UACM

Universidad Autónoma
de la Ciudad de México

NADA HUMANO ME ES AJENO

Introducción a la ingeniería de Software.

Profesor: Raúl Jara.

Proyecto Final.

Zavala Pérez Velia.

Semestre 2025-II.

Introducción.

Introducción

El presente proyecto final tiene como objetivo integrar los conocimientos adquiridos sobre arreglos/buffers, funciones, apuntadores, memoria dinámica (malloc, calloc, realloc y free), uso de estructuras (struct), y la generación de archivos, mediante la creación de un editor de texto básico capaz de producir documentos personalizados para un conjunto de personas.

El sistema permite:

- Crear y modificar un texto base.
- Registrar múltiples personas (nombre y cargo).
- Combinar automáticamente el texto con los datos de cada persona.
- Mostrar los documentos generados en pantalla.
- Crear archivos individuales por persona.
- Crear un solo archivo general con todos los documentos.
- Controlar el flujo mediante un menú con sentencia switch.

El proyecto también se integra con Git como base para ampliaciones en prácticas posteriores.

Desarrollo.

Análisis del Problema

Se necesita simular un editor de textos simple pero funcional, capaz de almacenar un mensaje y adaptarlo para diferentes destinatarios.

Los requisitos incluyen:

a) Manejo de texto (buffer y cadenas dinámicas)
- Capturar un texto base.
- Modificarlo tantas veces como el usuario desee.
- Gestionar dinámicamente la memoria necesaria.

b) Memoria dinámica obligatoria
- malloc() → para inicializar el texto y cadenas.
- calloc() → para inicializar la lista inicial de personas.
- realloc() → modificar memoria del texto o del arreglo de personas.
- free() → liberar todos los recursos al final.

c) Flujo principal con switch

El programa debe presentar un menú con opciones:

- Crear texto
- Modificar texto
- Capturar personas
- Mostrar documentos en pantalla
- Generar archivos individuales
- Generar archivo único
- Salida + liberar memoria

d) Generación de archivos

- Individuales: documento_1.txt, documento_2.txt, etc.
- Archivo general: documento_general.txt.

Solución Propuesta.

La solución se basa en dividir el programa en funciones especializadas:

crearTexto() malloc Captura y guarda el mensaje base

modificarTexto() realloc Permite reescribir el texto base

capturarPersonas() calloc + malloc Registra nombre/cargo para N personas

mostrarDocumentos() Muestra mensajes personalizados

generarArchivos() Genera un archivo por persona

generarArchivoUnico() Genera un único documento global

liberarMemoria() free Limpia toda la memoria dinámica

Código fuente.

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

// ------------------------------------------
//              STRUCT PERSONA
```

```
// ------------------------------------------------
typedef struct {
    char *nombre;
    char *cargo;
} Persona;

// Variables globales
char *texto = NULL;          // texto base
Persona *personas = NULL;    // arreglo dinámico de personas
int totalPersonas = 0;

// ------------------------------------------------
//       CREAR TEXTO BASE (USA malloc)
// ------------------------------------------------
void crearTexto() {
    char buffer[500];

    printf("\nEscribe el texto base:\n> ");
    fflush(stdin);
    fgets(buffer, sizeof(buffer), stdin);

    texto = (char *) malloc(strlen(buffer) + 1);

    if (texto == NULL) {
        printf("Error al asignar memoria.\n");
        return;
    }
```

```
    strcpy(texto, buffer);
    printf("Texto creado correctamente.\n");
}

// ------------------------------------------------
//      MODIFICAR TEXTO (USA realloc)
// ------------------------------------------------
void modificarTexto() {
    if (texto == NULL) {
        printf("\nPrimero debes crear el texto.\n");
        return;
    }

    char buffer[500];

    printf("\nIngresa el nuevo texto:\n> ");
    fflush(stdin);
    fgets(buffer, sizeof(buffer), stdin);

    texto = (char *) realloc(texto, strlen(buffer) + 1);

    if (texto == NULL) {
        printf("Error al reasignar memoria.\n");
        return;
    }

