

PROYECTO FINAL

# SEGUIMIENTO DE ACTIVIDADES FÍSICAS

```
COUT << "PROGRAMACIÓN II"<<ENL;
```





# ÍNDICE

1. Resumen
2. Introducción
3. Definiciones
4. Funcionamiento del programa
5. Ejemplos de ejecución
6. Conclusiones
7. Recomendaciones
8. Referencias





# INTRODUCCIÓN

- Abordamos la funcionalidad y desarrollo del seguimiento de actividades físicas, un tema relevante en la salud y el bienestar.
- El seguimiento de actividades físicas implica recopilar y registrar información sobre diferentes actividades como caminar, correr, nadar y levantar pesas. Esto ayuda a evaluar el nivel de actividad física, establecer metas para mejorar la salud y detectar patrones y tendencias en la rutina de ejercicios.

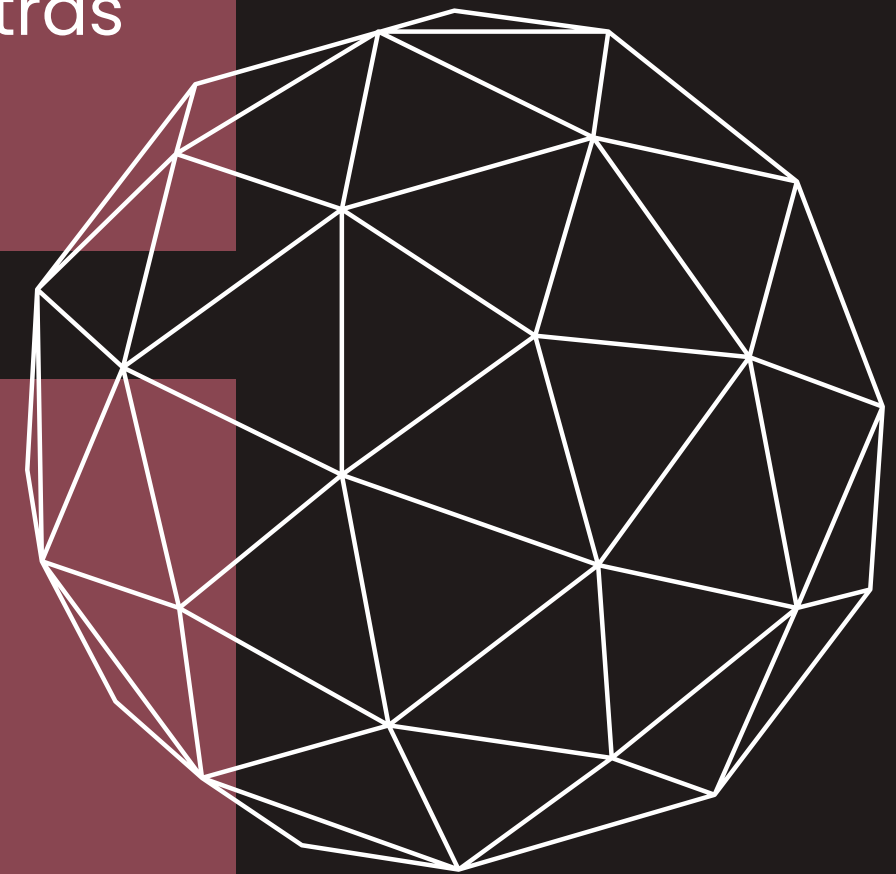
# FUNCIONAMIENTO DEL PROGRAMA

## MÓDULO MAIN.CPP

Controla la ejecución de nuestro programa dirigiendo las llamadas a otras funciones del programa.

### ESTRUCTURA:

1. Creación de Base de datos
2. Menú principal y bucle de ejecución
3. Agregar usuario a base de datos
4. Agregar ejercicios
5. Uso de python para gráficas de progreso de calorías quemadas
6. Generación Reportes individuales
7. Exportación de reportes
8. Liberación de memoria



# MENÚ.H

## • Menú principal

Muestra el menú principal del programa y solicita al usuario que elija una opción del 1 al 7.

### MENU PRINCIPAL

1. Ingresar datos de un nuevo Usuario
2. Ingresar la rutina semanal de un Usuario
3. Mostrar gráficas de progreso de un Usuario
4. Mostrar reporte individual de un Usuario
5. Mostrar reporte general de todos los Usuarios
6. Exportar reporte general
7. Finalizar

## • Menú usuarios

Muestra un menú que lista los usuarios disponibles y solicita al usuario que elija uno ingresando el índice correspondiente.

