



Universidad de Guanajuato

División de Ingenierías Campus Irapuato-Salamanca

Ingeniería de Software

PROYECTO FINAL

Juego Poker Clásico en C

Serrano Lule Javier

Salamanca, Guanajuato

08 de Diciembre del 2016

Descripción del proyecto

El respectivo trabajo que se presenta para la materia de Ingeniería de Software tiene como objetivo demostrar todos los conocimientos obtenidos, demostrar el uso de las técnicas vistas y generar un código de calidad para que el producto tenga la menor cantidad de defectos al momento de terminarse.

Como tal, se pretende describir en el presente reporte, el proyecto de un juego sencillo de Poker clásico realizado en lenguaje C y compilado en Codeblocks.

Requerimientos del proyecto

El proyecto tiene como objetivos principales los siguientes puntos:

- Poder jugar de manera igual con una persona humana el juego de póker, usando las mismas reglas para ambos.
- Presentar un reto para el jugador, creando una simulación de inteligencia tomando métodos de heurísticas y predicciones para la realización de jugadas.
- Dar incentivo al jugador de que puede dejar su registro al hacer muy buenas jugadas.
- Probar todas y cada una de las funciones para ejercitar todo camino posible.
- Documentar totalmente la realización con el apoyo de un servicio de repositorio.
- Crear un proyecto (o dividirlos de manera distinta en su caso) por cada librería a probar, tanto pruebas unitarias, como las pruebas de integración del proyecto.

Además, se pretende que las librerías sean completamente funcionales para proyectos futuros o modificaciones posibles.

Desarrollo

El desarrollo se hizo de la siguiente manera:

1. Se tomó la idea y se investigó acerca de lo complejo que sería la realización de un proyecto de tal magnitud. Al verificar si los tiempos eran adecuados se hizo la propuesta al profesor acerca de la decisión de realizar el juego de Póker.
2. Se creó un repositorio en mi cuenta personal el cuál fue llamado "Poker-C" (<https://github.com/JavierSLX/Poker-C>) donde se documentaría y se haría un seguimiento de cada acción a realizarse en el proyecto, controlar cambios además de tener un control con las distintas versiones del proyecto.
3. Se tomó la decisión de que el proyecto se realizaría con el IDE Codeblocks debido a la comodidad de su editor y compilador, se trató de que la versión de tal fuera independiente y que con cualquiera de ellas pudiera ejecutarse sin problemas.
4. Se documentó cada función que se realizaba usando el formato Markdown (*.md), donde se describen pruebas unitarias, de integración y de humo, se realizó su respectivos grafos en cada una de ellas y se registró el resultado de cada prueba (tales resultados y procesos se pueden encontrar en la carpeta "Documentación" anexada al proyecto).
5. Se hicieron 3 reuniones formales e informales respectivamente, con fechas estratégicas para poder identificar cuál sería la mejor forma de seguir el proyecto y detectar posibles errores y carencias.

6. Se realizaron dos pruebas Alpha, las cuales fue posible detectar un error crucial en el código.

Las actividades anteriores describen de manera general la realización del proyecto, los detalles de tales pueden observarse en la carpeta donde se documentan o como tal, a lo largo del presente reporte.

Cronograma de actividades (Tabla).

Actividad	Fecha de realización
Creación del repositorio	27/Octubre/2016
Creación del proyecto principal “Baraja” y funciones principales.	27/Octubre/2016
Creación de la función: imprimirElementoCarta , pruebas y documentación.	02/Noviembre/2016
Creación de la función : imprimirCaractCarta , pruebas y documentación.	04/Noviembre/2016
Creación de la funciones imprimirCaractBaraja , definirCarta , liberarMemoria y crearBaraja pruebas y documentación.	05/Noviembre/2016
Creación de la función : barajear , pruebas y documentación.	08/Noviembre/2016
Creación de la función : quitarJokers y repartirMano , pruebas y documentación.	09/Noviembre/2016
Creación de la función: imprimirMano , pruebas y documentación.	10/Noviembre/2016
1er Reunión Informal.	11/Noviembre/2016
Creación de las funciones: asignarValor , ordenarCartas y prioridadTipo	12/Noviembre/2016
1er Reunión Formal.	14/Noviembre/2016
Creación de las funciones: borde , datos , apostar , sacarCarta y cambios	16/Noviembre/2016
2da Reunión Informal.	18/Noviembre/2016
Creación de la función : probarEscaleraColor , pruebas y documentación.	21/Noviembre/2016
Creación de las funciones: comprobarEscaleraColor , inicioDescarte , probarEscaleraReal , y comprobarEscaleraReal .	22/Noviembre/2016
2da. Reunión Formal.	23/Noviembre/2016
Reacomodo de los drivers baraja y póker.	24/Noviembre/2016
Creación de las funciones del proyecto de jugadas, apuestas y proyecto.	26/Noviembre/2016
Pruebas y documentación: sacarCarta , cambios .	27/Noviembre/2016

