HITO INDIVIDUAL PROGRAMACIÓN

Carlos Hernández

Fase 1

a:

Explicar qué es un algoritmo con el ejemplo de Registro de cliente. Escribe con pseudocódigo siguiendo los ejemplos cómo serían los datos de entrada, el procedimiento y los datos de salida.

Un algoritmo es una secuencia precisa de pasos que resuelven un problema en un tiempo finito.

Para que este algoritmo pueda estar bien ejecutado, debe seguir una serie de propiedades o requisitos:

• Deben finalizar.

• Las instrucciones serán concretas.

• Pasos en orden definido.

• Funciona sea cual sea los datos de entrada.

• Problema con varias soluciones. Optimización.

• Independiente del lenguaje y de la máquina.

Para poder hacer el ejemplo de registro necesitaré introducir los **datos de entrada** principalmente, por ejemplo:

- Nombre del cliente
- Apellido del cliente
- Dirección del cliente
- Correo electrónico
- Teléfono
- Tarjeta de crédito.

Estos serán los datos que deberemos pedir en el procedimiento

El **procedimiento** principalmente será:

1. Pedir al cliente que ingrese su nombre.
2. Pedir al cliente que ingrese su apellido.
3. Pedir al cliente que ingrese su dirección.
4. Pedir al cliente que ingrese su teléfono.
5. Pedir al cliente que ingrese su correo.
6. Pedir al cliente que ingrese su tarjeta de crédito.
7. Guardar los datos del cliente en la base de datos
8. Mostrar un mensaje de confirmación al usuario para indicar que el registro se ha completado

Los **datos de salida** serán un mensaje de confirmación del registro exitoso, con un mensaje de este estilo: “cliente registrado exitosamente”.

Aquí va el código:

```
# Datos de entrada
nombre = input("Ingrese el nombre del cliente: ")
apellido = input("Ingrese el apellido del cliente: ")
direccion = input("Ingrese la dirección del cliente: ")
telefono = input("Ingrese el teléfono del cliente: ")
email = input("Ingrese el correo electrónico del cliente: ")
tarjeta = input("Ingrese el número de tarjeta")

# Guardando los datos en un diccionario
cliente = {
    "nombre": nombre,
    "apellido":apellido,
    "direccion": direccion,
    "telefono": telefono,
    "email": email,
    "tarjeta": tarjeta
}

# Guardando el diccionario en una lista
clientes_registrados = []
clientes_registrados.append(cliente)

# Mostrando el mensaje de confirmación
print("El cliente ha sido registrado exitosamente.")
```

b. Diagrama de flujo.

