Programación 1

Tema 15

Ficheros binarios

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

Información sobre protección de datos de carácter personal en el tratamiento de gestión de grabaciones de docencia

Sesión con grabación

Tratamiento: Gestión de grabaciones de docencia

Finalidad: Grabación y tratamiento audiovisual de docencia y su evaluación

Base Jurídica: Art. 6.1.b), c) y d) Reglamento General de Protección de Datos

Responsable: Universidad de Zaragoza.

Ejercicio de Derechos de acceso, rectificación, supresión, portabilidad, limitación u oposición al tratamiento ante el gerente de la Universidad conforme a https://protecciondatos.unizar.es/procedimiento-seguir

Información completa en:

https://protecciondatos.unizar.es/sites/protecciondatos.unizar.es/files/users/lopd/gdocencia_extensa.pdf

Propiedad intelectual: Queda prohibida la difusión, distribución o divulgación de la grabación y particularmente su compartición en redes sociales o servicios dedicados a compartir apuntes. La infracción de esta prohibición puede generar responsabilidad disciplinaria, administrativa y de índole civil o penal.

Fuente de las imágenes: https://pixabay.com/es

Información sobre protección de datos de carácter personal en el tratamiento de gestión de grabaciones de docencia

- Se recuerda que la grabación de las clases por medios distintos a los usados por el profesor o por personas diferentes al profesor sin su autorización expresa no está permitida, al igual que la difusión de esas imágenes o audios.

Índice

- Ficheros binarios
  - Diferencia con ficheros de texto
- Herramientas de C++ para trabajar con ficheros binarios
- Problemas básicos con ficheros binarios
  - Creación
  - Lectura

Ficheros binarios

- Almacenan una secuencia de datos codificados en binario.
  - Cada dato se almacena como un grupo consecutivo de *bytes*.
  - Para cada dato, se utiliza una codificación binaria idéntica a la que se utiliza en la memoria del computador.

Ficheros binarios

- Ejemplo:
  - Un dato de tipo **`int`** se almacena en un fichero binario como 4 *bytes* consecutivos en los que el entero está codificado en binario en complemento a 2.

Ficheros binarios

- Ventajas
  - Reducción del tamaño de los ficheros
  - Se facilitan las operaciones de lectura y escritura
    - Simplificación de las instrucciones que es necesario programar
    - Reducción del tiempo de lectura o escritura
- Desventajas
  - No legibles por seres humanos
  - Pueden aparecer problemas de portabilidad

Diferencias entre un fichero binario y un fichero de texto

- Ejemplo: 26173, dato de tipo **`int`**
  - Codificación en un **fichero de texto**:
    - 00110010 00110110 00110001 00110111 00110011
    - (= Secuencia de *bytes* 50, 54, 49, 55 y 51)
    - (= Secuencia de caracteres de códigos 50, 54, 49, 55 y 51)
    - (= Secuencia de caracteres '2', '6', '1', '7' y '3')
  - Codificación en un **fichero binario**:
    - 00111101 01100110 00000000 00000000
    - (= Secuencia de *bytes* 61, 102, 0 y 0)
    - (= 4 *bytes* que codifican el número 26173 en base 2 en complemento a 2, con el byte menos significativo en primer lugar)
    - (= $61 \times 256^0 + 102 \times 256^1 + 0 \times 256^2 + 0 \times 256^3$)

Diferencias entre un fichero binario y un fichero de texto

	Fichero de texto	**Fichero binario**
Interpretación de la secuencia de *bytes*	Caracteres	Codificación interna binaria de datos
¿Estructurado en líneas?	Sí	No
Necesidad de separadores entre datos	Habitualmente, sí	Habitualmente, no
Legible por una persona	Sí	No

