

Juego Zanahoria

0.1

Generado por Doxygen 1.8.0

Domingo, 6 de Mayo de 2012 01:14:19

Índice general

1	Índice de módulos	1
1.1	Módulos	1
2	Índice de estructura de datos	3
2.1	Estructura de datos	3
3	Índice de archivos	5
3.1	Lista de archivos	5
4	Documentación de módulos	7
4.1	Codigos Funciones de Colores.h	7
4.1.1	Descripción detallada	7
4.2	Cadenas De Color y Estilo	8
4.2.1	Descripción detallada	9
4.3	Cadenas De Color y Estilo	10
4.4	Configuracion Tablero	11
4.4.1	Descripción detallada	11
4.5	Elementos de tablero.	12
4.5.1	Descripción detallada	12
4.6	Movimientos de juego.	13
4.6.1	Descripción detallada	13
4.7	Codigo de Movimiento	14
4.7.1	Descripción detallada	14
4.8	Constantes con el aumento de puntaje	15
4.8.1	Descripción detallada	15
4.9	Tasas de aumentos para las etaoas	16
4.9.1	Descripción detallada	16
4.10	Configuracion para los tampolines	17
4.10.1	Descripción detallada	17
4.11	Configuracion del Ranking	18
4.11.1	Descripción detallada	18
5	Documentación de las estructuras de datos	19

5.1	Referencia de la Estructura itemRanking	19
5.1.1	Descripción detallada	19
6	Documentación de archivos	21
6.1	Referencia del Archivo include/colores.h	21
6.1.1	Descripción detallada	23
6.1.2	Documentación de las funciones	24
6.1.2.1	fijarColorEstilo	24
6.1.2.2	fijarColorFondo	24
6.1.2.3	fijarColorTexto	24
6.1.2.4	fijarColorTextoEstilo	24
6.1.2.5	fijarColorTextoFondo	24
6.1.2.6	fijarColorTextoFondoEstilo	25
6.2	Referencia del Archivo include/memoria.h	25
6.2.1	Descripción detallada	25
6.2.2	Documentación de las funciones	26
6.2.2.1	liberarMemoriaMatriz	26
6.2.2.2	liberarMemoriaMatrizCaracter	26
6.2.2.3	liberarMemoriaMatrizDos	26
6.2.2.4	liberarMemoriaMatrizEntera	26
6.2.2.5	pedirMemoriaMatriz	26
6.2.2.6	pedirMemoriaMatrizCaracter	27
6.2.2.7	pedirMemoriaMatrizDos	27
6.2.2.8	pedirMemoriaMatrizEntera	27
6.3	Referencia del Archivo include/my_function.h	27
6.3.1	Descripción detallada	28
6.3.2	Documentación de las enumeraciones	28
6.3.2.1	Bool	28
6.3.3	Documentación de las funciones	29
6.3.3.1	mygets	29
6.3.3.2	pausaMensaje	29
6.3.3.3	pedirCadena	29
6.3.3.4	preguntayn	29
6.3.3.5	println	29
6.3.3.6	redondeoEntero	30
6.4	Referencia del Archivo include/printascii.h	30
6.4.1	Descripción detallada	31
6.4.2	Documentación de las funciones	31
6.4.2.1	printLuminoso	31
6.4.2.2	printMasMenos	31

6.4.2.3	printMsjError	31
6.4.2.4	printMsjErrorPausa	31
6.4.2.5	printMsjInfo	32
6.4.2.6	printMsjInfoPausa	32
6.4.2.7	printMsjOk	32
6.4.2.8	printMsjOkPausa	32
6.5	Referencia del Archivo include/zanahoria.h	32
6.5.1	Descripción detallada	35
6.5.2	Documentación de las funciones	35
6.5.2.1	cargarPartida	35
6.5.2.2	cargarRanking	35
6.5.2.3	ejecutarMovimientoConejos	36
6.5.2.4	ejecutarMovimientoZanahoria	36
6.5.2.5	ejecutarTeletransportacion	37
6.5.2.6	guardarPartida	37
6.5.2.7	guardarRanking	37
6.5.2.8	ingresarRanking	37
6.5.2.9	inicializarRanking	38
6.5.2.10	mostrarRanking	38
6.5.2.11	mostrarRankingDestacado	38
6.5.2.12	pedirConejosIniciales	38
6.5.2.13	pedirDimensionTablero	38
6.5.2.14	pedirSiguienteMovimiento	39
6.5.2.15	posicionZanahoria	39
6.5.2.16	tablero_ini	39
6.5.2.17	tablero_pretty_view	39
6.5.2.18	tablero_view	40
6.5.2.19	ubicarConejosIniciales	40
6.5.2.20	ubicarTrampolines	40
6.5.2.21	ubicarZanahoriaInicial	40
6.5.2.22	verificarPrimeraVecindadZanahoria	41
6.5.2.23	verificarSegundaVecindadZanahoria	41
6.5.2.24	verificarVecindadZanahoria	41
6.6	Referencia del Archivo main.c	41
6.6.1	Descripción detallada	42

Capítulo 1

Indice de módulos

1.1. Módulos

Lista de todos los módulos:

Codigos Funciones de Colores.h	7
Cadenas De Color y Estilo	8
Cadenas De Color y Estilo	10
Configuracion Tablero	11
Elementos de tablero.	12
Movimientos de juego.	13
Codigo de Movimiento	14
Constantes con el aumento de puntaje	15
Tasas de aumentos para las etaoas	16
Configuracion para los tampolines	17
Configuracion del Ranking	18

Capítulo 2

Índice de estructura de datos

2.1. Estructura de datos

Lista de estructuras con una breve descripción:

itemRanking	
Estructura de Datos para los datos del Ranking	19

Capítulo 3

Indice de archivos

3.1. Lista de archivos

Lista de todos los archivos documentados y con descripciones breves:

main.c	Juego de la Zanahoria	41
include/colores.h	Para implementar colores en terminales ANSI	21
include/memoria.h	Libreria para solicitar y Liberar Memoria	25
include/my_function.h	Mis Funciones Reutilizables	27
include/printascii.h	Libreria para la impresion de Arte ASCII	30
include/zanahoria.h	Funciones Exclusivas para el juego	32

Capítulo 4

Documentación de módulos

4.1. Codigos Funciones de Colores.h

'defines'

- #define **ESTILO_NORMAL** 0
- #define **ESTILO_CLARO** 1
- #define **ESTILO_SUBRAYADO** 4
- #define **ESTILO_PARPADANTE** 5
- #define **ESTILO_INVERSO** 7
- #define **ESTILO_OCULTO** 8
- #define **COLOR_NEGRO** 30
- #define **COLOR_ROJO** 31
- #define **COLOR_VERDE** 32
- #define **COLOR_MARRON** 33
- #define **COLOR_AZUL** 34
- #define **COLOR_PURPURA** 35
- #define **COLOR_CYAN** 36
- #define **COLOR_GNIS** 37
- #define **FONDO_NEGRO** 40
- #define **FONDO_ROJO** 41
- #define **FONDO_VERDE** 42
- #define **FONDO_MARRON** 43
- #define **FONDO_AZUL** 44
- #define **FONDO_PURPURA** 45
- #define **FONDO_CYAN** 46
- #define **FONDO_GNIS** 47

4.1.1. Descripción detallada

Constantes con los codigos disponibles para pasarle a las funciones definidas en [colores.h](#) que representan los diferentes colores, estilos y fondos que son capaces de representar.

