ALGORITMOS Y ESTRUCTURAS DE DATOS

Proyecto Virtual LCD

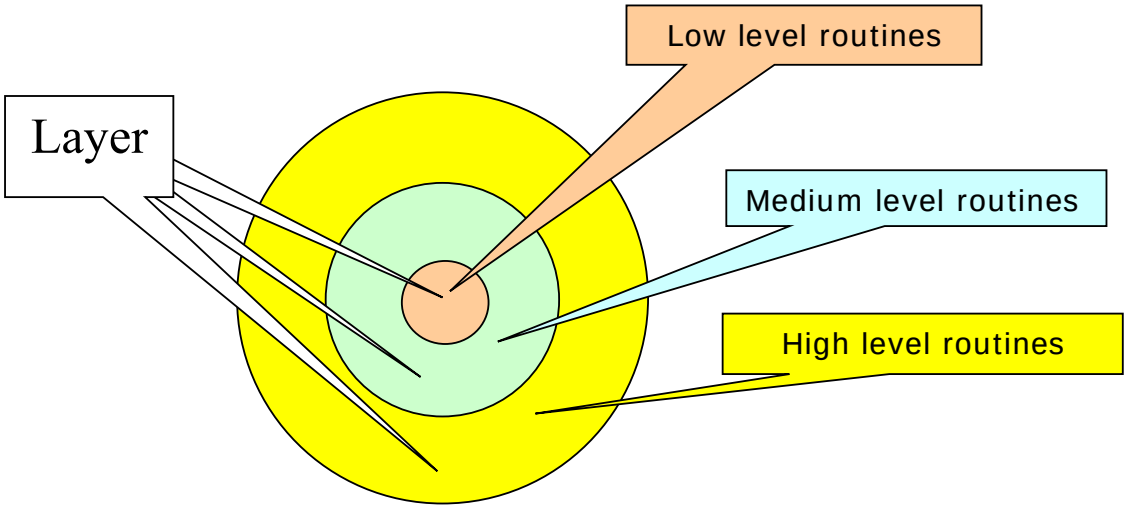
Introducción

El presente trabajo práctico le propone al alumno trabajar con un display de cuarzo líquido (LCD) alfanumérico de 2x16 caracteres.

El alumno deberá programar una LCD virtual utilizando Allegro.

Niveles o capas de abstracción

Cuando se programa hardware siempre se debe incrementar el nivel de abstracción desde los niveles más bajos hasta los más altos. Los niveles más bajos son también conocidos como funciones o rutinas de bajo nivel o drivers. Estas funciones entran en contacto con el hardware en forma directa y serán invocadas por funciones de mayor abstracción que forman una capa por encima de ellas. Así se puede continuar hasta llegar a la capa más alta del sistema (usualmente la que está en contacto con el usuario)



El número de capas dependerá del problema en particular. Que es lo que separa una capa de la otra está dado por el nivel de abstracción. Este nivel de abstracción esta en directa relación con los datos que maneja cada capa. Cada capa maneja un "lenguaje" (abstracciones) y se debe evitar que haya solapamiento cuando se definen estas capas.

Lo que siempre se trata de hacer es crear una capa alrededor de los drivers de manera de independizarse del mismo esto es lo que se conoce como Hardware Abstraction Layer o HAL. De esta manera diferentes aplicaciones pueden usar un hardware sin que realmente sepan como es.

Cuando instalamos un modem o una impresora lo único que cambia es el driver.

Take home

El objetivo del trabajo es implementar una clase que nos permita manipular un LCD Virtual.

Abajo se incluye el prototipo de un clase abstracta a implementar junto con una descripción de cada una de las funciones que la componen.

Deberán implementar la clase basicLCD como una clase abstracta (todas sus funciones miembro públicas virtuales e igualadas a 0, a excepción del constructor) respetando absolutamente el prototipo de abajo. No está permitido incluir nuevas funciones miembro públicas a la clase ni eliminar funciones o variables miembros del prototipo especificado abajo.

Además deberán crear una clase concreta a partir de basicLCD que contenga todas las implementaciones de las funciones descriptas en basicLCD para un LCD virtual que deberán crear ustedes a visualizar en Allegro.

Podrán agregar funciones y variables miembros privadas para “customizar” la clase concreta (la clase abstracta no deber tener miembros privados ya que no tiene sentido).

Se deberá realizar también un programa de prueba (main) que permita probar una instancia (objeto) de la clase que ustedes armen heredada de basicLCD.

