UNIVERSIDAD AUTÓNOMA DE SAN LUIS POTOSÍ

FACULTAD DE INGENIERÍA

“PROYECTO: PARED APLASTA CUBOS”

ESTRUCTURAS DE DATOS Y ALGORITMOS “A”

PROFR. SILVIA LUZ VACA RAMIREZ

ALUMNO: EDUARDO MONTES HERNÁNDEZ

INTRODUCCION



MANUAL DEL PROGRAMADOR

ESTAS SON LAS CONSTANTES SIMBOLICAS QUE SE UTILIZAN EN EL PROGRAMA:

#define NUM 15 NUMERO DE CUBOS POR LADO.

#define TAM 15 TAMAÑO DE CADA CUBO.

#define ESC 27 CODIGO DE LA TECLA ESC.

#define XCOO 300 COORDENADA EN X DONDE SE VA A PINTAR EL ARREGLO.

#define YCOO 350 COORDENADA EN Y DONDE SE VA A PINTAR EL ARREGLO.

ESTAS SON LAS UNICAS ESTRUCTURAS QUE SE USAN EN EL JUEGO; LA PRIMERA ES PARA DARLE LAS COORDENADAS A CADA CUBO ASI COMO EL TIPO Y LA SEGUNDA ESTRUCTURA ES PARA GUARDAR LOS RECORDS:

typedef struct

{

int x,y;

int color,existe;

}Tcubo;

typedef struct

{

char nombre[30];

int puntos;

long tiempo;

}Records;

ESTA ES LA FUNCION PRINCIPAL, LA CUAL INICIA LA PORTADA DEL JUEGO MOSTRANDO LOS DATOS DEL PROGRAMADOR Y DESPUES INICIALIZA EL MODO GRAFICO, CREA EL APUNTADOR AL CUBO DE CUBOS, RESERVA LA MEMORIA PARA EL ARREGLO, MANDA LLAMAR A LA FUNCION GENERA CONTENEDOR LA QUE LE DA LAS COORDENADAS A CADA CUBO, INICIALIZA EL CONTENEDOR, DESPUES MANDA LLAMAR A LA FUNCION MENU; CUANDO SE SALEN DEL MENU SE TIENE QUE LIBERAR LA MEMORIA UTILIZADA EN EL CONTENEDOR Y ESO SE HACE CON LA FUNCIN LIBERA CONTENEDOR, POR ULTIMO SE CIERRA EL MODO GRAFICO PARA ASI TERMINAR EL PROGRAMA:

int main()

{

initwindow(800,600); //SE INICIA LA VENTANA DE GRAFICOS

Tcubo\*\*\* contenedor; //SE DECLARA EL ARREGLO TRIDIMENSIONAL DEL CONTENEDOR

portada(); //SE MUESTRA LOS DATOS DEL PROGRAMADOR Y DEL PROGRAMA

cleardevice(); //LIMPIA LA PANTALLA DE LA PORTADA

contenedor=reservaContenedor(); //RESERVA LA MEMORIA DEL CONTENEDOR

generaCont(contenedor);

EstadoInicial(contenedor); //INICIA EL AMBIENTE DEL CONTENEDOR

menu(contenedor); //INICIA EL MENU DEL JUEGO

liberaContenedor(contenedor); //LIBERA LA MEMORIA DEL CONTENEDOR

closegraph(); //CIERRA LA VENTANA DEL GRAFICO

}

ESTA FUNCION ES LA QUE SE ENCARGA DE RESERVAR LA MEMORIA QUE SE NECESITA PARA EL APUNTADOR:

Tcubo\*\*\* reservaContenedor()

{

int i,j,k; // VARIABLES DE LA FILA,COLUMNA Y MATRIZ DEL CONTENEDOR

Tcubo\*\*\* aux;

aux=(Tcubo\*\*\*)malloc(sizeof(Tcubo\*\*)\*NUM); //MEMORIA PARA LAS MATRICES

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

{

\*(aux+i)=(Tcubo\*\*)malloc(sizeof(Tcubo\*)\*NUM); //FILAS DEL CONTENEDOR

for(j=0;j<NUM;j++)

{

\*(\*(aux+i)+j)=(Tcubo\*)malloc(sizeof(Tcubo)\*NUM); //COLUMNAS DEL CONTENEDOR

}

}

return(aux); //SE RETORNA EL CONTENEDOR

}

ESTA FUNCION SE ENCARGA DE DARLE LAS COORDENADAS A CADA CUBO DEL ARREGLO; A TODOS LES DA EL EXISTE=0 PARA QUE NO PINTARLOS PORQUE DESPUES SE VA A MODIFICAR LO QUE SE QUIERE PINTAR; TAMBIEN LES DA UN COLOR DE MANERA ALEATORIA:

void generaCont(Tcubo\*\*\* cont)

{

int i,j,k; // VARIABLES DE LA FILA,COLUMNA Y MATRIZ DEL CONTENEDOR

for(k=0;k<NUM;k++) //CICL0S ANIDADOS PARA GENERAR EL CONTENEDOR

{

for(j=0;j<NUM;j++)

{

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

{

(\*(\*(\*(cont+k)+j)+i)).x=(i\*TAM-k\*TAM/2);

(\*(\*(\*(cont+k)+j)+i)).y=(-j\*TAM+k\*TAM/2);

(\*(\*(\*(cont+k)+j)+i)).color=2;

(\*(\*(\*(cont+k)+j)+i)).existe=0;

}

}

}

}

ESTA FUNCION LE DA LA ESTRUCTURA QUE SE NECESITA PARA EL JUEGO; EN ESTE CASO SE MODIFICA EL EXISTE PARA QUE SE PINTE UNA PLATAFORMA EN LA PARTE DE ABAJO:

void EstadoInicial(Tcubo\*\*\* cont)

{

int i, j; //VARIABLES PARA LA FILA Y MATRIZ DEL CONTENEDOR

for(j=0;j<NUM;j++) //CICLOS ANIDADOS QUE GENERAN LA PLATAFORMA

{

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

{

(\*(\*(\*(cont+j)+0)+i)).existe=1;

