Prácticas Resumen de la Asignatura

METODOLOGÍA Y TECNOLOGÍA DE LA PROGRAMACIÓN(curso 2006-2007).

Módulo II (2º Cuatrimestre)

Marzo de 2007.

1º Ingeniero Técnico en Informática de Gestión.
1º Ingeniero Técnico en Informática de Sistemas.

Práctica RESUMEN I

Enunciado

Realizar un programa que implemente una serie de funciones para ocultar y leer mensajes ocultos en imágenes.

Imagen digital

Una imagen digital es una matriz bidimensional de puntos (pixels) en la que cada uno tiene asociado un nivel de luminosidad cuyos valores están en el conjunto {0, 1, ..., 255} de forma que el 0 indica la menor luminosidad (negro) y el 255 la mayor luminosidad (blanco). Los restantes valores indican niveles intermedios de luminosidad (grises), siendo más oscuros cuanto menor sea su valor. Con esta representación cada pixel requiere únicamente un byte (*unsigned char*).
Cada casilla de esta matriz representa a un punto de la imagen y el valor guardado en ésta indica

- En imágenes de niveles de gris: su nivel de luminosidad.
- En imágenes en color: su código de color (representación por tabla de color) o la representación del color en el esquema usado (RGB, IHV, etc).

Nuestro interés se centra únicamente en imágenes de niveles de gris, por lo que cada casilla contiene niveles de luminosidad que se representan con valores del conjunto {0, 1, ..., 255} con la convención explicada anteriormente.
Las imágenes se almacenan en ficheros con un determinado formato. Nosotros vamos a trabajar con imágenes en formato PGM.

Imagen de tamaño *6 filas x 6 columnas*

255	255	255	255	255	255
255	0	125	125	125	255
255	125	0	125	125	255
255	125	125	0	125	255
255	125	125	125	0	255
255	255	255	255	255	255

Descripción del formato PGM

El formato PGM constituye un ejemplo de los llamados formatos con cabecera. Un fichero PGM tiene, desde el punto de vista del programador, un **formato mixto texto-binario**: la cabecera se almacena en formato texto y la imagen en sí en formato binario.

Con más detalle, la descripción del formato PGM es:

1. Cabecera. La cabecera está en formato texto y consta de:
    - Un "número mágico" para identificar el tipo de fichero. Un fichero PGM que contiene una imagen de niveles de gris (nuestro caso) tiene asignado como identificador los caracteres P5.
    - Un número indeterminado de comentarios (marcados delante con #).
    - Número de columnas (*c*) y número de filas (*f*), separados por un número indeterminado de espacios.
    - Valor del mayor nivel de gris que puede tener la imagen (*m*).

   Cada una de estas informaciones está terminada, por un salto de línea.
2. Contenido de la imagen

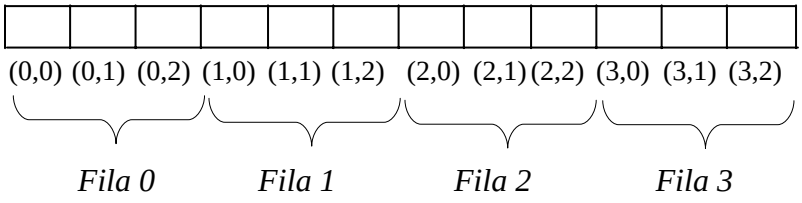
   Una secuencia binaria de *f x c* bytes (*unsigned char*), con valores entre 0 y *m*. Cada uno de estos valores representa un nivel de gris de un pixel. La imagen se almacena por filas, es decir, el primer pixel es la esquina superior izquierda, el segundo el que está a su derecha, etc.

Algunas aclaraciones respecto a este formato

- El número mágico, el número de filas, columnas y mayor nivel de gris se especifica en modo texto, esto es, cada dígito viene en forma de carácter.
- Los comentarios son de línea completa y están precedidos por el carácter #. La longitud máxima es de 70 caracteres.
- Aunque el mayor nivel de gris sea *m*, no tiene porqué haber algún pixel con este valor.