Prototipos

```
struct cursorPosition
{
	int row;
	int column;
};


class lcdError
{
	public:

	string getErrorName();
	string getErrorDescription();
	unsigned long getErrorCode();
};


class basicLCD
{
  public:

/*=====================================================
* Name: basicLCD
* Entra: -
* Resulta: Constructor de la clase. Inicializa el LCD y deja
*          todo listo comenzar a utilizarlo.
*
* cadd =1 (cursor address) (ver NOTA 1)
*=====================================================*/
  basicLCD();

/*=====================================================
* Name: ~basicLCD
* Entra: -
```

```
* Resulta: Destructor de la clase. Libera cualquier recurso
*          que se hubiera tomado de forma de evitar
*          "resources leak".
*=========================================================*/
  ~basicLCD();

/*=========================================================
* Name: lcdInitOk
* Entra: -
* Resulta: No genera ningún cambio en el display.
* Devuelve en su nombre “true” si el display se inicializó
* correctamente (el constructor no tuvo errores) o “false
* en caso contrario.
*=========================================================*/
virtual bool lcdInitOk() = 0;

/*=========================================================
* Name: lcdGetError
* Entra: -
* Resulta: No genera ningún cambio en el display.
* Devuelve en su nombre un lcdError&
*=========================================================*/
virtual FT_STATUS lcdGetError() = 0;

/*=========================================================
* Name: lcdClear
* Entra: -
* Resulta: Borra el display y el cursor va a HOME
*
* Devuelve en su nombre “true” si fue satisfactoria “false”
* en caso contrario.
*=========================================================*/
virtual bool lcdClear() = 0;

/*=========================================================
* Name: lcdClearToEOL
* Entra: -
* Resulta: Borra el display desde la posición actual
* del cursor hasta el final de la línea.
*
* Devuelve en su nombre “true” si fue satisfactoria “false”
* en caso contrario.
*=========================================================*/
virtual bool lcdClearToEOL()= 0;

/*=========================================================
* Name: operator<<()
* Entra: Un carácter
* Resulta: Pone el carácter en la posición actual
* del cursor del display y avanza el cursor a la próxima
* posición respetando el gap (si el carácter no es imprimible
* lo ignora)
*
* Devuelve en su nombre una referencia a un basicLCD que permite
* encascar la función:
*     basicLCD lcd;
*     lcd << ‘a’ << ‘b’ << ‘c’;
*=========================================================*/
virtual basicLCD& operator<<(const unsigned char c) = 0;
```

```
/*=====================================================
* Name: operator<<()
* Entra: Una cadena de caracteres NULL terminated
* Resulta: imprime la cadena de caracteres en la posición actual
* del cursor y avanza el cursor al final de la cadena respetando
* el gap (si algún carácter no es imprimible lo ignora). Si recibe una
* cadena de más de 32 caracteres, muestra los últimos 32 en el display.
*
* Devuelve en su nombre una referencia a un basicLCD que permite
* encascar la función:
*     basicLCD lcd;
*     lcd << "Hola" << " " << "Mundo";
*=====================================================*/
virtual basicLCD& operator<<(const unsigned char * c) = 0;

/*=====================================================
* Name: lcdMoveCursorUp
*
* Entra: -
* Resulta: Pasa el cursor a la primera línea del display sin
* alterar la columna en la que estaba.
*
* Devuelve en su nombre "true" si fue satisfactoria "false"
* en caso contrario.
*=====================================================*/
virtual bool lcdMoveCursorUp() = 0;

/*=====================================================
* Name: lcdMoveCursorDown
*
* Entra: -
* Resulta: Pasa el cursor a la segunda línea del display sin
* alterar la columna en la que estaba.
*
* Devuelve en su nombre "true" si fue satisfactoria "false"
* en caso contrario.
*=====================================================*/
virtual bool lcdMoveCursorDown() = 0;

/*=====================================================
* Name: lcdMoveCursorRight
*
* Entra: -
* Resulta: Avanza el cursor una posición
*
* Devuelve en su nombre "true" si fue satisfactoria "false"
* en caso contrario.
*=====================================================*/
virutal bool lcdMoveCursorRight() = 0;

/*=====================================================
* Name: lcdMoveCursorLeft
*
* Entra: -
* Resulta: Retrocede el cursor una posición
*
* Devuelve en su nombre "true" si fue satisfactoria "false"
* en caso contrario.
*=====================================================*/
virtual bool lcdMoveCursorLeft() = 0;
```

```
/*=====================================================
* Name: lcdSetCursorPosition
* Entra: Recibe una estructura tipo cursorPosition
* Resulta: Posiciona el cursor en la posición dada
* por row y column. row[0-1] col[0-19]. Ante un valor inválido
* de row y/o column ignora la instrucción (no hace nada).
*
* Devuelve en su nombre “true” si fue satisfactoria “false”
* en caso contrario.
*=====================================================*/
virtual bool lcdSetCursorPosition(const cursorPosition pos) = 0;

/*=====================================================
* Name: lcdGetCursorPosition
* Entra: -
* Resulta: Devuelve la posición actual del cursor.
*
*
* Devuelve una estructura tipo cursorPosition
*=====================================================*/
virtual cursorPosition lcdGetCursorPosition() = 0;
```

Bibliografía

C++

1. Effective C++ Third Edition 55 Specific Ways to Improve Your Programs and Designs. Scott Meyers. May 12, 2005. ISBN 0-321-33487-6.
2. C++: The Complete Reference, 4th Edition. Herbert Schildt. Nov 19, 2002. ISBN 0-072-22680-3.
3. www.cplusplus.com