MANUAL DE USUARIO
Actividad de Aprendizaje - Proyecto Final

Teoría de la Computación

Maestro: Luis Fernando Curi Quintal

Integrantes del equipo:

- Canul Duran Nicolás Israel
- Couoh Concha Valeria Fernanda
- Gonzalez Alcocer Fabio Leonardo
- Gonzalez Chable Said Alfredo

Índice

1. Presentación del Proyecto

El presente documento constituye el Manual de Usuario del compilador desarrollado para analizar programas escritos en el lenguaje MIO. Este manual ha sido actualizado conforme a la nueva versión del proyecto, en la cual se reescribieron y optimizaron los módulos del analizador léxico y el analizador sintáctico. Debido a estos cambios, el manual anterior quedó obsoleto y no representa adecuadamente el funcionamiento actual del sistema, por lo que este documento sirve como reemplazo oficial.

Este compilador es producto de una actividad académica en la asignatura *Teoría de la Computación*, cuyo propósito principal es introducir al estudiante en el funcionamiento interno de los compiladores, el análisis de lenguajes formales y el procesamiento automatizado de código mediante técnicas reales de análisis léxico y sintáctico.

El manual está diseñado para usuarios sin experiencia previa en compiladores, proporcionando explicaciones claras y detalladas sobre cada parte del sistema y ofreciendo ejemplos prácticos mediante las Prueba 1 y Prueba 2, que funcionan como casos de uso ilustrativos del comportamiento del compilador.

2. ¿Qué es el Compilador MIO?

El Compilador MIO es una herramienta que permite analizar archivos escritos en el lenguaje MIO. Este lenguaje, creado con fines académicos, incluye una serie de instrucciones, estructuras de control y expresiones aritméticas diseñadas para ejemplificar la estructura formal de un lenguaje de programación.

El compilador se divide en dos etapas principales:

a) Análisis Léxico

En esta fase, el compilador examina el archivo .mio carácter por carácter, separando la entrada en componentes válidos llamados *tokens*.
Un token puede ser:

- Una palabra reservada (como PROGRAMA, SI, IMPRIME, FINPROG)
- Un identificador
- Un número hexadecimal
- Un literal de texto
- Un operador
- Un símbolo como =

Si el analizador encuentra un carácter o patrón desconocido, detiene el proceso y reporta un error léxico.

b) Análisis Sintáctico

Una vez generado el archivo .lex, el compilador verifica que la secuencia de tokens respete la gramática del lenguaje MIO.
Durante este proceso el sistema identifica:

- Sentencias válidas
- Estructuras de control correctamente anidadas
- Expresiones bien formadas
- Comparaciones válidas
- Aparición obligatoria de PROGRAMA y FINPROG

Si el orden o la estructura no coincide con la gramática, se produce un error sintáctico, indicando la posición exacta del problema.

A diferencia de compiladores complejos, este sistema no ejecuta el programa, sino que verifica su validez estructural.

3. Componentes del Proyecto

El proyecto consta de dos módulos principales encargados del análisis del lenguaje, y ambos fueron reescritos o modificados para esta versión del compilador.

3.1 Analizador Léxico — *analex.py*

Este módulo es responsable de abrir el archivo .mio y traducir su contenido a una secuencia de tokens válidos.

El analizador reconoce:

- Palabras reservadas: PROGRAMA, FINPROG, SI, MIENTRAS, IMPRIME, LEE, etc.
- Identificadores: nombres de variables definidos por el usuario.
- Valores hexadecimales: expresiones como 0xA, 0x10 o 0xFF.
- Literales de texto: secuencias encerradas entre comillas.

- Operadores aritméticos (+, -, *, /)
- Operadores relacionales (<, <=, >=, ==, <>, >)
- Símbolo de asignación (=)

El analizador también valida aspectos como:

- Identificadores demasiado largos
- Hexadecimales sin dígitos
- Literales sin cerrar
- Caracteres no reconocidos

Si la entrada es válida, el módulo genera dos archivos:

- .lex — Secuencia ordenada de tokens
- .sim — Tabla de símbolos, que registra variables, textos y valores

En caso de error, se detiene el análisis y se muestra el problema con su ubicación exacta.

3.2 Analizador Sintáctico — *anasin.py*

Este módulo toma el archivo .lex y determina si su estructura coincide con la gramática MIO. Está construido como un parser recursivo descendente, lo cual facilita identificar:

- La estructura principal PROGRAMA … FINPROG
- Sentencias válidas
- Orden correcto de palabras reservadas
- Anidamientos correctos en condicionales
- Bucles correctamente delimitados
- Expresiones válidas
- Comparaciones permitidas

Si el programa cumple con la gramática, la salida es:

Compilación exitosa

De lo contrario, el analizador muestra:

- Token encontrado

- Token esperado
- La posición del error
- Un mensaje explicativo

Este módulo es crucial para detectar problemas de estructura aun cuando la entrada léxica es correcta.

3.3 Ejecución Manual de Pruebas

Aunque la versión anterior del proyecto incluía un ejecutor automático, en esta versión se trabaja directamente con los módulos léxico y sintáctico para analizar manualmente los casos de uso.