4.2. Cadenas De Color y Estilo

'defines'

- #define ESTILO_NORMAL_S "\033[0m"
- #define ESTILO_NORMAL_S ""
- #define ESTILO_CLARO_S "\033[1m"
- #define ESTILO_CLARO_S ""
- #define ESTILO_SUBRAYADO_S "\033[4m"
- #define ESTILO_SUBRAYADO_S ""
- #define ESTILO_PARPADEANTE_S "\033[5m"
- #define ESTILO_PARPADEANTE_S ""
- #define ESTILO_INVERSO_S "\033[7m"
- #define ESTILO_INVERSO_S ""
- #define ESTILO_OCULTO_S "\033[8m"
- #define ESTILO_OCULTO_S ""
- #define COLOR_NEGRO_S "\033[0;30m"
- #define COLOR_NEGRO_S "\033[0;30m"
- #define COLOR_ROJO_S "\033[0;31m"
- #define COLOR_ROJO_S "\033[0;31m"
- #define COLOR_VERDE_S "\033[0;32m"
- #define COLOR_VERDE_S "\033[0;32m"
- #define COLOR_MARRON_S "\033[0;33m"
- #define COLOR_MARRON_S "\033[0;33m"
- #define COLOR_AZUL_S "\033[0;34m"
- #define COLOR_AZUL_S "\033[0;34m"
- #define COLOR_PURPURA_S "\033[0;35m"
- #define COLOR_PURPURA_S "\033[0;35m"
- #define COLOR_CYAN_S "\033[0;36m"
- #define COLOR_CYAN_S "\033[0;36m"
- #define COLOR_GRIS_S "\033[0;37m"
- #define COLOR_GRIS_S "\033[0;37m"
- #define COLOR_NEGRO_CLARO_S "\033[1;30m"
- #define COLOR_NEGRO_CLARO_S "\033[1;30m"
- #define COLOR_ROJO_CLARO_S "\033[1;31m"
- #define COLOR_ROJO_CLARO_S "\033[1;31m"
- #define COLOR_VERDE_CLARO_S "\033[1;32m"
- #define COLOR_VERDE_CLARO_S "\033[1;32m"
- #define COLOR_MARRON_CLARO_S "\033[1;33m"
- #define COLOR_MARRON_CLARO_S "\033[1;33m"
- #define COLOR_AZUL_CLARO_S "\033[1;34m"
- #define COLOR_AZUL_CLARO_S "\033[1;34m"
- #define COLOR_PURPURA_CLARO_S "\033[1;35m"
- #define COLOR_PURPURA_CLARO_S "\033[1;35m"
- #define COLOR_CYAN_CLARO_S "\033[1;36m"
- #define COLOR_CYAN_CLARO_S "\033[1;36m"
- #define COLOR_GRIS_CLARO_S "\033[1;37m"
- #define COLOR_GRIS_CLARO_S "\033[1;37m"
- #define FONDO_NEGRO_S "\033[0;40m"
- #define FONDO_NEGRO_S "\033[0;40m"
- #define FONDO_ROJO_S "\033[0;41m"
- #define FONDO_ROJO_S "\033[0;41m"
- #define FONDO_VERDE_S "\033[0;42m"
- #define FONDO_VERDE_S "\033[0;42m"

- #define FONDO_MARRON_S "\033[0;43m"
- #define FONDO_MARRON_S "\033[0;43m"
- #define FONDO_AZUL_S "\033[0;44m"
- #define FONDO_AZUL_S "\033[0;44m"
- #define FONDO_PURPURA_S "\033[0;45m"
- #define FONDO_PURPURA_S "\033[0;45m"
- #define FONDO_CYAN_S "\033[0;46m"
- #define FONDO_CYAN_S "\033[0;46m"
- #define FONDO_GRIS_S "\033[0;47m"
- #define FONDO_GRIS_S "\033[0;47m"
- #define FONDO_NEGRO_CLARO_S "\033[1;40m"
- #define FONDO_NEGRO_CLARO_S "\033[1;40m"
- #define FONDO_ROJO_CLARO_S "\033[1;41m"
- #define FONDO_ROJO_CLARO_S "\033[1;41m"
- #define FONDO_VERDE_CLARO_S "\033[1;42m"
- #define FONDO_VERDE_CLARO_S "\033[1;42m"
- #define FONDO_MARRON_CLARO_S "\033[1;43m"
- #define FONDO_MARRON_CLARO_S "\033[1;43m"
- #define FONDO_AZUL_CLARO_S "\033[1;44m"
- #define FONDO_AZUL_CLARO_S "\033[1;44m"
- #define FONDO_PURPURA_CLARO_S "\033[1;45m"
- #define FONDO_PURPURA_CLARO_S "\033[1;45m"
- #define FONDO_CYAN_CLARO_S "\033[1;46m"
- #define FONDO_CYAN_CLARO_S "\033[1;46m"
- #define FONDO_GRIS_CLARO_S "\033[1;47m"
- #define FONDO_GRIS_CLARO_S "\033[1;47m"

4.2.1. Descripción detallada

Cadenas de colores, estilos y fondo para el texto en la consola basado, soportado por todas las consolas ANSI

4.3. Cadenas De Color y Estilo

La consola de comandos por defecto no soporta el estandar ANSI por lo tanto las cadenas se mostrarian como caracteres extranos por eso para agregar compatibilidad se usan las mismas constantes pero como una cadena vacia.

4.4. Configuración Tablero

'defines'

- #define **TABLERO_MIN_SIZE** 5
- #define **TABLERO_MAX_SIZE** 25

4.4.1. Descripción detallada

Configuración del tamaño máximo y mínimo de tablero el tamaño mínimo es debido a la cantidad mínima para que se desarrolle un juego, con al menos dos conejos El tamaño máximo está dado por el espacio que ocuparía el tablero en una consola a pantalla completa (resolución 1280x800) para que el tablero pudiera caer entero y ser jugable sin hacer scroll

4.5. Elementos de tablero.

'defines'

- #define **ZANAHORIA** 'z'
- #define **CONEJO** '&'
- #define **CADAVER** '#'
- #define **TRAMPOLIN** '<'
- #define **CONEJO_TRAMPOLIN** '?'
- #define **CELDA_VACIA** ' '
- #define **ZANAHORIA_MUERTA** 'x'

4.5.1. Descripción detallada

Constantes que representan a los elementos del tablero de juegos

4.6. Movimientos de juego.

'defines'

- #define **MOV_TOP_N** '8'
- #define **MOV_TOP_C** 'u'
- #define **MOV_TOPLEFT_N** '7'
- #define **MOV_TOPLEFT_C** 'y'
- #define **MOV_LEFT_N** '4'
- #define **MOV_LEFT_C** 'h'
- #define **MOV_BOTTOMLEFT_N** '1'
- #define **MOV_BOTTOMLEFT_C** 'n'
- #define **MOV_BOTTOM_N** '2'
- #define **MOV_BOTTOM_C** 'm'
- #define **MOV_BOTTOMRIGHT_N** '3'
- #define **MOV_BOTTOMRIGHT_C** ','
- #define **MOV_RIGHT_N** '6'
- #define **MOV_RIGHT_C** 'k'
- #define **MOV_TOPRIGHT_N** '9'
- #define **MOV_TOPRIGHT_C** 'i'
- #define **MOV_CENTER_N** '5'
- #define **MOV_CENTER_C** 'j'
- #define **MOV_TRANSPORT_C** 't'
- #define **ACTION_QUIT_C** 'q'
- #define **ACTION_SAVE_C** 's'

4.6.1. Descripción detallada

Comstantes que representan a las teclas validas como comandos de juego.

4.7. Código de Movimiento

'defines'

- #define **MOVE_TOP** 8
- #define **MOVE_TOPLEFT** 7
- #define **MOVE_LEFT** 4
- #define **MOVE_BOTTOMLEFT** 1
- #define **MOVE_BOTTOM** 2
- #define **MOVE_BOTTOMRIGHT** 3
- #define **MOVE_RIGHT** 6
- #define **MOVE_TOPRIGHT** 9
- #define **MOVE_CENTER** 5
- #define **MOVE_TRANSPORT** 10
- #define **ACTION_QUIT** 11
- #define **ACTION_SAVE** 12

4.7.1. Descripción detallada

Codigos internos de las diferentes acciones dentro del juego

4.8. Constantes con el aumento de puntaje

'defines'

- #define **PUNTAJE_MOV** 5
- #define **PUNTAJE_CHOQUE** 50
- #define **PUNTAJE_NIVEL** 100

4.8.1. Descripción detallada

Constantes con la cantidad de puntaje que se otorga por cada accion

4.9. Tasas de aumentos para las etaoas

'defines'

- #define **TASA_AUMENTO_CONEJOS** 0.25
- #define **TASA_AUMENTO_TRAMPOLINES** 0.2

4.9.1. Descripción detallada

Tasas de incremento para los conejos y los trampolines a medida que avanza el juego

4.10. Configuración para los trampolines

'defines'

- #define **NIVEL_TRAMPOLINES_START** 2
- #define **TRAMPOLINES_INICIALES** 2

4.10.1. Descripción detallada

Configuración de nivel en donde inician los trampolines y la cantidad de trampolines iniciales

4.11. Configuración del Ranking

'defines'

- #define **MAX_SIZE_NAME** 20
- #define **RANKING_NUM** 10
- #define **FILE_RANKING** "ranking.dat"

4.11.1. Descripción detallada

Configuración del tamaño máximo del nombre Cantidad de entradas en el ranking y archivo en donde se guardará el ranking

Capítulo 5

Documentación de las estructuras de datos

5.1. Referencia de la Estructura itemRanking

Estructura de Datos para los datos del Ranking.

```
#include <zanahoria.h>
```

Campos de datos

- char **nombre** [MAX_SIZE_NAME+1]
- int **puntaje**

5.1.1. Descripción detallada

Estructura de Datos para los datos del Ranking.

