

# ESTRUCTURA DE DATOS Y ALGORITMOS I

## PROYECTO FINAL



### **“Tour del caballo”**

Presenta:

Leslie Janeth Quincosa Ramírez

Profesora:

Dra. Claudia Esteves Jaramillo

Licenciatura en Matemáticas

Departamento de Matemáticas de la Universidad de Guanajuato

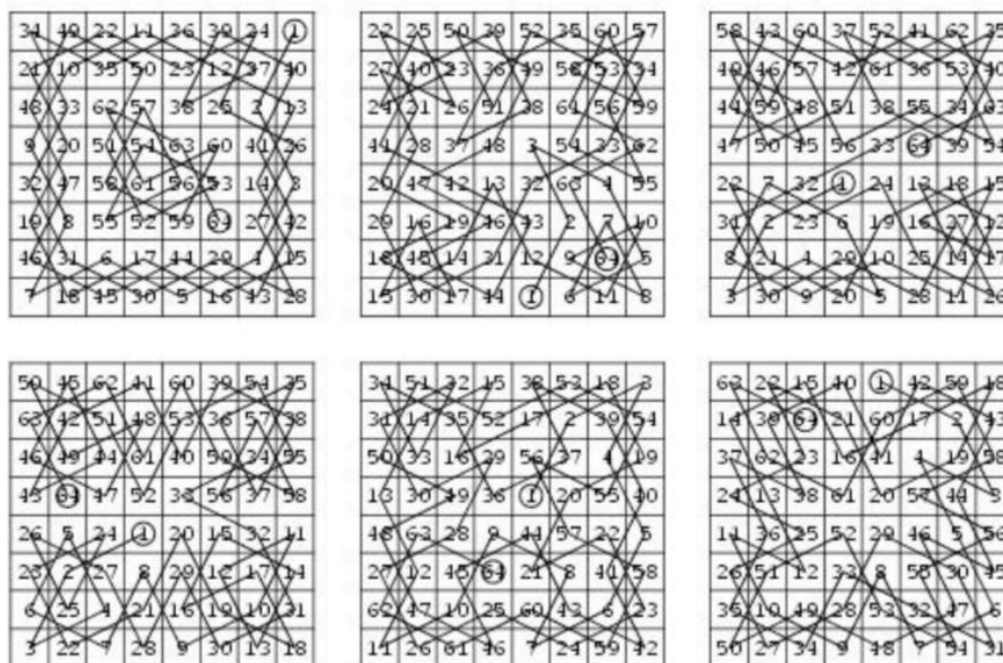
Segundo semestre, grupo B

Fecha: 07/Junio/2019

## Introducción:

El **tour del caballo** es un antiguo problema matemático en el que se pide que, teniendo una cuadrícula de  $n \times n$  casillas y un caballo de ajedrez colocado en una posición cualquiera (  $x, y$  ), el caballo pase por todas las casillas y una sola vez. Lo que resulta en  $n^2 - 1$  movimientos.

Algunos ejemplos del tour del caballo son los siguientes:



Muchos matemáticos han buscado una solución matemática a este problema, entre ellos Leonhard Euler.

Se han encontrado muchas soluciones a este problema y de hecho no se sabe con seguridad de cuántas maneras diferentes es posible solucionarlo.

Algunas variaciones de este problema han sido estudiadas por los matemáticos, tales como:

- Buscar soluciones cíclicas, en la cual se debe llegar a la misma casilla de la cual se partió.
- Tableros de diferente número de columnas o diferente número de filas.
- Juegos de dos jugadores basados en la idea.
- Problemas usando ligeras variaciones en la forma de moverse el caballo.

El tour del caballo es una forma del problema más general problema de la ruta Hamiltoniana en la teoría de grafos.

## **PROYECTO:**

El presente proyecto se basa en el algoritmo que recorre todas las casillas de un tablero de ajedrez de 8x8, se saben varias herramientas para la solución de este problema, sin embargo, en este caso utilicé las siguientes implementaciones:

- Estruct, template.
- class priority\_queue
- Stacks.