Pruebas y documentación: datos.	28/Noviembre/2016
3ra. Reunión Informal.	29/Noviembre/2016
Pruebas y documentación: apostar.	01/Diciembre/2016
Pruebas y documentación: Funciones de manejo de jugadas.	03/Diciembre/2016
Pruebas y documentación: Funciones de manejo de mano y comparación de jugadas.	04/Diciembre/2016
3ra. Reunión Formal	05/Diciembre/2016
Pruebas finales	07/Diciembre/2016
Pruebas Alpha	07/Diciembre/2016

Reportes de Revisiones técnicas e informales

Reuniones Técnicas Informales

1era Reunión

FECHA: 11/Noviembre/2016

LUGAR: Salvatierra, Guanajuato.

NOMBRES: Samantha García Ortiz, Javier Serrano Lule

DURACIÓN: 20 minutos

OBSERVACIONES:

Tras dar las ideas del proyecto y acerca de lo que trataba, la srita comenzó a hacerme las siguientes preguntas: *¿funciona lo básico? ¿qué porcentaje del proyecto va hecho?* Tras el análisis del total del proyecto realizado y mostrar las pruebas ya hechas se llegó a las siguientes conclusiones:

1. Las pruebas estaban bien hechas pero había que encontrar la forma de acelerarlas.
2. El proyecto iba atrasado aparentemente.
3. Aún faltaba buscar la forma de reemplazar correctamente los símbolos de las cartas.

Una sugerencia más que se dió, fue la de quitar en algunas funciones un parámetro que se vio que es innecesario, por lo que se anotó quitarlo en las pruebas de tal.

2da Reunión

FECHA: 18/Noviembre/2016

LUGAR: Salvatierra, Guanajuato.

NOMBRES: Samantha García Ortiz, Javier Serrano Lule

DURACIÓN: 15 minutos

OBSERVACIONES:

Se observó que una de las librerías ya estaba completamente terminada, aún así se sugirió que se hiciera un cambio en cuanto a la presentación de las cartas, además de que las pruebas iban muy lentas a pesar del tiempo. También se dio la idea de comparación para las manos en las funciones que se construirían a continuación.

3ra Reunión

FECHA: 29/Noviembre/2016

LUGAR: Salamanca, Guanajuato.

NOMBRES: Adalberto Martínez Blancarte, Javier Serrano Lule

DURACIÓN: 30 minutos

OBSERVACIONES:

Las pruebas estaban a punto de finalizarse. Se sugirió que algunas funciones afectaran variables principales para su comodidad, debido a la integración de algunas funciones y el funcionamiento de ellas. También se sugirió una heurística apropiada para el manejo de manos de la computadora y que lo más adecuado sería simplemente tener tres oponentes en el juego para que se ejecutara de manera eficiente.

Reuniones Técnicas Formales

1era Reunión

FECHA: 14/Noviembre/2016

LUGAR: Salvatierra, Guanajuato.

NOMBRES: Samantha García Ortiz, Héctor Zavala, Javier Serrano Lule

DURACIÓN: 60 minutos

OBSERVACIONES:

Antecedentes: La reunión comenzó dando un resumen de lo que era el proyecto. Tras hablar y describir lo que en sí terminaría siendo, y mostrar los avances.

¿Qué fue lo que se revisó?: Las funciones iniciales de la librería Baraja, así como las ideas establecidas de como se realizaría la otra parte del proyecto.

