Universidad Americana

Facultad de Ingeniería y Arquitectura

Algoritmos y Estructuras de Datos

Grupo 4

Guía Práctica #2: Manejo de Listas y Funciones

Realizado por:

Joaquín Alberto Pérez Zúñiga

Docente:

MSc. César Marín López

Ejercicio #1- Código:

```python
"""
Crear un programa que simule la gestión de un inventario en una tienda.
Utilizar un menú para agregar, eliminar, modificar y consultar productos
en el inventario. Cada producto tendrá un código, nombre, cantidad y precio.
"""

from os import system  # para limpiar la pantalla
from typing import Optional


class Producto:
    """
    Clase de datos para representar un producto con:
    - un código,
    - un nombre,
    - una cantidad,
    - un precio
    """

    def __init__(
        self,
        codigo: int = 0,
        nombre: str = "",
        cantidad: int = 0,
        precio: float = 0.0
    ):
        self.codigo = codigo
        self.nombre = nombre
        self.cantidad = cantidad
        self.precio = precio


def pedir_codigo(accion: str) -> int:
    """
    Pide el código del producto a mostrar/modificar/eliminar.
    """

    try:
        codigo = int(input(f"\nIngrese el código del producto que desea {accion}: "))
        if codigo < 0:
            raise ValueError

    except (TypeError, ValueError):
        input("¡El código debe ser un número entero positivo!")
        return pedir_codigo(accion)  # volver a pedir input

    return codigo


def buscar_producto(codigo: int, inventario: list[Producto]) -> Optional[Producto]:
    """
    Busca el índice del producto con el código indicado
    en el inventario de productos registrados.
    Retorna -1 si el producto no existe.
    """

    for producto in inventario:
        if producto.codigo == codigo:
            return producto

    return None


