma \vdash \texttt{println}(e) : \texttt{unit}}$$

6.- ***(3 puntos)*** Implementa errores de manera detallada en cada proceso de la compilación.
En este momento cuando tenemos un error de sintaxis la manera en la que se nos informa (si es que se implementó el ejercicio extra de la práxtica 3) es con un mensaje de este estilo:

```
error: no fue posible procesar un token
    en: SMC
```

Pero este mensaje es muy genérico porque pueden haber muchas instancias de ese token en el código que buscamos compilar, por eso para este ejercicio deberás modificar desde el parser y el lexer para que se informe dónde está el token que genera el problema de la siguiente manera:

```
error: no fue posible procesar un token
    linea : columna SMC
```

Donde linea y columna obviamente son la posicion del lexema en el código fuente que está provocando el error. Además, de que los errores que arroje el sistema de tipos deben informar el problema que se produjo detalladamente.

## Entrega.

Igual que con las prácticas, la solución del proyecto se va a entregar en github classroom.
El repositorio deberá encontrarse bien organizado y con el siguiente contenido:

- Archivo `readme.pdf` explicando detalladamente el desarrollo del proyecto, la gramática del lenguaje resultante de su proyecto, los ejercicios que eligieron, como resolvieron cada uno y las instrucciones para ejecutar su proyecto.
- Un ejemplo para corroborar el funcionamiento de **cada** ejercicio seleccionado.
- Archivos que conforman el compilador con un nombre adecuado a su función.

**Cada especificación no respetada se penalizará con un punto menos** sobre la calificación del proyecto y las siguientes son las razones por las que tu proyecto puede ser **evaluado con cero**:

- Existe un error de ejecución o interpretación en tu proyecto.
- Existe copia de algún ejercicio o práctica en el proyecto.

**Es imposible que entreguen con retraso ya que perderán el acceso de escritura a su repositorio el día de entrega a las 23:59.**