**LABERINTOS**

CASTELLANO, CYNTHIA SOLEDAD

DELGADO MARTIN SEGUNDO MANUEL

INDICE

MANUAL DE USO…....................................................................3

CABECERA DEL ARCHIVO............................................................4

DESCRIPCION DE FUNCIONES CREADAS.....................................5

Generador del laberinto....................................................5

Juego..................................................................................7

CONCLUSIONES...........................................................................8

BIBLIOGRAFIA/WEBGRAFIA........................................................8

***MANUAL DE USO***

La primera aplicación que brindamos se trata de un generador de laberintos, con la cual al ejecutarla, se podrá generar un laberinto de manera aleatoria en el cual los 0 representan las paredes y los 1 los senderos. Quedará a disposición del usuario decidir si desea guardar el laberinto generado para su posterior juego.

La segunda aplicación que presentamos aquí lee el laberinto generado en la primera aplicación y permite jugarlo.

Se podrá recorrer el laberinto con las teclas de dirección, dirigiendo el cursor hasta la salida. Al llegar a la salida se entiende como ganado el juego.

***CABECERA DEL ARCHIVO***

La cabecera del archivo binario en el que guardamos el laberinto generado cuenta con los siguientes parámetros :

nombre: Donde hace referencia al nombre del archivo.

posx: Es la posición inicial en x(columnas) del cursor al momento de generar el laberinto.

posy: Es la posición inicial en y(filas) del cursor al momento de generar el laberinto.

endx: Es la posición final en x del cursor ya generado el laberinto.

endy: Es la posición final en y del cursor ya generado el laberinto.

tamanio: tamaño de la matriz usada para generar el laberinto.

***DESCRIPCION DE FUNCIONES CREADAS:***

**GENERADOR DE LABERINTO:**

* Inicializar(): Esta función coloca en todos las filas y

columnas de la matriz el elemento 0, luego a las paredes de cada elemento las clasifica “true” indicando así que existen y a cada elemento le coloca “false” indicando que no fue visitado. Estas acciones las realiza a través de los parámetros declarados en la estructura de tipo “elemento”.

Luego recorre la matriz de tamaño [TAM-1] \* [TAM-1], siendo TAM el tamaño de la matriz, y a las paredes de los elementos de los “bordes” las indica “false” para que así el al generarse el laberinto no pueda romper los bordes.

* ClearScreen(): Función que limpia la pantalla.
* Dibujar(): Dibuja el elemento indicado a través del parámetro mostrar de la estructura de tipo elemento.
* Caminos(): Es la función generadora del camino, esta función lo primero que hace es controlar si el tamaño de la matriz es par o no, si resulta par el generador inicia el camino desde la posición lab[1][TAM-2] , si es impar lo hace desde el medio del laberinto para asi no tener problema con los demás controles.

Al parámetro elementos visitados lo coloca en 1 ya que el inicio del laberinto ocupa un elemento y calcula el total de elementos disponibles para generar el camino.

Para generar el camino controla que mientras el total de elementos visitados sea menor que el total de elementos disponibles realice los siguientes controles:

Al estar posicionado en un elemento disponible controla que si el elemento de dos lugares arriba exista, que exista la pared de arriba, y que exista la pared del elemento de dos lugares abajo; ídem para abajo, izquierda y derecha, según el valor del parámetro aleatorio de la aplicación, controlando así que al generarse el camino no queden huecos y que el camino no sea demasiado ancho, ya que el camino solo ocupara un lugar de un elemento. Al mismo tiempo se guarda la última posición del último elemento generado en una pila y también se suman los elementos visitados cada vez que se cumplen las condiciones y se genera el camino.

Si no se cumple ninguna condición elimina de la pila el primer elemento y limpia la pantalla liberando así la última posición pudiendo ser visitada luego.

Al finalizar coloca el número 3 en la salida y el número 4 en la entrada para así reconocerlas más fácilmente.

* Guardar(): Guarda el laberinto generado en un archivo binario, guardando también los elementos correspondientes de la cabecera.

**JUEGO:**

* Abrir(): abre el archivo binario y muestra el laberinto cambiando los caracteres que indicaban caminos por espacios, las paredes por el carácter 177 del código ASCII, la entrada por espacio y la salida por el carácter 175 del código ASCII.
* Gotoxy(): posiciona el cursor según las coordenadas que se le indica.
* Jugar(): mueve el cursor de acuerdo a la tecla presionada siempre y cuando se presionen teclas de selección hasta llegar al final del laberinto.

**CONCLUSIONES**

Nosotros al recibir la propuesta del proyecto nos entusiasmamos y nos pusimos manos a la obra, tuvimos una primer prueba que generaba el laberinto pero al no realizar los controles necesarios se generaban huecos sin salida en los lugares por donde no pasaba el sendero haciéndolo así muy fácil de resolver. Luego fuimos descubriendo los controles que se deben hacer para evitar estos problemas , investigando pudimos generar un laberinto basándonos en el método de búsqueda en profundidad . El único problema es que nuestro generador solo admite matrices de tamaño impar, ya que al indicar un tamaño par se rompen los bordes al generar el camino(por el avance de 2 en 2 elementos).

Fue una propuesta muy productiva y una hermosa experiencia ya que tuvimos la ayuda de toda la cátedra y asi pudimos lograr nuestro objetivo.

**BIBLIOGRAFIA/ WEBGRAFIA**

- Wikipedia.

- Material de Clases.