Herramientas para trabajar con ficheros binarios en C++

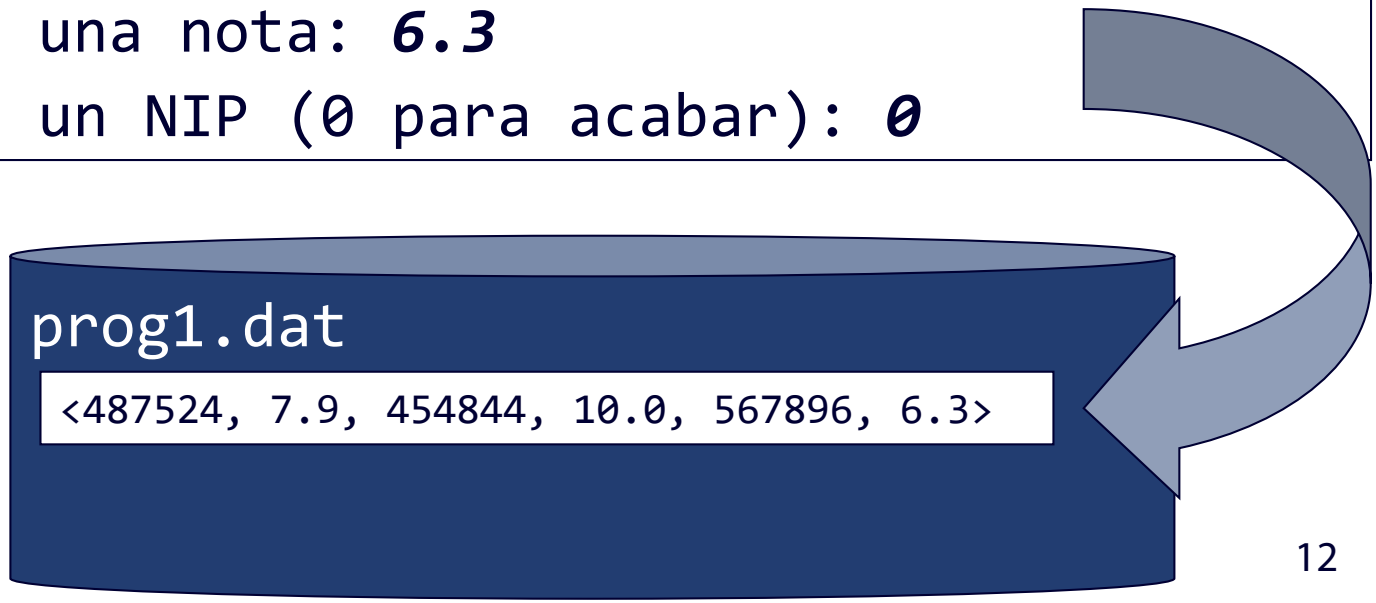
- Flujos de las clases **`ifstream`** y **`ofstream`** utilizados de una forma específica

