Laboratorio No 0. Lógica de programación

Integrantes

Catalina Arcila Grisales

Angela María Bolaños Moncayo

Juan Camilo Poveda Delgado

Ingeniería electrónica

Programación

Profesor

Alexander López - Parrado

26 de agosto del 2024

Laboratorio 0, Lógica de programación

Resumen- En el siguiente documento se busca retomar aquellos conocimientos y habilidades adquiridos anteriormente en Lógica de Programación. Para tal fin, se pondrá en práctica aspectos relacionados con la lógica para la creación de algoritmos y su implementación a través de los lenguajes de programación C y Python.

En ese sentido, la práctica de laboratorio contempla el repaso de estructuras de programación tales como: declaración de variables, sentencias condicionales, sentencias para ciclos, arreglos y funciones.

Palabras clave - Algoritmo, sentencias condicionales, arreglos, funciones, Python.

I. INTRODUCCIÓN

En este documento se demuestra el uso de los conocimientos y habilidades anteriormente adquiridas durante el proceso de aprendizaje en lenguaje C, esto, para aplicarlo y aprender un nuevo lenguaje llamado Python, usando arreglos de datos, sentencias condicionales e iteraciones y funciones, donde el docente da a resolver 3 ejercicios, cada uno basado en un código proporcionado al que se debe modificar según lo solicitado, utilizando lenguaje C y Python, con la finalidad de comprender y aprender a utilizar correctamente el lenguaje Python.

A continuación, se hará un análisis a 3 ejemplos ofrecidos por el docente y se dará una explicación más a fondo sobre los retos propuestos por el docente, además de la solución que se ofrece a cada uno de estos, reflejando el uso y aprendizaje de lo enseñado en clases.

II. OBJETIVOS

Objetivo general:

Analizar el problema estipulado por el docente para hallar la solución más eficaz mediante conocimientos anteriormente adquiridos, además de aprender a manejarlos en un nuevo lenguaje llamado Python.

Objetivos específicos:

- Fortalecer el pensamiento lógico.
- Comprender el funcionamiento de arreglos de datos en Python.
- Comprender el funcionamiento de sentencias condicionales en Python.
- Comprender el lenguaje Python y diferenciarlo del lenguaje C.
- Trabajar en equipo para hallar soluciones.

III. MÉTODOS E INSTRUMENTOS

A continuación, la solución de los ejercicios propuestos por el docente en Lenguaje C y Python con sus respectivos diagramas de flujo:

1. Compile y ejecute el programa, y realice pruebas con diferentes valores de peso y estatura.
2. Cree un programa a partir de bmi.c que le informe al usuario si se encuentra en alguna de las siguientes condiciones: bajo peso, peso normal, sobrepeso u obesidad.
3. Mediante la herramienta de Inteligencia Artificial (IA) Generativa ChatGPT, genere el equivalente en Python (bmi.py) del programa bmi.c. Identifique las similitudes y diferencias entre ambas implementaciones.
4. Re-implemente el programa del punto 2 en Python, **esta vez sin usar ChatGPT**.

Fig. 1. Ejercicio 1, sentencias condicionales.

Para el primer ejercicio se pide modificar el código proporcionado en lenguaje C, de manera que el ims, resultado de la operación entre Kg y altura ingresados por el usuario, indique con sentencias condicionales si está en bajo peso, peso normal o sobrepeso.

Inicialmente, se propone el siguiente código modificado en c:

```c
// Include header file for prototypes of scanf and printf
#include <stdio.h>

// Prototype of calcBmi function
float calcBmi(float h,float w);

// Main function - entry point
int main(){
    // Local variable declaration
    float height;
    float weight;
    float bmi;

    // Ask user for height in cms
    printf("Enter your height in cms: ");
    scanf("%f",&height);

    // Ask user for weight in kgs
    printf("Enter your weight in kgs: ");
    scanf("%f",&weight);

    // Call calcBmi function
    bmi=calcBmi(height,weight);


    // Print result
    printf("Your body mass index is %f kg/m^2\n",bmi);

   if (bmi<=18.49){
       printf("bajo peso\n");
    }
```

```c
    else if ((18.5<=bmi)&(bmi<=24.99)){
        printf("peso normal\n");
    }
    else if (bmi>=25){
        printf("sobrepeso\n");
    }
}

// Implementation of calcBmi function
float calcBmi(float h,float w){
    float bmi;

    // Compute body mass index
    bmi=w/((h/100.0)*(h/100.0));

    return bmi;
}
```

Fig 1.1. Código modificado del ejercicio 1 en lenguaje C.