La documentación para esta estructura fue generada a partir del siguiente fichero:

- include/[zanahoria.h](#)

Capítulo 6

Documentación de archivos

6.1. Referencia del Archivo include/colores.h

Para implementar colores en terminales ANSI.

```
#include <stdio.h>
```

'defines'

- #define **ESTILO_NORMAL** 0
- #define **ESTILO_CLARO** 1
- #define **ESTILO_SUBRAYADO** 4
- #define **ESTILO_PARPADEANTE** 5
- #define **ESTILO_INVERSO** 7
- #define **ESTILO_OCULTO** 8
- #define **COLOR_NEGRO** 30
- #define **COLOR_ROJO** 31
- #define **COLOR_VERDE** 32
- #define **COLOR_MARRON** 33
- #define **COLOR_AZUL** 34
- #define **COLOR_PURPURA** 35
- #define **COLOR_CYAN** 36
- #define **COLOR_GRIS** 37
- #define **FONDO_NEGRO** 40
- #define **FONDO_ROJO** 41
- #define **FONDO_VERDE** 42
- #define **FONDO_MARRON** 43
- #define **FONDO_AZUL** 44
- #define **FONDO_PURPURA** 45
- #define **FONDO_CYAN** 46
- #define **FONDO_GRIS** 47
- #define **ESTILO_NORMAL_S** "\033[0m"
- #define **ESTILO_CLARO_S** "\033[1m"
- #define **ESTILO_SUBRAYADO_S** "\033[4m"
- #define **ESTILO_PARPADEANTE_S** "\033[5m"
- #define **ESTILO_INVERSO_S** "\033[7m"
- #define **ESTILO_OCULTO_S** "\033[8m"
- #define **COLOR_NEGRO_S** "\033[0;30m"
- #define **COLOR_ROJO_S** "\033[0;31m"

- #define COLOR_VERDE_S "\033[0;32m"
- #define COLOR_MARRON_S "\033[0;33m"
- #define COLOR_AZUL_S "\033[0;34m"
- #define COLOR_PURPURA_S "\033[0;35m"
- #define COLOR_CYAN_S "\033[0;36m"
- #define COLOR_GRIS_S "\033[0;37m"
- #define COLOR_NEGRO_CLARO_S "\033[1;30m"
- #define COLOR_ROJO_CLARO_S "\033[1;31m"
- #define COLOR_VERDE_CLARO_S "\033[1;32m"
- #define COLOR_MARRON_CLARO_S "\033[1;33m"
- #define COLOR_AZUL_CLARO_S "\033[1;34m"
- #define COLOR_PURPURA_CLARO_S "\033[1;35m"
- #define COLOR_CYAN_CLARO_S "\033[1;36m"
- #define COLOR_GRIS_CLARO_S "\033[1;37m"
- #define FONDO_NEGRO_S "\033[0;40m"
- #define FONDO_ROJO_S "\033[0;41m"
- #define FONDO_VERDE_S "\033[0;42m"
- #define FONDO_MARRON_S "\033[0;43m"
- #define FONDO_AZUL_S "\033[0;44m"
- #define FONDO_PURPURA_S "\033[0;45m"
- #define FONDO_CYAN_S "\033[0;46m"
- #define FONDO_GRIS_S "\033[0;47m"
- #define FONDO_NEGRO_CLARO_S "\033[1;40m"
- #define FONDO_ROJO_CLARO_S "\033[1;41m"
- #define FONDO_VERDE_CLARO_S "\033[1;42m"
- #define FONDO_MARRON_CLARO_S "\033[1;43m"
- #define FONDO_AZUL_CLARO_S "\033[1;44m"
- #define FONDO_PURPURA_CLARO_S "\033[1;45m"
- #define FONDO_CYAN_CLARO_S "\033[1;46m"
- #define FONDO_GRIS_CLARO_S "\033[1;47m"
- #define ESTILO_NORMAL_S ""
- #define ESTILO_CLARO_S ""
- #define ESTILO_SUBRAYADO_S ""
- #define ESTILO_PARPADEANTE_S ""
- #define ESTILO_INVERSO_S ""
- #define ESTILO_OCULTO_S ""
- #define COLOR_NEGRO_S "\033[0;30m"
- #define COLOR_ROJO_S "\033[0;31m"
- #define COLOR_VERDE_S "\033[0;32m"
- #define COLOR_MARRON_S "\033[0;33m"
- #define COLOR_AZUL_S "\033[0;34m"
- #define COLOR_PURPURA_S "\033[0;35m"
- #define COLOR_CYAN_S "\033[0;36m"
- #define COLOR_GRIS_S "\033[0;37m"
- #define COLOR_NEGRO_CLARO_S "\033[1;30m"
- #define COLOR_ROJO_CLARO_S "\033[1;31m"
- #define COLOR_VERDE_CLARO_S "\033[1;32m"
- #define COLOR_MARRON_CLARO_S "\033[1;33m"
- #define COLOR_AZUL_CLARO_S "\033[1;34m"
- #define COLOR_PURPURA_CLARO_S "\033[1;35m"
- #define COLOR_CYAN_CLARO_S "\033[1;36m"
- #define COLOR_GRIS_CLARO_S "\033[1;37m"
- #define FONDO_NEGRO_S "\033[0;40m"
- #define FONDO_ROJO_S "\033[0;41m"
- #define FONDO_VERDE_S "\033[0;42m"

- `#define FONDO_MARRON_S "\033[0;43m"`
- `#define FONDO_AZUL_S "\033[0;44m"`
- `#define FONDO_PURPURA_S "\033[0;45m"`
- `#define FONDO_CYAN_S "\033[0;46m"`
- `#define FONDO_GRIS_S "\033[0;47m"`
- `#define FONDO_NEGRO_CLARO_S "\033[1;40m"`
- `#define FONDO_ROJO_CLARO_S "\033[1;41m"`
- `#define FONDO_VERDE_CLARO_S "\033[1;42m"`
- `#define FONDO_MARRON_CLARO_S "\033[1;43m"`
- `#define FONDO_AZUL_CLARO_S "\033[1;44m"`
- `#define FONDO_PURPURA_CLARO_S "\033[1;45m"`
- `#define FONDO_CYAN_CLARO_S "\033[1;46m"`
- `#define FONDO_GRIS_CLARO_S "\033[1;47m"`

Funciones

- void `fijarColorNormal` ()
Resetea la terminal a sus colores originales.
- void `fijarColorTexto` (const int color)
Fija el color del texto a color.
- void `fijarColorFondo` (const int fondo)
Fija el color del fondo a fondo.
- void `fijarColorEstilo` (const int estilo)
Fija el estilo de la terminal.
- void `fijarColorTextoFondo` (const int texto, const int fondo)
Fija el color y el fondo del texto.
- void `fijarColorTextoFondoEstilo` (const int texto, const int fondo, const int estilo)
Fija el color, el estilo y el fondo del texto.
- void `fijarColorTextoEstilo` (const int texto, const int estilo)
Fija el color y el estilo del texto.

6.1.1. Descripción detallada

Para implementar colores en terminales ANSI. Con esta libreria se podran utilizar las características de las terminales ANSI para poder dar color, fondo y estilo a la terminal y así agregar un poco de riqueza al contenido mostrado.

Versión

0.1

Fecha

22/04/2012

Autor

JesusGoku

6.1.2. Documentación de las funciones

6.1.2.1. void fijarColorEstilo (const int *estilo*)

Fija el estilo de la terminal.

