MEJORA DEL HITO INDIVIDUAL PROGRAMACIÓN

NOMBRE: Daniel Manzano Núñez
CURSO: 1ºDAM

ÍNDICE

Fase 1:

- *Definición Algoritmia*
- *Diagrama de flujo*
- *Caso de uso*

Fase 2:

- *Implementación código Python*

Fase 3:

- *Análisis Crítico*
- *Paradigmas de programación*

Fase 1:

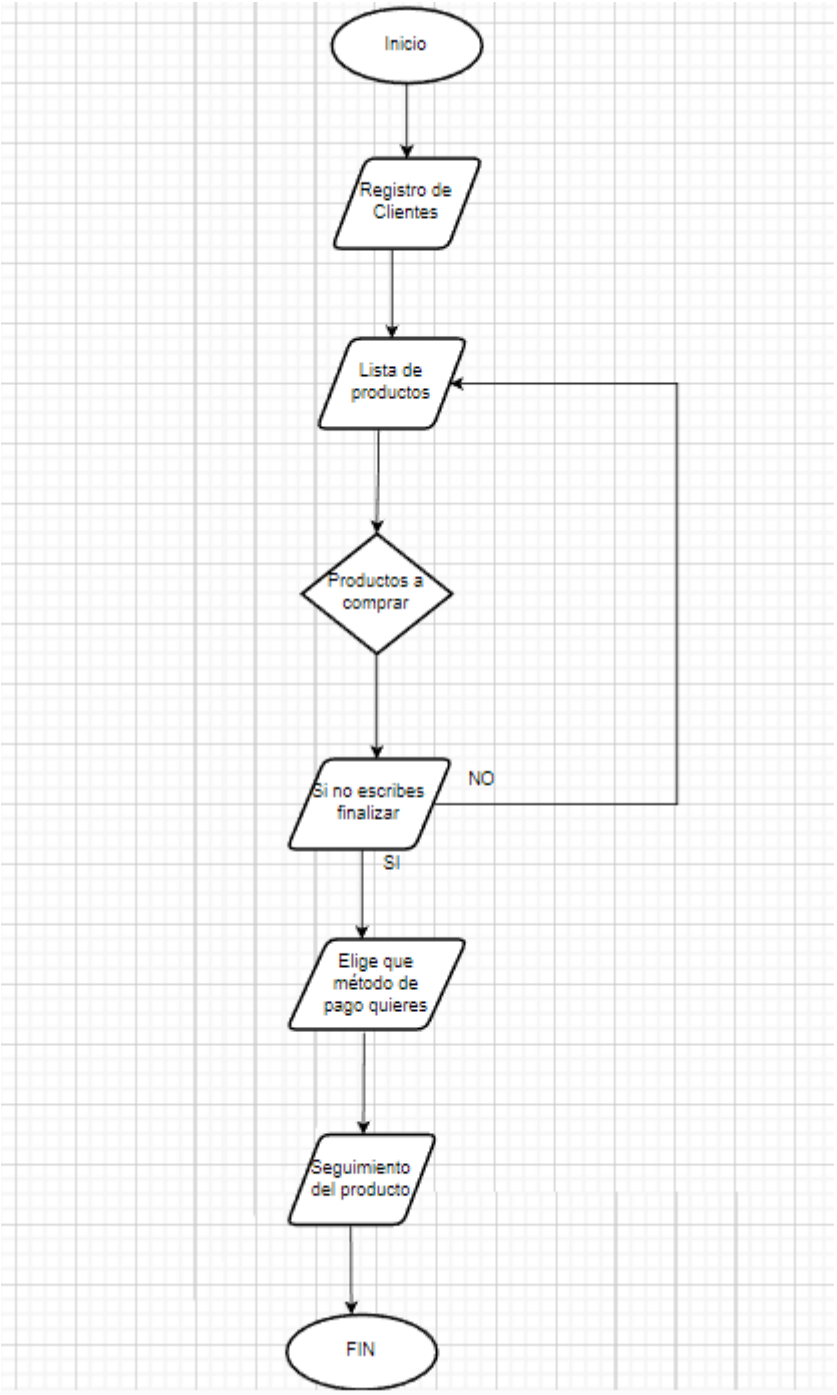
- Algoritmia

Un algoritmo es una secuencia de pasos que permiten resolver un problema paso a paso en un tiempo finito.