¿Cuáles fueron los descubrimientos y las conclusiones? Se encontró un error en el código, el cual liberaba memoria antes de hacer un cambio lo que ocasionaba que la baraja no se mostrara de manera correcta y ocasiona un error de lógica. Se sugirió probar la memoria antes de seguir realizando el proyecto.

2da Reunión

FECHA: 23/Noviembre/2016

LUGAR: Salvatierra, Guanajuato.

NOMBRES: Samantha García Ortiz, Héctor Zavala, Javier Serrano Lule

DURACIÓN: 75 minutos

OBSERVACIONES:

Antecedentes: Se mostraron los avances. Se describió como se terminó la librería baraja y el avanza de la librería póker. Se dieron los avances de las funciones que eran capaces de manejar las manos de los jugadores.

¿Qué fue lo que se revisó?: Absolutamente todas las funciones que fueron hechas en la librería baraja, y las funciones iniciales de la librería poker.

¿Cuáles fueron los descubrimientos y las conclusiones? Se verificaron las pruebas, se observó que el proyecto iba con avances lentos además de que la forma en la que se estaba programando estaba siendo poco productiva. Se dieron algunos consejos, como la forma en la que se debería de analizar la mano y las jugadas y como se deberían de mostrar los resultados.

3ra Reunión

FECHA: 05/Diciembre/2016

LUGAR: Salvatierra, Guanajuato.

NOMBRES: Samantha García Ortiz, Héctor Zavala, Javier Serrano Lule

DURACIÓN: 70 minutos

OBSERVACIONES:

Antecedentes: Se mostró toda la documentación, el proyecto ya terminado y las funciones con las que ya se contaba. Se explicó la manera de jugar y la forma en la que se apostaba y el registro de los nombres.

¿Qué fue lo que se revisó?: Todo el proyecto, las pruebas unitarias que se realizaron y las pruebas de integridad.

¿Cuáles fueron los descubrimientos y las conclusiones? Al probar el proyecto, saltó de inmediato la idea de cambiar una función para dar más eficiencia al programa. Se sugirió que la función ChecarMano fuera cambiada y obtuviera algunos datos extras para poder trabajar más sencillo con ellos. Además se sugirió que se hicieran las pruebas Alpha con alguien que no supiera absolutamente nada del proyecto.

Bugs corregidos

Los siguientes bugs fueron corregidos gracias a las distintas pruebas (unitarias, integración) a las reuniones y a las pruebas Alpha:

- El desborde de la asignación de cartas en una baraja.
- El análisis de un elemento más en la repartición de manos, lo que ocasionaba un error lógico en el programa.
- Ajustes de apuestas incorrectos.
- Cambio de cartas similares que no deberían hacerse.
- El recorrido de la variable carry que se comportaba de manera extraña.
- Asignaciones de referencia que no se hacían en la función apostar.
- Mensajes extraños que mostraba la función de marcadores.
- La función escalera no detectaba cuando se encontraba tal jugada.
- La función escalera real no funcionaba cuando detectaba una jugada parecida.
- Se tomaba los valores de los comodines de manera incorrecta en la función póker.
- Extraño comportamiento cuando todos los jugadores cambiaban más de 4 cartas en el juego.
- No detectaba en ocasiones quien poseía la mano más alta al finalizar el juego.

Documentación

Toda la documentación de las pruebas y el desarrollo, por lo extenso que es, se encuentran en una carpeta del mismo nombre, por lo que para consultar resultados de las pruebas, grafos, etc. Se recomienda dar un vistazo a tal carpeta.

Cómo se usa el software

El funcionamiento del programa se puede describir de la siguiente forma:

Al correr el programa, se inicia con un menú, el cual da 3 opciones. La primera (1) es el juego de póker clásico el cual permite jugar contra otros 3 oponentes, la segunda opción (2) muestra los marcadores más altos del juego y la última permite salir del programa.

```
"C:\Respaldo\FIMEE\Plan Semestral\Semestre 2\Ingenieria de Software\Baraja\bin\Debug\Baraja.exe"

***** JUEGO POKER CLASICO *****
Opciones:
1.Juego
2.Marcadores
3.Salir

Ingenieria de Software
By: Serrano Lule Javier

Num: █
```

