main.cpp

```cpp
#include "Libreria.h"
int main()
{
    srand(time(NULL));
    Tdatos Arreglo[CantidadMaximaDeRegistros];
    menu(Arreglo);
    return 0;
}
```

pedrolib.h

```cpp
// PCBC_ACT12_01.cpp
// Pedro Carlos Balderrama Campos
// 5 de mayo del 2024
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
//
#define CantidadMaximaDeRegistros 1500
#define LlenadoAutomatico 10
#define RegistrosAImprimir 17
#define MAX_RUTA 200
// Struct
typedef struct StructDatos
{
    int Estatus;
    int Matricula;
    char Nombre[13];
    char ApellidoPaterno[13];
    char ApellidoMaterno[13];
    int Edad;
    int Genero;
} Tdatos;
// Listas
char ListaNombresHombre[21][12] = {"Pedro", "Carlos", "Diego", "Alan",
"Emiliano", "Alejandro", "Raul", "Hector",
                                   "Francisco", "Gabriel", "Maximo", "Alonso",
"Fernando", "Rogelio", "Ariel", "Juan",
                                   "Erick", "Christian", "Rodrigo", "Joel"};
char ListaNombresMujer[21][12] = {"Dayanara", "Karla", "Ashley", "Aurora",
"Jessica", "Alejandra", "Ramona", "Ermelina",
                                  "Francisca", "Gabriela", "Marissa", "Josefina",
"Fernanda", "Roberta", "Esperanza",
                                  "Erika", "Linda", "Cristina", "Dulce",
"Jazmin"};
char ListaApellidos[21][12] = {"Balderrama", "Campos", "Gartner", "Duran",
"Martinez", "Hernandez", "Ramirez", "Benitez",
```

```
                                "Valenzuela", "Rocha", "Ramos", "Migoni", "Yepiz",
"Garza", "Prieto",
                                "Leyva", "Gonzalez", "Corona", "Zamarripa",
"Solorzano"};
//
// Prototipos
void menu(Tdatos Arreglo[]);
//
void OpcionUnoCargarArchivo(Tdatos Arreglo[]);
void OpcionDosAgregar(Tdatos Arreglo[]);
void OpcionTresEliminar(Tdatos Arreglo[]);
void OpcionCuatroBuscar(Tdatos Arreglo[]);
void OpcionCincoOrdenar(Tdatos Arreglo[]);
void OpcionSeisMostrarTodo(Tdatos Arreglo[]);
void OpcionSieteGenerarArchivo(Tdatos Arreglo[]);
//
int BuscarSiExiste(Tdatos Arreglo[], int Matricula);
int BuscarIndiceDeMatricula(Tdatos Arreglo[], int Matricula);
int CantidadDeDatosActuales(Tdatos Arreglo[]);
int MatriculaRepetida(Tdatos Arreglo[], int IndiceAgregar, int
MatriculaTemporal);
void ImprimirTablaDeRegistros(Tdatos Arreglo[]);
//
void ImprimirMenuPrincipal();
void EncabezadoDeLista();
//
void clear();
void pause();
void pauseclear();
//

// Desarrollo de funciones
void menu(Tdatos Arreglo[])
{
    int EstadoArchivoCargado = 0, MenuPrincipalActivo = 1, OpcionInt,
EstadoOrdenado = 0;
    char OpcionChar[2];
    do
    {
        clear();
        ImprimirMenuPrincipal();
        if (EstadoArchivoCargado == 1 || CantidadDeDatosActuales(Arreglo) > 0)
        {
            printf("\nHay una cantidad de %d datos en el arreglo\n",
CantidadDeDatosActuales(Arreglo));
            printf("El registro se encuentra: ");
            switch (EstadoOrdenado)
            {
```

```
            case 0:
                printf("desordenado");
                break;
            case 1:
                printf("ordenado");
                break;
            }
            printf("\n");
        }
        gets(OpcionChar);
        OpcionInt = atoi(OpcionChar);

        switch (OpcionInt)
        {
        case 1:
            if (EstadoArchivoCargado == 1)
            {
                printf("Solo puedes cargar el archivo una vez\n");
            }
            else
            {
                OpcionUnoCargarArchivo(Arreglo);
                EstadoArchivoCargado++;
            }
            break;
        case 2:
            OpcionDosAgregar(Arreglo);
            EstadoOrdenado = 0;
            break;
        case 3:
            OpcionTresEliminar(Arreglo);
            break;
