



Universidad Autónoma de Baja
California
Facultad de Ingeniería Arquitectura
y Diseño



Materia: Programación Estructurada

Alumno: Fabian Aguiar Sergio

Carrera: Ingeniero en computación

Fecha: 22/03/2024

Matrícula: 374317

Maestro: Pedro Nuñez Yepiz

Práctica No. #8

INTRODUCCIÓN

Los arreglos en C son estructuras de datos que permiten almacenar múltiples elementos del mismo tipo bajo un mismo nombre, organizados en una secuencia contigua de memoria. Cada elemento del arreglo puede ser accedido mediante un índice, que indica su posición dentro del arreglo. Los arreglos en C tienen un tamaño fijo que se determina en el momento de su declaración y no puede cambiar durante la ejecución del programa. Son muy útiles para almacenar y manipular conjuntos de datos de manera eficiente en programas C

COMPETENCIA

Utilice funciones para llenar los vectores con números dentro de los rangos especificados.

Implemente una función para llenar el tercer vector con los datos de los dos primeros vectores.

mejorar la presentación del menú y la interacción con el usuario.

Añadiremos comentarios explicativos en el código para una mejor comprensión.

Esto ayudándome a poder saber cómo funcionan los arreglos y mejorar al usarlos.

FUNDAMENTOS

El código presentado en lenguaje C se centra en el manejo de arreglos para llevar a cabo operaciones como llenar vectores con valores específicos o aleatorios, sumar vectores y llenar matrices utilizando los datos de estos vectores. Se emplean funciones para dividir el código en tareas específicas, como el llenado de vectores, la suma de estos y la impresión de resultados. Además, se utilizan estructuras de control de flujo para mostrar un menú al usuario y ejecutar acciones según su selección. En síntesis, este código representa una práctica básica para la manipulación de arreglos en el contexto del lenguaje de programación C, facilitando la organización, manipulación y visualización de datos almacenados en vectores y matrices.

Los arreglos (arrays) son variables del mismo tipo de dato que tienen el mismo nombre y que se distinguen y diferencian por un índice.

Sintaxis:

<tipo> <variable> [N]

Se declara un arreglo de nombre **<variable>** con **N** elementos de tipo **<tipo>**, (**N es una constante**).

Ejemplo:

```
int a[10];
```

Los arreglos se caracterizan por:

- Almacenan los elementos en posiciones contiguas

de memoria Tienen un mismo nombre de variable que representa a todos los elementos. Para hacer referencia a esos elementos es necesario utilizar un índice que especifica el lugar que ocupa cada elemento dentro del archivo.

PROCEDIMIENTO

El procedimiento del código implica mostrar un menú al usuario con varias opciones, como llenar vectores, sumarlos, llenar matrices y mostrar resultados. El usuario elige una opción y el programa ejecuta la acción correspondiente utilizando funciones específicas. Después de completar una acción, el programa vuelve al menú para permitir al usuario seleccionar otra opción o salir del programa. Este ciclo se repite hasta que el usuario elige salir.

ACTIVIDAD 8

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 VECTOR 2)
- 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**

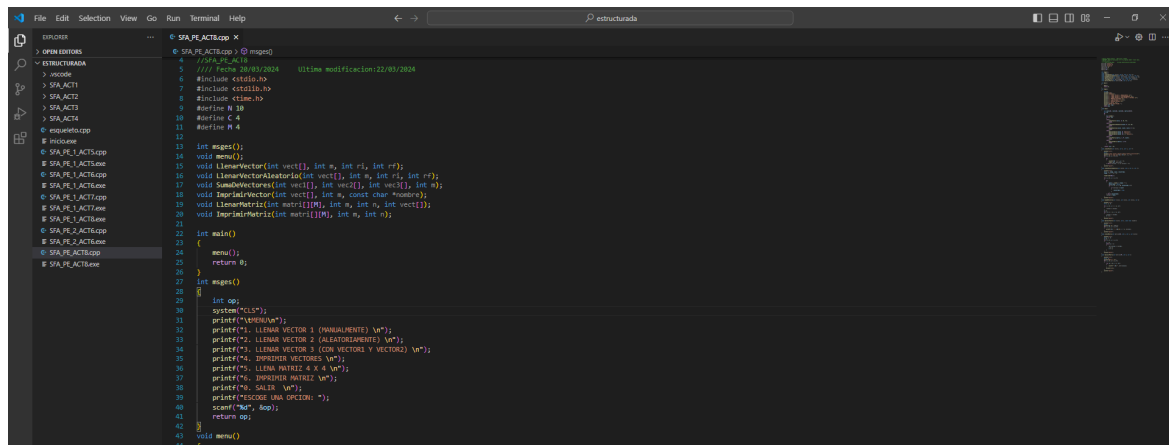
NOTA 3: EL **VECTOR 2** DE **10** POSICIONES CON NÚMEROS GENERADOS ALEATORIAMENTE DEL **1 AL 20 (SIN REPETIR)**

NOTA 4: EL **VECTOR 3** DE **20** POSICIONES, CON LOS DATOS DEL ARREGLO 1 Y ARREGLO 2

NOTA 5: MATRIZ 4 X 4 LLENARLA CON LOS DATOS DEL VECTOR1 Y VECTOR 2.

