main.cpp

```cpp
// Balderrama Campos Pedro Carlos
// Lunes 29 de abril del 2024
// PCBC_ACT11_01.cpp
#include "pedrolib.h"

// main
int main()
{
    srand(time(NULL));

    Tdatos Arreglo[TamanoDelArreglo];
    int MatriculasRegistradas[TamanoDelArreglo];
    int EstadoOrdenado = 0;

    menu(Arreglo, MatriculasRegistradas, EstadoOrdenado);
}
//
```

pedrolib.h

```cpp
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
//
#define TamanoDelArreglo 2000
#define LlenadoAutomatico 100
#define RegistrosAImprimir 15
//
int ContadorUniversal = 0;
//
// 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"};
//
// Structs anidados
typedef struct _snombre
{
    char Nombre[25];
    char ApellidoPaterno[20];
    char ApellidoMaterno[20];
} Tnombre;
typedef struct _sfecha
{
    int DiaDeNacimiento;
    int MesDeNacimiento;
    int AnioDeNacimiento;
} Tfecha;
// Struct del CURP
typedef struct _sdatos
{
    int Estatus;
    Tnombre NombreYApellidos;
    Tfecha FechaDeNacimiento;
    int Genero;
    int EstadoDeNacimiento;
    int Matricula;
    char CURP[20];
} Tdatos;
// Prototipos
void menu(Tdatos Arreglo[], int MatriculasRegistradas[], int EstadoOrdenado);
int OpcionUnoCargar(Tdatos Arreglo[], int MatriculasRegistradas[]);
void OpcionDosEliminar(Tdatos Arreglo[]);
void OpcionTresBuscar(Tdatos Arreglo[], int MatriculasRegistradas[]);
void OpcionCuatroOrdenar(Tdatos Arreglo[], int MatriculasRegistradas[], int
EstadoOrdenado);
void OpcionCincoImprimir(Tdatos Arreglo[], int MatriculasRegistradas[]);
void OpcionSeisGuardarDocumento(Tdatos Arreglo[], int MatriculasRegistradas[],
char nom[]);
//
void OrdenarMetodoDeBurbuja(Tdatos Arreglo[], int MatriculasRegistradas[]);
void ImprimirTablaDeRegistros(Tdatos Arreglo[], int MatriculasRegistradas[]);
int TamanoDeUnaCadena(char Cadena[]);
void LlenadoAutomaticoDeCurp(Tdatos Arreglo[], int AnioMenosSiglos);
void CuidarDesbordamiento(int CantidadAutomatica);
int MatriculaRepetida(int MatriculaTemporal, int MatriculasRegistradas[]);
void ImprimirListaDeMatriculasAgregadas(int CantidadAutomatica, int
MatriculasRegistradas[]);
void MatriculaExistente(Tdatos Arreglo[], int i);
//
```

```
void JuntarNombre(char x[], char y[], char z[], char NombreCompleto[]);
char LetraMayuscula(char c);
char PrimeraConsonante(char Palabra[]);
char PrimeraVocal(char Palabra[]);
//
void EncabezadoDeLista();
void ImprimirMenu();
//
void clear();
void pause();
//
// Funciones del menu, opciones y funciones de apoyo
void menu(Tdatos Arreglo[], int MatriculasRegistradas[], int EstadoOrdenado)
{
    int OpcionDelMenuInt;
    do
    {
        clear();
        ImprimirMenu();
        printf("\n");
        printf("Hay una cantidad de %d datos registrados\n", ContadorUniversal);
        printf("El arreglo se encuentra: ");
        switch (EstadoOrdenado)
        {
        case 0:
            printf("Vacio");
            break;
        case 1:
            printf("Desordenado");
            break;

        case 2:
            printf("Ordenado");
            break;
        }
        printf("\n\nSelecciona una opcion: ");
        scanf("%d", &OpcionDelMenuInt);
        switch (OpcionDelMenuInt)
        {
        case 1:
            OpcionUnoCargar(Arreglo, MatriculasRegistradas);
            EstadoOrdenado = 1;
            break;
        case 2:
            OpcionDosEliminar(Arreglo);
            break;
        case 3:
            OpcionTresBuscar(Arreglo, MatriculasRegistradas);
```

```
            break;
        case 4:
            OpcionCuatroOrdenar(Arreglo, MatriculasRegistradas, EstadoOrdenado);
            if (EstadoOrdenado != 0)
            {
