Universidad Autónoma de Baja California

Facultad de Ingeniería Arquitectura y Diseño

Ingeniero en Software y tecnologías emergentes

Materia: Programación Estructurada / Clave **36276**

Alumno: Teresa Rivas Gómez

Matrícula: 372565

Maestro: Pedro Núñez Yépiz

Actividad No. : 9

Tema - Unidad : **FUNCIONES y METODOS DE ORDENACION Y BUSQUEDA**

Ensenada, Baja California a 08 de Octubre del 2023

Introducción:

Para la actividad numero 9, repasamos el tema visto en clase acerca de la búsqueda y ordenación de un vector. También estaremos viendo como se hizo uso de nuestra propia librería externa al programa principal para recurrir a nuestras funciones mas usadas con el propósito de visualizar nuestro código de mejor manera.

Este tema nos sera fundamental en nuestro desarrollo como programadores y para la realización de las próximas actividades impartidas en esta materia y en otras relacionadas.

Competencias

La idea es que nos volvamos unos expertos en Métodos de Ordenación y Búsqueda. Esto nos permitirá crear programas avanzados sin tener que depender constantemente de buscar en Internet o consultar libros. En otras palabras, estaremos programando de manera más eficiente y lógica.

Además, cuando usemos las librerías de nuestra propiedad, tendremos más libertad para programar, ya que no tendremos que escribir todas las funciones fundamentales una y otra vez desde cero cada vez que las necesitemos porque estarán ya disponibles.

Documentación

Vectores (Arrays):

Un vector, también conocido como arreglo, es una estructura de datos que almacena elementos del mismo tipo en una secuencia contigua de memoria. El tamaño de un vector debe especificarse al declararlo y no puede cambiar durante la ejecución del programa.

Ejemplo de declaración y uso de un vector en C:

```
int Vector[5];  // Declaración de un vector de enteros con 5 elementos

Vector[0] = 10; // Asignación de un valor al primer elemento

int valor = Vector[2]; // Acceso al tercer elemento
```

Matrices:

Una matriz es una estructura de datos bidimensional que se utiliza para almacenar datos en filas y columnas. En C, las matrices son vectores de vectores, lo que significa que cada elemento de la matriz es un vector.

Ejemplo de declaración y uso de una matriz en C:

```
int Matriz[3][3];  // Declaración de una matriz de enteros 3x3

Matriz[0][0] = 1; // Asignación de un valor al primer elemento (fila 0, columna 0)

int valor = Matriz[1][2]; // Acceso al elemento en la fila 1, columna 2
```

Ordenación de Vectores y Matrices:

La ordenación implica organizar los elementos de un vector o matriz en un cierto orden, como orden ascendente o descendente, según el valor de los elementos. Los algoritmos de ordenación comunes incluyen:

- Ordenación de Burbuja (Bubble Sort): Este algoritmo compara pares de elementos adyacentes y los intercambia si están en el orden incorrecto.
- Ordenación por Selección (Selection Sort): En este algoritmo, se busca el elemento mínimo y se coloca al principio del vector.
- Ordenación por Inserción (Insertion Sort): Este algoritmo construye una lista ordenada uno a uno, tomando cada elemento del vector original y colocándolo en su posición correcta en la lista ordenada.

Ordenación Rápida (Quick Sort): Utiliza la técnica de dividir y conquistar para ordenar los elementos.

Búsqueda en Vectores y Matrices:

La búsqueda implica encontrar un elemento específico dentro de un vector o matriz. Algunos algoritmos de búsqueda comunes son:

- Búsqueda Lineal: Recorre secuencial mente los elementos del vector o matriz uno por uno hasta encontrar el elemento deseado.
- Búsqueda Binaria: Este algoritmo funciona en vectores o matrices ordenadas. Divide repetidamente el conjunto en dos mitades y compara el valor buscado con el valor central.
- Búsqueda en Matrices: Si se busca en una matriz, puedes usar una combinación de búsqueda binaria en filas o columnas, dependiendo de cómo esté organizada la matriz.

Por ejemplo, para buscar en una matriz ordenada por filas y columnas, puedes utilizar la búsqueda binaria en la primera columna para encontrar la fila donde podría estar el elemento, y luego aplicar la búsqueda binaria en esa fila.

Procedimiento

ACTIVIDAD 9

Realiza programa en C utilizando librería propia, el programa deberá tener el siguiente menú.