```

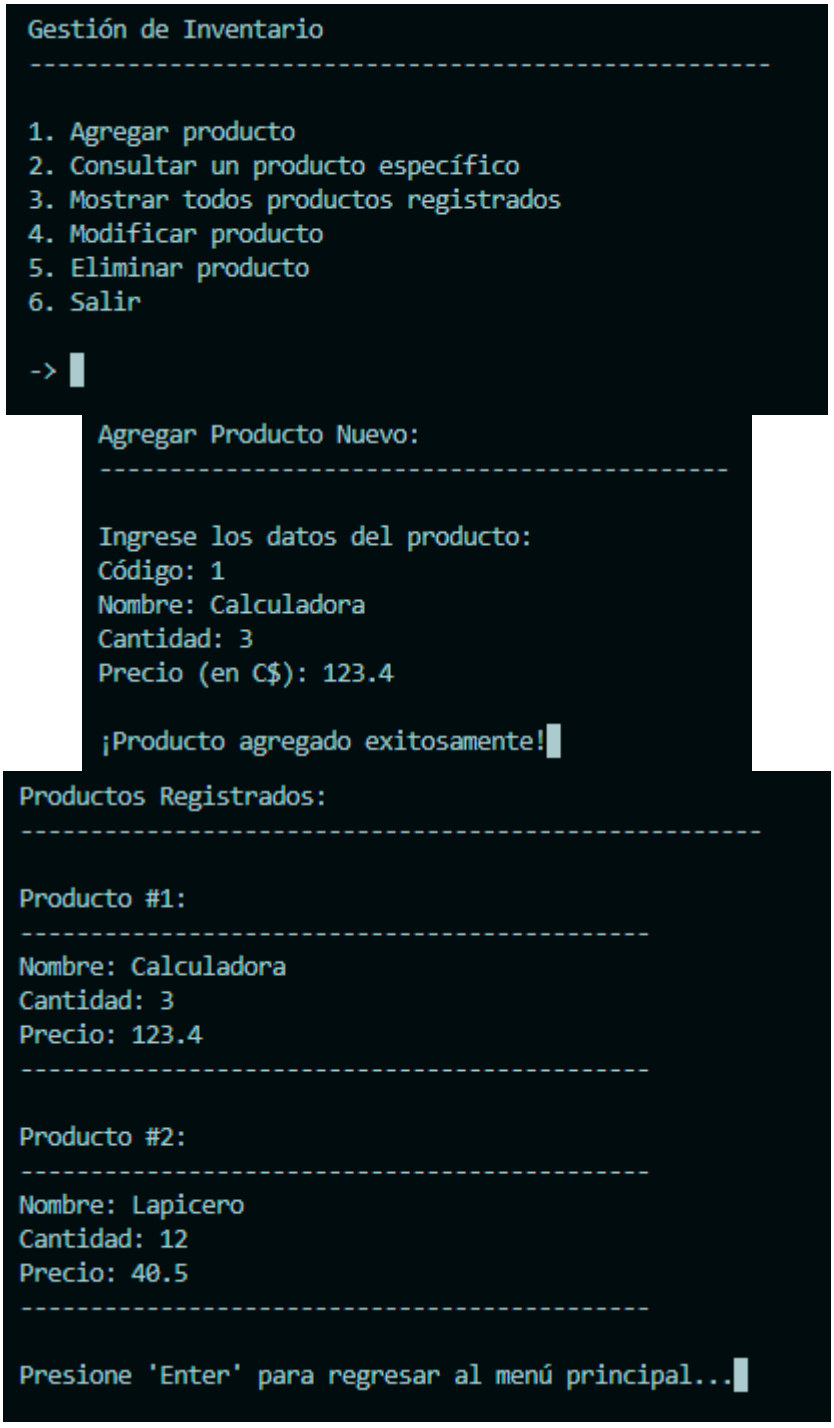
```python
def agregar_producto(inventario: list[Producto]) -> None:
    """
    Pide los datos de un producto nuevo y lo agrega al inventario.
    """

    print("\nAgregar Producto Nuevo:")
    print("-----------------------------------------")

    print("\nIngrese los datos del producto:")
    try:
        codigo = int(input("Código: "))
        nombre = input("Nombre: ")
        cantidad = int(input("Cantidad: "))
        precio = float(input("Precio (en C$): "))

        if codigo < 0 or cantidad < 0 or precio <= 0:
            raise ValueError

    except (TypeError, ValueError):
        input(
            "\n¡El código y la cantidad del producto deben ser números enteros"
            + " positivos, y el precio debe ser un número real positivo!"
        )

        system("cls || clear")
        return agregar_producto(inventario)  # volver a pedir input

    inventario.append(Producto(codigo, nombre, cantidad, precio))
    input("\n¡Producto agregado exitosamente!")
    return None


def consultar_producto(codigo: int, inventario: list[Producto]) -> None:
    """
    Busca un producto específico en el inventario y muestra sus datos.
    """

    producto = buscar_producto(codigo, inventario)
    if producto is None:
        input(f"\n¡No existe un producto con el código '{codigo}'!")
        return

    print(f"\nProducto #{producto.codigo}:")
    print("-----------------------------------------")
    print(f"Nombre: {producto.nombre}")
    print(f"Cantidad: {producto.cantidad}")
    print(f"Precio: {producto.precio}")
    print("-----------------------------------------")


def mostrar_productos(inventario: list[Producto]) -> None:
    """
    Llama consultar_producto() para todos los productos registrados.
    """

    print("\nProductos Registrados:")
    print("-------------------------------------------------")

    for producto in inventario:
        consultar_producto(producto.codigo, inventario)


