Universidad Autónoma de Baja California
Facultad de Ingeniería Arquitectura y Diseño

Ingeniería en Software y Tecnologías Emergentes.

Programación estructurada

Actividad 09

Vectores y matrices

Danna Guadalupe Sandez Islas

373080

1.- LLENAR VECTOR .- Llenar vector con 15 números, los números generados aleatoriamente, los números entre el rango de 100 al 200 **(no repetidos)**

Codigo:

```c
// Funcion para llenar un vector con datos random
void llenarVect(int vector[], int n, int ri, int rf)
{
    // variables locales
    int num;
    int i;
    int rango = (rf - ri) + 1;
    /* desarrollo de la funcion */
    for (i = 0; i < n; i++)
    {
        do
        {
            num = rand() % rango + ri;
        } while (repetidoVect(num, i, vector) != 0);
        vector[i] = num;
    }
}
```

Implementacion en actividad:

```c
        switch (op)
        {
        case 1:
            llenarVect(vector, 15, 100, 200);
            printf("El vector se lleno exitosamente!\n");
            break;
```

Salida:

```
El vector se lleno exitosamente!
Presione una tecla para continuar . . .
```

2.- LLENAR MATRIZ .- Llenar la matriz de 4x4 con con números generados aleatoriamente, números entre el rango de 1 al 16 (no repetidos)

Desarrollo:

```c
void llenarMatriz4x4(int matriz[][M], int n, int m, int vect[])
{
    // Variables locales
    int i, j, k = 0;
    /* Desarrollo de funcion */
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < m; j++)
        {
            matriz[i][j] = vect[k];
            k++;
        }
    }
}
```

Implementacion en actividad:

```c
        break;
    case 2:
        llenarVect(vect, (N * M), 1, 16);
        llenarMatriz4x4(matriz, N, M, vect);
        printf("La matriz se lleno exitosamente!\n");
        break;
```

Salida:

```
La matriz se lleno exitosamente!
Presione una tecla para continuar . . .
```

3.- IMPRIMIR VECTOR .- Imprime el vector que se envíe, donde la función recibe como parámetro el vector,tamaño, nombre del vector.

Desarrollo:

```c
void printVect(int vector[], int n)
{
    int i;
    printf("--Vector --\n");
    for (i = 0; i < n; i++)
    {
        printf("Posicion[%d] --> %d\n", i + 1, vector[i]);
    }
}
```

Implementación en la actividad:

```c
case 3:
    printVect(vector, 15);
    break;
```

Salida:

```
--Vector --
Posicion[1] --> 149
Posicion[2] --> 171
Posicion[3] --> 154
Posicion[4] --> 110
Posicion[5] --> 104
Posicion[6] --> 158
Posicion[7] --> 165
Posicion[8] --> 151
Posicion[9] --> 180
Posicion[10] --> 184
Posicion[11] --> 173
Posicion[12] --> 167
Posicion[13] --> 132
Posicion[14] --> 108
Posicion[15] --> 170
Presione una tecla para continuar . . .
```

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**

Desarrollo:

```
void printMatriz(int n, int m, int matriz[][m])
{
    printf("~~~ Matriz ~~~\n");
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            printf("%d ", matriz[i][j]);
        }
        printf("\n");
    }
}
```

Implementación en la actividad:

```
case 4:
    printMatriz(N,M,matriz);
    break;
```

Salida:

```
~~~ Matriz ~~~
16 13 8 15
5 6 10 7
4 9 14 1
11 12 3 2
Presione una tecla para continuar . . .
```

5.- ORDENAR VECTOR.- Usar función que ordene el vector por el método de ordenación de la Burbuja mejorada.

Desarrollo:

```c
void ordenar(int vector[], int n)
{
    int i, j;
    int temporal;
    for (i = 0; i < n; i++)
    {
        for (j = i + 1; j < n; j++)
        {
            if (vector[j] < vector[i])
            {
                temporal = vector[i];
                vector[i] = vector[j];
                vector[j] = temporal;
            }
        }
    }
}
```

Implementacion en la actividad:

```c
case 5:
    ordenar(vector, 15);
    printf("Vector ordenado exitosamente!\n");
    break;
```

Salida:

```
--Vector --
Posicion[1] --> 104
Posicion[2] --> 108
Posicion[3] --> 110
Posicion[4] --> 132
Posicion[5] --> 149
Posicion[6] --> 151
Posicion[7] --> 154
Posicion[8] --> 158
Posicion[9] --> 165
Posicion[10] --> 167
Posicion[11] --> 170
Posicion[12] --> 171
Posicion[13] --> 173
Posicion[14] --> 180
Posicion[15] --> 184
Presione una tecla para continuar . . .
```

6.- BUSCAR VALOR EN VECTOR.- Buscar un valor en el vector usando el método de búsqueda secuencial.

Desarrollo:

```c
int busq_seq(int vector[], int n, int ri, int rf)
{
    int num;
    num = validar("Que valor te gustaria buscar entre 100-200?: ", ri, rf);
    int i;
    for (i = 0; i < n; i++)
    {
        if (vector[i] == num)
        {
            return i;
        }
    }
    return -1;
}
```

Implementacion en la actividad:

```c
case 6:
    num= busq_seq(vector, 15,100,200);
    if (num == -1)
    {
        printf("El numero no se encuentra dentro del vector\n");
    }
    else
    {
        printf("Numero en la posicion %d",num);
    }
    break;
}
```

Salida:

```
Que valor te gustaria buscar entre 100-200?: 120
El numero no se encuentra dentro del vector
Presione una tecla para continuar . . .
```

```
Que valor te gustaria buscar entre 100-200?: 110
Numero en la posicion 2Presione una tecla para continuar . . .
```