

Informe técnico

Programación Bajo Plataformas Abiertas

Proyecto Final: Programación de juego de
“Poker”

Profesor:
Juan Carlos Coto

Integrantes:
Fabian Sander Hangen C07336
Gabriel Gamboa Vargas B73098
Pablo Cordero Bermúdez C02337
Grupo 03

II-2022

Índice:

Introducción	3
Diseño general	5
Principales retos	11
Conclusiones	12
Bibliografía	13

Introducción

El proyecto que se va a presentar en este documento es una descripción narrativa del proceso de construcción de un juego de "Poker" en el lenguaje de programación de "C".

Cuando se le otorgó el juego de Poker a los estudiantes en cuestión, ninguno sabía la manera en la que se jugaba este juego. En un principio el estudiantado se puso a averiguar la dinámica de juego de esta actividad. Lo primero que se nota al investigar más sobre esta actividad de recreación, es que este, tiene que ver con la suerte y las posibilidades que tiene cada jugador de ganar o de que le salga una carta en específico, por eso es que el Poker se puede encontrar en muchos casinos o casas de apuestas a lo largo del mundo. Lo segundo que se encontró es que hay muchas maneras y versiones de jugar este juego. Según la revista en internet "NOROESTE", las versiones del poker más conocidas nivel mundial son: "Texas Hold'em", "5 Card Draw", "5/7 Card Stud", "Poker 224" y "Omaha Hold'em" ([Externo, 2022](#)). Según lo leído, todos los tipos de Poker tienen una base similar a la hora de ser jugados, en todas las formas de Poker se tiene que apostar y reconocer de manera visual y auditiva la manera en la que juega su oponente. Esto mismo hace que este juego sea tan estresante para algunos, ya que psicológicamente es muy agotador estar pretendiendo y tratando de engañar al oponente. Por otro lado, también tiene mucha relación con las matemáticas y la probabilidad que le salga una carta en específico que el jugador necesita o no quiere que le salga en su juego.

Lo que resta es explicar la manera en la que se escogió el tipo de Poker el cual se programó en el lenguaje de C para este proyecto. En un principio se consideró la opción de programar el mundialmente conocido como "5 Card Stud". Este es un tipo de Poker en el cual se tiene un repartidor (el cual claramente es el programa) y los demás jugadores, en este juego se puede jugar de dos a 5 jugadores por mesa y se requiere que se le repartan, al inicio, 2 cartas por jugador, una tapada y una destapada para que toda la mesa pueda ver esta carta. En este tiempo de juego no hay un montón de cartas comunitarias, por eso es que no se simplifica la programación. El juego consiste en que el repartidor tiene que ir repartiendo cada ronda, una carta, a cada jugador, cada ronda se tiene que apostar (como en casi todos los juegos de Poker). Lo emocionante de este juego es que

cada jugador de la mesa conoce las cartas de cada jugador excepto de una, entonces no saben si tienen el potencial para construir una mano fuerte que pueda vencer a su mano. En este caso, la “Figura 1” vuelve a ser protagonista, ya que cada jugador tiene que apostar según la “jerarquía de manos”, ya que un par no es tan poderoso como dos pares o cuatro del mismo palo o una escalera.




 Royal Flush	10♥ J♥ Q♥ K♥ A♥
 Straight Flush	4♣ 5♣ 6♣ 7♣ 8♣
 Four of a Kind	K♠ K♥ K♣ K♦ 3♠
 Full House	10♥ 10♠ 10♦ A♠ A♣
 Flush	10♠ K♠ 2♠ 6♠ 7♠
 Straight	7♣ 8♠ 9♦ 10♠ J♥
 Three of a Kind	5♠ 5♥ 5♣ J♦ A♦
 Two Pair	A♠ A♥ 3♣ 3♠ J♣
 One Pair	Q♦ Q♥ 2♥ 8♠ 9♣

Figura 1. Jerarquía de cartas en el Poker (A. 2014).