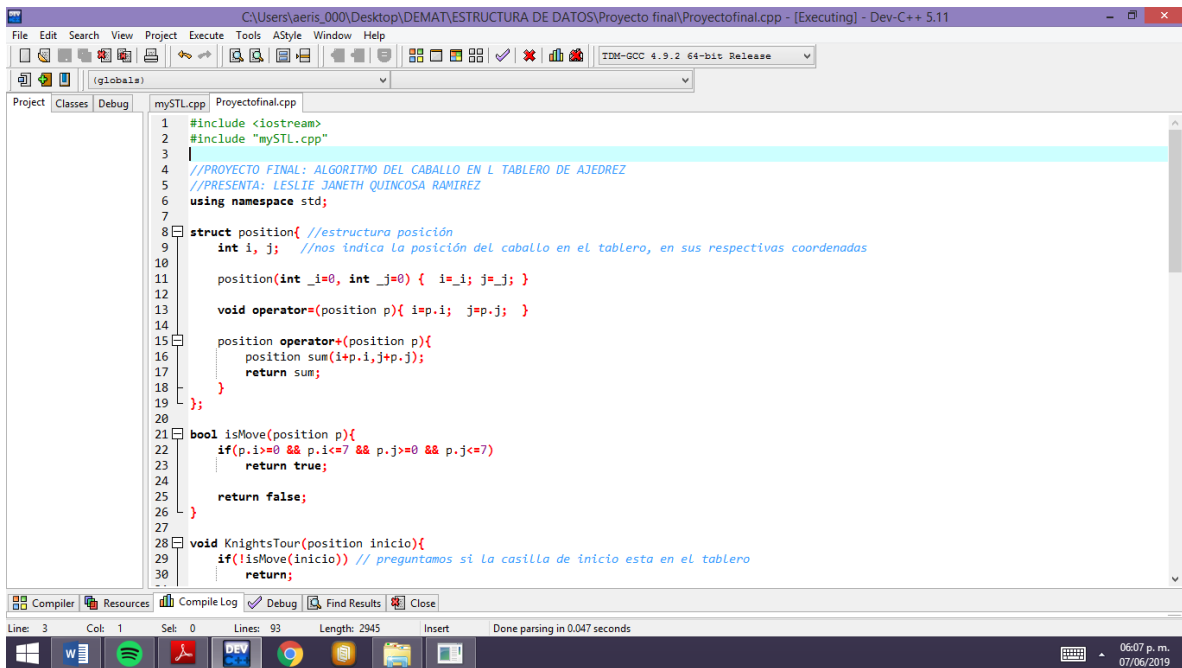
- Funciones tipo: bool, void.
- Iteraciones: for.
- Condicionales: if, while.
- Memoria dinámica.
- Matriz.

Aclaraciones del código: El tablero es una matriz que indica el movimiento del caballo, es decir, el 1 es la posición inicial del caballo y así sucesivamente hasta llegar al 64 donde termina el recorrido del caballo.

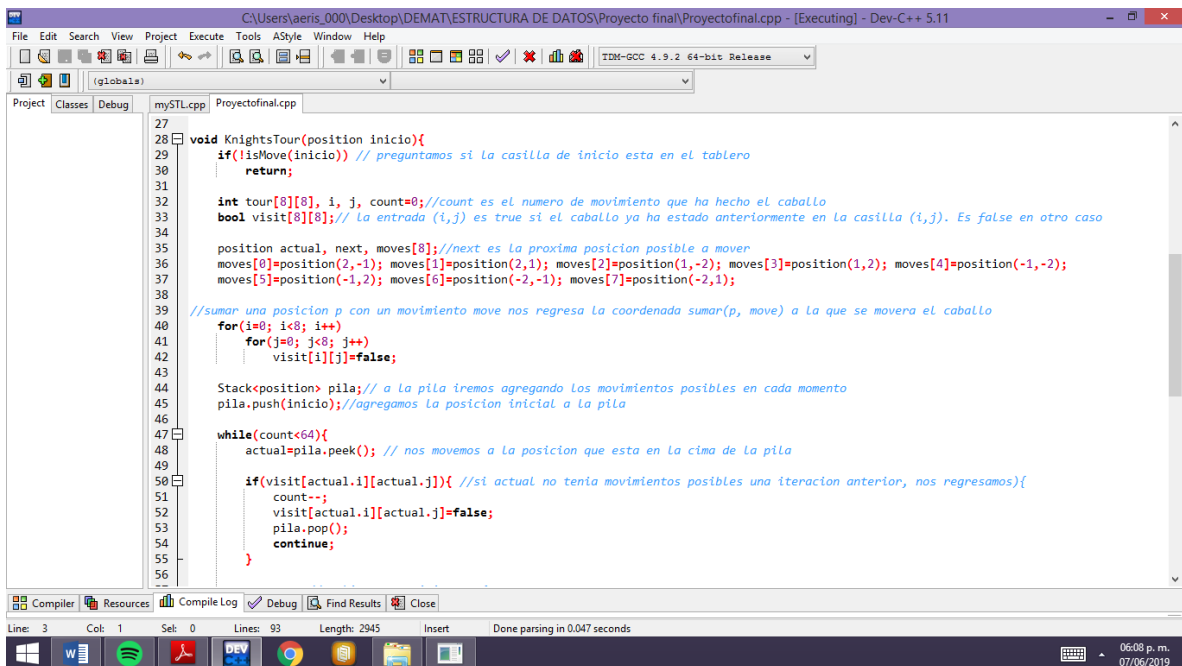
Hay caminos que son más tardados de ejecutarse que otros.

