



Programación avanzada en C

Archivos

Objetivos

El alumno conocerá y aplicará el concepto de archivo para el almacenamiento y recuperación de datos persistentes.

Abrir archivos y almacenar su contenido en algún tipo de dato en C que más facilite su posterior manipulación junto con otros datos dentro de un programa.

*Escribir en archivos *.txt y *.xls.*

Qué necesitamos para trabajar con archivos?

Un archivo.

Un **apuntador** a un archivo. **FILE *archivo;**

Función de apertura indicando **nombre** del archivo y **modo** de acceso.

Validar que el acceso es permitido.

Función para **leer** carácter por carácter, por línea completa o con formato.

Función para **escribir** carácter por carácter, por línea completa o con formato.

Cerrar el archivo.

Funciones en archivos

Nombre	Función
fopen()	Abre un archivo.
fclose()	Cierra un archivo.
fgets()	Lee una cadena de un archivo.
fputs()	Escribe una cadena en un archivo
fseek()	Busca un byte específico de un archivo.
fprintf()	Escribe una salida con formato en el archivo.
fscanf()	Lee una entrada con formato desde el archivo.
feof()	Devuelve cierto si se llega al final del archivo.
ferror()	Devuelve cierto si se produce un error.
rewind()	Coloca el localizador de posición del archivo al principio del mismo.
remove()	Borra un archivo.
fflush()	Vacía un archivo.

Apertura del archivo

`FILE *archivo= NULL; //es como un alias para el archivo`
`archivo= fopen("nuevos.txt", "r");`

Modo	Significado
r	Abre un archivo de texto para lectura.
w	Crea un archivo de texto para escritura.
a	Abre un archivo de texto para añadir.
rb	Abre un archivo binario para lectura.
wb	Crea un archivo binario para escritura.
ab	Abre un archivo binario para añadir.
r+	Abre un archivo de texto para lectura / escritura.
w+	Crea un archivo de texto para lectura / escritura.
a+	Añade o crea un archivo de texto para lectura / escritura.
r+b	Abre un archivo binario para lectura / escritura.
w+b	Crea un archivo binario para lectura / escritura.
a+b	Añade o crea un archivo binario para lectura / escritura.

Lectura carácter por carácter

fgetc sin ciclo

```
char c;
```

```
c=fgetc(archivo);  
printf("%c", c);
```

```
c= fgetc(archivo);  
printf("%c", c);
```

Como solo trae un carácter se debe **repetir** la función para cada carácter que se encuentre en el archivo pero se sobrescribe.

Hecho por Huicho :)

fgetc con ciclo

```
char c;
```

```
while(feof(archivo)==0)  
//while(!feof(archivo))  
{  
    c= fgetc(archivo);  
    printf("%c", c);  
}
```

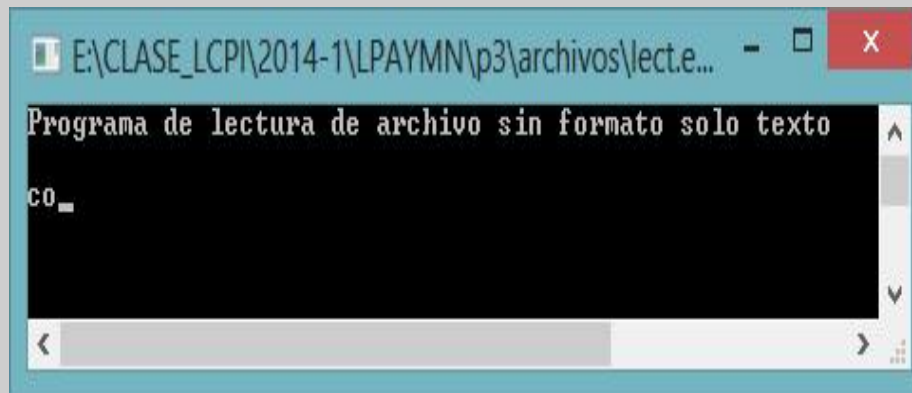
El ciclo se encarga de repetir la instrucción aunque también se sobrescribe.

La función feof devuelve 0 si no ha llegado al final del archivo.