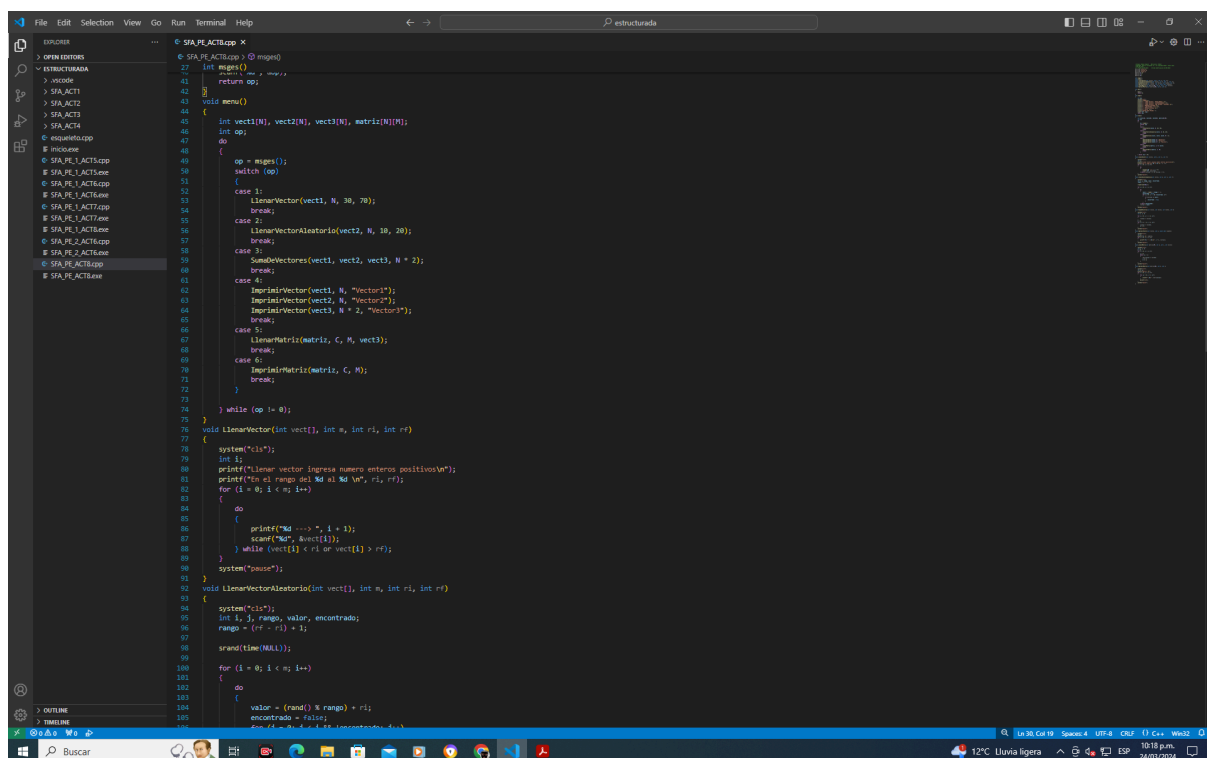
RESULTADOS

Se proporciona al usuario la capacidad de ingresar manualmente valores para el Vector 1 en el rango de 30 a 70, mientras que el Vector 2 se llena con valores aleatorios únicos entre 10 y 20, evitando duplicados. Luego, se suman los elementos de Vector 1 y 2 para formar el Vector 3, que duplica su tamaño. Se muestran los tres vectores en pantalla. Posteriormente, se llena una matriz 4x4 con los datos del Vector 3, organizándose por filas, y se imprime esta matriz.



```
1  // SPA_P3_1.cpp
2  // Fecha: 20/03/2024
3  // Última modificación: 22/03/2024
4  #include <iostream>
5  #include <cstdlib>
6  #include <ctime>
7  #include <conio.h>
8  #include <time.h>
9  #define N 30
10 #define C 4
11 #define M 4
12
13 int main()
14 {
15     void llenarVector(int vect[], int n, int ri, int rf);
16     void llenarVectorAleatorio(int vect[], int n, int ri, int rf);
17     void SumaDeVectores(int vect1[], int vect2[], int vect3[], int n);
18     void ImprimirVector(int vect[], int n, const char *mensaje);
19     void ImprimirMatriz(int mat[][M], int n, int m);
20     void ImprimirMatriz(int mat[][M], int n, int m);
21
22     int op;
23     int main()
24     {
25         menu();
26         return 0;
27     }
28
29     int op;
30     system("cls");
31     printf("\nMENU\n");
32     printf("1. LLENAR VECTOR 1 (MANUALMENTE) W\n");
33     printf("2. LLENAR VECTOR 2 (ALEATORIAMENTE) W\n");
34     printf("3. LLENAR VECTOR 3 (CON VECTORES 1 Y VECTORES 2) W\n");
35     printf("4. IMPRIMIR VECTORES W\n");
36     printf("5. LLENAR MATRIZ 4 X 4 W\n");
37     printf("6. IMPRIMIR MATRIZ W\n");
38     printf("7. SALIR W\n");
39     printf("Escriba una opcion: ");
40     scanf("%d", &op);
41     return op;
42 }
43 void menu()
44 {
```

