**Trabajo práctico N° 3**

***SSL Lenguajes***

| **Nombre y Apellido** | **Legajo** |
| --- | --- |
| Stefano Alejandro Gassmann | 208.380-2 |

***Índice:***

* ***Resolución***
  + *Planteo inicial*
  + *Funcionamiento del programa*
  + *Ejemplo 1*
  + *Ejemplo 2*

**RESOLUCIÓN DEL PROBLEMA**

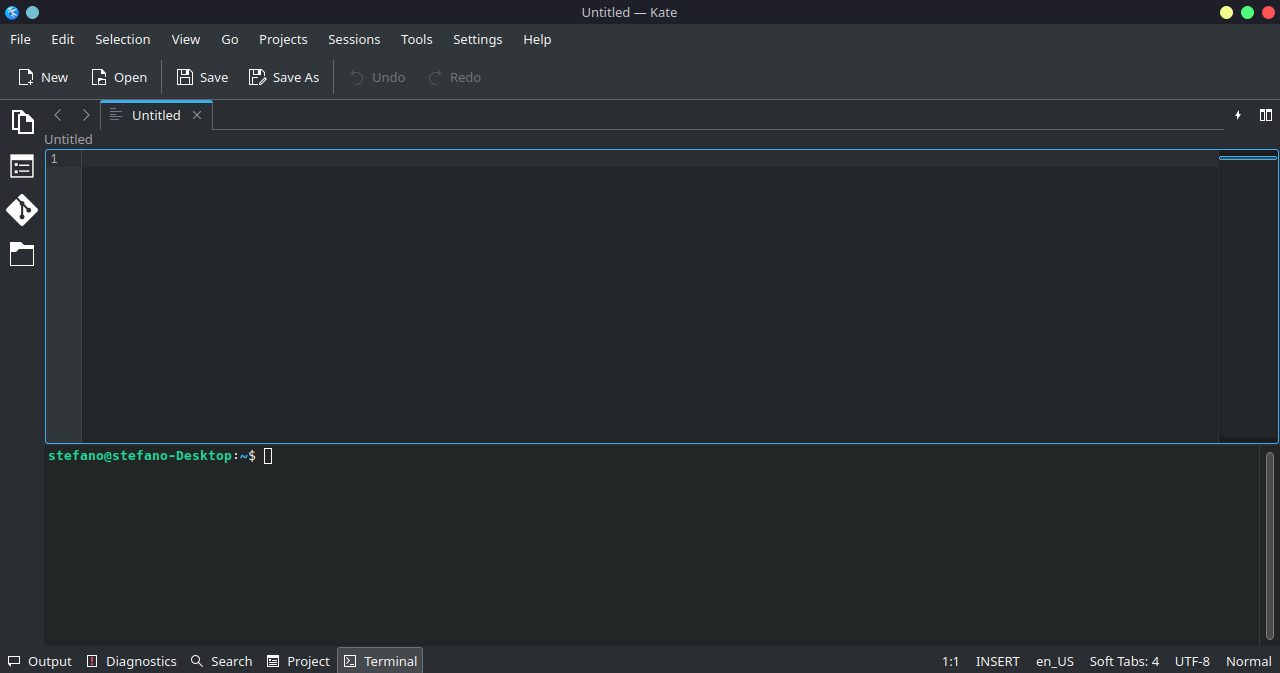
***Planteo inicial:***

Para iniciar este proyecto, procedí a instalar las herramientas esenciales necesarias para ejecutar **Flex** en mi máquina con sistema operativo ***Ubuntu 23.10(LTS)***.

Descargué la imagen oficial desde los repositorios, marcando así el primer paso en la configuración de mi entorno de desarrollo. **Flex** es una herramienta fundamental para el análisis léxico en la construcción de compiladores y será muy útil a lo largo del proyecto.

Tras completar la instalación y configuración exitosa de **Flex**, el siguiente paso es la instalación de **Bison**(un generador de analizadores sintácticos de propósito general que convierte una descripción para una gramática independiente o libre de contexto en un programa en ***C*** que analiza esa gramática), sin embargo, este proceso resultó un poco más desafiante debido a las dependencias adicionales requeridas. Luego de dedicar unos minutos a la búsqueda en la web, logré instalar **Bison** de manera correcta, superando así los obstáculos asociados con las dependencias y avanzando en la configuración de mi entorno de desarrollo.

Opté por utilizar un editor de texto enriquecido llamado **Kate**. Este bloc de notas proporciona una interfaz amigable y funcional para facilitar la escritura y edición de código. **Kate** incluye una terminal interactiva que posibilita la ejecución directa de comandos sin necesidad de recurrir a programas externos, esta funcionalidad adicional me permitió una mayor versatilidad y agilidad en la ejecución de tareas.

Vista del entorno de desarrollo, utilizando la herramienta Kate

Inicié el desarrollo del programa utilizando el archivo con el nombre **'flex.l'** (puede encontrar el enlace adjunto). Este archivo sirvió como punto de partida para la implementación, estableciendo las bases del análisis léxico en mi proyecto.

Después de completar la implementación en **Flex**, procedí a la creación del programa en **Bison** con el nombre **'bison.y'**. Realicé pruebas exhaustivas para garantizar el correcto funcionamiento del sistema. Siguiendo la práctica común en el ámbito informático, automatizar estos procesos de prueba para mejorar la eficiencia y asegurar la consistencia del programa en desarrollo.

Para llevar a cabo dicha automatización, se empleó la Terminal/Consola de la herramienta ya mencionada.

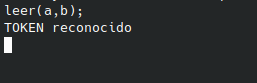


***Funcionamiento del programa:***

Al comenzar la ejecución del programa, se introduce una sentencia/instrucción perteneciente al lenguaje de programación Micro, en este caso utilizaremos la sentencia “**leer(a,b)**”.



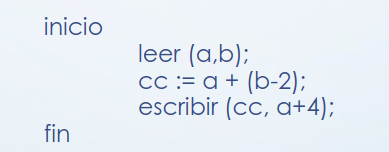
Al presionar enter, el programa no informa que se ha reconocido y es válida la instrucción.

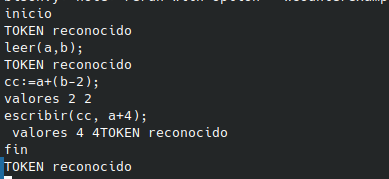


El programa se queda a la espera de más sentencias para ser evaluadas.

***Ejemplo1:***

Se toma el programa dado en la PPT de la clase n°13 que tiene la siguiente estructura:





Cómo se muestra, todo el programa se ha reconocido correctamente, e incluso se destacan algunos elementos importantes del mismo.

***Ejemplo2:***

Se toma un programa inventado que tiene la siguiente estructura:

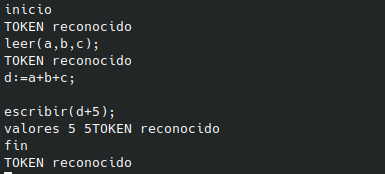
inicio

leer(a,b,c);

d:=a+b+c;

escribir(d+5);

fin



Nuevamente todos lo Tokens son reconocidos y el programa se ejecuta correctamente.