Universidad Autónoma de Baja California

Facultad de Ingeniería Arquitectura y Diseño

Ingeniero en Software y tecnologías emergentes

Materia: Programación Estructurada / Clave **36276**

Alumno: Maria Fernanda Medrano Calderon

Matrícula: 373143

Maestro: Pedro Núñez Yépiz

Actividad No. : 8

Tema - Unidad : Unidad 1 - ARREGLOS EN C
(Cadenas [vectores y matrices])

Ensenada Baja California a 24 de Marzo del 2024.

Universidad Autónoma de Baja California

Facultad de Ingeniería Arquitectura y Diseño

1. INTRODUCCIÓN

En esta práctica, se mostrarán los ejercicios realizados en base a los temas vistos en clase, los ejercicios fueron programados en Visual Studio Code en formato ".cpp" y los elementos principales para realizar la práctica son:

- Estructuras de control repetitivas
- Funciones
- Ciclos
-Metodos de optimizacion
-Cadenas
-Vectores
-Matrices

Con lo visto en clase y videos de ayuda, realizaremos algunos ejercicios para poner en práctica lo aprendido en clase.

2. COMPETENCIA

Con esta práctica lograremos entender las cadenas, que son, como funcionan, tanto su características principales como maneras de realizar procedimientos con ayuda de ciclos

3. FUNDAMENTOS

En esta práctica, lo que consulté para realizar los ejercicios fueron códigos que realizó en semestres pasados, además del código base proporcionado por el profesor ("esqueleto.c") y videos del material extra proporcionado por el profesor junto a videos de youtube.

4. PROCEDIMIENTO

Realiza programa en C el programa deberá tener el siguiente menú.

MENÚ

1.- LLENAR VECTOR 1 (MANUALMENTE)
2.- LLENAR VECTOR 2 ALEATORIAMENTE
3.- LLENAR VECTOR 3 (CON VECTOR1 Y VECTOR2)
4.- IMPRIMIR VECTORES
5.- LLENA MATRIZ 4 X 4
6.- IMPRIMIR MATRIZ
0.- SALIR

NOTA: EL PROGRAMA DEBERÁ REPETIRSE CUANTAS VECES LO DESEE EL USUARIO
NOTA 2: EL **VECTOR 1** DE **10** POSICIONES, NÚMEROS DEL **30 AL 70**
NOTA 3: EL **VECTOR 2** DE **10** POSICIONES CON NÚMEROS GENERADOS ALEATORIAMENTE DEL **1 AL 20 (SIN REPETIR)**
NOTA 4: EL **VECTOR 3** DE **20** POSICIONES, CON LOS DATOS DEL ARREGLO1 Y ARREGLO2
NOTA 5: MATRIZ 4 X 4 LLENARLA CON LOS DATOS DEL VECTOR1 Y VECTOR2,

1.- Inicie creando las funciones para cada una de las opciones del menú

```
void llenarvect(int vect[], int m);
void llenarvect2(int vect[], int m);
void llenarvect3(int vect[], int vect1[], int vect2[], int m);
void imprimirvect(int vect[], int m);
void llenarMat(int matriz[][M], int vect1[], int vect2[]);
void imprimirmat(int matriz[][M]);
```

2,- En lo siguiente solamente llame al menú en el "main()" para agilizar la parte de mostrar el menú, selección, etc.

```
int inicio()
{
    int op;
    system("CLS");
    printf(" ----- MENU ----- \n");
    printf("1.- LLENAR VECTOR 1 (MANUALMENTE)\n");
    printf("2.- LLENAR VECTOR 2 ALEATORIAMENTE \n");
    printf("--- Para la opcion 3 en adelante es obligatorio realizar las opci
    printf("3.- LLENAR VECTOR 3 (CON VECTOR1 Y VECTOR2) \n");
    printf("4.- IMPRIMIR VECTORES \n");
    printf("5.- LLENA MATRIZ 4 X 4 \n");
    printf("6.- IMPRIMIR MATRIZ \n");
    printf("0.- SALIR \n");
    printf("ESCOGE UNA OPCION: ");
    scanf("%d", &op);
    return op;
}

void menu()
{
    int vect2[N], vect1[N], vect3[N * 2], matriz[M][M];
    int op;
    do
    {
        op = inicio();
        switch (op)
        {
        case 1:
            llenarvect(vect1, N);
            system("pause");
            break;
        case 2:
```

