

**Ingeniería en Ciencias de la Computación**

**Introducción a la Programación**

*“Documentación de Tema de Exposición de Tercer Parcial”*

*Tema 5: Archivos Binarios o Directos*

Tercer Parcial

|  |  |
| --- | --- |
| **Equipo:** | **Nombre del Maestro:** |
| 3 | Ricardo Blanco Vega |
| **Matrícula:** | **Nombre del Alumno:** |
| 338811 | Valeria Sofía Nevárez Juárez |
| 338900 | Marley Zaragoza Balderrama |

Fecha de entrega: **11/11/2019**

**Tabla de contenidos**

[**Objetivos**](#_sgjqbntl9lnn) **3**

[**Desarrollo**](#_u23z1vuziyxc) **4**

[¿Qué son los archivos binarios?](#_wcyiu9ov5rvz) 4

[Ventajas de archivos binarios](#_z43ld5yty7bu) 5

[Diferencias entre funciones de archivos de texto y archivos binarios](#_e4p8t971w1db) 7

[**Actividad**](#_ou1xsbrf0yhv) **10**

[**Conclusiones**](#_3xfajefb45v4) **12**

[**Recomendaciones**](#_53fi7uvqsrja) **12**

[**Bibliografía**](#_yvhmdtww31fd) **13**

# **Objetivos**

El objetivo de esta documentación formal de exposición del tema archivos binarios o directos es el de dar a conocer a nuestros compañeros sobre la importancia y los usos que se pueden dar a los archivos binarios.

Lo que esperamos obtener es un conocimiento más amplio sobre este tema para así poderlo aplicar en la vida diaria y profesional. En el documento se explicará en qué consisten los archivos binarios para que los compañeros puedan usarlo e involucrarse con la actividad realizada.

# 

# **Desarrollo**

## ¿Qué son los archivos binarios?

Los archivos binarios son muy similares a los arreglos de estructuras, excepto que los arreglos están en un archivo de disco en lugar de en un arreglo en la memoria. Debido a que las estructuras en un archivo binario están en el disco, se pueden crear colecciones muy grandes de ellas (limitadas solo por su espacio disponible en el disco). También son permanentes y siempre están disponibles. La única desventaja es la lentitud que proviene del tiempo de acceso al disco.

Los archivos binarios tienen dos características que los distinguen de archivos de texto:

* Puedes saltar instantáneamente a cualquier estructura del archivo, lo cual nos proporciona acceso aleatorio como en un arreglo.
* Se pueden cambiar los componentes de una estructura en cualquier lugar del archivo en cualquier momento

Los archivos binarios también suelen tener tiempos de lectura y escritura más rápidos que los archivos de texto, porque una imagen binaria del registro se almacena directamente de la memoria al disco (o viceversa). En un archivo de texto, todo tiene que convertirse de un lado a otro en texto, y esto lleva tiempo.

C admite el concepto de archivo de estructuras de forma muy limpia. Una vez que abre el archivo, puede leer una estructura, escribir una estructura o buscar cualquier estructura en el archivo. Este concepto de archivo admite el concepto de un puntero de archivo. Cuando se abre el archivo, el puntero apunta al registro 0 (el primer registro del archivo). Cualquier operación de lectura lee la estructura apuntada actualmente y mueve el puntero hacia abajo una estructura. Cualquier operación de escritura escribe en la estructura apuntada actualmente y mueve el puntero hacia abajo una estructura. Buscar mueve el puntero al registro solicitado.

Tenga en cuenta que C piensa en todo en el archivo del disco como bloques de bytes leídos del disco a la memoria o leídos de la memoria al disco. C utiliza un puntero de archivo, pero puede apuntar a cualquier ubicación de byte en el archivo.

Algunas funciones aquí son fread, fwrite y fseek. La función fread toma cuatro parámetros:

* Una dirección de memoria
* El número de bytes a leer por bloque
* El número de bloques para leer.
* La variable de archivo

Los archivos binarios son muy similares a los arreglos, excepto por el hecho de que los arreglos son almacenamiento temporal en la memoria, pero los archivos binarios son almacenamiento permanente en los discos. La diferencia más importante entre los archivos binarios y un archivo de texto es que en un archivo binario, se puede buscar, escribir o leer desde cualquier posición dentro del archivo e insertar estructuras directamente en los archivos.

## Ventajas de archivos binarios

**1. Las operaciones de entrada y salida son mucho más rápidas con datos binarios.**

Por lo general, los archivos de texto grandes contienen millones de números. Se necesita mucho tiempo para convertir enteros de 32 bits en caracteres legibles. Esta conversión no es necesaria en el caso de archivos binarios, ya que los datos pueden almacenarse directamente en forma de bits.