MENÚ
1.- LLENAR VECTOR
2.- LLENAR MATRIZ
3.- IMPRIMIR VECTOR
4.- IMPRIMIR MATRIZ
5.- ORDENAR VECTOR
6.- BUSCAR VALOR EN VECTOR
0.- SALIR

Universidad Autónoma de Baja California

Facultad de Ingeniería Arquitectura y Diseño

NOTA: El programa deberá repetirse cuantas veces lo desee el usuario, Validado el menú con la función vali_num

I N S T R U C C I O N E S

1.- LLENAR VECTOR .- Llenar vector con 15 números, los números generados aleatoriamente, los números entre el rango de 100 al 200 **(no repetidos)**
2.- LLENAR MATRIZ .- Llenar la matriz de 4x4 con con números generados aleatoriamente, números entre el rango de 1 al 16 **(no repetidos)**
3.- IMPRIMIR VECTOR .- Imprime el vector que se envíe, donde la función recibe como parámetro el vector,tamaño, nombre del vector.
4.- IMPRIMIR MATRIZ.- Imprime la matriz sin importar el tamaño de la matriz recibiendo como parámetros la matriz, la cantidad de renglones y columnas, así como nombre que se le dará a la matriz
5.- ORDENAR VECTOR.- Usar función que ordene el vector por el método de ordenación de la **Burbuja mejorada**.
6.- BUSCAR VALOR EN VECTOR.- Buscar un valor en el vector usando el método de **búsqueda secuencial**.
0.- SALIR

Resultados y Conclusiones

PROGRAMA PRINCIPAL

En esta parte agregare las partes mas relevantes de mi código con su respectiva documentación y comentarios.

- **Declaración de funciones a utilizar:**

```
// Menu para el usuario
int msges();
void menu();

// Buscar en Vector
int BuscarVector(int vector1[], int n);
```

- **Menu Principal:**

```
// Menu
int main()
{
    srand(time(NULL));
    menu();
    return 0;
}
```

- **Mensajes y menu para el usuario:**

```
// Menu para el usuario
int msges()
{
    int op;


system ("CLS");
    printf ("\n MENU DE LA ACTIVIDAD 8 \n");
    printf("1.- LLENAR VECTOR \n");
    printf("2.- LLENAR MATRIZ \n");
    printf("3.- IMPRIMIR VECTOR \n");
    printf("4.- IMPRIMIR MATRIZ \n");
    printf("5.- ORDENAR VECTOR \n");
    printf("6.- BUSCAR VALOR EN VECTOR \n");
    printf("0.- SALIR  \n");
    op = ValidarCadena("Escribe el numero de opcion que elegiste: \n", 0, 6);
    // Retorno a opcion
    return op;
}
// Casos para el menu segun la opcion que elijan
void menu()
{
    int op;
    int n = 15;
    int vector1[15];
    int matriz[4][4];
    // Ciclo
    do{
        system("CLS");
        op=msges();
        // Casos
        switch (op)
        {
        case 1:
                LlenarVectorSinRepetir(vector1, n, 100, 200);
                printf("Vector llenado.\n");
                EsperarUsuario();
                break;
        case 2:
                LlenarMatrizSinRepetir(4, 4, matriz, 1, 16);
                printf("Matriz llenada.\n");
                EsperarUsuario();
                break;
        case 3:
                ImprimirVector(vector1, n);
                EsperarUsuario();
                break;
        case 4:
                ImprimirMatriz(4, 4, matriz);
                EsperarUsuario();
                break;
        case 5:
```

```
                OrdenarVector(vector1, n);
                printf("Vector ordenado, vuelva a seleccionar la opcion 3.\n");
            EsperarUsuario();
            break;
        case 6:
            BuscarVector(vector1, n);
            EsperarUsuario();
            break;
        case 0:
            printf("Saliendo del programa.\n");
            break;
        }
    } while (op != 0);
}
```

- **Función para mostrar e imprimir mensajes de buscar en el vector un numero:**

```
// BUSCAR VECTOR
int BuscarVector(int vector1[], int n)
{



    // Declarar variable
    int ValorBuscar, posicion;
    // Validar el valor buscado
    ValorBuscar = ValidarCadena("Ingresa el valor que buscas (Entre 100 y 200):  ", 100, 200);
    // Realizar la búsqueda utilizando BusqSecVector
    posicion = BusqSecVector(vector1, n, ValorBuscar);
    // Condición para comprobar si se encontró o no
    if (posicion != -1)
    {
        printf("El numero ha sido encontrado, esta en la posicion: %d\n", posicion);
    }
    else
    {
        printf("El numero introducido no esta en el vector\n");
    }
    // Retorna la posicion encontrada o -1 si no se encuentra
    return posicion;
}
```

PROGRAMA LIBRERIA mcdonalds.h

- **Declaración de funciones a utilizar:**

```
// - FUNCIONES - //

// PARA VALIDAR:
int ValidarCadena(char mensj[], int ri, int rf);
int DigitoEncontrado(int vector[], int longitud, int num);
```

```
// PARA ENTRADA Y SALIDA DE DATOS:
void EsperarUsuario();
// PARA VECTORES Y MATRICES:
void LlenarVectorSinRepetir(int vector1[], int n, int ri, int rf);
void LlenarMatrizSinRepetir(int m, int n, int matriz[][4], int ri, int rf);
void ImprimirVector(int vector[], int n);
void ImprimirMatriz(int n, int m, int matriz[][4]);
void OrdenarVector(int vector[], int n);
int BusqSecVector(int vector[], int n, int num);
/*-------------------------------------------------------*/
```

