

## Entrega I

Decidimos separar el juego en varios archivos a fin de aislar los problemas y mejorar la navegacion del codigo. Más adelante se describiran cada uno de los archivos.

Nuestra matriz es un vector largo **lamatriz [26][7][2]** en donde las posiciones(i) funcionan con **lamatriz[1...25]**, en **lamatriz[i][0][0]** se guarda el numero del dado en esa posicion (j) funcionan con **lamatriz[i][0...6]**, en la posicion **lamatriz[i][j][0]** se guarda a quien le pertenece el dado de valor (j), si es 1 es al humano y si es 2 es a la máquina.

## TPCphalopoid.c

Es el archivo principal donde se inicia el juego, y donde está la **función** que verifica el ganador del juego. El usuario elige si ir primero o si la computadora va primero. Luego entramos a un **while** que repetira los turnos de jugador y computadora.

## Jugador.c/.h

- void dadoUno(int);
- void dadoMas(int,int,int);
- void sumarDado(int,int);
- int validarJugada(void);
- int jugador(int);
- void ponerDadojugador(int);

Se le pide al jugador una posicion, luego en la función **validarJugada** con un **while** se verifica mediante **if** y si es válido el dato introducido se llama a la función limpiarBuffer para luego llamar a la función **ponerDadojugador** el cual recibe como parámetro el valor de la posicion introducido por el usuario.

Se agrupa posiciones con características parecidas y se suma los dados alrededor de la posicion seleccionada. Se llama a la función **sumarDado** que permite al jugador elegir cuántos dados, que dados, y si se arrepiente o no de elegir los dados y los come y suma siempre y cuando la suma sea menor a 6. Se utilizaron 4 while y 2 for para lograr esto(no se representa en la foto porque la función **dadoMas** es muy larga). El algebra utilizado se basa en la manipulacion de las posiciones de la matriz donde si la jugada es “arriba” se le resta 5 a la posicion,”abajo” se le suma 5, “izquierda” se le resta uno y “derecha se le suma uno.

Si es que no existe un movimiento el cual permite aumentar el valor del dado, el dado puesto en el tablero será 1.

```
int derecha = lamatriz[jugada + 1][0][0];
int abajo = lamatriz[jugada + 5][0][0];
int izquierda = lamatriz[jugada - 1][0][0];
int arriba = lamatriz[jugada - 5][0][0];
int vector[] = { izquierda, derecha, arriba, abajo, '\0' };
int suma = 0;
int resp = 0;
int comercantidad = 0;
int terminar = 0;
int ref[] = { -1, +1, -5, +5, '\0' };

void dadoUno(int jugada) {
    lamatriz[jugada][0][0] = 1;
    lamatriz[jugada][1][0] = 1;
}

void sumarDado(int suma,int jugada){
    lamatriz[jugada][0][0] = suma;
    lamatriz[jugada][suma][0] = 1;
}
```

```
if (jugada == 6 || jugada == 11 || jugada == 16) {
    suma = derecha + arriba + abajo;
    int j = 0;
    for (int i = 0; i < 4; i++) {
        if (vector[i] == 0) {
            suma -= vector[i];
        }
        if (vector[i] != 0 && i != 0 && vector[i] != 6) {
            dif++;
            ubi[j] = ref[i];
            j++;
        }
    }
    if (suma < 7 && dif == 2) {
        sumarDado(suma, jugada);
        for (int i = 0; i < dif; i++) {
            lamatriz[jugada + ubi[i]][0][0] = '\0';
        }
    } else if (dif > 2) {
        int invalido = 1;
        dadoMas(invalido, jugada, dif);
    } else {
        dadoUno(jugada);
    }
}
```

## Computadora.c

Al entrar en computadora.c, lo primero que se ve es la función “ponerDado”, que recibe como parámetro la posición la cual la compu eligió usar, que veremos después cómo lo hace, y lo que hace es revisar si se puede sumar dados a su alrededor, y lo hace dividiendo en secciones:

- 1) las esquinas 1,5,21,25;



2) - las filas de arriba:  $2 < x < 5$ ; las filas de abajo  $21 < x < 25$ :



3) las columnas del costado izquierdo y derecho respectivamente:  $x == 6 \parallel 11 \parallel 16$  y  $x == 10 \parallel 15 \parallel 20$ :



4) y la la seccion mas grande que seria el bloque del medio:  $6 < x < 10; 11 < x < 15; 16 < x < 20$



Tuve que separar en secciones [así](#) por la forma en la que trabajé.

