

**Materia:** Programación y Estructura de Datos (COM104)

**Profesor:** Francisco Javier Aguilar Juárez

**Fecha de entrega: 16 de noviembre de 2022**

**Ciclo:** 1228

**Nombre del proyecto:**

**Chess Simulator**

Miembros del Equipo		
ID	Nombre	Carrera
0243036	Diego Arenas Trevilla	IIDC
0244643	Sara Rocío Miranda Mateos	IIDC

## Chess Simulator

### Análisis *Descripción detallada del problema a resolver*

Nuestra aplicación es una aplicación lúdica que tiene como propósito simular una partida de la vida real de un juego de ajedrez de dos personas. A diferencia de un juego normal, este simulador tendrá las reglas incluidas, es decir que no te dejará mover una pieza si es que esta pieza no debe moverse de esa manera, por ejemplo, un peón no puede moverse más de una casilla.

Esta simulación de ajedrez nos permite evitar cualquier trampa ya que solo le da una oportunidad de mover una pieza a cada jugador en su respectivo turno. Además, en caso de estar aprendiendo o enseñando a jugar ajedrez, es muy útil poder regresar movimientos por lo cual este programa nos permitirá regresar los movimientos que hemos hecho, esto es de gran utilidad ya que de esta manera puedes conocer los diferentes caminos, técnicas y jugadas para adquirir práctica y mejorar en este deporte.

### Diseño *Descripción de las Estructuras de Datos que se emplearán y la explicación de cómo estas estructuras resuelven el problema*

Para la elaboración del proyecto decidimos usar cómo estructura de datos una pila ligada.

Una pila es una estructura de datos lineal, de entradas ordenadas, sus entradas son del mismo tipo de dato y el ingreso de los datos es de tipo LIFO (Last In, First Out). Una pila ligada tiene la ventaja de que la información no se guarda en localidades consecutivas de memoria, cada elemento de una pila ligada está conformada por dos partes, la parte de la información y el enlace con el elemento anterior. Con esta pila se pueden realizar operaciones, cómo:

- ❖ Insertar (push): Se inserta un dato al final de la pila.
- ❖ Extraer (pop): Se elimina el último dato de la pila.
- ❖ Consultar (peek): Muestra el último dato de la pila.

En nuestro caso la pila ligada la usamos para poder guardar los movimientos que se hacen en el tablero, es necesario usar una pila ya que si estamos en el movimiento 9 y queremos regresar al 7 primero tendríamos que ver el último movimiento hecho, extraerlo, acomodar el tablero y repetir hasta llegar al movimiento 7.

### Analysis

This is a ludic app that has as a purpose to simulate a real-life chess game for two people. The difference between a normal chess game and this simulator is that this program has the rules

integrated, in other words it has the validations in order to respect each piece's movements, for example, a pawn cannot move more than one square.

This chess simulator avoids any type of cheating because the program will just let us move a piece if it's our turn. Furthermore, this app can be useful in order to teach and learn because the simulator will let us reverse our moves. This is really handy due to the facility of knowing the different ruts, techniques and moves we can do in order to get practice and be better at this sport.

## Design

For this project elaboration we decided to implement the linked stack as a data structure. A stack is a linear data structure of ordered inputs, its inputs must be the same data type and it is structured as LIFO (Last In, First Out). A linked stack has as an advantage that its information isn't storage in different consecutive memory localities, the information is linked with the previous data. With this stack we can do operations as:

- ❖ Insert (push): We insert a data at the end of the stack
- ❖ Extract (pop): We delete the last data in the stack
- ❖ Consult (peek): It shows the last data in the stack

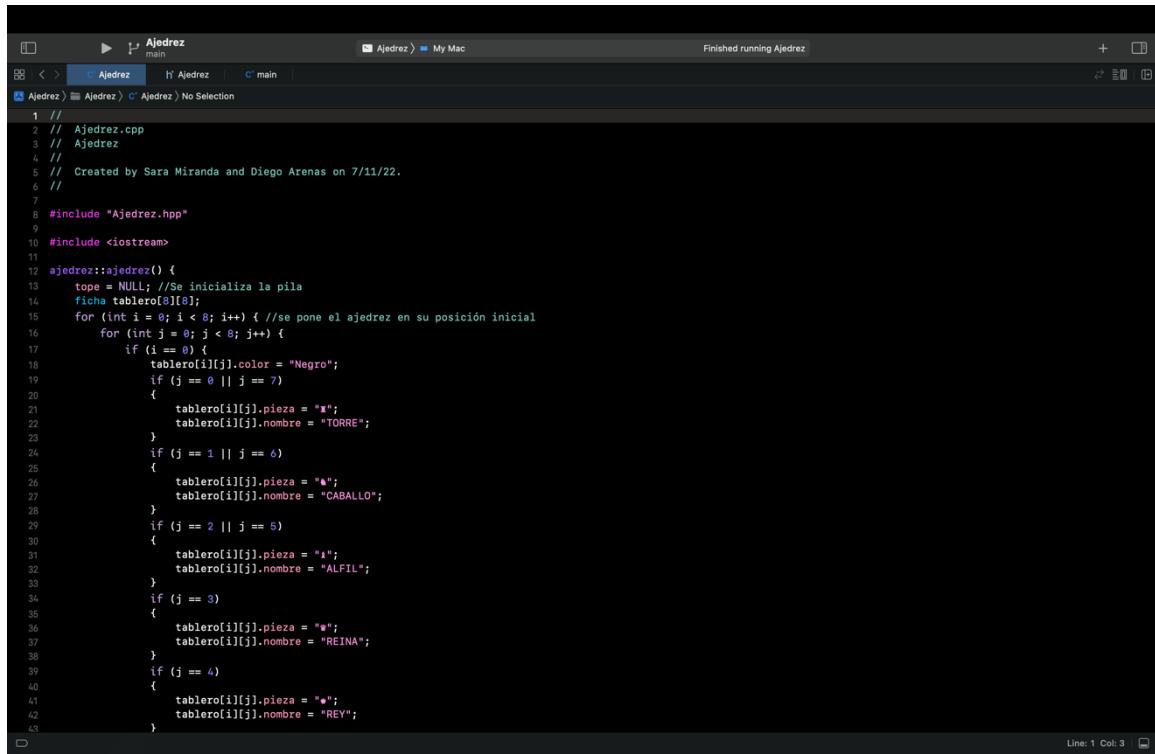
In our app we use a linked stack in order to save the moves the user had made on the board. It is necessary to use a stack because if we are in move number 9 and we want to go back to move number 7, first we should watch the last move done, extract it, order the board and repeat it until we get to the move we wanted to see.

## Implementación. Código completo de la aplicación / Implementation. Complete application code

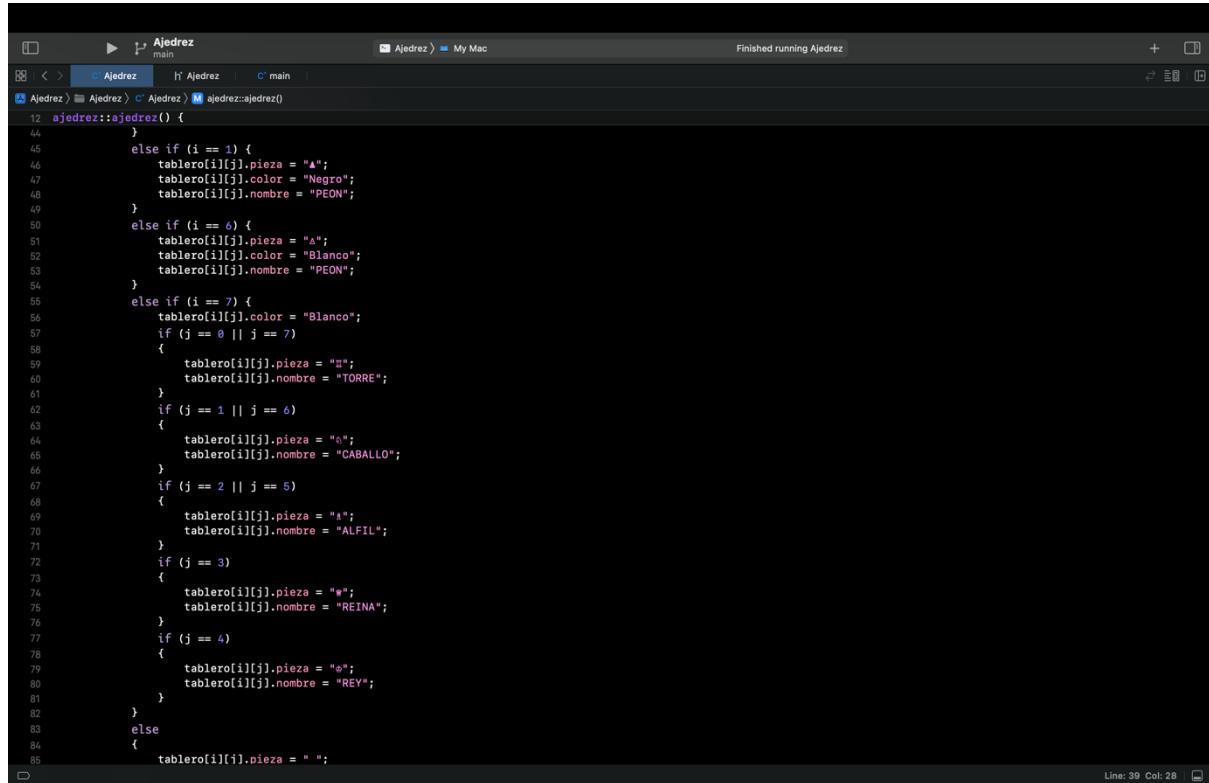
### Header Ajedrez.h

```
7 #ifndef Ajedrez_hpp
8 #define Ajedrez_hpp
9
10
11 #include <stdio.h>
12 #include <iostream>
13
14 using namespace std;
15
16 struct ficha {
17     string pieza; //Nombre de la pieza
18     string color; //Color de la pieza --y si ponemos color con una bandera
19     string nombre;
20 };
21
22 struct tabl {
23     ficha tablero[8][8] = {};
24     tabl* ante; //para guardar los movimientos en una pila
25 };
26
27 class ajedrez {
28 public:
29     ajedrez();
30     int Extraer(); //Regresar movimientos
31     void Mostrar_tablero(tabl); //Muestra la matriz de fichas
32     void Jugar(string); //Juega
33     tabl Consultar(); //Devuelve el tablero actual
34     //Hara falta una funcion para comer pieza?
35 private:
36     void Insertar(ficha t[8][8]);
37     tabl* tope, * nuevo;
38 };
39 #endif /* Ajedrez_hpp */
40 |
```