```

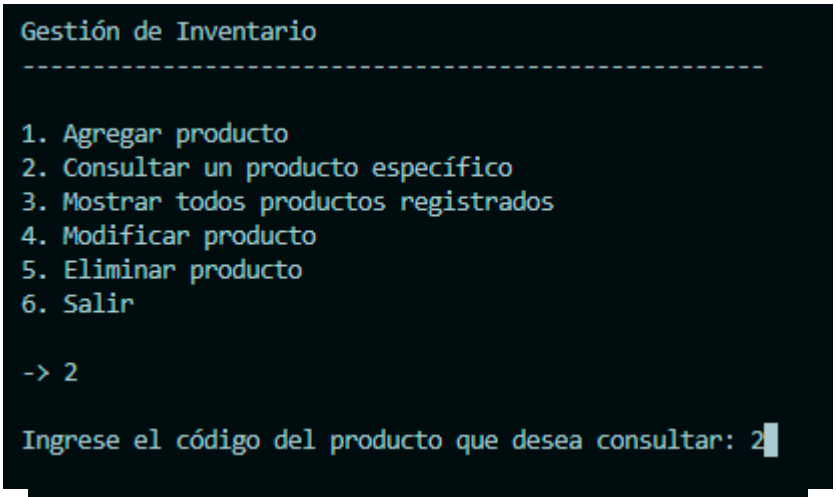
```python
def modificar_producto(codigo: int, inventario: list[Producto]) -> None:
    """
    Permite al usuario ingresar los datos de un producto de nuevo
    para modificarlos (excepto el código, ya que se utiliza
    para identicar los productos).
    """

    producto_modificado = buscar_producto(codigo, inventario)
    if producto_modificado is None:
        input(f"\n¡No existe un producto con el código '{codigo}'!")
        return None

    # eliminar el producto original para sobreescribirlo
    inventario.remove(producto_modificado)

    print("\nModificar Producto:")
    print("-------------------------------------------")

    consultar_producto(codigo, inventario)
    print()

    # pedir nuevos datos
    print("\nIngrese los nuevos datos del producto (dejar en blanco para no modificar):")
    nuevo_nombre = input("Nombre: ")
    nueva_cantidad = input("Cantidad: ")
    nuevo_precio = input("Precio: ")

    # los strings vacios se evaluan como False,
    # entonces si el usuario decide no modificar un atributo,
    # el valor del objeto original permanecera igual

    if nuevo_nombre:
        producto_modificado.nombre = nuevo_nombre

    try:
        if nueva_cantidad:
            producto_modificado.cantidad = int(nueva_cantidad)
        if nuevo_precio:
            producto_modificado.precio = float(nuevo_precio)

        if producto_modificado.cantidad < 0 or producto_modificado.precio <= 0:
            raise ValueError

    except (TypeError, ValueError):
        input(
            "\n¡El código y la cantidad del producto deben ser números enteros"
            + " positivos, y el precio debe ser un número real positivo!"
        )

        system("cls || clear")
        return modificar_producto(codigo, inventario)

    inventario.append(producto_modificado)
    print(
        f"\n¡Datos del producto #{producto_modificado.codigo}"
        + " fueron actualizados exitosamente!"
    )

    return None

```

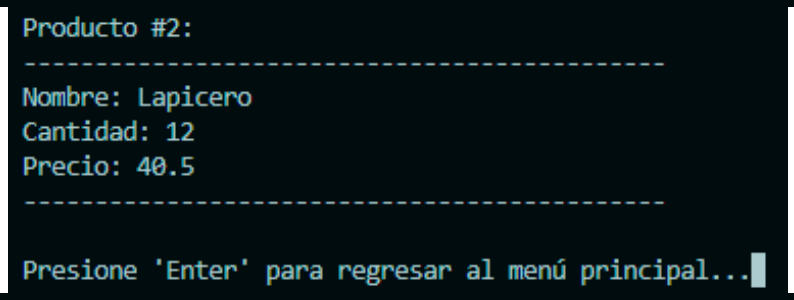
```python
def eliminar_producto(codigo: int, inventario: list[Producto]) -> None:
    """
    Encuentra el índice del producto a eliminar, y lo
    quita del inventario de productos.
    """

    producto = buscar_producto(codigo, inventario)
    if producto is None:
        input(f"\n¡No existe un producto con el código '{codigo}'!")
        return

    print("\nEliminar Producto:")
    print("------------------------------------------")

    consultar_producto(codigo, inventario)
    print()

    confirmacion = input(
        "\n¿Está seguro que desea eliminar el producto seleccionado? (s/n) "
    ).lower()

    if confirmacion == "s":
        inventario.remove(producto)
        input(f"\n¡Producto #{producto.codigo} eliminado exitosamente!")
    elif confirmacion == "n":
        input("\n¡De acuerdo! Regresando al menú principal...")
    else:
        input("\n¡Respuesta inválida! Regresando al menú principal...")


def menu_principal(inventario: list[Producto]) -> None:
    """
    Loop principal del programa que le muestra un menú de opciones al usuario.
    """

    opcion = ""
    while opcion != "6":
        system("cls || clear")
        print("\nGestión de Inventario")
        print("------------------------------------------------------\n")
        print("1. Agregar producto")
        print("2. Consultar un producto específico")
        print("3. Mostrar todos productos registrados")
        print("4. Modificar producto")
        print("5. Eliminar producto")
        print("6. Salir")

        opcion = input("\n-> ")

        match opcion:
            case "1":
                system("cls || clear")
                agregar_producto(inventario)
            case "2":
                codigo = pedir_codigo("consultar")
                system("cls || clear")
                consultar_producto(codigo, inventario)
                input("\nPresione 'Enter' para regresar al menú principal...")
            case "3":
                system("cls || clear")
                mostrar_productos(inventario)
            case "4":
```

```
            case "4":
                codigo = pedir_codigo("modificar")
                system("cls || clear")
                modificar_producto(codigo, inventario)
            case "5":
                codigo = pedir_codigo("eliminar")
                system("cls || clear")
                eliminar_producto(codigo, inventario)
            case "6":
                print("\nSaliendo del programa...\n")
            case _:
                input("\n¡Opción inválida, intente nuevamente!")
        inventario.sort(key=lambda p: p.codigo)


def main() -> None:
    """
    Ejecución del programa.
    """

    inventario: list[Producto] = []
    menu_principal(inventario)


if __name__ == "__main__":
    main()