                EstadoOrdenado = 2;
            }
            break;
        case 5:
            OpcionCincoImprimir(Arreglo, MatriculasRegistradas);
            break;
        case 6:
            clear();
            printf("Funcion aun no implementada\n");
            pause();
            /*
            char NombreDelDocumento[15];
            printf("Escribe el nombre que quieres que tenga el documento: ");
            fflush(stdin);
            gets(NombreDelDocumento);
            OpcionSeisGuardarDocumento(Arreglo, MatriculasRegistradas,
NombreDelDocumento);
            */
            break;
        case 0:
            printf("Has escogido salir del programa. Hasta pronto");
        default:
            clear();
            break;
        }
    } while (OpcionDelMenuInt != 0);
}
int OpcionUnoCargar(Tdatos Arreglo[], int MatriculasRegistradas[])
{
    clear();
    int EstatusTemporal = 1;
    int ValorAleatorioGenero, CantidadAutomatica = LlenadoAutomatico,
MatriculaTemporal, VerificadorDeMatricula = 0, AnioMenosSiglos;
    char DiaTemporal[3];

    // Cuidar Desbordamiento
    CuidarDesbordamiento(CantidadAutomatica);
    //

    if (CantidadAutomatica > 0)
    {
        for (int i = 0; i < CantidadAutomatica; i++)
        {
```

```
            // Verificar si el arreglo está lleno
            if (ContadorUniversal >= TamanoDelArreglo)
            {
                printf("El arreglo está completamente lleno. No se pueden agregar
más datos.");
                pause();
                return 0;
            }
            //

            // Asignar estatus EXISTENTE
            Arreglo[ContadorUniversal].Estatus = 1;
            //

            // Asignar matricula y verificar que no este repetida
            do
            {
                MatriculaTemporal = rand() % 100000 + 300000; // entre 300,000 y
399,999;
                VerificadorDeMatricula = MatriculaRepetida(MatriculaTemporal,
MatriculasRegistradas);
            } while (VerificadorDeMatricula == 0);
            Arreglo[ContadorUniversal].Matricula = MatriculaTemporal;
            MatriculasRegistradas[ContadorUniversal] = MatriculaTemporal;
            //

            // Asignar genero y nombre
            ValorAleatorioGenero = rand() % 2;
            Arreglo[ContadorUniversal].Genero = ValorAleatorioGenero;
            if (ValorAleatorioGenero == 0)
            {
                strcpy(Arreglo[ContadorUniversal].NombreYApellidos.Nombre,
ListaNombresHombre[rand() % 20]);
            }
            if (ValorAleatorioGenero == 1)
            {
                strcpy(Arreglo[ContadorUniversal].NombreYApellidos.Nombre,
ListaNombresMujer[rand() % 20]);
            }
            //

            // Asignar apellidos
            strcpy(Arreglo[ContadorUniversal].NombreYApellidos.ApellidoPaterno,
ListaApellidos[rand() % 20]);
            strcpy(Arreglo[ContadorUniversal].NombreYApellidos.ApellidoMaterno,
ListaApellidos[rand() % 20]);
            //
```

```
            // Asignar fecha de nacimiento
            Arreglo[ContadorUniversal].FechaDeNacimiento.MesDeNacimiento =
(rand() % 12) + 1;     // Entre 1 y 12
            Arreglo[ContadorUniversal].FechaDeNacimiento.AnioDeNacimiento =
(rand() % 45) + 1980; // Entre 1980 y 2024
            if (Arreglo[ContadorUniversal].FechaDeNacimiento.MesDeNacimiento ==
2)
            {
                if
((Arreglo[ContadorUniversal].FechaDeNacimiento.AnioDeNacimiento % 4) == 0)
                {
                    Arreglo[ContadorUniversal].FechaDeNacimiento.DiaDeNacimiento
= (rand() % 29) + 1;
                }
                else
                {
