

PROYECTO RISK EN C++

JUAN DIEGO TENJO

JUAN DIEGO PALACIOS

JAIRO ANDRÉS SIERRA COMBARIZA



Pontificia Universidad
JAVERIANA
Colombia

FACULTAD DE INGENIERIA

ESTRUCTURAS DE DATOS

PONTIFICIA UNIVERSIDAD JAVERIANA

BOGOTÁ D.C

2023

Introducción:

En el mundo de los juegos de mesa estratégicos, Risk® ha sido una elección preferida por generaciones de entusiastas de estrategia y diplomacia. Con sus complejas interacciones geopolíticas y la necesidad de tomar decisiones estratégicas críticas, Risk® ofrece un desafío emocionante para jugadores de todas las edades. En este informe, abordaremos un emocionante proyecto que busca llevar el juego Risk® al mundo digital mediante la implementación de un sistema de apoyo basado en estructuras de datos.

El objetivo del proyecto se sumerge en una exploración detallada de cómo se puede implementar Risk® en un programa en lenguaje c++ versión 11 utilizando estructuras de datos. No solo se trata de replicar el juego en línea, sino de proporcionar un sistema que facilite la experiencia de juego, viéndolo desde otra perspectiva y permitiendo a los jugadores disfrutar de todas las complejidades estratégicas que ofrece Risk® de manera más accesible y diferente.

Además de describir la implementación técnica, este informe establece los requisitos y criterios de para el proyecto. Con los entregables claves incluyen un documento de diseño que detalla la arquitectura y funcionamiento del sistema, un plan de pruebas exhaustivo para garantizar la calidad del software, código fuente compilable que sirve como base para el sistema, y una visión de todos los temas vistos en la materia de Estructuras de Datos que involucrará la aplicación de estos conceptos a todos los miembros del grupo para el desarrollo del proyecto.

Componente 1

TADs:

Carpeta Asignación:

Territorio (en asignacion.h):

Descripción: Representa un territorio en el juego Risk® con sus atributos como nombre, jugador dueño, unidades de ejército, continente al que pertenece y listas de territorios vecinos y vecinos enemigos.

Atributos:

id (int): Identificador único del territorio.

nombre (string): Nombre del territorio.

jugador (string): Nombre del jugador que posee el territorio.

unidades_ejercito (int): Número de unidades de ejército en el territorio.

continente (string): Continente al que pertenece el territorio.

territorios_vecinos (vector<int>): Lista de identificadores de territorios vecinos.

vecinos_enemigos (vector<VecinoEnemigo>): Lista de vecinos enemigos con sus nombres y estados de enemistad.

Jugador (en asignacion.h):

Descripción: Representa un jugador en el juego Risk® con sus atributos como identificador, número de piezas, nombre, color y una lista de territorios y tarjetas que posee.

Atributos:

id (int): Identificador único del jugador.

numPiezas (int): Número de piezas que el jugador tiene disponibles.

nombre (string): Nombre del jugador.

color (string): Color asignado al jugador.

territorio (vector<Territorio>): Lista de territorios que el jugador posee.

tarjetas (vector<int>): Lista de tarjetas del jugador.

VecinoEnemigo (en asignacion.h):

Descripción: Representa a un vecino enemigo de un territorio, con información sobre su identificador, nombre y estado de enemistad.

Atributos:

id (int): Identificador único del vecino enemigo.

nombre (string): Nombre del territorio vecino.

esEnemigo (bool): Estado de enemistad del vecino enemigo.

Funciones (en asignación.cxx)

bool inicializarJuego(std::vector<Jugador>& jugadores, std::vector<Territorio>& territorios):
Función que inicializa el juego, solicita información a los jugadores, asigna territorios y realiza otras configuraciones iniciales.

void seleccionAleatoriaTerritorios(std::vector<Jugador>& jugadores, std::vector<Territorio>& territorios): Función que selecciona territorios aleatoriamente para los jugadores.

void mostrarTerritoriosAsignados(const std::vector<Jugador>& jugadores): Función que muestra los territorios asignados a cada jugador.

