#ifndef \_LIBUTILS\_H\_

#define \_LIBUTILS\_H\_

#include "common.h"

#include "menu.h"

#include "kernel.cuh"

#include <stdarg.h>

#include <stdbool.h>

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <unistd.h>

/\*\*

\* Mensaje de ayuda para mostrar el funcionamiento del programa.

\*/

#define MSG\_AYUDA "\n\

Práctica de Ampliación de Programación Avanzada.\n\

Daniel Estangüi y Miguel García\n\

Uso correcto:\n\

./candy [-hman:f:c:v]\n\

-h\n\

Muestra este mensaje de ayuda y sale del programa\n\

-a | -m\n\

Habilita la ejecución automática (-a) o manual (-m). Si no se\

especifica, se habilita la ejecución automática por defecto. Estas opciones son\

excluyentes\n\

-n <nivel>\n\

Si se especifica, establece el nivel de inicio (del 1 al 3)\n\

-f <nº\_filas>\n\

Establece el número de filas de la matriz de juego\n\

-c <nº\_columnas>\n\

Establece el número de columnas de la matriz de juego\n\

-v\n\

Incrementa el nivel de detalle\n\

"

/\*\*

\* Colores

\*/

#define NOR "\x1B[0m" //Color normal

#define ROJ "\x1B[31m"

#define VER "\x1B[32m"

#define AMA "\x1B[33m"

#define AZU "\x1B[34m"

#define ROS "\x1B[35m"

#define CYN "\x1B[36m"

#define BLA "\x1B[30m"

#define RST "\x1B[0m" //RESET

/\*\*

\* Niveles de detalle

\*/

#define DETALLE\_LOG 0

#define DETALLE\_DEBUG 1

#define DETALLE\_EXTRA 2

/\* ------------------------ \*/

/\* DECLARACIÓN DE FUNCIONES \*/

/\* ------------------------ \*/

/\*\*

\* Procesa los argumentos pasados por línea de comandos

\*

\* @param

\* Número de argumentos pasados (nº de elementos en argv).

\*

\* @param argv

\* Array de cadenas con los argumentos.

\*

\*

\* @return

\* -> ERR\_ARGS si se ha especificado alguna opción no reconocida.

\* -> SUCCESS si se han procesado los argumentos correctamente y se

\* debe proseguir con la ejecución.

\* -> SUCC\_ARGS si se ha procesado un argumento y se debe terminar la

\* ejecución (p.ej.: tras procesar '-h').

\*/

int procesar\_args (int argc, char \*argv []);

/\*\*

\* Imprime toda la información de las variables globales del juego.

\*/

void imprimir\_info ();

/\*\*

\* Cambia el valor de los parámetros del juego.

\*

\* @param nuevos\_params

\* Estructura de tipo Malla (definida en 'common.h') con los nuevos nivel

\* y dimensiones del tablero de juego.

\*/

void cambiar\_params (Malla malla);

/\*\*

\* Devuelve una estructura Malla con los valores especificados (nivel y dimensiones),

\* pero sin ninguna memoria reservada para la matriz.

\*

\*

\* @return

\* Una nueva instancia de tipo Malla, con los valores especificados por

\* línea de comandos.

\*/

Malla ver\_params ();

/\*\*

\* Permite guardar la malla en el fichero especificado.

\*

\* @param malla

\* Estructura con toda la información del juego actual (nivel, dimensiones

\* de la matriz y el contenido de la matriz).

\*

\* @param nombre\_fichero

\* Nombre del fichero en el que se deben guardar los datos. Si ya existe se

\* sobrescribirá; si no, se creará.

\*

\*

\* @return

\* SUCCESS si los datos se han guardado correctamente.

\* ERR\_ARCHIVO si no se pudo abrir o cerrar correctamente el archivo.

\*/

int guardar (Malla malla, const char \*nombre\_fichero);

/\*\*

\* Carga desde el fichero especificado el juego guardado.

\*

\* @param malla

\* Estructura en la que se va a cargar la información del juego.

\*

\* @param nombre\_fichero

\* Nombre del fichero que contiene la información del juego.

\*

\*

\* @return

\* SUCCESS si el archivo se cargó correctamente.

\* ERR\_ARCHIVO si hubo algún error al abrir o cerrar el fichero.

\*/

int cargar (Malla \*malla, const char \*nombre\_fichero);

/\*\*

\* Reserva la memoria necesaria para el tablero de juego

\*

\* @param malla

\* Estructura de tipo Malla (definida en 'common.h') con las dimensiones de

\* la matriz y su contenido.

\*

\*

\* @return

\* SUCCESS si todo ha salido correctamente.

\* ERR\_MEM si hubo algún error al intentar reservar la memoria.

\*/

int reservar\_mem (Malla \*malla);

/\*\*

\* Rellena la matriz de juego con diamantes aleatorios.

\*

\* @param malla

\* Estructura de tipo Malla (definida en 'common.h') con las dimensiones de

\* la matriz y su contenido.

\*

\*

\* @return

\* SUCCESS si todo ha salido correctamente.

\* ERR\_CUDA si alguna función CUDA ha fallado.

\*/

int rellenar (Malla \*malla);

/\*\*

\* Imprime por pantalla el contenido de la matriz.

\*

\* @param malla

\* Estructura de tipo Malla (definida en 'common.h') con las dimensiones de

\* la matriz y su contenido.

\*/

void mostrar\_malla (Malla malla);

/\*\*

\* Crea un diamante del tipo especificado.

\*

\* @param num

\* Número del tipo de diamante a crear.

\*

\* @return

\* Una nueva instancia de diamante.

\*/

Diamante crear\_diamante (int num);

/\*\*

\* Imprime por pantalla la cadena pasada como argumento sólo si el nivel de detalle es

\* el especificado (o mayor).

\*

\* @param detalle

\* Nivel de detalle mínimo para imprimir el mensaje

\*

\* @param cadena

\* Cadena con formato para imprimir

\*

\* @param ...

\* Argumentos para el formato de la cadena

\*/

void imprimir (int detalle, const char \*cadena, ...);

/\*\*

\* Obtiene el máximo diamante a generar, según el nivel especificado en la malla

\*

\* @param Malla

\* Estructura con la información del nivel actual.

\*

\*

\* @return

\* El valor máximo del diamante que se puede generar, en función del nivel.

\*/

int max\_nv (Malla malla);

/\*\*

\* Obtiene el nivel de detalle actual.

\*/

int ver\_nv\_detalle ();

#endif