Ingeniero en Software y tecnologías emergentes

Materia: Programación Estructurada / Clave **36276**

Alumno: Solano Meza Angel Daniel

Matrícula: 372453

Maestro: Pedro Núñez Yépiz

Actividad No. : 9

Tema - Unidad : Funciones, metodos de ordenacion y busqueda - Unidad 5

Ensenada Baja California a 30 de Septiembre del 2023

Incluir libreria

```
// Solano Meza Angel Daniel Matr. 372453
// 05/10/2023
// Creacion de nuestra libreria e implementacion de funciones propias
// ADSM_ACT9_932

#include "miti.h"
```

Llamada a funciones

```
void menu()
{
    int op, num, i;
    int vector[15], matriz[4][4];
    do
    {
        op = msges();
        switch (op)
        {
        case 1:
            LlenarVector(vector, 15, 100, 200, 1);
            break;
        case 2:
            LlenarMatrizR(matriz, 4, 4, 1, 16);
            break;
        case 3:
            ImprimirVector(vector, 15, "VECTOR");
            break;
        case 4:
            ImprimirMatriz(matriz, 4, 4, "MATRIZ");
            break;
        case 5:
            Ordenar(vector, 15);
            break;
        case 6:
            system("CLS");
            printf("NUMERO QUE DESEAS ENCONTRAR: \n");
            num = Validar(100, 200);
            i = BusquedaSec(vector, 15, num);
            if (i != -1)
            {
                printf("EL VALOR SE ENCUENTRA EN EL INDICE %d\n",i);
            }
            else
            {
                printf("EL VALOR NO SE ENCUENTRA EN EL VECTOR\n");
            }
            system("PAUSE");
```

```
            }
            system("PAUSE");
            break;
        }

    } while (op != 0);
}
```

Capturas de mi codigo

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int Validar(int inferior, int superior);
void VectorMatriz(int vector[], int m, int ri, int rf);
void LlenarVector(int vector[], int m, int ri, int rf, int op);
void LlenarMatrizM(int matriz[][4], int m, int n, int ri, int rf);
void LlenarMatrizR(int matriz[][4], int m, int n, int ri, int rf);
int BusquedaSec(int vector[], int n, int num);
void ImprimirVector(int vector[], int m, char msg[]);
void ImprimirMatriz(int matriz[][4], int m, int n, char msg[]);
void Ordenar(int vector[], int n);
//*********************
```

ImprimirMatriz(int [][4], int, int, char []) (function)

```
int Validar(int inferior, int superior) // Parametros para funcionar
{
    int num;
    char cadena[10];
    do
    {
        fflush(stdin);
        gets(cadena);       // Tomas los datos como cadena
        num = atoi(cadena); // Transforma la entrada a entero
    } while ((num < inferior) || (num > superior));
    return num;
}
//*********************
void VectorMatriz(int vector[], int m, int ri, int rf)
{
    int i, num, rango;
    rango = rf - ri + 1;
    for (i = 0; i < m; i++) // Le asgina un valor a cada indice del vector
    {

        do
        {
            num = rand() % rango + ri;
        } while (BusquedaSec(vector, i, num) != -1);
        vector[i] = num;
    }
}
//*********************
```

```c
void LlenarVector(int vector[], int m, int ri, int rf, int op)
{
    system("CLS");
    int i, num, rango;
    rango = rf - ri + 1;
    if (op == 1) // Opcion rellenado automatico
    {
        for (i = 0; i < m; i++) // Le asgina un valor a cada indice del vector
        {

            do
            {
                num = rand() % rango + ri;
            } while (BusquedaSec(vector, i, num) != -1);
            vector[i] = num;
        }
    }
    else // Opcion rellenado manual
    {
        for (i = 0; i < m; i++)
        {
            printf("    Espacio [%d]\n", i + 1);
            vector[i] = Validar(30, 70); // Valida la entrada manual
        }
    }
    printf("VECTOR RELLENADO\n");
    system("PAUSE");
}
```

```c
void LlenarMatrizR(int matriz[][4], int m, int n, int ri, int rf)
{
    system("CLS");
    int i, j, k, vect[m * n];
    VectorMatriz(vect, m * n, ri, rf);
    for (i = 0, k = 0; i < m; i++)
    {
        for (j = 0; j < n; j++)
        {
            matriz[i][j] = vect[k++];
        }
    }
    printf("MATRIZ RELLENADA\n");
    system("PAUSE");
}
//***************************
void LlenarMatrizM(int matriz[][4], int m, int n, int ri, int rf)
{
    system("CLS");
    int i, j;
    for (i = 0; i < m; i++)
    {
        for (j = 0; j < n; j++)
        {
            matriz[i][j] = Validar(ri, rf);
        }
    }
    printf("MATRIZ RELLENADA\n");
    system("PAUSE");
}
//***************************
```

```c
int BusquedaSec(int vector[], int n, int num)
{
    system("CLS");
    int i;
    for (i = 0; i < n; i++)
    {
        if (vector[i] == num)
        {
            return i;
        }
    }
    return -1;
}
//****************************
void ImprimirVector(int vector[], int m, char msg[])
{
    system("CLS");
    int i;
    printf("    %s\n", msg);
    for (i = 0; i < m; i++)
    {
        printf("[%d] --> %d\n", i + 1, vector[i]);
    }
    system("PAUSE");
}
//****************************
```

```c
void Ordenar(int vector[], int n)
{
    system("CLS");
    int i, j, temp;
    for (i = 0; i < n; i++)
    {
        for (j = i + 1; j < n; j++)
        {
            if (vector[j] < vector[i])
            {
                temp = vector[i];
                vector[i] = vector[j];
                vector[j] = temp;
            }
        }
    }
    ImprimirVector(vector, n, "VECTOR ORDENADO");
}
//****************************
void ImprimirMatriz(int matriz[][4], int m, int n, char msg[])
{
    system("CLS");
    int i, j;
    printf("    %s\n", msg);
    for (i = 0; i < m; i++)
    {
        for (j = 0; j < n; j++)
        {
            printf("[%d][%d] --> %d\n", i + 1, j + 1, matriz[i][j]);
        }
    }
    system("PAUSE");
}
```