Al comenzar con el juego, se reparten las 5 cartas de manera oculta y se comienza con la primera fase de apuestas. Los jugadores controlados por la computadora abre con una cantidad donde el jugador puede aumentar o igualar la apuesta. Es importante observar que la cantidad que se debe de ingresar debe ser igual o superior a la que se muestra, en caso de no contar con tal, el sistema sacará del juego a la persona e inmediatamente le comentará que ha perdido el juego. Si el jugador ingresa 0, el programa terminará debido a que se decidió abandonar la partida.

```
"C:\Respaldo\FIMEE\Plan Semestral\Semestre 2\Ingenieria de Software\Baraja\bin\Debug\Baraja.exe"

--> JUGADOR <--
Carta 1      Carta 2      Carta 3      Carta 4      Carta 5
* * ****    * * ****    * * ****    * * ****    * * ****

Fondo: $1000
Apuesta: $0

--> PC1 <--
Carta 1      Carta 2      Carta 3      Carta 4      Carta 5
* * ****    * * ****    * * ****    * * ****    * * ****

Fondo: $****
Apuesta: $0

--> PC2 <--
Carta 1      Carta 2      Carta 3      Carta 4      Carta 5
* * ****    * * ****    * * ****    * * ****    * * ****

Fondo: $****
Apuesta: $0

--> PC3 <--
Carta 1      Carta 2      Carta 3      Carta 4      Carta 5
* * ****    * * ****    * * ****    * * ****    * * ****

Fondo: $****
Apuesta: $0

### SE ABREN LAS APUESTAS ###


El jugador 2 inicia la apuesta con $10

El jugador 3 iguala la apuesta de $10

El jugador 4 iguala la apuesta de $10

Quieres igualar o aumentar la apuesta? (0 = retirarse):
```

En la siguiente fase, la mano del jugador es mostrada y se le pregunta que cartas quiere cambiar. Es importante hacer notar que primero se registra la cantidad de cartas que se quiere cambiar, e inmediatamente después el número de cartas que se cambian. El mismo sistema irá indicando que carta sale y cual entra a la mano. El sistema no permite el cambio de cartas que acaban de salir para evitar trampa o confusión del jugador. El sistema al finalizar los cambios del jugador indica cuantas cartas cambiaron los otros jugadores.

 "C:\Respaldo\FIMEE\Plan Semestral\Semestre 2\Ingenieria de Software\Baraja\bin\Debug\Baraja.exe"

```
--> PC1 <--
Carta 1      Carta 2      Carta 3      Carta 4      Carta 5
* * ****    * * ****    * * ****    * * ****    * * ****

Fondo: $****
Apuesta: $50

--> PC2 <--
Carta 1      Carta 2      Carta 3      Carta 4      Carta 5
* * ****    * * ****    * * ****    * * ****    * * ****

Fondo: $****
Apuesta: $50

--> PC3 <--
Carta 1      Carta 2      Carta 3      Carta 4      Carta 5
* * ****    * * ****    * * ****    * * ****    * * ****

Fondo: $****
Apuesta: $50

TOTAL = $200
```

PRIMERA RONDA DE CAMBIOS

```
Cuántas cartas quiere cambiar: 3
Que carta quiere cambiar?(1-5): 2

Cambio carta (2): 2 C rojo -----> 10 T negro

Que carta quiere cambiar?(1-5): 4

Cambio carta (4): 10 C rojo -----> 8 C rojo

Que carta quiere cambiar?(1-5): 5

Cambio carta (5): K R rojo -----> 3 T negro

El jugador 2 cambia 3 carta(s)
El jugador 3 cambia 3 carta(s)
El jugador 4 cambia 4 carta(s)
```

Se abre la segunda ronda de apuestas, donde hay que igualar o superar la apuesta de la computadora. E igual que la primera ronda, si no se cuenta con la cantidad suficiente o en caso de que se ingrese 0, el juego termina.