        case 4:
            OpcionCuatroBuscar(Arreglo);
            break;
        case 5:
            if (EstadoOrdenado == 0)
            {
                OpcionCincoOrdenar(Arreglo);
            }
            else
            {
                clear();
                printf("El arreglo ya se encuentra ordenado");
                pauseclear();
            }
            EstadoOrdenado = 1;
            break;
```

```
        case 6:
            OpcionSeisMostrarTodo(Arreglo);
            break;
        case 7:
            OpcionSieteGenerarArchivo(Arreglo);
            break;
        case 0:
            exit(NULL);
            break;
        default:
            printf("No seleccionaste una opcion valida\n");
            pauseclear();
            break;
        }
    } while (MenuPrincipalActivo);
}
// Opciones del menu
void OpcionUnoCargarArchivo(Tdatos Arreglo[])
{
    char NombreDelArchivo[50];
    char ruta[MAX_RUTA];
    int CicloDeCargaActivo = 1;
    do
    {
        printf("Escribe el nombre del archivo que quieres cargar: (escribir sin
extension)\n");
        fflush(stdin);
        fgets(NombreDelArchivo, sizeof(NombreDelArchivo), stdin);
        NombreDelArchivo[strcspn(NombreDelArchivo, "\n")] = '\0';

        snprintf(ruta, sizeof(ruta), "/Users/pedrocbcmp/Desktop/Tercer
semestre/Programacion estructurada/Actividad 12/%s.txt", NombreDelArchivo);

        FILE *archivo = fopen(ruta, "r");
        if (archivo == NULL)
        {
            printf("No se encontró ningún archivo con ese nombre\n");
            pauseclear();
            return;
        }
        else
        {
            int i = 0;
            for (int i = 0; i < CantidadMaximaDeRegistros; i++)
            {
                fscanf(archivo, "%d %d %s %s %s %d %d",
                       &Arreglo[i].Estatus,
                       &Arreglo[i].Matricula,
```

```
                        Arreglo[i].Nombre,
                        Arreglo[i].ApellidoPaterno,
                        Arreglo[i].ApellidoMaterno,
                        &Arreglo[i].Edad,
                        &Arreglo[i].Genero);
            }
            fclose(archivo);
            CicloDeCargaActivo = 0;
        }
    } while (CicloDeCargaActivo);
}
void OpcionDosAgregar(Tdatos Arreglo[])
{
    clear();
    int CantidadAutomatica = LlenadoAutomatico, MatriculaTemporal,
VerificadorDeMatricula = 0, IndiceAgregar, CantidadDelCiclo;
    char DiaTemporal[3];
    // Establecer desde donde seguiremos llenando los datos
    IndiceAgregar = CantidadDeDatosActuales(Arreglo);
    //

    // Verificar si el arreglo está lleno y cuidar desbordamiento
    if (IndiceAgregar >= CantidadMaximaDeRegistros - 1)
    {
        printf("El arreglo está completamente lleno. No se pueden agregar más
datos.");
        pause();
        return;
    }
    for (int j = 0; j < LlenadoAutomatico; j++)
    {
        if (CantidadAutomatica + IndiceAgregar >= CantidadMaximaDeRegistros)
        {
            CantidadAutomatica--;
        }
        else
        {
            j = LlenadoAutomatico;
        }
    }
    if (CantidadAutomatica != LlenadoAutomatico)
    {
        printf("Para evitar el desbordamiento de buffer, la cantidad automatica
de datos a agregar fue cambiada de %d a %d", LlenadoAutomatico,
CantidadAutomatica);
        pause();
    }
    //
```

```
    CantidadDelCiclo = CantidadAutomatica + IndiceAgregar;
    if (CantidadAutomatica != 0)
    {
        printf("Lista de datos agregados al arreglo: \n\n");
        for (int i = IndiceAgregar; i < CantidadDelCiclo; i++, IndiceAgregar++)
        {
            // Asignar estatus activo y genero, nombre, apellidos, edad y
matricula aleatorios
            Arreglo[IndiceAgregar].Estatus = 1;