**2. Los archivos binarios son mucho más pequeños que los archivos de texto.**

Para los datos en forma de imágenes, audio o video, esto es muy importante. Tamaño pequeño significa menos espacio de almacenamiento y transmisión más rápida. Por ejemplo, un dispositivo de almacenamiento puede almacenar una gran cantidad de datos binarios en comparación con los datos en formato de caracteres.

**3. Algunos datos no se pueden convertir a formatos de caracteres.**

Por ejemplo, los compiladores de Java generan bytecodes después de la compilación.

Dicho esto, pasemos a manejar las operaciones de E / S en un archivo binario en C. Los parámetros básicos que aceptan las funciones de lectura y escritura de los archivos binarios son:

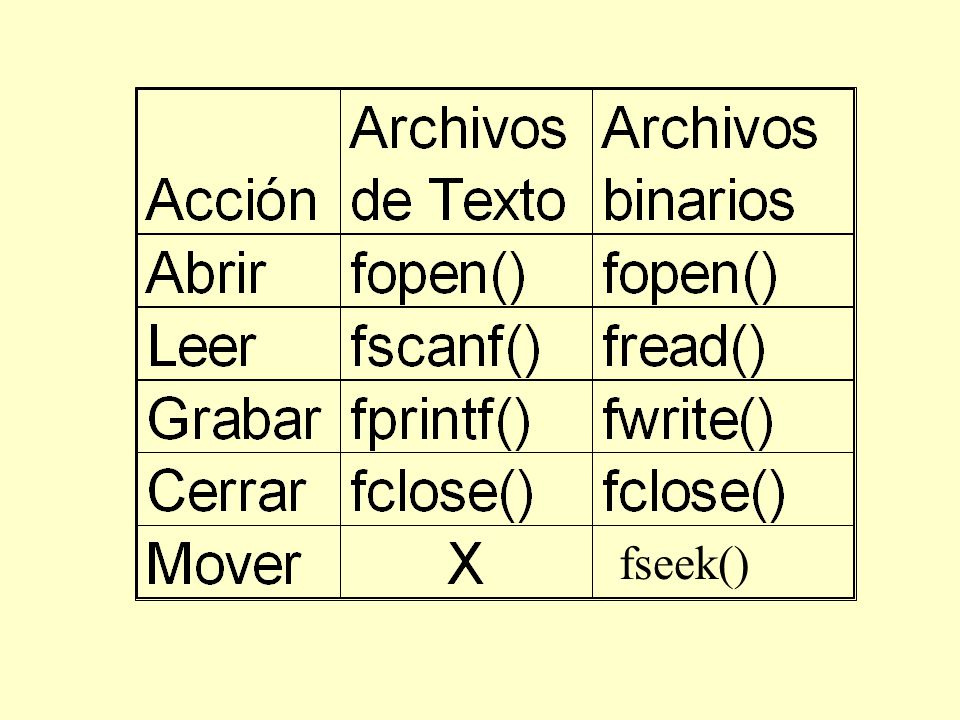
* la dirección de memoria del valor a escribir o leer
* la cantidad de bytes a leer por bloque
* el número total de bloques para leer
* el puntero de archivo

Las bibliotecas C proporcionan funciones para buscar, leer y escribir en archivos binarios.

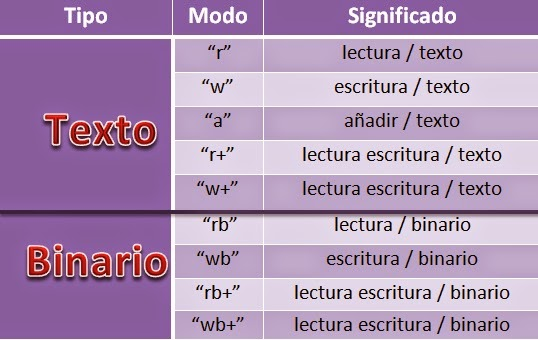
## Diferencias entre funciones de archivos de texto y archivos binarios

**Apertura y Cierre de Archivo Binarios.**

Las operaciones de apertura y cierre para archivos binarios son exactamente las mismas que las vistas para archivos de texto fopen y fclose. También los modos de apertura son los mismos. La única diferencia radica en que hay que añadir una b de "binary" al modo de apertura, es decir, rb, wb, ab, rb+, wb+ y ab+, 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 modo de acceso, las operaciones de lectura y escritura han de ser las específicas para cada tipo de archivo.



**Lectura y Escritura de Archivo 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 archivo binarios esto es innecesario. Por este motivo hay que utilizar otras funciones. 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.

La especificación de las funciones fread y fwrite es la siguiente:

size\_t fread(void \*p, size\_t size, size\_t n, FILE \*pf)

size\_t fwrite(void \*p, size\_t size, size\_t n, FILE \*pf)

Para fread:

* El puntero p apunta a la variable en la cual se almacenarán los datos leídos del 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 han de leerse.
* El cuarto argumento es el puntero o descriptor de fichero que devuelve la función fopen.