## Ajedrez.cpp



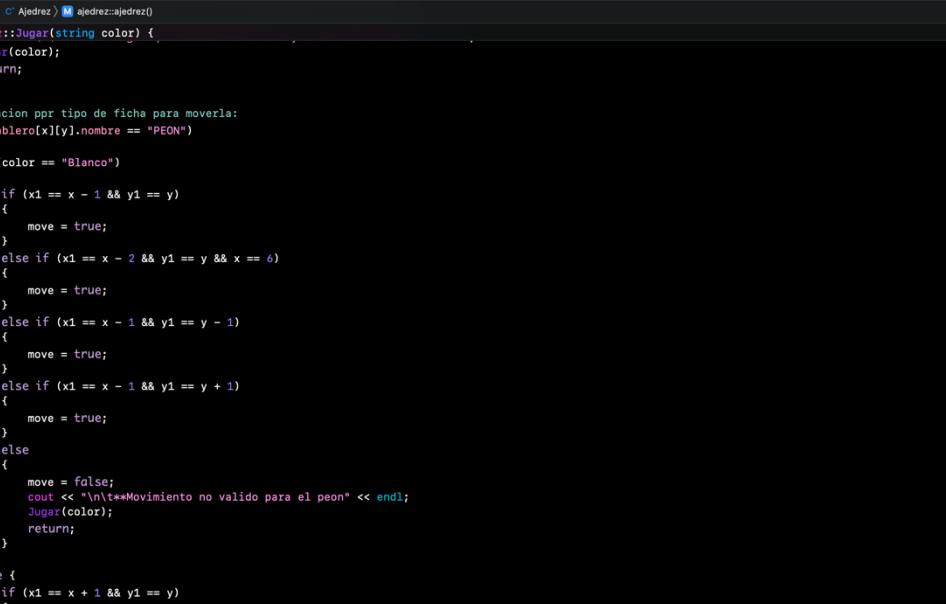
```
1 // 
2 //  Ajedrez.cpp
3 //  Ajedrez
4 //
5 //  Created by Sara Miranda and Diego Arenas on 7/11/22.
6 //
7
8 #include "Ajedrez.hpp"
9
10 #include <iostream>
11
12 ajedrez::ajedrez() {
13     tope = NULL; //Se inicializa la pila
14     ficha tablero[8][8];
15     for (int i = 0; i < 8; i++) { //se pone el ajedrez en su posición inicial
16         for (int j = 0; j < 8; j++) {
17             if (i == 0) {
18                 tablero[i][j].color = "Negro";
19                 if (j == 0 || j == 7)
20                 {
21                     tablero[i][j].pieza = "x";
22                     tablero[i][j].nombre = "TORRE";
23                 }
24                 if (j == 1 || j == 6)
25                 {
26                     tablero[i][j].pieza = "▲";
27                     tablero[i][j].nombre = "CABALLO";
28                 }
29                 if (j == 2 || j == 5)
30                 {
31                     tablero[i][j].pieza = "■";
32                     tablero[i][j].nombre = "ALFIL";
33                 }
34                 if (j == 3)
35                 {
36                     tablero[i][j].pieza = "•";
37                     tablero[i][j].nombre = "REINA";
38                 }
39                 if (j == 4)
40                 {
41                     tablero[i][j].pieza = "●";
42                     tablero[i][j].nombre = "REY";
43                 }
44             }
45             else if (i == 1) {
46                 tablero[i][j].pieza = "▲";
47                 tablero[i][j].color = "Negro";
48                 tablero[i][j].nombre = "PEON";
49             }
50             else if (i == 6) {
51                 tablero[i][j].pieza = "▲";
52                 tablero[i][j].color = "Blanco";
53                 tablero[i][j].nombre = "PEON";
54             }
55             else if (i == 7) {
56                 tablero[i][j].color = "Blanco";
57                 if (j == 0 || j == 7)
58                 {
59                     tablero[i][j].pieza = "■";
60                     tablero[i][j].nombre = "TORRE";
61                 }
62                 if (j == 1 || j == 6)
63                 {
64                     tablero[i][j].pieza = "△";
65                     tablero[i][j].nombre = "CABALLO";
66                 }
67                 if (j == 2 || j == 5)
68                 {
69                     tablero[i][j].pieza = "■";
70                     tablero[i][j].nombre = "ALFIL";
71                 }
72                 if (j == 3)
73                 {
74                     tablero[i][j].pieza = "•";
75                     tablero[i][j].nombre = "REINA";
76                 }
77                 if (j == 4)
78                 {
79                     tablero[i][j].pieza = "●";
80                     tablero[i][j].nombre = "REY";
81                 }
82             }
83             else
84             {
85                 tablero[i][j].pieza = " ";
86             }
87         }
88     }
89 }
```



```
12 ajedrez::ajedrez() {
13
14     }
15     else if (i == 1) {
16         tablero[i][j].pieza = "▲";
17         tablero[i][j].color = "Negro";
18         tablero[i][j].nombre = "PEON";
19     }
20     else if (i == 6) {
21         tablero[i][j].pieza = "▲";
22         tablero[i][j].color = "Blanco";
23         tablero[i][j].nombre = "PEON";
24     }
25     else if (i == 7) {
26         tablero[i][j].color = "Blanco";
27         if (j == 0 || j == 7)
28         {
29             tablero[i][j].pieza = "■";
30             tablero[i][j].nombre = "TORRE";
31         }
32         if (j == 1 || j == 6)
33         {
34             tablero[i][j].pieza = "△";
35             tablero[i][j].nombre = "CABALLO";
36         }
37         if (j == 2 || j == 5)
38         {
39             tablero[i][j].pieza = "■";
40             tablero[i][j].nombre = "ALFIL";
41         }
42         if (j == 3)
43         {
44             tablero[i][j].pieza = "•";
45             tablero[i][j].nombre = "REINA";
46         }
47         if (j == 4)
48         {
49             tablero[i][j].pieza = "●";
50             tablero[i][j].nombre = "REY";
51         }
52     }
53     else
54     {
55         tablero[i][j].pieza = " ";
56     }
57 }
```

```
□ ▶ Ajedrez main Ajedrez Ajedrez My Mac Finished running Ajedrez + □
Ajedrez Ajedrez Ajedrez ajedrez::ajedrez()
12 ajedrez::ajedrez() {
13     }
14 }
15 Insertar(tablero); //Inserta el tablero inicial (el que se acaba de crear) en la pila
16 }
17
18 void ajedrez::Insertar(ficha t[8][8]) { //Inserta en la pila la matriz que se le pasa como parámetro
19     nuevo = new tabl;
20     for (int i = 0; i < 8; i++) { //agregando datos
21         for (int j = 0; j < 8; j++)
22             nuevo->tablero[i][j] = t[i][j];
23     }
24     nuevo->ante = tope;
25     tope = nuevo;
26 }
27
28 int ajedrez::Extraer() { //Extrae el ultimo movimiento de la pila
29     if (tope->ante != NULL) { //Si no es el tablero inicial
30         tabl aux = tope;
31         tope = tope->ante;
32         delete aux;
33         return 1;
34     }
35     else {
36         return -1;
37     }
38 }
39
40 tabl ajedrez::Consultar() { //Regresa el tablero que esta en el tope
41     tabl aux;
42     if (tope != NULL) {
43         aux = *tope;
44         return aux;
45     }
46     else{
47         cout << "\n\t**No hay movimientos anteriores" << endl;
48     }
49 }
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
Line: 39 Col: 28
```

```
□ ▶ Ajedrez main Ajedrez Ajedrez My Mac Finished running Ajedrez + □
Ajedrez Ajedrez Ajedrez ajedrez::ajedrez()
127 void ajedrez::Mostrar_tablero(tabl t) {
128
129     cout << "||||||| JUGADOR 2 (NEGROS)" << endl;
130     cout << "||||| 1 2 3 4 5 6 7 8 " << endl;
131     for (int i = 0; i < 8; i++) {
132         cout << "\t\t-----" << endl;
133         cout << "\t\t| ";
134         cout << "\t\t| -----";
135         for (int j = 0; j < 8; j++)
136             cout << " | " << t.tablero[i][j].pieza << " ";
137         cout << " | " << i + 1 << endl;
138     }
139     cout << "\t\t-----" << endl;
140     cout << "\n||||| 1 2 3 4 5 6 7 8 " << endl;
141     cout << "\n||||| JUGADOR 1 (BLANCOS)" << endl;
142 }
143
144 void ajedrez::Jugar(string color) {
145     int x, y, x1, y1;
146     bool move = false;
147     tabl t = Consultar();
148     Mostrar_tablero(t);
149     cout << "Ingrese la posicion de la pieza que desea mover" << endl;
150     cout << "Ingrese la fila: "; cin >> x;
151     cout << "Ingrese la columna: "; cin >> y; //Pide las coordenadas de la ficha que quiere mover
152     x--; y--;
153     if (t.tablero[x][y].color != color)
154     {
155         cout << "\n\t**Casilla seleccionada incorrecta" << endl;
156         Jugar(color);
157         return;
158     }
159
160     cout << "\nHas seleccionado " << t.tablero[x][y].nombre << " en la posicion " << x+1 << "," << y+1 << endl;
161
162     cout << "\nIngrese la posicion a la que desea mover la pieza" << endl;
163     cout << "Ingrese la fila: "; cin >> x1;
164     cout << "Ingrese la columna: "; cin >> y1; //Pide las coordenadas a las que quiere mover la ficha
165     x1--; y1--;
166
167     if (t.tablero[x1][y1].color == color) { //Validacion para mover ficha
168         move = false;
169         cout << "\n\t**En el lugar que deseas moverte hay una ficha aliada" << endl;
Line: 39 Col: 28
```



