**Avance Final**

**Analizador para lenguaje “Go” con “PlY”**

**Integrantes:**

* Juan Demera
* Jorge Herrera
* Paula Peralta

SOFG1009: Lenguajes de Programación

M.Sc. Rodrigo Alexander Saraguro Bravo

Jueves, 30 de noviembre 2023

**PAO:**

2023 – II

**Contents**

[Implementación de una herramienta que valida el léxico, sintaxis y semántica del Lenguaje de Programación GO 3](#_Toc149601329)

[Introducción 3](#_Toc149601330)

[Componentes Léxicos 4](#_Toc149601331)

[Declaración de variables y constantes 4](#_Toc149601332)

[Tipos de datos 4](#_Toc149601333)

[Palabras Reservadas 6](#_Toc149601334)

[Símbolos o Caracteres Especiales 6](#_Toc149601335)

[Comentarios 7](#_Toc149601336)

[Reglas de sintaxis 8](#_Toc149601337)

[Definición de variables 8](#_Toc149601338)

[Comparaciones de valores 10](#_Toc149601339)

[Estructuras de control 12](#_Toc149601340)

[Estructuras de datos 14](#_Toc149601341)

[Declaración de funciones: 17](#_Toc149601342)

[Describa reglas semánticas 18](#_Toc149601343)

[Interfaz gráfica 22](#_Toc149601344)

[Bibliography 23](#_Toc149601345)

# Implementación de una herramienta que valida el léxico, sintaxis y semántica del Lenguaje de Programación GO

## Introducción

El proyecto consiste en un analizador para el lenguaje de programación Go, implementado con PIY (Python-It-Yourself), una adaptación personalizada de la herramienta PLY (Python Lex-Yacc). Este analizador aborda las fases clave del procesamiento de código: léxico, sintáctico y semántico.

Python es un lenguaje de programación de alto nivel creado por Guido Van Rossun, es el lenguaje de elección en muchas disciplinas profesionales, ya que tiene las siguientes características como: sintaxis limpia y fácil de entender, adicionalmente debido a su versatilidad y legibilidad tiene una inmensa popularidad hace que en la actualidad sea una caja de herramientas esencial para los desarrolladores de software, científicos de datos y aficionados. la principal ventaja de Python es que tiene una gran comunidad y recursos que enriquecen el ecosistema de librerías como la PLY(Python Lex-Yacc) es una herramienta que consta de dos módulos, ‘ply.ley’ y ‘ply.yacc’ que nos permite crear analizadores de código léxicos y sintácticos.

GO o Golang es un lenguaje de programación moderno desarrollado por Google sus principales características es que es de código abierto, concurrente lo que le permite hacer varias tareas al mismo tiempo, compilado, tiene sintaxis clara y legible, facilidad de compilación cruzada, enfocado en el rendimiento, además cuenta con un garbage collector que se encarga de la gestión de la memoria así entre otras características.

Aprender Go , es importante porque pertenece al gigante tecnológico Google, debido a su influencia en el mundo ya sea en el campo de la inteligencia artificial o la gestión de datos que está en aumento. así mismo se acrecienta la demanda de profesionales con habilidades en Go. Te puede abrir oportunidades de empleo y desarrollo profesional en estos campos de gran demanda actualmente.

## Componentes Léxicos

### Declaración de variables y constantes

Para poder declarar una variable o una constante en Go debemos saber que este lenguaje es de tipado fuerte, al momento de declarar una variable, esa variable no cambiará su tipo de dato asignado en todo el programa.

En el proceso de declaración de variable a parte del nombre, intervienen *palabras reservadas*, en el caso de una variable es **var** y en el caso de una constante es **const**.

Para continuar se debe dar un nombre por el cual accederemos al valor de la variable y existen restricciones al momento de declarar el nombre de la variable en Go.

* No debe llevar guiones (Ej.: mi-nombre)
* No pueden comenzar con un número (Ej.: 4variable)
* No pueden llevar símbolos (Ej.: $usuario)
* No pueden tener más de una palabra. (Ej.: mi nombre)

A continuación, se usa el símbolo igual (=) seguido del valor respectivo al tipo de dato dado. Cuando se llegue a la sección de sintaxis en el documento se hablará de una *forma larga* y *forma corta* de declaración de variable, en la forma corta usaremos otro símbolo, dos puntos e igual (:=) esto ayudará mucho a reducir la sintaxis de una declaración de variables.

