**COLEGIUL NAȚIONAL DE INFORMATICĂ “GR. MOISIL” BRAȘOV**

**LUCRARE PENTRU DOBÂNDIREA COMPETENȚELOR PROFESIONALE**

**Chess Engine**

**-Program de șah cu inteligență artificială -**

**Elev: Tișcă Cătălin George**

**Clasa: a XII-a D**

**Profesor îndrumător: Trandabur Alexandra / Șerban Manuela**

**Brașov - Mai 2025**

**CUPRINS**

[**1. MOTIVAȚIA ALEGERII TEMEI LUCRĂRII 2**](#_Toc196661442)

[**2. UTILITATEA APLICAȚIEI 3**](#_Toc196661443)

[**3. STRUCTURA APLICAȚIEI 4**](#_Toc196661444)

[**4. DETALII DE IMPLEMENTARE 7**](#_Toc196661445)

* [**Autentificarea și înregistrarea** 7](#_Toc196661446)
* [**Structura navigației între funcționalități** 7](#_Toc196661447)
* [**Calculatorul matematic** 7](#_Toc196661448)
* [**Reprezentarea grafică a funcțiilor** 8](#_Toc196661449)
* [**Formularul de Feedback** 10](#_Toc196661450)
* [**Butonul de Back** 10](#_Toc196661451)

[**5. RESURSE HARDWARE ȘI SOFTWARE 10**](#_Toc196661452)

* [**Resurse hardware minime pentru folosire** 10](#_Toc196661453)
* [**Resurse software minime pentru folosire** 10](#_Toc196661454)
* [**Resurse hardware pe care a fost implementată aplicația** 11](#_Toc196661455)
* [**Resurse software pe care a fost implementată aplicația** 11](#_Toc196661456)

[**6. POSIBILITĂȚI DE DEZVOLTARE 11**](#_Toc196661457)

[**7. BIBLIOGRAFIE 12**](#_Toc196661458)

# MOTIVAȚIA ALEGERII TEMEI LUCRĂRII

Am ales să dezvolt acest joc de șah cu scopul de a adresa provocările tehnice și conceptuale legate de implementarea unui motor de șah puternic și eficient. Alegerea unui algoritm bazat pe Negamax, optimizat cu Alpha-Beta Pruning, mi-a permis să îmbunătățesc performanța și să reduc semnificativ timpul de procesare necesar pentru analiza pozițiilor, chiar și în cazurile de joc avansate. De asemenea, implementarea acestui motor de șah îmi oferă o oportunitate excelentă de a lucra cu tehnici de inteligență artificială și de a înțelege mai profund modul în care algoritmii pot fi utilizați pentru a lua decizii strategice într-un joc complex.

Un alt factor motivant a fost alegerea limbajului Rust, care mi-a permis să dezvolt aplicația cu un nivel ridicat de siguranță în manipularea memoriei și performanță în procesare. Rust este un limbaj modern, optimizat pentru performanță și siguranță, care mi-a oferit flexibilitatea necesară pentru a implementa algoritmi complexi și pentru a interacționa eficient cu hardware-ul. Explorarea acestui limbaj a fost un aspect esențial al proiectului, deoarece mi-a permis să învăț și să mă familiarizez cu caracteristicile sale unice, cum ar fi gestionarea explicită a memoriei și abordarea sa riguroasă față de concurența între thread-uri.

În final, dezvoltarea acestui joc de șah nu doar că mi-a permis să îmbunătățesc abilitățile mele de programare, dar mi-a oferit și o oportunitate excelentă de a crea un produs care poate fi folosit atât pentru distracție, cât și pentru educație, demonstrând aplicabilitatea teoriei algoritmice în jocurile de strategie.

# UTILITATEA APLICAȚIEI

Scopul principal al aplicației este de a oferi o platformă pentru jocul de șah, care să includă atât o componentă de joc împotriva unui adversar uman, cât și o componentă în care utilizatorul poate juca împotriva unui motor de inteligență artificială bazat pe Negamax cu Alpha-Beta Pruning. Aceasta este utilă pentru cei care doresc să își îmbunătățească abilitățile de șah, oferind o provocare constantă și adaptabilă la nivelul de dificultate al jucătorului. De asemenea, aplicația poate fi folosită și în scop educativ, pentru a demonstra utilizarea algoritmilor de inteligență artificială în jocurile de strategie.