Las pruebas incluidas en este manual (“Prueba 1” y “Prueba 2”) fueron procesadas ejecutando manualmente ambos analizadores, generando archivos .lex, .sim y salida.log.

4. ¿Cómo usar el compilador?

A continuación se muestra el proceso correcto para analizar un archivo del lenguaje MIO.

4.1 Análisis de un archivo MIO

1. Ejecución del Analizador Léxico

python analex.py archivo.mio

La salida será:

- archivo.lex → tokens generados
- archivo.sim → tabla de símbolos

Si se detecta un error léxico, el proceso se detiene y se muestra el problema.

2. Ejecución del Analizador Sintáctico

python anasin.py archivo.lex

Posibles salidas:

- Compilación exitosa
- Mensaje de error sintáctico con su posición

Con esto, el usuario puede identificar si el programa .mio cumple con la gramática.

5. Prueba 1

La Prueba 1 representa un caso básico utilizado para comprobar los componentes esenciales del compilador. Esta prueba contiene un programa sencillo en MIO que permite validar tanto el análisis léxico como el sintáctico sin involucrar estructuras complejas.

Objetivo de la Prueba 1

- Verificar que el analizador léxico identifique correctamente los tokens esenciales del lenguaje.
- Validar que el analizador sintáctico acepte la estructura de un programa simple.
- Confirmar que no existan errores de reconocimiento en elementos básicos como:
    - Identificadores
    - Valores hexadecimales
    - Expresiones sencillas
    - Sentencias fundamentales

Comportamiento del Analizador Léxico

Al procesar el archivo .mio correspondiente a Prueba 1, el analizador debe producir un .lex que contiene:

- Los tokens de inicio de programa (PROGRAMA [id])
- Una secuencia de sentencias simples
- Su terminación en FINPROG

La tabla .sim debe registrar:

- Los identificadores usados
- Los valores numéricos

- Códigos internos asignados correctamente

Comportamiento del Analizador Sintáctico

El archivo .lex generado debe ser válido y producir:

Compilación exitosa

Esto confirma que el compilador reconoce correctamente la estructura principal del lenguaje y que el flujo básico del programa es aceptado según la gramática de MIO.

Importancia Académica de la Prueba 1

Esta prueba asegura que:

- El entorno de ejecución funciona adecuadamente
- Los analizadores están correctamente instalados
- La ruta básica del compilador (léxica y sintáctica) es estable

Es un punto de partida ideal para comenzar a trabajar con el compilador.

6. Prueba 2

La Prueba 2 incorpora elementos adicionales que permiten evaluar un comportamiento más profundo del compilador. Aunque continúa siendo un ejemplo sencillo, introduce aspectos que permiten validar un mayor espectro del sistema.

Objetivo de la Prueba 2

- Verificar reconocimiento de estructuras adicionales del lenguaje
- Evaluar expresiones más completas
- Comprobar el tratamiento correcto de operadores
- Generar una tabla de símbolos más extensa
- Ver cómo el compilador maneja variaciones en orden y complejidad sintáctica

Comportamiento del Analizador Léxico

En esta prueba, el léxico debe:

- Detectar correctamente identificadores adicionales
- Procesar expresiones compuestas

- Reconocer operadores según corresponda
- Registrar múltiples elementos en .sim

El archivo .lex resultante permite observar una secuencia más rica de tokens, mostrando cómo el compilador traduce un programa MIO en su forma interna.

Comportamiento del Analizador Sintáctico

El analizador sintáctico debe aceptar el .lex generado y producir:

Compilación exitosa

Esto confirma que la gramática puede manejar casos con expresiones más elaboradas o sentencias en orden diferente al de Prueba 1.

Importancia Académica de la Prueba 2

Permite evaluar:

- La robustez del analizador léxico ante mayor variedad de entradas
- La precisión del analizador sintáctico ante estructuras sintácticas válidas
- La coordinación entre ambos módulos

En conjunto, esta prueba complementa la Prueba 1 para demostrar que el compilador puede manejar distintos tipos de programas dentro de MIO.

7. Interpretación de Resultados

El compilador puede generar tres tipos de archivo mientras analiza un programa MIO:

a) **.lex**

Lista la secuencia de tokens exactamente en el orden del programa.
Permite verificar si el léxico interpretó correctamente cada elemento.

b) **.sim**

Contiene:

- Identificadores
- Literales de texto
- Valores numéricos

- Códigos internos

Es fundamental para entender cómo el analizador reconoce elementos semánticos del programa.

c) **salida.log**

Este archivo compila la información del análisis:

- Si hubo errores léxicos
- Si se generaron los archivos .lex y .sim
- Si la compilación sintáctica fue exitosa
- En caso de error, la descripción detallada

Mensajes del Analizador Sintáctico

- Compilación exitosa:
  Indica que el programa cumple la gramática del lenguaje MIO.
- Error sintáctico:
  Cuando se encuentra un token inesperado, el analizador reporta:
  - Qué token se esperaba
  - Qué token se encontró
  - La posición del error

Esto permite al usuario corregir el archivo .mio de manera precisa.