Juego Zanahoria

0.1

Generado por Doxygen 1.8.0

Jueves, 10 de Mayo de 2012 10:01:41

Índice general

Capítulo 1

Indice de módulos

1.1. Módulos

Lista de todos los módulos:

Capítulo 2

Índice de estructura de datos

2.1. Estructura de datos

Lista de estructuras con una breve descripción:

Capítulo 3

Indice de archivos

3.1. Lista de archivos

Lista de todos los archivos documentados y con descripciones breves:

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_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

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_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"

4.2.1. Descripción detallada

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

4.3. Configuracion Tablero

'defines'

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

4.3.1. Descripción detallada

Configuracion del tamano maximo y minimo de tablero el tamano minimo es debido a la cantidad minima para que se desarrolle un juego, con al menos dos conejos El tamano maximo esta dado por el espacio que ocuparia el tablero en una consola a pantalla completa (resolucion 1280x800) para que el tablero pudiera caer entero y ser jugable sin hacer scroll

4.4. Configuracion Conejos Iniciales

'defines'

- #define **CONEJOS_INICIALES_MIN** 2
- #define **CONEJOS_INICIALES_MAX_TASA** 0.1

4.4.1. Descripción detallada

Configuracion de la cantidad minima de conejos y la Tasa maxima de conejos en relacion al tamano del tablero

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

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

4.7. Codigo 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. Configuracion para los tampolines

'defines'

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

4.10.1. Descripción detallada

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

4.11. Configuracion del Ranking

'defines'

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

4.11.1. Descripción detallada

Configuracion del tamano maximo del nombre Cantidad de entradas en el ranking y archivo en donde se guardara 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

5.2. Referencia de la Estructura NodoNombre

Estructura Nodo para almacenar un campo de texto.

`#include <lista.h>`

Campos de datos

- char **nombre** [NODO_NOMBRE_MAX_LENGTH+1]
- struct NodoNombre * **sig**

5.2.1. Descripción detallada

Estructura Nodo para almacenar un campo de texto.

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

- include/lista.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"

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 caracteristicas de las terminales ANSI para poder dar color, fondo y estilo a la terminal y asi agregar un poco de riquesa 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

estilo	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

fondo	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

color	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

texto	corresponde a las constantes COLOR_XXX donde XXX puede ser cualquier color de los disponibles
estilo	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

texto	corresponde a las constantes COLOR_XXX donde XXX puede ser cualquier color de los disponibles
fondo	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

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

6.2. Referencia del Archivo include/lista.h

Funciones para la implementacion de una lista simplemente enlazada.

Estructuras de datos

- struct NodoNombre

  Estructura Nodo para almacenar un campo de texto.

'defines'

- #define NODO_NOMBRE_MAX_LENGTH 50

  Largo maximo de la cadena contenida por el campo nombre en el nodo.

'typedefs'

- typedef struct NodoNombre TipoNodoNombre

  Estructura Nodo para almacenar un campo de texto.

Funciones

- TipoNodoNombre * lista_vacia ()

  Inicializa la lista a NULL.
- int es_lista_vacia (TipoNodoNombre *lista)

  Verifica si la lista esta vacia.
- TipoNodoNombre * insertar_por_cabeza (TipoNodoNombre *lista, char *cadena)

  Inserta un nodo a la cabeza.
- int longitud_lista (TipoNodoNombre *lista)

  Devuelve la cantidad de nodos de la lista.
- TipoNodoNombre * insertar_por_cola (TipoNodoNombre *lista, char *cadena)

  Inserta un nodo a la cola de la lista.
- TipoNodoNombre * borar_cabeza (TipoNodoNombre *lista)

  Borra el primer nodo de la lista.
- TipoNodoNombre * borrar_cola (TipoNodoNombre *lista)

  Borra el ultimo elemento de la lista.
- int pertenece (TipoNodoNombre *lista, char *cadena)

  Verifica si un valor pertenece a la lista.
- TipoNodoNombre * borrar_primera_ocurrencia (TipoNodoNombre *lista, char *cadena)

