Universidad Autónoma de San Luis Potosí

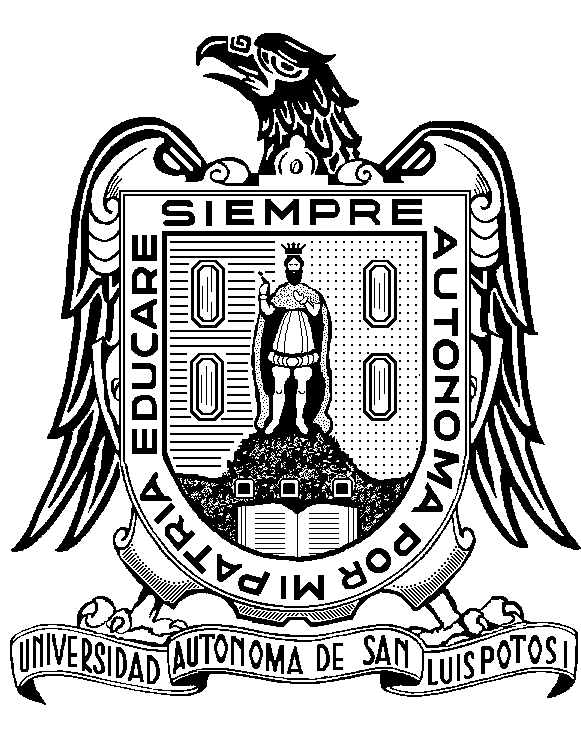
Facultad de Ingeniería

Área de Computación e Informática

Estructuras de Datos y Algoritmos B

VANGUARD

Manual de Programación



Ing. Raymundo Antonio Gonzales Grimaldo

Luis Alberto Díaz Villanueva

Índice

1.-Introducción………...…………………………….……………………………………………….4

2.-Objetivos……………………………...……….…………………………………………………..4

3.-Aplicaciones………...……………….…………………………………………………..….…….4

4.-Librerias………...…………………….……………………………………………………….......5

4.1 Constantes……………………………….………………………………………………….......5

4.1.2 Constantes definidas para el escenario…….………………………….……………….....5

4.1.3 Constantes definidas para las imágenes ….………………………………………….......5

4.1.5 Movimientos para las diagonales ………….…………………………….……...……...….5

5.-Estructuras utilizadas……………………………..……………………………………………...6

5.1 estructura para las imágenes…………………….…………….……………………………...6

5.2 Estructura de las balas en el juego……………….………………………………...………...6

5.3 Estructura lista de la estructura de balas…………………………………………………….6

5.4 Estructura del enemigo en el juego………………………………….………………………..6

5.5 Estructura de la lista de enemigos………………………………….…………………….…..6

5.6 Estructura del nodo con el que se crea la malla…………………………………………….7

5.7 Estructura del jugador…………………………………………………………………………..8

5.8 Estructura de los obstaculos…………………………………………………………………...8

7.-Funciones…………………………………………………………………………………………9

7.1 Funcion main…………………………………………………………………………………....9

7.2.-Menu…………………………………………………………………………………………….9

7.3 funcion Jugar….………………………………………………….……………………………..9

7.4 Funcion juego………………………………………….………………………........................10

7.5 Funcio Movimiento a la derecha……………..…………...……………………………….….10

7.6 Mueve izquierda……………………..……………………………………..……………….…..10

7.7 Funcion mueve abajo……………………………………….………………………………….10

7.8 Funcion mueve arriba……………………………..…………………………………………...10

7.9 Lee Tecla………………………………….………………………………………………….….11

7.1.1 Funcion Crea Nodo………….……………………………………………………………….11

7.1.2 Genera nodos……….………………………………………………………………………..11

7.1.3 Funcion lee imagen………………………………………..…………………..…………….11

7.1.4 Dibuja imagen...............................................................................................................11

7.1.5 Borrar imagen...............................................................................................................11

7.1.7 Inicializa Jugador……………………………………….…………………………………….11

7.1.8 Inicializa Obstaculos…………………………..…..…………………………………………11

7.1.9 Posicion del jugador…………………………………………...……………………………11

7.2.1 Crea enemigo…………………………………………...……………………………………12

7.2.2 Mueve Enemigos………………………………………………………………………….....12

7.2.3 Mueve Enemigo…………………………………………………………………………...…12

7.2.4 Crea Enemigos……………………………………………………………………………….12

7.2.5 Cuenta Lista de enemigos……………………………………….…………..……………...13

7.2.6 Explocion…………………………………………………………..…………..……………...13

7.2.7 Elimina Enemigo…………………………………………………………..………………....13

7.2.8 inicializa Balas………………………………………………………………………………..13

7.2.9 Crea Bala………………………………………………..……….…………………………..14

7.3.1 Mueve Balas………………………………….………………….………………………......14

7.3.2 Mueve Balas………………………..…………………………….………………………….14

7.3.3 Dispara…………………….……………………………………..…………………………..14

7.3.4 Elimina Balas…………………………………………………..………….…………………14

7.3.4 Elimina Balas y Enemigos……………………………………...………………………….14

7.3.5 Elimina Lista de balas………………………………………..………………….………….14

7.3.6 Esenario………………………………………………………..…………………………….15

7.3.7 Interios…………………………………………………………..……………………………15

7.3.8 Mouse…………………………………………………………………………….…………..15

7.3.9 Pause…………………………………………………………………….…….……..……...16

7.4.1 Lee texto………………………………………………………………….…..….…………..16

7.4.2 Escoje tu nave………………………………………………………………………..……..16

7.4.3 Perdiste……………………………………………………..….…………………….……...16

7.4.4 Ayuda……………………………………………...…………….…………………..……….17

7.4.5 Guardar records…………………………..……………………..……………….………….18

4.4.6 Imprime records……………….………………………………...…………….…………….18

5 Programa completo………………………………………………………………………..…….18

Introducción

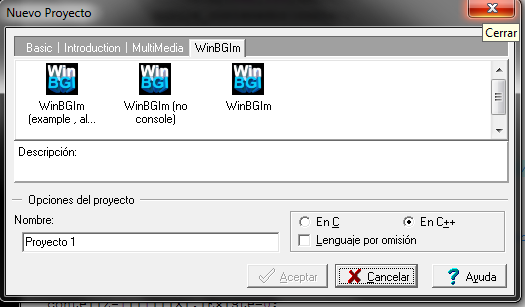
El juego que se desarrolló para la materia de estructuras de datos y algoritmos "B", se estuvo realizando en el lenguaje de programación C mediante el compilador Devc++.

Este juego consiste en una nave que viaja por el espacio esquivando extraterrestres y asteroides y todo lo que pueda surgir en el camino, minetras se desarroya el jugeo puedes recoger diamantes y herramientas que te serán útiles en los niveles siguientes.

* Para este programa se utilizó el compliadordevc++



* Y la librería winBGlm



Objetivos:

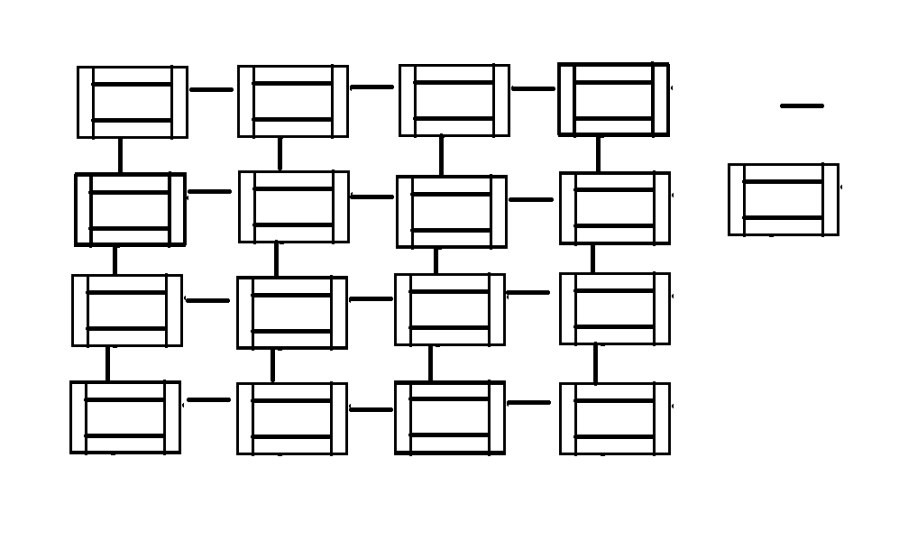
El objetivo del programa es utilizar las habilidades aprendidas durande el curso de estructuras de datos y algoritmos B, para así desarrollar nuevas habilidades de lógica.

Aplicaciones:

Este proyecto funciona con el teclado y el mouse para jugar el usuario presiona una tecla la cual será reconocida por el programa y será comparada para así realizar las acciones que tiene asignada cada tecla.

Utilizo también archivos donde leo las ayudas del programa y para escribir los records de los jugadores.

La parte mas importante del programa se desarrollo dentro de una malla enlazada que fue creada con estructuras llamadas nodos. Cada nodo tiene 8 apuntadores que enlaza, arriba, abajo, izquierda y derecha, y las diagonales.