            Arreglo[IndiceAgregar].Edad = (rand() % 30) + 18; // Entre 1 y 12
            Arreglo[IndiceAgregar].Genero = rand() % 2;
            if (Arreglo[IndiceAgregar].Genero == 0)
            {
                strcpy(Arreglo[IndiceAgregar].Nombre, ListaNombresHombre[rand() %
20]);
            }
            if (Arreglo[IndiceAgregar].Genero == 1)
            {
                strcpy(Arreglo[IndiceAgregar].Nombre, ListaNombresMujer[rand() %
20]);
            }
            strcpy(Arreglo[IndiceAgregar].ApellidoPaterno, ListaApellidos[rand()
% 20]);
            strcpy(Arreglo[IndiceAgregar].ApellidoMaterno, ListaApellidos[rand()
% 20]);
            Arreglo[IndiceAgregar].Edad = (rand() % 30) + 18;
            do
            {
                MatriculaTemporal = rand() % 100000 + 300000; // entre 300,000 y
399,999;
                VerificadorDeMatricula = MatriculaRepetida(Arreglo,
IndiceAgregar, MatriculaTemporal);
            } while (VerificadorDeMatricula == 0);
            Arreglo[IndiceAgregar].Matricula = MatriculaTemporal;

            printf("Indice: %d\n Estatus: %d\n", IndiceAgregar,
Arreglo[IndiceAgregar].Estatus);
            printf(" Genero: ");
            switch (Arreglo[IndiceAgregar].Genero)
            {
            case 0:
                printf("hombre\n");
                break;
            case 1:
                printf("mujer\n");
                break;
            }
```

```
            printf(" Nombre: %s\n Apellido paterno: %s\n Apellido materno: %s\n
Edad: %d\n Matricula: %d\n\n", Arreglo[IndiceAgregar].Nombre,
Arreglo[IndiceAgregar].ApellidoPaterno, Arreglo[IndiceAgregar].ApellidoMaterno,
Arreglo[IndiceAgregar].Edad, Arreglo[IndiceAgregar].Matricula);
        }
        printf("Cantidad de datos agregados: %d\n", CantidadAutomatica);
        pause();
    }
}
void OpcionTresEliminar(Tdatos Arreglo[])
{
    char MatriculaABorrarChar[7];
    int MatriculaABorrarint, MatriculaExistente, IndiceMatricula;
    clear();
    printf("Escribe la matricula de los datos que quieres eliminar: ");
    gets(MatriculaABorrarChar);
    MatriculaABorrarint = atoi(MatriculaABorrarChar);
    MatriculaExistente = BuscarSiExiste(Arreglo, MatriculaABorrarint);
    if (MatriculaExistente == 1)
    {
        int MenuConfirmar = 1, OpcionInt;
        char OpcionChar[2];
        do
        {
            printf("Estas seguro de que deseas eliminar los datos de la
matricula #%d\n", MatriculaABorrarint);
            printf("[1] SI, quiero eliminar los datos\n");
            printf("[2] NO, quiero mantener los datos\n");
            fflush(stdin);
            gets(OpcionChar);
            OpcionInt = atoi(OpcionChar);
            if (OpcionInt == 1 || OpcionInt == 2)
            {
                switch (OpcionInt)
                {
                case 1:
                    IndiceMatricula = BuscarIndiceDeMatricula(Arreglo,
MatriculaABorrarint);
                    Arreglo[IndiceMatricula].Estatus = 0;
                    printf("Matricula eliminada con exito\n");
                    pauseclear();
                    return;
                    break;
                case 2:
                    printf("Escogiste no eliminar los datos\n");
                    pauseclear();
                    return;
                    break;
```

```
                }
            }
            else
            {
                printf("No seleccionaste una opcion correcta\n");
                pauseclear();
            }
        } while (MenuConfirmar);
    }
}
void OpcionCuatroBuscar(Tdatos Arreglo[])
{
    int MatriculaExiste = 0, MatriculaBuscarInt, IndiceMatricula;
    char MatriculaBuscarChar[7];
    clear();
    printf("Escribe la matricula que estas buscando: ");