}

}

}

LA SIGUIENTE FUNCION LLAMADA “MENU” ES MUY IMPORTANTE PORQUE DESDE ESA FUNCION SE ACCEDE OTRAS FUNCIONES COMO SON LA AYUDA, LOS RECORDS Y AL JUEGO:

void menu(Tcubo \*\*\*cont)

{

int op=0;

int x,y,a; //x,y PARA LEER LAS COORDENADAS DEL MOUSE; a VARIABLE PARA RETORNAR

//EL JUEGO AL MENU

presentacion(); //se inicia el grafico del menú

while(!op)

{

x=mousex(); //se lee la coordenada del mouse en x

y=mousey(); //se lee la coordenada del mouse en y

if("waitForLeftMouseClick()") //SI SE DETECTA UN CLICK

{

if(x>300 && x<600 && y>150 && y<200) //boton jugar

{

cleardevice(); //LIMPIA PANTALLA

a=juego(cont); //INICIA EL JUEGO

presentacion(); //MUESTRA NUEVAMENTE EL MENU DEL JUEGO

}

if(x>300 && x<600 && y>250 && y<300) //boton puntuaciones

{

cleardevice();

ayuda(); //SE MUESTRA LA AYUDA

presentacion();

}

if(x>300 && x<600 && y>350 && y<400) //boton creditos

{

cleardevice();

muestra\_records(); //SE MUESTRAN LOS RECORDS

presentacion();

}

if(x>300 && x<600 && y>450 && y<500) //boton salir

{

op=1; //SALIR DEL JUEGO

}

}

}

}

LA FUNCION PRESENTACION INICIA EL GRAFICO DE LOS BOTONES DEL MENÚ

void presentacion()

{

cleardevice();

setfillstyle(1,BLACK);

rectangle(300,100,900,650);

floodfill(301,101,6);

setcolor(BLUE);

settextstyle(3,0,4);

outtextxy(150,40,"P A R E D A P L A S T A C U B O S !");

settextstyle(3,0,3);

setcolor(BLUE);

setcolor(WHITE);

setfillstyle(1,0);

rectangle(300,150,600,200); //jugar

floodfill(301,151,0);

rectangle(300,250,600,300); //ayuda

floodfill(301,251,0);

rectangle(300,350,600,400); //puntuaciones

floodfill(301,351,0);

rectangle(300,450,600,500); //salir

floodfill(301,451,0);

setcolor(BLUE);

outtextxy(370,160,"J U G A R");

outtextxy(370,260,"A Y U D A");

outtextxy(370,360,"R E C O R D S");

outtextxy(370,460,"S A L I R");

waitForLeftMouseClick();

}

EN ESTA FUNCION SE DETECTA SI HAY UN CLICK IZQUIERDO CON EL MOUSE

void waitForLeftMouseClick()

{

clearmouseclick(WM\_LBUTTONDOWN); //LIMPIA EL BUFFER DEL MOUSE

const int DELAY = 50;

int x, y; //VARIABLES PARA DETECTAR CORRDENADAS DEL MOUSE

while (!ismouseclick(WM\_LBUTTONDOWN))

delay(DELAY);

getmouseclick(WM\_LBUTTONDOWN, x, y);

}

ESTA FUNCION LEE UNA TECLA YA SEA DE CODIGO NORMAL O EXTENDIDO; CUANDO ES DE CODIGO EXTENDIDO EL PRIMER CARÁCTER ES UN CERO, POR ESO SE PONE EL IF:

char leetecla()

{

char tecla;

tecla=getch(); //AGREGA LA TECLA A LA VARIABLE

if(getch==0) //SI ES TECLA DE CODIGO EXTENDIDO VUELVE A LEER

{

tecla=getch();

}

return(tecla);

}

ESTA FUNCION ES DONDE SE DESARROLLA TODO EL JUEGO

int juego(Tcubo\*\*\* contenedor)

{

char tecla, cad[10], \*nom; //VARIABLE CAD ES PARA IMPRIMIR EL TIEMPO Y \*NOM EL NOMBRE

int punt=0, pagina=0,a ,i=0,j=0,c,l=0;//PUNTOS, PAGINADO, CAIDA PARED, CONTADOR PARA

//LA REPETICION DE LA PARED

Int izq=7,der=7;//VARIABLES DE LAS COORDENADAS DE POSICION DE LA FIGURA EN EL ESTADO

//INICIAL

long d; //EN ESTA VARIABLE SE GUARDA EL TIEMPO

time\_t ha, hi; //TIPO DE ESTRUCTURAS QUE GUARDAN EL TIEMPO

//HA HORA ACTUAL, HI HORA INICIAL

inicializa\_jugador(contenedor); //LLAMADA A LA FUNCION QUE ASIGNA AL CONTENEDOR

// CIERTOS CUBOS DESTINADOS AL JUGADOR (existe=2)

nom=captura\_nombre(); //FUNCION QUE CAPTURA EL NOMBRE DEL JUGADOR

setactivepage(pagina); //PARTE DEL PAGINADO PARA QUE LA PANTALLA NO PARPADEE

dibujaContenedor(contenedor,XCOO,YCOO); //FUNCION QUE PINTA EL CONTENEDOR

time(&hi); //FUNCION QUE TOMA LA HORA ACTUAL DEL SISTEMA

//EN ESTE CASO LA GUARDA EN LA HORA INICIAL

while(tecla!=27) //CICLO PRINCIPAL DEL JUEGO, SE TERMINA CUANDO SE PRECIONA “ESC”

{

setvisualpage(pagina); //PAGINACION

if(pagina==1)

{

pagina=0;

}

else

{

pagina=1;

}

setactivepage(pagina);

time(&ha); //TOMA LA HORA ACTUAL

d=(long)difftime(ha, hi); //ASIGNA A “d” LA DIFERENCIA ENTRE HI Y HA

sprintf(cad, "Tiempo: %02d:%02d:%02d", d/3600, d%3600/60, d%3600%60);

