

Video31 (1)

Conceptos Principales

1. a
2. b
3. c

Notas

1. Apertura de archivos: `fopen`

- Se usa para abrir archivos en **modo texto o binario**.
 - Sintaxis: `FILE *fopen(const char *nombreArchivo, const char *modo);`
 - **Argumentos:**
 - `nombreArchivo`: ruta o nombre del archivo.
 - `modo`: define la operación:
 - `"r"`: lectura
 - `"w"`: escritura (trunca el contenido si existe)
 - `"a"`: append, escribir al final
 - `"rb"`, `"wb"`: versiones binarias
 - Devuelve un **puntero a FILE** (descriptor de archivo) que contiene información como nombre, tamaño y posición actual de lectura/escritura.
-

2. Cierre de archivos: `fclose`

- Libera el archivo y los recursos del sistema.
 - Muy importante porque si el archivo queda abierto, **otros programas no pueden acceder a él**.
-

3. Escritura de datos: `fwrite`

- Se puede usar para **archivos binarios y texto**, aunque es más útil en binario.
 - Sintaxis: `size_t fwrite(const void *ptr, size_t size, size_t count, FILE *stream);`
 - **Argumentos:**
 - `ptr`: puntero al bloque de memoria que se quiere escribir.
 - `size`: tamaño de cada elemento (en bytes).
 - `count`: cantidad de elementos.
 - `stream`: archivo donde se escriben los datos.
 - Permite escribir **estructuras completas** directamente en archivos binarios.
-

4. Posicionamiento dentro del archivo: `fseek` y `ftell`

- `fseek(FILE *stream, long offset, int origin):`

- Mueve la posición de lectura/escritura dentro del archivo.
 - **origin** puede ser:
 - **SEEK_SET**: desde el inicio del archivo
 - **SEEK_CUR**: desde la posición actual
 - **SEEK_END**: desde el final del archivo
 - **ftell(FILE *stream)**:
 - Devuelve la posición actual dentro del archivo.
 - Permite **simular punteros** en archivos binarios, útil para listas o estructuras persistentes.
-

5. Aplicaciones principales

- Manipulación de **estructuras de datos persistentes**.
- Guardar y recuperar listas o estructuras directamente desde el archivo.
- Control preciso de **bytes y posiciones** dentro del archivo.