En la imagen siguiente se muestra, a modo de ejemplo, como se almacena una imagen de 4 filas y 3 columnas en este formato:

```
P5
#ejemplo
3 4
255
```



Esteganografía

La esteganografía es la rama de la criptología que trata sobre la ocultación de mensajes, para evitar que se perciba la existencia del mismo.

La esteganografía se puede aplicar a imágenes mediante diferentes técnicas, una de ellas consiste en la modificación del bit de menor peso (LSB: *less significant bit*) de algunos pixels de la imagen. Al tratarse del bit de menor peso de un píxel, éste se ve sometido a un cambio imperceptible de color.

Descripción del programa principal

El programa principal realizará las siguientes operaciones:

1. Pedir el tipo de operación: codificar o decodificar.
2. Para decodificar:
    a. Pedir nombre del fichero con la imagen.
    b. Pedir nombre del fichero donde se guardará el mensaje.
    c. Leer la imagen con el mensaje en formato *unsigned char*.
    d. Obtener la matriz LSB con los bits menos significativos de cada píxel.
    e. Obtener el mensaje codificado en la matriz LSB. Cada carácter del mensaje está codificado con 1 byte (8 bits). El mensaje comienza en el primer píxel de la imagen y termina con una secuencia especial de tres caracteres '***'.
    f. Escribir el mensaje en un fichero de texto.
3. Para codificar:
    a. Pedir nombre del fichero con la imagen original.
    b. Pedir nombre del fichero con la imagen que contendrá el mensaje.
    c. Pedir nombre del fichero con el mensaje original.
    d. Leer la imagen original en formato *unsigned char*.
    e. Leer el mensaje y almacenarlo en una cadena de caracteres. Al mensaje del fichero habrá que añadir la secuencia "***" de fin de mensaje.
    f. Comprobar que el mensaje cabe dentro de la imagen. Cada carácter del mensaje está codificado con 1 byte (8 bits). Habrá que tener en cuenta, además, la secuencia '***' de fin de mensaje.
    g. Pasar el mensaje a binario.
    h. Esconder el mensaje en los bits menos significativos de la imagen original.
    i. Escribir la nueva imagen en un fichero.

Para ello, el programa deberá implementar las siguientes funciones:

Cargar una imagen desde un fichero

Dado un fichero que contiene una imagen digital en el formato PGM anteriormente descrito, la función deberá leer la cabecera del fichero, para obtener las características de la imagen (número de columnas, número de filas y niveles de gris) y, con estos datos, crear la matriz en la que se almacenará la imagen, para a continuación leer la imagen y guardarla en la matriz creada.
Observaciones:

- El fichero se abrirá en modo "*rb*".
- La lectura de la cabecera se realizará con funciones de lectura de ficheros de texto.
- La lectura de la matriz de píxeles se realizará con funciones de lectura de ficheros binarios.

Guardar una imagen *unsigned char* en un fichero

Esta función deberá crear un fichero que contenga la imagen almacenada en una matriz de *unsigned char*. El fichero deberá tener formato PGM. Para ello, en primer lugar habrá que grabar la cabecera del fichero en formato texto, de acuerdo a la especificación dada, y, a continuación, la imagen en formato binario. El fichero se abrirá en modo "*wb*".

Obtener una matriz con los bits menos significativos

Dada una imagen, *I*, en formato *unsigned char*, se creará una matriz de las mismas dimensiones de la imagen, *LSB*, en la que se almacenará el bit menos significativo de cada pixel de la imagen original.
Para esto, habrá que implementar una función que, dado un número *unsigned char*, devuelva el bit menos significativo.

Imagen

	0	1	2	3	4	5	6
0	115	120	125	125	125	125	125
1	115	120	125	125	125	125	130
2	115	120	125	130	130	130	130
3	115	120	125	130	130	130	130
4	115	120	125	130	130	130	130
5	115	120	125	130	130	130	130

Matriz LSB

	0	1	2	3	4	5	6
0	1	0	1	1	1	1	1
1	1	0	1	1	1	1	0
2	1	0	1	0	0	0	0
3	1	0	1	0	0	0	0
4	1	0	1	0	0	0	0
5	1	0	1	0	0	0	0

Convertir 8 bits en un *unsigned char* (paso de binario a decimal)

Dado un vector de 8 elementos, cuyos valores podrán ser 0 o 1, la función transformará el vector en un *unsigned char* correspondiente a su equivalente en decimal.

