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 3 Creación de un editor de texto básico, Uso de Arreglos/Buffer, Funciones, apuntadores y memoria dinámica

Ariel Sofia Lopez Amaya
Matricula: 21-003-1193

Introducción

En la programación en lenguaje C, la manipulación de cadenas de texto y el uso de memoria dinámica constituyen conceptos fundamentales para la creación de aplicaciones eficientes y flexibles. Este proyecto tiene como objetivo desarrollar un **simulador de editor de texto básico** que permita **ingresar texto, buscar palabras o frases específicas y reemplazarlas por otras**, empleando **arreglos dinámicos (buffers)** y el uso de las funciones de administración de memoria dinámica: malloc(), realloc() y free().

El programa está diseñado para funcionar desde consola, permitiendo al usuario ingresar líneas de texto hasta que escriba la palabra "FIN". Posteriormente, el sistema solicita el texto a buscar y el texto de reemplazo, mostrando el resultado final después de aplicar las modificaciones. Este ejercicio integra los conocimientos de manejo de **punteros, cadenas, funciones y memoria dinámica**, reforzando las competencias para el desarrollo de programas más complejos en C.

Desarrollo

Análisis del Problema

En los editores de texto es común la necesidad de **buscar una palabra o frase dentro del contenido** y **reemplazarla por otra**, una tarea que implica recorrer la cadena, identificar coincidencias y reconstruir el texto.

Sin embargo, en C el manejo de cadenas es limitado, ya que estas se manejan como arreglos estáticos de caracteres, lo que restringe su tamaño. Por ello, el uso de **memoria dinámica** permite construir un texto cuyo tamaño se ajusta al contenido real ingresado por el usuario, optimizando el uso de recursos y mejorando la flexibilidad del programa.

El problema se puede desglosar en los siguientes subprocesos:

1. Capturar texto de manera dinámica, sin límite predefinido.
2. Permitir ingresar la palabra o frase a buscar.
3. Permitir ingresar la palabra o frase que reemplazará a la anterior.
4. Realizar la sustitución dentro del texto.
5. Mostrar el resultado final y liberar la memoria utilizada.

Solución propuesta

La solución se implementó mediante tres funciones principales:

- **llenarTexto()**
  Permite al usuario ingresar texto línea por línea hasta escribir "FIN".
  Usa realloc() para ir ampliando el espacio de memoria según la longitud total del texto.

- **Buscar y Reemplazar()**
  Busca todas las apariciones de una subcadena dentro del texto usando strstr(), y construye una nueva cadena con los reemplazos aplicados.
  Usa realloc() para aumentar dinámicamente el tamaño del texto resultante.
- **mostrarTexto()**
  Imprime el texto actual en pantalla.

El programa hace uso de punteros para recorrer y manipular el texto, y libera toda la memoria utilizada con free() al final del proceso.

Tiempos estimados y reales de desarrollo

Etapa del desarrollo	**Actividades principales**	**Tiempo estimado**	**Tiempo real**
Análisis del problema	Identificación de requerimientos y diseño de funciones	1 Dia	4 hora
Implementación inicial	Codificación de funciones básicas (llenarTexto, mostrarTexto)	1 Dia	3 horas
Incorporación de memoria dinámica	Uso de malloc, realloc, free	3 hora	2 horas
Implementación de búsqueda y reemplazo	Creación de función con punteros y strstr()	2 horas	2 horas
Pruebas y corrección de errores	Ejecución con distintos textos	1 hora	1 hora
Total		**2 días 6 horas**	**12 horas**

Ventajas del uso de memoria dinámica

- Permite **manejar textos de tamaño variable** sin desperdiciar memoria.
- Mejora la **eficiencia** en el uso de recursos del sistema.
- Facilita el manejo de datos introducidos por el usuario en tiempo de ejecución.
- Demuestra el uso práctico de **punteros y funciones de biblioteca estándar**.

Conclusión

El desarrollo del simulador de editor de texto básico permitió comprender y aplicar conceptos clave de programación en C, particularmente el **manejo de memoria dinámica**, el **uso de punteros** y la **manipulación de cadenas**.

La solución implementada demostró ser eficiente y funcional, logrando que el usuario pueda **ingresar texto de cualquier longitud**, **buscar y reemplazar subcadenas**, y **visualizar el resultado final** sin restricciones de espacio.
El uso de malloc, realloc y free resultó fundamental para administrar correctamente la memoria, evitando errores de segmentación y fugas de memoria.