Carpeta Turno:

Funciones:

`void turnoJugador(std::vector<Jugador>& jugadores, std::vector<Territorio>& territorios):` Función que representa el turno de un jugador en el juego Risk®. Realiza diversas acciones como obtener nuevas unidades, realizar ataques y fortificaciones.

`void salir():` Función que permite salir del juego.

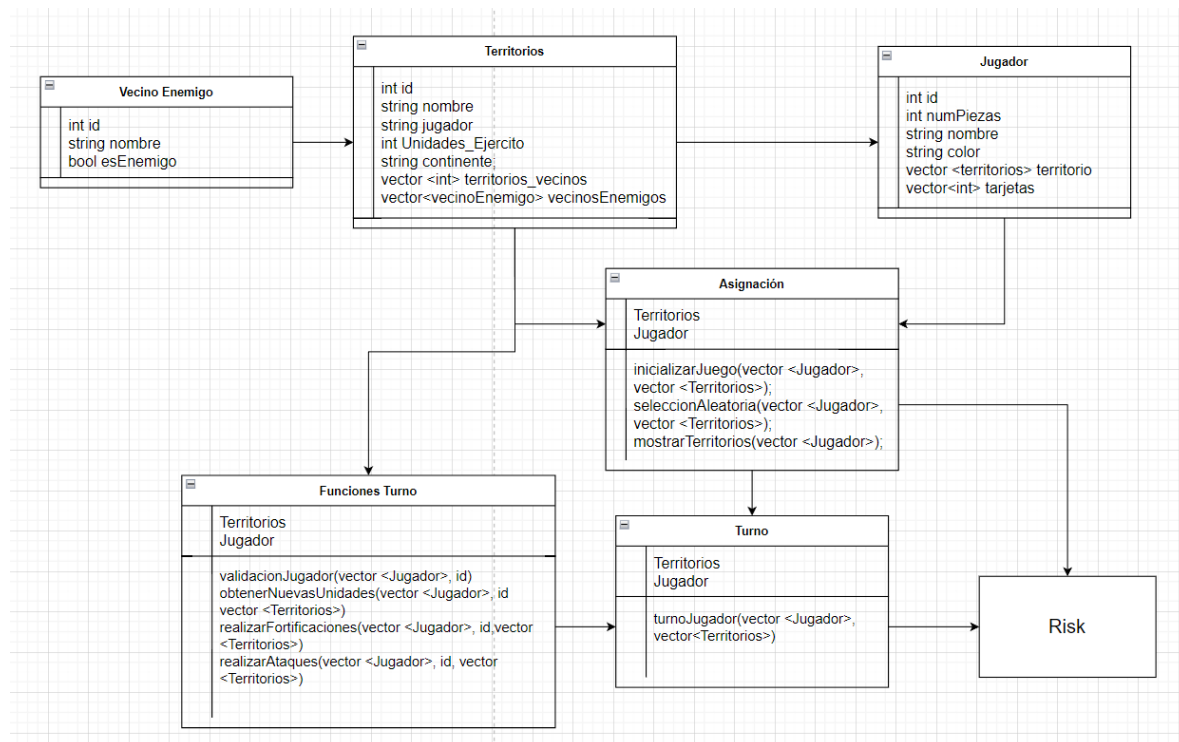
`bool validacionJugador(std::vector<Jugador>& jugadores, int id_jugador_actual):` Función que valida si el ID del jugador actual es válido.

`void obtenerNuevasUnidades(std::vector<Jugador>& jugadores, int id_jugador_actual, std::vector<Territorio>& territorios):` Función que realiza la acción de obtener nuevas unidades para el jugador actual.