Este programa en lenguaje C calcula el Índice de Masa Corporal (IMC) de una persona. Primero, solicita al usuario su altura en centímetros y su peso en kilogramos. Luego, llama a la función `calcBmi`, que calcula el IMC dividiendo el peso por la altura al cuadrado (convertida a metros).

El programa imprime el IMC y clasifica el resultado en tres categorías: bajo peso, peso normal y sobrepeso, según los valores de IMC obtenidos. La estructura del código es rigurosa en cuanto a la sintaxis. Requiere punto y coma (`;`) al final de cada línea y llaves (`{}`) para definir bloques de código. Es un lenguaje donde se debe declarar el tipo de cada variable antes de usarla, ejemplo (int, float).

Después, se propone el siguiente código en Python con IA:

```python
# Función para calcular el IMC
def calc_bmi(height, weight):
    # Convertir altura de cm a metros y calcular el IMC
    height_m = height / 100
    bmi = weight / (height_m ** 2)
    return bmi

# Solicitar la altura
height = float(input("Introduce tu altura en cm: "))

# Solicitar el peso
weight = float(input("Introduce tu peso en kg: "))

# Llamar a la función calc_bmi
bmi = calc_bmi(height, weight)

# Imprimir el resultado
print(f"Tu índice de masa corporal es {bmi:.2f} kg/m^2")

# Clasificación según el IMC
if bmi <= 18.49:
    print("Bajo peso")
elif 18.5 <= bmi <= 24.99:
    print("Peso normal")
elif bmi >= 25:
    print("Sobrepeso")
```

Fig. 1.2. Código del ejercicio 1 en Python.

Para empezar, se observan las primeras diferencias entre el lenguaje C y Python, Python identifica las funciones como “def” y no necesita identificar las variables dentro de la función “(height, weight)”. Por otro lado, se observa que no se utiliza “;” ni “scanf”, además para imprimir mensajes en pantalla, se utiliza “input” y “print”. (Observar fig. 1.2).

En la línea 9 se puede notar mejor la nueva forma para pedir valores al usuario y guardar esta información, además de que la variable solicitada (height) es identificada al mismo tiempo como “float”. A continuación, en la línea 18 “print (f” Tu índice de masa corporal es {bmi:.2f} kg/m^2")”, se pueden observar dos diferencias, siendo la primera de estas, como el valor que se quiere imprimir es identificado antes de las comillas como “f”, además de que para introducir el valor calculado “bmi” utiliza corchetes y dentro de estos bmi: .2f. Por último, para las condiciones, no es necesario colocar la condición dentro del paréntesis, aparte de eso, se utiliza “:” para indicar que proceso le corresponde a la condición. En lo que respecta a su estructura, es la misma que en lenguaje C.

Por último, código nuevamente en Python sin IA:

codigo python 1.3.py > ...

```python
height = float(input("Enter your height in cm: "))
weight = float(input("Enter your weight in kg: "))
bmi = weight / ((height / 100.0) * (height / 100.0))
print(f"Your Body Mass Index (BMI) is {bmi:.2f} kg/m^2")

if bmi <= 18.49:
    print("Underweight")
elif 18.5 <= bmi <= 24.99:
    print("Normal weight")
elif bmi >= 25:
    print("Overweight")
```

Fig. 1.3. Código del ejercicio 1 en Python.

A continuación, el diagrama de flujo:

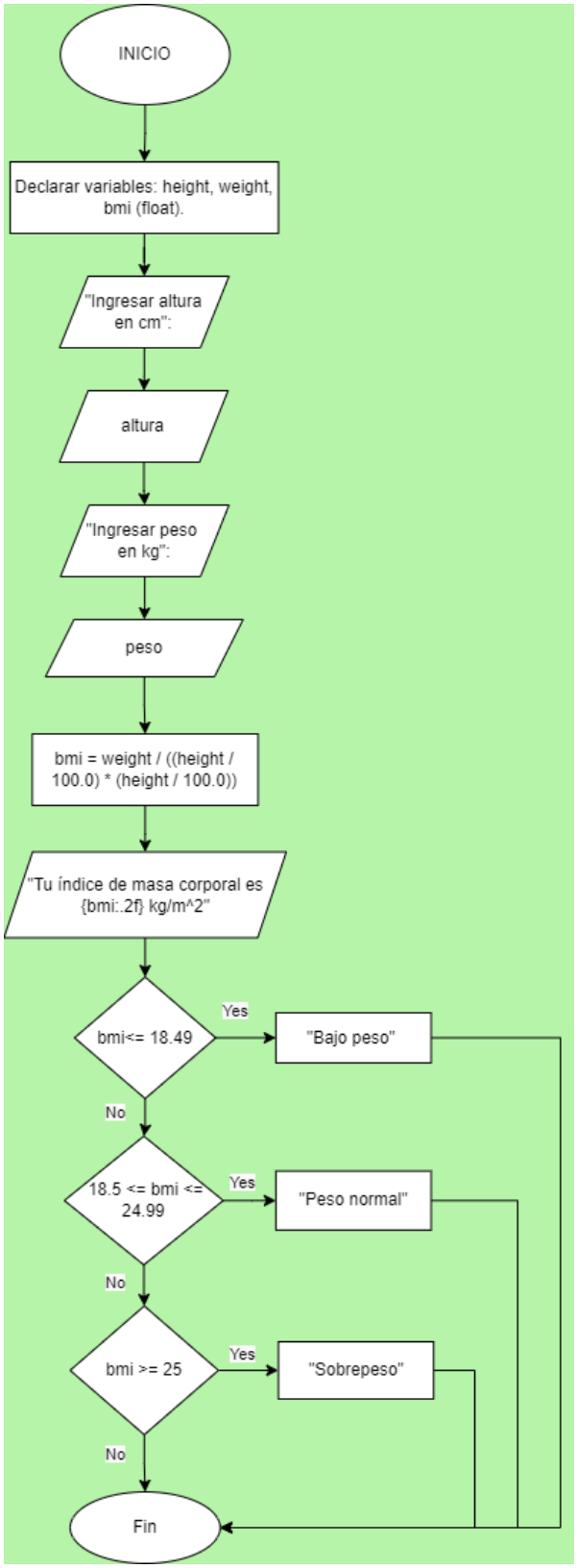

Fig 1.4. Diagrama de flujo del ejercicio 1.

Fig. 2. Ejercicio 2, ciclos y arreglos.

Para el segundo ejercicio se pide modificar el código anteriormente proporcionado, de manera que el imc, resultado de la operación entre Kg y altura ingresados por el usuario, indique con sentencias condicionales si está en bajo peso, peso normal o sobrepeso, además de añadir la cantidad de personas a calcular el imc para guardar los datos en un arreglo.

Por otro lado, después de esto, pide imprimir en porcentaje la cantidad de personas en bajo peso, peso normal y sobrepeso

Inicialmente, se proponen los siguientes códigos:

```c
#include <stdio.h>

// Función para calcular el IMC
float calc_bmi(float height, float weight) {
    float height_m = height / 100;
    float bmi = weight / (height_m * height_m);
    return bmi;
}

int main() {
int personas, i;
float bmi, weight, height;
float bajo_peso = 0, peso_normal = 0, sobrepeso = 0;

printf("Ingrese el numero de personas: ");
scanf("%d", &personas);

// Arreglo para almacenar los valores de BMI
float bmi_list[personas];

// Bucle para calcular el BMI de cada persona
for (i=0; i<personas; i++) {
    //Solicitar al usuario la altura en cm
    printf("Introduce tu altura en cm: ");
    scanf("%f", &height);

    //Solicitar al usuario el peso en kg
    printf("Introduce tu peso en kg: ");
    scanf("%f", &weight);

    //Llamar a la función calc_bmi
    bmi = calc_bmi(height, weight);

    //Categorizar según el BMI
    if (bmi < 18.5) {
        bajo_peso += 1;
        }
    else if (18.5 <= bmi && bmi<= 24.99) {
        peso_normal += 1;
        }
    else if (25 <= bmi) {
        sobrepeso += 1;
    }
    //Agregar el resultado a la lista de BMI
    bmi_list[i] = bmi;

    printf("Tu indice de masa corporal es %.2f\n", bmi);
    }

for (i = 0; i < personas; i++) {
        printf("Persona %d: %.2f\n", i + 1, bmi_list[i]);
    }
return 0;
}
```

Fig 2.1. Código de arreglos en lenguaje C.