### Tipos de datos

El objetivo de los tipos de datos es poder clasificar datos de la vida real dentro de la programación, dentro de Go tenemos varios tipos en los cuales podemos guardar números, textos, entre otros, estos tipos de datos pueden clasificarse en datos básicos o compuestos.

#### Tipos de Datos Básicos. Son tipos de datos fundamentales en la gran mayoría de lenguajes de programación, entre los cuales tenemos:

##### Booleanos (bool). Representa un tipo de dato booleano, un valor de verdad, puede ser true o false.

##### String (string). Representa una cadena de texto, puede presentar cualquier dato en formato texto, estos van escritos entre comillas dobles. Ej.: “Esta es una cadena de texto”

##### Integer (int). Este tipo representa números enteros, los cuales pueden ser negativos o positivos. El léxico es solo escribir el número, ej.: 65

##### Float (float32, float64). Estos son los tipos de dato de coma flotante, es decir, decimales. El tipo float32 se usa para precisión simple y el tipo float64 para precisión doble. Su léxico son dos números separados por un punto (.), ej., float32: 1.75, float64: 3.1416

#### Tipos de datos compuestos. Son formados por la unión de uno o más valores mencionados en los tipos de datos básicos.

##### Array (array). Es una colección de elementos del mismo tipo de dato básico, su longitud es fija. Su léxico es la cantidad de elementos entre corchete seguido del tipo de datos que tendrá el array, por último, entre llaves irán los datos separados por comas (,), ej.: [4]int{5, 2, 8, 1}

##### Slice (slice). Es un tipo de dato similar al array, con la diferencia de que tienen un tamaño variable, es decir, son dinámicas en longitud. La única diferencia en su léxico es que no hay un número entre los corchetes, ej.: []int{5, 2, 8, 1}

##### Map (map). Es un tipo de dato en los que existen claves y valores, las claves dentro del map son únicas, los valores a los que se puede acceder son cualquier tipo de datos. Su léxico es map[key]value{}, siendo key, el tipo de dato que se usará para acceder al valor tiene que ser inmutable como int o string, y value el tipo de dato del valor, este puede ser cualquier tipo de dato, ej.: map[string]int{"Juan": 28, "Ana": 30}

### Palabras Reservadas

Las palabras reservadas en Go son consideradas “duras” ya que no se pueden usar como identificadores, teniendo un significado fijo y específico dentro del lenguaje de programación. Algunas de ellas son:

**`import`**. Sirve para la importación de paquetes externos.

#### `package`. Define el paquete al que pertenece el código fuente.

#### `func`. Sirve para definir una función.

#### `var`. Sirve para declarar una variable.

#### `const`. Sirve para declarar una constante.

#### `return`. Se usa dentro de las funciones para retornar un valor.

#### `if, else`. Se usan para crear estructuras de decisión.

#### `for`. Usada al momento de implementar un bucle.

#### `switch`, `case`, `default`. Se utilizan para crear estructuras de control de selección.

#### `defer`. Se usa cuando se desea una sección de código que se ejecute al terminar una función, como parte de la limpieza de recursos.

### Símbolos o Caracteres Especiales

Los operadores son símbolos para realizar operaciones matemáticas, ya sea de aritmética, relacional o lógicas, así mismo también hay de asignación.

#### Operaciones aritméticas. Suma (+), resta (-), multiplicación (-), división(/), módulo (%), incremento en 1 (++), decremento en 1 (--).

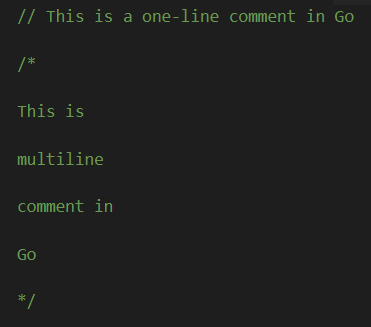
#### Operaciones relacionales. Igualdad (==), diferencia (!=), mayor que (>), menor que (<), mayor o igual que (>=), menor o igual que (<=).

#### Operaciones lógicas: Conjunción (&&), disyunción (||), negación (!).

#### Operaciones de asignación: Asignación simple (=), suma y asignación (+=), resta y asignación (-=), multiplicación y asignación (\*=), división y asignación (/=), módulo y asignación (%=).

### Comentarios

En Go existen dos tipos de comentarios, comentarios de una sola línea y comentarios multilínea. Para agregar un comentario de una línea debemos usar el doble slash (//) en cambio para los comentarios multilínea se usa slash asterisco para abrir, y asterisco slash para cerrar el comentario multilínea.



