Video5 (1)

Introducción a C – Ejemplos de Strings

Conceptos Principales

1. a
2. b
3. c

Notas

- **Propósito del video**: Mostrar ejemplos prácticos del manejo de **strings** en C, reforzando los conceptos teóricos del video anterior.
- **Almacenamiento de strings**:
  - Los strings se representan como **arreglos de `char`** o como punteros a áreas de memoria.
  - Ejemplo: `char *texto = "Hola mundo";`
    - `"Hola mundo"` es una **literal** almacenada en la sección de **constantes/lectura**.
    - El puntero `texto` contiene la **dirección de memoria** de la literal, no los caracteres en sí.
    - Las literales **no se pueden modificar**; si se quiere un string modificable, se debe asignar memoria dinámica con `calloc` o `malloc`.
- **Variables automáticas vs heap**:
  - Variables automáticas (ejemplo: `char texto[12];`) se almacenan en la **pila** y desaparecen al salir de la función.
  - Variables en heap (`calloc`) permanecen hasta que se libere explícitamente con `free`.
  - Se recomienda **siempre usar heap** para strings que deban persistir fuera de la función.
- **Ejemplo: calcular la longitud de un string**
  - Se recorre el arreglo hasta encontrar el **carácter nulo (`\0`)**.
  - Es importante inicializar correctamente los contadores y validar la presencia de `\0`.
  - Si se olvida el `\0`, puede ocurrir un **segmentation fault** al intentar leer memoria no asignada.
- **Ejemplo: buscar un carácter dentro de un string**
  - Se recorre el string comparando cada carácter con el objetivo.
  - Se retorna `1` si se encuentra el carácter, `0` si se llega al final sin encontrarlo.
  - Recordar la diferencia entre **carácter (`'a'`)** y **string (`"a"`)**.
- **Buenas prácticas**:
  - Siempre inicializar variables antes de usarlas.
  - Documentar y comentar el código.
  - Mantener consistencia en espacios y formato para un código profesional.
  - Evitar errores comunes de strings dinámicos: retorno de punteros a memoria automática que desaparece.
- **Resumen conceptual**:
  - **Strings literales** → const, no modificables, puntero a sección de memoria de solo lectura.

- **Strings en heap** → modificables, permanecen hasta `free`.
- **Variables automáticas** → temporales en stack, se borran al salir de la función.
- Recorrer strings → detenerse en `\0` para evitar errores de segmentación.
- Distinción entre **carácter** y **string** fundamental en C.