Manipulación de imágenes

Práctica de programación en C

Informática

Grado en Ingeniería en Tecnologías Industriales

Grado en Ingeniería de la Energía

v.2526

Sesión 12

Objetivos de la práctica

En la práctica se plantea la resolución de un problema específico que nos permita lograr los siguientes objetivos:

- Utilizar ficheros para lectura y almacenamiento de imágenes.
- Utilizar matrices de enteros para el almacenamiento y manipulación de imágenes.
- Aplicar técnicas de programación modular.

Problema a resolver: Manipulación de imágenes

La práctica consiste en escribir un programa en C que nos permita leer y manipular imágenes representadas en formato **`PGM`**.

La imagen original que se suministra en la práctica se corresponde con una imagen que ha servido de prueba para los algoritmos de compresión de imagen y se ha convertido de facto en un estándar industrial y científico.

El programa nos ofrecerá las siguientes opciones:

1. **Rotar la imagen**. El programa generará otra imagen resultado de rotar 90º en sentido antihorario la imagen original. Ejemplo:

Imagen Original

Imagen Resultado

2. **Blanco y negro**. El programa generará una imagen resultado de cambiar a blanco y negro la imagen original. Ejemplo:

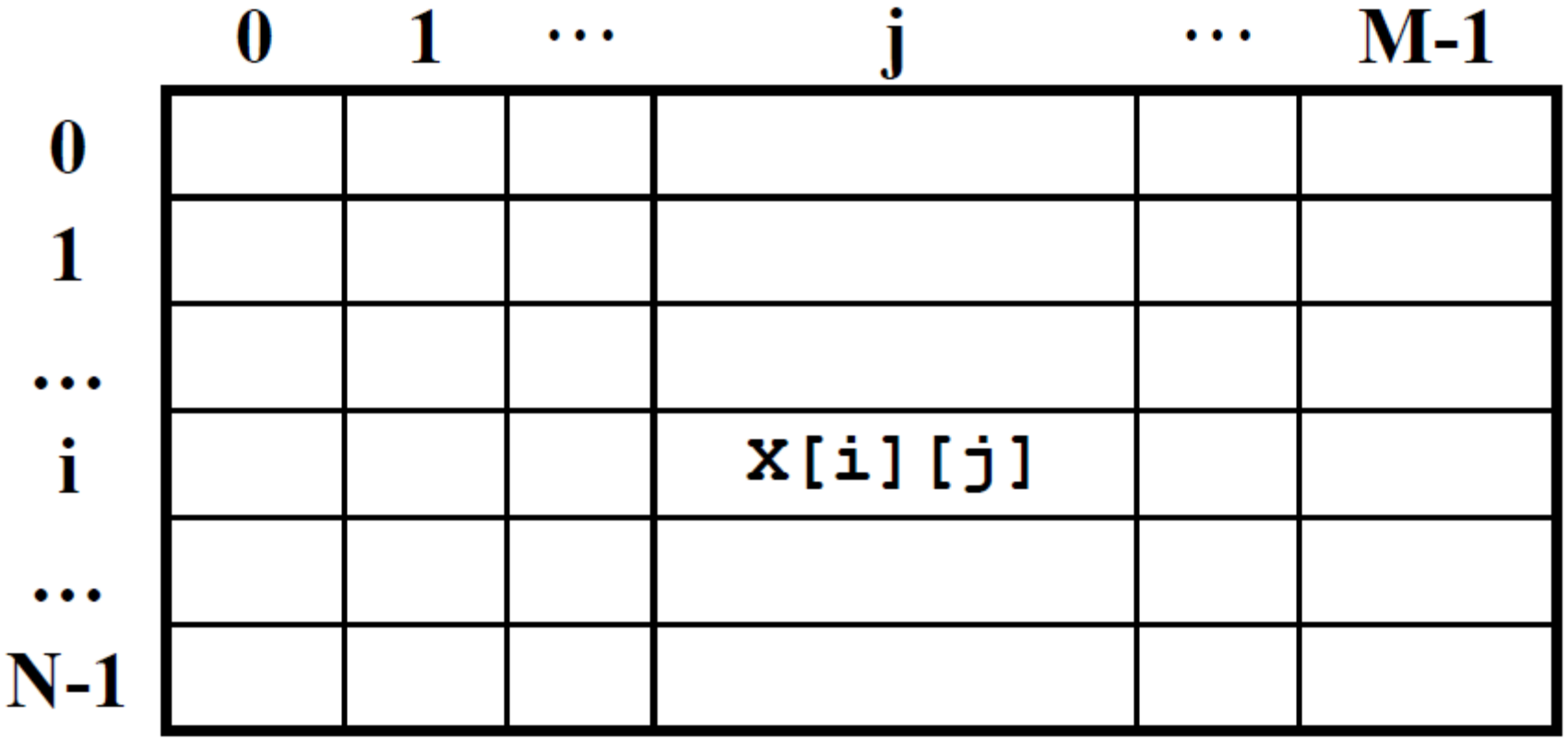
Imagen Original

Imagen Resultado

Vectores multidimensionales: matrices

- Declaración de una matriz estática de dimensión N x M (N filas y M columnas):

```
Tipo_dato  nombre_matriz[N][M];
```



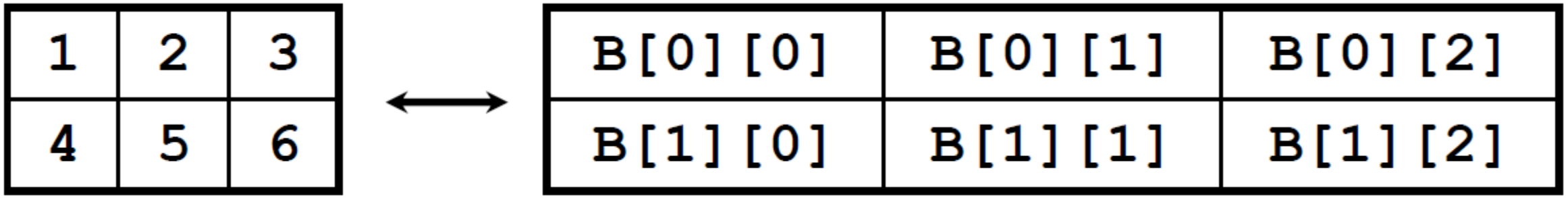
Vectores multidimensionales: matrices

- Acceso al elemento que ocupa la fila i-ésima y columna j-ésima:

nombre_matriz[i][j]

- Ejemplo:

```
int B[2][3]={{1, 2, 3}, {4, 5, 6}};
int C[2][3]={1, 2, 3, 4, 5, 6}; //Las matrices B y C son iguales
```

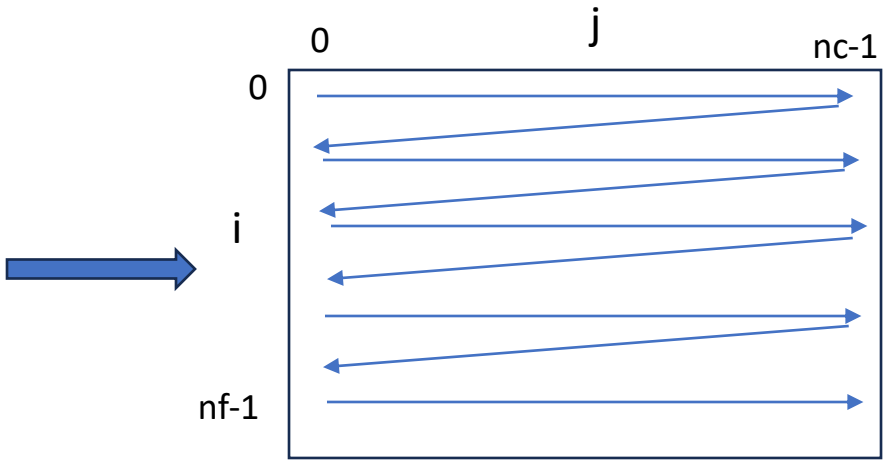



```
printf("El elemento de B que ocupa la fila 1 y columna 2 vale %d", B[1][2]);
```

Vectores multidimensionales: matrices

- Para recorrer una matriz A de **nf** filas y **nc** columnas se usarían dos bucles, uno anidado en el otro:

```
for(i=0; i<nf; i++)
    for(j=0; j<nc; i++){
        asignación, lectura o escritura del elemento A[i][j];
        ……;
}
```



- Ejemplo: Para escribir la matriz B del ejemplo anterior, utilizaríamos el siguiente código:

```
for(i=0; i<nf; i++)
    for(j=0; j<nc; i++){
        printf("A[%d][%d]=%d", i, j, A[i][j]);
}
```