"C:\Respaldo\FIMEE\Plan Semestral\Semestre 2\Ingenieria de Software\Baraja\bin\Debug\Baraja.exe"

--> JUGADOR <--

Carta 1	Carta 2	Carta 3	Carta 4	Carta 5
3 E negro	3 T negro	10 T negro	3 C rojo	8 C rojo

Fondo: \$950

Apuesta: \$50

--> PC1 <--

Carta 1	Carta 2	Carta 3	Carta 4	Carta 5
* * ****	* * ****	* * ****	* * ****	* * ****

Fondo: \$****

Apuesta: \$50

--> PC2 <--

Carta 1	Carta 2	Carta 3	Carta 4	Carta 5
* * ****	* * ****	* * ****	* * ****	* * ****

Fondo: \$****

Apuesta: \$50

--> PC3 <--

Carta 1	Carta 2	Carta 3	Carta 4	Carta 5
* * ****	* * ****	* * ****	* * ****	* * ****

Fondo: \$****

Apuesta: \$50

TOTAL = \$200

SEGUNDA RONDA DE APUESTAS

El jugador 2 aumenta la apuesta en \$39

El jugador 3 iguala la apuesta de \$39

El jugador 4 iguala la apuesta de \$39

Quieres igualar o aumentar la apuesta? (0 = retirarse):

Se abre la última ronda de cambios, y de igual forma, al finalizar los cambios del jugador, se muestra cuantos cambios hicieron los jugadores controlados por la computadora.

"C:\Respaldo\FIMEE\Plan Semestral\Semestre 2\Ingenieria de Software\Baraja\bin\Debug\Baraja.exe"

```
--> JUGADOR <--
Carta 1      Carta 2      Carta 3      Carta 4      Carta 5
3 E negro    3 T negro    10 T negro   3 C rojo     8 C rojo

Fondo: $900
Apuesta: $100

--> PC1 <--
Carta 1      Carta 2      Carta 3      Carta 4      Carta 5
* * ****    * * ****    * * ****    * * ****    * * ****

Fondo: $****
Apuesta: $100

--> PC2 <--
Carta 1      Carta 2      Carta 3      Carta 4      Carta 5
* * ****    * * ****    * * ****    * * ****    * * ****

Fondo: $****
Apuesta: $100

--> PC3 <--
Carta 1      Carta 2      Carta 3      Carta 4      Carta 5
* * ****    * * ****    * * ****    * * ****    * * ****

Fondo: $****
Apuesta: $100

TOTAL = $400
```

ULTIMA RONDA DE CAMBIOS

Cuántas cartas quiere cambiar: 2
Que carta quiere cambiar?(1-5): 3

Cambio carta (3): 10 T negro -----> 9 C rojo

Que carta quiere cambiar?(1-5): 5

Cambio carta (5): 8 C rojo -----> Q R rojo

Se abre la última ronda de apuestas, donde como las anteriores un 0 representa que el jugador se retira o si no cuenta con la cantidad mínima, el sistema lo retirará.

"C:\Respaldo\FIMEE\Plan Semestral\Semestre 2\Ingenieria de Software\Baraja\bin\Debug\Baraja.exe"

--> JUGADOR <--

Carta 1	Carta 2	Carta 3	Carta 4	Carta 5
3 E negro	3 T negro	3 C rojo	9 C rojo	Q R rojo

Fondo: \$900
Apuesta: \$100

--> PC1 <--

Carta 1	Carta 2	Carta 3	Carta 4	Carta 5
* * ****	* * ****	* * ****	* * ****	* * ****

Fondo: \$****
Apuesta: \$100

--> PC2 <--

Carta 1	Carta 2	Carta 3	Carta 4	Carta 5
* * ****	* * ****	* * ****	* * ****	* * ****

Fondo: \$****
Apuesta: \$100

--> PC3 <--

Carta 1	Carta 2	Carta 3	Carta 4	Carta 5
* * ****	* * ****	* * ****	* * ****	* * ****

Fondo: \$****
Apuesta: \$100

TOTAL = \$400

ULTIMA RONDA DE APUESTAS


El jugador 2 aumenta la apuesta en \$25

El jugador 3 iguala la apuesta de \$25

El jugador 4 iguala la apuesta de \$25

Quieres igualar o aumentar la apuesta? (0 = retirarse):

Se inicia la ronda del destape, donde uno a uno se va mostrando la jugada que cada jugador cuenta en su mano.

