**¿Qué es Go?**

Es un leguaje open source creado por Google, su aparición fue en el año 2009 y sus diseñadores son Robert Griesemer, Rob Pike y Ken Thompson. Además es un lenguaje de **programación concurrente, compilado, estructurado, tipado estático e imperativo.** También no es un lenguaje orientado a objetos y cuenta con un recolector de basura.

**Características de Go**

* Go utiliza un sintaxis parecida a la de C.
* Posee un tipado estático y resulta ser igual de eficiente que C.
* Tiene muchas características y facilidades de lenguajes dinámico como Python.
* No es orientado a objetos debido a que no existe una jerarquía de tipos, pero si implementa la interface.
* Declaraciones de variables simple.
* Duck typing – Tipificación dinámica de datos.
* No tiene excepciones.

**Declaración de variables**

Para declarar una variable se usa la palabra reservada VAR, haciendo referencia que es una variable después el nombre de la variable y por último el tipo.

A diferencia de otros lenguajes de programación la declaración en GO es al revés. GO inicializa la variable con un valor por defecto.



Figure 1: Declaración de variable en Go

En este caso **numero,** Go lo inicializará con el valor de 0 pero podemos asignarle un valor por defecto.

Figure 2: Asignación de variable

Para declarar una variable de manera rápida determinando el tipo de manera dinámica una vez que la variable se le asigna una tipo este ya no se puede cambiar el operador **:=** solo se cuándo se declara una nueva variable.



Figure 3: Declaración rápida de una variable

Otra forma de declarar variables omitiendo el tipo es usando **var + nombre** y dándole una asignación.

Figure 4: Otra forma de declaración usando el Duck typing

En GO se pueden asignar valores de forma consecutiva.



Figure 5: Asignación de valores de forma consecutiva

Esta asignación de valores puede resultar útil al momento de intercambiar valores entre variables del mismo tipo.



Figure 6: Intercambio de valores entre variables



Figure 7: Variables Intercambiadas

Esta forma de asignación de valores se puede utilizar cuando se declara una nueva variable.

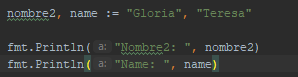


Figure 8: Declaración de una variable y asignación

En este caso **nombre2** es la variable creada y su valor es **Gloria**, mientras que **name** era una variable previamente declarada por lo cual su valor asignado es **Teresa**.

Para declarar múltiples variables se usa **var(),** dentro de los paréntesis se declaran las variables con sus asignaciones. Al usar esta forma de declaración se puede asignar el tipo o bien usando el Duck typing (tipado dinámico).

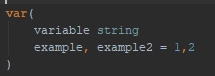


Figure 9: Declaración de múltiples variables

**NOTAS:**

En **Go** toda variable que se es declarada debe de ser utilizada de la misma forma sucede con los **import** sino se usa una librería importada el compilador obliga a eliminarla o usarla.

La librería fmt es una abreviación de format, esta se usa para poder imprimir e ingresar datos.

**Alcance de las variables**

En Go el alcance de las variables es como cualquier otro lenguaje de programación es decir las variables pueden funcionar de manera global o de manera local.

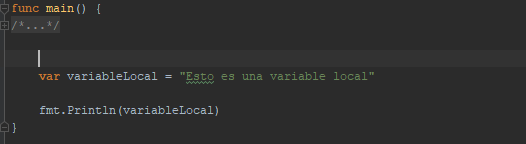


Figure 10: Ejemplo de variable local

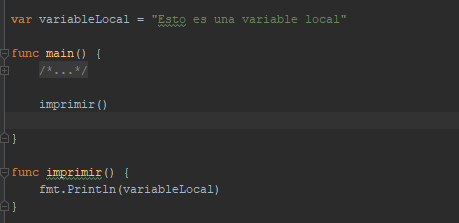


Figure 11: Ejemplo de variable global

**NOTAS:**

Cuando se declara una variable global no se permite usar el operador **:=** por lo que es obligado declararla de forma apropiada.

**Tipos de datos en Go**

Go posee 4 categorias de datos: básico, conjuntos o sets, referencias e interfaces.

Comencemos por los tipos básicos, estos pueden ser **números (int), booleanos y cadenas.**

Tipos numéricos:

|  |  |  |
| --- | --- | --- |
| Tipo | Rango | Consumo de memoria |
| uint8 | 0 a 255 | 1 byte |
| uint16 | 0 a 65535 | 2 byte |
| uint32 | 0 a 4294967295 | 4 byte |
| uint64 | 0 a 18446744073709551615 | 8 byte |

**Enteros sin signo**

**Enteros con signo**

|  |  |  |
| --- | --- | --- |
| Tipo | Rango | Consumo de memoria |
| int8 | -128 a 127 | 1 byte |
| int16 | -32768 a 32767 | 2 byte |
| int32 | -2147483648 a 2147483647 | 4 byte |
| int64 | -9223372036854775808 a 9223372036854775807 | 8 byte |

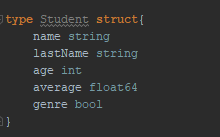
**Alias**

|  |  |  |
| --- | --- | --- |
| Tipo | Rango | Consumo de memoria |
| byte (lo mismo que uint8) | 0 a 255 | 1 byte |
| rune (lo mismo que int32) | -2147483648 a 2147483647 | 4 byte |

**Tipos dependientes de la plataforma**

|  |  |
| --- | --- |
| Tipo | Rango |
| uint | De pende de la plataforma (32bits o 64 bits) |
| int | De pende de la plataforma (32bits o 64 bits) |
| uintptr | Entero suficientemente largo para almacenar un puntero |

En los tipos de conjuntos tenemos los **arreglos** y las **estructuras.**



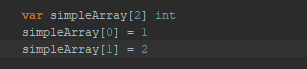


Figure 13: Ejemplo de un array con su inicialización

Figure 12: Ejemplo de una estructura

**Casting de valores**

En Go el casting tiene que ser de forma explicito debido que Go no es un lenguaje de gran potencia como lo son Java o C# que se puede hacer un casteo de forma implícita.