- **Funcion de validación para evitar el uso de scanf:**

```
// - VALIDACION - //
// CADENA: El valor del numero entre los rangos especificados.
int ValidarCadena(char mensj[], int ri, int rf)
{
    // Declarar variables
    int num;
    // Cadena que va a leer el mensaje ingresado
    char cadena[200];
    do
    {
        printf("%s", mensj);
        // Borrar basura
        fflush(stdin);
        gets(cadena);
        num = atoi(cadena);
    } while (num < ri || num > rf);
    // Retorna el valor qUe num, entre los rangos dados
    return num;
}
```

- **Función para encontrar un digito en el vector, la podemos implementar en funciones como búsqueda de un digito/numero:**

```
// DIGITO: Busca un valor en especifico en el vector
int DigitoEncontrado(int vector[], int longitud, int num)
{
    // Declarar variable
    int i;
    // Ciclo para recorrer el vector
    for (i = 0; i < longitud; i++)
    {
        if (vector[i] == num)
        {
            // Retorna a la posicion en la que esta el numero en el vector
            return i;
        }
```

```
        }
        // Retorna a -1 si no esta
    return -1;
}
```

- **Para esperar la entrada del usuario:**

```
/*---------------------------------------------------------*/
// - DATOS - //
// ESPERA DE DATOS: para mostrar un mensaje de espera y esperar la entrada del usuario
void EsperarUsuario()



{
    printf("\nPresiona ENTER para continuar...");
    // Espera a que el usuario presione ENTER
    getchar();
}
```

- **Llenar vector:**

```
/*---------------------------------------------------------*/
// - VECTORES Y MATRICES - //
// LLENAR VECTOR: Random sin repetir
void LlenarVectorSinRepetir(int vector1[], int n, int ri, int rf)
{
    // Declarar variables
    int num, rango;
    int i;
    // Inicializar variables
    num = 0;
    rango = (rf - ri) + 1;
    // Ciclo para llenar el vector con los valores random
    for (i = 0; i < n; i++)
    {
        // Ciclo para que no haya repeticiones
        do
        {
            num = (rand() % rango) + ri;
            // Se manda a llamar a la funcion para encontrar digitos
        } while (DigitoEncontrado(vector1, i, num) != -1);
        vector1[i] = num;
    }
}
```

- **Llenar matriz:**

```
// LLENAR MATRIZ: Random sin repetir
void LlenarMatrizSinRepetir(int m, int n, int matriz[][4], int ri, int rf)
{
```

```
    // m y n son las dimensiones de la matriz
    // Declarar variables, llenamos el vector multiplicando
    int vector[m * n];
    LlenarVectorSinRepetir(vector, m * n, ri, rf);
    // 2 ciclos anidados para asignarle los valores que tiene el vector a la matriz
    int k = 0;
    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < n; j++)
        {
            matriz[i][j] = vector[k++];
        }
    }
}
```

- **Imprimir vector:**

```
// IMPRIMIR VECTOR: Random sin repetir
void ImprimirVector(int vector[], int n)
{
    // Declarar variable


 int i;
    // n = Longitud del vector
    for (i = 0; i < n; i++)
    {
        // Vector que se imprimira
        printf("Vector [%d]: [%2d]\n", i, vector[i]);
    }
}
```

- **Imprimir matriz:**

```
// IMPRIMIR MATRIZ: Random sin repetir
void ImprimirMatriz(int n, int m, int matriz[][4])
{
    // 2 ciclos anidados para poder recorrer la matriz completamente.
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            // Imprime cada elemento con \t para darle formato
            printf("[%2d]\t", matriz[i][j]);
        }
        printf("\n");
    }
}
```

- **Ordenar:**

```c
// ORDENAR VECTOR
void OrdenarVector(int vector[], int n)
{
    // Declarar funciones
    int i, j;
    int temp;
    // Ciclo 1
    for(i = 0; i < n; i++)
    {
        // Ciclo 2 para comparar
        for(j = 0; j < n; j++)
        {
            // Comparacion
            if(vector[j] > vector[i])
            {
                // Usar variable temporal como espacio para la comparacion
                temp = vector[i];
                vector[i] = vector[j];
                vector[j] = temp;
            }
        }
    }
}
```

- **Búsqueda Secuencial:**

```c
// BUSQUEDA SECUENCIAL
int BusqSecVector(int vector[], int n, int num)
{
    // Declarar función
    int i;


   // Ciclo para recorrer el vector
    for(i = 0; i < n; i++)
    {
        if(vector[i] == num)
        {
            return i;
        }
    }
    return -1;
}
```

Referencias

Sitios Web:

Metodos de Ordenación de Vectores. (2012, November 30). Vectores Y Matrices.

https://vectoradsi.wordpress.com/2012/11/29/metodos-de-ordenacion-de-vectores/

Búsqueda en arreglos y vectores - Programación en C. (n.d.). Solución Ingenieril.

https://solucioningenieril.com/programacion_en_c/busqueda_en_arreglos_y_vectores

PDF:

https://kesquivel.files.wordpress.com/2012/09/tema_1_algoritmosordenacionbusqueda1.pdf