Trabajo con ficheros binarios

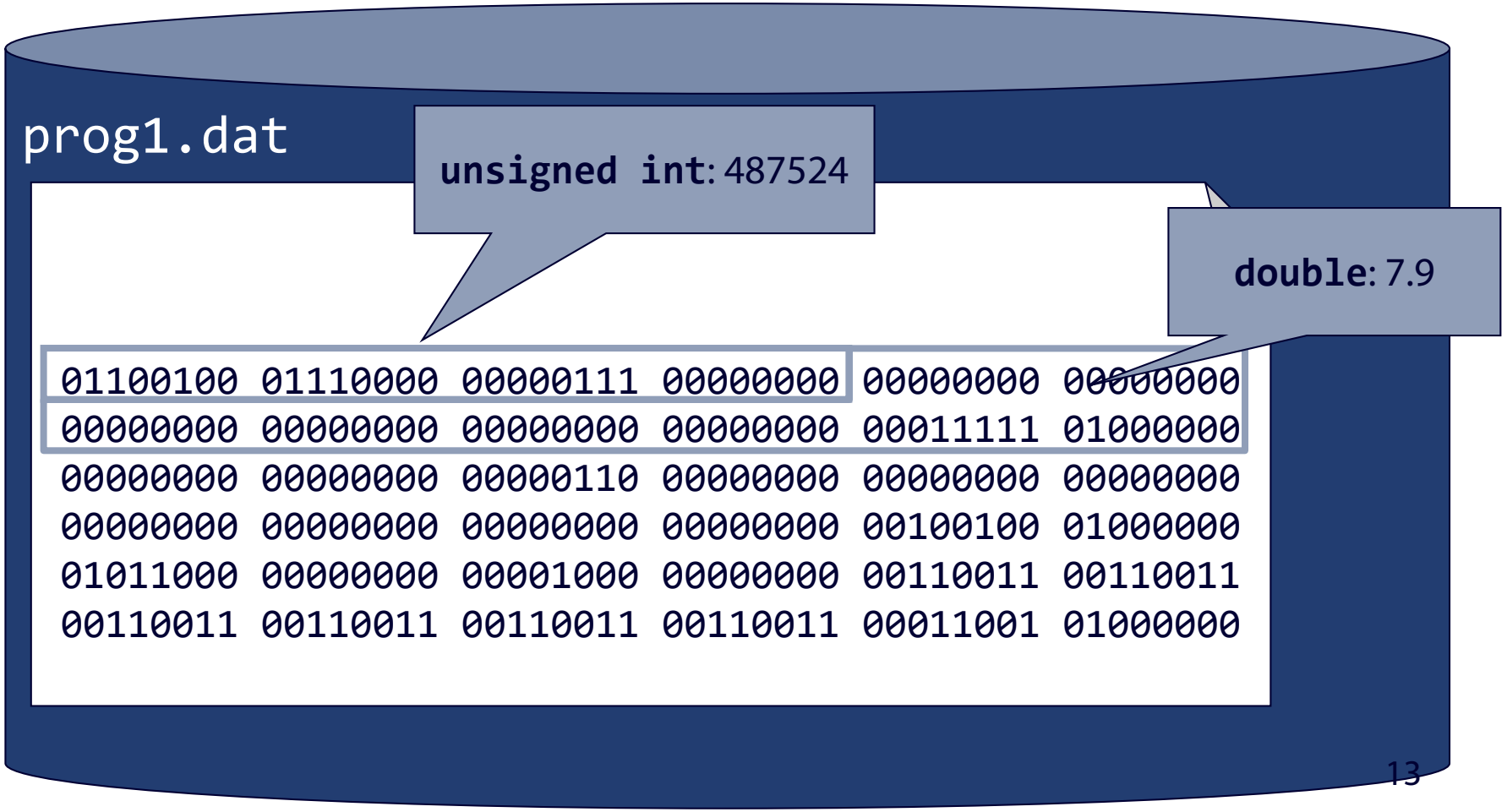
- Asociación
  - `f.open(const string cadena, ios::binary)`
- Lectura
  - `f.read(char buffer[], streamsize n)`
    - `f.read(reinterpret_cast<char*>(&dato), sizeof(dato))`
- Escritura
  - `f.write(const char buffer[], streamsize n)`
    - `f.write(reinterpret_cast<const char*>(&dato), sizeof(dato))`

Creación de un fichero binario

```
Introduzca un NIP (0 para acabar): 487524
Introduzca una nota: 7.9
Introduzca un NIP (0 para acabar): 454844
Introduzca una nota: 10.0
Introduzca un NIP (0 para acabar): 567896
Introduzca una nota: 6.3
Introduzca un NIP (0 para acabar): 0
```



Creación de un fichero binario



Creación de un fichero binario. Sintaxis

```
<fichero_de_notas> ::= { <nota> }
<nota> ::= <nip> <calificación>
<nip> ::= unsigned int
<calificación> ::= double
```

Creación de un fichero binario

```
/*
 * Pre:  ---
 * Post: Ha creado un fichero binario de
 *       nombre «nombreFichero» compuesto
 *       por una secuencia de pares (NIP,
 *       nota) solicitados
 *       interactivamente al usuario.
 */
void crearFicheroNotas(
                const string nombreFichero);
```

