****

**“Compilador para un Lenguaje Propio”**

Curso: Fundamentos de Cs. De la ComputaciónIntegrante: Javier Catalán Vargas  
Profesor: José Luis Veas  
Fecha de Entrega: 17/06/2025

1. **Introducción**

El presente proyecto consiste en el diseño e implementación de un compilador para un lenguaje propio. El objetivo fue crear un lenguaje capaz de realizar operaciones aritméticas básicas, exponentes mediante bucles while, operaciones de comparación y sentencias de control de flujo como if, else y while, todo esto manejado desde entrada por consola.

**2. Objetivos**

* Implementar un analizador léxico que identifique correctamente tokens.
* Construir un analizador sintáctico funcional que soporte expresiones matemáticas y estructuras de control.
* Delegar los cálculos a funciones especializadas para cumplir el principio de modularidad.

**3. Desarrollo del Proyecto**

**3.1 Análisis Léxico**

Gracias a la utilización de “Flex” se definió el archivo “lenguaje.l” para reconocer números, operadores aritméticos (+, -, \*, /, ^), paréntesis y palabras clave (if, else, while). Los espacios y tabulaciones se ignoran y se genera un mensaje en caso de detectar caracteres inválidos. Por ejemplo, se pueden identificar correctamente:

1. “+” como un token de SUMA.
2. “if” como una palabra clave IF.
3. “2.5” como un número decimal (NUM)
   1. **Análisis Sintáctico**

Usando bison se construyó el archivo lenguaje.y, donde se definen las reglas gramaticales para:

* Operaciones: suma, resta, multiplicación, división, potenciación.
* Comparaciones: ==, !=, <, >, <=, >=.
* Sentencias de control: if, if-else, while.
  1. **Funciones Auxiliares**

Se definieron funciones en C como:

* resolver\_suma, resolver\_resta, etc., cada una imprime y retorna el resultado.
* resolver\_potencia, que usa un bucle while para calcular la potencia.
  1. **Ejecución**

El archivo main.c inicia el parser con yyparse(). Se puede ingresar expresiones directamente por consola.

Ejemplo:

./compilador

4 + 5;

Resultado: 9

if (2 < 3) (10 / 2);

Resultado: 5

1. **Resultados Obtenidos**

* El compilador procesa correctamente expresiones con resultados decimales.
* Se detectan y reportan errores sintácticos (ej: if (3 < 2); sin bloque).
* Las funciones especializadas imprimen el resultado con claridad.

1. **Conclusiones**

* Se cumplieron los objetivos funcionales propuestos para el lenguaje.
* El uso de flex y bison permitió estructurar de forma modular el compilador.
* A pesar de no implementar un AST formal, se logró un flujo sintáctico correcto.

1. **Archivos del Proyecto**

* lenguaje.l: analizador léxico.
* lenguaje.y: analizador sintáctico.
* main.c: entrada principal.
* Makefile: automatización de compilación.