Fig.1 MENÚ



```
27 int main()
28 {
29     return op;
30 }
31 void menu()
32 {
33     int vect1[N], vect2[N], vect3[N], matriz[M][M];
34     int op;
35     do
36     {
37         op = menu();
38         switch (op)
39         {
40             case 1:
41                 llenarVector(vect1, N, 30, 70);
42                 break;
43             case 2:
44                 llenarVectorAleatorio(vect2, N, 10, 20);
45                 break;
46             case 3:
47                 SumaDeVectores(vect1, vect2, vect3, N * 2);
48                 break;
49             case 4:
50                 ImprimirVector(vect1, N, "vector1");
51                 ImprimirVector(vect2, N, "vector2");
52                 ImprimirVector(vect3, N * 2, "vector3");
53                 break;
54             case 5:
55                 llenarMatriz(matriz, C, M, vect3);
56                 break;
57             case 6:
58                 ImprimirMatriz(matriz, C, M);
59                 break;
60         }
61     } while (op != 0);
62 }
63 void llenarVector(int vect[], int n, int ri, int rf)
64 {
65     system("cls");
66     int i;
67     printf("Llenar vector ingrese numero enteros positivos W\n");
68     printf("En el rango del %d al %d W\n", ri, rf);
69     for (i = 0; i < n; i++)
70     {
71         do
72         {
73             printf("%d ----> ", i + 1);
74             scanf("%d", &vect[i]);
75             while (vect[i] < ri || vect[i] > rf)
76             {
77                 printf("Error\n");
78                 continue;
79             }
80             system("pause");
81         } while (true);
82     }
83     void llenarVectorAleatorio(int vect[], int n, int ri, int rf)
84     {
85         system("cls");
86         int i, j, rango, valor, encontrado;
87         rango = (rf - ri) + 1;
88         srand(time(NULL));
89         for (i = 0; i < n; i++)
90         {
91             do
92             {
93                 valor = (rand() % rango) + ri;
94                 encontrado = false;
95                 for (j = 0; j < i; j++)
96                 {
97                     if (vect[j] == valor)
98                     {
99                         encontrado = true;
100                         break;
101                     }
102                 }
103                 if (!encontrado)
104                     vect[i] = valor;
105             } while (encontrado);
106         }
107     }
108 }
```

Fig.2 La parte de llenar vector y llenar vector aleatorio

```
118 void SumaDeVectores(int vec1[], int vec2[], int vec3[], int n)
119 {
120     system("cls");
121     int i, j, k;
122     for (i = 0; i < n / 2; i++)
123     {
124         vec3[i] = vec1[i];
125     }
126     k = 0;
127     for (j = n / 2; j < n; j++)
128     {
129         vec3[j] = vec2[k];
130         k++;
131     }
132     system("pause");
133 }
134 void ImprimirVector(int vec1[], int n, const char *nombre)
135 {
136     system("cls");
137     int i;
138     printf("Mostrar %s", nombre);
139     for (i = 0; i < n; i++)
140     {
141         printf("%d --> [%d] %s", i + 1, vec1[i]);
142     }
143     system("pause");
144 }
145 void LlenarMatriz(int mat1[][M], int m, int n, int vec1[])
146 {
147     system("cls");
148     int i, j, k;
149     for (i = 0; i < m; i++)
150     {
151         j = 0;
152         while (j < n)
153         {
154             mat1[i][j] = vec1[k];
155             j++;
156             k++;
157         }
158     }
159     system("pause");
160 }
161 void ImprimirMatriz(int mat1[][M], int m, int n)
162 {
163     system("cls");
164     int i, j;
165     printf("Mostrar Matriz %s", mat1);
166     for (i = 0; i < m; i++)
167     {
168         for (j = 0; j < n; j++)
169         {
170             printf(" [%d] ", mat1[i][j]);
171         }
172         printf("\n");
173     }
174     system("pause");
175 }
```

Fig 3. La parte de suma de vector, llenar matriz e imprimir matriz.

CONCLUSIÓN

En resumen, el código proporciona una interfaz interactiva para realizar operaciones con vectores y matrices en el lenguaje de programación C. A través de un menú de opciones, el usuario puede realizar diversas acciones como llenar vectores manualmente o de manera aleatoria, sumarlos, imprimirlos y utilizar sus datos para llenar una matriz. Estas funcionalidades ofrecen una introducción práctica al trabajo con arreglos en C, lo que facilita la manipulación y visualización de datos estructurados en forma de vectores y matrices. Además, el código ilustra el uso de funciones, estructuras de control y generación de números aleatorios en C.

REFERENCIAS

https://drive.google.com/drive/folders/13q-7WNvxi_TC9ssK-Cqozzt9xKToSkI?usp=sharing

GIT HUB

https://github.com/sergiofabian05/SFA_PE_ACT8.pe.git