Ya dentro de la función, todo se basa en los if, si la “jugada”, que es el entero que dicta la posición que la compu eligió, esta en cierto rango, se entra en ese if, y si no, se pasa al siguiente if:

```
if((jugada>1)&&(jugada<5)){
```

(si no se cumple, se pasa al siguiente)

```
if((jugada==6)|| (jugada==11)|| (jugada==16)){ //comienza
```

y así sucesivamente.

Ahora, entremos al primer if:

```
if((jugada>1)&&(jugada<5)){  
  
    if (lamatriz[jugada - 1][0][0] != '\0')  
        contador++;  
    if (lamatriz[jugada + 1][0][0] != '\0')  
        contador++;  
    if (lamatriz[jugada + 5][0][0] != '\0')  
        contador++;
```

Si “jugada” esta entre la posición 2 o 4, se entra aqui; seguidamente, se entra a unos if que lo que hacen es mirar si alrededor suyo (en este caso a la izquierda, a la derecha, y abajo) hay algun dado (en este código, si una posición NO es ‘\0’, significa que hay un dado), y si se cumple, contador sube 1.

```
if(contador<2){  
    lamatriz[jugada][1][0]=2;  
}
```

Si el contador es menor que 2, no se suma, ya que, o hay solo un dado, o no hay ninguno, y eso no se puede sumar porque así son las reglas del juego.

```

if(contador==2){
    //si el contador llega a 2, solo sumara la posicion izq (jugada-1) y derecha(jugada+1), y revisa si se puede sumar para no pasa
    if(lamatriz[jugada+1][0][0]!='\0' && lamatriz[jugada-1][0][0]!='\0' && (lamatriz[jugada+1][0][0]+lamatriz[jugada-1][0][0]<7){
        lamatriz[jugada][0][0]=lamatriz[jugada+1][0][0]+lamatriz[jugada-1][0][0];

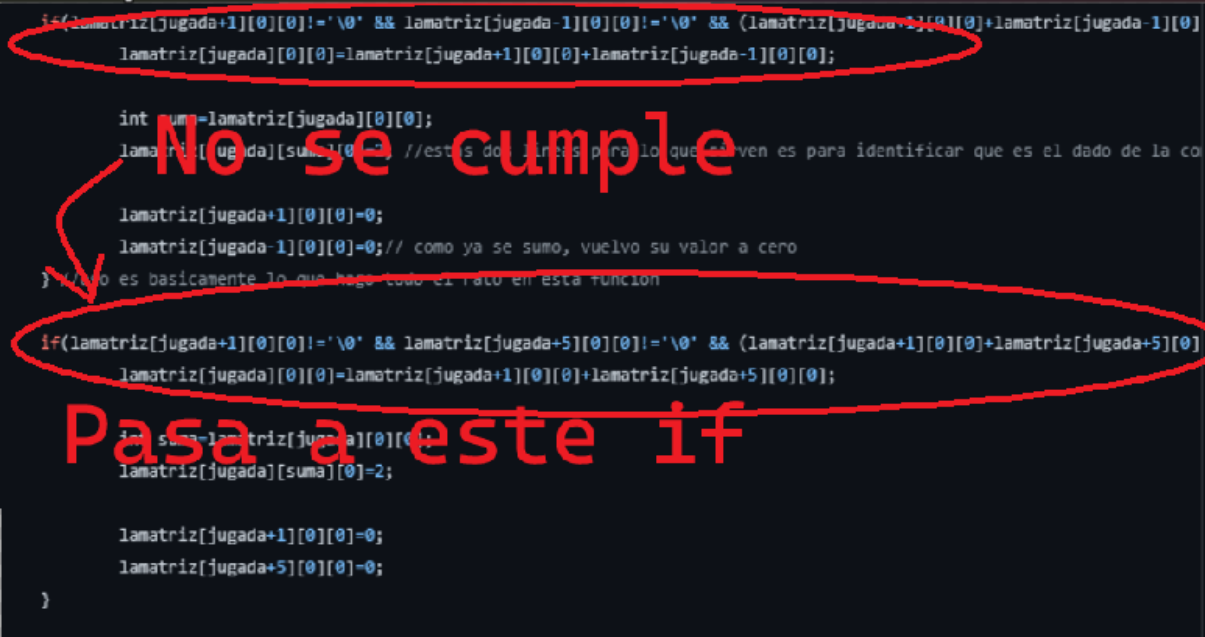
        int suma=lamatriz[jugada][0][0];
        lamatriz[jugada][suma][0]=2; //estas dos lineas para lo que sirven es para identificar que es el dado de la compu

        lamatriz[jugada+1][0][0]=0;
        lamatriz[jugada-1][0][0]=0; // como ya se sumo, vuelvo su valor a cero
    } //eso es basicamente lo que hago todo el rato en esta funcion
}