//IMPRESION DE DATOS EN UNA CADENA PARA MOSTRAR EL TIEMPO

outtextxy(50,50,cad); //IMPRESION DE LA CADENA EN LA PANTALLA

outtextxy(480,50,"Nombre:"); //IMPRESION DEL NOMBRE

outtextxy(600,50,nom);

sprintf(cad, "Puntos: %d",punt); //IMPRESION DE LOS PUNTOS

if(j%120==0) //PARA QUE APAREZCA UNA NEVA PARED

{if(j%120==0)

{

EstadoInicial(contenedor); //INICIALIZA NUEVAMENTE EL CONTENEDOR

switch(v) //SEGUN LA VARIABLE QUE LLEGA CAMBIA LA FIGURA JUGADOR

{

case 0:

inicializa\_jugador(contenedor,&izq,&der);//JUGADOR CUADRADO CON LAS COORDENADAS DE LA POSICION

inicializa\_pared(contenedor,&c,v);//VUELVE A INICIAR LA PARED MANDANDO LA FIGURA

v++;// AUMENTA v PARA QUE CAMBIE LA FIGURA LA PROXIMA VEZ

break;

case 1:

jugadorlinea(contenedor,&izq,&der); //FIGURA LINEA, SE MANDAN LAS COORDENADAS

inicializa\_pared(contenedor,&c,v);

v++;

break;

case 2:

jugadorL(contenedor,&izq,&der);

inicializa\_pared(contenedor,&c,v);

v++;

break;

case 3:

jugadorT(contenedor,&izq,&der);

inicializa\_pared(contenedor,&c,v);

v=0;

break;

}

if(punt%50==0) //PARA AUMENTAR LA VAOCIDAD DE LA CAIDA DE LA PERED

l++;}

j++;

if(j>120) //REGRESA EL CONTADOR DE j PARA QUE NO SE ELEVE DEMASIADO EL NUMERO

j=1;

if(i%(10-l)==0) //PARA LA VELOCIDAD DE LA CAIDA DE LA PARED

actualiza\_pared(contenedor, &punt);

i++;

if(i>(10-l)) //REGRESA EL CONTADOR DE i PARA QUE NO SE ELEVE DEMASIADO

i=1;

if(kbhit()) //SI SE PRESIONA UNA TECLA

{

tecla=leetecla();

switch(tecla) //LEEMOS LA TECLA QUE SE PRESIONO

{

case 77: //SI PRESIONO LA FLECHA HACIA LA DERECHA

mueve\_derecha(contenedor); //MUEVE AL JUGADOR HACIA LA DERECHA

der++;//ACTUALIZA LA COORDENADA DE LA POSICION DE LA FIGURA

break;

case 75: //SI PRESIONO LA FLECHA HACIA LA IZQUIERDA

mueve\_izquierda(contenedor); //MUEVE AL JUGADOR HACIA LA IZQUIERDA

der--;//ACTUALIZA LA COORDENADA DE LA POSICION DE LA FIGURA

break;

case 72: //SI PRESIONO LA FLECHA HACIA ARRIBA

mueve\_adelante(contenedor); //MUEVE AL JUGADOR HACIA ARRIBA

izq--;//ACTUALIZA LA COORDENADA DE LA POSICION DE LA FIGURA

break;

case 80: //SI PRESIONO LA FLECHA HACIA ABAJO

mueve\_atras(contenedor); //MUEVE AL JUGADOR HACIA ABAJO

izq++;//ACTUALIZA LA COORDENADA DE LA POSICION DE LA FIGURA

break;

case 'a': //SI PRESIONO LA TECLA “A”

giraIzquierda(contenedor); //GIRA EL CONTENEDOR HACIA LA IZQUIERDA

break;

case 'd': //SI PRESIONO LA TECLA “D”

giraDerecha(contenedor); //GIRA EL CONTENEDOR HACIA LA DERECHA

break;

case 27: //SI PRESIONO ESC

setvisualpage(pagina); //PAGINACION

cleardevice(); //LIMPIA LA PANTALLA

return(1); //REGRESA A LA FUNCION QUE LA LLAMO

break;

}

setfillstyle(1,BLACK); //ESTABLECE EL ESTILO DE RELLENO A SOLIDO COLOR NEGRO

bar(XCOO-150,YCOO-250,XCOO+240,YCOO+130); //CON ESTE RECTANGULO

//BORRAMOS LO QUE SE PINTO DEL CONTENEDOR PARA QUE NO SE QUEDEN

//PINTADOS ALGUNOS CUBOS MIENTRAS SE VAN MOVIENDO

dibujaContenedor(contenedor,XCOO,YCOO); //DIBUJA EL CONTENEDOR

delay(30);

}

else //SI NO SE PRESIONO NINGUNA TECLA

{

setfillstyle(1,BLACK); //BORRAMOS RASTROS

bar(XCOO-150,YCOO-250,XCOO+240,YCOO+130);

dibujaContenedor(contenedor,XCOO,YCOO); //SE PINTA EL CONTENEDOR

delay(30);

}if((\*(\*(\*(contenedor+NUM-1)+2)+0)).existe==4)//SI SE CUMPRE QUIERE DECIR QUE NO

{punt+=10; //CADA VEZ QUE ENTRA SE AUMENTAN LOS PUNTOS

//SE ATINO AL AGUJERO Y SE FINALIZA LA PARTIDA

if(pcuad(contenedor)) {

R.puntos=punt; //SE AGREGARN LOS RECORDS EN UN ARREGLO

R.tiempo=d; //SE COPIAN LOS PUNTOS

strcpy(R.nombre,nom); //SE COPIA LA CADENA DE CARACTERES

setvisualpage(pagina);

cleardevice(); //LIMPIA LA PANTALLA

outtextxy(150,200,"FIN DEL JUEGO"); //IMPRIME EL FINAL DEL JUEGO

getch();

inicializa\_records();

agrega\_records(R)

cleardevice();

muestra\_records(); //MUESTRA LOS RECORDS

getch();

setvisualpage(1); //FUNCIONES PARA LIMPIAR LA PAGINACION

setactivepage(1);

cleardevice();

setvisualpage(0);

setactivepage(0);

cleardevice();

return(1); //REGRESAMOS AL MENU

}

}

}

}

ESTA FUNCION SIRVE PARA CAPTURAR EL NOMBRE DEL JUGADOR

char\* captura\_nombre()