La “Figura 1”, presenta la jerarquía de cartas que se tiene que seguir para ganar el juego de Poker. Esta escalera va de arriba para abajo, esto significa que el “Royal Flush” le gana a todas las combinaciones posibles, el “Straight Flush” le gana a todas los de abajo pero pierde contra un “Royal Flush”, y así en adelante.

“Texas Hold’em” consiste en que el repartidor tiene que repartir solamente dos cartas a cada jugador, estas cartas no son conocidas por el resto de los jugadores de la mesa. Cada ronda, este repartidor tiene que ir agregando una carta al montón de cartas que hay en el centro de la mesa (inicialmente, el repartidor ya había repartido tres cartas en el centro de la mesa). En este caso se tiene que ir apostando cada ronda, y cada ronda el repartidor tiene que ir agregando una carta más a la mesa, hasta que hayan 5 cartas comunitarias en la mesa. El objetivo de cada jugador es perder las menos fichas posibles o ganar las mayores fichas posibles, según la combinación de cartas que haya recibido cada jugador.

En esta caso, este juego es más sencillo de programar, ya que se tiene que comparar con el montón del centro, y no se tiene que comparar cada mano con la

de los demás, por eso es que se escogió este tipo de juego para el proyecto de programación, el cual se va a describir en el siguiente apartado.

Diseño general

A la hora de empezar a escribir el código del juego, se le pide al estudiante que realice un menú, el cual tiene que recopilar las diferentes opciones. La primera opción, la cual es la más importante, es la opción de “jugar”. En este caso, esta opción es la que inicia el juego y la que conlleva la mayor parte del trabajo a la hora de programar un juego como es “Texas Hold’em”. En este caso, la segunda opción en el menú es la de las “reglas”, en las cuales, para hacer el trabajo más fácil para el programador y para el jugador, se le puso un link a una página web, la cual contiene todas las diferentes reglas y especificaciones. En este caso, se le puso como opciones al jugador que escogiera entre dos diferentes links, los dos links de los que puede escoger el jugador son: “<https://bicyclecards.com/how-to-play/basics-of-poker>”. La tercera opción es la de revisar cuáles son los puntajes más altos que se hayan jugado en este juego. Como se está jugando Poker, para el caso de los “puntajes más altos”, se escogió poner las mejores ganancias que les hayan tocado a los jugadores en las partidas que jugaron.

```
printf("\n\n¡Bienvenido al poker! \n ¿Que desea hacer?\n\n");  
// opciones  
printf("\n(1)   ¡Jugar!");  
printf("\n(2)   Reglas");  
printf("\n(3)   Mejores manos ");  
printf("\n(4)   Salir\n");
```

Figura 2. Menú del programa.

En la “Figura 2” se puede observar todo lo que se acaba de comentar, lo cual es, las diferentes opciones del menú de entrada al juego de Poker que se programó. Seguidamente se indicará el funcionamiento del juego en sí, se proporcionarán fotos de la estructura del programa y de cómo se lograron los objetivos cumplidos.

Seguidamente, después de que el usuario pasa el filtro del menú y empieza a jugar el juego, se le tienen que pedir al usuario diferentes datos del panorama del juego que son importantes para que el programa sepa lo que tiene que hacer.

Primero, se le enseña al usuario un mensaje de bienvenida, el cual se realizó con muchas funciones de “print”.

```
printf(".....\n");
printf("|B.--. ||I.--. ||E.--. ||N.--. ||V.--. ||E.--. ||N.--. ||I.--. ||D.--. ||O.--. |\n");
printf("| :(): || (\V) || (\V) || :(): || :(): || (\V) || :(): || (\V) || :/\V: || :/\V: |\n");
printf("| ()() || :VV: || :VV: || ()() || ()() || :VV: || ()() || :VV: || ( ) || :VV: |\n");
printf("| '---'B|| '---'I|| '---'E|| '---'N|| '---'V|| '---'E|| '---'N|| '---'I|| '---'D|| '---'O|\n");
printf(".....\n");
```

Figura 3. Mensaje de Bienvenida.

