****

**Universidad Alva Edison**

**Sistemas Computacionales**

**Programación Avanzada**

**Mejia Cholula Brenda Lisset**

**Amaro Ortega Axel**

**Profesor : Ma. Raquel Guevara Sotomayor**

**25/09/2019**

**Introducción**

Este proyecto consiste en la investigación acerca de la instalación, declaración y programación en el lenguaje de programación de GO, tomando en cuenta Git y Github para realizar repositorios y así poder alojar proyectos utilizando el sistema de control de versiones.

**GO**

* Producto realizado por Google 2007
* Primera versión
* Producción → Sabemos lo que vamos a obtener / Código
* Libera memoria
* GIT Sistema de control de versiones, sistema de control distribuido

**Las características son:**

* Base de Pascal en su sintaxis
* Lenguaje de compilación
* Multiplataforma
* De fácil transportación
* Recolector de basura y eliminación y liberación de espacio en la RAM de sus variables
* No existen excepciones se llaman errores
* Es nativo de UTF-8 por lo que no existen ningún impedimento en el manejo de código con caracteres especiales

**Herramientas necesarias de Go:**

Descargar e instalar lo siguiente como se va presentando a continuación:

* Go: golang.ar
* Git git-sam.com/download/win

(Para windows)

* Atom: atom.io

**El orden de instalación es:**

1. GIT
2. GO
3. ATOM.

**ATOM**

1. File / setting / install
2. En el área de buscar escribir “atom-i18n”
3. Setting
4. Language → spanish

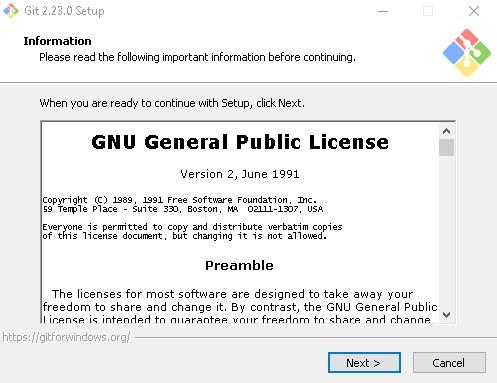
**PACKAGE**

**A**rchivo / ajustes / package

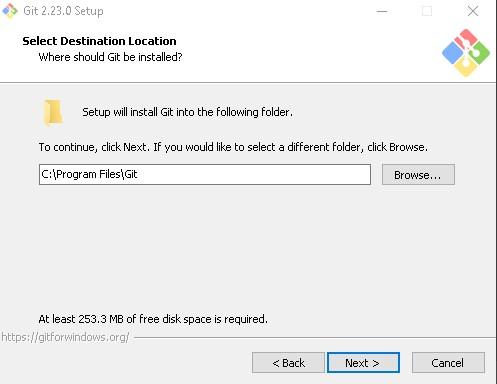
Buscar → Go-plus → instalar

**Instalación Go**

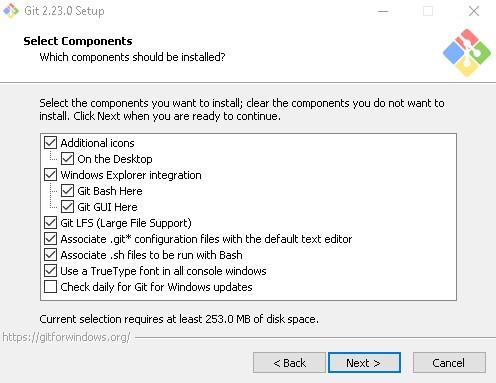
* Ejecutar como Administrador.



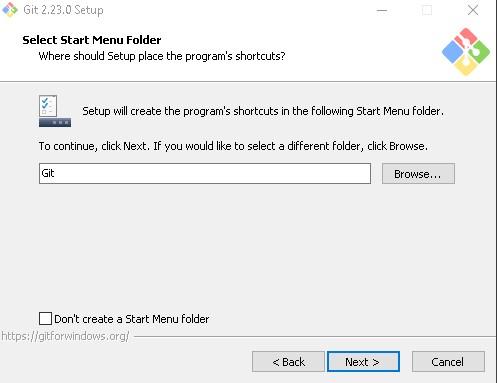
* Respetar la ubicación por Default



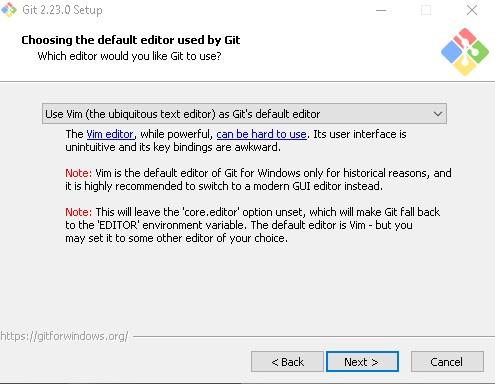
* Marcar las opciones desmarcadas de “Aditional Icons” y “Use a TrueType Font in all console Windows”



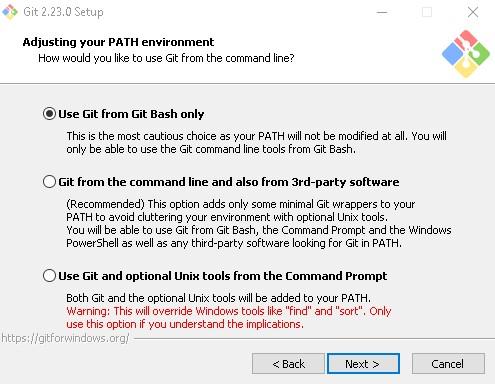
* Dejar el Folder del menú de inicio sin modificar.



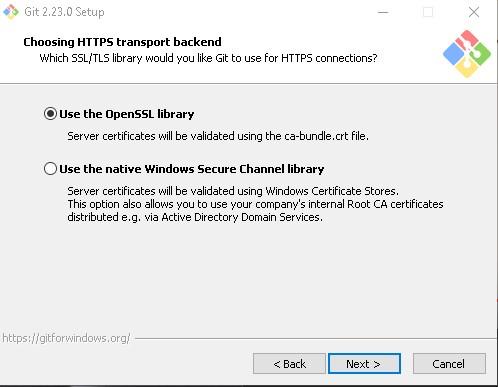
* Al desplegar combo box dejamos la opción que viene por default.



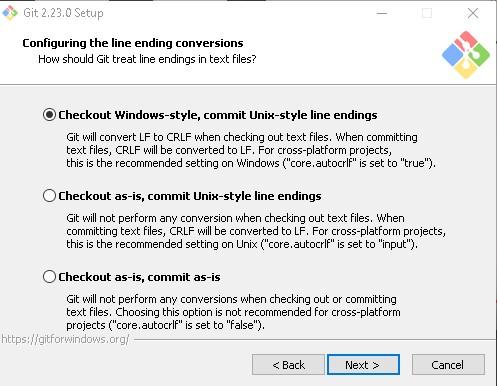
* La línea de comandos, la dejamos con la opción de “Use Git from Git Bash only”



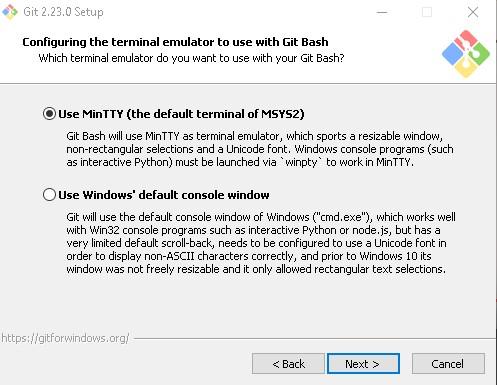
* La librería del “HTTPS Transport backend” dejar por default.



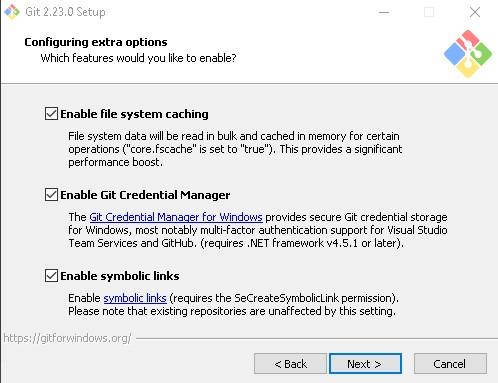
* Los estilos de windows dejar por default