{

char \*cadena; //APUNTADOR A CHAR PARA GUARDAR EL NOMBRE

int out;

setcolor(BLUE); //DEFINIMOS EL COLOR CON EL QUE VAMOS A MOSTRAR LAS LETRAS

settextstyle(1,0,2); //DEFINIMOS EL TIPO DE LETRA

cadena=(char \*)malloc(sizeof(char)\*20); //ASIGNAMOS MEMORIA A LA CADENA

out=inttextxy(100,50,cadena,20); //MANDAMOS LLAMAR A LA FUNCION

cleardevice(); //CON LA QUE CAPTURAMOS EL TEXTO

return(cadena); //REGRESAMOS LA DIRECCION DONDE

} //GUARDAMOS LA CADENA

int inttextxy(int x, int y, char \*cad, int max) //X, Y SON COORDENADAS DE DONDE VAMOS A

{ //IMPRIMIR EL NOMBRE MIENTRAS SE ESTA CAPTURANDO

char car; //VARIABLE PARA GUARDAR UN CARACTER

int pos=0; //POSICION EN LA CADENA

(\*(cad+0))='\0'; //PRIMER CARACTER DE LA CADENA IGUAL A CARACTER NULO

do //CICLO PARA CAPTURAR CARACTERES

{

outtextxy(x-20,y,"Nombre:"); //IMPRESIÓN DE LA PALABRA “NOMBRE:”

car=leetecla(); //CAPTURA DE UNA TECLA

switch(car) //LEEMOS QUE SE TECLEO

{

case 13:return(1); //SI PRESIONAN ENTER SALIMOS DE LA FUNCION

break;

case 27:(\*(cad+0))='\0'; //SI PRESIONAN ESC SALIMOS DE LA FUNCION

return(0);

break;

case 8: //SI PRESIONAN “BACKSPACE” BORRAMOS UN CARACTER

if(pos>=0) //DE LA CADENA

{

pos--;

(\*(cad+pos))='\0';

}

break;

default: //SI SE PRESIONO CUALQUIER OTRA TECLA

if(pos<max-1) //SI LA CADENA NO ESTA LLENA

{

(\*(cad+pos))=car; //COPIAMOS EL CARÁCTER EN LA CADENA

pos++; //AUMENTAMOS LA POSICION DE LA CADENA

(\*(cad+pos))='\0'; //ASIGNAMOS EL CARÁCTER NULO A LA SIGUIENTE

} //POSICION DE LA CADENA

break;

}

cleardevice(); //LIMPIAMOS LA PANTALLA

outtextxy(x+110,y,cad); //IMPRIMIMOS LA CADENA

}while(pos<max); //MIENTRAS LA POSICION SEA MENOR MAX

return(1); //REGRESAMOS A LA FUNCION

}

ESTA FUNCION PINTA EL CONTENEDOR EN LA PANTALLA

void dibujaContenedor(Tcubo\*\*\* cont,int x, int y) //X Y Y SON COORDENADAS

{

int i,j,k;

setcolor(BLACK); //DEFINIMOS EL COLOR

line(x+TAM/2,-15+y+TAM/2,x+TAM/2,y-(NUM\*TAM)+TAM/2);

for(k=0;k<NUM;k++)

{

for(j=0;j<NUM;j++)

{

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

{

if((\*(\*(\*(cont+k)+j)+i)).existe!=0) //SI EL CUBO EXISTE(ES DIFERENTE DE CERO)

{

setfillstyle(1,(\*(\*(\*(cont+k)+j)+i)).color); //DEFINIMOS EL TIPO DE RELLENO

cubo(x+((\*(\*(\*(cont+k)+j)+i))).x,y+((\*(\*(\*(cont+k)+j)+i))).y); //FUNCION CUBO

}

}

}

}

contornoContenedor(x,y,NUM); //FUNCION QUE PINTA LOS VERTICES DEL CONTENEDOR

}

ESTA FUNCION PINTA UN CUBO, RECIBE LAS COORDENADAS EN X Y EN Y; LA VARIABLE TAM ES UNA CONSTANTE SIMBOLICA.

void cubo( int x, int y)

{

int \*punto;

punto=(int\*)malloc(sizeof(int)\*8); //ASIGNACION DE MEMORIA, EL ARREGLO GUARDA LAS

//COORDENADAS DE UNA TAPA DEL CUBO (NECESITA 3)

//TAPA FRONTAL//

\*(punto+0)=x;

\*(punto+1)=y;

\*(punto+2)=x+TAM;

\*(punto+3)=y;

\*(punto+4)=x+TAM;

\*(punto+5)=y+TAM;

\*(punto+6)=x;

\*(punto+7)=y+TAM;

fillpoly(4,punto); //FUNCION QUE IMPRIME UN POLIGONO DE LOS VERTICES QUE LE

//INDIQUEMOS

//TAPA SUPERIOR//

\*(punto+0)=x;

\*(punto+1)=y;

\*(punto+2)=x+TAM/2;

\*(punto+3)=y-TAM/2;

\*(punto+4)=x+TAM+TAM/2;

\*(punto+5)=y-TAM/2;

\*(punto+6)=x+TAM;

\*(punto+7)=y;

fillpoly(4,punto);

//TAPA LATERAL//

\*(punto+0)=x+TAM;

\*(punto+1)=y;

\*(punto+2)=x+TAM+TAM/2;

\*(punto+3)=y-TAM/2;

\*(punto+4)=x+TAM+TAM/2;

\*(punto+5)=y+TAM/2;

\*(punto+6)=x+TAM;

\*(punto+7)=y+TAM;

fillpoly(4,punto);

}

FUNCION QUE IMPRIME LOS VERTICES DEL CONTENEDOR

void contornoContenedor(int x,int y,int p)

