Universidad Autónoma de la Ciudad de México
Nada humano me es ajeno

Licenciatura en Ingeniería de Software
UACM San Lorenzo Tezonco
introducción a la ingeniería en Software

Practica 5 Creación de un editor de texto básico, Uso de

Arreglos/Buffer, Funciones, apuntadores y memoria dinámica y Uso de Struc

Ariel Sofia Lopez Amaya
Matricula: 21-003-1193

Introducción

El presente documento tiene como finalidad describir el desarrollo de un programa en lenguaje C que integra dos funcionalidades principales:

1. la gestión de información mediante estructuras (struct),
2. la simulación de un editor de texto básico con memoria dinámica.

El sistema permite capturar un texto base para comunicar un mensaje, modificarlo cuando sea necesario, registrar datos de personas con nombre y cargo, y finalmente generar documentos personalizados combinando automáticamente el texto con la información de cada persona.

Este proyecto busca reforzar el uso de arreglos, punteros, memoria dinámica, estructuras, funciones y menús mediante sentencias *switch*, siguiendo la propuesta de prácticas previas. Asimismo, se incluyen los códigos como anexos y una explicación del funcionamiento, junto con la evidencia del correcto desempeño del programa y los tiempos estimados y reales del desarrollo.

Desarrollo

Análisis del problema

La práctica requiere crear un programa que:

- Simule un editor de texto básico.
- Genere mensajes personalizados dirigidos a un conjunto de personas.
- Utilice:
    - Arreglos o buffers para capturar cadenas.
    - Apuntadores.
    - Funciones de memoria dinámica (malloc, realloc).
    - Estructuras (struct) para almacenar nombre y cargo.
    - Un menú con *switch* para controlar las funciones.
- Permita:
    - Crear un texto.
    - Modificarlo.

- Registrar personas.
- Combinar y mostrar mensajes personalizados en pantalla.

Problemas identificados:

- El programa original no conservaba el texto si se creaba otro nuevo.
- Era necesario evitar que el texto se sobrescribiera a menos que se usara “Modificar texto”.
- La combinación texto–persona debía mostrarse claramente como documento personalizado.

Estos puntos fueron resueltos y reflejados en el código final.

Solución propuesta

Para cumplir los requisitos, se diseñó un programa modular con las siguientes características:

a) Uso de struct Persona

Se definió una estructura con dos campos:

- nombre
- cargo

Se almacena un máximo de 50 personas.

b) Uso de memoria dinámica

El texto base se guarda mediante un apuntador char *textoBase, usando:

- malloc() para crearlo,
- realloc() solo cuando se modifica,
- free() al finalizar.

Esto cumple con el requisito de utilizar memoria dinámica para cadenas.

c) Buffer para capturar texto

Se usa un arreglo local buffer[500] para recibir la entrada antes de guardarla dinámicamente.

d) Menú con switch

Las opciones incluyen:

1. Crear texto base.
2. Modificar texto.
3. Capturar personas.
4. Mostrar documento personalizado.

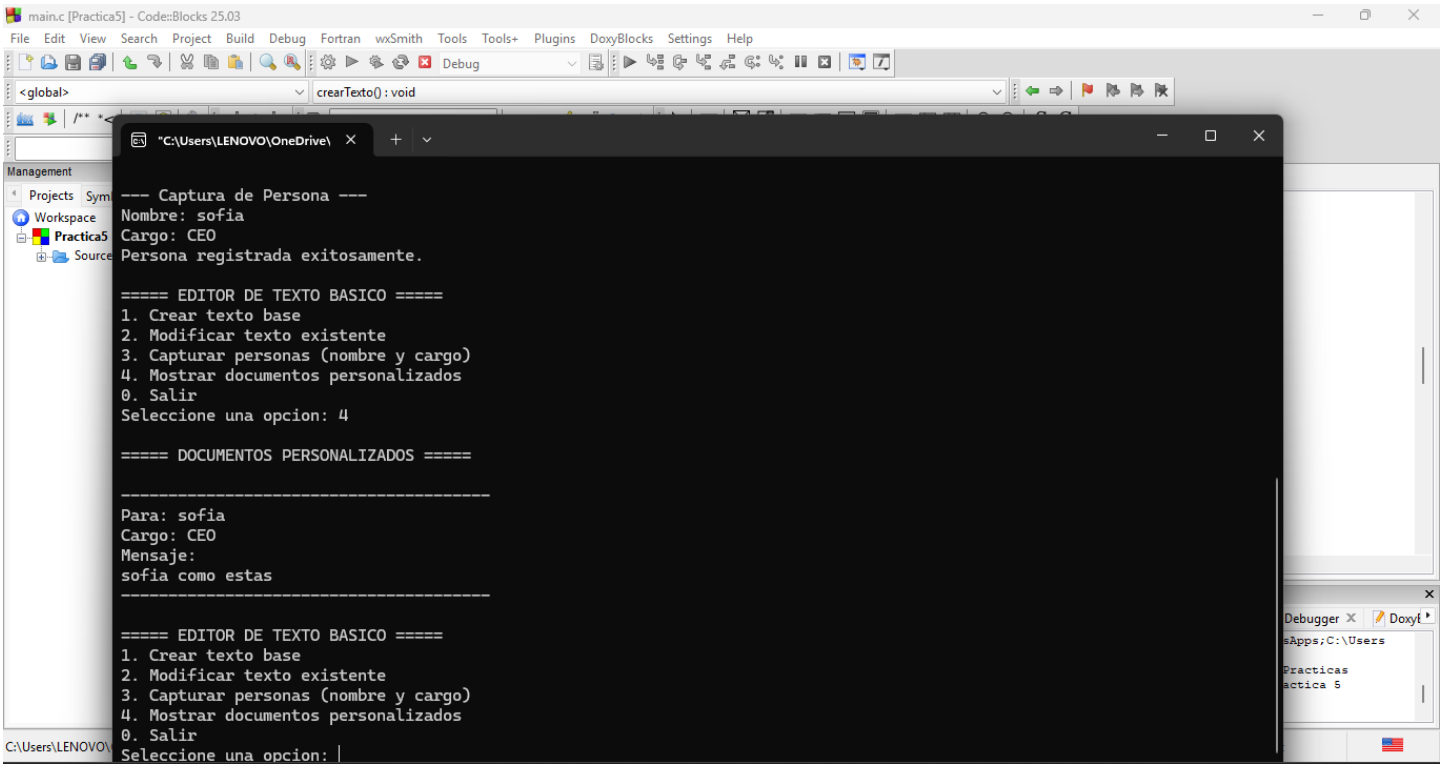
e) Validaciones

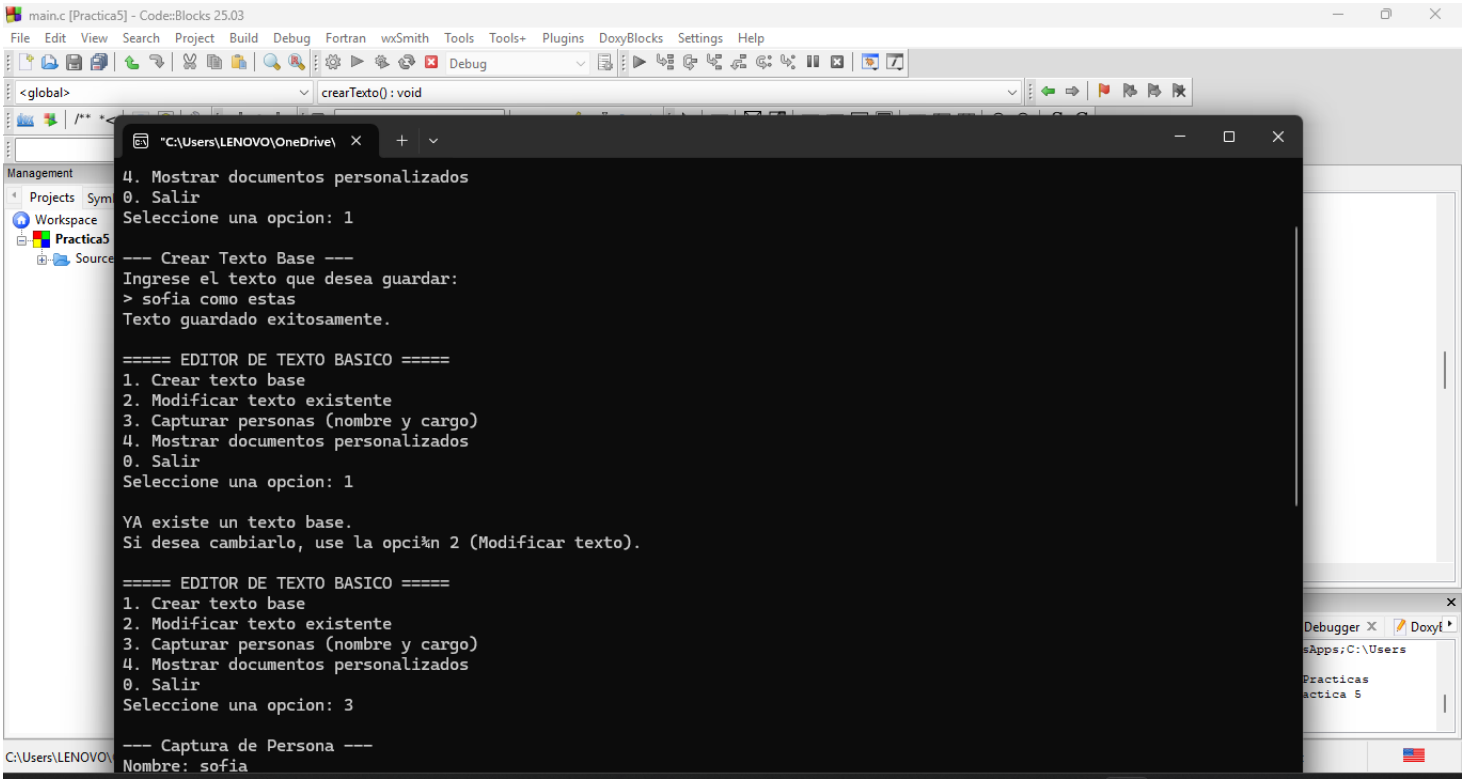
- El texto solo puede crearse si no existe.
- Si se quiere cambiar, debe usarse “Modificar texto”.
- No se pueden agregar más personas si se llega al máximo.

