|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nombre de la práctica** | **Manual de Practicas** | | | **No.** | **1** |
| **Asignatura:** | **Estructura de Datos** | **Carrera:** | **IGN.SISTEMAS COMPU.** | **Duración de la práctica (Hrs)** |  |

**Nombre del Alumno: Kevin Yoan Calderon Garcia**

**I. Competencia(s) específica(s):**

**II. Lugar de realización de la práctica (laboratorio, taller, aula u otro):**

**Salón de clases:**

**Aula**

**Casa**

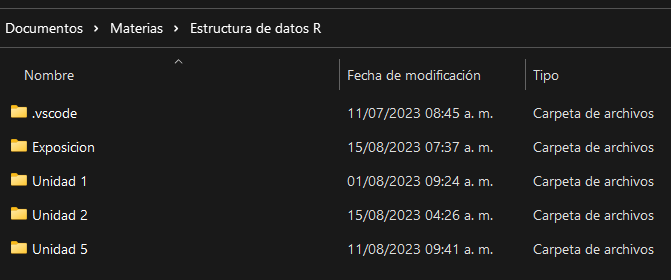
**III. Material empleado:**

Computadora

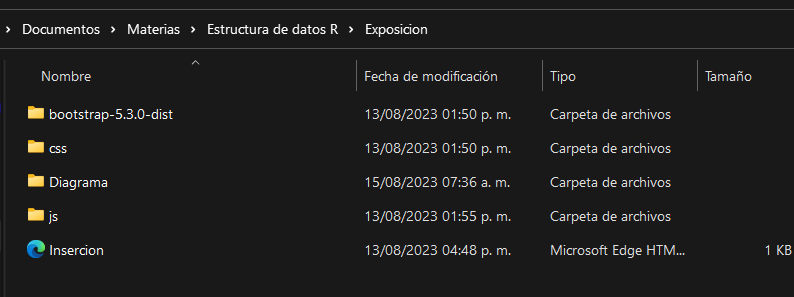
Sitios web

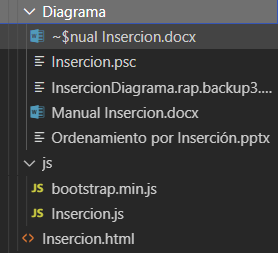
Url

**IV. Desarrollo de la práctica:**

1.- Creamos primero nuestras carpetas en donde colocaremos nuestros trabajos.

2. Agregamos las carpetas de Bootstrap, css y js a nuestra carpeta en donde guardaremos el código.



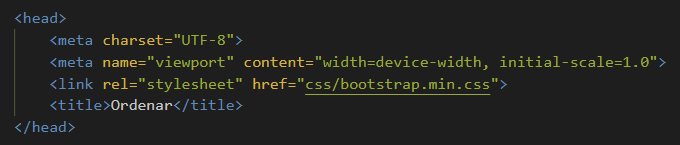
3. Ahora crearemos un formulario con “.html” en donde daremos el diseño y los botones.

4. Utilizamos “<!**DOCTYPE html>**” esta es una declaración que indica al navegador que la página está escrita en HTML, ahora clocamos lo siguiente “**<html lang="es">**” aquí se inicia el elemento raíz del documento HTML. “**lang="es"”** especifica que el idioma utilizado en la página es el español.



5. Ahora abrimos un “**<head>**”, aquí es donde se incluyen etiquetas para configurar el título de la página, vincular hojas de estilo y realizar otras configuraciones, “**<title>Recursividad</title>**” esta etiqueta define el título de la página.

**<link rel="stylesheet" href="css/bootstrap.min.css">**: Esta línea vincula una hoja de estilo externa a la página. Bootstrap es un marco de diseño CSS y JavaScript que proporciona una apariencia y funcionalidades predefinidas para facilitar el diseño y la maquetación de páginas web.

****

6. **<body>**: Es el área donde se muestra el contenido real de la página web, como texto, imágenes, formularios, etc.

**class="bg-primary"**: Las clases son un mecanismo de CSS que permite aplicar estilos o reglas de diseño a uno o varios elementos HTML. En este caso, la clase aplicada es "bg-primary".



7. Aquí solo agregaremos etiquetas para darle título a nuestro formulario.

8. Colocamos un botón con el texto "Registrar". Cuando se hace clic en él, se ejecuta la función **enviarDatos()**.



9. Ahora enlazaremos nuestro formulario con el archivo Insercion.js que nos permitirá darle funcionamiento a nuestro formulario.



10. **Obtener el tamaño del arreglo:** El código comienza declarando una función de flecha llamada **enviarDatos**. Dentro de esta función, se obtiene el valor del tamaño del arreglo desde un elemento de la página HTML con el nombre **'txtNumero'**. La instrucción **document.getElementsByName('txtNumero')[0].value** obtiene el valor del primer elemento con el nombre **'txtNumero'** en el documento HTML y lo almacena en la variable **tamanioArreglo**.