{

setcolor(WHITE);

line(x-(p\*TAM/2)+TAM/2,y-(p\*TAM/2)+TAM/2,x+(p\*TAM/2)+TAM/2,y-(p\*TAM/2)+TAM/2);

line(x-(p\*TAM/2)+TAM/2,y+(p\*TAM/2)+TAM/2,x-(p\*TAM/2)+TAM/2,y-(p\*TAM/2)+TAM/2);

line(x+(p\*TAM/2)+TAM/2,y+(p\*TAM/2)+TAM/2,x+(p\*TAM/2)+TAM/2,y-(p\*TAM/2)+TAM/2);

line(x+(p\*TAM)+TAM/2,y+TAM/2,x+(p\*TAM)+TAM/2,y-(p\*TAM)+TAM/2);

line(x+TAM/2,y-(p\*TAM)+TAM/2,x+(p\*TAM)+TAM/2,y-(p\*TAM)+TAM/2);

line(x+TAM/2,y-(p\*TAM)+TAM/2,x-(p\*TAM/2)+TAM/2,y-(p\*TAM/2)+TAM/2);

line(x+(p\*TAM)+TAM/2,y-(p\*TAM)+TAM/2,x+(p\*TAM/2)+TAM/2,y-(p\*TAM/2)+TAM/2);

}

FUNCION QUE INICIALIZA LA PARED DE CUBOS

void inicializa\_pared(Tcubo\*\*\* cont, int \*c, int v) //RECIBE LA VARIABLE ALEATORIA DONDE APARECE

{ //EL AGUJERO DE LA PARED, Y UNA VARIABLE v

int i,j; //QUE INDICA CUAL FIGURA VA A APARECER

\*c=rand()%(NUM-2); //CREA ALEATORIAMENTE EL AGUJERO

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

for(j=0;j<NUM;j++)

{

(\*(\*(\*(cont+i)+NUM-1)+j)).existe=4; //PINTA LA PARED EN LA PARTE SUPERIOR

(\*(\*(\*(cont+i)+NUM-1)+j)).color=4;

(\*(\*(\*(cont+i)+0)+j)).existe=1;

(\*(\*(\*(cont+i)+0)+j)).color=2;

}

switch(v) //SEGUN EL NUMERO QUE RECIBA VA A INICIAR UNA FIGURA DIFERENTE

{

case 0:

cuadrado(cont,c); //CUARDRADO

break;

case 1:

linea(cont,c); //LINEA DE 4 CUBOS

break;

case 2:

letraL(cont,c); //LETRA L

break;

case 3:

letraT(cont,c); // LETRA T

break;

}

}

FUNCION QUE MUEVE A PARED HACIA ABAJO

void actualiza\_pared(Tcubo\*\*\* cont, int \*p)

{

int i, j, k;

for(k=0;k<NUM;k++)

{

for(j=0;j<NUM;j++)

{

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

{

if((\*(\*(\*(cont+k)+j)+i)).existe==4) //VERIFICA QUE HAYA PARED

{

if(j>2) //SI SE CUMPLE LA PARED CAE

{

(\*(\*(\*(cont+k)+j)+i)).existe=0;

(\*(\*(\*(cont+k)+j-1)+i)).color=(\*(\*(\*(cont+k)+j)+i)).color;

(\*(\*(\*(cont+k)+j-1)+i)).existe=4;

}

else //DESAPARECE LA PARED

{

(\*(\*(\*(cont+k)+j)+i)).existe=0;

(\*(\*(\*(cont+k)+0)+i)).color=2;

}

}

}

}

}

}

FUNCION QUE MUEVE AL JUGADOR HACIA LA IZQUIERDA

void mueve\_izquierda(Tcubo\*\*\* cont) //JUGADOR EXISTE=2

{

int i, j, k;

for(k=0;k<NUM;k++)

{

for(j=0;j<NUM;j++)

{

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

{

if((\*(\*(\*(cont+k)+j)+i)).existe==2) //SI ENCUENTRA UN CUBO DEL JUGADOR

{

(\*(\*(\*(cont+k)+j)+i-1)).existe=2; //LO MUEVE HACIA LA IZQUIERDA

(\*(\*(\*(cont+k)+j)+i)).existe=0;

(\*(\*(\*(cont+k)+j)+i-1)).color=(\*(\*(\*(cont+k)+j)+i)).color;

}

}

}

}

}

FUNCION QUE MUEVE AL JUGADOR HACIA LA DERECHA

void mueve\_derecha(Tcubo\*\*\* cont)

{

int i, j, k;

for(k=0;k<NUM;k++)

{

for(j=0;j<NUM;j++)

{

for(i=NUM;i>=0;i--)

{

if((\*(\*(\*(cont+k)+j)+i)).existe==2) //SI ENCUENTRA UN CUBO DEL JUGADOR

{

(\*(\*(\*(cont+k)+j)+i+1)).existe=2; //LO MUEVE HACIA LA DERECHA

(\*(\*(\*(cont+k)+j)+i)).existe=0;

(\*(\*(\*(cont+k)+j)+i+1)).color=(\*(\*(\*(cont+k)+j)+i)).color;

}

}

}

}

}

FUNCION QUE MUEVE AL JUGADOR HACIA LA ADELANTE

void mueve\_adelante(Tcubo\*\*\* cont)

{

int i, j, k;

for(k=0;k<NUM;k++)

{

for(j=0;j<NUM;j++)

{

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

{

if((\*(\*(\*(cont+k)+j)+i)).existe==2) //SI ENCUENTRA UN CUBO DEL JUGADOR

{

(\*(\*(\*(cont+k-1)+j)+i)).existe=2; //LO MUEVE HACIA ADELANTE

(\*(\*(\*(cont+k)+j)+i)).existe=0;

(\*(\*(\*(cont+k-1)+j)+i)).color=(\*(\*(\*(cont+k)+j)+i)).color;

}

}

}

}

}

FUNCION QUE MUEVE AL JUGADOR HACIA LA ATRAS

void mueve\_atras(Tcubo\*\*\* cont)

{

int i, j, k;

for(k=NUM-1;k>=0;k--)

{

for(j=0;j<NUM;j++)