El usuario puede insertar la coordenada donde desea comenzar el recorrido.

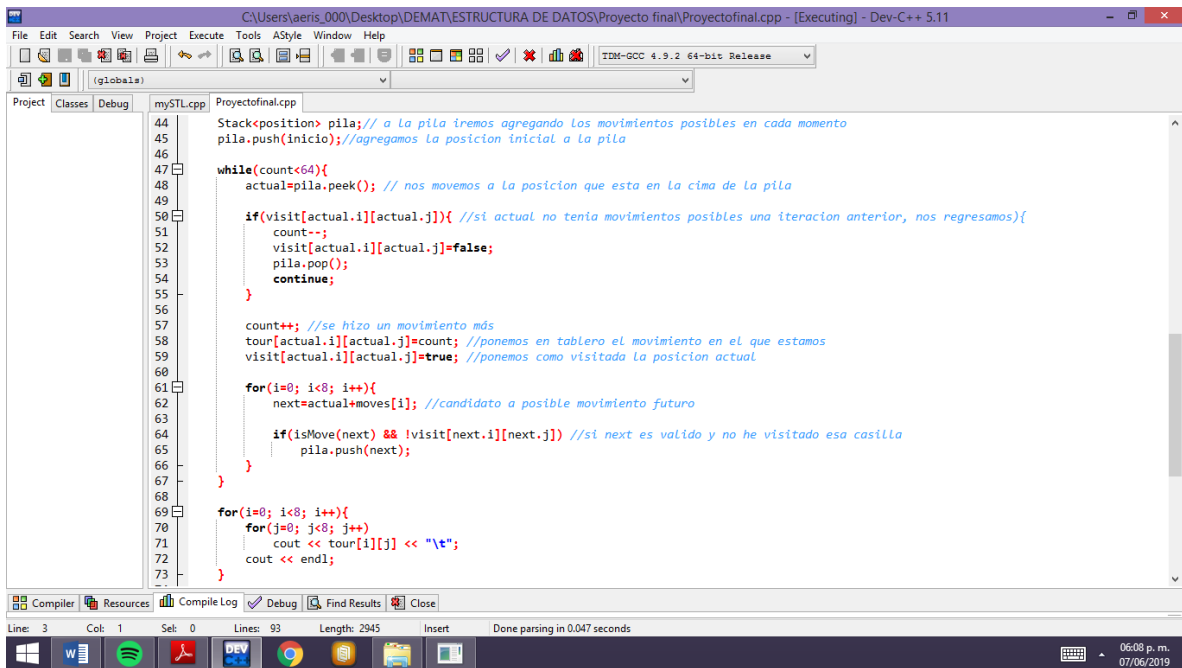
A continuación se muestra la evidencia del código con sus respectivos comentarios, y ejemplos de la implementación:



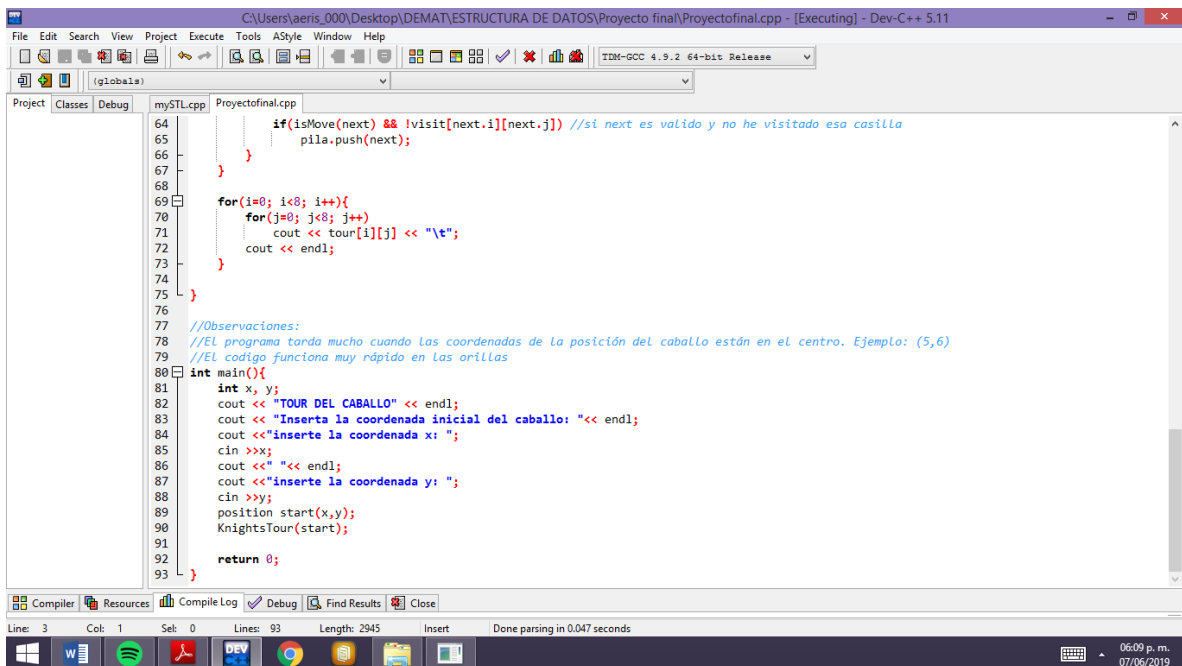
```
1 #include <iostream>
2 #include "mySTL.cpp"
3
4 //PROYECTO FINAL: ALGORITMO DEL CABALLO EN L TABLERO DE AJEDREZ
5 //PRESENTA: LESLIE JANETH QUINCOSA RAMIREZ
6 using namespace std;
7
8 struct position{ //estructura posición
9     int i, j; //nos indica la posición del caballo en el tablero, en sus respectivas coordenadas
10
11     position(int _i=0, int _j=0) { i=_i; j=_j; }
12
13     void operator=(position p){ i=p.i; j=p.j; }
14
15     position operator+(position p){
16         position sum(i+p.i,j+p.j);
17         return sum;
18     }
19 };
20
21 bool isMove(position p){
22     if(p.i>0 && p.i<=7 && p.j>=0 && p.j<=7)
23         return true;
24
25     return false;
26 }
27
28 void KnightsTour(position inicio){
29     if(!isMove(inicio)) // preguntamos si la casilla de inicio esta en el tablero
30         return;
```



```
27
28 void KnightsTour(position inicio){
29     if(!isMove(inicio)) // preguntamos si la casilla de inicio esta en el tablero
30         return;
31
32     int tour[8][8], i, j, count=0; //count es el numero de movimiento que ha hecho el caballo
33     bool visit[8][8]; // la entrada (i,j) es true si el caballo ya ha estado anteriormente en la casilla (i,j). Es false en otro caso
34
35     position actual, next, moves[8]; //next es la proxima posicion posible a mover
36     moves[0]=position(2,-1); moves[1]=position(2,1); moves[2]=position(1,-2); moves[3]=position(1,2); moves[4]=position(-1,-2);
37     moves[5]=position(-1,2); moves[6]=position(-2,-1); moves[7]=position(-2,1);
38
39     //sumar una posicion p con un movimiento move nos regresa la coordenada sumar(p, move) a la que se movera el caballo
40     for(i=0; i<8; i++)
41         for(j=0; j<8; j++)
42             visit[i][j]=false;
43
44     Stack<position> pila; // a la pila iremos agregando los movimientos posibles en cada momento
45     pila.push(inicio); //agregamos la posicion inicial a la pila
46
47     while(count<64){
48         actual=pila.peek(); // nos movemos a la posicion que esta en la cima de la pila
49
50         if(visit[actual.i][actual.j]){ //si actual no tenía movimientos posibles una iteracion anterior, nos regresamos{
51             count--;
52             visit[actual.i][actual.j]=false;
53             pila.pop();
54             continue;
55         }
56     }
```



```
44 Stack<position> pila; // a la pila iremos agregando los movimientos posibles en cada momento
45 pila.push(inicio); // agregamos la posición inicial a la pila
46
47 while(count<64){
48     actual=pila.peek(); // nos movemos a la posición que está en la cima de la pila
49
50     if(visit[actual.i][actual.j]){ // si actual no tenía movimientos posibles una iteración anterior, nos regresamos){
51         count--;
52         visit[actual.i][actual.j]=false;
53         pila.pop();
54         continue;
55     }
56
57     count++; // se hizo un movimiento más
58     tour[actual.i][actual.j]=count; // ponemos en tablero el movimiento en el que estamos
59     visit[actual.i][actual.j]=true; // ponemos como visitada la posición actual
60
61     for(i=0; i<8; i++){
62         next=actual+moves[i]; // candidato a posible movimiento futuro
63
64         if(isMove(next) && !visit[next.i][next.j]) // si next es válido y no he visitado esa casilla
65             pila.push(next);
66     }
67
68     for(i=0; i<8; i++){
69         for(j=0; j<8; j++){
70             cout << tour[i][j] << "\t";
71             cout << endl;
72         }
73     }
```



```
64         if(isMove(next) && !visit[next.i][next.j]) // si next es válido y no he visitado esa casilla
65             pila.push(next);
66     }
67
68     for(i=0; i<8; i++){
69         for(j=0; j<8; j++){
70             cout << tour[i][j] << "\t";
71             cout << endl;
72         }
73     }
74
75 }
76
77 // Observaciones:
78 // El programa tarda mucho cuando las coordenadas de la posición del caballo están en el centro. Ejemplo: (5,6)
79 // El código funciona muy rápido en las orillas
80 int main(){
81     int x, y;
82     cout << "TOUR DEL CABALLO" << endl;
83     cout << "Inserta la coordenada inicial del caballo: " << endl;
84     cout << "inserte la coordenada x: ";
85     cin >> x;
86     cout << " " << endl;
87     cout << "inserte la coordenada y: ";
88     cin >> y;
89     position start(x,y);
90     KnightsTour(start);
91
92     return 0;
93 }
```