## Reglas de sintaxis

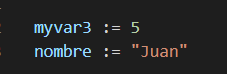
### Definición de variables

En el lenguaje de programación GO, la asignación y la inicialización de valores se las puede realizar de las siguientes maneras:

**Asignación de valores**

Para poder declarar una variable tenemos varias formas, que se dividen en dos grupos, la forma larga y la forma corta. La forma larga consta de declarar la variable **var** seguido del nombre de la variable, sigue el símbolo igual con el tipo de dato. Un ejemplo con el tipo de dato integer (int) es:

Asignación Simple:Puedes hacer una asignación usando el operador **“:=”** que infiere automáticamente el tipo de datos de la variable. El uso de este operador solo es válido dentro de una función y se utiliza para declarar e inicializar variables locales.



Como se puede observar, no hemos dado un tipo de dato en la declaración de la variable, sin embargo, Go infiere que tipo de dato vamos a usar, a pesar de esto, se mencionó que este lenguaje es de tipado fuerte, no se podrá cambiar el tipo de dato inferido en la variable, es decir que no podremos asignar un valor de otro tipo de dato a esta misma variable. Esto también puede hacerse para la declaración larga de variable siempre y cuando se asigne el valor en la misma línea de en la que se declara la variable.

**Asignación Explicita:** Si deseas asignar un valor a una variable especificando el tipo de dato se puede usar la sintaxis “var nombre tipo = valor”.



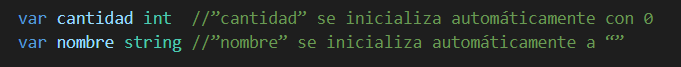
Esta forma larga sirve para poder declarar variables sin inicializarla en la misma línea:



No obstante, se ha dado como buena práctica en la comunidad de Go usar la forma larga de declaración de variable solo cuando no se inicialice la variable con un valor en la misma línea, caso contrario, usaremos la forma corta de declaración de variable cuando podamos o debamos asignarle un valor en cuanto declaremos la variable.

**Inicialización de valores:**

**Inicialización automática:** En el lenguaje GO se asigna de forma predeterminada un valor a las variables que no tienen un valor ya asignado. Como ejemplo se tienen las variables numéricas que se inicializan con 0 o las cadenas de texto con una cadena vacía.



**Inicialización expresa:** Al declarar una variable, puedes asignarle un valor al mismo tiempo. Esto se hace especificando el valor después del tipo de datos utilizando el operador “=”



**Inicialización múltiple:** Se tiene la opción de inicializar varias variables en una sola línea de código. Los valores se separan por comas y se corresponden con las variables en el mismo orden



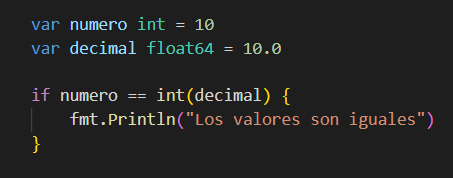
Es fundamental tomar en cuenta que una vez que una variable es declarada debe ser utilizada en el programa, de lo contrario GO lanzará un error de compilación.

### Comparaciones de valores

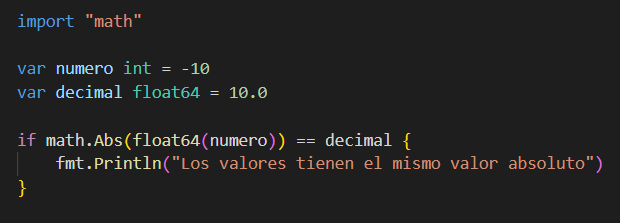
Comparar valores en Go puede ser un poco complicado debido a la fuerte tipificación estática del lenguaje, es decir que los tipos de datos de las variables se determinan en tiempo de compilación y no pueden cambiar durante la ejecución del programa. Sin embargo, existen diferentes formas de realizar comparaciones entre tipos diferentes.

A computer screen with numbers and symbols

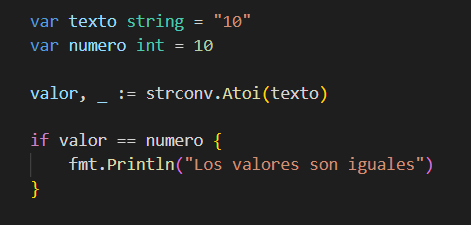
Description automatically generated**Comparación directa:** Si los tipos de datos son los mismos o compatibles, se pueden comparar directamente entre ellos utilizando operadores de comparación como: **(“==”, “!=”, “<”, “>”, “<=”, “>=”).**

**Conversiones de Tipo (Type Assertion):** Se hace primero una conversión entre variables para que sea compatibles en tipo y se puedan comparar.

