Licenciatura en Sistemas
Programación de computadoras

Equipo docente: Jorge Golfieri, Natalia Romero, Romina Masilla y Nicolás Perez
Mails: jgolfieri@hotmail.com , nataliab_romero@yahoo.com.ar , romina.e.mansilla@gmail.com, nperez_dcao_smn@outlook.com
Facebook: https://www.facebook.com/groups/171510736842353
Git: ***http://***github.com/UNLASistemasPrograma cion/Programacion-de-Computadoras

---

Unidad Extra:

Punteros. Concepto. Tipos de punteros. Punteros como argumentos de funciones. Manejo dinámico de memoria (malloc, free). Estructuras dinámicas. Problemas: implementación y testeo.

Bibliografía citada:

Ceballos, Francisco Javier. C/C++ Curso de Programación. 2da Edición. Editorial RA-MA, 2002.

Luis Joyanes Aguilar, Andrés Castillo Sanz, y Lucas Sánchez García (2005) - C algoritmos, programación y estructuras de datos - McGraw-Hill España.

Gottfried, Byron S. Programación en C. McGraw Hill, 1991.

Archivos – Guía Teórica

Los datos que hemos tratado hasta el momento han residido en la **memoria principal.** Sin embargo, las grandes cantidades de datos se almacenan normalmente en un dispositivo de memoria secundaria. Los **ficheros son estructuras de datos almacenadas en memoria secundaria.**

Hay dos tipos de archivos, **archivos de texto y archivos binarios**.

Un archivo de texto es una secuencia de caracteres organizadas en líneas terminadas por un carácter de nueva línea. Los archivos de texto se caracterizan por ser planos, es decir, **todas las letras tienen el mismo formato y no hay palabras subrayadas, en negrita, o letras de distinto tamaño.**

Un archivo binario es una secuencia de bytes que tienen una correspondencia uno a uno con un dispositivo externo. Así que no tendrá lugar ninguna traducción de

caracteres. Además, el número de bytes escritos (leídos) será el mismo que los encontrados en el dispositivo externo. Ejemplos de estos archivos son: Fotografías, imágenes, texto con formatos, archivos ejecutables (aplicaciones), etc.

Los archivos en C:

En C, **un archivo es un concepto lógico** que puede aplicarse a muchas cosas desde archivos de disco hasta terminales o una impresora. Se asocia una secuencia con un archivo específico realizando una operación de apertura. Una vez que el archivo está abierto, la información puede ser intercambiada entre este y el programa. Se puede conseguir la entrada y la salida de datos a un archivo a través del uso de la biblioteca de funciones; C no tiene palabras claves que realicen las operaciones de E/S.

El estándar de C contiene funciones para la edición de ficheros, estas están definidas en la cabecera **"stdio.h"** y por lo general empiezan con la letra **f**, haciendo referencia a FILE. Adicionalmente se agrega un tipo **FILE**, el cual se usará como apuntador a la información del fichero.

Funciones para manejo de 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 especifico 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.

¿Cómo manejar un archivo?:

La secuencia que usaremos para realizar operaciones será la siguiente:

- Crear un apuntador del tipo **FILE *.**
- Abrir el archivo utilizando la función **fopen** y asignándole el resultado de la llamada a nuestro apuntador.
- Hacer las diversas operaciones (lectura, escritura, etc).
- Cerrar el archivo utilizando la función **fclose**.

¿Por qué utilizar un puntero?:

Un puntero a un archivo es un puntero a una información que define varias cosas sobre él, incluyendo el nombre, el estado y la posición actual del archivo. En esencial identificar un archivo especifico y utilizar la secuencia asociada para dirigir el funcionamiento de las funciones de E/S con buffer. Es una **variable de tipo puntero a la estructura FILE** que se define en **"stdio.h"** para el manejo de ficheros. Un programa necesita utilizar punteros a archivos para leer o escribir en los mismos. La definición de la estructura **FILE** 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, al programador le interesa su uso como estructura FILE.

¿Cómo abrir un archivo?:

Para poder operar con un fichero, exista previamente o no, es necesario "abrirlo" mediante la función **fopen**.