```

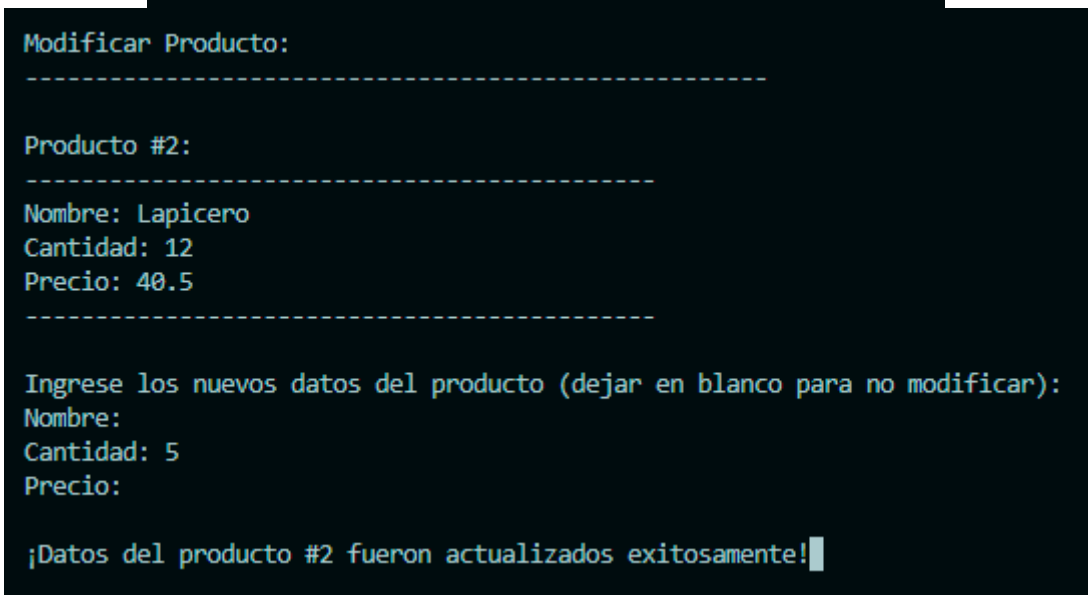
Ejercicio #1 – Ouput:

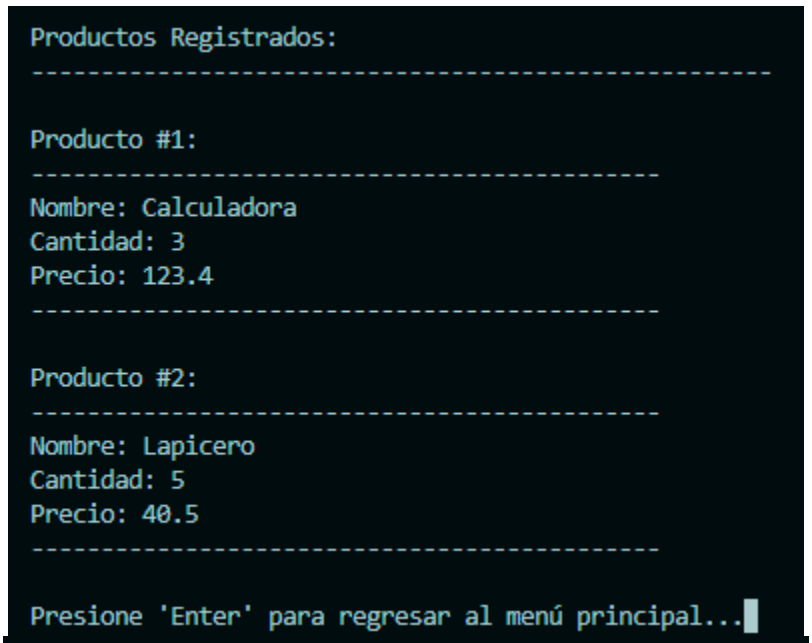


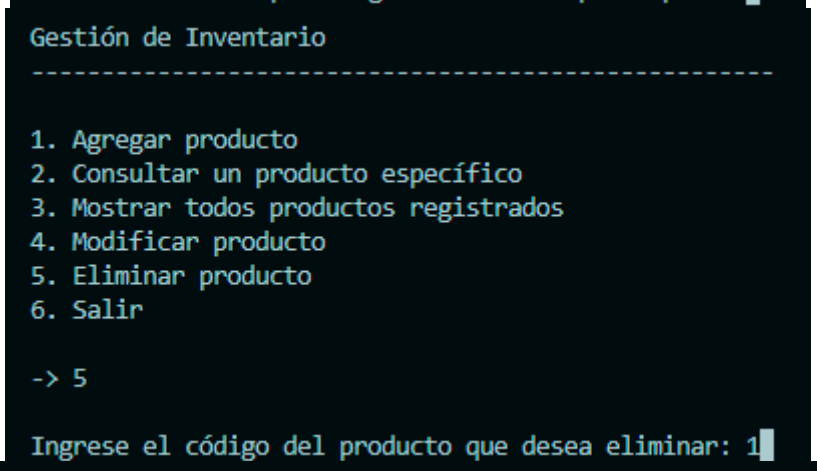

Gestión de Inventario
1. Agregar producto
2. Consultar un producto específico
3. Mostrar todos productos registrados
4. Modificar producto
5. Eliminar producto
6. Salir
-> 2
Ingrese el código del producto que desea consultar: 2

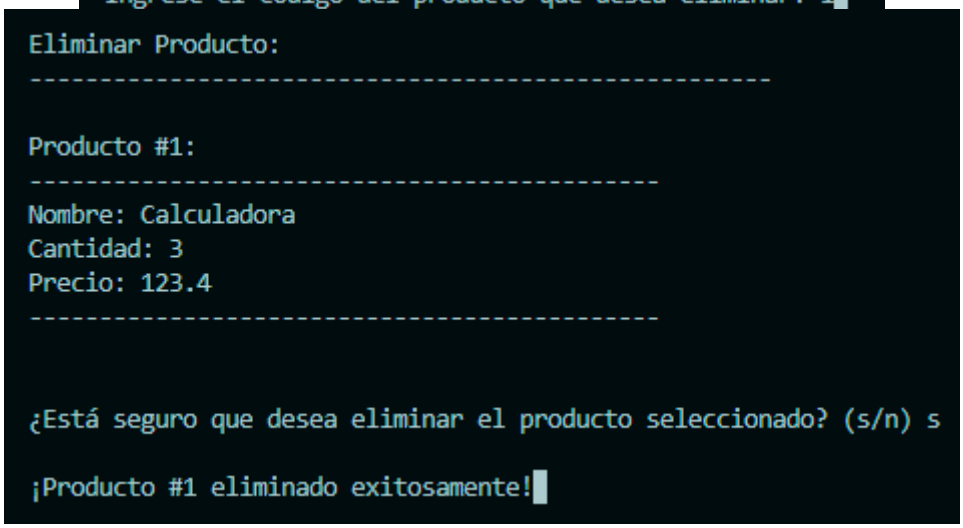

Producto #2:
Nombre: Lapicero
Cantidad: 12
Precio: 40.5
Presione 'Enter' para regresar al menú principal...

