```
void mostrarMenu(int &opc);
```

OPCION 1 (JUGAR)

```
void cargarMazoDeCartas(string vecCartas[], int vecPuntajeCartas[]);
```

`string vecCartas[20]` Nombres de los palos

`int vecPuntajeCartas[20]` Puntaje de los naipes

Codigo	Carta	Puntaje
0	10 de Corazon	10
1	J de Corazon	11
2	Q de Corazon	12
3	K de Corazon	15
4	A de Corazon	20
5	10 de Diamantes	10
6	J de Diamantes	11
7	Q de Diamantes	12
8	K de Diamantes	15
9	A de Diamantes	20
10	10 de Pica	10
11	J de Pica	11
12	Q de Pica	12
13	K de Pica	15
14	A de Pica	20
15	10 de Trebol	10
16	J de Trebol	11
17	Q de Trebol	12
18	K de Trebol	15
19	A de Trebol	20
	Lo que se carga en vecCartas[]	Lo que se carga en vecPuntajeCartas

```
void determinarContrincantes(string &jugador1, string &jugador2);
```

```
vecCodigosValores[10]
void GenerarNaipes(int vecCodigosValores[]);
```

Genera un vector booleano `bool vectorBanderas[20];` del 0 al 19 inicializado en false.

Genera 10 números aleatorios 1-19 y por cada uno, confirma que en el vector booleano, dicho índice no esté ya cargado. En caso afirmativo, vuelve a generar un número aleatorio. Carga `vecCodigosValores[10]` con los códigos aleatorios.

```
void obtenerNaipe(int codigo, string &carta, string vecCartas[]);
```

Mediante la utilización de `string vecCartas[20]` (núm. + nombre) e `(int codigo,` (índice) devuelve un naipe ej.: “10 de CORAZONES”.

```
int vecCartasJugador1Ronda[5], vecCartasJugador2Ronda[5];
void Repartir(int vecCodigosValores[], string vecCartas[], int vecCartasJugador1Ronda1[],
```

```
int vecCartasJugador2Ronda1[], string nombreJugador1, string nombreJugador2);
```

Mediante el uso de la función `void obtenerNaipe` asigna a los vectores **vecCArtasJugador** (1 y 2) **Ronda** los naipes correspondientes a la actual ronda y los imprime por pantalla.

```
void DarVueltaEmbaucadora(string &nombre);
```
Genera un número aleatorio del 0 al 4 e imprime por pantalla el palo respectivo.

```
void ManejarEmbaucadora(string &nombre, int ronda, string nombreJugador1, string nombreJugador2,
int puntajeTotalJugador1, int puntajeTotalJugador2, bool &menos20J1, bool &menos20J2)
```

Dependiendo de la ronda y del puntaje disponible de los jugadores, da la opción de cambiar la Embaucadora y registra mediante booleanos cuál de ellos usó 20 puntos para dicho beneficio.

==========

```
int calcularPuntajeDeRonda(string figura, int vecCartasDelJugador[], int vecPuntajeCartas[], bool& menos20);
```

Determina el palo de la Embaucadora y su respectivo 1º índice, relativo a la siguiente tabla:

	Carta	Puntaje
0	10 de Corazon	10
1	J de Corazon	11
2	Q de Corazon	12
3	K de Corazon	15
4	A de Corazon	20
5	10 de Diamantes	10
6	J de Diamantes	11
7	Q de Diamantes	12
8	K de Diamantes	15
9	A de Diamantes	20
10	10 de Pica	10
11	J de Pica	11
12	Q de Pica	12
13	K de Pica	15
14	A de Pica	20
15	10 de Trebol	10
16	J de Trebol	11
17	Q de Trebol	12
18	K de Trebol	15
19	A de Trebol	20
	Lo que se carga en vecCartas[]	Lo que se carga en vecPuntajeCartas

(Corazones = 0, Diamantes = 5, Picas = 10 y Tréboles = 15).
Determina si el palo de la embaucadora, mediante rangos, **no** corresponde a cada una de las 5 cartas del jugador y dado el caso, acumula el puntaje correspondiente a la ronda.

En main, cada ronda para ambos jugadores se asignan a los vectores:

```
int vecPuntajesRondasJugador1[3]={}, vecPuntajesRondasJugador2[3]={};
```

==========

```
int calcularPuntajeTotal(int vecPuntajesRondasJugador[]);
```

Recibe el vector con los puntajes de las tres rondas de cada jugador, los suma y los devuelve.

En main se asigna a

```
int totalPuntosJugador1 = 0, totalPuntosJugador2 = 0;
```

==========

```
void modificarPuntajeRondaAnterior(int ronda, bool menos20, int vecPuntajesRondasJugador[], string nombreJugador);
```

Determina el índice de los vectores

```
int vecPuntajesRondasJugador1[3]={}, vecPuntajesRondasJugador2[3]={};
```

Se posiciona en la ronda anterior y dependiendo de si el jugador gastó 20 puntos para cambiar la Embaucadora `bool menos20` los resta en la ronda correspondiente.

==========

FIN DE LA PARTIDA

==========

```
void mostrarTabla(int vecPuntajesRondasJugador1[], int vecPuntajesRondasJugador2[], int totalPuntosJugador1,
int totalPuntosJugador2, string nombreJugador1, string nombreJugador2);
```

Hace uso de los vectores

```
int vecPuntajesRondasJugador1[3]={}, vecPuntajesRondasJugador2[3]={};
```

..de las variables

```
int totalPuntosJugador1 = 0, totalPuntosJugador2 = 0;
```

..y de `nombreJugador1, nombreJugador2,`

..para imprimir una tabla con puntajes por ronda/jugador y puntaje total/jugador.

==========

```
void determinarGanador(string nombreJugador1, string nombreJugador2, int totalPuntosJugador1, int totalPuntosJugador2,
int vecPuntajesRondasJugador1[], int vecPuntajesRondasJugador2[], string &ganadorHistorico, int &ptsGanadorHistorico);
```

Mediante el uso de los puntajes totales, determina el ganador.
En caso de empate, compara los puntajes de ambos jugadores en cada ronda para desempatar.

Hace uso de un registro histórico de puntaje máximo para mantener actualizadas las estadísticas respecto al puntaje máximo obtenido con su respectivo jugador.

==========

```
void resetearValores(int &ronda, int &totalPuntosJugador1, int &totalPuntosJugador2,
int vecPuntajesRondasJugador1[], int vecPuntajesRondasJugador2[]);
```

Pone en cero:

```
int totalPuntosJugador1 = 0, totalPuntosJugador2 = 0;
```

```
int vecPuntajesRondasJugador1[3]={}, vecPuntajesRondasJugador2[3]={};
```

y posiciona en 1 la variable **int ronda**

==========

OPCION 2 (ESTADÍSTICAS)

```
void mostrarEstadisticas(int ptsGanadorHistorico, string ganadorHistorico);
```

Usa los valores determinados para las variables

```
string ganadorHistorico;
int ptsGanadorHistorico=0;
```

En la función **determinarGanador** e imprime en pantalla el ganador histórico con su puntaje obtenido.

==========

OPCION 3 (CREDITOS)

```
void mostrarCreditos();
```

==========

OPCION 0 (SALIR)

```
return 0;
```

No se ejecuta el while principal

```
while(true)
```

==========