Ficheros gráficos en formato PGM

El formato **Portable Graymap Format** (PGM) es un formato de gráficos simple que almacena imágenes en escala de grises. Cada píxel de la imagen se representa por un número entero que indica la intensidad de gris de ese punto. El formato de un archivo PGM es el siguiente:

- **Primera línea**: Aparece un identificador (cadena mágica) que indica el tipo de fichero: de texto o binario. En nuestro caso, como los ficheros que utilizaremos en la práctica son en formato de texto, en la primera línea del fichero siempre aparecerá la cadena P2.
- **Segunda línea**: Aparecen dos números enteros separados por un espacio blanco que se corresponden con el ancho y el alto de la imagen en píxeles. Es decir, el número de columnas (ancho) y de filas (alto) que se utilizarán en la matriz.
- **Tercera línea**: Aparece un número entero que indica el máximo valor de la escala de grises. Generalmente es 255.
- **Cuarta línea y siguientes**: A partir de la cuarta línea hasta el final de fichero se indican los valores en escala de grises de cada píxel de la imagen, línea a línea.

```
P2
8 8
255
0   0   0   0   0   0   0   0
40  255 255 255 255 255 255 0
80  255 150 255 255 255 255 0
120 255 100 100 255 255 255 0
160 255 50  50  50  255 255 0
200 255 0   0   0   0   255 0
240 255 255 255 255 255 255 0
0   0   0   0   0   0   0   0
```

- Valor 0 corresponde al negro, 255 al blanco, demás valores a escalas de grises entre el negro y el blanco
- Para visualizar las imágenes puedes utilizar alguno de los visores de imágenes que se encuentran en Poliformat
- **IrfanView** es un visor gratuito que lo puedes obtener en http://www.irfanview.com/ o en Microsoft Store

Fichero Lenna.pgm (Poliformat)

Lenna.pgm

```
P2
256 256
255
161 161 162 162 163 162 160 158 158 159 157 156 157 156 15
161 161 161 162 162 160 158 157 156 158 156 155 156 155 15
161 161 161 160 160 159 157 155 155 156 155 154 156 155 15
159 159 159 159 159 158 156 155 154 156 154 153 155 154 15
158 158 158 158 159 158 157 156 155 157 154 153 154 153 15
156 156 156 156 157 157 157 156 156 157 155 153 155 153 15
157 156 156 156 156 156 156 155 156 157 155 154 155 154 15
157 157 156 156 156 156 155 154 155 157 155 154 156 155 15
157 156 156 156 157 157 157 156 155 156 156 155 153 153 15
156 156 155 156 157 157 157 157 157 158 157 155 153 154 15
156 155 155 156 157 157 157 157 159 159 158 155 153 154 15
156 155 155 156 157 158 158 158 159 159 157 155 154 155 15
155 155 155 156 157 158 158 158 158 157 156 155 155 158 16
157 156 156 157 158 159 159 158 157 157 157 157 158 161 16
158 158 157 158 159 159 159 158 158 158 159 160 162 165 16
159 158 158 158 159 159 159 158 160 160 161 163 164 167 16
158 157 156 156 158 159 159 159 161 160 161 163 166 168 16
158 157 157 157 159 161 162 162 162 162 164 166 168 169 16
160 159 158 159 160 162 163 163 163 164 167 168 168 168 16
163 162 160 160 160 161 161 162 163 166 168 168 167 165 16
```

Funciones a implementar

Rotar la imagen. El programa generará otra imagen resultado de rotar 90^o en sentido antihorario la imagen original. Ejemplo: [1]

Imagen Original

Imagen Resultado

Funciones a implementar

Blanco y negro. El programa generará una imagen en blanco y negro del original. Ejemplo:

Funciones a implementar

Cambiar Luminosidad. El programa generará otra imagen resultado de cambiar la luminosidad de la imagen original. Ejemplo:

Funciones a implementar

Espejo. El programa generará otra imagen que será el resultado de darle la vuelta a la imagen original. Ejemplo:

Fichero imagenes.c (descargar de Poliformat)


```
int main(){
  /* Declaracion de variables */
  int opc;
  /* Leer fichero pgm con la imagen -> Funcion lee_imagen */
  /* si error de lectura -> Finalizar programa */
  do{
     opc = menu(); /*Función ya implementada*/
     switch(opc){
        case 1:
           /* Opcion Rotar imagen -> Funcion rotar_imagen */
           break;
        case 2:
           /* Opcion Blanco y negro -> Funcion blanco_negro */
           break;
        case 3:
           /* Opcion cambiar_luz */
           /* Solicitar el valor de variación de intensidad (k)  */
           /* Cambiar la luminosidad -> Funcion cambiar_luz */
           break;
        case 4:
           /* Opcion espejo -> Funcion espejo */
           break;
     }
     /* Si opcion distinta de terminar escribir la imagen resultante -> Funcion
escribe_imagen */
  }while (opc != 0);
  return 0;
}
```

int lee_imagen (int img[MAXF][MAXC], int *nf, int *nc)

Esta función leerá de un fichero en formato pgm la imagen y la almacenará en una matriz.

