INFORME

Alumnos: Sandra Rojas Benjamín Gálvez

Descripción del juego

1-Como se juega: En este juego el crupier reparte dos cartas visibles a cada jugador. El valor del As puede ser 11 o 1, las figuras (J, Q, K) valen 10, y las cartas numéricas su valor nominal. El valor del As se ajusta según sea necesario para no pasar de 21. Si un jugador recibe un As junto con una carta de valor 10 en las dos primeras cartas, obtiene blackjack automáticamente y gana la apuesta, a menos que el crupier también obtenga blackjack. Después de repartir las dos primeras cartas a cada jugador, el crupier muestra su primera carta boca arriba, visible para todos los jugadores, quienes basan sus decisiones en esa carta. El crupier tiene una segunda carta boca abajo, que se revela más tarde. Cada jugador compite únicamente contra el crupier y no contra los otros jugadores.

2-Baraja y Cartas: Se juega típicamente con una o más barajas de cartas estándar (52 cartas cada una). Cada carta tiene un valor numérico o de figura:

- Las cartas del 2 al 10 valen su valor nominal.
- Las cartas J, Q y K valen 10 puntos cada una.
- El As puede valer 1 u 11 puntos, dependiendo de la mano

3-Objetivo: El jugador gana si su mano suma más puntos que la del crupier sin exceder 21 puntos. Si el jugador supera 21 puntos, pierde automáticamente (se llama "bust").

4-Desarrollo del Juego:

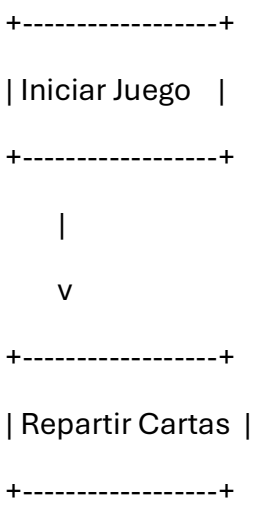
- **Reparto Inicial**: El crupier reparte dos cartas visibles a cada jugador, incluyéndose a sí mismo. El crupier muestra una de sus cartas y deja la otra boca abajo ("carta tapada").
- **Juego del Jugador**: Cada jugador decide si pide más cartas ("pedir") para acercarse a 21 o se planta con su mano actual ("plantarse"). También puede optar por "doblar" su apuesta al recibir dos cartas iniciales.

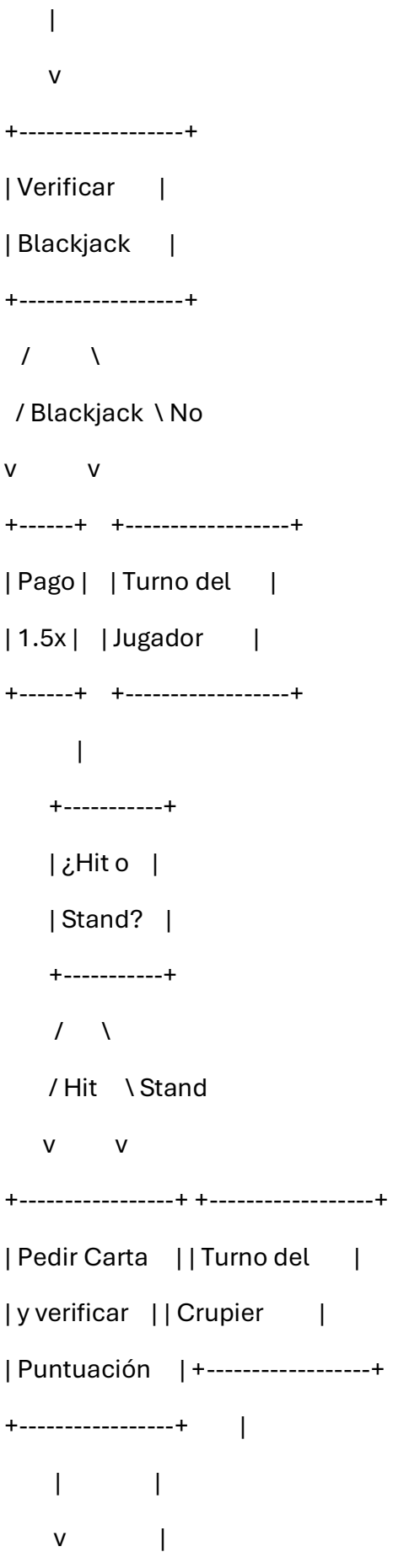
- **Juego del Crupier**: Una vez que todos los jugadores han tomado sus decisiones, el crupier revela su carta tapada. El crupier debe seguir reglas predefinidas, generalmente pedirá cartas adicionales hasta alcanzar un valor mínimo (por ejemplo, 17).
- **Determinación del Ganador**: Se comparan las manos del jugador y del crupier:
  Si el jugador tiene una mano más cercana a 21 que el crupier sin pasarse, el jugador gana y recibe una ganancia igual a su apuesta.
  Si el crupier supera los 21 puntos, todos los jugadores que no hayan superado 21 ganan.
  Si el crupier tiene una mano más cercana a 21 que todos los jugadores, estos pierden su apuesta.
  Un "Blackjack" se forma con un As y una carta de valor 10 en las dos primeras cartas, ganando 1.5 veces la apuesta inicial, a menos que el crupier también tenga un Blackjack.