3.- Ejemplos de las funciones que llame para cada paso

```
        break;
    case 2:
        llenarvect2(vect2, N);
        printf("Se lleno el VECTOR No.2\n");
        system("pause");
        break;
    case 3:
        llenarvect3(vect3, vect1, vect2, N * 2);
        printf("Se lleno el VECTOR No.3\n");
        system("pause");
        break;
    case 4:
        printf("VECTOR No. 1:\n");
        imprimirvect(vect1, N);
        printf("VECTOR No. 2:\n");
        imprimirvect(vect2, N);
        printf("VECTOR No. 3:\n");
        imprimirvect(vect3, N * 2);
        system("pause");
        break;

    case 5:
        llenarMat(matriz, vect1, vect2);
        printf("Se lleno la MATRIZ\n");
        system("pause");
        break;

    case 6:
        printf("La MATRIZ:\n");
        imprimirmat(matriz);
        system("pause");
        break;
```

5.- En la primera función sólo guardaba los datos dados por el usuario en cada posición y con la condición de que debían ser números entre el 30 y el 70.

```
void llenarvect(int vect[], int m)
{
    int num, i;
    for (i = 0; i < m; i++)
    {
        printf("Ingresa el numero de la posicion No. %d: ", i);
        scanf("%d", &num);
        vect[i] = num;
        if (num < 30 || num > 70)
        {
            printf("El numero debe estar entre el 30 al 70!!\n");
            i--;
        }
    }
}
```

6.- En la siguiente que se debía llenar aleatoriamente aplique un ciclo de condición que al generar un número lo guardaba y en cada vez que se generaba uno nuevo debía ser comparado con los que ya estaban guardados para que no hubiera ninguno repetido.

```
void llenarvect2(int vect[], int m)
{
    int num_existen[20] = {0};
    int i, num;

    for (i = 0; i < m; i++)
    {
        do
        {
            num = rand() % 20 + 1;
        } while (num_existen[num - 1]);
        vect[i] = num;
        num_existen[num - 1] = 1;
    }
}
```

7.- Para el tercer vector se debía llenar con los anteriores por lo que solamente cree el vector para ingresar los del primer vector y después de superar el número posiciones (0 al 9), continuará llenado lo que faltaba con el vector 2.

```
void llenarvect3(int vect[], int vect1[], int vect2[], int m)
{
    int i;
    for (i = 0; i < m; i++)
    {
        if (i < N)
        {
            vect[i] = vect1[i];
        }
        else
        {
            vect[i] = vect2[i - N];
        }
    }
}
```

8.- para mostrar los vectores solamente los llamaba con su número correspondiente con la siguiente función

```
void imprimirvect(int vect[], int m)
{
    int i;
    for (i = 0; i < m; i++)
    {
        printf("%d ----> [%d] \n", i, vect[i]);
    }
}
```

9.- En la parte de las matrices realice algo parecido para llenar el tercer vector solo que esta vez tenía que ingresar los valores dependiendo la posición por lo que al final de llenar la matriz solo se usan algunos del vector 2

```
void llenarMat(int matriz[][M], int vect1[], int vect2[])
{
    int i, j, n = 0;
    for (i = 0; i < M; i++)
    {
        for (j = 0; j < M; j++)
        {
            if (n < N)
            {
                matriz[i][j] = vect1[n];
            }
            else
            {
                matriz[i][j] = vect2[n - N];
            }
            n++;
        }
    }
}
```

10.- Y finalmente para presentar la matriz imprimía cada posición , pues en todas ya había un valor asignado (con la función antes explicada.

```
void imprimirmat(int matriz[][M])
{
    int i, j;
    for (i = 0; i < M; i++)
    {
        for (j = 0; j < M; j++)
        {
            printf("[%d]\t", matriz[i][j]);
        }
        printf("\n");
    }
}
```

5. RESULTADOS Y CONCLUSIONES

En conclusión, en esta práctica mejore en lo que fue la comprensión de los vectores, sin embargo tuve algunas complicaciones, al momento de las pociones, pues eso junto a evitar la repetición del vector 2, tuve revisar muchos ejemplos de internet principalmente para poder entender y así poder hacer de mejor manera las funciones, pero aparte de eso no fue una práctica con la que tuve grandes complicaciones, pues en semestres pasados a había realizado matrices y vectores (tanto manuales como aleatorias) por lo que creo que se cumplio en su mayoria el objetivo de esta practica.

6. ANEXOS

PDF con los códigos y salida en ANEXO……

7. REFERENCIAS

Diseño de algoritmos y su codificación en lenguaje C

Corona, M.A. y Ancona, M.A. (2011)..

España: McGraw-Hill.

ISBN: 9786071505712

Programación estructurada a fondo:implementación de algoritmos en C

:Pearson Educación.Sznajdleder, P. A. (2017)..

Buenos Aires,Argentina: Alfaomega

Como programar en C/C++

H.M. Deitel/ P.J. Deitel

Segunda edición

Editorial: Prentice Hall.

ISBN:9688804711

Programación en C.Metodología, estructura de datos y objetos

Joyanes, L. y Zahonero, I. (2001)..

España:McGraw-Hill.

ISBN: 844813013