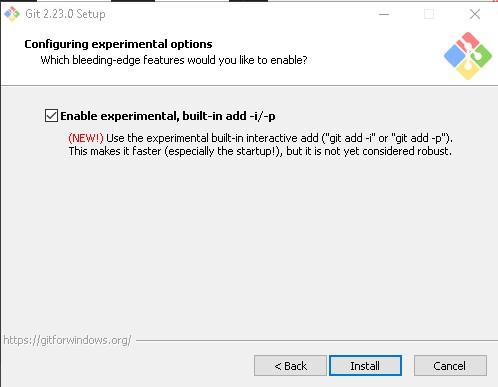
* Dejar la consola por default



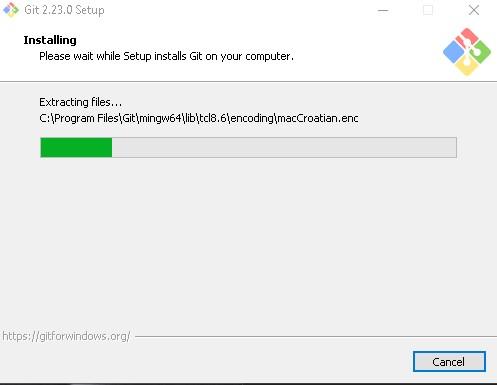
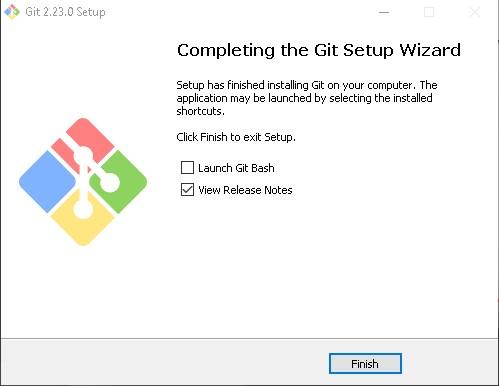
* Activar los tres puntos



* Tener marcado el recuadro de “built -in add -i/-p”

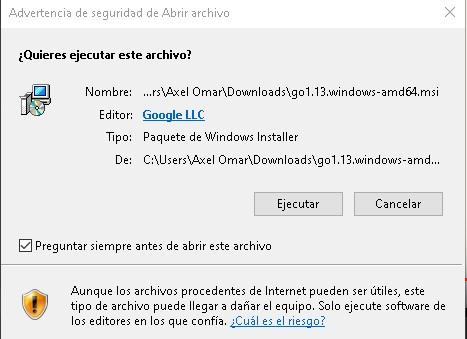


* Dejar instalar el programa y finalizar.

