UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE

Nombre: Criollo Solano Jahir **NRC:** 29583

Fecha: 13/01/2026

Tema: CRUD

1. EJERCICIOS

Ejercicio 1 (Datos Población)

Crear un código ejecutable en Code Blocks e implementar el CRUD completo y probarlo con al menos 10 registros.

1.- Crear el proyecto en Code Blocks y pegar el código en un solo archivo main.c
2.- Ejecutar y agregar 3 estudiantes. Verificar que se haya creado estudiantes.txt y cada estuante quede en una sola línea.
3.- Probar la validación de ID duplicado (intentar agregar el mismo ID)
4.- Listar y verificar el formato de tabla en consola.
5.- Consultar por ID existente y por uno inexistente.
6.- Actualizar un estudiante y verificar que los cambios persistan al listar de nuevo.
7.- Eliminar un estudiante y verificar que ya no existe.
8.- Abrir estudiantes.txt con un editor de texto y comprobar que el formato sigue siendo “;” por campos.

Objeto	Nombre	Valor	Tipo
OP. Menú	op	Variable	Entero
Identificar del estudiante	id	Variable	Entero
Nom. estudiante	nombre	Variable	Cadena
Ape. estudiante	apellido	Variable	Cadena
Edad estudiante	edad	Variable	Entero
Contador	i	Variable	Entero
Estado de búsqueda	encontrado	Variable	Entero
Estado de eliminación	eliminado	Variable	Entero
Estructura estudiante	estudiante	Tipo definido	Estructura
Registro estudiante	e	Variable	Estructura
Archivo principal	f	Variable	Archivo
Archivo temporal	temp	Variable	Archivo
Ruta del archivo	ARCHIVO	Constante	Cadena
Límite de estudiantes	MAX	Constante	Entero
Cadena de limpieza	cad	Variable	Cadena
Función verificar ID	existeID	Función	Entero
F. Agregar estudiante	agregarEstudiante	Función	Procedimiento
F. Listar estudiantes	listarEstudiantes	Función	Procedimiento
F. Consultar estudiante	consultarEstudiante	Función	Procedimiento
F. Actualizar estudiante	actualizarEstudiante	Función	Procedimiento
F. Eliminar estudiante	eliminarEstudiante	Función	Procedimiento
F. Limpiar salto de línea	limpiarSalto	Función	Procedimiento