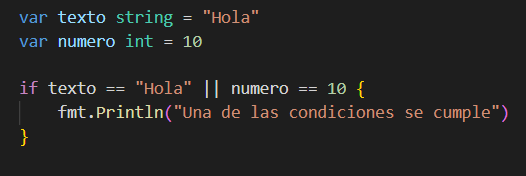
**Comparación de valor absoluto:** Se tiene la opción de comparar valores numéricos junto con la función “math.Abs()”. Ejemplo: decimal con un entero.



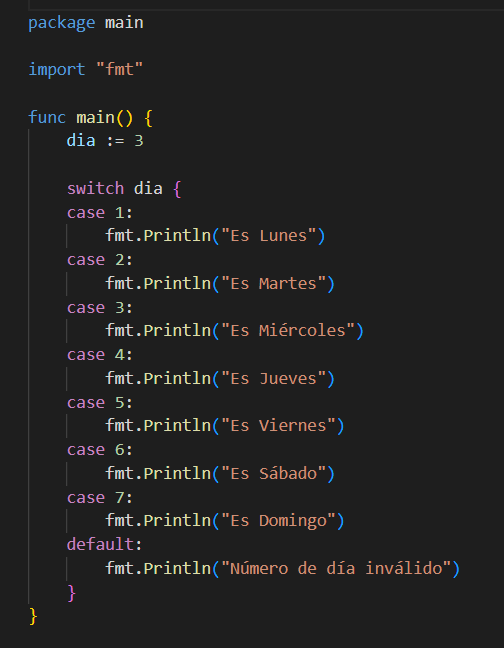
**Cadenas a Enteros o Flotantes:** Se pueden convertir cadenas a números para luego hacer la comparación.



**Comparación Lógica:** En esta comparación se hace uso de las comparaciones lógicas para determinar si los valores son iguales o cumplen con ciertas condiciones.



**Comparación con el operador “switch”:** El operador “switch” permite evaluar expresiones y ejecutar el bloque de código correspondiente en caso de que coincida con el valor de la expresión.

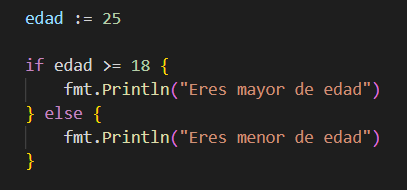


### Estructuras de control

Las estructuras de control más utilizadas en el lenguaje Go son:

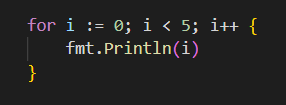
**If-else:**

La estructura “if-else” se utiliza para tomar decisiones en base a una condición en específico. Si la condición es verdadera (“true”), se ejecuta un bloque de código y si es (“false”) se ejecuta otro bloque de código.



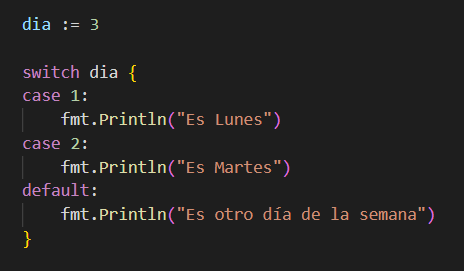
**For:**

La estructura “for” se utiliza para repetir un bloque de código en un numero especifico de veces o mientras una condición se cumpla. Puede usarse para implementar bucles finitos o indefinidos.



**Switch:**

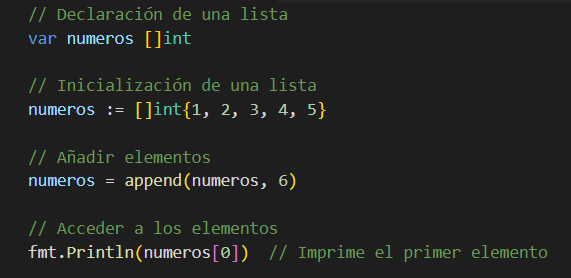
La estructura “switch” se utiliza para seleccionar entre varias opciones basadas en el valor de una expresión. Es una forma más limpia y eficiente de escribir múltiples “if-else if-else”.



### Estructuras de datos

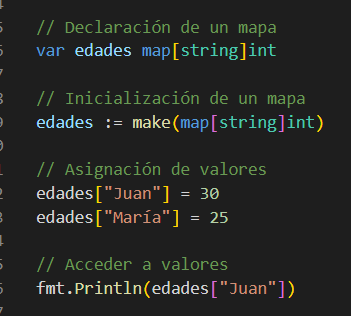
**Listas:**

Las listas son secuencias de elementos del mismo tipo, pero a diferencia de los arreglos su tamaño puede cambiar de forma dinámica. Son muy flexibles y se utiliza para almacenar y manipular colecciones de datos.