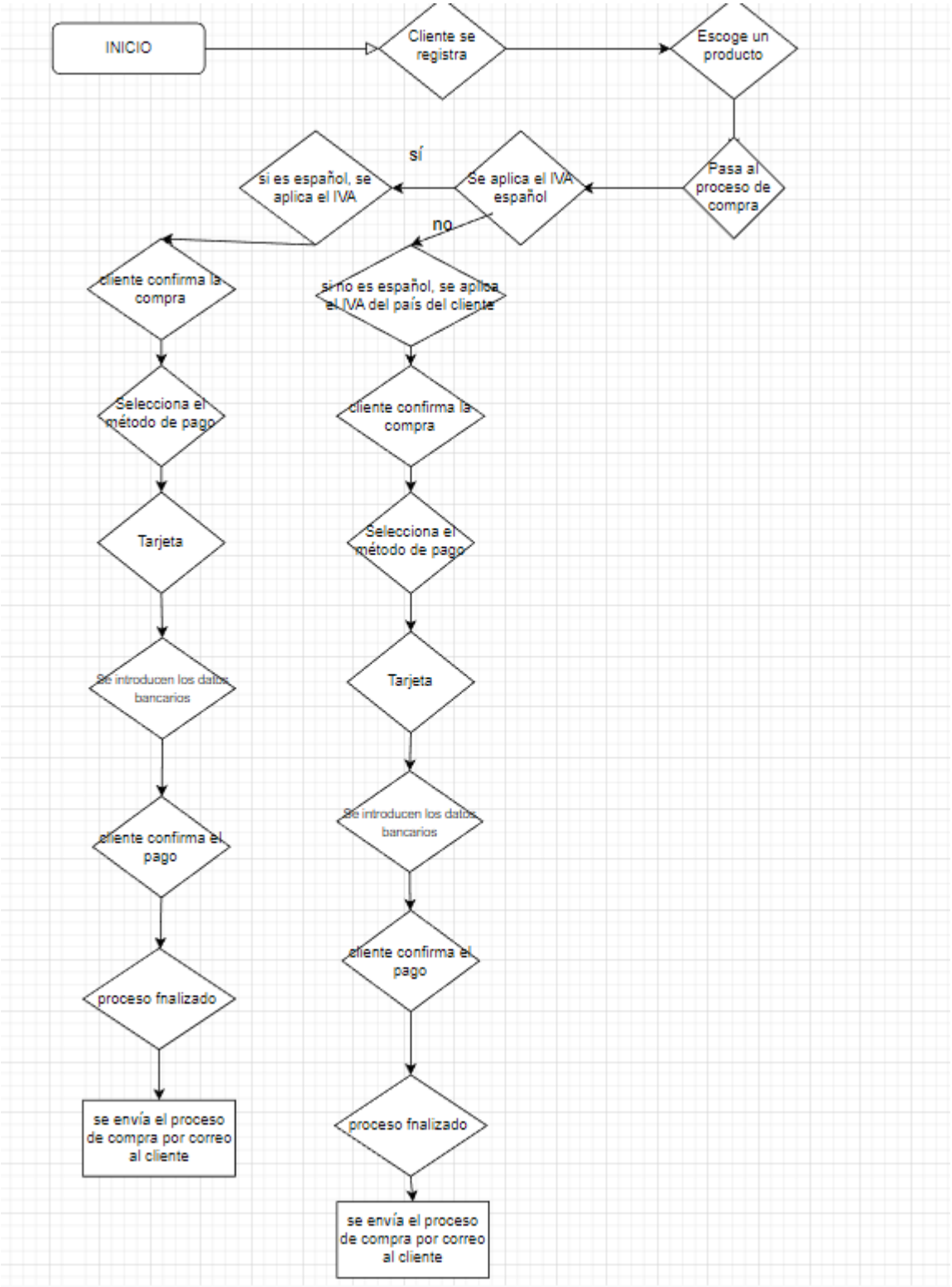
Explica cómo sería un diagrama de flujo a la hora de realizar una compra de un producto. Fíjate que el cliente puede tener un iva español o de otro país. Puedes dibujar el diagrama de flujo con LucidChart.
Ejemplo de crear diagrama de flujo.

Un diagrama de flujo es una representación gráfica que muestra la secuencia de pasos en un proceso.

A continuación explicaré cómo realizaré el diagrama de flujo:

- El proceso comenzará cuando el cliente se registre en la app.
- Luego tendrá que seleccionar el producto que desea comprar.
- Sin embargo, preguntaremos si el cliente tiene un IVA español:
- Si tiene IVA español, se le aplicará el IVA correspondiente al precio del producto que desea comprar y se continúa con el proceso de compra.
- Si su IVA no es Español, se preguntará a qué país pertenece el IVA del cliente.
- Al saber cuál es el IVA del cliente, se aplicará el IVA correspondiente al cliente.
- El cliente confirma la compra.
- Introducirá los datos de su tarjeta
- El proceso de compra se completa y finaliza.
- Se envía un correo con la comprobación de compra.

A continuación va el diagrama de flujo ya representado:



c. Caso de uso.

Explica un caso de uso para el caso de un cliente que se da de alta y de un administrador de nuestra empresa que da de alta un producto.

Un ejemplo podría ser el siguiente:

1. El cliente visita la página web de la empresa y hace clic en el enlace “Registro”.
2. Aparece el formulario de registro en el que el cliente introducirá sus datos personales como nombre, tlf, contraseña, etc.
3. Cuando el cliente haya ingresado los datos, se registrará.
4. Se le mandará al correo electrónico un correo para que pueda validar su registro.
5. Lo verifica.

Ahora el caso de un administrador que da de alta un producto:

1. El administrador accede al sistema de administración, al apartado de productos.
2. El administrador agrega un nuevo producto y rellena el formulario de los datos específicos del producto.
3. Agrega el producto.
4. El producto ya aparece en la web y los clientes podrán comprarlo.