Parámetros

<i>estilo</i>	corresponde a la constantes ESTILO_XXX donde XXX son los estilos dentro de los disponibles
---------------	--

6.1.2.2. void fijarColorFondo (const int *fondo*)

Fija el color del fondo a fondo.

Parámetros

<i>fondo</i>	corresponde a las constantes FONDO_XXX donde XXX puede ser cualquiera de los fondos disponibles
--------------	---

6.1.2.3. void fijarColorTexto (const int *color*)

Fija el color del texto a color.

Parámetros

<i>color</i>	corresponde a las constantes COLOR_XXX donde XXX puede ser cualquier color de los disponibles
--------------	---

6.1.2.4. void fijarColorTextoEstilo (const int *texto*, const int *estilo*)

Fija el color y el estilo del texto.

Parámetros

<i>texto</i>	corresponde a las constantes COLOR_XXX donde XXX puede ser cualquier color de los disponibles
<i>estilo</i>	corresponde a las constantes ESTILO_XXX donde XXX puede ser cualquier estilo de los disponibles

6.1.2.5. void fijarColorTextoFondo (const int *texto*, const int *fondo*)

Fija el color y el fondo del texto.

Parámetros

<i>texto</i>	corresponde a las constantes COLOR_XXX donde XXX puede ser cualquier color de los disponibles
<i>fondo</i>	corresponde a las constantes FONDO_XXX donde XXX puede ser cualquiera de los fondos disponibles

6.1.2.6. void fijarColorTextoFondoEstilo (const int *texto*, const int *fondo*, const int *estilo*)

Fija el color, el estilo y el fondo del texto.

Parámetros

<i>texto</i>	corresponde a las constantes COLOR_XXX donde XXX puede ser cualquier color de los disponibles
<i>fondo</i>	corresponde a las constantes FONDO_XXX donde XXX puede ser cualquiera de los fondos disponibles
<i>estilo</i>	corresponde a las constantes ESTILO_XXX donde XXX puede ser cualquier estilo de los disponibles

6.2. Referencia del Archivo include/memoria.h

Librería para solicitar y Liberar Memoria.

```
#include <stdio.h>
#include <stdlib.h>
```

Funciones

- int ** pedirMemoriaMatrizEntera (const int filas, const int columnas)
Pedir memoria para una matriz de enteros.
- void liberarMemoriaMatrizEntera (int **pm)
Liberar memoria de una matriz entera.
- char ** pedirMemoriaMatrizCaracter (const int filas, const int columnas)
Pedir memoria para una matriz de caracteres.
- void liberarMemoriaMatrizCaracter (char **pm)
Liberar memoria de una matriz de caracteres.
- void ** pedirMemoriaMatriz (const int filas, const int columnas, const char tipo)
Pedir memoria para una matriz.
- void liberarMemoriaMatriz (void **pm)
Liberar memoria de una matriz.
- void ** pedirMemoriaMatrizDos (const int filas, const int columnas, const char tipo)
Pedir memoria para una matriz (Metodo por Filas).
- void liberarMemoriaMatrizDos (void **pm, const int filas)
Liberar memoria pedida por filas.

6.2.1. Descripción detallada

Librería para solicitar y Liberar Memoria. Hay funciones para pedir memoria facilmente para una matriz en una sola llamada y liberarla tambien.

Para solicitar la memoria se utiliza un metodo para reducir el numero de llamadas a malloc solo a dos Lo cual lo hace eficiente en tiempo de procesador, pero como la memoria solicitada debe ser continua se corre el riesgo de no encontrar el suficiente espacio para la matriz, este riesgo se reduce al pedir la memoria de cada fila por separado, pero considerando que los computadores actuales tienen suficiente memoria opte por utilizar este metodo.

Versión

0.1

Fecha

22/04/2012

Autor

JesusGoku

6.2.2. Documentación de las funciones

6.2.2.1. void liberarMemoriaMatriz (void ** *pm*)

Liberar memoria de una matriz.

Parámetros

<i>pm</i>	Puntero a la memoria que se desea liberar
-----------	---

6.2.2.2. void liberarMemoriaMatrizCaracter (char ** *pm*)

Liberar memoria de una matriz de caracteres.

Parámetros

<i>pm</i>	Puntero a la memoria que se desea liberar
-----------	---

6.2.2.3. void liberarMemoriaMatrizDos (void ** *pm*, const int *filas*)

Liberar memoria pedida por filas.

Parámetros

<i>pm</i>	Puntero a la memoria que se desea liberar
<i>filas</i>	Cantidad de filas que seran liberadas

6.2.2.4. void liberarMemoriaMatrizEntera (int ** *pm*)

Liberar memoria de una matriz entera.

Parámetros

<i>pm</i>	Puntero a la memoria que se desea liberar
-----------	---

6.2.2.5. void** pedirMemoriaMatriz (const int *filas*, const int *columnas*, const char *tipo*)

Pedir memoria para una matriz.

Te permite pedir memoria para una matriz de entre 3 tipos (Enteros, Caracteres, Flotantes) especificandolo en el tercer parametro

Parámetros

<i>filas</i>	Filas de la matriz a solicitar
<i>columnas</i>	Columnas de la matriz a solicitar
<i>tipo</i>	Cacter que representa al tipo de la matriz c -> char i -> int f -> float

Devuelve

Puntero a la matriz solicitada, se debe hacer un casting al tipo pedido

6.2.2.6. char pedirMemoriaMatrizCaracter (const int *filas*, const int *columnas*)**

Pedir memoria para una matriz de caracteres.

Parámetros

<i>filas</i>	Filas de la matriz a solicitar
<i>columnas</i>	Columnas de la matriz a solicitar

Devuelve

Puntero a la matriz solicitada

6.2.2.7. void pedirMemoriaMatrizDos (const int *filas*, const int *columnas*, const char *tipo*)**

Pedir memoria para una matriz (Metodo por Filas).

Te permite pedir memoria para una matriz de entre 3 tipos (Enteros, Caracteres, Flotantes) especificandolo en el tercer parametro.

Esta pide la memoria fila por fila, para ser utilizada en maquinas con poca memoria continua

Parámetros

<i>filas</i>	Filas de la matriz a solicitar
<i>columnas</i>	Columnas de la matriz a solicitar
<i>tipo</i>	Cacter que representa al tipo de la matriz c -> char i -> int f -> float

Devuelve

Puntero a la matriz solicitada, se debe hacer un casting al tipo pedido

6.2.2.8. int pedirMemoriaMatrizEntera (const int *filas*, const int *columnas*)**

Pedir memoria para una matriz de enteros.

Parámetros

<i>filas</i>	Filas de la matriz a solicitar
<i>columnas</i>	Columnas de la matriz a solicitar

Devuelve

Puntero a la matriz solicitada

6.3. Referencia del Archivo include/my_function.h

Mis Funciones Reutilizables.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <string.h>
```

Enumeraciones

- enum **Bool** { **False**, **True** }
Emulacion del tipo Booleano.

Funciones

- int **println** (const char *formato,...)
Como el print de python.
- int **redondeoEntero** (float num)
: Redondea un flotante a entero
- void **clearScr** ()
Limpiar Buffer de Entrada.
- void **clearStdin** ()
Limpiar Pantalla.
- void **pausa** ()
Introducir una pausa en la ejecucion.
- void **pausaMensaje** (const char *mensaje)
Muestra un mensaje e introduce una pausa.
- int **preguntayn** (const char *pregunta)
Pregunta al usuario dando la posibilidad de responder y/n.
- void **mygets** (char *cadena, const int max)
Captura una cadena desde la entrada estadar.
- void **pedirCadena** (const char *mensaje, char *destino, const int max)
Pide una cadena al usuario de forma segura.

6.3.1. Descripción detallada

Mis Funciones Reutilizables. En esta libreria se encuentras funciones de uso general

Versión

0.1

Fecha

22/04/2012

Autor

JesusGoku

6.3.2. Documentación de las enumeraciones

6.3.2.1. enum **Bool**

Emulacion del tipo Booleano.

6.3.3. Documentación de las funciones

6.3.3.1. void mygets (char * *cadena*, const int *max*)

Captura una cadena desde la entrada estadar.

Mas segura que gets ya que se debe especificar el limite a capturar y se elimina el "ENTER" dejado por fgets

Parámetros

<i>cadena</i>	indica donde se guardara la cadena capturada
<i>max</i>	langitud maxima de caracteres a capturar

6.3.3.2. void pausaMensaje (const char * *mensaje*)

Muestra un mensaje e introduce una pausa.

Parámetros

<i>mensaje</i>	mensaje que se desea mostrar antes de introducir la pausa
----------------	---

6.3.3.3. void pedirCadena (const char * *mensaje*, char * *destino*, const int *max*)

Pide una cadena al usuario de forma segura.

Para su utilizacion utiliza la funcion mygets de esta misma libreria

Parámetros

<i>mensaje</i>	indicacion de la cadena a ingresar
<i>destino</i>	cadena donde se guardara la cadena ingresada por el usuario
<i>max</i>	longitud maxima de caracteres a capturar

6.3.3.4. int preguntayn (const char * *pregunta*)

Pregunta al usuario dando la posibilidad de responder y/n.

Parámetros

<i>pregunta</i>	pregunta a mostrar al usuario
-----------------	-------------------------------

Devuelve

respuesta del usuario, 1 para y 0 para n

6.3.3.5. int println (const char * *formato*, ...)

Como el print de python.

Da un salto de carro luego de imprimir, por lo demas se comporta igual que printf

Parámetros

<i>formato</i>	cadena de formato
...	los distintos atributos especificados en formato

Devuelve

cantidad de parametros imprimidos

6.3.3.6. int redondeoEntero (float num)

: Redondea un flotante a entero

Parámetros

<i>num</i>	numero que desea ser redondeado
------------	---------------------------------

Devuelve

numero redondeado

6.4. Referencia del Archivo include/printascii.h

Libreria para la impresion de Arte ASCII.