El prototipo de dicha función es: **FILE *fopen (const char *filename, const char *mode);**

Respecto a este prototipo:

Nótese que fopen devuelve un valor de tipo **FILE *** (o sea, un puntero a FILE). Por supuesto, se supone que el valor devuelto será asignado a una variable de tipo FILE *, que se usará en otras funciones que manipulen dicho fichero.
Si la apertura del fichero falla, fopen devuelve un puntero nulo.
El argumento **filename** (una cadena de caracteres) es el nombre "real" del fichero que va a abrirse mediante fopen (es decir, es el nombre con el que el fichero aparece en el disco).
El argumento **mode** (una cadena de caracteres): indica qué tipo de operaciones se realizarán sobre el fichero abierto y el tipo de datos que puede contener, de texto o binarios:.

Los posibles valores de mode son:

r: El fichero se abre para lectura. Si el fichero no existe, se devuelve un puntero nulo.
w: Se crea el fichero para escritura. Si ya existe un fichero con ese nombre, el fichero antiguo será eliminado.
a: Si ya existe un fichero con ese nombre, se abre para escritura (al final del fichero). Si

no existe, se crea.
r+: Si el fichero existe, se abre para lectura y escritura (al principio del fichero).
w+: Se crea el fichero para lectura y escritura. Si ya existe un fichero con ese nombre, el fichero antiguo será eliminado.
a+: Si el fichero existe, se abre para lectura y escritura (al final del fichero). Si el fichero no existe, se crea.

En estos modos no se ha establecido el tipo de archivo, para ello se utilizará t para especificar un archivo de texto o b para binario.

t: tipo texto, si no se especifica "t" ni "b", se asume por defecto que es **"t".**
b: tipo binario.

Es decir: **"rt", "wt", "at", "r+t", "w+t", "a+t"** o bien **"rb", "wb", "ab", "r+b", "w+b", "a+b"**

¿Cómo cerrar el archivo?:

Cerrar un fichero 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. Además permite que otros programas puedan abrir el fichero para su uso. Muy a menudo, los ficheros no pueden ser compartidos por varios programas.

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**. El parámetro es un puntero a la estructura FILE del fichero que queremos cerrar.

Para cerrar un fichero, se usa la función **fclose**.

Lectura y escritura:

Las funciones empleadas con ficheros en formato texto son:
Para lectura y escritura con formato: fscanf y fprintf.
Para leer y escribir un carácter: fgetc y fputc.
Para leer y escribir una cadena: fgets y fputs.

Para leer y escribir cualquier dato con formato en un fichero de texto se emplean las funciones fscanf y fprintf. Los prototipos de dichas funciones son:

```
int fscanf(FILE *fp, const char *fmt,…);
int fprintf(FILE *fp, const char *fmt,…);
```

El valor entero devuelto por ambas funciones indica si la operación se ha llevado a cabo con éxito.

Ejemplo 1: Programa en C que lee dos caracteres de un archivo y los copia en otro.

```
#include <stdio.h>

#include <stdlib.h>

int main()

{

FILE *fp1,*fp2; char c1,c2;


fp1=fopen("archivo.txt","r"); if(fp1==NULL)

{

printf("Error al abrir el archivo para leer"); system("PAUSE");

exit(1);

}


fp2=fopen("copia.txt","w");

if(fp2==NULL)

{

printf("Error al abrir el archivo copia.txt"); system("PAUSE");

exit(1);
```

```
}

fscanf(fp1,"%c%c",&c1,&c2);
fprintf(fp2,"%c%c",c1,c2);



fclose(fp1);
fclose(fp2);
return 0;
}
```

Ejemplo 2: Programa en C que escribe un carácter leído del teclado en un fichero.

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

int main()
{
FILE *fp; char caracter;
printf("\nIntroduce caracter:\n");
 scanf("%c",&caracter);
 fp=fopen("caracter.txt","w");
fputc(caracter,fp);
fclose(fp);
printf("\nCaracter guardado en el fichero!!\n");
system("PAUSE");
return 0;
}
```

Ejemplo 3: Programa en C que imprime un carácter leído de un fichero.

