# UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE

**Nombre:** Guerrero Sagnai Jair Adalberto

**NRC:** 29583

**Fecha:** 14/01/2026

**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.

**CÓDIGO**

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

#define ARCHIVO_EST "estudiantes.txt"

typedef struct {
    char id[20];
    char apellidos[50];
    char nombres[50];
    int edad;
} Estudiante;
```

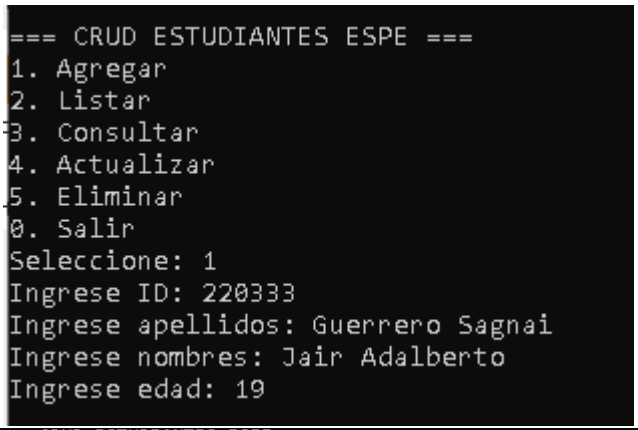
```
void limpiarNuevaLinea(char *s) {
   s[strcspn(s, "\n")] = '\0';
}

void leerCadena(const char *etiqueta, char *dest, int tam)
{
   printf("%s", etiqueta);
   fgets(dest, tam, stdin);
   limpiarNuevaLinea(dest);
}

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 invalida. Intente de nuevo.\n");
   }
}

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);
}

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;
}

void agregarEstudiante() {
```

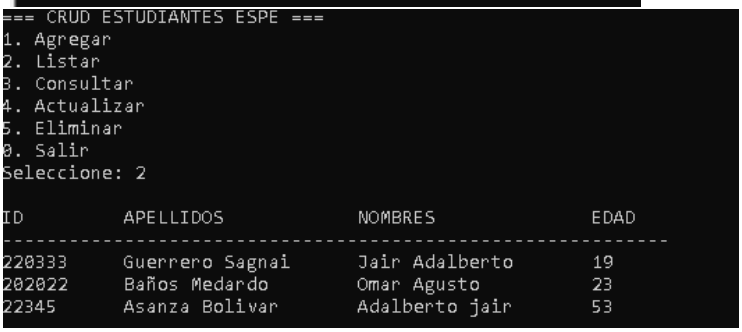
```
    Estudiante nuevo;
    leerCadena("Ingrese ID: ", nuevo.id, sizeof(nuevo.id));
    if (existeId(nuevo.id)) {
        printf("Ya existe un estudiante con ese ID.\n");
        return;
    }
    leerCadena("Ingrese apellidos: ", nuevo.apellidos,
sizeof(nuevo.apellidos));
    leerCadena("Ingrese nombres: ", nuevo.nombres,
sizeof(nuevo.nombres));
    nuevo.edad = leerEntero("Ingrese edad: ");

    FILE *f = fopen(ARCHIVO_EST, "a");
    if (f == NULL) {
        printf("Error: no se pudo 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");
}

void listarEstudiantes() {
    FILE *f = fopen(ARCHIVO_EST, "r");
    if (f == NULL) {
        printf("No hay datos aun (archivo inexistente).\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);
}

void consultarEstudiante() {
    char id[20];
    FILE *f = fopen(ARCHIVO_EST, "r");
    if (f == NULL) {
        printf("No existe el archivo.\n");
```

```
        return;
    }
    leerCadena("Ingrese ID a buscar: ", id, sizeof(id));
    char linea[200];
    Estudiante e;
    int encontrado = 0;
    while (fgets(linea, sizeof(linea), f)) {
        if (parsearEstudiante(linea, &e) && strcmp(e.id, id)
== 0) {
            printf("Encontrado: %s %s (Edad: %d)\n",
e.nombres, e.apellidos, e.edad);
            encontrado = 1;
            break;
        }
    }
    if (!encontrado) printf("No se encontro el ID.\n");
    fclose(f);
}

void actualizarEstudiante() {
    char idObj[20];
    leerCadena("Ingrese ID a actualizar: ", idObj,
sizeof(idObj));
    FILE *f = fopen(ARCHIVO_EST, "r");
    FILE *t = fopen("tmp.txt", "w");
    if (f == NULL || t == NULL) {
        if (f) fclose(f);
        if (t) fclose(t);
        printf("Error al abrir archivos.\n");
        return;
    }

    char linea[200];
    Estudiante e;
    int act = 0;
    while (fgets(linea, sizeof(linea), f)) {
        if (parsearEstudiante(linea, &e) && strcmp(e.id,
idObj) == 0) {
            leerCadena("Nuevos apellidos: ", e.apellidos,
sizeof(e.apellidos));
            leerCadena("Nuevos nombres: ", e.nombres,
sizeof(e.nombres));
            e.edad = leerEntero("Nueva edad: ");
            fprintf(t, "%s;%s;%s;%d\n", e.id, e.apellidos,
e.nombres, e.edad);
            act = 1;
        } else if (parsearEstudiante(linea, &e)) {
```