En conclusión, este proyecto representa un ejemplo práctico del potencial del lenguaje C para resolver problemas de manipulación de texto con estructuras de memoria flexibles y eficientes, sentando las bases para futuros desarrollos más avanzados en el área de procesamiento de datos.

Codigo :

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

// -------------------------
// Función para llenar texto dinámicamente
// -------------------------
char* llenarTexto() {
    char *texto = NULL;
    char buffer[200];   // Buffer temporal
    size_t tamano = 0;  // Tamaño actual del texto

    printf("Ingrese el texto base (escriba 'FIN' para terminar):\n");

    while (1) {
        fgets(buffer, sizeof(buffer), stdin);
        buffer[strcspn(buffer, "\n")] = '\0'; // Eliminar salto de línea

        // Si el usuario escribe "FIN", se detiene
        if (strcmp(buffer, "FIN") == 0)
            break;

        // Calcular el nuevo tamaño y usar realloc
        size_t nuevoTamano = tamano + strlen(buffer) + 2; // +1 espacio y +1 '\0'
        char *temp = realloc(texto, nuevoTamano);

        if (temp == NULL) {
            printf("Error al asignar memoria.\n");
            free(texto);
            exit(1);
        }
```

```
        texto = temp;

        // Copiar o concatenar el texto
        if (tamano == 0)
            strcpy(texto, buffer);
        else {
            strcat(texto, " ");
            strcat(texto, buffer);
        }

        tamano = strlen(texto);

        printf("[Memoria usada: %zu bytes]\n", nuevoTamano);
    }

    return texto;
}

// -------------------------
// Función para mostrar texto
// -------------------------
void mostrarTexto(const char *texto) {
    printf("%s\n", texto);
}

// -------------------------
// Función para buscar y reemplazar
// -------------------------
char* buscarYReemplazar(char *texto, const char *buscar, const char *reemplazar) {
    char *resultado = NULL;
    char *pos = texto;
    size_t tamResultado = 0;
    size_t lenBuscar = strlen(buscar);
    size_t lenReemplazar = strlen(reemplazar);

    while (1) {
        char *encontrado = strstr(pos, buscar);
        if (!encontrado) {
            // Copiar el resto del texto
            size_t lenResto = strlen(pos);
            resultado = realloc(resultado, tamResultado + lenResto + 1);
            if (!resultado) {
                printf("Error de memoria.\n");
                exit(1);
            }
            strcpy(resultado + tamResultado, pos);
            break;
        }

        // Copiar texto antes del encontrado
```

```
        size_t lenAntes = encontrado - pos;
        resultado = realloc(resultado, tamResultado + lenAntes + lenReemplazar + 1);
        if (!resultado) {
            printf("Error de memoria.\n");
            exit(1);
        }

        strncpy(resultado + tamResultado, pos, lenAntes);
        tamResultado += lenAntes;

        // Agregar el texto reemplazado
        strcpy(resultado + tamResultado, reemplazar);
        tamResultado += lenReemplazar;

        pos = encontrado + lenBuscar; // Mover el puntero
    }

    return resultado;
}

// -------------------------
// Función principal
// -------------------------
int main() {
    char *texto = NULL;
    char buscar[50];
    char reemplazar[50];

    // Llenar texto dinámicamente
    texto = llenarTexto();

    printf("\nTexto original:\n");
    mostrarTexto(texto);

    // Pedir texto a buscar y reemplazar
    printf("\nIngrese el texto a buscar: ");
    fgets(buscar, sizeof(buscar), stdin);
    buscar[strcspn(buscar, "\n")] = '\0';

    printf("Ingrese el texto a reemplazar: ");
    fgets(reemplazar, sizeof(reemplazar), stdin);
    reemplazar[strcspn(reemplazar, "\n")] = '\0';

    printf("\nTexto a buscar: %s\n", buscar);
    printf("Texto a reemplazar: %s\n", reemplazar);

    // Reemplazar texto usando memoria dinámica
    char *nuevoTexto = buscarYReemplazar(texto, buscar, reemplazar);

    // Mostrar el resultado final
    printf("\nTexto final:\n");
```

```
    mostrarTexto(nuevoTexto);

    // Liberar memoria
    free(texto);
    free(nuevoTexto);

    return 0;
}
```