```
#include <stdio.h>

int main()
{
FILE *fp;
char leido;
fp=fopen("archivo.txt","r");
leido=fgetc(fp);
fclose(fp);

printf("\nEl Caracter Leido es: %c\n",leido);
 system("PAUSE");
return 0;
}
```

¿Cómo leer o escribir cadenas?:

Para leer y escribir una cadena de caracteres en un fichero se emplean las funciones fgets **y fputs**. Los prototipos de dichas funciones son:

```
int fgetc(FILE *fp);

int fputc(int c,FILE *fp);
```

Respecto a estos prototipos:

fgets lee una serie de caracteres de un fichero y los asigna a una cadena.
El nombre de la cadena debe indicarse como primer argumento, y el tamaño de la misma como segundo argumento.
Si se indica un tamaño de cadena **n** se leerán n-1 caracteres del fichero, salvo que se llegue a un carácter \n, en cuyo caso se detiene la lectura.
El valor devuelto por **fgets** es el primer argumento (el puntero a la cadena) si la

operación se lleva a cabo con éxito, y un puntero nulo si la operación falla.
fputs copia los caracteres que forman la cadena (salvo el carácter de terminación) indicada como primer argumento en el fichero.
fputs devuelve el valor ASCII del último carácter escrito si la operación se lleva a cabo con éxito.

Ejemplo 4: Programa en C que muestra la lectura y escritura de una cadena de N caracteres en un fichero.

```
#include <stdio.h>

int main()
{
FILE *fp;
char cadena[]="Algoritmo y Estructura de Datos",leida[20];
fp=fopen("ejemplo5.txt","w");
fputs(cadena,fp); fclose(fp);
fp=fopen("ejemplo5.txt","r");
fgets(leida,sizeof(leida)/sizeof(char),fp); fclose(fp);
printf("\nCadena leida: %s\n",leida); system("PAUSE");

return 0;
}
```

Archivos – Guía Práctica

Ejercicio 1: Crea un programa que vaya leyendo las frases que el usuario teclea y las guarde en un fichero de texto llamado "registroDeUsuario.txt". Terminará cuando la frase introducida sea "fin" (esa frase no deberá guardarse en el fichero).

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

int main()
{
  FILE* ptFichero;
  char fin[]="fin";
```

```
  char frase[60];

  ptFichero = fopen("registroDeUsuario.txt", "wt");
  printf(" PROGRAMA para ESCRIBIR FRASES.\nCuando quiera salir,"
       "escriba la palabra fin.\n\n");
  do
  {
    puts("\nEscriba una FRASE:\n(o fin). \n");
    gets(frase);
    if (strcmp(frase, fin) == 0)
      break;
    fprintf(ptFichero, "%s\n", frase);
  }
  while (strcmp(frase, fin) != 0);

  fclose(ptFichero);
  return 0;
}
```

Ejercicio 2: **Un programa que pida al usuario que teclee frases, y las almacene en el fichero "frases.txt". Acabará cuando el usuario pulse Intro sin teclear nada. Después deberá mostrar el contenido del fichero.**

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

int main()
{
  FILE* ficheroU;
  char frase[61];
  int i=0;

  ficheroU = fopen("frases.txt", "wt");
```

```
  printf(" PROGRAMA para ESCRIBIR y almacenar FRASES.\n"
    "Cuando quiera salir, simplemente pulse \"Intro\".\n\n");
  do
  {
    if (i == 0)
      puts("\nEscriba una FRASE:\n(o pulse \"Intro\"). \n");
    else
      puts("\nEscriba otra FRASE:\n(o pulse \"Intro\"). \n");
    gets(frase);
    fprintf(ficheroU, "%s\n", frase);
    i++;
  }
  while (strcmp(frase, "") != 0);
  printf("He aqui lo que escribio:\n\n");
  fclose(ficheroU);

  ficheroU = fopen("frases.txt", "rt");
  do
  {
    fgets(frase, 60, ficheroU);
    puts(frase);
  }
  while (!feof(ficheroU));
  getchar();
  printf("...Hasta luego!");
  getchar();
  fclose(ficheroU);

  return 0;
}
```

Ejercicio3: Un programa que pregunte un nombre de fichero y muestre en pantalla el contenido de ese fichero, haciendo una pausa después de cada 25 líneas, para que dé tiempo a leerlo. Cuando el usuario pulse intro, se mostrarán las siguientes 25 líneas, y así hasta que termine el fichero.

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

int main()
{
   FILE *fichero;
   char linea[100], nombre[40];
   int i=0;
  do
  {
    printf("\nNombre de fichero: ");
    gets(nombre);

    fichero = fopen(nombre, "rt");

    if (fichero == NULL)
    {
       printf("No existe el fichero\n\n");
       i++;
       if (i == 5)
          exit(1);
    }
  }
  while (fichero == NULL);
  while (!feof(fichero))
  {
     for (i=0; i<25; i++){
        fgets(linea, 100, fichero);
```

```
        if (!feof(fichero))
        {
            puts(linea);
        }
    }
    getchar();
    }
    fclose(fichero);
    return 0;
}
```