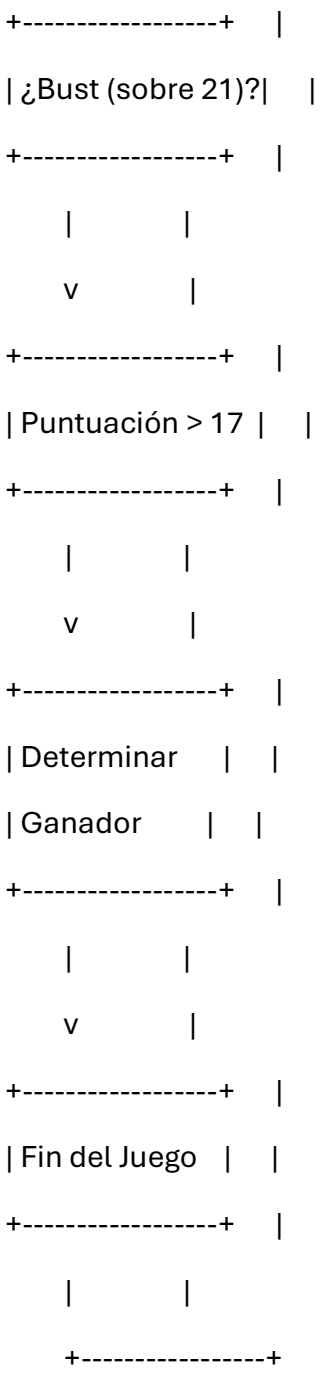
  5- Decisiones y Estrategias
  Los jugadores deben tomar decisiones basadas en la carta visible del crupier y en su propia mano para maximizar sus posibilidades de ganar, como pedir cartas adicionales, dividir pares o doblar la apuesta

  6- **Variantes y Reglas Adicionales**: Hay numerosas variantes del Blackjack que pueden incluir reglas adicionales como dividir cartas, rendirse antes de que se revele la carta tapada del crupier, o asegurar contra un posible Blackjack del crupier.

Diagrama




Verificar Blackjack
Blackjack
No
Pago 1.5x
Turno del Jugador
¿Hit o Stand?
Hit
Stand
Pedir Carta y verificar Puntuación
Turno del Crupier



Este diagrama de flujo detalla los pasos del juego de blackjack, incluyendo la verificación de blackjack inicial, las decisiones del jugador (pedir carta o quedarse), el juego del crupier hasta que alcanza una puntuación mínima de 17, la determinación del ganador y el final del juego.

PASO POR PASO

- `iostream`: Necesario para entrada y salida estándar.
- `vector`: Utilizado para almacenar las cartas de la baraja y gestionarlas dinámicamente.
- `cstdlib` y `ctime`: Utilizados para generación de números aleatorios (`rand()` y `srand()` respectivamente).

```
#include <iostream>
#include <vector>
#include <cstdlib>
#include <ctime>
```

`obtenerNumeroAleatorio`: Función que devuelve un número aleatorio entre `min` y `max`, utilizando `rand()` para la generación y `srand()` para la inicialización.

```
int obtenerNumeroAleatorio(int min, int max) {
    static bool inicializado = false;
    if (!inicializado) {
        srand(time(nullptr));  // Inicialización del generador de números aleatorios
        inicializado = true;
    }
    return min + rand() % ((max + 1) - min);
}
```

