Carrera: Electrónica y automatización **Asignatura:** Fundamentos de programación
Tema: Crud de estudiantes con archivos txt en C **Docente:** Jenny Ruiz
Integrantes: Francisco Comina, Dylan Tutillo, Omar Alquinga **Fecha:** 12/01/2026
NRC: 29583

Código

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

#define ARCHIVO_EST "estudiantes.txt"
/* ============================
   ESTRUCTURA
   ============================ */
typedef struct {
    char id[20];
    char apellidos[50];
    char nombres[50];
    int edad;
} Estudiante;
/* ============================
   UTILIDADES DE ENTRADA
   ============================ */

// Elimina el '\n' dejado por fgets
void limpiarNuevaLinea(char *s) {
    s[strcspn(s, "\n")] = '\0';
```

```
}
// Lee una cadena de forma segura
void leerCadena(const char *etiqueta, char *dest, int tam) {
    printf("%s", etiqueta);
    fgets(dest, tam, stdin);
    limpiarNuevaLinea(dest);
}
// Lee un entero de forma robusta
int leerEntero(const char *etiqueta) {
    char buf[50];
    long val;
    char *endptr;

    while (1) {
        printf("%s", etiqueta);
        fgets(buf, sizeof(buf), stdin);
        val = strtol(buf, &endptr, 10);

        if (endptr != buf) {
            return (int)val;
        }
        printf("Entrada inválida. Intente de nuevo.\n");
    }
}
/* ==========================
   PARSER DE LÍNEA
   ========================== */

// Convierte "id;apellidos;nombres;edad" a Estudiante
int parsearEstudiante(const char *linea, Estudiante *e) {
    int n = sscanf(linea, "%19[^;];%49[^;];%49[^;];%d",
```

```
            e->id, e->apellidos, e->nombres, &e->edad);
    return (n == 4);
}
/* ============================
   VALIDAR ID EXISTENTE
   ============================ */
int existeId(const char *idBuscado) {
    FILE *f = fopen(ARCHIVO_EST, "r");
    if (f == NULL) return 0;
    char linea[200];
    Estudiante e;
    while (fgets(linea, sizeof(linea), f)) {
        if (parsearEstudiante(linea, &e) &&
            strcmp(e.id, idBuscado) == 0) {
            fclose(f);
            return 1;
        }
    }
    fclose(f);
    return 0;
}
/* ============================
   AGREGAR ESTUDIANTE
   ============================ */
void agregarEstudiante() {
    Estudiante nuevo;
    leerCadena("Ingrese ID: ", nuevo.id, sizeof(nuevo.id));
    if (existeId(nuevo.id)) {
        printf("Ya existe un estudiante con ese ID.\n");
        return;
    }
```

```
    leerCadena("Ingrese apellidos: ", nuevo.apellidos, sizeof(nuevo.apellidos));
    leerCadena("Ingrese nombres: ", nuevo.nombres, sizeof(nuevo.nombres));
    nuevo.edad = leerEntero("Ingrese edad: ");

    FILE *f = fopen(ARCHIVO_EST, "a");
    if (f == NULL) {
        printf("Error al abrir el archivo.\n");
        return;
    }
    fprintf(f, "%s;%s;%s;%d\n",
            nuevo.id, nuevo.apellidos, nuevo.nombres, nuevo.edad);
    fclose(f);
    printf("Estudiante agregado correctamente.\n");
}
/* ==========================
   LISTAR ESTUDIANTES
   ========================== */
void listarEstudiantes() {
    FILE *f = fopen(ARCHIVO_EST, "r");
    if (f == NULL) {
        printf("No hay datos registrados.\n");
        return;
    }
    char linea[200];
   Estudiante e;
    printf("\n%-10s %-20s %-20s %-5s\n",
           "ID", "APELLIDOS", "NOMBRES", "EDAD");
    printf("-----------------------------------------------------------\n");
    while (fgets(linea, sizeof(linea), f)) {
        if (parsearEstudiante(linea, &e)) {
            printf("%-10s %-20s %-20s %-5d\n",
```