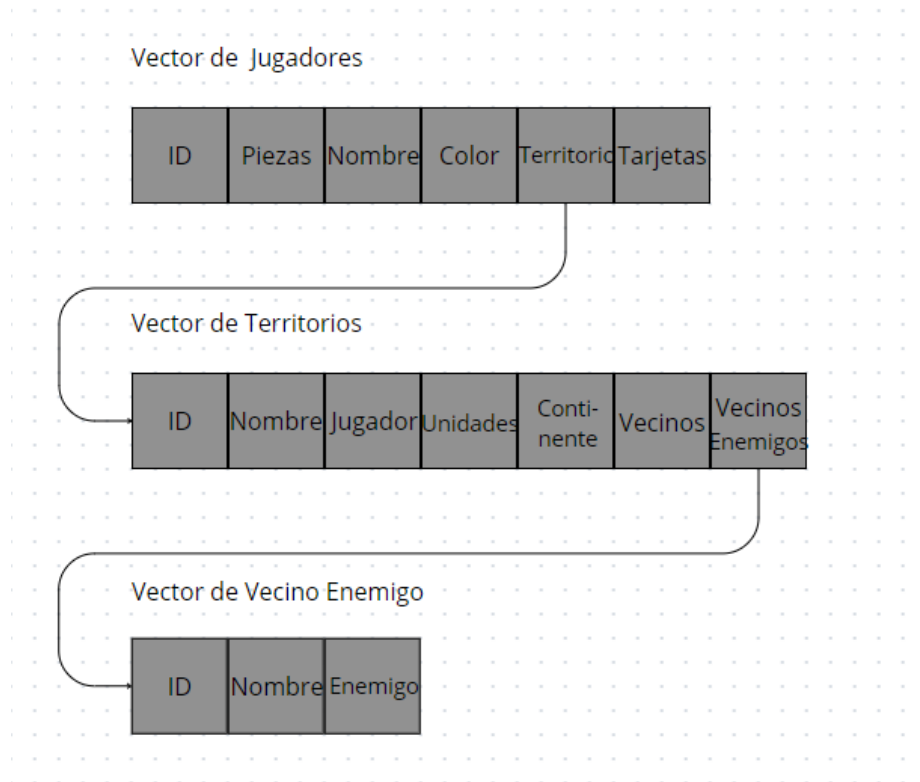
`void realizarAtaques(std::vector<Jugador>& jugadores, std::vector<Territorio>& territorios, int id_jugador_actual):` Función que permite al jugador actual realizar ataques.

`void realizarFortificaciones(std::vector<Jugador>& jugadores, std::vector<Territorio>& territorios, int id_jugador_actual):` Función que permite al jugador actual realizar fortificaciones.

Diagramas de relación:



Explicación del código:



Tenemos representados los vectores que enviamos a través de las funciones para la manipulación de estos, teniendo en cuenta que, según el jugador, el vector se combina con una lista de ellos, para representar a cada jugador.

Plan de pruebas Componente 1:

Inicializar juego (comando i):

Prueba con tres jugadores.

```

Ingresando a la funcion inicializarJuego.
Bienvenido a la inicializacion del juego Risk!
Ingrese el numero de jugadores (2-6): 3
Jugador 1: Ingrese su nombre (solo un nombre): juan
Seleccione su color (verde, azul, rojo, amarillo, negro, gris): verde
Jugador 2: Ingrese su nombre (solo un nombre): jairo
Seleccione su color (verde, azul, rojo, amarillo, negro, gris): azul
Jugador 3: Ingrese su nombre (solo un nombre): tenjo
Seleccione su color (verde, azul, rojo, amarillo, negro, gris): rojo
Jugadores registrados:
ID: 1| Nombre: juan| Color: verde
ID: 2| Nombre: jairo| Color: azul
ID: 3| Nombre: tenjo| Color: rojo
Presione una tecla para continuar . . .
  
