# EXAMEN FINAL

# TAREA

## Gramática de libre contexto y comprobador de tipos

**Objetivo:** Diseñar una gramática de libre contexto y construir su comprobador de tipos.

**Instrucciones Generales:**

1. En grupos de hasta seis estudiantes realizarán un trabajo de aplicación de los conceptos de compiladores estudiados en la III y IV Unidad.
2. Cada grupo presentará un informe en digital (Word o pdf), que **será subido al aula virtual por uno de los integrante**s.
3. La presentación grupal e individual será hasta el miércoles 06 de julio a las 5:50pm.
4. El informe debe incluir las siguientes partes como mínimo:
    - Caratula (Universidad, Facultad, Escuela, Titulo de la monografía, Curso, Autores, Fecha)
    - Resumen
    - Índice
    - Contenido:
        a) Componentes léxicos (tokens) del lenguaje de programación (Debe indicar los componentes y para cada uno su expresión regular).
        b) Cinco ejemplos correctos de programas fuente del lenguaje diseñado (Cada ejemplo debe ser de al menos 10 líneas).
        c) Gramática de libre contexto del lenguaje en bison (Debe ser diferente a la que se muestra en el ejemplo).
        d) Comprobador de tipos (se escribe sobre la estructura de la GLC).
    - Conclusiones
    - Referencias bibliográficas (según modelo APA, opcional)

**Instrucciones específicas:**

1. Siguiendo como mínimo las características de un lenguaje de alto nivel, que se describen más adelante, el grupo diseña una gramática de libre contexto y sobre la estructura de esta construye su comprobador de tipos.

    **CARACTERISTICAS DEL LENGUAJE**

    El lenguaje de programación tendrá las siguientes características:
    a. Permita los siguientes tipos de datos:
        - enteros

- reales
- cadenas de caracteres

b. Incluya un conjunto de operadores (aritméticos y de relación) que permitan construir expresiones utilizando variables y/o constantes de los tipos de datos mencionados.
c. Debe incluir las siguientes sentencias de control de flujo:
    - condicional (if...else)
    - bucles (for y while)
d. Admite instrucciones de entrada/salida (Lectura/escritura) de los distintos tipos de datos.
e. Admite la instrucción de asignación de expresiones aritméticas.
f. Otras sentencias importantes: bloque y otras que se puedan considerar importantes.

**RUBRICA PARA CALIFICAR LA TAREA (EXAMEN FINAL)**

| | **PUNTAJE** | | | | |
|---|---|---|---|---|---|
| **CATEGORIA** | **4** | **3** | **2** | **1** | **PUNTAJE** |
| **Calidad de la Gramática** | La gramática tiene todas sus reglas correctamente diseñadas | La gramática tiene una regla incorrectamente diseñada. | La gramática tiene dos reglas incorrectamente diseñadas. | La gramática tiene tres o reglas incorrectamente diseñadas. | |
| **Amplitud de la Gramática** | La gramática incluye todas las característica del lenguaje solicitadas | La gramática no incluye una característica del lenguaje solicitadas | La gramática no incluye dos característica del lenguaje solicitadas | La gramática no incluye tres o más característica del lenguaje solicitadas | |
| **Calidad del comprobador de tipos** | El comprobador de tipos tiene todas las comprobaciones correctamente construidas | El comprobador de tipos tiene una comprobación incorrectamente construida. | El comprobador de tipos tiene dos comprobaciones incorrectamente construidas | El comprobador de tipos tiene tres o más comprobaciones incorrectamente construidas | |
| **Amplitud del comprobador de tipos** | La gramática incluye todas las comprobaciones necesarias | La gramática no incluye una comprobación necesaria | La gramática no incluye dos comprobaciones necesarias | La gramática no incluye tres o más comprobaciones necesarias | |
| **Puntualidad** | Envía la tarea puntual (2) | | Envía la monografía Impuntual (0) | | |
| **Partes** | Incluye todas las partes mínimas solicitadas (2) | | Incluye algunas de las partes mínimas solicitadas (1) | | |
| | | | | **NOTA** | |

**Ejemplo de GLC**

```
%{

int yystopparser=0;
%}

%token MAIN STDIO CONIO CTYPE VOID IDENT INT FLOAT CHAR READ WRITE WHILE
%token MAYIGUAL MENIGUAL IGUALA DIFERENTE ENTERO REAL CADENA

%start MiniC

%%

MiniC          : Libreria TipoPrograma MAIN '(' VOID ')' '{' Cuerpo '}'
               ;
Libreria       :STDIO Libreria
               |CONIO Libreria
               |CTYPE Libreria
               |/*vacio*/
               ;
TipoPrograma   : INT
               | VOID
               ;
Cuerpo         : Variables Sentencias
               ;
Variables      : TipoVariable IDENT MasIdent ';' Variables
               | /*vacio*/
               ;
TipoVariable   : INT
               |FLOAT
               |CHAR
               ;
MasIdent       : ',' IDENT MasIdent
               |/*vacio*/
               ;
Sentencias     : Asignacion Sentencias
               |Leer Sentencias
               |Escribir Sentencias
               |Mientras Sentencias
               |Asignacion
               |Leer
               |Escribir
               |Mientras
               ;
Leer           : READ '(' IDENT MasIdent ')' ';'
               ;
Mientras       : WHILE '('  Condicion ')' '{' Sentencias '}'
               ;
Condicion      : Expresion OpRelacional Expresion
```

```
                ;
OpRelacional    : '>'
                |'<'
                |MAYIGUAL
                |MENIGUAL
                |IGUALA
                |DIFERENTE
                ;
Expresion       : Expresion '+' Termino
                | Expresion '-' Termino
                |Termino
                ;
Termino         : Termino '*' Factor
                |Termino '/' Factor
                | Factor
                ;
Factor          : ENTERO
                |REAL
                | IDENT
                | '('Expresion')'
                ;
Asignacion      : IDENT '=' Expresion ';'
                ;
Escribir        : WRITE '(' Contenido ')' ';'
                ;
Contenido       : Expresion MasContenido
                |CADENA MasContenido
                ;
MasContenido    : ',' Contenido
                |/*Vacio*/
                ;
```