 "C:\Respaldo\FIMEE\Plan Semestral\Semestre 2\Ingenieria de Software\Baraja\bin\Debug\Baraja.exe"

```
--> PC1 <--  
Carta 1      Carta 2      Carta 3      Carta 4      Carta 5  
5 E negro    9 T negro    J T negro    K T negro    K C rojo
```

Fondo: \$****
Apuesta: \$600

```
--> PC2 <--  
Carta 1      Carta 2      Carta 3      Carta 4      Carta 5  
7 T negro    Q C rojo     A C rojo     2 R rojo     A R rojo
```

Fondo: \$****
Apuesta: \$600

```
--> PC3 <--  
Carta 1      Carta 2      Carta 3      Carta 4      Carta 5  
N N blanco   4 E negro    8 E negro    A E negro    J R rojo
```

Fondo: \$****
Apuesta: \$600

TOTAL = \$2400

DESTAPES!

JUGADOR:
Obtuvo una TERCIA
Valor: 3

PC1:
Obtuvo un PAR
Valor: K

PC2:
Obtuvo un PAR
Valor: A

PC3:
Obtuvo un PAR
Valor: A

Y al final, el sistema indica que mano fue la ganadora y a quien le entregará el monto de la apuesta total. Además se le pregunta si quiere seguir jugando, donde un 1 representa un sí y un 2 representa un no.

"C:\Respaldo\FIMEE\Plan Semestral\Semestre 2\Ingenieria de Software\Baraja\bin\Debug\Baraja.exe"

```
--> JUGADOR <--
Carta 1      Carta 2      Carta 3      Carta 4      Carta 5
3 E negro    3 T negro    3 C rojo     9 C rojo     Q R rojo

Fondo: $400
Apuesta: $600

--> PC1 <--
Carta 1      Carta 2      Carta 3      Carta 4      Carta 5
5 E negro    9 T negro    J T negro    K T negro    K C rojo

Fondo: $****
Apuesta: $600

--> PC2 <--
Carta 1      Carta 2      Carta 3      Carta 4      Carta 5
7 T negro    Q C rojo     A C rojo     2 R rojo     A R rojo

Fondo: $****
Apuesta: $600

--> PC3 <--
Carta 1      Carta 2      Carta 3      Carta 4      Carta 5
N N blanco   4 E negro    8 E negro    A E negro    J R rojo

Fondo: $****
Apuesta: $600

TOTAL = $2400
```

```
### PREMIACION ###

JUGADOR GANADOR!
Obtuvo una TERCIA
Valor: 3
Obtiene $2400

Quiere seguir jugando(1=Si/2=No): _
```

Es importante mencionar que si se retira el jugador del juego, solamente si se el jugador rompió un record ya registrado, se le pedirá que ingresé su nombre para poderlo registrar.

Documentación de las funciones internas

Las funciones creadas y su descripción breve de cada una de ellas son las siguientes:

Librería Baraja

- ***void imprimirElementoCarta (carta nombre, int n)***. Imprime el elemento específico de una carta.
- ***void imprimirCaractCarta(carta nombre)***. Imprime todas las características de una carta.
- ***void imprimirCaractBaraja(carta baraja[], int length)***. Imprime todas las características de un arreglo de cartas (baraja).
- ***int definirCarta(int id, int num, int value, char type, char* color, carta *elemento)***. Crea una carta con todos los elementos que contiene.
- ***void liberarMemoria(carta baraja[], int length)***. Libera la memoria de los elementos de un arreglo que usan memoria dinámica.

- ***int crearBaraja(carta baraja[])***. Crea un arreglo de tipo carta (baraja) con todos los tipos de cartas, y los elementos de cada una de ellas.
- ***void barajear (carta baraja[])***. Desordena la baraja para que los elementos siempre salgan de manera aleatoria.