{

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

{

if((\*(\*(\*(cont+k)+j)+i)).existe==2) //SI ENCUENTRA UN CUBO DEL JUGADOR

{

(\*(\*(\*(cont+k+1)+j)+i)).existe=2; //LO MUEVE HACIA ATRAS

(\*(\*(\*(cont+k)+j)+i)).existe=0;

(\*(\*(\*(cont+k+1)+j)+i)).color=(\*(\*(\*(cont+k)+j)+i)).color;

}

}

}

}

}

FUNCION QUE GIRA EL CONTENEDOR HACIA LA DERECHA

void giraDerecha(Tcubo\*\*\* cont)

{

Tcubo\*\*\*c; //CONTENEDOR AUXILIAR PARA COPIAR TODO LO QUE TIENE EL ORIGINAL

int i,j,k;

c=reservaContenedor(); //ASIGNACION DE MEMORIA AL CUBO AUXILIAR

for(k=0;k<NUM;k++)

for(j=0;j<NUM;j++)

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

{ //AQUI SE COPIA EL CONTENEDOR ORIGINAL AL TEMPORAL

(\*(\*(\*(c+k)+j)+i)).existe=(\*(\*(\*(cont+k)+j)+i)).existe;

(\*(\*(\*(c+k)+j)+i)).color=(\*(\*(\*(cont+k)+j)+i)).color;

}

for(k=0;k<NUM;k++)

for(j=0;j<NUM;j++)

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

{ //AQUI ES DONDE SE COPIA EL CONTENIDO DEL CONTENEDOR

//DE CIERTA MANERA QUE EN EL ORIGINAL QUEDA GIRADO

(\*(\*(\*(cont+NUM-i-1)+j)+k)).existe=(\*(\*(\*(c+k)+j)+i)).existe;

(\*(\*(\*(cont+NUM-i-1)+j)+k)).color=(\*(\*(\*(c+k)+j)+i)).color;

}

liberaContenedor(c); //LIBERACION DE MEMORIA PARA EL CONTENEDOR TEMPORAL

}

FUNCION QUE GIRA EL CONTENEDOR HACIA LA IZQUIERDA

FUNCIONA IGUAL QUE EL ANTERIOR SOLO CAMBIA LA FORMA EN QUE SE COPIAN LOS DATOS DEL CONTENEDOR TEMPORAL AL ORIGINAL

void giraIzquierda(Tcubo\*\*\* cont)

{

Tcubo\*\*\*c;//auxiliar

int i,j,k;

c=reservaContenedor();

//respaldo de contenedor

for(k=0;k<NUM;k++)

for(j=0;j<NUM;j++)

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

{

(\*(\*(\*(c+k)+j)+i)).existe=(\*(\*(\*(cont+k)+j)+i)).existe;

(\*(\*(\*(c+k)+j)+i)).color=(\*(\*(\*(cont+k)+j)+i)).color;

}

for(k=0;k<NUM;k++)

for(j=0;j<NUM;j++)

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

{

(\*(\*(\*(cont+i)+j)+NUM-k-1)).existe=(\*(\*(\*(c+k)+j)+i)).existe

(\*(\*(\*(cont+i)+j)+NUM-k-1)).color=(\*(\*(\*(c+k)+j)+i)).color;

}

liberaContenedor(c);

}

FUNCION QUE CREA EL ARCHIVO DE LOS RECORDS SI ESTE NO SE ENCUENTRA EN EL DIRECTORIO DEL JUEGO

void inicializa\_records()

{

int i;

Records R={"Anonimo",100,0}; //CREA UNA ESTRUCTURA DE LOS RECORDS

FILE \*fp; //APUNTADOR A FILE

fp=fopen("records.dat","rb"); //ABRE EL ARCHIVO DE LOS RECORDS

if(fp!=NULL) //SI NO SE ENCUENTRA EL ARCHIVO

{

fclose(fp);

outtextxy(150,50,"Error al abrir el archivo de los records"); //MUESTRA EL ERROR

getch();

cleardevice();

return; //REGRESA A LA FUNCION

}

fp=fopen("records.dat","wb"); //CREA UN ARCHIVO NUEVO PARA CREAR LOS RECORDS

if(fp==NULL) //SI NO LO PUEDE CREAR

{

outtextxy(150,50,"Error al crear el archivo"); //MUESTRA EL ERROR

getch();

cleardevice();

return; //REGRESA A LA FUNCION

}

for(i=0;i<5;i++) //SI LO PUDO CREAR

{

fwrite(&R,sizeof(Records),1,fp); //LE ESCRIBE LOS DATOS QUE TIENE R

}

fclose(fp); //CIERRA EL ARCHIVO

}

FUNCION QUE MUESTRA LOS RECORDS QUE ESTAN GUARDADOS EN EL ARCHIVO

void muestra\_records()

{

char cad[40]; //CADENA QUE SIRVE PARA IMPRIMIR EN PANTALLA LOS RECORDS

int i=100;

Records R; //VARIABLE TIPO RECORDS

FILE \*fp;

fp=fopen("records.dat","rb"); //ABRE EL ARCHIVO PARA SU LECTURA

if(fp==NULL) //SI NO LO PUDO ABRIR

{

inicializa\_records(); //CREA UNO NUEVO CON ESTA FUNCION

muestra\_records(); //MUESTRA LOS RECORDS

}

else //SI LO PUDO ABRIR

{

fread(&R,sizeof(Records),1,fp); //LEE LO QUE TIENE EL ARCHIVO Y LO GUARDA EN R

outtextxy(300,60,"RECORDS"); //IMPRIME LA PALABRA “RECORDS”

while(!feof(fp)) //MIENTRAS NO SEA FINAL DE ARCHIVO

{

sprintf(cad, "Nombre: %s Puntos: %d Tiempo: %02d:%02d:%02d", R.nombre, R.puntos, R.tiempo/3600, R.tiempo%3600/60, R.tiempo%3600%60);

//IMPRIME EL NOMBRE, LOS PUNTOS Y EL TIEMPO EN LA CADENA

outtextxy(50,i,cad); //IMPRIME LA CADENA EN PANTALLA

fread(&R,sizeof(Records),1,fp); //LEE EL SIGUIENTE RECORD

i+=30;