Ejercicio 4: Crear un struct que almacene los siguientes datos de una persona: nombre, edad, ciudad de residencia. Pedir al usuario esos datos de una persona y guardarlos en un fichero llamado "gente.dat". Cerrar el fichero, volverlo a abrir para lectura y mostrar los datos que se habían guardado.

```
#include <stdio.h>

int main()
{
  struct datos
  {
    char nombre[21];
    int edad;
    char ciudad[21];
  }persona;

  FILE *fichero;
  int i;

  fichero = fopen("gente.dat", "wt");
  if (fichero == NULL)
    printf("No se pudo abrir el archivo.\n");
```

```
else
{
  puts("Nombre:");
  gets(persona.nombre);
  fprintf(fichero, "%s ", persona.nombre);
  puts("Edad:");
  scanf("%d", &persona.edad);
  getchar();
  fprintf(fichero, "%d ", persona.edad);
  puts("Ciudad:");
  gets(persona.ciudad);
  fputs(persona.ciudad, fichero);
  fclose(fichero);
}
fichero = fopen("gente.dat", "rt");
if (fichero == NULL)
  printf("No se pudo abrir el archivo.\n");
else
{
   while (!feof(fichero))
  // while (feof(fichero) == 0)
   {
     fgets(persona.nombre, 60, fichero);
     puts(persona.nombre);
     fscanf(fichero, "%d", &persona.edad);
     getchar();
     printf("%d", persona.edad);
     fgets(persona.ciudad, 60, fichero);
     puts(persona.ciudad);
   }
fclose(fichero);
}
```

```
  return 0;
}
```

Ejercicio 5: Ampliar el programa anterior para que use un "array de structs", de forma que se puedan tener datos de 10 personas. Se deberá pedir al usuario los datos de las 10 personas y guardarlos en el fichero. Después se pedirá al usuario un número del 1 al 10 y se mostrarán los datos de la persona indicada por ese número, que se deberán leer de fichero (1 será la primera ficha, y 10 será la última). Por ejemplo, si el usuario indica que quiere ver los datos de la persona 3 (tercera), se deberá leer las dos primeras, ignorando su contenido, y después leer la tercera, que sí se deberá mostrar.

Ejercicio 6: Un programa que pida al usuario que teclee frases, y las almacene en el fichero "registro.txt", que puede existir anteriormente (y que no deberá borrarse, sino añadir al final de su contenido). Cada sesión acabará cuando el usuario pulse Intro sin teclear nada.

Ejercicio 7: Crear un programa que pida al usuario pares de números enteros y escriba su suma (con el formato "20 + 3 = 23") en pantalla y en un fichero llamado "sumas.txt", que se encontrará en un subdirectorio llamado "resultados". Cada vez que se ejecute el programa, deberá añadir los nuevos resultados a continuación de los resultados de las ejecuciones anteriores.

Ejercicio 8: Una agenda que maneje los siguientes datos: nombre, dirección, tlf móvil, email, y día, mes y año de nacimiento (estos tres últimos datos deberán ser números enteros cortos). Deberá tener capacidad para 100 fichas. Se deberá poder añadir un dato nuevo, visualizar los nombres de las fichas existentes, o mostrar todos los datos de una persona (se preguntará al usuario cual es el nombre de esa persona que quiere visualizar). Al empezar el programa, leerá los datos de un fichero llamado "agenda.dat" (si existe). Al terminar, guardará todos los datos en ese fichero.

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

int main()
{
  struct agenda
  {
    char nombre [21];
```

```
  char direccion [31];
  char celular[13];
  char email[31];
  unsigned short int dia, mes, anyo;
}persona[100];

