Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Santa Fe

Desarrollo de aplicaciones avanzadas de ciencias computacionales

ITC (Gpo 501)

Proyecto 3: Analizador Semántico

Estudiante:

David Medina Domínguez - A01783155

Docentes:

Dr. Víctor Manuel de la Cueva Hernández

Fecha de entrega

14 de mayo de 2025

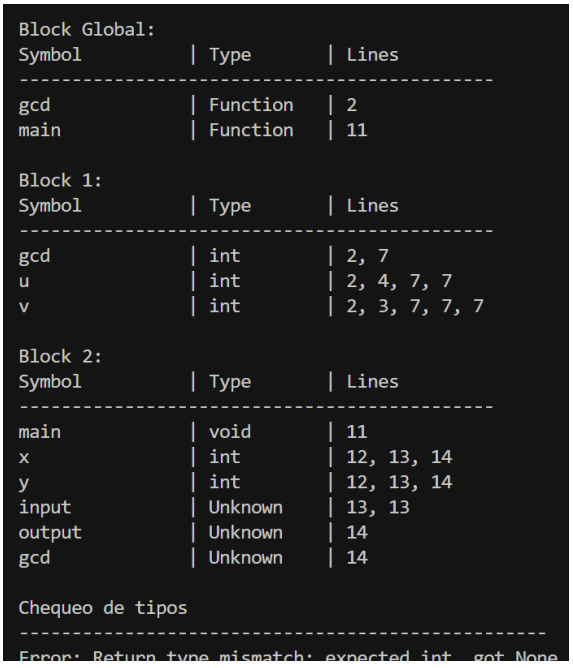
Este proyecto desarrolla un analizador semántico que trabaja sobre el árbol de sintaxis abstracta generado previamente, con el fin de validar las reglas del lenguaje. El proceso de verificación se realiza de forma recursiva, comenzando desde el punto de entrada del programa y evaluando cada declaración en su contexto correspondiente. Solo se permiten dos tipos de datos: enteros y vacíos, siendo este último exclusivo para funciones y no aplicable a declaraciones simples.

Para controlar la visibilidad y el alcance de los elementos, se emplea una estructura en pila que organiza los diferentes entornos de ejecución. Cada nuevo bloque crea un nuevo contexto, y al finalizar, se regresa al anterior. Las funciones deben definirse claramente antes de ser utilizadas, no pueden redefinirse, y si devuelven un valor, este debe ser coherente con el tipo declarado. Las funciones sin valor de retorno también deben seguir ciertas restricciones.

Durante el análisis, se revisa que todas las operaciones sean válidas según los tipos involucrados. También se comprueba que las estructuras de datos se accedan correctamente, respetando tanto el tipo como los límites definidos. Las expresiones utilizadas en condiciones deben ser del tipo adecuado para que el flujo de control sea válido.

Las llamadas a procedimientos se validan asegurando que existan, que el número de argumentos coincida, y que los tipos esperados sean compatibles. Además, se permite el uso de estructuras como condicionales y ciclos, siempre que las expresiones de control cumplan con las reglas semánticas del lenguaje.

En caso de encontrar errores, estos se reportan junto con su ubicación en el código fuente, permitiendo continuar el análisis mediante un mecanismo de recuperación que evita detener el proceso por completo.
Tabla:



```
Block Global:
Symbol          | Type       | Lines
-------------------------------------------
gcd             | Function   | 2
main            | Function   | 11

Block 1:
Symbol          | Type       | Lines
-------------------------------------------
gcd             | int        | 2, 7
u               | int        | 2, 4, 7, 7
v               | int        | 2, 3, 7, 7, 7

Block 2:
Symbol          | Type       | Lines
-------------------------------------------
main            | void       | 11
x               | int        | 12, 13, 14
y               | int        | 12, 13, 14
input           | Unknown    | 13, 13
output          | Unknown    | 14
gcd             | Unknown    | 14

Chequeo de tipos
-----------------------------------------------
Error: Return type mismatch: expected int, got None
```

Reglas de inferencia

Operadores aritméticos:

Operador: + $$\frac{\vdash e1: Int, \vdash e2: Int}{\vdash e1 + e2: Int}$$	Operador: * $$\frac{\vdash e1: Int, \vdash e2: Int}{\vdash e1 * e2: Int}$$
Operador: - $$\frac{\vdash e1: Int, \vdash e2: Int}{\vdash e1 - e2: Int}$$	Operador: / $$\frac{\vdash e1: Int, \vdash e2: Int}{\vdash e1 / e2: Int}$$

Operadores relacionales:

Operador: > $$\frac{\vdash e1: Int, \vdash e2: Int}{\vdash e1 > e2: Int}$$	Operador: >= $$\frac{\vdash e1: Int, \vdash e2: Int}{\vdash e1 >= e2: Int}$$
Operador: > $$\frac{\vdash e1: Int, \vdash e2: Int}{\vdash e1 > e2: Int}$$	Operador: >= $$\frac{\vdash e1: Int, \vdash e2: Int}{\vdash e1 >= e2: Int}$$
Operador: < $$\frac{\vdash e1: Int, \vdash e2: Int}{\vdash e1 < e2: Int}$$	Operador: <= $$\frac{\vdash e1: Int, \vdash e2: Int}{\vdash e1 <= e2: Int}$$

Operador: **==** $\frac{\vdash e1: Int, \vdash e2: Int}{\vdash e1 == e2: Int}$	Operador: **!=** $\frac{\vdash e1: Int, \vdash e2: Int}{\vdash e1 != e2: Int}$

Asignación de variables:

Básico: $\frac{\vdash [a: Int] \vdash O(a): Int}{\vdash a: Int}$	Array: ○ $\frac{\vdash [a: Int[]] \vdash e: Int}{O \vdash a[e]: Int}$

Operación de asignación:

$$\frac{\vdash e1: Int, \vdash e2: Int}{\vdash e1 = e2: Int}$$

Asignación de funciones:

Void: $\frac{O[f: void] \vdash O(f): void}{\vdash f: void}$	Int: $\frac{O[f: Int] \vdash O(f): Int}{\vdash f: Int}$

Tipos return:

Void: $\frac{O \vdash O(f): void}{\vdash f: void}$	Int: $\frac{O \vdash O(f): Int, e: Int}{\vdash f(e): Int}$