Cada algoritmo tiene un procedimiento paso a paso. En primer lugar se busca cual es el problema y se analiza. Después, se prepara un planteamiento para resolverlo y se lleva a cabo. Una vez finalizado dicho proceso, se ejecuta de manera que no haya fallos a la hora de llevar a cabo el proyecto. Si no encuentras fallos, has llegado a la solución.

En el caso de Registrar un cliente, primero observas información sobre el tema a tratar. Después, en el planteamiento creas una clase llamada cliente y ahí introduces los datos personales del cliente tales como: nombre, apellidos, correo electrónico, contraseña o número de teléfono. Una vez introducidos los datos del cliente creas una función llamada RegistroCliente( ): y ahí le dices que imprima los datos del cliente. Finalmente añades los clientes y los llamas junto con el método de registrar al cliente. Como podemos observar estamos ante un algoritmo el cual hemos desarrollado paso a paso. Para acabar con el algoritmo solo nos quedaría ejecutar y ver si diera algún fallo.

- Diagrama de Flujo



En este diagrama podemos observar como una etiqueta Inicio marca el comienzo del algoritmo o proyecto.
A continuación le indicamos que registre al cliente. Luego decimos que muestre la lista de pedidos disponibles a adquirir y le decimos que mientras no indique finalizar puedas adquirir los productos que desees, pero cuando indiques finalizar te muestre el precio y pases al método de pago. Finalmente concluimos con el seguimiento de

producto y el fin del proyecto. Este diagrama es el esquema que usaremos para la implementación en Python.

- Caso de uso

- Nuestra aplicación registra al cliente dado de alta con sus datos personales tales como nombre, apellidos, teléfono, correo o contraseña entre otros usos.
- Además los administradores de la aplicación son los encargados de dar de alta un producto usando el id del pedido, nombre, fecha en que se realizó entre los datos relacionados al producto.
- Cuando un cliente se dé de alta en nuestra aplicación, se comprobará que el cliente ha introducido datos verídicos utilizando las bases de datos (si fuera necesario) para validar la cuenta con su teléfono y correo electrónico (el cual debe llevar un "@")
- Lo mismo se debe realizar con el producto de manera que no haya unidades de más a las que se hayan puesto en venta. Es decir tenemos que dar de alta los productos y mostrarle esos productos al cliente ya dado de alta.
- Una vez mostrados los productos dados de alta es el cliente el cual elige o selecciona los productos que desea comprar y elige el método de pago para pagar dichos productos.
- Finalmente el administrador comprobará que el pago se haya realizado correctamente y le entregará la compra final al cliente.

Fase 2:

- Documento de Python explicado paso a paso

A continuación, en esta fase 2 veremos el código interpretado en Python.

```python
#creamos una clase llamada cliente
class Cliente:
    #definimos el método init con todos los datos del cliente
    def __init__(self, nombre, apellido, pais, telefono, correo, password):
        self.nombre = nombre
        self.apellido = apellido
        self.pais = pais
        self.telefono = telefono
        self.correo = correo
        self.password = password
    #definimos la función para que nos muestre por consola que el usuario se ha registrado
    def RegistroCliente(self):
        print(f'El cliente {self.nombre}, ha sido registrado con sus datos')
#Aquí llamamos a los dos clientes a registrar
cliente1 = Cliente("Juan", "Pérez", "España", "654879210", "juanperez@gmail.com", "juaitopz")
cliente2 = Cliente("Ana", "González", "España", "694852015", "anagolez@gmail.com", "anitagon")
#y por último llamamos al método registroCliete con los dos clientes anteriores
cliente1.RegistroCliente()
cliente2.RegistroCliente()
```

Aquí podemos ver el inicio del programa o aplicación. En este caso tenemos que registrar al cliente mediante la creación de una clase llamada Cliente en la que registramos los datos del cliente mediante un método llamado "__init__".

