**Deal or No Deal**

**Trabajo Práctico I**

**Alumno:** Agustín Delger

**Profesor:** Jonathan L. Pepe

**Materia:** Programación II

**Carrera:** Desarrollo de videojuegos

**Año:** 2016

El desarrollo del juego puede dividirse en dos partes. Por un lado, la preparación del escenario de juego, donde se leen los archivos y se cargan y guardan los datos, y por el otro, el juego propiamente dicho.

Para la primera parte, se creó una estructura de datos llamada “DealOrNoDeal”, la cual, como es de esperarse, contiene toda la información necesaria para modelar una partida del juego.

struct DealOrNoDeal {

string nombre;

int maletines[CANTIDAD\_MALETINES];

int maletinesOrdenados[CANTIDAD\_MALETINES];

int posicionesOrdenado[CANTIDAD\_MALETINES]; ///En la posición i posicionesOrdenado, nos dice en qué posición de maletines está maletinesOrdenados[i]; DE FORMA TAL QUE MALETINESORDENADOS[I] == MALETINES[POSICIONES[I]]

int mascara[CANTIDAD\_MALETINES] = {0,0,0,0,0,0,0,0};

int maletinPrincipal;

bool esPrimeraVez = true; ///Primera vez que se elige un maletin.

int maletinesDisponibles; //cantidad de maletines restantes.

int premio;

} ;

Si bien a simple vista esta estructura puede parecer un poco confusa, uno termina de darse cuenta que no lo es. Dado que todos los objetos poseen nombres autodeclarativos, son pocos los que deben ser explicados. De todas formas, repasaremos la utilidad de los 4 arreglos utilizados en la estructura para asegurar el correcto entendimiento de ésta.

**int maletines[CANTIDAD\_MALETINES];**

Como el nombre lo indica, este arreglo contendrá los valores de los maletines. La información guardada en éste se obtiene del archivo *maletines\_oficiales.txt,* de forma tal que el contenido del maletín *i*, se encontrará en la posición *i-1* del arreglo. Esto se debe a que los maletines contienen números entre 1 y 9, y las posiciones del arreglo van entre 0 y 8.

**int maletinesOrdenados[CANTIDAD\_MALETINES];**

Si bien en el arreglo *maletines* ya se guarda la información de los maletines, es de gran utilidad también tener un segundo arreglo con los mismos valores, pero en forma ordenada. Esto se debe a que al momento de mostrarle al usuario los valores posibles dentro de los maletines, tiene que ser de forma ordenada, ya que si se mostraran en el orden que aparecen en el arreglo *maletines*, fácilmente uno podría asociar la posición donde se encuentra el valor con el número de maletín, y así asegurarse de siempre llevarse el premio mayor.

**int mascara[CANTIDAD\_MALETINES] = {0,0,0,0,0,0,0,0};**

Tal como lo indica su nombre, este arreglo es utilizado como máscara para el arreglo *maletines*. Esto quiere decir que cada posición del primer arreglo está asociada a la misma posición del segundo. Esto se utiliza para saber qué maletines fueron elegidos, de una forma muy sencilla. El arreglo se inicializa con todos ceros, dado que en un principio todos los maletines están disponibles. Luego, se elige un maletín, y se guarda la elección introduciendo un 2 en la posición correspondiente de la máscara. Una vez comenzado el juego, uno debe elegir varios maletines más para ser mostrados. Nuevamente, para almacenar esta información, se guarda un 1 en la máscara para cada maletín elegido.

De esta forma, la máscara es un arreglo que en cada posición nos indica si el maletín está disponible para ser elegido/mostrado, si es el maletín que elegimos para nosotros, o si ya fue mostrado.

**int posicionesOrdenado[CANTIDAD\_MALETINES];**

La frutillita del postre. Como se mencionó previamente, es de utilidad tener un segundo arreglo con los valores de los maletines ordenados. Pero es claro que uno quiere utilizar tanto *maletines* como *maletinesOrdenados* en conjunto, ya que si elige un maletín, se debe bloquear en indicar que fue elegido, pero en la lista visible para el usuario, es decir, la lista ordenada. Para lograr esto, uno podría, para cada maletín que se elige, realizar una búsqueda en el segundo arreglo para saber en qué posición se encuentra el valor obtenido, pero hacerlo de esta forma resultaría muy poco práctico y eficiente.

Para lograr trabajar los dos arreglos en simultáneo, se utiliza el tercer arreglo *posicionesOrdenado.* Lo que éste contiene en su interior no es nada más ni nada menos que la relación entre ellos, es decir, cómo se asocian las posiciones en los dos arreglos para un mismo valor. Por ejemplo, si en la primera posición de *maletinesOrdenados* tenemos $100, la primera posición de *posicionesOrdenado* nos indicará en qué posición de *maletines* se encuentra ese mismo valor, cumpliendo la siguiente ecuación:

*maletinesOrdenados*[i] == *maletines*[*posicionesOrdenado*[i]]

Es importante aclarar que la utilización de este arreglo no implica ningún costo extra, ya que se genera al mismo tiempo que se ordena el arreglo *maletinesOrdenados*, y permite recorrer ambos arreglos de maletines con un mismo índice.

Para asegurar la completa comprensión de estos arreglos, se implementó la función *ImprimirDeal*, la cual muestra por pantalla todos los valores de forma muy prolija. Esta función no tiene más propósito que ser utilizada por cualquier programador o lector del código para comprender la estructura *DealOrNoDeal*. No tiene ningún uso en el juego.

En cuanto a la lectura de archivos, se implementaron dos algoritmos muy similares para ambos, aprovechando el formato de éstos. Ambos contienen dos parámetros en cada línea (número de maletín y su valor o nombre y puntaje de cada jugador) separados por un espacio, se utilizó el operador “>>” para obtener los dos parámetros por separado y guardarlos en diferentes variables. La única diferencia se encuentra en que al número de maletín hay que quitarle la coma antes del espacio antes de convertirlo a entero.

Sobre la segunda parte, el juego propiamente dicho, no es mucho lo que hace falta aclarar, ya que no consiste en más que llevar a cabo el juego e ir guardando toda la información en el objeto correspondiente de la estructura DealOrNoDeal. De todas formas, sí es necesario describir la estrategia utilizada por la banca para realizar la oferta en cada caso.

Como ofrecer el promedio entre todos los valores que no han salido al momento resultaba ser una oferta demasiado generosa, originalmente se planteó multiplicar este valor por la fracción , para reducirla levemente. En un principio esta constante parecía ser apropiada, sobretodo en ofertas tempranas del juego, donde uno quiere incentivar a los jugadores a no retirarse de forma tan temprana, pero resultaba ser una oferta muy baja hacia el final, ya que cuando quedaban pocos maletines, al reducir el promedio de tal forma, en casos quedaba muy cerca del valor más bajo posible, haciéndola casi obsoleta. Por este motivo, se decidió que la constante que multiplica al promedio vaya variando según la cantidad de maletines restantes, de forma que vaya aumentando y pareciéndose más 1 con el paso de los turnos. Esto tiene mucho sentido ya que, en caso de haber solo dos maletines restantes, resulta coherente que si restan dos montos posibles, uno pueda arriesgarse a ir por el premio mayor, o jugar seguro y retirarse con un valor intermedio, es decir, el promedio o un valor cercano a éste.

Con este fin, si llamamos *k* a la constante mencionada, su valor está dado por:

Notar que a medida que la cantidad de maletines disponibles disminuye, la fracción se acerca más a 1.