UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Sintaxis y Semántica de los lenguajes

Trabajo Práctico N°4
Compilador de Micro en C

GRUPO 7

<Zoe Iael Mancini>	<206.464-9>	<zmancini@frba.utn.edu.ar>	<zoemancini>
<Sofia Zalazar>	<204.178-9>	<szalazar@utn.frba.utn>	<SOFIAZALAZAR>
<Román Brezman>	<203.432-3>	<rbrezman@frba.utn.edu.ar>	<rabrezman>
<Mariano Bruno>	<138.794-7>	<marbruno@frba.utn.edu.ar>	<marianonahuelbruno>
<Guido Moreyra >	<209.802-7>	<gmoreyra@frba.utn.edu.ar	<GuidoMoreyra>

Docente: *Roxana Leituz* ***Curso:*** *K2055*

Índice

Objetivos

Compilar y ejecutar el código del compilador de Micro hecho en C. Probar el compilador con distintos códigos en Micro, forzando errores, y analizar el comportamiento del mismo.

Link al repositorio de GitHub

En el siguiente repositorio se encuentran subidos todos los archivos que se utilizaron para llevar a cabo el presente trabajo: https://github.com/rabrezman/tp-analizador-grupo7

Compilación y ejecución del Compilador de Micro

Luego de generar nuestro archivo .c con el código visto en clase, procedimos a compilarlo y generar el ejecutable del mismo:

```
zoema@ASUS-ZOEMANCINI MINGW64 ~/Desktop/ssl/tp-analizador-grupo7 (main)
$ gcc compilador.c -o compilador
```

Una vez que tuvimos el ejecutable, procedimos a correrlo pasandole los distintos códigos en micro que habíamos generado:

```
zoema@ASUS-ZOEMANCINI MINGW64 ~/Desktop/ssl/tp-analizador-grupo7 (main)
$ ./compilador microCorrecto.m
```

```
zoema@ASUS-ZOEMANCINI MINGW64 ~/Desktop/ssl/tp-analizador-grupo7 (main)
$ ./compilador microErrorLexico.m
```

```
zoema@ASUS-ZOEMANCINI MINGW64 ~/Desktop/ssl/tp-analizador-grupo7 (main)
$ ./compilador microErrorSem.m
```

Prueba 1: código en Micro sin errores

Detallamos a continuación el código que utilizamos para hacer la prueba con código Micro correcto:

```
1 inicio
2    leer (a,b);
3    cc := a + (b-2);
4    escribir (cc, a+4);
5 fin
```

La salida obtenida al pasarle este código al compilador fue la siguiente:

```
zoema@ASUS-ZOEMANCINI MINGW64 ~/Desktop/ssl/tp-analizador-grupo7 (main)
$ ./compilador microCorrecto.m
Declara a,Entera,
Read a,Entera,
Declara b,Entera,
Read b,Entera,
Declara cc,Entera,
Declara Temp&1,Entera,
Restar b,2,Temp&1
Declara Temp&2,Entera,
Sumar a,Temp&1,Temp&2
Almacena Temp&2,cc,
Write cc,Entera,
Declara Temp&3,Entera,
Sumar a,4,Temp&3
Write Temp&3,Entera,
Detiene ,,
```

Como vemos, y como esperábamos, el compilador no detectó ningún error léxico, sintáctico o semántico en el código.

También vemos que el compilador puede "entender" cuando una variable está siendo declarada, también cuando se está leyendo algún valor, sumando, restando y mostrando por pantalla valores. Finalmente, vemos como la palabra reservada "fin" la lee y comprende que es el final del programa, imprimiendo por pantalla la leyenda "Detiene".