# STRUCTURA APLICAȚIEI

Aplicația este compusă din patru module principale:

* Interfața grafică: Interfața este realizată folosind biblioteca macroquad din Rust, care permite crearea unui mediu de joc simplu și intuitiv. Aceasta include tabla de șah, piesele, butoanele pentru interacțiune și alte elemente vizuale pentru a face jocul mai plăcut.
* Reprezentarea tablei: Tabla de șah este reprezentată cu valori sentinel. Piesele sunt stocate într-un cuvânt de 1 byte, în care:
* Primul bit reprezintă culoarea piesei.
* Următorii biți sunt utilizați pentru valori adiționale, cum ar fi starea mutării regelui (pentru rocada, de exemplu).
* Ultimii 3 biți sunt folosiți pentru a reprezenta tipul piesei (rege, regină, turn, cal, nebun, pion).
* Evaluare: Funcția de evaluare este simplificată conform metodei descrise în ChessProgramming Wiki. Aceasta ia în considerare factori precum valoarea pieselor și poziționarea acestora pe tablă.
* Căutare: Algoritmul de căutare folosește Negamax optimizat cu Alpha-Beta Pruning pentru a analiza posibilele mișcări ale computerului și a selecta cele mai bune mutări. Acest algoritm îmbunătățește eficiența căutării în arborele de decizie prin eliminarea ramurilor nepromițătoare.

# DETALII DE IMPLEMENTARE

* **Reprezentarea tablei**:  
  În aplicația de șah, tabla de șah este reprezentată folosind un vector bidimensional de valori sentinel, unde fiecare element reprezintă o piesă de pe tablă. Fiecare piesă este codificată într-un cuvânt de 1 byte. Aceasta se face astfel:
  + Primul bit reprezintă culoarea piesei (alb sau negru).
  + Următorii biți sunt utilizați pentru stocarea informațiilor adiționale despre piesă, cum ar fi dacă regele a fost mutat (pentru rocada).
  + Ultimii 3 biți sunt folosiți pentru a reprezenta tipul piesei (rege, regină, turn, cal, nebun sau pion).

Acest model de reprezentare reduce complexitatea calculului și crește eficiența în comparație cu alte tehnici de stocare a pieselor pe tablă. Implementarea acestui sistem de valori sentinel este realizată într-o structură simplă, care este accesată și manipulată rapid în Rust.

* **Motorul de șah (Negamax cu Alpha-Beta Pruning)**:  
  Algoritmul de căutare utilizat este Negamax, care este o variantă simplificată a Minimax. Acest algoritm este optimizat cu Alpha-Beta Pruning pentru a reduce numărul de noduri verificate în arborele de decizie.

În implementarea Rust, algoritmul este structurat astfel:

* + **Funcția** negamax(): Această funcție recursivă parcurge arborele de joc, evaluând fiecare posibilă mișcare a pieselor pe tablă. Pentru fiecare mutare posibilă, se generează o ramură a arborelui de decizie, iar evaluarea poziției curente este realizată pe baza unui scor numeric.
  + **Alpha-Beta Pruning**: Această tehnică este utilizată pentru a tăia ramurile nepromițătoare ale arborelui de decizie, care nu vor influența rezultatul final. Implementarea în Rust folosește două variabile, alpha și beta, pentru a urmări cele mai bune scoruri găsite pe parcursul căutării și a evita explorarea inutilă a ramurilor.

Am folosit tipuri de date simple și eficiente pentru a minimiza costurile de calcul, iar Rust a fost alegerea ideală datorită performanței sale excelente și a siguranței în manipularea memoriei.

* **Evaluarea pozițiilor**:  
  Funcția de evaluare a poziției este implementată conform metodei simplificate descrise în ChessProgramming Wiki. Aceasta ia în considerare:
  + **Valoarea pieselor**: Piesele sunt evaluate pe baza unei valori predefinite (de exemplu, regina are valoarea 9, turnul 5 etc.).
  + **Poziția pieselor pe tablă**: Anumite poziții pe tablă sunt considerate mai favorabile pentru anumite tipuri de piese (de exemplu, un cal este mai valoros în centru decât pe marginea tablei).

