main.cpp

```
#include "pedrolib.c"

int main()
{
    srand(time(NULL));
    clear();
    menu();
    return 0;
}
```

pedrolib.c

```
#include "pedrolib.h"
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
char listanombreshombre[20][15] = {"Pedro", "Carlos", "Diego", "Alan",
"Emiliano", "Alejandro", "Raul", "Hector",
                                   "Francisco", "Gabriel", "Maximo", "Alonso",
"Fernando", "Rogelio", "Ariel", "Jose",
                                   "Erick", "Christian", "Rodrigo", "Joel"};
char listanombresmujer[20][15] = {"Dayanara", "Karla", "Ashley", "Aurora",
"Jessica", "Alejandra", "Ramona", "Ermelina",
                                  "Francisca", "Gabriela", "Maria", "Josefina",
"Fernanda", "Roberta", "Esperanza",
                                  "Erika", "Linda", "Cristina", "Dulce",
"Jazmin"};
char listaapellidos[20][15] = {"Balderrama", "Campos", "Gartner", "Duran",
"Martinez", "Hernandez", "Ramirez", "Benitez",
                               "Valenzuela", "Rocha", "Ramos", "Migoni", "Yepiz",
"De la garza", "Prieto",
                               "Leyva", "Gonzalez", "Corona", "Zamarripa",
"Solorzano"};

typedef struct _salumnos
{
    bool status;
    int matricula;
    char appat[20];
    char apmat[20];
    char nombres[25];
    int edad;
    int sexo;
} Talumnos;

Talumnos vect[tamanodelvector];

void clear()
{
```

```
    system("clear");
}
void pause()
{
    printf("\nPresiona Enter para continuar...");
    getchar();
}
void menu()
{
    int op, salida = 0, contadordedatos = 0;
    char opchar[2];

    do
    {
        clear();
        opcionesmenu();
        fflush(stdin);
        gets(opchar);

        op = atoi(opchar);

        switch (op)
        {
        case 1:
            agregadoautomatico(&contadordedatos);
            break;
        case 2:
            agregadomanual(&contadordedatos);
            break;
        case 3:
            eliminarregistro(&contadordedatos);
            break;
        case 4:
            buscarregistro(contadordedatos);
            break;
        case 5:
            ordenarregistros(contadordedatos);
            break;
        case 6:
            imprimirregistros(contadordedatos);
            break;
        case 0:
            salida++;
            break;
        default:
            printf("No seleccionaste una opcion valida.\n");
            pause();
            break;
```

```
        }
    } while (salida == 0);
}
void opcionesmenu()
{
    printf("\n");
    printf("[1] Agregar automaticamente %d registros \n", cantidadautomatica);
    printf("[2] Agregar manualmente 1 registro \n");
    printf("[3] Eliminar manualmente 1 registro \n");
    printf("[4] Buscar y mostrar un registro \n");
    printf("[5] Ordenar registros por matrícula \n");
    printf("[6] Imprimir todos los registros \n");
    printf("\n");
    printf("[0] Salir \n");
    printf("\n");
}
void copiarcadena(char destino[], const char origen[])
{
    int i = 0;
    while (origen[i] != '\0')
    {
        destino[i] = origen[i];
        i++;
    }
    destino[i] = '\0';
}

void agregadomanual(int *contadordedatos)
{
    clear();

    bool statustemp = true;
    int matriculatempman, edadtemp, validadordeedad = 0, validadordesexo = 0,
sexotemp;
    char nombretempman[25], appattemp[20], apmattemp[20], edadtempchar[2],
sexotempchar[2];

    printf("\nHas elegido agregar manualmente un registro\n");

    matriculatempman = rand() % 100000 + 300000; // entre 300,000 y 399,999
    printf("La matricula que se ha escogido automaticamente es: %d\n",
matriculatempman);

    printf("Escribe el nombre o nombres del estudiante: \n");
    fflush(stdin);
    gets(nombretempman);

    printf("Escribe el apellido paterno del estudiante: \n");
```

```
    fflush(stdin);
    gets(appattemp);

    printf("Escribe el apellido materno del estudiante: \n");
    fflush(stdin);
    gets(apmattemp);

    do
    {
        printf("Escribe la edad del estudiante: \n");

        fflush(stdin);
        gets(edadtempchar);
        edadtemp = atoi(edadtempchar);

        if (edadtemp != 0)
        {
            validadordeedad++;
        }
        else
        {
            printf("\nNo escribiste una edad correcta, vuelve a introducir
una\n");
        }
    } while (validadordeedad == 0);

    do
    {
        printf("El estudiante es hombre o mujer? \n");
        printf("[1] Hombre\n");
        printf("[2] Mujer\n");

        fflush(stdin);
        gets(sexotempchar);
        sexotemp = atoi(sexotempchar);

        if (sexotemp == 1 || sexotemp == 2)
        {
            validadordesexo++;
        }
        else
        {
            printf("\nNo escribiste una opcion correcta, vuelve a introducir
una\n");
        }
    } while (validadordesexo == 0);

    vect[*contadordedatos].status = statustemp;
```

```
    vect[*contadordedatos].matricula = matriculatempman;
    copiarcadena(vect[*contadordedatos].nombres, nombretempman);
    copiarcadena(vect[*contadordedatos].appat, appattemp);
    copiarcadena(vect[*contadordedatos].apmat, apmattemp);
    vect[*contadordedatos].edad = edadtemp;
    vect[*contadordedatos].sexo = sexotemp;

    printf("\nAgregado de registro con exito!\n Se agrego en la posicion [%d]\n",
*contadordedatos);
    pause();

    (*contadordedatos)++;
}
void agregadoautomatico(int *contadordedatos)