Elija una opción: 4

### MENU USUARIOS

1. Guillermo Galvez

Ingrese el índice de un Usuario: 1

## • Menú Ejercicios

Muestra un menú para seleccionar el tipo de ejercicio (Cardio, Fuerza, Flexibilidad) o volver al menú principal.

```
MENU EJERCICIOS
1. Cardio
2. Fuerza
3. Flexibilidad
4. Volver al Menu Principal
Elija una opción: 2
```

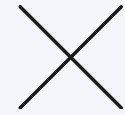
## • Menú cardio

Cuando el usuario elige una opción, se crea una instancia de la clase Cardio correspondiente al ejercicio seleccionado.

```
MENU CARDIO
1. Correr
2. Ciclismo
3. Nadar
4. Saltar Cuerda
5. Volver al Menu Ejercicios
Elija una opción: 5
-----CONSEJO : Cocina en casa para tener control sobre los ingredientes.---
```

## • Menú fuerza

Cuando el usuario elige una opción, se crea una instancia de la clase Fuerza correspondiente al ejercicio seleccionado.



### MENU FUERZA

```
1. Press Banca
2. Peso Muerto
3. Sentadillas
4. Dominadas
5. Flexiones
6. Volver al Menu Ejercicios
Elija una opción: 5
```

### MENU FLEXIBILIDAD

```
1. Yoga
2. Danza
3. Arte Marcial
4. Estiramientos
5. Volver al Menu Ejercicios
Elija una opción: 5
```

-----CONSEJO : Cocina en casa para tener control sobre los ingredientes.---

## • Menú flexibilidad

cuando el usuario elige una opción, se crea una instancia de la clase Flexibilidad correspondiente al ejercicio seleccionado.





# EJERCICIO.H



Esta clase sirve como la clase base para diferentes tipos de ejercicios.

## ATRIBUTOS:

- Nombre
- FCE
- CQE
- Frecuencia
- FA

## MÉTODOS:

- Ingresar datos
- Mostrar y exportar información
- Hallar CQ-FC

```
// Destructor
virtual ~Ejercicio() { cout << "Ejercicio destruido" << endl; }

virtual void hallar_CQ_FC(double TBM, int FCM) = 0;

virtual void ingresar_datos() = 0;

virtual void mostrar_informacion() = 0;

virtual void exportacion_informacion() = 0;

double getFCE() { return FCE; }

double getCQE() { return CQE; }

const string &getNombre() const {
    return nombre;
}
```



# FLEXIBILIDAD.H



Se define la clase Flexibilidad, que hereda de la clase base Ejercicio.

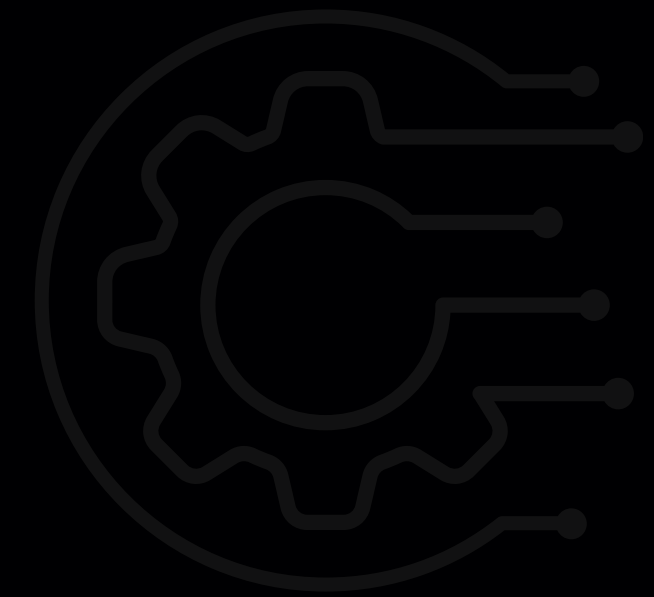
## ATRIBUTOS:

- Duración
- Dificultad

## MÉTODOS:

- Ingresar datos
- Mostrar y exportar información
- Hallar CQ-FC

```
Elija una opción: 2
Ingrese la duración (minutos): 20
Dificultad (fácil/media/difícil): media
Ingrese la frecuencia (diaria/interdiaria/semanal): semanal
-----CONSEJO : Encuentra inspiración en modelos a seguir que hayan alcanzado objetivos similares.-----
```



# FUERZA.H

Se define la clase Fuerza, que hereda de la clase base Ejercicio.