Ejemplos del funcionamiento del programa.

```
C:\Users\aelis_000\Desktop\DEMAT\ESTRUCTURA DE DATOS\Proyecto final\Pr... - [ ] [X]
TOUR DEL CABALLO
Inserta la coordenada inicial del caballo:
inserte la coordenada x: 2

inserte la coordenada y: 1
64      25      2      15      12      9      4      7
27      16      63      24      3      6      13     10
62      1      26      17      14      11      8      5
49      28      23      42      57      18      21     34
40      61      50      29      22      33      56     19
51      48      41      58      43      20      35     32
60      39      46      53      30      37      44     55
47      52      59      38      45      54      31     36

-----
Process exited after 358.6 seconds with return value 0
Presione una tecla para continuar . . . _
```

```
C:\Users\aelis_000\Desktop\DEMAT\ESTRUCTURA DE DATOS\Proyecto final\Pr... - [ ] [X]
TOUR DEL CABALLO
Inserta la coordenada inicial del caballo:
inserte la coordenada x: 4

inserte la coordenada y: 4
19      22      5      8      17      12      3      10
6       25      18      21      4       9      16     13
23      20      7      26      15      2      11     28
32      35      24      41      30      27      14     43
37      52      31      34      1       42      29     56
62      33      36      53      40      57      44     47
51      38      63      60      49      46      55     58
64      61      50      39      54      59      48     45

-----
Process exited after 8.91 seconds with return value 0
Presione una tecla para continuar . . . _
```

```
C:\Users\Aeris_000\Desktop\DEMAT\ESTRUCTURA DE DATOS\Proyecto final\Pr...
TOUR DEL CABALLO
Inserta la coordenada inicial del caballo:
inserte la coordenada x: 2

inserte la coordenada y: 7
18 21 4 7 14 11 2 9
5 24 17 20 3 8 15 12
22 19 6 25 16 13 10 1
49 58 23 34 51 26 29 32
40 35 50 59 30 33 52 27
57 48 39 36 43 28 31 62
38 41 46 55 60 63 44 53
47 56 37 42 45 54 61 64

-----
Process exited after 42.95 seconds with return value 0
Presione una tecla para continuar . . . _
```

## Conclusión:

La implementación de pilas y memoria dinámica nos permite guardar el recorrido del caballo y así ir comparando rutas optimas, hasta conseguir la adecuada y donde recorra todo el tablero.

La STL nos permite aplicar funciones que nos facilitan la implementación de pilas y colas. El presente proyecto dio uso de los distintos métodos, entre otras aplicaciones y sus funciones vistas en clase.