Nodo

Apuntador isq

Apuntador abaj

Apuntador ant

Apuntador arib

arr

**4.-LIBRERIAS**

|  |  |  |
| --- | --- | --- |
| #include <graphics.h> |  |  |
| #include<string.h> |  |  |
| #include<windows.h> |  |  |
| #include<mmsystem.h> |  |  |
| #include<stdlib.h> |  |  |
|  |  |  |

**4.1 Constantes**

|  |  |  |
| --- | --- | --- |
| 4.1.2 Constantes definidas para el escenario |  |  |
| #define NX 30 | Constante para el numero de nodos (escenario) en X |  |
| #define NY 20 | Constante para el numero de nodos (escenario) en Y |  |
| #define pantallaX 900 | Constante para el tamaño del escenario en x |  |
| #define pantallaY 600 | Constante para el tamaño del escenario en Y |  |

|  |  |  |
| --- | --- | --- |
| 4.1.3 Constantes definidas para las imágenes |  |  |
| #define IMAGJUG 14 | Numero de imágenes para el jugador |  |
| #define IMAGOBST 3 | Numero de obstáculos diferentes |  |
| #define IMAGENEM 5 | Numero de imágenes para enemigos |  |
| #define IMAGOBJ 5 | Numero de imágenes para objetos |  |

|  |  |
| --- | --- |
| 4.1.4 Constantes definidas para las teclas |  |
| #define ENTER 13 | Corresponde a la tecla enter |
| #define BACKSPACE 8 | Corresponde a la tecla espaciado |
| #define ESC 27 | Corresponde a la tecla Esc |
|  |  |

|  |  |  |
| --- | --- | --- |
| 4.1.5 Movimientos para las diagonales |  |  |
| #define O 79 | Diagonal Superior Derecha |  |
| #define L 76 | Diagonal inferior izquierda |  |
| #define A 65 | Diagonal superior izquierda |  |
| #define D 68 | Diagonal inferior derecha |  |

5.-Estructuras utilizadas

**5.1 estructura para las imágenes**

|  |
| --- |
| Estructura que tiene las imágenes utilizadas en el juego |

**typedef struct** {

**char** nomarch[100];

**int** pX, pY;

**int** \*\*pixels;

}IMAGEN;

|  |  |
| --- | --- |
| char nomarch[100]; | Constante para el nombre del archivo |
| int pX, pY; | Enteros para el numero de píxeles de la imagen |
| int \*\*pixels; | Arreglo de pixeles(colores) |

**5.2 Estructura de las balas en el juego**

**typedef struct bal**{

**int**  arma;

**struct** **nod** \*Posbala;

**struct** **bal** \*sig,\*ant;

**int** iNum;

}BALA;

|  |  |
| --- | --- |
| int arma; | Entero que almacena el tipo de arma actual del jugador |
| struct nod \*Posbala; | Apuntador a la malla de la bala |
| struct bal \*sig,\*ant; | Apuntadores a la estructura bala atrás y adelante |
| int iNum; | Entero que almacena el numero de pasos que da la bala |

**5.3 Estructura de lista de balas**

**typedef struct**{

BALA \*prim,\*ult;

IMAGEN imgenem[3];

}LISTBALAS;

|  |  |
| --- | --- |
| BALA \*prim,\*ult; | Apuntadores a la primera bala y a la ultima de la lista |
| IMAGEN imgenem[1]; | Arreglo de imágenes para las balas |

**5.4 Estructura del enemigo en el juego**

**typedef struct** enem{

**int** tipo;

**struct** **nod** \*Posenemigo;

**struct** enem \*sig, \*ant;

**int** iNum;

}ENEMIGO;

|  |  |
| --- | --- |
| int tipo; | Entero que almacena el tipo del enemigo |
| struct nod \*Posenemigo; | Apuntador a la malla |
| struct enem \*sig, \*ant; | Apuntadores a anterior sig del nodo enemigo |
| int iNum; | Número de enemigos |

**5.5 Estructura de la lista de enemigos**

**typedef struct**{

ENEMIGO \*prim,\*ult;

IMAGEN imgenem[NIE];

}LISTENEM;

|  |  |
| --- | --- |
| ENEMIGO \*prim,\*ult; | Apuntadores a el principio de la lista y al fnal de la lista enemigos |
| IMAGEN imgenem[NIE]; | Imágenes para los enemigos |

**5.6 Estructura del nodo con el que se crea la malla**

**typedef struct** nod{

**struct nod** \*ant,\*sig,\*arriba,\*abajo;

**struct nod** \*dsi, \*dsd\*,dii, \*did;

**int** coorX, coorY;

**int** obstaculo;

**int** objeto;

BALA \*balas;

ENEMIGO \*enemigo;

}NODOD;

|  |  |
| --- | --- |
| struct nod \*ant,\*sig,\*arriba,\*abajo; | Apuntadores que enlazan los nodos arriba,abajo,anterior,siguiente |
| struct nod \*dsi, \*dsd\*,dii, \*did; | Apuntadores que enlazan a los nodos en forma de diagonales |
| int coorX, coorY; | Enteros que almacenan las coordenadas de cada nodo en la malla |
| int obstaculo; | Entero que almacena un obstáculo(el nodo puede tener o no obstáculo) 0 si no hay mayor que cero si hay |
| int objeto; | Entero que almacena un objeto(puntos o armas) 0 si no hay, mayor que cero si hay |
| BALA \*balas; | Apuntador a enemigo... NULL= no hay enemigo en el nodo |
| ENEMIGO \*enemigo; | Apuntador a enemigo... NULL= no hay enemigo en el nodo |

**5.7 Estructura del jugador**

**typedef struct**{

IMAGEN arr[NumIJ];

NODOD \*Pos;

**char** cNombre[15];

**int** iPuntos;

}JUGADOR;

|  |  |
| --- | --- |
| IMAGEN arr[NumIJ]; | Numero de imágenes del jugador |
| NODOD \*Pos; | Posición del jugador |
| char cNombre[15]; | Nombre del jugador |
| int iPuntos; | Total de puntos del jugador |

**5.8 Estructura de los obstáculos**

**typedef struct**{

IMAGEN obstimg[NOBST];

IMAGEN objetosimg[OBJ];

}OBSTACULO;

|  |  |
| --- | --- |
| IMAGEN obstimg[NOBST]; | Numero de imágenes de los obstáculos |
| IMAGEN objetosimg[OBJ]; | Numero de imágenes de los objetos |

**7.-Funciones**

**7.1 Función main();**

**int** main()

|  |  |
| --- | --- |
| initwindow(1200,670, "Vanguard" ); | Abre la ventana de gráficos |
| menu(); | Llama a la función menú |
| closegraph( ) ; | cierra los graficos |

**7.2.-Menu**

**void** menu(){

|  |
| --- |
| Se genera una ambientación de juego con una imagen y rectángulos e impresiones de texto |

**7.3 Funcion Jugar()**

**void** jugar()

|  |
| --- |
| Funcion que inicializa los obstáculos los enemigos y a el jugador y manda llamar a la función principal en donde se desarolla el juego también el tamaño de los nodos. |

**7.4 Función juego**

**void** Juego(int tamXNod,int tamYNod, JUGADOR J, LISTENEM le,LISTBALAS balas ,NODOD \*Principal,NODOD \*PosJug,OBSTACULO obst)

|  |
| --- |
| Funcion principal en donde se desarolla el juego, en esta función manda llamar a la gran mayoría de las funciones |

|  |  |
| --- | --- |
| OBSTACULO obst | Estructura de los obstáculos, se envía para poder dibujar un obstáculo durante el juego |
| int tamXNod | Tamaño de el nodo en X |
| int tamYNod | Tamaño del nodo en Y |
| JUGADOR J | Estructura del jugador, es enviada para manipular el jugador durante el juego |
| LISTENEM le | Estructura de la lista de Enemigos, permite crear una lsta de enemigos con diferentes imágenes y tipos |
| LISTBALAS balas | Estructura de la lista de balas, permite crear una lsta de balas con diferentes imágenes y tipos |
| NODOD \*Principal | Nodo que apunta a el primer nodo de la malla es enviado para reorrer la malla |
| NODOD \*PosJug | Posicion del jugador con respecto a la malla |

**7.5 Función Movimiento a la derecha**

**void** Mueve\_Derecha(int tamXNod, int tamYNod,JUGADOR \*J,int iNave)

|  |
| --- |
| Funcion que mueve la posición del jugador con respecto a la malla, mueve el nodo que apunta a la malla a la derecha |

|  |  |
| --- | --- |
| int tamXNod | Tamaño de el nodo en X |
| int tamYNod | Tamaño de el nodo en Y |
| JUGADOR \*J | La estructura es enviada por referencia para poder ser modificada, el apuntador de la estructura jugador que apunta a la malla es modificada, |
| int iNave | Entero que se resta al total de las imágenes que permite dibujar una imagen especifica de acuerdo a la que se escojio previamente |

7.6 **Función Movimiento a la izquierda**

void Mueve\_Izquierda(int tamXNod, int tamYNod,JUGADOR \*J,int iNave)

|  |
| --- |
| Funcion que mueve la posición del jugador con respecto a la malla, mueve el nodo que apunta a la malla a la derecha |

|  |  |
| --- | --- |
| int tamXNod | Tamaño de el nodo en X |
| int tamYNod | Tamaño de el nodo en Y |
| JUGADOR \*J | La estructura es enviada por referencia para poder ser modificada, el apuntador de la estructura jugador que apunta a la malla es modificada, |
| int iNave | Entero que se resta al total de las imágenes que permite dibujar una imagen especifica de acuerdo a la que se escojio previamente |

7.7 **Función Movimiento a la abajo**

void Mueve\_Abajo(int tamXNod, int tamYNod,JUGADOR \*J,int iNave)

|  |
| --- |
| Funcion que mueve la posición del jugador con respecto a la malla, mueve el nodo que apunta a la malla a la derecha |

|  |  |
| --- | --- |
| int tamXNod | Tamaño de el nodo en X |
| int tamYNod | Tamaño de el nodo en Y |
| JUGADOR \*J | La estructura es enviada por referencia para poder ser modificada, el apuntador de la estructura jugador que apunta a la malla es modificada, |
| int iNave | Entero que se resta al total de las imágenes que permite dibujar una imagen especifica de acuerdo a la que se escojio previamente |

7.8 **Función Movimiento a la arriba**

void Mueve\_Arriba(int tamXNod, int tamYNod,JUGADOR \*J,int iNave)

|  |
| --- |
| Funcion que mueve la posición del jugador con respecto a la malla, mueve el nodo que apunta a la malla a la derecha |

|  |  |
| --- | --- |
| int tamXNod | Tamaño de el nodo en X |
| int tamYNod | Tamaño de el nodo en Y |
| JUGADOR \*J | La estructura es enviada por referencia para poder ser modificada, el apuntador de la estructura jugador que apunta a la malla es modificada, |
| int iNave | Entero que se resta al total de las imágenes que permite dibujar una imagen especifica de acuerdo a la que se escojio previamente |