CÓDIGO

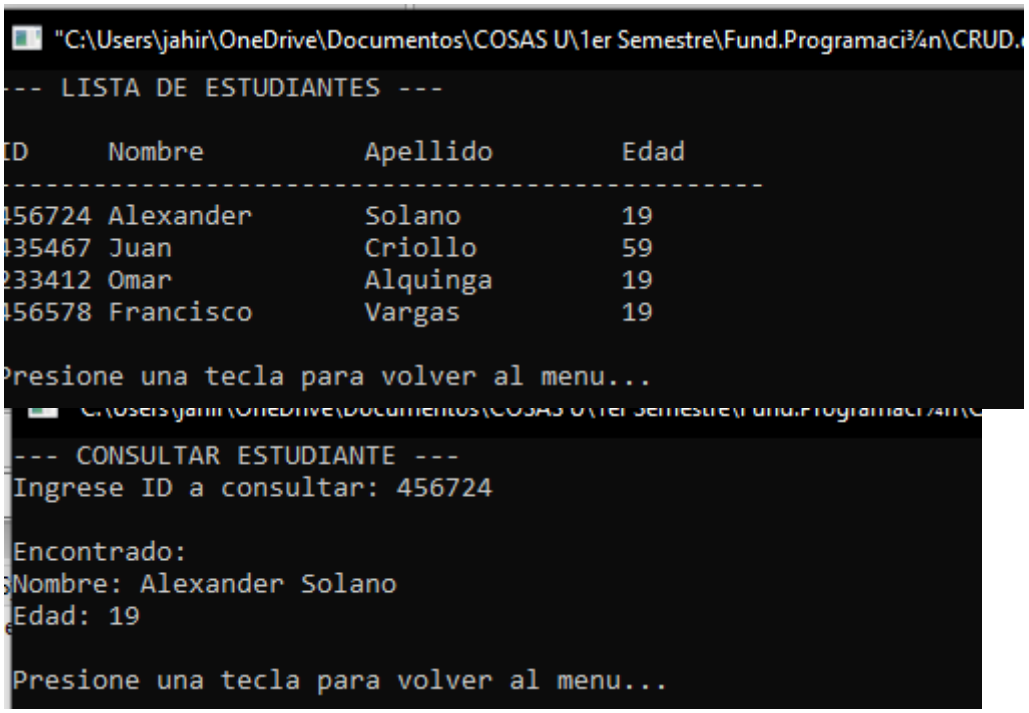
```
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <stdlib.h>

#define MAX 5
#define ARCHIVO "estudiantes.txt"

struct Estudiante {
   int id;
   char nombre[30];
   char apellido[30];
   int edad;
};

int existeID(int id);
void agregarEstudiante();
void listarEstudiantes();
void consultarEstudiante();
void actualizarEstudiante();
void eliminarEstudiante();
void limpiarSalto(char *cad);

int main() {
   int op;

   do {
      system("cls");
      printf("--- MENU ---\n");
      printf("1. Agregar estudiante\n");
      printf("2. Listar estudiantes\n");
      printf("3. Consultar por ID\n");
      printf("4. Actualizar estudiante\n");
      printf("5. Eliminar estudiante\n");
      printf("0. Salir\n");
      printf("Opcion: ");
      scanf("%d", &op);
      getchar();

      system("cls");

      switch (op) {
         case 1: agregarEstudiante(); break;
         case 2: listarEstudiantes(); break;
         case 3: consultarEstudiante(); break;
         case 4: actualizarEstudiante(); break;
         case 5: eliminarEstudiante(); break;
      }

      if (op != 0) {
         printf("\nPresione una tecla para volver al menu...");
         getch();
```

```
        }

    } while (op != 0);

    return 0;
}


void limpiarSalto(char *cad) {
    cad[strcspn(cad, "\n")] = '\0';
}


int existeID(int id) {
    FILE *f = fopen(ARCHIVO, "r");
    struct Estudiante e;

    if (f == NULL) return 0;

    while (fscanf(f, "%d;%29[^;];%29[^;];%d\n",
                &e.id, e.nombre, e.apellido, &e.edad) == 4) {
        if (e.id == id) {
            fclose(f);
            return 1;
        }
    }
    fclose(f);
    return 0;
}


void agregarEstudiante() {
    FILE *f = fopen(ARCHIVO, "a");
    struct Estudiante e;

    if (f == NULL) {
        printf("Error al abrir archivo\n");
        return;
    }

    printf("--- AGREGAR ESTUDIANTE ---\n");
    printf("ID: ");
    scanf("%d", &e.id);
    getchar();

    if (existeID(e.id)) {
        printf("ERROR: ID duplicado\n");
        fclose(f);
        return;
    }

    printf("Nombre: ");
    fgets(e.nombre, sizeof(e.nombre), stdin);
    limpiarSalto(e.nombre);
```

```
    printf("Apellido: ");
    fgets(e.apellido, sizeof(e.apellido), stdin);
    limpiarSalto(e.apellido);

    printf("Edad: ");
    scanf("%d", &e.edad);

    fprintf(f, "%d;%s;%s;%d\n",
            e.id, e.nombre, e.apellido, e.edad);

    fclose(f);
    printf("\nEstudiante agregado correctamente\n");
}


void listarEstudiantes() {
    FILE *f = fopen(ARCHIVO, "r");
    struct Estudiante e;

    printf("--- LISTA DE ESTUDIANTES ---\n");

    if (f == NULL) {
        printf("No hay estudiantes registrados\n");
        return;
    }

    printf("\n%-6s %-15s %-15s %-4s\n", "ID", "Nombre", "Apellido", "Edad");
    printf("-------------------------------------------------\n");

    while (fscanf(f, "%d;%29[^;];%29[^;];%d\n",
                  &e.id, e.nombre, e.apellido, &e.edad) == 4) {
        printf("%-6d %-15s %-15s %-4d\n",
               e.id, e.nombre, e.apellido, e.edad);
    }
    fclose(f);
}

void consultarEstudiante() {
    FILE *f = fopen(ARCHIVO, "r");
    struct Estudiante e;
    int id, encontrado = 0;

    printf("--- CONSULTAR ESTUDIANTE ---\n");

    if (f == NULL) {
        printf("Archivo no existe\n");
        return;
    }

    printf("Ingrese ID a consultar: ");
    scanf("%d", &id);

    while (fscanf(f, "%d;%29[^;];%29[^;];%d\n",
                  &e.id, e.nombre, e.apellido, &e.edad) == 4) {
        if (e.id == id) {
```