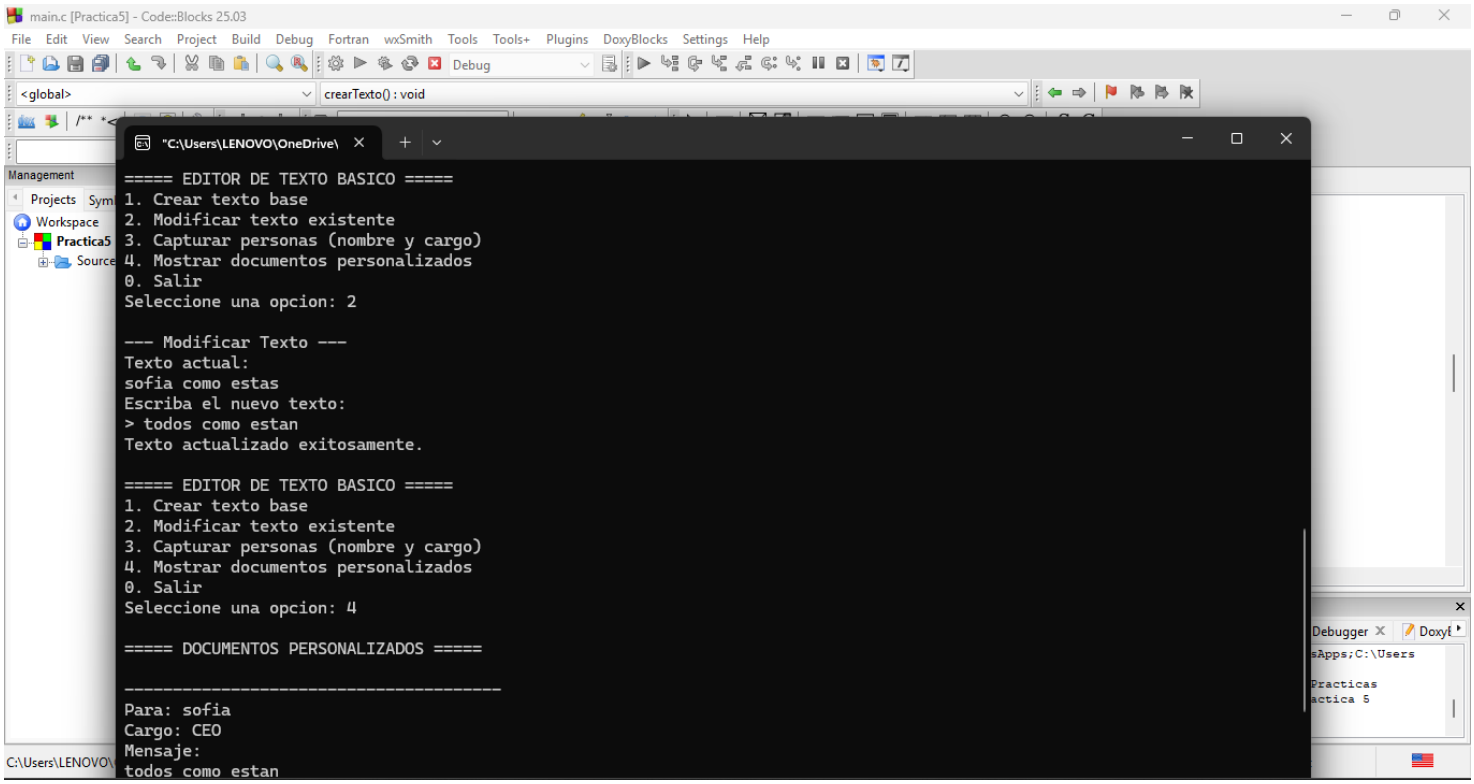
Tiempo estimado y tiempo real

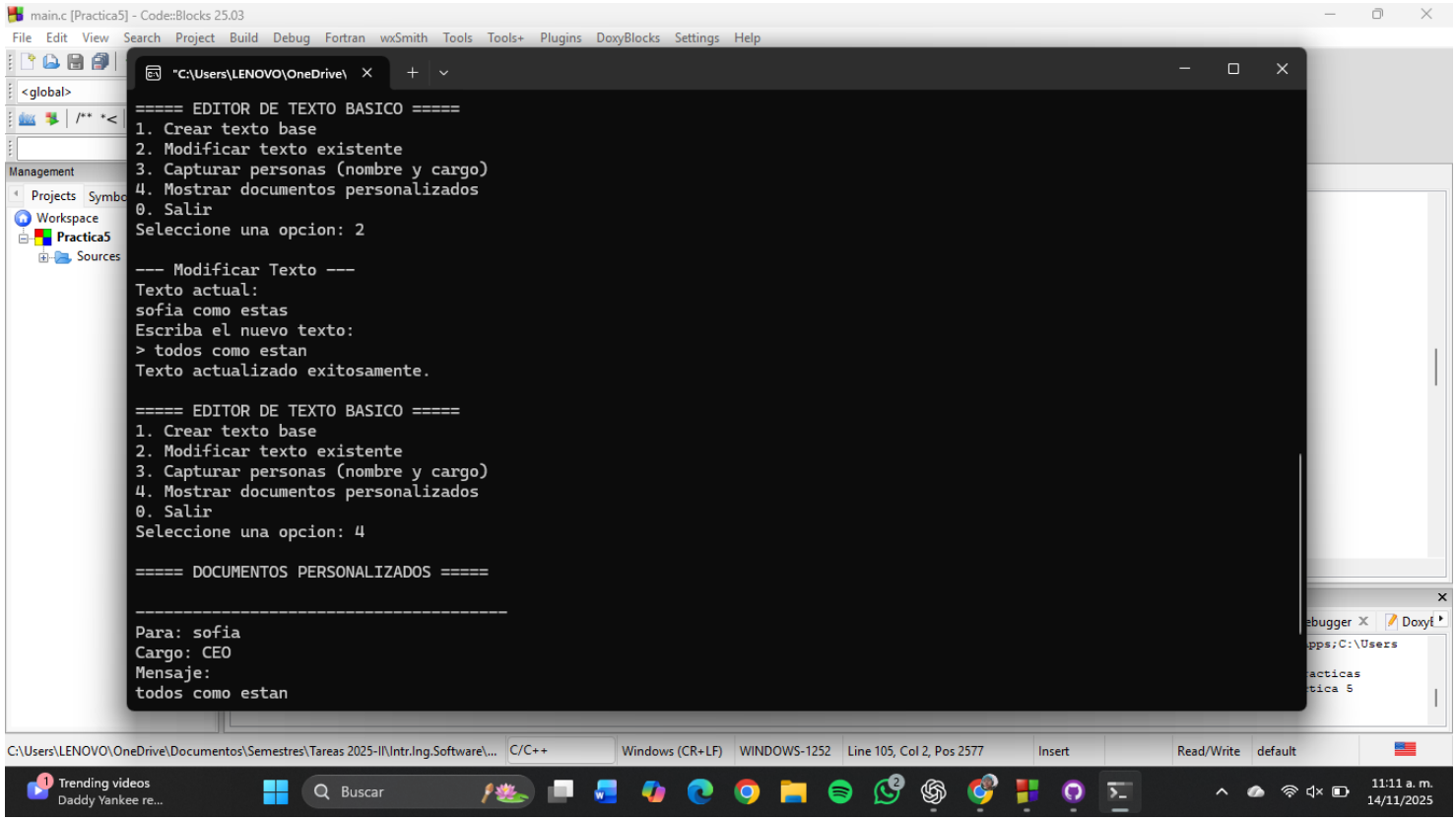
Actividad	**Tiempo estimado**	**Tiempo real**
Análisis del problema	20 min	25 min
Diseño del programa	25 min	30 min
Programación inicial	45 min	2 dias
Integración del editor de texto	40 min	45 min
Pruebas y correcciones	30 min	20 min
Documentación	20 min	1 hora

Se concluye que el tiempo real fue ligeramente superior al estimado, principalmente debido a ajustes de memoria dinámica y validaciones adicionales.


main.c [Practica5] - Code::Blocks 25.03
"C:\Users\LENOVO\OneDrive\
--- Captura de Persona ---
Nombre: sofia
Cargo: CEO
Persona registrada exitosamente.
===== EDITOR DE TEXTO BASICO =====
1. Crear texto base
2. Modificar texto existente
3. Capturar personas (nombre y cargo)
4. Mostrar documentos personalizados
0. Salir
Seleccione una opcion: 4
===== DOCUMENTOS PERSONALIZADOS =====
Para: sofia
Cargo: CEO
Mensaje:
sofia como estas
===== EDITOR DE TEXTO BASICO =====
1. Crear texto base
2. Modificar texto existente
3. Capturar personas (nombre y cargo)
4. Mostrar documentos personalizados
0. Salir
Seleccione una opcion:


main.c [Practica5] - Code::Blocks 25.03
"C:\Users\LENOVO\OneDrive\
4. Mostrar documentos personalizados
0. Salir
Seleccione una opcion: 1
--- Crear Texto Base ---
Ingrese el texto que desea guardar:
> sofia como estas
Texto guardado exitosamente.
===== EDITOR DE TEXTO BASICO =====
1. Crear texto base
2. Modificar texto existente
3. Capturar personas (nombre y cargo)
4. Mostrar documentos personalizados
0. Salir
Seleccione una opcion: 1
YA existe un texto base.
Si desea cambiarlo, use la opci%n 2 (Modificar texto).
===== EDITOR DE TEXTO BASICO =====
1. Crear texto base
2. Modificar texto existente
3. Capturar personas (nombre y cargo)
4. Mostrar documentos personalizados
0. Salir
Seleccione una opcion: 3
--- Captura de Persona ---
Nombre: sofia


main.c [Practica5] - Code::Blocks 25.03
"C:\Users\LENOVO\OneDrive\
===== EDITOR DE TEXTO BASICO =====
1. Crear texto base
2. Modificar texto existente
3. Capturar personas (nombre y cargo)
4. Mostrar documentos personalizados
0. Salir
Seleccione una opcion: 2
--- Modificar Texto ---
Texto actual:
sofia como estas
Escriba el nuevo texto:
> todos como estan
Texto actualizado exitosamente.
===== EDITOR DE TEXTO BASICO =====
1. Crear texto base
2. Modificar texto existente
3. Capturar personas (nombre y cargo)
4. Mostrar documentos personalizados
0. Salir
Seleccione una opcion: 4
===== DOCUMENTOS PERSONALIZADOS =====
Para: sofia
Cargo: CEO
Mensaje:
todos como estan


main.c [Practica5] - Code::Blocks 25.03
"C:\Users\LENOVO\OneDrive\
===== EDITOR DE TEXTO BASICO =====
1. Crear texto base
2. Modificar texto existente
3. Capturar personas (nombre y cargo)
4. Mostrar documentos personalizados
0. Salir
Seleccione una opcion: 2
--- Modificar Texto ---
Texto actual:
sofia como estas
Escriba el nuevo texto:
> todos como estan
Texto actualizado exitosamente.
===== EDITOR DE TEXTO BASICO =====
1. Crear texto base
2. Modificar texto existente
3. Capturar personas (nombre y cargo)
4. Mostrar documentos personalizados
0. Salir
Seleccione una opcion: 4
===== DOCUMENTOS PERSONALIZADOS =====
Para: sofia
Cargo: CEO
Mensaje:
todos como estan

CODIGO:

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

#define MAX_PERSONAS 50

// -------------------------
//  STRUCT PERSONA
// -------------------------
typedef struct {
    char nombre[50];
    char cargo[50];
} Persona;

// -------------------------
// VARIABLES GLOBALES
// -------------------------
Persona personas[MAX_PERSONAS];
int contadorPersonas = 0;

char *textoBase = NULL;    // Mensaje dinámico
int textoLong = 0;

// -------------------------
// PROTOTIPOS
// -------------------------
void menuEditor();
void crearTexto();
```

```
void modificarTexto();
void capturarPersonas();
void mostrarDocumentos();


// -------------------------
// PROGRAMA PRINCIPAL
// -------------------------
int main() {
    menuEditor();

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

    return 0;
}

// -------------------------
// MENÚ PRINCIPAL DEL EDITOR
// -------------------------
void menuEditor() {
    int opc;

    do {
        printf("\n===== EDITOR DE TEXTO BASICO =====\n");
        printf("1. Crear texto base\n");
        printf("2. Modificar texto existente\n");
        printf("3. Capturar personas (nombre y cargo)\n");
        printf("4. Mostrar documentos personalizados\n");
        printf("0. Salir\n");
```

```
        printf("Seleccione una opcion: ");
        scanf("%d", &opc);
        getchar();

        switch (opc) {
            case 1: crearTexto(); break;
            case 2: modificarTexto(); break;
            case 3: capturarPersonas(); break;
            case 4: mostrarDocumentos(); break;
            case 0: printf("Saliendo...\n"); break;
            default: printf("Opcion NO valida.\n");
        }

