PRÁCTICA
MANEJO DE ARCHIVOS

Objetivos

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

Al final de esta práctica el alumno podrá:

1. *Utilizar las herramientas para el almacenamiento y recuperación de datos contenidos en archivos.*

Antecedentes

1. Manejar ciclos de repetición en lenguaje C.
2. Manipular sentencias de control de flujo en lenguaje C.
3. Haber aplicado del concepto de apuntador para el manejo de datos en memoria principal

Introducción

Archivos

Un archivo o fichero, es un conjunto de datos que se encuentran almacenados en memoria secundaria y se distingue a través de un nombre. A diferencia de los arreglos y de las estructuras de datos *struct* de C, los archivos mantienen sus datos sin importar si el programa que accede a ellos está en ejecución o ha terminado. El nombre de un archivo consta de un nombre en sí y una extensión, donde la extensión indica el tipo de archivo del que se trata, pudiendo ser de texto plano (txt) o de imágenes (jpg, bmp, etc) o de cualquier otro tipo de datos.

Acceso a archivos

En el lenguaje C, para manipular un archivo se requiere una variable apuntador, la cual irá conteniendo la dirección de almacenamiento del dato del archivo al que se accederá. Dicha variable se declara de tipo apuntador a *FILE* (archivo). Un ejemplo de la definición de un apuntador a un archivo es:

```
FILE *apArch;
```

Una vez definido dicho apuntador, hay que "abrir" el archivo para acceder a los datos que contenga o que se grabarán en él; esto se hace con la función *fopen()*, la cual recibe como parámetros el nombre del archivo y el modo en que va a ser empleado.

El modo, indica la acción que se va a realizar sobre el archivo. A continuación se presenta una tabla con los diferentes modos de acceso al archivo que se "abrirá".

Modo	Acción
"r"	Abre un archivo para lectura
"w"	Crea un archivo para escritura; si existe, borra y escribe nueva información en el archivo.

"a"	Abre o crea un archivo y añade datos al final del mismo
"rb"	Abre un archivo en modo binario para lectura
"wb"	Crea un archivo binario para escritura; si existe, borra y escribe nueva información en el archivo.
"ab"	Abre o crea un archivo binario y añade datos al final del mismo

El valor que entrega la función *fopen()* es la dirección de la primera localidad de memoria del archivo. Así, una sentencia de apertura de un archivo generalmente tiene el siguiente aspecto:

apArch = fopen("datos.txt", "w");

donde *apArch* ha sido declarado como un apuntador a archivo.

Ejemplos de manejo de archivos

Lectura de un archivo

Ejemplo 1

Programa que lee el contenido de un archivo llamado *text.txt* y lo despliegua en pantalla. (Nota: para la correcta ejecución de este programa, debe existir un archivo de texto simple con el nombre "texto.txt" en donde se vaya a ejecutar este programa).

-Explicación de variables

f: Apuntador de tipo *FILE*, auxiliar en la lectura de un archivo.
c: Variable de tipo char, auxiliar en el almacenamiento y despliegue de los caracteres del archivo en pantalla.

-Código

```
#include<stdio.h>

void main(){
   FILE *f;
   char c;
   f=fopen("texto.txt","r");
   c=fgetc(f);
   while(c!=EOF){
       printf("%c",c);
       c=fgetc(f);
   }
   fclose(f);
}
```

Descripción del ejemplo 1

Con la instrucción `FILE *f` se define la variable de tipo apuntador a archivo *f*. Mediante este apuntador se manipulará al archivo que el programa leerá.

Luego se abre el archivo con *fopen("texto.txt","r")* asignando el valor que devuelve la función a nuestro apuntador *f*, o sea, la dirección de la primera localidad de memoria del archivo.

Con la función *fgetc(f)*, se obtiene el carácter que está apuntando *f*; al término de esta operación, *f* actualiza su valor para apuntar al siguiente carácter. Con la sentencia

```
c=fgetc(f);
```

la variable tipo carácter *c* recibe el carácter apuntado por *f*. Esta sentencia se encuentra en un ciclo *while* para leer carácter por carácter del archivo e irlo imprimiendo.

Cabe mencionar que al final de un archivo siempre hay un carácter especial para indicar que ya no hay más datos contenidos en el archivo. Este carácter especial, en el lenguaje C, está definido con la constante simbólica EOF (End Of File); es por ello que en el ciclo *while* del programa tiene como condición que mientras `c!=EOF` siga imprimiendo y leyendo un carácter.

Una vez que se ha terminado de utilizar el archivo es conveniente "cerrarlo"; por esto, el programa contiene la sentencia

```
fclose(f);
```

Escritura en un archivo

Ejemplo 2

Programa que crea un archivo y escribe datos en él. Los datos que se almacenan en el archivo creado se leen del teclado y se toman como caracteres. Cuando ya no se ingresen más datos desde el teclado se deben dar dos saltos de línea juntos.