`barajarCartas`: Función que baraja aleatoriamente las cartas en el vector `baraja`, utilizando `obtenerNumeroAleatorio()` para seleccionar índices aleatorios.

```
void barajarCartas(vector<Carta> &baraja) {
    for (size_t i = 0; i < baraja.size(); ++i) {
        int indiceAleatorio = obtenerNumeroAleatorio(0, baraja.size() - 1);
        swap(baraja[i], baraja[indiceAleatorio]);
    }
}
```

- `crearBaraja`: Función que crea y devuelve una baraja completa de cartas.
- Utiliza vectores para almacenar las cartas con diferentes palos, valores y puntajes.

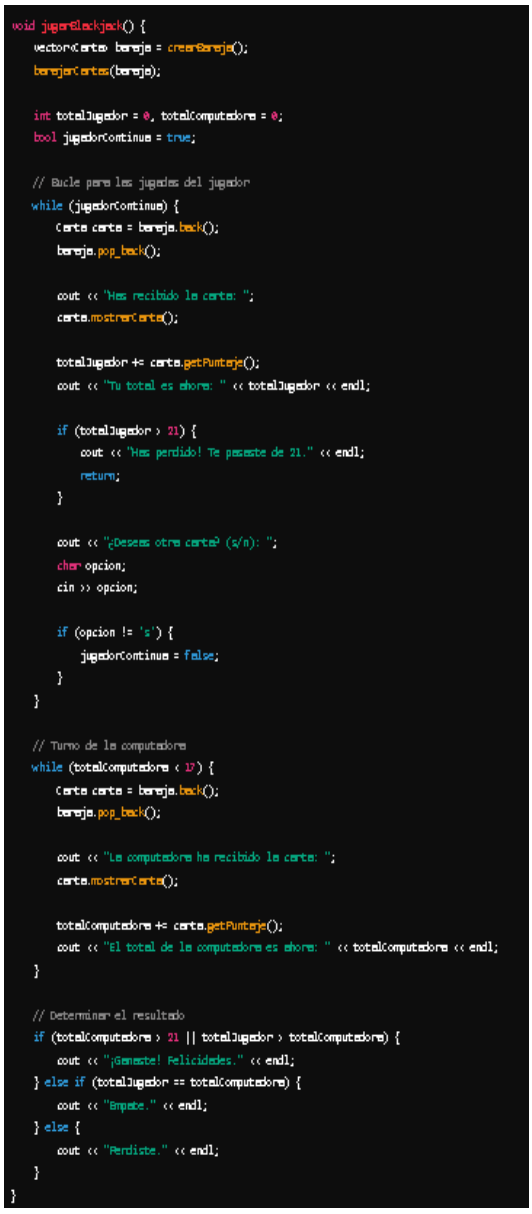
```
vector<Carta> crearBaraja() {
    vector<Carta> baraja;
    string palos[] = {"Corazones", "Diamantes", "Treboles", "Picas"};
    string valores[] = {"As", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K"
    int puntajes[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 10, 10};

    for (const auto &palo : palos) {
        for (size_t i = 0; i < sizeof(valores) / sizeof(valores[0]); ++i) {
            baraja.push_back(Carta(palo, valores[i], puntajes[i]));
        }
    }

    return baraja;
}
```

- `Carta`: Clase que representa una carta con atributos `palo`, `valor` y `puntaje`.
- El constructor inicializa estos atributos.
- `getPuntaje()`: Devuelve el puntaje de la carta.
- `mostrarCarta()`: Muestra la carta por consola en formato "Valor de Palo".

```
class Carta {
private:
    string palo;
    string valor;
    int puntaje;
public:
    Carta(string p, string v, int punt) : palo(p), valor(v), puntaje(punt) {}
    int getPuntaje() { return puntaje; }
    void mostrarCarta() {
        cout << valor << " de " << palo << endl;
    }
};
```