    } while (opc != 0);
}


// -------------------------
// 1. CREAR TEXTO (memoria dinámica)
// -------------------------
void crearTexto() {
    // Si ya existe texto, NO debe reemplazarlo
    if (textoBase != NULL) {
        printf("\nYA existe un texto base.\n");
        printf("Si desea cambiarlo, use la opción 2 (Modificar texto).\n");
        return;
    }

    char buffer[500];
```

```
    printf("\n--- Crear Texto Base ---\n");
    printf("Ingrese el texto que desea guardar:\n> ");
    fgets(buffer, sizeof(buffer), stdin);
    buffer[strcspn(buffer, "\n")] = 0;

    textoLong = strlen(buffer) + 1;
    textoBase = (char*)malloc(textoLong);

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

    strcpy(textoBase, buffer);
    printf("Texto guardado exitosamente.\n");
}

// -------------------------
// 2. MODIFICAR TEXTO
// -------------------------
void modificarTexto() {
    if (textoBase == NULL) {
        printf("No hay texto aún. Use opción 1 primero.\n");
        return;
    }

    char buffer[500];
    printf("\n--- Modificar Texto ---\n");
    printf("Texto actual:\n%s\n", textoBase);
    printf("Escriba el nuevo texto:\n> ");
```

```
    fgets(buffer, sizeof(buffer), stdin);
    buffer[strcspn(buffer, "\n")] = 0;

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

    strcpy(textoBase, buffer);
    printf("Texto actualizado exitosamente.\n");
}



// -------------------------
// 3. CAPTURAR PERSONAS
// -------------------------
void capturarPersonas() {
    if (contadorPersonas >= MAX_PERSONAS) {
        printf("No se pueden agregar más personas.\n");
        return;
    }

    printf("\n--- Captura de Persona ---\n");

    printf("Nombre: ");
    fgets(personas[contadorPersonas].nombre, 50, stdin);
    personas[contadorPersonas].nombre[strcspn(personas[contadorPersonas].nombre, "\n")]
= 0;

    printf("Cargo: ");
    fgets(personas[contadorPersonas].cargo, 50, stdin);
    personas[contadorPersonas].cargo[strcspn(personas[contadorPersonas].cargo, "\n")] = 0;
```

```
    contadorPersonas++;

    printf("Persona registrada exitosamente.\n");
}


// -------------------------
// 4. MOSTRAR DOCUMENTO FINAL
// -------------------------
void mostrarDocumentos() {
    if (textoBase == NULL) {
        printf("No existe texto base.\n");
        return;
    }
    if (contadorPersonas == 0) {
        printf("No hay personas registradas.\n");
        return;
    }

    printf("\n===== DOCUMENTOS PERSONALIZADOS =====\n");

    for (int i = 0; i < contadorPersonas; i++) {
        printf("\n---------------------------------------\n");
        printf("Para: %s\nCargo: %s\n",
            personas[i].nombre, personas[i].cargo);
        printf("Mensaje:\n%s\n", textoBase);
        printf("---------------------------------------\n");
    }
}
```

Conclusiones

El desarrollo de este programa permitió integrar múltiples temas fundamentales de programación estructurada en C: manejo de cadenas, arreglos, estructuras, punteros, memoria dinámica y control mediante menús.

Se logró implementar un editor de texto básico que cumple exactamente con los requisitos solicitados: crear y modificar un texto, registrar personas con nombre y cargo, y generar documentos personalizados combinando el mensaje con los datos capturados.

El uso de memoria dinámica garantizó flexibilidad en el manejo del contenido del texto, mientras que la estructura de personas permitió asociar adecuadamente los datos de cada individuo con el mensaje. El menú mediante *switch* proporciona un control intuitivo y organizado del flujo del programa.

El programa fue probado y validado correctamente, demostrando su funcionamiento estable y la integración adecuada de todos los conceptos solicitados en la práctica. Además, el proceso permitió reforzar habilidades en modularidad, documentación y depuración.