Lectura carácter por carácter

fgetc sin ciclo


```
char c;  
  
c=fgetc(archivo);  
printf("%c", c);  
  
c=fgetc(archivo);  
printf("%c", c);
```



```
E:\CLASE_LCPI\2014-1\LPAYMN\p3\archivos\lect.e...  
Programa de lectura de archivo sin formato solo texto  
co_
```

fgetc con ciclo

```
char c;  
  
while(feof(archivo)==0)  
    //while(!feof(archivo))  
{  
    c=fgetc(archivo);  
    printf("%c", c);  
}
```



```
E:\CLASE_LCPI\2014-1\LPAYMN\p3\archivos\3_1...  
Programa de lectura de archivo sin formato solo texto  
cosas  
casas  
quesos  
minutos horas segundos
```

Lectura carácter por carácter

```
#include <stdio.h>

int main(int argc, char *argv[])
{
    FILE *archivo=NULL;
    char c='\0';

    printf("Programa de lectura de archivo sin formato solo texto\n\n");

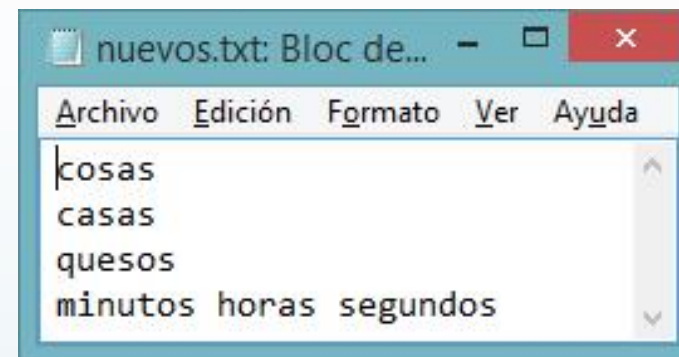
    //se abre el archivo para solo lectura
    archivo=fopen("nuevos.txt", "r");

    //ciclo q toma un caracter a la vez mientras no se llegue al final
    //feof devuelve cero mientras no se llegue al final del archivo,
    // cuando llega se obtiene el valor verdadero
    while(feof(archivo)==0)
    //while(!feof(archivo))
    {
        c=fgetc(archivo);
        printf("%c", c);
    }

    //se cierra el archivo
    fclose(archivo);

    getchar();
    return 0;
```

Hecho por Huicho :)



Al recibir el dato siempre en la variable "c" se pierde el dato anterior para almacenar el nuevo que trae la función fgetc

Validación en la apertura del archivo

Se debe validar si el archivo puede ser abierto para evitar terminación inesperada del programa ya sea porque:

Está siendo usado por otro programa.

No existe.

No se encuentra en la ruta indicada.

```
if((archivo=fopen("nuevos.txt","r"))==NULL)
//if(!(archivo=fopen("nuevos.txt","r")))
{
    printf("Error al abrir");
    getchar();
    exit(0); //termina la ejecución justo ahí
}
else
{
    código para la lectura del archivo
}
```

```
#include <stdio.h>
#include <stdlib.h> //para exit

int main(int argc, char *argv[])
{
    FILE *archivo=NULL;
    char c='\0';

    printf("Programa de lectura de archivo sin formato solo texto\n\n");

    //validar si el archivo no pudo ser abierto, caso contrario continuar
    if((archivo=fopen("nuevos.txt", "r"))==NULL)
    {
        printf("Error al abrir");
        getchar();
        exit(0); //termina la ejecucion justo ahi
    }
    else
    {
        //ciclo q toma un caracter a la vez mientras no se llegue al final
        //feof devuelve cero mientras no se llegue al final del archivo,
        // cuando llega se obtiene el valor verdadero
        while(feof(archivo)==0)
        //while(!feof(archivo))
        {
            c=fgetc(archivo);
            printf("%c", c);
        }

        //se cierra el archivo
        fclose(archivo);
    } //cierre del else

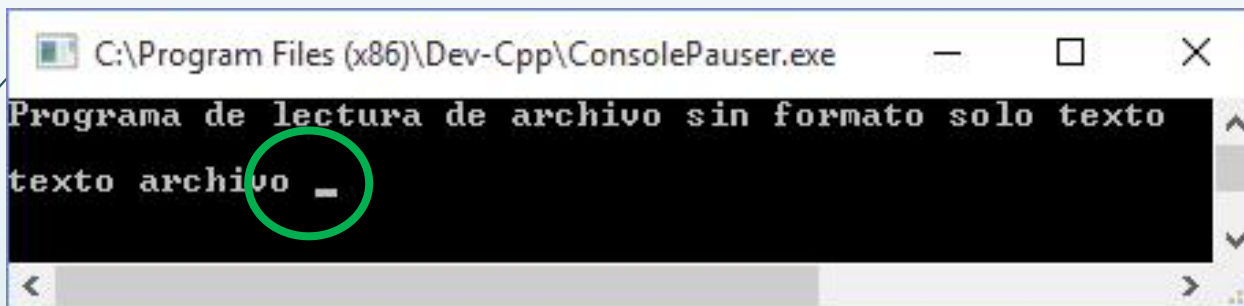
    getchar();
    return 0;
}
```