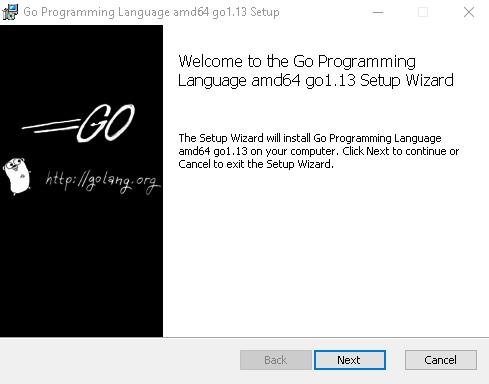
 

**Instalación de Go**

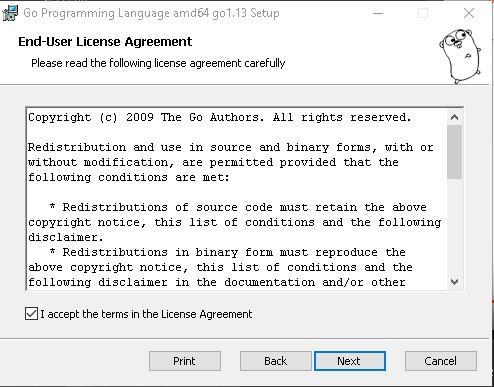
* Ejecutar



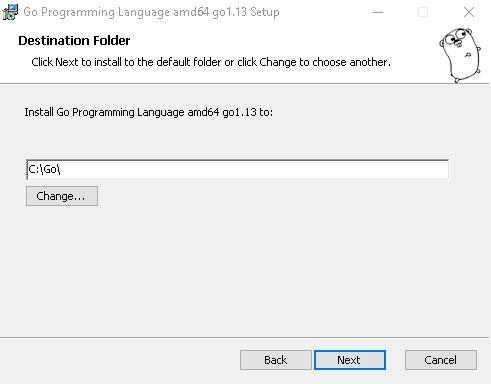
* Clickear en siguiente



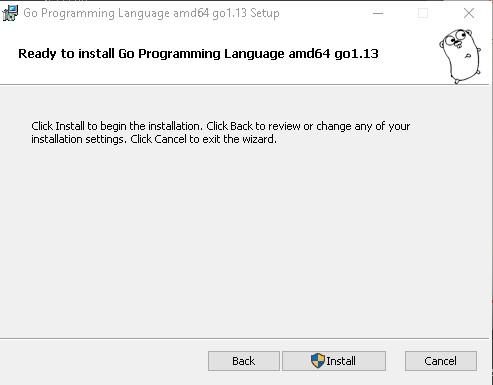
* Aceptar los términos



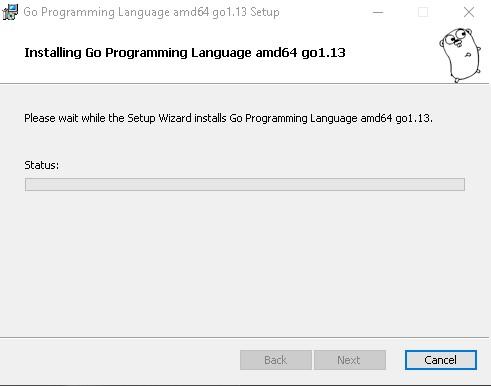
* Dejar la dirección de la carpeta por default



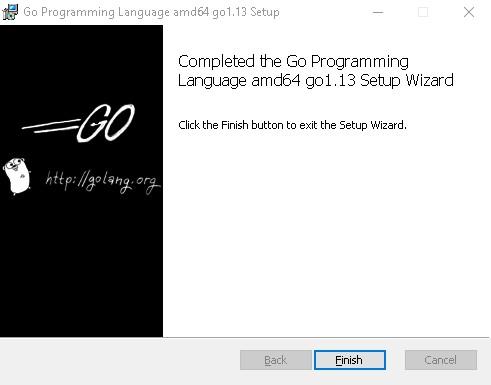
* Instalar el programa



* Esperar a que finalice la instalación.

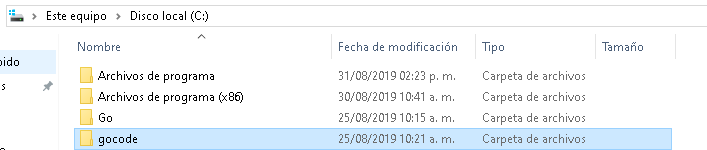


* Finalizar la instalación.



**Crear la carpeta para guardar los Trabajos.**

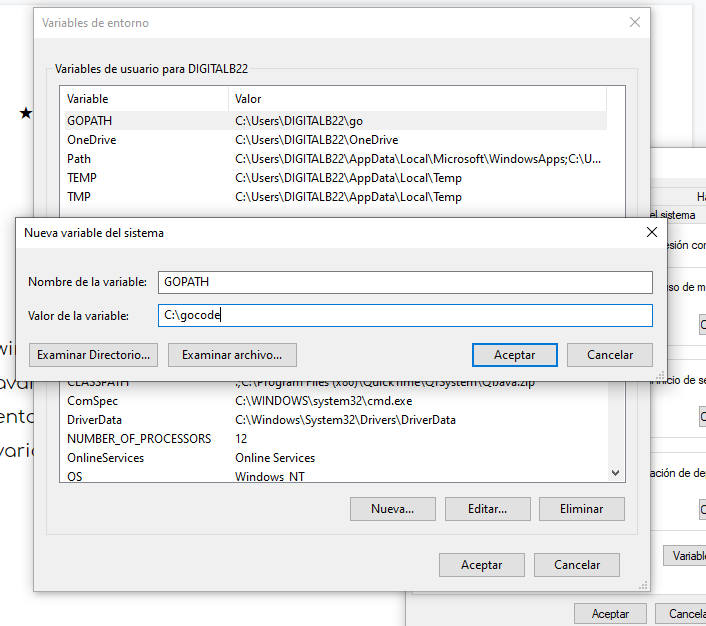
* Crear carpeta contenedora de código en la unidad C: con el nombre “gocode”



* Crear la siguiente estructura en la carpeta de gocode
  + c:/ gocode
    - src (código fuente)
      * pkg (archivo de paquetes)
        + bin

