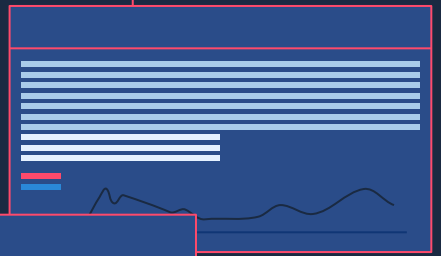
Tecnicatura en Programación Universitaria

Comisión

B

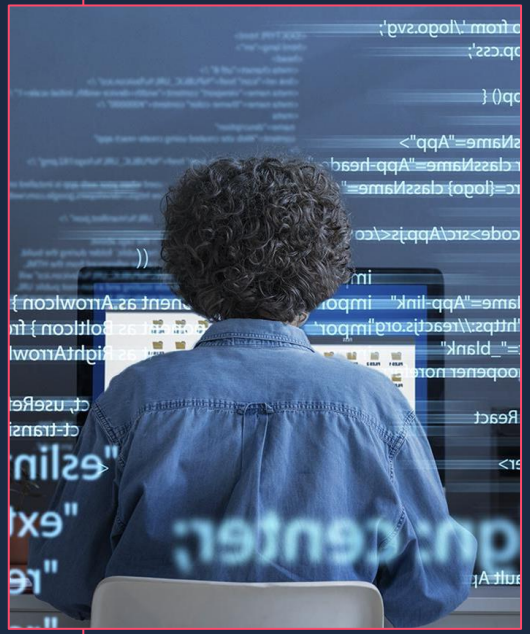
Clase 10



Clase Practica

Martes 11 de Junio 2024

Ing. Oviedo Codigoni Carlos Nicolas



/*

… Manejo de Archivos …

*/

Funciones para el Manejo de Archivos

NOMBRE	FUNCION
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 True si se llega al final del archivo
ferror()	Devuelve True si se produce un error
rewind()	Coloca un localizador de posición del archivo al principio de este
remove()	Borra un archivo
fflush()	Vacía un archivo

fopen()

```
FILE* fichero; fichero = fopen("ejemplo.txt", "w");
```

- 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

EOF (End-Of-File)

EOF (End-Of-File), en el contexto de la programación, se refiere a un marcador o indicador especial que se utiliza para señalar el final de un archivo. Es decir, cuando se alcanza el EOF, significa que no hay más datos disponibles para leer en el archivo

¿En qué se utiliza EOF?

- Lectura de archivos: La función feof() se utiliza para verificar si se ha alcanzado el final de un archivo. Esto es útil para controlar bucles de lectura de archivos, ya que permite detener la lectura cuando no hay más datos disponibles.
- Escritura de archivos: Aunque no se utiliza directamente un marcador EOF para la escritura de archivos, algunas funciones de escritura, como fput(), pueden devolver un valor especial para indicar que se ha producido un error o que se ha alcanzado el final del archivo.

Ejemplo fopen() & fprintf()

```c
#include <stdio.h>

int main() {
  // Se declara una variable de tipo FILE llamada "archivo" para almacenar el puntero al archivo
  FILE *archivo;
  // Se abre el archivo "ejemplo.txt" en modo escritura
  archivo = fopen("ejemplo.txt", "w");
  // Se comprueba si la apertura del archivo fue exitosa
  if (archivo == NULL) {
    // Se imprime un mensaje de error si no se pudo abrir el archivo
    printf("Error al abrir el archivo");
    // Se termina el programa con un código de error 1
    return 1;
  }
// Se escribe la cadena "Esta es una línea de texto" en el archivo
  fprintf(archivo, "Esta es una línea de texto\n");
  // Se escribe la cadena "Otra línea de texto" en el archivo
  fprintf(archivo, "Otra línea de texto\n");
// Se cierra el archivo
  fclose(archivo);
// Se indica que el programa se ha ejecutado correctamente
  return 0;
}
```

Ejemplo fopen() & fwrite()

```c
#include <stdio.h>

int main() {
  // Se declaran tres variables: dos de tipo int y una de tipo float
  int num1 = 10;
  int num2 = 20;
  float num3 = 3.1415f;

  // Se declara una variable de tipo FILE para almacenar el puntero al archivo
  FILE *archivo;

  // Se abre el archivo "datos.txt" en modo escritura binaria ("wb")
  archivo = fopen("datos.txt", "wb");

  // Se comprueba si la apertura del archivo fue exitosa
  if (archivo == NULL) {
    printf("Error al abrir el archivo"); // Se imprime un mensaje de error si no se pudo abrir el archivo
    return 1; // Se termina el programa con un código de error 1
 }

  // Se escriben las variables num1 y num2 en el archivo como enteros
  fwrite(&num1, sizeof(int), 1, archivo);
  fwrite(&num2, sizeof(int), 1, archivo);

  // Se escribe la variable num3 en el archivo como un float
  fwrite(&num3, sizeof(float), 1, archivo);

  // Se cierra el archivo
  fclose(archivo);

  // Se imprime un mensaje en pantalla para indicar que las variables se han guardado correctamente en el archivo
  printf("Las variables se han guardado correctamente en el archivo\n");

  return 0; // Se indica que el programa se ha ejecutado correctamente
}
```