```

Colocando valores incorrectos en los jugadores: (valor mayor al permitido)

Dato ingresado: 7

Retorno esperado del programa: (Número de jugadores no valido)

```
Ingresando a la funcion inicializarJuego.  
Bienvenido a la inicializacion del juego Risk!  
Ingrese el numero de jugadores (2-6): 7  
Numero de jugadores no valido.  
Ingrese el numero de jugadores (2-6):
```

Colocando valores incorrectos en el color:

Dato ingresado: morado

Retorno esperado del programa: (Color no valido. Seleccione un color correcto.)

```
Ingresando a la funcion inicializarJuego.  
Bienvenido a la inicializacion del juego Risk!  
Ingrese el numero de jugadores (2-6): 3  
Jugador 1: Ingrese su nombre (solo un nombre): juan  
Seleccione su color (verde, azul, rojo, amarillo, negro, gris): morado  
Color no valido. Seleccione un color correcto.  
Seleccione su color (verde, azul, rojo, amarillo, negro, gris):
```

Opción para escoger los territorios a mano o que el programa los escoja:

```
Todos los jugadores han seleccionado un territorio, ¿ Como desea continuar?  
1. Seleccionar el resto de los territorios uno por uno.  
2. Seleccion aleatoria del resto de los territorios.
```

Opción 1: el usuario escoge por sí mismo

Dato ingresado 1:

Retorno esperado del programa: Nombrar al jugador y mostrar una lista de continentes

```
Turno de juan para seleccionar continentes y territorios:  
Lista de continentes disponibles:  
1. America del Norte  
2. America del Sur  
3. Europa  
4. Africa  
5. Asia  
6. Oceania  
Seleccione un continente (1-6):
```

Opción 2: el usuario escoge aleatorio

Valor esperado del programa: lista de cada jugador con su respectivo territorio asignado

```
Territorios asignados a cada jugador:
Jugador: juan | Color: verde
Territorios asignados:
- Territorio: Ural | Continente:  | Piezas: 1
- Territorio: Venezuela | Continente:  | Piezas: 1
- Territorio: Alberta | Continente:  | Piezas: 1
- Territorio: Madagascar | Continente:  | Piezas: 1
- Territorio: Europa del norte | Continente:  | Piezas: 1
- Territorio: Africa del Sur | Continente:  | Piezas: 1
- Territorio: Egipto | Continente:  | Piezas: 1
- Territorio: Groenlandia | Continente:  | Piezas: 1
- Territorio: Estados Unidos Occidentales | Continente:  | Piezas: 1
- Territorio: Islandia | Continente:  | Piezas: 1
- Territorio: Gran Bretania | Continente:  | Piezas: 1
- Territorio: Australia Occidental | Continente:  | Piezas: 1
- Territorio: Irkutsk | Continente:  | Piezas: 1
- Territorio: Australia Oriental | Continente:  | Piezas: 1
Total de piezas de juan: 20
-----
Jugador: jairo | Color: azul
Territorios asignados:
- Territorio: Congo | Continente:  | Piezas: 1
- Territorio: Alaska | Continente:  | Piezas: 1
- Territorio: Africa Oriental | Continente:  | Piezas: 1
- Territorio: Africa del norte | Continente:  | Piezas: 1
- Territorio: Estados Unidos Orientales | Continente:  | Piezas: 1
- Territorio: Japon | Continente:  | Piezas: 1
- Territorio: Brasil | Continente:  | Piezas: 1
- Territorio: Siam | Continente:  | Piezas: 1
- Territorio: Mongolia | Continente:  | Piezas: 1
- Territorio: India | Continente:  | Piezas: 1
- Territorio: Escandinavia | Continente:  | Piezas: 1
- Territorio: Europa Occidenal | Continente:  | Piezas: 1
- Territorio: Indonesia | Continente:  | Piezas: 1
- Territorio: Kamchatka | Continente:  | Piezas: 1
Total de piezas de jairo: 20
-----
Jugador: tenjo | Color: rojo
Territorios asignados:
- Territorio: America Central | Continente:  | Piezas: 1
- Territorio: Siberia | Continente:  | Piezas: 1
- Territorio: Ontario | Continente:  | Piezas: 1
- Territorio: Europa del sur | Continente:  | Piezas: 1
- Territorio: Afghanistan | Continente:  | Piezas: 1
- Territorio: Argentina | Continente:  | Piezas: 1
- Territorio: Medio oriente | Continente:  | Piezas: 1
- Territorio: Peru | Continente:  | Piezas: 1
- Territorio: Ucrania | Continente:  | Piezas: 1
- Territorio: Nueva Guinea | Continente:  | Piezas: 1
- Territorio: Quebec | Continente:  | Piezas: 1
- Territorio: Yakutsk | Continente:  | Piezas: 1
- Territorio: China | Continente:  | Piezas: 1
- Territorio: Territorio Noroccidental | Continente:  | Piezas: 1
Total de piezas de tenjo: 20
-----
```