**VARIABLES DE ENTORNO**

windows / panel de control / buscar “sistema” / configuraciòn avanzada de sistema/ opciones avanzadas/ variables de entorno/ nueva/ windows + e “C:\gocode”/ nombre de la variable “ GOPATH”



**GO VERSIÓN : CONTROLADOR DE VERSIONES.**

package main → iniciando paquetes

import "fmt" → mandar a escribir en pantalla

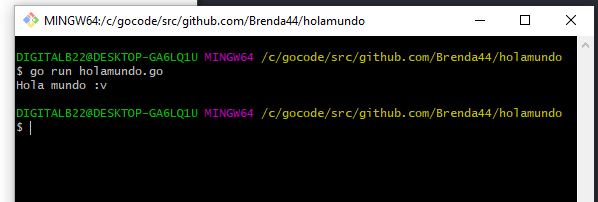
func main → manejo por funciones

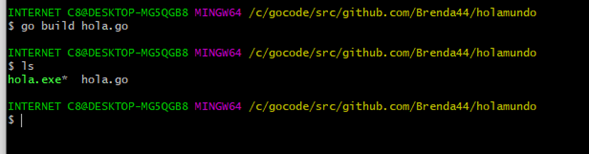
Para imprimir mandas el fmt.Println(“”)

No se utiliza “;”

go run “nombre del archivo” → para correrlo

el ejecutable go build nombre del archivo





crsgithubgolang-es/gocursom-go

Para ingresar a un manual de GO: <https://golang.org/ref/spec#types>

**SINTAXIS PARA AGREGAR VARIABLES EN ATOM:**

var nombre tipo\_dato

var nombre1 nombre2 tipo de dato

nombre: = valor ← le dices a la variable que ese tipo de valor

nombre1, nombre2 := valor1,valor2

**Condicionales:**

**IF**

* Instrucción inicial en condición

if isValid:=true; isValid{}

**SWITCH**

Ejemplo 1;

switch variable {

case:

instrucción

default:

instrucción

}

Ejemplo 2:

switch variable {

case \_\_:

fallthrough

}

Ejemplo 3:

switch a: = 3; {

case a>=0 && a>=5:

introducción

}

**FOR**

for i: = Valor; condición; incremento{

Instruccion

}

**TRABAJOS DE GO**

* Al empezar, se necesita llamar la paqueteria del main, para eso lo hacemos escribiendo “**package main**”, y a continuación llamamos una función de la misma paqueteria, que en este caso es “fmt”. Esto se llogra escribiendo ‘**import "fmt"’**
* Después escribiremos la función donde estará nuestro código, esto lo lograremos escribiendo la siguiente sintaxis:   
  **func main(){  
  }**

Ejercicio 1:

Estructura principal de GO.

Declarar usando, primeramente

eso declarando variables, nombre, apellido materno y paterno.

edad, cuota 'lo que pagamos en la escuela', acreditar.

**package main**

**import "fmt"**

**func main(){**

**Nom, ApePat, ApeMat, Edad, Cuo, Acr:= "Axel", "Amaro", "Ortega", 22, 1500, false**

**fmt.Println(Nom, ApePat, ApeMat, Edad, Cuo, Acr)**

**}**

1. Dentro de la función principal, declararemos las variables ya con sus valores respectivos de string, int y booleano de la siguiente manera:   
   **Nom, ApePat, ApeMat, Edad, Cuo, Acr:= "Axel", "Amaro", "Ortega", 22, 1500, false**
2. Por último, mandaremos a imprimir a consola usando la función de la paquetería que importamos, el **"fmt"** de la siguiente manera:   
   **fmt.Println(Nom, ApePat, ApeMat, Edad, Cuo, Acr)**
3. Con esto, imprimimos los valores de cada variable en una sola línea de código.

Ejercicio 2:

Indicar el nombre del alumno, edad, cuánto paga arriba de los 6 mil pesos se escribe caso contrario o no.