A screenshot of a terminal window titled "Ajedrez" running on "My Mac". The window shows the following code:

```
144 void ajedrez::Jugar(string color) {
145     Jugar(color);
146     return;
147 }
148 //Validacion ppr tipo de ficha para moverla:
149 if (t.tablero[x][y].nombre == "PEON")
150 {
151     if (color == "Blanco")
152     {
153         if (x1 == x - 1 && y1 == y)
154         {
155             move = true;
156         }
157         else if (x1 == x - 2 && y1 == y && x == 6)
158         {
159             move = true;
160         }
161         else if (x1 == x - 1 && y1 == y - 1)
162         {
163             move = true;
164         }
165         else if (x1 == x - 1 && y1 == y + 1)
166         {
167             move = true;
168         }
169         else
170         {
171             move = false;
172             cout << "\n\t*Movimiento no valido para el peon" << endl;
173             Jugar(color);
174             return;
175         }
176     }
177     else
178     {
179         if (x1 == x + 1 && y1 == y)
180         {
181             move = true;
182         }
183         else if (x1 == x + 2 && y1 == y && x == 1)
184         {
185             move = true;
186         }
187         else
188         {
189             move = false;
190             cout << "\n\t*Movimiento no valido para el peon" << endl;
191             Jugar(color);
192             return;
193         }
194     }
195 }
```