Prueba 2: código en Micro con un error léxico

Detallamos a continuación el código que utilizamos para hacer la prueba con código Micro con un error léxico:

```
1 inicio
2     a*2;
3 fin
```

El código es muy simple y se puede ver claramente cuál es el error léxico: el lenguaje Micro no soporta la multiplicación, sólo suma y resta, por lo tanto no comprende el operador "*".
Veamos cuál es la salida del programa al pasarle este código:

```
zoema@ASUS-ZOEMANCINI MINGW64 ~/Desktop/ssl/tp-analizador-grupo7 (main)
$ ./compilador microErrorLexico.m
Declara a,Entera,
Error Lexico
Error Sintactico
Almacena 2,a,
Detiene ,,
```

Podemos ver claramente que el scanner primero lee el caracter "a", luego al leer el * entiende que "a" es un token y se lo pasa al parser. El parser comprende que está declarando una variable cuyo identificador es "a" y por eso vemos que se imprime la leyenda "Declara a".
Luego de esto, el scanner lee el * y no lo comprende, por ello se indica "Error Léxico". También aparece un "Error Sintáctico" que se debe a cómo está implementado el compilador, pues este continúa analizando a pesar de que ya detectó un error léxico. Es probable que el scanner luego de hallar el error léxico el siguiente token que encuentre es "2", se lo pasa al parser y este indica que hay un error sintáctico porque luego de la declaración de la variable a no encuentra un estado de aceptación con el token que le está pasando el scanner. Al no haber considerado el * como un operador es como si no hubiera ningún operador entre los operandos, por eso considera el Error Sintáctico.

Prueba 3: código en Micro con error sintáctico y semántico

Detallamos a continuación el código que utilizamos para hacer la prueba con código Micro con un error sintáctico:

```
1 inicio
2    leer (a,b);
3    cc := a + ((b-2);
4    escribir (cc, a+4);
5 fin
```

En la línea 3 está el error sintáctico, la expresión tiene un paréntesis de apertura de más que podemos ver que nunca se cierra.
Y con un error semántico:

```
1 inicio
2 inicio
3    leer (a,b);
4    cc := a + (b-2);
5    escribir (cc, a+4);
6 fin
```

En la línea 2 está el error semántico, se repite la palabra reservada para indicar comienzo del programa.

Veamos cual es la salida al pasarle el primer código al compilador:

```
zoema@ASUS-ZOEMANCINI MINGW64 ~/Desktop/ssl/tp-analizador-grupo7 (main)
$ ./compilador microErrorSin.m
Declara a,Entera,
Read a,Entera,
Declara b,Entera,
Read b,Entera,
Declara cc,Entera,
Declara Temp&1,Entera,
Restar b,2,Temp&1
Error Sintactico
Declara Temp&2,Entera,
Sumar a,Temp&1,Temp&2
Almacena Temp&2,cc,
Error Sintactico
Error Sintactico
Error Sintactico
Detiene ,,
```

Vemos que comienza analizando la expresión que se encuentra dentro del paréntesis y arroja la leyenda “Restar b,2”. Luego, indica “Error Sintáctico” pues encuentra que los paréntesis no están balanceados. Sin embargo, continúa con el análisis: encuentra que suma lo anterior con

la variable “a” y todo ello se almacena en cc. Luego vemos que se indican más errores sintácticos aunque no los haya, esto puede deberse al mismo motivo por el que en el caso anterior nos indicaba un error sintáctico a pesar de que no hubiera ninguno: el compilador continúa el análisis incluso habiendo ya detectado errores, y esto puede generar que interprete otras cosas como posibles errores. Es probáble que los últimos errores lanzados sean porque el parser continúa buscando el paréntesis de cierre de la expresión y no lo encuentra.

Vamos a analizar la salida del código indicado con error semántico:

```
zoema@ASUS-ZOEMANCINI MINGW64 ~/Desktop/ssl/tp-analizador-grupo7 (main)
$ ./compilador microErrorSem.m
Error Sintactico
Error Sintactico
Detiene ,,
```

Nuevamente vemos detección de errores sintácticos donde no los hay. La implementación del compilador es claro que contiene rutinas semánticas, pues sino el error que forzamos con el código ingresado sería indetectable. Sin embargo, debe haber un problema en la señalización del tipo de error hallado. También vemos el error repetido en este caso, producto de lo mismo que causó la repetición de errores previamente: el analizador continúa a pesar de ya haber encontrado errores.
Por otro lado, vemos que el error es tan grave que el compilador no hace el intento de comprender el bloque de código que sí tiene sentido. Lo único que llega a comprender es la marca de fin de programa.