UACM

Universidad Autónoma de la Ciudad de México

*Nada humano me es ajeno*

PROYECTO FINAL (CALCULAR EL NÚMERO DE DIFERENTES CAMINOS DE DOS REINAS SIN CRUZARSE EN EL TABLERO DE AJEDREZ)

ALUMNO: LARA DE LA CRUZ GUADALUPE VIRIDIANA
MUNGUIA MORALES DANIELA PERLA

MATRÍCULA: 15-011-0530
22-011-0181

MATERIA: MATEMÁTICAS DISCRETAS

PROFESORA: ELIZABETH LOPEZ LOZADA

# ÍNDICE

# RESUMEN

En este trabajo abordaremos el tema acerca del tablero de ajedrez el cómo se van moviendo las piezas, pero específicamente las reinas cuántos caminos diferentes pueden tomar sin cruzarse y atacarse mutuamente.

El objetivo de nuestro proyecto será implementar algoritmos en el lenguaje c++ que permitirá calcular los movimientos de diferentes caminos de ambas reinas sin atacarse, para ello se tendrá que entender cómo se juega en un tablero de ajedrez y el cómo funciona los movimientos. Para ello utilizaremos el algoritmo de blacktracking que funciona para explorar todas las posibles configuraciones de las dos reinas en el tablero de ajedrez.

Para mejor comprensión de la solución se dará una breve explicación, se integrará el pseudocódigo y el código comentado a detalle para un mejor entendimiento, también se utilizara fuentes de información que nos ayudara para poder resolver el problema. Como resultado del proyecto el programa resultante tendrá que ser capaz de resolver el problema planteado de las dos reinas..

# INTRODUCCIÓN

Para el siguiente proyecto tendremos que llegar como resultado el calcular cuántos caminos puede recorrer dos reinas en un tablero de ajedrez de n*n, sin que se crucen y puedan atacarse mutuamente este cálculo se llevará a cabo en programación en el lenguaje c++ para ellos ¿Que es el ajedrez ?

El ajedrez es un popular juego de mesa de tradición ancestral, cuya práctica frecuente y deportiva en Occidente data del siglo XV. El juego simula sobre un tablero cuadriculado el enfrentamiento entre dos ejércitos antiguos y asigna a cada jugador uno de los bandos, con el propósito de vencer al contrario y atrapar a su rey.

El ajedrez es un juego muy difundido y practicado en todo el mundo y en su momento fue considerado un “juego de reyes” porque era muy común entre la nobleza.

De hecho, los campeonatos mundiales de ajedrez se celebran desde 1866, y en ellos se han coronado jugadores de nacionalidades muy distintas, como Alemania, Cuba, India, Estados Unidos y, sobre todo, Rusia.

Las piezas del ajedrez son siempre las mismas, aunque son de dos colores diferentes (blancas y negras, o claras y oscuras) y están distribuidas en la misma proporción en

ambos bandos enfrentados. Cada una posee un rango propio de movimientos permitidos y una valoración en puntos. De este modo, cada jugador contará con:

- Un (1) rey, del que depende el partido. Es la única pieza que no puede ser capturada normalmente, sino que debe ser arrinconada en una posición tal que no tenga movimientos posibles y se halle bajo ataque de alguna pieza enemiga (o sea, en *jaque*). Cuando el rey está en jaque y no tiene escapatoria ni modo de defenderse, se dice que está *jaque mate* y ello sentencia el final de la partida. El rey puede moverse una sola casilla por vez en cualquier dirección deseada, y cualquier pieza que se halle en su trayectoria puede ser capturada por el rey, siempre y cuando ello no implique que el rey se exponga al *jaque*.
- Una (1) dama o reina, la pieza más poderosa del partido, con un valor de 9 puntos al ser capturada. La reina puede moverse cualquier número de casillas en cualquier dirección que se desee (horizontal, vertical o diagonal), y puede capturar cualquier pieza en las mismas direcciones.
- Dos (2) torres, con un valor de 5 puntos cada una. Estas piezas pueden moverse únicamente en recorridos horizontales y verticales, pero pueden hacerlo tantas casillas como se desee, pudiendo capturar piezas en la misma medida.
- Dos (2) alfiles, con un valor de 3 puntos cada uno. Estas piezas pueden moverse únicamente en recorridos verticales, pero pueden hacerlo tantas casillas como se desee. Por esa razón hay uno en casillas negras y otro en casillas blancas, exclusivamente. Los alfiles capturan piezas en sus respectivas casillas diagonales.
- Dos (2) caballos, con un valor de 3 puntos cada uno. Estas piezas son las más ágiles del juego, y en su recorrido pueden pasar por encima (“saltar”) otras piezas que estén atravesadas, cosa imposible en el resto de los casos. Sin embargo, su movimiento debe siempre trazar una “L”, es decir, dos casillas seguidas en una misma dirección horizontal o vertical (nunca diagonal) y luego una casilla vertical u horizontal. Por lo tanto, para capturar una pieza, ésta debe estar ubicada en esa última casilla de la “L”, y no en las casillas iniciales “saltadas” por el caballo.
- Ocho (8) peones, las piezas con menos valor del juego: 1 punto cada una. Estas piezas pueden moverse únicamente hacia adelante y una casilla por vez, excepto por su movimiento inicial (conocido como “salida”) en el que pueden optar entre avanzar una casilla o dos. Además, los peones sólo pueden capturar las piezas enemigas que se encuentren en sus dos casillas diagonales delanteras inmediatas (diagonal derecha y diagonal izquierda). Sin embargo, cuando un peón avanza sin interrupciones hasta el final de su fila en el costado enemigo del tablero, podrá ser “coronado” y cambiarse por cualquier otra pieza de juego, excepto por el rey.

Existe un movimiento especial conocido como “enroque”, en el que un rey y una torre que no se han movido aún en la partida y no se encuentran bajo amenaza alguna, pueden intercambiar sus posiciones originales, ocupando las casillas intermedias para

guarecer al rey. Existe un enroque corto (con la torre del rey) y un enroque largo (con la torre de la dama).

El tablero de ajedrez es cuadrado y está dividido en 64 casillas idénticas formando una matriz de 8×8. Mirándolo de frente, la casilla más a la izquierda debe ser siempre blanca (o de algún color claro), mientras que las dos que le siguen (arriba y abajo) deben ser negras (o de algún color oscuro) y las siguientes a esas claras, alternándose sucesivamente.

Las reglas del ajedrez se pueden resumir de la siguiente manera:

- Podrán jugar sólo dos jugadores por partida, cada uno a cargo de un bando (el claro o el oscuro). Cada bando debe sortearse antes de la partida.
- El jugador del bando claro siempre tendrá la primera jugada. El objetivo del juego es derrotar al bando contrario capturando sus piezas hasta poder acorralar al rey contrario y conducirlo al *jaque mate*.
- El juego se desarrolla por turnos. En cada turno un jugador puede mover una (1) pieza por vez. Ningún jugador puede saltarse su turno, ni está permitido retroceder las movidas a turnos previos.
- El turno de cada jugador durará lo que sea necesario, o podrá medirse a través de un reloj. En este último caso, el jugador que agote el tiempo total establecido para la partida en su turno, perderá la partida.
- Cuando una pieza enemiga es capturada, debe ser retirada del tablero y la pieza propia que la capturó pasa a ocupar su lugar. Las piezas capturadas no pueden volver al tablero. A menos que sean capturadas, las piezas en juego tampoco pueden abandonar el tablero.
- El rey es la única pieza que no puede desplazarse para ocupar una casilla amenazada por una pieza contraria. Es decir, el rey no puede nunca exponerse voluntariamente al *jaque*. Del mismo modo, ningún jugador podrá ignorar un jaque a su rey.
- Cualquier jugador puede renunciar a la partida cuando así lo desee. Ello, sin embargo, se considera otorgarle al otro la victoria.

El juego puede culminar en un empate (llamado *tablas*) cuando se dé alguna de las siguientes condiciones:

- Ambos jugadores están de acuerdo en suspender la partida sin que haya un ganador declarado.
- A ningún jugador le quedan piezas suficientes para dar al otro jaque mate.
- La misma posición de las piezas en el tablero se repite durante tres turnos consecutivos.
- Sin estar en jaque, el jugador en turno no puede realizar ninguna jugada legal (se dice que está ahogado).
- Cuando transcurren 50 jugadas consecutivas sin que se haya capturado una pieza o haya avanzado algún peón.