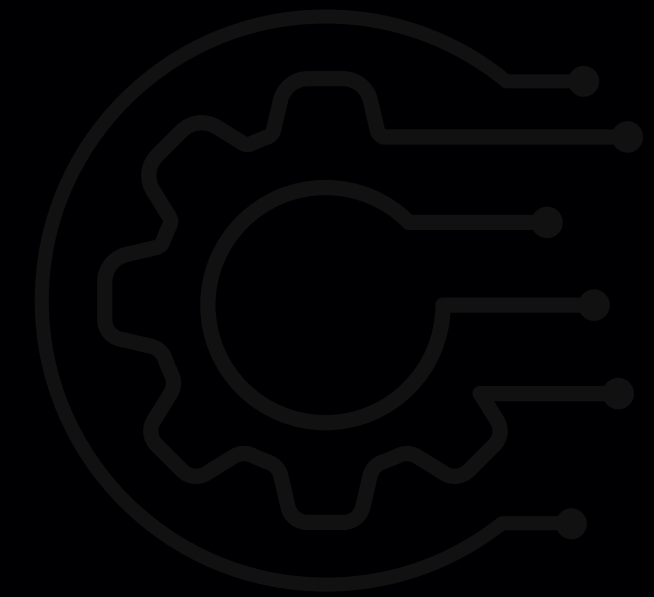
## ATRIBUTOS:

- Repeticiones
- Series
- Peso (Kg)

## MÉTODOS:

- Ingresar datos
- Mostrar y exportar información
- Hallar CQ-FC

```
Ingrese el número de repeticiones: 8
Ingrese el número de series: 3
Ingrese el peso que usa para el ejercicio (kg): 5
Ingrese la frecuencia (diaria/interdiaria/semanal): interdiaria
-----CONSEJO : Programa tus entrenamientos como citas inquebrantables en tu agenda.-----
```



# CARDIO.H

Se define la clase Cardio, que hereda de la clase base Ejercicio.

## ATRIBUTOS:


- Duración (min)
- Velocidad (Km/h)

## MÉTODOS:

- Ingresar datos
- Mostrar y exportar información
- Hallar CQ-FC

### MENU CARDIO

```
1. Correr
2. Ciclismo
3. Nadar
4. Saltar Cuerda
5. Volver al Menu Ejercicios
Elija una opción: 1
Ingrese la velocidad (kilometros por hora): 5
Ingrese la duración (minutos): 20
Ingrese la frecuencia (diaria/interdiaria/semanal): interdiaria
-----CONSEJO : Prioriza alimentos frescos y naturales.-----
```





# USUARIO.H

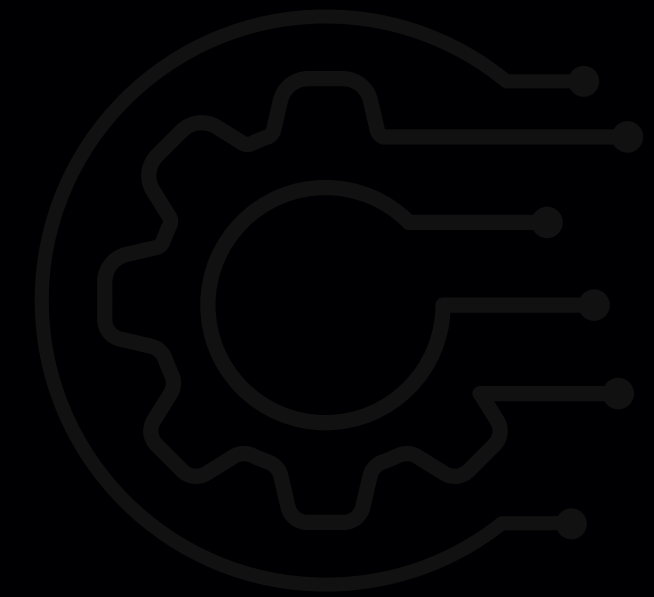
Se define la clase Usuario, que establece un lazo de amistad con la clase Ejercicio.

