

AEDD - Guía Práctica 18: Archivos

En C++, aunque el uso de la clase *string* es común y práctico para manejar cadenas de caracteres, en ciertas circunstancias, como la lectura y escritura de archivos, es preferible usar arreglos de caracteres (*char[]*) en lugar de *string*. Esto se debe a que los arreglos de caracteres permiten un control más preciso sobre el formato y la longitud de los datos, especialmente en operaciones de entrada/salida (I/O) con archivos. Los *string* pueden introducir ciertos comportamientos no deseados relacionados con la gestión de memoria dinámica que no siempre son compatibles o eficientes con los sistemas de archivos.

A continuación, se muestra un ejemplo de cómo manejar la lectura y escritura de archivos utilizando arreglos de caracteres para una estructura que almacena información de una persona con los campos *nombre*, *apellido* y *año de nacimiento*.

```
struct Persona {
    string nombre;
    string apellido;
    int anioNacimiento;
};

struct PersonaArchivo {
    char nombre[50];
    char apellido[50];
    int anioNacimiento;
};

/*
 *
 *Para facilitar el guardado en el archivo se crea una función que transforma
 *entre las estructuras.
 *
 */
PersonaArchivo convertirAFormatoArchivo(Persona persona) {
    PersonaArchivo personaArchivo;

    // Para realizar la copia del string utilizamos la función strcpy que nos permite copiar cadenas de
    // caracteres
    // tambien hacemos uso de la función de string c_str para que nos devuelva una cadena de
    // caracteres.
    strncpy(personaArchivo.nombre, persona.nombre.c_str(), sizeof(personaArchivo.nombre) - 1);
    personaArchivo.nombre[sizeof(personaArchivo.nombre) - 1] = '\0'; // Nos aseguramos que el
    // ultimo caracter se el fin de cadena.

    strncpy(personaArchivo.apellido, persona.apellido.c_str(), sizeof(personaArchivo.apellido) - 1);
    personaArchivo.apellido[sizeof(personaArchivo.apellido) - 1] = '\0';

    personaArchivo.anioNacimiento = persona.anioNacimiento;
    return personaArchivo;
}

/*
 *
 *Para facilitar la lectura del archivo se crea una función que transforma
 *entre las estructuras.
 *
 */
```

*/

```
Persona convertirDesdeFormatoArchivo( PersonaArchivo personaArchivo) {  
    Persona persona;  
    persona.nombre = string(personaArchivo.nombre);  
    persona.apellido = string(personaArchivo.apellido);  
    persona.anioNacimiento = personaArchivo.anioNacimiento;  
    return persona;  
}
```

Ejercicios propuestos:

1. Leer un archivo de texto y contar cuántos dígitos y letras contiene. Mostrar el resumen al finalizar la ejecución del programa. ✓
2. Leer un archivo de texto (original.txt) y escribir en clave (enclave.txt). Escribirlo en clave no es más que sumar una cantidad al número en código ASCII de cada carácter. Esta cantidad deberá ingresarse por teclado. ✓
3. Leer un archivo de texto e informar:
 - a. La cantidad de palabras del texto.
 - b. La longitud de la palabra más larga.
 - c. La cantidad de palabras que comienzan y terminan con letras elegidas por el usuario.
4. Crear un archivo binario de acceso aleatorio de números naturales comprendidos entre 0 y un cierto valor dado.
5. Escribir un programa que permita cargar desde teclado registros que contengan los siguientes campos: título del libro, autor, ISBN, precio, cantidad en existencia y cantidad vendida. Luego, guardar esta información en un archivo *binario* de "inventario".
6. Escribir un programa que utilizando el archivo creado en el problema anterior nos permita:
 - a. Recuperar del archivo el 3er libro y mostrar por pantalla su título y autor.
 - b. Reemplazar en el archivo el 2do libro con uno nuevo, cuyos datos cargará el usuario por teclado.
 - c. Mostrar por pantalla el inventario completo.
7. Las cuentas de usuario de una aplicación se almacenan en un archivo binario llamado Usuarios.dat, que tendrá la siguiente información:

Tipo	Campo	Observaciones
char[11]	nombre_usuario	
char[37]	clave	
Fecha	ultimo_acceso	Fecha de último acceso del usuario a la aplicación. en formato dd/mm/aaaa hh:mi:ss

- a. Declarar un arreglo de hasta 100 usuarios e inicializarlo manualmente con 3 usuarios.
- b. Escribir el archivo Usuarios.dat en base a los datos cargados en el arreglo de usuarios.
- c. Limpiar el arreglo y volver a cargarlo con todos los usuarios almacenados en el archivo Usuarios.dat.

Aclaración: utilizaremos la estructura `tm` de la librería `time.h` para obtener fecha y hora del sistema:

```
struct tm{           //representación del tiempo en formato de calendario
    (fecha/hora)
    int tm_sec;       //Indica los segundos después de un minuto(0 - 59)
    int tm_min;       //Indica los minutos después de una hora (0 - 59)
    int tm_hour;      //Hora [0,23]
    int tm_mday;      //Día del mes[1,31]
    int tm_mon;       //Meses que han pasado desde enero [0,11]
    int tm_year;      //Años desde 1900, si quieres saber el año actual sumas
1900
    int tm_wday;      //Día de la semana, desde el domingo [0,6]
    int tm_yday;      //Días desde el 1 de Enero [0,365]
    int tm_isdst;     //No se xDD, algo de daylight
}

time_t time(time_t *)
//devuelve la fecha/hora (time_t) actual o -1 en caso de no ser posible. Si
el //argumento que se le pasa no es NULL, también asigna la fecha/hora actual
a dicho //argumento.

struct tm *localtime(time_t *)
//recibe un puntero a una variable de tiempo (time_t*) y
// devuelve su conversión como fecha/hora LOCAL.
```

A continuación, un ejemplo:

```
#include <time.h>
using namespace std;

int main(){
    time_t ahora;
    struct tm *fecha;
    time(&ahora);
    fecha = localtime(&ahora);
    cout << fecha->tm_mday <<"-"<< fecha->tm_mon+1 <<"-"<< fecha->tm_year+1900;
}
```

8. Se propone crear un simple editor de texto, llamado Palabras, el cual permite generar textos. Éstos consisten en un conjunto de páginas (hasta 1000), las cuales poseen a su vez hasta 100 líneas de texto de 100 caracteres. A su vez, la página contiene el número de la misma, y dos líneas de encabezado y pie. Tener en cuenta que una página puede no haber usado todas las líneas de texto.

Implementar la estructura de datos de textos, y con la misma:

- Escribir una función que permite guardar el texto en un archivo de tipo binario.
- Escribir una función que, dado un archivo, permita modificar una línea del texto, y lo vuelva a guardar.
- Escribir una función que, dado un archivo y un número de página, imprima esta página en pantalla.
- Escribir una función que imprima el libro completo en pantalla.