# Archivos

Algorítmica y programación I

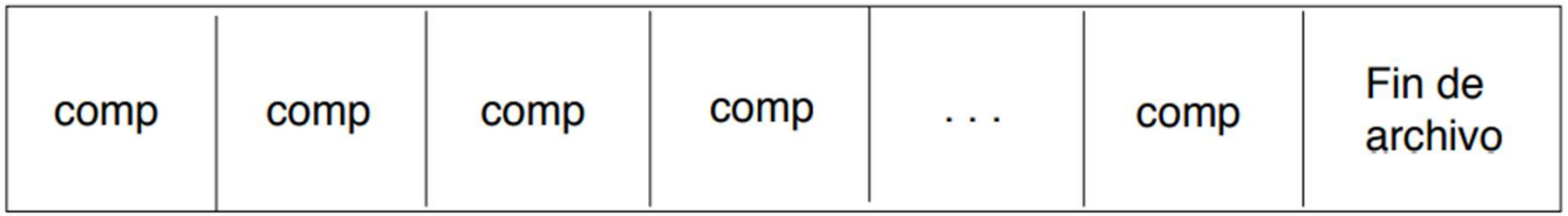
# Archivos

Que es un archivo?

Es un conjunto de datos que se almacenan en un dispositivo de almacenamiento, por ejemplo, una unidad de disco.

Archivos en C:

- En C, los archivos se manejan a través de punteros de tipo FILE
- Es una estructura, con una secuencia datos del mismo tipo, de tamaño no fijado de antemano.
- Se puede representar como una fila de celdas en las que se almacenan los datos.
- Contaremos con una marca especial que señala el fin del archivo.



# Archivos

Porque crear archivos?

Para guardar datos que posteriormente podrán ser leídos por el mismo programa o por otro.

*Supongamos un programa que pide información de los alumnos para generar una base de datos de los mismos, la idea sería poder almacenar dicha información para luego volver a consultarla o actualizarla.*

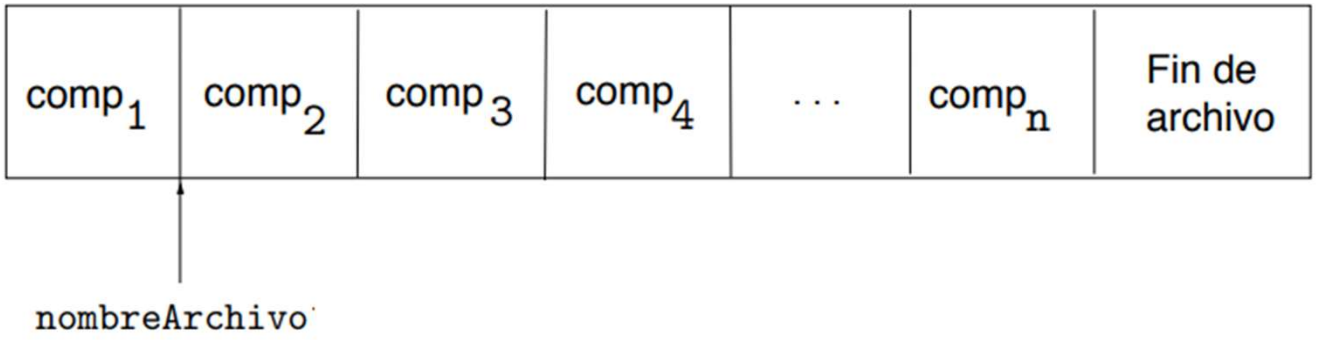
```
typedef struct {
    char nombre[50];
    int edad;
    float nota;
} Estudiante;

int main() {
    FILE *filePointer;
    Estudiante estudiante1 = {"Juan", 20, 85.5}, estudiante2;

    // Abriendo/creando un archivo en modo escritura binaria
    filePointer = fopen("archivoEjemplo.bin", "wb");
    if (filePointer == NULL) {
        printf("No se pudo abrir o crear el archivo.\n");
        return 1;
    }
    fwrite(&estudiante1, sizeof(Estudiante), 1, filePointer);
    fclose(filePointer);

    // Abriendo el archivo en modo lectura binaria
    filePointer = fopen("archivoEjemplo.bin", "rb");
    if (filePointer == NULL) {
        printf("No se pudo abrir el archivo.\n");
        return 1;
    }
    fread(&estudiante2, sizeof(Estudiante), 1, filePointer);
    printf("Datos leídos:\nNombre: %s\nEdad: %d\nNota: %.2f\n",
           estudiante2.nombre, estudiante2.edad, estudiante2.nota);
    fclose(filePointer);
    return 0;
}
```

# Archivos - Eof

**Cursor del archivo**: al leer de un archivo leeremos una posición señalada por el cursor del archivo.

Al momento de abrir el archivo este cursor apuntará a la posición inicial del mismo. A medida que vayamos leyendo el cursor ira avanzando hasta llegar al final.

Para saber si llegamos al final del archivos contamos con la función **feof()**. Esta función nos devolverá verdadero si hemos llegado al final del archivo.