Ya que conocemos un poco sobre el contexto del ajedrez ahora en las piezas que nos interesa enfocarnos es en el movimiento de las reinas pero llevándolo a la programación para ello utilizaremos el algoritmo de BACKTRACKTING que consiste en una técnica poderosa para resolver problemas combinatorios y de búsqueda exhaustiva. Se utiliza para explorar todas las posibles soluciones de un problema de manera sistemática, descartando aquellas que no cumplen ciertas condiciones.

# TRABAJOS RELACIONADOS

Encontramos un trabajo relacionado con nuestro proyecto el cual utilizan la misma función de backtracking y explican un poco de como funciona, la diferencia fue que en este trabajo plantea como problema calcular n reinas (lo cual quiere decir que el usuario puede ingresar el número de reinas deseados) y nuestro trabajo sólo utiliza las dos reinas del tablero.

En este trabajo que encontramos nos basamos o tuvimos como una idea mas amplia para aplicar la función deseada en nuestro proyecto (se deja una muestra del trabajo del usuario).

### Ejemplo: Problema de las N Reinas

Un ejemplo clásico de aplicación de backtracking es el problema de las N Reinas, donde se deben colocar N reinas en un tablero de ajedrez de tal manera que no se ataquen entre sí en ninguna dirección. Aquí hay el código completo de cómo se vería la implementación en C++ utilizando las clases solucionador, solución y candidatos:

```
#include <iostream>
#include <vector>

using namespace std;

class Solucion {
public:
    Solucion(int N) : tablero(N, vector<int>(N, 0)), N(N) {}
```

```
                cout << (tablero[i][j] ? "Q" : ".");
            }
            cout << endl;
        }
    }

    vector<vector<int>> tablero;
    int N;
};

class Candidatos {
public:
    Candidatos(int N) : disponibles(N, true) {}

    vector<int> obtener() const {
        vector<int> candidatos;
        for (int i = 0; i < disponibles.size(); ++i) {
            if (disponibles[i]) {
                candidatos.push_back(i);
            }
        }
        return candidatos;
```

```
    void marcar(int pos) {
        disponibles[pos] = false;
    }

    void desmarcar(int pos) {
        disponibles[pos] = true;
    }

private:
    vector<bool> disponibles;
};

class Solucionador {
public:
    Solucionador(int N) : N(N) {}

    bool resolverNReinas(Solucion& solucion, int fila) {
        if (fila == N) {
            return true;  // Todas las reinas están colocadas
        }

        Candidatos candidatos(N);
```

```
        Candidatos candidatos(N);
        vector<int> cands = candidatos.obtener();

        for (int columna : cands) {
            if (esSeguro(solucion, fila, columna)) {
                solucion.tablero[fila][columna] = 1;
                candidatos.marcar(columna);

                if (resolverNReinas(solucion, fila + 1)) {
                    return true;
                }

                candidatos.desmarcar(columna);
                solucion.tablero[fila][columna] = 0;  // Retroceder si no
 se encuentra solución
            }
        }

        return false;  // No se encontró una posición segura para colocar
 la reina en esta fila
    }
```

```
private:
    bool esSeguro(const Solucion& solucion, int fila, int columna) const
{
        for (int i = 0; i < fila; ++i) {
            if (solucion.tablero[i][columna] == 1) {
                return false;
            }
        }

        for (int i = fila, j = columna; i >= 0 && j >= 0; --i, --j) {
            if (solucion.tablero[i][j] == 1) {
                return false;
            }
        }

        for (int i = fila, j = columna; i >= 0 && j < N; --i, ++j) {
            if (solucion.tablero[i][j] == 1) {
                return false;
            }
        }

        return true;
```

```
    int N;
};

int main() {
    int N;
    cout << "Ingrese el tamaño del tablero (N): ";
    cin >> N;

    Solucion solucion(N);
    Solucionador solucionador(N);

    if (solucionador.resolverNReinas(solucion, 0)) {
        cout << "Solución encontrada:" << endl;
        solucion.imprimir();
    } else {
        cout << "No se encontró solución para N = " << N << endl;
    }

    return 0;
}
```

## LA DESCRIPCIÓN DE LA SOLUCIÓN QUE ESTÁS IMPLEMENTANDO.

Después de haber entrado un poco en contexto podemos decir que para resolver el problema planteado utilizaremos el algoritmo backtracking en cual me será de utilidad para poder combinar los diferentes caminos que pueden tomar las reinas .

A continuación presentaré el pseudocódigo y el código de la solución del problema.