L e c t u r a carácter por carácter con validación de apertura

La apertura del archivo debe colocarse entre paréntesis para poder compararse correctamente con NULL dentro del if de lo contrario el compilador devolverá un error.

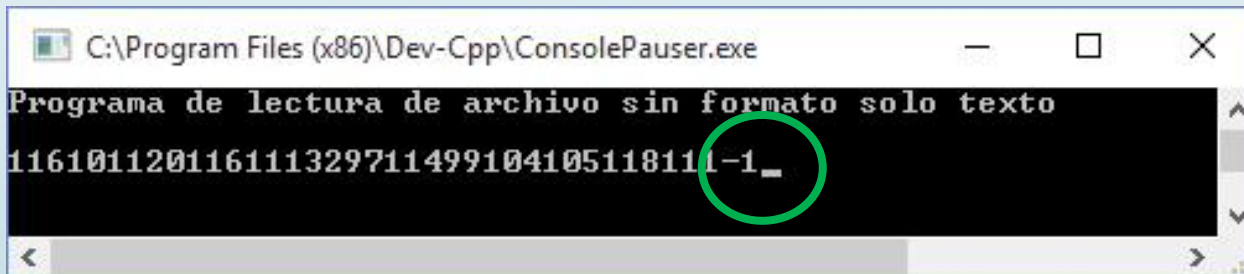
Problema en la lectura con fgetc

Al ejecutar el programa se observa un espacio después del último carácter que contiene el archivo de texto, éste corresponde al carácter identificador de final de archivo con valor -1

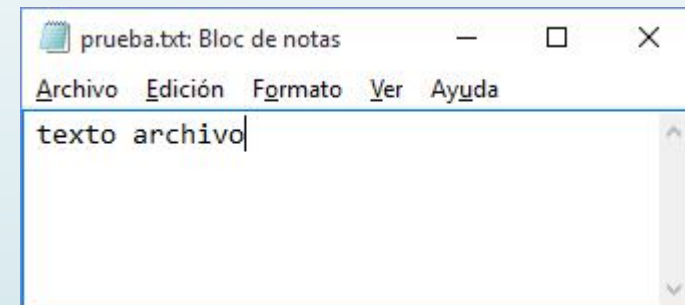


```
C:\Program Files (x86)\Dev-Cpp\ConsolePauser.exe
Programa de lectura de archivo sin formato solo texto
texto archivo _
```