```
#include <stdio.h>
#include <my_function.h>
#include <string.h>
#include <colores.h>
```

Funciones

- void [printZanahoria](#) ()
Imprime el logo de bienvenida al juego de la zanahoria.
- void [printConejoZanahoria](#) ()
Imprime el logo de un conejo comiendo una zanahoria.
- void [printConejoGameOver](#) ()
Imprime un conejo con el texto de GAME OVER!.
- void [printGraciasPorJugar](#) ()
Imprime gracias por jugar.
- void [printLuminoso](#) (char *cadena)
- void [printMasMenos](#) (char *cadena)
Imprime un letrero con letras parpadeantes encerrado por signos + y -.
- void [printMsjError](#) (const char *msj)
Imprime un mensaje de error en color rojo.
- void [printMsjErrorPausa](#) (const char *msj)
Imprimir un mensaje de error en color rojo e introducir una pausa en la ejecucion.
- void [printMsjOk](#) (const char *msj)
Imprime un mensaje de error en color verde.
- void [printMsjOkPausa](#) (const char *msj)
Imprimir un mensaje de confirmacion en color verde e introducir una pausa en la ejecucion.
- void [printMsjInfo](#) (const char *msj)
Imprime un mensaje de error en color marron.
- void [printMsjInfoPausa](#) (const char *msj)
Imprimir un mensaje de informacion en color marron e introducir una pausa en la ejecucion.

6.4.1. Descripción detallada

Libreria para la impresion de Arte ASCII. Colecciones de logos para impresion de arte ascii en la consola

Autor

JesusGoku

Versión

0.1

Fecha

22/04/2012

6.4.2. Documentación de las funciones

6.4.2.1. void printLuminoso (char * *cadena*)

Imprime un letrero con letras parpadeantes con el string apuntado por cadena.

Parámetros

<i>cadena</i>	puntero al string que se quiere mostrar en el mensaje
---------------	---

6.4.2.2. void printMasMenos (char * *cadena*)

Imprime un letrero con letras parpadeantes encerrado por signos + y -.

Parámetros

<i>cadena</i>	puntero al string que se quiere mostrar en el mensaje
---------------	---

6.4.2.3. void printMsjError (const char * *msj*)

Imprime un mensaje de error en color rojo.

Parámetros

<i>msj</i>	puntero a cadena con el mensaje que se desea mostrar
------------	--

6.4.2.4. void printMsjErrorPausa (const char * *msj*)

Imprimir un mensaje de error en color rojo e introducir una pausa en la ejecucion.

Parámetros

<i>msj</i>	puntero a cadena con el mensaje a mostrar
------------	---

6.4.2.5. void printMsjInfo (const char * *msj*)

Imprime un mensaje de error en color marron.

Parámetros

<i>msj</i>	puntero a cadena con el mensaje que se desea mostrar
------------	--

6.4.2.6. void printMsjInfoPausa (const char * *msj*)

Imprimir un mensaje de informacion en color marron e introducir una pausa en la ejecucion.

Parámetros

<i>msj</i>	puntero a cadena con el mensaje a mostrar
------------	---

6.4.2.7. void printMsjOk (const char * *msj*)

Imprime un mensaje de error en color verde.

Parámetros

<i>msj</i>	puntero a cadena con el mensaje que se desea mostrar
------------	--

6.4.2.8. void printMsjOkPausa (const char * *msj*)

Imprimir un mensaje de confirmacion en color verde e introducir una pausa en la ejecucion.

Parámetros

<i>msj</i>	puntero a cadena con el mensaje a mostrar
------------	---

6.5. Referencia del Archivo include/zanahoria.h

Funciones Exclusivas para el juego.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include <ctype.h>
#include <my_function.h>
#include <memoria.h>
#include <colores.h>
```

Estructuras de datos

- struct [itemRanking](#)

Estructura de Datos para los datos del Ranking.

'defines'

- #define **TABLERO_MIN_SIZE** 5
- #define **TABLERO_MAX_SIZE** 25
- #define **ZANAHORIA** 'z'
- #define **CONEJO** '&'
- #define **CADAVER** '#'
- #define **TRAMPOLIN** '<'
- #define **CONEJO_TRAMPOLIN** '?'
- #define **CELDA_VACIA** ' '
- #define **ZANAHORIA_MUERTA** 'x'
- #define **MOV_TOP_N** '8'
- #define **MOV_TOP_C** 'u'
- #define **MOV_TOPLEFT_N** '7'
- #define **MOV_TOPLEFT_C** 'y'
- #define **MOV_LEFT_N** '4'
- #define **MOV_LEFT_C** 'h'
- #define **MOV_BOTTOMLEFT_N** '1'
- #define **MOV_BOTTOMLEFT_C** 'n'
- #define **MOV_BOTTOM_N** '2'
- #define **MOV_BOTTOM_C** 'm'
- #define **MOV_BOTTOMRIGHT_N** '3'
- #define **MOV_BOTTOMRIGHT_C** ','
- #define **MOV_RIGHT_N** '6'
- #define **MOV_RIGHT_C** 'k'
- #define **MOV_TOPRIGHT_N** '9'
- #define **MOV_TOPRIGHT_C** 'i'
- #define **MOV_CENTER_N** '5'
- #define **MOV_CENTER_C** 'j'
- #define **MOV_TRANSPORT_C** 't'
- #define **ACTION_QUIT_C** 'q'
- #define **ACTION_SAVE_C** 's'
- #define **MOVE_TOP** 8
- #define **MOVE_TOPLEFT** 7
- #define **MOVE_LEFT** 4
- #define **MOVE_BOTTOMLEFT** 1
- #define **MOVE_BOTTOM** 2
- #define **MOVE_BOTTOMRIGHT** 3
- #define **MOVE_RIGHT** 6
- #define **MOVE_TOPRIGHT** 9
- #define **MOVE_CENTER** 5
- #define **MOVE_TRANSPORT** 10
- #define **ACTION_QUIT** 11
- #define **ACTION_SAVE** 12
- #define **PUNTAJE_MOV** 5
- #define **PUNTAJE_CHOQUE** 50
- #define **PUNTAJE_NIVEL** 100
- #define **TASA_AUMENTO_CONEJOS** 0.25
- #define **TASA_AUMENTO_TRAMPOLINES** 0.2
- #define **NIVEL_TRAMPOLINES_START** 2
- #define **TRAMPOLINES_INICIALES** 2
- #define **FILE_PARTIDA** "partida.dat"
- Archivo para guardar y recuperar la partida.*
- #define **MAX_SIZE_NAME** 20
- #define **RANKING_NUM** 10
- #define **FILE_RANKING** "ranking.dat"

'typedefs'

- typedef struct [itemRanking](#) [ItemRanking](#)
Estructura de Datos para los datos del Ranking.

Funciones

- void [pedirConejosIniciales](#) (int *f, int *c, int *ci)
Solicita la cantidad de conejos iniciales al usuario.
- void [pedirDimensionTablero](#) (int *f, int *c)
Solicita al usuario las dimensiones del tablero.
- void [tablero_ini](#) (char **tablero, int f, int c)
Inicializa el tablero al equivalente a CELDA_VACIA.
- void [tablero_view](#) (char **tablero, int f, int c)
Muestra el tablero en pantalla.
- void [tablero_pretty_view](#) (char **tablero, const int filas, const int columnas)
Muestra el tablero en pantalla de forma espectacular.
- void [ubicarZanahoriaInicial](#) (char **tablero, const int f, const int c)
Ubica a la zanahoria en el centro del escenario.
- void [ubicarConejosIniciales](#) (char **tablero, const int m, const int n, const int conejos)
Ubica una cierta cantidad de conejos aleatoriamente por el tablero.
- void [ubicarTrampolines](#) (char **tablero, const int m, const int n, const int trampolines)
Ubicar los trampolines aleatoriamente por el tablero.
- void [posicionZanahoria](#) (char **tablero, const int filas, const int columnas, int *coordZF, int *coordZC)
Devuelve las coordenadas de la zanahoria en el tablero.
- int [pedirSiguienteMovimiento](#) ()
Pide el siguiente movimiento a ejecutar al usuario.
- int [ejecutarMovimientoZanahoria](#) (const int mov, char **tablero, const int filas, const int columnas, const int nivel)
Ejecuta el movimiento de la zanahoria hasta mov.
- int [ejecutarMovimientoConejos](#) (char **tablero, char **tableroCopy, const int filas, const int columnas, int *conejosVivos, int *puntaje)
Ejecuta el movimiento de todos los conejos persiguiendo al conejo.
- void [ejecutarTeletransportacion](#) (char **tablero, const int filas, const int columnas)
Posicion al conejo en una posicion aleatoria del tablero.
- int [verificarVecindadZanahoria](#) (char **tablero, const int m, const int n, const int f, const int c, const int salto)
Verifica la vecindad de la zanahoria para verificar si hay peligro al ubicarse hay.
- int [verificarPrimeraVecindadZanahoria](#) (char **tablero, const int m, const int n, const int f, const int c)
Verifica la primera vecindad en busca de peligros.
- int [verificarSegundaVecindadZanahoria](#) (char **tablero, const int m, const int n, const int f, const int c)
Verifica la segunda vecindad en caso de peligros.
- char ** [cargarPartida](#) (char *ficheroName, int *filas, int *columnas, int *conejosIniciales, int *conejosVivos, int *nivel, int *puntaje)
Carga una partida desde un fichero.
- int [guardarPartida](#) (char *ficheroName, char **tablero, const int filas, const int columnas, const int conejosIniciales, const int conejosVivos, const int nivel, const int puntaje)
Guarda una partida en un fichero.
- void [mostrarRanking](#) ([ItemRanking](#) *ranking, const int n)
Muestra el ranking.
- void [mostrarRankingDestacado](#) ([ItemRanking](#) *ranking, const int n, const int pos)
Muestra el ranking destacando una poscion.

- int **ingresarRanking** (**ItemRanking** *ranking, const int n, **ItemRanking** *elemento)
Ingresar un elemento al ranking.
- int **guardarRanking** (**ItemRanking** *ranking, const int n)
Guarda el ranking a un archivo.
- int **cargarRanking** (**ItemRanking** *ranking, const int n)
Carga el ranking desde un archivo.
- void **inicializarRanking** (**ItemRanking** *ranking, const int n)
Inicializa las variables del ranking con valores iniciales.
- void **mostrarAyuda** ()
Muestra las instrucciones del juego.

6.5.1. Descripción detallada

Funciones Exclusivas para el juego. Aquí encontraras las funciones que son de utilidad para este juego y que difícilmente podran ser utilizadas en otros proyectos a menos que sean muy similares.

Versión

0.1

Fecha

22/04/2012

Autor

JesusGoku

6.5.2. Documentación de las funciones

6.5.2.1. char cargarPartida (char * ficheroName, int * filas, int * columnas, int * conejosIniciales, int * conejosVivos, int * nivel, int * puntaje)**

Carga una partida desde un fichero.

Parámetros

<i>ficheroName</i>	puntero a cadena con el nombre del archivo que contiene la partida
<i>filas</i>	puntero a entero donde se almacena la cantidad de filas del tablero guardado
<i>columnas</i>	puntero a entero donde se almacena la cantidad de columnas del tablero guardado
<i>conejosIniciales</i>	puntero a entero donde se almacenara la cantidad de conejos iniciales
<i>conejosVivos</i>	puntero a entero donde se almacenara la cantidad de conejos vivos en la partida guardada
<i>nivel</i>	puntero a entero donde se almacenara el nivel de la partida guardada
<i>puntaje</i>	puntero a entero donde se almacenara el puntaje en la partida guardada

Devuelve

devuelve un puntero al tablero con los datos de la partida guardada

6.5.2.2. int cargarRanking (ItemRanking * ranking, const int n)

Carga el ranking desde un archivo.

Parámetros

<i>ranking</i>	puntero a un arreglo de estructura itemRanking que contendra el ranking
<i>n</i>	entero correspondiente al tamaño del ranking que se espera en el archivo

Devuelve

devuelve 1 si logro abrir el archivo y recuperar el ranking, 0 de lo contrario

6.5.2.3. int ejecutarMovimientoConejos (char ** *tablero*, char ** *tableroCopy*, const int *filas*, const int *columnas*, int * *conejosVivos*, int * *puntaje*)

Ejecuta el movimiento de todos los conejos persiguiendo al conejo.

Parámetros

<i>tablero</i>	puntero a la matriz que representa al tablero
<i>tableroCopy</i>	puntero a la matriz que representa a una copia del tablero para poder realizar los movimientos
<i>filas</i>	entero con la cantidad de filas del tablero
<i>columnas</i>	entero con la cantidad de columnas del tablero
<i>conejosVivos</i>	puntero a entero donde se almacena la cantidad de conejos vivos que se disminuira en caso de colision de conejos
<i>puntaje</i>	puntero a entero donde se lleva el puntaje que ira aumentando en caso de colision de acuerdo a los establecido en la constante PUNTAJE_CHOQUE

Devuelve

Devuelve 0 en caso de que un conejo haya caído sobre la zanahoria, en caso contrario retorna 1

6.5.2.4. int ejecutarMovimientoZanahoria (const int *mov*, char ** *tablero*, const int *filas*, const int *columnas*, const int *nivel*)

Ejecuta el movimiento de la zanahoria hasta *mov*.

El movimiento se ejecuta siempre de que sea un movimiento valido es decir que no se salga del tablero o que no haya otro objeto en la casilla a la que se desea mover.

Ademas se debe validar que para que el movimiento sea valido, no exista el riesgo de que en la casilla a donde se va a mover vaya a ser comido. Por eso se utilizan dos funciones auxiliares para verificar las dos vecindades de si hay algun potencial conejo que pueda hacer invalida la jugada. Por ese motivo se pasa el nivel que a primeras podria parecer innecesario pero a como los trampolines solo comienzan a aparecer desde el 2do nivel y estos son lo que permiten a los conejos saltar de a dos espacios, no tiene sentido gastar tiempo llamando a la funcion que se encarga de revisar la segunda vecindad si no hay riesgo aun.

Para realizar el ahorro comentado en el parrafo anterior se aprovecha la evaluacion de expresiones logicas de C por cortocircuito evaluando si el nivel es menor que el nivel en que comienzan a aparecer los trampolines O verificar la vecindad, y ya que hasta que se alcance el nivel la primera expresion siempre dara verdadero, se ya sabe que la expresion es verdadera y no ejecutara la funcion para revisar la vecindad. No asi cuando se alcanza el nivel para los trampolines donde la primera expresion dara falso y C se vera obligado a evaluar la segunda expresion, osea la funcion que verifica el segundo cuadrante para conocer el valor de verdad de la expresion, dependiendo ahora exclusivamente de ella el valor de la expresion.

Parámetros

<i>mov</i>	entero que representa al movimiento que solicito el usuario
<i>tablero</i>	puntero a la matriz que representa al tablero
<i>filas</i>	entero con la cantidad de filas del tablero
<i>columnas</i>	entero con la cantidad de columnas del tablero
<i>nivel</i>	entero con el nivel en el que se encuentra el juego

Devuelve

Devuelve 0 si el movimiento no es valido, y 1 si lo es y la mueve hasta la posicion

6.5.2.5. void ejecutarTeletransportacion (char ** *tablero*, const int *filas*, const int *columnas*)

Posicion al conejo en una posicion alatoria del tablero.

Parámetros

<i>tablero</i>	puntero a la matriz que representa al tablero
<i>filas</i>	entero con la cantidad de filas del tablero
<i>columnas</i>	entero con la cantidad de columnas del tablero

6.5.2.6. int guardarPartida (char * *ficheroName*, char ** *tablero*, const int *filas*, const int *columnas*, const int *conejosIniciales*, const int *conejosVivos*, const int *nivel*, const int *puntaje*)

Guarda una partida en un fichero.

Parámetros

<i>ficheroName</i>	puntero a cadena con el nombre del fichero donde se guardara la partida
<i>tablero</i>	puntero a el tablero que se desea guardar
<i>filas</i>	entero con la cantidad de filas del tablero
<i>columnas</i>	entero con la cantidad de columnas del tablero
<i>conejosIniciales</i>	entero con la cantidad inicial de conejos ingresada por el usuario
<i>conejosVivos</i>	entero con la cantidad de conejos vivos al momento de llamar la funcion
<i>nivel</i>	entero con el nivel al momento de llamar a la funcion
<i>puntaje</i>	entero con el puntaje al momento de llamar a la funcion

6.5.2.7. int guardarRanking (ItemRanking * *ranking*, const int *n*)

Guarda el ranking a un archivo.

El ranking se guarda un archivo binario ya que hace mucho mas sencillo el recuperar la informacion posteriormente.

Parámetros

<i>ranking</i>	puntero a un arreglo de estructura itemRanking que contiene el ranking
<i>n</i>	entero correspondiente al tamano del ranking

Devuelve

1 si se logro abrir y guardar el ranking, 0 de lo contrario

6.5.2.8. int ingresarRanking (ItemRanking * *ranking*, const int *n*, ItemRanking * *elemento*)

Ingresa un elemento al ranking.

Parámetros

<i>ranking</i>	puntero a un arreglo de estructura itemRanking que contiene el ranking
<i>n</i>	entero correspondiente al tamano del ranking
<i>elemento</i>	puntero a estructura con los datos que quieres ingresar al ranking

Devuelve

devuelve 0 si no puede ingresar al ranking o el numero de la posicion en que quedo en el ranking

6.5.2.9. void inicializarRanking (ItemRanking * ranking, const int n)

Inicializa las variables del ranking con valores iniciales.

Parámetros

<i>ranking</i>	puntero a un arreglo de estructura itemRanking que apunta a el ranking
<i>n</i>	entero correspondiente al tamaño del ranking

6.5.2.10. void mostrarRanking (ItemRanking * ranking, const int n)

Muestra el ranking.

Parámetros

<i>ranking</i>	puntero al arreglo de estructuras que contiene el ranking
<i>n</i>	cantidad de entradas que tiene el ranking

6.5.2.11. void mostrarRankingDestacado (ItemRanking * ranking, const int n, const int pos)

Muestra el ranking destacando una posición.

Parámetros

<i>ranking</i>	puntero a estructura que contiene el ranking
<i>n</i>	entero correspondiente al tamaño del ranking
<i>pos</i>	entero con la posición que se desea destacar

6.5.2.12. void pedirConejosIniciales (int * f, int * c, int * ci)

Solicita la cantidad de conejos iniciales al usuario.

Se deben enviar las dimensiones del tablero para calcular una número máximo para la cantidad de conejos ingresadas por el usuario, cosa de que el juego tenga sentido. Se ha establecido arbitrariamente al 10% del tamaño total de casillas disponibles en el tablero esto pensando en que a medida de que avance el juego la cantidad de conejos ira aumentando.

También se fija la cantidad mínima de conejos en 2 ya que de lo contrario sería imposible que hubiera un choque entre ellos.

Parámetros

<i>f</i>	puntero a entero con el número de filas del tablero de juego
<i>c</i>	puntero a entero con el número de columnas del tablero de juego
<i>ci</i>	puntero a entero donde se almacenara el número de conejos iniciales ingresado.

6.5.2.13. void pedirDimensionTablero (int * f, int * c)

Solicita al usuario las dimensiones del tablero.

Se ha establecido arbitrariamente que las dimensiones del tablero deben de ser impar con el objeto de que haya un solo centro en lugar de cuatro cuando las dimensiones del tablero es par.

Tambien se ha establecido como tamaño minimo de tablero el numero 5 ya que es el minimo aceptable para que caiga la zanahoria en el centro y un 10% de conejos y se pueda desarrollar un juego.

Como comentario adicional se establecen la cantidad de filas y columnas por separado, aun cuando para este juego la dimension del tablero es cuadrada, pero he decidido dejarla asi para que pueda ser modificado facilmente en caso de que se desee que no sea asi.

Parámetros

<i>f</i>	puntero a entero donde se almacenara el numero de filas
<i>c</i>	puntero a entero donde se almacenara el numero de columnas

6.5.2.14. int pedirSiguienteMovimiento ()

Pide el siguiente movimiento a ejecutar al usuario.

Se encarga de validar que al menos sea un comando de juego valido o lo vuelve a pedir

Devuelve

devuelve un entero que representa el movimiento ingresado por el usuario

6.5.2.15. void posicionZanahoria (char ** tablero, const int filas, const int columnas, int * coordZF, int * coordZC)

Devuelve las coordenadas de la zanahoria en el tablero.

Parámetros

<i>tablero</i>	puntero a la matriz que representa al tablero
<i>filas</i>	entero con la cantidad de filas del tablero
<i>columnas</i>	entero con la cantidad de columnas del tablero
<i>coordZF</i>	puntero a entero donde se almacenara la coordena fila de la zanahoria
<i>coordZC</i>	puntero a entero donde se almacenara la coordena columna de la zanahoria

6.5.2.16. void tablero_ini (char ** tablero, int f, int c)

Inicializa el tablero al equivalente a CELDA_VACIA.

Llena todo las casillas del tablero con el caracter representado por la constante CELDA_VACIA

Parámetros

<i>tablero</i>	puntero a la matriz que representa al tablero
<i>f</i>	entero con la cantidad de filas del tablero
<i>c</i>	entero con la cantidad de columnas del tablero

6.5.2.17. void tablero_pretty_view (char ** tablero, const int filas, const int columnas)

Muestra el tablero en pantalla de forma espectacular.

Esta es una version mejorar la version anterior que muestra el tablero de una forma mas organizada a la vista al simular verdaderamente un tablero y sus casillas por donde se desplazan la zanahoria y los conejos que quieren devorarla. Tiene la desventaja de ocupar mas espacio para su representacion, lo que limita el tamaño de los

tableros posibles.

Parámetros

<i>tablero</i>	puntero a la matriz que representa al tablero
<i>filas</i>	entero con la cantidad de filas del tablero
<i>columnas</i>	entero con la cantidad de columnas del tablero

6.5.2.18. void `tablero_view` (*char ** tablero*, *int f*, *int c*)

Muestra el tablero en pantalla.

Parámetros

<i>tablero</i>	puntero a la matriz que representa al tablero
<i>f</i>	entero con la cantidad de filas del tablero
<i>c</i>	entero con la cantidad de columnas del tablero

6.5.2.19. void `ubicarConejosIniciales` (*char ** tablero*, *const int m*, *const int n*, *const int conejos*)

Ubica una cierta cantidad de conejos aleatoriamente por el tablero.

Parámetros

<i>tablero</i>	puntero a la matriz que representa al tablero
<i>m</i>	entero con la cantidad de filas del tablero
<i>n</i>	entero con la cantidad de columnas del tablero
<i>conejos</i>	entero con la cantidad de conejos a ubicar aleatoriamente

6.5.2.20. void `ubicarTrampolines` (*char ** tablero*, *const int m*, *const int n*, *const int trampolines*)

Ubicar los trampolines aleatoriamente por el tablero.