# Archivos – operaciones

**Declaración:**

```
FILE *filePointer;
```

**Apertura del archivo**

```
filePointer = fopen("archivo.bin", "wb");  // Escribir o crear archivo binario
filePointer = fopen("archivo.bin", "rb");  // Leer archivo binario
```

**Cerrar un archivo**

```
fclose(filePointer);
```

**Lectura y escritura**

```
// Escribir
fwrite(&estructura, sizeof(TipoEstructura), 1, filePointer);
// Leer
fread(&estructura, sizeof(TipoEstructura), 1, filePointer);
```

**Posicionamiento**:

Puedes mover el cursor del archivo a una posición específica o obtener su posición actual.

```
// Moverse a una posición específica en bytes
fseek(filePointer, desplazamientoEnBytes, origen);
// Obtener posición actual en bytes
long posicion = ftell(filePointer);
```

# Posicionamiento en archivos

**Cursor del archivo**: El cursor del archivo nos indica la posición actual desde donde leeremos o escribiremos el archivo.

- Al abrir un archivo para lectura con "r" o lectura/escritura con "r+", el cursor se posiciona al inicio del archivo.
- Al abrir un archivo para escritura con "w", "w+" o agregar con "a", el cursor se posiciona al final del archivo (a menos que escribamos algo, en cuyo caso con "w" se sobrescribirá el archivo).

**Posicionamiento**: Para mover el cursor a una posición específica en el archivo, utilizamos la función fseek.

```
fseek(filePointer, desplazamientoEnBytes, origen);
```

- filePointer: es el puntero al archivo.
- desplazamientoEnBytes: es el número de bytes a desplazarse desde el origen.
- origen: puede ser SEEK_SET (comienzo del archivo), SEEK_CUR (posición actual del cursor) o SEEK_END (final del archivo).

# Posicionamiento en archivos

**Posición lógica**: La posición lógica en el archivo se refiere a la ubicación actual del cursor en términos de bytes desde el comienzo del archivo. En C, podemos obtener la posición actual usando ftell.

```
long posicionActual = ftell(filePointer);
```

# Archivos – leer hasta el final…

```
typedef struct {
    char nombre[50];
    int edad;
    float nota;
} Estudiante;
int main() {
    FILE *filePointer;
    Estudiante estudiante;
    // Abre el archivo en modo lectura binaria
    filePointer = fopen("archivoEjemplo.bin", "rb");
    if (filePointer == NULL) {
        printf("No se pudo abrir el archivo.\n");
        return 1;
    }
    // Lee el archivo hasta que feof() devuelva verdadero
    while (!feof(filePointer)) {
        size_t itemsRead = fread(&estudiante, sizeof(Estudiante), 1, filePointer);
        if (itemsRead == 1) {  // Verifica que se leyó un registro completo
            printf("Nombre: %s, Edad: %d, ", estudiante.nombre, estudiante.edad);
        }
    }
    fclose(filePointer);
    return 0;
}
```

# Archivos – Modificación de un registro

Para modificar un registro debemos

- Posicionarnos en el registro que queremos modificar
  - Recorriendo hasta encontrar el registro que queremos modificar o
  - Usando la función **fseek(filePointer, desplazamiento, origen)**.
- Usar la función fwrite para escribir sobre el mismo.

# Archivos – borrado de registro

**Borrado lógico**: En este caso, el registro no se borra del archivo, pero usaremos alguna marca que nos indicará que el registro esta libre.

**Borrado físico**: eliminaremos el registro, para ello la mejor técnica es crear un archivo temporal, al que copiemos los registros que siguen vigentes. Luego se elimina el archivo original y se renombra el archivo temporal.

**Cambiar nombre a un archivo**

```
rename("nombreActual.txt", "nuevoNombre.txt");
```

**Borrar un archivo:**

```
remove("nombreArchivo.txt");
```

# Archivos de texto

A diferencia de los archivos de registro o binarios, estos solo guardan texto y pueden ser explorados por cualquier editor de texto.

```
int main() {
    FILE *filePointer;
    char dataToBeWritten[100] = "¡Bienvenido a la programación en C!";
    char dataToBeRead[100];

    // Abriendo/creando un archivo en modo escritura de texto
    filePointer = fopen("archivoTextoEjemplo.txt", "w");
    if (filePointer == NULL) {
        printf("No se pudo abrir o crear el archivo.\n");
        return 1;
    }
    fputs(dataToBeWritten, filePointer);
    fclose(filePointer);

    // Abriendo el archivo en modo lectura de texto
    filePointer = fopen("archivoTextoEjemplo.txt", "r");
    if (filePointer == NULL) {
        printf("No se pudo abrir el archivo.\n");
        return 1;
    }
    while (fgets(dataToBeRead, sizeof(dataToBeRead), filePointer) != NULL) {
        printf("%s", dataToBeRead);
    }
    fclose(filePointer);

    return 0;
}
```

# Ejercicios

**Ejercicio 1: Registro de Productos** Crea un programa que permita al usuario registrar productos en un archivo binario. Cada producto debe tener un ID, nombre y precio.

El programa debe permitir:

- Añadir un nuevo producto al archivo.
- Listar todos los productos en el archivo.
- Buscar un producto por su ID.

**Ejercicio 2: Actualización de Estudiantes** Dado un archivo binario que contiene registros de estudiantes (nombre, matrícula y calificación promedio), escribe un programa que:

- Permita al usuario buscar a un estudiante por su matrícula.
- Si el estudiante es encontrado, el programa debe permitir al usuario actualizar la calificación promedio del estudiante.
- Si el estudiante no es encontrado, se debe mostrar un mensaje adecuado.