Universidad de Costa Rica
CI-0113: Programación 2
Laboratorio #2

Objetivo: Familiarizar al estudiante con el concepto y uso de matrices en C++.

Enunciado

Resuelva los siguientes problemas. Escriba un código en C/C++ que resuelva los siguientes problemas:

1. Implemente una función que reciba una matriz de enteros, su cantidad de filas y columnas. La función debe empezar a rellenar la matriz con ceros y unos. Para ello contará el número de las casillas: si el número de casilla es número compuesto entonces guardará 0, si es número primo entonces guardará 1.
    a. Empezará a contar en 1.
    b. La matriz se procesa en orden filas-primero (*row-major*)
    c. La matriz puede venir vacía, el número de la casilla no es lo mismo que el valor almacenado en ella.
    d. Ejemplo, una matriz de 3 filas y 4 columnas:

```
original          número de celda     resultado guardado en matriz
[ 0 0 0 0 ]       [ 1  2  3  4 ]      [ 0  1  1  0 ]
[ 0 0 0 0 ] ------> [ 5  6  7  8 ]  ------> [ 1  0  1  0 ]
[ 0 0 0 0 ]       [ 9 10 11 12]       [ 0  0  1  0 ]
```

2. El juego de la vida de Conway es un experimento clásico en la computación ideado en 1970 por el matemático británico John Horton Conway (quien falleció producto del COVID-19 en abril). Este experimento toma inspiración en el comportamiento de organismos unicelulares que dependen de otros de su misma especie para sobrevivir. La idea inicial de Conway es que podemos simular el proceso de "vida" y "muerte" de un conjunto de células dentro de la computadora utilizando una matriz y siguiendo un conjunto de reglas simples:
    a. Cada cuadrícula/célula puede estar en 0 (si la célula está muerta) o en 1 (si la célula está viva).
    b. Cada "ciclo" o "paso" de la simulación:
        i. Las células muertas que estaban rodeadas por exactamente 3 células vivas ahora están vivas (por reproducción).
        ii. Las células vivas que estaban rodeadas por más de 3 células vivas mueren (por sobrepoblación).
        iii. Las células vivas que estaban rodeadas por menos de 2 células vivas mueren (por soledad).
    c. Implemente una función que recibe como parámetro una matriz de enteros, su cantidad de filas y de columnas. Luego aplique un paso de la simulación de Conway, modificando los valores ubicados en la matriz. Tome en

consideración que los cambios deben ocurrir de manera instantánea con base en la matriz original recibida.

d. Si lo desea, puede asumir que la matriz es toroidal (la primera fila conecta con la última, la primera columna con la última).
e. Puede utilizar el siguiente pulsar para probar sus resultados:

```
unsigned char pulsarMat[17][17] =
{{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                        {0,0,0,0,1,1,1,0,0,0,1,1,1,0,0,0,0},
                        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                        {0,0,1,0,0,0,0,1,0,1,0,0,0,0,1,0,0},
                        {0,0,1,0,0,0,0,1,0,1,0,0,0,0,1,0,0},
                        {0,0,1,0,0,0,0,1,0,1,0,0,0,0,1,0,0},
                        {0,0,0,0,1,1,1,0,0,0,1,1,1,0,0,0,0},
                        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                        {0,0,0,0,1,1,1,0,0,0,1,1,1,0,0,0,0},
                        {0,0,1,0,0,0,0,1,0,1,0,0,0,0,1,0,0},
                        {0,0,1,0,0,0,0,1,0,1,0,0,0,0,1,0,0},
                        {0,0,1,0,0,0,0,1,0,1,0,0,0,0,1,0,0},
                        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                        {0,0,0,0,1,1,1,0,0,0,1,1,1,0,0,0,0},
                        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}};
```

Sin embargo para poder accesarlo deberá crear el arreglo:

```
unsigned char** pulsar = new unsigned char*[17];
for(int i=0;i<17;i++){ pulsar[i] = pulsarMat[i]; }
```

3. La codificación de imágenes dentro de una computadora ha evolucionado con el paso de los años: desde imágenes que estaban constituidas por un máximo de 16 colores, a imágenes de 256 colores, a la codificación RGB. La codificación RGB (red-green-blue) es una de la más utilizadas hoy en día y se basa en la representación de los colores de una imagen por medio de 3 bytes que representan la cantidad de rojo, verde y azul presentes en cada pixel (para un total de 16 millones de colores posibles). De esta manera, una imagen codificada en RGB se puede ver como un arreglo multidimensional de ALTO x LARGO x 3, donde cada pixel está compuesto por sus 3 componentes de color.
    a. Cree un método que reciba un arreglo de bytes (unsigned char) de 3 dimensiones, su cantidad de filas y columnas.
    b. Genere un “color” al azar (un valor aleatorio de rojo, verde y azul) que se utilizará para “pintar” la matriz. Recuerde que estos valores son de tamaño 1 byte, por lo que toman valores en el intervalo [0,255].
    c. Para cada pixel de la imagen, “coloree” su contenido con el color inicial usando una degradación lineal de los valores desde el centro hacia los extremos.

i. La degradación lineal se basa en multiplicar el color inicial por un valor en el intervalo [0,1].
ii. El valor a utilizar se calcula con base en la proximidad al centro de la imagen: 1 - (distancia(pixel,centro)*2/diagonal)
iii. La distancia se calcula utilizando distancia euclidiana:

$$d = \sqrt{(x_{centro} - x_{punto})^2 + (y_{centro} - y_{punto})^2}$$

iv. Para calcular la raíz cuadrada incluya la biblioteca “cmath” (o math.h en C) y podrá utilizar el comando sqrt(x).

4. Incluya un código main con el código necesario para probar sus algoritmos, cree los arreglos y haga los llamados a las funciones.

Usar codigo del profesor. Así puedo probar el codigo.

Nota: Para facilitar la verificación del correcto funcionamiento de sus algoritmos, puede descargar el archivo Bitmap.cpp proveído por el profesor. Este documento incluye una pequeña biblioteca para crear archivos de imagen (bitmap .bmp) que le permitirá guardar las imágenes generadas. Guarde el archivo en la misma carpeta que su programa y a su programa inclúyale **#include "Bitmap.cpp"**

Si la variable de su imagen del primer ejercicio se llama “primos” entonces en su main escriba:

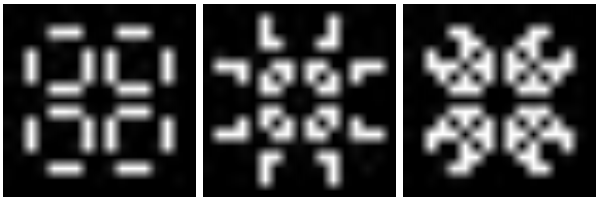
```
Bitmap::save_8bitmap(“primos.bmp”,primos,w,h);
```

Donde **w** es el ancho de la imagen y **h** su alto.

Su resultado obtenido debería ser algo similar a (w=1050 y h=590):

Puede usar un llamado similar para probar el resultado de su algoritmo de Conway.

Las siguiente imágenes representan el estado del sistema antes del llamado al algoritmo, después del primer llamado al algoritmo y después de volver a llamar al algoritmo, respectivamente.



Si la variable de su imagen del tercer ejercicio se llama “imagen” entonces en su main escriba:

```
Bitmap::save_24bitmap("radial.bmp",imagen,w,h);
```

Donde **w** es el ancho de la imagen y **h** su alto.

Su resultado obtenido debería ser algo similar a (con w=2310 y h=1300):