Impresión del texto como ASCII:



```
C:\Program Files (x86)\Dev-Cpp\ConsolePauser.exe
Programa de lectura de archivo sin formato solo texto
116101120116111329711499104105118111-1_
```



```
prueba.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda
texto archivo|
```

```
#include <stdio.h>
#include <stdlib.h> //para exit
#include <locale.h> //para setlocale

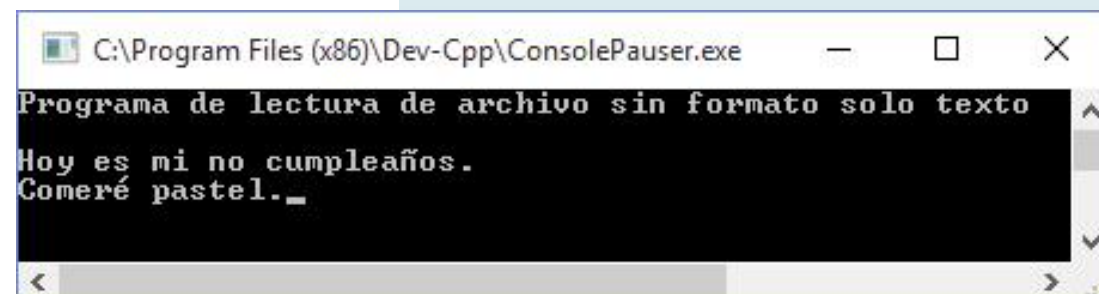
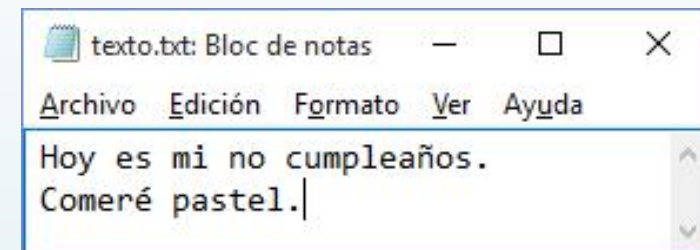
int main(int argc, char *argv[])
{
    setlocale(LC_ALL, "Spanish_Mexico");

    FILE *archivo=NULL;
    unsigned char c='\0';

    printf("Programa de lectura de archivo sin formato solo texto\n\n");

    //validar si el archivo no pudo ser abierto, caso contrario continuar
    if((archivo=fopen("texto.txt", "r"))==NULL)
    {
        printf("Error al abrir");
        getchar();
        exit(0); //termina la ejecucion justo ahi
    }
    else
    {
        c= fgetc(archivo);
        //ciclo q toma un caracter a la vez mientras no se llegue al final
        //feof devuelve cero mientras no se llegue al final del archivo,
        // cuando llega se obtiene el valor verdadero
        while(feof(archivo)==0)
        //while(!feof(archivo))
        {
            printf("%c", c);
            c= fgetc(archivo);
        }
        fclose(archivo); //se cierra el archivo
    } //cierre del else
    getchar();
    return 0;
}
```

Lectura carácter por carácter completa




```

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

int main(int argc, char *argv[])
{
    FILE *archivo=NULL;
    int contadora= 0;
    char arreglo[100];
    char c='\0';

    printf("Programa de lectura de archivo sin formato a un arreglo\n\n");
    //validar si el archivo no pudo ser abierto, caso contrario continuar
    if((archivo=fopen("origen1.txt", "r"))==NULL)
    //if(!(archivo=fopen("origen1.txt", "r")))
    {
        printf("Error al abrir");
        getchar();
        exit(0);
    }
    else
    {
        c= fgetc(archivo);
        while(feof(archivo)==0)
        //while(!feof(archivo))
        {
            arreglo[contadora]= c;
            printf("%c", arreglo[contadora]);
            contadora++;
            c=fgetc(archivo);
        }
        arreglo[contadora]= '\0'; //colocar caracter de fin de cadena
        printf("\n\nArchivo guardado en arreglo: \n%s",arreglo);
    }
    fclose(archivo); //se cierra archivo
    getchar();
    return 0;
}