```
                e.id, e.apellidos, e.nombres, e.edad);
        }
    }
    fclose(f);
}
/* ==========================
   BUSCAR ESTUDIANTE
   ========================== */
int buscarPorId(const char *idBuscado, Estudiante *salida) {
    FILE *f = fopen(ARCHIVO_EST, "r");
    if (f == NULL) return 0;
    char linea[200];
    Estudiante e;

    while (fgets(linea, sizeof(linea), f)) {
        if (parsearEstudiante(linea, &e) &&
            strcmp(e.id, idBuscado) == 0) {
            *salida = e;
            fclose(f);
            return 1;
        }
    }
    fclose(f);
    return 0;
}
void consultarEstudiante() {
    char id[20];
    Estudiante e;
    leerCadena("Ingrese ID a buscar: ", id, sizeof(id));
    if (buscarPorId(id, &e)) {
        printf("Encontrado: %s %s (Edad: %d)\n",
```

```
            e.nombres, e.apellidos, e.edad);
    } else {
        printf("No se encontró el ID.\n");
    }
}
/* ==========================
   ACTUALIZAR ESTUDIANTE
   ========================== */
void actualizarEstudiante() {
    char idObjetivo[20];
    leerCadena("Ingrese ID a actualizar: ", idObjetivo, sizeof(idObjetivo));

    FILE *f = fopen(ARCHIVO_EST, "r");
    if (f == NULL) {
        printf("No existe el archivo.\n");
        return;
    }/
    FILE *temp = fopen("tmp.txt", "w");
    if (temp == NULL) {
        fclose(f);
        printf("No se pudo crear archivo temporal.\n");
        return;
    }
    char linea[200];
    Estudiante e;
    int actualizado = 0;

    while (fgets(linea, sizeof(linea), f)) {
        if (parsearEstudiante(linea, &e) &&
            strcmp(e.id, idObjetivo) == 0) {
            leerCadena("Nuevos apellidos: ", e.apellidos, sizeof(e.apellidos));
```

```
            leerCadena("Nuevos nombres: ", e.nombres, sizeof(e.nombres));
            e.edad = leerEntero("Nueva edad: ");

            fprintf(temp, "%s;%s;%s;%d\n",
                    e.id, e.apellidos, e.nombres, e.edad);
            actualizado = 1;
        } else if (parsearEstudiante(linea, &e)) {
            fprintf(temp, "%s;%s;%s;%d\n",
                    e.id, e.apellidos, e.nombres, e.edad);
        }
    }
    fclose(f);
    fclose(temp);
    remove(ARCHIVO_EST);
    rename("tmp.txt", ARCHIVO_EST);

    if (actualizado)
        printf("Registro actualizado.\n");
    else
        printf("No se encontró el ID.\n");
}
/* ==========================
   ELIMINAR ESTUDIANTE
   ========================== */
void eliminarEstudiante() {
    char idEliminar[20];
    leerCadena("Ingrese ID a eliminar: ", idEliminar, sizeof(idEliminar));

    FILE *f = fopen(ARCHIVO_EST, "r");
    if (f == NULL) {
        printf("No existe el archivo.\n");
```

```
    return;
  }
  FILE *temp = fopen("tmp.txt", "w");
  if (temp == NULL) {
    fclose(f);
    printf("No se pudo crear archivo temporal.\n");
    return;
  }
  char linea[200];
  Estudiante e;
  int eliminado = 0;
  while (fgets(linea, sizeof(linea), f)) {
    if (parsearEstudiante(linea, &e)) {
      if (strcmp(e.id, idEliminar) != 0) {
        fprintf(temp, "%s;%s;%s;%d\n",
                e.id, e.apellidos, e.nombres, e.edad);
      } else {
        eliminado = 1;
      }
    }
  }
  fclose(f);
  fclose(temp);
  remove(ARCHIVO_EST);
  rename("tmp.txt", ARCHIVO_EST);
  if (eliminado)
    printf("Registro eliminado.\n");
  else
    printf("No se encontró el ID.\n");
}
/* ========================
```

```
   MENÚ
   ========================== */
int menu() {
    char opcion[10];

    printf("\n=== CRUD Estudiantes (TXT) ===\n");
    printf("1. Agregar\n");
    printf("2. Listar\n");
    printf("3. Consultar por ID\n");
    printf("4. Actualizar\n");
    printf("5. Eliminar\n");
    printf("0. Salir\n");
    printf("Seleccione una opción: ");

    fgets(opcion, sizeof(opcion), stdin);
    return atoi(opcion);
}

/* ==========================
   MAIN
   ========================== */
int main() {
    int op;

    do {
        op = menu();
        switch (op) {
            case 1: agregarEstudiante(); break;
            case 2: listarEstudiantes(); break;
            case 3: consultarEstudiante(); break;
            case 4: actualizarEstudiante(); break;
```