**7.9 Lee Tecla**

|  |
| --- |
| Función que regresa la tecla precionada |

**7.1.1 Funcion Crea Nodo**

NODOD \*CreaNodo(int i, int j,int tamXNod,int tamYNod)

|  |
| --- |
| Reserva memoria para para el nodo pone los apuntadores en NULL obstaculos y objetos a cero Regresa el nodo creado |

|  |  |
| --- | --- |
| int i | Entero que contiene la posición en x del nodo a crear |
| int j | Entero que contiene la psocion en y del nodo a crear |
| int tamXNod | Tamaño del nodo X |
| int tamYNod | Tamaño del nodo Y |

**7.1.2 Genera nodos**

NODOD \*GeneraNodos(int tamXNod, int tamYNod)

|  |
| --- |
| Funcion que enlasa los nodos en forma de malla con diagonales |

|  |  |
| --- | --- |
| int tamXNod | Entra el tamaño del nodo para poder generarlo y enlazarlo |
| int tamYNod | Entra el tamaño del nodo para poder generarlo y enlazarlo |

**7.1.3 Funcion lee imagen**

int \*\* LeeImagen(char nombre[100],int \*pX,int \*pY)

|  |
| --- |
| Se abre el archive si f es igual a NULL muestra un menzaje de error |

|  |  |
| --- | --- |
| char nombre[100] | Nombre del archivo que se va a leer |
| int \*pX | Numero de pixeles en X |
| int \*pY | Numero de pixeles en Y |

**7.1.4 Dibuja imagen**

void DibujaImagen(int tamXnod,int tamYnod,IMAGEN imgn,int coorX,int coorY)

|  |
| --- |
| Se generan las constantes del tamaño de la imagen con función del tam y la imagen |

|  |  |
| --- | --- |
| int tamXnod | Entra el tamaño del nodo para poder dibujar la imagen con las dimensiones correctas |
| int tamYnod | Entra el tamaño del nodo para poder dibujar la imagen con las dimensiones correctas |
| IMAGEN imgn | Entra la estructura de la imagen para acceder a los piceles y a el arreglo de apuntadores que contiene los puntos de la imagen |
| int coorX | Entra la coordenada en x para poder dibujarla en la posición correcta |
| int coorY | Entra la coordenada en x para poder dibujarla en la posición correcta |

**7.1.5 Borrar imagen**

void borraimagen(int tamXnod, int tamYnod,int coorX,int coorY)

|  |
| --- |
| Pone el estilo de la barra en negro y dibuja una barra del tamaño de la imagen para asi borrarla |

|  |  |
| --- | --- |
| int tamXnod, | Entra el tamaño del nodo para adobtar las dimenciones y poder tapar a la imgen que ya esta por una barra negra |
| int tamYnod | Entra el tamaño del nodo para adobtar las dimenciones y poder tapar a la imgen que ya esta por una barra negra |
| int coorX | Las coordenadas en donde se va a dibujar la barra negra |
| int coorY | Las coordenadas en donde se va a dibujar la barra negra |

**7.1.6 Dibuja obstaculos**

void DibujaObstaculos(NODOD \*inicio,int tamXNod,int tamYNod, OBSTACULO obst)

|  |
| --- |
| Nodo auxiliar que apunta al inicio de la malla |

|  |  |
| --- | --- |
| NODOD \*inicio | El nodo que apunta al inicio de la malla para poder recorrerla |
| int tamXNod | Tmaño del nodo para llamar a la función dibuja imagen y esta necesita este parámetro para funcionar |
| int tamYNod | Tmaño del nodo para llamar a la función dibuja imagen y esta necesita este parámetro para funcionar |
| OBSTACULO obst | La estructura obstaculo que contiene las imágenes de los obstáculos a dibujar |

**7.1.7 Inicializa Jugador**

void inicializaJugador(JUGADOR \*J,NODOD \*Posicion)

|  |
| --- |
| Se crea el nombre del archivo y se lee, cuando encutra un numero espesifico regresa la posición y la asigna a el nodo posición |

|  |  |
| --- | --- |
| JUGADOR \*J | Entra la estructura para poderla apuntar a cualquier nodo |
| NODOD \*Posicion | Entra el nodo principal que recorre la malla |

**7.1.8 Inicializa Obstaculos**

OBSTACULO inicializaObstaculos()

|  |
| --- |
| Se incia el for hasa el numero de imagines que existen de los obstáculos regresa el obstáculo ya inicializado |

**7.1.8 Inicializa Enemigos**

void inicializaEnemigos(LISTENEM \*le)

|  |
| --- |
| inicializa las imágenes de los enemigos con un ciclo |

|  |  |
| --- | --- |
| LISTENEM \*le | La lista de enemigos para poder modificar las imágenes de la estructura |

**7.1.9 Posicion del jugador**

NODOD \*PosJugador(NODOD \*Principal)

|  |
| --- |
| Se crean auxiliares para recorrer la malla y se abre un archivo de texto |

|  |  |
| --- | --- |
| NODOD \*Principal | Para poder modificar la posición del jugador con respecto a la malla |

**7.2.1 Crea enemigo**

ENEMIGO \*CreaEnemigo(int tipo,NODOD \*Pos)

|  |
| --- |
| Se reserva memoria para el nodo de tipo enemigo, el tipo de enemigo y la posicion a la que apunta a la malla |

|  |  |
| --- | --- |
| int tipo | El tipo de enemigo que se va a crear |
| NODOD \*Pos | La posición de dicho enemigo |

**7.2.2 Mueve Enemigos**

void MueveEnemigos(LISTENEM \*le,int tamXNod,int tamYNod)

|  |
| --- |
| Se recorre la lista de enemigos y se llama a la función MueveEnemigo para asi mover cada uno de ellos |

|  |  |
| --- | --- |
| LISTENEM \*le | La lista se recorre con la alluda de un auxiliar |
| int tamXNod | La función MueveEnemigos necesita este paramtero para dibujar a los enemigos |
| int tamYNod | La función MueveEnemigos necesita este paramtero para dibujar a los enemigos |

**7.2.3 Mueve Enemigo**

void MueveEnemigo(ENEMIGO \*enem,int tamXNod,int tamYNod,IMAGEN imgn[NIE])

|  |
| --- |
| Modifica la posición del enemigo con respecto a la malla borra a el enemigo y luego lo dibuja en su nueva posicion |

|  |  |
| --- | --- |
| ENEMIGO \*enem | Entra el enemigo que se va a mover |
| IMAGEN imgn[NIE | La imagen del enemigo que se va a mover |
| int tamXNod | Los tamaños de los nodos |
| int tamYNod | Los tamaños de los nodos |

**7.2.4 Crea Enemigos**

void Crea\_enemigos(NODOD \*Principal,LISTENEM \*le,int iRan,int iTipo)

|  |
| --- |
| Se Rcorre la malla y se busca la posición ultima en X y una posición aleatoria en Y y se genera un enemigo |

|  |  |
| --- | --- |
| NODOD \*Principal, | Nodo que apunta a la malla para podre recorrerla |
| LISTENEM \*le | La lista a la cual se insertara un nuevo enemigo |
| int iRan | La posición aleatoria en Y |
| int iTipo | El tipo de enemigo a crear |

**7.2.5 Cuenta Lista de enemigos**

int cuentalista(LISTENEM \*le)

|  |
| --- |
| Recorre la lista de enemigos y los cuenta regresa al numero de enemigos |

|  |  |
| --- | --- |
| LISTENEM \*le | Para poder recorrer la lista de enemigos y contarlos |

**7.2.6 Explocion**

void Explocion(int tamXNod,int tamYNod, JUGADOR J,int \*iVidaMenos){

|  |
| --- |
| Dibuja la imagen del jugador de tipo Esplocion para hacer una animacion |

|  |  |
| --- | --- |
| int tamXNod | Tamaño del nodo ya que se necesita para poder dibujar la imagen |
| int tamYNod | Tamaño del nodo ya que se necesita para poder dibujar la imagen |
| JUGADOR J | Estructura jugador para acceder a sus imagenes |
| int \*iVidaMenos | Se resta una vida |

**7.2.7 Elimina Enemigo**

**void** EliminaEnemigo(LISTENEM \*le,int numero)

|  |
| --- |
| Elimina un enemigo de la lista |

|  |  |
| --- | --- |
| LISTENEM \*le | Modifica la lista eliminando un elemento |
| int numero | Es el numero de enemigo a eliminar |

**7.2.8 inicializa Balas**

**void** inicializaBalas(LISTBALAS \*balas)

|  |
| --- |
| Ciclo para cargar imágenes de las balas |

|  |  |
| --- | --- |
| LISTBALAS \*balas | Para acceder a las imágenes y cargarlas en la estructura |

**7.2.9 Crea Bala**

BALA \*CreaBala(int tipoArma,NODOD \*Pos)

|  |
| --- |
| Se reserve memoria para el nodo de tipo de bala, y la posición a la que apunta a la malla |

|  |  |
| --- | --- |
| int tipoArma | El tipo de arma a crear |
| NODOD \*Pos | La posición de la bala con respecto a la malla |

**7.3.1 Mueve Balas**

void MueveBalas(LISTBALAS \*balas ,int tamXNod,int tamYNod)

|  |
| --- |
| Recorre la lista de balas y manda llamar a otra función que mueve bala por bala de la lista |

|  |  |
| --- | --- |
| LISTBALAS \*balas | La lista de balas se va a modificar |
|  |  |
| int tamXNod | El tamaño de los nodos que se necesita para poder dibujar |
| int tamYNod | El tamaño de los nodos que se necesita para poder dibujar |

**7.3.2 Mueve Balas**

void MueveBala(BALA \*bala,int tamXNod,int tamYNod,IMAGEN imgn[NIE])

|  |
| --- |
| Mueve la bala modificando la posicion |

|  |  |
| --- | --- |
| BALA \*bala, | Se modifica la posicion de la bala con respect a la malla |
| int tamXNod | El tamaño del nodo es necesario para poder dibujar las imagenes |
| int tamYNod | El tamaño del nodo es necesario para poder dibujar las imagenes |
| IMAGEN imgn[NIE] | La imagen de las balas |

**7.3.3 Dispara**

void Dispara(NODOD \*Principal,LISTBALAS \*balas, JUGADOR J,int iArma)

|  |
| --- |
| Recorre la malla hasta que encuetra la posicion del jugador y guarda la posicion siguiente y se crea la bala con dicha posicion |

|  |  |
| --- | --- |
| NODOD \*Principal | Nodo para poder recorrer la malla |
| LISTBALAS \*balas | La lista de balas para poder meidificarla e insertar una nueva |
| JUGADOR J | Para poder encontrar la posicion |
| int iArma | EL tipo de arma con la que se va a crear la bala |

**7.3.4 Elimina Balas**

void EliminaBala(LISTBALAS \*bal,int numero)

|  |
| --- |
| Busca y elimina una bala especifica |

|  |  |
| --- | --- |
| LISTBALAS \*bal, | La lista de balas que se va a modificar al eliminar una de las balas |
| int numero | Numero de la bala a eliminar |

**7.3.4 Elimina Balas y Enemigos**

int EliminaEB(LISTBALAS \*balas,LISTENEM \*le ,int \*num)

|  |
| --- |
| Recorre la lista de balas y pregunta si existe un enemigo en su posicion y elimina enemigo y bala |

|  |  |
| --- | --- |
| LISTBALAS \*balas, | Se va a modificar la lista de balas |
| LISTENEM \*le | Se va a modifical la lista de enemigos |
| int \*num | El numero de eliminaciones |

**7.3.5 Elimina Lista de balas**

void EliminaLista(LISTBALAS \*bal)

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | LISTBALAS \*bal | Elimina toda la lista de balas | |

**7.3.6 Esenario**

void Esenario(int \*f,int \*x,int iDistancia,int iVidas)

|  |
| --- |
| Crea un esenario con la function interior |

|  |  |
| --- | --- |
| int \*f | Se usa para modificar la posicion del esenario |
| int \*x | Se usa para modificar la posicion del esenario |
| int iDistancia | La distancia que se imprime en esta function |
| int iVidas | El numero de las vidas para crear rectangulos que simulen la vida |

**7.3.7 Interios**

void interior(int x1,int y1,int x2,int y2,int cual)

|  |
| --- |
| Esta function acomoda barras en 3 D |

|  |  |
| --- | --- |
| int x1 | Posicion inicial en x |
| int y1 | Posicion inicial en y |
| int x2 | Posicion final en x |
| int y2 | Posicion final en y |
| int cual | Color de las barras |

**7.3.8 Mouse**

int mouse(int \*x,int \*y)

|  |
| --- |
| Guarda la posicion en y y Y del mouse |

|  |  |
| --- | --- |
| int \*x | Se usa para almacenar la posicion del mouse |
| int \*y | **Se usa para almacenar la posicion del mouse** |

**7.3.9 pause**

int iPause()

|  |
| --- |
| Se pausa el juego por el getch() hasta que la tecla sea igual a enter |

**7.4.1 lee texto**

void lee\_texto(int x,int y, char cadena[5])

|  |
| --- |
| Lee y guarda una cadena de texo |

|  |  |
| --- | --- |
| Int x | La posicion en donde se dibuja la introducción de texto |
| Int y | La posicion en donde se dibuja la introducción de texto |
| Char cadena | La cadena que se modifica |

**7.4.2 escoje tu nave**

int EscojeTuNave()

|  |
| --- |
| Se genera un menu en el cual puedes escojer una nave y regresa el numero de la nave elegida |

**7.4.3 perdiste**

int perdiste()

|  |
| --- |
| Se imprime un contador y una imagen con las opciones volver a jugar o salir |

**7.4.4 ayuda**

void ayuda()

|  |
| --- |
| Lee un archive de texto e imprime en pantalla su contenido |

**7.4.5 Guardar records**

void guardarR(char nom\_arch[50],JUGADOR J)

|  |
| --- |
| Crea o modifica un archive binario de texto |

|  |  |
| --- | --- |
| char nom\_arch[50] | El nombre del archivo que se va a abrir |
| JUGADOR J | Estructura que contiene todos los datos del jugador |

**4.4.6 imprime records**

void imprime\_record(char nom\_arch[50])

|  |
| --- |
| Si el archivo no existe muestra un menzaje de error si si existe lee el archivo e imprime en pantalla su contenido |

|  |
| --- |
|  |

**Programa completo**

#include <graphics.h>

#include<string.h>

#include<windows.h>

#include<mmsystem.h>

#include<stdlib.h>

#include<time.h>

#define NX 30

#define NY 20

#define pantallaX 900

#define pantallaY 600

#define IMAGENESJUGADOR 14

#define IMAGOBST 3

#define IMAGENEM 5

#define IMAGOBJ 5

#define IMAGENBAL 3

#define ENTER 13

#define BACKSPACE 8

#define ESC 27

#define O 79

#define L 76

#define A 65

#define D 68

typedef struct {

char nomarch[100];

int pX, pY;

int \*\*pixels;

}IMAGEN;

typedef struct bal{

int arma;

struct nod \*Posbala;

struct bal \*sig,\*ant;

int iNum;

}BALA;

typedef struct{

BALA \*prim,\*ult;

IMAGEN imgenem[IMAGENBAL];

}LISTBALAS;

typedef struct enem{

int tipo;

struct nod \*Posenemigo;

struct enem \*sig, \*ant;

int iNum;

}ENEMIGO;

typedef struct{

ENEMIGO \*prim,\*ult;

IMAGEN imgenem[IMAGENEM];

}LISTENEM;

typedef struct nod{

struct nod \*ant,\*sig,\*arriba,\*abajo;

struct nod \*dsi;

struct nod \*dsd;

struct nod \*dii;

struct nod \*did;

int coorX;

int coorY;

int obstaculo;

int objeto;

BALA \*balas;

ENEMIGO \*enemigo;

}NODOD;

typedef struct{

IMAGEN arr[IMAGENESJUGADOR];

NODOD \*Pos;

char cNombre[15];

int iPuntos;

}JUGADOR;

typedef struct{

IMAGEN obstimg[IMAGOBST];

IMAGEN objetosimg[IMAGOBJ];

}OBSTACULO;

BALA \*CreaBala(int tipoArma,NODOD \*Pos);

void inicializaBalas(LISTBALAS \*balas);

void Dispara(NODOD \*Principal,LISTBALAS \*balas, JUGADOR J,int iArma);

void MueveBalas(LISTBALAS \*balas,LISTENEM \*le,int tamXNod,int tamYNod);

void MueveBala(BALA \*bala,int tamXNod,int tamYNod,IMAGEN imgn[IMAGENBAL]);

void EliminaBala(LISTBALAS \*bal,int numero);

ENEMIGO \*CreaEnemigo(int tipo,NODOD \*Pos);

void inicializaEnemigos(LISTENEM \*le);

void Crea\_enemigos(NODOD \*Principal,LISTENEM \*le,int iRan,int iTipo);

void MueveEnemigos(LISTENEM \*le,int tamXNod,int tamYNod);

void MueveEnemigo(ENEMIGO \*enem,int tamXNod,int tamYNod,IMAGEN imgn[IMAGENEM]);

void EliminaEnemigo(LISTENEM \*le,int numero);

int cuentalista(LISTENEM \*le);

OBSTACULO inicializaObstaculos();

void DibujaObstaculos(NODOD \*inicio,int tamXNod,int tamYNod, OBSTACULO obst);

NODOD \*CreaNodo(int coorX, int coorY);

NODOD \*GeneraNodos(int tamXNod, int tamYNod);

NODOD \*PosJugador(NODOD \*Principal);

void inicializaJugador(JUGADOR \*J,NODOD \*Posicion);

void Mueve\_Derecha(int tamXNod, int tamYNod,JUGADOR \*J,int iNave);

void Mueve\_Izquierda(int tamXNod, int tamYNod,JUGADOR \*J,int iNave);

void Mueve\_Abajo(int tamXNod, int tamYNod,JUGADOR \*J,int iNave);

void Mueve\_Arriba(int tamXNod, int tamYNod,JUGADOR \*J,int iNave);

void menu();

void Juego(int tamXNod,int tamYNod, JUGADOR J, LISTENEM le, LISTBALAS balas,NODOD \*Principal,NODOD \*PosJug,OBSTACULO obst);

void Explocion(int tamXNod,int tamYNod, JUGADOR J,int \*iVidaMenos);

void Esenario(int \*f,int \*x,int iDistancia,int iVidas,int iNivel);

char LeeTecla();

int \*\*LeeImagen(char nombre[],int \*pX,int \*pY);

void DibujaImagen(int tamXnod,int tamYnod,IMAGEN imgn,int coorX,int coorY);

void borraimagen(int tamXnod, int tamYnod,int coorX,int coorY);

int EliminaEB(LISTBALAS \*balas,LISTENEM \*le,int \*num);

void EliminaLista(LISTBALAS \*bal);

void interior(int x1,int y1,int x2,int y2,int cual);

void jugar();

void ayuda();

void imprime\_record(char nom\_arch[50]);

void guardarR(char nom\_arch[50],JUGADOR J);

void lee\_texto(int x,int y, char cadena[5]);

int mouse(int \*x,int \*y);

int iPause();

int perdiste();

int EscojeTuNave();

void musica(char Nombre[50]);

void portada();

int espacio(int iColor,int iMueve);

int cuentalistaB(LISTBALAS balas);

int iSiguiente\_Nivel(int iNivel);

typedef BOOL (\*PlaySoundA\_ptr) (char\*, HMODULE, DWORD);

PlaySoundA\_ptr fn\_PlaySound = 0;

int main()

{

initwindow(1200,670, "Vanguard" );

musica("musicajuego.wav");

portada();

menu();

closegraph( );

}

void menu()