{
    clear();
    bool statustemp = true;
    int nombrealeatorio, primerapellidorandom, segundoapellidorandom,
listadematriculasagregadas[cantidadautomatica], matriculaaleatoria, sexorandom,
edadaleatoria;

    for (int i = 0; i < cantidadautomatica; i++)
    {
        matriculaaleatoria = rand() % 100000 + 300000; // entre 300,000 y 399,999
        sexorandom = rand() % 2;                       // del 0 al 1
        nombrealeatorio = rand() % 20;                 // del 0 al 19 (total 20
nombres)
        edadaleatoria = rand() % 51 + 18;              // de 18 a 68
        primerapellidorandom = rand() % 20;            // del 0 al 19 (total 20
apellidos)
        segundoapellidorandom = rand() % 20;           // del 0 al 19 (total 20
apellidos)

        vect[*contadordedatos].status = statustemp;           // LISTO
        vect[*contadordedatos].matricula = matriculaaleatoria; // LISTO
        if (sexorandom == 1)
        {
            copiarcadena(vect[*contadordedatos].nombres,
listanombreshombre[nombrealeatorio]);
        }
        else
        {
            copiarcadena(vect[*contadordedatos].nombres,
listanombresmujer[nombrealeatorio]);
        }
        copiarcadena(vect[*contadordedatos].appat,
listaapellidos[primerapellidorandom]);
```

```
        copiarcadena(vect[*contadordedatos].apmat,
listaapellidos[segundoapellidorandom]);

        vect[*contadordedatos].edad = edadaleatoria; // LISTO
        vect[*contadordedatos].sexo = sexorandom;    // LISTO

        listadematriculasagregadas[i] = matriculaaleatoria;

        (*contadordedatos)++;
    }
    printf("Las matriculas agregadas son: \n");
    for (int i = 0; i < cantidadautomatica; i++)
    {
        printf("[%d] %d\n", i, listadematriculasagregadas[i]);
    }
    pause();
}
void eliminarregistro(int *contadordedatos)
{
    int matricula;
    printf("Ingrese la matrícula del registro que desea eliminar: ");
    scanf("%d", &matricula);

    bool encontrado = false;
    for (int i = 0; i < *contadordedatos; i++)
    {
        if (vect[i].matricula == matricula)
        {
            encontrado = true;
            vect[i].status = false; // Marcar como inactivo
            printf("El registro con matrícula %d ha sido eliminado.\n",
matricula);
            break;
        }
    }

    if (!encontrado)
    {
        printf("No se encontró ningún registro con la matrícula
proporcionada.\n");
    }

    pause();
}
void buscarregistro(int contadordedatos)
{
    int buscadortemp = 0;
    char buscadortempchar[7];
```

```
    clear();
    printf("Escribe el numero de matricula que quieres buscar: ");
    fflush(stdin);
    gets(buscadortempchar);

    buscadortemp = atoi(buscadortempchar);

    for (int i = 0; i < contadordedatos; i++)
    {
        if (vect[i].matricula == buscadortemp)
        {
            clear();
            printf("La matricula si existe!\n");
            printf("Status de la persona: %s\n", vect[i].status ? "activo" : "no
activo");
            printf("Matricula de la persona: %d\n", vect[i].matricula);
            printf("Nombre de la persona: %s\n", vect[i].nombres);
            printf("Apellido paterno de la persona: %s\n", vect[i].appat);
            printf("Apellido materno de la persona: %s\n", vect[i].apmat);
            printf("Edad de la persona: %d\n", vect[i].edad);
            printf("Sexo de la persona: %s\n", vect[i].sexo == 1 ? "hombre" :
"mujer");
            pause();
            clear();
            return;
        }
    }
    printf("No se encontró ningún registro con la matrícula proporcionada.\n");
    pause();
}
void ordenarregistros(int contadordedatos)
{
    int i, j;
    Talumnos temp;

    for (i = 0; i < contadordedatos - 1; i++)
    {
        for (j = 0; j < contadordedatos - i - 1; j++)
        {
            if (vect[j].matricula > vect[j + 1].matricula)
            {
                // Intercambiar los registros
                temp = vect[j];
                vect[j] = vect[j + 1];
                vect[j + 1] = temp;
            }
        }
```

```
    }

    printf("Registros ordenados por matrícula.\n");
    pause();
}
void imprimirregistros(int contadordedatos)
{
    clear();
    for (int i = 0; i < contadordedatos; i++)
    {
        printf("Status de la persona: %s\n", vect[i].status ? "activo" : "no
activo");
        printf("Matricula de la persona: %d\n", vect[i].matricula);
        printf("Nombre de la persona: %s\n", vect[i].nombres);
        printf("Apellido paterno de la persona: %s\n", vect[i].appat);
        printf("Apellido materno de la persona: %s\n", vect[i].apmat);
        printf("Edad de la persona: %d\n", vect[i].edad);
        printf("Sexo de la persona: %s\n", vect[i].sexo == 1 ? "hombre" :
"mujer");
        printf("====================================\n");
    }
    pause();
}
```

pedrolib.h

```
#define tamanodelvector 500
#define cantidadautomatica 10

void clear();
void pause();
void menu();
void opcionesmenu();
void copiarcadena(char destino[], const char origen[]);

void agregadoautomatico(int *contadordedatos);
void agregadomanual(int *contadordedatos);
void eliminarregistro(int *contadordedatos);
void buscarregistro(int contadordedatos);
void ordenarregistros(int contadordedatos);
void imprimirregistros(int contadordedatos);
void imprimirunregistro();
```