```

Lectura carácter por carácter a un arreglo

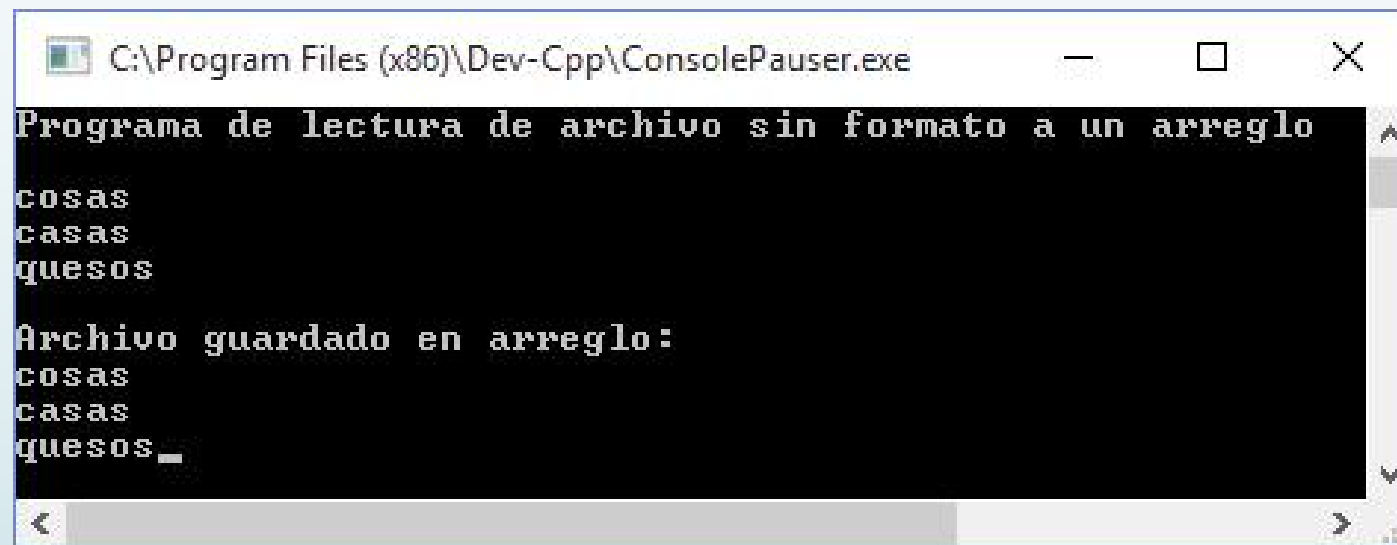
Guardar en un arreglo para evitar que se sobrescriba la variable.

La variable **contadora** se **incrementa** en uno para que el nuevo carácter leído se almacene en el siguiente cuadrado.

Asignar carácter de terminación al final para que se guarde como cadena.

Lectura carácter por carácter a un arreglo

También se inicia con una llamada a la función `fgetc` fuera del ciclo para evitar imprimir y guardar el carácter de final de archivo.



```
C:\Program Files (x86)\Dev-Cpp\ConsolePauser.exe
Programa de lectura de archivo sin formato a un arreglo
cosas
casas
quesos

Archivo guardado en arreglo:
cosas
casas
quesos_
```

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

int main(int argc, char *argv[])
{
    FILE *archOrigen=NULL;
    FILE *archDest=NULL;
    char c='\0';

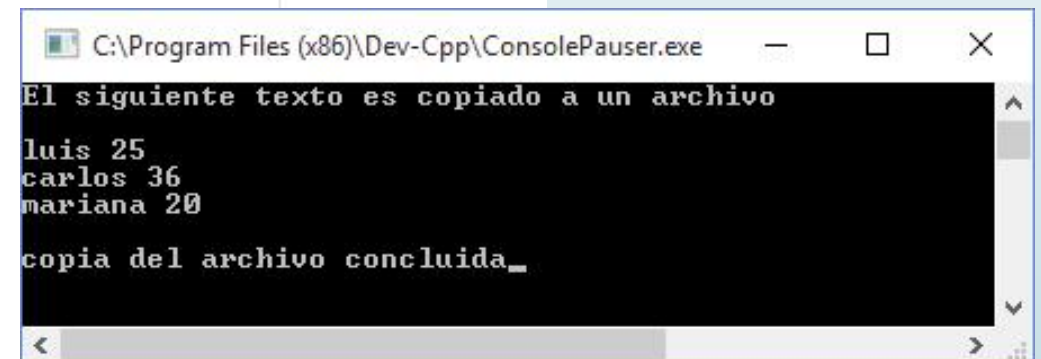
    if((archOrigen=fopen("origen.txt","r"))==NULL || (archDest=fopen("destino.txt","w"))==NULL)
        //if(!(archOrigen=fopen("origen.txt","r")) || !(archDest=fopen("destino.txt","w")))
    {
        printf("Error al abrir");
        getchar();
        exit(0);
    }
    else
    {
        printf("El siguiente texto es copiado a un archivo\n\n");

        c=fgetc(archOrigen);
        while(!feof(archOrigen))
            //while(!feof(archOrigen))
        {
            fputc(c,archDest);
            printf("%c", c);
            c=fgetc(archOrigen);
        }

        printf("\n\ncopia del archivo concluida");
    }
    fclose(archOrigen); //se cierra el archivo origen
    fclose(archDest); //se cierra archivo destino
    getchar();
    return 0;
}
```

Lectura y escritura carácter por carácter

La función `fputc` permite escribir al archivo el carácter pasado como argumento.



A screenshot of a Windows console window titled "C:\Program Files (x86)\Dev-Cpp\ConsolePauser.exe". The window displays the output of the C program. It shows the message "El siguiente texto es copiado a un archivo" followed by three lines of text: "luis 25", "carlos 36", and "mariana 20". Below these, it shows "copia del archivo concluida_" with a cursor. The window has standard Windows controls (minimize, maximize, close) and a scrollbar on the right.