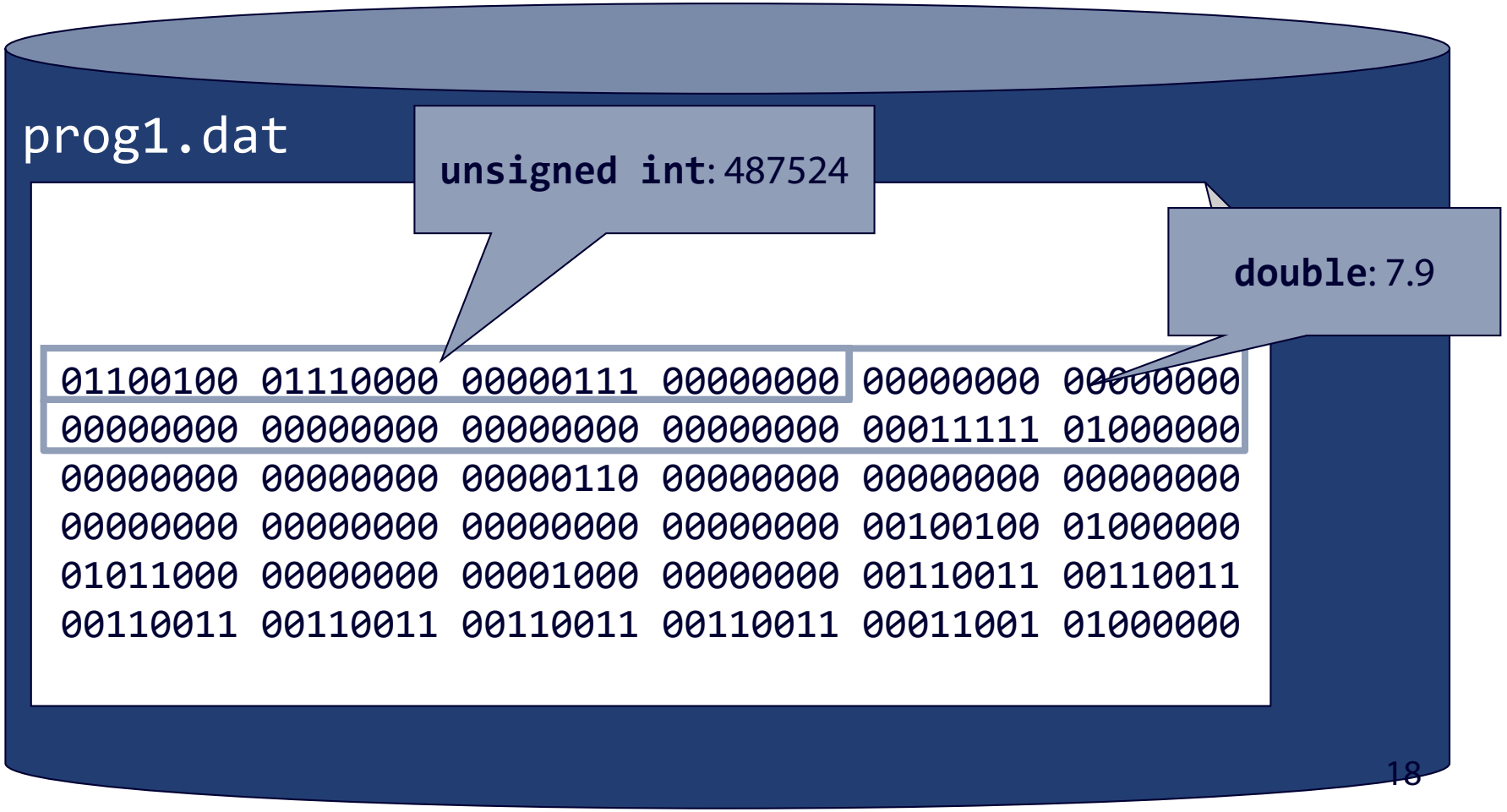
Creación de un fichero binario

```
void crearFicheroNotas(const string nombreFichero) {
    ofstream f(nombreFichero, ios::binary);
    if (f.is_open()) {
        cout << "Introduzca un NIP (0 para acabar): ";
        unsigned int nip;
        cin >> nip;
        while (nip != 0) {
            cout << "Introduzca una nota: ";
            double nota;
            cin >> nota;
            f.write(reinterpret_cast<const char*>(&nip),
                    sizeof(nip));
            f.write(reinterpret_cast<const char*>(&nota),
                    sizeof(nota));
            cout << "Introduzca un NIP (0 para acabar): ";
            cin >> nip;
        }
        ...
```