```

Ya si el contador es igual a 2, significa que sí ya hay dos dados, se pueden sumar, y cuando se entra al if, se verifica primero si las posiciones que hay a su alrededor no están vacías, por ejemplo en este if, se verifica si hay un dado a su derecha y a su izquierda, y después se verifica si la suma de esas dos posiciones son menores que 7, como dicen las reglas del juego. Si se verifican estas condiciones, se hace la suma, se pone que la suma en esa posición pertenece a la compu, y nosotros decidimos que sea el 2 su identificador, y por último, se reinician los valores los cuales se usó para sumar, ahora su valor es igual a cero.

Ahora imaginemos que no cumplió una de las condiciones, digamos que estaba vacía a la izquierda del dado, entonces se pasa al siguiente if:



```

if(lamatriz[jugada+1][0][0]!='\0' && lamatriz[jugada-1][0][0]!='\0' && (lamatriz[jugada+1][0][0]+lamatriz[jugada-1][0][0]<7){
    lamatriz[jugada][0][0]=lamatriz[jugada+1][0][0]+lamatriz[jugada-1][0][0];

    int suma=lamatriz[jugada][0][0];
    lamatriz[jugada][suma][0]=2; //estas dos lineas para lo que sirven es para identificar que es el dado de la compu

    lamatriz[jugada+1][0][0]=0;
    lamatriz[jugada-1][0][0]=0; // como ya se sumo, vuelvo su valor a cero
} //eso es basicamente lo que hago todo el rato en esta funcion

if(lamatriz[jugada+1][0][0]!='\0' && lamatriz[jugada+5][0][0]!='\0' && (lamatriz[jugada+1][0][0]+lamatriz[jugada+5][0][0]<7){
    lamatriz[jugada][0][0]=lamatriz[jugada+1][0][0]+lamatriz[jugada+5][0][0];

    int suma=lamatriz[jugada][0][0];
    lamatriz[jugada][suma][0]=2;

    lamatriz[jugada+1][0][0]=0;
    lamatriz[jugada+5][0][0]=0;
}

```

**No se cumple**

**Pasa a este if**

Ahora revisará si hay un dado para sumar a su derecha y a su izquierda.