    fflush(stdin);
    gets(MatriculaBuscarChar);
    MatriculaBuscarInt = atoi(MatriculaBuscarChar);
    MatriculaExiste = BuscarSiExiste(Arreglo, MatriculaBuscarInt);
    if (MatriculaExiste == 1)
    {
        if (Arreglo[IndiceMatricula].Estatus == 1)
        {
            IndiceMatricula = BuscarIndiceDeMatricula(Arreglo,
MatriculaBuscarInt);
            printf("Matricula: %d \n", Arreglo[IndiceMatricula].Matricula);
            printf("Nombre: %s \n", Arreglo[IndiceMatricula].Nombre);
            printf("Apellido paterno: %s \n",
Arreglo[IndiceMatricula].ApellidoPaterno);
            printf("Apellido materno: %s \n",
Arreglo[IndiceMatricula].ApellidoMaterno);
            printf("Edad: %d\n", Arreglo[IndiceMatricula].Edad);
            printf("Genero: ");
            switch (Arreglo[IndiceMatricula].Genero)
            {
            case 0:
                printf("hombre\n");
                break;
            case 1:
                printf("mujer\n");
                break;
            }
        }
        pauseclear();
    }
    else
    {
```

```
        printf("Esa matricula no existe o ha sido eliminada");
        pauseclear();
    }
}
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);
}
void OpcionSeisMostrarTodo(Tdatos Arreglo[])
{
    clear();
    printf("Has escogido imprimir los registros, se imprimiran en tablas de %d
datos", RegistrosAImprimir);
    pauseclear();
    ImprimirTablaDeRegistros(Arreglo);
    pause();
}
void OpcionSieteGenerarArchivo(Tdatos Arreglo[])
{
    char NombreDelArchivo[30];
```

```
    char ruta[MAX_RUTA];
    clear();
    printf("Escribe el nombre con el que quieres guardar el archivo (la extension
del archivo se escribira automaticamente como .txt)\n");
    fflush(stdin);
    gets(NombreDelArchivo);
    snprintf(ruta, sizeof(ruta), "/Users/pedrocbcmp/Desktop/Tercer
semestre/Programacion estructurada/Actividad 12/%s.txt", NombreDelArchivo);
    FILE *archivo = fopen(ruta, "w");

    if (archivo == NULL)
    {
        printf("No se pudo crear el archivo\n");
        pause();
        return;
    }
    else
    {
        fprintf(archivo, "%s %s %s %s %s %s %s \n", "S", "Matricula", "Nombre",
"ApellidoPaterno", "ApellidoMaterno", "Edad", "Sexo");
        for (int i = 0; i < CantidadDeDatosActuales(Arreglo); i++)
        {
            if (Arreglo[i].Estatus == 1)
            {
                fprintf(archivo, "%d %d %s %s %s %d",
                        Arreglo[i].Estatus,
                        Arreglo[i].Matricula,
                        Arreglo[i].Nombre,
                        Arreglo[i].ApellidoPaterno,
                        Arreglo[i].ApellidoMaterno,
                        Arreglo[i].Edad);
                switch (Arreglo[i].Genero == 0)
                {
                case 1:
                    fprintf(archivo, " Hombre\n");
                    break;

                case 0:
                    fprintf(archivo, " Mujer\n");
                    break;
                }
            }
            else
            {
                fprintf(archivo, "%d %d Los datos relacionados con esta matricula
han sido eliminados",
                        Arreglo[i].Estatus,
                        Arreglo[i].Matricula);
```

```
            }
        }
        printf("Archivo %s.txt guardado con exito!", NombreDelArchivo);
        pause();
    }
    fclose(archivo);
}
//
void ImprimirMenuPrincipal()
{
    printf("[1] Cargar Archivo\n");
    printf("[2] Agregar automaticamente %d datos\n", LlenadoAutomatico);
    printf("[3] Eliminar registro\n");
    printf("[4] Buscar matricula\n");
    printf("[5] Ordenar arreglo\n");
    printf("[6] Mostrar todo el arreglo\n");
    printf("[7] Generar archivo\n\n");