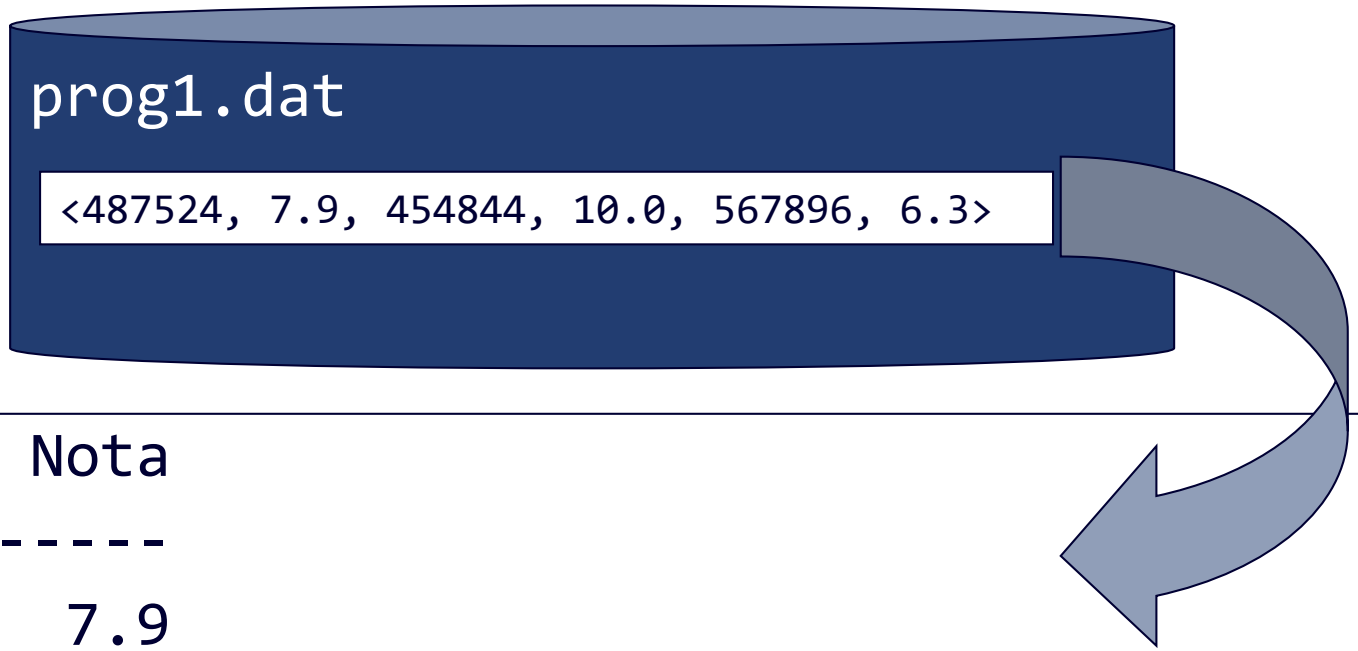
Creación de un fichero binario

```
void crearFicheroNotas(
                        const string nombreFichero) {
        ...
        f.close();
    }
    else {
        cerr << "No se ha podido escribir en el"
             << " fichero \"" << nombreFichero
             << "\"" << endl;
    }
}
```

Creación de un fichero binario



Lectura de un fichero binario




```
  NIP    Nota
------------
487524    7.9
454844   10.0
567896    6.3
```

Lectura de un fichero binario. Sintaxis

```
<fichero_de_notas> ::= { <nota> }
<nota> ::= <nip> <calificación>
<nip> ::= unsigned int
<calificación> ::= double
```

Lectura de un fichero binario

```
/*
 * Pre:  «nombreFichero» es el nombre de un fichero existente
 *       binario cuya estructura consiste en una secuencia de
 *       pares (NIP, nota), de tipos unsigned int y double,
 *       respectivamente.
 * Post: Ha mostrado en la pantalla del contenido del fichero de
 *       nombre «nombreFichero», de acuerdo con el siguiente
 *       formato de ejemplo:
 *
 *         NIP   Nota
 *       ------------
 *       487524   7.9
 *       454844  10.0
 *       567896   6.3
 */
void mostrarFicheroNotas(const string nombreFichero);
```