Este código en C calcula el Índice de Masa Corporal (IMC) para varias personas. Primero, pide al usuario la cantidad de personas a procesar y luego, en un bucle, solicita la altura y el peso de cada persona. Con estos datos, calcula el IMC usando una función llamada calc_bmi. Dependiendo del valor del IMC, clasifica a la persona en una de tres categorías: bajo peso, peso normal o sobrepeso. Cada resultado de IMC se almacena en un arreglo (bmi_list). Al finalizar, el programa imprime el IMC de cada persona junto con su categoría correspondiente.

```python
# Función para calcular el IMC
def calc_bmi(height, weight):
    # Convertir altura de cm a metros y calcular el IMC
    height_m = height / 100
    bmi = weight / (height_m ** 2)
    return bmi

# Número de personas
personas = int(input("Ingrese el número de personas: "))

# Lista para almacenar los valores de BMI
bmi_list = []

for i in range(personas):
    # Solicitar al usuario la altura en cm
    height = float(input("Introduce tu altura en cm: "))

    # Solicitar al usuario el peso en kg
    weight = float(input("Introduce tu peso en kg: "))

    # Llamar a la función calc_bmi
    bmi = calc_bmi(height, weight)

    # Agregar el resultado a la lista de BMI
    bmi_list.append(bmi)

    # Imprimir el resultado
    print(f"Tu índice de masa corporal es {bmi:.2f} kg/m^2")

# Imprimir todos los valores de BMI
for idx, bmi in enumerate(bmi_list, start=1):
    print(f"Persona {idx}: {bmi:.2f} kg/m²")
```

Fig 2.2. Código de arreglos en Python.

En este código se pueden observar diferencias respecto al uso de arreglos, en la línea 12, a diferencia del lenguaje C, no necesita definir el arreglo como “float”, además se puede definir directamente como un arreglo vacío, lo cual permite mayor versatilidad al ser usado más adelante. Por otra parte, en la línea 14 se introduce el “for”, donde, a diferencia del lenguaje C, utiliza a “i” en un rango de “personas”. (Observar fig. 2.2).

En la línea 25 y 31, cabe resaltar nuevas diferencias, en la primera, “bmi_list.append(bmi)”, se observa como la función append añade un nuevo elemento al final de la lista. En este caso, la variable bmi, que contiene un valor del Índice de Masa Corporal (IMC) calculado, se agrega a la lista bmi_list y en la segunda, se utiliza el segundo “for”, el cual analizará los datos en “bmi_list” y los imprimirá en pantalla, esta línea recorre la lista bmi_list y, en cada iteración, te da tanto el índice (idx) comenzando desde 1, como el valor del IMC (bmi).

```c
#include <stdio.h>

// Función para calcular el IMC
float calc_bmi(float height, float weight) {
    float height_m = height / 100;
    float bmi = weight / (height_m * height_m);
    return bmi;
}

int main() {
int personas, i;
float bmi, weight, height;
float bajo_peso = 0, peso_normal = 0, sobrepeso = 0;

printf("Ingrese el numero de personas: ");
scanf("%d", &personas);

// Bucle para calcular el BMI de cada persona
for (i=0; i<personas; i++) {
    //Solicitar al usuario la altura en cm
    printf("Introduce tu altura en cm: ");
    scanf("%f", &height);

    //Solicitar al usuario el peso en kg
    printf("Introduce tu peso en kg: ");
    scanf("%f", &weight);

    //Llamar a la función calc_bmi
    bmi = calc_bmi(height, weight);

    //Categorizar según el BMI
    if (bmi < 18.5) {
        bajo_peso += 1;
        }
    else if (18.5 <= bmi && bmi<= 24.99) {
        peso_normal += 1;
        }
    else if (25 <= bmi) {
        sobrepeso += 1;
    }

    printf("Tu índice de masa corporal es %.2f kg/m^2\n", bmi);
    }

float porcentaje_bajo_peso = (bajo_peso / personas) * 100;
float porcentaje_peso_normal = (peso_normal / personas) * 100;
float porcentaje_sobrepeso = (sobrepeso / personas) * 100;

printf("El porcentaje de las personas con bajo peso: %.2f%%\n", porcentaje_bajo_peso);
printf("El porcentaje de las personas con peso normal: %.2f%%\n", porcentaje_peso_normal);
printf("El porcentaje de las personas con sobrepeso: %.2f%%\n", porcentaje_sobrepeso);

return 0;
}
```

Fig. 2.3. Código de porcentajes en lenguaje C.