Todo eso pasaba si el contador era igual a 2, ahora, si el contador es igual a 3, significa que la compu puede elegir o no si sumar todos o solo 2, y en esta entrega, la compu no sabe todavía pensar si le conviene o no sumar todos, entonces suma todo:

```

if(contador==3 && (lamatriz[jugada-1][0][0]+lamatriz[jugada+1][0][0]+lamatriz[jugada+5][0][0]<7){
    //si el contador llega a 3 significa que hay 3 dados para que pueda sumar
    lamatriz[jugada][0][0]=lamatriz[jugada-1][0][0]+lamatriz[jugada+1][0][0]+lamatriz[jugada+5][0][0];

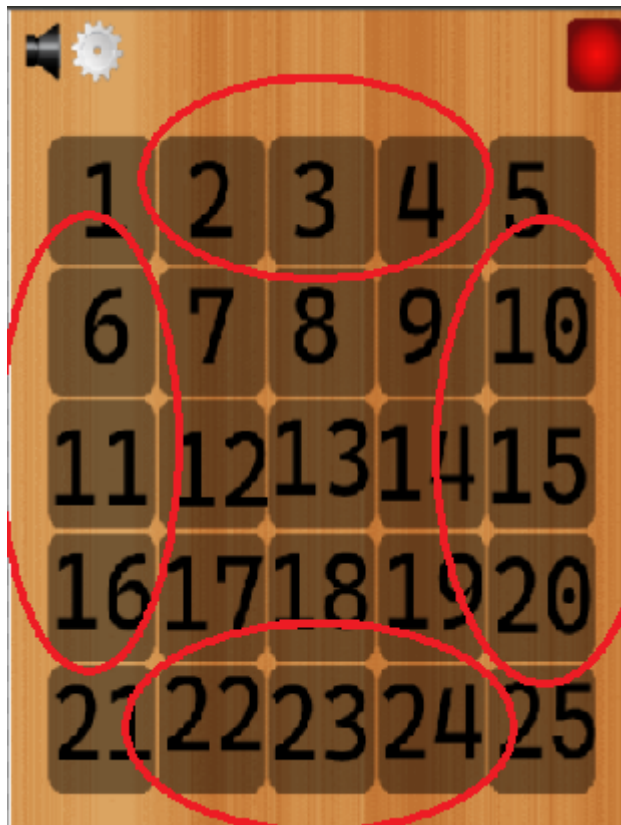
    int suma=lamatriz[jugada][0][0];
    lamatriz[jugada][suma][0]=2;

    lamatriz[jugada-1][0][0]=0;
    lamatriz[jugada+1][0][0]=0;
    lamatriz[jugada+5][0][0]=0;

    int jugada2=jugada+1;
    int jugada3=jugada-1;
    int jugada4=jugada+5; //estas declaraciones son solamente para poder imprimir como se ve a continuacion
    printf("La compu eligió sumar las posiciones %d,%d y %d\n",jugada2,jugada3,jugada4);
    contador=0;
    return;
}

```

Esta estructura de la función se cumple para las filas de arriba y abajo, las columnas de la izquierda y derecha, cada uno con sus respectivos límites.