Librería Poker

- ***void quitarJokers(carta baraja[], carta nBaraja[])***. Quita dos elementos carta del arreglo y crea una nueva baraja sin ellos.
- ***int repartirMano(carta baraja[], carta mano[], int n, int *carry, int max)***. Toma del arreglo baraja cinco cartas para darle a cada jugador.
- ***void imprimirMano(carta mano[])***. Imprime la mano que tiene el jugador.
- ***void asignarValor(carta baraja[], int length)***. Define el valor de las cartas a este juego en específico.
- ***void ordenarCartas(carta monton[], int length)***. Ordena las cartas de acuerdo al tipo y al valor de la carta para hacer un poco más visible la mano.
- ***int prioridadTipo(char tipo)***. Auxilia a la función “*void ordenarCartas(carta monton[], int length)*”.
- ***int sacarCarta(carta baraja[], carta mano[], int posicion, int carry, int max)***. Permite sacar una carta de la baraja y la reemplaza por la carta que se le indique.
- ***int cambios(carta baraja[], carta mano[], int *carry, int max)***. Realiza los cambios de cartas de los jugadores.
- ***void borde(int n)***. Imprime un par de líneas que simula un borde.
- ***void datos(jugador travis, int oculto)***. Imprime los datos básicos del jugador para que se pueda observar su avance en el juego.
- ***int apostar(int *fondo, int *apuesta, int *total, int numero, int n)***. Registra la apuesta del jugador (humano).
- ***void inicioDescarte(int a[])***. Inicializa un arreglo de 5 elementos a -1.
- ***int contarComodines (carta mano[])***. Cuenta los comodines existentes en la mano.
- ***int checarRepeticionTipo (carta mano[], char *tipo)***. Cuenta cuantas cartas son repetidas en cuanto a tipo (espadas, tréboles, corazones y rombos).
- ***int cartaMasAlta (carta mano[])***. Obtiene el ID de la carta más alta que se encuentra en la mano.
- ***int comprobarPares (carta mano[], int pares[], int posiciones[], int comodines)***. Comprueba si hay 1 o 2 pares en la mano.
- ***void probarPar (carta mano[], int posiciones[])***. Da las 4 cartas que se deben cambiar para obtener mínimo un par.
- ***int comprobarTrio (carta mano[], int posiciones[], int comodines)***. Comprueba si en la mano existe un trio de cartas.
- ***int probarTrio (carta mano[], int posiciones[], int comodines)***. Da las cartas faltantes para formar un trío, y las posiciones de cambio.
- ***int comprobarEscalera(carta mano[], int comodines)***. Comprueba si en la mano existe una jugada escalera.
- ***int probarEscalera (carta mano[], int posiciones[], int comodines)***. Da las cartas faltantes para formar una jugada escalera y las posiciones de cambio.
- ***int comprobarColor (carta mano[], int comodines)***. Comprueba si en la mano existe una jugada color.

- ***int probarColor (carta mano[], int posiciones[], int comodines)***. Da las cartas faltantes para formar una jugada color y las posiciones de cambio.
- ***int comprobarFullHouse (carta mano[], int posiciones[], int comodines)***. Comprueba si en la mano existe una jugada full.
- ***int probarFullHouse (carta mano[], int posiciones[], int comodines)***. Da las cartas faltantes para formar una jugada full y las posiciones de cambio.
- ***int comprobarPoker (carta mano[], int posiciones[], int comodines)***. Comprueba si en la mano existe una jugada poker.
- ***int probarPoker (carta mano[], int posiciones[], int comodines)***. Da las cartas faltantes para formar una jugada de poker y las posiciones de cambio.
- ***int probarEscaleraColor(carta mano[], int posiciones[], char palo, int comodines)***. Analiza cuantas cartas se necesitan cambiar para formar una escalera de color y da las posiciones de dichas cartas.
- ***int comprobarEscaleraColor (carta mano[], char tipo, int comodines)***. Checa si existe una escalera de color en la mano del jugador.
- ***int probarEscaleraReal (carta mano[], int posiciones[], char palo, int comodines)***. Analiza cuantas cartas se necesitan cambiar para formar una escalera real y da las posiciones de dichas cartas.
- ***int comprobarEscaleraReal (carta mano[], char tipo, int comodines)***. Checa si existe una escalera real en la mano del jugador.
- ***int checarMano (jugador hp)***. Permite checar las posibles jugadas de la computadora.
- ***int sacarCambios(int a[])***. Regresa la cantidad de cambios que debe hacerse en una mano.
- ***int sacarArregloCambios(carta mano[], int jugadas[], int a[])***. Saca un arreglo que contiene las posiciones de cambio dependiendo de la jugada de la PC.
- ***int sacarValorJugada(int jugadas[])***. Transforma la jugada de la mano, en un valor entero que la representa para comparar con otras jugadas.
- ***int sacarMontoTotal(jugador *a, jugador *b, jugador *c, jugador *d, int n, int fase)***. Realiza las apuestas del juego y hace sus respectivos cálculos.
- ***int sacarValorNumJugada(int ventaja[])***. Regresa del arreglo de jugadas, la más alta de ellas y la que representará al jugador.
- ***void mensajeManoJugada (int jugada, int ventaja[])***. Muestra en un mensaje, la jugada que tiene cada jugador en la mano.
- ***int analizarGanador (jugador a, jugador b, jugador c, jugador d)***. Analiza y compara las jugadas de cada jugador y regresa quien tiene la mejor de ellas.
- ***int menu(void)***. Imprime el menú de entrada y regresa la opción elegida por el usuario.
- ***int checarRegistro (int cantidad)***. Checa en el registro, si la cantidad del jugador es mayor a una de las que ya se encuentran.
- ***void nuevoRegistro(char nombre[], int cantidad, int posicion)***. Crea un nuevo registro (actualiza) con el nombre y la cantidad a ingresadas.
- ***void mostrarMarcadores(void)***. Muestra los marcadores al usuario.