- Parámetros: La función tiene como parámetros de salida la matriz img donde se almacenará la imagen y dos punteros que indican el número de filas y de columnas de la matriz img.
- Valor de retorno: Para controlar errores de lectura, la función devolverá -1 si el fichero no ha podido abrirse correctamente o 0 si todo ha ido bien.

```
int lee_imagen (int img[MAXF][MAXC], int *nf, int *nc){
  FILE *f;
  char nomfich[MAXCAD], tipo[MAXCAD]; //variable tipo sirve para leer la 1ª línea
  int nivmax, col, fil, i, j;
  printf("Introduce el nombre de la imagen\n");
  fflush(stdin);
  gets(nomfich);

//Abrir fichero con tratamiento de errores
//leer tipo del fichero (no se usa después)
//leer el número de columnas (col) y filas (fil) del fichero
//leer nivel de grises (no se usa después)
  for(i=0;i<fil;i++)//Recorre de arriba hacia abajo y de izquierda a derecha los valores de los pixels
    for(j=0;j<col;j++)
      //Leer del fichero img[i][j]
  *nf = fil;  *nc = col;
  fclose(f);
  return 0;
}
```

Ejemplo de ficher pgm:

P2

8 8

255

0 0 0 0 0 0 0 0

1 0 1 0 1 1 0 1

.....

```
int escribe_imagen(int img[MAXF][MAXC], int nf, int nc)
```


Esta función escribirá la información de una imagen en un fichero. Los datos de la imagen se escribirán en el fichero siguiendo el formato pgm.

- Parámetros: La función tiene como parámetros la matriz img con los datos de la imagen y el número de filas y el número de columnas de la matriz img.
- Valor de retorno: Para controlar errores de escritura, la función retorna -1 si el fichero no ha podido abrirse correctamente o o si todo ha ido bien.

Funcionamiento:

- Abrir y leer por teclado el nombre del fichero. Son los mismos pasos que en la función anterior, pero ahora escribiendo todos los datos en img. Primero se escribirán las tres primeras líneas y después se recorrerán los elementos de la matriz mediante 2 bucles, uno anidado en el otro.

```
P2
nc nf
255
x  x  x  x  x  x  x  x
……..
```

```
void blanco_negro(int img[MAXF][MAXC], int img2[MAXF][MAXC], int nf, int nc)
```


Esta función debe transformar la imagen original en una imagen en blanco y negro.

- Parámetros: La función recibe como parámetros dos matrices: la matriz img con la imagen original, la matriz img2 donde almacenar la imagen en blanco y negro y el número de filas **nf** y el número de columnas **nc** de la matriz imagen img.
- Valor de retorno: Ninguno.

Funcionamiento:

- Deberemos almacenar en la imagen resultado todos los píxeles al valor mínimo (0) o al valor máximo (255).
- La función recorrerá la imagen original. Para cada píxel que sea inferior a 127 se almacenará en la imagen resultado el valor 0 y 255 en caso contrario.

```
void cambiar_luz(int img[MAXF][MAXC], int img2[MAXF][MAXC], int nf, int nc, int k)
```


Esta función cambia la luminosidad de la imagen original.

- Parámetros: La matriz **img** con la imagen original, la matriz **img2** donde almacenar la imagen resultante, el número de filas **nf** y el número de columnas **nc** de la imagen, y finalmente el valor k, el cual puede ser un valor positivo, en cuyo caso se aumenta la luz, o un valor negativo en cuyo caso se oscurece.
- Valor de retorno: Ninguno.

Funcionamiento:

- La función recorrerá la imagen original (mediante dos bucles, uno anidado dentro del otro) y almacenará en la imagen resultado el valor del píxel sumándole el valor k.
- Hay que controlar que, al sumar k a un píxel, el valor resultante no sea superior al valor máximo (255) ni inferior al valor mínimo (0).