En segundo, se le pide al usuario la cantidad de dinero con la cual los jugadores van a estar jugando en la partida. En esta función, se hace lo posible para que el entero que se ingrese en el juego sea positivo. Este dato es importante, para que el programa sepa, después en las apuestas, con cuanto dinero se puede apostar y que ningún jugador trate de pasarse del límite de apuestas con dinero que no tiene. También, en un casino real, funciona para que el repartidor sepa cuándo una persona ya apostó todo y está “all in”.

```
int pedir_dinero_inicial () {
    printf("Ingrese el dinero inicial para cada jugador \n");
    int dinero_inicial = pedir_input();

    while (dinero_inicial <= 0) {
        printf("El dinero inicial debe ser positivo\n");
        dinero_inicial = pedir_input();
    }

    return dinero_inicial;
}
```

Figura 4. Función de dinero inicial.

De segundo se puede ver que hay una función que le pide al usuario la cantidad de jugadores, esta tiene que ser mayor o igual a dos y menor o igual a tres, esto, para que las cartas en un mazo, no se acaben en media partida y para que el programa sepa cuántos turnos tiene que tener cada ronda. Esta función también es

muy importante ya que en un mazo de naipes, solo hay 52 cartas disponibles, por lo cual, si se reparten muchas cartas, el juego de probabilidad de este modo, no serviría ya que los del juego, podrían saber qué carta va a aparecer en la siguiente ronda.

```
int pedir_cantidad_de_jugadores () {  
    printf("Ingrese la cantidad de jugadores \n");  
    int cantidad_de_jugadores = pedir_input();  
  
    while (cantidad_de_jugadores < 2 || cantidad_de_jugadores > 5) {  
        printf("La cantidad de jugadores debe ser entre 2 y cinco \n");  
        cantidad_de_jugadores = pedir_input();  
    }  
  
    return cantidad_de_jugadores;  
}
```

Figura 5. Función para pedir cantidad de jugadores

El siguiente dato que se le pide al usuario es el nombre de los jugadores. Esta función no es tan importante, ya que para jugar Poker, no se necesitan los nombres de los participantes, y en ningún casino se le pide a los jugadores que den su nombre, todo es muy anónimo. En este caso, se le piden los nombres a los jugadores, para que el programa fluya con fluidez y para que sea más personal para el observador y el usuario. Esta función requiere que se le dé como parámetro la cantidad de jugadores. Esta función al final devuelve una lista con los nombres de los jugadores que se formó con tres bucles de “for” y “while”. En este caso no se enseña toda la función ya que es muy extensa.

```
jugador* pedir_nombres (int cantidad_de_jugadores) {  
    jugador* jugadores = malloc(cantidad_de_jugadores*sizeof(jugador));  
  
    return jugadores;
```

Figura 5. Función para pedir nombre de los jugadores.

De ahora en adelante, empieza el juego a correr por sí mismo en lo cual se necesita una función grande que tenga todas las funciones. En el juego de poker,

todos los juegos tienen la misma estructura, por eso es que una función principal siempre va a tener las funciones de: crear mazo, repartir cartas, de apostar y al final de comparar las cartas para ver quién tiene el mejor juego.

En este caso la función de crear el mazo ya se había hecho en la clase como un laboratorio, esta función no recibe ningún parámetro ya que todos los mazos son iguales en el poker. Al final hacer “return” de un arreglo llamado “nuevo_mazo”.

```
mazo* crear_mazo () {  
  
    int j = 0;  
    mazo *nuevo_mazo = malloc(sizeof(mazo));  
    int k = 0;  
  
    for (j = 0; j<4; j++) {  
        for (int i = 1; i<=13; i++) {  
            carta nueva_carta;  
  
            nueva_carta.tipo_carta = j;  
            nueva_carta.valor_carta = i;  
            nueva_carta.eliminado = 1;  
  
            (*(nuevo_mazo)).baraja[k] = nueva_carta;  
            k++;  
        }  
    }  
  
    (*(nuevo_mazo)).restantes = 52;  
    return nuevo_mazo;  
}
```

Figura 6. función para crear mazo.

