PRÁCTICA 2

Juan Carlos Hermoso Quesada
41512475-M
Jose Luis Pedraza Roman
80108849-X
Metodología de Programación
2016/2017
Universidad de Granada

1. DEFINICIÓN DEL PROBLEMA

En esta práctica simularemos que tenemos un bloque con 8 leds y que mediante una serie de funciones iremos manipulando su estado o, más concretamente, el de los los leds.

Cada led sólo puede tener dos estados, encendido o apagado (true o false, respectivamente).

Un led representa un bit, por lo cual ocho leds representarán un byte y en C++ las variables de tipo unsigned char ocupan exactamente un byte. Por tanto, para esta simulación declararemos un alias para esta variable llamado bloqueLed y así ir declarando variables de este tipo:

```
typedef unsigned char bloqueLed
```

2. FUNCIONES

Se nos ha facilitado previamente un archivo llamado bloqueLed.h en el cual se encontraban las funciones necesarias para realizar el programa y en el cual venía una breve explicación de lo que hacía cada función en particular. A continuación, una lista de todas ellas:

- **void on(bloqueLed &b, int pos):** mediante una variable b de tipo bloqueLed y una posición del led (de 0 a 7) moveremos tantas veces como la cifra de la posición indique un 1 en 0x1 (0000 0001). Dicha serie binaria con el 1 movido de sitio (o no) será el contenido de otra variable tipo bloqueLed, llamada mask (máscara). Una vez creada la máscara se realizará la operación lógica or con la variable b. Al hacer esto, la posición a cambiar se quedará a 1 y, por tanto, el led se encenderá.

- **void off(bloqueLed &b, int pos):** es el proceso contrario al anterior: consiste en apagar un led. El proceso es el mismo sólo que, cuando ya hemos creado la máscara, la tenemos que negar mediante el símbolo ~. De esta forma, tendremos un 0 en vez de un 1 y al realizar la operación lógica and tendremos un 0 en la posición a cambiar y, por tanto, el led se apagará.

- **bool get(bloqueLed b, int pos):** esta función consiste en, dada una posición, saber cual es el estado de esa posición o, lo que es lo mismo, saber si ese led esta encendido o no. Para ello, crearemos otra máscara dentro de la función y una varia ble booleana llamada estado e inicializada a false. Realizaremos otra vez la operación lógica or. En el caso de que b resulte mayor que 0 significará que el led estará encendido, por lo cual asignaremos a estado el valor de true. Finalmente, devolveremos el estado.

- **string bloqueLedToString(bloqueLed b):** función usada para imprimir el estado del bloque al completo. Para ello, declararemos una variable string llamada bloque

y una variable booleana llamada estado e inicializada a false. Una vez hecho esto, se realiza un bucle for desde 0 hasta 7 y pasaremos cada iteración de éste y la variable b a una llamada al método get(). El estado se almacenará en la variable estado del método que estamos explicando. Si sale true, sumamos 1 al bloque. En caso contrario, sumamos un 0.

- **void encender(bloqueLed &b):** este método sirve para encender todos los leds. En este caso, simplemente hay que crear una máscara y pasarle el valor de ocho 1s. Para esto tenemos que igualar la máscara a 0xFF y realizar una operación or con la variable b.

- **void apagar(bloqueLed &b):** este método sirve para apagar todos los leds. Para esto, creamos otra máscara y la rellenamos con ocho valores a 0 igualándola pues a 0x00 y realizando una operación and.

- **void asignar(bloqueLed &b, const bool v[]):** antes de explicar este método queríamos aclarar previamente un problema que hemos tenido. Para realizar este método, necesitamos pasar por parámetro un vector de booleanos inicializados previamente de la siguiente forma: 10100000. Nuestra interpretación fue que los bloques iban de izquierda a derecha ascendentemente, es decir, empezando por la posición 0 a la izquierda y acabando con la siete a la derecha. Nos dimos cuenta de nuestra equivocación al imprimir los resultados. El problema es que desconocemos como podemos hacer para que surjan los resultados como se nos indica, por lo cual, hemos cambiado en el main.cpp la organización del vector, por lo cual lo hemos colocado así: 00000101. Esperemos que no suponga problema alguno, ya que de esta forma nuestro programa funciona a la perfección.

  Este método consiste en, una vez pasados por parámetro la variable b y un vector de booleanos (inicializado previamente), realizar un bucle for de 0 a 7. En el caso de que la posición del vector correspondiente a una iteración del bucle dé true; se encenderá el led correspondiente a dicha posición en el bloque. En caso contrario, se apagará.

- **void encendidos(bloqueLed b, int posic[], int &cuantos):** este método nos indica cuáles de los leds están encendidos. Para ello, pasamos por parámetro la variable b, un vector de enteros inicializado en el main.cpp a 8 y una variable entera llamada cuantos (por referencia) y declaramos una variable booleana estado, una variable entera llamada j e inicializada a 0 e inicializamos la variable cuantos a 0.

  Una vez hecho esto, hacemos un bucle for de 0 a 7 y pasamos cada iteración por parámetro al método get();. Cada resultado se almacenará en la variable estado. Si da true, se almacenará el número de la iteración en la posición igual al valor j del vector de posiciones (originalmente inicializada a 0). Entonces, se aumentará en uno el valor de j, para que, si se repite el mismo caso, se colocará la siguiente iteración en la siguiente posición del vector de posiciones. También se aumentará en uno la variable cuantos, así cuando en el main.cpp tengamos que imprimir las posiciones correspondientes a los leds encendidos imprima exactamente sus posiciones.

- **void print(bloqueLed b):** Hemos prescindido de este método.
- **void volcar(bloqueLed b, bool v[]):** Hemos prescindido de este método.

3. RESULTADO

A continuación, una captura que describe gráficamente el resultado de cada método.