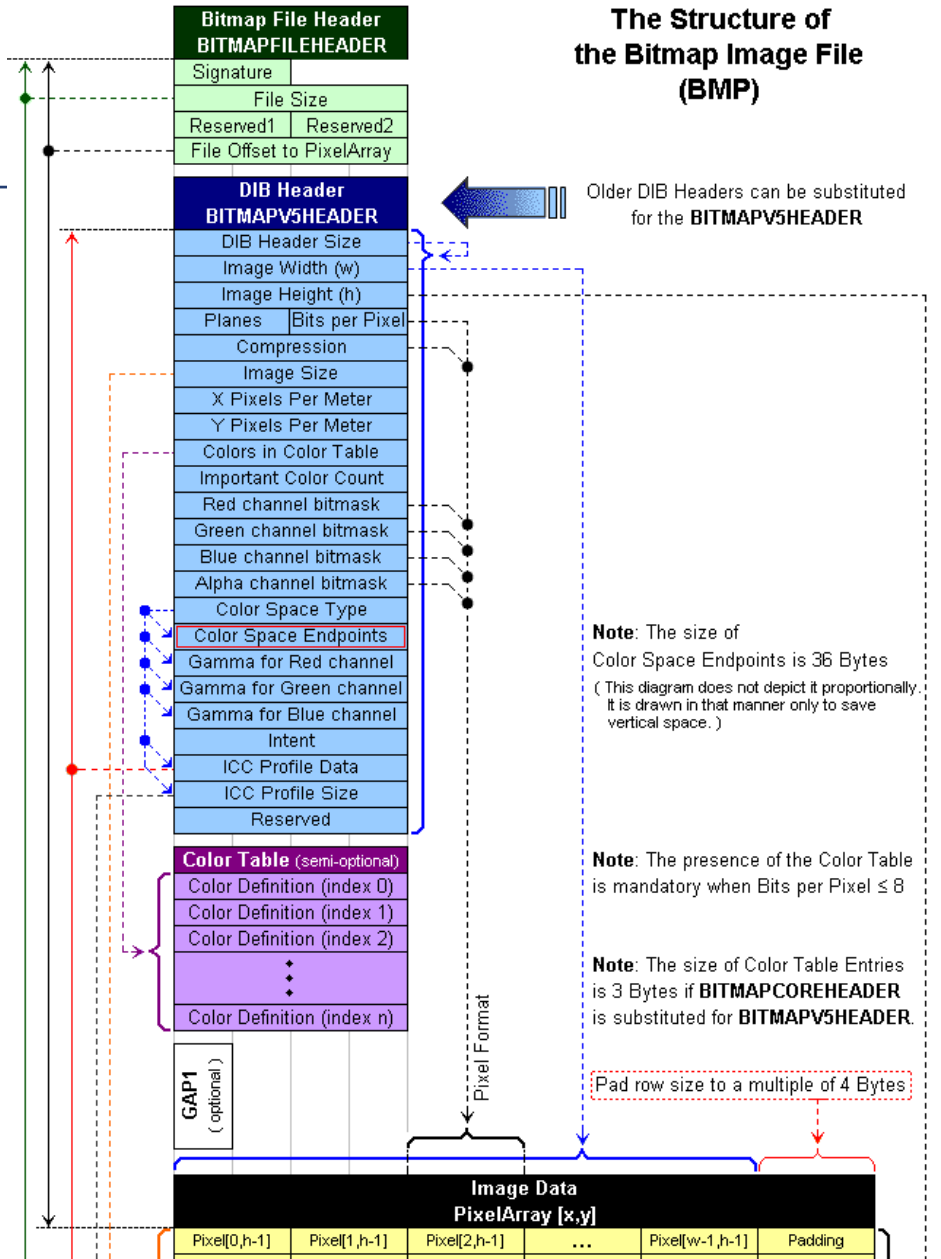
Lectura de un fichero binario

```
void mostrarFicheroNotas(const string nombreFichero) {
    ifstream f(nombreFichero, ios::binary);
    if (f.is_open()) {
        cout << "  NIP   Nota" << endl;
        cout << "------------" << endl;
        cout << fixed << setprecision(1);
        unsigned int nip;
        f.read(reinterpret_cast<char*>(&nip), sizeof(nip));
        while (!f.eof()) {
            double nota;
            f.read(reinterpret_cast<char*>(&nota),
                   sizeof(nota));
            cout << setw(6) << nip << " " << setw(5) << nota
                 << endl;
            f.read(reinterpret_cast<char*>(&nip), sizeof(nip));
        }
        ...
```