## ATRIBUTOS:

- Nombre
- Apellido
- Edad
- DNI
- Género
- Altura
- Peso, peso objetivo y peso inicial
- Masa muscular y masa muscular objetivo
- TMB
- FCM

```
// metodos
void agregar_ejercicio(Ejercicio* ejercicio);
float calcular_IMC();
float frecuencia_cardiaca_prom();
void reporte_individual();
void reporte_individual_exportacion();
void meta_peso();
void meta_masa_muscular();
void calorías_quemadas_rango(double c, int n);
void actualizaciones_imcs();
string to_string_historial_calorias();
string to_string_historial_imcs();
```

# BASE DE DATOS.H



Se define la clase Cardio, que hereda de la clase base Ejercicio.

## ATRIBUTOS:

- Usuarios\*

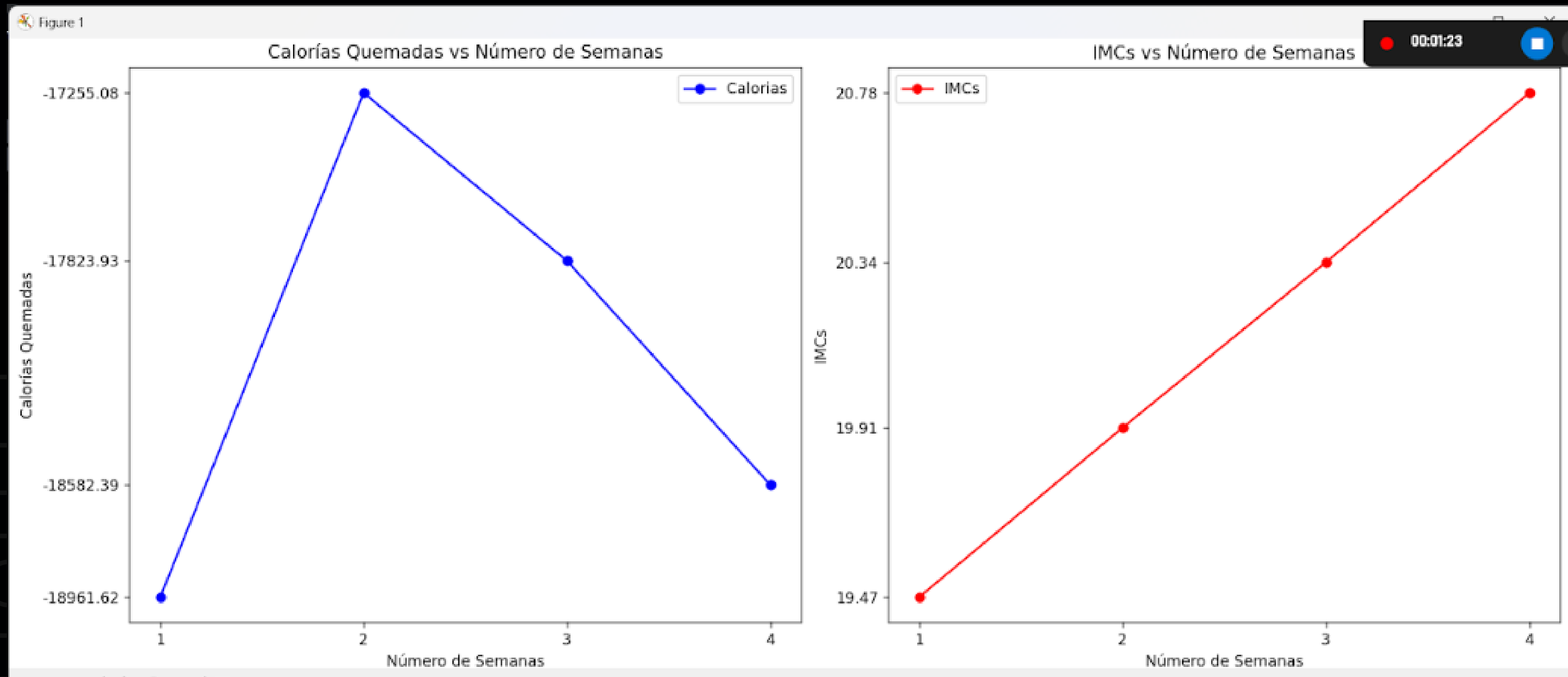
## MÉTODOS:

- Agregar usuario
- Reporte general
- Exportar reportes
- Existen usuarios con DNI

```
BaseDeDatos();  
  
void agregar_usuario();  
  
void reporte_general();  
  
void exportarReportes();  
  
bool existeUsuarioConDNI(const string& dni);  
  
vector<Usuario *> &getUsuarios();  
  
virtual ~BaseDeDatos();
```

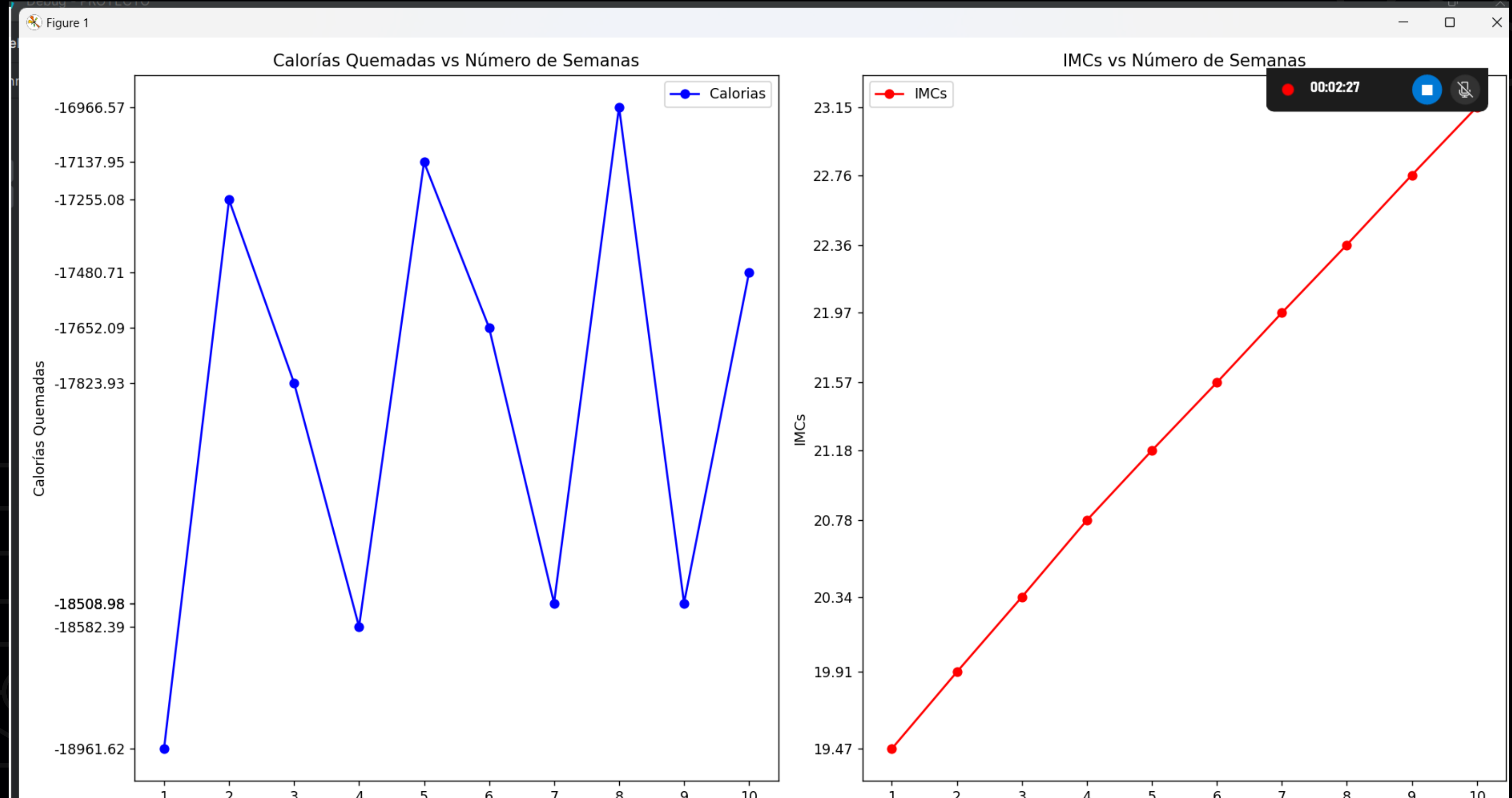