Resultado

Los resultados fueron en gran parte los obtenidos ya que muchas de las metas planteadas con anterioridad fueron cumplidas.

Requerimientos cumplidos (y no cumplidos)

Los requerimientos cumplidos fueron casi en su totalidad:

- Poder jugar de manera igual con una persona humana el juego de póker, usando las mismas reglas para ambos.
- Presentar un reto para el jugador, creando una simulación de inteligencia tomando métodos de heurísticas y predicciones para la realización de jugadas.
- Dar incentivo al jugador de que puede dejar su registro al hacer muy buenas jugadas.
- Probar todas y cada una de las funciones para ejercitar todo camino posible.
- Documentar totalmente la realización con el apoyo de un servicio de repositorio.
- Crear un proyecto (o dividirlos de manera distinta en su caso) por cada librería a probar, tanto pruebas unitarias, como las pruebas de integración del proyecto.

Sólo detalles insignificantes pero el programa en sí cumple, con cada una de las metas propuestas anteriormente.

Bugs conocidos no depurados

Los bugs que no fueron depurados y que se encontraron fueron dos principalmente:

- Uno de ellos que se desborda la memoria después de estar jugando ya determinado tiempo, se encontró en las pruebas Alpha, y por limitación de tiempo no se encontró la razón de tal efecto. Esto provoca que el programa falle y se cierre.
- El reemplazo de las letras por símbolos. A lo largo del proyecto se buscó reemplazar las letras que representaban los tipos de cartas por sus respectivos símbolos, pero si saber la causa, estos solo eran aceptados en otras computadoras con la misma versión del compilador. Por más que se buscó la solución a tal problema, no se encontró una forma que pudiera arreglarlo.

Conclusiones

El proyecto aunque lo sentí sencillo, me permitió realizar pruebas sobre las que vimos en el curso. Pude notar la presión del tiempo y como, a pesar de que haces las pruebas en cada función que se realiza, se siente como no se avanza en el trabajo. También, experimenté en carne propia, la utilidad de las pruebas y de las reuniones ya que gracias a ellas, pude observar errores que por mí solo me hubiera costado mucho más trabajo y tiempo hacerlo, lo que demuestra que la gráfica al inicio del curso mostrada al inicio del curso es cierta, aparentemente se avanza más lento, pero al final el trabajo es más robusto y el tiempo es menor a que si se hubiera hecho sin una técnica de programación.

También me quedé con la sensación de seguir desarrollando el proyecto, con miles de ideas de implementar e incluso pasar el código a otra plataforma, pero por el mismo tiempo, creo que son

cosas que iré dando poco a poco además de ir puliendo el código. El proyecto me dejó bien fundamentado lo que son todos los términos que se vieron en clase.

Valoración de la calidad del proyecto.

Considero que el producto final cumple con lo esperado, se juega de manera eficiente y la computadora responde de acuerdo a lo indicado. Errores que pudieron ser fatales se corrigieron a tiempo a la vez que se incrementaba en la eficiencia del juego. En sí, creo que puedo decir que estoy orgulloso del producto final y de la manera en la que se obtuvo.