}

getch();

cleardevice();

}

fclose(fp); //CIERRA EL ARCHIVO

}

FUNCION QUE AGREGA LOS RECORDS AL FINAL DEL JUEGO

void agrega\_records(Records r) //RECIBE EL NUEVO RECORD

{

Records registros[NR]; //CREA EL ARREGLO DE RECORDS

int i;

FILE \*f; //VARIABE ARCHIVO

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

{ strcpy(registros[i].nombre," "); //INICIALIZA EL ARREGLO

registros[i].puntos=0;

}

f=fopen("Records.dat","rb+"); //ABRE EL ARCHIVO

if(f==NULL) //SI NO SE PUDO ABRIR MANDA ERROR

{ printf("No se encontro el archivo");

registros[0]=r;

f=fopen("Records.dat","wb");

fwrite(registros,sizeof(Records),NR,f);

fclose(f);

}

else

{

fread(registros,sizeof(Records),NR,f); //SE LEEN LOS RECORD GUARDADOS Y LOS

//ACOMODA DE MAYOR A MENOR

for(i=NR-1;i>=0;i--)

{ if(registros[i].puntos<=r.puntos && i+1<NR)

registros[i+1]=registros[i];

else

if(registros[i].puntos>=r.puntos && i+1<NR)

{ registros[i+1]=r;

i=-1;

}

}

if(registros[0].puntos<=r.puntos)

registros[0]=r;

rewind(f);

fwrite(registros,sizeof(Records),NR,f);

fclose(f);

}

}

ESTA FUNCION MUESTRA LA AYUDA EN PANTALLA DESDE UN ARCHIVO DE TEXTO

void ayuda()

{

char C[80]; //CADENA DONDE SE GUARDA CADA RENGLON

FILE \*fp; // VARIABLE DE TIPO ARCHIVO

int y=25; //SEPARACION ENTRE CADA RECORD

fp=fopen("ayuda.txt","r"); //SE ABRE EL ARCHIVO DE AYUDA

settextstyle(3,0,1);

if(fp==NULL) //SI NO SE PUEDE ABRIR

{

outtextxy(20,20,"No se puede abrir el archivo"); //SE MUESTRA EL ERROR

}

else //SI LO PUDO ABRIR

{

while(!feof(fp)) //MIENTRAS NO SEA FIN DE ARCHIVO

{

fgets(C,79,fp); //COPIA LA CADENA DESDE EL ARCHIVO A LA VARIABLE

outtextxy(20,y,C); //MUESTRA EN PANTALLA LA CADENA

y+=25;

}

}

getch();

cleardevice(); //LIMPIA LA PANTALLA

}

ESTAS SON LAS FUNCIONES PARA INICIALIZAR LAS DIFERENTES FIGURAS EN LA PARED QUE VA CAYENDO

void cuadrado(Tcubo \*\*\*cont, int \*c) //LA FIGURA ES UN CUADRADO, RECIBE EL VALOR \*c POR

{ //POR REFERENCIA, ESTA SIRVE PARA INDICAR DONDE

srand(time(NULL)); //VA A APARECER LA FIGURA EN FORMA ALEATORIA

\*c=rand()%(NUM-3);

(\*(\*(\*(cont+\*c)+NUM-1)+\*c)).existe=5;

(\*(\*(\*(cont+\*c)+0)+\*c)).existe=5;

(\*(\*(\*(cont+\*c+1)+NUM-1)+\*c)).existe=5; //DAN LA FORMA AL CUBO

(\*(\*(\*(cont+\*c+1)+0)+\*c)).existe=5;

(\*(\*(\*(cont+\*c)+NUM-1)+\*c+1)).existe=5;

(\*(\*(\*(cont+\*c)+0)+\*c+1)).existe=5;

(\*(\*(\*(cont+\*c+1)+NUM-1)+\*c+1)).existe=5;

(\*(\*(\*(cont+\*c+1)+0)+\*c+1)).existe=5;

}

void linea(Tcubo \*\*\*cont, int \*c) //LA FIGURA ES UNA LINEA

{

srand(time(NULL)); \*c=rand()%(NUM-3); //DE IGUAL MANERA SE PASA LA VARIABLE \*c

(\*(\*(\*(cont+\*c)+NUM-1)+\*c)).existe=5; // POR REFERENCIA PARA APARECER LA FIGURA

(\*(\*(\*(cont+\*c)+0)+\*c)).existe=5; //DE FORMA ALEATORIA

(\*(\*(\*(cont+\*c+1)+NUM-1)+\*c)).existe=5;

(\*(\*(\*(cont+\*c+1)+0)+\*c)).existe=5;

(\*(\*(\*(cont+\*c+2)+NUM-1)+\*c)).existe=5; //DAN LA FORMA A LA FIGURA

(\*(\*(\*(cont+\*c+2)+0)+\*c)).existe=5;

(\*(\*(\*(cont+\*c+3)+NUM-1)+\*c)).existe=5;

(\*(\*(\*(cont+\*c+3)+0)+\*c)).existe=5;

}

void letraL(Tcubo \*\*\*cont, int \*c) //LA FIGURA ES UNA LETRA L

{

srand(time(NULL)); //RECIBE \*c DE FORMA ALEATORIA PARA APARECER EL HUECO

\*c=rand()%(NUM-2); //EN LA PARED

(\*(\*(\*(cont+\*c)+NUM-1)+\*c)).existe=5;

(\*(\*(\*(cont+\*c)+0)+\*c)).existe=5;

(\*(\*(\*(cont+\*c)+NUM-1)+\*c+1)).existe=5;

(\*(\*(\*(cont+\*c)+0)+\*c+1)).existe=5;

(\*(\*(\*(cont+\*c)+NUM-1)+\*c+2)).existe=5; //DA LA FORMA DE UNA LETRA L

(\*(\*(\*(cont+\*c)+0)+\*c+2)).existe=5;

(\*(\*(\*(cont+\*c+1)+NUM-1)+\*c+2)).existe=5;