Para fwrite:

* El puntero p apunta a la variable en 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.

El tipo de dato size\_t está definido en el fichero <stdio.h>. Su definición es la siguiente:

typedef unsigned size\_t

El tipo size\_t no es más que un entero sin signo; la razón de utilizar este tipo de dato y no la de unsigned directamente no es otra que la de distinguir el tipo de dato cuando se utiliza en la lectura o escritura de archivos binarios.

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

Es posible almacenar varios datos en disco a la vez y no necesariamente uno a uno.

# **Actividad**

La siguiente actividad consiste en crear un archivo binario para poder utilizar y comprender las funciones necesarias para poder manipularlo. Esta actividad ayudará al alumno a obtener una mejor comprensión de manera dinámica e involucrándose en lo explicado. Utilizaremos el programa Dev C++, ya que es un programa eficaz y fácil de manipular.

#include <stdio.h>

/\* random record description - could be anything \*/

struct rec

{

int x,y,z;

};

/\* writes and then reads 10 arbitrary records

from the file "junk". \*/

int main()

{

int i,j;

FILE \*f;

struct rec r;

/\* create the file of 10 records \*/

f=fopen("junk","w");

if (!f)

return 1;

for (i=1;i<=10; i++)

{

r.x=i;

fwrite(&r,sizeof(struct rec),1,f);

}

fclose(f);

/\* read the 10 records \*/

f=fopen("junk","r");

if (!f)

return 1;

for (i=1;i<=10; i++)

{

fread(&r,sizeof(struct rec),1,f);

printf("%d\n",r.x);

}

fclose(f);

printf("\n");

/\* use fseek to read the 10 records

in reverse order \*/

f=fopen("junk","r");

if (!f)

return 1;

for (i=9; i>=0; i--)

{

fseek(f,sizeof(struct rec)\*i,SEEK\_SET);

fread(&r,sizeof(struct rec),1,f);

printf("%d\n",r.x);

}

fclose(f);

printf("\n");

/\* use fseek to read every other record \*/

f=fopen("junk","r");

if (!f)

return 1;

fseek(f,0,SEEK\_SET);

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

{

fread(&r,sizeof(struct rec),1,f);

printf("%d\n",r.x);

fseek(f,sizeof(struct rec),SEEK\_CUR);

}

fclose(f);

printf("\n");

/\* use fseek to read 4th record,

change it, and write it back \*/

f=fopen("junk","r+");

if (!f)

return 1;

fseek(f,sizeof(struct rec)\*3,SEEK\_SET);

fread(&r,sizeof(struct rec),1,f);

r.x=100;

fseek(f,sizeof(struct rec)\*3,SEEK\_SET);

fwrite(&r,sizeof(struct rec),1,f);

fclose(f);

printf("\n");

/\* read the 10 records to insure

4th record was changed \*/

f=fopen("junk","r");

if (!f)

return 1;

for (i=1;i<=10; i++)

{

fread(&r,sizeof(struct rec),1,f);

printf("%d\n",r.x);

}

fclose(f);

return 0;

}

# **Conclusiones**

Aprender sobre los usos de los archivos binarios es de gran utilidad, puesto que nos será de ayuda en un futuro, y de esta manera podremos escribir programas que sean más efectivos y rápidos a la hora de su creación. Contar con este conocimiento a la hora de programar desarrollará nuestra lógica y nuestra facilidad para escribir códigos cada vez más complejos.

# **Recomendaciones**

Para poder dominar el tema presentado, es necesario involucrarse durante la realización de la actividad del tema y de las demás prácticas que se realicen en el aula, para así adquirir el conocimiento necesario que servirá a corto y largo plazo. A su vez, es necesario practicar en casa y leer independientemente, así como preguntar dudas durante las horas de clase para que el tema quede totalmente comprendido.

Al programar en C, se recomiendan los siguientes IDEs:

* Dev C++
* NetBeans
* Eclipse
* Visual Studio Code

# **Bibliografía**

Moisse, D. (s.f.). Archivos binarios: creación y grabación de tipo de datos primitivos (fopen, fwrite, fclose). Recuperado 11 noviembre, 2019, de <https://www.tutorialesprogramacionya.com/cya/detalleconcepto.php?punto=71&codigo=71&inicio=60>

Pollette, C., & Brain, M. (s.f.). Binary Files. Recuperado 11 noviembre, 2019, de <https://computer.howstuffworks.com/c39.htm>

Goswami, M. (s.f.). Writing & Reading Binary Files in C Programming - Video & Lesson Transcript | Study.com. Recuperado 11 noviembre, 2019, de <https://study.com/academy/lesson/writing-reading-binary-files-in-c-programming.html>