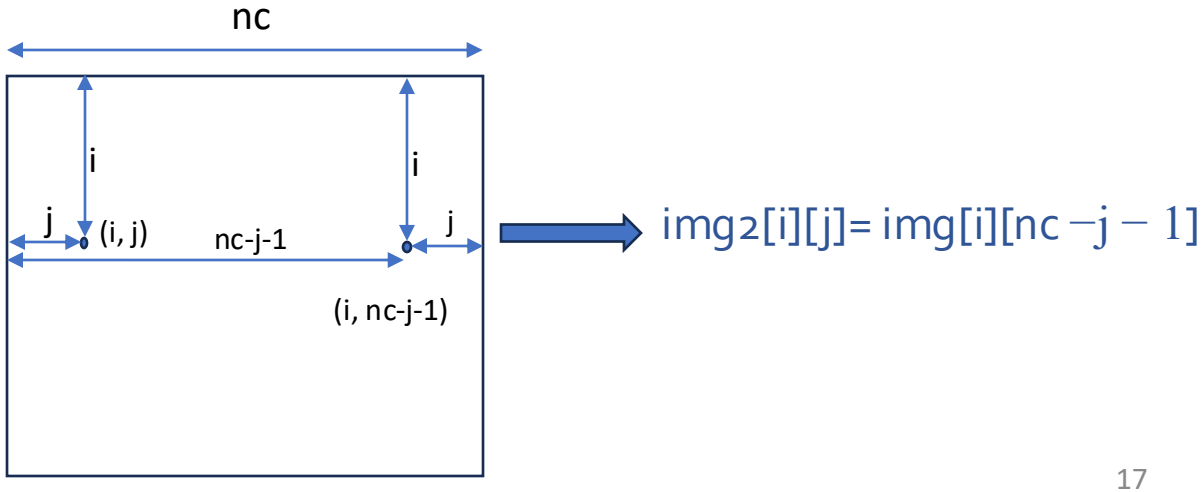
```
void espejo(int img[MAXF][MAXC], int img2[MAXF][MAXC], int nf, int nc)
```

Esta función calcula la imagen que se obtiene al darle la vuelta a la imagen original.

- Parámetros: La matriz img con la imagen original, la matriz img2 donde almacenar la imagen "espejo", y el número de filas **nf** y el número de columnas **nc** de la matriz imagen img.
- Valor de retorno: Ninguno.

Funcionamiento:

- La función recorrerá la imagen original (mediante dos bucles, uno anidado dentro del otro) y almacenará los valores adecuados en la imagen resultado, de manera que a cada píxel de la imagen resultante se le asigna el simétrico de la original.
- Por ejemplo, para cada fila i, el valor img2[i][j] de la imagen "espejo" deberá ser igual a img[i][nc −j − 1] de la imagen original.



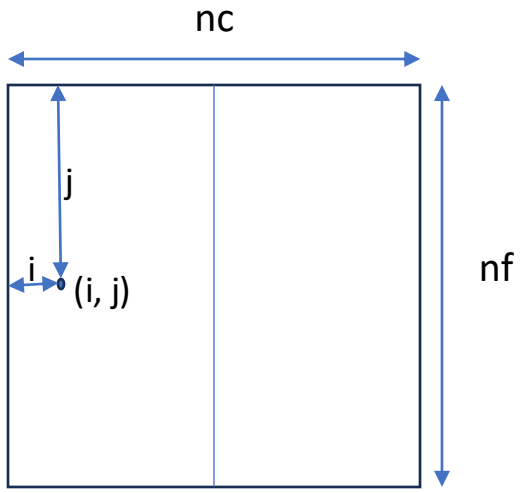
```
void rotar_imagen(int img[MAXF][MAXC], int img2[MAXF][MAXC], int nf, int nc)
```

Esta función debe rotar la imagen original 90º en sentido antihorario.

- Parámetros: La matriz img con la imagen original, la matriz img2 donde almacenar la imagen rotada, y el número de filas **nf** y el número de columnas **nc** de la matriz imagen img .
- Valor de retorno: Ninguno.

Funcionamiento:

- La función deberá recorrer la imagen original (mediante dos bucles, uno anidado dentro del otro) y almacenará los valores adecuados en la imagen resultado, de manera que se obtenga una imagen rotada.
- Para ello, dentro del bucle anidado el valor img2[i][j] de la imagen modificada deberá ser igual a img[j][nc-i-1] de la imagen original.



img2[i][j]= img[j][nc-i-1]

Muchas gracias
y
¡ a programar !