  Borra la primera ocurrencia que coincida con cadena.
- TipoNodoNombre * borrar_valor (TipoNodoNombre *lista, char *cadena)

  Borra todos los nodos que contengan cadena.
- TipoNodoNombre * insertar_en_posicion (TipoNodoNombre *lista, int pos, char *cadena)

  Inserta un nodo en una posicion especificada.
- void modificar_valor_posicion (TipoNodoNombre *lista, int pos, char *cadena)

  Modifica el valor de una posicion.
- TipoNodoNombre * concatenar_listas (TipoNodoNombre *a, TipoNodoNombre *b)

  Concatena dos listas de nodos y devuelve una nueva lista.
- TipoNodoNombre * liberar_lista (TipoNodoNombre *lista)

  Elimina todos los nodos de la lista.

6.2.1. Descripción detallada

Funciones para la implementacion de una lista simplemente enlazada. Funciones para trabajar con una lista enlazada simple de un campo de texto de tamano NODO_NOMBRE_MAX_LENGTH

Versión

0.1

Fecha

08/05/2012

Autor

JesusGoku

6.2.2. Documentación de las funciones

6.2.2.1. TipoNodoNombre* borrar_cabeza (TipoNodoNombre * *lista*)

Borra el primer nodo de la lista.

Parámetros

lista	puntero al primer nodo de la lista

Devuelve

puntero al nuevo primer elemento de la lista

6.2.2.2. TipoNodoNombre* borrar_cola (TipoNodoNombre * *lista*)

Borra el ultimo elemento de la lista.

Parámetros

lista	puntero al primer elemento de la lista

Devuelve

Retorna el puntero lista o NULL en caso de haber un solo nodo

6.2.2.3. TipoNodoNombre* borrar_primera_ocurrencia (TipoNodoNombre * *lista,* char * *cadena*)

Borra la primera ocurrencia que coincida con cadena.

Parámetros

lista	puntero al primer nodo de la lista
cadena	puntero a la cadena que se desea eliminar

Devuelve

Devuelve un puntero al primer nodo de la lista

6.2.2.4. TipoNodoNombre* borrar_valor (TipoNodoNombre * *lista,* char * *cadena*)

Borra todos los nodos que contengan cadena.

Parámetros

lista	puntero al primer nodo de la lista
cadena	puntero a cadena con la cadena a buscar para eliminar nodo

Devuelve

Retorno un puntero al primer nodo de la lista

6.2.2.5. TipoNodoNombre* concatenar_listas (TipoNodoNombre * *a,* TipoNodoNombre * *b*)

Concatena dos listas de nodos y devuelve una nueva lista.

Parámetros

a	puntero a una lista de nodos
b	puntero a una lista de nodos

Devuelve

Retorna un puntero al primer nodo de una nueva lista con los nodos de a y b

6.2.2.6. int es_lista_vacia (TipoNodoNombre * *lista*)

Verifica si la lista esta vacia.

Parámetros

lista	puntero al primer nodo de la lista

Devuelve

Retorna 1 para vacia y 0 para almenos un elemento

6.2.2.7. TipoNodoNombre* insertar_en_posicion (TipoNodoNombre * *lista,* int *pos,* char * *cadena*)

Inserta un nodo en una posicion especificada.

Parámetros

lista	puntero al primer nodo de la lista
pos	entero que indica la posicion en la que se desea ingresar el elemento (0 es antes del primer elemento)
cadena	puntero a la cadena para crear el nuevo nodo

Devuelve

Puntero al primer elemento de la lista

6.2.2.8. TipoNodoNombre* insertar_por_cabeza (TipoNodoNombre * *lista,* char * *cadena*)

Inserta un nodo a la cabeza.

Parámetros

lista	puntero al primer nodo de la lista
cadena	puntero a la cadena con la que se creara el nuevo nodo

Devuelve

Retorna un puntero a la nueva cabeza de la lista

6.2.2.9. TipoNodoNombre* insertar_por_cola (TipoNodoNombre * *lista,* char * *cadena*)

Inserta un nodo a la cola de la lista.