1	1	1	0	1	0	1	0	=> 234

Escribir mensaje en un fichero de texto

El mensaje decodificado se escribirá en un fichero de texto.

Leer mensaje de un fichero de texto

El mensaje a codificar estará escrito en un fichero de texto.

Transformar un *unsigned char* en binario (paso de decimal a binario)

Dado un número en formato *unsigned char*, correspondiente a un carácter del mensaje, se transformará en un vector de 0 y 1 correspondiente a su representación en binario con 8 bits.

234 =>

1	1	1	0	1	0	1	0

Modificar el bit menos significativo de un *unsigned char*

Dado un número en formato *unsigned char*, correspondiente a un pixel de la imagen original, se modificará su bit menos significativo con un valor que será 0 o 1.

Ejemplo de codificación

Imagen original:

Imagen

	0	1	2	3	4	5	6
0	115	120	125	125	125	125	125
1	115	120	125	125	125	125	130
2	115	120	125	130	130	130	130
3	115	120	125	130	130	130	130
4	115	120	125	130	130	130	130
5	115	120	125	130	130	130	130

Mensaje a codificar: "*so****"

1. Comprobar si el mensaje cabe dentro de la imagen:
   - Tamaño mínimo de la imagen: 5 caracteres * 8 píxeles/ carácter = 40 píxeles.
   - Tamaño de la imagen actual: 7x6=42 píxeles.
2. Obtener el mensaje en decimal:
   - 115 111 42 42 42.
3. Obtener el mensaje en binario binario:
   - 01110011 01101111 00101010 00101010 00101010
4. Esconder el mensaje en la imagen:

Imagen con mensaje

	0	1	2	3	4	5	6
0	114	121	125	125	124	124	125
1	115	120	125	125	124	125	131
2	115	121	124	130	131	130	131
3	114	121	124	130	130	131	130
4	115	120	125	130	130	130	131
5	114	121	124	131	130	130	130

Ejemplo de decodificación

Imagen con mensaje

	0	1	2	3	4	5	6
0	114	121	125	125	124	124	125
1	115	120	125	125	124	125	131
2	115	121	124	130	131	130	131
3	114	121	124	130	130	131	130
4	115	120	125	130	130	130	131
5	114	121	124	131	130	130	130

1. Obtener los bits menos significativos.

Matriz LSB

	0	1	2	3	4	5	6
0	0	1	1	1	0	0	1
1	1	0	1	1	0	1	1
2	1	1	0	0	1	0	1
3	0	1	0	0	0	1	0
4	1	0	1	0	0	0	1
5	0	1	0	1	0	0	0

2. Extraer el mensaje:
   - 01110011 01101111 00101010 00101010 00101010
3. Decodificar el mensaje:
   - "so***"

Requisitos y Recomendaciones

- El programa se estructurará en 5 ficheros:
  - *resumen.c*: en el que se encuentra sólo el main.
  - *imagen.c*: para las funciones relacionadas con la lectura/escritura de imágenes en formato PGM.
  - *imagen.h*: fichero de cabecera correspondiente a *imagen.c*.
  - *esteganografia*.c: para las funciones relacionadas con esconder y leer mensajes.
  - *esteganografía.h*: fichero de cabecera correspondiente a *esteganografía.c*.
- Para simplificar la cabecera de los ficheros PGM, los ejemplos que se dejan para probar el programa (candy.pgm, simpson.pgm y camera.pgm), no contienen ningún comentario (no es necesario comprobar si existen y leerlos). No obstante, se debe tener en cuenta que si se utilizan otros ficheros de imágenes, éstas pueden presentar comentarios en la cabecera.
- Las dimensiones de las imágenes no se conocen en tiempo de compilación, por lo que las matrices que las albergarán deberán reservarse en tiempo de ejecución una vez se conozcan el número de filas y columnas de la misma (esto ocurrirá al leer la cabecera del fichero).
- Para visualizar las imágenes en lucano se utilizará el programa GIMP. Su sintaxis es:

gimp <fichero con la imagen>

Semanas de asistencia: 2 (semana del 23 al 27 de Abril y semana del 30 al 4 de Mayo).