**package main**

**import "fmt"**

**func main(){**

**Nombre, Ap\_Pater, Ap\_Mater, Edad, Cuota, Acreditado := "Brenda", "Mejia", "cholula", 21 , 210, true**

**if Acreditado == true{**

**if Cuota > 2000{**

**fmt.Println(Nombre, Ap\_Pater, Ap\_Mater, Edad, Cuota, Acreditado)**

**}else{**

**fmt.Println("No es la cuota")**

**}**

**}else{**

**fmt.Println("No acredita")**

**}**

**}**

1. Insertamos un “if” donde escribiremos nuestra condición de “Si la variable ‘Acreditar’ es igual a ‘true’”, se anidará con otro if en donde escribiremos otra condición, el cual será “Si la variable Cuota es mayor al valor asignado”.

if Acreditado == true{

if Cuota > 2000{

}

}

2. Por consiguiente, dentro de las llaves del segundo If, escribiremos la instrucción

“fmt.Println(Nombre, Ap\_Pater, Ap\_Mater, Edad, Cuota, Acreditado)”

3. Después de la llave del segundo if, se escribe el “Else”, este término hará que en caso de que no se cumpla la condición ejecute la excepción.

else{

fmt.Println("No es la cuota")

}

4. Y para finalizar, en la llave final del primer “if” insertamos el siguiente else:

else{

fmt.Println("No acredita")

}

5. Si las dos condiciones se cumplen, imprimirán las variables, en caso de que la primera condición no se cumpla, mostrará un mensaje donde indique que la cuota no es la indicada y si la segunda condición no se cumple, mostrará otro mensaje en la cual indicará que no acredite.

Ejercicio 3:

**func main(){**

**Nom, ApePat, ApeMat, Edad, Cuo, Acr := "Axel", "Amaro", "Ortega", 22, 2500.00, true**

**fmt.Println("La cuota incrementara cada mes un 5% al Mes")**

**fmt.Println(Nom, ApePat, ApeMat, Edad, Cuo, Acr)**

**var Inc, Cuo\_Fin float64**

**for i := 0; i < 12; i++ {**

**Inc= .5 \* Cuo**

**Cuo\_Fin = Cuo + Inc**

**}**

**fmt.Println(Cuo\_Fin)**

**}**

**1.** Se visualizará con la sentencia de “for” como irá cambiando e incrementando el valor de la cuota en cada mes.

Ejercicio 4:

**package main**

**import "fmt"**

**func main(){**

**Cuota, mensaje := 2000.0,"La cuota aumentara un 5% en cada mes:"**

**fmt.Println(mensaje, Cuota)**

**var meses =[] string {"Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio", "Julio", "Agosto", "Septiembre", "Octubre", "Noviembre", "Diciembre"}**

**i :=0**

**for i != 12{**

**i++**

**var resultado float64 = (Cuota \* 0.05) + Cuota**

**Cuota = resultado**

**fmt.Println(meses[i])**

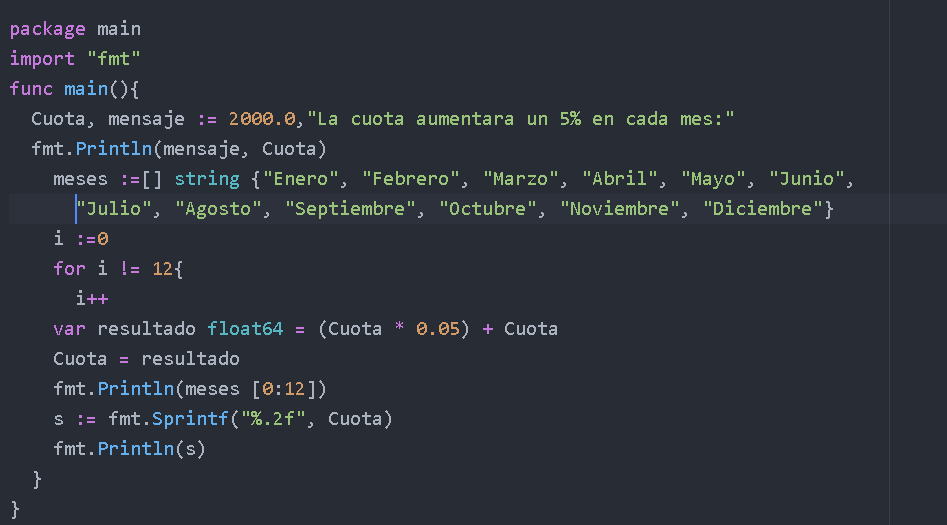
**s := fmt.Sprintf("%.2f", Cuota)**

**fmt.Println(s)**

**}**

**}**

**1. fmt.Sprintf("%.2f", Cuota) ←** En esta declaración al momento de ejecutar una operación, si tiene muchos puntos decimales, lo truncará solo para dejar un valor corto.