A continuación creamos otro método llamado RegistroCliente y le decimos que el cliente con nombre "Juán" se ha registrado correctamente. Posteriormente debajo creamos dos clientes con los datos almacenados en el método init. Solo nos quedaría llamar al método RegistroCliente de manera que nos muestre el mensaje anterior.

```python
def es_direccion_correo_valida(direccion):
  # Utiliza una expresión regular para verificar si la dirección de correo electrónico cumple con el formato válido
  coincidencia = re.match('^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(\.[a-z0-9-]+)*(\.[a-z]{2,4})$', direccion)
  if coincidencia == None:
    return False
  return True

# Prueba la función con algunas direcciones de correo electrónico
direcciones = ['juanperez@gmail.com', 'mi@correo', 'anagolez@gmail.com']
for direccion in direcciones:
  resultado = es_direccion_correo_valida(direccion)
  print(f'{direccion}: {resultado}')
```

Aquí podemos observar que en coincidencia tenemos la forma clara de un correo electrónico. Justo debajo podemos ver que si la coincidencia no es de ese estilo diga que el correo es falso, sino que diga que el correo es verdadero. Justo debajo tenemos la prueba con los correos de Juan y Ana los cuales son verdaderos ya que tienen la estructura de un correo electrónico con un "@" y ".com o .es" mientras que "mi@correo" es falso debido a la ausencia del ".com o .es".

```python
#Creamos un diccionario en el cual introducimos los productos con su nombre, precio y cantidad
productos = [
  { "nombre": "Zapatillas de deporte", "precio": 100, "cantidad": 5 },
  { "nombre": "Camisa de algodón", "precio": 50, "cantidad": 3 },
  { "nombre": "Pantalón de mezclilla", "precio": 75, "cantidad": 4 }
]

# Creamos el diccionario de la compra
compra = {
  "cliente": input('dime que cliente ha efectuado la compra: '),
  "pais": "España",
  "productos": [],
  "total": 0
}
```

En este paso, creamos un diccionario llamado productos, de manera que guarde la información de cada producto con sus unidades disponibles y el precio por unidad comprada.

Luego creamos otro diccionario llamado compra donde guardamos los datos sobre la compra como: ¿Quién ha realizado la compra?, ¿Desde que país se ha realizado?, entre otros.

```python
# Solicitamos al usuario que seleccione los productos a comprar
print("Selecciona los productos a comprar:")
for i, producto in enumerate(productos):
  print(f"{i+1}. {producto['nombre']} - Precio: {producto['precio']} - Cantidad disponible: {producto['cantidad']}")
print("Escribe el número de los productos separados por comas (ej: 1,3,4) o escribe 'finalizar' para terminar la compra")
seleccion = input("> ")

# Mientras el usuario no escriba "finalizar", seguimos pidiendo productos
while seleccion != "finalizar":
  try:
    # Convertimos la selección a una lista de números
    seleccion = [int(x) for x in seleccion.split(",")]
    # Agregamos cada producto seleccionado al diccionario de la compra
    for i in seleccion:
      # Decrementamos la cantidad disponible del producto
      productos[i-1]["cantidad"] -= 1
      # Agregamos el producto a la compra
      compra["productos"].append(productos[i-1])
      # Incrementamos el total de la compra
      compra["total"] += productos[i-1]["precio"]
    # Volvemos a solicitar productos al usuario
    print("Selecciona otros productos o escribe 'finalizar' para terminar la compra")
    seleccion = input("> ")
  except ValueError:
    print("Por favor, ingresa una lista de números separados por comas o escribe 'finalizar' para terminar la compra")
    seleccion = input("> ")
```

Le preguntamos al cliente qué productos y unidades desea adquirir mediante un input. Utilizamos un "while" para que mientras el cliente no introduzca "finalizar" se le volverá a hacer la pregunta de qué productos y unidades desea, pero cuando el cliente introduzca "finalizar" le muestre la cantidad del precio a abonar.

```python
#queremos hacerle un descuento del 10% por su compra navideña
precio_original = int(input('dime el precio total de su pedido: '))
porcentaje_descuento = 10

#introducimos las formulas del descuento y el precio final
descuento = precio_original * porcentaje_descuento / 100
precio_con_descuento = precio_original - descuento

#imprimimos por consola el precio con descuento
print(f"Precio con descuento: ${precio_con_descuento:.2f}")

# Define las opciones de pago disponibles
payment_options = ['tarjeta de crédito', 'Pay Pal', 'efectivo']

# Pregunte al usuario por su opción de pago
print("Seleccione una opción de pago:")

# Muestra una lista de opciones de pago al usuario
for i, option in enumerate(payment_options):
    print(f"{i+1}. {option}")

# Pida al usuario que ingrese su opción de pago
selected_option = input("Ingrese el número de la opción deseada: ")

# Convierte la opción seleccionada por el usuario a un número entero
selected_option = int(selected_option)

# Obtiene la opción de pago seleccionada por el usuario
selected_payment_option = payment_options[selected_option - 1]

# Muestra la opción de pago seleccionada por el usuario
print(f"Ha seleccionado la opción de pago: {selected_payment_option}")

#enviamos la factura al cliente
print(f"la factura total de ${precio_con_descuento:.2f} euros ha sido enviada al correo mediante un documento PDF")
```

