# Práctica No. 11

## Uso de campos de bits

**Objetivo:** Crear programas de software donde se almacene la información en memoria secundaria, seleccionando de manera responsable para cada escenario el tipo de archivo que sea adecuado y así garantizar el uso eficiente de los recursos y la integridad de los datos.

**Material:**
- Computadora Personal (PC)
- Programa Editor de texto (ASCII), compilador GCC

**Equipo:**
- Computadora Personal

### Introducción

### Campos de bits

El método que se utiliza en C para operar con campos de bits, está basado en las estructuras (structs) que todos conocemos del lenguaje. Una estructura compuesta por bits se denomina campo de bits. La forma general de definición de un campo de bits es:

Estructura genérica de campos de bits

```
typedef struct bits_
{
   <tipo_dato> nombre1 : <cantidad_bits>;
   <tipo_dato> nombre2 : <cantidad_bits>;
   . . .
   <tipo_dato> nombre3 : <cantidad_bits>;
} campo_bits;
```

Cada campo de bits puede declararse como char, unsigned char, int, unsigned int, etc y su longitud puede variar entre 1 y el número máximo de bits acorde al tipo de dato manejado. Es decir, supongamos que trabajamos en una arquitectura de 32 bits, sabemos que un char o unsigned char ocupa 1 Byte (8 bits), por lo tanto para el char, el campo de bits puede variar entre 1 bit y 8 Bits. Ahora, sabemos que un int o unsigned int, ocupa (en esta arquitectura) 32 bits, por lo tanto, este campo de bits puede variar entre 1 bit y 32 bits.

## Aplicaciones

- En ocasiones es más sencillo la utilización de campos de bits en lugar de emplear máscaras.
- Se facilita la interacción con el hardware debido a que se comunican a través de un protocolo de bytes en donde cada conjunto de bits (incluido conjunto de 1 bit) tienen algún significado.

### Limitaciones

Las variables de campos de bits tienen ciertas restricciones, algunas de ellas son:

- No se puede aplicar el operador & sobre una de estas variables para averiguar su dirección de memoria (es decir, de algún elemento del struct).
- No se pueden construir arrays de un campo de bits (es decir, de algún elemento del struct).
- En algunos entornos, los bits se dispondrán de izquierda a derecha y, en otras, de derecha a izquierda, por lo que la portabilidad puede verse comprometida.

### Ventajas y Desventajas de Operar Bit a Bit

Las ventajas que podemos obtener son:

- Mayor velocidad de procesamiento al momento de ejecutar el programa.
- Más ahorro de memoria en caso de querer modificar variables.
- Son ventajosos al momento de querer programar Microcontroladores (o Microprocesadores).
- Ocupan poco espacio.

Las desventajas que podemos obtener son:

- El código se puede tornar menos legible y bastante confuso.
- Puede quitarle portabilidad al programa.
- Seguido de la anterior, son dependientes de la arquitectura en donde se esté trabajando.

### Desarrollo de la práctica

Leer el encabezado de una imagen de tipo BMP RGB24 e imprimir cada uno de sus valores.

En el siguiente enlace se describe a detalle las partes del encabezado https://en.wikipedia.org/wiki/BMP_file_format.

| Offset | Size | Hex Value | Value | Description |
|---|---|---|---|---|
| BMP Header | | | | |
| 0h | 2 | 42 4D | "BM" | ID field (42h, 4Dh) |
| 2h | 4 | 46 00 00 00 | 70 bytes (54+16) | Size of the BMP file |
| 6h | 2 | 00 00 | Unused | Application specific |
| 8h | 2 | 00 00 | Unused | Application specific |
| Ah | 4 | 36 00 00 00 | 54 bytes (14+40) | Offset where the pixel array (bitmap data) can be found |
| DIB Header | | | | |
| Eh | 4 | 28 00 00 00 | 40 bytes | Number of bytes in the DIB header (from this point) |
| 12h | 4 | 02 00 00 00 | 2 pixels (left to right order) | Width of the bitmap in pixels |
| 16h | 4 | 02 00 00 00 | 2 pixels (bottom to top order) | Height of the bitmap in pixels. Positive for bottom to top pixel order. |
| 1Ah | 2 | 01 00 | 1 plane | Number of color planes being used |
| 1Ch | 2 | 18 00 | 24 bits | Number of bits per pixel |
| 1Eh | 4 | 00 00 00 00 | 0 | BI_RGB, no pixel array compression used |
| 22h | 4 | 10 00 00 00 | 16 bytes | Size of the raw bitmap data (including padding) |
| 26h | 4 | 13 0B 00 00 | 2835 pixels/meter horizontal | Print resolution of the image, 72 DPI × 39.3701 inches per meter yields 2834.6472 |
| 2Ah | 4 | 13 0B 00 00 | 2835 pixels/meter vertical | |
| 2Eh | 4 | 00 00 00 00 | 0 colors | Number of colors in the palette |
| 32h | 4 | 00 00 00 00 | 0 important colors | 0 means all colors are important |
| Start of pixel array (bitmap data) | | | | |
| 36h | 3 | 00 00 FF | 0 0 255 | Red, Pixel (0,1) |
| 39h | 3 | FF FF FF | 255 255 255 | White, Pixel (1,1) |
| 3Ch | 2 | 00 00 | 0 0 | Padding for 4 byte alignment (could be a value other than zero) |

| | | | | |
|---|---|---|---|---|
| 3Eh | 3 | FF 00 00 | 255 0 0 | Blue, Pixel (0,0) |
| 41h | 3 | 00 FF 00 | 0 255 0 | Green, Pixel (1,0) |
| 44h | 2 | 00 00 | 0 0 | Padding for 4 byte alignment (could be a value other than zero) |

Tabla 1. Ejemplo de una imagen BMP de 2x2 pixeles con un formato de píxeles RGB24. Fuente: https://en.wikipedia.org/wiki/BMP_file_format

**Tareas a realizar (50%):**

1. Definir una estructura en la cual se pueda guardar el encabezado de una imagen BMP.
2. Leer la información de la imagen ejemplo.bmp y ejemplo2.bmp e imprimir los siguientes campos del header (sección sombreada en amarillo):
    a. ID
    b. Tamaño del archivo BMP
    c. Ancho de la imagen
    d. Largo de la imagen

**Preguntas a contestar (50%):**

1. Ilustre la(s) estructuras utilizadas en este ejercicio.
2. ¿Por qué una estructura con los campos descritos en el encabezado no mide los 54 bytes calculados?
3. ¿De qué forma se puede leer de forma correcta el archivo?
4. ¿Qué soluciones se proponen para almacenar el encabezado en una estructura?
5. Proponga un procedimiento para escribir en otro archivo el mismo encabezado pero con ciertas modificaciones.

**Recomendaciones**

- Usar un lector de archivos binarios en hexadecimal
- Comparar los valores del encabezado con los de la imagen
-

**Restricciones**

1. El código fuente deberá contener comentarios que indique el objetivo del programa y descripción de instrucciones más relevantes.
2. Evitar nombres de variables y funciones que no reflejen su propósito claramente.
3. Cuando se haga uso de memoria dinámica es importante no olvidar liberar la memoria.

**Conclusiones y Comentarios.**