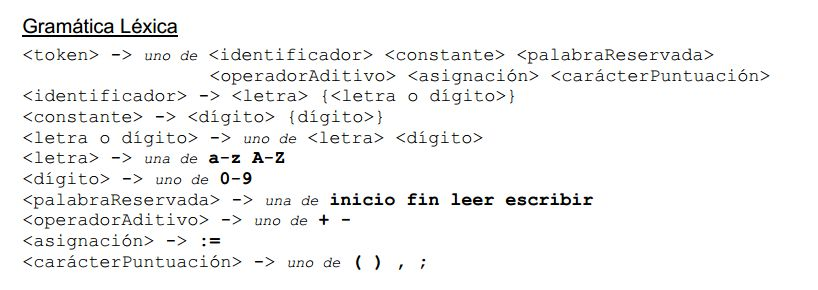
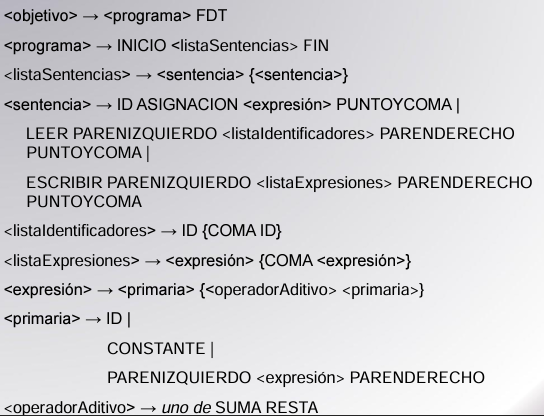
Resolución del trabajo

Para la realización del trabajo, como primer paso, se utilizó **Flex** para realizar el análisis léxico del programa, teniendo en cuenta la gramática léxica de **Micro**. Se incluyó además el carácter de espacio (" ") que no retorna nada y el token "fdt" como la marca de finalización del programa.

Posteriormente, se utilizó Bison para realizar el análisis sintáctico, apoyándonos en la gramática correspondiente.:



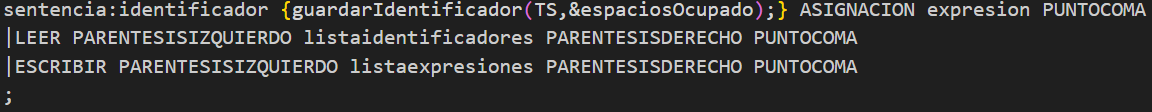
Detección de errores

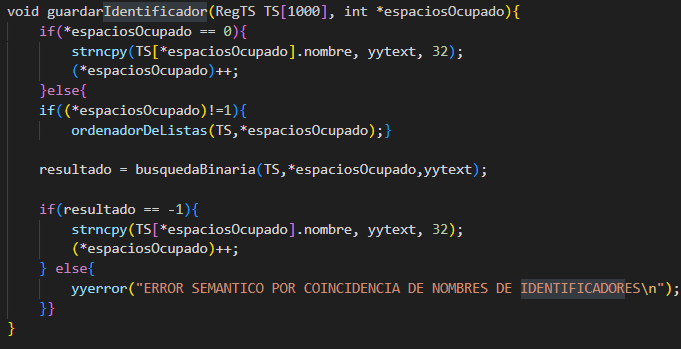
Para la detección de errores se utilizó la función yyerror y se llamó en caso de ser necesario.

Rutinas semánticas

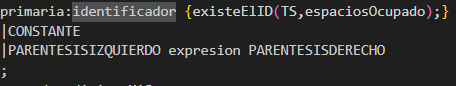
Adicionalmente, se implementaron las siguientes rutinas semánticas:

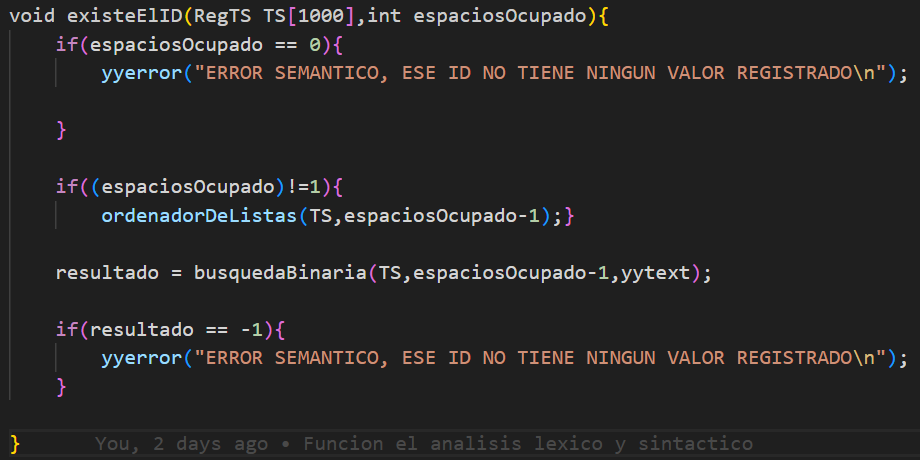
* **Mensaje de finalización**: Se agregó un printf que notifica cuando se ha leído la marca de fin de texto ("fdt"). Esto indica que cualquier entrada posterior generará un error, ya que el programa no espera más datos después de esta marca.
* **Tabla de símbolos simplificada**: Se implementó una tabla de símbolos que únicamente almacena los identificadores.No se registra el tipo ni el contexto de los identificadores, dado que solo existe un único tipo en este lenguaje. Las palabras reservadas se detectan directamente con el token correspondiente y las constantes no se almacenan porque no se utilizan en otras partes del programa.
* **Validación de identificadores**:Además cada vez que se ingresa un identificador se verifica que no tenga más de 32 caracteres y si es una sentencia posteriormente se registra en la tabla de símbolos (si no estaba previamente). En caso contrario, se genera un error semántico.

