UNIVERSIDAD TECNOLÓGICA DE TULANCINGO

Nombre del trabajo:

Actividad 2.2 -arreglos y matrices

Materia:

” estructura de datos”

Catedrático:

Víctor Manuel Ramírez Soto

Alumno: anhuar fernando martinez islas

Matricula:1722110139

Grupo: TI 41

DESCRIPCIÓN DE LA ACTIVIDAD

ordenamiento y busqueda

```
# Definición de la función de ordenamiento burbuja
def burbuja(A):
    n = len(A)  # Obtiene la longitud del arreglo A
    for i in range(n):  # Bucle externo para recorrer el arreglo
        for j in range(0, n-i-1):  # Bucle interno para comparar elementos y hacer intercambios
            if A[j] > A[j+1]:  # Compara elementos adyacentes
                A[j], A[j+1] = A[j+1], A[j]  # Intercambia los elementos si están en el orden incorrecto

# Definición del arreglo a ordenar
arreglo = [215, 1118, 1110, 897, 686, 408, 172, 888, 885, 180, 56, 1183, 1317, 1148, 408, 818, 606, 774, 663,
           1127, 694, 1351, 794, 1341, 686, 809, 592, 51, 780, 882, 1033, 5, 754, 12, 391, 352, 202, 1112, 1343,
           1067, 1073, 244, 1125, 37, 159, 1288, 1227, 1304, 136, 648, 802, 260, 1106, 1067, 554, 950, 750, 142,
           751, 1315, 1309, 223, 643, 112, 1032, 1030, 249, 440, 1342, 809, 575, 611, 264, 791, 10, 314, 1137,
           66, 435, 1104, 1183, 1224, 462, 1274, 959, 144, 836, 96, 586, 710, 562, 1037, 187, 39, 402, 745, 864,
           104, 1296, 1120, 337, 1322, 1189, 1369, 1378, 449, 251, 1041, 595, 949, 33, 729, 252, 1393, 1232,
           648, 997, 1392, 612, 1115]

# Llama a la función de ordenamiento burbuja para ordenar el arreglo
burbuja(arreglo)

# Imprime el arreglo ordenado
print("Arreglo ordenado:")
print(arreglo)

# Solicita al usuario que ingrese un número
numero = int(input("Ingrese un número: "))
valor_buscado = numero
```

```
posicion = -1  # Inicialmente asumimos que el valor no se encuentra en el arreglo

# Bucle para buscar el valor ingresado en el arreglo
for i in range(len(arreglo)):
    if arreglo[i] == valor_buscado:
        posicion = i
        break

# Muestra el resultado de la búsqueda
if posicion != -1:
    print(f"El valor {valor_buscado} se encuentra en la posición {posicion}.")
else:
    print(f"El valor {valor_buscado} no se encuentra en el arreglo.")
```

gato

```
# Función para imprimir el tablero
def imprimir_tablero(tablero):
    for fila in tablero:
        print(" | ".join(fila))  # Imprime una fila del tablero uniendo las casillas con
"|"
        print("-" * 9)  # Imprime una línea horizontal para separar las filas

# Función para verificar si alguien ha ganado
def verificar_ganador(tablero, jugador):
    # Verificar filas
    for fila in tablero:
        if all(casilla == jugador for casilla in fila):
            return True  # Si todas las casillas en una fila son del mismo jugador,
devuelve True

    # Verificar columnas
    for columna in range(3):
        if all(tablero[fila][columna] == jugador for fila in range(3)):
            return True  # Si todas las casillas en una columna son del mismo jugador,
devuelve True

    # Verificar diagonales
    if all(tablero[i][i] == jugador for i in range(3)) or all(tablero[i][2 - i] ==
jugador for i in range(3)):
        return True  # Si todas las casillas en una diagonal son del mismo jugador,
devuelve True
```

```
    return False  # Si no se encontró una combinación ganadora, devuelve False

# Función principal del juego
def jugar_tic_tac_toe():
    tablero = [[" " for _ in range(3)] for _ in range(3)]  # Crea un tablero vacío (una lista 3x3 de espacios en blanco)
    jugador_actual = "X"  # Inicia el juego con el jugador X
    jugadas = 0  # Contador de jugadas

    print("¡Bienvenido al juego de Tic Tac Toe!")

    while True:  # Bucle principal del juego
        imprimir_tablero(tablero)  # Imprime el tablero actual
        fila = int(input(f"Jugador {jugador_actual}, elige un lugar de forma vertical (0, 1, 2): "))  # El jugador elige una fila
        columna = int(input(f"Jugador {jugador_actual}, elige un lugar de forma horizontal (0, 1, 2): "))  # El jugador elige una columna

        if tablero[fila][columna] == " ":  # Verifica si la casilla seleccionada está vacía
            tablero[fila][columna] = jugador_actual  # Marca la casilla con el jugador actual
            jugadas += 1  # Incrementa el contador de jugadas

            if verificar_ganador(tablero, jugador_actual):  # Verifica si el jugador actual ha ganado
                imprimir_tablero(tablero)
                print(f"¡Jugador {jugador_actual} ha ganado!")
                break  # Sale del bucle si hay un ganador
            elif jugadas == 9:  # Si se han realizado 9 jugadas y no hay ganador, es un empate
                imprimir_tablero(tablero)
                print("¡Es un empate!")
                break  # Sale del bucle en caso de empate

            jugador_actual = "O" if jugador_actual == "X" else "X"  # Cambia al siguiente jugador (X a O o viceversa)
        else:
            print("Esa casilla ya está ocupada. Intenta de nuevo.")  # Mensaje de error si la casilla está ocupada
```

```python
if __name__ == "__main__":
    jugar_tic_tac_toe()  # Inicia el juego cuando se ejecuta el script
```