Archivos

Técnico Universitario en Programación - Laboratorio de Computación II

Simón, Angel

Hemos estado trabajando con tipos de datos más complejos que representan un objeto o registro. Pero hasta el momento la limitación más importante que se presentaba es que la información se almacenaba en la memoria de la computadora y, al finalizar el programa, se perdía. Por lo que si cargabamos una lista de diez clientes y nuestro programa finalizaba, esa lista se perdía y era necesario cargarla de vuelta.

Este problema se soluciona haciendo uso de archivos. Ya que la información puede ser persistente en algún medio de almacenamiento. Generalmente el almacenamiento principal.

Un archivo de datos como los que trabajaremos no es más que una sucesión de bytes en formato binario que se encuentra en algún dispositivo de almacenamiento con un nombre identificatorio.

Si hacemos una clasificación sencilla de los archivos podemos destacar:

Archivos del sistema

Archivos ejecutables

Archivos de datos

Los archivos del sistema son archivos necesarios para que el sistema operativo trabaje. Contienen configuraciones, drivers, bibliotecas, etc. No es común que el usuario común manipule estos archivos.
Los archivos ejecutables son aplicaciones que contienen el conjunto de instrucciones en lenguaje máquina para resolver una tarea.
En cambio, los archivos de datos son aquellos que permiten a una aplicación persistir información. Ya sea una hoja de cálculo, un dibujo, un archivo de texto, etc. Estos archivos suelen tener un formato y/o estructura específica para que ciertos programas puedan crearlos, visualizarlos, modificarlos, etc.
En las aplicaciones que realizaremos, estos archivos de datos contendrán información de entidades. Por ejemplo, llevar un registro de exámenes, de ventas, de artículos, etc. Para poder lograr este objetivo es necesario que sigamos una serie de reglas puntuales.

Registros de longitud fija

Todos los registros de un archivo debieran tener la misma longitud (en bytes). Esto determina de cuanto en cuanto debe desplazarse dentro del archivo para realizar la lectura de información. Por ejemplo, imaginemos que estamos trabajando con un archivo de exámenes. Indistintamente de los datos almacenados un esquema general del archivo podría ser algo como el siguiente:

Imagen 1 · Representación gráfica de un archivo con registros de longitud fija.

Podemos observar que el archivo cuenta con cinco registros y que cada registro tiene una longitud fija de 120 bytes. De ahí surge que el primer registro comienza en el byte 0 y finaliza en el byte 119. El segundo comienza en el byte 120 y finaliza en el 239 y así sucesivamente. En el byte 600 comenzaría el sexto registro que aún no fue cargado.

Lograr que los registros tengan longitud fija no es un problema. Ya que cuando creamos objetos de una misma clase, todos tendrán el mismo tamaño en bytes.

Registros identificados por valores únicos e irrepetibles

Algo importante al trabajar con archivos es poder identificar a los registros dentro del mismo de alguna manera. Es importante destacar que el "número de registro" no puede ser una manera válida de identificarlo ya que, por varias razones, puede ocurrir que el archivo se vea modificado y el orden original de los registros cambie (por ejemplo, al ordenar un archivo).

En general, se debe definir uno o varios tributos de nuestra clase como identificadores del registro. Y deberemos asegurar que dicho valores o conjunto de valores no se repitan dentro del archivo. Por ejemplo:

Código ISO	País	Capital	Superficie km^2
AR	Argentina	Buenos Aires	2780400
AS	Samoa Americana	Pago Pago	199
AT	Austria	Viena	83871
AU	Australia	Canberra	7741220

Tabla 1 · Datos de un archivo de países

En la Tabla 1, podemos observar un archivo de países compuesto por los atributos País, Código ISO, Capital y Superficie. El código ISO es una cadena de dos caracteres que identifica unívocamente al registro. Luego, no podrá haber dos registros con el valor AU en el Código ISO ya que ese valor es el que identifica al país Australia. De la misma manera, si quisieramos saber la longitud de Austria, deberíamos recorrer el archivo en la búsqueda del código AT. Si lo encontramos sabremos la longitud del país.

Archivo no debe admitir registros duplicados

En relación a lo mencionado en el ítem anterior. De cumplir la restricción de que uno o varios atributos sean únicos dentro del archivo. No podría darse la situación en la que dos registros sean iguales. De suceder, sería conceptualmente incorrecto.

Funciones de Archivos de C

Una vez aclarados los conceptos generales para trabajar con archivos podemos visualizar cuáles son las funciones para poder hacer uso de ellos. Las acciones principales para trabajar con archivos son:

Acción	Descripción	Función de C
Apertura	Abrir un archivo en un modo particular, crearlo si es necesario.	fopen
Cierre	Cerrar un archivo previamente abierto.	fclose
Escribir	Escribir datos de la memoria a un archivo.	fwrite
Leer	Leer datos de un archivo a la memoria.	fread
Desplazar	Desplazarse una cantidad determinada de bytes dentro del archivo.	fseek
Tamaño	Obtener el tamaño total de bytes desplazados desde el comienzo a la ubicación actual.	ftell

Tabla 2 · Acciones generales con el trabajo con archivos

Otra acción muy común y necesaria será la de poder saber cuánto ocupa en memoria una variable o un registro de una clase en particular. Para ello utilizaremos el operador **sizeof**. Las sintaxis para utilizarlo son las siguientes:

```
size_t sizeof (tipo de dato)
size_t sizeof variable
```

En cualquier caso va a devolver el tamaño en bytes del elemento indicado. Por ejemplo:

```
int numero;
cout << sizeof (Examen);
cout << sizeof numero;
```

Código 1 · Ejemplo de uso del operador sizeof

En el primer uso de sizeof, el cout nos mostrará por pantalla el tamaño en bytes del tipo de dato Examen (todo objeto de la clase Examen ocupará ese tamaño en memoria).
En el segundo uso de sizeof, el cout nos mostrará por pantalla el tamaño en bytes de la variable número. Es decir, nos mostrará por pantalla el tamaño en bytes de un int.

