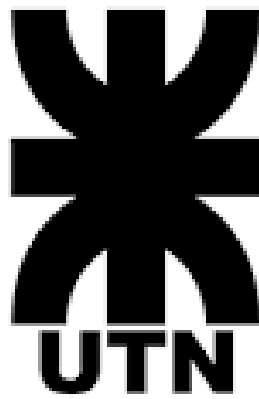


# INFORME TP 4 COMPILADOR

Materia: SSL

## Explicación del pensamiento detrás del trabajo



Fecha de entrega: 24/11/2023

Integrantes Grupo 25:

-CACACE, Guillermo Federico

-CALÓ, Ignacio

-GOMEZ PEREYRA, Manuel Francisco

- MAJER, Cecilia Alejandra

-TROSSERO, Agustín Francisco

- 1) Compilar el Código entregado en clase.

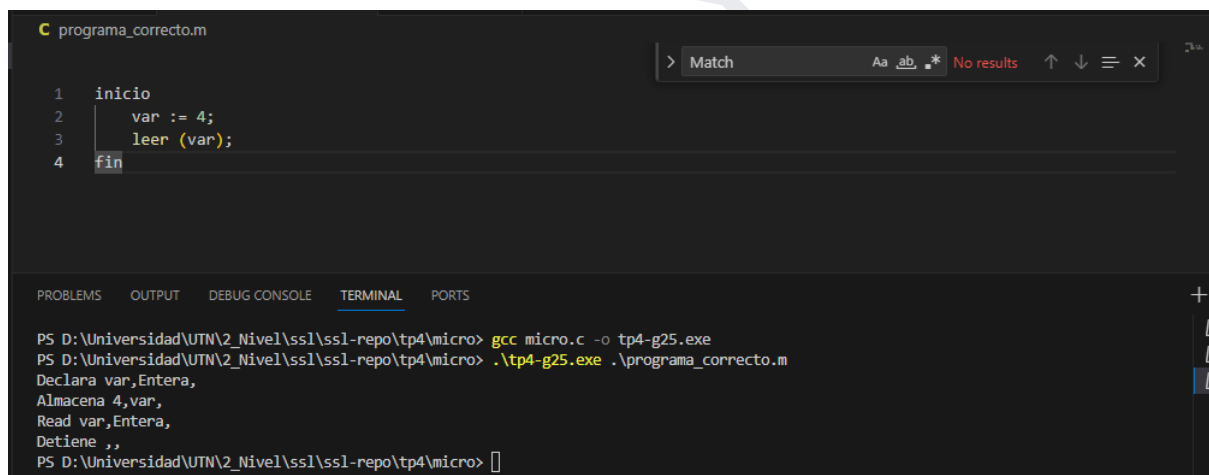
El programa base incluía anotaciones y código comentado que irrumpía en la compilación del programa de C. Se corrigió de modo tal que el programa compile como debe.

- 2) Preparar 3 programas en micro: 1 correcto , 1 con error léxico , 1 con error sintáctico y semántico.

### **PROGRAMA CORRECTO**

```
//PROGRAMA CORRECTO

inicio
    id:= 4;
fin
```



The screenshot shows a C compiler interface with a code editor and a terminal. The code editor displays the following C code:

```
1 inicio
2     var := 4;
3     leer (var);
4 fin
```

The terminal shows the following commands and output:

```
PS D:\Universidad\UTN\2_Nivel\ssl\ssl-repo\tp4\micro> gcc micro.c -o tp4-g25.exe
PS D:\Universidad\UTN\2_Nivel\ssl\ssl-repo\tp4\micro> .\tp4-g25.exe .\programa_correcto.m
Declara var,Entera,
Almacena 4,var,
Read var,Entera,
Detiene ,,
PS D:\Universidad\UTN\2_Nivel\ssl\ssl-repo\tp4\micro>
```

Se compila el programa de C, para luego agregar el “Programa correcto” de Micro. Su funcionamiento adecuado muestra por consola las declaraciones, el valor almacenado, y el resultado de la función leer, como estaba definido en el código base, devuelve el tipo de dato leído.

