Manipulació d'imatges

Pràctica de programació en C

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Sessió 12

Objectius de la pràctica

En la pràctica es planteja la resolució d'un problema específic que ens permeta aconseguir els següents objectius:

- Utilitzar fitxers per a lectura i emmagatzematge d'imatges.
- Utilitzar matrius d'enters per a l'emmagatzematge i manipulació d'imatges.
- Aplicar tècniques de programació modular.

Problema a resoldre: Manipulació d'imatges

La pràctica consisteix a escriure un programa en C que ens permeta llegir i manipular imatges representades en format **PGM**.

La imatge original que se subministra en la pràctica es correspon amb una imatge que ha servit de prova per als algorismes de compressió d'imatge i s'ha convertit de facto en un estàndard industrial i científic.

El programa ens oferirà les següents opcions:

1. **Girar la imatge**. El programa generarà una altra imatge resultat de girar 90º en sentit antihorari la imatge original. Exemple:

Imatge Original

Imatge Resultat

2. **Blanc i negre**. El programa generarà una imatge resultat de canviar a blanc i negre la imatge original. Exemple:

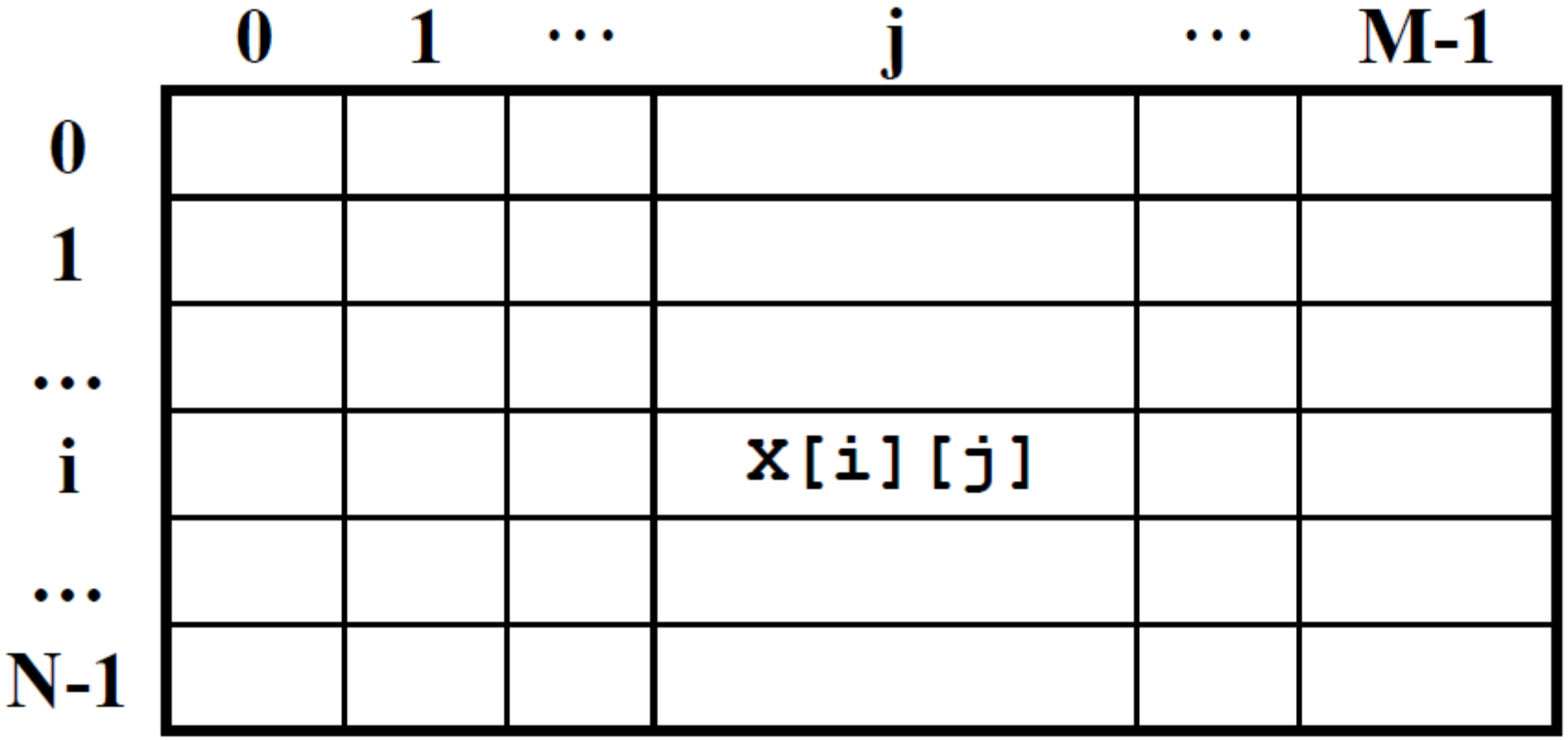
Imatge Original

Imatge Resultat

Vectores multidimensionales: matrices

- Declaració d'una matriu estàtica de dimensió N x M (N files i M columnes):

  Tipus_de_dada nombre_matriz[N][M];



Vectores multidimensionales: matrices

- Accés a l'element que ocupa la fila i-èsima i la columna j-èsima:

nombre_matriu[i][j]

- Exemple:

```
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
```

1	2	3
4	5	6

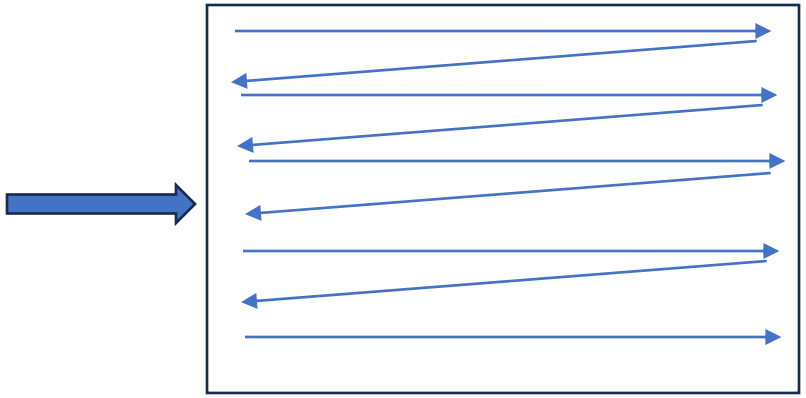
B[0][0]	B[0][1]	B[0][2]
B[1][0]	B[1][1]	B[1][2]

```
printf("L'element de B que ocupa la fila 2 i la columna 3 %d", B[2][3]);
B[1][3]=2*B[1][1]; //El elemento (1,3) de B pasa a valer 2*1=2
```

Vectores multidimensionales: matrices