-Explicación de variables

f: Apuntador de tipo *FILE*, auxiliar en la escritura de un archivo
c: Variable de tipo *char*, auxiliar en el almacenamiento e impresión de los caracteres dentro del archivo.
salir: Variable de tipo entero, auxiliar para finalizar el programa

-Código

```
#include<stdio.h>

void main(){
      FILE *f;
      char c;
      int salir=0;
      f=fopen("escritura.txt","w");
      c=getchar();
      while(salir<1){
            if(c=='\n'){
                  salir++;
            }
            else{
                  salir=0;
            }
            fputc(c,f);
            c=getchar();
      }
      fclose(f);
}
```

Descripción del ejemplo 2

Al igual que con el programa anterior, se debe declarar un apuntador de tipo FILE, y abrir el archivo en modo escritura.

```
FILE *f;
f=fopen("escritura.txt","w");
```

Con la función getchar() se obtiene el carácter tecleado y se asigna a la variable *c*

```
c=getchar();
```

El ciclo *while* se emplea para ir leyendo carácter por carácter desde el teclado e irlo almacenándolo en el archivo utilizando la función *fputc()*, la cual recibe como parámetros, el carácter a escribir, y el apuntador al archivo donde se va a escribir. En el programa la sentencia que realiza esta escritura es

```
fputc(c,f);
```

El ciclo while se detiene cuando el pograma lea dos saltos de línea juntos.

Por último, es conveniente cerrar el archivo cuando se termina de escribir datos, para asegurar el correcto almacenamiento de ellos en el archivo. Para realizar esta operación, en el programa se encuentra la sentencia

```
fclose(f);
```

Otras funciones para el manejo de archivos.

Existen otras funciones en el lenguaje C que permiten manejar archivos, algunas de ellas se muestran en la siguiente tabla:

Función	Descripción	Ejemplo

feof()	Comprueba el indicador de final de archivo. Regresa un 1 si encuentra el EOF.	while(!feof(f)){
		}
		Verifica que no haya llegado al final del archivo.

fgets()	Obtiene una secuencia de caracteres de numero_MAX-1.	fgets(variable_donde_guardar, numero_MAX, apuntador_a_archivo);
		
		fgets(cadena,20,f);

fprintf()	Realiza la misma función que *printf()*, pero escribe en archivo en lugar de en pantalla.	fprintf(f,"El número escogido es: %d",num);
fscanf()	Realiza la misma función que *scanf()*, pero lee de un archivo.	fscanf(f,"%d %d",&var1,&var2);

Manejo de archivos de datos numéricos.

Es común para un estudiante de ingeniería almacenar datos numéricos en archivos. La manipulación de archivos que contengan datos numéricos no difiere mucho de uno de tipo texto.

Existen varias formas de almacenar números en un archivo. Por ejemplo, si se desea almacenar los elementos de un arreglo de 5 enteros en un archivo, el siguiente código podría ser parte de un programa que realizaría esta operación.

```
FILE *apArch;
int arr[]={10,20,14,9,5};
/*Apertura del archivo para escritura */
apArch=fopen("numeros.txt","w");
for (int i=0;i<5;i++) {
     fprintf(apArch,"%d ",arr[i]);
}
fclose(apArch);
```

La sentencia

```
fprintf(apArch,"%d ",arr[i]);
```

escribe en el archivo apuntado por *apArch* el valor del *i*-ésimo elemento del arreglo *arr* y deja un espacio. Como esta sentencia está en un ciclo *for*, los cinco números escritos en el archivo estarán situados en un mismo renglón separados por un espacio.

Para leer 5 números enteros de un archivo que están separados por un espacio, y colocarlos en un arreglo, puede emplearse el siguiente código como parte de un programa:

```
FILE *apArch;
int arr[5];
/*Apertura del archivo para lectura */
apArch=fopen("numeros.txt","r");
for (int i=0;i<5;i++) {
     fscanf(apArch,"%d ",&arr[i]);
}
fclose(apArch);
```

En caso de que los números a almacenar o a leer de un archivo sean reales o flotantes, sólo se cambia el formato por el adecuado (%f o %g).

En ocasiones los datos numéricos están almacenados en el archivo en modo texto, por lo que hay que hacer la conversión de cadena a número o de número a cadena; para ello se sugiere revisar las siguientes funciones de C, que realizan las conversiones antes descritas

atoi()
atof()
itoa()
ftoa()

EJERCICIOS PROPUESTOS

1. Elaborar un programa que lea texto de un archivo y que lo escriba tal cual en otro archivo.
2. Construir un programa que reciba un carácter, lea un archivo e imprima en pantalla el número de incidencias del carácter dentro del archivo.
3. Hacer un programa que lea de un archivo e imprima el texto incluido pero en orden inverso.
4. Elaborar un programa que lea de un archivo una matriz cuadrada entera de orden 5 (ver NOTA) y que la despliegue en pantalla.
5. Hacer un programa que lea de un archivo una matriz cuadrada entera de orden 5 (ver NOTA) y que despliegue en pantalla sólo los elementos de la diagonal principal.
6. Modificar el programa del punto 4 para que guarde en otro archivo la matriz por renglones, cada renglón seguido de su suma, y después del último renglón, que agregue otro renglón con la suma de las columnas.
7. Elaborar un programa que lea, de dos archivos diferentes, dos matrices enteras de 3x3 y que guarde en otro archivo la suma de las matrices.
8. Modificar el programa anterior, para que realice la multiplicación y que despliegue en pantalla la matriz resultante.
9. Modificar el programa del inciso 8, para que obtenga el determinante de ambas matrices y que imprima en otro archivo únicamente la matriz con el valor más alto del determinante.

NOTA: La separación entre valores de un renglón, deberá ser únicamente un espacio o tabulador, y la separación entre renglones, por un solo salto de línea.

El profesor podr§ dise¶ar sus propios ejercicios, siempre y cuando cubra por completo el objetivo de la pr§ctica.