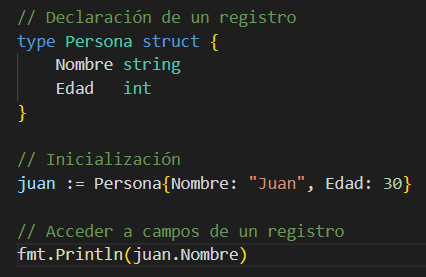
**Mapas:**

Los mapas son una estructura de datos que almacena pares de clave-valor, donde cada clave es única. Proporcionan una forma eficiente de buscar y recuperar valores a partir de una clave dada.



**Registros:**

Los registros o estructuras permiten agrupar diferentes tipos de datos desde un mismo nombre, en si es una forma de definir un nuevo tipo de dato personalizado. Son fundamentales para crear tipos de datos complejos y representar objetos.

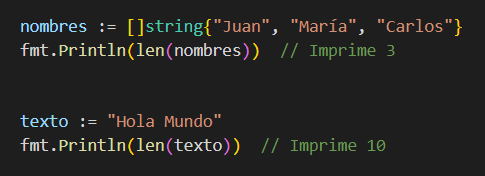


**Operaciones básicas:**

**Longitud, impresión, ingreso por teclado.**

**Longitud de una lista o String:**

Para obtener la longitud de una lista o de una cadena de texto en Go, se utiliza la función “len()”. Esta función devuelve el número de elementos que están en la lista o la cantidad de caracteres en una cadena.

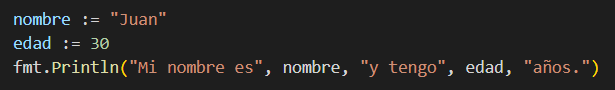


**Impresión en pantalla:**

Para poder imprimir en la consola en Go, se utiliza la función “fmt.Println()”.

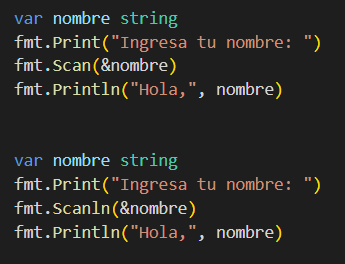


También se tiene la opción de agregar variables en la función de la siguiente manera:



**Ingreso por teclado:**

Para recibir la entrada del usuario desde la consola de Go, se debe utilizar el paquete “fmt” y la función “Scan” o “Scanln” que permiten capturar y almacenar el valor ingresado en una variable.



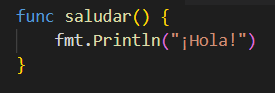
En los ejemplos se muestra “&” que se utiliza para obtener la dirección de memoria de la variable donde se almacenara la entrada del usuario.

### Declaración de funciones:

Para definir un conjunto de acciones se hacen uso de las declaraciones de funciones que pueden ser ejecutadas en un momento posterior dentro del programa. En sí, una declaración de funciones proporciona un nombre para un conjunto de código y puede aceptar datos de entrada y devolver un resultado.

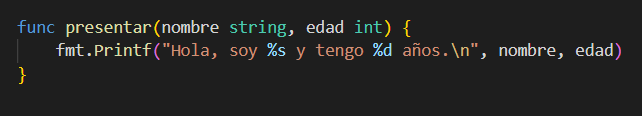
**Función sin parámetros ni valor de retorno:**

En esta declaración la función no toma ningún parámetro y no devuelve ningún valor.



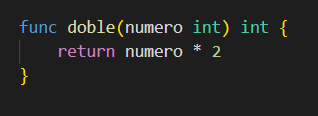
**Función con parámetros, pero sin valor de retorno:**

En el siguiente ejemplo se puede ver que la función toma dos parámetros (“nombre”, “edad”) pero no tiene ningún valor de retorno.



**Función con parámetros y valor de retorno:**

En el ejemplo se puede ver que la función toma un parámetro (“numero”) y devuelve el doble del valor ingresado.