```
            case 5: eliminarEstudiante(); break;
            case 0: printf("Saliendo...\n"); break;
            default: printf("Opción inválida.\n");
        }
    } while (op != 0);

    return 0;
}
```

```
=== CRUD Estudiantes (TXT) ===
1. Agregar
2. Listar
3. Consultar por ID
4. Actualizar
5. Eliminar
0. Salir
Seleccione una opci├│n: 1
Ingrese ID: 1727383497
Ingrese apellidos: Tutillo Heredia
Ingrese nombres: Dylan Josue
Ingrese edad: 19
Estudiante agregado correctamente.
```

```
=== CRUD Estudiantes (TXT) ===
1. Agregar
2. Listar
3. Consultar por ID
4. Actualizar
5. Eliminar
0. Salir
Seleccione una opci├│n: 1
Ingrese ID: 17164345766
Ingrese apellidos: Moran Heredia
Ingrese nombres: Cristian Daniel
Ingrese edad: 20
Estudiante agregado correctamente.
```

```
=== CRUD Estudiantes (TXT) ===
1. Agregar
2. Listar
3. Consultar por ID
4. Actualizar
5. Eliminar
0. Salir
Seleccione una opci├│n: 1
Ingrese ID: 1726358797
Ingrese apellidos: Morales Espinoza
Ingrese nombres: Pedro Josue
Ingrese edad: 25
Estudiante agregado correctamente.
```

```
=== CRUD Estudiantes (TXT) ===
1. Agregar
2. Listar
3. Consultar por ID
4. Actualizar
5. Eliminar
0. Salir
Seleccione una opci├│n: 2

ID         APELLIDOS            NOMBRES              EDAD
------------------------------------------------------------
1727383497 Tutillo Heredia      Dylan Josue          19
17164345766 Moran Heredia        Cristian Daniel      20
1726358797 Morales Espinoza     Pedro Josue          25
```

```
=== CRUD Estudiantes (TXT) ===
1. Agregar
2. Listar
3. Consultar por ID
4. Actualizar
5. Eliminar
0. Salir
Seleccione una opci├│n: 3
Ingrese ID a buscar: 1727383497
Encontrado: Dylan Josue Tutillo Heredia (Edad: 19)
```

```
=== CRUD Estudiantes (TXT) ===
1. Agregar
2. Listar
3. Consultar por ID
4. Actualizar
5. Eliminar
0. Salir
Seleccione una opci├│n: 4
Ingrese ID a actualizar: 1727383497
Nuevos apellidos: Moran Tutillo
Nuevos nombres: Erick Mateo
Nueva edad: 24
Registro actualizado.
```

```
=== CRUD Estudiantes (TXT) ===
1. Agregar
2. Listar
3. Consultar por ID
4. Actualizar
5. Eliminar
0. Salir
Seleccione una opci├│n: 2

ID         APELLIDOS            NOMBRES              EDAD
-------------------------------------------------------------
1727383497 Moran Tutillo        Erick Mateo          24
17164345766 Moran Heredia        Cristian Daniel      20
1726358797 Morales Espinoza     Pedro Josue          25
```

```
=== CRUD Estudiantes (TXT) ===
1. Agregar
2. Listar
3. Consultar por ID
4. Actualizar
5. Eliminar
0. Salir
Seleccione una opci├│n: 5
Ingrese ID a eliminar: 1727383497
Registro eliminado.

=== CRUD Estudiantes (TXT) ===
1. Agregar
2. Listar
3. Consultar por ID
4. Actualizar
5. Eliminar
0. Salir
Seleccione una opci├│n: 2

ID         APELLIDOS            NOMBRES              EDAD
-------------------------------------------------------------
17164345766 Moran Heredia        Cristian Daniel      20
1726358797 Morales Espinoza     Pedro Josue          25
```

Conclusión:

Al haber aprendido el manejo de estructuras logramos desarrollar este programa que es un buen ejemplo académico y práctico de cómo aplicar lo aprendido en clases, manejo de archivos y control de errores en C. También sirve como base sólida para futuros proyectos, como validaciones más estrictas, ordenamiento de registros, incluso integración con bases de datos.

Recomendación:

Es importante tomar en cuenta que este tipo de códigos tan extensos que contienen muchos procesos que guardan y cumplen funciones, tienen un gran alto nivel de complejidad si es que no se comprende el tema y lo que mas se recomienda es saber todo tipo de conceptos de para que sirve cada función, librería, estructura y así poder crear nuestros códigos de una forma más fácil.