PSEUDOCÓDIGO:

1.-Se define una estructura para la posición de las reinas y el tablero.
2.-Se define una función para que la posición de una reina sea segura.
3.-Se define una función recursiva de backtracking para colocar las reinas en el tablero y contar los caminos.
4.-Iniciar una función recursiva desde la posición inicial y contar los caminos válidos.

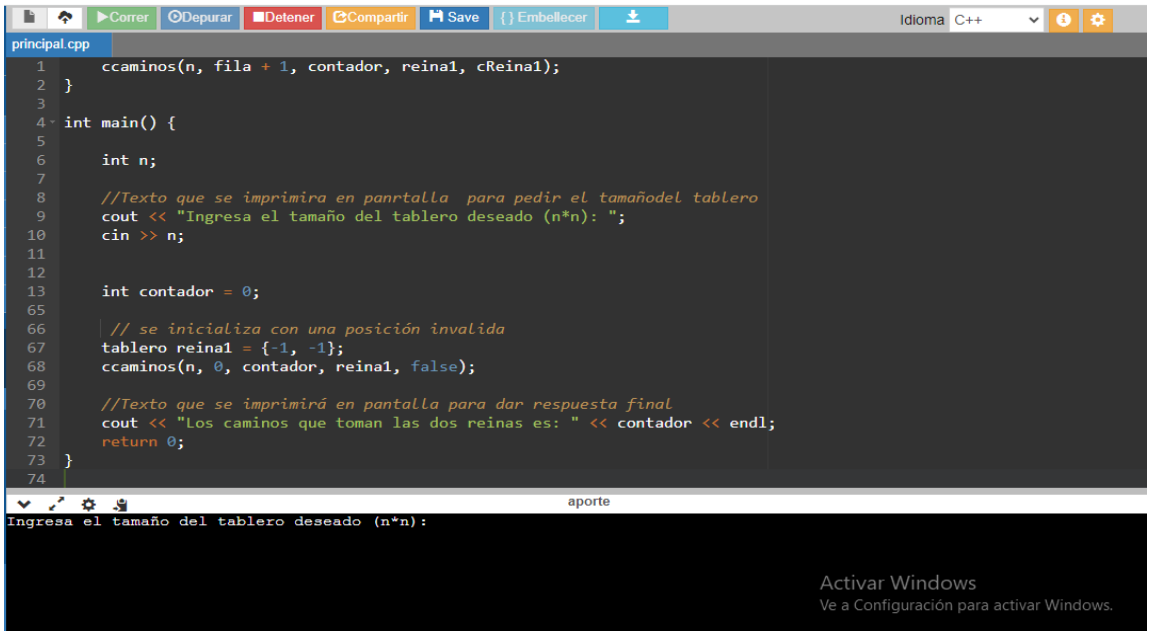
CÓDIGO:

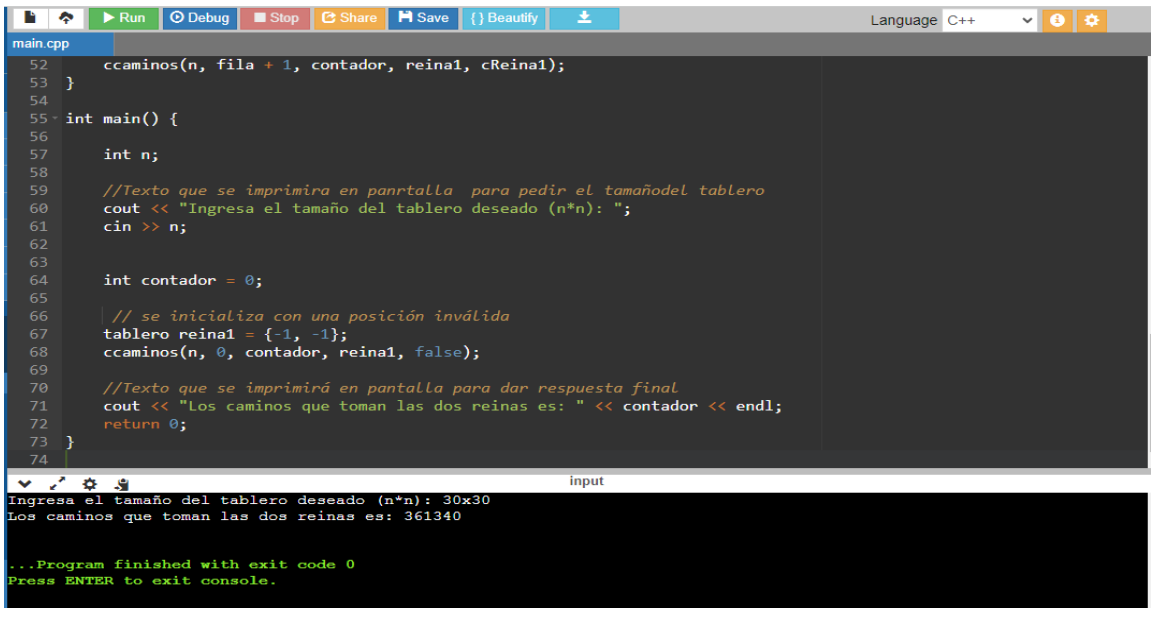
```
//librerias
#include <iostream>
#include <cmath>

using namespace std;

// Función de estructura para tener una posición en el tablero
struct tablero {
    int fila, columna;
};
```

```
// Función para ver si dos reinas se atacan
bool sAta(tablero reina1, tablero reina2) {

    //Se compara la  isma fila, columna o diagonal
    return (reina1.fila == reina2.fila || reina1.columna == reina2.columna ||
            abs(reina1.fila - reina2.fila) == abs(reina1.columna - reina2.columna));
}

// Se utiliza la función de backtracking para contar las posiciones de las reinas
void ccaminos(int n, int fila, int &contador, tablero reina1, bool cReina1) {
    if (fila == n) {
        return;
    }

    for (int columna = 0; columna < n; columna++) {
        if (!cReina1) {

            // función para poner a la primera reina
            tablero nReina1 = {fila, columna};
            ccaminos(n, fila + 1, contador, nReina1, true);


        } else {

            // función para poner a la segunda reina
            tablero reina2 = {fila, columna};
            if (!sAta(reina1, reina2)) {
                contador++;
            }
        }
    }

    ccaminos(n, fila + 1, contador, reina1, cReina1);
}

int main() {

    int n;

    //Texto que se imprimira en panrtalla  para pedir el tamañodel tablero
    cout << "Ingresa el tamaño del tablero deseado (n*n): ";
    cin >> n;


    int contador = 0;
```

```
    // se inicializa con una posición inválida
    tablero reina1 = {-1, -1};
    ccaminos(n, 0, contador, reina1, false);

    //Texto que se imprimirá en pantalla para dar respuesta final
    cout << "Los caminos que toman las dos reinas es: " << contador << endl;
    return 0;
}
```

# RESULTADO



# DISCUSIÓN

El proyecto fue lo que esperábamos no fue tan fácil de resolver ya que en cuanto al lenguaje de programación que escogimos (c++) conocemos algunas funciones pero no todas en su totalidad, también porque no conocíamos en su totalidad como se jugaba el ajedrez es por ellos que tuvimos que hacer una investigación algo profunda para conocer el juego de ajedrez y cómo podríamos llevarlo a la programación para calcular los movimientos de dos piezas del tablero.

En el trabajo que encontramos tiene solo de similitud el haber usado la función de bracktracking y el utilizar a la reina para calcular los diferentes caminos que pueden tener , solo que en aquel trabajo utilizan mas de una reina y en nuestro trabajo utilizamos solo las dos reinas que viene en el tablero. Tal vez se pueda mejor el código y podamos encontrar alguna otra función que nos pueda ayudar a resolver el problema pero de igual manera tenemos que hacer una investigación.
Al final después de una investigación amplia encontramos una función el cual nos ayudó para resolver el planteamiento del problema y llegar a un resultado exitoso.

# CONCLUSIÓN

Este proyecto nos pareció interesante ya que nos permitió ampliar una investigación y de un juego tan simple poder llevar algún problema dentro de la programación y a sí mismo poder conocer más funciones no vistas con anterioridad.

# REFERENCIAS BIBLIOGRÁFICAS


- [1] Editorial Etecé 2013-2024 enciclopedia concepto [online] https://concepto.de/ajedrez/

- [2] sw hosting 2024 [online] https://www.swhosting.com/es/comunidad/manual/en-que-consiste-el-algoritmo-de-backtracking-y-como-aplicarlo-en-c
- [3] David Cabrera 11 abril 2023 MEDIUM [online] https://medium.com/@davidcabreraygarcia/la-recursividad-de-backtracking-3e6739609b08