    strcpy(texto, buffer);
    printf("Texto modificado correctamente.\n");
```

```
}

// ------------------------------------------------
//     CAPTURAR PERSONAS (usa calloc + malloc)
// ------------------------------------------------
void capturarPersonas() {
    printf("\nCuantas personas deseas registrar?: ");
    scanf("%d", &totalPersonas);
    getchar();

    personas = (Persona *) calloc(totalPersonas, sizeof(Persona));

    if (personas == NULL) {
        printf("Error al asignar memoria.\n");
        return;
    }

    for (int i = 0; i < totalPersonas; i++) {

        char bufferNombre[100];
        char bufferCargo[100];

        printf("\nPersona %d\n", i + 1);

        printf("Nombre: ");
        fgets(bufferNombre, sizeof(bufferNombre), stdin);
        bufferNombre[strcspn(bufferNombre, "\n")] = 0;
```

```
        printf("Cargo: ");
        fgets(bufferCargo, sizeof(bufferCargo), stdin);
        bufferCargo[strcspn(bufferCargo, "\n")] = 0;

        personas[i].nombre = (char *) malloc(strlen(bufferNombre) + 1);
        personas[i].cargo  = (char *) malloc(strlen(bufferCargo) + 1);

        strcpy(personas[i].nombre, bufferNombre);
        strcpy(personas[i].cargo, bufferCargo);
    }

    printf("\nPersonas registradas.\n");
}

// ------------------------------------------------
//     MOSTRAR DOCUMENTOS EN PANTALLA
// ------------------------------------------------
void mostrarDocumentos() {
    if (texto == NULL || personas == NULL) {
        printf("\nDebes capturar texto y personas primero.\n");
        return;
    }

    printf("\n--- DOCUMENTOS GENERADOS ---\n");

    for (int i = 0; i < totalPersonas; i++) {

        printf("\n========================================\n");
```

```
        printf(" Documento para: %s (%s)\n", personas[i].nombre, personas[i].cargo);
        printf("=======================================\n");
        printf("%s\n", texto);
    }
}

// ------------------------------------------------
//  GENERAR ARCHIVOS INDIVIDUALES
// ------------------------------------------------
void generarArchivos() {
    if (texto == NULL || personas == NULL) {
        printf("\nDebes capturar texto y personas primero.\n");
        return;
    }

    for (int i = 0; i < totalPersonas; i++) {

        char nombreArchivo[120];
        sprintf(nombreArchivo, "documento_%d.txt", i + 1);

        FILE *archivo = fopen(nombreArchivo, "w");

        if (!archivo) {
            printf("No se pudo abrir el archivo %s\n", nombreArchivo);
            continue;
        }

        fprintf(archivo, "Documento para: %s (%s)\n", personas[i].nombre,
personas[i].cargo);
```

```
        fprintf(archivo, "----------------------------------------\n");
        fprintf(archivo, "%s\n", texto);

        fclose(archivo);
    }

    printf("\nArchivos generados exitosamente.\n");
}

// ------------------------------------------------
//  GENERAR ARCHIVO ÚNICO CON TODOS LOS DOCUMENTOS
// ------------------------------------------------
void generarArchivoUnico() {
    if (texto == NULL || personas == NULL) {
        printf("\nDebes capturar texto y personas primero.\n");
        return;
    }

    FILE *archivo = fopen("documento_general.txt", "w");

    if (!archivo) {
        printf("No se pudo crear el archivo general.\n");
        return;
    }

    fprintf(archivo, "========== DOCUMENTO GENERAL ==========\n\n");

    for (int i = 0; i < totalPersonas; i++) {
```

```
        fprintf(archivo, "----------------------------------------\n");
        fprintf(archivo, " Documento para: %s (%s)\n", personas[i].nombre, personas[i].cargo);
        fprintf(archivo, "----------------------------------------\n");
        fprintf(archivo, "%s\n", texto);
    }

    fclose(archivo);

    printf("\nArchivo unico 'documento_general.txt' generado correctamente.\n");
}

// ------------------------------------------------
//        LIBERAR MEMORIA (free)
// ------------------------------------------------
void liberarMemoria() {

