

Archivos Binarios

Archivos de acceso directo

Archivos de acceso directo

- Concepto de archivo binario

Un archivo binario es un tipo de archivo que permite almacenar un bloque de datos de cualquier tipo. Los archivos binarios los puede crear únicamente el programa y el acceso a sus elementos sólo es posible a través del programa. El contenido de un archivo binario es ilegible ya que utiliza un esquema de representación binario interno. Este esquema depende de la computadora que se use, por lo que no puede ser visualizado mediante un editor de textos.

- Diferencia entre archivo binario y archivo de texto

Las componentes de un archivo de texto son de tipo *char* y están organizados en líneas mientras que las componentes de un archivo binario pueden ser de cualquier tipo predefinido o definido por el usuario (excepto de tipo archivo).

Taller de Lenguajes I

- Apertura y cierre de archivos binarios

- Las operaciones de apertura y cierre para archivos binarios son exactamente las mismas que las vistas para archivos de texto: *fopen* y *fclose*.
- Los modos de apertura también son los mismos que para archivos de texto. La única diferencia radica en que hay que añadir una 'b' de "binary" al modo de apertura, es decir, "rb", "wb", "ab", "r+b", "w+b" y "a+b", manteniendo el mismo significado que para archivos de texto.
- Una vez creado un archivo con un tipo determinado (binario o de texto), ya no se le puede cambiar el mismo.

Taller de Lenguajes I

- Lectura y escritura de archivos binarios

- Las funciones de lectura y escritura ya no son las mismas que para los archivos de texto. Mientras que *fscanf* y *fprintf* realizan una conversión del tipo de dato a cadena de caracteres, en los archivos binarios esto no es necesario. La función utilizada para **leer** datos de un archivo binario es *fread* y la función utilizada para **escribir** datos en un archivo binario es *fwrite*. Estas funciones son independientes del tipo de dato que se lea o escriba, es decir, no realizan ninguna interpretación del tipo de dato.

- Sintaxis de las funciones *fread* y *fwrite*

```
size_t fread(const void *p, size_t size, size_t n, FILE *arch)
size_t fwrite(const void *p, size_t size, size_t n, FILE *arch)
```

- Ambas funciones devuelven el número de elementos leídos o escritos, o 0 en caso de que no se haya leído o escrito ninguno.

Taller de Lenguajes I

- Sintaxis de la función *fread*

- El argumento **p** es la dirección de memoria a partir de la cual se almacenará la secuencia de bytes leída desde el archivo binario.
- El argumento **size** contiene el número de bytes que componen cada uno de los elementos que van a ser leídos.
- El argumento **n** contiene el número de elementos de tamaño **size** que van a leerse.
- El cuarto argumento es el puntero o descriptor de fichero que devuelve la función *fopen*.

- Sintaxis de la función *fwrite*

- El argumento **p** es la dirección de memoria a partir de la cual se encuentran los datos que se escribirán en el fichero.
- El argumento **size** contiene el número de bytes que componen cada uno de los elementos que van a ser escritos en el archivo binario.
- El argumento **n** contiene el número de elementos de tamaño **size** que se van a escribir.
- El cuarto argumento es el puntero o descriptor de fichero que devuelve la función *fopen*.

Ejemplo 1:

- **Escritura en un archivo binario**

Dada la siguiente estructura:

```
struct alumno{  
    char Nombre[20];  
    char Apellido[20];  
    float NotaObtenida;  
};
```

Leer desde teclado la información de 5 alumnos y almacenarla en un archivo binario.

Ejemplo 1

```
#include <stdio.h>
#include <stdlib.h>
```

```
struct alumno{
    char Nombre[20];
    char Apellido[20];
    float NotaObtenida;
};
```

```
void leerAlumno(struct alumno * a);
void almacenarInformacion(FILE*, struct alumno a);
```

```
int main(){
    FILE* arch; int i; struct alumno a;
```

```
    arch=fopen("Alumnos.dat","wb");
```

```
    if(arch) {
```

```
        for(i=0;i<5;i++) {
```

```
            leerAlumno(&a);
```

```
            almacenarInformacion(arch,a);
```

```
        }
```

```
        fclose(arch);
```

```
    }
```

```
    else
```

```
        printf("Error al abrir el archivo.");
```

```
    return 0;
```

```
}
```

Se asocia la variable *arch* con el archivo donde se almacenarán los datos

Note que se pasa un FILE*

Se cierra el archivo al finalizar la lectura

Ejemplo 1 (continuación)

```
void leerAlumno (struct alumno *a) {  
    printf("Nombre: "); scanf("%s", a->Nombre);  
    printf("Apellido: "); scanf("%s", a->Apellido);  
    printf("NotaObtenida: "); scanf("%f", &(a->NotaObtenida));  
}
```

```
void almacenarInformacion(FILE*arch, struct alumno a){  
    fwrite(&a, sizeof(struct alumno), 1, arch);  
}
```



Con fwrite se escribe el registro del alumno *i* en el archivo; le debo pasar la dirección de la estructura.