Ejemplo fopen() & fread()

```c
#include <stdio.h>
int main() {
  // Se declaran las variables para almacenar los datos leídos del archivo
  int num1;
  int num2;
  float num3;

  // Se declara una variable de tipo FILE para almacenar el puntero al archivo
  FILE *archivo;

  // Se abre el archivo "datos.txt" en modo lectura binaria ("rb")
  archivo = fopen("datos.txt", "rb");

  // Se comprueba si la apertura del archivo fue exitosa
  if (archivo == NULL) {
    printf("Error al abrir el archivo"); // Se imprime un mensaje de error si no se pudo abrir el archivo
    return 1; // Se termina el programa con un código de error 1
  }

  // Se lee la variable num1 del archivo como un entero
  fread(&num1, sizeof(int), 1, archivo);

  // Se lee la variable num2 del archivo como un entero
  fread(&num2, sizeof(int), 1, archivo);

  // Se lee la variable num3 del archivo como un float
  fread(&num3, sizeof(float), 1, archivo);

  // Se cierra el archivo
  fclose(archivo);

  // Se imprimen las variables leídas en pantalla
  printf("Las variables leídas del archivo son:\n");
  printf("num1: %d\n", num1);
  printf("num2: %d\n", num2);
  printf("num3: %f\n", num3);

  return 0; // Se indica que el programa se ha ejecutado correctamente
}
```

include<exercise.h>

Ejercicio 1:

- **Objetivo**: Crear un archivo de texto llamado "archivo_texto.txt" y un archivo binario llamado "archivo_binario.dat".
- **Descripción**: El programa deberá crear dos archivos: uno de texto y otro binario. El archivo de texto puede estar vacío o contener un mensaje simple. El archivo binario no deberá contener datos específicos, solo se debe crear el archivo vacío.
- **Requisitos**:
  - Utilizar las funciones fopen() y fclose() para la manipulación de archivos.
  - Verificar si la apertura de los archivos fue exitosa.
  - Cerrar los archivos una vez finalizada la operación.

Ejercicio 2:

- **Objetivo**: Crear un archivo llamado "datos.dat" y almacenar en él tres variables: dos de tipo int y una de tipo float.
- **Descripción**: El programa deberá crear un archivo binario llamado "datos.dat" y escribir en él los valores de tres variables: dos enteras y una flotante. Se deben utilizar las funciones adecuadas para la escritura de datos binarios en el archivo.
- **Requisitos**:
  - Incluir la biblioteca stdio.h para la entrada y salida de datos.
  - Declarar tres variables: dos de tipo int y una de tipo float.
  - Asignar valores a las variables.
  - Utilizar la función fopen() para abrir el archivo "datos.dat" en modo binario de escritura ("wb").
  - Utilizar la función fwrite() para escribir los valores de las variables en el archivo. El primer argumento de fwrite() debe ser un puntero a las variables, el segundo argumento debe ser el tamaño en bytes de cada elemento a escribir, el tercer argumento debe ser la cantidad de elementos a escribir y el cuarto argumento debe ser un puntero al modo de apertura del archivo.
  - Verificar si la escritura en el archivo fue exitosa.
  - Utilizar la función fclose() para cerrar el archivo.

include<exercise.h>

Ejercicio 3:

- **Objetivo**: Leer un archivo llamado "datos.dat" y mostrar en pantalla los valores de tres variables: dos de tipo int y una de tipo float.
- **Descripción**: El programa deberá leer un archivo binario llamado "datos.dat" y obtener los valores de tres variables: dos enteras y una flotante. Se deben utilizar las funciones adecuadas para la lectura de datos binarios del archivo.
- **Requisitos**: Incluir la biblioteca stdio.h para la entrada y salida de datos.

  - Declarar tres variables: dos de tipo int y una de tipo float.Utilizar la función fopen() para abrir el archivo "datos.dat" en modo binario de lectura ("rb").
  - Utilizar la función fread() para leer los valores del archivo en las variables.
  - El primer argumento de fread() debe ser un puntero a las variables, el segundo argumento debe ser el tamaño en bytes de cada elemento a leer, el tercer argumento debe ser la cantidad de elementos a leer y el cuarto argumento debe ser un puntero al modo de apertura del archivo.
  - Verificar si la lectura del archivo fue exitosa.
  - Mostrar en pantalla los valores de las variables leídas.
  - Utilizar la función fclose() para cerrar el archivo.

include<exercise.h>

Ejercicio 4:

- **Objetivo**: Crear un archivo llamado "mensaje.txt" y almacenar en él una cadena de caracteres.
- **Descripción**: El programa deberá crear un archivo de texto llamado "mensaje.txt" y escribir en él una cadena de caracteres. Se debe utilizar la función fprintf() para escribir la cadena en el archivo.
- **Requisitos**:
  - Incluir la biblioteca stdio.h para la entrada y salida de datos.
  - Declarar una variable de tipo char para almacenar la cadena de caracteres.
  - Asignar una cadena de caracteres a la variable.Utilizar la función fopen() para abrir el archivo "mensaje.txt" en modo escritura ("w").
  - Utilizar la función fprintf() para escribir la cadena de caracteres en el archivo.
  - El primer argumento de fprintf() debe ser un puntero al archivo, el segundo argumento debe ser la cadena de caracteres a escribir y el tercer argumento debe ser un puntero a la cadena de formato (en este caso, se puede omitir).
  - Verificar si la escritura en el archivo fue exitosa.
  - Utilizar la función fclose() para cerrar el archivo.

Ejercicio 5:

Leer un archivo llamado "mensaje.txt" y mostrar en pantalla. El archivo deberá tener un texto, el cual se deberá modificar todas las letras “a” por “e”.
Primero deberan cargar el archivo con el texto y luego leerlo y realizar lo que pide el enunciado.