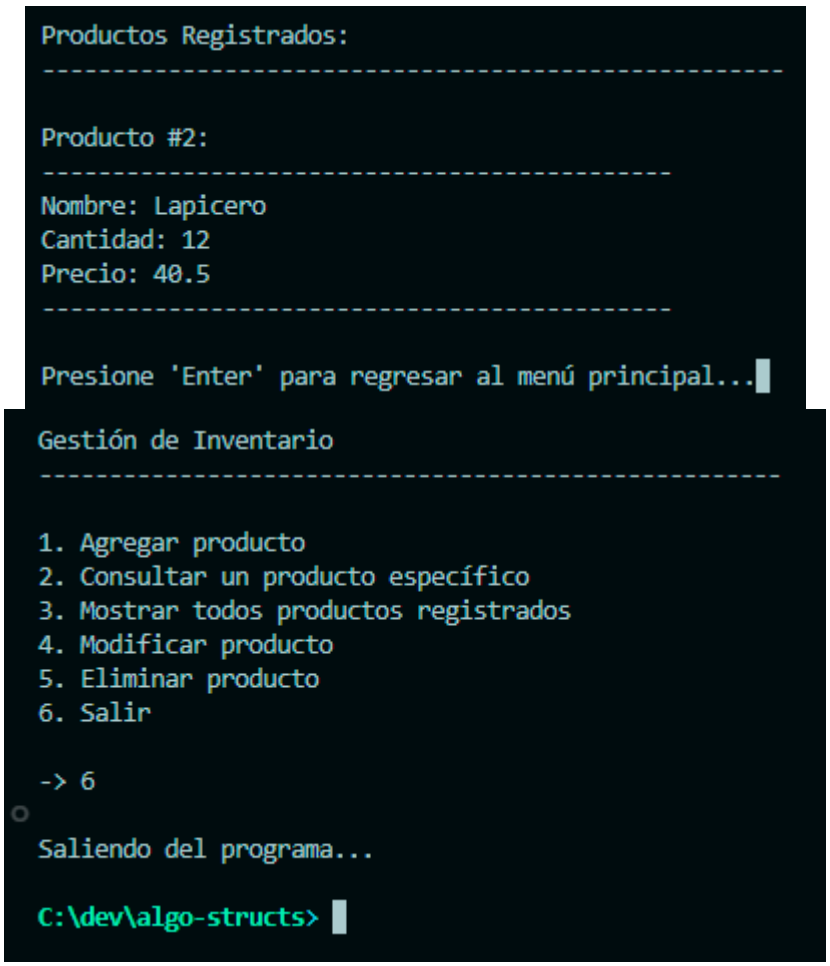

Gestión de Inventario
1. Agregar producto
2. Consultar un producto específico
3. Mostrar todos productos registrados
4. Modificar producto
5. Eliminar producto
6. Salir
-> 4
Ingrese el código del producto que desea modificar: 2


Modificar Producto:
Producto #2:
Nombre: Lapicero
Cantidad: 12
Precio: 40.5
Ingrese los nuevos datos del producto (dejar en blanco para no modificar):
Nombre:
Cantidad: 5
Precio:
¡Datos del producto #2 fueron actualizados exitosamente!


Productos Registrados:
Producto #1:
Nombre: Calculadora
Cantidad: 3
Precio: 123.4
Producto #2:
Nombre: Lapicero
Cantidad: 5
Precio: 40.5
Presione 'Enter' para regresar al menú principal...


Gestión de Inventario
1. Agregar producto
2. Consultar un producto específico
3. Mostrar todos productos registrados
4. Modificar producto
5. Eliminar producto
6. Salir
-> 5
Ingrese el código del producto que desea eliminar: 1


Eliminar Producto:
Producto #1:
Nombre: Calculadora
Cantidad: 3
Precio: 123.4
¿Está seguro que desea eliminar el producto seleccionado? (s/n) s
¡Producto #1 eliminado exitosamente!


Productos Registrados:
Producto #2:
Nombre: Lapicero
Cantidad: 12
Precio: 40.5
Presione 'Enter' para regresar al menú principal...
Gestión de Inventario
1. Agregar producto
2. Consultar un producto específico
3. Mostrar todos productos registrados
4. Modificar producto
5. Eliminar producto
6. Salir
-> 6
Saliendo del programa...
C:\dev\algo-structs>