    if (texto != NULL)
        free(texto);

    if (personas != NULL) {
        for (int i = 0; i < totalPersonas; i++) {
            free(personas[i].nombre);
            free(personas[i].cargo);
        }
        free(personas);
    }
}
```

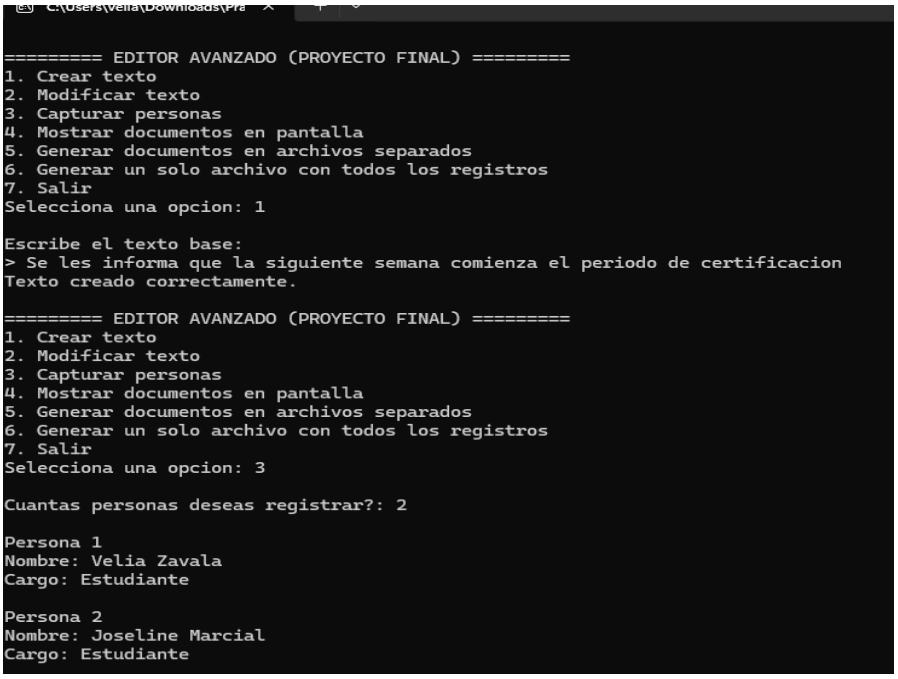
```
// ------------------------------------------------
//                MENÚ (CON SWITCH)
// ------------------------------------------------
int main() {

    int opcion;

    do {
        printf("\n========= EDITOR AVANZADO (PROYECTO FINAL) =========\n");
        printf("1. Crear texto\n");
        printf("2. Modificar texto\n");
        printf("3. Capturar personas\n");
        printf("4. Mostrar documentos en pantalla\n");
        printf("5. Generar documentos en archivos separados\n");
        printf("6. Generar un solo archivo con todos los registros\n");
        printf("7. Salir\n");
        printf("Selecciona una opcion: ");

        scanf("%d", &opcion);
        getchar();

        switch (opcion) {

            case 1: crearTexto(); break;
            case 2: modificarTexto(); break;
            case 3: capturarPersonas(); break;
            case 4: mostrarDocumentos(); break;
```

```
            case 5: generarArchivos(); break;
            case 6: generarArchivoUnico(); break;

            case 7:
                printf("Saliendo...\n");
                liberarMemoria();
                break;

            default:
                printf("Opcion no valida.\n");
        }

    } while (opcion != 7);