Una de las funciones más importantes es la de repartir cartas. Esta función recibe como parámetros la cantidad de jugadores en la mesa y el mazo nuevo que se acaba de crear con la función anterior. Al final se tiene que actualizar el mazo ya que se tienen que restar las cartas que se hayan repartido. En este caso se repartieron las caras de forma aleatoria para que no siempre se repartan las cartas de forma igual en todos los juegos.


```

node* repartir(mazo* baraja, int cantidad, node* mano) {

    if (cantidad <= (*baraja).restantes) {
        time_t t;
        srand((unsigned) time(&t));

        for (int i = 0 ; i < cantidad ; i++) {

            carta nueva_carta;
            int random = rand() % 52;

            if ( (*baraja).baraja[random].eliminado == 0) {
                i--;
                continue;
            }

            nueva_carta = (*baraja).baraja[random];
            mano = insert(nueva_carta, mano);

            (*baraja).baraja[random].eliminado = 0;
            (*baraja).restantes --;
        }
    }

    return mano;
}

```

Figura 7. Función de repartir cartas.

La siguiente función es la de apostar el dinero que cada jugador tenga. En cada caso, el jugador que le toca, va a tener que apostar la cantidad de dinero que tenga disponible. Esta función tiene como parámetros, la cantidad de jugadores, el nombre de cada uno, la cantidad de dinero con la que se decidió jugar. En este caso la función de apostar tiene que ir sumando la cantidad de dinero que hay en el “canasto público”, que todos ya apostaron, para que el que gane la ronda pueda ganarse esa cantidad. Solo se va a enseñar el principio de esta función ya que es extremadamente larga.

```

void apuesta (juego* juego, jugador* jugador, pot_node* bote) {

    printf("\n\n\n\n");
    printf("Es el turno de %s su dinero actual es: %d\n", jugador -> nombre, jugador -> dinero);

    printf("Las cartas sobre la mesa son: \n\n");
    imprimir_mano(juego->mesa);

    printf("Su mano es: \n");
    imprimir_mano(jugador -> mano);
}

```

Figura 8. Pedazo de la función de apostar.

Se puede ver que en cada paso, se le imprime la información del jugador para que vaya viendo qué tiene en su mano y lo que los demás tienen. Además se le enseña la cantidad de dinero que aún le queda y que no ha apostado.

La última función de la que se va a hablar sirve como ejemplo en el proceso de comparación de las diferentes manos que se pueden tener en el Poker. En este caso se tiene la función de comparación de “straight flush”, la cual comprueba si se tiene un juego de “straight flush” en la mano de los jugadores. En este caso, saber lo que se tiene en la mano es muy importante, ya que, para saber el ganador de cada ronda, se tiene que saber quién tiene el mejor juego de todos los jugadores en la mesa.

Por último, se tiene una función muy importante para la recopilación del juego. En este caso, se le pidió al estudiante que guardara las mejores puntuaciones de todos los juegos que se habían jugado. En este caso, se creó una función que abre un documento de texto en el que se escriben las mejores puntuaciones posibles. En este caso se quiere que la mejor mano que se haya sacado sea guardada. Esto se consiguió por medio de el documento “puntuaciones_numero.txt”, la cual, al final de cada juego se tiene que estar abriendo y cerrando para que se pueda escribir en ella. Esta función recibe como parámetros el puntero al nombre de la persona (usuario) que se ganó la mano más alta, y también recibe el número entero del dinero que ganó la persona.

```
void puntuacion (char * nombre, int dinero_ganado) {  
    FILE* file = fopen("puntuaciones_numero.txt", "r+");
```

Figura 9. Función de mejor puntuación.

Principales retos

Al principio de la escritura del código hubo diferentes retos, los cuales se tuvieron que enfrentar. El primero de esos retos fue la organización del equipo. Este reto se pudo pasar gracias a la buena comunicación de los integrantes del grupo. Se discutió cuáles eran las diferentes debilidades de los tres diferentes estudiantes y cuáles eran las diferentes fortalezas. En algunos casos, se tuvo que repartir el trabajo de manera diferente, en los cual alguno del equipo tenía que escribir en el código más que otro y en algunos casos, uno tenía que escribir en el reporte más que otro. Al final, con la ayuda de la buena comunicación se pudo resolver y todos los integrantes del grupo trabajaron de manera equitativa.