A terminal window titled "Ajedrez" is shown, indicating the program has finished running. The command used was "ajedrez:ajedrez". The code itself is a C++ program that handles chess moves, specifically pawn moves. It includes logic for validating moves, handling knight moves, and managing the board state (represented by a 2D array). The code uses standard C++ syntax with loops, conditionals, and function calls.

```
144 void ajedrez::Jugar(string color) {
145     move = true;
146 }
147 else if (x1 == x + 1 && y1 == y - 1)
148 {
149     move = true;
150 }
151 else if (x1 == x + 1 && y1 == y + 1)
152 {
153     move = true;
154 }
155 else
156 {
157     move = false;
158     cout << "\n\t**Movimiento no valido para el peon" << endl;
159     Jugar(color);
160     return;
161 }
162 }
163 move = true;
164 }
165
166 if (t.tablero[x][y].nombre == "TORRE")
167 {
168     move = false;
169     if (x1 == x && y1 != y) { //Si se mueve horizontal
170         if (x1 < x) { //Si se mueve hacia la izquierda
171             for (int i = x - 1; i > x1; i--) //Recorre las posiciones entre la posicion inicial y la final
172                 if (t.tablero[i][y].nombre != " ") { //Si encuentra una ficha en el camino
173                     if (t.tablero[i][y].color == color) { //Si la ficha es del color
174                         x1 = i - 1; //La posicion final es la anterior a la ficha
175                         break;
176                     }
177                     else { //Si la ficha es del otro color
178                         t.tablero[i][y] = {};
179                         x1 = i; //La posicion final es la ficha
180                         break;
181                     }
182                 }
183             }
184         }
185     }
186 }
```

```
□ ▶ Ajedrez main □ Ajedrez □ Ajedrez > C main | Finished running Ajedrez + □
Ajedrez Ajedrez > Ajedrez > ajedrez::ajedrez()
144 void ajedrez::Jugar(string color) {
250     else { //Si se mueve hacia la derecha
251         for (int i = x + 1; i < x1; i++) { //Recorre las posiciones entre la posicion inicial y la final
252             if (t.tablero[i][y].pieza != " ") { //Si encuentra una ficha en el camino
253                 if (t.tablero[i][y].color == color) { //Si la ficha es del color
254                     x1 = i + 1; //La posicion final es la siguiente a la ficha
255                     break;
256                 }
257                 else { //Si la ficha es negra
258                     t.tablero[i][y] = {};
259                     x1 = i; //La posicion final es la ficha
260                     break;
261                 }
262             }
263         }
264     }
265     move = true;
266 }
else if (y1 == y && x1 != x) { //Si se mueve vertical
267     if (y1 < y) { //Si se mueve hacia abajo
268         for (int i = y - 1; i > y1; i--) { //Recorre las posiciones entre la posicion inicial y la final
269             if (t.tablero[x][i].pieza != " ") { //Si encuentra una ficha en el camino
270                 if (t.tablero[x][i].color == color) { //Si la ficha del color
271                     y1 = i - 1; //La posicion final es la anterior a la ficha
272                     break;
273                 }
274                 else { //Si la ficha es negra
275                     t.tablero[x][i] = {};
276                     y1 = i; //La posicion final es la ficha
277                     break;
278                 }
279             }
280         }
281     }
282 }
else { //Si se mueve hacia arriba
283     for (int i = y + 1; i < y1; i++) { //Recorre las posiciones entre la posicion inicial y la final
284         if (t.tablero[x][i].pieza != " ") { //Si encuentra una ficha en el camino
285             if (t.tablero[x][i].color == color) { //Si la ficha es del color
286                 x1 = i + 1; //La posicion final es la siguiente a la ficha
287                 break;
288             }
289         }
290     }
291 }
else { //Si se mueve hacia la izquierda
292     for (int i = x - 1; i > x1; i--) { //Recorre las posiciones entre la posicion inicial y la final
293         if (t.tablero[i][y].pieza != " ") { //Si encuentra una ficha en el camino
294             if (t.tablero[i][y].color == color) { //Si la ficha es del color
295                 x1 = i; //La posicion final es la anterior a la ficha
296                 break;
297             }
298         }
299     }
300     move = true;
301 }
302 if (t.tablero[x][y].nombre == "ALFIL")
{
    move = false;
    if (x1 != x && y1 != y) { //Si se mueve diagonal
304        if (x1 < x && y1 < y) { //Si se mueve hacia abajo a la izquierda
305            for (int i = x - 1, j = y - 1; i > x1 && j > y1; i--, j--) { //Recorre las posiciones entre la posicion inicial y la final
306                if (t.tablero[i][j].pieza != " ") { //Si encuentra una ficha en el camino
307                    if (t.tablero[i][j].color == color) { //Si la ficha es del color
308                        x1 = i - 1; //La posicion final es la anterior a la ficha
309                        y1 = j - 1; //La posicion final es la anterior a la ficha
310                        break;
311                    }
312                    else { //Si la ficha es negra
313                        t.tablero[i][j] = {};
314                        x1 = i; //La posicion final es la ficha
315                        y1 = j; //La posicion final es la ficha
316                        break;
317                    }
318                }
319            }
320        }
321    }
322    else if (x1 < x && y1 > y) { //Si se mueve hacia abajo a la derecha
323        for (int i = x - 1, j = y + 1; i > x1 && j < y1; i--, j++) { //Recorre las posiciones entre la posicion inicial y la final
324            if (t.tablero[i][j].pieza != " ") { //Si encuentra una ficha en el camino
325                if (t.tablero[i][j].color == color) { //Si la ficha es del color
326                    x1 = i - 1; //La posicion final es la anterior a la ficha
327                    y1 = j + 1; //La posicion final es la siguiente a la ficha
328                    break;
329                }
330            }
331        }
332    }
333 }
```

```
□ ▶ Ajedrez main □ Ajedrez □ Ajedrez > C main | Finished running Ajedrez + □
Ajedrez Ajedrez > Ajedrez > ajedrez::ajedrez()
144 void ajedrez::Jugar(string color) {
250     else { //Si se mueve hacia la derecha
251         for (int i = x + 1; i < x1; i++) { //Recorre las posiciones entre la posicion inicial y la final
252             if (t.tablero[i][y].pieza != " ") { //Si encuentra una ficha en el camino
253                 if (t.tablero[i][y].color == color) { //Si la ficha es del color
254                     x1 = i + 1; //La posicion final es la siguiente a la ficha
255                     break;
256                 }
257                 else { //Si la ficha es negra
258                     t.tablero[i][y] = {};
259                     x1 = i; //La posicion final es la ficha
260                     break;
261                 }
262             }
263         }
264     }
265     move = true;
266 }
else if (y1 == y && x1 != x) { //Si se mueve vertical
267     if (y1 < y) { //Si se mueve hacia abajo
268         for (int i = y - 1; i > y1; i--) { //Recorre las posiciones entre la posicion inicial y la final
269             if (t.tablero[x][i].pieza != " ") { //Si encuentra una ficha en el camino
270                 if (t.tablero[x][i].color == color) { //Si la ficha del color
271                     y1 = i - 1; //La posicion final es la anterior a la ficha
272                     break;
273                 }
274                 else { //Si la ficha es negra
275                     t.tablero[x][i] = {};
276                     y1 = i; //La posicion final es la ficha
277                     break;
278                 }
279             }
280         }
281     }
282 }
else { //Si se mueve hacia arriba
283     for (int i = y + 1; i < y1; i++) { //Recorre las posiciones entre la posicion inicial y la final
284         if (t.tablero[x][i].pieza != " ") { //Si encuentra una ficha en el camino
285             if (t.tablero[x][i].color == color) { //Si la ficha es del color
286                 x1 = i + 1; //La posicion final es la siguiente a la ficha
287                 break;
288             }
289         }
290     }
291 }
else { //Si se mueve hacia la izquierda
292     for (int i = x - 1; i > x1; i--) { //Recorre las posiciones entre la posicion inicial y la final
293         if (t.tablero[i][y].pieza != " ") { //Si encuentra una ficha en el camino
294             if (t.tablero[i][y].color == color) { //Si la ficha es del color
295                 x1 = i; //La posicion final es la anterior a la ficha
296                 break;
297             }
298         }
299     }
300     move = true;
301 }
302 if (t.tablero[x][y].nombre == "ALFIL")
{
    move = false;
    if (x1 != x && y1 != y) { //Si se mueve diagonal
304        if (x1 < x && y1 < y) { //Si se mueve hacia abajo a la izquierda
305            for (int i = x - 1, j = y - 1; i > x1 && j > y1; i--, j--) { //Recorre las posiciones entre la posicion inicial y la final
306                if (t.tablero[i][j].pieza != " ") { //Si encuentra una ficha en el camino
307                    if (t.tablero[i][j].color == color) { //Si la ficha es del color
308                        x1 = i - 1; //La posicion final es la anterior a la ficha
309                        y1 = j - 1; //La posicion final es la anterior a la ficha
310                        break;
311                    }
312                    else { //Si la ficha es negra
313                        t.tablero[i][j] = {};
314                        x1 = i; //La posicion final es la ficha
315                        y1 = j; //La posicion final es la ficha
316                        break;
317                    }
318                }
319            }
320        }
321    }
322    else if (x1 < x && y1 > y) { //Si se mueve hacia abajo a la derecha
323        for (int i = x - 1, j = y + 1; i > x1 && j < y1; i--, j++) { //Recorre las posiciones entre la posicion inicial y la final
324            if (t.tablero[i][j].pieza != " ") { //Si encuentra una ficha en el camino
325                if (t.tablero[i][j].color == color) { //Si la ficha es del color
326                    x1 = i - 1; //La posicion final es la anterior a la ficha
327                    y1 = j + 1; //La posicion final es la siguiente a la ficha
328                    break;
329                }
330            }
331        }
332    }
333 }
```

```
□ ▶ Ajedrez main Ajedrez | C main | Finished running Ajedrez + □
Ajedrez Ajedrez > Ajedrez > M ajedrez::ajedrez()
146 void ajedrez::Jugar(string color) {
329
330     else { //Si la ficha es negra
331         t.tablero[i][j] = {};
332         x1 = i; //La posicion final es la ficha
333         y1 = j; //La posicion final es la ficha
334         break;
335     }
336 }
337 }
338 }
339 else if (x1 > x && y1 < y) { //Si se mueve hacia arriba a la izquierda
340     for (int i = x + 1, j = y - 1; i < x1 && j > y1; i++, j--) { //Recorre las posiciones entre la posicion inicial y la final
341         if (t.tablero[i][j].pieza != " ") { //Si encuentra una ficha en el camino
342             if (t.tablero[i][j].color == color) { //Si la ficha es del color
343                 x1 = i + 1; //La posicion final es la siguiente a la ficha
344                 y1 = j - 1; //La posicion final es la anterior a la ficha
345                 break;
346             }
347             else { //Si la ficha es negra
348                 t.tablero[i][j] = {};
349                 x1 = i; //La posicion final es la ficha
350                 y1 = j; //La posicion final es la ficha
351                 break;
352             }
353         }
354     }
355 }
356 else { //Si se mueve hacia arriba a la derecha
357     for (int i = x + 1, j = y + 1; i < x1 && j < y1; i++, j++) { //Recorre las posiciones entre la posicion inicial y la final
358         if (t.tablero[i][j].pieza != " ") { //Si encuentra una ficha en el camino
359             if (t.tablero[i][j].color == color) { //Si la ficha es del color
360                 x1 = i + 1; //La posicion final es la siguiente a la ficha
361                 y1 = j + 1; //La posicion final es la siguiente a la ficha
362                 break;
363             }
364             else { //Si la ficha es negra
365                 t.tablero[i][j] = {};
366                 x1 = i; //La posicion final es la ficha
367                 y1 = j; //La posicion final es la ficha
368                 break;
369             }
370         }
371     }
372 }
373 move = true;
374 }
375 }
376 if (t.tablero[x][y].nombre == "REY")
{
    move = true;
    if (x == x1)
    {
        if (y1 > y + 1 || y1 < y - 1 || (t.tablero[x][y1].pieza != " " & t.tablero[x][y1].color == color))
            move = false;
    }
    else if (y == y1)
    {
        if (x1 > x + 1 || x1 < x - 1 || (t.tablero[x1][y].pieza != " " & t.tablero[x1][y].color == color))
            move = false;
    }
    else
        move = false;
}
391 if (t.tablero[x][y].nombre == "REINA")
{
    move = false;
    if (x1 == x && y1 != y) { //Si se mueve horizontal
        if (x1 < x) { //Si se mueve hacia la izquierda
            for (int i = x - 1; i > x1; i--) { //Recorre las posiciones entre la posicion inicial y la final
                if (t.tablero[i][y].pieza != " ") { //Si encuentra una ficha en el camino
                    if (t.tablero[i][y].color == color) { //Si la ficha es del color
                        x1 = i - 1; //La posicion final es la anterior a la ficha
                        break;
                    }
                    else { //Si la ficha es negra
                        t.tablero[i][y] = {};
                        x1 = i; //La posicion final es la ficha
                        break;
                    }
                }
            }
        }
    }
}
Line: 39 Col: 28
```

```
□ ▶ Ajedrez main Ajedrez | C main | Finished running Ajedrez + □
Ajedrez Ajedrez > Ajedrez > M ajedrez::ajedrez()
146 void ajedrez::Jugar(string color) {
370
371     }
372 }
373 move = true;
374 }
375 }
376 if (t.tablero[x][y].nombre == "REY")
{
    move = true;
    if (x == x1)
    {
        if (y1 > y + 1 || y1 < y - 1 || (t.tablero[x][y1].pieza != " " & t.tablero[x][y1].color == color))
            move = false;
    }
    else if (y == y1)
    {
        if (x1 > x + 1 || x1 < x - 1 || (t.tablero[x1][y].pieza != " " & t.tablero[x1][y].color == color))
            move = false;
    }
    else
        move = false;
}
391 if (t.tablero[x][y].nombre == "REINA")
{
    move = false;
    if (x1 == x && y1 != y) { //Si se mueve horizontal
        if (x1 < x) { //Si se mueve hacia la izquierda
            for (int i = x - 1; i > x1; i--) { //Recorre las posiciones entre la posicion inicial y la final
                if (t.tablero[i][y].pieza != " ") { //Si encuentra una ficha en el camino
                    if (t.tablero[i][y].color == color) { //Si la ficha es del color
                        x1 = i - 1; //La posicion final es la anterior a la ficha
                        break;
                    }
                    else { //Si la ficha es negra
                        t.tablero[i][y] = {};
                        x1 = i; //La posicion final es la ficha
                        break;
                    }
                }
            }
        }
    }
}
Line: 39 Col: 28
```