Fase 2

Esta fase en la implementación. el cómo se hace
Es necesario definir las clases principales. cuáles son?
Cada clase debe tener atributos y constructores. Debes definirlos.
También realizar los métodos de la clase. Por ejemplo, el cliente tendrá un método llamado registrar.
Puedes crear una clase llamada Pedidos. Entre otros, tendrá un método llamado imprimirFactura. Este método mostrar por consola "la factura se ha impreso en pdf".

En esta fase definiremos las clases. Son la clase cliente, la clase pedidos

La clase **cliente**: tendrá atributos como nombre, dirección, número de teléfono, correo electrónico y número de tarjeta. y un constructor que comience con estos atributos y un método que los registrará.

La clase **producto**: almacena información sobre los productos, como nombre, precio, descripción, etc. Podría tener atributos como nombre, precio, descripción y un constructor para asignar estos valores al objeto de la clase.

La clase **pedido**: almacena información sobre los pedidos realizados, como productos, cantidades, fecha de entrega, etc. Podría tener atributos como productos, cantidades, fechaEntrega, numeroPedido, costoTotal y un constructor para asignar estos valores al objeto de la clase.
Métodos: dependiendo de la funcionalidad deseada, cada clase podría tener métodos asociados. Por ejemplo, la clase Cliente podría tener un método llamado "registrar" que permita agregar un nuevo cliente al sistema, la clase Pedido podría tener un método llamado "imprimirFactura" que muestre por consola "la factura se ha impreso en pdf".

Aquí va el código:

```
class Cliente:
    def __init__(self, nombre, direccion, telefono):
        self.nombre = nombre
        self.direccion = direccion
        self.telefono = telefono

    def registrar(self):
        #método con el que registraremos al cliente en el
sistema
        pass

class Producto:
    def __init__(self, nombre, precio, descripcion):
        self.nombre = nombre
        self.precio = precio
        self.descripcion = descripcion

class Pedido:
    def __init__(self, productos, cantidades,
fecha_entrega, numero_pedido):
        self.productos = productos
        self.cantidades = cantidades
        self.fecha_entrega = fecha_entrega
```

```
        self.numero_pedido = numero_pedido
        self.costo_total = 0


    def imprimir_factura(self):
        # Lógica para imprimir la factura
        print("La factura se ha impreso en pdf.")
```

Fase 3

En esta fase se define el por qué.
Esta fase es un análisis de la aplicación. Un análisis crítico.
En la ud7 tendremos la presentación de paradigmas. Debes explicar qué paradigma has utilizado. Por qué.
Qué características son las que se incluyen en cada paradigma.
Qué validación has utilizado y si se produce algún error porque el usuario no introduce los datos correctos, cómo funcionaría el programa.

Un paradigma de programación es un estilo de programación de software.

Existen diferentes formas de diseñar un lenguaje de programación y varios modos de trabajar para obtener los resultados que necesitan los programadores.

Realmente se trata de un proceso en el cuál un conjunto de métodos sistemáticos se aplican en los niveles de diseño de programas para poder resolver problemas computacionales.

Los lenguajes de programación adoptan varios paradigmas en función del tipo de órdenes que permiten implementar como, por ejemplo, Python o JavaScript, que son multiparadigmas.

Existen diferentes tipos de paradigmas de programación, son los siguientes:

- Paradigma imperativo
- Paradigma declarativo
- Paradigma funcional
- Paradigma de programación orientado a objetos

El paradigma que usé finalmente fue el de **programación orientado a objetos,** porque podía resolver el problema del registro del cliente, ya que al usar el método de encapsulación, conseguí desarrollar el código de tal manera que al resolver los errores del código, pude encontrarlos de una manera más efectiva, y no ir buscando línea por línea todo el código.

De esta manera pude definir las clases, los atributos y métodos del código que necesitaba usar. Gracias al encapsulamiento también conseguí mantener y proteger la información de las clases, atributos y métodos.

Otros métodos del paradigma de programación orientada a objetos son el polimorfismo o la herencia.

Para que el programa funcionase correctamente, si hubiera algún error, el bucle se cerraría y comenzaría de nuevo el registro.