Otro problema que se enfrentó fue a la hora de escoger el tipo de Poker que se iba a programar. En este caso, todos los integrantes del grupo se reunieron a informarse sobre los diferentes tipos de Poker que había. Entonces, se escogió el “Texas Hold'em”, ya que era el tipo de Poker que más se adapta a las capacidades de los estudiantes y temario del curso en sí.

Otro reto que tiene más relación con el código en sí fue el reto de usar datos que estuvieran en el heap, usando listas enlazadas para las manos de los jugadores y en las cartas de la mesa. En este caso, este tema no se profundizó con mucha intensidad en el curso, pero al final, con ayuda de la búsqueda en internet y algunos videos, se pudo alcanzar el resultado esperado.

Un reto, el cual se fue muy difícil de superar fue el de poder procesar “strings” en un lenguaje de programación diferente a “Python”. En un curso anterior, se sabía cómo recorrer un string y también poder combinarlos con diferentes valores de enteros en una carta de un mazo. En este proceso se pudo aprender sobre este tema y también se pudo agrandar el conocimiento que se tenía antes de empezar con el proyecto.

El último reto por el cual se tuvo que pasar fue el de comparar todos los resultados de todos los jugadores, para saber cuál estaba más arriba en la jerarquía de cartas de juego. En este caso se tuvo que comparar 4 manos diferentes en base a unas reglas de juego que no tenían nada que ver con números y su orden. En este caso, la comparación de las manos de poker en una ronda es lo más importante de un juego, esto, para saber quién es el que se tiene que llevar el dinero de todos en la ronda final.

Conclusiones

En conclusión, se puede decir que se aprendió sobre el trabajo en equipo y la preparación que se tiene que llevar a cabo antes de empezar un proyecto como es el de programar un juego en el lenguaje de programación de C. En este recorrido se aprendió a comunicarse de manera ejecutiva y clara con los compañeros del grupo y también se aprendió sobre la importancia de un “plan de juego”, ya que en un principio no se puede comenzar a programar y a investigar de primer plano. En un proceso bien planteado, se tiene que discutir con el equipo cuáles son las metas, cuál es el trabajo de cada integrante del equipo y hay que tener claro cuáles son las fortalezas y debilidades de cada compañero. Esto es algo que se aprendió en el transcurso de la programación de este juego y que se tiene que implementar a lo largo de la vida personal y laboral de cada persona. También se puede decir que se afilaron las destrezas de programación y de búsqueda en Internet, ya que en la vida de un programador, esta es una de las herramientas más importantes que se tienen. Hay que saber cómo y dónde buscar información sobre el código y funciones que puedan ser útiles.

Bibliografía

A. (2014, May 16). *Los puntajes del texas hold'em poker: puntajes de las manos más altas*. Poker. Retrieved November 26, 2022, from

<https://es.poker-online-gratis.net/las-jugadas-en-el-texas-holdem-poker/>

C Files I/O: Opening, Reading, Writing and Closing a file. (s. f.).

<https://www.programiz.com/c-programming/c-file-input-output>

EOF, getc() and feof() in C. (s. f.).

<https://www.tutorialspoint.com/eof-getc-and-feof-in-c>

Externo, C. (2022, November 16). *Diferentes versiones del póker*.

www.noroeste.com.mx.

<https://www.noroeste.com.mx/internacional/diferentes-versiones-del-poker-HI2965823>

GeeksforGeeks. (2017, 2 junio). *fseek() in C/C++ with example*.

<https://www.geeksforgeeks.org/fseek-in-c-with-example/>

GeeksforGeeks. (2022, 12 abril). *snprintf() in C library*.

<https://www.geeksforgeeks.org/snprintf-c-library/>

Strings in C (With Examples). (s. f.).

<https://www.programiz.com/c-programming/c-strings>