En este código, al igual que el anterior, se añaden 6 nuevas variables para hallar el porcentaje de personas en bajo peso, peso normal y sobrepeso. A diferencia del código de arreglos, estos no se utilizan y se añaden sentencias condicionales dentro del “for”, esto, para almacenar a la persona en uno de los tres rangos, al finalizar el ciclo, los valores finales que quedaron en cada categoría serán utilizados en una nueva ecuación para hallar el porcentaje, para esto, se usa el valor final de cada categoría, siendo el primero de estos “bajo_peso” sobre “personas”, el resultado se multiplica por 100 y así se obtiene el porcentaje: porcentaje_bajo_peso = $(\frac{\square\square\square\square.\square\square\square\square}{\square\square\square\square\square\square\square\square\square}) \times 100$

Esta misma ecuación se aplica para las otras categorías, al finalizar se añaden los códigos para imprimir los resultados de cada categoría y se termina con “return 0;”. (Observar fig. 2.3).

```python
# Función para calcular el IMC
def calc_bmi(height, weight):
    # Convertir altura de cm a metros y calcular el IMC
    height_m = height / 100
    bmi = weight / (height_m ** 2)
    return bmi

personas = int(input("Ingrese el número de personas: "))

bajo_peso = 0
peso_normal = 0
sobrepeso = 0

# Bucle para calcular el BMI de cada persona
for i in range(personas):
    # Solicitar al usuario la altura en cm
    height = float(input("Introduce tu altura en cm: "))

    # Solicitar al usuario el peso en kg
    weight = float(input("Introduce tu peso en kg: "))

    # Llamar a la función calc_bmi
    bmi = calc_bmi(height, weight)

    # Categorizar según el BMI
    if bmi < 18.5:
        bajo_peso += 1
    elif 18.5 <= bmi <= 24.99:
        peso_normal += 1
    elif 25 <= bmi:
        sobrepeso += 1

    print(f"Tu índice de masa corporal es {bmi:.2f} kg/m^2")

# Calcular porcentajes
porcentaje_bajo_peso = (bajo_peso / personas) * 100
porcentaje_peso_normal = (peso_normal / personas) * 100
porcentaje_sobrepeso = (sobrepeso / personas) * 100

print(f"El porcentaje de las personas con bajo peso: {porcentaje_bajo_peso:.2f}%")
print(f"El porcentaje de las personas con peso normal: {porcentaje_peso_normal:.2f}%")
print(f"El porcentaje de las personas con sobrepeso: {porcentaje_sobrepeso:.2f}%")
```

Fig. 2.4. Código de porcentajes en Python.

En este código, al igual que en lenguaje C, se añaden 6 nuevas variables para hallar el porcentaje de personas en bajo peso, peso normal y sobrepeso. En lo que respecta a su estructura, no hay cambios significativos que no hayan sido explicados anteriormente. A continuación, los diagramas de flujo de este punto:

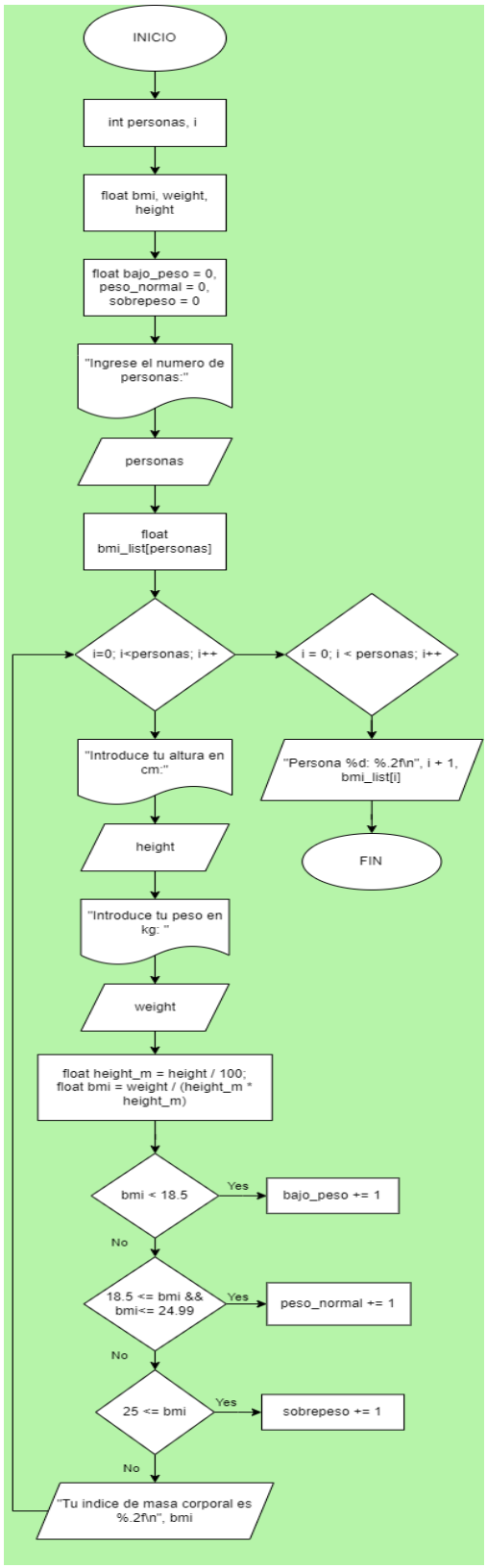

Fig. 2.5. Diagrama de flujo del código de arreglos.

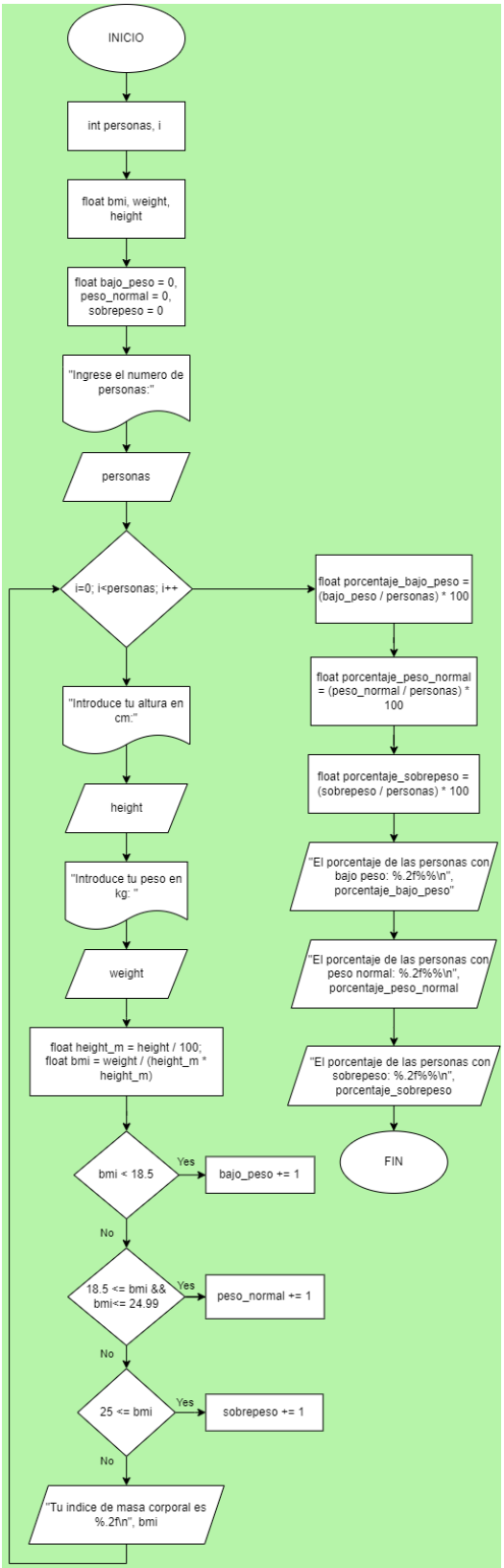

Fig. 2.6. Diagrama de flujo del código de arreglos y porcentajes.

Para lo anterior, considere los programas que construyó en los puntos 2 y 4 de la sección Sentencias Condicionales.

1. Realice una modificación a ambos programas para que se le informe al usuario el peso ideal para lograr la condición de peso normal, esto solo en el caso de que el usuario se encuentre en algunas de las siguientes condiciones: bajo peso, sobrepeso u obesidad. Para lo anterior cree una función que realice el cálculo del peso ideal.

Fig 3. ejercicio 3, funciones.

Para el tercer y último ejercicio se pide modificar el código proporcionado en lenguaje C, de manera que se usen funciones para calcular el rango del peso ideal e imprimir el resultado al usuario.

