Capítulo 1

Manipulación de archivos estructurados

1.1. Introducción

Definición Un archivo es un medio de almacenamiento no volátil, es decir permanece luego de terminar la aplicación que la utiliza o luego de apagar el computador. Un archivo se almacena en memoria secundaria como el disco duro, cd, dvd, etc.

Los archivos llevan asociado un nombre y una extensión, ésta última permite identificar el tipo del archivo, por ejemplo doc, jpg, bmp, cpp, etc.

Los archivos estructurados se caracterizan por que almacenan los datos respetando un patrón específico que ocupa un determinado espacio. Una de las aplicaciones más comunes en la que se utiliza archivos estructurados es el de las bases de datos. Supongamos que deseamos almacenar datos de libros en un archivo, entonces los datos se estructurarán de acuerdo a lo que interesa almacenar de cada libro, en este caso podemos necesitar representar el título, autor, año de edición, ISBN y editorial. Entonces un conjunto de datos de varios libros se verán como en la figura 1.1.

Cada uno de los libros en el archivo se representa por un registro. Todos los registros del archivo se almacenan siguiendo una estructura (struct en C o C++), discutiremos en la siguiente sección cómo crear una estructura para los registros, seguiremos usando el ejemplo del libro.

1.2. Estructuras (struct)

Supongamos que tenemos la siguiente estructura para cada registro de libro:

```
-------------------------------------------------
  ISBN | TITULO | AUTOR |ANIO EDICION | EDITORIAL
-------------------------------------------------
```

Lo primero es determinar el tipo de dato de cada elemento de la estructura, cada elemento se denomina campo, en el ejemplo hay 5 campos.

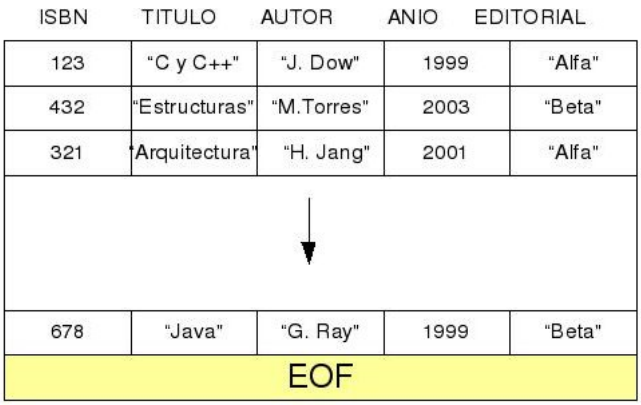

Figura 1.1: Vista lógica de un archivo de registros de libros.

- ISBN: un entero
- TITULO: cadena de 20
- AUTOR: cadena de 20
- ANIO EDICION: entero
- EDITORIAL: cadena de 20

La implementación en C o C++ es como sigue:

```
typedef struct Libro
{
int ISBN;
char titulo[20];
char autor[20];
int  anio;
char editorial[20];
}TLibro;
```

donde TLibro se utiliza como un nuevo tipo de dato que representa un tipo para cada libro, entonces un libro en particular se puede definir como:

```
LIBRO miLibro;
```

y para acceder a cada uno de sus campos utilizaremos el operador punto (.), por ejemplo:

```
miLibro.ISBN=2344;
miLibro.titulo="C y C++";
miLibro.autor="D. Ritchie";
miLibro.anio=1999;
miLibro.editorial="Alfa";
```

Ahora, utilizaremos un archivo para guardar cada registro, esto lo veremos en la siguiente sección.

1.3. Operaciones con archivos

Para manipular un archivo utilizaremos el tipo FILE (en realidad crearemos un puntero de tipo FILE) y usaremos las siguientes funciones de stdlib.h.