Componente 2:

Errores Encontrados Componente #1:

Asignación: En los territorios designados, encontramos una falla en el tiempo estimado que tarda la asignación de los 42 territorios si son pocos jugadores, por lo que implementamos la selección aleatoria que tarda unos segundos, solo teniendo el nombre y el color de los jugadores.

Turno: Por otra parte, obtuvimos un error en identificar los países vecinos de cada territorio para su ataque, por lo que decidimos implementar otra estructura que indicaba si el vecino era enemigo o no, teniendo en cuenta el flag dentro de la estructura para la especificación de los procesos con los territorios vecinos que si son enemigos.

TADs:

Guardar Partida (guardado.h):

Descripción: El guardado, tiene que en un archivo de texto ingresar la información de los participantes, con los territorios que ha conquistado y el numero de tropas que tiene cada uno, además de su identificador como participante del juego.

Funciones (guardado.cxx):

void inicializarPartida (std::string nombre_archivo, std::vector<Jugador>& jugadores, std::vector<Territorio>& territorios);
Esta función, lee los datos de los archivos e inicializa alguna partida iniciada antes que haya sido guardada.

void guardarPartida (std::string nombre_archivo, std::vector<Jugador>& jugadores, std::vector<Territorio>& territorios);
La función define los datos que se ingresan en el archivo, con la manipulación de los vectores iniciales que se manipulan durante todo el juego.

void guardarPartidaComprimida (std::string nombre_archivo, std::vector<Jugador>& jugadores, std::vector<Territorio>& territorios);
La función define, el guardado con la codificación de Huffman, la cual mediante este método, transforma el archivo de texto con la información guardada en un archivo binario según el recorrido del árbol, que se genera para su codificación.

Nodo (huffman.h):

Descripción: Es necesario representar los nodos del árbol, con la estructura teniendo en cuenta la forma en la que se crea el mismo para el proceso de codificación.

Atributos:

ch (char): El carácter que representa el nodo.

freq (int): La frecuencia con la que aparece el carácter en el archivo a codificar.

*left, *right (nodo): Apunta ya sea al nodo de la derecha o izquierda con los mismos atributos del nodo.

Comp (huffman.h):

Descripción: La estructura, retorna la frecuencia de los nodos, organizándola así por el que tiene mayor prioridad en la tabla de frecuencia construida.

Atributos:

l (Nodo*): Apunta al nodo de mayor prioridad.

r (Nodo*): Apunta al nodo de menor prioridad.

Funciones (huffman.cxx):

bool isLeaf (Node* node);

La función verifica si el árbol, contiene solo un nodo.

Node* getNode (char ch, int freq, Node* left, Node* right);

Asigna un nuevo nodo al árbol.

Node* buildHuffmanTree (const std::string& text);

Construye el árbol según la frecuencia del nodo.