> ❗ El tipo de dato size_t es un entero largo sin signo → unsigned long

Abrir un archivo - fopen

```
FILE * fopen(const char *nombre, const char *modo);
```

Abre un archivo cuyo nombre es indicado por el contenido de la cadena apuntada por nombre, el segundo argumento es una cadena que indica el modo de apertura del archivo.

Parámetros:
nombre – la ruta del archivo, puede ser relativa o absoluta si el sistema lo permite.
modo – el modo de apertura del archivo:

Modo	Nombre	Descripción
wb	write-binary	Destruye el archivo a 0 bytes y lo abre como escritura. Si no existe el archivo, lo crea.
rb	read-binary	Abre el archivo como lectura desde el byte 0. No admite escritura.

ab	append-binary	Abre el archivo desde el final y lo abre como escritura. No admite modificar lo existente, sólo agregar. Si no existe el archivo, lo crea.
rb+	read-binary plus overwrite	Abre el archivo desde el byte 0 y lo abre como escritura. Admite modificar registros existentes.

Tabla 3 · Modos de apertura de archivos binarios con fopen.

Valor de retorno: La función fopen retorna un puntero FILE que controlará el archivo. Si el proceso de apertura no es exitoso, entonces retorna un carácter nulo.

Ejemplo:

```
bool existeArchivo(const char *strArchivo){
      FILE *p;
      p = fopen(strArchivo, "rb");
      if (p == NULL){
        fclose(p);
        return false;
      }
      return true;
}
```

Código 2 · Ejemplo de uso de fopen y fclose

Cerrar un archivo - fclose

```
int fclose(FILE *archivo);
```

El archivo apuntado por *archivo* es liberado y cerrado.

Parámetros:
archivo - Puntero a una estructura FILE previamente abierta con fopen.

Valor de retorno:
La función fclose retorna cero si el archivo es cerrado exitosamente, si ocurren errores devuelve EOF.

Lectura - fread

```
size_t fread(void *ptr, size_t tam, size_t cant, FILE *archivo);
```

La función fread lee desde el archivo apuntado por *archivo*, *cant* elementos de tamaño *tam* y los almacena en el bloque de memoria apuntado por *ptr*.

Parámetros:
ptr – puntero a un bloque de memoria. Generalmente será la dirección de un objeto.
tam – tamaño en bytes de cada elemento a ser leído.
cant – cantidad de elementos a leer, cada uno de tamaño tam.
archivo – puntero a una estructura FILE que especifica el flujo de datos.

Valor de retorno:
La función fread devuelve la cantidad de registros que pudo leer correctamente y en caso contrario 0.

Escribir - fwrite

```
size_t fread(void *ptr, size_t tam, size_t cant, FILE *archivo);
```

La función fwrite escribe en el archivo apuntado por *archivo*, *cant* elementos de tamaño *tam* provenientes del bloque de memoria apuntado por *ptr*.

Parámetros:
ptr – puntero a un bloque de memoria desde donde se escribirán los datos. Generalmente será la dirección de un objeto.
tam – tamaño en bytes de cada elemento a ser escrito.
cant – cantidad de elementos a escribir, cada uno de tamaño tam.
archivo – puntero a una estructura FILE que especifica el flujo de datos.

Valor de retorno:
La función fwrite devuelve la cantidad de registros que pudo escribir correctamente y en caso contrario 0.

Ejemplo de alta de un registro de la clase Examen

```
#include <iostream>
using namespace std;
#include <cstdio>
#include "Examen.h"

int main(){
  Examen a;
  a.cargar();

  FILE *p = fopen("examenes.dat", "ab");
  if (p == NULL){
      cout << "No se pudo abrir el archivo.";
      return 1;
  }
  bool ok= fwrite(&a, sizeof(Examen), 1, p);
  if (ok == true){
      cout << "Registro guardado exitosamente." << endl;
  }
  else{
      cout << "No se pudo guardar el registro." << endl;
  }
  fclose(p);
  return 0;
}
```

Código 3 · Ejemplo de carga y almacenamiento en un archivo de exámenes.

Ejemplo de lectura de un registro de la clase Examen

```
#include <iostream>
using namespace std;
#include <cstdio>
#include "Examen.h"

int main(){
  Examen a;

  FILE *p = fopen("examenes.dat", "rb");
  if (p == NULL){
      cout << "No se pudo abrir el archivo.";
      return 1;
  }
  bool ok= fread(&a, sizeof(Examen), 1, p);
  if (ok == true){
      a.mostrar();
```

```
  }
  else{
      cout << "No se pudo leer el registro." << endl;
  }
  fclose(p);
  return 0;
}
```

Código 4 · Ejemplo de lectura y listado de un registro del archivo de exámenes.