Parámetros

lista	puntero al primer nodo de la lista
cadena	puntero a la cadena con la que se creara el nuevo nodo

Devuelve

Retorna el mismo puntero lista o un puntero al nuevo elemento creada en caso de estar vacia

6.2.2.10. TipoNodoNombre* liberar_lista (TipoNodoNombre * *lista*)

Elimina todos los nodos de la lista.

Parámetros

lista	puntero al primer nodo de la lista

Devuelve

La funcion retorna NULL

6.2.2.11. int longitud_lista (TipoNodoNombre * *lista*)

Devuelve la cantidad de nodos de la lista.

Parámetros

lista	puntero al primer nodo de la lista

Devuelve

entero que representa al numero de nodos de la lista

6.2.2.12. void modificar_valor_posicion (TipoNodoNombre * *lista,* int *pos,* char * *cadena*)

Modifica el valor de una posicion.

Parámetros

lista	puntero al primer nodo de la lista
pos	entero a la posicion que se desea modifica (primer nodo posicion 0)
cadena	puntero a cadena con el valor nuevo para la posicion

6.2.2.13. int pertenece (TipoNodoNombre * *lista,* char * *cadena*)

Verifica si un valor pertenece a la lista.

Parámetros

lista	puntero al primer nodo de la lista
cadena	puntero a la cadena que se desea verificar si es parte de algun nodo

Devuelve

1 si la cadena pertenece a algun nodo, 0 no pertenece

6.3. Referencia del Archivo include/memoria.h

Libreria 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.

6.3.1. Descripción detallada

Libreria 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

6.3.2. Documentación de las funciones

6.3.2.1. void liberarMemoriaMatriz (void ∗∗ *pm*)

Liberar memoria de una matriz.

Parámetros

pm	Puntero a la memoria que se desea liberar

6.3.2.2. void liberarMemoriaMatrizCaracter (char ∗∗ *pm*)

Liberar memoria de una matriz de caracteres.

Parámetros

pm	Puntero a la memoria que se desea liberar

6.3.2.3. void liberarMemoriaMatrizDos (void ∗∗ *pm,* const int *filas*)

Liberar memoria pedida por filas.

Parámetros

pm	Puntero a la memoria que se desea liberar
filas	Cantidad de filas que seran liberadas

6.3.2.4. void liberarMemoriaMatrizEntera (int ∗∗ *pm*)

Liberar memoria de una matriz entera.

Parámetros

pm	Puntero a la memoria que se desea liberar

6.3.2.5. 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

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

Devuelve

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

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

Pedir memoria para una matriz de caracteres.

Parámetros

filas	Filas de la matriz a solicitar
columnas	Columnas de la matriz a solicitar

Devuelve

Puntero a la matriz solicitada

6.3.2.7. 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

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

Devuelve

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

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

Pedir memoria para una matriz de enteros.

Parámetros

filas	Filas de la matriz a solicitar
columnas	Columnas de la matriz a solicitar

Devuelve

Puntero a la matriz solicitada

6.4. 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.

6.4.1. Descripción detallada

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

Versión

0.1

Fecha

22/04/2012

Autor

JesusGoku

6.4.2. Documentación de las enumeraciones

6.4.2.1. enum Bool

Emulacion del tipo Booleano.

6.4.3. Documentación de las funciones

6.4.3.1. 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

cadena	indica donde se guardara la cadena capturada
max	langitud maxima de caracteres a capturar

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

Muestra un mensaje e introduce una pausa.

Parámetros

mensaje	mensaje que se desea mostrar antes de introducir la pausa

6.4.3.3. 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

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

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

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

Parámetros

pregunta	pregunta a mostrar al usuario

Devuelve

respuesta del usuario, 1 para y 0 para n

6.4.3.5. 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

formato	cadena de formato
...	los distintos atributos especificados en formato

Devuelve

cantidad de parametros imprimidos

6.4.3.6. int redondeoEntero (float *num*)

: Redondea un flotante a entero

Parámetros

num	numero que desea ser redondeado

Devuelve

numero redondeado

6.5. 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 printJuegoDeLaZanahoriaLetras ()

  Imprime el logo de bienvenida simplicado.
- 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 mesaje 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 mesaje 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 mesaje de informacion en color marron e introducir una pausa en la ejecucion.

6.5.1. 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

6.5.2. Documentación de las funciones

6.5.2.1. void printLuminoso (char * *cadena*)

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

Parámetros

cadena	puntero al string que se quiere mostrar en el mensaje

6.5.2.2. void printMasMenos (char * *cadena*)

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

Parámetros

cadena	puntero al string que se quiere mostrar en el mensaje

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

Imprime un mensaje de error en color rojo.

Parámetros

msj	puntero a cadena con el mensaje que se desea mostrar

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

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

Parámetros

msj	puntero a cadena con el mensaje a mostrar

6.5.2.5. void printMsjInfo (const char * msj)

Imprime un mensaje de error en color marron.

Parámetros

msj	puntero a cadena con el mensaje que se desea mostrar

6.5.2.6. void printMsjInfoPausa (const char * msj)

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

Parámetros

msj	puntero a cadena con el mensaje a mostrar

6.5.2.7. void printMsjOk (const char * msj)

Imprime un mensaje de error en color verde.

Parámetros

msj	puntero a cadena con el mensaje que se desea mostrar

6.5.2.8. void printMsjOkPausa (const char * msj)

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

Parámetros

msj	puntero a cadena con el mensaje a mostrar

6.6. 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>
#include <lista.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 **CONEJOS_INICIALES_MIN** 2
- #define **CONEJOS_INICIALES_MAX_TASA** 0.1
- #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"
- #define FOLDER_PARTIDAS "partidas/"
- #define FILE_PARTIDA_PREFIX "partida-"
- #define FILE_PARTIDA_EXT ".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 tamprolines aleatoriamente por el tablero.
- void posicionZanahoria (char **tablero, const int filas, const int columnas, int *coordZF, int *coordZC)

  Devuelve las coordenas de la zanahoria en el tablero.
- int pedirSiguienteMovimiento ()

  Pide el siguiemte 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 alatoria del trablero.
- int verificarVecindadZanahoria (char **tablero, const int m, const int n, const int f, const int c, const int salto)

  Verifica la vencindad 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 vencindad 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, char *partida_nombre, const int conejosIniciales, const int conejosVivos, const int nivel, const int puntaje)

  Guarda una partida en un fichero.
- TipoNodoNombre * generar_lista_partidas ()

  Genera una lista con las partidas disponibles en el directorio de partidas.
- void mostrarRanking (ItemRanking *ranking, const int n)

  Muesta 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)

  Ingresa 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.

6.6.1. Descripción detallada

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

Versión

0.1

Fecha

22/04/2012

Autor

JesusGoku

6.6.2. Documentación de los 'defines'

6.6.2.1. #define FILE_PARTIDA_EXT ".dat"

Extension de los archivos de partida

6.6.2.2. #define FILE_PARTIDA_PREFIX "partida-"

Prefijo para las partidas guardas

6.6.2.3. #define FOLDER_PARTIDAS "partidas/"

Carpeta donde se guardas las partidas

6.6.3. Documentación de las funciones

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

Carga una partida desde un fichero.

Parámetros

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

Devuelve

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

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

Carga el ranking desde un archivo.

Parámetros

ranking	puntero a un arreglo de estructura itemRanking que contendra el ranking
n	entero correspondiente al tamano del ranking que se espera en el archivo

Devuelve

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

6.6.3.3. 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

tablero	puntero a la matriz que representa al tablero
tableroCopy	puntero a la matriz que representa a una copia del tablero para poder realizar los movimientos
filas	entero con la cantidad de filas del tablero
columnas	entero con la cantidad de columnas del tablero
conejosVivos	puntero a entero donde se almacena la cantidad de conejos vivos que se disminuira en caso de colision de conejos
puntaje	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 caido sobre la zanahoria, en caso contrario retorna 1

6.6.3.4. 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 inecesario 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 menos que el nivel en que comienzan a aparecer los trampolines O verificar la vecindad, y ya que hasta que se alcanze 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 seguna expresion, osea la funcion que verifica el segundo cuandrante para conocer el valor de verdad de la expresion, dependiendo ahora exclusivamente de ella el valor de la expresion.

Parámetros

mov	entero que representa al movimiento que solicito el usuario
tablero	puntero a la matriz que representa al tablero
filas	entero con la cantidad de filas del tablero
columnas	entero con la cantidad de columnas del tablero
nivel	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

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

Posicion al conejo en una posicion alatoria del trablero.

Parámetros

tablero	puntero a la matriz que representa al tablero
filas	entero con la cantidad de filas del tablero
columnas	entero con la cantidad de columnas del tablero

6.6.3.6. TipoNodoNombre* generar_lista_partidas ( )

Genera una lista con las partidas disponibles en el directorio de partidas.

Va intentando abrir archivos consecutivos que sean posibles partidas guardadas, cuando el primero de ellos no se puede abrir significa que no hay mas.

Devuelve

devulve un puntero a la lista de partidas

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

Guarda una partida en un fichero.

Parámetros

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

6.6.3.8. 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

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

Devuelve

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

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

Ingresa un elemento al ranking.

Parámetros

ranking	puntero a un arreglo de estructura itemRanking que contiene el ranking
n	entero correspondiente al tamano del ranking
elemento	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

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

Inicializa las variables del ranking con valores iniciales.

Parámetros

ranking	puntero a un arreglo de estructura itemRanking que apunta a el ranking
n	entero correspondiente al tamano del ranking

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

Muesta el ranking.

Parámetros

ranking	puntero al arreglo de estructuras que contiene el ranking
n	cantidad de entradas que tiene el ranking

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

Muestra el ranking destacando una poscion.

Parámetros

ranking	puntero a estructura que contiene el ranking
n	entero correspondiente al tamano del ranking
pos	entero con la posicion que se desea destacar

6.6.3.13. 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 numero max 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 avanze el juego la cantidad de conejos ira aumentando.

Tambien se fija la cantidad minina de conejos en 2 ya que de lo contrario seria imposible que hibiera un choque entre ellos.

Parámetros

f	puntero a entero con el numero de filas del tablero de juego
c	puntero a entero con el numero de columnas del tablero de juego
ci	puntero a entero donde se almacenara el numero de conejos iniciales ingresado.

6.6.3.14. 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

f	puntero a entero donde se almacenara el numero de filas
c	puntero a entero donde se almacenara el numero de columnas

6.6.3.15. int pedirSiguienteMovimiento ( )

Pide el siguiemte 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

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

Devuelve las coordenas de la zanahoria en el tablero.

Parámetros

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

6.6.3.17. 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

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

6.6.3.18. 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 tamano de los tableros posibles.

Parámetros

tablero	puntero a la matriz que representa al tablero
filas	entero con la cantidad de filas del tablero
columnas	entero con la cantidad de columnas del tablero

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

Muestra el tablero en pantalla.

Parámetros

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

6.6.3.20. void **ubicarConejosIniciales** (char ∗∗ *tablero,* const int *m,* const int *n,* const int *conejos*)

Ubica una cierta cantidad de conejos aleatoriamente por el tablero.

Parámetros

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

6.6.3.21. void **ubicarTrampolines** (char ∗∗ *tablero,* const int *m,* const int *n,* const int *trampolines*)

Ubicar los tamprolines aleatoriamente por el tablero.

Parámetros

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

6.6.3.22. void **ubicarZanahoriaInicial** (char ∗∗ *tablero,* const int *f,* const int *c*)

Ubica a la zanahoria en el centro del escenario.

Parámetros

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

6.6.3.23. 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

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

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

Verifica la segunda vencindad en caso de peligros.

Parámetros

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

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

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

Parámetros

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

6.7. 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>
#include <lista.h>
```

'defines'

- #define **IS_UNIX** 1

Funciones

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

6.7.1. Descripción detallada

Juego de la Zanahoria.

Autor

JesusGoku

Fecha

22/04/2012

Versión

0.1

Índice alfabético