ST0244 - Programming Languages

Practice III

School of Applied Sciences and Engineering - EAFIT University

Lecturer Alexander Narváez Berrío.

May 2025

PRACTICE III - VALUE: 16% OF THE FINAL GRADE OF THE COURSE.

POO PRACTICE: SYNTACTIC PARSER OF CHESS GAMES WITH BINARY TREE VISUALIZATION BY TURNS.

Objective:

Design and implement a program that syntactically validates a chess game written in standard algebraic notation (SAN), and, if it complies with the rules defined in the provided BNF grammar, represents it visually as a binary tree interlaced by turns.

Description: The program must allow as input a chess game written in standard algebraic notation (SAN), for example:

1. d4 d5 2. Bf4 Nf6 3. e3 e6 4. c3 c5 5. Nd2 Nc6 6. Bd3 Bd6 7. Bg3 O-O 8. Ngf3 Qe7 9. Ne5 Nd7 10. Nxc6 bxc6 11. Bxd6 Qxd6 12. Nf3 a5 13. O-O Ba6 14. Re1 Rfb8 15. Rb1 Bxd3 16. Qxd3 c4 17. Qc2 f5 18. Nd2 Rb5 19. b3 cxb3 20. axb3 Rab8 21. Qa2 Qc7 22. c4 Rb4 23. cxd5 cxd5 24. Rbc1 Qb6 25. h3 a4 26. bxa4 Rb2 27. Qa3 Rxd2 28. Qe7 Qd8 29. Qxe6+ Kh8 30. Qxf5 Nf6 31. g4 Ne4 32. Rf1 h6 33. Rc6 Qh4 34. Rc8+ Rxc8 35. Qxc8+ Kh7 36. Qf5+

The program should have two phases:

Syntactic analysis:

- Implement a parser that verifies that each individual move and each complete turn complies with the rules of chess.
- Based on the simplified BNF grammar (see below) that describes the structure of the SAN notation.
- If any move is invalid, an error message should be displayed indicating the move and the reason.

Binary tree: If the match is valid, a binary tree must be constructed where:

- The root is “Partida”.
- Each turn adds two children: white move (left) and black move (right).
- The generated tree should look similar to this (corresponds to the example entry):

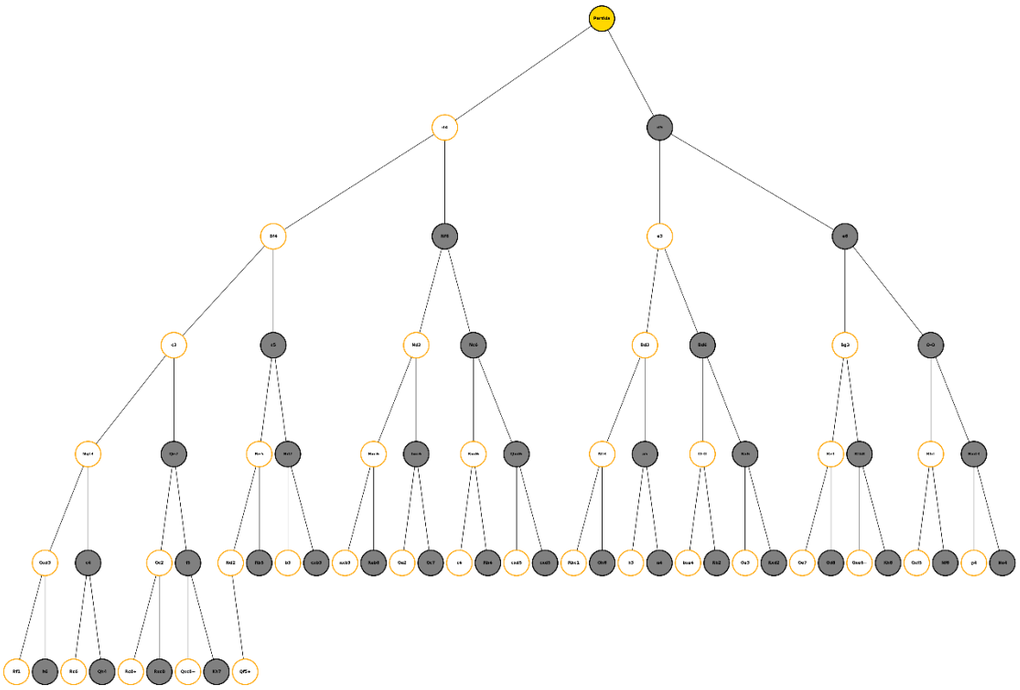

Figure 1: Binary Interlocked Binary Tree by Turns generated from a Game in SAN notation (Zoom to see it in detail)

BNF grammar for SAN notation:

<partida> ::= { <turno> }

<turno> ::= <numero_turno> "." <jugada_blanca> [<jugada_negra>]

<jugada_blanca> ::= <jugada>
<jugada_negra> ::= <jugada>

<jugada> ::= <enroque>
| <movimiento_pieza>
| <movimiento_peon>

<enroque> ::= "O-O" | "O-O-O"

<movimiento_pieza> ::= <pieza> <desambiguacion>? <captura>? <casilla>
<promocion>? <jaque_mate>?

<movimiento_peon> ::= <peon_captura> | <peon_avance>

<peon_avance> ::= <casilla> <promocion>? <jaque_mate>?

<peon_captura> ::= <letra> "x" <casilla> <promocion>? <jaque_mate>?

<desambiguacion> ::= <letra> | <numero> | <letra><numero>

<captura> ::= "x"

<casilla> ::= <letra><numero>

<letra> ::= "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h"
<numero> ::= "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8"

<promocion> ::= "=" <pieza>

<jaque_mate> ::= "+" | "#"

<pieza> ::= "K" | "Q" | "R" | "B" | "N"

<numero_turno> ::= <digito> {<digito>}
<digito> ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

Examples of valid and invalid plays in BNF-based SANs

Example in SAN notation	**¿Valid?**	**Motive if invalid**
e4	Yes	Pawn movement
Nf3	Yes	Horse movement
exd5	Yes	Pawn capture
O-O	Yes	Valid short cast
e8=Q	Yes	Valid promotion
Qh5+	Yes	Check to the king
1. e4 e5	Yes	Full shift
Qxe	No	Destination box incomplete
e9	No	Row 9 does not exist
Pxe4	No	No capital '**P**' used for pawns in SAN
O-O-O-O	No	Invalid long cast

Table 1. Syntactic validation of moves in SAN notation according to BNF

Tips:

- You can work with the PyQT library as I showed them in class.
- If you want to do it in C++, you can also use QTCreator as I showed them.

Supporting material: the Python and C++ project repositories with **graphical interface** shared by the teacher.

Delivery:

- Implement the solution applying the object-oriented paradigm in any language.
- That the program works exactly as requested and shows the expected results.
- Deadline: **two weeks after being assigned**.
- Please share through **Github**, include **README.md** file repository with the names of the members.
- Finally, make a short video presentation explaining and showing how the practice works on your PC with all the options described.

Assessment:

- Solution of the practice applying the object-oriented paradigm (40%).
- The defense of the practice must be done in person in class (60%).

Requirements:

- The practice can be done in teams of up to two (2) people.
- The delivery is done through a link to a repository on GitHub, therefore, the developed exercise must be uploaded to GitHub and share the link to access the proposed solution.
- The delivered solution must be based on object-oriented programming.
- In the GitHub repository, a "**Readme.md**" file must be included containing the full names of the members, the language, the version of the compiler they selected and the IDE they used for the development of the practice.
- The presentation of this practice must be coordinated with the teacher.