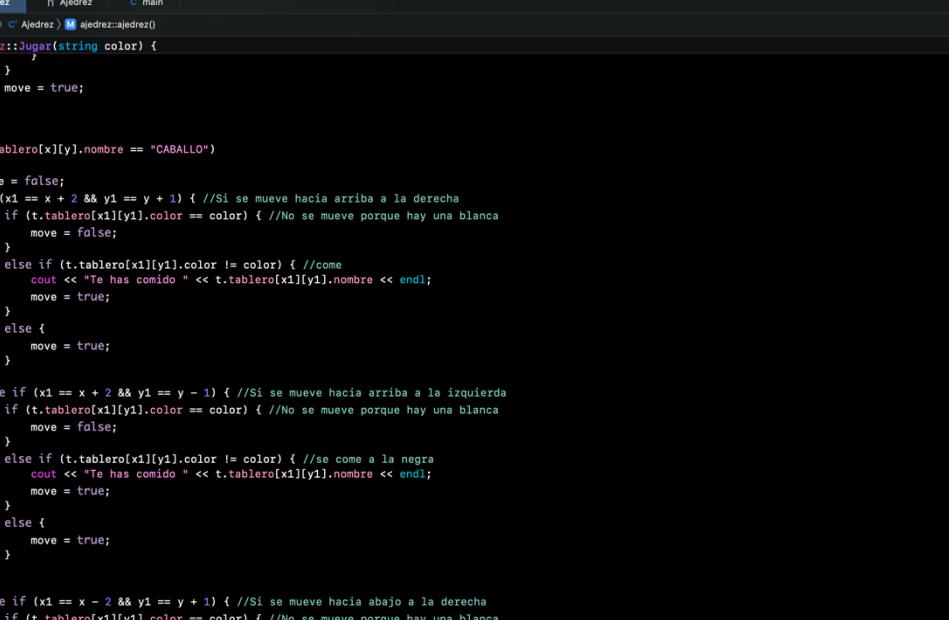
```
□ ▶ Ajedrez main Ajedrez My Mac Finished running Ajedrez + □
Ajedrez < Ajedrez > Ajedrez > C Ajedrez ajedrez::ajedrez()
144 void ajedrez::Jugar(string color) {
409     }
410     else { //Si se mueve hacia la derecha
411         for (int i = x + 1; i < x1; i++) { //Recorre las posiciones entre la posicion inicial y la final
412             if (t.tablero[i][y].pieza != " ") { //Si encuentra una ficha en el camino
413                 if (t.tablero[i][y].color == color) { //Si la ficha es del color
414                     x1 = i + 1; //La posicion final es la siguiente a la ficha
415                     break;
416                 }
417                 else { //Si la ficha es negra
418                     t.tablero[i][y] = {};
419                     x1 = i; //La posicion final es la ficha
420                     break;
421                 }
422             }
423         }
424     }
425     move = true;
426 }
427 else if (y1 == y && x1 != x) { //Si se mueve vertical
428     if (y1 < y) { //Si se mueve hacia abajo
429         for (int i = y - 1; i > y1; i--) { //Recorre las posiciones entre la posicion inicial y la final
430             if (t.tablero[x][i].pieza != " ") { //Si encuentra una ficha en el camino
431                 if (t.tablero[x][i].color == color) { //Si la ficha es del color
432                     y1 = i - 1; //La posicion final es la anterior a la ficha
433                     break;
434                 }
435                 else { //Si la ficha es negra
436                     t.tablero[x][i] = {};
437                     y1 = i; //La posicion final es la ficha
438                     break;
439                 }
440             }
441         }
442     }
443     else { //Si se mueve hacia arriba
444         for (int i = y + 1; i < y1; i++) { //Recorre las posiciones entre la posicion inicial y la final
445             if (t.tablero[x][i].pieza != " ") { //Si encuentra una ficha en el camino
446                 if (t.tablero[x][i].color == color) { //Si la ficha es del color
447                     y1 = i + 1; //La posicion final es la siguiente a la ficha
448                     break;
449                 }
450             }
451         }
452     }
453     move = true;
454 }
455 else { //Si se mueve diagonal
456     if (x1 != x && y1 != y) { //Si se mueve diagonal
457         if (x1 < x && y1 < y) { //Si se mueve hacia abajo a la izquierda
458             for (int i = x - 1, j = y - 1; i > x1 && j > y1; i--, j--) { //Recorre las posiciones entre la posicion inicial y la final
459                 if (t.tablero[i][j].pieza != " ") { //Si encuentra una ficha en el camino
460                     if (t.tablero[i][j].color == color) { //Si la ficha es blanca
461                         x1 = i - 1; //La posicion final es la anterior a la ficha
462                         y1 = j - 1; //La posicion final es la anterior a la ficha
463                         break;
464                     }
465                     else { //Si la ficha es negra
466                         t.tablero[i][j] = {};
467                         x1 = i; //La posicion final es la ficha
468                         y1 = j; //La posicion final es la ficha
469                         break;
470                     }
471                 }
472             }
473         }
474     }
475     else if (x1 < x && y1 > y) { //Si se mueve hacia abajo a la derecha
476         for (int i = x - 1, j = y + 1; i > x1 && j < y1; i--, j++) { //Recorre las posiciones entre la posicion inicial y la final
477             if (t.tablero[i][j].pieza != " ") { //Si encuentra una ficha en el camino
478                 if (t.tablero[i][j].color == color) { //Si la ficha es blanca
479                     x1 = i - 1; //La posicion final es la anterior a la ficha
480                     y1 = j + 1; //La posicion final es la siguiente a la ficha
481                     break;
482                 }
483                 else { //Si la ficha es negra
484                     t.tablero[i][j] = {};
485                     x1 = i; //La posicion final es la ficha
486                     y1 = j; //La posicion final es la ficha
487                     break;
488                 }
489             }
490         }
491     }
492 }
493 move = true;
494 }
```

```
□ ▶ Ajedrez main Ajedrez My Mac Finished running Ajedrez + □
Ajedrez < Ajedrez > Ajedrez > C Ajedrez ajedrez::ajedrez()
144 void ajedrez::Jugar(string color) {
409     }
410     else { //Si la ficha es negra
411         t.tablero[x][i] = {};
412         y1 = i; //La posicion final es la ficha
413         break;
414     }
415 }
416 else { //Si la ficha es negra
417     t.tablero[x][i] = {};
418     y1 = i; //La posicion final es la ficha
419     break;
420 }
421 }
422 move = true;
423 }
424 else if (x1 != x && y1 != y) { //Si se mueve diagonal
425     if (x1 < x && y1 < y) { //Si se mueve hacia abajo a la izquierda
426         for (int i = x - 1, j = y - 1; i > x1 && j > y1; i--, j--) { //Recorre las posiciones entre la posicion inicial y la final
427             if (t.tablero[i][j].pieza != " ") { //Si encuentra una ficha en el camino
428                 if (t.tablero[i][j].color == color) { //Si la ficha es blanca
429                     x1 = i - 1; //La posicion final es la anterior a la ficha
430                     y1 = j - 1; //La posicion final es la anterior a la ficha
431                     break;
432                 }
433                 else { //Si la ficha es negra
434                     t.tablero[i][j] = {};
435                     x1 = i; //La posicion final es la ficha
436                     y1 = j; //La posicion final es la ficha
437                     break;
438                 }
439             }
440         }
441     }
442     else if (x1 < x && y1 > y) { //Si se mueve hacia abajo a la derecha
443         for (int i = x - 1, j = y + 1; i > x1 && j < y1; i--, j++) { //Recorre las posiciones entre la posicion inicial y la final
444             if (t.tablero[i][j].pieza != " ") { //Si encuentra una ficha en el camino
445                 if (t.tablero[i][j].color == color) { //Si la ficha es blanca
446                     x1 = i - 1; //La posicion final es la anterior a la ficha
447                     y1 = j + 1; //La posicion final es la siguiente a la ficha
448                     break;
449                 }
450                 else { //Si la ficha es negra
451                     t.tablero[i][j] = {};
452                     x1 = i; //La posicion final es la ficha
453                     y1 = j; //La posicion final es la ficha
454                     break;
455                 }
456             }
457         }
458     }
459 }
460 move = true;
461 }
```

Ajedrez main

Ajedrez Ajedrez Ajedrez ajedrez::ajedrez()

```
144 void ajedrez::Jugar(string color) {
489     t.tablero[i][j] = {};
490     x1 = i; //La posicion final es la ficha
491     y1 = j; //La posicion final es la ficha
492     break;
493 }
494 }
495 }
496 }
497 else if (x1 > x && y1 < y) { //Si se mueve hacia arriba a la izquierda
498     for (int i = x + 1, j = y - 1; i < x1 && j > y1; i++, j--) { //Recorre las posiciones entre la posicion inicial y la final
499         if (t.tablero[i][j].pieza != " ") { //Si encuentra una ficha en el camino
500             if (t.tablero[i][j].color == color) { //Si la ficha es blanca
501                 x1 = i + 1; //La posicion final es la siguiente a la ficha
502                 y1 = j - 1; //La posicion final es la anterior a la ficha
503                 break;
504             }
505             else { //Si la ficha es negra
506                 t.tablero[i][j] = {};
507                 x1 = i; //La posicion final es la ficha
508                 y1 = j; //La posicion final es la ficha
509                 break;
510             }
511         }
512     }
513 }
514 else { //Si se mueve hacia arriba a la derecha
515     for (int i = x + 1, j = y + 1; i < x1 && j < y1; i++, j++) { //Recorre las posiciones entre la posicion inicial y la final
516         if (t.tablero[i][j].pieza != " ") { //Si encuentra una ficha en el camino
517             if (t.tablero[i][j].color == color) { //Si la ficha es blanca
518                 x1 = i + 1; //La posicion final es la siguiente a la ficha
519                 y1 = j + 1; //La posicion final es la siguiente a la ficha
520                 break;
521             }
522             else { //Si la ficha es negra
523                 t.tablero[i][j] = {};
524                 x1 = i; //La posicion final es la ficha
525                 y1 = j; //La posicion final es la ficha
526                 break;
527             }
528         }
529     }
530 }
```