                    Arreglo[ContadorUniversal].FechaDeNacimiento.DiaDeNacimiento
= (rand() % 28) + 1;
                }
            }
            if (Arreglo[ContadorUniversal].FechaDeNacimiento.MesDeNacimiento == 1
|| Arreglo[ContadorUniversal].FechaDeNacimiento.MesDeNacimiento == 3 ||
Arreglo[ContadorUniversal].FechaDeNacimiento.MesDeNacimiento == 5 ||
Arreglo[ContadorUniversal].FechaDeNacimiento.MesDeNacimiento == 7 ||
Arreglo[ContadorUniversal].FechaDeNacimiento.MesDeNacimiento == 8 ||
Arreglo[ContadorUniversal].FechaDeNacimiento.MesDeNacimiento == 10 ||
Arreglo[ContadorUniversal].FechaDeNacimiento.MesDeNacimiento == 12)
            {
                Arreglo[ContadorUniversal].FechaDeNacimiento.DiaDeNacimiento =
(rand() % 30) + 1;
            }
            if (Arreglo[ContadorUniversal].FechaDeNacimiento.MesDeNacimiento == 4
|| Arreglo[ContadorUniversal].FechaDeNacimiento.MesDeNacimiento == 6 ||
Arreglo[ContadorUniversal].FechaDeNacimiento.MesDeNacimiento == 9 ||
Arreglo[ContadorUniversal].FechaDeNacimiento.MesDeNacimiento == 11)
            {
                Arreglo[ContadorUniversal].FechaDeNacimiento.DiaDeNacimiento =
(rand() % 30) + 1;
            }
            //

            // Asignar estado de nacimiento
            Arreglo[ContadorUniversal].EstadoDeNacimiento = (rand() % 33) + 1;
            //

            // Llenar CURP
            LlenadoAutomaticoDeCurp(Arreglo, AnioMenosSiglos);
            //
```

```
            ContadorUniversal++;
        }
        ImprimirListaDeMatriculasAgregadas(CantidadAutomatica,
MatriculasRegistradas);
    }
    printf("\n");
    pause();
}
void OpcionDosEliminar(Tdatos Arreglo[])
{
    clear();
    int MatriculaInt, ConfirmarEliminadoInt;
    char MatriculaChar[7], ConfirmarEliminadoChar[2];
    printf("Escribe la matricula que deas eliminar: ");
    fflush(stdin);
    gets(MatriculaChar);
    MatriculaInt = atoi(MatriculaChar);

    for (int i = 0; i < ContadorUniversal; i++)
    {
        if (Arreglo[i].Matricula == MatriculaInt)
        {
            if (Arreglo[i].Estatus == 1)
            {
                MatriculaExistente(Arreglo, i);
                printf("\n\nEstas seguro de que quieres eliminar el registro con
la matricula #%d?\n", MatriculaInt);
                printf("[0] Eliminar matricula\n");
                printf("[1] No eliminar matricula\n");
                fflush(stdin);
                gets(ConfirmarEliminadoChar);
                ConfirmarEliminadoInt = atoi(ConfirmarEliminadoChar);

                if (ConfirmarEliminadoInt == 0 || ConfirmarEliminadoInt == 1)
                {
                    if (ConfirmarEliminadoInt == 0)
                    {
                        Arreglo[i].Estatus = 0;
                        printf("Datos eliminados con exito!\n");
                    }
                    if (ConfirmarEliminadoInt == 1)
                    {
                        printf("Has escogido no eliminar los datos!\n");
                    }
                }
                pause();
            }
        }
```

```
    }
}
void OpcionTresBuscar(Tdatos Arreglo[], int MatriculasRegistradas[])
{
    char MatriculaChar[15];
    int MatriculaInt, MatriculaEncontrada;
    clear();
    printf("Escribe la matricula que quieres buscar: ");
    fflush(stdin);
    gets(MatriculaChar);
    MatriculaInt = atoi(MatriculaChar);

    for (int i = 0; i < ContadorUniversal; i++)
    {
        if (Arreglo[i].Matricula == MatriculaInt)
        {
            if (Arreglo[i].Estatus == 1)
            {
                MatriculaExistente(Arreglo, i);
            }
            if (Arreglo[i].Estatus == 0)
            {
                printf("Los datos existian pero han sido eliminados");
            }
        }
    }
    pause();
    fflush(stdin);
}
void OpcionCuatroOrdenar(Tdatos Arreglo[], int MatriculasRegistradas[], int
EstadoOrdenado)
{
    clear();
    if (EstadoOrdenado == 2)
    {
        printf("El arreglo ya se encuentra ordenado\n");
    }
    if (EstadoOrdenado == 1)
    {
        OrdenarMetodoDeBurbuja(Arreglo, MatriculasRegistradas);
        printf("Se utilizo el metodo de la burbuja para ordenar el arreglo\n");
    }
    if (EstadoOrdenado == 0)
    {
        printf("El arreglo se encuentra vacio, por lo tanto no hay datos que
ocupen ser ordenados\n");
    }
    pause();
```

```
}
void OpcionCincoImprimir(Tdatos Arreglo[], int MatriculasRegistradas[])
{
    clear();
    printf("Has escogido imprimir los registros, se imprimiran en tablas de %d
datos", RegistrosAImprimir);
    pause();
    clear();
    ImprimirTablaDeRegistros(Arreglo, MatriculasRegistradas);
    pause();
}
void OpcionSeisGuardarDocumento(Tdatos Arreglo[], int MatriculasRegistradas[],
char nom[])
{
    FILE *arch;
    arch = fopen(nom, "w+");

    if (arch != NULL)
    {
        char NombreCompleto[28];

        for (int i = 0; i < ContadorUniversal; i++)
        {
            fprintf(arch, "[%4d] | ", i);
            fprintf(arch, "%d | ", Arreglo[i].Matricula);
            JuntarNombre(Arreglo[i].NombreYApellidos.Nombre,
Arreglo[i].NombreYApellidos.ApellidoPaterno,
Arreglo[i].NombreYApellidos.ApellidoMaterno, NombreCompleto);
            fprintf(arch, "%-28s | ", NombreCompleto);
            fprintf(arch, "%d/%d/%d | ",
Arreglo[i].FechaDeNacimiento.DiaDeNacimiento,
Arreglo[i].FechaDeNacimiento.MesDeNacimiento,
Arreglo[i].FechaDeNacimiento.AnioDeNacimiento);
            fprintf(arch, "%18s\n", Arreglo[i].CURP);
        }

        printf("Documento creado con exito!");
        fclose(arch);
    }
    else
    {
        printf("Error al abrir el archivo.");
    }
}
//
void OrdenarMetodoDeBurbuja(Tdatos Arreglo[], int MatriculasRegistradas[])
{
    int Temporal;
```

```
    for (int i = 0; i < ContadorUniversal - 1; i++)
    {
        for (int j = 0; j < ContadorUniversal - 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;

                Temporal = MatriculasRegistradas[j];
                MatriculasRegistradas[j] = MatriculasRegistradas[j + 1];
                MatriculasRegistradas[j + 1] = Temporal;
            }
        }
    }
}
void ImprimirTablaDeRegistros(Tdatos Arreglo[], int MatriculasRegistradas[])
{
    char ContinuarChar[2], NombreCompletoJunto[28];
    int ContinuarInt, Validador = 0;
    EncabezadoDeLista();
    for (int i = 0; i < ContadorUniversal; i++)
    {
        if (Arreglo[i].Estatus == 1)
        {
            char NombreCompleto[36];
            printf("[%4d] | ", i);
            printf("%d | ", MatriculasRegistradas[i]);
            JuntarNombre(Arreglo[i].NombreYApellidos.Nombre,
Arreglo[i].NombreYApellidos.ApellidoPaterno,
Arreglo[i].NombreYApellidos.ApellidoMaterno, NombreCompleto);
            printf("%-28s | ", NombreCompleto);
            // printf("%s %s %s | ", Arreglo[i].NombreYApellidos.Nombre,
Arreglo[i].NombreYApellidos.ApellidoPaterno,
Arreglo[i].NombreYApellidos.ApellidoMaterno);
            printf("%d/%d/%d | ", Arreglo[i].FechaDeNacimiento.DiaDeNacimiento,
Arreglo[i].FechaDeNacimiento.MesDeNacimiento,
Arreglo[i].FechaDeNacimiento.AnioDeNacimiento);
            printf("%18s", Arreglo[i].CURP);
            printf("\n");
        }
        if (Arreglo[i].Estatus == 0)
        {
            printf("[%d] | ", 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 = ContadorUniversal;
                        Validador++;
                    }
                    if (ContinuarInt == 1)
                    {
                        Validador++;
                        pause();
                    }
                }
                else
                {
                    printf("No escogiste una opcion valida\n");
                    pause();
                    clear();
                }

            } while (Validador == 0);
            clear();
            EncabezadoDeLista();
        }
    }
}
int TamanoDeUnaCadena(char Cadena[])
{
    int Tamano = 0;
    while (Cadena[Tamano] != '\0')
    {
        Tamano++;
    }
    return Tamano;
}
void LlenadoAutomaticoDeCurp(Tdatos Arreglo[], int AnioMenosSiglos)
{
    // B A CP 040420 H CH L M D A0
```

```
    // 0 1 23 456789 0 12 3 4 5 67
    Arreglo[ContadorUniversal].CURP[0] =
Arreglo[ContadorUniversal].NombreYApellidos.ApellidoPaterno[0];
    Arreglo[ContadorUniversal].CURP[1] =
LetraMayuscula(PrimeraVocal(Arreglo[ContadorUniversal].NombreYApellidos.ApellidoP
aterno));
    Arreglo[ContadorUniversal].CURP[2] =
Arreglo[ContadorUniversal].NombreYApellidos.ApellidoMaterno[0];
    Arreglo[ContadorUniversal].CURP[3] =
Arreglo[ContadorUniversal].NombreYApellidos.Nombre[0];

    if (Arreglo[ContadorUniversal].FechaDeNacimiento.AnioDeNacimiento >= 2000)
    {
        AnioMenosSiglos =
Arreglo[ContadorUniversal].FechaDeNacimiento.AnioDeNacimiento - 2000;
    }
    else
    {
        AnioMenosSiglos =
Arreglo[ContadorUniversal].FechaDeNacimiento.AnioDeNacimiento - 1900;
    }

    Arreglo[ContadorUniversal].CURP[4] = (char)(AnioMenosSiglos / 10 + '0');
    Arreglo[ContadorUniversal].CURP[5] = (char)(AnioMenosSiglos % 10 + '0');

    Arreglo[ContadorUniversal].CURP[6] =
(char)(Arreglo[ContadorUniversal].FechaDeNacimiento.MesDeNacimiento / 10 + '0');
    Arreglo[ContadorUniversal].CURP[7] =
(char)(Arreglo[ContadorUniversal].FechaDeNacimiento.MesDeNacimiento % 10 + '0');

    Arreglo[ContadorUniversal].CURP[8] =
(char)(Arreglo[ContadorUniversal].FechaDeNacimiento.DiaDeNacimiento / 10 + '0');
    Arreglo[ContadorUniversal].CURP[9] =
(char)(Arreglo[ContadorUniversal].FechaDeNacimiento.DiaDeNacimiento % 10 + '0');

    if (Arreglo[ContadorUniversal].Genero == 0)
    {
        Arreglo[ContadorUniversal].CURP[10] = 'H';
    }
    if (Arreglo[ContadorUniversal].Genero == 1)
    {
        Arreglo[ContadorUniversal].CURP[10] = 'M';
    }
    switch (Arreglo[ContadorUniversal].EstadoDeNacimiento)
    {
    case 1:
        Arreglo[ContadorUniversal].CURP[11] = 'A';
        Arreglo[ContadorUniversal].CURP[12] = 'S';
```

```
        break;
    case 2:
        Arreglo[ContadorUniversal].CURP[11] = 'B';
        Arreglo[ContadorUniversal].CURP[12] = 'C';
        break;
    case 3:
        Arreglo[ContadorUniversal].CURP[11] = 'B';
        Arreglo[ContadorUniversal].CURP[12] = 'S';
        break;
    case 4:
        Arreglo[ContadorUniversal].CURP[11] = 'C';
        Arreglo[ContadorUniversal].CURP[12] = 'M';
        break;
    case 5:
        Arreglo[ContadorUniversal].CURP[11] = 'C';
        Arreglo[ContadorUniversal].CURP[12] = 'S';
        break;
    case 6:
        Arreglo[ContadorUniversal].CURP[11] = 'C';
        Arreglo[ContadorUniversal].CURP[12] = 'H';
        break;
    case 7:
        Arreglo[ContadorUniversal].CURP[11] = 'C';
        Arreglo[ContadorUniversal].CURP[12] = 'H';
        break;
    case 8:
        Arreglo[ContadorUniversal].CURP[11] = 'C';
        Arreglo[ContadorUniversal].CURP[12] = 'S';
        break;
    case 9:
        Arreglo[ContadorUniversal].CURP[11] = 'C';
        Arreglo[ContadorUniversal].CURP[12] = 'L';
        break;
    case 10:
        Arreglo[ContadorUniversal].CURP[11] = 'D';
        Arreglo[ContadorUniversal].CURP[12] = 'G';
        break;
    case 11:
        Arreglo[ContadorUniversal].CURP[11] = 'C';
        Arreglo[ContadorUniversal].CURP[12] = 'M';
        break;
    case 12:
        Arreglo[ContadorUniversal].CURP[11] = 'G';
        Arreglo[ContadorUniversal].CURP[12] = 'T';
        break;
    case 13:
        Arreglo[ContadorUniversal].CURP[11] = 'G';
        Arreglo[ContadorUniversal].CURP[12] = 'R';
```

```
        break;
    case 14:
        Arreglo[ContadorUniversal].CURP[11] = 'H';
        Arreglo[ContadorUniversal].CURP[12] = 'G';
        break;
    case 15:
        Arreglo[ContadorUniversal].CURP[11] = 'J';
        Arreglo[ContadorUniversal].CURP[12] = 'C';
        break;
    case 16:
        Arreglo[ContadorUniversal].CURP[11] = 'M';
        Arreglo[ContadorUniversal].CURP[12] = 'N';
        break;
    case 17:
        Arreglo[ContadorUniversal].CURP[11] = 'M';
        Arreglo[ContadorUniversal].CURP[12] = 'S';
        break;
    case 18:
        Arreglo[ContadorUniversal].CURP[11] = 'N';
        Arreglo[ContadorUniversal].CURP[12] = 'T';
        break;
    case 19:
        Arreglo[ContadorUniversal].CURP[11] = 'N';
        Arreglo[ContadorUniversal].CURP[12] = 'L';
        break;
    case 20:
        Arreglo[ContadorUniversal].CURP[11] = 'O';
        Arreglo[ContadorUniversal].CURP[12] = 'C';
        break;
    case 21:
        Arreglo[ContadorUniversal].CURP[11] = 'P';
        Arreglo[ContadorUniversal].CURP[12] = 'L';
        break;
    case 22:
        Arreglo[ContadorUniversal].CURP[11] = 'Q';
        Arreglo[ContadorUniversal].CURP[12] = 'T';
        break;
    case 23:
        Arreglo[ContadorUniversal].CURP[11] = 'Q';
        Arreglo[ContadorUniversal].CURP[12] = 'R';
        break;
    case 24:
        Arreglo[ContadorUniversal].CURP[11] = 'S';
        Arreglo[ContadorUniversal].CURP[12] = 'P';
        break;
    case 25:
        Arreglo[ContadorUniversal].CURP[11] = 'S';
        Arreglo[ContadorUniversal].CURP[12] = 'N';
```

```
        break;
    case 26:
        Arreglo[ContadorUniversal].CURP[11] = 'S';
        Arreglo[ContadorUniversal].CURP[12] = 'R';
        break;
    case 27:
        Arreglo[ContadorUniversal].CURP[11] = 'T';
        Arreglo[ContadorUniversal].CURP[12] = 'C';
        break;
    case 28:
        Arreglo[ContadorUniversal].CURP[11] = 'T';
        Arreglo[ContadorUniversal].CURP[12] = 'S';
        break;
    case 29:
        Arreglo[ContadorUniversal].CURP[11] = 'T';
        Arreglo[ContadorUniversal].CURP[12] = 'L';
        break;
    case 30:
        Arreglo[ContadorUniversal].CURP[11] = 'V';
        Arreglo[ContadorUniversal].CURP[12] = 'Z';
        break;
    case 31:
        Arreglo[ContadorUniversal].CURP[11] = 'Y';
        Arreglo[ContadorUniversal].CURP[12] = 'N';
        break;
    case 32:
        Arreglo[ContadorUniversal].CURP[11] = 'Z';
        Arreglo[ContadorUniversal].CURP[12] = 'S';
        break;
    case 33:
        Arreglo[ContadorUniversal].CURP[11] = 'N';
        Arreglo[ContadorUniversal].CURP[12] = 'E';
        break;
    }
    Arreglo[ContadorUniversal].CURP[13] =
LetraMayuscula(PrimeraConsonante(Arreglo[ContadorUniversal].NombreYApellidos.Apel
lidoPaterno));
    Arreglo[ContadorUniversal].CURP[14] =
LetraMayuscula(PrimeraConsonante(Arreglo[ContadorUniversal].NombreYApellidos.Apel
lidoMaterno));
    Arreglo[ContadorUniversal].CURP[15] =
LetraMayuscula(PrimeraConsonante(Arreglo[ContadorUniversal].NombreYApellidos.Nomb
re));
    if (Arreglo[ContadorUniversal].FechaDeNacimiento.AnioDeNacimiento >= 2000)
    {
        Arreglo[ContadorUniversal].CURP[16] = 'A';
        Arreglo[ContadorUniversal].CURP[17] = '0';
    }
```

```
    if (Arreglo[ContadorUniversal].FechaDeNacimiento.AnioDeNacimiento < 2000)
    {
        Arreglo[ContadorUniversal].CURP[16] = Arreglo[ContadorUniversal].CURP[4];
        Arreglo[ContadorUniversal].CURP[17] = Arreglo[ContadorUniversal].CURP[5];
    }
}
void CuidarDesbordamiento(int CantidadAutomatica)
{
    if ((CantidadAutomatica + ContadorUniversal) > TamanoDelArreglo)
    {
        CantidadAutomatica = TamanoDelArreglo - ContadorUniversal;
        if (CantidadAutomatica != 0)
        {
            printf("Para evitar el desbordamiento del buffer, se ha ajustado la
cantidad automática de datos a agregar de %d a %d\n", LlenadoAutomatico,
CantidadAutomatica);
        }
    }
}
int MatriculaRepetida(int MatriculaTemporal, int MatriculasRegistradas[])
{
    // 0 Matricula repetida
    // 1 Matricula no repetida
    for (int i = 0; i < ContadorUniversal; i++)
    {
        if (MatriculaTemporal == MatriculasRegistradas[i])
        {
            return 0;
        }
    }
    return 1;
}
void ImprimirListaDeMatriculasAgregadas(int CantidadAutomatica, int
MatriculasRegistradas[])
{
    // ImprimirListaDeMatriculasAgregadas(CantidadAutomatica,
MatriculasRegistradas[])
    printf("Se agregaron con éxito %d datos\n", CantidadAutomatica);
    printf("La lista de matrículas agregadas es: \n\n");
    for (int i = ContadorUniversal - CantidadAutomatica; i < ContadorUniversal;
i++)
    {
        printf("[%d] %d\n", i, MatriculasRegistradas[i]);
    }
}
void MatriculaExistente(Tdatos Arreglo[], int i)

{
```

```
    clear();
    printf("La matricula si existe!\n\n");
    printf("Matricula de la persona: %d\n", Arreglo[i].Matricula);
    printf("Nombre de la persona: %s\n", Arreglo[i].NombreYApellidos.Nombre);
    printf("Apellido paterno de la persona: %s\n",
Arreglo[i].NombreYApellidos.ApellidoPaterno);
    printf("Apellido materno de la persona: %s\n",
Arreglo[i].NombreYApellidos.ApellidoMaterno);
    printf("Fecha de nacimiento de la persona: %d/%d/%d\n",
Arreglo[i].FechaDeNacimiento.DiaDeNacimiento,
Arreglo[i].FechaDeNacimiento.MesDeNacimiento,
Arreglo[i].FechaDeNacimiento.AnioDeNacimiento);
    printf("Edad de la persona: %d\n", 2024 -
Arreglo[i].FechaDeNacimiento.AnioDeNacimiento);
    if (Arreglo[i].Genero == 0)
    {
        printf("Sexo de la persona: Hombre\n");
    }
    if (Arreglo[i].Genero == 1)
    {
        printf("Sexo de la persona: Mujer\n");
    }
    printf("Estado de nacimiento de la persona: ");
    switch (Arreglo[i].EstadoDeNacimiento)
    {
    case 1:
        printf("Aguascalientes\n");
        break;
    case 2:
        printf("Baja California\n");
        break;
    case 3:
        printf("Baja California Sur\n");
        break;
    case 4:
        printf("Campeche\n");
        break;
    case 5:
        printf("Chiapas\n");
        break;
    case 6:
        printf("Chihuahua\n");
        break;
    case 7:
        printf("Ciudad de México\n");
        break;
    case 8:
        printf("Coahuila\n");
```

```
        break;
    case 9:
        printf("Colima\n");
        break;
    case 10:
        printf("Durango\n");
        break;
    case 11:
        printf("Estado de México\n");
        break;
    case 12:
        printf("Guanajuato\n");
        break;
    case 13:
        printf("Guerrero\n");
        break;
    case 14:
        printf("Hidalgo\n");
        break;
    case 15:
        printf("Jalisco\n");
        break;
    case 16:
        printf("Michoacán\n");
        break;
    case 17:
        printf("Morelos\n");
        break;
    case 18:
        printf("Nayarit\n");
        break;
    case 19:
        printf("Nuevo León\n");
        break;
    case 20:
        printf("Oaxaca\n");
        break;
    case 21:
        printf("Puebla\n");
        break;
    case 22:
        printf("Querétaro\n");
        break;
    case 23:
        printf("Quintana Roo\n");
        break;
    case 24:
        printf("San Luis Potosí\n");
```

```
        break;
    case 25:
        printf("Sinaloa\n");
        break;
    case 26:
        printf("Sonora\n");
        break;
    case 27:
        printf("Tabasco\n");
        break;
    case 28:
        printf("Tamaulipas\n");
        break;
    case 29:
        printf("Tlaxcala\n");
        break;
    case 30:
        printf("Veracruz\n");
        break;
    case 31:
        printf("Yucatán\n");
        break;
    case 32:
        printf("Zacatecas\n");
        break;
    case 33:
        printf("Extranjero\n");
        break;
    }
    printf("El CURP de la persona es: %s", Arreglo[i].CURP);
}
//
// Funciones para manejo de caracteres
void JuntarNombre(char x[], char y[], char z[], char NombreCompleto[])
{
    sprintf(NombreCompleto, "%s %s %s", x, y, z);
}
char LetraMayuscula(char c)
{
    if (c >= 'a' && c <= 'z')
    {
        return c - ('a' - 'A');
    }
    else
    {
        return c;
    }
}
```

```
char PrimeraConsonante(char Palabra[])
{
    for (int i = 1; i < 20; i++)
    {
        if (Palabra[i] == 'b' || Palabra[i] == 'c' || Palabra[i] == 'd' ||
Palabra[i] == 'f' || Palabra[i] == 'g' ||
            Palabra[i] == 'h' || Palabra[i] == 'j' || Palabra[i] == 'k' ||
Palabra[i] == 'l' || Palabra[i] == 'm' ||
            Palabra[i] == 'n' || Palabra[i] == 'p' || Palabra[i] == 'q' ||
Palabra[i] == 'r' ||
            Palabra[i] == 's' || Palabra[i] == 't' || Palabra[i] == 'v' ||
Palabra[i] == 'w' || Palabra[i] == 'x' ||
            Palabra[i] == 'y' || Palabra[i] == 'z' || Palabra[i] == 'B' ||
Palabra[i] == 'C' || Palabra[i] == 'D' ||
            Palabra[i] == 'F' || Palabra[i] == 'G' || Palabra[i] == 'H' ||
Palabra[i] == 'J' || Palabra[i] == 'K' ||
            Palabra[i] == 'L' || Palabra[i] == 'M' || Palabra[i] == 'N' ||
Palabra[i] == 'P' ||
            Palabra[i] == 'Q' || Palabra[i] == 'R' || Palabra[i] == 'S' ||
Palabra[i] == 'T' || Palabra[i] == 'V' ||
            Palabra[i] == 'W' || Palabra[i] == 'X' || Palabra[i] == 'Y' ||
Palabra[i] == 'Z')
        {
            return Palabra[i];
        }
    }
}
char PrimeraVocal(char Palabra[])
{
    for (int i = 1; i < 25; i++)
    {
        if (Palabra[i] == 'a' || Palabra[i] == 'e' || Palabra[i] == 'i' ||
Palabra[i] == 'o' || Palabra[i] == 'u' ||
            Palabra[i] == 'A' || Palabra[i] == 'E' || Palabra[i] == 'I' ||
Palabra[i] == 'O' || Palabra[i] == 'U')
        {
            return Palabra[i];
        }
    }
}
//
// Funciones sin logica
void EncabezadoDeLista()
{
    printf(" #   Matricula    Nombre   Apellidos        Fecha de nacimiento
CURP    \n");
}
void ImprimirMenu()
```

```
{
    clear();
    printf("[1] Cargar automaticamente \n");
    printf("[2] Eliminar\n");
    printf("[3] Buscar\n");
    printf("[4] Ordenar\n");
    printf("[5] Imprimir\n");
    printf("[6] Archivo Texto\n");
    printf("\n");
    printf("[0] Salir\n");
}
//
// Funciones genericas
void clear()
{
    system("clear");
}
void pause()
{
    printf("\nPresiona Enter para continuar...");
    fflush(stdin);
    getchar();
}
```