- Per a recórrer una matriu A de nf files y nc columnes s'empraran dos bucles, un dins de l'altre:

```
for(i=0; i<nf; i++)
    for(j=0; j<nc; i++){
        Assignació, lectura o escriptura de l'element A[i][j];
        ……;
}
```



- Per a escriure la matriu B de l'exemple anterior, utilitzaríem el codi següent.

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

Fitxers gràfics en format PGM

El format **Portable Graymap Format** (PGM) és un format de gràfics simple que emmagatzema imatges en escala de grisos. Cada píxel de la imatge es representa per un nombre enter que indica la intensitat de grisa d'aqueix punt. El format d'un arxiu PGM és el següent:

- **Primera línia**: Apareix un identificador (cadena màgica) que indica el tipus de fitxer: de text o binari. En el nostre cas, com els fitxers que utilitzarem en la pràctica són en format de text, en la primera línia del fitxer sempre apareixerà la cadena P2.
- **Segona línia**: Apareixen dos nombres enters separats per un espai blanc que es corresponen amb l'ample i l'alt de la imatge en píxels. És a dir , el nombre de columnes (ample) i de files (alt) que s'utilitzaran en la matriu.
- **Tercera línia**: Apareix un nombre enter que indica el màxim valor de l'escala de grisos. Generalment és 255.
- **Quarta línia i següents**: A partir de la quarta línia fins al final de fitxer s'indiquen els valors en escala de grisos de cada píxel de la imatge. línia a línia.

```
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
```

- El valor 0 correspon a negre, 255 a blanc, i els altres valors a escales de gris entre el negre i el blanc
- Per a visualitzar la imatge pots utilitzar algun dels visors d'imatges enllaçats en la carpeta de la pràctica de poliformat
- **IrfanView** és un visor gratuït que el pots obtindre en https://www.irfanview.com/ o en Microsoft Store.