Ejercicio 5:

**package main**

**import "fmt"**

**func main(){**

**Cuota, mensaje := 2000.0,"La cuota aumentará un 5% en cada mes:"**

**fmt.Println(mensaje, Cuota)**

**meses :=[] string {"Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio", "Julio", "Agosto", "Septiembre", "Octubre", "Noviembre", "Diciembre"}**

**size := len(meses)**

**i :=0**

**for i != size{**

**i++**

**var resultado float64 = (Cuota \* 0.05) + Cuota**

**Cuota = resultado**

**fmt.Println(meses[i])**

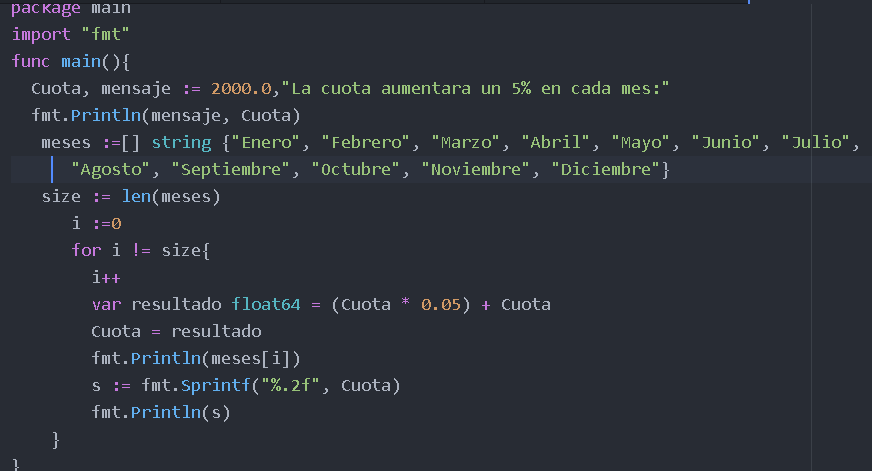
**s := fmt.Sprintf("%.2f", Cuota)**

**fmt.Println(s)**

**}**

**}**

1. **size := len(meses)** El len se utiliza para sacar el número de datos que contiene una matriz , en este caso se utilizó para una declaración dentro del “for”, al momento de que se ejecute tendrá por default el valor de datos en la matriz..

****

Ejercicio 6:

**package main**

**import "fmt"**

**func main(){**

**Nombre, Ap\_Pater, Ap\_Mater, Edad, Cuota, Acreditado,mensaje := "Brenda", "Mejia", "cholula", 21 , 2000.0, true,"La cuota aumentará un 5% en cada mes:"**

**fmt.Println(mensaje, Cuota)**

**var meses [] string**

**meses = append(meses, "Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio", "Julio", "Agosto", "Septiembre", "Octubre", "Noviembre", "Diciembre", "Hola","Nuevos")**

**fmt.Println(meses)**

**meses [1] = "Brenda"**

**meses [11] = "Hola"**

**fmt.Println("Su tamaño es: ",len(meses)," y su capacidad es",cap(meses))**

**fmt.Println(Nombre, Ap\_Pater, Ap\_Mater, Edad, Acreditado)**

**if Acreditado == true{**

**if Cuota > 1999{**

**fmt.Println("Acreditado")**

**i :=0**

**for i != 12{**

**i++**

**var resultado float64 = (Cuota \* 0.05) + Cuota**

**Cuota = resultado**

**fmt.Println(meses[i])**

**s := fmt.Sprintf("%.2f", Cuota)**

**fmt.Println(s)**

**}**

**}else{**

**fmt.Println("No es la cuota")**

**}**

**}else{**

**fmt.Println("No acredita")**

**}**

**}**

1. En este último proyecto se verá en conjunto todo lo realizado en los trabajos anteriores, integrando la matriz, sustitución de matriz, eliminación de decimales, dando como tal una respuesta final de acreditación y cuota final con el paso de los meses.
2. **var meses [] string** ← declaración de inicio para una matriz
3. **meses = append(meses, "valores”)** ← en esta ejecución se manda a traer la variable ya declarada, tomando el append para integrar el nombre de la matriz y los valores que llevará. Con esto lo que logramos es solo mandar a traer la matriz con solo el nombre que en este caso seria “meses”
4. **meses [1] = "Brenda"** ← se utilizará cuando quieras cambiar el valor de una matriz.