Lectura de un fichero binario

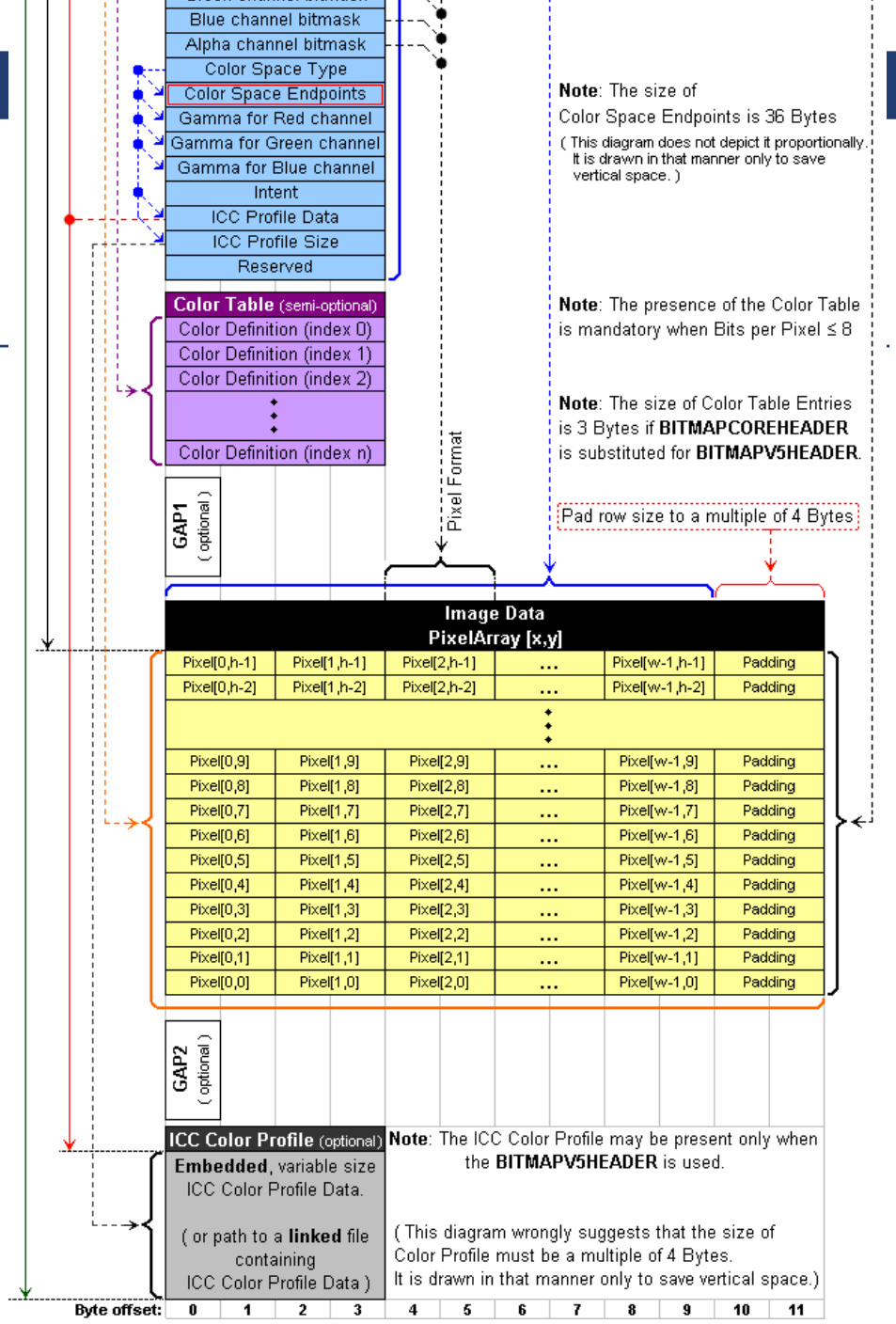
```
void mostrarFicheroNotas(
                        const string nombreFichero) {
        ...
        f.close();
    }
    else {
        cerr << "No se ha podido leer el "
             << "fichero \"" << nombreFichero
             << "\"" << endl;
    }
}
```

Ejemplo. Ficheros BMP



Fuente: File:BMPfileFormat.png. (2020, September 9). *Wikimedia Commons, the free media repository*. Retrieved 18:31, December 4, 2020 from https://commons.wikimedia.org/w/index.php?title=File:BMPfileFormat.png&oldid=452662221.