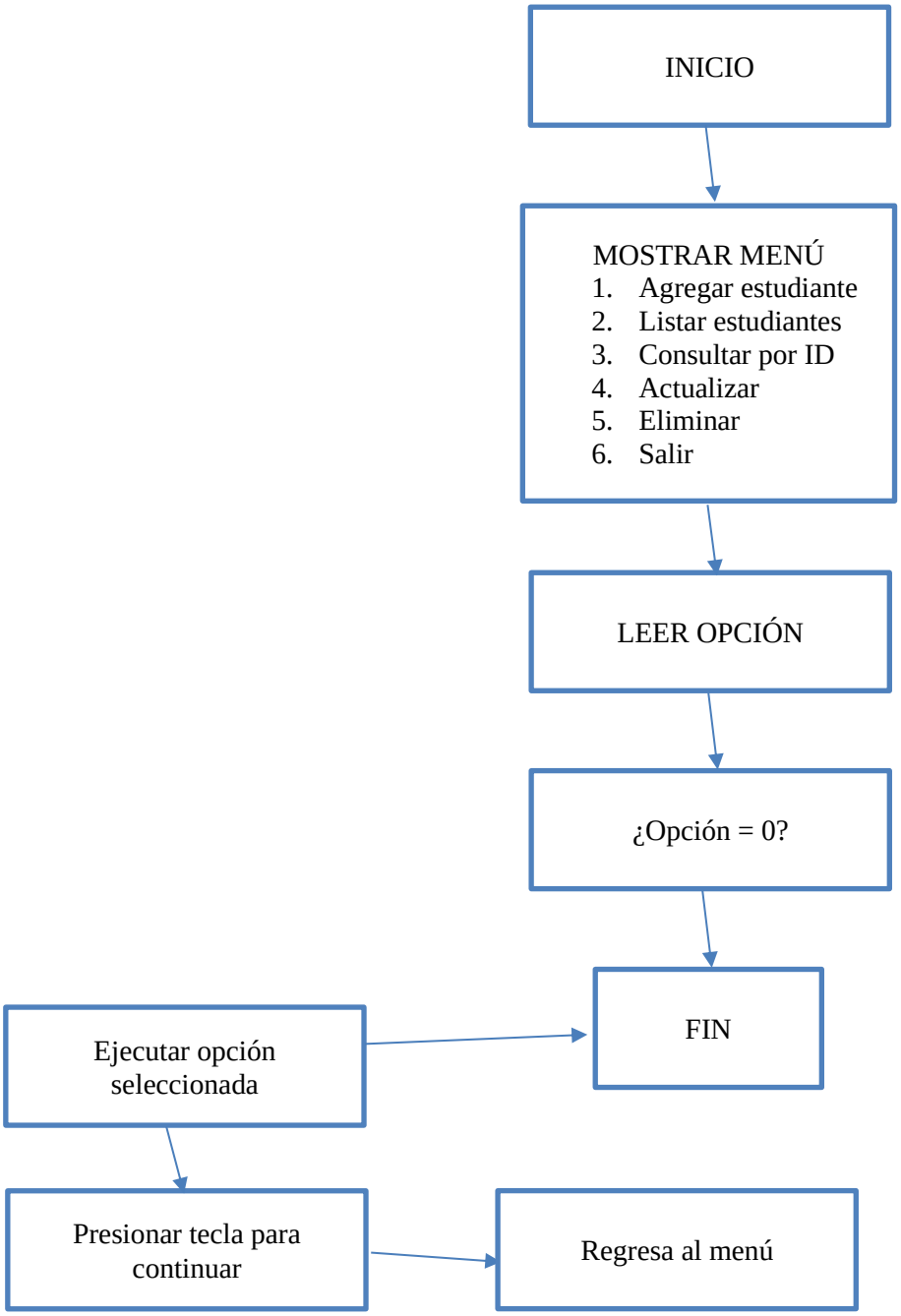
```
            printf("\nEncontrado:\n");
            printf("Nombre: %s %s\nEdad: %d\n",
                  e.nombre, e.apellido, e.edad);
            encontrado = 1;
            break;
        }
    }

    if (!encontrado)
        printf("\nID no encontrado\n");

    fclose(f);
}

void actualizarEstudiante() {
    FILE *f = fopen(ARCHIVO, "r");
    FILE *temp = fopen("temp.txt", "w");
    struct Estudiante e;
    int id, encontrado = 0;

    printf("--- ACTUALIZAR ESTUDIANTE ---\n");
    printf("ID a actualizar: ");
    scanf("%d", &id);
    getchar();

    while (fscanf(f, "%d;%29[^;];%29[^;];%d\n",
                &e.id, e.nombre, e.apellido, &e.edad) == 4) {
        if (e.id == id) {
            printf("Nuevo nombre: ");
            fgets(e.nombre, sizeof(e.nombre), stdin);
            limpiarSalto(e.nombre);

            printf("Nuevo apellido: ");
            fgets(e.apellido, sizeof(e.apellido), stdin);
            limpiarSalto(e.apellido);

            printf("Nueva edad: ");
            scanf("%d", &e.edad);
            encontrado = 1;
        }
        fprintf(temp, "%d;%s;%s;%d\n",
              e.id, e.nombre, e.apellido, e.edad);
    }

    fclose(f);
    fclose(temp);

    remove(ARCHIVO);
    rename("temp.txt", ARCHIVO);

    if (encontrado)
        printf("\nEstudiante actualizado correctamente\n");
    else
        printf("\nID no encontrado\n");
}
```