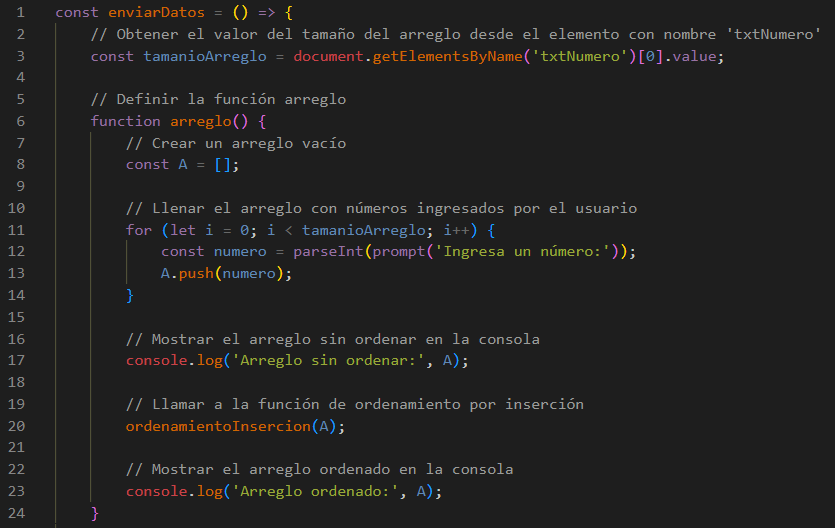
**Definir la función arreglo:** Luego se define otra función dentro de **enviarDatos** llamada **arreglo**. Esta función tiene la tarea de crear un arreglo vacío y llenarlo con números ingresados por el usuario.

**Llenar el arreglo:** La función **arreglo** utiliza un bucle **for** para iterar desde 0 hasta el valor de **tamanioArreglo**. En cada iteración, se muestra un cuadro de diálogo (**prompt**) pidiendo al usuario que ingrese un número. El número ingresado se convierte en un entero usando **parseInt** y luego se agrega al arreglo **A** utilizando el método **push**.

**Mostrar el arreglo sin ordenar:** Después de llenar el arreglo, se muestra en la consola el contenido del arreglo **A** sin ordenar utilizando **console.log('Arreglo sin ordenar:', A)**.

**Llamar a la función de ordenamiento por inserción:** Luego se llama a una función no definida en este fragmento de código llamada **ordenamientoInsercion**. Supongo que esta función debería ser una implementación del algoritmo de ordenamiento por inserción.

**Mostrar el arreglo ordenado:** Después de realizar el posible ordenamiento con la función **ordenamientoInsercion**, se muestra en la consola el contenido del arreglo **A** ya ordenado usando **console.log('Arreglo ordenado:', A)**



11. **Comenzar el ciclo de ordenamiento:** La función **ordenamientoInsercion** toma un arreglo **ar** como argumento. Comienza con un bucle **for** que inicia desde el segundo elemento del arreglo (**i = 1**). Esto se debe a que el primer elemento se considera "ordenado" ya que no tiene elementos a su izquierda para comparar.

**Guardar el valor actual:** En cada iteración del bucle, se guarda el valor del elemento actual en la variable **valorActual**. Esto es necesario porque vamos a desplazar elementos y necesitamos mantener este valor en memoria.

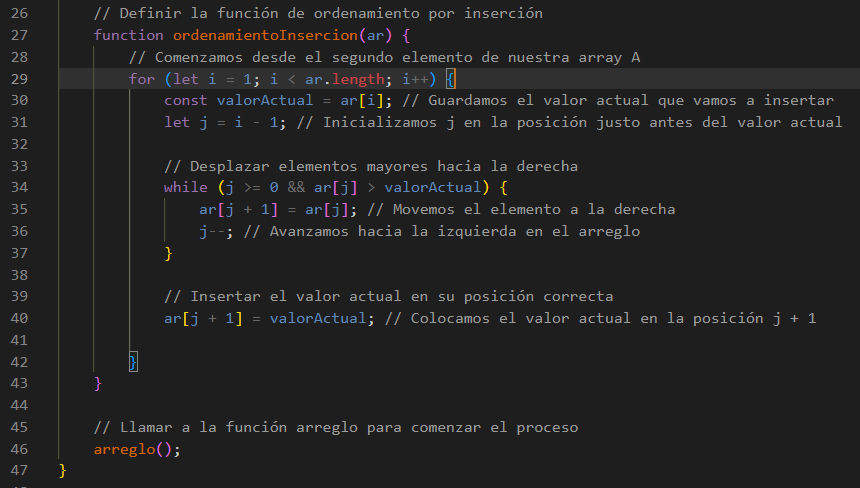
**Inicializar el índice j:** Se inicializa una variable **j** en la posición justo antes del valor actual (**i - 1**). Este índice se utilizará para comparar y desplazar elementos en el arreglo.