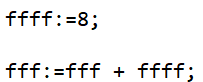




* **Asignación previa de identificadores**: Durante la evaluación de una expresión, se verifica que cualquier identificador utilizado como operando haya sido asignado previamente.

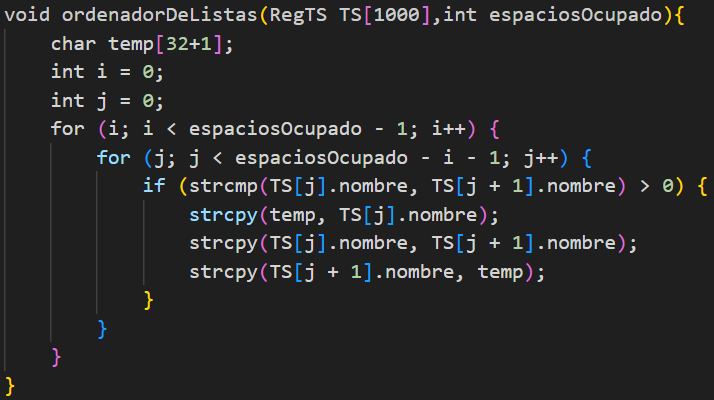
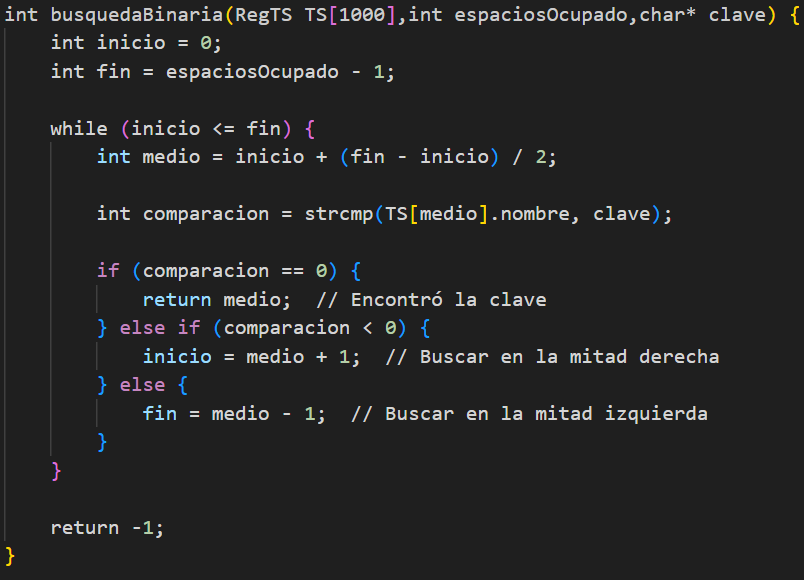




Una cosa a tener en cuenta es que no toma en cuenta al último identificador ingresado debido a que este es al que se le está asignando mismamente toda la ecuación que contiene al otro identificador. Por ejemplo, en el caso de 

El primer fff no se toma en cuenta para analizar el fff dentro de expresion. Si se pusiera al revés ( es decir fff:=ffff + fff) tampoco habría problema porque el ffff no se guarda en la tabla por lo que el análisis es el mismo osea sin el ultimo.

Para la realización de ambas funciones se utilizó una función de búsqueda binaria y a su vez como esta requiere que el vector este ordenado se hizo una función de ordenamiento del vector:

Manual de usuario

Para su uso en caso de no tener el ejecutable, los .c y .h de bison y flex se debe escribir en consola:

|  |
| --- |
| bison -yd microbison.y |
| flex micro.l |
| gcc y.tab.c lex.yy.c -o eje |

Una vez hecho esto a continuación se puede ingresar la prueba por consola o con un archivo:

* En caso de querer hacerlo por consola solo se debe escribir:

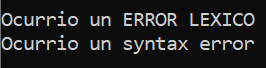
|  |
| --- |
| eje |

Darle a enter y posteriormente escribir la prueba que quiera realizar

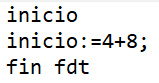
* En caso de querer hacerlo con un archivo debe escribir el txt y posteriormente escribir:

|  |
| --- |
| eje < (nombre del txt con el .txt) |

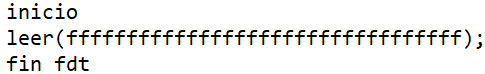
Ejemplo1:

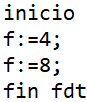
Ejemplo2:



Ejemplo3:



Ejemplo4:



Ejemplo5:

