Universidad Autónoma de San Luis Potosí

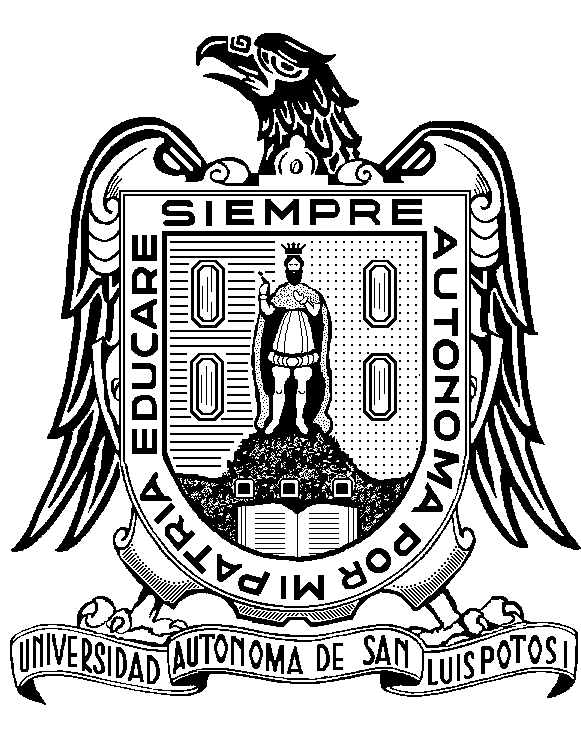
Facultad de Ingeniería

Área de Computación e Informática

Estructuras de Datos y Algoritmos B

--------------

Manual de Programación



Ing.----------------------------

-------------------

Índice

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

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

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

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

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

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

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

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 -

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

* Para este programa se utilizó el compilador
* Y la librería

Objetivos:

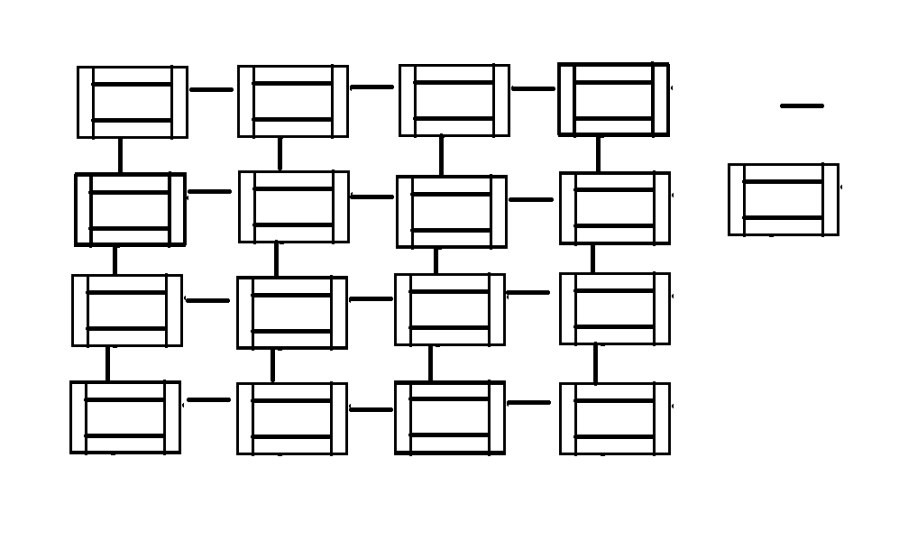
El objetivo del programa es utilizar las habilidades aprendidas durante el curso de estructuras de datos y algoritmos B, para así desarrollar nuevas habilidades de lógica.

Aplicaciones:

Este proyecto funciona con el teclado, 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 se leen las ayudas del programa y para escribir los records de los jugadores.

La parte más importante del programa se desarrollo dentro de una malla enlazada que fue creada con estructuras llamadas nodos. Cada nodo tiene 4 apuntadores que enlaza, arriba, abajo, izquierda y derecha.



Apuntador der

Nodo

Apuntador abaj

Apuntador arib

arr

Apuntador izq

**4.-Librerias**

|  |  |
| --- | --- |
| graphics.h |  |
| stdio.h |  |
| iostream |  |
| time.h |  |

**4.1.- Constantes**

|  |  |
| --- | --- |
| TamNodo | Tamaño que tiene el nodo |
| TamMallaX | **Tamaño de la malla en x** |
| TamMallaY | **Tamaño de la malla en y** |

**5.-Estructuras**

**5.1 Estructura Nodo**

|  |
| --- |
| Estructura principal del juego, se crean y enlazan nodos para generar la malla en donde se desarrolla el juego |

typedef struct nodo

{

struct nodo \*izq;

struct nodo \*der;

struct nodo \*arri;

struct nodo \*aba;

int x1,y1,x2,y2;

struct bala \*NodoB;

struct enem \*NodoE;

struct enemf \*NodoEF;

struct ju \*NodoJ;

int arma;

}Nodo;

|  |  |
| --- | --- |
| struct nodo izq,der,arri,aba | Apuntadores a nodos |
| Int x1,x2,y1,y2 | Posición del nodo en la pantalla |
| struct bala \*NodoB; | Apuntador a la bala, si no hay bala en el nodo es igual a NULL |
| struct enem \*NodoE; | Apuntador al enemigo, si no hay enemigo es igual a NULL |
| struct enemf \*NodoEF; | Apuntador al enemigo final |
| struct ju \*NodoJ; | Apuntador al jugador indica en que nodo está el jugador |
| int arma; | Indica si existe alguna arma en el nodo |

**5.2 Estructura Imagen**

|  |
| --- |
| Estructura principal para las imágenes. |

typedef struct

{

int \*\*malla;

}dibujo;

|  |  |
| --- | --- |
| int \*\*malla | Matriz dinámica que almacena los colores de las imágenes del juego |

**5.3 Estructura Bala**

|  |
| --- |
| Estructura principal para generar balas |

typedef struct bala

{

Nodo \*PosB;

struct bala \*ant,\*sig;

int itipo;

int indice;

}Bala;

|  |  |
| --- | --- |
| Nodo \*PosB; | Posición de la bala en la malla, se utiliza para mover a la bala |
| struct bala \*ant,\*sig; | Apuntadores a balas, se usa para generar una lista de balas |
| int itipo; | Indica que tipo de bala es |
| int indice; | Indica el numero de la bala, se usa para la eliminación |

**5.4 Estructura ListBalas**

|  |
| --- |
| Estructura que se utiliza para generar una lista de balas |

typedef struct{

Bala \*prim,\*ult;

dibujo iconoBal[10];

}ListBalas;

|  |  |
| --- | --- |
| Bala \*prim,\*ult; | Cabeceras que apuntan a el primer y el último elemento de la lista |
| dibujo iconoBal[10]; | Imágenes que las balas pueden tener |

**5.5 Estructura Enemigo**

|  |
| --- |
| Estructura que se utiliza para crear a un enemigo en el juego |

typedef struct enem

{

Nodo \*PosE;

struct enem \*ant,\*sig;

int tipo;

int indice;

int pasos;

int numPasos;

}Enemigo;

|  |  |
| --- | --- |
| Nodo \*PosE; | Posición del jugador en la malla |
| struct enem \*ant,\*sig; | Apuntadores que se utilizan para la lista de enemigos |
| int tipo; | Tipo de enemigo, se utiliza para saber que tipo de enemigo se creo |
| int indice; | Se utiliza para la eliminación, es un identificador de enemigo |
| int pasos; | Se utiliza para un enemigo especifico, éste se mueve cuando se da cierta cantidad de pasos |
| int numPasos; | Se utiliza para un enemigo especifico, éste se mueve cuando se da cierta cantidad de pasos |

**5.6 Estructura ListEnemigos**

|  |
| --- |
| Se utiliza para generar una lista de enemigos |

typedef struct

{

Enemigo \*prim,\*ult;

dibujo iconoEnem[10];

}ListEnemigos;

|  |  |
| --- | --- |
| Enemigo \*prim,\*ult; | Cabeceras a los enemigos primero y último de la lista |
| dibujo iconoEnem[10]; | Iconos para los enemigos |

**5.7 Estructura EnemigoFinal**

|  |
| --- |
| Se utiliza para crear un enemigo final |

typedef struct enemf

{

Nodo \*PosEF;

Nodo \*PosArma1;

Nodo \*PosArma2;

int dir;

int pasos;

int R;

int imag;

bool activo;

dibujo iconoEnemF[3];

}EnemigoFinal;

|  |  |
| --- | --- |
| Nodo \*PosEF; | Posición del enemigo en la malla |
| Nodo \*PosArma1; | Posición de las armas que tiene el enemigo |
| int dir; | Se utiliza para la dirección del enemigo, 1 izquierda y 0 derecha |
| int pasos; | Numero de pasos, se utiliza para saber cuándo se debe cambiar de dirección |
| int R; | Es la resistencia que tiene el enemigo |
| int imag; | Imagen actual del enemigo |
| bool activo; | Indica si el enemigo debe moverse |
| dibujo iconoEnemF[3]; | Imágenes para el enemigo |

**5.8 Estructura Records**

|  |
| --- |
| Estructura exclusiva para los reccords |

typedef struct

{

char cNombre[10];

int puntos;

}Records;

|  |  |
| --- | --- |
| char cNombre[10]; | Nombre del jugador |
| int puntos; | Puntos |

**5.9 Estructura Jug**

|  |
| --- |
| Estructura del jugador, se utiliza para generar un jugador |

typedef struct ju

{

Nodo \*PosJ;

int iNave;

int iTipoB;

int iNumB;

int iVidas;

int iNivel;

int enemigosEliminados;

bool muerto;

Records R;

dibujo iconoJug[10];

}jug;

|  |  |
| --- | --- |
| Nodo \*PosJ; | Posición actual del jugador en la malla |
| int iNave; | Nave que el jugador tiene |
| int iTipoB; | Tipo de balas |
| int iNumB; | Numero de balas que tiene el jugador |
| int iVidas; | Numero de vidas |
| int iNivel; | Nivel en el que esta el jugador |
| int enemigosEliminados; | Numero de enemigos que el jugador a eliminado |
| bool muerto; | Indica si el jugador esta vivo o muerto |
| Records R; | Almacena los records |
| dibujo iconoJug[10]; | Iconos del jugador |

**5.10 Estructura scroll**

|  |
| --- |
| Genera el scroll en el juego, se utiliza para generar un arreglo de puntos en la pntalla y estos se recorren |

typedef struct

{

int x;

int y;

}scroll;

|  |  |
| --- | --- |
| int x; | Posición en x, |
| int y; | Posición en y |

**6.-Funciones**

**6.1 Función creaNodo**

Nodo \*creaNodo(int x1,int y1);

|  |  |
| --- | --- |
| Reserva memoria para un Nodo y lo regresa |  |
| int x1,int y1 | Son las posiciones del nodo |

**6.2 Función creaMaya**

Nodo \*creaMaya();

|  |
| --- |
| Crea nodos y los enlaza para generar una malla, regresa el primer nodo que creo |

**6.3 Función Dibujaicono**

void Dibujaicono(int \*\*imagen,int x,int y,int tamx,int tamy,int pixel);

|  |  |
| --- | --- |
| Pinta un icono en la pantala |  |
| int \*\*imagen | Matriz dinámica, que tiene los colores del dibujo |
| int x,int y | Posición en donde se pintara el icono |
| int tamx,int tamy | Numero de pixeles en X y Y |
| int pixel | Tamaño de la imagen |

**6.4 Función borraImagen**

void borraImagen(int x1,int y1,int x2,int y2);

|  |  |
| --- | --- |
| Pone una barra de un color espesifico | |
| int x1,int y1, int x2,int y2 | Posición en la que se debe de poner la barra |

**6.5 Función abreimagen**

int \*\*abreimagen(char nombre[]);

|  |  |
| --- | --- |
| Busca un archivo con el nombre especificado lo lee y regresa la malla con los colores que están en ese archivo | |
| char nombre[]); | Nombre del archivo |

**6.6 Función creaJugador**

void creaJugador(jug \*J,Nodo \*principal);

|  |  |
| --- | --- |
| Inicializa al Jugador | |
| jug \*J | El jugador |
| Nodo \*principal | El nodo en donde iniciará el jugador |

**6.7 Función mueveJugador**

void mueveJugador(jug \*J,int \*tecla,ListBalas \*LB,bool EnemFActivo);

|  |  |
| --- | --- |
| Cambia de posición el apuntador que tiene el jugador y lo dibuja en una nueva posición | |
| jug \*J | **Jugador, para ser modificado** |
| int \*tecla | **La tecla se modifica** |
| ListBalas \*LB | **La lista de balas se modifica cuando el jugador oprime una tecla** |
| bool EnemFActivo | **Si el enemigo está activo límites de la pantalla para el jugador cambian** |

**6.8 Función creaBala**

bala \*creaBala(Nodo \*pos,int itipo);

|  |  |
| --- | --- |
| Reserva memoria para una bala y la regresa | |
| Nodo \*pos | Posición de la malla en donde iniciara la bala |
| int itipo | Tipo de bala, se usa para las imágenes |

**6.9 Función imagenesBalas**

void imagenesBalas(ListBalas \*balas);

|  |  |
| --- | --- |
| Carga las imágenes de las balas | |
| ListBalas \*balas | Se envía la lista de balas, en esta estructura están las imágenes de las balas |

**6.10 Función dispara**

void dispara(Nodo \*pos,ListBalas \*balas,int tipo);

|  |  |
| --- | --- |
| Agrega una bala a la lista de balas | |
| Nodo \*pos | Posición en donde iniciara la nueva bala |
| ListBalas \*balas | Lista en donde se agregara la nueva bala |
| int tipo | Tipo de bala que se va a agregar |

**6.11 Función MueveListBalas**

void MueveListBalas(ListBalas \*balas);

|  |  |
| --- | --- |
| Funcion que mueva la lista de balas en la malla | |
| ListBalas \*balas | Cambia la posición de las balas se recorre la lista y se mueve cada bala una por una |

**6.12 Función MueveBala**

void MueveBala(bala \*bal,dibujo iconoBal[]);

|  |  |
| --- | --- |
| Mueve una sola bala | |
| bala \*bal | Bala que se cambiara de posición en la malla |
| dibujo iconoBal[]); | Dibujo de esta bala |

**6.13 Función CondicionesEliminaBalasEnemigo**

void CondicionesEliminaBalasEnemigo(ListBalas \*balas,ListEnemigos \*enemigos,jug \*J);

|  |  |
| --- | --- |
| Tiene todas las condiciones bajo las cuales se debe eliminar una bala | |
| ListBalas \*balas | La lista se modificara ya que se eliminaran balas |
| ListEnemigos \*enemigos | La lista de enemigos es necesaria por si la bala toca a un enemigo |
| jug \*J | El jugador es enviado para saber cuántas balas tiene la lista de balas |

**6.14 Función EliminaBala**

void EliminaBala(ListBalas \*balas,int numero);

|  |  |
| --- | --- |
| Elimina la bala espesificada | |
| ListBalas \*balas | La lista de balas se modifica |
| int numero | El índice de la bala que se va a eliminar |

**6.15 Función creaEnemigo**

Enemigo \*creaEnemigo(Nodo \*pos,int tipo);

|  |  |
| --- | --- |
| Reserva memoria para un enemigo y lo regresa | |
| Nodo \*pos | Posición del enemigo donde se va a iniciar en la malla |
| int tipo | Tipo de enemigo que se crea |

**6.16 Función imagenesEnemigos**

void imagenesEnemigos(ListEnemigos \*enemigos);

|  |  |
| --- | --- |
| Carga las imágenes que van a tener los enemigos | |
| ListEnemigos \*enemigos | En la estructura lista enemigos está el arreglo de imágenes para los enemigos |

**6.17 Función agregaEnemigo**

void agregaEnemigo(ListEnemigos \*enemigos,Nodo \*pos,int tipo);

|  |  |
| --- | --- |
| Agrega un enemigo a la lista con su posición y su tipo | |
| ListEnemigos \*enemigos | Lista que se va a modificar en donde se va a agregar un enemigo nuevo |
| Nodo \*pos | Posición en la que se inicia el enemigo |
| int tipo | Tipo de enemigo |

**6.18 Función MueveListEnemigos**

void MueveListEnemigos(ListEnemigos \*enemigos);

|  |  |
| --- | --- |
| Mueve la lista de enemigos la recorre | |
| ListEnemigos \*enemigos | Lista que se va a recorrer uno por uno |

**6.19 Función MueveEnemigo**

void MueveEnemigo(Enemigo \*enemigo,dibujo iconoEnem[]);

|  |  |
| --- | --- |
| Mueve un enemigo especifico | |
| Enemigo \*enemigo | Enemigo que se va a recorrer |
| dibujo iconoEnem[]); | Dibujo del enemigo |

**6.20 Función CondicionesEliminaEnemigoBalas**

void CondicionesEliminaEnemigoBalas(ListEnemigos \*enemigos,ListBalas \*balas,jug \*J);

|  |  |
| --- | --- |
| Funcion que tiene todas las condiciones bajo las cuales se debe eliminar un enemigo | |
| ListEnemigos \*enemigos | Lista de enemigos |
| ListBalas \*balas | Las balas por si un enemigo toca una bala |
| jug \*J | El jugador pos si la bala lo toca |

**6.21 Función EliminaEnemigo**

void EliminaEnemigo(ListEnemigos \*enemigos,int numero);

|  |  |
| --- | --- |
| Elimina un enemigo espesifico | |
| ListEnemigos \*enemigos | Lista que se va a modificar cuando el enemigo se elimine |
| int numero | Índice del enemigo que se va a modificar |

**6.22 Función creaEnemigoFinal**

void creaEnemigoFinal(EnemigoFinal \*EF,Nodo \*principal);

|  |  |
| --- | --- |
| Crea un enemigo final | |
| EnemigoFinal \*EF | Inicia todos los parámetros del enemigo final |
| Nodo \*principal | Nodo en donde estará en enemigo |

**6.23 Función mueveEnemigoFinal**

void mueveEnemigoFinal(EnemigoFinal \*EF,ListEnemigos \*LE,bool final);

|  |  |
| --- | --- |
| Condiciones del movimiento del enemigo final | |
| EnemigoFinal \*EF | EL enemigo se modificara la posición |
| ListEnemigos \*LE | Lista de enemigos para que en enemigo final pueda crear enemigos |
| bool final | Indica cuando el enemigo debe moverse |

**6.24 Función BuscaNodoPosicion**

Nodo \*BuscaNodoPosicion(Nodo \*principal,int num);

|  |  |
| --- | --- |
| Busca un nodo en la malla y lo regresa | |
| Nodo \*principal | EL primer nodo de la malla para poder recorrerla |
| int num | El número del nodo que se necesita |

**6.25 Función enemigosAleatorios**

void enemigosAleatorios(Nodo \*nodo,ListEnemigos \*LE,int nivel,jug \*J,EnemigoFinal \*EF,Nodo \*principal);

|  |  |
| --- | --- |
| Genera un enemigo aleatorio en una posición aleatoria | |
| Nodo \*nodo | Nodo principal para poder crear un enemigo en una posición aleatoria |
| ListEnemigos \*LE | Lista de enemigos para poder agregar un enemigo |
| int nivel | El nivel para saber qué tipo de enemigos se deben agregar |
| jug \*J | El jugador para saber el nivel actual del juego |
| EnemigoFinal \*EF | El enemigo para saber cuándo se debe activar |
| Nodo \*principal | Nodo principal |

**6.26 Función juego**

void juego(jug J,ListBalas LB,ListEnemigos LE,EnemigoFinal EF,Nodo \*nodo,scroll p[]);

|  |  |
| --- | --- |
| Funcion donde se desarrolla el juego | |
| jug J | El jugador |
| ListBalas LB | La lista de balas |
| ListEnemigos LE | La lista de enemigos |
| EnemigoFinal EF | El enemigo final |
| Nodo \*nodo | El nodo principal |
| scroll p[]); | arreglo para el scroll |

**6.27 Función paginacion**

void paginacion(int \*pagina);

|  |  |
| --- | --- |
| Funcion que se encarga de cambiar la página activa | |
| int \*pagina | Pagina actual, esta se modifica |

**6.28 Función iniciaScroll**

void iniciaScroll(scroll p[]);

|  |  |
| --- | --- |
| Inicia una serie de puntos en un arreglo | |
| scroll p[]); | Variable scroll que tiene una posición de x y y |

**6.29 Función dibujaScroll**

void dibujaScroll(scroll p[]);

|  |  |
| --- | --- |
| Recorre el arreglo p que tiene una serie de puntos | |
| scroll p[]); | El scroll se recorre |

**6.30 Función imprimeTiempo**

void imprimeTiempo(int tiempo);

|  |  |
| --- | --- |
| Funcion que imprime el tiempo en la pantalla y lo acomoda en forma de minutos y segundos | |
| int tiempo | Manda el tiempo para imprimirse |

**6.31 Función** **dibujaEsenario**

void (Nodo \*principal,jug J);

|  |  |
| --- | --- |
| Dibuja el esenario con los puntos y las vidas | |
| Nodo \*principal |  |
| jug J |  |

**6.32 Función ingresaNombre**

void ingresaNombre(char nombre[]);

|  |  |
| --- | --- |
| Funcion que pide el nombre al jugador y lo guarda en un arreglo de caracteres | |
| char nombre[] | Areglo de caracteres para el nombre |

**6.33 Función ayuda**

void ayuda(jug J,ListEnemigos LE);

|  |  |
| --- | --- |
| Muestra la ayuda que esta en un archivo | |
| jug J | El jugador se utiliza para las imágenes |
| ListEnemigos LE | La lista de enemigos se utiliza para mostrar las imágenes |

**6.34 Función insertarRecord**

void insertarRecord(Records bomber);

|  |  |
| --- | --- |
| Inserta un record en el archivo records | |
| Records bomber |  |

**6.1 Función imprimirRecords**

void imprimirRecords();

|  |
| --- |
| Imprime los records |

**6.35 Función opcion**

int opcion(jug J,scroll p[]);

|  |  |
| --- | --- |
| Mestra un menú y regresa la opción | |
| jug J | El jugador, se utliza para mostrar la imagen del jugador en la pantalla |
| scroll p[]); | El scroll se utiliza para mostrar una animación mientras se selecciona una opción del menú |

**6.36 Función menu**

int menu(jug J,ListBalas LB,ListEnemigos LE,EnemigoFinal EF,Nodo \*nodo,scroll p[]);

|  |  |
| --- | --- |
| En esta función se encarga de mandar llamar a las funciones principales para el juego utilizando la función de opción | |
| jug J | El jugador para mandar llamar la función de juego |
| ListBalas LB | Lista de balas para la función del juego |
| ListEnemigos LE | Lista de enemigos para la función juego y la función ayuda |
| EnemigoFinal EF | El enemigo final para la función juego |
| Nodo \*nodo | El nodo principal que lo utiliza la función juego |
| scroll p[]); | El scroll para la función opción y para a función juego |