    return 0;
}
```

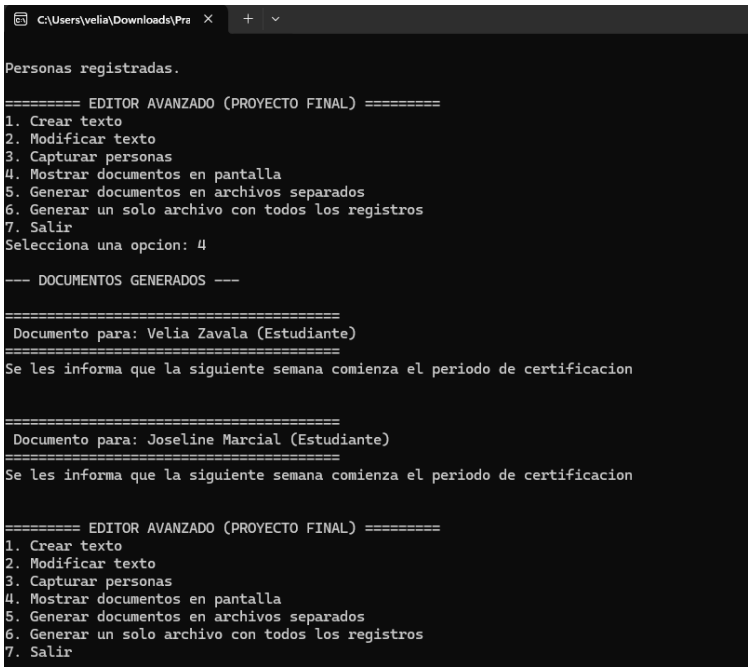
Tiempos estimados vs tiempos reales.

Actividad	Tiempo estimado	Tiempo real
Análisis	**40 minutos**	**30 minutos**
Diseño	**1 hora**	**1 hora**
Programación	**3 horas**	**90 minutos**
Pruebas	**1 hora**	**40 minutos**
Documentación	**2 horas**	**1 hora**
Total	**7 horas 40 minutos**	**4 horas 40 minutos**

Evidencia.



```
========= EDITOR AVANZADO (PROYECTO FINAL) =========
1. Crear texto
2. Modificar texto
3. Capturar personas
4. Mostrar documentos en pantalla
5. Generar documentos en archivos separados
6. Generar un solo archivo con todos los registros
7. Salir
Selecciona una opcion: 1

Escribe el texto base:
> Se les informa que la siguiente semana comienza el periodo de certificacion
Texto creado correctamente.

========= EDITOR AVANZADO (PROYECTO FINAL) =========
1. Crear texto
2. Modificar texto
3. Capturar personas
4. Mostrar documentos en pantalla
5. Generar documentos en archivos separados
6. Generar un solo archivo con todos los registros
7. Salir
Selecciona una opcion: 3

Cuantas personas deseas registrar?: 2

Persona 1
Nombre: Velia Zavala
Cargo: Estudiante

Persona 2
Nombre: Joseline Marcial
Cargo: Estudiante
```



```
C:\Users\velia\Downloads\Pra

Personas registradas.

========= EDITOR AVANZADO (PROYECTO FINAL) =========
1. Crear texto
2. Modificar texto
3. Capturar personas
4. Mostrar documentos en pantalla
5. Generar documentos en archivos separados
6. Generar un solo archivo con todos los registros
7. Salir
Selecciona una opcion: 4

--- DOCUMENTOS GENERADOS ---

======================================
 Documento para: Velia Zavala (Estudiante)
======================================
Se les informa que la siguiente semana comienza el periodo de certificacion


======================================
 Documento para: Joseline Marcial (Estudiante)
======================================
Se les informa que la siguiente semana comienza el periodo de certificacion


========= EDITOR AVANZADO (PROYECTO FINAL) =========
1. Crear texto
2. Modificar texto
3. Capturar personas
4. Mostrar documentos en pantalla
5. Generar documentos en archivos separados
6. Generar un solo archivo con todos los registros
7. Salir
```

```
File    Edit    View                                                    H1

========== DOCUMENTO GENERAL ==========

----------------------------------------
 Documento para: Velia Zavala (Estudiante)
----------------------------------------
Se les informa que la siguiente semana comienza el periodo de certificacion

----------------------------------------
 Documento para: Joseline Marcial (Estudiante)
----------------------------------------
Se les informa que la siguiente semana comienza el periodo de certificacion
```

Conclusiones

El proyecto final permitió aplicar de manera integral los conceptos fundamentales de programación en C: memoria dinámica, estructuras, buffers, apuntadores y archivos. Se logró un sistema completamente funcional capaz de generar documentos personalizados tanto en pantalla como en archivos externos.