Inicialmente, se propone el siguiente código modificado en c:

```c
#include <stdio.h>

// Función para calcular el IMC
float calc_bmi(float height, float weight) {
    return weight / ((height / 100.0) * (height / 100.0));
}

// Función para calcular el rango de peso ideal
void peso_ideal(float height, float *min, float *max) {
    float h = height / 100.0;
    *min = 18.5 * (h * h);
    *max = 24.9 * (h * h);
}

int main() {
    float height, weight, bmi, min_weight, max_weight;

    printf("Introduce tu altura en cm: ");
    scanf("%f", &height);

    printf("Introduce tu peso en kg: ");
    scanf("%f", &weight);

    bmi = calc_bmi(height, weight);

    printf("Tu índice de masa corporal es %.2f kg/m^2\n", bmi);

    if (bmi < 18.5) {
        printf("Bajo peso\n");
        peso_ideal(height, &min_weight, &max_weight);
        printf("Para un peso normal, debe pesar entre %.2f kg y %.2f kg.\n", min_weight, max_weight);

    } else if (bmi >= 18.5 && bmi <= 24.9) {
        printf("Peso normal\n");

    } else if (bmi >= 25) {
        printf("Sobrepeso\n");
        peso_ideal(height, &min_weight, &max_weight);
        printf("Para un peso normal, debe pesar entre %.2f kg y %.2f kg.\n", min_weight, max_weight);
    }

    return 0;
}
```

Fig 3.1. Código del ejercicio 3 en lenguaje C.

Para este código (Observar figura 3.1), la primera modificación que se hace del código de sentencias condicionales, es añadir una nueva función llamada “peso_ideal” (línea 9), en la que se utilizan dos principales ecuaciones, la primera sirve para hallar el peso ideal mínimo, teniendo en cuenta que el imc mínimo para un peso normal es de 18.5, con esto, se usa la altura elevada al cuadrado multiplicado por el valor mínimo anterior: $* \square\square\square = 18.5 \times (h^2)$

Así mismo se utiliza el mismo método para hallar el peso ideal, teniendo en cuenta que el ims máximo es de 24.9: $* \square\square\square = 24.9 \times (h^2)$

Como se puede notar en la línea 30, al llamar a la función, esta aparece con dos variables distintas, esto se debe a que en lenguaje C, los valores de la función “peso_ideal” (*min, *max), deben llevar un asterisco porque son punteros, al utilizar punteros en una función, se puede modificar directamente el valor de las variables que fueron pasadas a la función desde la función que los llama. Así que, cuando dentro de peso_ideal asignas valores a *min y *max, se está modificando las variables min_weight y max_weight en main con los valores calculados en la función peso_ideal.

A continuación, en la función principal, se añaden en las condiciones de bajo peso y sobrepeso un printf nuevo, este sirve para imprimir los resultados de las anteriores ecuaciones, sin antes llamar a la función “peso_ideal” para realizar estas mismas e imprimir en pantalla el valor.

Después, se propone el siguiente código en Python:

```python
def calc_bmi(height, weight):
    # Convertir altura de cm a metros y calcular el IMC
    h = height / 100
    bmi = weight / (h ** 2)
    return bmi

def peso_ideal(height):
    h = height / 100
    # Calcular el rango de peso ideal
    min = 18.5 * (h ** 2)
    max = 24.9 * (h ** 2)
    return min, max

# Solicitar la altura
height = float(input("Introduzca su altura en cm: "))

# Solicitar el peso
weight = float(input("Introduzca su peso en kg: "))

# Llamar a la función calc_bmi
bmi = calc_bmi(height, weight)

# Imprimir el resultado
print(f"El índice de masa corporal es {bmi:.2f}")
# Clasificación según el IMC
if bmi < 18.5:
    print("Bajo peso")
    min, max = peso_ideal(height)
    print(f"Para alcanzar un peso normal, deberías pesar entre {min:.2f} kg y {max:.2f} kg.")
elif 18.5 <= bmi <= 24.9:
    print("Peso normal")
elif 25 <= bmi:
    print("Sobrepeso")
    min, max = peso_ideal(height)
    print(f"Para un peso normal, debe pesar entre {min:.2f} kg y {max:.2f} kg.")
```

Fig 3.2. Código del ejercicio 3 en Python.

Para el código en Python, se usa la misma estructura explicada en códigos anteriores, acá se puede observar que en la función “peso_ideal” ya no es necesario cambiar el nombre de la variable para su uso al ser llamada. Finalmente añade la misma nueva función en las condiciones para poder imprimir el rango del peso ideal, y al igual que en lenguaje C, se llama a la función antes de este proceso.