```

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

int main(int argc, char* argv[])
{
    FILE *archivo=NULL;
    char nombre[20];
    int edad=0;
    int datos=0;

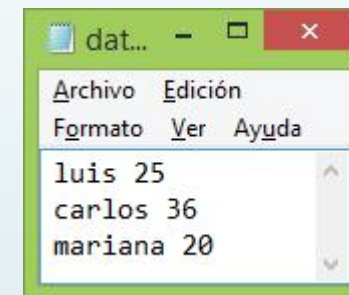
    if((archivo=fopen("datos.txt","r"))==NULL)
        //if(!(archivo=fopen("datos.txt","r")))
    {
        printf("Error al abrir");
        getchar();
        exit(0);
    }
    else
    {
        while(feof(archivo)==0)
            //while(!feof(archivo))
        {
            //fscanf devuelve un entero con el numero de elementos encontrados
            datos=fscanf(archivo,"%20s %2d",nombre,&edad);

            if(datos>0)
                printf("Nombre: %s Edad: %d Datos: %d\n",nombre,edad,datos);
        }
        fclose(archivo);
    }
    getchar();
    return 0;
}

```

Lectura con formato

Se indica como viene el renglón del archivo, como una cadena, espacio y un entero:



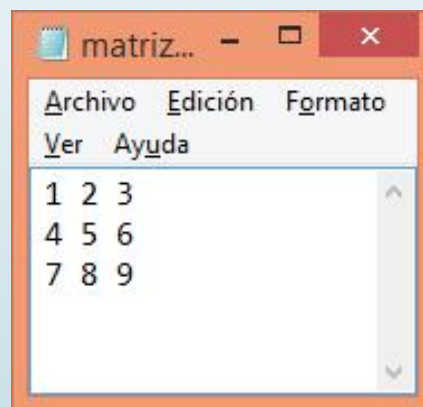
fscanf lee el renglón completo y devuelve el número de elementos encontrados.

Lectura y escritura con formato

Se validan los dos archivos.

La función `fscanf` permite la lectura con formato de cada renglón completo del archivo para almacenarlo directamente en una matriz sin tener que convertir o hacer operaciones extra.

La función `fprintf` escribe hacia el archivo con el mismo formato que se manda a la consola.



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

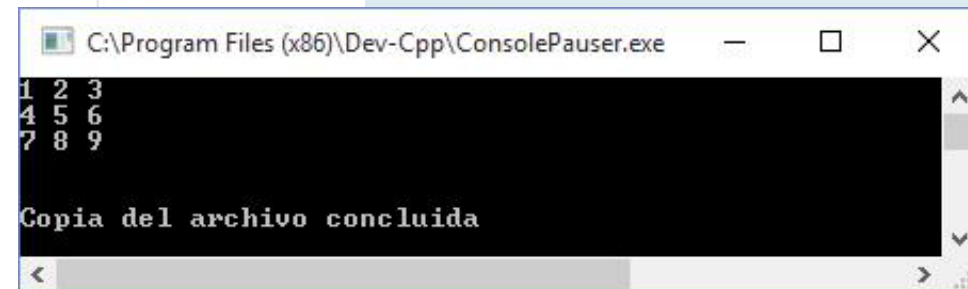
int main(int argc, char* argv[])
{
    //apuntadores a archivos origen y destino
    FILE *archOrigen=NULL;
    FILE *archDest=NULL;

    int matriz[3][3];
    int i=0;

    if((archOrigen=fopen("matriz1.txt","r"))==NULL || (archDest=fopen("matriz2.txt","w"))==NULL)
    //if(!(archOrigen=fopen("matriz1.txt","r")) || !(archDest=fopen("matriz2.txt","w")))
    {
        printf("Error al abrir");
        getchar();
        exit(0);
    }
    else
    {
        while(feof(archOrigen)==0)
        //while(!feof(archOrigen))
        {
            fscanf(archOrigen,"%d %d %d",&matriz[i][0],&matriz[i][1],&matriz[i][2]);
            printf("%d %d %d\n",matriz[i][0],matriz[i][1],matriz[i][2]);
            fprintf(archDest,"%d %d %d\n",matriz[i][0],matriz[i][1],matriz[i][2]);
            i++;
        }
        fclose(archOrigen);
        fclose(archDest);

        printf("\n\nCopia del archivo concluida");
    }
    getchar();
    return 0;
}
```