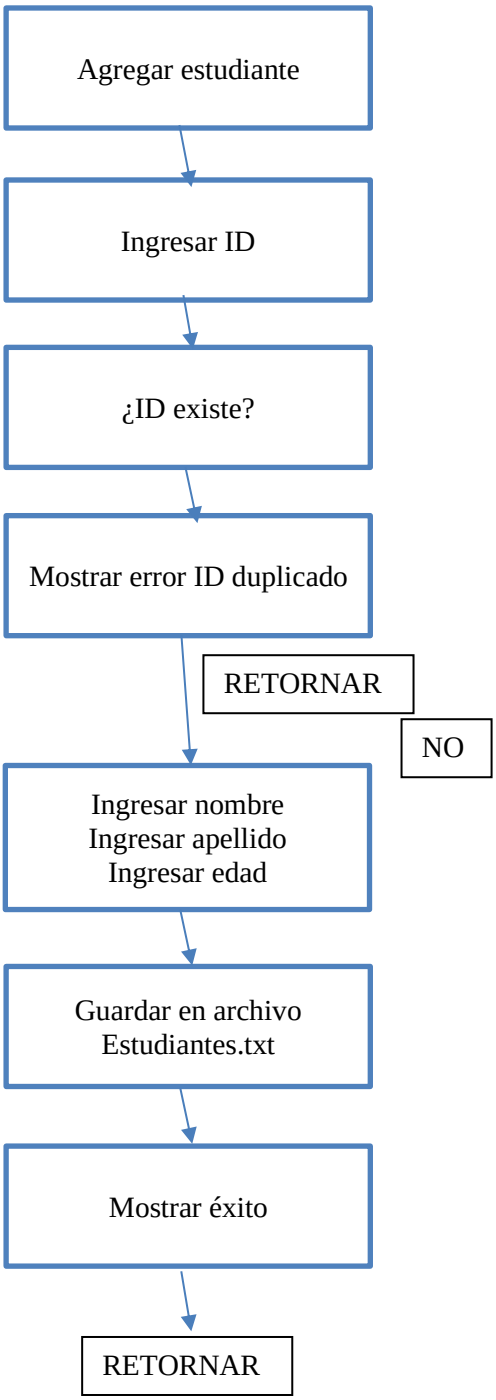
```
void eliminarEstudiante() {
   FILE *f = fopen(ARCHIVO, "r");
   FILE *temp = fopen("temp.txt", "w");
   struct Estudiante e;
   int id, eliminado = 0;

   printf("--- ELIMINAR ESTUDIANTE ---\n");
   printf("ID a eliminar: ");
   scanf("%d", &id);

   while (fscanf(f, "%d;%29[^;];%29[^;];%d\n",
            &e.id, e.nombre, e.apellido, &e.edad) == 4) {
      if (e.id != id) {
         fprintf(temp, "%d;%s;%s;%d\n",
              e.id, e.nombre, e.apellido, e.edad);
      } else {
         eliminado = 1;
      }
   }

   fclose(f);
   fclose(temp);

   remove(ARCHIVO);
   rename("temp.txt", ARCHIVO);

   if (eliminado)
      printf("\nEstudiante eliminado correctamente\n");
   else
      printf("\nID no encontrado\n");
}
```