## PROGRAMA ERROR LÉXICO

```
//PROGRAMA ERROR LEXICO

inicio
    id:= 4/4; // NO RECONOCE EL LEXEMA '/'
fin
```

C error\_lexico.m

```
1 inicio
2     id:= 4/4;
3 fin
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\Universidad\UTN\2_Nivel\ssl\ssl-repo\tp4\micro> .\tp4-g25.exe .\error_lexico.m
Declara id,Entera,
Error Lexico
Almacena 4,id,
Error Sintactico
Error Sintactico
Error Sintactico
Detiene ,,
PS D:\Universidad\UTN\2_Nivel\ssl\ssl-repo\tp4\micro> █
```

Detecta el error léxico, pero la falta de la completitud de esa expresión termina generando los errores sintácticos. Es decir, cuando el léxico detecta el carácter “/” se dispara el mensaje de error e ignora ese carácter, por lo que el analizador sintáctico intenta leer cada token siguiente, sin entender la estructura, generando que por cada uno de los tres tokens siguientes, los toma como errores sintácticos.

**PROGRAMA ERROR SINTÁCTICO/SEMÁNTICO**

```
// ERROR SINTACTICO/SEMANTICO
```

```
inicio
```

```
    leer (a)    // ERROR SINTACTICO (NO TERMINA LA EXPRESION CON ';')
```

```
    // ERROR SEMANTICO (NO SE DECLARA LA VARIABLE 'a')
```

```
fin
```

C error\_sint-seman.m X

C error\_sint-seman.m

1 inicio

2 leer (a)

3 fin

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\Universidad\UTN\2_Nivel\ssl\ssl-repo\tp4\micro> .\tp4-g25.exe .\error_sint-seman.m
Declara a,Entera,
Read a,Entera,
Error Sintactico
Error Sintactico
Detiene ,,
PS D:\Universidad\UTN\2_Nivel\ssl\ssl-repo\tp4\micro> []
```

- 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.

### **Análisis léxico:**

Este se encarga de analizar carácter a carácter el código, formando “tokens” a partir del reconocimiento de lexemas a través del uso de ERs. En el momento en que el analizador reconoce un carácter que no esté en ninguna ER, este falla generando un mensaje de error léxico, el cuál no traba la compilación del código, sino que ignora el elemento desconocido para completar este proceso.

Para el caso citado en el punto anterior, el error léxico nace en incluir en el programa un carácter que no forma parte del vocabulario del lenguaje.

### **Análisis sintáctico:**

Con este análisis se verifica que la “estructura” del código esté bien implementada. Gracias a los tokens que va recibiendo del analizador léxico, una estructura previamente definida gramaticalmente y un diccionario de datos, este analizador es capaz de reconocer palabras reservadas, sentencias, expresiones, etc. Una forma de ver cómo trabaja este analizador es a través del PAS el cuál por medio de la función ***match*** le va “pidiendo” tokens al analizador léxico para poder seguir identificando estructuras. El error sintáctico se basa en esta predefinición del “camino” que tienen que seguir los tokens y palabras reservadas para que el código esté bien escrito. En el momento en que este camino se “desvía” el analizador romperá el proceso de compilación.

En el caso citado anteriormente, la estructura de la sentencia que declara a la variable “id” está incompleta debido a la falta del punto y coma (“;”) en el cierre de la sentencia.

**Análisis semántico:**

En esta etapa del análisis, el programa se centra más en cosas que tengan que ver con la memoria, ya sea definiciones, matcheo de tipos, contexto de las variables, etc. Básicamente esto funciona como una revisión del código para pasar al proceso de síntesis. Un error semántico está ligado con la falencia de los casos anteriormente mencionados, y al igual que el analizador sintáctico, cuando detecta un error, corta el flujo de compilación.