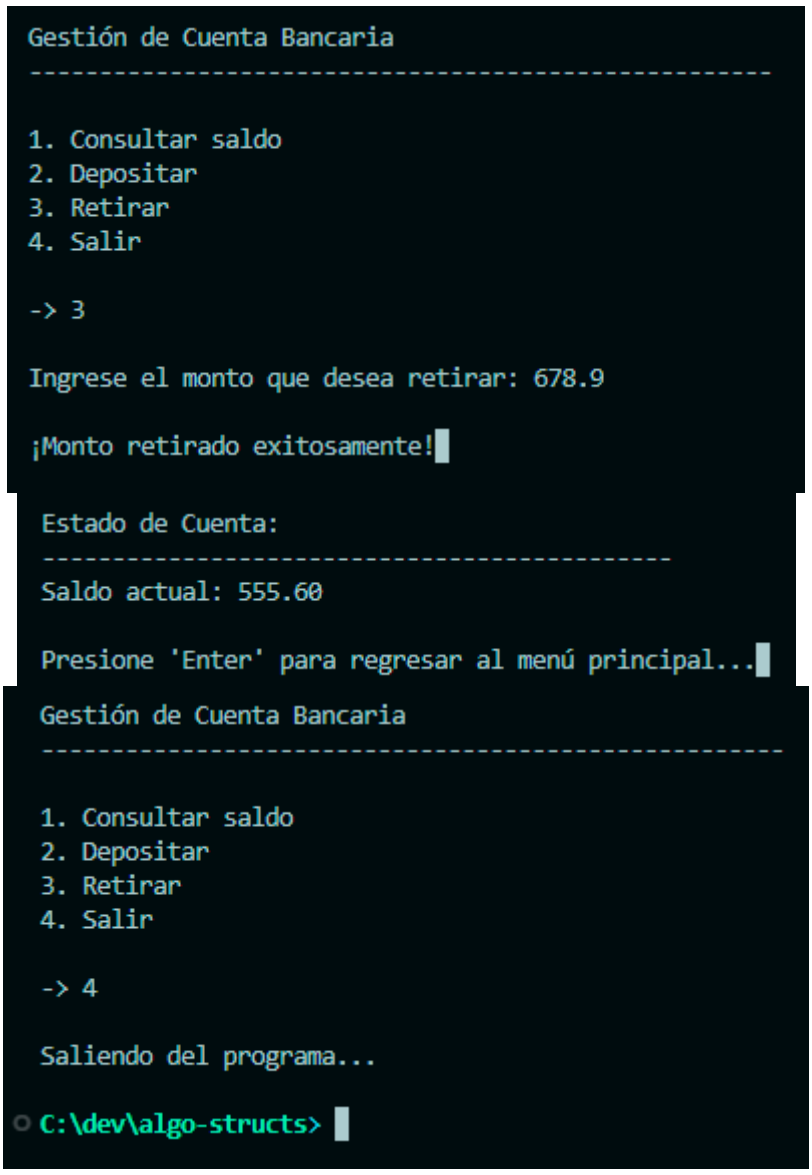
Ejercicio #2 – Código:

```python
"""
Desarrollar un programa que permita al usuario gestionar una cuenta bancaria.
El programa deberá utilizar un menú que permita realizar diferentes operaciones
sobre el saldo de la cuenta. El programa debe permitir al usuario ingresar la
cantidad para depositar o retirar, actualizar el saldo y mostrar los resultados.
Si se elige la opción de retiro, debe verificar que el saldo sea suficiente.
"""

from os import system


def consultar_saldo(saldo: float) -> None:
    """
    Muestra el saldo de la cuenta del usuario.
    """

    print("\nEstado de Cuenta:")
    print("-----------------------------------------")
    print(f"Saldo actual: {saldo:.2f}")
    input("\nPresione 'Enter' para regresar al menú principal...")


def pedir_monto(accion: str) -> float:
    """
    Le pide un monto al usuario para realizar
    una acción con su cuenta bancaria.
    """

    try:
        monto = float(input(f"\nIngrese el monto que desea {accion}: "))
        if monto <= 0:
            raise ValueError
    except (TypeError, ValueError):
        input("\n¡El monto debe ser un número real positivo!")
        return pedir_monto(accion)

    return monto


def depositar(saldo: float) -> float:
    """
    Deposita el monto deseado en la cuenta del usuario.
    """

    saldo += pedir_monto("depositar")
    input("\n¡Monto depositado exitosamente!")
    return saldo


```

```python

def retirar(saldo: float) -> float:
    """
    Retira el monto deseado de la cuenta del usuario.
    """

    monto = pedir_monto("retirar")

    if monto > saldo:
        input(
            "\n¡Error! Monto ingresado es mayor que el saldo"
            + " actual, por favor intente nuevamente..."
        )
    else:
        saldo -= monto
        input("\n¡Monto retirado exitosamente!")

    return saldo


def menu_principal(saldo: float) -> None:
    """
    Loop principal del programa que le muestra un menú de opciones al usuario.
    """

    opcion = ""
    while opcion != "4":
        system("cls || clear")
        print("\nGestión de Cuenta Bancaria")
        print("-----------------------------------------------------\n")
        print("1. Consultar saldo")
        print("2. Depositar")
        print("3. Retirar")
        print("4. Salir")