Después de que muestre el total de la compra y antes de realizar el pago, la aplicación le quiere hacer un descuento del 10% a todos sus clientes por navidad. Por lo tanto, introducimos las fórmulas correctas para la realización del 10% de descuento sobre el precio total de la compra. Una vez hecho y habiendo mostrado el precio con el descuento añadido, introducimos las opciones de pago que tiene el cliente dándole opción de abonar en efectivo, tarjeta o PayPal. Una vez que hayas mostrado las opciones de pago el cliente obtiene una y la factura con el precio se enviará al correo electrónico.

```python
#introduce los datos de su compra
idcompra=int(input('introduce el id de compra enviado en tu factura: '))
fecha=input('introduce la fecha de compra en dd/mm/yyyy: ')
telefono=int(input('introduce tu número de telefono: '))

#imprime el mensaje  de que puede seguir la compra a través del SMS
print('Puedes seguir la compra mediante el SMS de tu telefono móvil')
```

Aquí pedimos al cliente que introduzca los datos de su compra y que al final le muestre un mensaje por consola que puede seguir su compra por SMS.

Fase 3:

- Análisis crítico

La aplicación ha sido creada con la función de que cualquier cliente que desee pueda darse de alta con sus datos personales y efectuar la compra de determinados productos adquiriendo la factura en un documento PDF al correo electrónico y teniendo un seguimiento de su compra.

La primera cosa importante que te pide la aplicación es el registro del cliente, y para ello se ha utilizado una clase para definir las variables y los datos de los clientes a realizar.

Segundo, se pide una selección de productos de manera que el cliente pueda ver que productos posees y al precio que lo puede obtener. Para ello, se ha usado un diccionario de datos que guarde y almacene los productos adquiridos.

La tercera cosa que se pide es efectuar la compra de los productos, hacer el pago y recibir la factura en pdf en el correo electrónico. Finalmente se pide que se haga un seguimiento del producto.

En conclusión, podemos ver en este análisis crítico todos los puntos a tratar de manera que podamos observar que la aplicación cumple con lo acordado.

- Paradigma usado

El paradigma usado para la aplicación es el paradigma imperativo. En este Paradigma, se describen procedimientos hasta llegar al resultado final o solución.
Otra de las características del paradigma imperativo que podemos observar es la utilización de estructuras “while”, “for” para llevar a cabo el programa o aplicación.
Este paradigma imperativo utiliza muchas líneas de código por lo que es otra de las características por las que la aplicación se decanta hacia un paradigma imperativo.

A su vez, no puede faltar el paradigma de programación orientada a objetos ya que podemos observar clases, métodos e instancias. En este caso, podemos ver una clase llamada Cliente, junto con métodos para registrar sus datos y para que por consola nos muestre que el usuario se ha registrado correctamente.

Nos encontramos también ante una programación asíncrona es decir es una sola persona la que realiza las tareas dentro de la aplicación. Por ejemplo, si Juan efectúa la compra y el pago de los productos Ana no puede hacer lo mismo al mismo tiempo.

- Validación

En cuanto a la validación, una de las que se emplean en mi aplicación es que en el caso de que el cliente ponga más unidades de las que posee la tienda, la aplicación va a dar fallo y no va a poner la factura.

También junto con la clase registrar podemos observar que valida y comprueba los correos electrónicos válidos. En ese caso podemos observar que tanto el correo de Juan como el de Ana son correos válidos mientras que el correo de ejemplo es incorrecto.