```
            fprintf(t, "%s;%s;%s;%d\n", e.id, e.apellidos,
e.nombres, e.edad);
        }
    }
    fclose(f); fclose(t);
    remove(ARCHIVO_EST); rename("tmp.txt",
ARCHIVO_EST);
    if (act) printf("Registro actualizado.\n");
    else printf("ID no encontrado.\n");
}

void eliminarEstudiante() {
    char idElim[20];
    leerCadena("Ingrese ID a eliminar: ", idElim,
sizeof(idElim));
    FILE *f = fopen(ARCHIVO_EST, "r");
    FILE *t = fopen("tmp.txt", "w");
    if (f == NULL || t == NULL) {
        if (f) fclose(f);
        if (t) fclose(t);
        printf("Error al abrir archivos.\n");
        return;
    }

    char linea[200];
    Estudiante e;
    int elim = 0;
    while (fgets(linea, sizeof(linea), f)) {
        if (parsearEstudiante(linea, &e)) {
            if (strcmp(e.id, idElim) != 0) {
                fprintf(t, "%s;%s;%s;%d\n", e.id, e.apellidos,
e.nombres, e.edad);
            } else {
                elim = 1;
            }
        }
    }
    fclose(f); fclose(t);
    remove(ARCHIVO_EST); rename("tmp.txt",
ARCHIVO_EST);
    if (elim) printf("Registro eliminado.\n");
    else printf("ID no encontrado.\n");
}

int main() {
    char opc[10];
    int op;
    do {
```

```
        printf("\n=== CRUD ESTUDIANTES ESPE
===\n");
        printf("1. Agregar\n2. Listar\n3. Consultar\n4.
Actualizar\n5. Eliminar\n0. Salir\nSeleccione: ");
        fgets(opc, sizeof(opc), stdin);
        op = atoi(opc);
        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("Opcion invalida.\n");
        }
    } while (op != 0);
    return 0;
}
```



```
=== CRUD ESTUDIANTES ESPE ===
1. Agregar
2. Listar
3. Consultar
4. Actualizar
5. Eliminar
0. Salir
Seleccione: 3
Ingrese ID a buscar: 220333
Encontrado: Jair Adalberto Guerrero Sagnai  (Edad: 19)
```

```
Seleccione: 4
Ingrese ID a actualizar: 220333
Nuevos apellidos: Paredes Bañoz
Nuevos nombres: Jonatan Antonio
Nueva edad: 30
Registro actualizado.

=== CRUD ESTUDIANTES ESPE ===
1. Agregar
2. Listar
3. Consultar
4. Actualizar
5. Eliminar
0. Salir
Seleccione: 2

ID         APELLIDOS            NOMBRES              EDAD
------------------------------------------------------------
220333     Paredes Bañoz        Jonatan Antonio      30
202022     Baños Medardo        Omar Agusto          23
22345      Asanza Bolivar       Adalberto jair       53
```

```
=== CRUD ESTUDIANTES ESPE ===
1. Agregar
2. Listar
3. Consultar
4. Actualizar
5. Eliminar
0. Salir
Seleccione: 5
Ingrese ID a eliminar: 220333
Registro eliminado.

=== CRUD ESTUDIANTES ESPE ===
1. Agregar
2. Listar
3. Consultar
4. Actualizar
5. Eliminar
0. Salir
Seleccione: 2

ID         APELLIDOS            NOMBRES              EDAD
------------------------------------------------------------
202022     Baños Medardo        Omar Agusto          23
22345      Asanza Bolivar       Adalberto jair       53
```

## 2. CONCLUSIONES

1. Se entendió el empleo de las estructuras (funciones) y los archivos de texto (CRUD) a lo largo de la práctica, lo que permitió poner en práctica lo aprendido en clase.

2. Usando el CRUD, conseguimos profundizar nuestros entendimientos acerca de la lógica de programación, lo que nos posibilita avanzar más en los nuevos temas relacionados con esta..

## 3. RECOMENDACIONES

1. Leer y entender bien las diferentes clases de funciones, para saber en qué momento y cómo usarlas, ya que estas ayudan a desarrollar ciertos programas.

## REFERENCIAS

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