# GRÁFICAS.PY

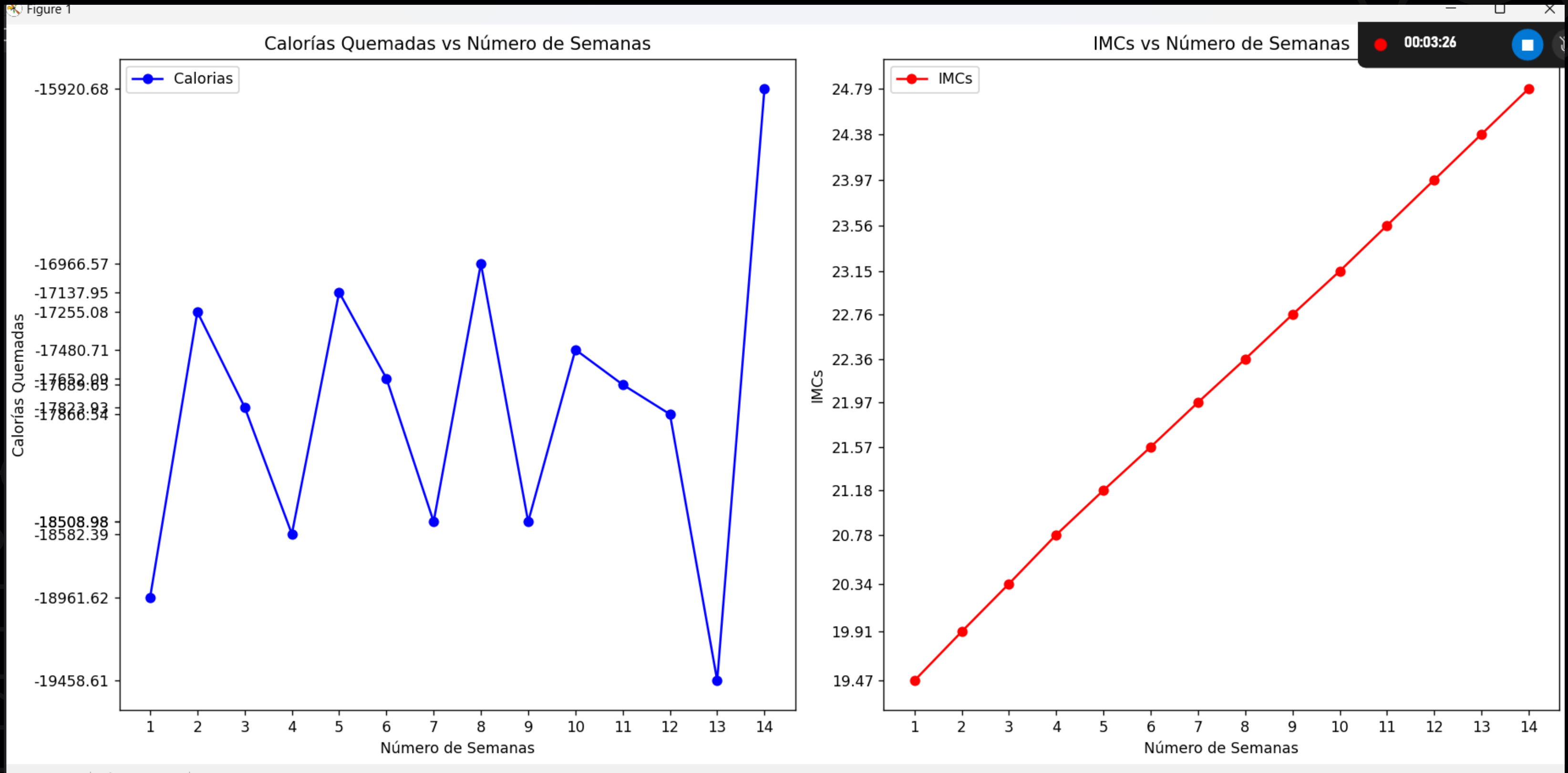
La función del siguiente código es crear un gráfico de la historia de los pesos (weight\_history) o de la historia de los índices de medición (imc\_history) según los argumentos proporcionados en la línea de comandos.





# GRÁFICAS.PY





```
<!--PROGRAMACIÓN II-->
```

Gracias {

```
<Por="GRUPO N° "/>
```

}