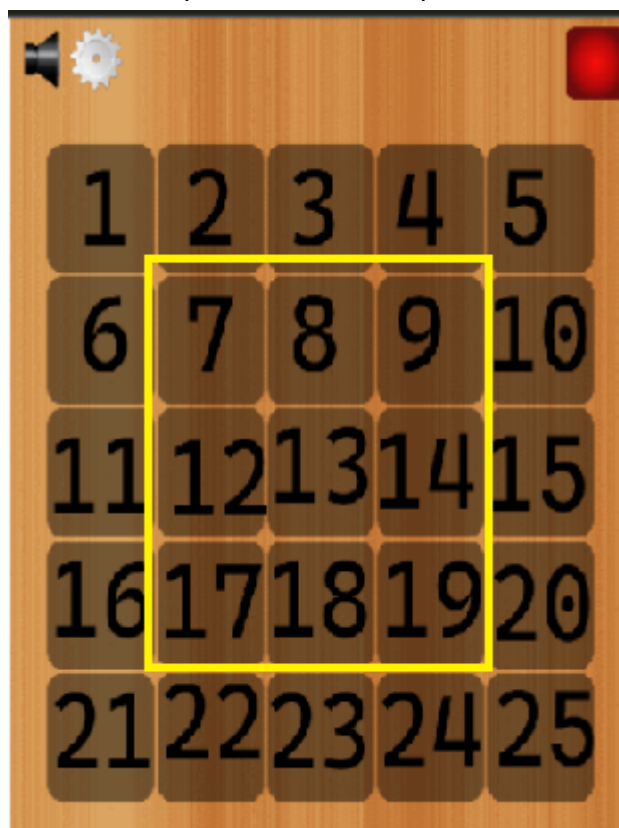


Para las esquinas, es todo mas simple ya que el contador nunca pasara de 2, y por ejemplo en la esquina de posicion 1, si se puede sumar algo, siempre se sumar la posicion 2 y 6:



Y así con todas las esquinas.

El unico escenario que falta es si la posicion en el bloque del medio:



Aquí, lo único que cambia es que el contador puede ser 2,3 o 4, osea puede sumar arriba, abajo, derecha e izquierda, pero la filosofía sigue igual. Y hay un



pequeño cambio que me solucionó ~~facilito~~facilitó bastante la ~~creacion~~creación de esta parte de la ~~funcion~~función:

```
if (lamatriz[jugada +5][0][0] != '\0'){
    arrayposiciones[contador]=jugada +5; //meto las p
    contador++;
}
if (lamatriz[jugada + 1][0][0] != '\0'){
    arrayposiciones[contador]=jugada +1;
    contador++;
}
if (lamatriz[jugada - 5][0][0] != '\0'){
    arrayposiciones[contador]=jugada -5;
    contador++;
}
if (lamatriz[jugada -1][0][0] != '\0'){
    arrayposiciones[contador]=jugada -1;
    contador++;
}
```

Al entrar en el if del bloque del medio, no solo se sube el ~~numero~~número del contador, si se encuentra un dado, la ~~posicion~~posición del dado se guarda en un array de posiciones, ~~asi~~así no tengo que hacer tantos if.

Y obviamente, si en la ~~funcion~~función no se cumple algunos if y se llega hasta el final, no se suma nada.

Eso fue la función “ponerDado”, ahora entramos al main “computadora”.

Ya al iniciar “computadora”, “turno” se vuelve 1, ya que “turno” es el valor que retorna la función “computadora”, y si turno es igual a 1, es turno del jugador, pero esto que estoy diciendo no pertenece a la función “computadora”.

Si el jugador elige que la compu debe jugar primero:

```
if(turnoinicial==2){

    srand(time(NULL));
    int numerorandom = rand() % 26;
    lamatriz[numerorandom][0][0]=1;
    printf("\n numero random %d",numerorandom);
    printf("\n posicion %d",lamatriz[numerorandom][0][0]);
    turnoinitial=1;
    return(turno); //en el caso de que el usuario quiera que la compu comience, tira uno random
}
```

se crea un número random hasta 25 y se mete un 1 en la posición del número random.

Si fue el jugador el que comenzó, la compu hace algo muy sencillo para determinar la posición en la que quiere poner:





Si es 7 donde el jugador eligió poner su dado, la compu revisará si se puede poner arriba, abajo, a la derecha o la izquierda, y si no puede, tirara un numero random para la posición:

```

srand(time(NULL));
int numerorandom = rand() % 25 + 1;
lamatriz[numerorandom][0][0]=1;
while(lamatriz[numerorandom][0][0]!='\0'){
    srand(time(NULL));
    numerorandom = rand() % 25 + 1;
    lamatriz[numerorandom][0][0]=1;
    printf("\n vale pene \n");

}
ponerDado(numerorandom);

return(turno);

```

Esto ya es al fondo de "computadora".

Como ciertas posiciones varían sus limites (osea, el jugador puso en la posición 1, no se puede poner en la posición -1 o -5, como si se puede en la posición 25, porque si ya existe ese límite en esa posición), hay muchos if, y se dividen como hice en las funciones:



Para cada fila o columna, un if diferente.