Taller de Lenguajes I

- Posicionamiento en un archivo

- La función ***fseek*** es una herramienta útil cuando se desea trabajar con archivos binarios. Al tener la información almacenada en bloques fijos de bytes continuos, permite tratar al archivo como si fuese un vector y moverse directamente a cualquier byte del mismo.

- Sintaxis de la función ***fseek***

```
int fseek(FILE *stream, long offset, int whence);
```

- En caso de éxito la función devuelve el número de bytes que se ha desplazado o -1 en caso de error.

- Posicionamiento en un archivo

- El primero de los tres argumentos es un puntero al archivo que se está utilizando.
- El segundo argumento se denomina *offset* e indica cuán lejos se puede mover desde la posición actual, desde principio del archivo o desde el final del mismo (esta posición se indica a través del tercer argumento). *offset* puede ser positivo (se desplaza hacia adelante), negativo (se desplaza hacia atrás) o cero (se queda en el lugar).
- El tercer argumento se denomina *whence* y tiene tres modos:

<i>whence</i>	<i>Tipo de posicionamiento</i>
SEEK_SET	Relativo al principio del fichero
SEEK_CUR	Relativo a la posición actual
SEEK_END	Relativo al final del fichero

- Posicionamiento en un archivo
 - Para un archivo binario, la nueva posición, medida en caracteres desde el principio del fichero, es obtenida mediante la suma de ***offset*** y la posición especificada por ***whence*** (SEEK_SET, SEEK_CUR o SEEK_END).
 - Para un archivo de texto, o bien ***offset*** será cero, o bien ***offset*** será un valor retornado por una llamada anterior a la función *ftell* al mismo archivo y ***whence*** será SEEK_SET

Ejemplo 2:

- Lectura y posicionamiento en un archivo binario

Modificar el programa del ejemplo anterior para leer la información del tercer alumno almacenado en el archivo utilizando un único modo de apertura.

Ejemplo 2

```
#include <stdio.h>
#include <stdlib.h>
struct alumno{
    char Nombre[20];
    char Apellido[20];
    float NotaObtenida;
};
void leerAlumno(struct alumno * a);
void almacenarInformacion(FILE*, struct alumno a);
void recuperarTercerAlumno(FILE*);
void imprimirInformacion(struct alumno a);
int main(){
    FILE* arch; int i; struct alumno a;
    arch=fopen("Alumnos.dat","wb+");
    if (arch) {
        for(i=0;i<5;i++) {
            leerAlumno(&a);
            almacenarInformacion(arch,a); }
        recuperarTercerAlumno(arch);
        fclose(arch); }
    else
        printf("Error al abrir el archivo.");
    return 0; }
```




Modifico el modo de apertura para lectura y escritura.


Ejemplo 2 (continuación)

```
void recuperarTercerAlumno(FILE*arch){  
    struct alumno a;  
    fseek(arch,2*sizeof(struct alumno),SEEK_SET);  
  
    fread(&a,sizeof(struct alumno),1,arch);  
    imprimirInformacion(a);  
}
```

Después de la escritura el indicador queda posicionado al final del archivo; lo posiciono en el tercer alumno.



Copio en la variable *a* el registro del tercer alumno



```
void imprimirInformacion(struct alumno a){  
    printf("Nombre %s", a.Nombre);  
    printf("\nApellido %s", a.Apellido);  
    printf("\nNota: %f", a.NotaObtenida);  
}
```

Taller de Lenguajes I

- Ejercicios propuestos

- Generar en un archivo binario denominado “inventario.dat”, el cual corresponde al inventario de una juguetería. De cada juguete se conoce el nombre, código, categoría (1..9), precio y el stock actual (50). La información se lee desde teclado hasta ingresar un juguete con código 0.
- El propietario de la juguetería decidió aumentar en un 25% los juguetes con categoría 2 y 3. Como encargado, debe modificar el inventario aplicando el aumento indicado.