UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Ingeniería en sistemas de Información
Sintaxis y semántica de los lenguajes – 2023

TURNO: Noche CURSO: K2055

DOCENTE A CARGO: Ing. Roxana Leituz
AYUDANTE:

TRABAJO PRÁCTICO TEORICO N°4
"Compilador Micro ASDR"

ÁREA TEMÁTICA: Proceso de compilación

GRUPO N° 30
Rodríguez Lucas Ariel
Golato Barcia Ivan Nahuel
Sayago Pablo
Rabahia Maron Leonel
Schinca Mauro

FECHA DE VENCIMIENTO: 24/11/2023
FECHA DE PRESENTACIÓN: /11/2023
FECHA DE DEVOLUCIÓN: __/__/__

CALIFICACIÓN: _____ FIRMA PROFESOR: _____

Compilación del programa CompiladorMicro.c:

Para compilar el programa CompiladorMicro.c vamos a hacerlo por medio de un archivo MakeFile que contiene el siguiente código:

```
ifdef OS
    RM = del /Q
    ECHO = @echo
else
    RM = rm -f
    ECHO = echo
endif

BIN = compiladorMicro.exe
OBJ = compiladorMicro.o
CC = gcc
CFLAGS = -std=c18
LFLAGS = -lm

$(BIN): $(OBJ)
    $(CC) $(OBJ) -o $(BIN) $(CFLAGS) $(LFLAGS)

test: $(BIN)
    $(ECHO) Programa micro correcto:
    ./$(BIN)  programaCorrecto.m
    $(ECHO) ------------------------------------------------------------------------------
    $(ECHO) Programa micro con error lexico:
    ./$(BIN)  ErrorLexico.m
    $(ECHO) ------------------------------------------------------------------------------
    $(ECHO) Programa micro con error sintactico y semantico:
    ./$(BIN)  ErrorSinYSem.m
    $(ECHO) ------------------------------------------------------------------------------

all: $(BIN)

compiladorMicro.o: compiladorMicro.c
    $(CC) -c compiladorMicro.c -o compiladorMicro.o $(CFLAGS)

clean:
    $(RM) *.o compiladorMicro.exe
```

Para realizar la compilación solo basta con abrir una consola en la ubicación de nuestros archivos y ejecutar el comando **make** seguido de la tecla ENTER, como se ve en la siguiente imagen:

```
PS C:\Users\PC1\Documents\Facultad\SSL\SSl-trabajos\CompiladorMicroDescendente> make
gcc -c compiladorMicro.c -o compiladorMicro.o -std=c18
gcc compiladorMicro.o -o compiladorMicro.exe -std=c18 -lm
```

Esto es equivalente a los comandos gcc que aparecen debajo de la ejecución del make.

Pero el MakeFile no solo nos permite realizar la compilación del programa por medio del comando make, sino que también podemos realizar una limpieza de los archivos .o y .exe por medio del comando **make clean**, esto en el caso que sea necesario realizar algún cambio y volverlo a compilar de una manera rápida.

```
PS C:\Users\PC1\Documents\Facultad\SSL\SSl-trabajos\CompiladorMicroDescendente> make clean
rm -rf *.o compiladorMicro.exe
PS C:\Users\PC1\Documents\Facultad\SSL\SSl-trabajos\CompiladorMicroDescendente>
```

Ejecución y errores:

Por último el MakeFile nos permite ejecutar el programa con la entrada que quisiéramos, como vemos en el código del MakeFile ejecutamos los diferentes programas, tanto con errores como el correcto, como entrada del programa compiladorMicro.exe .Esto realizamos por medio del comando **make test**

```
PS D:\UTN\Sintaxis\tp4\Compilador-micro-ASDR> mingw32-make test
Programa micro correcto:
./compiladorMicro.exe  programaCorrecto.m
Declara variable1,Entera,
Almacena 25,variable1,
Declara variable2,Entera,
Almacena 0,variable2,
Declara variable3,Entera,
Almacena 80,variable3,
Write variable1,Entera,
Declara varable2,Entera,
Read varable2,Entera,
Detiene ,,
--------------------------------------------------------------------------------
Programa micro con error lexico:
./compiladorMicro.exe  ErrorLexico.m
Error Lexico
Error Sintactico
Error Sintactico
Detiene ,,
--------------------------------------------------------------------------------
Programa micro con error sintactico y semantico:
./compiladorMicro.exe  ErrorSinYSem.m
Declara variable1,Entera,
Almacena 25,variable1,
Declara varable3,Entera,
Almacena 80,varable3,
Declara variable2muyExtensaMasDeTreintaYdosC♦,Entera,
Almacena 10,variable2muyExtensaMasDeTreintaYdosC♦,
Declara variable4,Entera,
Almacena variable4,variable4,
Error Sintactico
Error Sintactico
Error Sintactico
Detiene ,,
--------------------------------------------------------------------------------
```

Error Léxico:

```
--------------------------------------------------------------------------------
Programa micro con error lexico:
./compiladorMicro.exe  ErrorLexico.m
Error Lexico
Error Sintactico
Error Sintactico
Detiene ,,
--------------------------------------------------------------------------------
```

Como error léxico propusimos **.variable4 := 0;** en este caso estamos intentando declarar un identificador que empiece con un carácter no reconocido del lenguaje, como podemos ver en su gramática.

Gramática Léxica

```
<token> -> uno de <identificador> <constante> <palabraReservada>
                  <operadorAditivo> <asignación> <carácterPuntuación>
<identificador> -> <letra> {<letra o dígito>}
<constante> -> <dígito> {dígito>}
<letra o dígito> -> uno de <letra> <dígito>
<letra> -> una de a-z A-Z
<dígito> -> uno de 0-9
<palabraReservada> -> una de inicio fin leer escribir
<operadorAditivo> -> uno de + -
<asignación> -> :=
<carácterPuntuación> -> uno de ( ) , ;
```

Esta clase de errores son detectados por el autómata implementado en el compiladorMicro, el cual devuelve el código correspondiente al token ERRORLEXICO, él cual se obtiene por medio de la función scanner();.

Error sintáctico y semántico:

```
--------------------------------------------------------------------------------
Programa micro con error sintactico y semantico:
./compiladorMicro.exe  ErrorSinYSem.m
Declara variable1,Entera,
Almacena 25,variable1,
Declara varable3,Entera,
Almacena 80,varable3,
Declara variable2muyExtensaMasDeTreintaYdosC♦,Entera,
Almacena 10,variable2muyExtensaMasDeTreintaYdosC♦,
Declara variable4,Entera,
Almacena variable4,variable4,
Error Sintactico
Error Sintactico
Error Sintactico
Detiene ,,
--------------------------------------------------------------------------------
```

Como error sintáctico propusimos la sentencia **variable4 := escribir (variable1-variable3);** la cual no coincide con ninguna de las reglas de la gramática sintáctica de micro

Gramática Sintáctica

```
<programa> -> inicio <listaSentencias> fin
<listaSentencias> -> <sentencia> {<sentencia>}
<sentencia> -> <identificador> := <expresión> ; |
               leer ( <listaIdentificadores> ) ; |
               escribir ( <listaExpresiones> ) ;
<listaIdentificadores> -> <identificador> {, <identificador>}
<listaExpresiones> -> <expresión> {, <expresión>}
<expresión> -> <primaria> {<operadorAditivo> <primaria>}
<primaria> -> <identificador> | <constante> |
              ( <expresión> )
```

El programa intenta hacerla coincidir con una sentencia de asignación ya que machea en primera instancia con **variable4 :=** pero la regla necesita una **expresion** como siguiente token y no una palabra reservada u otro tipo de token.

El error semántico está ubicado en la siguiente sentencia que no respeta el límite de 32 caracteres que puede llegar a tener un identificador en el lenguaje micro **variable2muyExtensaMasDeTreintaYdosCaracteres := 0.**

Este error no está contemplado en ninguna gramática (y por lo tanto no es reconocido ni por el scanner ni por el parser) ya que se trata de la parte semántica del lenguaje, la cual se define en lenguaje natural y se implementa mediante una rutina semántica programada en lenguaje c.