Fitxer 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
```

Funcions a implementar

1. **Girar la imatge**. El programa generarà una altra imatge resultat de girar 90° en sentit antihorari la imatge original. Exemple:

Imatge Original

Imatge Resultat

Funcions a implementar

2. **Blanc i negre**. El programa generarà una imatge resultat de canviar a blanc i negre la imatge original. Exemple:

Imatge Original

Imatge Resultat

Funcions a implementar

3. **Canviar lluminositat**. El programa generarà una altra imatge resultat de canviar la lluminositat de la imatge original. Exemple:

Imatge Original

Imatge Resultat

Funcions a implementar

4. **Espill**. El programa generarà una altra imatge que serà el resultat de donar-li la volta a la imatge original. Exemple:

Imatge Original

Imatge Resultat

Ficher 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)`

Aquesta funció llegirà d'un fitxer en format PGM la imatge i l'emmagatzemarà en una matriu.

- Paràmetres: La funció té com a paràmetres d'eixida la matriu img, on s'emmagatzemarà la imatge, i dos punters que indiquen el nombre de files i de columnes de la matriu img.
- Valor de retorn: Per a controlar errors de lectura, la funció retornarà -1 si el fitxer no s'ha pogut obrir correctament o 0 si tot ha anat bé.

```
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)`

Aquesta funció escriurà la informació d'una imatge en un fitxer. Les dades de la imatge s'escriuran en el fitxer seguint el format PGM.

- Paràmetres: La funció té com a paràmetres la matriu img amb les dades de la imatge i el nombre de files i de columnes de la matriu img.
- Valor de retorn: Per a controlar errors d'escriptura, la funció retornarà -1 si el fitxer no s'ha pogut obrir correctament o 0 si tot ha anat bé.

Funcionament:

Obrir i llegir per teclat el nom del fitxer. Són els mateixos passos que en la funció anterior, però ara escrivint totes les dades d'img. Primer s'escriuran les tres primeres línies i després es recorreran els elements de la matriu mitjançant dos bucles, un dins de l'altre.

```
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)
```


Aquesta funció ha de transformar la imatge original en una imatge en blanc i negre.

- Paràmetres: La funció rep com a paràmetres dues matrius: la matriu **img** amb la imatge original, la matriu **img2** on s'emmagatzemarà la imatge en blanc i negre, i el nombre de files **nf** i el nombre de columnes **nc** de la matriu d'imatge img.
- Valor de retorn: Cap.

Funcionament:

- Haurem d'emmagatzemar en la imatge resultat tots els píxels al valor mínim (0) o al valor màxim (255).
- La funció recorrerà la imatge original: per a cada píxel que siga inferior a 127 s'emmagatzemarà en la imatge resultat el valor 0, i 255 en cas contrari.

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


Aquesta funció canvia la lluminositat de la imatge original.

- Paràmetres: La matriu img amb la imatge original **img**, la matriu **img2** on s'emmagatzemarà la imatge resultant, el nombre de files **nf** i el nombre de columnes **nc** de la imatge, i finalment el valor **k**, que pot ser un valor positiu (augmenta la llum) o un valor negatiu (l'enfosqueix).
- Valor de retorn: Cap.