## Describa reglas semánticas

* + **Declaración de variables y Ámbito(Scope)**
    - Las variables se declaran utilizando la palabra reservada “var”, seguida del nombre de la variable, por ejemplo: “var x int”
    - No se puede utilizar una variable sin ser declarada.
    - La variable solo es visible dentro la función, ámbito o bloque donde fue declara.
* **Firma de Funciones y Valores de retorno.**
  + Las funciones se declaran con la palabra reservada “func”, seguida del nombre de la función, la lista de parámetros entre paréntesis y los tipos de retorno, si existieran.
  + La firma de la función incluye tanto los tipos de parámetros como el tipo de retorno.
  + Texto

    Descripción generada automáticamenteSi se ingresa argumentos con otro tipado o esperas un tipo de retorno diferente habrá un error.

* **Importación de paquetes.**
* Los paquetes o bibliotecas de código se importan utilizando la palabra reservada “import”.
* Los nombres de los paquetes que se importan deben coincidir con el nombre entre comillas definido en el directorio.
* Se importa todos los paquetes que se necesite antes de utilizarlos.

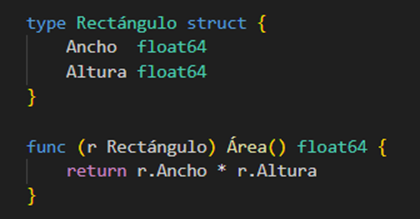
* **Interfaz y polimorfismo.**
  + Las interfaces son un tipo especial que define un conjunto de métodos que un tipo debe implementar para cumplir con la interfaz. Por esto se da el polimorfismo, donde varias estructuras diferentes pueden cumplir con una misma interfaz y por lo tanto ser tratadas de forma uniforme.
  + Texto

    Descripción generada automáticamente con confianza mediaLas interfaces en Golang permiten escribir código genérico y reutilizable.
* **Uso de punteros.**
  + Son variables que almacenan direcciones de memoria.
  + Los punteros son utilizados para pasar referencias en lugar de copias, lo que es útil para mejora el rendimiento y modificación de los valores originales.
  + No se permite la aritmética de punteros.
  + Gracias al garbage collector se simplifica la gestión de memoria.Interfaz de usuario gráfica, Texto

    Descripción generada automáticamente
* **Mapeo de errores.**
  + Texto

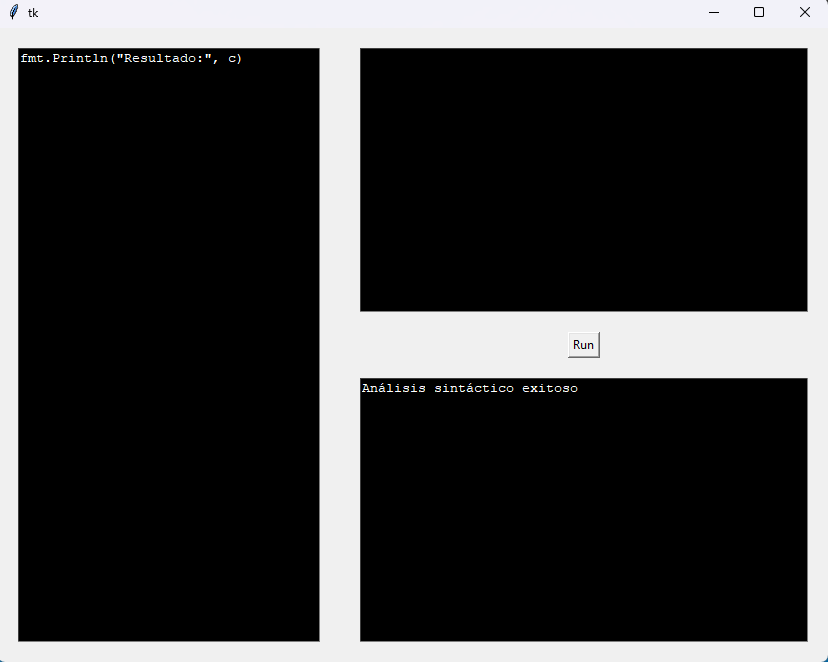
    Descripción generada automáticamenteLos errores se representan mediante el tipo “error” junto con el resultado de la función para manejar las situaciones inusuales., por ejemplo:
* **Goroutines y Concurrencia.**
  + Los Gouroutines son hilos de ejecución ligeros que permiten la concurrencia.
  + Se inician Gouroutines para ejecutar funciones en paralelo.
  + Logotipo