Lectura y escritura con formato



```
C:\Program Files (x86)\Dev-Cpp\ConsolePauser.exe
1 2 3
4 5 6
7 8 9

Copia del archivo concluida
```

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

int main(int argc, char*argv[])
{
    FILE *archivo;
    FILE *archivo_dest;

    int datos=0;

    struct
    {
        char nombre[50];
        int edad;
    }alumno;

    if((archivo=fopen("alumno.txt","r"))==NULL || (archivo_dest=fopen("alumno_dest.txt","w"))==NULL)
        //if(! (archivo=fopen("datos.txt","r")))
        {
            printf("Error al abrir");
            getchar();
            exit(0);
        }
    else
    {
        //fscanf(archivo,"%20s %2d",nombre,&edad);

        while(feof(archivo)==0)
            //while(!feof(archivo))
            {
                //printf("Nombre: %s Edad: %d\n",nombre,edad);
                //fscanf devuelve un entero con el numero de elementos encontrados
                datos=fscanf(archivo,"%20s %2d",alumno.nombre,&alumno.edad);

                if(datos>0)
                {
                    printf("Nombre: %s Edad: %d\n",alumno.nombre,alumno.edad);
                    fprintf(archivo_dest,"Nombre: %s Edad: %d\n",alumno.nombre,alumno.edad);
                }
            }
        fclose(archivo);
        fclose(archivo_dest);
        printf("\n\n Archivo de destino guardado!");
    }

    getchar();
    return 0;
}
```

Archivos y estructuras



Funciones comunes en archivos

FILE

C define la estructura de datos FILE en el fichero de cabecera stdio.h para el manejo de ficheros. La definición de ésta estructura depende del compilador, pero en general mantienen un campo con la posición actual de lectura/escritura, un buffer para mejorar las prestaciones de acceso al fichero y algunos campos para uso interno.

FUNCIÓN: fopen

Sintaxis

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

Esta función sirve para abrir y crear ficheros en disco. El valor de retorno es un puntero a una estructura [FILE](#)

Parámetros de entrada de fopen

Nombre: una cadena que contiene un nombre de fichero válido, el nombre puede incluir el camino completo

Modo: especifica en tipo de fichero que se abrirá o se creará y el tipo de datos que puede contener, de texto o binarios:

Parámetros de entrada de fopen

r	Solo lectura (fichero debe existir)
w	Se abre para escritura
a	Añadir, cursor se situa al final del fichero y si no existe se crea
r+	Lectura y escritura (el fichero debe existir)
w+	Lectura y escritura (se crea fichero nuevo o sobrescribe existente)
a+	Añadir, lectura y escritura (cursor se situa al final del fichero)
t	Tipo de texto si no especifica por defecto es “t”
b	Tipo binario

FUNCIÓN: fclose

Sintaxis

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

Esta función sirve para cerrar un fichero, ya que almacena los datos que aún están en el buffer de memoria, y actualiza algunos datos de la cabecera del fichero que mantiene el sistema operativo.

Un valor de retorno cero indica que el fichero ha sido correctamente cerrado, si ha habido algún error, el valor de retorno es la constante EOF.

FUNCIÓN: fgetc

Sintaxis

```
int fgetc (FILE *fichero);
```

Esta función lee un carácter desde un fichero. El valor de retorno es el carácter leído como un unsigned char convertido a int, si no hay ningún carácter disponible, el valor de retorno es EOF

FUNCIÓN: fputc

Sintaxis

```
int fputc(int carácter, FILE *fichero);
```

Esta función escribe un carácter a un fichero. El valor de retorno es el carácter escrito, si la operación fue completada con éxito, en caso contrario será EOF.

FUNCIÓN: feof

Sintaxis