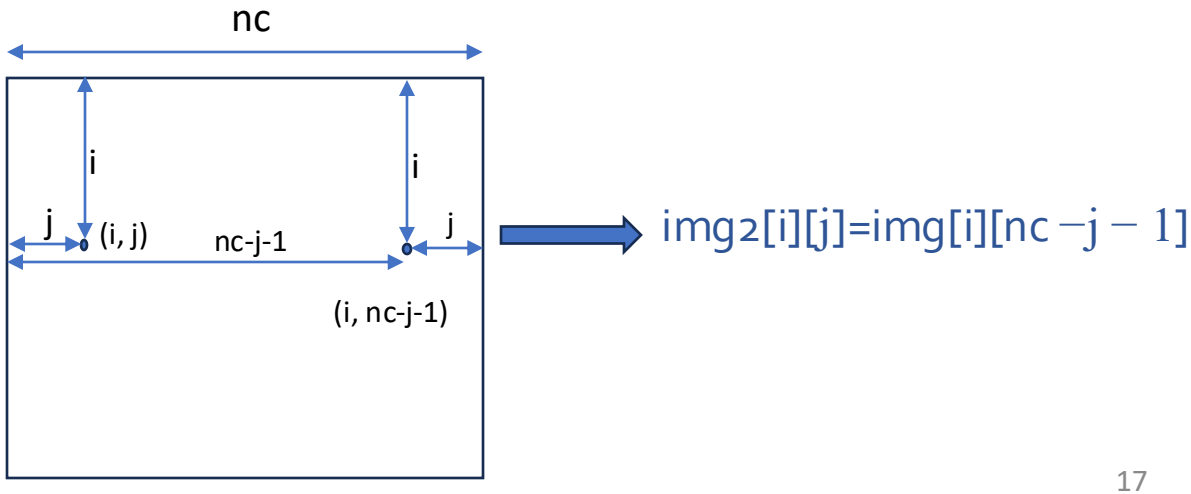
- Funcionament:
- La funció recorrerà la imatge original (mitjançant dos bucles, un dins de l'altre) i emmagatzemarà en la imatge resultat el valor del píxel sumant-li el valor k.
- Cal controlar que, en sumar k a un píxel, el valor resultant no siga superior al valor màxim (255) ni inferior al valor mínim (0).

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

Aquesta funció calcula la imatge que s'obté en donar-li la volta a la imatge original.

- Paràmetres: La matriu img amb la imatge original, la matriu img2 on s'emmagatzemarà la imatge "espill", i el nombre de files nf i el nombre de columnes nc de la matriu d'imatge img.
- Valor de retorn: Cap.

- Funcionament:
  - La funció recorrerà la imatge original (mitjançant dos bucles, un dins de l'altre) i emmagatzemarà els valors adequats en la imatge resultat, de manera que a cada píxel de la imatge resultant se li assigna el simètric de l'original.
  - Per exemple, per a cada fila i, el valor img2[i][j] de la imatge "espill" haurà de ser igual a img[i][nc-j-1] de la imatge original



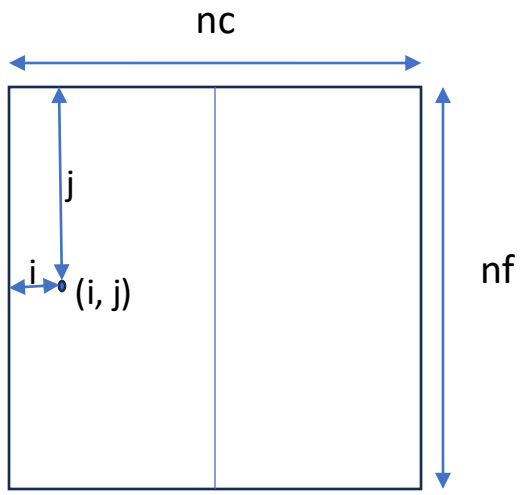
```
void rotar_imagen(int img[MAXF][MAXC], int img2[MAXF][MAXC], int nf, int nc)
```


Aquesta funció ha de rotar la imatge original 90° en sentit antihorari.

- Paràmetres: La matriu **img** amb la imatge original, la matriu **img2** on s'emmagatzemarà la imatge rotada, i el nombre de files **nf** i el nombre de columnes **nc** de la matriu d'imatge **img**.
- Valor de retorn: Cap.

Funcionament:

- La funció haurà de recórrer la imatge original (mitjançant dos bucles, un dins de l'altre) i emmagatzemar els valors adequats en la imatge resultat, de manera que s'obtinga una imatge rotada.
- Per a això, dins del bucle imbricat el valor img2[i][j] de la imatge modificada haurà de ser igual a img[j][nc-i-1] de la imatge original.



Moltes gràcies
y
¡ a programar !