

# Calculadora BISON

Leidy Valeria Larrea Guerrero

## 1. Ejercicios y pruebas

### 1.1 Instalación

```
tulipan1637@tulipan1637-TUF-GAMING-FX504GE-FX80GE:~  
→ ~ sudo apt-get update  
sudo apt-get install bison  
[sudo] password for tulipan1637:  
Get:1 file:/var/cuda-repo-ubuntu2204-11-7-local InRelease [1.575 B]  
Get:1 file:/var/cuda-repo-ubuntu2204-11-7-local InRelease [1.575 B]  
Hit:2 https://dl.google.com/linux/chrome/deb stable InRelease  
Hit:3 http://co.archive.ubuntu.com/ubuntu jammy InRelease  
Hit:4 http://co.archive.ubuntu.com/ubuntu jammy-updates InRelease  
Hit:5 http://co.archive.ubuntu.com/ubuntu jammy-backports InRelease  
Hit:6 https://packages.microsoft.com/repos/code stable InRelease  
Hit:7 http://archive.lambdalabs.com/ubuntu jammy InRelease  
Hit:8 http://security.ubuntu.com/ubuntu jammy-security InRelease  
Reading package lists... Done  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
bison is already the newest version (2:3.8.2+dfsg-1build1).  
0 upgraded, 0 newly installed, 0 to remove and 27 not upgraded.  
→ ~
```

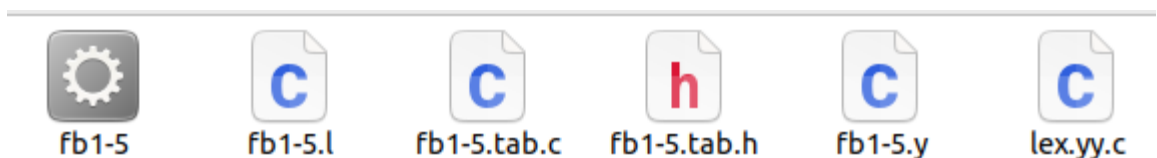
### 1.2 Descarga de archivos de prueba

<div> <div>+ Nuevo ▾</div> <div> <div>↑ Cargar ▾</div> <div> <div>Editar en vista de cuadrícula</div> <div>...</div> </div> </div> <div>Todo</div> </div>			
Documentos > General <div>Nombre</div> Example_Bison > prueba			
<div>📄</div> Nombre ▾	Modificado ▾	Modificado por ▾	
<div>📄</div> fb1-5.l	2 de agosto	Joaquin Fernandez	
<div>📄</div> fb1-5.tab.c	2 de agosto	Joaquin Fernandez	
<div>📄</div> fb1-5.tab.h	2 de agosto	Joaquin Fernandez	
<div>📄</div> fb1-5.y	2 de agosto	Joaquin Fernandez	
<div>📄</div> lex.yy.c	2 de agosto	Joaquin Fernandez	

### 1.3 Ejecución

```
./fb1-5
→ Downloads cd Example_Bison/prueba
→ prueba ls
fb1-5 fb1-5.l fb1-5.tab.c fb1-5.tab.h fb1-5.y lex.yy.c
→ prueba ./fb1-5
> 45+(45-89)*2
= -43
> 1-90
= -89
> 4673+8941*89-5/(527/8*2-(32+6))/6
= 800422
> 
```

## 2. Explicación



### Tokenización

/home/tulipan1637/Downloads/Example\_Bison/prueba/fb1-5.l

El lexer o analizador léxico se encarga de dividir la entrada en componentes significativos llamados tokens. Cada vez que el lexer identifica un patrón en la entrada, genera el token correspondiente y lo pasa al siguiente paso en el proceso de análisis.

"+" se convierte en el token **ADD**.

"-" se convierte en el token **SUB**.

"\*" se convierte en el token **MUL**.

"/" se convierte en el token **DIV**.

"|" se convierte en el token **ABS**.

"(" se convierte en el token OP (**paréntesis de apertura**).

")" se convierte en el token CP (**paréntesis de cierre**).

[0-9]+ se convierte en el token NUMBER, y el valor numérico se almacena en **yyval**.

\n se convierte en el token EOL (**fin de línea**).

"//".\* se utiliza para ignorar comentarios.

[ \t] se ignoran los espacios en blanco.

## **Análisis**

El archivo fb1-5.y define las reglas gramaticales para analizar las expresiones matemáticas.

- **calclist**: Esta regla define la lista de cálculos. Puede ser una lista de expresiones seguidas por EOL o una lista vacía. Cada vez que se encuentra una expresión seguida de EOL, imprime el resultado de la expresión.
- **exp**: Define cómo se construyen las expresiones. Una expresión puede ser un factor, o puede ser una expresión con operaciones de adición o sustracción.
- **factor**: Un factor puede ser un term, o puede involucrar multiplicación o división.

- **term:** Un term puede ser un número, una expresión con valor absoluto o una expresión entre paréntesis.

**fb1-5.tab.c** es un archivo generado por Bison, incluye el código necesario para implementar el parser. Este código es responsable de leer la entrada, aplicar las reglas gramaticales definidas en el archivo fb1-5.y, y construir un árbol de sintaxis para la entrada.

#### **Estructura:**

- **Definiciones de Tokens:** Incluye definiciones para los tokens que Bison espera, basadas en las declaraciones en el archivo .y.
- **Funciones de Parsing:** Implementa la función `yyparse()` que realiza el análisis sintáctico.
- **Manejo de Errores:** Contiene funciones para manejar errores de análisis.

### **3. Evaluación**

**3.1 Árbol de sintaxis:** Antes de evaluar una expresión, se construye un árbol de sintaxis a partir de la entrada. Este árbol es una representación jerárquica de la expresión basada en las reglas gramaticales. Cada nodo en el árbol representa una operación o un valor, y las ramas representan los operandos o sub-expresiones. En el código generado por Bison (y el archivo de Bison en particular), cada regla de gramática define cómo se debe calcular el valor de las expresiones.

**3.2 Reglas Gramaticales:** Cada regla en el archivo .y define cómo se combinan los tokens para formar expresiones válidas y cómo se deben evaluar

Aquí, `$$` representa el valor de la expresión resultante, y `$1`, `$2`, `$3` representan los valores de los operandos y operadores. Las expresiones son evaluadas usando las operaciones definidas en las reglas.

**3.3 Implementación del Evaluador:** El evaluador recorre el árbol de sintaxis (implementado en el archivo .tab.c) y aplica las operaciones aritméticas definidas en las reglas gramaticales.

yyparse() se encarga de ejecutar el análisis y evaluación. Los resultados de las expresiones se imprimen en la salida estándar.