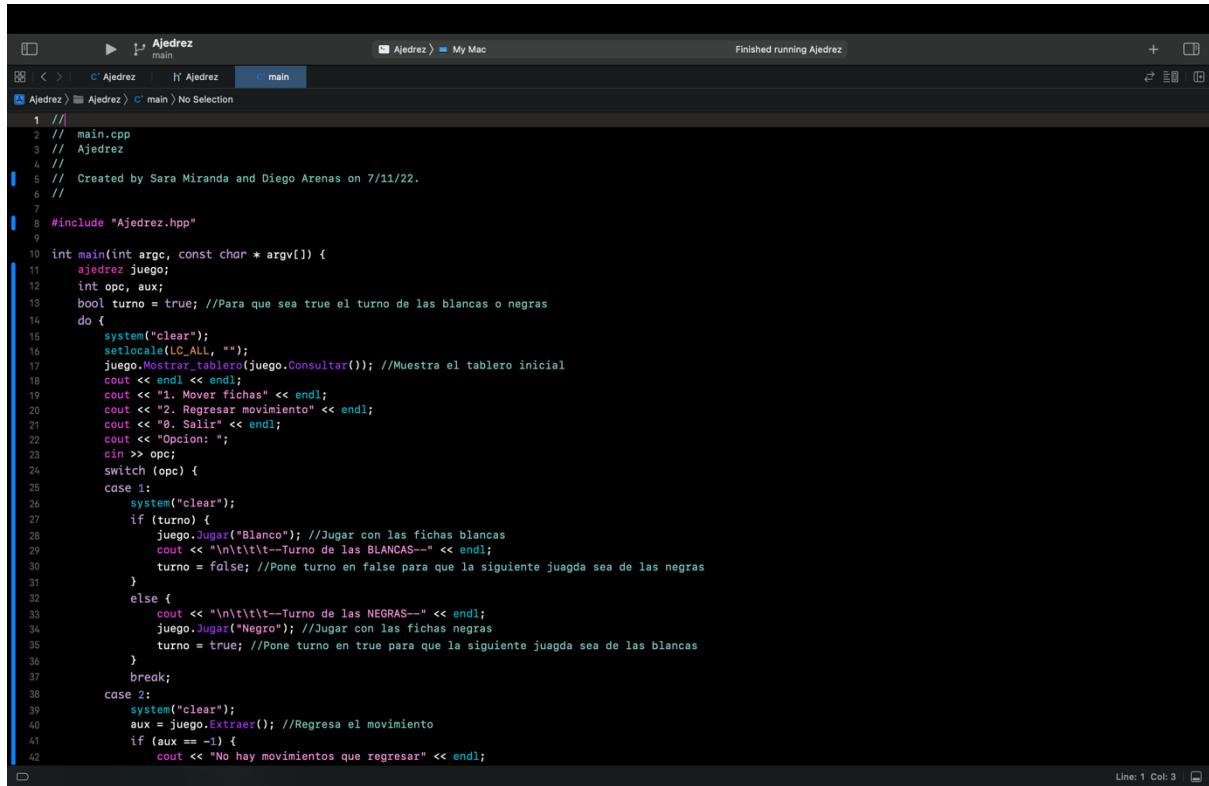
```
Finished running Ajedrez
Line: 39 Col: 28
```

```
144 void ajedrez::Jugar(string color) {
145     move = true;
146 }
147
148 if (t.tablero[x][y].nombre == "CABALLO") {
149     move = false;
150     if (x1 == x + 2 && y1 == y + 1) { //Si se mueve hacia arriba a la derecha
151         if (t.tablero[x1][y1].color == color) { //No se mueve porque hay una blanca
152             move = false;
153         }
154         else if (t.tablero[x1][y1].color != color) { //come
155             cout << "Te has comido " << t.tablero[x1][y1].nombre << endl;
156             move = true;
157         }
158         else {
159             move = true;
160         }
161     }
162     else if (x1 == x + 2 && y1 == y - 1) { //Si se mueve hacia arriba a la izquierda
163         if (t.tablero[x1][y1].color == color) { //No se mueve porque hay una blanca
164             move = false;
165         }
166         else if (t.tablero[x1][y1].color != color) { //se come a la negra
167             cout << "Te has comido " << t.tablero[x1][y1].nombre << endl;
168             move = true;
169         }
170         else {
171             move = true;
172         }
173     }
174     else if (x1 == x - 2 && y1 == y + 1) { //Si se mueve hacia abajo a la derecha
175         if (t.tablero[x1][y1].color == color) { //No se mueve porque hay una blanca
176             move = false;
177         }
178         else if (t.tablero[x1][y1].color != color) { //se come a la negra
179             cout << "Te has comido " << t.tablero[x1][y1].nombre << endl;
180         }
181     }
182 }
```