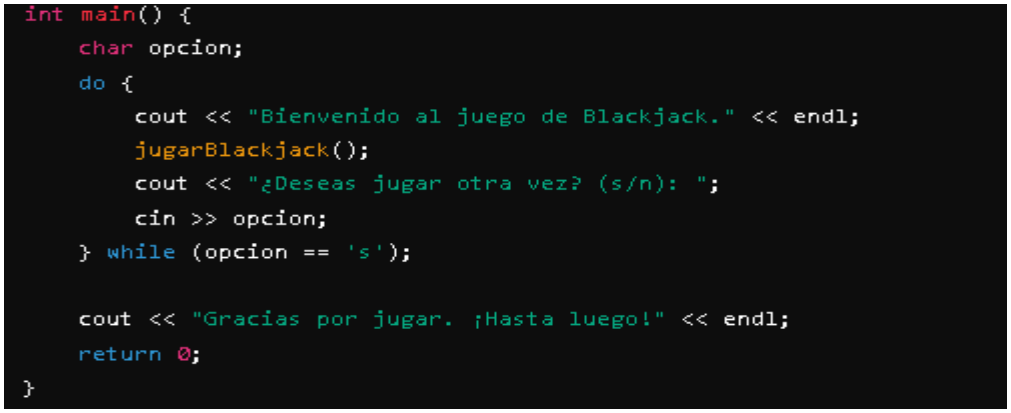
- `jugarBlackjack`: Función principal que maneja todo el flujo del juego de blackjack.
- Crea una nueva baraja, la baraja, la baraja y distribuye las cartas.
- Permite al jugador tomar decisiones sobre qué hacer después de recibir cada carta.

- La computadora juega automáticamente hasta alcanzar o superar un total de 17.
- Finalmente, determina y muestra el resultado de la partida.



```
void jugarBlackjack() {
    vector<Carta> baraja = crearBaraja();
    barajarCartas(baraja);

    int totalJugador = 0, totalComputadora = 0;
    bool jugadorContinua = true;

    // Bucle para las jugadas del jugador
    while (jugadorContinua) {
        Carta carta = baraja.back();
        baraja.pop_back();

        cout << "Has recibido la carta: ";
        carta.mostrarCarta();

        totalJugador += carta.getPuntaje();
        cout << "Tu total es ahora: " << totalJugador << endl;

        if (totalJugador > 21) {
            cout << "Has perdido! Te pasaste de 21." << endl;
            return;
        }

        cout << "¿Deseas otra carta? (s/n): ";
        char opcion;
        cin >> opcion;

        if (opcion != 's') {
            jugadorContinua = false;
        }
    }

    // Turno de la computadora
    while (totalComputadora < 17) {
        Carta carta = baraja.back();
        baraja.pop_back();

        cout << "La computadora ha recibido la carta: ";
        carta.mostrarCarta();

        totalComputadora += carta.getPuntaje();
        cout << "El total de la computadora es ahora: " << totalComputadora << endl;
    }

    // Determinar el resultado
    if (totalComputadora > 21 || totalJugador > totalComputadora) {
        cout << "¡Ganaste! Felicidades." << endl;
    } else if (totalJugador == totalComputadora) {
        cout << "Empate." << endl;
    } else {
        cout << "Perdiste." << endl;
    }
}
```

- `main`: Función principal del programa.
- Utiliza un bucle `do-while` para permitir al jugador jugar varias rondas.
- Llama a `jugarBlackjack()` para iniciar y manejar una partida completa.
- Pregunta al jugador si desea jugar otra vez al final de cada partida.



```
int main() {
    char opcion;
    do {
        cout << "Bienvenido al juego de Blackjack." << endl;
        jugarBlackjack();
        cout << "¿Deseas jugar otra vez? (s/n): ";
        cin >> opcion;
    } while (opcion == 's');

    cout << "Gracias por jugar. ¡Hasta luego!" << endl;
    return 0;
}
```

CONCLUSION

En resumen, el juego de blackjack implementado en C++ proporciona una experiencia interactiva donde los jugadores pueden enfrentarse a la computadora en partidas emocionantes. A través del código proporcionado, hemos explorado cómo se gestionan las cartas, se toman decisiones estratégicas y se determina el ganador al final de cada ronda. Este proyecto no solo ha demostrado conceptos fundamentales de programación como estructuras de datos, funciones y control de flujo, sino también cómo aplicarlos en un contexto de juego realista.

Agradezco la oportunidad de haber trabajado en este proyecto y espero que el informe proporcione una comprensión clara del proceso de desarrollo y funcionamiento del juego de blackjack en C++. Si hay alguna pregunta adicional o alguna área que requiera más detalle, no dudes en contactarme. Gracias por tu atención y consideración.