Universidad Zaragoza

Proyecto

Fundamentos de Informática

Grado en Ingeniería de Mecánica

Curso 2017-2018

Universidad de Zaragoza
Escuela de Ingeniería y Arquitectura
Departamento de Informática e Ingeniería de Sistemas
Área de Lenguajes y Sistemas Informáticos

OBJETIVOS

Los objetivos de este proyecto son los siguientes:

- Resolver ejercicios de dificultad similar a los que aparecerán en el examen, y poder afrontarlos y resolverlos con tiempo y calma.
- Introducirse en el manejo de ficheros.

1. Registros, vectores y matrices

En la resolución de diversos problemas de ingeniería se utilizan **matrices cuasivacías** (MCV), en inglés sparse matrices, que son aquellas que contienen un pequeño número de elementos no nulos. Un modo muy simple de describirlas consiste en indicar sus dimensiones y, a continuación, todos los valores no nulos junto con su posición (pueden estar en cualquier orden). Una manera de representar una matriz cuasivacía es mediante un vector de registros, tal y como se muestra a continuación:

matriz	representación	significado
$\begin{pmatrix} 3.6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.2 & 0 \\ 0 & 0 & 0 & 0 & -3.0 \\ 0 & 0 & 6.7 & 0 & 0 \end{pmatrix}$	4 5 0.0 0 0 3.6 1 3 1.2 2 4 -3.0 3 2 6.7	*matriz de 4 filas y 5 columnas* *en fila 1, columna 1, valor 3.6* *en fila 2, columna 4, valor 1.2* *en fila 3, columna 5, valor -3.0* *en fila 4, columna 3, valor 6.7*

Diseñar un programa para trabajar con matrices cuasivacías. Supondremos que la mayor dimensión de una matriz cuasivacía será de 1000 x 1000 y que el máximo número de elementos no nulos de la matriz es el 0.1% de la dimensión de la matriz y que los elementos de la matriz son valores en el intervalo (-10.0, 10.0). El registro se definirá mediante:

```
struct tpCV{

int f; int c; double valor;

};
```

Se pide diseñar los siguientes procedimientos y funciones:

```
1. vector< vector< tpCV > > generarMCV()

// Genera una matriz cuasivacía de manera aleatoria

2. vector< vector<double> > MCVToMatrix( vector< vector< tpCV > > & mcv )

// Construye la matriz representada por la matriz cuasivacía mcv

3. vector< vector< tpCV > > matrixToMCV( vector< vector<double> > & m )

// Construye la matriz representada por la matriz m

4. void analizaMCV( vector< vector< tpCV > > & mcv )

// Analiza la matriz cuasivacia mcv y muestra por pantalla el porcentaje de
elementos no nulos y el numero de filas con todos sus telementos no nulos

5. int main()
```

```
// Programa principal que genera y muestra una matriz cuasivacia, genera y
muestra la matriz representada, regenera y muestra representacion cuasivacia y
finalmente analiza esta ultima representacion.
```

Ejemplo de ejecución:

```
Generando matriz cuasivacía…
4 5 0.0
1 3 1.2
0 0 3.6
3 2 6.7
2 4 -3.0
Matriz representada…
3.6 0.0 0.0 0.0 0.0
0.0 0.0 0.0 1.2 0.0
0.0 0.0 0.0 0.0 -3.0
0.0 0.0 6.7 0.0 0.0
Matriz cuasivacia regenerada…
4 5 0.0
0 0 3.6
1 3 1.2
2 4 -3.0
3 2 6.7
Analisis…
Porcentaje de elementos no nulos: 0.2%
Numero de filas con todos sus elementos no nulos: 4
```

Nombre del fichero: quasiv.cpp

2. Ficheros de texto

En la práctica 3 construiste un programa que calculaba el cifrado César de una frase, dado un desplazamiento. Para descifrar un texto cifrado, es necesario conocer el desplazamiento con el que se codificó para recuperar la letra. No obstante, si no se conoce dicho desplazamiento, se puede deducir. Si el texto está en español se sabe que, estadísticamente, la letra que más se utiliza es la letra "e" (tanto mayúscula como minúscula). Por tanto, se puede asumir que la letra que más aparece en el texto cifrado es la representación de la "e" y con ello deducir el desplazamiento.

Escribe un programa que descifre un fichero de texto (cuyo nombre es introducido por el usuario) utilizando la técnica anteriormente expuesta y lo muestre por pantalla, junto con la letra que más se repite y el desplazamiento usado.

Ejemplo de ejecución:

Contenido del fichero 'texto.txt'

```
Kr sgy hxkbk jkyvoyzk ky kr zktkx ktikxxgjuy ruy sktkyzkxky jkr jkyzotu.
```

Salida del programa

```
Introduce nombre de fichero: texto.txt
La letra que mas aparece es la: k
El desplazamiento estimado es de: 6
texto.txt descifrado es:
El mas breve despiste es el tener encerrados los menesteres del destino.
```

Para poder descifrar el texto, debes haber deducido primero el desplazamiento y, para ello, debes haber buscado la letra que más aparece, y compararla con la "e" para calcular el desplazamiento.