- **fopen**: Para abrir un archivo y asociarlo con un puntero de tipo FILE. La sintaxis es como sigue:

  FILE *fopen(char *nombre, char *modo)

  donde *nombre* es el nombre del archivo, en el que se puede incluir la ruta y *modo* indica la forma en que puede usarse el archivo:

"w "	Para crear un archivo.
"r "	Para leer un archivo.
"a "	Para aadir datos al archivo.

  Ejemplo:

  ```
  FILE *f = fopen("libros.txt","r");
  ```

- **fread**: Lee datos del archivo y los pasa a la memoria principal.

  ```
  fread(void *puntero, size_t tamanyo, size_t nmemb, FILE *f)
  ```

  donde *puntero* indica el espacio de memoria en dónde se almacenarán los datos, *tamanyo* es la cantidad de bytes a leer, *nmemb* es la cantidad de veces que se leerá (asumiremos igual a 1), y f representa el archivo.

- **fwrite**: Lee datos de la memoria principal y los pasa al archivo.

  ```
  fwrite(void *puntero, size_t tamanyo, size_t nmemb, FILE *f)
  ```

  donde *puntero* indica el espacio de memoria desde dónde se leeran los datos, *tamanyo* es la cantidad de bytes a escribir, *nmemb* es la cantidad de veces que se escribirán (asumiremos igual a 1), y f representa el archivo.

- **fseek**: Permite ubicarnos en cualquier posición del archivo para actualizar datos.

  ```
  int fseek(FILE *f, long int desplazamiento, int origen);
  ```

  La función *fseek* activa el indicador de posicin de ficheros de f y los desplaza seún el parámetro *desplazamiento* donde la posición inicial es *origen*. Se puede utilizar las constantes SEEK_SET para indicar como origen el inicio, SEEK_CUR en caso de que origen sea la posición actual y SEEK_END si es el final. Frecuentemente desplazamiento es un valor obtenido mediante una llamada a la función *ftell*. Por ejemplo, si se tiene la posición de un registro X, entonces en cualquier momento podemos regresar a ubicarnos en ese registro y actualizar sus dato.

- **ftell**: La función *ftell* obtiene el valor actual del indicador de posición de fichero para un determinado archivo.

  ```
  long int ftell(FILE *f);
  ```

1.4. Implementacion de las operaciones

Siguiendo con nuestra política de implementar un programa modularmente, crearemos dos módulos, uno para guardar un registro y otro para reportar todos los registro del archivo.

GUARDAR

```
void guardarLibro(TLibro L, FILE *f)
{
  fwrite(&L,sizeof(TLibro),1,f);
}
```

REPORTAR

```
void reportarLibro(char *file)
{
  FILE *f=fopen(file,"r");
  TLibro L;
  if (f!=NULL)
  {
    fread(&L,sizeof(TLibro),1,f);
    while(!feof(f))
    {
      imprimir(L);
      fread(&L,sizeof(TLibro),1,f);
    }
  }
  else
  {
   printf( "Error al abrir el archivo \n" );
   exit(1);
  }
  fclose(f);
}
```

En el módulo anterior usamos la función *feof()* para determinar si ya se ha leído el final del archivo. Además usamos el módulo imprimir para mostrar los datos:

```
void imprimir(TLibro L)
{
  printf("ISBN  : %s \n", L.ISBN);
  printf("Titulo: %s \n", L.titulo);
  printf("Anio  : %s \n", L.autor);
  printf("Anio  : %d \n", L.anio);
  printf("Anio  : %s \n", L.editorial);
}
```

1.5. Implementación completa

Se adjunta la implementación completa para guardar y reportar datos de un archivo.

1.6. Ejercicios

- Utilizando el código base de la sección anterior, agregue lo necesario para que el programa soporte búsqueda de un libro por ISBN, es decir se da el ISBN y el programa debe devolver los datos completos del libro si lo encuentra o un mensaje de error si no se encuntra un libro con tal ISBN.
- Complete el código anterior de modo que su programa soporte modificación de datos. Por ejemplo supongamos que se desee modificar los datos del autor de un libro con ISBN determinado.