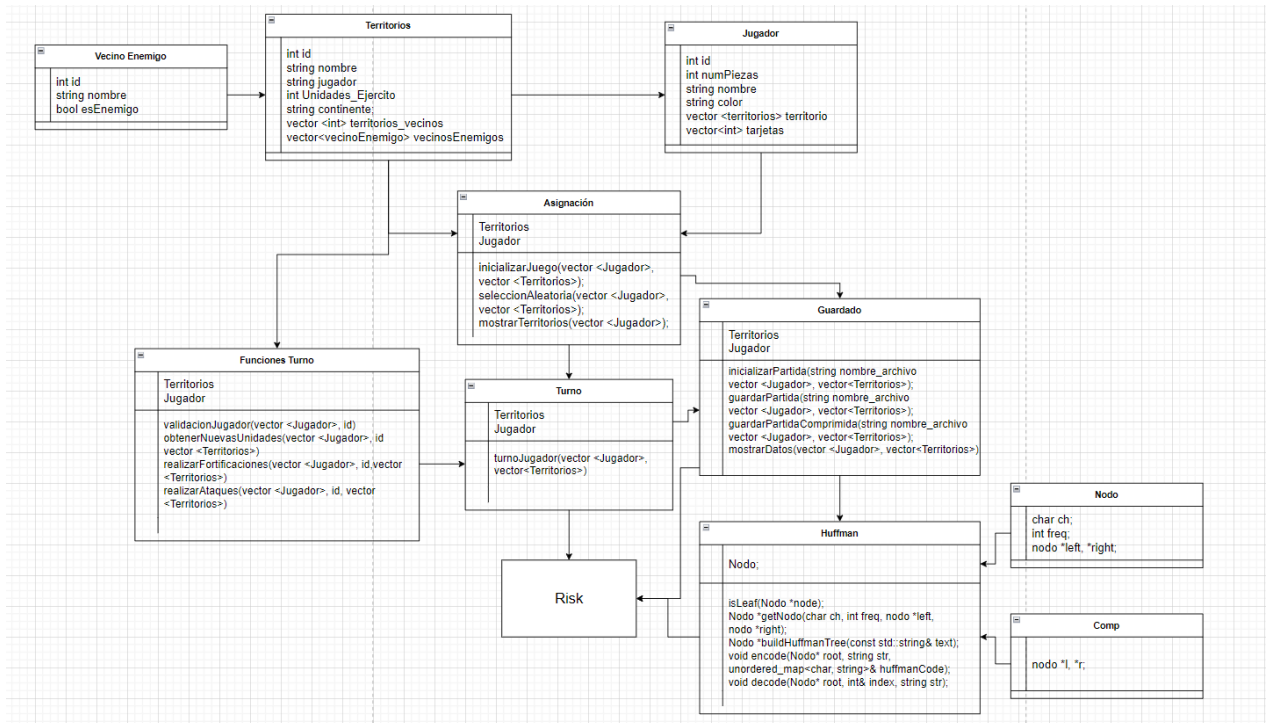
void encode (Node* root, string str, unordered_map<char, string>& huffmanCode);

La función recorre el árbol y codifica los caracteres, almacenándolos en un mapa binario.

void decode (Node* root, int& index, string str);

La función recorre el mapa binario y decodifica los caracteres según la información binaria asociada al árbol.

Diagramas de relación:

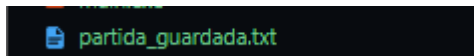


Explicación del código:

Plan de Pruebas Componente 2:

Guardar partida:

Valores esperado del programa: archivo.txt del jugo generado



```

1  JUGADOR
2  ID: 1
3  Nombre: juan
4  Número de Piezas: 20
5  Color: verde
6  Territorios
7    ID: 1
8    Nombre: Madagascar
9    PerteneceJugador: juan
0    Unidades de Ejército: 1
1    Continente: Africa
2    Territorios Vecinos:
3    Vecinos Enemigos
4      ID: 2
5      NombreVecino: Gran Bretania
6      Es Enemigo: true
7      ID: 3
8      NombreVecino: Medio oriente
9      Es Enemigo: true
0      ID: 5
1      NombreVecino: Australia Occidental
2      Es Enemigo: true
3      ID: 6
4      NombreVecino: Ucrania
5      Es Enemigo: true
6      ID: 8
7      NombreVecino: Escandinavia
8      Es Enemigo: true
9      ID: 9
0      NombreVecino: Europa Occidenal
1      Es Enemigo: true
2      ID: 11
3      NombreVecino: Europa del norte
4      Es Enemigo: true
5      ID: 12
6      NombreVecino: Egipto