Al representar el nombre del fichero de texto como un string necesitarás hacer uso del método `c_str()` de la calse string cuando invoques el método "abrir". Ejemplo: `in.open( nombre.cstr() );`

Nombre del fichero: descifrado.cpp

3. Ficheros binarios

Una **imagen digital** no es más que una matriz bidimensional de elementos llamados píxeles, cuyos valores numéricos indican el color de la correspondiente región de la foto. En el caso de una imagen en **escala de grises**, estos valores numéricos van desde 0 (que indica negro) hasta 255 (que indica blanco). Los valores intermedios representan, por tanto, distintas gradaciones de gris. Es muy habitual que una imagen digital vaya acompañada de datos sobre la propia imagen, datos que se conocen como *metadatos*. Ejemplos de metadatos de una imagen digital son: fecha en que se tomó la imagen, tipo de cámara, identificador de la imagen, coordenadas GPS donde se tomó, resolución y tamaño, los parámetros con los que se realizó la imagen, etc.

Las imágenes digitales se pueden almacenar en diferentes tipos de ficheros binarios. En este caso utilizaremos el formato bmp de 24 bits, que es un formato de fichero binario con la siguiente estructura: los primeros 54 bytes son diferentes cabeceras donde, entre otros datos, se almacena la anchura y altura de la imagen que ocupan la posicion 18 y 22 empezando a contar desde 0; los siguientes bytes hasta el final del fichero son la información de la imagen, tres bytes por cada píxel que representan las componentes del color en formato RGB.

Por ejemplo, si disponemos de una imagen de tamaño 40x40 píxeles, el fichero bmp de 24 bits tendría 54 bytes de cabeceras más 40x40x3=4800 bytes de información de la imagen, haciendo un total de 4854 bytes. En este proyecto vamos a trabajar con imágenes en formato bmp de 24 bits tales que su altura y anchura sea un múltiplo de ocho menor de 127. Estas restricciones hacen que el problema sea más sencillo de resolver.

El color en formato RGB se puede representar como una estructura con tres valores enteros que representan la intensidad de cada canal de color:

```
struct tpColor{
   int r, g, b;
};
```

Existen numerosas técnicas de tratamiento de imagen digital que se basan en alterar los datos de una imagen. Así, por ejemplo, el **filtro de media** difumina la imagen. Para el cálculo de un filtro de media, se sustituye el valor de cada pixel por la media de los valores de dicho píxel y sus vecinos (píxeles de arriba, abajo, izquierda, derecha, y los ubicados contiguamente en las diagonales). Es importante resaltar que un píxel puede tener 8, 5 ó 3 vecinos, dependiendo de su posición en la imagen.

Se pide diseñar los siguientes procedimientos y funciones:

1. `vector< vector<tpColor> > cargarImagen( string & nombre )`

```
// Carga una imagen en formato bmp de 24 bits desde el fichero con nombre pasado como parametro y la guarda en la estructura de datos tpImagen. Para que los valores almacenados en la imagen se encuentren entre 0 y 255 es necesario asignar el dato leido del fichero de la siguiente manera: imagen[i][j].r = (int)((unsigned char)leido); // Entre 0 y 255
```

2. `void guardarImagen( string & nombre, vector< vector<tpColor> > & imagen )`

```
// Guarda una imagen en formato bmp de 24 bits en el fichero con nombre pasado como parametro desde la estructura de datos de imagen
```

3. `void filtroMedia(vector< vector<tpColor> > & original, vector< vector<tpColor> > & filtrada)`

```
// Aplica un filtro de media a la imagen original pasada como parametro y guarda el resultado en la imagen filtrada.
```

```
4. int main()

// Programa que pide el nombre de la imagen a filtrar y el nombre de la imagen filtrada, carga la imagen a filtrar en memoria, aplica el filtro de media y guarda el resultado con el nombre de la imagen filtrada.
```

Nota: para probar tus programas se recomienda utilizar las imágenes proporcionadas.

Ejemplo de interacción del programa principal:

```
Fichero de origen:  imagen.bmp
Fichero de destino: filtrada.bmp
Filtrando imagen…
Imagen filtrada con éxito.
```

Al representar el nombre del fichero de texto como un string necesitarás hacer uso del método `c_str()` de la calse string cuando invoques el método "abrir". Ejemplo: `in.open( nombre.c_str() );`

Nombre del fichero: filtroMedia.cpp

PARA ENTREGAR

Se entregará un fichero .zip que contiene el código fuente del trabajo realizado para el proyecto. Para ello se deberá enviar el fichero utilizando el link correspondiente al Proyecto de la asignatura en Moodle. El fichero .zip contendrá los ficheros quasiv.cpp, descifrado.cpp, y fitroMedia.cpp correspondiente a cada uno de los problemas propuestos. Para la evaluación del proyecto se utilizarán los siguientes criterios de evaluación:

- Un código fuente que no compile será evaluado con un 0
- Se valorará la utilización de comentarios que aclaren el método de solución utilizado en las partes mas conflictivas del código
- Se valorará especialmente el diseño y la utilización de procedimientos y funciones de una manera coherente y cohesiva, indicadores de un buen diseño de programas. Se pueden utilizar estrategias de refactorización durante el desarrollo del proyecto.
- Se valorará especialmente la legibilidad del código fuente entregado
- Se valorará la ausencia de errores semánticos

La fecha límite para realizar la entrega es el **6 de junio de 2018 a las 23:59 horas**, límite de entrega de las actividades de evaluación continua.