(\*(\*(\*(cont+\*c+1)+0)+\*c+2)).existe=5;

}

void letraT(Tcubo \*\*\*cont, int \*c) //LA FIGURA ES UNA LETRA T

{

srand(time(NULL)); //REIBE LA VARIABLE \*c POR REFERENCIA PARA APARECER LA

\*c=rand()%(NUM-3); //FIGURA DE FORMA ALEATORIA EN LA PARED

(\*(\*(\*(cont+\*c)+NUM-1)+\*c)).existe=5;

(\*(\*(\*(cont+\*c)+0)+\*c)).existe=5;

(\*(\*(\*(cont+\*c)+NUM-1)+\*c+1)).existe=5;

(\*(\*(\*(cont+\*c)+0)+\*c+1)).existe=5;

(\*(\*(\*(cont+\*c)+NUM-1)+\*c+2)).existe=5; //DAN LA FORMA DE UNA LETRA T

(\*(\*(\*(cont+\*c)+0)+\*c+2)).existe=5;

(\*(\*(\*(cont+\*c-1)+NUM-1)+\*c+1)).existe=5;

(\*(\*(\*(cont+\*c-1)+0)+\*c+1)).existe=5;

}

LA SIGUIENTE FUNCION ES PARA DETERMINAR SI NO SE ATINA AL HUECO DE LA PARED DE CUBOS, SI NO SE ATINA, EL JUEGO REGRESA UN 1 QUE INDICA QUE SE HA PERDIDO EL JUEGO

int pcuad(Tcubo \*\*\*contenedor)

{

int j,k,res=0; //VARIABLES PARA EL CICLO DE BUSQUEDA DEL JUGADOR

for(j=0;j<TAM&&(\*(\*(\*(contenedor+j)+1)+k)).existe!=2;j++)

for(k=0;(\*(\*(\*(contenedor+j)+1)+k)).existe!=2&&k<TAM;k++);

if((\*(\*(\*(contenedor+j-1)+2)+k)).existe!=4&&(\*(\*(\*(contenedor+j-1)+2)+k)).existe==4)

return(res);

return(res+1);

}

LAS SIGUIENTES FUNCIONES SON LAS QUE INICIALIZAN AL JUGADOR EN LA FUNCION JUEGO CON LAS DIFERENTES FIGURAS QUE HAY EN EL JUEGO, TODAS RECIBEN LAS VARIABLES DEL CONTENEDOR \*\*\*cont, \*i Y \*d QUE INDICAN LAS COORDENADAS DEL JUGADOR PARA QUE CUANDO APAREZCA UNA NUEVA FIGURA SE UBIQUE EL EL LUGAR DONDE ESTABA LA ANTERIOR

void inicializa\_jugador(Tcubo\*\*\* cont, int \*i, int \*d)//FIGURA DE UN CUADRADO

{

(\*(\*(\*(cont+\*i)+1)+\*d)).existe=2;

(\*(\*(\*(cont+\*i)+1)+\*d)).color=7;

(\*(\*(\*(cont+\*i)+1)+\*d+1)).existe=2;

(\*(\*(\*(cont+\*i)+1)+\*d+1)).color=7;

(\*(\*(\*(cont+\*i+1)+1)+\*d+1)).existe=2;

(\*(\*(\*(cont+\*i+1)+1)+\*d+1)).color=7;

(\*(\*(\*(cont+\*i+1)+1)+\*d)).existe=2;

(\*(\*(\*(cont+\*i+1)+1)+\*d)).color=7;

}

void jugadorlinea(Tcubo\*\*\* cont,int \*i,int \*d)//LA FIGURA ES UNA LINEA

{

(\*(\*(\*(cont+\*i)+1)+\*d)).existe=2;

(\*(\*(\*(cont+\*i)+1)+\*d)).color=7;

(\*(\*(\*(cont+\*i+1)+1)+\*d)).existe=2; //DA FORMA Y COLOR A LA LINEA

(\*(\*(\*(cont+\*i+1)+1)+\*d)).color=7;

(\*(\*(\*(cont+\*i+2)+1)+\*d)).existe=2;

(\*(\*(\*(cont+\*i+2)+1)+\*d)).color=7;

(\*(\*(\*(cont+\*i+3)+1)+\*d)).existe=2;

(\*(\*(\*(cont+\*i+3)+1)+\*d)).color=7;

}

void jugadorL(Tcubo\*\*\* cont, int \*i, int \*d)//LA FGURA ES LA LETRA L

{

(\*(\*(\*(cont+\*i)+1)+\*d)).existe=2;

(\*(\*(\*(cont+\*i)+1)+\*d)).color=7;

(\*(\*(\*(cont+\*i)+1)+\*d+1)).existe=2;

(\*(\*(\*(cont+\*i)+1)+\*d+1)).color=7; //DAN FORMA Y COLOR A LA FIGURA

(\*(\*(\*(cont+\*i)+1)+\*d+2)).existe=2;

(\*(\*(\*(cont+\*i)+1)+\*d+2)).color=7;

(\*(\*(\*(cont+\*i+1)+1)+\*d+2)).existe=2;

(\*(\*(\*(cont+\*i+1)+1)+\*d+2)).color=7;

}

void jugadorT(Tcubo\*\*\* cont, int \*i, int \*d)//LA FIGURA ES UNA LETRA T

{

(\*(\*(\*(cont+\*i)+1)+\*d)).existe=2;

(\*(\*(\*(cont+\*i)+1)+\*d)).color=7;

(\*(\*(\*(cont+\*i)+1)+\*d+1)).existe=2;

(\*(\*(\*(cont+\*i)+1)+\*d+1)).color=7; //DA FORMA Y COLOR A LA FIGURA

(\*(\*(\*(cont+\*i)+1)+\*d+2)).existe=2;

(\*(\*(\*(cont+\*i)+1)+\*d+2)).color=7;

(\*(\*(\*(cont+\*i-1)+1)+\*d+1)).existe=2;

(\*(\*(\*(cont+\*i-1)+1)+\*d+1)).color=7

}