```
int feof(FILE *fichero);
```

Esta función sirve para comprobar si se ha alcanzado el final del fichero. La forma que suelen tener los bucles para leer todos los datos de un archivo es permanecer leyendo mientras no se detecte el fin del fichero. El valor de retorno es distinto de cero sólo si no se ha alcanzado el fin del fichero.

FUNCIÓN: rewind

Sintaxis

```
void rewind(FILE *fichero);
```

Es una función heredada de los tiempos de las cintas magnéticas, rebobinar, es decir que para volver al principio de un archivo almacenado en cinta. Sitúa al cursor de lectura/escritura al principio del archivo

FUNCIÓN: fgets

Sintaxis

```
char *fgets (char *cadena, int n, FILE *fichero);
```

Esta función está diseñada para leer cadenas de caracteres. Leerá hasta $n-1$ caracteres o hasta que lea un retorno de línea (este último carácter también es leído). El parámetro n nos permite limitar la lectura para evitar desbordar el espacio disponible en la cadena.

El valor de retorno es un puntero a la cadena leída, si se leyó con éxito, y es NULL si se detecta al final del fichero o si hay un error.

FUNCIÓN: fputs

Sintaxis

```
int fputs (const char *cadena, FILE *stream);
```

Esta función escribe una cadena en un fichero. No se añade el carácter de retorno de línea ni el carácter nulo final.

El valor de retorno es un número no negativo o EOF en caso de error.

FUNCIÓN: fread

Sintaxis

```
size_t fread(void *puntero, size_t tamaño, size_t nregistros, FILE *fichero);
```

Esta función está pensada para trabajar con registros de longitud constante. Es capaz de leer desde un fichero uno o varios registros de la misma longitud y a partir de una dirección de memoria determinada.

El valor de retorno es el número de registros leídos, no el número de bytes

FUNCIÓN: fwrite

Sintaxis

```
size_t fwrite(void *puntero, size_t tamaño, size_t nregistros, FILE *fichero);
```

Esta función también está pensada para trabajar con registros de longitud constante y forma pareja con fread. Es capaz de escribir hacia un fichero uno o varios registros de la misma longitud almacenados a partir de una dirección de memoria determinada.

El valor de retorno es el número de registros escritos, no el número de bytes.

FUNCIÓN: fprintf

Sintaxis

```
int fprintf(FILE *fichero, const char *formato, ...);
```

Esta función funciona igual que printf en cuanto a parámetros, pero la salida es dirigida a un fichero en lugar de a la pantalla

FUNCIÓN: fscanf

Sintaxis

```
int fscanf(FILE *fichero, const char *formato, ....);
```

Esta función funciona igual que scanf en cuanto a parámetros, pero la entrada se toma de un fichero en lugar del teclado.

FUNCIÓN: fflush

Sintaxis

```
int fflush(FILE *fichero);
```

Esta función fuerza la salida de los datos acumulados en el buffer de salida del fichero. Para mejorar las prestaciones del manejo de ficheros se utilizan buffers, almacenes temporales de datos en memoria, las operaciones de salida se hacen a través del buffer, y sólo cuando el buffer se llena se realiza la escritura en el disco y se vacía el buffer.

El valor de retorno es cero si la función se ejecutó con éxito, y EOF si hubo algún error.



FUNCIONES ESPECÍFICAS PARA FICHEROS DE ACCESO ALEATORIO

FUNCIÓN: fseek

Sintaxis

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

Esta función sirve para situar el cursor del fichero para leer o escribir en el lugar deseado.

El valor de retorno es cero si la función tuvo éxito, y un valor distinto de cero si hubo algún error.

FUNCIÓN: `fseek`

El parámetro origen puede tener tres posibles valores:

1. `SEEK_SET` el desplazamiento se cuenta desde el principio del fichero.
El primer byte tiene un desplazamiento de cero
2. `SEEK_CUR` el desplazamiento se cuenta desde la posición actual del cursor
3. `SEEK_END` el desplazamiento se cuenta desde el final del fichero

FUNCIÓN: ftell

Sintaxis

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

La función ftell sirve para averiguar la posición actual del cursor de lectura/ escritura de un fichero.

El valor de retorno será esa posición, o -1 si hay algún error