{

char cOp=0;

int pagina=0;

int iX,iX2,iXm,iYm;

iX=50;

iX2=200;

settextstyle(4, 0, 4);

do{

setactivepage(pagina);

cleardevice();

readimagefile("inicio.jpg",20,20,1050,540);

rectangle(iX,350+200,iX2,385+200);

rectangle(48,348+200,202,387+200);

rectangle(218,348+200,372,387+200);

rectangle(388,348+200,542,387+200);

rectangle(558,348+200,712,387+200);

setcolor(YELLOW);

rectangle(0,0,1199,669);

rectangle(10,10,1189,659);

setcolor(15);

settextstyle(3, 0, 4);

outtextxy( 65 ,352+200,"-Jugar" );

outtextxy( 230 ,352+200,"-Ayuda" );

outtextxy( 391 ,352+200,"-Record" );

outtextxy( 570 ,352+200,"-Salir" );

setvisualpage(pagina);

if(pagina==0)

pagina=1;

else

pagina=0;

if(kbhit()){

cOp=getch();

switch(cOp){

case 77: if(iX2!=710){

iX+=170;

iX2+=170;

}

break;

case 75: if(iX!=50){

iX-=170;

iX2-=170;

}

break;

}

if(cOp==13){

switch(iX){

case 50: jugar();

break;

case 220: ayuda();

break;

case 390:imprime\_record("Records.dat");

break;

case 560: cOp=27;

break;

}

}

}

if(iXm>50 && iYm>348+200 && iXm<200 && iYm<387+200){

iX=50;

iX2=200;

}

if(iXm>220 && iYm>348+200 && iXm<370 && iYm<387+200){

iX=220;

iX2=370;

}

if(iXm>390 && iYm>348+200 && iXm<540 && iYm<387+200){

iX=390;

iX2=540;

}

if(iXm>560 && iYm>348+200 && iXm<710&& iYm<387+200){

iX=560;

iX2=710;

}

if(mouse(&iXm,&iYm)){

if(iXm>50 && iYm>348+200 && iXm<220 && iYm<387+200)

jugar();

if(iXm>220 && iYm>348+200 && iXm<390 && iYm<387+200)

ayuda();

if(iXm>390 && iYm>348+200 && iXm<540 && iYm<387+200)

imprime\_record("Records.dat");

if(iXm>560 && iYm>348+200 && iXm<710&& iYm<387+200)

cOp=27;

}

}while(cOp!=27);

}

void jugar()

{

OBSTACULO obst;

obst=inicializaObstaculos();

int x,y,;

setcolor(WHITE);

cleardevice();

int tamXNod, tamYNod;

musica("musicajuego.wav");

LISTENEM le;

LISTBALAS balas;

JUGADOR J;

NODOD \*Principal;

tamXNod=(int)((pantallaX)/NX);

tamYNod=(int)((pantallaY)/NY);

Principal=GeneraNodos(tamXNod,tamYNod);

inicializaEnemigos(&le);

inicializaBalas(&balas);

NODOD \*PosJug=PosJugador(Principal);

inicializaJugador(&J,PosJug);

DibujaObstaculos(Principal,tamXNod,tamYNod,obst);

Juego(tamXNod,tamYNod,J,le,balas,Principal,PosJug,obst);

cleardevice();

}

void Juego(int tamXNod,int tamYNod, JUGADOR J, LISTENEM le,LISTBALAS balas,NODOD \*Principal,NODOD \*PosJug,OBSTACULO obst)

{

int iNave=EscojeTuNave();

lee\_texto(25,65,J.cNombre);

int iArma=0;

char cTecla;

int iExplocion=1;

int iTotal=0;

char cTotal[5];

int iVidas=5;

char cVidas[5];

int iDiamantes=0;

char cDiamantes[5];

int iNotas=0;

char cNotas[5];

int NEliminaciones=0;

char NEL[5];

int iNivel=0;

char cNivel[5];

int iTipoB=0;

char cTipoB[5];

int iDistancia=0;

int iRan=3;

int iNumB=1;

int iNumBalas=0;

char B[5];

int iTiempo=100;

int iTipo=1;

int tiemp;

int MueveEsen=0;

char cPuntos[5];

int enemigo\_eliminado;

int npag=0;

DibujaImagen(tamXNod,tamYNod,J.arr[14-iNave],J.Pos->coorX,J.Pos->coorY);

int En=12;

Esenario(&tiemp,&MueveEsen,iDistancia,iVidas,iNivel);

do{

setactivepage(npag);

cleardevice();

Esenario(&tiemp,&MueveEsen,iDistancia,iVidas,iNivel);

outtextxy(1015,25,J.cNombre);

MueveEnemigos(&le,tamXNod,tamYNod);

MueveBalas(&balas,&le,tamXNod,tamYNod);

itoa(NEliminaciones,NEL,10);

outtextxy(1160,500,NEL);

itoa(iTotal,cTotal,10);

outtextxy(1000,560,cTotal);

itoa(iDiamantes,cDiamantes,10);

outtextxy(1050,520,cDiamantes);

itoa(iNotas,cNotas,10);

outtextxy(1000,540,cNotas);

itoa(iNivel,cNivel,10);

outtextxy(1070,350,cNivel);

itoa(iTipoB,cTipoB,10);

outtextxy(1160,440,cTipoB);

iNumBalas=cuentalistaB(balas);

itoa(iNumB,B,10);

outtextxy(1160,420,B);

itoa(iVidas,cVidas,10);

outtextxy(1070,375,cVidas);

if(iExplocion!=0)

if(npag==0)

DibujaImagen(tamXNod,tamYNod,J.arr[14-iNave],J.Pos->coorX,J.Pos->coorY);

else

DibujaImagen(tamXNod,tamYNod,J.arr[15-iNave],J.Pos->coorX,J.Pos->coorY);

DibujaObstaculos(Principal,tamXNod,tamYNod,obst);

setvisualpage(npag);

if(npag==0)

npag=1;

else

npag=0;

iDistancia+=5;

MueveEsen+=5;

iRan++;

if(iRan==En){

iRan=rand()%11;

Crea\_enemigos(Principal,&le,iRan+4,iTipo);

iRan=0;

}

if(iExplocion==0){

iVidas--;

EliminaLista(&balas);

Explocion(tamXNod,tamYNod,J,&iExplocion);

inicializaJugador(&J,PosJug);

}

enemigo\_eliminado=EliminaEB(&balas,&le,& NEliminaciones);

if(kbhit()){

cTecla=LeeTecla();

switch(cTecla)

{

case KEY\_RIGHT: Mueve\_Derecha(tamXNod,tamYNod,&J,iNave);

break;

case KEY\_LEFT: Mueve\_Izquierda(tamXNod,tamYNod,&J,iNave);

break;

case KEY\_UP: Mueve\_Arriba(tamXNod,tamYNod,&J,iNave);

break;

case KEY\_DOWN: Mueve\_Abajo(tamXNod,tamYNod,&J,iNave);

break;

case 'f': if(iNumBalas<iNumB)

Dispara(Principal,&balas,J,iArma);

break;

case 13:iPause();

break;

}

}

if(J.Pos->enemigo!=NULL){

iExplocion--;

EliminaEnemigo(&le,J.Pos->enemigo->iNum);

}

if(J.Pos->objeto==1){

J.Pos->objeto=0;

iDiamantes++;

}

if(J.Pos->objeto==2){

J.Pos->objeto=0;

iNotas++;

}

if(J.Pos->objeto==3){

J.Pos->objeto=0;

if(iArma<2){

iTipoB++;

iArma++;}

}

if(J.Pos->objeto==4){

J.Pos->objeto=0;

if(iNumB<4);

iNumB++;

}

if(iDistancia==1005){

outtextxy(19,10,"Nivel 1");

iSiguiente\_Nivel(iNivel);

En=10;

iNivel++;

iTipo=2;

}

if(iDistancia==2005){

outtextxy(19,10,"Nivel 2");

iSiguiente\_Nivel(iNivel);

En=9;

iNivel++;

iTipo=3;

}

if(iDistancia==3005){

outtextxy(19,10,"Nivel 3");

iSiguiente\_Nivel(iNivel);

En=7;

iNivel++;

iTipo=4;

}

if(iDistancia==4005){

outtextxy(19,10,"Nivel 4");

iSiguiente\_Nivel(iNivel);

En++;

iNivel++;

iTipo=5;

}

if(iDistancia==5005){

iNivel++;

iSiguiente\_Nivel(iNivel);

cTecla=ESC;

}

if(iVidas==0)

cTecla=perdiste();

if(cTecla==1){

iNivel=0;

iVidas=5;

iArma=0;

iNotas=0;

iDiamantes=0;

NEliminaciones=0;

iNumBalas=0;

iDistancia=0;

}

iTotal=iNotas+iDiamantes+NEliminaciones;

}while(cTecla!=ESC);

J.iPuntos=iTotal;

guardarR("Records.dat",J);

imprime\_record("Records.dat");

}

void Mueve\_Derecha(int tamXNod, int tamYNod,JUGADOR \*J,int iNave)

{

if(J->Pos->sig->sig->sig->sig->sig!=NULL && (J->Pos->sig)->obstaculo==0)

{

borraimagen(tamXNod,tamYNod,J->Pos->coorX,J->Pos->coorY);

J->Pos=J->Pos->sig;

DibujaImagen(tamXNod,tamYNod,J->arr[15-iNave],J->Pos->coorX,J->Pos->coorY);

}

}

void Mueve\_Izquierda(int tamXNod, int tamYNod,JUGADOR \*J,int iNave)

{

if(J->Pos->ant->ant!=NULL && (J->Pos->ant)->obstaculo==0)

{

borraimagen(tamXNod,tamYNod,J->Pos->coorX,J->Pos->coorY);

J->Pos=J->Pos->ant;

DibujaImagen(tamXNod,tamYNod,J->arr[14-iNave],J->Pos->coorX,J->Pos->coorY);

}

}

void Mueve\_Abajo(int tamXNod, int tamYNod,JUGADOR \*J,int iNave)

{

if(J->Pos->abajo->abajo->abajo->abajo->abajo->abajo!=NULL && (J->Pos->abajo)->obstaculo==0)

{

borraimagen(tamXNod,tamYNod,J->Pos->coorX,J->Pos->coorY);

J->Pos=J->Pos->abajo;

DibujaImagen(tamXNod,tamYNod,J->arr[17-iNave],J->Pos->coorX,J->Pos->coorY);

}

}

void Mueve\_Arriba(int tamXNod, int tamYNod,JUGADOR \*J,int iNave)

{

if(J->Pos->arriba->arriba->arriba->arriba->arriba!=NULL && (J->Pos->arriba)->obstaculo==0)

{

borraimagen(tamXNod,tamYNod,J->Pos->coorX,J->Pos->coorY);

J->Pos=J->Pos->arriba;

DibujaImagen(tamXNod,tamYNod,J->arr[16-iNave],J->Pos->coorX,J->Pos->coorY);

}

}

char LeeTecla()

{

char tecla;

tecla=getch();

if(tecla=='\0')

tecla=getch();

return tecla;

}

NODOD \*CreaNodo(int i, int j,int tamXNod,int tamYNod)

{

NODOD \*nuevo;

nuevo=(NODOD \*)malloc(sizeof(NODOD));

if(nuevo){

nuevo->coorX=50+(j\*tamXNod);

nuevo->coorY=50+(i\*tamYNod);

nuevo->ant=NULL;

nuevo->sig=NULL;

nuevo->arriba=NULL;

nuevo->abajo=NULL;

nuevo->dsi=NULL;

nuevo->dsd=NULL;

nuevo->dii=NULL;

nuevo->did=NULL;

nuevo->enemigo=NULL;

nuevo->balas=NULL;

nuevo->obstaculo=0;

nuevo->objeto=0;

}

return (nuevo);

}

NODOD \*GeneraNodos(int tamXNod, int tamYNod)

{

NODOD \*nuevo;

NODOD \*primero;

NODOD \*auxArr=NULL;

NODOD \*auxant=NULL;

NODOD \*auxdiag2=NULL;

NODOD \*auxdiag3=NULL,\*auxd=NULL;

NODOD \*iniciofila,\*iniciodiag=NULL;

int i, j;

for(i=0; i<NY; i++)

{for(j=0; j<NX; j++)

{

nuevo=CreaNodo(i,j,tamXNod,tamYNod);

if(nuevo)

{

if(i==0 && j==0)

{

primero=nuevo;

auxd=nuevo;

}

if(j==0)

{

iniciofila=nuevo;

iniciodiag=nuevo;

nuevo->dsi=nuevo->dii=NULL;

nuevo->ant=auxant;

nuevo->dsd=auxdiag2;

}

if(j==1 && i>0)

{

auxdiag3=auxd;

auxd=auxdiag3->abajo;

}

nuevo->arriba=auxArr;

if(j>0 && i>0)

{

nuevo->dsd=auxdiag2;

nuevo->dsi=auxdiag3;

}

if(auxdiag3 && j!=0)

{

auxdiag3->did=nuevo;

auxdiag3=auxdiag3->sig;

}

if(auxdiag2)

{

auxdiag2->dii=nuevo;

auxdiag2=auxdiag2->sig;

}

if(auxant)

{

auxant->sig=nuevo;

nuevo->ant=auxant;

}

if(auxArr)

{

auxArr->abajo=nuevo;

auxArr=auxArr->sig;

}

auxant=nuevo;

}

}

auxant=NULL;

auxArr=iniciofila;

auxdiag2=iniciodiag->sig;

}

return (primero);

}

int \*\* LeeImagen(char nombre[100],int \*pX,int \*pY)

{

int iX,iY;

int \*\*malla;

FILE \*f;

f=fopen(nombre,"rb");

if(f==NULL)

{cleardevice();

outtextxy((textwidth("ERROR: Archivo no encontrado")/2),400-textheight("ERROR: Archivo no encontrado")/2,"ERROR: Archivo no encontrado");

getch();

exit(1);}

fread(pX,sizeof(int),1,f);

fread(pY,sizeof(int),1,f);

malla=(int\*\*)malloc((sizeof(int\*))\*(\*pX));

for(iX=0; iX<\*pX;iX++)

malla[iX]=(int\*)malloc(sizeof(int)\*(\*pY));

for(iX=0; iX<\*pX; iX++)

{for(iY=0; iY<\*pY; iY++)

{

fread(&malla[iX][iY],sizeof(int),1,f);

}}

fclose(f);

return(malla);

}

void DibujaImagen(int tamXnod,int tamYnod,IMAGEN imgn,int coorX,int coorY)

{

int tamx,tamy,x,y;

tamx=(int) tamXnod/imgn.pX;

tamy=(int) tamYnod/imgn.pY;

for(x=0; x<imgn.pX; x++)

for(y=0; y<imgn.pY; y++)

{

setfillstyle(1,imgn.pixels[x][y]);

bar(coorX+(tamx\*x),coorY+(tamy\*y),coorX+(tamx\*(x+1)),coorY+(tamy\*(y+1)));

}

}

void borraimagen(int tamXnod, int tamYnod,int coorX,int coorY)

{

setfillstyle(1,0);

bar(coorX,coorY,coorX+tamXnod,coorY+tamYnod);

}

void DibujaObstaculos(NODOD \*inicio,int tamXNod,int tamYNod, OBSTACULO obst)

{

NODOD \*aux=inicio;

while(inicio!=NULL)

{

while(aux!=NULL)

{

if(aux->obstaculo!=00)

DibujaImagen(tamXNod,tamYNod,obst.obstimg[aux->obstaculo-1],aux->coorX,aux->coorY);

if(aux->objeto!=0)

DibujaImagen(tamXNod,tamYNod,obst.objetosimg[aux->objeto],aux->coorX,aux->coorY);

aux=aux->sig;

}

inicio=inicio->abajo;

aux=inicio;

}

}

void inicializaJugador(JUGADOR \*J,NODOD \*Posicion)

{

int i;

for(i=0; i<IMAGENESJUGADOR; i++)

{

sprintf(J->arr[i].nomarch,"jugador%d.dso",i);

J->arr[i].pixels=LeeImagen(J->arr[i].nomarch,&J->arr[i].pX,&J->arr[i].pY);

}

J->Pos=Posicion;

}

OBSTACULO inicializaObstaculos()

{

OBSTACULO obst;

int c;

char cad[100];

for(c=0; c<IMAGOBST; c++)

{

sprintf(cad,"obstaculo%d.dso",c+1);

obst.obstimg[c].pixels=LeeImagen(cad,&obst.obstimg[c].pX,&obst.obstimg[c].pY);}

for(c=0; c<IMAGOBJ; c++)

{

sprintf(cad,"puntos%d.dso",c+1);

obst.objetosimg[c].pixels=LeeImagen(cad,&obst.objetosimg[c].pX,&obst.objetosimg[c].pY);}

return obst;

}

void inicializaEnemigos(LISTENEM \*le)

{

int i;

le->prim=NULL;

le->ult=NULL;

for(i=0; i<IMAGENEM; i++)

{

sprintf(le->imgenem[i].nomarch,"enemigo%d.dso",i+1);

le->imgenem[i].pixels=LeeImagen(le->imgenem[i].nomarch,&le->imgenem[i].pX,&le->imgenem[i].pY);

}

}

NODOD \*PosJugador(NODOD \*Principal)

{

FILE \*f;

NODOD \*POSJUG, \*aux, \*auxP;

char cad[50];

int x,Num=2,Cont=0;

int y;

int lectura;

auxP=Principal;

aux=Principal;

sprintf(cad,"escenario%d.txt",1);

f=fopen(cad,"r");

{

rewind(f);

for(y=0; y<NY; y++)

{

for(x=0; x<NX; x++)

{

fscanf(f,"%d",&lectura);

if(lectura==99)

{

if(aux->obstaculo!=1){

POSJUG=aux;

return (POSJUG);

}

}

aux=aux->sig;

}

auxP=auxP->abajo;

aux=auxP;

}

}

return (POSJUG);

}

ENEMIGO \*CreaEnemigo(int tipo,NODOD \*Pos){

ENEMIGO \*nuevo;

nuevo=(ENEMIGO \*)malloc(sizeof(ENEMIGO));

nuevo->ant=NULL;

nuevo->sig=NULL;

nuevo->Posenemigo=Pos;

nuevo->tipo=tipo;

nuevo->iNum=0;

return (nuevo);

}

void MueveEnemigos(LISTENEM \*le,int tamXNod,int tamYNod)

{

ENEMIGO \*aux=le->prim;

while(aux)

{

MueveEnemigo(aux,tamXNod,tamYNod,le->imgenem);

if(aux->Posenemigo->ant->objeto!=0)

aux->Posenemigo->ant->objeto=0;

if(aux->Posenemigo->ant->ant==NULL){

if(le->prim==le->ult){

free(le->prim);

le->prim=le->ult=NULL;

}else{

ENEMIGO \*aux2=le->ult;

le->ult=le->ult->ant;

le->ult->sig=NULL;

aux2->Posenemigo->enemigo=NULL;

free(aux2);

}

}

aux=aux->sig;

}

}

void MueveEnemigo(ENEMIGO \*enem,int tamXNod,int tamYNod,IMAGEN imgn[IMAGENEM])

{

if(enem->Posenemigo->ant!=NULL)

{

borraimagen(tamXNod,tamYNod,enem->Posenemigo->coorX,enem->Posenemigo->coorY);

enem->Posenemigo->enemigo=NULL;

enem->Posenemigo=enem->Posenemigo->ant;

enem->Posenemigo->enemigo=enem;

DibujaImagen(tamXNod,tamYNod,imgn[enem->tipo],enem->Posenemigo->coorX,enem->Posenemigo->coorY);

}

}

void Crea\_enemigos(NODOD \*Principal,LISTENEM \*le,int iRan,int iTipo)

{

ENEMIGO \*nuevoE;

NODOD \*aux, \*auxP;

int x;

int y;

auxP=Principal;

aux=Principal;

for(y=0; y<NY; y++)

{

for(x=0; x<NX; x++)

{

if(x==25 && y==iRan)

{

int iR=rand()%iTipo;

nuevoE=CreaEnemigo(iR,aux);

if(nuevoE)

{

if(le->prim==NULL)

{

le->prim=nuevoE;

le->ult=nuevoE;

le->prim->iNum=1;

}

else{

nuevoE->sig=le->prim;

le->prim->ant=nuevoE;

le->prim=nuevoE;

le->prim->iNum+=le->prim->sig->iNum+1;

if(le->prim->iNum==11)

le->prim->iNum=0;

}

aux->enemigo=nuevoE;

}

}

aux=aux->sig;

}

auxP=auxP->abajo;

aux=auxP;

}

}

int cuentalista(LISTENEM \*le)

{

int i=0;

ENEMIGO \*aux=le->prim;

while(aux!=NULL)

{

i+=1;

aux=aux->sig;

}

return i;

}

void Explocion(int tamXNod,int tamYNod, JUGADOR J,int \*iVidaMenos){

int a=1;

do{

DibujaImagen(tamXNod+50,tamYNod+50,J.arr[12],J.Pos->coorX-10,J.Pos->coorY-10);

delay(100);

DibujaImagen(tamXNod+50,tamYNod+50,J.arr[13],J.Pos->coorX-10,J.Pos->coorY-10);

a++;

}while(a!=5);

\*iVidaMenos=1;

}

void EliminaEnemigo(LISTENEM \*le,int numero)

{

ENEMIGO \*aux2=le->ult;

ENEMIGO \*aux=le->prim;

NODOD \*Noso=NULL;

if(le->prim==le->ult){

aux->Posenemigo->enemigo=NULL;

free(le->prim);

le->prim=le->ult=NULL;

}else

if(le->ult->iNum==numero){

Noso=aux2->Posenemigo;

if(aux2->tipo==2){

Noso->obstaculo=2;

}else{

Noso->objeto=rand()%IMAGOBJ;

}

le->ult=le->ult->ant;

le->ult->sig=NULL;

aux2->Posenemigo->enemigo=NULL;

free(aux2);

}else

if(le->prim->iNum==numero){

Noso=aux->Posenemigo;

if(aux->tipo==2){

Noso->obstaculo=2;

}else{

Noso->objeto=rand()%IMAGOBJ;

}

le->prim=le->prim->sig;

le->prim->ant=NULL;

aux->Posenemigo->enemigo=NULL;

free(aux);//2

}else{

while(aux != NULL && aux->iNum != numero)

{

aux=aux->sig;

}

if(aux->iNum == numero)

{

Noso=aux->Posenemigo;

if(aux->tipo==2){

Noso->obstaculo=2;

}else{

Noso->objeto=rand()%IMAGOBJ;

}

aux->sig->ant=aux->ant;

aux->ant->sig=aux->sig;

aux->sig=NULL;

aux->ant=NULL;

aux->Posenemigo->enemigo=NULL;

free(aux);

}

}

}

void inicializaBalas(LISTBALAS \*balas)

{

int i;

balas->prim=NULL;

balas->ult=NULL;

for(i=0; i<3; i++)

{

sprintf(balas->imgenem[i].nomarch,"balas%d.dso",i+1);

balas->imgenem[i].pixels=LeeImagen(balas->imgenem[i].nomarch,&balas->imgenem[i].pX,&balas->imgenem[i].pY);

}

}

BALA \*CreaBala(int tipoArma,NODOD \*Pos)

{

BALA \*nuevo;

nuevo=(BALA \*)malloc(sizeof(BALA));

nuevo->ant=NULL;

nuevo->sig=NULL;

nuevo->Posbala=Pos;

nuevo->arma=tipoArma;

return (nuevo);

}

void MueveBalas(LISTBALAS \*balas,LISTENEM \*le,int tamXNod,int tamYNod)

{

BALA \*aux=balas->prim;

NODOD \*aux2=NULL;

NODOD \*aux3=NULL;

NODOD \*aux4=NULL;

srand(time(NULL));

while(aux)

{

MueveBala(aux,tamXNod,tamYNod,balas->imgenem);

if(aux->Posbala->obstaculo!=0 || aux->Posbala->objeto!=0){

aux->Posbala->obstaculo=0;

aux->Posbala->objeto=0;}

if(aux->Posbala->sig->sig->sig->sig->sig==NULL){

EliminaBala(\*(&balas),aux->iNum);}

aux=aux->sig;

}

}

void MueveBala(BALA \*bala,int tamXNod,int tamYNod,IMAGEN imgn[IMAGENBAL])

{

if(bala->Posbala->sig!=NULL)

{

borraimagen(tamXNod,tamYNod,bala->Posbala->coorX,bala->Posbala->coorY);

bala->Posbala->balas=NULL;

bala->Posbala=bala->Posbala->sig;

bala->Posbala->balas=bala;

DibujaImagen(tamXNod,tamYNod,imgn[bala->arma],bala->Posbala->coorX,bala->Posbala->coorY);

}

}

void Dispara(NODOD \*Principal,LISTBALAS \*balas, JUGADOR J,int iArma)

{

BALA \*nuevoE;

NODOD \*aux, \*auxP;

int x;

int y;

auxP=Principal;

aux=Principal;

for(y=0; y<NY; y++)

{

for(x=0; x<NX; x++)

{

if(aux==J.Pos)

{

nuevoE=CreaBala(iArma,aux);

if(nuevoE)

{

if(balas->prim==NULL)

{

balas->prim=nuevoE;

balas->ult=nuevoE;

balas->prim->iNum=1;

}

else{

nuevoE->sig=balas->prim;

balas->prim->ant=nuevoE;

balas->prim=nuevoE;

balas->prim->iNum+=balas->prim->sig->iNum+1;

}

aux->balas=nuevoE;

}

}

aux=aux->sig;

}

auxP=auxP->abajo;

aux=auxP;

}

}

void EliminaBala(LISTBALAS \*bal,int numero)

{

BALA \*aux2=bal->ult;

NODOD \*Noso=NULL;

BALA \*aux=bal->prim;

if(bal->prim==bal->ult){

free(bal->prim);

bal->prim=bal->ult=NULL;

}else

if(bal->ult->iNum==numero){

bal->ult=bal->ult->ant;

bal->ult->sig=NULL;

aux2->Posbala->balas=NULL;

free(aux2);

}else

if(bal->prim->iNum==numero){

bal->prim=bal->prim->sig;

bal->prim->ant=NULL;

aux->Posbala->balas=NULL;

free(aux);

}else{

while(aux != NULL && aux->iNum != numero)

{

aux=aux->sig;

}

if(aux->iNum == numero)

{

aux->sig->ant=aux->ant;

aux->ant->sig=aux->sig;

aux->sig=NULL;

aux->ant=NULL;

aux->Posbala->balas=NULL;

free(aux);

}}

}

int EliminaEB(LISTBALAS \*balas,LISTENEM \*le ,int \*num)

{

BALA \*aux=balas->prim;

NODOD \*siguiente=NULL;

while(aux!=NULL)

{

siguiente=aux->Posbala->ant;

if(aux->Posbala->enemigo!=NULL){

\*num+=1;

EliminaEnemigo(le,aux->Posbala->enemigo->iNum);

EliminaBala(\*(&balas),aux->Posbala->balas->iNum);

return 1;

}else{

if(siguiente->enemigo!=NULL){

\*num+=1;

EliminaEnemigo(le,siguiente->enemigo->iNum);

EliminaBala(\*(&balas),aux->Posbala->balas->iNum);

return 1;

}}

aux=aux->sig;

}

}

void EliminaLista(LISTBALAS \*bal)

{

BALA \*aux=bal->prim;

while(bal->prim!=NULL)

{

aux=bal->prim;

bal->prim=bal->prim->sig;

free(aux);

}

}

void Esenario(int \*f,int \*x,int iDistancia,int iVidas,int iNivel)

{

espacio(iNivel+1,\*x);

interior(5,5,834,80,7);

interior(835,5,917,586,7);

interior(82,587,917,662,7);

interior(5,81,80,662,7);

interior(919,5,1195,662,7);

setfillstyle(1,0);

char cDistancia[10];

rectangle(1000,80,1185,300);

rectangle(929,80,980,285);

if(iVidas>=5){

setfillstyle(1,3);

bar(935,85,975,100);

setfillstyle(1,3);

bar(935,105,975,120);

}

if(iVidas>=4){

setfillstyle(1,2);

bar(935,125,975,140);

setfillstyle(1,2);

bar(935,145,975,160);

}

if(iVidas>=3){

setfillstyle(1,14);

bar(935,165,975,180);

setfillstyle(1,14);

bar(935,185,975,200);

}

if(iVidas>=2){

setfillstyle(1,6);

bar(935,205,975,220);

setfillstyle(1,6);

bar(935,225,975,240);

}

if(iVidas>=1){

setfillstyle(1,4);

bar(935,245,975,260);

setfillstyle(1,4);

bar(935,265,975,280);

}

settextstyle(8, 0, 1);

setfillstyle(1,0);

bar(929,20,1185,60);

outtextxy(930,25,"Nombre:");

bar(935,500,1185,586);

outtextxy(940,500,"Enemigos eliminados");

outtextxy(940,520,"Diamantes");

outtextxy(940,540,"Notas");

outtextxy(940,560,"Total");

bar(1000,350,1185,400);

outtextxy(1005,350,"Nivel");

outtextxy(1005,375,"Vidas");

bar(1000,420,1185,470);

outtextxy(1005,420,"Num. de Balas");

outtextxy(1005,445,"Tipo");

bar(10,10,200,75);

settextstyle(3, 0, 4);

itoa(iDistancia,cDistancia,10);

outtextxy(25,25,cDistancia);

outtextxy(125,25,"Km");

settextstyle(8, 0, 1);

if(\*x==810)

\*x=0;

}

void interior(int x1,int y1,int x2,int y2,int cual){

setfillstyle(1,cual);

bar(x1,y1,x2,y2);

setfillstyle(1,15);

line(x1,y1,x2+2,y1);

line(x1,y1,x1,y2+2);

setfillstyle(1,15);

line(x1,y2+2,x2+2,y2+2);

line(x2+2,y2+2,x2+2,y1);

setfillstyle(1,0);

}

int mouse(int \*x,int \*y)

{

\*x = mousex();

\*y = mousey();

if(ismouseclick(WM\_LBUTTONDOWN)==1)

{

clearmouseclick(WM\_LBUTTONDOWN);

return(1);

}

return(0);

}

int iPause(){

int iPause;

settextstyle(3, 0, 6);

outtextxy(28,15,"Pause");

char cTecla;

bar(81,80,837,587);

do{

cTecla=getch();

switch( cTecla){

}

}while(cTecla!=13);

}

void lee\_texto(int x,int y, char cadena[5])

{

cleardevice();

rectangle(0,0,1199,669);

rectangle(10,10,1189,659);

rectangle(20,55,250,100);

outtextxy(15,20,"Cual es tu nombre\_");

char tecla;

int contador=0;

cadena[0]='\0';

do{

do{

setcolor(15);

outtextxy(x+textwidth(cadena),y,"\_");

delay(50);

setcolor(0);

outtextxy(x+textwidth(cadena),y,"\_");

delay(50);

setcolor(15);

}while(!kbhit());

tecla=getch();

if(tecla==0)

tecla=getch();

if(tecla==8 && contador > 0)

{

setcolor(BLACK);

outtextxy(x,y,cadena);

cadena[--contador]='\0';

setcolor(WHITE);

outtextxy(x,y,cadena);

}

else

{

if(tecla!=13)

{

cadena[contador++]=tecla;

cadena[contador]='\0';

outtextxy(x,y,cadena);

}

}

}while(tecla!=13 && contador!=6);

settextstyle(0, 0, 0);

}

int EscojeTuNave()

{

char cOp;

int pagina=0;

int iY=200;

int iY2=500;

int iXm,iYm;

settextstyle(3, 0, 4);

int X1,X2;

X1=470;

X2=730;

int iColor=10;

do{

setactivepage(pagina);

cleardevice();

settextstyle(3, 0, 4);

outtextxy( 15,15 , "Escoge tu nave::" );

readimagefile("Nave1.jpg",475-300,200,725-300,500);

rectangle(475-300,200,725-300,500);

readimagefile("Nave3.jpg",475+300,200,725+300,500);

rectangle(475+300,200,725+300,500);

readimagefile("Nave2.jpg",475,200,725,500);

rectangle(475,200,725,500);

rectangle(0,0,1199,669);

rectangle(10,10,1189,659);

setvisualpage(pagina);

if(pagina==0)

pagina=1;

else

pagina=0;

rectangle(X1,195,X2,505);

if(kbhit()){

cOp=getch();

switch(cOp){

case KEY\_RIGHT:

if(X1<770){

X1+=300;

X2+=300;

}

if(X1==170){

iColor=14;

}

if(X1==470){

iColor=10;

}

if(X1==770){

iColor=6;

}

break;

case KEY\_LEFT:

if(X2>470){

X1-=300;

X2-=300;

}

if(X1==170){

iColor=14;

}

if(X1==470){

iColor=10;

}

if(X1==770){

iColor=6;

}

break;

} }

if(iXm>470 && iYm>200 && iXm<730 && iYm<500){

X1=470;

X2=730;

iColor=10;

}

if(iXm>170 && iYm>200 && iXm<430 && iYm<500){

X1=170;

X2=430;

iColor=14;

}

if(iXm>770 && iYm>200 && iXm<1030 && iYm<500){

X1=770;

X2=1030;

iColor=6;

}

if(mouse(&iXm,&iYm)){

if(iXm>170 && iYm>200 && iXm<430 && iYm<500)

cOp=13;

if(iXm>470 && iYm>200 && iXm<730 && iYm<500)

cOp=13;

if(iXm>770 && iYm>200 && iXm<1030 && iYm<500)

cOp=13;

}

}while(cOp!=13);

return(iColor);

}

int perdiste(){

cleardevice();

int iA,iB=10,I;

char cB[5];

int i=10;

char c[5];

setcolor(15);

do{

rectangle(0,0,1199,669);

rectangle(10,10,1189,659);

readimagefile("Pierdes.jpg",20,20,750,600);

settextstyle(3,0,2);

outtextxy(180, 18 ,"<<..•.¸¸•´¯`•.¸¸¤ Has sido derrotado ¤¸¸.•´¯`•¸¸.•..>>");

outtextxy(160, 600,"???¤°.¸¸.·´¯`»® Enter para volver a intentar ®«´¯`·.¸¸.°¤???");

settextstyle(3, 0, 9);

itoa(i,c,10);

outtextxy(900, 300,c);

delay(900);

i--;

if(kbhit())

iA=getch();

}while(iA!=13 && i!=0);

setcolor(15);

if(iA==13)

return(1);

if(i==0)

return(ESC);

}

void ayuda()

{

musica("musicajuego.wav");

int iTecla;

FILE \*f;

cleardevice();

rectangle(0,0,1199,669);

rectangle(10,10,1189,659);

char c[255];

int x=50,y=50;

settextstyle(4,0,3);

f=fopen("Ayuda.txt","r");

while(!feof(f))

{

fgets(c,255,f);

c[strlen(c)-1]='\0';

outtextxy(x,y,c);

y+=textheight("H");

}

fclose(f);

getch();

}

void guardarR(char nom\_arch[50],JUGADOR J)

{

FILE \*f;

JUGADOR p[5],aux[5];

int i;

f=fopen(nom\_arch,"rb+");

if(f==NULL)

{

f=fopen(nom\_arch,"wb");

strcpy(p[0].cNombre,J.cNombre);

p[0].iPuntos=J.iPuntos;

for(i=1;i<5;i++)

{

strcpy(p[i].cNombre,"- - - - - - - - - -");

p[i].iPuntos=0;

}

fwrite(p,sizeof(JUGADOR),5,f);

}

else

{

fread(p,sizeof(JUGADOR),5,f);

fseek(f,0,SEEK\_SET);

if(J.iPuntos > p[0].iPuntos)

{

strcpy(aux[0].cNombre,J.cNombre);

aux[0].iPuntos=J.iPuntos;

for(i=0;i<4;i++)

{

strcpy(aux[i+1].cNombre,p[i].cNombre);

aux[i+1].iPuntos=p[i].iPuntos;

}

}

else

{

i=0;

do

{

strcpy(aux[i].cNombre,p[i].cNombre);

aux[i].iPuntos=p[i].iPuntos;

i++;

if(i==5)

break;

}while(p[i].iPuntos > J.iPuntos);

if(i!=5)

{

strcpy(aux[i].cNombre,J.cNombre);

aux[i].iPuntos=J.iPuntos;

for(;i<4;i++)

{

strcpy(aux[i+1].cNombre,p[i].cNombre);

aux[i+1].iPuntos=p[i].iPuntos;

}

}

}

fwrite(aux,sizeof(J),5,f);

}

fclose(f);

}

void imprime\_record(char nom\_arch[50])

{

FILE \*f;

rectangle(0,0,1199,669);

rectangle(10,10,1189,659);

JUGADOR rec[5];

char cadena[60];

int i;

char cOp;

f=fopen(nom\_arch,"rb");

musica("musicajuego.wav");

cleardevice();

setcolor(15);

settextstyle (0,0,1);

do{

if(f==NULL)

{

settextstyle (4,0,3);

outtextxy(40,60,"Aun no hay records almacenados");

outtextxy(40,100,"Preciona enter para volver");

settextstyle (10,0,1);

}

else

{

settextstyle (4,0,3);

outtextxy(40,60,"Preciona enter para volver");

settextstyle (4,0,1);

outtextxy(40,100,"Puntuaciones mas Altas: ");

fread(rec,sizeof(JUGADOR),5,f);

for(i=0;i<5;i++)

{

settextstyle (10,0,3);

itoa(rec[i].iPuntos,cadena,10);

outtextxy(150,200+(i+1)\*30,rec[i].cNombre);

outtextxy(600,200+(i+1)\*30,cadena);

}

}

cOp=getch();

}while(cOp!=13);

fclose(f);

}

void musica(char cNombre[50])

{

HMODULE Lib = LoadLibrary("winmm.dll");

if (Lib)

{

fn\_PlaySound =(PlaySoundA\_ptr)GetProcAddress(Lib, "PlaySoundA");

if (fn\_PlaySound)

fn\_PlaySound(cNombre, NULL, SND\_FILENAME | SND\_ASYNC);

else

outtextxy(150,200,"Error No se encontro la función PlaySoundA");

FreeLibrary(Lib);

}}

void portada(){

readimagefile("UASLP.jpg",420,150,820,500);

settextstyle(4, 0, 2);

setcolor(15);

rectangle(98,39,668,62);

setcolor(14);

outtextxy( 100 , 40 , "Universidad Autonoma de San Luis Potosi" );

setcolor(15);

rectangle(223,99,527,122);

setcolor(14);

outtextxy( 225, 100 , "Facultad de Ingenieria" );

setcolor(15);

rectangle(138,159,616,182);

setcolor(14);

outtextxy( 140 , 160 , "Estructuras de datos y algoritmos B" );

setcolor(15);

rectangle(188,219,569,242);

setcolor(14);

outtextxy( 190 , 220 , "Luis Alberto Diaz Villanueva" );

setcolor(15);

rectangle(218,279,536,302);

setcolor(14);

outtextxy( 220 , 280 , "Vanguard" );

setcolor(11);

settextstyle(1,0,1);

outtextxy(40,450,"Presiona enter para continuar. . .");

setcolor(15);

int q;

do{

if(kbhit())

q=getch();

}while(q!=13);

}

int espacio(int iColor,int iMueve){

int e=1;

for(e=0;e<13;e++)

interior(85+65\*e-iMueve,140,150+65\*e-iMueve,150,iColor);

e=1;

for(e=0;e<13;e++)

interior(85+65\*e-iMueve,500+20,150+65\*e-iMueve,510+20,iColor);

e=1;

for(e=0;e<12;e++)

interior(930+65\*e-iMueve,140,1080+65\*e-iMueve,150,iColor);

e=1;

for(e=0;e<12;e++)

interior(930+65\*e-iMueve,500+20,11080+65\*e-iMueve,510+20,iColor);

}

int cuentalistaB(LISTBALAS balas)

{

int i=0;

BALA \*aux=balas.prim;

while(aux!=NULL)

{

i+=1;

aux=aux->sig;

}

return i;

}

int iSiguiente\_Nivel(int iNivel){

int q=0;

if(iNivel==5){

readimagefile("Ganaste.jpg",81,80,917,587);

}else{

readimagefile("Siguiente.jpg",81,80,917,587);

}

do{

if(kbhit())

q=getch();

}while(q!=13);

}