    printf("[0] Salir\n");
}
//
int BuscarSiExiste(Tdatos Arreglo[], int Matricula)
{
    for (int i = 0; i < CantidadMaximaDeRegistros; i++)
    {
        if (Arreglo[i].Matricula == Matricula)
        {
            return 1;
        }
    }
    return 0;
}
int BuscarIndiceDeMatricula(Tdatos Arreglo[], int Matricula)
{
    // Buscar indice de matricula
    int MatriculaExiste = BuscarSiExiste(Arreglo, Matricula);
    if (MatriculaExiste == 1)
    {
        for (int i = 0; i < CantidadMaximaDeRegistros; i++)
        {
            if (Arreglo[i].Matricula == Matricula)
            {
                return i;
            }
        }
    }
    return 0;
}
int CantidadDeDatosActuales(Tdatos Arreglo[])
```

```
{
    // Esta funcion devuelve la cantidad de datos actuales dentro del arreglo
    for (int i = 0; i < CantidadMaximaDeRegistros; i++)
    {
        if (Arreglo[i].Matricula == 0)
        {
            return i;
        }
    }
}
int MatriculaRepetida(Tdatos Arreglo[], int IndiceAgregar, int MatriculaTemporal)
{
    // 0 Matricula repetida
    // 1 Matricula no repetida
    for (int i = 0; i < IndiceAgregar; i++)
    {
        if (MatriculaTemporal == Arreglo[i].Matricula)
        {
            return 0;
        }
    }
    return 1;
}
void ImprimirTablaDeRegistros(Tdatos Arreglo[])
{
    char ContinuarChar[2], CantidadActual = CantidadDeDatosActuales(Arreglo);
    int ContinuarInt, Validador = 0;
    EncabezadoDeLista();
    for (int i = 0; i < CantidadActual; i++)
    {
        if (Arreglo[i].Estatus == 1)
        {
            printf("[%4d] | ", i);
            printf("%6d | ", Arreglo[i].Matricula);
            printf("%s %s %-12s ", Arreglo[i].Nombre, Arreglo[i].ApellidoPaterno,
Arreglo[i].ApellidoMaterno);
            printf("%4d ", Arreglo[i].Edad);
            switch (Arreglo[i].Genero)
            {
            case 0:
                printf("%-6s", "Hombre");
                break;
            case 1:
                printf("%-6s", "Mujer");
                break;
            }
            printf("\n");
        }
```

```
        if (Arreglo[i].Estatus == 0)
        {
            printf("[%4d] | ", i);
            printf("%d | ", Arreglo[i].Matricula);
            printf("Los datos existian pero han sido eliminados\n");
        }
        if (i % RegistrosAImprimir == 0 && i != 0)
        {
            do
            {
                printf("\nQuieres continuar imprimiendo datos?\n");
                printf("[0] Dejar de imprimir\n");
                printf("[1] Continuar imprimiendo\n");
                fflush(stdin);
                gets(ContinuarChar);
                ContinuarInt = atoi(ContinuarChar);
                if (ContinuarInt == 0 || ContinuarInt == 1)
                {
                    if (ContinuarInt == 0)
                    {
                        i = CantidadActual;
                        Validador++;
                    }
                    if (ContinuarInt == 1)
                    {
                        Validador++;
                        pause();
                    }
                }
                else
                {
                    printf("No escogiste una opcion valida\n");
                    pause();
                    clear();
                }

            } while (Validador == 0);
            clear();
            EncabezadoDeLista();
        }
    }
}
//
void EncabezadoDeLista()
{
    printf(" #   Matricula    Nombre   Apellidos        Edad        Genero
\n");
}
```

```c
//
void clear()
{
    system("clear");
}
void pause()
{
    printf("\nPresiona Enter para continuar...");
    fflush(stdin);
    getchar();
}
void pauseclear()
{
    pause();
    clear();
}
```