În Rust, această funcție este implementată eficient folosind un set de condiții pentru a evalua fiecare piesă pe baza poziției sale și a valorii sale relative. Valoarea totală a unei poziții este calculată și utilizată în algoritmul Negamax pentru a ghida procesul de selecție a celor mai bune mișcări.

* **Interfața grafică (realizată cu macroquad)**:  
  Interfața utilizator este realizată cu ajutorul bibliotecii macroquad, o bibliotecă simplă și rapidă pentru crearea de aplicații grafice în Rust. Funcționalitățile interfeței sunt implementate astfel:
  + **Afișarea tablei de șah**: Tabla este desenată folosind elemente grafice 2D, iar fiecare piesă este reprezentată prin imagini corespunzătoare. Piesele sunt plasate pe tablă în funcție de valorile stocate în vectorul de valori sentinel.
  + **Interacțiunea cu utilizatorul**: Butoanele și interacțiunile sunt gestionate folosind funcționalitățile oferite de macroquad, care permit citirea inputurilor de la tastatură și mouse pentru a permite utilizatorului să aleagă mișcări și să efectueze alte acțiuni.

Fiecare element grafic (tabla, piesele și butoanele) este desenat într-un loop de redare continuă, care asigură actualizarea constantă a interfeței în timpul jocului.

* **Stocarea jocurilor anterioare**  
  Jocurile anterioare sunt serializate într-un fisier. Fiecare joc este înregistrat cu informațiile relevante (cum ar fi jucătorii, scorurile și istoricul mișcărilor).
  + **Salvarea jocului**: După finalizarea unui joc, rezultatele (jucători, mișcări și scoruri) sunt salvate în baza de date pentru a putea fi accesate ulterior.
  + **Accesarea jocurilor anterioare**: Utilizatorii pot încărca jocurile anterioare stocate într-un fișier iar aplicația poate încărca și analiza aceste jocuri pentru a ajuta utilizatorul să își îmbunătățească performanțele. Fișierul se încarcă din parametrii.
* **Suport FEN**  
  În interfață există un câmp pentru introducerea unui FEN string (Forsyth-Edwards Notation), un format standard care descrie poziția curentă de pe tabla de șah (așezare a pieselor, rândul de mutare, drepturi de rocada etc.).

# HARDWARE ȘI SOFTWARE

## Resurse hardware minime pentru folosire

* + Procesor: Intel Core i3 (generația 3 sau mai nouă) sau AMD echivalent
  + Memorie RAM: minim 4 GB
  + Spațiu de stocare disponibil: minim 500 MB
  + Placă video integrată
  + Mouse, tastatură

## Resurse software minime pentru folosire

* + Sistem de operare: Windows 10 sau Windows 11 (64-bit)
  + Linux x86 / x64

## Resurse hardware pe care a fost implementată aplicația

* + 16GB RAM
  + CPU Intel i5-11400
  + GPU NVIDIA GTX 1050ti
* **Resurse software cu care a fost implementată aplicația**
  + Ubuntu Linux
  + RustRover IDE
  + Rustc 1.86.0
  + Macroquad, care se foloseste de API-ul grafic OpenGL
  + Serde JSON, pentru serializarea datelor in operatiile cu fisiere

# POSIBILITĂȚI DE DEZVOLTARE

Aplicația poate fi îmbunătățită în mai multe moduri, inclusiv:

* **Adăugarea unui modul de multiplayer online**, care să permită utilizatorilor să joace împotriva altor jucători la distanță.
* **Îmbunătățirea interfeței grafice**, adăugând efecte vizuale mai avansate și animarea mișcărilor pieselor.
* **Implementarea unui algoritm de învățare automată**, folosind perceptronul multi-strat (MLP), care să permită motorului de șah să învețe din jocurile anterioare și să îmbunătățească strategia sa de joc.

# BIBLIOGRAFIE

* [***https://www.rust-lang.org/***](https://www.rust-lang.org/)
* [***https://www.chessprogramming.org/Main\_Page***](https://www.chessprogramming.org/Main_Page)
* [***https://macroquad.rs/***](https://macroquad.rs/)
* [***https://sqlite.org/***](https://sqlite.org/)
* [***https://docs.rs/sqlite/latest/sqlite/***](https://docs.rs/sqlite/latest/sqlite/)

