|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
| **Nombre de la materia** | | | **Curso:** |
| Sintaxis y Semántica de los Lenguajes | | | K2054 |
| **ALUMNOS** | **Nombre y apellido** | | |
| Valentin isolabella  Constanza sabino  Juan cruz sánchez busatto  Juan manuel moré  Alejandro país  Bruno maximiliano chvapil | | |
| **Trabajo Practico Nº4** | | | |
| **Profesora:** | | Roxana Leituz | |

Contenido

[Consigna 2](#_Toc182145421)

[Resolución 3](#_Toc182145422)

[Link repositorio github 3](#_Toc182145423)

[Código Correcto 3](#_Toc182145424)

[Código – Error Léxico 3](#_Toc182145425)

[Código – Error Sintáctico y Error Semántico 5](#_Toc182145426)

# **Consigna**

**Trabajo Práctico N° 4 – Análisis del compilador de micro realizado en C**

1. Compilar el Código entregado en clase.
2. Preparar 3 programas en micro: 1 correcto, 1 con error léxico, 1 con error sintáctico y semántico.
3. Analizar cómo funciona el análisis de cada uno de los errores expresando con sus palabras porque se produce el error, no relatando las líneas del código sino con lenguaje natural.

Lineamientos de entrega:

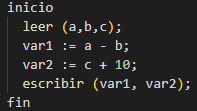
* Carátula con todos los integrantes
* Ejecutable
* Documento con
  1. pantallas mostrando que el programa funciona
  2. análisis de los errores solicitados
* Archivos anexos: se deben incluir los archivos que se utilicen en las pruebas
* Grupo: el mismo de todo el año
* Caso de copia: desaprueban la materia y deberán recuperar en marzo perdiendo por completo la posibilidad de promoción.
* El trabajo para darse por cumplido debe entregarse con todos los ítems desarrollados.
* Se entrega por campus una copia por grupo.

# **Resolución**

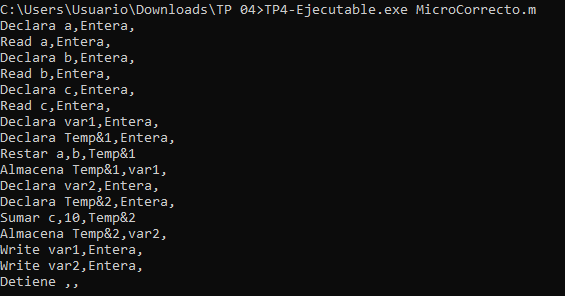
# **Link repositorio github**

<https://github.com/AlePais/SSL-2024/tree/557dce4b9df28d731c18c5fb13fa3f0056f94866/TP%2004>

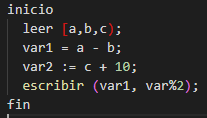
## **Código Correcto**



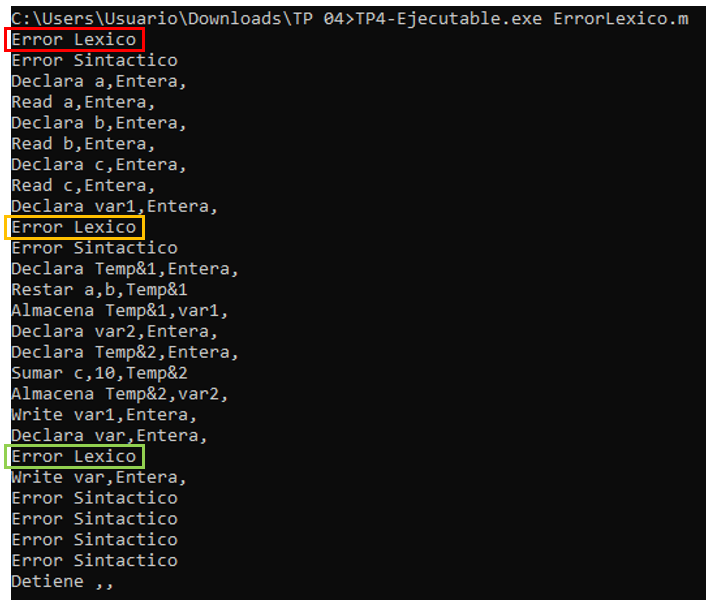
**Ejecución del Programa**



## **Código – Error Léxico**



**Ejecución del Programa**



**Errores**

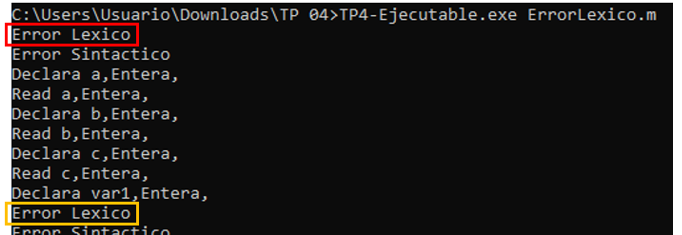
* Carácter de Puntuación: Se usó el corchete [ en vez del token Paréntesis Izquierdo.
* Carácter de Asignación: Se usó = como asignación en vez del token de Asignación :=
* Identificador: Se usó el caracter % en el nombre de una variable.