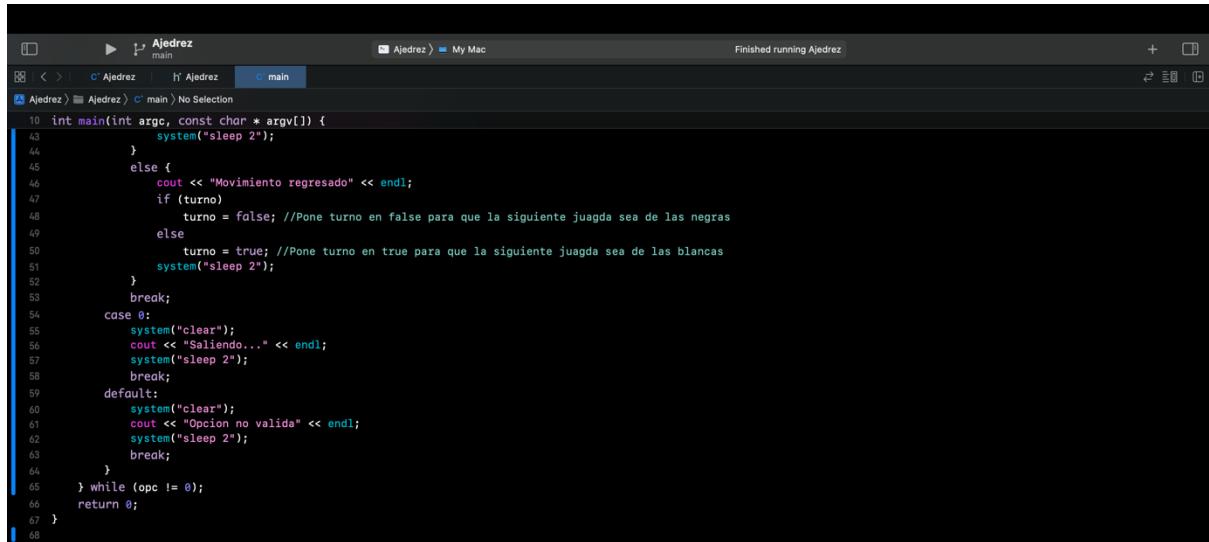
```
Finished running Ajedrez
Ajedrez > Ajedrez > ajedrez::ajedrez()
144 void ajedrez::Jugar(string color) {
569     move = true;
570 }
571 else {
572     move = true;
573 }
574 }
575 else if (x1 == x - 2 && y1 == y - 1) { //Si se mueve hacia abajo a la izquierda
576     if (t.tablero[x1][y1].color == color) { //No se mueve porque hay una blanca
577         move = false;
578     }
579     else if (t.tablero[x1][y1].color != color) { //se come a la negra
580         cout << "Te has comido " << t.tablero[x1][y1].nombre << endl;
581         move = true;
582     }
583     else {
584         move = true;
585     }
586 }
587 else if (x1 == x + 1 && y1 == y + 2) { //Si se mueve hacia arriba a la derecha
588     if (t.tablero[x1][y1].color == color) { //No se mueve porque hay una blanca
589         move = false;
590     }
591     else if (t.tablero[x1][y1].color != color) { //se come a la negra
592         cout << "Te has comido " << t.tablero[x1][y1].nombre << endl;
593         move = true;
594     }
595     else {
596         move = true;
597     }
598 }
599 else if (x1 == x + 1 && y1 == y - 2) { //Si se mueve hacia arriba a la izquierda
600     if (t.tablero[x1][y1].color == color) { //No se mueve porque hay una blanca
601         move = false;
602     }
603     else if (t.tablero[x1][y1].color != color) { //se come a la negra
604         cout << "Te has comido " << t.tablero[x1][y1].nombre << endl;
605         move = true;
606     }
607     else {
608         move = true;
609     }
610 }
611 else if (x1 == x - 1 && y1 == y + 2) { //Si se mueve hacia abajo a la derecha
612     if (t.tablero[x1][y1].color == color) { //No se mueve porque hay una blanca
613         move = false;
614     }
615     else if (t.tablero[x1][y1].color != color) { //se come a la negra
616         cout << "Te has comido " << t.tablero[x1][y1].nombre << endl;
617         move = true;
618     }
619     else {
620         move = true;
621     }
622 }
623 else if (x1 == x - 1 && y1 == y - 2) { //Si se mueve hacia abajo a la izquierda
624     if (t.tablero[x1][y1].color == color) { //No se mueve porque hay una blanca
625         move = false;
626     }
627     else if (t.tablero[x1][y1].color != color) { //se come a la negra
628         cout << "Te has comido " << t.tablero[x1][y1].nombre << endl;
629         move = true;
630     }
631     else {
632         move = true;
633     }
634 }
635 }
636 }
637 if (move == true)
{
    t.tablero[x1][y1] = t.tablero[x][y]; //mover ficha al lugar deseado
639    t.tablero[x][y] = {}; //Borrar lugar anterior de la ficha
640    t.tablero[x][y].pieza = " ";
641    Insertar(t.tablero); //Insertar tablero nuevo en la pila
642    Mostrar_tablero(Consultar()); //Mostrar ultimo tablero en la pila
643}
644 else
{
    cout << "La pieza " << t.tablero[x][y].nombre << " NO puede moverse de esa forma" << endl;
645}
646 Jugar(color);
647 return;
648 }
```

```
Finished running Ajedrez
Ajedrez > Ajedrez > ajedrez::ajedrez()
144 void ajedrez::Jugar(string color) {
569     move = true;
570 }
571 else {
572     move = true;
573 }
574 }
575 else if (x1 == x - 2 && y1 == y - 1) { //Si se mueve hacia abajo a la izquierda
576     if (t.tablero[x1][y1].color == color) { //No se mueve porque hay una blanca
577         move = false;
578     }
579     else if (t.tablero[x1][y1].color != color) { //se come a la negra
580         cout << "Te has comido " << t.tablero[x1][y1].nombre << endl;
581         move = true;
582     }
583     else {
584         move = true;
585     }
586 }
587 else if (x1 == x + 1 && y1 == y + 2) { //Si se mueve hacia arriba a la derecha
588     if (t.tablero[x1][y1].color == color) { //No se mueve porque hay una blanca
589         move = false;
590     }
591     else if (t.tablero[x1][y1].color != color) { //se come a la negra
592         cout << "Te has comido " << t.tablero[x1][y1].nombre << endl;
593         move = true;
594     }
595     else {
596         move = true;
597     }
598 }
599 else if (x1 == x + 1 && y1 == y - 2) { //Si se mueve hacia arriba a la izquierda
600     if (t.tablero[x1][y1].color == color) { //No se mueve porque hay una blanca
601         move = false;
602     }
603     else if (t.tablero[x1][y1].color != color) { //se come a la negra
604         cout << "Te has comido " << t.tablero[x1][y1].nombre << endl;
605         move = true;
606     }
607     else {
608         move = true;
609     }
610 }
611 else if (x1 == x - 1 && y1 == y + 2) { //Si se mueve hacia abajo a la derecha
612     if (t.tablero[x1][y1].color == color) { //No se mueve porque hay una blanca
613         move = false;
614     }
615     else if (t.tablero[x1][y1].color != color) { //se come a la negra
616         cout << "Te has comido " << t.tablero[x1][y1].nombre << endl;
617         move = true;
618     }
619     else {
620         move = true;
621     }
622 }
623 else if (x1 == x - 1 && y1 == y - 2) { //Si se mueve hacia abajo a la izquierda
624     if (t.tablero[x1][y1].color == color) { //No se mueve porque hay una blanca
625         move = false;
626     }
627     else if (t.tablero[x1][y1].color != color) { //se come a la negra
628         cout << "Te has comido " << t.tablero[x1][y1].nombre << endl;
629         move = true;
630     }
631     else {
632         move = true;
633     }
634 }
635 }
636 }
637 if (move == true)
{
    t.tablero[x1][y1] = t.tablero[x][y]; //mover ficha al lugar deseado
639    t.tablero[x][y] = {}; //Borrar lugar anterior de la ficha
640    t.tablero[x][y].pieza = " ";
641    Insertar(t.tablero); //Insertar tablero nuevo en la pila
642    Mostrar_tablero(Consultar()); //Mostrar ultimo tablero en la pila
643}
644 else
{
    cout << "La pieza " << t.tablero[x][y].nombre << " NO puede moverse de esa forma" << endl;
645}
646 Jugar(color);
647 return;
648 }
```

## Source.cpp



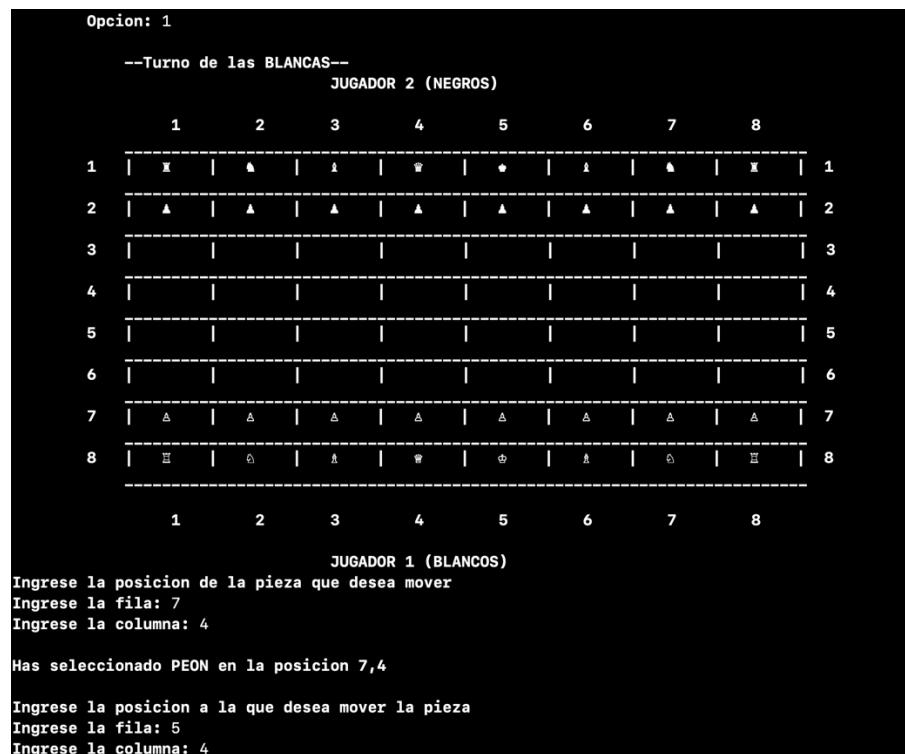
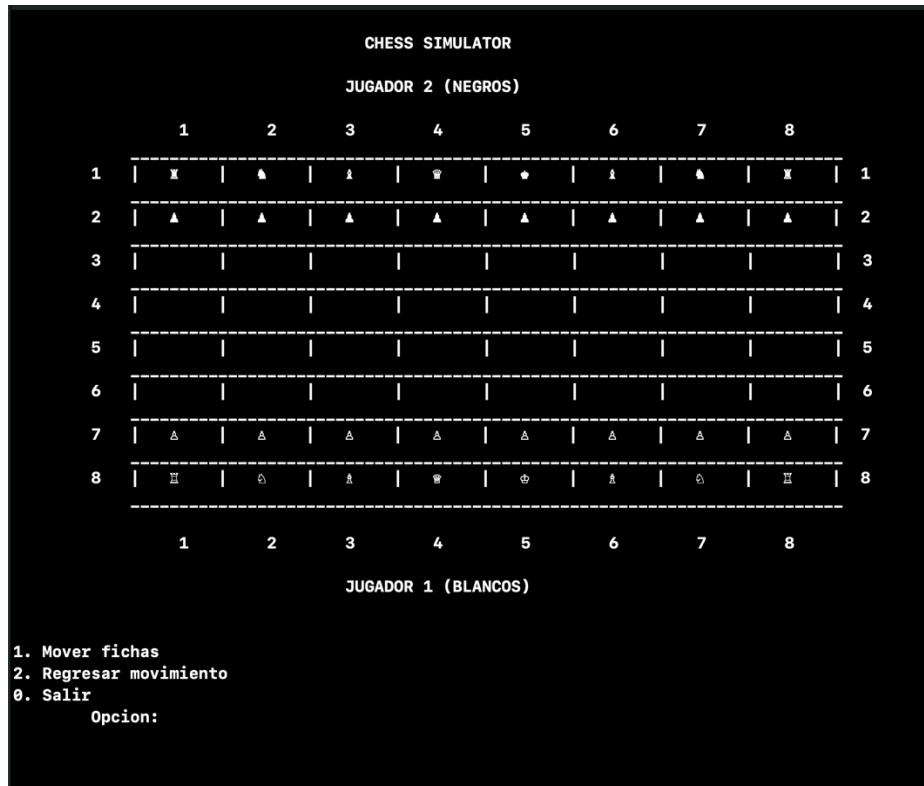
```
1 //| main.cpp
2 //| Ajedrez
3 //| Ajedrez
4 //| Created by Sara Miranda and Diego Arenas on 7/11/22.
5 //
6 #include "Ajedrez.hpp"
7
8 int main(int argc, const char * argv[]) {
9     ajedrez juego;
10    int opc, aux;
11    bool turno = true; //Para que sea true el turno de las blancas o negras
12    do {
13        system("clear");
14        setlocale(LC_ALL, "");
15        juego.Mostrar_tablero(); //Muestra el tablero inicial
16        cout << endl << endl;
17        cout << "1. Mover fichas" << endl;
18        cout << "2. Regresar movimiento" << endl;
19        cout << "0. Salir" << endl;
20        cout << "Opcion: ";
21        cin >> opc;
22        switch (opc) {
23            case 1:
24                system("clear");
25                if (turno) {
26                    juego.Jugar("Blanco"); //Jugar con las fichas blancas
27                    cout << "\n\t\t\t--Turno de las BLANCAS--" << endl;
28                    turno = false; //Pone turno en false para que la siguiente jugada sea de las negras
29                }
30                else {
31                    cout << "\n\t\t\t--Turno de las NEGRAS--" << endl;
32                    juego.Jugar("Negro"); //Jugar con las fichas negras
33                    turno = true; //Pone turno en true para que la siguiente jugada sea de las blancas
34                }
35            break;
36        case 2:
37            system("clear");
38            aux = juego.Extraer(); //Regresa el movimiento
39            if (aux == -1) {
40                cout << "No hay movimientos que regresar" << endl;
41            }
42        }
43    }
44    break;
45}
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
```



```
10 int main(int argc, const char * argv[]) {
11     system("sleep 2");
12     }
13     else {
14         cout << "Movimiento regresado" << endl;
15         if (turno)
16             turno = false; //Pone turno en false para que la siguiente jugada sea de las negras
17         else
18             turno = true; //Pone turno en true para que la siguiente jugada sea de las blancas
19         system("sleep 2");
20     }
21     break;
22 }
23
24 case 0:
25     system("clear");
26     cout << "Saliendo..." << endl;
27     system("sleep 2");
28     break;
29 default:
30     system("clear");
31     cout << "Opcion no valida" << endl;
32     system("sleep 2");
33     break;
34 }
35
36 } while (opc != 0);
37
38 return 0;
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
```