Ejemplo.
Ficheros BMP



Fuente: File:BMPfileFormat.png. (2020, September 9). *Wikimedia Commons, the free media repository*. Retrieved 18:31, December 4, 2020 from https://commons.wikimedia.org/w/index.php?title=File:BMPfileFormat.png&oldid=452662221.

Ejemplo. Ficheros BMP

```
const int ANCHO = 800;
const int ALTO = 800;
const unsigned int TAM_CABECERA_1 = 18;
const unsigned int TAM_CABECERA_2 = 28;

struct Pixel {
    char rojo, verde, azul;
};

struct Imagen {
    char cabecera1[TAM_CABECERA_1];
    char cabecera2[TAM_CABECERA_2];
    unsigned int ancho, alto;
    Pixel pixels[ANCHO][ALTO];
};
```

Ejemplo. Ficheros BMP

```
/*
 * Pre:  «nombreFichero» es un fichero binario en formato
 *       BMP.
 * Post: Si se ha encontrado el fichero y este tiene unas
 *       dimensiones correctas, tras ejecutar este
 *       procedimiento, «imagen» almacena en memoria la
 *       imagen almacenada en un fichero binario en
 *       formato BMP y la función ha devuelto true. En
 *       caso contrario, ha devuelto false y ha escrito en
 *       la pantalla un mensaje de error indicando la
 *       causa del mismo.
 */
bool leerImagen(const string nombreFichero,
                Imagen& imagen);
```

Ejemplo. Ficheros BMP

```
bool leerImagen(const string nombreFichero, Imagen& imagen) {
    ifstream infile(nombreFichero, ios::binary);
    if (infile.is_open()){
        infile.read(imagen.cabecera1, TAM_CABECERA_1);
        infile.read(reinterpret_cast<char*>(&imagen.ancho), sizeof(int));
        infile.read(reinterpret_cast<char*>(&imagen.alto), sizeof(int));
        infile.read(imagen.cabecera2, TAM_CABECERA_2);
        for (unsigned int i = 0; i < imagen.alto; i++){
            for (unsigned int j = 0; j < imagen.ancho; j++){
                infile.read(&imagen.pixels[i][j].rojo, sizeof(char));
                infile.read(&imagen.pixels[i][j].verde, sizeof(char));
                infile.read(&imagen.pixels[i][j].azul, sizeof(char));
            }
        }
        infile.close();
        return true;
    }
    …
```

Ejemplo.
Ficheros BMP

```
/*
 * Pre:  ---
 * Post: Tras ejecutar este procedimiento,
 *       almacena en disco en un fichero
 *       de nombre «nombreFichero» la
 *       imagen BMP de «imagen».
 */
void guardarImagen(
                  const string nombreFichero,
                  Imagen imagen);
```

Ejemplo. Ficheros BMP

```
void guardarImagen(const string nombreFichero, Imagen imagen) {
    ofstream outFile(nombreFichero, ios::binary);
    if (outFile.is_open()) {
        outFile.write(imagen.cabecera1, TAM_CABECERA_1);
        outFile.write(reinterpret_cast<char*>(&imagen.ancho), sizeof(int));
        outFile.write(reinterpret_cast<char*>(&imagen.alto), sizeof(int));
        outFile.write(imagen.cabecera2, TAM_CABECERA_2);
        for (unsigned int i = 0; i < imagen.alto; i++){
            for (unsigned int j = 0; j < imagen.ancho; j++){
                outFile.write(&imagen.pixels[i][j].rojo, sizeof(char));
                outFile.write(&imagen.pixels[i][j].verde, sizeof(char));
                outFile.write(&imagen.pixels[i][j].azul, sizeof(char));
            }
        }
        outFile.close();
}
    ...
}
```

¿Cómo se puede estudiar este tema?

- Repasando estas transparencias
- Trabajando con el código de estas transparencias
  - https://github.com/prog1-eina/tema-15-ficheros-binarios
- Leyendo
  - Capítulo 15 de los apuntes del profesor Martínez
    - Disponibles en Moodle