**Análisis Léxico**

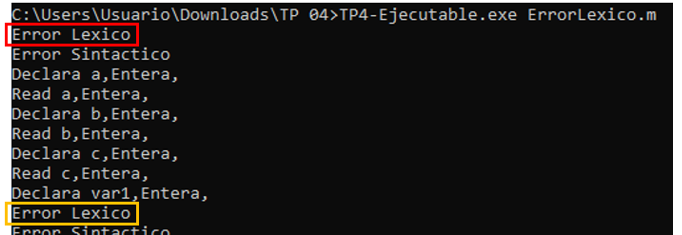
El análisis léxico de un programa fuente (en este caso en leguaje Micro) es llevado a cabo por el Scanner (analizador léxico). Este recibe un flujo de caracteres, y va reconociendo tokens. Está implementado como un AFD, que reconoce el LR que forma los tokens de Micro.

De este modo, comenzado el análisis del código, el Scanner recibirá carácter por carácter los lexemas *inicio*, luego *leer*, y los reconocerá como tokens válidos. Pero luego, en el flujo de caracteres, recibirá el carácter corchete izquierdo [ el cual no será reconocido como un token válido.

Como se mencionó, el Scanner está implementado como un AFD, con su correspondiente Tabla de Transiciones. Cuando lea el carácter [, partiendo del estado inicial 0, el autómata se pondrá en un estado de rechazo e informará que se halló un Error Léxico.



Continuando con el análisis, luego de un identificador y de un espacio en blanco, el Scanner recibe el carácter igual =. Nuevamente no reconocerá este lexema como un token válido. Esto se debe a que en su tabla de transición, si partimos del estado inicial y se consume un =, el autómata se coloca en un estado de rechazo.

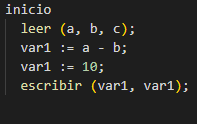


El siguiente y último error que detecta se da cuando recibe el carácter % entremedio del nombre de una variable. De acuerdo a la tabla de transición del AFD, se irán consumiendo los caracteres del lexema *var*, y al llegar al carácter %, el autómata devuelve un estado de aceptación en el cual reconoce a un Identificador (var).

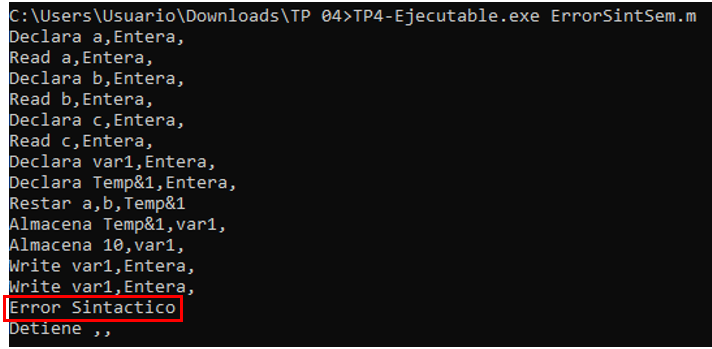
En este caso, el % fue tomado como un centinela, que indica el final de un token. Luego de esto, es devuelto al flujo de caracteres con un ungetc para ser analizado. En este punto, el autómata parte del estado inicial, consume el %, se coloca en Estado de Rechazo.



## **Código – Error Sintáctico y Error Semántico**



**Ejecución del Programa**



**Error Sintáctico**

* Falta de palabra reservada *fin* para indicar el fin del programa.

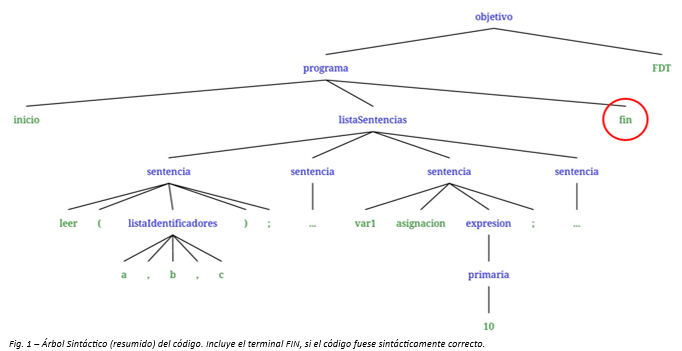
**Análisis Sintáctico**

El encargado del análisis sintáctico es el Parser (analizador sintáctico), implementado de forma Top-Down (es decir, es de tipo ASDR). El mismo consiste en una serie de procedimientos (PAS), que revisan que el código fuente cumpla con una serie de reglas, establecidas a partir de una gramática GIC.

Al realizar el análisis sintáctico, el parser ejecuta el procedimiento Objetivo(), que consiste en llamar al procedimiento Programa() y luego a la función Match(FDT). Esta última hace avanzar al Scanner para que le entregue al Parser el siguiente token.

En cuanto al Procedimiento Programa(), el Parser primero llama a Match(t) esperando recibir INICIO; luego llama al procedimiento ListaSentencias(). Hasta este punto no haya ningún problema.

Cuando llama nuevamente a la función Match(), el Parser espera recibir el token FIN. Pero en su lugar, recibe el token FDT. Esto se debe a que el código fuente no incluye la palabra reservada *fin* que indica el fin del programa. Es decir, el código fuente termino, y el ultimo carácter que reconoce el Scanner como token es el FDT, que hace de carácter centinela. Este carácter tiene un estado final propio en la Tabla de Transición del AFD del Scanner.

**

Como el Parser no recibe el token esperado, informa que se produjo un error sintáctico.

**Error Semántico**

* Declaración de dos variables con igual nombre var1.

**Análisis Semántico**

El Analizador Semántico es el encargado de generar las instrucciones para la máquina. Esto lo realiza por medio de rutinas semánticas (implementadas como funciones o procedimientos), incluidas dentro de los Procedimientos del Parser. Estas rutinas realizaran acciones para el chequeo semántico del código, la creación de registros semánticos, o la generación de las instrucciones mencionadas.

(…)