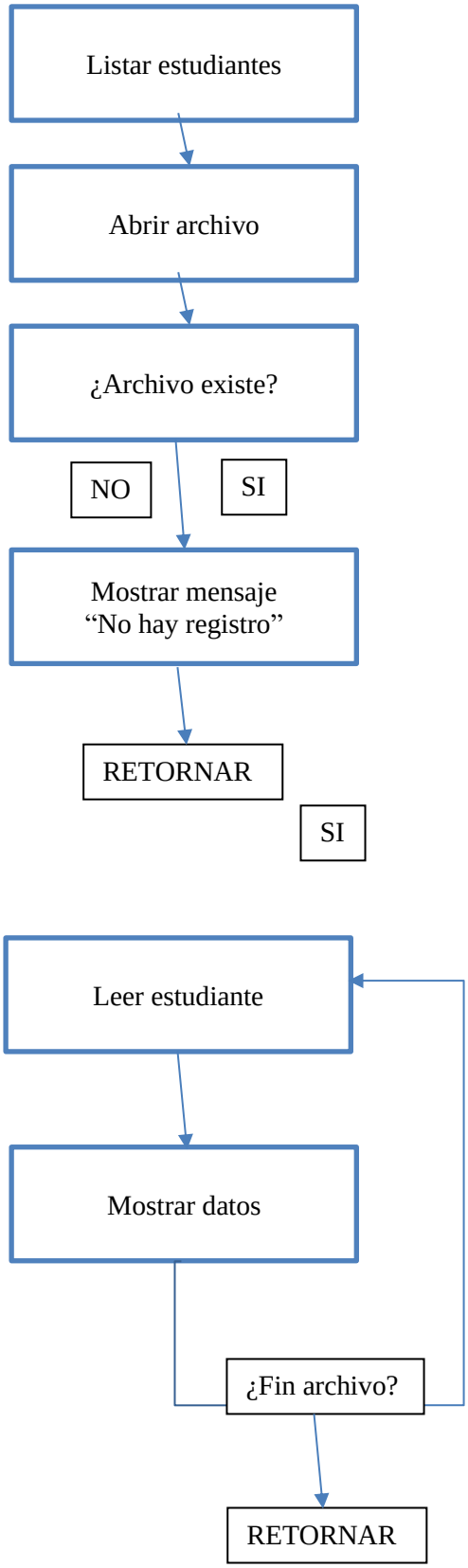


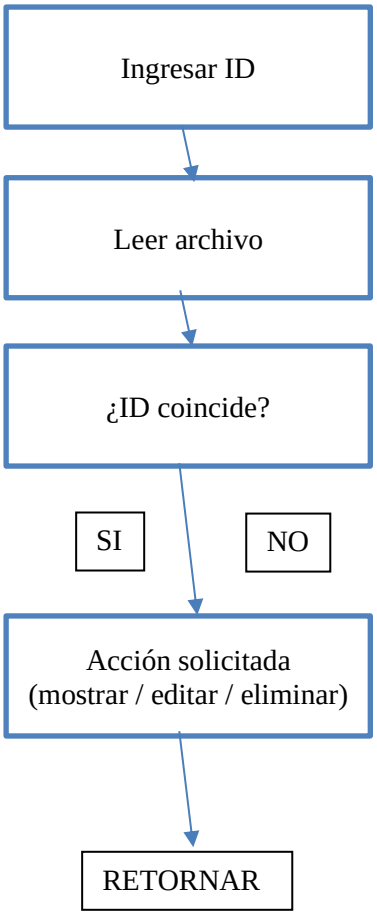
```
--- ACTUALIZAR ESTUDIANTE ---
ID a actualizar: 435467
Nuevo nombre: Alexander
Nuevo apellido: Mendoza
Nueva edad: 19

Estudiante actualizado correctamente

Presione una tecla para volver al menu...
```

```
"C:\Users\jahir\OneDrive\Documentos\COSAS U\1er Semestre\Fund.P
--- ELIMINAR ESTUDIANTE ---
ID a eliminar: 435467

Estudiante eliminado correctamente

Presione una tecla para volver al menu...
```




Agregar estudiante
Ingresar ID
¿ID existe?
Mostrar error ID duplicado
RETORNAR
NO
Ingresar nombre
Ingresar apellido
Ingresar edad
Guardar en archivo
Estudiantes.txt
Mostrar éxito
RETORNAR


Listar estudiantes
Abrir archivo
¿Archivo existe?
NO
SI
Mostrar mensaje
“No hay registro”
RETORNAR
SI
Leer estudiante
Mostrar datos
¿Fin archivo?
RETORNAR


Ingresar ID
Leer archivo
¿ID coincide?
SI
NO
Acción solicitada
(mostrar / editar / eliminar)
RETORNAR

2. CONCLUSIONES

1. Durante el desarrollo de la práctica se logró comprender el uso de las estructuras (funciones) y archivos de texto (CRUD), logrando aplicar lo enseñado en clases.
2. Mediante el uso del CRUD logramos reforzar nuestros conocimientos sobre la lógica de programación, permitiéndonos avanzar más en los nuevos temas sobre la programación.

3. RECOMENDACIONES

1. Leer y comprender bien los tipos de funciones, sabiendo cuando y como utilizarlos, ya que estos facilitan el desarrollo de algunos programas.

4. REFERENCIAS

Joyanes Aguilar, L. (2013). *Fundamentos de programación: algoritmos, estructuras de datos y objetos.*