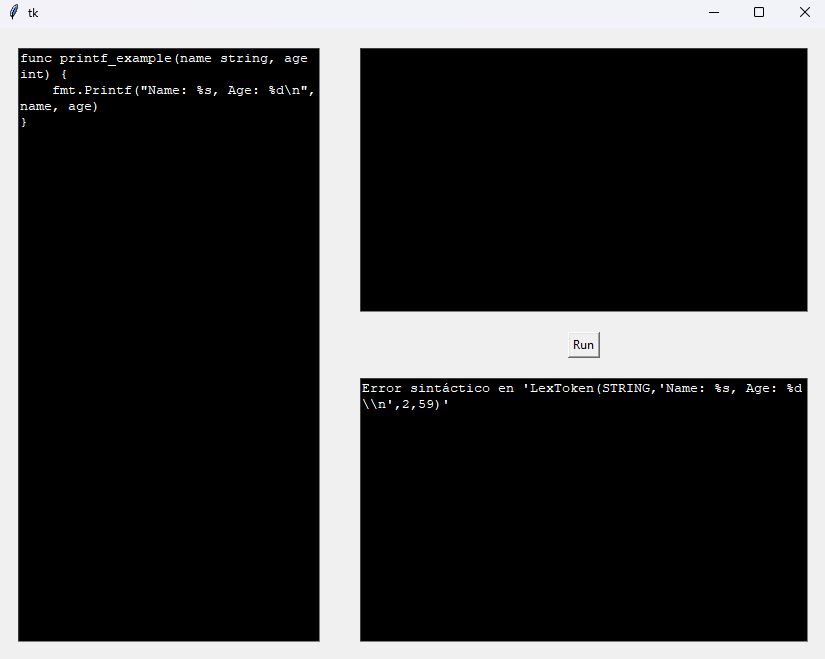
    Descripción generada automáticamenteTiene un mecanismo de comunicación entre gouroutines llamados canales, lo que permite la sincronización y compartir acceso a datos sin caer en deadlock.
* **Métodos y receptores.**
  + Los métodos son funciones asociadas a un tipo especifico y se utilizan para realizar operaciones en valores de ese tipo.
  + Los métodos permiten la encapsulación de comportamiento relacionado con un tipo y se declaran con un receptor como argumento, por ejemplo:



## 

## Interfaz gráfica





La propuesta de nuestra interfaz gráfica se compone de una entrada por texto para el código fuente, un botón para iniciar el proceso de compilado, una sección donde se mostrará el análisis léxico del código ingresado, y una ventana donde se mostrará el análisis semántico como paso final del proceso.

**Conclusiones.**

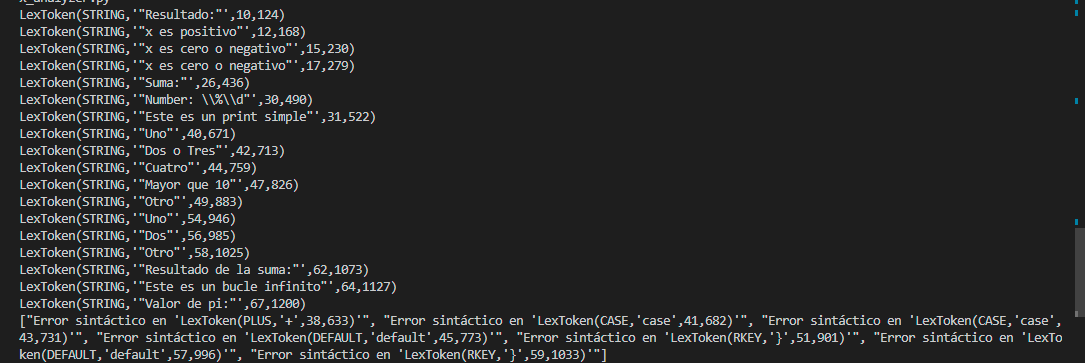
Se ha profundizado en la utilidad y el valor de la herramienta PLY (Python Lex-Yacc) en el ámbito especifico de la creación de analizadores léxicos y sintácticos. Esta herramienta ha proporcionado una perspectiva integral sobre el funcionamiento interno de los lenguajes de programación, destacando la importancia de la validación de gramática para comprender y estructurar el código de manera eficiente. dando apertura a la posibilidad de construcción de un nuevo lenguaje gracias a esta herramienta integral para desarrolladores de software, ya que permite la creación desde compiladores y analizadores para aplicaciones complejas. En el análisis de la gramática de Golang, PLY ha facilitado la profundización en la comprensión de reglas léxicas y sintácticas, el procesamiento de código fuente básicos, este enfoque ha mejorado significativamente la capacidad de aprendizaje de nuevos lenguajes de programación y a la vez conocer su estructura y reglas. Además, con herramientas de interfaz gráfica como tkinter, que permite a los desarrolladores crear desde ventanas, botones y otros elementos GUI. Es posible mostrar los resultados del proyecto de analizador de manera clara y efectiva.