FILE* fichero;
char textoAux[21];
int opcion, personas=0;
int i, j;

fichero = fopen("agenda.dat", "rt");
if (fichero != NULL)
{
    while (! feof(fichero))
    {
        fgets(textoAux, 20, fichero);
        if (feof(fichero)) break;
        if (strlen(textoAux) > 0)
            textoAux[strlen(textoAux) -1] = '\0';
        strcpy(persona[personas].nombre, textoAux);
        puts(persona[personas].nombre);

        fgets(textoAux, 20, fichero);
        if (feof(fichero)) break;
        if (strlen(textoAux) > 0)
            textoAux[strlen(textoAux) -1] = '\0';
        strcpy(persona[personas].direccion, textoAux);
        puts(persona[personas].direccion);

        fgets(textoAux, 20, fichero);
        if (feof(fichero)) break;
```

```
        if (strlen(textoAux) > 0)
            textoAux[strlen(textoAux) -1] = '\0';
        strcpy(persona[personas].celular, textoAux);
        puts(persona[personas].celular);

        fgets(textoAux, 20, fichero);
        if (feof(fichero)) break;
        if (strlen(textoAux) > 0)
            textoAux[strlen(textoAux) -1]  = '\0';
        strcpy(persona[personas].email, textoAux);
        puts(persona[personas].email);

        fgets(textoAux, 20, fichero);
        if (feof(fichero)) break;
        sscanf(textoAux, "%hd %hd %hd", &persona[personas].dia,
            &persona[personas].mes,
            &persona[personas].anyo);
        if ((persona[personas].dia == 0) || (persona[personas].mes == 0) ||
            (persona[personas].anyo == 0))
          break;
        puts(textoAux);
        personas++;
    }
    fclose(fichero);
}
if (personas == 0) printf("No hay ninguna ficha guardada.");
else
   if (personas == 1) printf("He ahi la unica ficha guardada.");
   else
      printf("He ahi las %d fichas guardadas.", personas);
do
{
```

```
puts("\n\n  MENU:\n");
puts("1.- Agregar un nuevo dato.");
puts("2.- Ver todos los datos existentes.");
puts("3.- Ver todos los datos de una persona.");
puts("0.- Terminar.");
puts("   Elija una opcion: ");

scanf("%d", &opcion);
getchar();

switch (opcion)
{
  case 1:    /* Agregar un nuevo dato */
     printf ("nombre: ");
     gets (persona[personas].nombre);
     fprintf(fichero, "%s", persona[personas+1].nombre);
     printf ("Direccion: ");
     gets (persona[personas].direccion);
     fprintf(fichero, "%s", persona[personas+1].direccion);
     printf ("Celular: ");
     gets (persona[personas].celular);
     fprintf(fichero, "%s", persona[personas+1].celular);
     printf ("Correo electronico: ");
     gets (persona[personas].email);
     fprintf(fichero, "%s", persona[personas+1].email);
     printf ("Dia de nacimiento: ");
     scanf("%hd", &persona[personas].dia);
     fprintf(fichero, "%hd", persona[personas+1].dia);
     printf ("Mes de nacimiento: ");
     scanf ("%hd", &persona[personas].mes);
     fprintf(fichero, "%hd", persona[personas+1].mes);
     printf ("Año de nacimiento: ");
```

```
   scanf ("%hd", &persona[personas].anyo);
   fprintf(fichero, "%hd", persona[personas].anyo);
   personas ++;
   break;

case 2:    /* Ver todos los nombres */
   j = 0;
   puts ("Ver todos los nombres existentes:");
   for (i=0; i<=personas; i++)
   {
     puts (persona[i].nombre);
     j ++;
   }
   if (j == 0)
     printf("No hay ningun nombre guardado.");
   break;

case 3:    /* Ver todos los datos de una persona */
   puts("Nombre a buscar:");
   gets(textoAux);
   j = 0;
   for (i=0; i<=personas; i++)
   {
     if (strcmp (textoAux, persona[i].nombre) == 0)
     {
       puts (persona[i].nombre);
       if (strcmp(persona[i].nombre, "") == 0)
         break;
       puts (persona[i].direccion);
       puts (persona[i].celular);
       puts (persona[i].email);
       printf("%hd %hd %hd.\n", persona[i].dia,
```

```
                persona[i].mes, persona[i].anyo);
              j ++;
            }
          }
          if (j == 0)
            printf("No existe esa ficha.");
          break;
        }
      }
      while (opcion != 0);

      fichero = fopen("agenda.dat", "wt");
      for (i=0; i<=personas; i++)
      {
        fprintf (fichero, "%s\n", persona[i].nombre);
        fprintf (fichero, "%s\n", persona[i].direccion);
        fprintf (fichero, "%s\n", persona[i].celular);
        fprintf (fichero, "%s\n", persona[i].email);
        fprintf (fichero, "%hd %hd %hd\n",
          persona[i].dia, persona[i].mes, persona[i].anyo);
      }
      fclose(fichero);
      return 0;

}
```