Parámetros

<i>tablero</i>	puntero a la matriz que representa al tablero
<i>m</i>	entero con la cantidad de filas del tablero
<i>n</i>	entero con la cantidad de columnas del tablero
<i>trampolines</i>	entero con la cantidad de tableros a ubicar aleatoriamente

6.5.2.21. void `ubicarZanahoriaInicial` (*char ** tablero*, *const int f*, *const int c*)

Ubica a la zanahoria en el centro del escenario.

Parámetros

<i>tablero</i>	puntero a la matriz que representa al tablero
<i>f</i>	entero con la cantidad de filas del tablero
<i>c</i>	entero con la cantidad de columnas del tablero

6.5.2.22. int verificarPrimeraVecindadZanahoria (char ** *tablero*, const int *m*, const int *n*, const int *f*, const int *c*)

Verifica la primera vecindad en busca de peligros.

Parámetros

<i>tablero</i>	puntero a la matriz que representa al tablero
<i>m</i>	entero con la cantidad de filas del tablero
<i>n</i>	entero con la cantidad de columnas del tablero
<i>f</i>	entero que indica la fila a donde estara ubicada la zanahoria
<i>c</i>	entero que indica la columna a donde estara ubicada la zanahoria

6.5.2.23. int verificarSegundaVecindadZanahoria (char ** *tablero*, const int *m*, const int *n*, const int *f*, const int *c*)

Verifica la segunda vecindad en caso de peligros.

Parámetros

<i>tablero</i>	puntero a la matriz que representa al tablero
<i>m</i>	entero con la cantidad de filas del tablero
<i>n</i>	entero con la cantidad de columnas del tablero
<i>f</i>	entero que indica la fila a donde estara ubicada la zanahoria
<i>c</i>	entero que indica la columna a donde estara ubicada la zanahoria
<i>salto</i>	entero, si valor es 1 verifica la primera vecindad, si valor es 2 verifica la segunda vecindad

6.5.2.24. int verificarVecindadZanahoria (char ** *tablero*, const int *m*, const int *n*, const int *f*, const int *c*, const int *salto*)

Verifica la vecindad de la zanahoria para verificar si hay peligro al ubicarse hay.

Parámetros

<i>tablero</i>	puntero a la matriz que representa al tablero
<i>m</i>	entero con la cantidad de filas del tablero
<i>n</i>	entero con la cantidad de columnas del tablero
<i>f</i>	entero que indica la fila a donde estara ubicada la zanahoria
<i>c</i>	entero que indica la columna a donde estara ubicada la zanahoria
<i>salto</i>	entero, si valor es 1 verifica la primera vecindad, si valor es 2 verifica la segunda vecindad

6.6. Referencia del Archivo main.c

Juego de la Zanahoria.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <string.h>
#include <math.h>
#include <time.h>
#include <my_function.h>
#include <zanahoria.h>
#include <memoria.h>
#include <colores.h>
#include <printascii.h>
```

Funciones

- `int main (int argc, char **argv)`

6.6.1. Descripción detallada

Juego de la Zanahoria.

Autor

JesusGoku

Fecha

22/04/2012

Versión

0.1

Índice alfabético

Bool
 my_function.h, 28

Cadenas De Color y Estilo, 8, 10

cargarPartida
 zanahoria.h, 35

cargarRanking
 zanahoria.h, 35

Codigo de Movimiento, 14

Codigos Funciones de Colores.h, 7

colores.h
 fijarColorEstilo, 24
 fijarColorFondo, 24
 fijarColorTexto, 24
 fijarColorTextoEstilo, 24
 fijarColorTextoFondo, 24
 fijarColorTextoFondoEstilo, 24

Configuracion del Ranking, 18

Configuracion para los trampolines, 17

Configuracion Tablero, 11

Constantes con el aumento de puntaje, 15

ejecutarMovimientoConejos
 zanahoria.h, 36

ejecutarMovimientoZanahoria
 zanahoria.h, 36

ejecutarTeletransportacion
 zanahoria.h, 37

Elementos de tablero., 12

fijarColorEstilo
 colores.h, 24

fijarColorFondo
 colores.h, 24

fijarColorTexto
 colores.h, 24

fijarColorTextoEstilo
 colores.h, 24

fijarColorTextoFondo
 colores.h, 24

fijarColorTextoFondoEstilo
 colores.h, 24

guardarPartida
 zanahoria.h, 37

guardarRanking
 zanahoria.h, 37

include/colores.h, 21

include/memoria.h, 25

include/my_function.h, 27

include/printascii.h, 30

include/zanahoria.h, 32

ingresarRanking
 zanahoria.h, 37

inicializarRanking
 zanahoria.h, 38

itemRanking, 19

liberarMemoriaMatriz
 memoria.h, 26

liberarMemoriaMatrizCaracter
 memoria.h, 26

liberarMemoriaMatrizDos
 memoria.h, 26

liberarMemoriaMatrizEntera
 memoria.h, 26

main.c, 41

memoria.h
 liberarMemoriaMatriz, 26
 liberarMemoriaMatrizCaracter, 26
 liberarMemoriaMatrizDos, 26
 liberarMemoriaMatrizEntera, 26
 pedirMemoriaMatriz, 26
 pedirMemoriaMatrizCaracter, 27
 pedirMemoriaMatrizDos, 27
 pedirMemoriaMatrizEntera, 27

mostrarRanking
 zanahoria.h, 38

mostrarRankingDestacado
 zanahoria.h, 38

Movimientos de juego., 13

my_function.h
 Bool, 28
 mygets, 29
 pausaMensaje, 29
 pedirCadena, 29
 preguntayn, 29
 println, 29
 redondeoEntero, 30

mygets
 my_function.h, 29

pausaMensaje
 my_function.h, 29

pedirCadena
 my_function.h, 29

pedirConejosIniciales
 zanahoria.h, 38

pedirDimensionTablero
 zanahoria.h, 38

pedirMemoriaMatriz
 memoria.h, 26

pedirMemoriaMatrizCaracter
 memoria.h, 27

pedirMemoriaMatrizDos
 memoria.h, 27

pedirMemoriaMatrizEntera
 memoria.h, 27

pedirSiguienteMovimiento
 zanahoria.h, 39

posicionZanahoria
 zanahoria.h, 39

preguntayn
 my_function.h, 29

printLuminoso
 printascii.h, 31

printMasMenos
 printascii.h, 31

printMsjError
 printascii.h, 31

printMsjErrorPausa
 printascii.h, 31

printMsjInfo
 printascii.h, 31

printMsjInfoPausa
 printascii.h, 32

printMsjOk
 printascii.h, 32

printMsjOkPausa
 printascii.h, 32

printascii.h
 printLuminoso, 31
 printMasMenos, 31
 printMsjError, 31
 printMsjErrorPausa, 31
 printMsjInfo, 31
 printMsjInfoPausa, 32
 printMsjOk, 32
 printMsjOkPausa, 32

println
 my_function.h, 29

redondeoEntero
 my_function.h, 30

tablero_ini
 zanahoria.h, 39

tablero_pretty_view
 zanahoria.h, 39

tablero_view
 zanahoria.h, 40

Tasas de aumentos para las etaoas, 16

ubicarConejosIniciales
 zanahoria.h, 40

ubicarTrampolines
 zanahoria.h, 40

ubicarZanahoriaInicial
 zanahoria.h, 40

verificarPrimeraVecindadZanahoria
 zanahoria.h, 40

verificarSegundaVecindadZanahoria
 zanahoria.h, 41

verificarVecindadZanahoria
 zanahoria.h, 41

zanahoria.h
 cargarPartida, 35
 cargarRanking, 35
 ejecutarMovimientoConejos, 36
 ejecutarMovimientoZanahoria, 36
 ejecutarTeletransportacion, 37
 guardarPartida, 37
 guardarRanking, 37
 ingresarRanking, 37
 inicializarRanking, 38
 mostrarRanking, 38
 mostrarRankingDestacado, 38
 pedirConejosIniciales, 38
 pedirDimensionTablero, 38
 pedirSiguienteMovimiento, 39
 posicionZanahoria, 39
 tablero_ini, 39
 tablero_pretty_view, 39
 tablero_view, 40
 ubicarConejosIniciales, 40
 ubicarTrampolines, 40
 ubicarZanahoriaInicial, 40
 verificarPrimeraVecindadZanahoria, 40
 verificarSegundaVecindadZanahoria, 41
 verificarVecindadZanahoria, 41