**Sintaxis del Lenguaje de Programación**

En este proyecto, se ha desarrollado un analizador para el lenguaje de programación GO, con la ayuda de la herramienta PLY (Python Lex-Yacc). Se ha abordado aspectos fundamentales de la creación de un lenguaje de programación, desde el análisis léxico hasta el análisis sintáctico y semántico.

Se han implementado un conjunto de diversas funcionalidades que a desde la declaración de variables y constantes hasta las estructuras de control avanzadas como if, else, switch, y bucles for. Además, el lenguaje GO admite la definición y llamada de funciones, así como operaciones aritméticas y lógicas. Todo esto se ha logrado gracias a una gramática bien definida y al uso de PLY para la generación de analizadores sintácticos.

**Resultados del proyecto**



Como se puede ver hay algunos tokens que no se han usado, pero si implementado como: AMPERSAND, AND y otros. Por otro lado, algunas líneas reflejan errores sintácticos que podrían ser a problemas en la gramática o en la entrada de datos que se está procesando.

Algunas reglas como assignment\_semantic y rule\_comparation se ve que están definidas, pero no se están usando en el código. En este caso simplemente se pueden eliminar estos tokens para reducir la complejidad del lexer.

A pesar de algunos errores, se ve que se generan las tablas LALR para el analizador sintactico y genera resultados de interpretación para ciertas partes del código fuente.

**Herramientas utilizadas**

## En este proyecto específico, se han utilizado varias herramientas y tecnologías para implementar el analizador para un lenguaje de programación inspirado en Go con PLY (Python Lex-Yacc). A continuación, se detallan las principales herramientas utilizadas:

## 1. PLY (Python Lex-Yacc): PLY es una herramienta que proporciona las capacidades de análisis léxico y sintáctico de Lex y Yacc, pero implementadas completamente en Python. Se ha empleado PLY para definir las reglas gramaticales, tokens, y construir el analizador léxico y sintáctico para el lenguaje.

## 2. Python: Python se utiliza como el lenguaje principal para implementar el analizador léxico y sintáctico. PLY es una biblioteca de Python, y Python en sí mismo es una elección común para tareas de procesamiento de lenguajes y desarrollo de compiladores.

## 3. Visual Studio Code (VSCode): VSCode u otro entorno similar puede haberse utilizado para escribir, organizar y depurar el código fuente del proyecto.

## 4. Git: Control de Versiones: Git podría haberse utilizado para realizar un seguimiento del historial de cambios en el código fuente, facilitando la colaboración y el mantenimiento del proyecto.

## 5. GitHub: GitHub podría haberse utilizado para alojar y compartir el código fuente del proyecto, permitiendo la colaboración y el seguimiento de problemas.

# Bibliography

Hancox, K., & Guides, G. (2020, February 7). *Digital Ocean*. Retrieved from Cómo usar variables y constantes en Go: https://www.digitalocean.com/community/tutorials/how-to-use-variables-and-constants-in-go-es

Moure, B. (July de 2023). *GitHub*. Obtenido de mouredev - one-day-one-language: https://github.com/mouredev/one-day-one-language/tree/main/08%20-%20Go

Oregoom. (n.d.). *Oregoom*. Retrieved from Tipos de Datos en Go: https://oregoom.com/go/tipos-de-datos/

Ruelas, U. (2016, 05 13). *coding or not*. Retrieved from Go to go: sintaxis, tipo de datos y palabras reservadas: https://codingornot.com/02-go-to-go-sintaxis-tipo-de-datos-y-palabras-reservadas

Zepeda, E. (2023, October 11). *Go, lenguaje de programación, introducción a variables y tipos de datos*. Coffee Bytes. <https://coffeebytes.dev/go-lenguaje-de-programacion-introduccion-a-variables-y-tipos-de-datos/>

*Estructuras de control de flujo en Go — Steemit*. (n.d.). Steemit. <https://steemit.com/cervantes/@orlmicron/estructuras-de-control-de-flujo-en-go>

Nivardo. (2023, March 31). *Operadores en Go*. Go. <https://oregoom.com/go/operadores/>

*The Go Programming Langu*age Specification - The Go Programming Language. (n.d.). <https://go.dev/ref/spec>

*PLY (Python Lex-Yacc) — ply 4.0 documentation*. (n.d.). <https://ply.readthedocs.io/en/latest/ply.html#ply-overview>