```

Guardado comprimido: comando (gpc):

Prueba de compresión:

Valor esperado del programa: codificación de cada carácter


:	11111	0001	n	0000	G	1110110010	
r	11110	T	011110001	B	0100100100	0111101011	
Q	11101111111	C	011110000	i	0011	11101001101	
	11101111111011	m	01110	b	010111	011111	
R	11101111111010	o	0110	3	0100101	10	
z	1110111111100	t	01010	f	0100100101	O	111011000
S	11101111110	E	11000	7	1110111000	N	110010
p	1110111110	l	0100111	0	010010011	I	110011
h	1110111011	D	010110	a	00101	e	1101
8	1110111010	s	01000	2	11101101	6	1110100111
P	1110111001	5	011110011	-	0100110	g	111000
M	0111101000	d	1110101	©	0111101001	u	111001
9	011110010	c	00100	1	01111011	U	111010000
k	111011110	4	01001000	v	0111101010	j	111010001
						A	111010010
						Y	11101001100
						J	1110110011

```

11101100111110100001110110010111010010010110111011000111011111
001001111011110111111100001010100011111101110110101001001100010
10011100101111111011011111101110100001111000101010011100011010
111010111011011000111010001011110101101111010011111000100001101
001000111111000011010011111110100100001100000110010001011000000
100001101010101100001000101100000001101011100011111000011011111
001111000011010000010101111000101010011100011010101011000010001
101110100100100111001010100011101111000101000110101010110000100
11110111011111000110101111101111100011001010001101010101100001
111111011101001001001001011111000110010000101101110110001111000
110010111111101101011111110100100001100000110111111011101100000
110011100101111111011010111111101001000011000001101111110111010
0110101101111111001111011111011010001101010101100100110011100101
111111001010111101110011101000110101010110011010110111111001111
0011110100000101000101010011111101010000001101010101100001000101
011111101110100100100100101111000111011101100101000000110100001
111111101101011111110100100001100000110111111011101100110010111
001011111110110101111111010010000110000011011111101110100101111
101100100110011100101111111011010111111101001000011000001101111
010101011001001100111001011111110110101111111010010000110000011
10111111011101101010010100011010101011001001100111001011111101
101110110101001000000110101010110010011001110010111111101101011
110111101001110001101010101100100110011100101111111011010111111
011011111101110110111101110000001101010101100100110011100101111

```

Valores ingresados en el archivo_comprimido.txt

 archivo_comprimido.txt



Datos usando la descompresión:

Valores esperados por el sistema: la partida con todos sus jugadores y territorios

```
The decoded string is:
JUGADOR
ID: 1
Nombre: juan
Nº de Piezas: 20
Color: verde
Territorios
  ID: 1
  Nombre: Ural
  PerteneceJugador: juan
  Unidades de Ejército: 1
  Continente:
  Territorios Vecinos:
  Vecinos Enemigos:
    ID: 2
    NombreVecino: Congo
    Es Enemigo: true
    ID: 3
    NombreVecino: America Central
    Es Enemigo: true
    ID: 5
    NombreVecino: Alaska
    Es Enemigo: true
    ID: 6
    NombreVecino: Siberia
    Es Enemigo: true
    ID: 8
    NombreVecino: Africa Oriental
    Es Enemigo: true
    ID: 9
    NombreVecino: Ontario
    Es Enemigo: true
    ID: 11
    NombreVecino: Africa del norte
    Es Enemigo: true
    ID: 12
```

Componente 3 “en proceso”.

Referencias:

1. Techie Delight. (2022, 20 de julio). Codificación Huffman. [Artículo de blog]. <https://www.techiedelight.com/es/huffman-coding/>
2. García Pérez, J. L. (2023, 20 de julio). Archivos en C++. [Documento en línea]. Recuperado de http://jbgarcia.webs.uvigo.es/asignaturas/TO/cursilloCpp/14_archivos.html