```

```python

        match opcion:
            case "1":
                system("cls || clear")
                consultar_saldo(saldo)
            case "2":
                saldo = depositar(saldo)
            case "3":
                saldo = retirar(saldo)
            case "4":
                print("\nSaliendo del programa...\n")
            case _:
                input("\n¡Opción inválida, intente nuevamente!")


def main() -> None:
    """
    Ejecución del programa.
    """

    saldo = 0.0
    menu_principal(saldo)


if __name__ == "__main__":
    main()

```

Ejercicio #2 – Output:


Gestión de Cuenta Bancaria
1. Consultar saldo
2. Depositar
3. Retirar
4. Salir
-> 3
Ingrese el monto que desea retirar: 678.9
¡Monto retirado exitosamente!
Estado de Cuenta:
Saldo actual: 555.60
Presione 'Enter' para regresar al menú principal...
Gestión de Cuenta Bancaria
1. Consultar saldo
2. Depositar
3. Retirar
4. Salir
-> 4
Saliendo del programa...
C:\dev\algo-structs>

Ejercicio #3 – Código:

```python
"""
Desarrollar un programa que se comporte como un diccionario Inglés-Español;
esto es, solicitará una palabra en inglés y escribirá la correspondiente
palabra en español. Para hacer más sencillo el ejercicio, el número de
parejas de palabras estará limitado a 5. Una vez finalizada la introducción
de las listas de palabras, pasamos al modo traducción, de forma que si
introducimos "green", la respuesta ha de ser "verde". Si la palabra
no se encuentra, se emitirá un mensaje que lo indique.
"""

from os import system


def crear_diccionario(diccionario: dict[str, str], limite: int = 5) -> dict[str, str]:
    """
    Crea un diccionario de palabras en inglés
    asociadas con su traducción al español.
    """

    if len(diccionario.keys()) == limite:
        return diccionario

    try:
        num_palabras = int(
            input("\n¿Cuántas palabras desea ingresar al diccionario? (máx. 5) ")
        )

        if num_palabras > 5:
            raise ValueError
    except (TypeError, ValueError):
        input(
            "\n¡Error! El tamaño máximo del diccionario es de 5 palabras;"
            + " por favor intente nuevamente..."
        )

        return diccionario

    for i in range(num_palabras):
        print(f"\n\nPalabra #{i + 1}:")
        print("-----------------------------------------")
        ingles = input("Inglés: ")
        espanol = input("Traducción a español: ")
        diccionario[ingles] = espanol

    input("\nPresione 'Enter' para regresar al menú principal...")
    return diccionario


```

```python
def traducir(diccionario: dict[str, str]) -> None:
    """
    Pide una palabra e imprime su traducción a la pantalla.
    """

    palabra = input("\nIngrese la palabra (en inglés) que desea traducir: ")
    if palabra not in diccionario.keys():
        input(
            "\n¡Error! La palabra ingresada no se encuentra en el diccionario;"
            + " por favor intente nuevamente..."
        )
    else:
        print(f"\nInglés: {palabra}\nEspañol: {diccionario[palabra]}")
        input("\nPresione 'Enter' para regresar al menú principal...")

def menu_principal(diccionario: dict[str, str], limite: int = 5) -> None:
    """
    Loop principal del programa que le muestra un menú de opciones al usuario.
    """

    opcion = ""
    while opcion != "6":
        system("cls || clear")
        print("\nDiccionario Inglés-Español:")
        print("-----------------------------------------------------\n")
        print("1. Introducir palabras")
        print("2. Traducir")
        print("3. Salir")

        opcion = input("\n-> ")

        match opcion:
```

```python

        match opcion:
            case "1":
                system("cls || clear")
                crear_diccionario(diccionario, limite)
            case "2":
                system("cls || clear")
                traducir(diccionario)
            case "3":
                print("\nSaliendo del programa...\n")
            case _:
                input("\n¡Opción inválida, intente nuevamente!")


def main() -> None:
    """
    Ejecución del programa.
    """

    limite_palabras = 5
    diccionario: dict[str, str] = {}

    menu_principal(diccionario, limite_palabras)


if __name__ == "__main__":
    main()

```

Ejercicio #3 – Output:

```
Ingrese la palabra (en inglés) que desea traducir: notebook

¡Error! La palabra ingresada no se encuentra en el diccionario; por favor intente nuevamente...
```

```
Ingrese la palabra (en inglés) que desea traducir: keyboard

Inglés: keyboard
Español: teclado

Presione 'Enter' para regresar al menú principal...
```

```
Ingrese la palabra (en inglés) que desea traducir: window

Inglés: window
Español: ventana

Presione 'Enter' para regresar al menú principal...
```