Y se devuelve turno en cada uno de los if:

```
if(posiocupada[1]==10 || posiocupada[1]==15 || posiocupada[1]==20 ){
    if(lamatriz[posiocupada[1]-5][0][0]=='\0'){
        lamatriz[posiocupada[1]-5][0][0]=1;
        ponerDado(posiocupada[1]-5);
        return(turno);
    }
    if(lamatriz[posiocupada[1]-1][0][0]=='\0'){
        lamatriz[posiocupada[1]-1][0][0]=1;
        ponerDado(posiocupada[1]-1);
        return(turno);
    }
    if(lamatriz[posiocupada[1]+5][0][0]=='\0'){
        lamatriz[posiocupada[1]+5][0][0]=1;
        ponerDado(posiocupada[1]+5);
        return(turno);
    }
}
```

Para que corte el if y sea turno del jugador.

Eso fue "computadora.c"

## Funciones .c/.h

Aquí se encuentran funciones de uso diverso que hacen lo que su nombre indica cómo limpiar el buffer, aplicar un ente a la consola, imprimir la matriz entre otros.

```
1 #ifndef FUNCIONES_H_
2 #define FUNCIONES_H_
3
4 void limpiarBuffer();
5 void vaciar_matriz();
6 void enter();
7
8 void imprimir_matriz();
9 int verificarTurno(int);
10 void tablaPosiciones();
11 void imprimirDecision();
12 void decirturnoini(int);
13 void decirparente();
14 int numeroAleatorio(int,int);
15 int verificarGanador(int);
16 void stats(char *cadena2, int gano);
17 int diceposisdispo();
18 int lugareslibres();
19
20 #endif /* FUNCIONES_H_ */
```

Las funciones más importantes que mencionar son:

- void **stats()** que maneja el uso de archivos en el programa
- int **lugareslibres()** es una función auxiliar para que la compu sepa cuáles lugares están libres y hacer elecciones en base a ello
- void **verificarGanador()** tiene la potestad de analizar la matriz en cada turno y dependiendo de la cantidad de dados decidir un ganador

## TP Final entrega 2

Nombres: Federico Alonso, Matias Ayala

Los cambios realizados fueron moderados, el archivo original junto con la función main original, ya que modularizamos bastante el código el trabajo fue de conectar las señales a los respectivos input de datos que antes se tomaban con scanf dentro de la consola.

Para esta entrega y simplificar el trabajo, hicimos que el Glade “vea” la matriz (la que teníamos en la primera entrega) y se actualice en torno a ella, suena mucho más fácil de lo que fue.

La función de sumar las casillas funcionan bien siempre y cuando la suma sea menor a 6 ya que esa función depende de mucho input del jugador y es difícil traducir en poco tiempo todo en una interfaz gráfica.

Escribir esta parte del código fue una experiencia estresante, larga y agotadora, pero fue interesante como Glade nos ayudaba a disminuir el tiempo de escritura del código..

Nos faltaría poner las jugadas en un documento, habilitar el modo computadora vs computadora, agregar estadísticas, agregar la función de juego nuevo (intentamos agregar ahora pero nos fue imposible) y agregar elementos decorativos a la interfaz.

### Entrega final

Para la entrega final nos dedicamos a la corrección de errores además de la escritura de archivos.

### Guía de utilización del programa

El programa cuenta con botones que hacen lo cual exponen, la interfaz es sencilla y muy humana.

Primero existen 6 botones toggleables los cuales establecen las opciones predeterminadas donde el juego se iniciara, al menos 2 de estos botones deben de ser seleccionados.

Luego existen 2 cuadros de texto que esperan recibir los nombres de los jugadores o jugador

En el juego el nombre del jugador principal se dispone arriba y el secundario abajo, se exponen además el número de dados actuales.

Abajo están los botones para manipular ciertas partes de la interfaz

Donde se incluyen reiniciar la partida, como jugar, créditos y estadísticas además de salir del juego

Al terminar la partida se expone el nombre del ganador