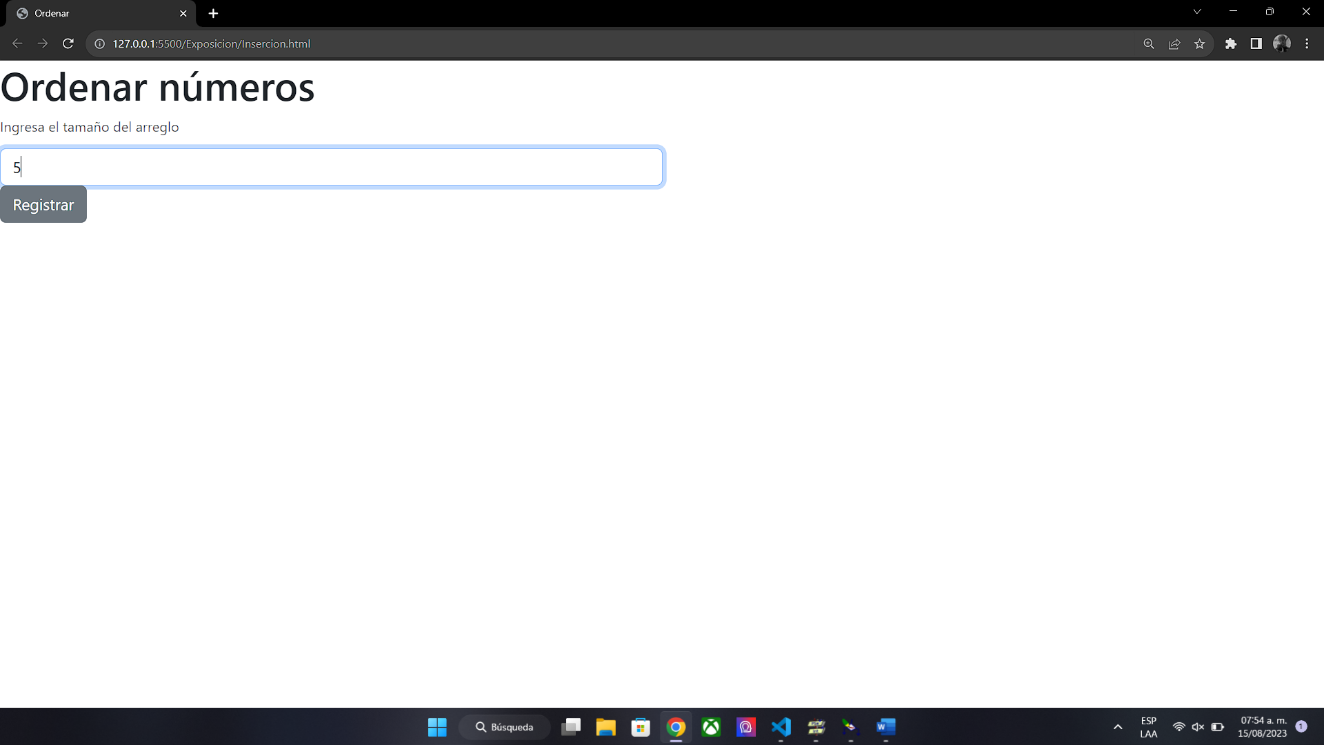
**Desplazar elementos mayores hacia la derecha:** Se entra en un bucle **while** que se ejecuta mientras **j** sea mayor o igual a cero y el valor en la posición **ar[j]** sea mayor que **valorActual**. Esto significa que estamos buscando la posición correcta para insertar **valorActual** dentro del segmento del arreglo que ya está ordenado.

Dentro de este bucle, se desplazan los elementos mayores hacia la derecha. Cada vez que encontramos un elemento mayor que **valorActual**, movemos ese elemento una posición a la derecha (**ar[j + 1] = ar[j]**) y decrementamos **j** para avanzar hacia la izquierda en el arreglo.

**Insertar el valor actual en su posición correcta:** Una vez que se encuentra la posición correcta para **valorActual** dentro del segmento ordenado del arreglo, se coloca **valorActual** en la posición **ar[j + 1]**. Esto coloca el valor actual en la posición que corresponde dentro del arreglo ordenado hasta el momento.

**Llamar a la función arreglo:** Finalmente, después de definir la función **ordenamientoInsercion**, el código llama a la función **arreglo** para comenzar el proceso de llenar y ordenar el arreglo



11. Como resultado nos da lo siguiente:

