
# Introducción a la Programación  Practica 8

Medina Martinez Jonathan Jason  2023640061

02 de junio del 2023

# Índice

## 1. Objetivo

Desarrollo de programas utilizando estructuras como aplicación para las bases de datos.

## 2. Introducción

El desarrollo de programas que utilicen estructuras como aplicación para las bases de datos es una habilidad fundamental en el ámbito de la programación. En esta práctica, nos centraremos en el desarrollo de un programa en lenguaje C que permita gestionar una lista de asistencia de alumnos. Para lograr esto, utilizaremos diferentes estructuras como Alumno, Profesor, Lista y Fecha. A través de un menú de opciones, los usuarios podrán registrar unidades de aprendizaje, grupos, fechas, profesores, así como agregar, editar y mostrar la lista de alumnos. Además, se utilizará un arreglo de tamaño dinámico para almacenar la lista de alumnos. A medida que avancemos en el desarrollo de esta práctica, exploraremos los conceptos clave de la programación estructurada y la gestión de bases de datos.

# 3. Desarrollo

## 3.1. Lista de Alumnos

Desarrolle un programa en C que permita tener una lista de asistencia de alumnos. Para esto, deberá crear las siguientes estructuras:

- Alumno: Nombre, Numero de Boleta, Año de Ingreso, Carrera, Turno.
- Profesor: Nombre, Numero de Empleado, Turno.
- Lista: Fecha, Unidad de Aprendizaje, Grupo, Profesor, Lista de Alumnos.
- Fecha: Día, Mes, Año.

El programa deberá tener el siguiente menú:

- Registrar Unidad de Aprendizaje.
- Registrar Grupo.
- Registrar Fecha.
- Registrar Profesor.
- Agregar Alumno.
- Editar Alumno.
- Mostrar Lista.
- Salir.

La Lista de Alumnos deberá ser un arreglo de tamaño dinámico.

### 3.1.1. Lista.c

```c
/**
* @file Lista.c
* @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
* @brief
* @version 0.1
* @date 2023-06-03
*
* @copyrigth GPlv3
*
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_NOMBRE 50
#define MAX_CARRERA 50
#define MAX_ALUMNOS 100

typedef struct {
	char nombre[MAX_NOMBRE];
	int numeroBoleta;
	int anioIngreso;
	char carrera[MAX_CARRERA];
	char turno;
} Alumno;

typedef struct {
	char nombre[MAX_NOMBRE];
	int numeroEmpleado;
	char turno;
} Profesor;

typedef struct {
	int dia;
	int mes;
	int anio;
} Fecha;

typedef struct {
	Fecha fecha;
	char unidadAprendizaje[MAX_NOMBRE];
	char grupo;
```

```
        Profesor profesor;
        Alumno alumnos[MAX_ALUMNOS];
        int numAlumnos;
} Lista;

void registrarUnidadAprendizaje(Lista *lista);
void registrarGrupo(Lista *lista);
void registrarFecha(Lista *lista);
void registrarProfesor(Lista *lista);
void agregarAlumno(Lista *lista);
void editarAlumno(Lista *lista);
void mostrarLista(const Lista *lista);
void liberarMemoria(Lista *lista);

int main() {
        Lista lista;
        lista.numAlumnos = 0;

        int opcion;
        char opcionStr[10];

        do {
                printf("\n--- MENU ---\n");
                printf("1. Registrar Unidad de Aprendizaje\n");
                printf("2. Registrar Grupo\n");
                printf("3. Registrar Fecha\n");
                printf("4. Registrar Profesor\n");
                printf("5. Agregar Alumno\n");
                printf("6. Editar Alumno\n");
                printf("7. Mostrar Lista\n");
                printf("8. Salir\n");
                printf("Ingrese una opcion: ");
                fgets(opcionStr, sizeof(opcionStr), stdin);
                opcionStr[strcspn(opcionStr, "\n")] = '\0';
                opcion = atoi(opcionStr);

                switch (opcion) {
                        case 1:
                        registrarUnidadAprendizaje(&lista);
                        break;
                        case 2:
                        registrarGrupo(&lista);
                        break;
                        case 3:
```

```
                        registrarFecha(&lista);
                        break;
                        case 4:
                        registrarProfesor(&lista);
                        break;
                        case 5:
                        agregarAlumno(&lista);
                        break;
                        case 6:
                        editarAlumno(&lista);
                        break;
                        case 7:
                        mostrarLista(&lista);
                        break;
                        case 8:
                        liberarMemoria(&lista);
                        printf("Hasta luego!\n");
                        break;
                        default:
                        printf("Opcion no valida. Intente de nuevo.\n");
                        break;
                }
        } while (opcion != 8);

        return 0;
}

void registrarUnidadAprendizaje(Lista *lista) {
        printf("\n--- Registrar Unidad de Aprendizaje ---\n");
        printf("Ingrese el nombre de la unidad de aprendizaje: ");
        fgets(lista->unidadAprendizaje, sizeof(lista->unidadAprendizaje), stdin);
        lista->unidadAprendizaje[strcspn(lista->unidadAprendizaje, "\n")] = '\0';
}

void registrarGrupo(Lista *lista) {
        printf("\n--- Registrar Grupo ---\n");
        printf("Ingrese el grupo: ");
        scanf(" %c", &(lista->grupo));
        getchar();
}

void registrarFecha(Lista *lista) {
        printf("\n--- Registrar Fecha ---\n");
        printf("Ingrese el dia: ");
```

```
132         scanf("%d", &(lista->fecha.dia));
133         printf("Ingrese el mes: ");
134         scanf("%d", &(lista->fecha.mes));
135         printf("Ingrese el anio: ");
136         scanf("%d", &(lista->fecha.anio));
137 }
138
139 void registrarProfesor(Lista *lista) {
140         printf("\n--- Registrar Profesor ---\n");
141         printf("Ingrese el nombre del profesor: ");
142         fgets(lista->profesor.nombre, sizeof(lista->profesor.nombre), stdin);
143         lista->profesor.nombre[strcspn(lista->profesor.nombre, "\n")] = '\0';
144         printf("Ingrese el numero de empleado: ");
145         scanf("%d", &(lista->profesor.numeroEmpleado));
146         printf("Ingrese el turno del profesor: ");
147         scanf(" %c", &(lista->profesor.turno));
148 }
149
150 void agregarAlumno(Lista *lista) {
151         printf("\n--- Agregar Alumno ---\n");
152         if (lista->numAlumnos < MAX_ALUMNOS) {
153                 Alumno *nuevoAlumno = &(lista->alumnos[lista->numAlumnos]);
154
155                 printf("Ingrese el nombre del alumno: ");
156                 fgets(nuevoAlumno->nombre, sizeof(nuevoAlumno->nombre), stdin);
157                 nuevoAlumno->nombre[strcspn(nuevoAlumno->nombre, "\n")] = '\0';
158                 printf("Ingrese el numero de boleta: ");
159                 scanf("%d", &(nuevoAlumno->numeroBoleta));
160                 printf("Ingrese el anio de ingreso: ");
161                 scanf("%d", &(nuevoAlumno->anioIngreso));
162                 printf("Ingrese la carrera del alumno: ");
163                 fgets(nuevoAlumno->carrera, sizeof(nuevoAlumno->carrera), stdin);
164                 nuevoAlumno->carrera[strcspn(nuevoAlumno->carrera, "\n")] = '\0';
165                 printf("Ingrese el turno del alumno: ");
166                 scanf(" %c", &(nuevoAlumno->turno));
167
168                 lista->numAlumnos++;
169         } else {
170                 printf("No se pueden agregar mas alumnos. Limite alcanzado.\n");
171         }
172 }
173
174 void editarAlumno(Lista *lista) {
175         printf("\n--- Editar Alumno ---\n");
```

```
176         int indice;
177         printf("Ingrese el indice del alumno a editar (1-%d): ", lista->numAlumnos);
178         scanf("%d", &indice);
179 
180         if (indice >= 1 && indice <= lista->numAlumnos) {
181                 Alumno *alumno = &(lista->alumnos[indice - 1]);
182 
183                 printf("Ingrese el nombre del alumno: ");
184                 fgets(alumno->nombre, sizeof(alumno->nombre), stdin);
185                 alumno->nombre[strcspn(alumno->nombre, "\n")] = '\0';
186                 printf("Ingrese el numero de boleta: ");
187                 scanf("%d", &(alumno->numeroBoleta));
188                 printf("Ingrese el anio de ingreso: ");
189                 scanf("%d", &(alumno->anioIngreso));
190                 printf("Ingrese la carrera del alumno: ");
191                 fgets(alumno->carrera, sizeof(alumno->carrera), stdin);
192                 alumno->carrera[strcspn(alumno->carrera, "\n")] = '\0';
193                 printf("Ingrese el turno del alumno: ");
194                 scanf(" %c", &(alumno->turno));
195 
196                 printf("Alumno editado con exito.\n");
197         } else {
198                 printf("Indice de alumno invalido.\n");
199         }
200 }
201 
202 void mostrarLista(const Lista *lista) {
203         printf("\n--- Mostrar Lista ---\n");
204         printf("Unidad de Aprendizaje: %s\n", lista->unidadAprendizaje);
205         printf("Grupo: %c\n", lista->grupo);
206         printf("Fecha: %02d/%02d/%d\n", lista->fecha.dia, lista->fecha.mes, lista->
                fecha.anio);
207         printf("Profesor: %s\n", lista->profesor.nombre);
208         printf("Numero de Empleado: %d\n", lista->profesor.numeroEmpleado);
209         printf("Turno del Profesor: %c\n", lista->profesor.turno);
210 
211         printf("\nAlumnos:\n");
212         for (int i = 0; i < lista->numAlumnos; i++) {
213                 const Alumno *alumno = &(lista->alumnos[i]);
214                 printf("%d. %s, Boleta: %d, Anio de Ingreso: %d, Carrera: %s, Turno:
                        %c\n",
215                 i + 1, alumno->nombre, alumno->numeroBoleta, alumno->anioIngreso,
                        alumno->carrera, alumno->turno);
216         }
```

```
217 }
218
219 void liberarMemoria(Lista *lista) {
220         if (lista->alumnos != NULL) {
221                 lista->numAlumnos = 0;
222                 memset(lista->alumnos, 0, sizeof(lista->alumnos));
223         }
224 }
```

## 4. Conclusión

En conclusión, esta práctica nos ha permitido adquirir conocimientos y habilidades fundamentales en el desarrollo de programas que utilizan estructuras como aplicación para las bases de datos. A lo largo del proceso, hemos aprendido a diseñar y utilizar estructuras como Alumno, Profesor, Lista y Fecha para gestionar una lista de asistencia de alumnos. Además, hemos implementado un menú de opciones que ofrece funcionalidades como el registro de unidades de aprendizaje, grupos, fechas y profesores, así como la capacidad de agregar, editar y mostrar la lista de alumnos. También hemos utilizado un arreglo de tamaño dinámico para adaptarnos a las necesidades cambiantes de la lista de alumnos. Esta práctica ha sido un paso importante en nuestro aprendizaje de la programación estructurada y la gestión de bases de datos, y nos ha preparado para enfrentar desafíos más complejos en el futuro.