Universidad Autónoma de Baja California

Ingeniería en Software y tecnologías emergentes

Materia: Programación Estructurada /Clave 36276

Alumno: Pedro Carlos Balderrama Campos

Matricula: 372267

Maestro: Pedro Núñez Yépiz

Actividad No. : 12

Tema-Unidad : Librerías en c, métodos de ordenación y búsqueda

1.- Introducción

El presente reporte de práctica presenta el desarrollo de un programa en lenguaje C que ofrece diversas funcionalidades a través de un menú interactivo. Este programa permite al usuario llevar un registro detallado de una base de datos de alumnos, la base de datos contempla datos como, la matricula, el nombre, apellidos y la edad. Como llave primaria se utilizara la matricula.

2.- Competencia

La competencia propuesta se centra en el refinamiento de habilidades en el manejo de cadenas de texto dentro del contexto del aprendizaje de programación. A través de una serie de ejercicios prácticos, los participantes tenemos la oportunidad de adquirir destrezas en la manipulación y operaciones con cadenas y estructuras, permitiendo desarrollar la capacidad de trabajar con ellas de manera efectiva y lograr los resultados deseados.

3.- Fundamentos

La habilidad para mover datos entre diferentes estructuras destaca lo importante que son las operaciones de control y la lógica en la creación de programas que sean eficientes y flexibles. Este enfoque práctico en estructuras de datos básicas como vectores y matrices nos ayuda a entender conceptos clave de informática, como recorrer colecciones de datos, generar números aleatorios de manera ordenada y organizar datos para trabajar con ellos.

4.- Procedimiento

El desarrollo se divide en funciones específicas, cada una encargada de una tarea particular, como la generación de números aleatorios para generar nombres, así como todos los otros datos de cada una de las personas registradas, el poner manualmente los datos de algún alumno, el ordenar los vectores, etc. Cada función es meticulosamente documentada y probada para garantizar su correcto funcionamiento. Este enfoque facilita la comprensión y el mantenimiento del código, al mismo tiempo que fortalece el aprendizaje de conceptos básicos en programación y manipulación de datos.

5.- Resultados y conclusiones

Gracias a que se reutilizo gran parte del código de practicas anteriores se pudo hacer con mucha mas facilidad, usando esta modularidad se pudo generar cada uno de los registros de forma automática y rápida.

```
void OpcionCincoOrdenar(Tdatos Arreglo[])
{
    int MetodoInt, CicloOrdenar = 1, Intercambiado, IndiceMinimo, n, Temporal, i,
j, CantidadDeDatos = CantidadDeDatosActuales(Arreglo);
    char MetodoChar[3];
    do
    {
        clear();
        printf("Selecciona que metodo quieres utlizar para ordenar el vector:
\n");
        printf("[1] Metodo de la burbuja\n");
        printf("[2] Metodo de la burbuja mejorada\n");
        printf("[3] Metodo de ordenacion por seleccion\n");
        fflush(stdin);
        gets(MetodoChar);
        MetodoInt = atoi(MetodoChar);

        switch (MetodoInt)
        {
        case 1:
            for (i = 0; i < CantidadDeDatos - 1; i++)
            {
                for (j = 0; j < CantidadDeDatos - i - 1; j++)
                {
                    if (Arreglo[j].Matricula > Arreglo[j + 1].Matricula)
                    {
                        Temporal = Arreglo[j].Matricula;
                        Arreglo[j].Matricula = Arreglo[j + 1].Matricula;
                        Arreglo[j + 1].Matricula = Temporal;
                    }
                }
            }
            printf("\nEl arreglo ahora se encuentra ordenado!");
            pause();
            CicloOrdenar = 0;
            break;
        case 2:
            for (i = 0; i < CantidadDeDatos - 1; i++)
            {
                Intercambiado = 0;
```

```
                for (j = 0; j < CantidadDeDatos - i - 1; j++)
                {
                    if (Arreglo[j].Matricula > Arreglo[j + 1].Matricula)
                    {
                        Temporal = Arreglo[j].Matricula;
                        Arreglo[j].Matricula = Arreglo[j + 1].Matricula;
                        Arreglo[j + 1].Matricula = Temporal;
                        Intercambiado = 1;
                    }
                }
            }
            printf("\nEl arreglo ahora se encuentra ordenado!");
            pause();
            CicloOrdenar = 0;
            break;
        case 3:
            for (i = 0; i < n - 1; i++)
            {
                IndiceMinimo = i;
                for (j = i + 1; j < n; j++)
                {
                    if (Arreglo[j].Matricula < Arreglo[IndiceMinimo].Matricula)
                        IndiceMinimo = j;
                }
                Temporal = Arreglo[IndiceMinimo].Matricula;
                Arreglo[IndiceMinimo].Matricula = Arreglo[i].Matricula;
                Arreglo[i].Matricula = Temporal;
            }
            printf("\nEl arreglo ahora se encuentra ordenado!");
            pause();
            CicloOrdenar = 0;
            break;
        default:
            printf("No seleccionaste una opcion valida, por favor introduce
otra");
            pauseclear();
            break;
        }
    } while (CicloOrdenar);
}
```

6.- 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: 8448130138