**Prueba** Imagen de algunas pantallas que ejemplifican la ejecución de la aplicación.

**Proof:** App Execution



CHESS SIMULATOR									
JUGADOR 2 (NEGROS)									
1	2	3	4	5	6	7	8		
1	x	▲	▲	*	◆	▲	▲	x	1
2	▲	▲	▲	▲	▲	▲	▲	▲	2
3									3
4									4
5				▲					5
6									6
7	▲	▲	▲		▲	▲	▲	▲	7
8	■	□	▲	*	◆	▲	□	■	8

1	2	3	4	5	6	7	8		
---	---	---	---	---	---	---	---	--	--

JUGADOR 1 (BLANCOS)

1. Mover fichas  
2. Regresar movimiento  
0. Salir  
Opcion:

Opcion: 1

JUGADOR 2 (NEGROS)									
1	2	3	4	5	6	7	8		
1	x	▲	▲	*	◆	▲	▲	x	1
2	▲	▲	▲	▲	▲	▲	▲	▲	2
3									3
4									4
5				▲					5
6									6
7	▲	▲	▲		▲	▲	▲	▲	7
8	■	□	▲	*	◆	▲	□	■	8

1	2	3	4	5	6	7	8		
---	---	---	---	---	---	---	---	--	--

JUGADOR 1 (BLANCOS)

Ingrese la posicion de la pieza que desea mover  
Ingrese la fila: 2  
Ingrese la columna: 5

Has seleccionado PEON en la posicion 2,5

Ingrese la posicion a la que desea mover la pieza  
Ingrese la fila: 4  
Ingrese la columna: 5

JUGADOR 2 (NEGROS)									
	1	2	3	4	5	6	7	8	
1	x	*	*	*	*	*	*	x	1
2	*	*	*	*	*	*	*	*	2
3									3
4					*				4
5					*				5
6									6
7	*	*	*	*	*	*	*	*	7
8	*	*	*	*	*	*	*	*	8

JUGADOR 1 (BLANCOS)

Has seleccionado REINA en la posicion 8,4

Ingrese la posicion a la que desea mover la pieza

Ingrese la fila: 6

Ingrese la columna: 4

JUGADOR 2 (NEGROS)

	1	2	3	4	5	6	7	8	
1	x	*	*	*	*	*	*	x	1
2	*	*	*	*	*	*	*	*	2
3									3
4					*				4
5					*				5
6				*					6
7	*	*	*	*	*	*	*	*	7
8	*	*	*	*	*	*	*	*	8

1      2      3      4      5      6      7      8

Opcion: 2  
Movimiento regresado

CHESS SIMULATOR

JUGADOR 2 (NEGROS)

	1	2	3	4	5	6	7	8	
1	X	◆	▲	●	*	▲	◆	X	1
2	▲	▲	▲	▲		▲	▲	▲	2
3									3
4					▲				4
5				▲					5
6									6
7	▲	▲	▲	▲	▲	▲	▲	▲	7
8	■	□	▲	●	●	□	■	■	8

1      2      3      4      5      6      7      8

JUGADOR 1 (BLANCOS)

Ingrese la posicion de la pieza que desea mover  
 Ingrese la fila: 1  
 Ingrese la columna: 2

Has seleccionado CABALLO en la posicion 1,2

Ingrese la posicion a la que desea mover la pieza  
 Ingrese la fila: 4  
 Ingrese la columna: 2  
 La pieza CABALLO NO puede moverse de esa forma

JUGADOR 2 (NEGROS)

	1	2	3	4	5	6	7	8	
1	X	◆	▲	●	*	▲	◆	X	1
2	▲	▲	▲	▲	▲	▲	▲	▲	2
3									3
4									4
5				▲					5
6									6
7	▲	▲	▲	▲	▲	▲	▲	▲	7
8	■	□	▲	●	●	□	■	■	8

1      2      3      4      5      6      7      8

0. Salir  
 Opcion: 0  
 Saliendo...  
 Program ended with exit code: 0

## Guía del usuario

**Objetivo:** Este manual tiene como finalidad dar a conocer a los pasos para utilizar el programa “Chess Simulator” para poder tener una experiencia amigable

### Requerimientos:

- Tener cualquier compilador de C++
- Idealmente se requiere un sistema operativo Linux u iOS, de lo contrario algunas funcionalidades de la terminal no serán mostradas de la manera más eficiente. A pesar de ello el programa funcionará

### Ingreso al sistema

- Opción 1: Con un IDE
  - o Abre el programa
  - o Click en “Run Program”
- Opción 2: Desde Terminal
  - o Abre la terminal de tu dispositivo
  - o Ingresa a la carpeta en donde se encuentre el programa, ejemplo  
`$ cd Desktop/Ajedrez`
  - o En caso de ser del sistema operativo Linux o iOS
    - Escribe los siguientes comandos  
`$ g++ *.cpp`  
`$ ./a.out`
  - o En caso de ser del sistema operativo Windows
    - Escribe el siguiente comando  
`$ g++ -Wall -std=c++14 Source.cpp -o main.exe`

### Opciones del sistema

*Seguir las instrucciones dentro del programa, en caso de no escribir la información correcta el programa te indicará que lo intentes de nuevo*

1. Mover pieza
  - 1) Ingrese la posición de la pieza que desea mover
  - 2) Ingrese la fila:
  - 3) Ingrese la columna:
  - 4) Ingrese la posición a la que desea mover la pieza
  - 5) Ingrese la fila:
  - 6) Ingrese la columna:
2. Regresar movimiento
3. Salir

# User's Guide

**Objective:** This guide has as a purpose to let the user know the steps to use the “Chess Simulator” program in order to have a friendly experience

## Requirements:

- Download any C++ compilator
- Ideally it requires a Linux or iOS operating system, on the contrary some terminal functionalities will not be available nor be shown in the most efficient way.  
Nevertheless, the program will work as it is supposed to.

## System Login

- Option 1: With a C++ IDE
  - o Open the program
  - o Click “Run Program”
- Option 2: From Terminal
  - o Open your device’s terminal
  - o Go into the folder which contains your program, for example  
`$ cd Desktop/Ajedrez`
  - o In the case it is a Linux or iOS operating system
    - Write the next commands  
`$ g++ *.cpp`  
`$ ./a.out`
  - o In the case it is a Windows operating system
    - Write the next commands  
`$ g++ -Wall -std=c++14 Source.cpp -o main.exe`

## System Options

*Follow up the program’s instructions. In case of not writing what is requested, the program will let you know*

1. Move piece
  - a. Write the piece’s position you’d like to move
  - b. Write the row:
  - c. Write the column:
  - d. Write the piece’s position you’d like to move to
  - e. Write the row:
  - f. Write the column:
2. Return movement
3. Exit