Relacionado a lo anterior, se genera el siguiente diagrama de flujo:

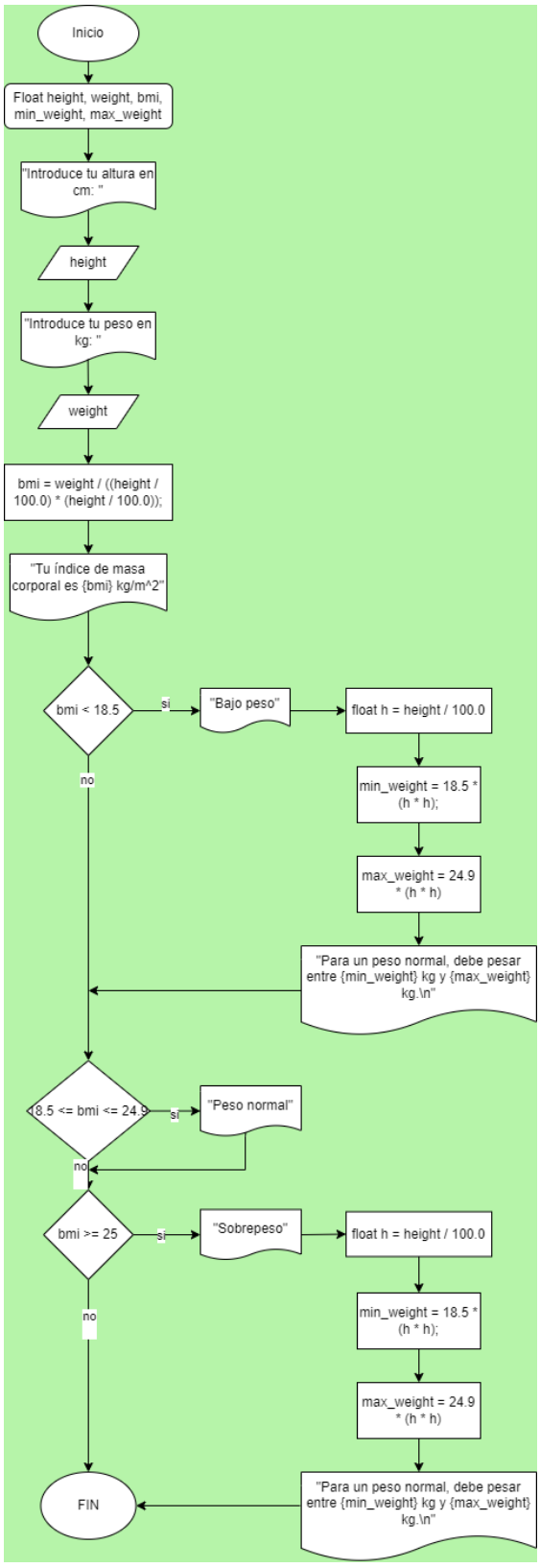

Fig 3.3. Diagrama de flujo del ejercicio 3.

CONCLUSIONES

1-La comparación entre C y Python permite destacar las diferencias en la sintaxis y estructura de ambos lenguajes. Python se destaca por su simplicidad y legibilidad, no requiere declaraciones de tipos explícitas, y se pueden realizar muchas operaciones con menos líneas de código en comparación con lenguaje C, el cual tiene una sintaxis más estricta. Los programadores deben declarar explícitamente los tipos de las variables y seguir una estructura más rígida.

2- El nuevo lenguaje de programación, Python, permite utilizar menos líneas de código e implementar las funciones anteriormente aprendidas en lenguaje C con mayor facilidad, gracias a que es un lenguaje muy versátil, teniendo en cuenta que a diferencia del lenguaje C, no necesita “;” ni especificar de qué tipo son la mayoría de variables, además de que en las sentencias condicionales y de iteración, no se necesitan paréntesis ni llaves, se puede manejar fácilmente mediante espacios, los cuales indican el proceso que se debe seguir, por otro lado, no se debe definir la función principal y tampoco utilizar return 0. Todo lo anterior ayuda bastante a evitar saltar detalles que puedan impedir la ejecución del programa, por lo que Python pasa a ser uno de los lenguajes más fáciles de manejar.

3-Esta práctica no solo solidifica la teoría aprendida, sino que también desarrolla habilidades críticas para el diseño y la implementación eficiente de algoritmos en contextos de programación reales.